PGS, TS. TRỊNH CHẤT
TS. TRỊNH ĐỒNG TÍNH

TỰ ĐỘNG HÓA THIẾT KẾ CƠ KHÍ



NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT

PGS.TS. TRỊNH CHẤT – TS. TRỊNH ĐỒNG TÍNH

TỰ ĐỘNG HOÁ THIẾT KẾ CƠ KHÍ

(In lần thứ nhất)

NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT
HÀ NỘI 2005

LỜI NÓI ĐẦU

Thông qua công cụ tin học, "Tự động hoá thiết kế cơ khí" nhằm củng cố và hoàn thiện các kiến thức về kết cấu cơ khí, nâng cao kỹ năng lập trình thiết kế tối ưu các chi tiết máy và bộ phận máy có công dụng chung và bồi dưỡng khả năng tiến hành thiết kế tự động các bản vẽ kỹ thuật.

Thuật ngữ và ký hiệu trong cuốn sách dựa theo tiêu chuẩn nhà nước và phù hợp với thuật ngữ và ký hiệu quốc tế.

Sách dùng làm giáo trình cho sinh viên ngành cơ - tin học kỹ thuật, làm tài liệu thiết kế cho sinh viên các ngành cơ khí của các trường đại học kỹ thuật và cao đẳng, làm tài liệu tham khảo cho các kỹ sư làm việc trong lĩnh vực thiết kế máy cũng như cho học viên cao học ngành cơ học máy. Sách còn được dùng làm tài liệu học tập cho thí sinh tham gia các kỳ thi olimpic quốc gia về tin học chi tiết máy.

Các tác giả tỏ lòng biết ơn Nhà xuất bản Khoa học và Kỹ thuật Hà Nội đã động viên và tạo điều kiện thuận lợi để tài liệu này sớm được hoàn thành và ra mắt bạn đọc. *Chúng tôi chân thành cảm ơn đồng nghiệp và bạn đọc về những ý kiến đóng góp quý báu gửi về địa chỉ: Bộ môn Cơ sở Thiết kế máy và Rôbôt, khoa Cơ khí, trường Đại học Bách khoa Hà nội.*

Các tác giả

Chương 1

NHỮNG VẤN ĐỀ CƠ BẢN VỀ THIẾT KẾ TỰ ĐỘNG

1.1. MỤC TIÊU VÀ NỘI DUNG THIẾT KẾ TỰ ĐỘNG

Trong quá trình nghiên cứu về "Cơ khí đại cương", "Chế tạo phôi", "Vẽ kỹ thuật", "Dung sai", "Nguyên lý máy", "Chi tiết máy", "Công nghệ chế tạo máy", "Máy cắt kim loại", "Kỹ thuật điện" và đặc biệt là "Đồ án môn học Chi tiết máy", các kiến thức về kết cấu cơ khí đã được hình thành và phát triển trong sinh viên. Thông qua các môn học này, người kỹ sư tương lai đã được bồi dưỡng khả năng phân tích, đánh giá các kết cấu cơ khí và thiết kế được những chi tiết máy, bộ phận máy, các cụm kết cấu và các thiết bị có kết cấu hợp lý, đáp ứng các yêu cầu kinh tế kỹ thuật. Các kết quả bước đầu này được tiếp tục bổ sung và phát triển thêm trong quá trình giải quyết các vấn đề liên quan đến thiết kế máy trong thực tế.

Đương nhiên với cùng một nội dung thiết kế luôn có nhiều giải pháp thực hiện và người thiết kế thường phải đề ra một số phương án tính toán, trên cơ sở đó tiến hành đánh giá, so sánh chúng với nhau nhằm xác định phương án có lợi nhất, đáp ứng tốt nhất các chỉ tiêu kinh tế kỹ thuật. Song do nhiều hạn chế về thời gian, về phương tiện và công cụ thực hiện, nội dung này trước đây chỉ được hoàn thành với một mức độ rất khiêm tốn. Ngày nay cùng với sự phát triển của khoa học kỹ thuật, đặc biệt là của công nghệ thông tin, những điều kiện thuận lợi để tiến hành lựa chọn tối ưu các thông số và kích thước cơ bản của chi tiết máy và tiến hành thiết kế tự động máy và kết cấu cơ khí nói chung đã được hình thành và phát triển rất nhanh, góp phần thúc đẩy sự phát triển mạnh mẽ lĩnh vực thiết kế cơ khí và toàn ngành cơ khí nói chung.

Tài liệu "Tự động hoá thiết kế cơ khí" được biên soạn nhằm đáp ứng các mục tiêu chính sau đây:

1) Củng cố, bổ sung và phát triển các kiến thức về kết cấu chi tiết máy, bộ phận máy có công dụng chung cũng như các kết cấu cơ khí khác thông qua công cụ tin học và các thành tựu mới về nghiên cứu thiết kế, chế tạo cơ khí.

2) Bồi dưỡng và nâng cao một bước kỹ năng lập trình đã được hình

thành và phát triển trong quá trình nghiên cứu về tin học đại cương và tin học chuyên ngành, nhờ đi sâu giải quyết các vấn đề về thiết kế tối ưu trong lĩnh vực tính toán thiết kế tự động các chi tiết máy, bộ phận máy và hệ dẫn động cơ khí.

3) Nâng cao chất lượng và hiệu quả thiết kế và rút ngắn thời gian thực hiện bản vẽ nhờ nắm vững và sử dụng thành thạo phần mềm AutoCAD để lập trình thiết kế tự động các bản vẽ cơ khí, bao gồm các bản vẽ chế tạo chi tiết, bản vẽ lắp các bộ phận máy và bản vẽ chung của toàn hệ thống cũng như thể hiện các kết cấu cơ khí khác.

Rõ ràng rằng các mục tiêu trên đây chỉ có thể được hoàn thành trên cơ sở:

a) Tìm hiểu, củng cố và trang bị thêm các kiến thức cơ bản về kết cấu và công nghệ cơ khí, nhờ đó có thể đề ra nhiều giải pháp và phương án khác nhau trong thiết kế và cuối cùng có thể lựa chọn một cách có căn cứ phương án tốt nhất để dùng.

b) Nắm vững và vận dụng có kết quả những ngôn ngữ lập trình sẽ được sử dụng (Pascal , C++, Visual Basic, v.v...) để thực hiện các nội dung tính toán thiết kế tự động và lựa chọn tối ưu các thông số và kích thước cơ bản của chi tiết máy, bộ phận máy và hệ dẫn động cơ khí.

c) Nắm vững các lệnh vẽ đơn giản, tìm hiểu và nghiên cứu kỹ các lệnh vẽ nhanh và các lệnh hiệu chỉnh của AutoCAD cũng như ngôn ngữ AutoLISP để lập trình thiết kế tự động các bản vẽ kỹ thuật.

Để thực hiện các mục tiêu nêu trên, thiết kế tự động cơ khí bao gồm các nội dung chính sau đây:

1) Xây dựng cơ sở dữ liệu trong thiết kế cơ khí: dữ liệu thiết kế cơ khí bao gồm toàn bộ số liệu cần thiết cho tính toán thiết kế, từ bảng tra các hệ số tính toán, các thông số và kích thước tiêu chuẩn của chi tiết máy, bộ phận máy, bảng tra cơ tính của vật liệu chế tạo máy hoặc các giá trị cho phép của công suất, áp suất, v.v... (dùng làm căn cứ để tính toán chi tiết máy), bảng tra dung sai lắp ghép, nhám bề mặt, v.v... đến dữ liệu đầu vào, kết quả chuyển tiếp và số liệu đầu ra. Tất cả các dữ liệu này cần được tổ chức, sắp xếp một cách khoa học, phục vụ các bài toán thiết kế chi tiết máy và bộ phận máy cụ thể, đảm bảo độ chính xác của các số liệu được sử dụng cũng như khả năng lựa chọn tối ưu các thông số và kích thước, đồng thời góp phần tăng tốc độ tính toán.

2) Lập trình tính toán thiết kế các chi tiết máy, bộ phận máy có công dụng chung cũng như các cụm kết cấu cơ khí điển hình. Cơ sở thiết kế ở đây chính là các chỉ tiêu chủ yếu về khả năng làm việc của chi tiết máy và

bộ phận máy cũng như các quan hệ động học, quan hệ kết cấu và khả năng công nghệ của từng đối tượng được thiết kế cụ thể. Tất cả các nội dung này đã được trình bày trong các tài liệu [1,2,3,4] cũng như trong sổ tay thiết kế cơ khí. Mục tiêu cần đạt được của các phần mềm này là:

- *Tính chính xác* của chương trình, thể hiện ở việc lựa chọn đúng chỉ tiêu chủ yếu về khả năng làm việc của chi tiết máy và bộ phận máy dùng làm cơ sở tính toán, sử dụng đúng các công thức tính toán độ bền và các quan hệ động học cũng như các bảng tra ứng với các điều kiện tra bảng cụ thể.

- *Tính tổng quát* của chương trình, tức là khả năng sử dụng phần mềm đã được soạn thảo để thiết kế các chi tiết máy và bộ phận máy ứng với một phạm vi đủ rộng của các dữ liệu đầu vào cũng như của cơ tính vật liệu được sử dụng để thiết kế.

- *Tính hiệu quả* của chương trình, thể hiện ở chỗ trong phần mềm có nhiều giải pháp thiết kế, nhờ đó có thể lựa chọn được phương án tối ưu để dùng, liên kết được các phần tính toán với phần thiết kế bản vẽ, nhờ đó có thể đưa ra được những thay đổi và hiệu chỉnh cần thiết trước khi quyết định lần cuối về kết cấu. Tính hiệu quả còn thể hiện ở tốc độ tính toán nhanh nhằm tiết kiệm tối đa thời gian thiết kế cũng như khả năng kiểm tra các kết quả tính toán khi cần thiết.

3) Lập trình thiết kế tự động các bản vẽ kỹ thuật theo đúng các yêu cầu đặt ra cho các tài liệu thiết kế. Ở đây người thiết kế không đối thoại trực tiếp với máy tính để thực hiện các lệnh vẽ thông dụng, lệnh vẽ nhanh cũng như các lệnh hiệu chỉnh của AutoCAD mà tiến hành xây dựng các tập tin đáp bản (Script File) hoặc tập tin AutoLISP để máy tính tự động thực hiện các bản vẽ kỹ thuật theo đúng các quy định về biểu diễn bản vẽ (TCVN 3826-83) và về khổ giấy (TCVN 2-74). Bản vẽ được hoàn thành theo một trình tự hợp lý nhằm biểu diễn rõ ràng và chính xác các kết cấu phức tạp, phân biệt rõ ràng về đường nét, mặt cắt, phân biệt về độ lớn và nét của chữ số ghi kích thước liên kết, kích thước khuôn khổ, kích thước lắp ghép hoặc dung sai.

Phần mềm thiết kế bản vẽ có thể thực hiện dưới dạng một *Unit* thể hiện nhiều kết cấu khác nhau (bản vẽ chế tạo) thuộc chương trình tính toán thiết kế một chi tiết máy có công dụng chung, cũng có thể là một phần mềm riêng (bản vẽ lắp, bản vẽ chung) gắn với chương trình tính toán thiết kế một hệ dẫn động gồm nhiều chi tiết máy và bộ phận máy khác nhau. Khi máy tính thực hiện những phần mềm này, các bản vẽ được thể hiện trong không gian 2D.

Trong tài liệu này cũng giành một phần quan trọng để trình bày cách sử dụng ngôn ngữ AutoLISP, một ngôn ngữ lập trình được thực thi thuận lợi trong môi trường AutoCAD và đặc biệt thích hợp cho quá trình biểu diễn đồ họa, để lập trình thiết kế tự động các bản vẽ kết cấu cơ khí.

Để trình bày các ý tưởng, hướng dẫn thực hiện các nội dung tính toán thiết kế các chi tiết máy và hệ dẫn động cơ khí và cho các thí dụ cụ thể, tài liệu "Tự động hoá thiết kế cơ khí" sử dụng ngôn ngữ PASCAL, một phần mềm có cấu trúc rõ ràng, dễ sử dụng, thuận tiện trong việc kết nối phần tính toán thiết kế nhằm xác định các thông số và kích thước cơ bản của chi tiết máy và bộ phận máy với phần vẽ tự động được thực hiện trong môi trường AutoCAD. Nói cách khác, bằng cách kết hợp *tập tin văn bản* ghi kết quả tính toán thiết kế với *tập tin đáp bản* ghi các lệnh vẽ, có thể dùng ngôn ngữ PASCAL để lập trình vẽ tự động các bản vẽ phức tạp có dung lượng lớn bằng cách phân một chương trình lớn thành nhiều chương trình con rồi chuyển chúng thành các *Unit Overlay* và chỉ gọi đến khi cần thiết. Mặt khác, sử dụng cùng một ngôn ngữ lập trình để thực hiện cả phần tính và phần vẽ rõ ràng là rất thuận tiện trong việc phát hiện kịp thời các sai sót trên bản vẽ, từ đó có thể thay đổi hiệu chỉnh nhanh chóng làm cho sản phẩm có kết cấu hợp lý hơn.

Đương nhiên, dựa theo các ý tưởng cũng như các vấn đề về phương pháp và thuật giải được trình bày đối với PASCAL, bạn đọc cũng có thể sử dụng các ngôn ngữ khác để lập trình tính toán thiết kế chi tiết máy và bộ phận máy có công dụng chung cũng như vẽ tự động các bản vẽ cơ khí. Vấn đề quan trọng ở đây là bạn cần nắm vững và vận dụng có kết quả các ngôn ngữ này phục vụ các nội dung chính và các mục tiêu đề ra khi tiến hành tự động hoá thiết kế cơ khí.

Ngoài các gợi ý, hướng dẫn về phương pháp và những vấn đề cần lưu ý khi lập trình tính toán thiết kế và vẽ tự động các bản vẽ kỹ thuật, để thuận tiện cho người sử dụng, trong tài liệu này còn cung cấp các công thức dùng để tính toán động học và xác định các thông số và kích thước chi tiết máy, các công thức thiết kế và kiểm nghiệm khả năng tải của chi tiết máy. Các bảng tra cần cho thiết kế và vẽ tự động được lấy từ tài liệu [2].

Một vấn đề khác về phương pháp không kém phần quan trọng nhằm hoàn thành đúng hạn và có chất lượng nội dung tự động hoá thiết kế cơ khí, đó là việc rèn luyện tác phong làm việc chính xác, khoa học, thường xuyên kiểm tra để phát hiện các lỗi và sửa chữa kịp thời, hoàn thành dứt điểm từng phần công việc, đồng thời luôn chú ý rút ra những nhận xét cần thiết, góp phần hoàn thiện các phần mềm đã được phác thảo lúc đầu.

1.2. PHƯƠNG PHÁP LẬP TRÌNH THIẾT KẾ CƠ KHÍ

Các thiết bị cơ khí và các dây chuyền công nghệ thường bao gồm nhiều chi tiết máy và bộ phận máy có công dụng chung, nhiều chi tiết tiêu chuẩn và những cụm kết cấu điển hình.Vì các chi tiết máy và bộ phận máy có công dụng chung được sử dụng ở rất nhiều thiết bị cơ khí cho nên vấn đề xây dựng phần mềm tính toán thiết kế chi tiết máy, bộ phận máy và hệ dẫn động cơ khí chiếm một phần rất quan trọng trong lập trình thiết kế cơ khí. Khi soạn thảo các phần mềm này, một mặt phải xuất phát từ các chỉ tiêu chủ yếu về khả năng làm việc của chi tiết máy để thiết lập các môđun tính toán cơ bản, mặt khác phải biết vận dụng những ưu điểm của các ngôn ngữ lập trình bậc cao để đề ra những giải thuật hợp lý nhằm thực thi có hiệu quả các ý tưởng trong thiết kế, xây dựng được những chương trình có cấu trúc rõ ràng, dễ sử dụng, đáp ứng được các yêu cầu về tối ưu hóa và tự động hoá trong thiết kế cơ khí.

Dưới đây gợi ý một số vấn đề về phương pháp lập trình tính toán thiết kế chi tiết máy và bộ phận máy có công dụng chung.

1.2.1. Viết lưu đồ thuật giải

Trước khi viết chương trình, nhất là các chương trình phức tạp bao gồm nhiều chương trình con, các thủ tục và hàm, cần xây dựng các lưu đồ thuật giải thể hiện các bước và nội dung tính toán, các yêu cầu cần đạt được về độ chính xác, về sự hợp lý của các thông số được lựa chọn, v.v... Khi trình bày lưu đồ thuật giải cần lưu ý những điểm sau:

a) Sử dụng theo quy ước các hình phẳng (elip, hình thang, hình chữ nhật, hình thoi, v.v...), bên trong đặt nội dung hoặc yêu cầu tính toán cùng số thứ tự công thức, đồng thời sử dụng các sơ đồ và ký hiệu quy ước để biểu diễn quá trình tính toán từ lúc bắt đầu cho tới khi kết thúc (xem thí dụ về lưu đồ thuật giải thiết kế truyền động bánh răng côn trong chương 2, hình 2.3).

b) Sử dụng các biến điều khiển i ,j , k,... kèm theo dấu và các hệ số với giá trị thích hợp để thực hiện các vòng lặp nhằm tăng hoặc giảm giá trị của các thông số thiết kế, nhờ đó đảm bảo được yêu cầu tối ưu trong tính toán hoặc lựa chọn các thông số.

c) Sắp xếp và bố trí các hình phẳng, các đường dẫn và mũi tên trên lưu đồ một cách khoa học, tương ứng với các bước tính toán thiết kế, tránh chồng chéo; kích thước (chiều dài và chiều rộng) các hình nên cố gắng chọn như nhau. Khi sử dụng hình thoi để thể hiện *lệnh điều kiện* (*If... then... else...*) hoặc *lệnh lựa chọn* (*Case... of...*) nhất thiết phải có

hai đường dẫn, một đường với ký hiệu Đ (đúng) đáp ứng điều kiện đặt ra dẫn đến bước tính tiếp theo, đường kia với ký hiệu S (sai) cần được xử lý trước khi thực hiện vòng lặp để thực hiện lại đoạn chương trình vừa qua.

d) Với các chương trình tính toán phức tạp, không thể biểu diễn trên một trang giấy có thể tách ra một số chương trình con để viết lưu đồ thuật giải riêng rồi gộp vào lưu đồ chung, khi đó trên các hình chữ nhật thể hiện các chương trình con hoặc một loạt các phương trình tính toán cần kẻ thêm hai gạch đứng (xem lưu đồ thuật giải chọn số răng - hình 2.4, 2.5 và lưu đồ thuật giải thiết kế truyền động bánh răng hành tinh trên hình 2.6).

Như vậy nhờ có lưu đồ thuật giải, các chương trình phần mềm sẽ có cấu trúc rõ ràng, tính toán được thực hiện một cách tuần tự, dễ phát hiện và sửa lỗi, khắc phục được các sai sót khi lập trình, đảm bảo được các yêu cầu đặt ra cho bài toán thiết kế.

1.2.2. Phép lặp và đặc điểm thiết kế chi tiết máy theo hai bước

Do điều kiện làm việc của chi tiết máy rất phức tạp, kết cấu chi tiết máy rất đa dạng nên thiết kế chi tiết máy thường được tiến hành theo hai bước: tính thiết kế và tính kiểm nghiệm. Ở đây các thông số và kích thước chi tiết máy được xác định sơ bộ trong bước tính thiết kế, sẽ được xử lý và thay đổi cho đến khi kiểm nghiệm đạt yêu cầu. Muốn vậy khi lập trình cần:

1. Sử dụng vòng lặp lồng nhau với các biến i, j, k,... có kèm dấu với yêu cầu sau một số vòng lặp nhất định tìm được các giá trị thích hợp nhất của các thông số và kích thước cần xác định.

2. Kết hợp vòng lặp với lệnh lựa chọn *Case* để tìm phương án thoả đáng trong trường hợp không đảm bảo chỉ tiêu tính toán. Chẳng hạn, khi thiết kế bộ truyền trục vít, với vật liệu đã chọn và ứng suất tiếp xúc đã được xác định, sau khi tính được khoảng cách trục a_w mà kiểm nghiệm không đảm bảo độ bền tiếp xúc, tức là tính ra $\sigma_H > [\sigma_H]$ thì có thể thay đổi các thông số theo một trong các phương án sau đây:

- Thay đổi môđun (do đó a_w sẽ thay đổi theo).
- Thay đổi hệ số đường kính q (do đó a_w sẽ thay đổi theo).
- Thay đổi khoảng cách trục a_w (do đó m và q sẽ thay đổi theo).
- Thay đổi số răng bánh vít z_2.

Tất cả các giải pháp xử lý đều làm thay đổi các thông số ăn khớp. Giải pháp được chọn sẽ là phương án vừa đảm bảo độ bền tiếp xúc, vừa có kích thước gọn, có kết cấu hợp lý và ít làm thay đổi tỉ số truyền của bộ truyền trục vít.

3. Tạo ra các đoạn chương trình phù hợp để lựa chọn tối ưu các thông số thiết kế với yêu cầu vi phạm ít nhất các dữ liệu tính toán. Chẳng hạn khi thiết kế bộ truyền trục vít, sau khi xác định được khoảng cách trục a_w, môđun m, hệ số đường kính q và số răng bánh vít z_2, nếu tính ra hệ số dịch chỉnh không đảm bảo điều kiện $-0,7 \leq x \leq 0,7$ thì cần thay đổi z_2 từ 1 đến 2 răng sao cho sai lệch tỉ số truyền ít nhất. Có nhiều đáp án nhưng đáp án có tỉ số truyền thay đổi ít nhất phải được chọn trước (xem chương 2, mục 2.3.3).

4. Dựa theo các giải pháp thiết kế hoặc các cách phối hợp khác nhau khi lựa chọn vật liệu và các quan hệ kết cấu, tiến hành tính toán thiết kế chi tiết máy theo một số phương án xác định, cuối cùng cần tổng hợp và viết ra màn hình giá trị của các thông số và kích thước cơ bản của chi tiết máy tương ứng với các phương án thiết kế khác nhau, từ đó có thể tiến hành so sánh, cân nhắc và chọn được phương án ưu việt nhất để dùng.

1.2.3. Xác định chiều của tải trọng

Khi tính toán thiết kế máy hoặc chi tiết máy cần xác định phương, chiều và cường độ của tải trọng (lực hoặc mômen) tác dụng lên chi tiết máy (trục, ổ lăn, v.v...) hoặc bộ phận máy. Trị số và chiều của tải trọng được xác định theo đặc điểm liên kết và điều kiện làm việc cụ thể của từng chi tiết máy. Khi lập trình tính toán tải trọng, trị số của lực hoặc mômen dễ dàng được xác định theo các công thức tương ứng, nhưng chiều của tải trọng chỉ có thể được xác định trong hệ tọa độ Oxyz theo những quy ước thống nhất về dấu, phụ thuộc vào các yếu tố ảnh hưởng. Nói cách khác, để xác định chiều của một tải trọng bất kỳ, trước hết hãy tìm ra các yếu tố ảnh hưởng, định nghĩa các yếu tố này là những biến nguyên, có giá trị bằng 1 và quy ước dấu (+ hoặc –) cho chúng rồi đưa vào phương trình tính toán tải trọng đó.

Chẳng hạn, lực dọc trục từ các bộ truyền tác dụng lên trục phụ thuộc vào chiều quay (cq), hướng răng (hr) và chi tiết quay (bánh răng, bánh vít, trục vít) là chủ động hay bị động (cb). Do đó khi xác định chiều của các lực này tác dụng lên trục, cần nhân thêm vào công thức tính lực các biến cq, hr và cb kèm theo dấu [xem các công thức tính lực từ các chi tiết quay tác dụng lên trục ở chương 3, từ công thức (3.1) đến (3.9)]. Bằng phương pháp tương tự cũng sẽ xác định chiều của tổng lực dọc trục tác dụng lên các gối đỡ của ổ lăn đỡ - chặn và do đó xác định được lực dọc trục tác dụng lên mỗi ổ.

1.2.4. Tạo các phương pháp tra bảng cho chương trình

Khi tính toán thiết kế chi tiết máy hoặc bộ phận máy thường phải sử dụng rất nhiều bảng số được lập ra từ các kết quả nghiên cứu về lý thuyết

và thực nghiệm. Để máy tính có thể đọc được các bảng số phục vụ các phần mềm thiết kế, tất cả các bảng này cần được lưu trữ lên đĩa. Một số bảng xác định sẽ được gọi đến khi tiến hành thiết kế một chi tiết máy cụ thể.

Để thuận tiện trong sử dụng và tránh trùng lặp, nên soạn thảo một *Unit* tra bảng dùng chung cho các chương trình tính toán thiết kế, trong đó chứa các chương trình con thể hiện các phương pháp tra bảng khác nhau, đáp ứng tính chất đa dạng của các bảng trong thiết kế (xem mục 1.3.1). *Unit* tra bảng này được gắn với từng chương trình tính toán chi tiết máy cũng như gắn với các chương trình thiết kế bộ phận máy hoặc cả hệ dẫn động cơ khí.

1.2.5. Soạn thảo các *Unit* riêng cho chương trình

Một chương trình tính toán thiết kế dù lớn hay nhỏ cần được phân thành nhiều phần, mỗi phần nằm trong *Unit* riêng, nói cách khác người lập trình cần tạo ra các *Unit* riêng cho bản thân mình, ngoài các *Unit* có sẵn trên máy như *Crt, Graph,...* Muốn vậy trước hết hãy viết các *Program* riêng, sau khi gỡ rối sẽ đổi từng *Program* riêng thành *Unit*, cuối cùng gắn các *Unit* vào chương trình chính.

Khi đổi *Program* thành *Unit* cần lưu ý rằng mỗi *Unit* thường gồm 5 phần [15]:

- Phần khai báo (để đặt tên cho *Unit*).

- Phần giao tiếp: *Interface* (để đặt các khai báo như: *Const, Type, Var* cũng như các khai báo *Procedure* và *Function*).

- Phần cài đặt bắt đầu bằng từ khoá *Implementation* để cài đặt các thủ tục, hàm đã khai báo ở phần giao tiếp.

- Phần khởi tạo có thể có hoặc không, nằm ngay trước phần kết thúc và bắt đầu bằng từ khoá *Begin*, theo sau là các lệnh để khởi tạo.

- Phần kết thúc chỉ gồm từ khoá *End*, báo cho chương trình biên dịch biết đã hết phần cài đặt *Unit*.

Sau khi soạn thảo *Unit* xong cần biên dịch *Unit* vào một tập tin trên đĩa có phần mở rộng *TPU*. Muốn vậy hãy chọn mục *Compile* rồi nhấn chuột hoặc bấm phím *Alt-F9*.

Nhờ các *Unit* đã chứa các thủ tục và hàm quan trọng nên việc xây dựng chương trình chính trở nên đơn giản hơn. Mặt khác, do các chương trình con trong *Unit* đã được dịch nên nó làm giảm đáng kể thời gian biên

dịch chương trình chính, đồng thời giảm được mã nguồn (*Code*) dùng để biểu diễn các thuật giải nhằm giải quyết các bài toán cụ thể.

1.2.6. Biến toàn cục, biến cục bộ và truyền dữ liệu

Để thu được kết quả tính toán chính xác và giảm bớt kích thước cho chương trình, cần sử dụng hợp lý biến toàn cục, biến cục bộ và biến thông số, đặc trưng phương pháp truyền dữ liệu cho các chương trình con, bao gồm tham số giá trị (tham trị), tham số biến (tham biến) hoặc cả hai. Nếu một biến có kiểu bất kỳ, có ý nghĩa như nhau ở nhiều chương trình con thì nó phải được định nghĩa là biến toàn cục mà không được xem là biến cục bộ để rồi phải định nghĩa nhiều lần ở đầu các chương trình con.

Chẳng hạn, khoảng cách trục aw[k] của bánh răng cấp k; số răng z[i,k] của bánh răng i, cấp k cùng với các biến kiểu *Byte* i và k là những biến toàn cục, nên được khai báo ở *Unit* đầu tiên mà các *Unit* khác hoặc chương trình sẽ gọi đến; số răng tương đương ze[i] (được dùng để tra hệ số dạng răng Y[i] khi tính bánh răng trụ răng nghiêng về độ bền uốn) là biến cục bộ, được khai báo ở đầu chương trình con, còn thí dụ về truyền dữ liệu có thể xem các chương trình con trong *Program vkttb* ở thí dụ 1 trong mục 1.3.3. Cần chú ý thêm rằng phương pháp truyền dữ liệu đặc biệt thích hợp đối với các chương trình con được sử dụng nhiều lần trong một chương trình hoặc cho nhiều chương trình tính toán thiết kế khác nhau ứng với những điều kiện sử dụng khác nhau.

1.2.7. Tốc độ tính toán

Muốn tăng được tốc độ tính toán, một mặt cần tạo ra nhiều đoạn chương trình để máy tự động thực hiện theo hướng vừa đảm bảo các chỉ tiêu tính toán, vừa lựa chọn được các thông số và kích thước tối ưu, mặt khác cần tìm cách giảm thời gian thao tác trên máy cũng như giảm thời gian đối thoại với máy. Chẳng hạn, có thể số hoá các dữ liệu, điều kiện (như vật liệu, nhiệt luyện, cách đúc, phương pháp gia công, v.v...) khi chọn vật liệu trong chương trình thiết kế truyền động bánh răng hoặc số hoá các loại động cơ điện trong chương trình tính toán chọn động cơ trong thiết kế máy.

Đương nhiên, khi thực hiện chương trình phải gán đúng những con số mà chương trình đã chọn. Để tránh trường hợp gán nhầm mà máy vẫn nhận số liệu dẫn đến kết quả sai lệch và buộc phải dừng, cần phải sử dụng *vòng lặp có điều kiện* cho các phép gán.

Thí dụ, nếu sử dụng các con số sau đây để chỉ loại động cơ của các Công ty:

1 : Việt - Hưng ; 2 : Điện cơ Hà Nội ; 3 : Nga,

thì có thể sử dụng đoạn chương trình dưới đây:

```
write (' Chọn loại động cơ (ldc) của công ty nào ? ');
write (' 1: VIET-HUNG ; 2: ĐIEN CO HA NOI ; 3: NGA ');
repeat
  write (' Chọn loại động cơ (ldc) của công ty nào ? ');
  readln (ldc);
until ldc in [1..3];
```

Sau đó nếu muốn viết ra màn hình động cơ của công ty nào đã được chọn, có thể sử dụng lệnh lựa chọn sau đây với *loaidc* được định nghĩa là *String[15]*:

```
case ldc of
  1:loaidc:='VIET-HUNG';
  2:loaidc:='DIEN CO HA NOI');
  3:loaidc:='NGA');
end;
writeln (' Dùng động cơ của công ty ', loaidc );
```

Như vậy chỉ khi gán đúng một trong các con số 1, 2 hoặc 3 cho loại động cơ được sử dụng thì chương trình mới được tiếp tục thực hiện.

1.2.8. Soạn thảo chương trình gồm cả phần tính toán và đồ họa

Trường hợp ngoài phần tính toán, chương trình soạn thảo gồm cả phần vẽ sơ đồ hoặc đồ thị thì cần lưu ý những vấn đề sau đây:

1) Khởi tạo đồ họa: Khi chuyển từ phần tính toán sang phần vẽ, màn hình thay đổi từ chế độ thể hiện văn bản sang chế độ thể hiện đồ họa, do đó trước khi vẽ cần phải khởi tạo đồ họa, chẳng hạn có thể sử dụng *Procedure* khởi tạo đồ họa sau đây:

```
Procedure ktdh;
Var  i, driver, mode : integer;
       duongdan : string[25];
begin
  driver: = detect;
  duongdan: = 'C:\TP\BGI';
  initgraph(driver,mode,duongdan);
  i: = graphresult;
  if i < > grok then
  begin
    writeln(' Duongdan chua dung,cho lai duongdan' );
    write('duongdan= ');
```

```
    readln(duongdan);
    driver: = detect;
    initgraph(driver,mode,duongdan);
    i: = graphresult;
    clrscr;
  end;
  setgraphmode(egahi) ;
end;
```

Sau đó nếu tiếp tục tính toán, cần đưa màn hình về chế độ văn bản nhờ lệnh *TEXTMODE*.

Chú ý rằng, với chương trình khởi tạo đồ họa trên đây, nếu lúc đầu đường dẫn không đúng với đường dẫn cho trong *Procedure ktdh* ở trên, tức là các *File* đồ họa không nằm trong thư mục *'C:\TP\BGI'* thì có thể gán lại đường dẫn phù hợp, khi đó chương trình sẽ được tiếp tục thực hiện.

2) Khi soạn thảo phần mềm đồ họa nên:

- Chọn tọa độ gốc để bắt đầu vẽ hình sau đó cố gắng chỉ sử dụng các tọa độ tương đối, như thế sẽ giảm được thời gian khi cần thay đổi sửa chữa.

- Có thể dùng nét và màu khác nhau để vẽ sơ đồ tải trọng nhưng cũng có thể dùng các cột màu để thể hiện các bậc tải trọng.

- Sử dụng các *Procedure* vẽ từng cụm riêng, viết chữ hoặc con số sau đó mới ghép các cụm lại với nhau.

3) Khi màn hình đang ở chế độ thể hiện văn bản, nếu cần tìm lỗi hoặc cần in kết quả tính toán ra màn hình cũng như khi cần cho số liệu từ bàn phím, thường dùng các lệnh *Write* hoặc *Writeln* và *Read* hoặc *Readln*. Tuy nhiên ở chế độ màn hình đồ họa, các lệnh này không có ý nghĩa, khi đó cần sử dụng các câu lệnh sau đây để thể hiện nội dung khác nhau trên màn hình đồ họa:

a) Viết chữ và số vào một vị trí xác định trên màn hình, chẳng hạn muốn viết chữ r = 50 ở vị trí x = 150, y = 120 thì dùng các lệnh sau:

```
Str(50:3:1, textr);              {biến 50 thành ký tự }
Cauviet := ' r = ' +textr;       {r = 50 được viết theo lệnh này
                                  dưới dạng ký tự }
Outextxy(150, 110, Cauviet);     {viết r = 50 ở x = 150 , y = 110 }
Val(textr, valr, result);        {biến ký tự thành số 50 }
```

trong đó *textr: string[3] ; valr: real ; result : integer.*

b) Chỉ viết chữ, khi đó bỏ lệnh *Str( ) và Val( )*, chẳng hạn:

```
Cauviet: = ' Kết quả tính toán ';
Outextxy(150,120, Cauviet);
```

c) Viết chữ là một ma trận, muốn vậy chữ cần được viết dưới dạng một ký tự có biến số, chẳng hạn muốn viết trị số của mômen uốn M[k,j] trong mặt phẳng thứ k, tại các tiết diện j, có thể sử dụng đoạn chương trình sau, với định nghĩa *valM : array [0..3,0..n] of real; textM : array[0..5] of string[10]; j , k: integer; textj , textk : string[3];*

```
Str(k,textk);
Str(j,textj);
Str(M[k,j]: 7:1, textM[k,j]);
Cauviet: = ' M['+textk+','+textj+'] = ' +textM[k,j];
Outextxy(rx[j], ry[j], Cauviet);
Val(textM[k,j], valM[k,j], result);
```

4) Vì đồ họa PASCAL chỉ dùng trị nguyên để vẽ cho nên sau khi xác định các tọa độ x, y là biến thực cần chuyển thành biến nguyên rx, ry để vẽ, chẳng hạn dùng các dòng lệnh

```
ry[j] : = round (y[j]) ;
rx[j]: = round (x[j]) ;
```

1.2.9. Soạn thảo chương trình gồm cả phần tính toán và thiết kế tự động bản vẽ

Như đã trình bày ở phần trên, khi tính toán thiết kế chi tiết máy và hệ dẫn động cơ khí cần phải sử dụng các ngôn ngữ lập trình bậc cao, còn khi thiết kế bản vẽ có thể sử dụng ngôn ngữ AutoLISP để lập trình vẽ tự động trong môi trường AutoCAD. Trường hợp sử dụng ngôn ngữ lập trình PASCAL, có thể tiến hành lập trình cả phần tính toán thiết kế và phần vẽ (bằng các lệnh của AutoCAD) trong cùng một chương trình phần mềm, bao gồm nhiều thủ tục, hàm và các *Unit*. Kết thúc phần tính toán thiết kế, các thông số và kích thước cơ bản của chi tiết máy (được xác định theo các chỉ tiêu chủ yếu về khả năng làm việc của chúng) sẽ được dùng làm căn cứ để tính toán thêm các yếu tố kết cấu khác và cùng với các yếu tố này sẽ xác định nên kết cấu, hình dáng và kích thước chi tiết máy hoặc bộ phận máy cần được thể hiện trên bản vẽ.

Như vậy để kết nối phần tính toán thiết kế và phần vẽ, trong cả hai trường hợp, các kết quả tính toán được thể hiện bằng các thông số và kích thước cơ bản của chi tiết máy cần phải được nạp vào *File* và trước khi lập trình thiết kế tự động bản vẽ, sẽ được gọi ra để kiểm tra, hiệu chỉnh và tính toán bổ sung các yếu tố kết cấu khác. Các *File* kết quả này nên là những *File Text*, cả phần tính toán phụ thêm và phần tạo *File* có thể được đặt trong cùng một chương trình con (xem thí dụ 2 ở mục 1.3.3).

1.2.10. Soạn thảo chương trình sử dụng *Unit Overlay*

Khi soạn thảo các chương trình tính toán thiết kế lớn thường phải dùng các *Overlay* (*Overlay* là các *Unit* nằm trên đĩa cho đến khi cần thiết sẽ nạp vào RAM, khi nạp chúng sẽ dùng chung bộ nhớ với các *Unit Overlay* khác). Sử dụng *Unit Overlay* cho phép phát triển chương trình lớn nằm ngoài khả năng cho phép 64 Kb bộ nhớ của TURBO PASCAL (các máy có bộ xử lý *32 Bit* như 80386 hoặc 80486 vẫn bị giới hạn ở mức *64 Kb* để tương thích với các thế hệ trước nó [15]).

Soạn thảo chương trình có dùng *Unit Overlay* tiến hành theo trình tự sau [14]:

1) Đổi các *Unit* thành các *Unit Overlay* bằng cách đưa dẫn hướng *{$O+,F+}* vào từng *Unit*, ngay phía dưới tên của *Unit*.

2) Biên dịch *Overlay* (tương tự như biên dịch *Unit*).

3) Đưa dẫn hướng *{$O+,F+}* lên đầu chương trình, trước tên của chương trình chính.

4) Chỉ định *Unit Overlay* trong khai báo *Uses* cùng các *Unit Overlay* riêng của chương trình và đưa dẫn hướng *{$O,X1}*, *{$O,X2}*, v.v... với *X1*, *X2*,... là tên của từng *Unit Overlay*.

5) Ở ngay đầu của thân chương trình chính cần gọi tới *OvrInit ('')* với tên của tập tin *Overlay* nằm trong hai dấu nháy trong ngoặc và thường cùng tên với chương trình nhưng có phần mở rộng là .OVR) để kiểm tra lỗi *Overlay* (xem mục 4.3.2 trong chương 4).

Khi chạy chương trình chính sẽ thu được hai tập tin .EXE và .OVR, trong đó tập tin .EXE chứa đoạn mã chương trình nằm thường trú trong bộ nhớ, còn tập tin .OVR chứa đoạn mã *Overlay* để trình quản lý *Overlay* nạp từ đĩa khi cần thiết.

1.2.11. Xử lý các chương trình lớn

Các chương trình thiết kế máy thường là những chương trình lớn, khi soạn thảo thường được chia thành nhiều đơn thể, nhiều chương trình con, v..v.., tuy phức tạp nhưng không khó khăn về mặt tổ chức chương trình. Tuy nhiên với giới hạn trên là 60000 ký tự, cần tìm giải pháp để trình xử lý văn bản của TURBO PASCAL có thể xử lý các chương trình lớn như vậy [14]:

1) Sử dụng thủ tục *Exec* của *Dos Unit* bằng cách:

- Chia chương trình lớn thành các đơn thể, tức là các chương trình con được thiết kế, biên dịch và kiểm tra độc lập nhưng có liên hệ với nhau thông qua các *file* dữ liệu (kết quả tính toán ở đơn thể này là dữ liệu tính toán cho đơn thể kia).

- Viết các trình điều khiển để gọi từng đơn thể. Ở đây thủ tục *Exec* cần hai tham số chuỗi để truyền tên của chương trình con cần chạy:

Exec(Path,Cmdline:string);

trong đó *Path* là tên của chương trình *EXE* hay *COM* cần chạy; *Cmdline* giữ bất kỳ thông tin nào cần truyền sang chương trình; trường hợp không có truyền thông tin thì để hai dấu nháy. Chẳng hạn ghi

Exec('C: Command. COM','');

- Hạn chế sử dụng bộ nhớ bằng cách đưa vào dòng đầu của chương trình dẫn hướng biên dịch đặc biệt sau đây:

{$M Stacksize,Heapmin,Heapmax}

trong đó *Stacksize* là số *Byte* có phạm vi từ thấp bằng 1024 đến cao là 65520; các giá trị *Heapmin* và *Heapmax* kiểm soát vùng nhớ còn trống bé nhất và lớn nhất.

Mỗi chương trình có yêu cầu *Stack* khác nhau phụ thuộc tổng số các cấp lồng nhau của tất cả các thủ tục và hàm cộng với số lượng và kiểu của các tham số truyền sang trình xử lý. Chẳng hạn có thể ghi

{$M 4000, 0, 0}

vào dòng đầu của chương trình chính.

- Lưu trữ các chương trình con độc lập và chương trình lớn gồm nhiều đơn thể trong tập tin *.PAS* rồi biên dịch thành *.EXE*.

2) Sử dụng tập tin *Include*:

Chia chương trình lớn thành nhiều phần nhỏ có thể soạn thảo được rồi chứa văn bản chương trình vào các tập tin *Include*. Đó là một tập tin văn bản riêng biệt trên đĩa sẽ được trình biên dịch đem vào tại một thời điểm nào đó khi biên dịch các văn bản khác. Muốn vậy cần đưa vào trước tên tập tin *Include* dẫn hướng {$I...}, khi trình biên dịch gặp dẫn hướng này nó sẽ đọc tập tin *Include* như một phần của chương trình gốc.

Chẳng hạn để trình biên dịch gộp tập tin có tên là *banhrang.pas* (là tập hợp các thủ tục và hàm mà chương trình 'thiết kế hệ dẫn động cơ khí' sử dụng), cần ghi

{$I.banhrang. pas}

1.2.12. Xây dựng giao diện cho chương trình

Thông thường số liệu được gán, kết quả từng bước hoặc các thông tin cần được kiểm tra, sẽ được lần lượt hiện ra màn hình từ trên xuống dưới, từ trái qua phải với cùng một màu đã định. Theo dõi các thông tin như vậy rõ ràng là không thuận tiện và không đáp ứng yêu cầu. Sử dụng các cửa sổ trên màn hình để tạo ra khung hiển thị với kiểu khung, đường viền, màu sắc và đường nét khác nhau, người thiết kế có điều kiện làm nổi bật nội dung cần được lưu ý, chẳng hạn như:

a) Gắn nhãn vào chương trình đang thực hiện.

b) Thể hiện các bước tính toán.

c) Hướng dẫn lựa chọn các thông số, kích thước.

d) Trình bày kết quả từng bước để có hướng xử lý phục vụ các bước tính tiếp theo.

e) Đối thoại với máy nhằm đáp ứng các yêu cầu trong thiết kế.

g) Nêu các chỉ dẫn cần thiết khi kết thúc một chương trình để chuyển sang phần tính toán thiết kế tiếp theo.

1.3. XÂY DỰNG CƠ SỞ DỮ LIỆU TRONG THIẾT KẾ CƠ KHÍ

Dữ liệu là những thông tin được máy tính xử lý, tồn tại dưới những dạng khác nhau và có ý nghĩa khác nhau. Dữ liệu thiết kế cơ khí có thể bao gồm: số liệu thiết kế (lực kéo hoặc công suất làm việc, vận tốc hoặc số vòng quay của trục công tác, v.v...), các đặc trưng cơ học của vật liệu (giới hạn bền, giới hạn chảy của vật liệu; độ rắn bề mặt sau nhiệt luyện, v.v...), bảng tra các hệ số trong các phương trình tính toán thiết kế (hệ số tải trọng, hệ số sử dụng, v.v...), các giá trị tiêu chuẩn của các thông số và kích thước chi tiết máy hoặc bộ phận máy (môđun của bánh răng, bước xích, đường kính bánh đai, kích thước ổ lăn, v.v...). Với các chương trình thiết kế tự động bản vẽ, dữ liệu thiết kế còn bao gồm các bảng tra miền dung sai và sai lệch giới hạn của trục và lỗ, trị số nhám bề mặt, kích thước lắp đặt của động cơ điện, v.v...

Để có thể sử dụng các loại dữ liệu trên đây phục vụ các nội dung tính toán thiết kế khác nhau, cần tiến hành phân loại, sắp xếp, ghi chép và lưu trữ các loại dữ liệu đó theo một phương pháp nhất quán, đảm bảo chính xác và tiết kiệm được bộ nhớ.

Dưới đây trình bày những vấn đề cần lưu ý khi xây dựng cơ sở dữ liệu cho thiết kế cơ khí.

1.3.1. Ghi số liệu lên đĩa và tra số liệu từ đĩa

Số liệu ở các bảng tra trong tài liệu thiết kế cơ khí có nội dung rất khác nhau, được sắp xếp theo số dòng i và số cột j không giống nhau. Chúng là các phần tử có kiểu khác nhau (thực, nguyên, ký tự, v.v...) nhưng có quan hệ với nhau. Để có thể sử dụng các bảng này cho các chương trình thiết kế cần soạn thảo các *Unit* ghi số liệu của các bảng lên các thiết bị nhớ ngoài (đĩa cứng, đĩa mềm) và các *Unit* tra số liệu từ đĩa.

1.3.1.1. Ghi các bảng số liệu lên đĩa

Có nhiều cách ghi các bảng số liệu lên đĩa để phục vụ các nội dung tính toán. Với ngôn ngữ Pascal dùng dữ liệu kiểu bản ghi (*File of Record*) là thích hợp hơn cả. Cấu trúc dữ liệu kiểu bản ghi được mô tả bắt đầu bằng *Record*, tiếp đến là các trường và kết thúc là *End*. Để thâm nhập vào các trường trong *Record* và để gọi tên trường gọn hơn thường dùng lệnh *With ... do...*

Thông qua *File* của tệp bản ghi, máy tính giao tiếp với bên ngoài nhờ thủ tục mở tập tin, đối với TURBO PASCAL là

Assign(biến_tệp, tên_tệp);

Sau khi mở tập tin với biến tệp, tên tệp, con trỏ của *File* viết vào một ô trên đĩa, thông qua lệnh

Rewrite(biến_tệp);

Lúc đầu con trỏ ở vị trí 0 (i = 0), muốn dịch chuyển con trỏ đến thành phần tiếp theo (i = i+1), dùng lệnh

Seek(biến_tệp, i);

để lần lượt viết số liệu thuộc các cột j trên cùng một dòng i của bảng lên màn hình, sau đó ghi từng dòng của tệp bản ghi vào *File*. cuối cùng để không mất thông tin, cần đóng các tập tin lại nhờ thủ tục

Close(biến_tệp);

Như vậy bất kỳ một bảng số liệu nào cũng đều có thể được ghi vào đĩa. Tuy nhiên do có sự khác nhau về *số lượng* các dòng và các cột giá trị của bảng và *kiểu* dữ liệu (nguyên, thực, chuỗi, mảng,...) cho nên cần quy cách hoá và thống nhất hoá các bảng để từ đó có thể mô tả chúng bằng một vài dạng *Record*, tạo thuận lợi cho việc lập trình ghi số liệu lên đĩa.

Khảo sát các bảng tra trong các tài liệu thiết kế cơ khí có thể đưa về hai dạng cơ bản sau đây để viết *Program* ghi các bảng số liệu lên đĩa:

a) Bản ghi *một trường*, đó là các *mảng số thực* (coi số nguyên là một

trường hợp của số thực) để ghi các bảng chỉ gồm toàn *con số*. Các mảng số thực là loại mảng một chiều (*Array [0.. n] of real*), trong đó $n \leq 8$.

b) Bản ghi *hai trường* gồm *biến chữ* và *biến số* với định nghĩa *biến chữ* là một *mảng xâu ký tự* (*Array [0.. p] of string*) với $p \leq 5$ và *biến số* là một *mảng số thực* một chiều với $n \leq 8$.

Việc hạn chế n và p như trên vừa phù hợp với số cột của phần lớn các bảng trong tài liệu thiết kế vừa giảm nhẹ được dung lượng cho chương trình. Trường hợp một vài bảng nào đó có số cột lớn hơn 9 có thể tách ra thành hai bảng để ghi vào đĩa, chẳng hạn bảng tra hệ số dạng răng (xem bảng 6.18 [2]) có thể tách thành 2 bảng ứng với 2 trường hợp hệ số dịch chỉnh $x > 0$ và $x \leq 0$.

1.3.1.2. Tra số liệu từ các bảng trên đĩa

Từ các bảng số liệu trên đĩa, để lấy ra một hoặc nhiều số liệu cần thiết phục vụ tính toán thiết kế, cần phải tìm ra phương pháp tra bảng thích hợp. Thông thường khi đọc một tệp dữ liệu, con trỏ được đặt ở vị trí đầu tiên (số liệu đầu tiên của *Record* là 0), mỗi biến được đọc lần lượt từ *File* trên đĩa và sau mỗi lần đọc, con trỏ tự động di chuyển đến vị trí (số hiệu) tiếp theo của *Record*. Rõ ràng ở đây không thể dùng cách truy xuất ngẫu nhiên mà phải tìm ra các điều kiện tra bảng và dựa vào các điều kiện này để ghi ra các số liệu cần cho tính toán.

Tuy nhiên điều kiện tra bảng liên quan đến *bản chất* của các số liệu và *số lượng* của các yếu tố mà chúng phụ thuộc.

Thật vậy, trong các bảng tra, số lượng của các yếu tố chi phối các số liệu rất khác nhau, trong khi số liệu có thể được dùng để biểu thị các tính chất sau đây:

1) Các đặc trưng cơ học (cơ tính của vật liệu) như giới hạn bền, giới hạn chảy, giới hạn mỏi,... được xác định bằng thực nghiệm, phụ thuộc vào loại vật liệu, phương pháp nhiệt luyện, kích thước phôi.

2) Các giá trị cho phép như công suất cho phép đối với bộ truyền đai hoặc bộ truyền xích, áp suất cho phép đối với ổ trượt, v. v..., được xác định bằng thực nghiệm ứng với các điều kiện thí nghiệm nhất định.

3) Các hệ số trong các phương trình tính toán (chẳng hạn hệ số tải trọng, hệ số sử dụng, v.v...). Các giá trị này được tra trong các bảng tương ứng tuỳ thuộc vào quan hệ kết cấu và điều kiện làm việc cụ thể, chẳng hạn dựa vào giá trị của vận tốc vòng và cấp chính xác để tra hệ số tải trọng động khi tính toán độ bền bánh răng.

4) Các giá trị tiêu chuẩn của các thông số và kích thước của các chi

tiết máy và bộ phận máy trích từ các tiêu chuẩn nhà nước (TCVN), tiêu chuẩn ngành (TCN), tiêu chuẩn vùng, thành phố (TCV) hoặc từ các tiêu chuẩn quốc tế (ISO) cũng như các tiêu chuẩn Liên Xô cũ (ГОСТ).

Như vậy, từ hai loại bảng đã được ghi vào đĩa, khi soạn thảo các *Program* tra bảng phục vụ các chương trình tính toán khác nhau, cần phân biệt nội dung và bản chất của các số liệu trong bảng cũng như số lượng các yếu tố mà chúng phụ thuộc. Có thể đưa về mấy dạng tra bảng sau đây và dựa vào chúng để viết các chương trình con tra bảng tương ứng:

a) Dựa vào một yếu tố tính toán s_v (trị số danh nghĩa) để tra ra thông số cần thiết tại cột k với điều kiện giá trị nằm tại cột j nào đó phải lớn hơn hoặc bằng s_v (thí dụ dựa vào công suất tính toán P_t để tra ra bước xích p với điều kiện công suất cho phép $[P] \geq P_t$).

b) Dựa vào một trị số tính toán s_v để tra ra hai giá trị ở hai dòng, hoặc dựa vào hai trị số để tra ra bốn giá trị ở hai dòng và hai cột, sau đó tiến hành nội suy để xác định số liệu cần tìm. Thí dụ, nội suy hai dòng để xác định hệ số phân bố không đều tải trọng trên chiều dài tiếp xúc $K_{H\beta}$ (bảng 6.7 [2]) hoặc nội suy hai dòng hai cột để xác định hệ số dạng răng Y_F (bảng 6.18 [2]).

c) Dựa vào một trị số tính toán s_v để tra giá trị gần nhất trong dãy số tiêu chuẩn (thí dụ, tra môđun m và hệ số đường kính q của trục vít - bảng 7.3 [2]).

d) Dựa vào hai trị số tính toán để tra một loạt các thông số (thí dụ, dựa vào khả năng tải động C và đường kính ngõng trục d để tra ra các thông số của ổ lăn tiêu chuẩn như đường kính trong d, đường kính ngoài D, chiều rộng ổ B, v.v...).

e) Dựa vào các *biến chữ* (chẳng hạn, tên vật liệu và cách nhiệt luyện) và *biến số* (tiết diện phôi) để tra cơ tính của vật liệu chế tạo bánh răng (bảng 6.1 [2]).

Về cấu trúc, chương trình tra số liệu từ đĩa cũng được bắt đầu bằng lệnh mở tập tin với các biến tệp, tên tệp, tiếp đến thay cho thủ tục

Rewrite(biến_tệp);

trong chương trình ghi số liệu lên đĩa, ta dùng thủ tục

Reset(biến_tệp);

sau đó đưa con trỏ của *File* từ vị trí đầu tiên (vị trí i = 0) đến thành phần tiếp theo (i = i+1) cũng bằng lệnh

Seek(biến_tệp, i);

để đọc các số liệu từ *File* nhằm thực hiện một cách tra bảng nào đó, cuối cùng để không mất thông tin, cũng dùng thủ tục đóng tập tin:

Close(biến_tệp);

1.3.1.3. Kiểm tra các bảng số liệu đã được ghi vào đĩa

Bằng các cách tra bảng khác nhau có thể đọc ra màn hình một giá trị hay một dòng bất kỳ của các bảng số liệu đã được ghi vào đĩa. Tuy nhiên cách kiểm tra ngẫu nhiên này chưa thể đảm bảo rằng những số liệu ở các vị trí khác đã được ghi chính xác. Vì vậy trước khi sử dụng các bảng phục vụ các chương trình tính toán nên viết một chương trình con dùng chung để kiểm tra toàn bộ các dòng và các cột của từng bảng và nếu có sai sót, cần bổ sung và sửa chữa các số liệu cho chính xác. Chỉ có bằng biện pháp này người lập trình mới chắc chắn rằng các dữ liệu tính toán là đáng tin cậy.

Về cấu trúc, chương trình kiểm tra các bảng số liệu đã được ghi vào đĩa cũng tương tự như chương trình tra bảng ở các phần mở *File*, thâm nhập vào *File* và đóng *File*, chỉ khác là không dùng lệnh di chuyển con trỏ - *Seek*(biến_tệp, i) - để đọc từng dòng của bảng mà ở đây máy thực hiện các lệnh đọc và ghi toàn bộ các số liệu của bảng ra màn hình để phục vụ mục đích kiểm tra.

1.3.2. Vào ra dữ liệu và kết quả tính toán

Dữ liệu đầu vào của các chương trình thiết kế máy có thể do bên đặt hàng đề ra, cũng có thể do người thiết kế căn cứ vào yêu cầu của bên đặt hàng và điều kiện thực tế để xác định. Trường hợp đơn giản nhất, dữ liệu đầu vào x được đưa vào chương trình tính toán qua bàn phím nhờ cặp thủ tục

```
Write(' x = ');
Readln(x);
```

đồng thời kết quả tính toán (số liệu đầu ra) cũng được viết ra màn hình nhờ thủ tục

Write(' x = ', x:5:3); hoặc *Writeln(' x = ', x:5:3);*

Cách vào dữ liệu như vậy không thuận tiện vì sau mỗi lần sửa lỗi lại phải vào lại dữ liệu qua bàn phím, tốn thời gian. Mặt khác đối với các chương trình lớn, dữ liệu đầu vào của các chương trình con không như nhau, kết quả tính toán ở chương trình trước có thể là dữ liệu tính toán của chương trình sau hoặc không thay đổi trong suốt quá trình tính toán. Trái lại, nếu nạp các dữ liệu này vào các *File* trên đĩa thì khi sử dụng có thể gọi trực tiếp ra màn hình để kiểm tra hoặc thay đổi, bổ sung thêm số liệu

rồi mới gọi ra để dùng, như vậy vừa thuận tiện, vừa đảm bảo được sự chính xác của dữ liệu tính toán.

Sử dụng tập tin văn bản, còn gọi là tệp văn bản *(Text File)* sẽ đáp ứng được yêu cầu trên đây của người thiết kế.

Tập tin văn bản có những đặc điểm sau đây:

1) Tệp văn bản được định nghĩa bằng từ chuẩn TEXT, được tổ chức thành từng dòng với độ dài mỗi dòng khác nhau. Chính vì vậy có thể coi tệp văn bản gần đúng là tệp bản ghi có độ dài thay đổi.

2) Tuy chứa các ký tự nhưng nhờ các thủ tục *Readln* và *Writeln*, tệp văn bản có thể đọc và ghi cả số nguyên, số thực, xâu ký tự *(String)* hoặc biến lôgic *(Boolean)* nhờ sự chuyển đổi tương ứng từ các giá trị này sang các dãy ký tự. Nói cách khác có thể đọc tệp văn bản như đọc các trang viết thông thường, điều này rõ ràng là không thể thực hiện trên tệp bản ghi.

3) Ở tệp văn bản có thể viết từng biến hoặc viết đồng thời một số biến vào biến tệp *(Filevar)*.

Tập tin văn bản có thể được tổ chức theo những dạng sau đây:

a) Vào số liệu qua bàn phím rồi viết vào tệp văn bản, trong đó tệp văn bản được mở đầu và kết thúc như tệp bản ghi, tức là cần dùng cặp thủ tục

Assign(biến_tệp, tên_tệp);
Rewrite(biến_tệp);

khi mở *File* và thủ tục

Close(biến_tệp);

khi đóng *File*.

Khi viết vào tệp văn bản cần chú ý dùng thủ tục *Writeln*(tên_tệp) nhằm đưa dấu hết dòng vào tệp. Trường hợp một số biến được viết trên cùng một dòng , phải cho trường với độ dài đủ lớn nhằm đảm bảo cho các số liệu tách biệt nhau, tránh gây nhầm lẫn khi máy đọc ra các số liệu này.

b) Nhập kết quả tính toán đã được thực hiện sau một giai đoạn tính toán xác định vào tệp văn bản, ở đây là *File* kết quả, để phục vụ các chương trình tính toán tiếp theo.

c) Bổ sung các kết quả vào tệp văn bản đã có bằng cách mở tập tin có sẵn để viết thêm vào nhờ cặp thủ tục

Assign(biến_tệp, tên_tệp);
Append(biến_tệp);

Phần còn lại cũng được thực hiện như khi nhập kết quả.

d) Gọi số liệu từ các tập tin đã được ghi vào đĩa để sử dụng cho chương trình tính toán hiện hành nhờ cặp thủ tục

Assign(biến_tệp, tên_tệp);
Reset(biến_tệp);

rồi viết ra màn hình nhờ cặp thủ tục

Readln(biến_tệp, tên_biến);
Writeln(tên_biến);

Khi vào ra dữ liệu cần lưu ý những điểm sau đây:

1) Nên sắp xếp các biến cùng kiểu để có thể ghi vào và đọc ra lần lượt một số biến cùng kiểu này rồi mới ghi và đọc các biến kiểu khác.

2) Thứ tự các biến cùng kiểu hoặc khác kiểu khi ghi vào tệp văn bản và đọc ra phải giống nhau.

3) Khi viết các biến thực vào *File* phải cho trường đủ rộng để tránh nhầm lẫn khi đọc ra.

1.3.3. Thí dụ

Thí dụ 1 :

Soạn thảo một *Program* để :

- Ghi dữ liệu lên đĩa.
- Kiểm tra số liệu đã được ghi lên đĩa.
- Tra số liệu từ đĩa.

Bảng số liệu có số dòng là k và gồm các chữ được sắp xếp trên jc cột và các con số được sắp xếp trên js cột.

Thực hiện :

Chương trình vào bảng, kiểm tra các số liệu của bảng và tra bảng gồm cả chữ và số được viết như sau:

```
1 : Program vkttb;
2 : uses crt;
3 : type   st12= string[12]
4          mata = array [0..8] of real ;
5 :        matb = array [0..5] of st12;
6 :    thongso = record
7 :                    bc: matb;
8 :                    bs: mata;
9 :                  end;
10 : Var  triso : thongso ;
```

```
11 :       filecs: file of thongso ;
12 :         i, j, k: byte ;
13 :       drmin, drmax, xmb, xmc, kq: mata;
14 :       vl,nl : matb;
{-------------------------------------------------------}
15 : Procedure vaobcs (jc,js:byte; tenfile: string[8]);
16 : begin
17 :   assign(filecs,'c:\ bang\'+tenfile);
18 :   rewrite(filecs);
19 :   i: = 0;
20 :   while i<=k do
21 :   begin
22 :     seek(filecs,i);
23 :     with triso  do
24 :     begin
25 :       for j:=1 to jc do
26 :       begin
27 :         write(' bc[ ',j , '] = '); readln(bc[j ] );
28 :       end;
29 :       for j:=1 to js do
30 :       begin
31 :         write(' bs[ ',j , '] = '); readln(bs[j ] );
32 :       end;
33 :     end;
34 :     write(filecs, triso);
35 :     i: = i+1;
36 :   end;
37 :   close(filecs);
38 : end.
{-------------------------------------------------------}
39 : Procedure ktbcs (jc,js: byte; tenfile: string[8]);
40 : var  bkt:string[8] ; kiem : boolean;
41 : begin
42 :   bkt:=' ';
43 :   assign(filecs,'c:\ bang\'+tenfile);
44 :   reset(filecs);
45 :   gotoxy(1,24);
46 :   kiem:=true;
47 :   while kiem do
48 :   begin
49 :     seek(filecs, i-1);
50 :     str(i, texti);
51 :     read(filecs, triso);
52 :     with triso  do
```

```
53 :      begin
54 :       goto(2,1);
55 :       write(' i ',bkt,' ');
56 :       for j:=1 to jc do write(' bc[',j,'] = ',bkt,' ');
57 :       for j:=1 to js do write(' bs[',j,'] = ',bkt,' ');
58 :       gotoxy(2,2);
59 :       write('-----------------------------------------------');
60 :       gotoxy(2, 2+i);
61 :       write(bkt,i:(2-length(texti)));
62 :       for j:=1 to jc do write(bkt,bc[j]:(12-length(texti)));
63 :       for j:=1 to js do write(bkt,bs[j]:(9-length(texti)));
64 :       i:=i+1;
65 :       writeln;
66:        if i = filesize(filecs)+1 then kiem:=false;
67 :      end;
68 :      close(filecs);
69 :   end;
70 : end;
{-------------------------------------------------------------------------------}
71 : Procedure trabcs (j:byte; bc1, bc2: st12; sv:real; tenfile: string[8];
                          var sr: math);
72 : var  xong : boolean;
73 : begin
74 :   assign(filecs,'c:\ bang\'+tenfile);
75 :   reset(filecs);
76 :   i: = 0 ; xong: = false;
77 :   while not(xong) do
78 :   begin
79 :    seek(filecs, i);
80 :    read(filecs,triso);
81 :    with triso  do
82 :    begin
83 :     if (bc1<>bc[1]) or (bc2<>bc[2]) or (sv<>bs[1]) then i:= i+1 else
84 :     begin
85 :       for j:=1 to jj do sr[j]:=bs[j];
86 :       xong: = true;
87 :     end;
88 :    end;
89 :  end;
90 :  close(filecs);
91 : end;
{-----------------------------------------}
92 : Begin
93 :   clrscr;
```

```
94 :    write(' k  = ') ; readln(k);
95 :    vaobcs(2,5,'b24.br');
96 :    ktbcs(2,5,'b24.br');
97 :    trabcs(5, vl[1], nl[1], td[1], 'b24.br', kq);
98 :    drmin: = kq[2] ; drmax: = kq[3] ; xmb: = kq[4] ; xmc: = kq[5];
99 :    writeln(' drmin= ', drmin:4:0, ' drmax= ', drmax: 4: 0);
100 :   writeln(' xmb = ', xrmb:4:0, ' xrmc= ', xrmc: 4: 0);
101 :   readkey;
102 : End.
```

Chú thích

- Dòng 11: *filecs* là một tệp bản ghi có hai trường.

- Dòng 15: chương trình con *vaobcs* sử dụng phương pháp truyền dữ liệu. Khi gọi tới *Procedure* này để ghi một bảng lên đĩa sẽ thay *jc* bằng số cột biến chữ thực tế và *js* bằng số cột biến số thực tế của bảng này nhưng *jc* ≤ 5 và js ≤ 8, đồng thời sẽ thay *tenfile* bằng tên bảng hoặc số thứ tự của bảng cần ghi. Tương tự như vậy đối với các chương trình con *ktbcs* (dòng 39) và *trabcs* (dòng 71).

- Dòng 17: mở một *File* để gán tệp cho một biến kiểu *File*. Một *File* có tên là *filecs* đã được tạo ra và sẵn sàng xử lý.

- Dòng 18: con trỏ của *File* bắt đầu để viết vào một ô trên đĩa.

- Dòng 22: dịch chuyển con trỏ tới thành phần thứ i của *File* (vị trí đầu tiên của *File* là 0).

- Dòng 23 đến 33: gán các biến chữ bc[j] và các biến số bs[j] trên màn hình.

- Dòng 34: viết từng dòng vào *File*.

- Dòng 94: gán k theo số dòng thực tế của bảng cần ghi vào đĩa ;

- Dòng 96: kiểm tra một bảng gồm có số cột chữ jc = 2 và số cột con số js = 5 với tên tệp là *b24.br*. Đó chính là bảng 6.1 của tài liệu [2], cho biết cơ tính của một số loại vật liệu chế tạo bánh răng. Ở đây cột độ rắn đã được tách thành hai cột chỉ độ rắn min và độ rắn max.

- Dòng 97: tra cơ tính cho vật liệu bánh răng 1 ở *File* có tên là *b24.br* với số biến số j = 5 ứng với các điều kiện: bc1= vl[1], bc2 = nl[1]; sv = td[1].

Thí dụ 2 :

Soạn thảo các chương trình con ghi và đọc dữ liệu tính toán truyền động bánh răng.

Thực hiện :

```
Procedure stbr(a,b:byte);
var h0,hgt:real;
begin
  gotoxy(4,b); write('Công suất          Pt[1](kW) =');b:=b+1;
  gotoxy(4,b); write("Số vòng quay      vq[1](vg/ph) =');b:=b+1;
  gotoxy(4,b); write('Ti số truyền              u =');b:=b+1;
```

```
gotoxy(4,b); write('Thời hạn sử dụng                    th =');b:=b+1;
gotoxy(4,b); write('Số chiều quay của bánh răng  socq =');b:=b+1;
gotoxy(4,b); write('Số bậc tải trọng                 sob  =');b:=b+1;
gotoxy(4,b);
a:=41;b:=3;
textcolor(11);
gotoxy(a,b); readln(Pt[1]); b:=b+1;
gotoxy(a,b); readln(vq[1]);b:=b+1;
gotoxy(a,b); readln(u); b:=b+1;
gotoxy(a,b); readln(th); b:=b+1;
gotoxy(a,b); readln(socq); b:=b+1;
if sob>1 then
begin
 for i:=1 to sob do
 begin
  textcolor(15); gotoxy(4,b);
  write('Hệ số tải trọng làm việc    ht[',i,'] = '); textcolor(11);
  readln(ht[i]);
  b:=b+1;
 end;
 for i:=1 to sob-1 do
 begin
  textcolor(15); gotoxy(4,b);
  write('Hệ số thời gian làm việc    hg[',i,'] = '); textcolor(11);
  readln(hg[i]);
  b:=b+1;
 end;
 hgt:=0;for i:=1 to sob-1 do hgt:=hgt+hg[i];
 hg[sob]:=1-hgt;
 gotoxy(4,b); textcolor(15);
 write('Hệ số thời gian làm việc    hg[',sob,'] = ',hg[sob]:3:2);
 readln;
end;
end;
{=====================================}
Procedure goisl(a,b:byte);
begin
 assign(ftk,'c:\hedd\dl\'+filename);
 reset(ftk);
 readln(ftk, Pt[1], vq[1], u, th, socq, sob);
 if sob>1 then
 begin
  for i:=1 to sob do readln(ftk,ht[i]);
  for i:=1 to sob do readln(ftk,hg[i]);
```

```
end;
close(ftk);
gotoxy(4,b); write(' Công suất          Pt[1](kW)  = ',Pt[1]:4:2); b:=b+1;
gotoxy(4,b);write(' Số vòng quay    vq[1](vg/ph) = ',vq[1]:5:0); b:=b+1;
gotoxy(4,b); write(' Tỉ số truyền                u = ',u:5:2); b:=b+1;
gotoxy(4,b); write(' Thời hạn sử dụng            th = ',th:5:0); b:=b+1;
gotoxy(4,b); write(' Số chiều quay của br      socq = ',socq); b:=b+1;
gotoxy(4,b); write(' Số bậc tải trọng          sob  = ',sob); b:=b+1;
gotoxy(4,b);
if sob>1 then
begin
 for i:=1 to sob do
 begin
  textcolor(15);
  gotoxy(4,b);
  write(' Hệ số tải trọng làm việc    ht[',i,'] = ', ht[i]:3:2); b:=b+1;
 end;
 for i:=1 to sob do
 begin
  textcolor(15);
  gotoxy(4,b);
  write(' Hệ số thời gian làm việc    hg[',i,'] = ', hg[i]:3:2); b:=b+1;
 end;
 textbackground(15);textcolor(12+blink); gotoxy(4,b);
 write(' Ấn một phím bất kỳ để tiếp tục ! ');
 ch:=readkey;
end;
end;
{=====================================}
Procedure nhapsolieu;
begin
 slbr(41,3);
 gotoxy(3,1);writeln(' Có ghi số liệu vào đĩa không(C/K) ? ');
 readln(ch);
 if upcase(ch)='C' then
 begin
  assign(ftk,'c:\hedd\dl\'+filename);
  rewrite(ftk);
  writeln(ftk, Pt[1],' ',vq[1],' ', u,' ', th,' ', socq,' ', sob);
  if sob>1 then
  begin
   for i:=1 to sob do writeln(ftk,ht[i]:3:2);
   for i:=1 to sob do writeln(ftk,hg[i]:3:2);
```

```
    end;
    close(ftk);
  end;
end;
{=====================================}
procedure ghisl;
begin
  write(' Cho dữ liệu từ đĩa hay từ bàn phím (D/F )? '); read(ch);
  clrscr;
  gotoxy(9,2); textcolor(yellow);
  write('     SỐ LIỆU CHO TRƯỚC');
  textcolor(white);
  if upcase(ch)=' F ' then nhapsolieu else goisl(41,3);
end;
```

Chú thích

1) Các biến tổng thể *Pt[1], vq[1], u, th, socq, sob* được định nghĩa một lần ở chương trình chính tính bánh răng hoặc ở *Unit* đầu tiên mà chương trình này sử dụng.

2) Tên của tệp dữ liệu (*Filename*) phục vụ tính toán bánh răng có thể là *'br.slt'* được nạp vào một *File* trong thư mục *'c:\hedd\dl*.

3) *ftk*: *Text File*.

4) Để thuận tiện cho việc sửa lỗi khi lập trình, nên sử dụng cùng một số liệu đã được nạp vào *File*, khi đó ở chương trình con *ghisl* cần gọi tới *Procedure goisl*, trái lại nếu muốn biết phần mềm đã được soạn thảo có thu được kết quả chính xác ứng với các dữ liệu thay đổi trong một phạm vi rộng hay không thì gọi tới chương trình con *nhapdulieu* để nhập dữ liệu mới từ bàn phím. Bộ dữ liệu mới này có thể được ghi hoặc không ghi vào file dữ liệu trên đĩa (xem *Procedure nhapsolieu*). Ngoài ra trong *Procedure* này đã dùng hai dấu nháy (' ') để tách các dữ liệu khi chúng được ghi vào *File* trên cùng một dòng nhằm tránh nhầm lẫn giữa các dữ liệu.

5) Hệ số tải trọng làm việc ht[i] là tỉ số của tải trọng tác dụng trong chế độ i và tải trọng lớn nhất T[1] : *ht[i] = T[i]/ T[1];*

Hệ số thời gian làm việc *hg[i]* là tỉ số của thời gian tác dụng của tải trọng *T[i]* trong chế độ i và thời gian tác dụng của một chu kỳ làm việc.

Thí dụ 3 :

Soạn thảo các chương trình con ghi và đọc kết quả tính toán truyền động bánh răng phục vụ việc vẽ chế tạo các bánh răng trong bộ truyền.

Thực hiện :

```
Procedure ghikqbrt(tenfile:string[7]);
begin
```

```
 for i:=1 to 2 do dc[i]:=d[i];
 assign(ftk,'c:\hedd\dl\'+tenfile);
 rewrite(ftk);
 writeln(ftk,T[1]:8:0,T[2]:8:0);
 writeln(ftk,loair:3,ccx:3);
 writeln(ftk,bw:5:2, m:5:1, beta:7:4);
 for i:=1 to 2 do writeln(ftk, dc[i]:7:2,,da[i]:7:2, df[i]:7:2);
 for i:=1 to 2 do writeln(ftk, z[i]:5, x[i]:5:2);
 for i:=1 to 2 do writeln(ftk,vl[i]);
close(ftk);
end;
{=================================}
Procedure goikqtbr(tenfile:ten);
begin
   assign(ftk,'c:\hedd\dl\'+tenfile);
   reset(ftk);
   readln(ftk,T[1], T[2]);
   readln(ftk, loair, ccx);
   readln(ftk, bw, m, beta);
   for i:=1 to 2 do readln(ftk, dc[i], da[i], df[i]);
   for i:=1 to 2 do readln(ftk, z[i], x[i]);
   for i:=1 to 2 do readln(ftk, vl[i]);
   close(ftk);
end;
```

Chú thích

1) Trước khi mở *File* để ghi các thông số và kích thước cần thiết, trong *Procedure ghikqbrt* đã tiến hành chuyển đổi đường kính chia d[i] của bánh răng thành ký hiệu dc[i] để tránh nhầm lẫn với d[i] sẽ dùng để ký hiệu đường kính các đoạn trục lắp bánh răng.

2) ftk: *Text File*.

3) Sử dụng các ký hiệu sau đây để chỉ các thông số của bộ truyền bánh răng:

loair	: loại răng (răng thẳng,răng nghiêng);
ccx	: cấp chính xác khi cắt răng;
bw,m, beta	: lần lượt là chiều rộng vành răng, môđun và góc nghiêng của răng;
dc[i],da[i],df[i]	: lần lượt là đường kính chia, đường kính đỉnh và đường kính đáy răng bánh răng;
z[i],x[i])	: lần lượt là số răng và hệ số dịch chỉnh bánh răng;
vl[i])	: vật liệu chế tạo bánh răng;

Các biến trên đây dùng chung cho cả hai Procedure, do đó cần được khai báo ở đầu chương trình.

Chương 2

XÂY DỰNG CHƯƠNG TRÌNH TÍNH TOÁN THIẾT KẾ TRUYỀN ĐỘNG CƠ KHÍ

Truyền động cơ khí là những cơ cấu truyền năng lượng từ động cơ điện đến các bộ phận công tác, thông thường có biến đổi vận tốc, lực hoặc mômen và đôi khi cả quy luật chuyển động. Trong tính toán thiết kế cũng như khi đánh giá truyền động cơ khí người ta sử dụng các thông số cơ bản sau đây:

- Công suất trên trục chủ động (kW) P_1 và trên trục bị động P_2.
- Hiệu suất truyền động $\eta = P_2 / P_1$.
- Số vòng quay trong một phút (vg/ph) n_1 trên trục chủ động và n_2 trên trục bị động.
- Tỉ số truyền $u = n_1/n_2$.
- Mômen xoắn (Nmm) trên trục chủ động T_1 và trên trục bị động T_2.

Dưới đây sẽ trình bày các công thức dùng trong tính toán động học, dùng để xác định các thông số và kích thước bộ truyền cũng như để tính toán khả năng tải, thuật giải dùng trong lập trình tính toán thiết kế và những nội dung cần lưu ý khi xây dựng phần mềm tính toán nhằm xác định tối ưu các thông số và kích thước cơ bản của các truyền động cơ khí:

- Truyền động bánh răng thường và bánh răng hành tinh.
- Truyền động trục vít.
- Truyền động đai.
- Truyền động xích.

2.1. XÂY DỰNG PHẦN MỀM TÍNH TOÁN THIẾT KẾ TRUYỀN ĐỘNG BÁNH RĂNG

2.1.1. Các quan hệ hình học và công thức tính toán độ bền

Các thông số cơ bản của bộ truyền bánh răng trụ và bánh răng côn phục vụ thiết kế được trình bày trong bảng 2.1 và bảng 2.2. Các công thức thiết kế các bộ truyền này về độ bền tiếp xúc cho trong bảng 2.3 và bảng 2.4, còn các công thức kiểm nghiệm các bộ truyền này về độ bền tiếp xúc và độ bền uốn được ghi trong bảng 2.5 và bảng 2.6.

Bảng 2.1. Các thông số cơ bản của bộ truyền bánh răng trụ

Tên thông số	*Ký hiệu*	*Công thức tính*
Khoảng cách trục chia	a	$a = 0,5(d_1 \pm d_2) = 0,5m\ (z_2 \pm z_1)/\cos\beta$
Khoảng cách trục	a_w	$a_w = a\cos\alpha_t / \cos\alpha_{tw}$ $a_w = a + ym = a + (x_2 \pm x_1 - \Delta y)\ m$
Đường kính chia	d	$d_1 = m\ z_1/\cos\beta$; $d_2 = m\ z_2/\cos\beta$
Đường kính lăn	d_w	$d_{w1} = 2a_w / (u \pm 1)$ hoặc $d_{w1} = d_1 + [2y/(z_2 \pm z_1)]d_1$ $d_{w2} = ud_{w1}$ hoặc $d_{w2} = d_2 + [2y/(z_2 \pm z_1)]d_2$
Đường kính đỉnh	d_a	*Ăn khớp ngoài :* $d_{a1} = d_1 + 2(1 + x_1 - \Delta y)m$; $d_{a2} = d_2 + 2(1 + x_2 - \Delta y)\ m$ *Ăn khớp trong :* $d_{a1} = d_1 + 2(1 + x_1)\ m$; $d_{a2} = d_2 - 2(1 - x_2 + \Delta y - k_2)\ m$ với $k_2 = 0.25 - 0,125x_2$ khi $x_2 < 2$ và $k_2 = 0$ khi $x_2 \geq 2$
Đường kính đáy răng	d_f	*Ăn khớp ngoài :* $d_{f1} = d_1 - (2,5 - 2x_1)m$ $d_{f2} = d_2 - (2,5 - 2x_2)m$ *Ăn khớp trong :* $d_{f2} = 2a_w + d_{a1} + 0,5\ m$
Đường kính cơ sở	d_b	$d_{b1} = d_1\cos\alpha$; $d_{b2} = d_2\cos\alpha$
Góc prôfin gốc	α	*Theo TCVN 1065-71* : $\alpha = 20°$
Góc prôfin răng	α_t	$\alpha_t = \text{arctg}\ (\text{tg}\alpha / \cos\beta)$
Góc ăn khớp	α_{tw}	$\alpha_{tw} = \arccos\ (a\cos\alpha_t / a_w)$
Tổng hệ số dịch chỉnh	x_t	$x_t = [(z_2 + z_1)(\text{inv}\alpha_{tw} - \text{inv}\alpha_t)]/(2\text{tg}\alpha)$
Hiệu hệ số dịch chỉnh	x_h	$x_h = [(z_2 - z_1)(\text{inv}\alpha_{tw} - \text{inv}\alpha_t)]/(2\text{tg}\alpha)$ với $\text{inv}\alpha = \text{tg}\alpha - \alpha$.
Hệ số trùng khớp ngang	ε_α	$\varepsilon_\alpha = \varepsilon_{\alpha 1} + \varepsilon_{\alpha 2}$, trong đó: $\varepsilon_{\alpha 1} = z_1(\text{tg}\alpha_{a1} - \text{tg}\alpha_{tw})/(2\pi)$ $\varepsilon_{\alpha 2} = z_2(\pm\text{tg}\alpha_{a2} \mp \text{tg}\alpha_{tw})/(2\pi)$

Chú thích : Ở các biểu thức trong bảng, dấu phía trên ứng với ăn khớp ngoài, dấu phía dưới ứng với ăn khớp trong.

Bảng 2.2. Các thông số cơ bản của bộ truyền bánh răng côn

Tên thông số	*Ký hiệu*	*Công thức tính*
Chiều dài côn ngoài	R_e	$R_e = 0{,}5m_{te}\sqrt{z_1^2 + z_2^2}$
Chiều rộng vành răng	b	$b = K_{be}R_e$
Chiều dài côn trung bình	R_m	$R_m = R_e - 0{,}5b$
Đường kính chia ngoài	d_e	$d_{e1} = m_{te}z_1$; $d_{e2} = m_{te}z_2$
Góc côn chia (lăn)	δ	$\delta_1 = \text{arctg}(z_1/z_2)$; $\delta_2 = 90° - \delta_1$
Chiều cao răng ngoài	h_e	$h_e = 2h_{te}m_{te}+c$ với $h_{te} = \cos\beta_m$; $c = 0{,}2m_{te}$
Chiều cao đầu răng ngoài	h_{ae}	$h_{ae1} = (h_{te} + x_{n1}\cos\beta_m)m_{te}$; $h_{ae2} = 2h_{te}m_{te} - h_{ae1}$
Hệ số dịch chỉnh	x_{n1}	$x_{t1} = 2(1 - 1/u^2)\sqrt{\cos^3\beta_m / z_1}$ hoặc tra bảng 6.20 [2]
Chiều cao chân răng ngoài	h_{fe}	$h_{fe1(2)} = h_e - h_{ae1(2)}$
Đường kính đỉnh răng ngoài	d_{ae}	$h_{ae1(2)} = d_{e1(2)} + 2h_{e1(2)}\cos\delta_{1(2)}$
Chiều dày răng ngoài	s_e	$s_{e1} = (0{,}5\pi + 2x_{n1}\text{tg}\alpha_n + x_{t1})m_{te}$; $s_{e2} = \pi m_{te} - s_{e1}$
Góc chân răng	θ_f	$\theta_{f1(2)} = \text{arctg}(h_{f1(2)}/R_e)$
Góc đầu răng	θ_a	$\theta_{a1} = \theta_{f2}$; $\theta_{a2} = \theta_{f1}$
Góc côn đỉnh	δ_a	$\delta_{a1(2)} = \delta_{1(2)} + \theta_{a1(2)}$
Góc côn đáy	δ_f	$\delta_{f1(2)} = \delta_{1(2)} - \theta_{f1(2)}$
Đường kính trung bình	d_m	$d_{m1(2)} = (1 - 0{,}5b/R_e)\, d_{e1(2)} = (1 - 0{,}5K_{be})\, d_{e1(2)}$
Khoảng cách từ đỉnh côn đến mặt phẳng đỉnh răng ngoài	B	$B_{1(2)} = R_e\cos\delta_{1(2)} - h_{ae1(2)}\sin\delta_{1(2)}$
Môđun vòng trung bình		$m_{tm} = m_{te}(1 - 0{,}5K_{be})$
Môđun pháp trung bình	m_{nm}	$m_{nm} = (m_{te} - \dfrac{b}{\sqrt{z_1^2 + z_2^2}})\cos\beta_m$

2.1.2. Những vấn đề cần lưu ý khi xây dựng phần mềm thiết kế truyền động bánh răng

a) Chỉ tiêu cơ bản để thiết kế truyền động bánh răng là độ bền tiếp xúc và độ bền uốn. Với các bộ truyền được che kín và đủ dầu bôi trơn, tróc rỗ mặt răng là dạng hỏng nguy hiểm nhất, vì vậy thường tiến hành

tính thiết kế bộ truyền theo độ bền tiếp xúc, sau đó tiến hành kiểm nghiệm về độ bền tiếp xúc, độ bền uốn và quá tải. Theo phương pháp này, thiết kế truyền động bánh răng được tiến hành lần lượt như sau:

1. Chọn vật liệu.
2. Xác định ứng suất cho phép.
3. Xác định sơ bộ một thông số cơ bản của bộ truyền theo độ bền tiếp xúc.
4. Xác định các thông số ăn khớp.
5. Kiểm nghiệm răng về độ bền tiếp xúc, độ bền uốn và quá tải.
6. Xác định lần cuối các thông số cơ bản và kích thước bộ truyền.

Bảng 2.3. Các công thức thiết kế bộ truyền bánh răng trụ

Tên thông số	*Công thức tính*	*№ công thức*
Khoảng cách trục	$a_w = K_a(u \pm 1)\sqrt[3]{\dfrac{T_1 K_{H\beta}}{[\sigma_H]^2 u \varphi_{ba}}}$	(2.1)
Ứng suất tiếp xúc cho phép	$[\sigma_{Hj}] = \sigma^0_{H\lim j} K_{HLj} / S_{Hj}$ với j =1; 2 .	(2.2)
	Khi $\beta = 0$ $[\sigma'_H] = [\sigma'_H]_{min}$	(2.3)
	Khi $\beta \neq 0$ $[\sigma'_H] = 0{,}5\,([\sigma'_{H1}] + [\sigma'_{H2}])$	(2.4)
	nhưng $[\sigma_H] \leq \lambda [\sigma_H]_{min}$, $\lambda = 1{,}25$	
	$\psi_{ba} = b_w / a_w$;	
	$\psi_{bd} = b_w / d_{w1} = 0{,}5\,\psi_{ba}(u \pm 1)$	
Hệ số tuổi thọ	$K_{HL} = \sqrt[m_H]{N_{H0} / N_{HE}}$ với $m_H = 6$	(2.5)
Số chu kỳ cơ sở	$N_{H0} = 30 H_{HB}^{2,4}$ với H_{HB} - độ rắn Brinen	(2.6)
Số chu kỳ chịu tải	$N_{HE} = 60c\Sigma(T_i / T_{max})^3 n_i t_i$	(2.7)
	ở đây :	
	c – số lần ăn khớp trong 1 vòng quay	
	$K_{H\beta}$ - tra bảng 6. 7 [2]	
	Với cặp bánh răng bằng thép :	
	$K_a = 49{,}5$ MPa$^{1/3}$ (răng thẳng)	
	$K_a = 43$ MPa$^{1/3}$ (răng nghiêng)	
Số răng bánh nhỏ	$z_1 = 2a_w \cos\beta / [m(u \pm 1)]$	(2.8)

b) Các bước thiết kế trên đây có thể dùng cho cả bộ truyền bánh răng trụ và bộ truyền bánh răng côn, chỉ khác nhau ở chỗ:

- Trong bước 3, thông số cơ bản cần xác định sơ bộ đối với bộ truyền bánh răng trụ là khoảng cách trục a_w hoặc đường kính vòng lăn bánh nhỏ d_{w1}, đối với bộ truyền bánh răng côn là chiều dài côn R_e hoặc đường kính trung bình bánh nhỏ d_{m1}.

- Trong bước 4, các quan hệ hình học và ăn khớp đối với bộ truyền bánh răng trụ được xác định theo bảng 2.1, đối với bộ truyền bánh răng côn - theo bảng 2.2.

Bảng 2.4. Các công thức thiết kế bộ truyền bánh răng côn

Tên thông số	*Công thức tính*	*№ công thức*
Chiều dài côn ngoài	$R_e = K_R \sqrt{u^2+1}\sqrt[3]{\dfrac{T_1 K_{H\beta}}{(1-K_{be})K_{be}[\sigma_H']^2}}$ trong đó $[\sigma_H']$ tính theo (2.3) hoặc (2.4) nhưng với $\lambda = 1,15$ $K_{H\beta}$ – tra bảng 6.21[2] $K_{be} = b/R_e = 0,25 \div 0,3$ (sử dụng trị số lớn khi $u \le 3$). Với cặp bánh răng bằng thép : Răng thẳng $K_R = 50\ MPa^{1/3}$; Răng tiếp tuyến $K_R = 43,5\ MPa^{1/3}$; Răng cung tròn $K_R = 41,7\ MPa^{1/3}$.	(2.9)
Đường kính chia ngoài	$d_{e1} = \dfrac{2R_e}{\sqrt{1+u^2}}$	(2.10)
Đường kính trung bình	$d_{m1} = (1-0,5K_{be})d_{e1}$	(2.11)
Môđun vòng trung bình	$m_{tm} = d_{m1}/z_1$	(2.12)
Môđun vòng ngoài	$m_{te} = m_{tm}/(1-0,5K_{be})$	(2.13)
Môđun pháp trung bình	$m_{nm} = (m_{te} - b/\sqrt{z_1^2+z_2^2})\cos\beta_m$	(2.14)

c) Trước khi lập trình, cần dựa vào các bước tính nêu trên để viết lưu đồ thuật giải thiết kế truyền động bánh răng, trong đó cần lưu ý đến các bước lặp và sử dụng các biến điều khiển. Thí dụ về lưu đồ thuật giải thiết kế truyền động bánh răng côn có thể xem trên hình 2.3.

Bảng 2.5. Các công thức kiểm nghiệm bộ truyền bánh răng trụ

Tên thông số	*Công thức tính*	*№ công thức*
Ứng suất tiếp xúc tính toán	$\sigma_H = Z_M Z_H Z_\varepsilon \sqrt{\dfrac{2T_1 K_H (u \pm 1)}{b_w d_{w1}^2 u}} \le [\sigma_H]$	(2.15)
Ứng suất tiếp xúc cho phép	$[\sigma_H] = \sigma^0_{H\lim} K_{HL} Z_R Z_V K_{xH} / S_H$	(2.16)
Hệ số kể đến dạng bề mặt tiếp xúc	$Z_H = \sqrt{2\cos\beta / \sin 2\alpha_w}$	(2.17)
Hệ số trùng khớp ngang	$\varepsilon_\alpha = \dfrac{\sqrt{d_{a1}^2 - d_{b1}^2} + \sqrt{d_{a2}^2 - d_{b2}^2} - 2a_w \sin\alpha_w}{2\pi m \cos\alpha_t / \cos\beta}$	(2.18)
Hệ số trùng khớp dọc	$\varepsilon_\beta = b_w \sin\beta / (m\pi)$	(2.19)
Hệ số kể đến sự trùng khớp	$Z_\varepsilon = \sqrt{(4-\varepsilon_\alpha)/3}$ khi $\varepsilon_\beta = 0$	(2.20)
	$Z_\varepsilon = \sqrt{\dfrac{(4-\varepsilon_\alpha)(1-\varepsilon_\beta)}{3} + \dfrac{\varepsilon_\beta}{\varepsilon_\alpha}}$ khi $\varepsilon_\beta < 1$	(2.21)
	$Z_\varepsilon = \sqrt{1/\varepsilon_\alpha}$ khi $\varepsilon_\beta \ge 1$	(2.22)
Ứng suất uốn trên chân răng :		
Bánh 1	$\sigma_{F1} = 2T_1 K_F Y_\varepsilon Y_\beta Y_{F1} / (b_w d_{w1} m) \le [\sigma_{F1}]$	(2.23)
Bánh 2	$\sigma_{F2} = \sigma_{F1} Y_{F2} / Y_{F1} \le [\sigma_{F2}]$	(2.24)
Hệ số kể đến răng nghiêng	$Y_\beta = 1 - \beta_n^o / 140$	(2.25)
Ứng suất uốn cho phép	$[\sigma_{Fj}] = (\sigma^0_{F\lim j} / S_{Fj}) K_{FL} K_{Fc} K_{xH} Y_R Y_S$	(2.26)
	ở đây $K_{FL} = \sqrt[m_F]{N_{F0} / N_{FE}}$ với $N_{F0} = 4.10^6$	(2.27)
	$N_{FE} = 60c\Sigma (T_i / T_{\max})^{m_F} n_i t_i$	(2.28)
Hệ số tải trọng	$K_H = K_{H\beta} K_{Hv} K_{H\alpha}$; $K_F = K_{F\beta} K_{Fv} K_{F\alpha}$ với $K_{H\beta}$ và $K_{F\beta}$ tra bảng 6.7 [2] : $K_{H\alpha}$ và $K_{F\alpha}$ tra bảng 6.14 [2] ; K_{Hv} theo công thức (6.41) và (6.42) [2]	(2.29)

Chú thích 1) Với bánh răng bằng thép $Z_M = 274\ MPa^{1/3}$; $m_F = 6$ khi HB ≤ 350 ; $m_F = 9$ khi HB >350 ;
2) Hệ số dạng răng Y_F tra bảng 6.18 [2] theo $z_v = z / \cos^3\beta$

Bảng 2.6. Các công thức kiểm nghiệm bộ truyền bánh răng côn

Tên thông số	*Công thức tính*	*№ công thức*
Ứng suất tiếp xúc tính toán	$\sigma_H = Z_M Z_H Z_\varepsilon \sqrt{\dfrac{2T_1 K_H \sqrt{u^2+1}}{0{,}85 b d_{m1}^2 u}} \le [\sigma_H]$ trong đó ứng suất tiếp xúc cho phép $[\sigma_H]$ xác định theo (2.17) ; K_H – theo (2.29); Z_H – theo (2.18); Z_ε -theo (2.20) đối với răng thẳng và theo (2.22) đối với bánh răng nghiêng và răng cung tròn, với ε - theo (2.31).	(2.30)
Hệ số trùng khớp ngang	$\varepsilon_\alpha = [1{,}88 - 3{,}2(1/z_1 + 1/z_2)]\cos\beta_m$	(2.31)
Ứng suất uốn trên chân răng:		
Bánh 1	$\sigma_{F1} = \dfrac{2T_1 K_F Y_\varepsilon Y_\beta Y_{F1}}{0{,}85 b m_{nm} d_{m1}} \le [\sigma_{F1}]$	(2.32)
Bánh 2	$\sigma_{F2} = \sigma_{F1} Y_{F2} / Y_{F1} \le [\sigma_{F2}]$ trong đó ứng suất uốn cho phép –theo (2.26); Y_β - theo (2.25); K_F –theo (2.29),với *$K_{H\beta}$ và $K_{F\beta}$ tra bảng 6.21 [2] ;* *$K_{H\alpha}$ và $K_{F\alpha}$ tra bảng 6.14 [2] ;* *K_{Hv} tính theo (6.63) và (6.64) [2].*	(2.33)

Chú thích 1) Với bánh răng bằng thép $Z_M = 274\ MPa^{1/3}$; $m_F = 6$ khi HB ≤ 350; $m_F = 9$ khi HB >350 ;

2) Hệ số dạng răng Y_F tra bảng 6.18 [2] theo $z_{vn} = z / \cos\delta$ đối với bánh côn răng thẳng và $z_{vn} = z / (\cos^3\beta_m \cos\delta)$ đối với bánh côn răng nghiêng và răng cung tròn.

3) Khi sử dụng công thức (2.32) cho bánh côn răng thẳng, thay m_{nm} bằng m_{tm}.

d) Các bước tính toán chỉ có thể bắt đầu sau khi cho dữ liệu thiết kế. Dữ liệu ở đây thường là công suất P (kW), số vòng quay n (vg/ph), tỉ số truyền u và thời hạn sử dụng L_h (giờ), có thể được nhập từ bàn phím nhưng tốt nhất là nên gọi từ *File dữ liệu* đã được nạp sẵn trên đĩa, như thế sau mỗi lần sửa chữa sai sót hoặc gỡ rối, không phải nạp lại dữ liệu ban đầu.

e) Chọn vật liệu phụ thuộc vào công suất cần truyền, yêu cầu về khối lượng, môi trường làm việc và khả năng công nghệ. Với công suất trung bình và không yêu cầu chặt chẽ về kích thước có thể chọn vật liệu nhóm

I (HB ≤ 350), với vật liệu hai bánh cùng mác thép hoặc vật liệu bánh nhỏ có cơ tính tốt hơn nhưng để tăng khả năng chạy mòn và khả năng chống dính của răng, nên nhiệt luyện bánh nhỏ đạt độ rắn cao hơn từ 25 đến 40 HB. Với công suất lớn hơn, yêu cầu khối lượng nhỏ, nên chọn vật liệu nhóm II có độ rắn HRC > 45 đồng thời cần quan tâm đến khả năng cung cấp thiết bị để gia công tinh bánh răng sau nhiệt luyện.

Chú ý rằng mỗi lần chọn vật liệu và nhiệt luyện cho mỗi bánh răng phải gán nhãn vật liệu và phương pháp nhiệt luyện (là những biến ký tự khá dài). Để giảm thời gian đối thoại với máy và tăng được tốc độ tính toán nên số hoá nhãn vật liệu và phương pháp nhiệt luyện. Muốn vậy hãy sử dụng các biến *vl, nl* và *vli, nli* với định nghĩa

```
vl,nl : array [0..5,0..2] of string[7] ;
vli,nli : array [0..10] of integer ;
```

khi đó chỉ cần gán các con số từ 1 đến 10 cho vli[i] ứng với 10 loại vật liệu chế tạo bánh răng (xem bảng 6.1[2]) và các con số từ 1 đến 5 cho nli[i] ứng với 5 loại nhiệt luyện là: thường hoá, tôi cải thiện, tôi, thấm cácbon và thấm nitơ, cuối cùng chuyển vli[i] thành vl[k,i] và nli[i] thành nl[k,i],... Khi chuyển vật liệu được số hoá thành vật liệu thực có thể sử dụng đoạn chương trình sau đây:

```
case vli[i] of
 1: vl[k,i] := '40';
 2: vl[k,i] := '45';
 ...
 10: vl[k,i] := '25CrMnTi';
end;
```

g) Độ bền bánh răng phụ thuộc số chu kỳ chịu tải. Theo [7], đối với thép kết cấu được tôi bề mặt hoặc thấm cacbon, ứng suất tiếp xúc cho phép $[\sigma_H]$ phụ thuộc vào 3 khoảng chu kỳ chịu tải: khi $N_{HE} \le 10^4$ chu kỳ, $[\sigma_H]$ được xác định theo độ bền tĩnh, khi $10^4 < N_{HE} \le 5.10^7$ chu kỳ, $[\sigma_H]$ được xác định theo giới hạn mỏi ngắn hạn và khi $N_{HE} > 5.10^7$chu kỳ, $[\sigma_H]$ được xác định theo giới hạn mỏi dài hạn. Chính vì vậy, khi tính $[\sigma_H]$ phải kể đến ảnh hưởng của số chu kỳ chịu tải thông qua hệ số tuổi thọ K_{HL} [xem công thức (2.5)]. Rõ ràng là K_{HL} phụ thuộc vào giới hạn bền tĩnh, giới hạn mỏi ngắn hạn và giới hạn mỏi dài hạn.

Tương tự, ứng với các mác thép của Nga, hệ số tuổi thọ được tính như sau [4]:

$K_{HL} = 1$ khi $N_{HE} \ge N_{H0}$ ứng với giới hạn mỏi dài hạn,

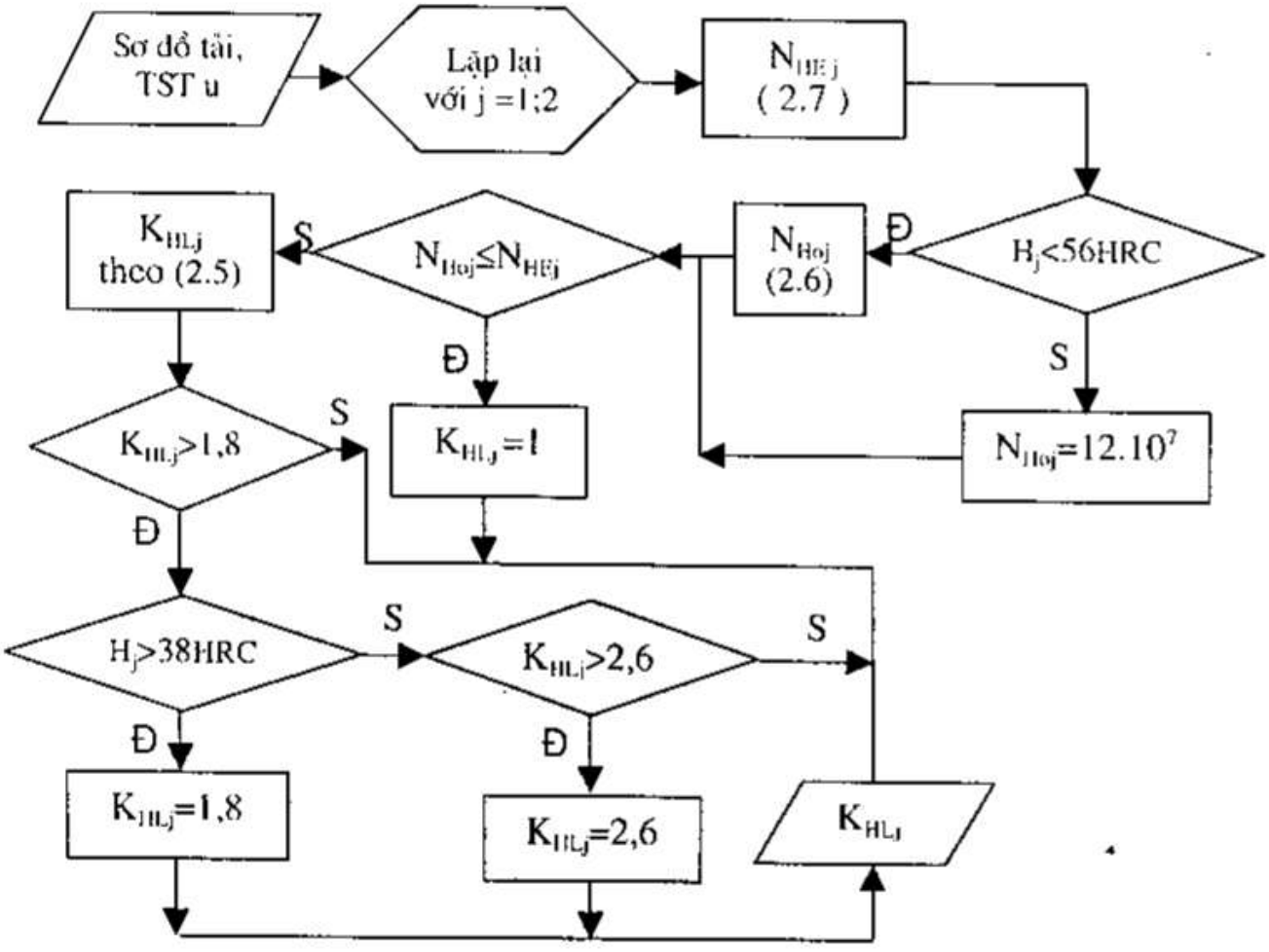
ở đây N_{HE} tính theo (2.7), N_{H0} tính theo (2.6) nhưng khi độ rắn mặt răng

H >563 HB thì lấy độ rắn H = 56 HRC, tương ứng $N_{H0} = 12.10^7$;

$K_{HL} = \sqrt[6]{N_{H0} / N_{HE}}$ khi $N_{HE} < N_{H0}$ ứng với giới hạn mỏi ngắn hạn,

nhưng với độ rắn mặt răng H >38 HRC nếu tính ra $K_{HL} > 1,8$ thì lấy $K_{HL} = 1,8$ trong khi đó với độ rắn mặt răng H ≤ 38 HRC nếu tính ra $K_{HL} > 2,6$ thì lấy $K_{HL} = 2,6$. Sử dụng các giá trị giới hạn 1,8 hoặc 2,6 nhằm đảm bảo cho $[\sigma_H]$ được xác định theo giới hạn mỏi ngắn hạn mà không phải theo độ bền tĩnh.

Sử dụng phương pháp xây dựng lưu đồ thuật giải trong [4], có thể xây dựng lưu đồ thuật giải xác định hệ số tuổi thọ K_{HL} như trên hình 2.1.



Hình 2.1. Lưu đồ thuật giải xác định hệ số tuổi thọ K_{HL}

h) Khi xác định ứng suất tiếp xúc cho phép cần lưu ý rằng lúc đầu vì chưa biết giá trị của vận tốc vòng, do đó chưa thể xác định được Z_v, chưa chọn được cấp chính xác động học cho bộ truyền, do đó chưa thể xác định được Z_R, dẫn đến chỉ có thể xác định sơ bộ ứng suất tiếp xúc cho phép $[\sigma'_H]$ và dùng nó để xác định sơ bộ một thông số bộ truyền (ở đây

là khoảng cách trục a_w). Đó chính là một trong những lý do giải thích vì sao đã tiến hành thiết kế bộ truyền bánh răng theo độ bền tiếp xúc, sau khi xác định được các thông số và kích thước bộ truyền vẫn cần đến bước tính kiểm nghiệm răng bánh răng về độ bền tiếp xúc, dựa vào ứng suất tiếp xúc cho phép $[\sigma_H]$ được xác định theo công thức (2.10) hoặc theo công thức sau:

$$[\sigma_H] = [\sigma_H] Z_R Z_V K_{xH} \qquad (2.34)$$

trong đó

Z_R – hệ số xét đến độ nhám mặt răng làm việc:

$Z_R = 1$ khi $R_a \le 1{,}25 \ldots 0{,}63$ μm;

$Z_R = 0{,}95$ khi $R_a = 2{,}5 \ldots 1{,}25$ μm và

$Z_R = 0{,}9$ khi $R_Z = 10 \ldots 40$ μm.

Z_v – hệ số xét đến ảnh hưởng của vận tốc vòng v,

+ Khi $v \le 5$ m/s : $Z_v = 1$;

+ Khi $v > 5$ m/s , lấy Z_v theo các giá trị dưới đây [4]:

$$Z_v = 0{,}85v^{0.1} \quad \text{khi } H \le 350 \text{ HB} \qquad (2.35)$$

$$Z_v = 0{,}925v^{0.05} \quad \text{khi } H > 350 \text{ HB} \qquad (2.36)$$

K_{xH} – hệ số xét đến ảnh hưởng của kích thước bánh răng, $K_{xH} = 1$ khi đường kính vòng đỉnh bánh răng $d_a \le 700$ mm.

Chú ý rằng, sau khi tính được $[\sigma_{H1}]$ và $[\sigma_{H2}]$ cần phân biệt hai trường hợp:

khi $\beta = 0$ hoặc $\beta_m = 0$: $[\sigma_H] = [\sigma_H]_{min}$ (2.37)

khi $\beta \neq 0$ hoặc $\beta_m \neq 0$:

- Nếu $H_1 > 350$ HB và $H_2 > 350$ HB: $[\sigma_H]$ tính theo công thức 2.37);
- Nếu $H_1 \le 350$ HB và $H_2 \le 350$ HB:

$$[\sigma_H] = ([\sigma_{H1}] + [\sigma_{H2}])/2 \qquad (2.38)$$

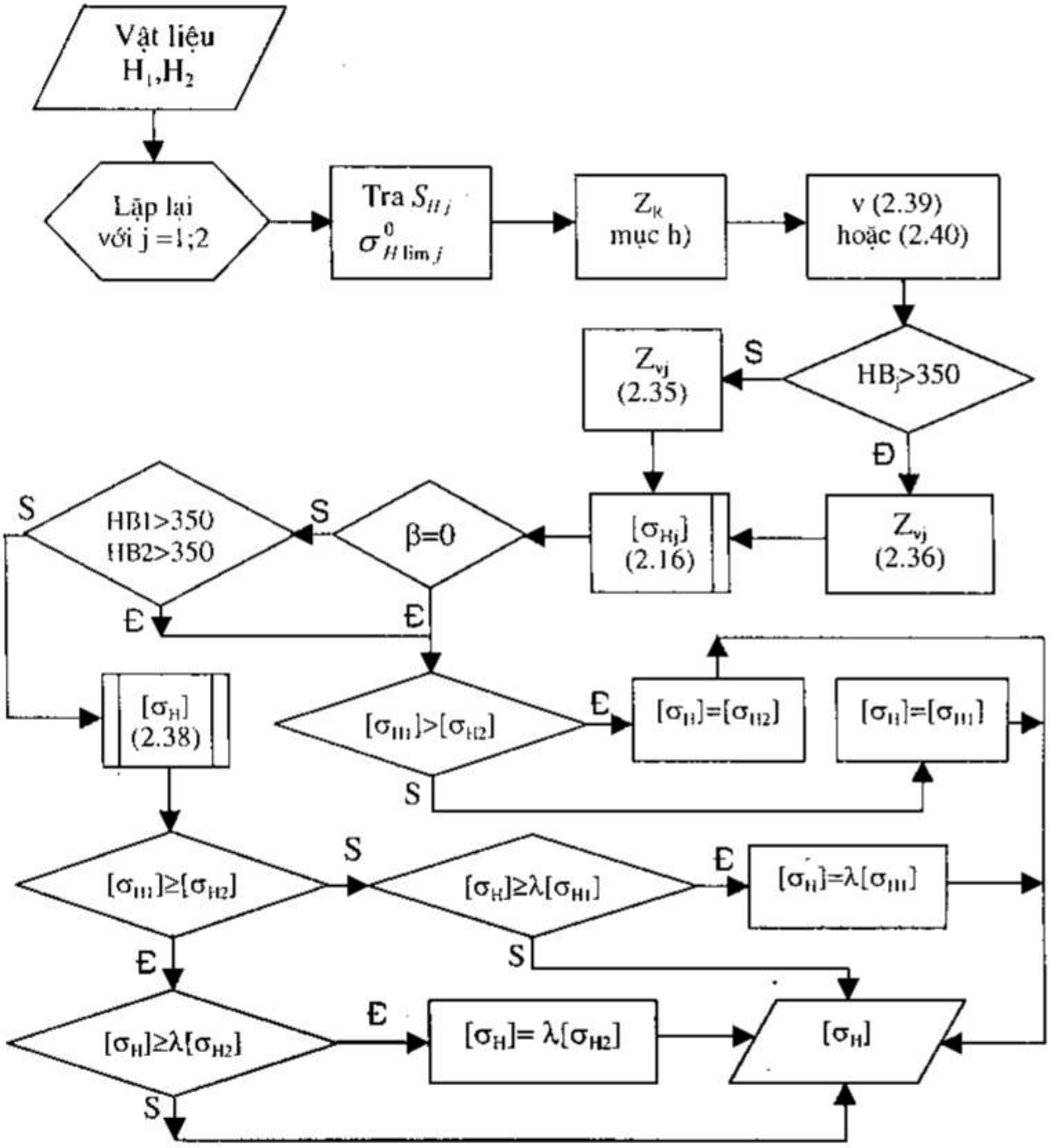
nhưng $[\sigma_H]$ tính theo (2.38) không được lớn hơn $\lambda[\sigma_H]_{min}$ với $\lambda = 1{,}25$ đối với bộ truyền bánh răng trụ và $\lambda = 1{,}15$ đối với bộ truyền bánh răng côn.

Lưu đồ thuật giải xác định ứng suất tiếp xúc cho phép $[\sigma_H]$ cho trên hình 2.2. Lưu đồ này dùng chung cho cả bánh răng trụ và bánh răng côn, chỉ khác nhau ở hệ số λ và công thức xác định vận tốc vòng:

Đối với bánh răng trụ $v = \pi d_{w1} n_1 /(60000)$ m/s (2.39)

Đối với bánh răng côn $v = \pi d_{m1} n_1 /(60000)$ m/s (2.40)

trong đó d_{W1} - đường kính vòng lăn bánh trụ nhỏ và d_{m1} - đường kính trung bình bánh côn nhỏ.



Hình 2.2. Lưu đồ thuật giải xác định ứng suất tiếp xúc cho phép $[\sigma_H]$

i) Khi thiết kế bộ truyền bánh răng trụ theo độ bền tiếp xúc, thông số được chọn để xác định sơ bộ thường là a_w tính theo công thức (2.1). Sau

khi tính được a_w nên chọn a_w nguyên và dùng nó làm căn cứ để xác định môđun theo công thức

$$m = (0,01 \div 0,02)a_w \qquad (2.41)$$

rồi chọn m tiêu chuẩn theo dãy ưu tiên sau đây: 1,5; 2; 2,5; 3; 4; 5; 6; 8; 10; 12 (trích từ bảng 6.8 [2]), trên cơ sở đó tiếp tục xác định các thông số khác của bộ truyền. Với bánh trụ răng thẳng, $\beta = 0$; chọn $\beta = 8 \div 20°$ đối với răng nghiêng và $\beta = 30 \div 40°$ đối với răng chữ V hoặc bánh răng phân đôi rồi tính số răng z_1 của bánh 1 theo công thức (2.8), lấy z_1 nguyên, tính $z_2 = uz_1$, lấy z_2 nguyên, từ đó tính lại a_w đối với răng thẳng hoặc giữ nguyên a_w và tính lại β đối với răng nghiêng hoặc răng chữ V với yêu cầu β phải nằm trong khoảng $8 \div 20°$ hoặc $30 \div 40°$. Trường hợp β nằm ngoài hai khoảng trên thì tăng giảm z_1 đi 1 răng : $z_1 = z_1 \pm 1$ răng rồi tính lại β. Các thông số còn lại được xác định theo các công thức trong bảng (2.1).

Riêng với bánh trụ răng thẳng, cần lưu ý rằng khoảng cách trục a_w tính theo m tiêu chuẩn, z_1 và z_2 nguyên nên kết quả thường là một số lẻ, vì vậy muốn có a_w nguyên cần tiến hành dịch chỉnh, khi đó có thể dựa theo những hướng dẫn trong tài liệu [2] để xác định các hệ số dịch chỉnh x_1 và x_2 thông qua các hệ số k_x và k_y được xác định bằng phương pháp đồ thị. Tuy nhiên cũng có thể xác định x_1 và x_2 nhằm đảm bảo khoảng cách trục định trước bằng cách tính toán theo trình tự sau đây [6]:

1. Tính góc ăn khớp theo công thức

$$\cos\alpha_{tw} = \frac{(z_1 + z_2)m\cos\alpha_t}{2a_t\cos\beta} \qquad (2.42)$$

2. Tính tổng hệ số dịch chỉnh

$$x_t = \frac{(z_1 + z_2)(inv\alpha_{tw} - inv\alpha_t)}{2tg\alpha_t}, \qquad (2.43)$$

trong đó $inv\alpha_t = tg\alpha_t - \alpha_t$; $inv\alpha_{tw} = tg\alpha_{tw} - \alpha_{tw}$.

3. Tính hệ số dịch tâm

$$y = (a_w - a_t)/m \qquad (2.44)$$

4. Tính hệ số giảm đỉnh răng

$$\Delta y = x_t - y \qquad (2.45)$$

5. Xác định hệ số dịch chỉnh bánh 1 và bánh 2

$$x_1 = 0{,}5[x_t - y(z_2 - z_1)/(z_1 + z_2)] \quad (2.46)$$

$$x_2 = x_t - x_1 \quad (2.47)$$

6. Xác định đường kính đỉnh răng và đường kính đáy răng cho các bánh răng i, với i = 1; 2:

$$d_{ai} = d_i + 2(1 + x_i - \Delta y)m \quad (2.48)$$

$$d_{fi} = d_i - (2{,}5 - 2x_i)m \quad (2.49)$$

k) Với bộ truyền bánh răng côn, do K_{be} dễ chọn hơn ($K_{be} = 0{,}25 \div 0{,}3$) nên cũng thường xác định sơ bộ chiều dài côn R_e theo độ bền tiếp xúc. Từ R_e xác định d_{e1} theo (2.10), dựa vào d_{e1} và loại răng xác định z_{1p} theo bảng (6.22) [2] rồi tính z_1 theo các công thức sau:

Khi độ rắn H_1, $H_2 \le 350$ HB $\quad z_1 = 1{,}6\, z_{1p}$ (2.50)

Khi độ rắn $H_1 \ge 45$ HRC , $H_2 \le 350$ HB $\quad z_1 = 1{,}3\, z_{1p}$ (2.51)

Khi độ rắn $H_1, H_2 \ge 45$ HRC $\quad z_1 = z_{1p}$ (2.52)

Sau đó tính d_{m1} theo (2.11), m_{tm} theo (2.12), m_{te} theo (2.13) đối với răng thẳng và m_{nm} theo (2.14) đối với răng nghiêng và răng cung tròn rồi lấy các giá trị môđun này theo tiêu chuẩn, từ đó xác định lại R_e theo công thức trong bảng 2.2.

l) Tuy nhiên khi thiết kế, thông số được chọn để xác định sơ bộ đối với bộ truyền bánh răng trụ cũng có thể là đường kính vòng lăn bánh nhỏ d_{w1} theo công thức (2.53) dưới đây và dùng nó làm căn cứ để tiếp tục xác định các thông số còn lại:

$$d_{w1} = K_d \sqrt[3]{\frac{T_1 K_{H\beta}(u \pm 1)}{[\sigma'_H]^2 u \varphi_{bd}}} \quad (2.53)$$

trong đó với cặp bánh răng bằng thép: $K_d = 77$ MPa$^{1/3}$ đối với răng thẳng và $K_d = 67{,}5$ MPa$^{1/3}$ đối với răng nghiêng và răng chữ V; dấu + ứng với ăn khớp ngoài, dấu – ứng với ăn khớp trong; $[\sigma'_H]$ tính theo (2.3) đối với răng thẳng và theo (2.4) đối với răng nghiêng; $K_{H\beta}$ - hệ số phân bố không đều tải trọng trên chiều rộng vành răng, tra bảng (6.7 [2]).

Tương tự, đối với bộ truyền bánh răng côn cũng có thể xác định sơ bộ đường kính chia ngoài theo công thức (2.54) dưới đây rồi dùng nó làm căn cứ để tiếp tục xác định các thông số còn lại:

$$d_{e1} = K_d \sqrt[3]{\frac{T_1 K_{H\beta}}{(1 - K_{be}) K_{be} u [\sigma'_H]^2}} \qquad (2.54)$$

trong đó với cặp bánh răng bằng thép: $K_d = 100$ MPa$^{1/3}$ đối với răng thẳng, $K_d = 87$ MPa$^{1/3}$ đối với răng nghiêng và $K_d = 83{,}5$ MPa$^{1/3}$ đối với răng tiếp tuyến; $[\sigma'_H]$ tính theo (2. 3) đối với răng thẳng và theo (2. 4) đối với răng nghiêng và răng cung tròn.

m) Sau khi xác định chính xác ứng suất tiếp xúc cho phép $[\sigma_H]$ theo (2.34) cần xác định ứng suất tiếp xúc tính toán σ_H theo công thức (2.15) đối với bộ truyền bánh răng trụ hoặc (2.30) đối với bộ truyền bánh răng côn và kiểm tra điều kiện sau đối với độ bền tiếp xúc:

$$k.[\sigma_H] \le \sigma_H \le [\sigma_H] \qquad (2.55)$$

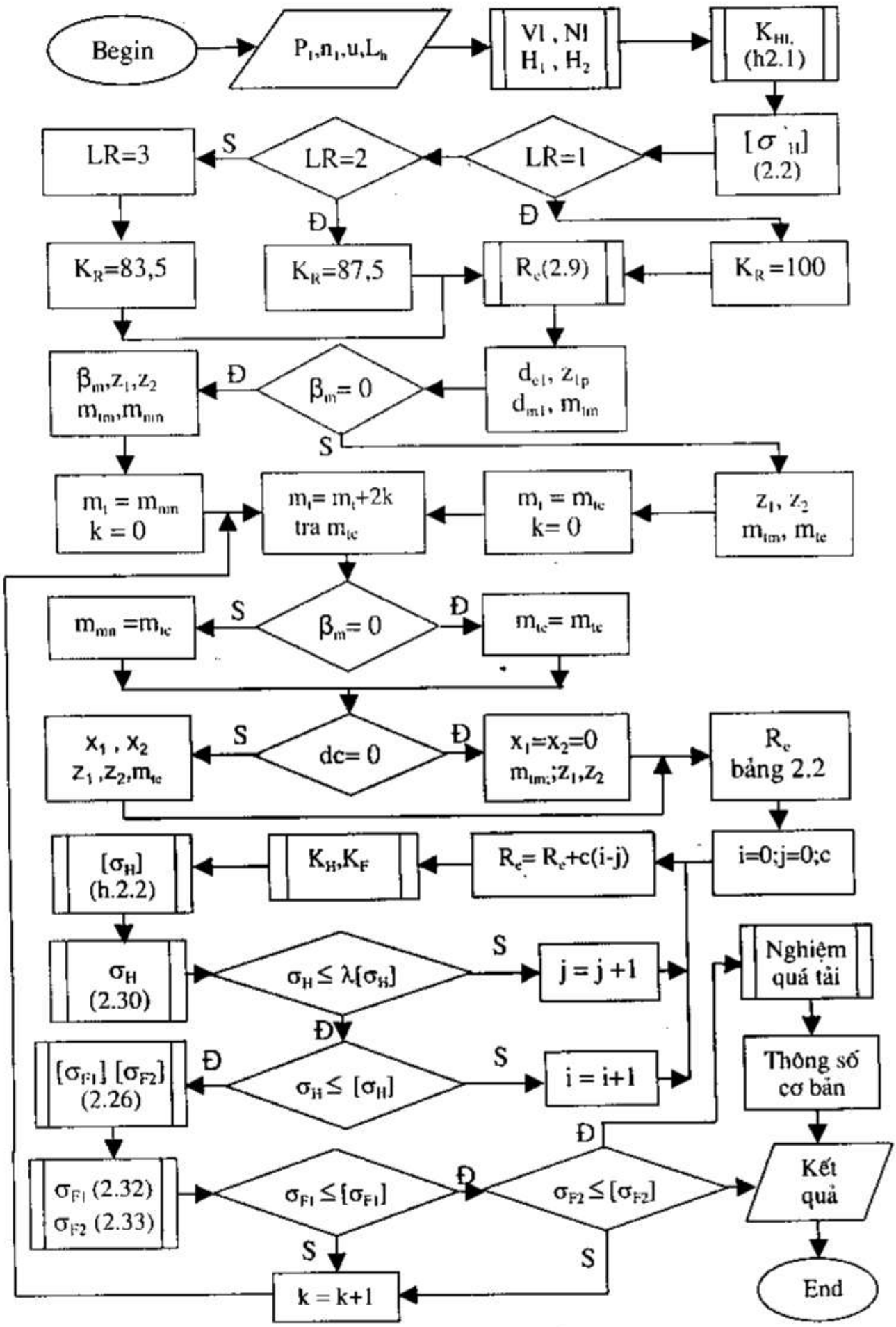
trong đó k là một hệ số nhỏ hơn 1 nhằm tránh cho bộ truyền không quá thừa bền. Trường hợp điều kiện (2.55) không thoả mãn, sử dụng biến i khi $\sigma_H > [\sigma_H]$ và biến j khi $\sigma_H < k.[\sigma_H]$ để thực hiện các vòng lặp nhằm tăng hoặc giảm a_w cho tới khi đảm bảo điều kiện (2.55) – xem lưu đồ thuật giải thiết kế truyền động bánh răng côn trên hình 2.3.

n) Khác với ứng suất tiếp xúc cho phép phải xác định hai lần nhưng chỉ sử dụng một giá trị để tính toán ($[\sigma'_H]$ để tính thiết kế và $[\sigma_H]$ để tính kiểm nghiệm), ứng suất uốn cho phép chỉ cần xác định một lần nhưng với hai giá trị ($[\sigma_{F1}]$ đối với bánh nhỏ và $[\sigma_{F2}]$ đối với bánh lớn). Khi kiểm nghiệm độ bền uốn cho từng bánh răng không đạt yêu cầu (thường xảy ra đối với trường hợp dùng vật liệu nhóm II có độ rắn lớn để chế tạo bánh răng và chọn môđun nhỏ), sử dụng biến k để chọn môđun tiêu chuẩn lớn hơn và các bước tính được lặp lại như thể hiện trên lưu đồ thuật giải thiết kế truyền động bánh răng côn trên hình 2.3.

p) Sau khi hoàn thành bước tính toán thứ sáu, tức là xác định lần cuối các thông số cơ bản và kích thước bộ truyền theo các công thức trong bảng 2.1 hoặc bảng 2.2, cần ghi các số liệu phục vụ các bước thiết kế tiếp theo vào các *File* tương ứng trên đĩa cũng như tổng hợp các số liệu cần thiết để ghi ra màn hình và in các kết quả tính toán thiết kế ứng với dữ liệu đã cho.

2.2. XÂY DỰNG PHẦN MỀM TÍNH TOÁN THIẾT KẾ TRUYỀN ĐỘNG BÁNH RĂNG HÀNH TINH

Khác với truyền động bánh răng thường, có trục quay cố định, truyền động bánh răng hành tinh có ít nhất một trục quay di động. Truyền động



Hình 2.3. Lưu đồ thuật giải thiết kế truyền động bánh răng côn

bánh răng hành tinh được phân thành ba loại: truyền động hành tinh đơn giản (khi cố định một trong các bánh trung tâm), truyền động vi sai (các khâu cơ bản là bánh trung tâm và cần đều quay) và truyền động vi sai kín (hai khâu cơ bản được nối với một bộ truyền nào đó).

2.2.1. Các quan hệ động học và công thức tính toán độ bền

2.2.1.1. Tỉ số truyền và số vòng quay các khâu

Trên bảng 2.7 giới thiệu các sơ đồ động của truyền động hành tinh cơ bản (truyền động A [2, 4]) với các ký hiệu: 0 – cần ; 1 – bánh trung tâm ăn khớp ngoài; 2 – bánh vệ tinh ; 3 – bánh trung tâm ăn khớp trong.

Trên bảng 2.7 cũng cho các công thức tính tỉ số truyền và số vòng quay của các khâu ứng với trường hợp khâu A hoặc B là khâu chủ động với các thông số và ký hiệu sau đây:

Tỉ số răng

$$p = z_3/z_1 \qquad (2.56)$$

Tỉ số truyền từ bánh trung tâm 1 đến bánh trung tâm 3 khi dừng cần

$$i_{13}^{0} = \frac{\omega_1 - \omega_0}{\omega_3 - \omega_0} = -\frac{z_3}{z_1} = -p \qquad (2.57)$$

Tỉ số truyền từ bánh trung tâm 1 đến cần khi bánh trung tâm 3 cố định

$$i_{10}^{3} = 1 - 1_{13}^{0} = 1 + p \qquad (2.58)$$

Tỉ số truyền của bộ truyền bánh răng 1- 2 và 2-3 khi dừng cần

$$u_{12} = - z_2/z_1 = (p - 1)/ 2 \qquad (2.59)$$

$$u_{23} = p / u_{12} \qquad (2.60)$$

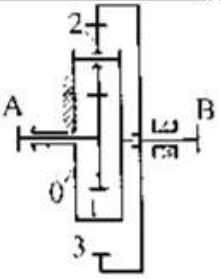
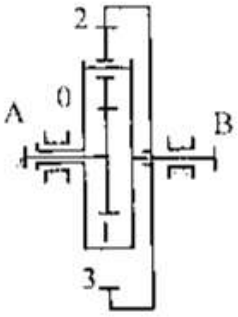
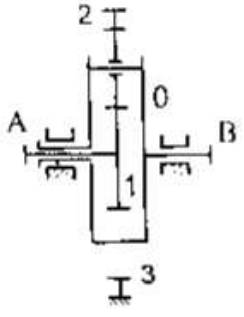
Tỉ số truyền và số vòng quay các khâu trong các truyền động khác xem ở bảng 6.24 và 6.25 [2]. Mômen xoắn tác dụng lên các khâu trong truyền động hành tinh, xem trong bảng 6.26 [2], còn công thức tính hiệu suất đối với các loại truyền động hành tinh cho trong bảng 6.27 [2].

2.2.1. 2. Chọn số răng

Khi chọn số răng cho các bánh răng trong truyền động hành tinh phải đảm bảo điều kiện đồng trục (bảng 6.23 [2]), điều kiện lắp ($z_1+z_3 = kq$ với k là nguyên và q là số bánh vệ tinh) và điều kiện kề [2]. Theo trị số đã biết của tỉ số truyền i_{10}^{3} từ bánh trung tâm 1 đến cần 0 và số răng bánh trung tâm z_1, từ (2.58) có thể xác định được số răng bánh trung tâm 3 theo công thức sau [4], trong đó R là phần nguyên (*round*) của biểu thức:

$$z_3 = R[(i_{10}^3 - 1)z_1] \qquad (2.61)$$

Bảng 2.7 . Tỉ số truyền và số vòng quay các khâu của truyền động A

Sơ đồ động	*Tỷ số truyền của cơ cấu khi*		*Vận tốc góc của các khâu*
	A là chủ động (i_{AB})	*B là chủ động* (i_{BA})	
	$i_{13}^0 = -z_3/z_1 = -p$	$i_{31}^0 = \frac{1}{i_{13}^0} = -\frac{1}{p}$	$\omega_0 = 0\ ;\ \omega_1 = -p.\omega_3$ $\omega_2 - \omega_0 = \frac{2p}{p-1}.\omega_3$
	$i_{03}^1 = \frac{1}{1-1/i_{13}^0} = \frac{p}{1+p}$	$i_{30}^1 = 1 - \frac{1}{i_{13}^0} = 1 + \frac{1}{p}$	$\omega_0 = 0; \omega_3 = (1+\frac{1}{p})\omega_0$ $\omega_2 - \omega_0 = \frac{2}{P-1}\omega_0$
	$i_{10}^3 = 1 - i_{13}^0 = 1 + z_3/z_1 = 1 + p$	$i_{01}^3 = 1/(1-i_{13}^0) = \frac{1}{1+p}$	$\omega_3 = 0\ ;\ \omega_1 = (1+p).\omega_0$ $\omega_2 - \omega_0 = \frac{2p}{1-p}.\omega_0$

Khi đó phần lẻ C_3 của biểu thức $z_1 + z_3 = k\ q$ ứng với điều kiện lắp, có dạng

$$C_3 = \frac{z_3 + z_1}{q} - R(\frac{z_3 + z_1}{q}) \qquad (2.62)$$

Mặt khác, từ điều kiện đồng trục đối với truyền động A không dịch chỉnh

$$z_3 = z_1 + 2z_2$$

có thể xác định được phần lẻ C_2 của biểu thức này theo công thức:

$$C_2 = \frac{z_3 - z_1}{2} - R(\frac{z_3 - z_1}{2}) \qquad (2.63)$$

Như vậy điều kiện lắp sẽ được thực hiện khi $C_3 = 0$ và điều kiện đồng trục đối với truyền động không dịch chỉnh sẽ được thực hiện khi $C_2 = 0$ (*). Tuy nhiên, trong thực tế tính toán nhiều khi rất khó đảm bảo điều kiện đồng trục bằng cách thay đổi z_1 và z_3 để $C_2 = 0$, khi đó có thể tiến hành dịch chỉnh bánh 3 với hệ số dịch chỉnh x_3 nhằm đảm bảo cho khoảng cách trục của bộ truyền ăn khớp trong và khoảng cách trục của bộ truyền ăn khớp ngoài bằng nhau ($a_n = a_t$), tức là cũng đảm bảo được điều kiện đồng trục. Hệ số dịch chỉnh x_3 suy từ hiệu hệ số dịch chỉnh x_h được tính theo công thức

$$x_h = [(z_3 - z_2)(\text{inv}\alpha_{tw3} - \text{inv}\alpha_{tw1})]/(2\text{tg}\alpha) \qquad (2.64)$$

trong đó

$$\alpha_{tw3} = \arccos[(z_3 - z_2)\cos\alpha_t/(z_1 + z_2)] \qquad (2.65)$$

$$\text{inv}\alpha_{tw3} = \text{tg}\alpha_{tw3} - \alpha_{tw3}\ ;\ \text{inv}\alpha_{tw1} = \text{tg}\alpha_{tw1} - \alpha_{tw1}.$$

Với bộ truyền bánh răng trụ răng thẳng ăn khớp ngoài không dịch chỉnh thì $\alpha_{tw1} = \alpha_{t1} = \alpha = 20^o$, khi đó do $x_2 = 0$ suy ra $x_3 = x_h$.

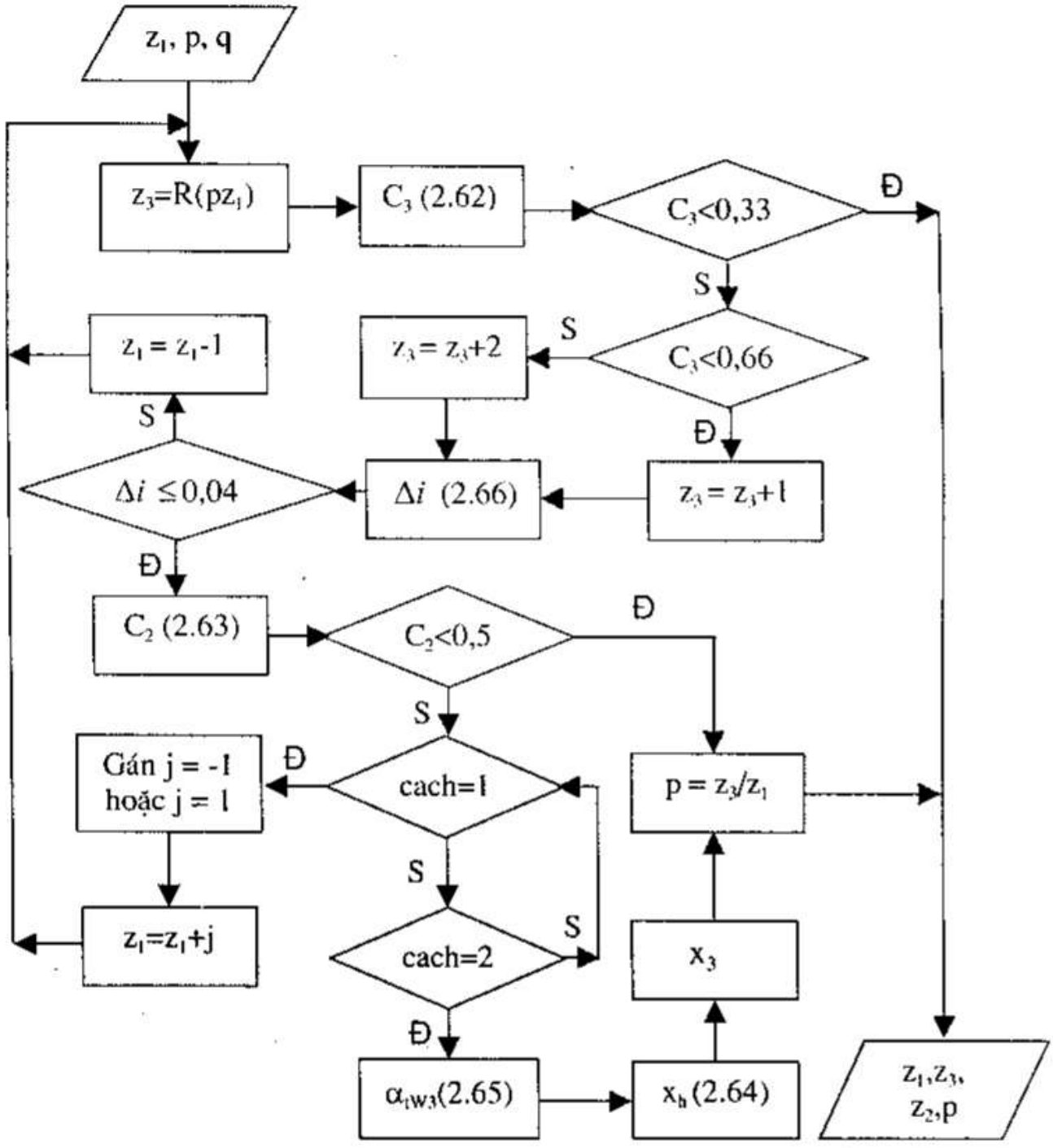
Lưu đồ thuật giải chọn số răng z_1, z_2 và z_3 đảm bảo được điều kiện lắp và điều kiện đồng trục ứng với số bánh vệ tinh q=3 theo tỉ số răng p và số răng z_1 chọn trước được trình bày trên hình 2. 4, trong đó các lệnh điều kiện $C_3 < 0{,}33$ và $C_3 < 0{,}66$ ứng với mỗi lần thay đổi số răng z_3 hoặc z_1 đi một răng. Vì sự thay đổi này có thể dẫn đến làm sai lệch tỉ số truyền so với tỉ số truyền thiết kế, do đó trước khi chuyển sang kiểm tra điều kiện đồng trục cần xác định sai lệch của tỉ số truyền Δi theo công thức:

$$\Delta i = \frac{i_{10}^3 - (1 + z_3/z_1)}{i_{10}^3} \qquad (2.66)$$

và hạn chế nó không quá 4%.

Tuy nhiên từ các công thức (2.61) và (2. 62) có thể thấy rằng khi thay đổi q, chỉ có phần lẻ C_2 không thay đổi, còn phần lẻ C_3 sẽ thay đổi ứng với bước thay đổi là 1/ q. Như vậy có thể sử dụng cùng một lưu đồ thuật giải để chọn số răng z_1, z_2 và z_3 đảm bảo điều kiện lắp và điều kiện đồng trục ứng với số bánh vệ tinh q khác nhau. Trên hình 2.5 trình bày lưu đồ thuật giải chọn số răng z_1 và z_3 ứng với q thay đổi từ 3 đến 6 đảm bảo điều kiện lắp (điều kiện đồng trục tương tự như trên hình 2.4, trong lưu đồ này không trình bày).

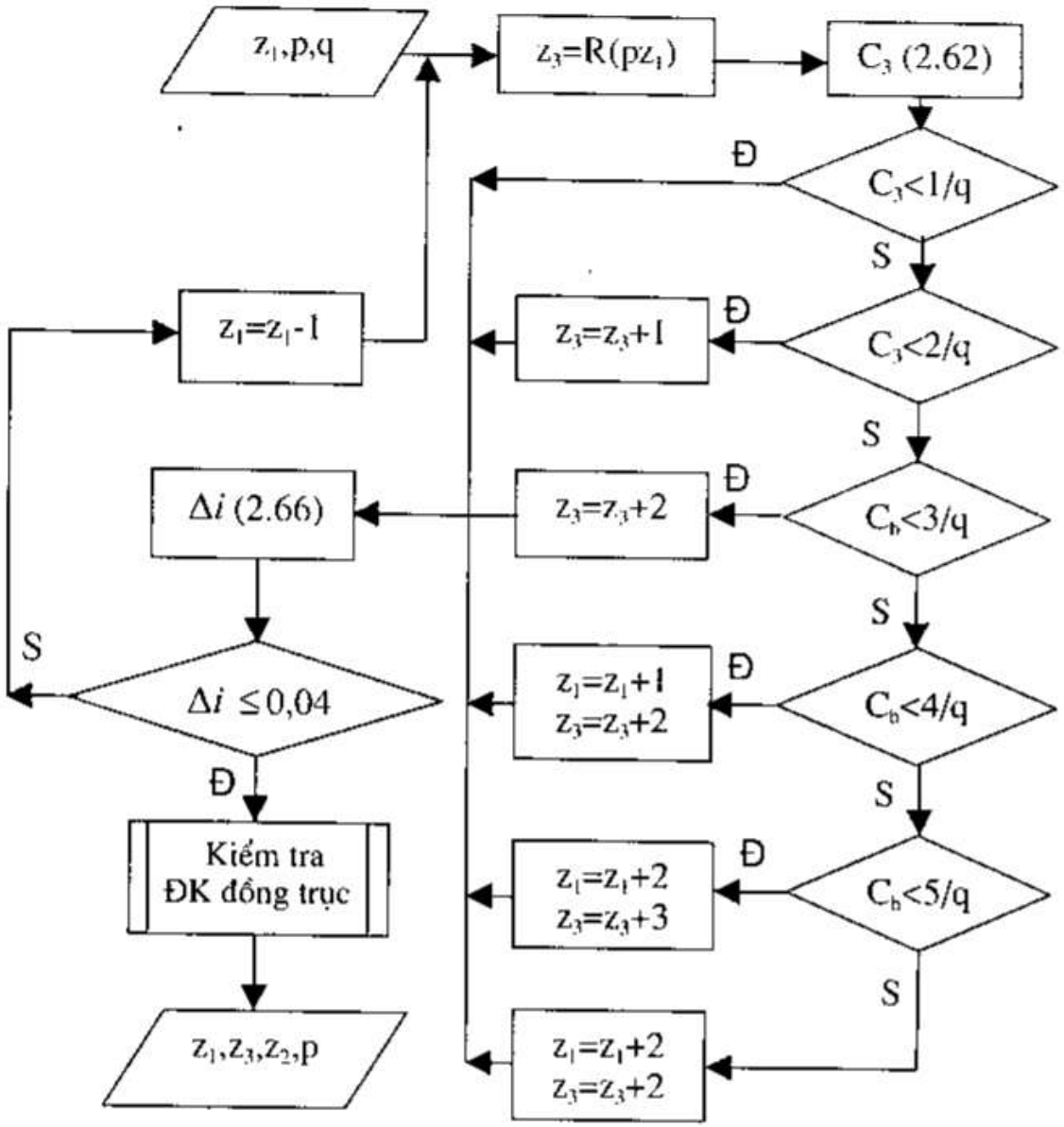
(*) Trường hợp số bánh vệ tinh q≤6, điều kiện kể thường được đảm bảo



Hình 2.4 : Lưu đồ thuật giải chọn số răng z_1,z_2 và z_3 đối với truyền động A

2.2.1.3. Các công thức tính toán độ bền

Tính toán độ bền bánh răng trong truyền động hành tinh tương tự như khi tính bánh răng trong truyền động thường. Các công thức thiết kế và kiểm nghiệm cho trong bảng 2.8.



Hình 2.5 : Lưu đồ thuật giải chọn số răng z_3 và z_1 ứng với $q \leq 6$

2.2.2. Những vấn đề cần lưu ý khi xây dựng phần mềm thiết kế truyền động hành tinh

Khi xây dựng phần mềm thiết kế truyền động bánh răng hành tinh cần lưu ý những nội dung sau đây:

1. Cũng như khi thiết kế truyền động bánh răng thường, trước khi chọn vật liệu và tiến hành tính toán truyền động bánh răng hành tinh, cần tạo *File dữ liệu* thiết kế và nạp sẵn trên đĩa, bao gồm: công suất P (kW), số vòng quay n (vg/ph), tỉ số truyền i và thời hạn sử dụng L_h (giờ).

2. Vì truyền động bánh răng hành tinh gồm hai cặp răng ăn khớp: cặp bánh răng ăn khớp ngoài 1-2 và cặp bánh răng ăn khớp trong 2-3 nên tính toán được tiến hành lần lượt cho mỗi cặp, bao gồm bước tính thiết kế và tính kiểm nghiệm cho cặp răng 1-2, còn đối với cặp răng 2-3 chỉ cần tính kiểm nghiệm.

Bảng 2.8 . Các công thức thiết kế truyền động bánh răng hành tinh

Tên thông số	*Công thức tính*	*Nº công thức*
Đường kính vòng lăn bánh trung tâm	$d_{w1} = K_d \sqrt[3]{\dfrac{T_1 K_{H\Sigma}(u+1)}{[\sigma_H]^2 u \varphi_{bd} q}}$	(2.67)
Hệ số phân bố không đều tải trọng (khi tính độ bền tiếp xúc) :	$K_{H\Sigma} = K_c + K^0_{H\beta} - 1$	(2.68)
Trên chiều rộng vành răng	$K^0_{H\beta}$ tra ở hình 6.5 [2] .	
Cho các bánh vệ tinh	K_c = 1,1 (khi số bánh vệ tinh q =3)	
Môđun ăn khớp	$m = \psi_{bd} d_{w1}$ /(12..15)	(2.69)
Tải trọng tương đương trên ổ lăn	$Q = \dfrac{Vs}{k_0 j}\sqrt{(\dfrac{4T_1 K_c}{d_{w1} q})^2 + F_{tt}^2}$ j - số ổ lăn trên bánh vệ tinh; V=1 khi vòng trong quay; V = 1,2 khi vòng ngoài quay; s - hệ số an toàn, s = 1,3; k_0 =1 khi j ≤ 2; k_0 = 1,3 khi j = 4.	(2.70)
Lực ly tâm đối với truyền động A	$F_{tt} = 6{,}7.10^{-11} d_2^2 b_w n_0^2 a_w \lambda_0$	(2.71)
Tuổi thọ thực tế của ổ lăn khi :		
Tải trọng không đổi	$L_i = (C/Q)^m$ triệu vòng quay	(2.72)
Tải trọng thay đổi, với s-số lần thay ổ	$L_{Et} = \dfrac{1}{s+1}\Sigma(\dfrac{T_{ij}}{T_1})^m L_i$ triệu vòng quay	(2.73)
Tải trọng không đổi và L_{ht} tính bằng giờ	$L_{ht} = \dfrac{10^6 L}{60(\lvert n_2 - n_0 \rvert)}$ giờ	(2.74)
Bậc của đường cong mỏi	m = 3 đối với ổ bi ; m = 10/3 đối với ổ đũa	

Chú thích : 1) Với bánh răng bằng thép K_d = 77 $MPa^{1/3}$ (đối với răng thẳng) và K_d = 67,5 $MPa^{1/3}$ (đối với răng nghiêng)

2) Hệ số phân bố không đều tải trọng khi tính uốn $K_{F\Sigma} = K_{H\Sigma}$

3. Vì lực tác dụng khi ăn khớp và môđun ăn khớp của các cặp răng là như nhau, trong khi đó độ bền của cặp răng ăn khớp trong cao hơn nên thường chọn vật liệu cặp bánh răng 1- 2 có cơ tính cao hơn (vật liệu nhóm II) và nhiệt luyện bánh răng đạt độ rắn cao hơn, còn bánh răng 3 thường được làm bằng vật liệu nhóm I và được tôi cải thiện để đạt độ rắn $HB_3 \leq 320$.

4. Cách xác định ứng suất cho phép $[\sigma'_H]$ và $[\sigma_H]$ tương tự như đối với truyền động thường, chỉ khác là khi xác định số chu kỳ chịu tải để tính hệ số tuổi thọ K_{HL} và K_{FL} thì phải xét các khâu trong chuyển động tương đối đối với cần, cụ thể là số chu kỳ thay đổi ứng suất tiếp xúc của các bánh răng 1, 2 và 3 sẽ là [4]:

$$N_{H1} = 60qt_{\Sigma}(|n_1 - n_0|) = 60qt_{\Sigma}n_0 p$$

$$N_{H2} = 60t_{\Sigma}(|n_2 - n_0|) = N_{H1}/(u_{12}q)$$

$$N_{H3} = N_{H2}q/u_{23},$$

trong đó q là số bánh vệ tinh; n_0, n_1 và n_2 lần lượt là số vòng quay của cần, của bánh trung tâm 1 và bánh vệ tinh 2 và p là tỉ số răng.

5. Vì truyền động hành tinh gồm q bánh vệ tinh ăn khớp đồng thời với các bánh trung tâm 1 và 3 nên sau khi xác định sơ bộ đường kính vòng lăn bánh trung tâm 1 (là bánh nhỏ của cặp răng 1-2) và môđun ăn khớp, cần xác định số răng của các bánh răng nhằm đảm bảo điều kiện lắp, điều kiện đồng trục và điều kiện kề trước khi tiếp tục xác định các thông số khác và tiến hành tính kiểm nghiệm cặp răng 1-2 và 2-3.

6. Khi kiểm nghiệm độ bền uốn của bánh răng ăn khớp trong 3 cần sử dụng hệ số dạng răng Y_F cho trên hình 6.6 [2] phụ thuộc z_3, x_3 và số răng dao xọc z_d đồng thời thay $K_{F\beta} = K_{F\Sigma} = K_{H\Sigma}$ tính theo (2.68).

7. Để xác định kích thước ổ lăn nhằm kiểm tra khả năng lắp ổ trên bánh vệ tinh hoặc lắp ổ trên cần, sau khi tính được kích thước của cặp bánh răng 1-2, cần tính tải trọng tương đương trên ổ lăn theo (2.70), trong đó:

j – số ổ lăn của bánh vệ tinh; $K_c \approx 1,1$;

k_0 – hệ số kể đến sự phân bố không đều tải trọng cho các ổ: $k_0=1$ khi $j \leq 2$; $k_0=1,3$ khi $j = 4$;

F_{ll} – lực ly tâm tính theo (2.71), ở đây hệ số $\lambda_0 = 1$ khi lắp ổ trên cần; $\lambda_0 = 0,5$ khi đặt ổ bên trong bánh vệ tinh,

sau đó dựa vào Q để chọn kích thước ổ theo khả năng tải động.

8. Trong nhiều kết cấu truyền động hành tinh thường sử dụng ổ lăn đặt trong lòng của bánh vệ tinh, do đó đường kính ngoài của ổ lăn D_n phải nhỏ hơn đường kính chân răng bánh vệ tinh, thường chọn

$$D_n = (0{,}65...0{,}75)d_{w2} \qquad (2.75)$$

Tuy nhiên để đảm bảo khả năng tải cần thiết của ổ lăn, có thể xảy ra trường hợp D_n lớn hơn giá trị tính được theo (2.75), khi đó có thể sử dụng một trong các biện pháp sau:

- Tăng thêm D_n tính theo (2.75) nhưng khoảng cách H từ D_n đến chân răng bánh vệ tinh phải đảm bảo điều kiện $H = 0{,}5(d_{f2} - D_n) \geq 2.m$, với m là môđun.
- Giảm thời hạn sử dụng ổ để giảm khả năng tải động C nhờ đó giảm được D_n;
- Tăng chiều rộng bánh vệ tinh nếu kết cấu cho phép (để đặt thêm ổ);
- Chọn cỡ ổ khác và xác định thời hạn sử dụng thực tế của ổ theo (2.72) hoặc (2.74) tuỳ theo tải trọng tĩnh (tính bằng triệu vòng quay hay tính bằng giờ) hoặc (2.73) khi tải trọng thay đổi.

Các bước tính toán thiết kế truyền động bánh răng hành tinh theo sơ đồ A được thể hiện bằng lưu đồ thuật giải trên hình 2.6.

2.3. XÂY DỰNG PHẦN MỀM TÍNH TOÁN THIẾT KẾ TRUYỀN ĐỘNG TRỤC VÍT - BÁNH VÍT

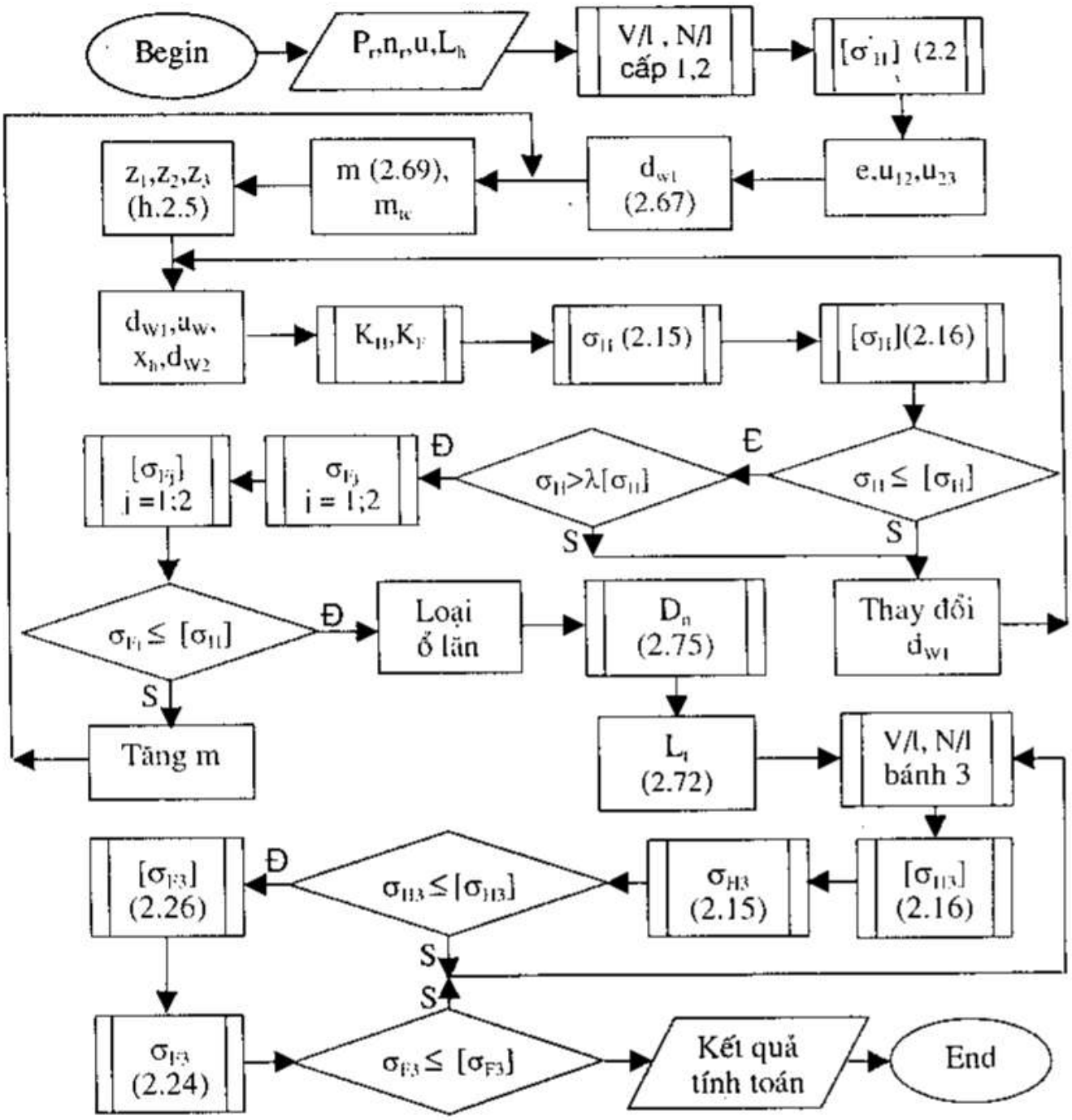
Truyền động trục vít dùng để truyền chuyển động quay giữa hai trục chéo nhau, góc giữa hai trục thường là 90° . Thông thường trục vít là chủ động và bánh vít là bị động. Dưới đây trình bày những vấn đề liên quan đến tính toán thiết kế trục vít trụ.

2.3.1. Các quan hệ hình học và công thức tính toán độ bền

Các thông số hình học của bộ truyền trục vít - bánh vít phục vụ thiết kế được trình bày trong bảng 2.9, còn các công thức thiết kế và kiểm nghiệm bộ truyền này về độ bền tiếp xúc và độ bền uốn cho trong bảng 2.10.

2.3.2. Các bước thiết kế và lưu đồ thuật giải

Khi bộ truyền trục vít làm việc cũng xuất hiện các dạng hỏng như trong truyền động bánh răng, bao gồm tróc rỗ, mòn, dính và gẫy, trong đó dạng hỏng mòn và dính là nguy hiểm hơn cả. Tuy nhiên vì mòn và dính cũng liên quan đến ứng suất tiếp xúc cho nên hiện nay vẫn tiến hành tính thiết kế bộ truyền theo độ bền tiếp xúc, sau đó kiểm nghiệm về uốn, tính



Hình 2.6 : Lưu đồ thuật giải thiết kế truyền động bánh răng hành tinh

nhiệt và khi cần thiết tiến hành kiểm nghiệm độ bền của thân trục vít. Nguy hiểm về dính và mòn sẽ được kể đến khi xác định ứng suất tiếp xúc cho phép.

Tính toán được tiến hành theo các bước sau đây:

1. Tính vận tốc trượt sơ bộ theo công thức

$$v_{sb} = 9{,}5.10^{-3} u \sqrt[3]{P_2 n_2^2} \qquad (2.76)$$

và dựa vào v_{sb} để chọn vật liệu bánh vít:

- Với $v_{sb} > 5$ m/s: dùng đồng thanh thiếc;

- Với $v_{sb} = 3 \div 5$ m/s: dùng đồng thanh nhôm sắt hoặc đồng thanh nhôm sắt niken;

- Với $v_{sb} < 3$ m/s và bộ truyền không quan trọng: dùng gang xám.

Bảng 2.9 . Các thông số hình học của bộ truyền trục vít - bánh vít

Tên thông số	*Ký hiệu*	*Công thức tính*
Khoảng cách trục	a_w	$a_w = 0{,}5m\,(q + z_2 + 2x)$
Đường kính vòng chia	d	$d_1 = m\,q$; $d_2 = m\,z_2$
Đường kính vòng lăn	d_w	$d_{w1} = (q + 2x)\,m$; $d_{w1} = d_2$
Đường kính vòng đỉnh	d_a	$d_{a1} = d_1 + 2m = m\,(q+2)$ $d_{a2} = m\,(z_2 + 2 + 2x)$
Đường kính vòng đáy	d_f	$d_{f1} = m\,(q - 2{,}4)$ $d_{f2} = m\,(z_2 - 2{,}4 + 2x)$
Đường kính ngoài của bánh vít	d_{aM2}	$d_{aM2} \le d_{a2} + 2m$ khi $z_1 = 1$ $d_{aM2} \le d_{a2} + 1{,}5m$ khi $z_1 = 2$ $d_{aM2} \le d_{a2} + m$ khi $z_1 = 4$
Chiều rộng bánh vít	b_2	$b_2 \le 0{,}75\,d_{a1}$ khi $z_1 = 1$ hoặc 2 $b_2 \le 0{,}6\,d_{a1}$ khi $z_1 = 4$
Góc ôm	δ	$\delta = \arcsin[b_2 / (d_{a1} - 0{,}5\,m)]$

2. Xác định ứng suất cho phép $[\sigma_H]$:

- Với bánh vít làm bằng đồng thanh không thiếc hoặc gang, $[\sigma_H]$ được xác định từ điều kiện tránh dính, tra trực tiếp từ bảng 7.2 [2] phụ thuộc vào trị số của vận tốc trượt;

- Với bánh vít làm bằng đồng thanh thiếc, $[\sigma_H]$ được xác định theo độ bền mỏi theo công thức sau:

$$[\sigma_H] = [\sigma_{H0}]\,K_{HL} \qquad (2.77)$$

trong đó

$[\sigma_{H0}]$ – ứng suất tiếp xúc cho phép ứng với 10^7 chu kỳ,

$[\sigma_{H0}] = (0{,}75 \div 0{,}9)\,\sigma_b$, với σ_b tra bảng 7.1 [2], hệ số 0,75 được dùng khi trục vít bằng thép không tôi, hệ số 0,9 được dùng khi trục vít được

Bảng 2.10. Các công thức thiết kế và kiểm nghiệm bộ truyền trục vít

Tên thông số	*Công thức tính toán*	*№ công thức*
Khoảng cách trục	$a_w = (z_2 + q)\sqrt[3]{(\frac{170}{z_2[\sigma_H]})^2 \frac{T_2 K_H}{q}}$	(2.78)
	trong đó $[\sigma_H]$ xác định theo (2.77)	
Mô đun	$m = 2a_w/(z_2 + q)$	(2.79)
Hệ số dịch chỉnh	$x = (a_w/m) - 0,5(q + z_2)$	(2.80)
Ứng suất tiếp xúc	$\sigma_H = (170/z_2)\sqrt{(\frac{z_2+q}{a_w})^3 \frac{T_2 K_H}{q}} \le [\sigma_H]$	(2.81)
Hệ số tải trọng	$K_H = K_{H\beta} K_{Hv}$	
	trong đó $K_{H\beta} = 1 + \frac{(z_2/\theta)^3}{1 - T_{2m}/T_{2\max}}$	
	$T_{2m} = \Sigma T_{2i} t_i n_{2i} / \Sigma t_i n_{2i}$	
Vận tốc trượt	$v_t = \pi d_{w1} n_1 /(60000 \cos\gamma_w)$	(2.82)
Góc vít lăn	$\gamma_w = arctg \frac{z_1}{q + z_2}$	
Hiệu suất	$\eta = 0,95 tg(\gamma_w)/tg(\gamma_w + \varphi)$	(2.83)
Ứng suất uốn	$\sigma_F = 1,4 T_2 Y_F K_F /(b_2 d_2 m_n) \le [\sigma_F]$	(2.84)
Mô đun pháp	$m_n = m\cos\gamma$	

thấm cacbon hoặc tôi đạt độ rắn HRC ≥ 45, mặt ren trục vít được mài và đánh bóng;

K_{HL} - hệ số tuổi thọ khi tính về tiếp xúc

$$K_{HL} = \sqrt[8]{10^7 / N_{HE}} \qquad (2.85)$$

trong đó số chu kỳ thay đổi ứng suất tương đương

$$N_{HE} = 60\Sigma(T_{2i}/T_{2\max})^4 n_{2i} t_i , \qquad (2.86)$$

với n_{2i}, t_i , T_{2i} lần lượt là số vòng quay trong một phút, số giờ làm việc và

mômen xoắn trên bánh vít ở chế độ i; T_{2max} là mômen xoắn lớn nhất trong các trị số T_{2i}; nếu tính ra $N_{HE} > 25.10^7$ thì lấy N_{HE} bằng 25. 10^7 chu kỳ để tính, nếu tính ra $N_{HE} < 5. 10^4$ thì lấy N_{HE} bằng 5. 10^4 chu kỳ để tính [1, 4].

3. Xác định sơ bộ khoảng cách trục a_w của bộ truyền theo độ bền tiếp xúc theo công thức (2.78). Muốn vậy hãy dựa vào tỉ số truyền u để chọn số mối ren trục vít z_1, tính số răng z_2 sao cho $26 \div 28 \leq z_2 \leq 60 \div 80$ rồi từ z_2 tính sơ bộ hệ số đường kính q theo công thức

$$q = (0,25 \div 0,3) z_2 \qquad (2.87)$$

và chọn q tiêu chuẩn (theo bảng 7.3 [2]). Chú ý chọn trị số 0,3 khi tỉ số truyền u lớn để trục vít có đường kính ($d_1 = qm$) đủ lớn nhằm đảm bảo độ cứng cho trục vít.

4. Xác định các thông số ăn khớp: trước hết xác định m theo (2.79), lấy m tiêu chuẩn, từ m và q tiêu chuẩn và z_2 đã chọn tính lại a_w. Muốn có a_w định trước cần xác định hệ số dịch chỉnh bánh vít x theo (2.80) và nên giới hạn nó trong khoảng $-0,7 \leq x \leq 0,7$. Nếu x không thoả mãn điều kiện này có thể tăng hoặc giảm q hoặc thay đổi z_2 đi 1 hoặc 2 răng nhưng với điều kiện sai số của tỉ số truyền không vượt quá 4%.

5. Kiểm nghiệm răng về độ bền tiếp xúc theo (2.81) đồng thời cũng cần kiểm tra điều kiện (2. 88) sau đây:

$$\sigma_H \geq k[\sigma_H] \qquad (2.88)$$

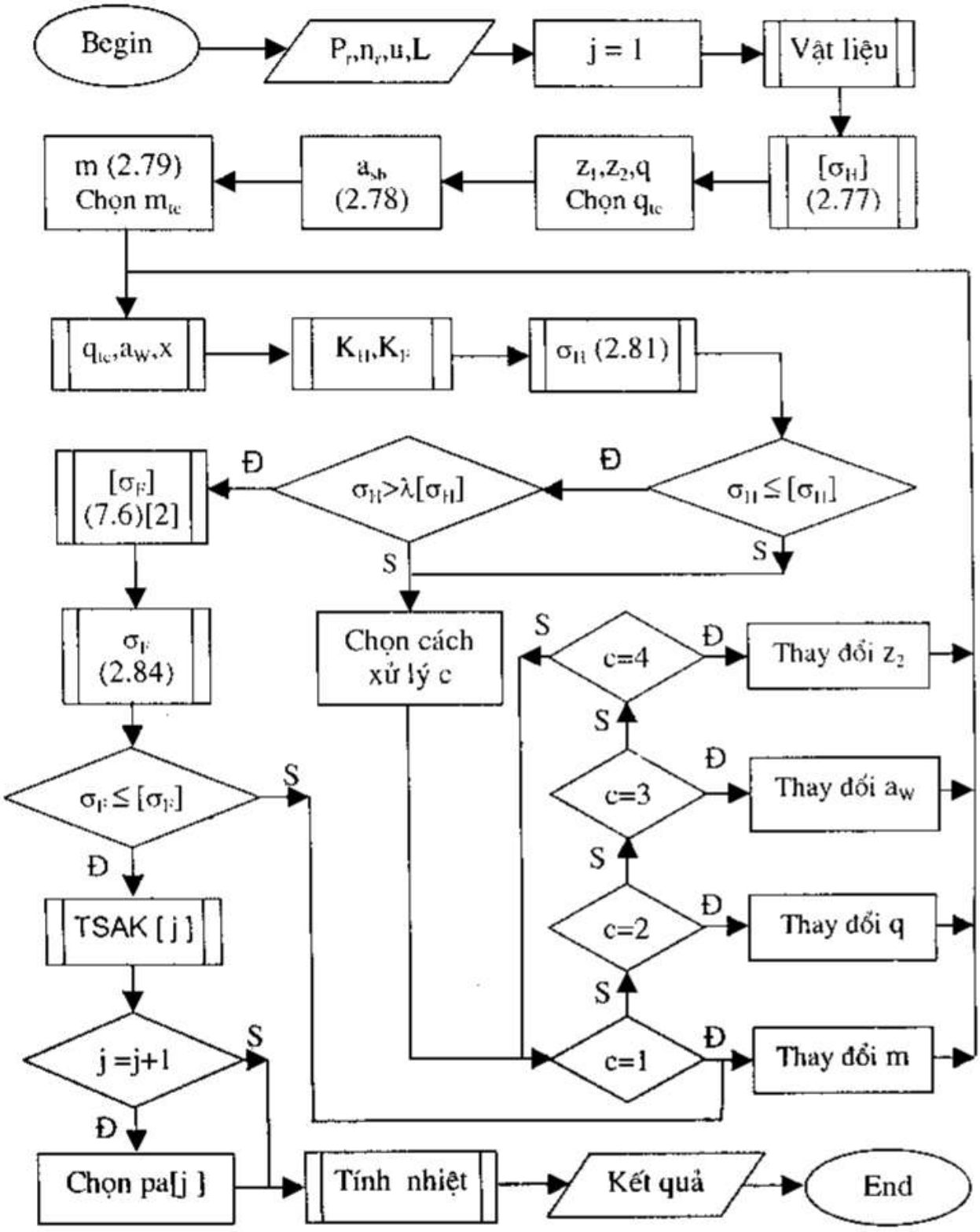
trong đó k là một hệ số nhỏ hơn 1 nhằm tránh cho bộ truyền không quá thừa bền.

6. Kiểm nghiệm răng về độ bền uốn theo (2.84) với ứng suất uốn cho phép $[\sigma_F] = [\sigma_{FO}]K_{FL}$, trong đó $[\sigma_{FO}]$ – ứng suất uốn cho phép ứng với 10^6 chu kỳ, tính theo công thức (7.7) [2] khi bộ truyền quay một chiều hoặc theo công thức (7.8) [2] khi bộ truyền quay hai chiều; K_{FL}- hệ số tuổi thọ khi tính về uốn, được xác định theo công thức (7.9) [2].

7. Xác định lần cuối các thông số cơ bản và kích thước bộ truyền theo các công thức ghi trong bảng 2.9.

8. Tính toán nhiệt nhằm xác định diện tích toả nhiệt cần thiết của hộp trục vít hoặc kiểm tra nhiệt độ do bộ truyền sinh ra không vượt quá nhiệt độ cho phép ($[t_d] = 70 \div 90^0C$).

Các bước tính toán thiết kế truyền động trục vít trên đây được thể hiện bằng lưu đồ thuật giải trình bày trên hình 2.7, trong đó TSAK[j] ký hiệu các thông số ăn khớp (m, q,...) ứng với lần tính thứ j .



Hình 2.7: Lưu đồ thuật giải thiết kế truyền động trục vít - bánh vít

2.3.3. Những vấn đề cần lưu ý khi xây dựng phần mềm thiết kế truyền động trục vít

Khi lập trình thiết kế truyền động trục vít cần lưu ý những vấn đề sau đây:

a) Cũng như khi thiết kế truyền động bánh răng, trước khi chọn vật liệu cũng như xác định ứng suất cho phép và tiến hành tính toán truyền động trục vít, cần tạo *File dữ liệu* thiết kế và nạp sẵn trên đĩa, bao gồm: công suất P (kW), số vòng quay n (vg/ph), tỉ số truyền u và thời hạn sử dụng L_h (giờ).

b) Như đã trình bày trong bước 4, số răng bánh vít z_2 có thể thay đổi so với lúc đầu nhằm đảm bảo cho hệ số dịch chỉnh x của bánh vít không quá lớn, tức là đảm bảo điều kiện $-0,7 \le x \le 0,7$ với yêu cầu sai lệch của tỉ số truyền nằm trong giới hạn cho phép ($\Delta u \le 4\%$), tuy nhiên đáp án có số răng thay đổi ít hơn phải được chọn trước. Muốn vậy cần soạn thảo một chương trình con chọn z_2 với vòng lặp thay đổi z_2 nhằm đảm bảo các điều kiện $-0,7 \le x \le 0,7$ và $\Delta u \le 4\%$.

Có thể sử dụng chương trình con sau đây:

```
Procedure chonz;
begin
  i:=1;
  repeat
    if ( i mode 2 ) = 0 then k:= 0 else k:= 2*i ;
    z2:= z2-k+i
    ut:= z2/ z1 ;
    du:= (u - ut) / u ;
    x:= aw/m -(q+z2) / 2 ;
    i:= i+1;
  until ((-0.7≤x ≤0.7) and (Δu≤0.04)) or ( i > 4 ) ;
end;
```

Kết quả thực hiện vòng lặp lần lượt sẽ là :

i	*1*	*2*	*3*	*4*
k	*2*	*0*	*6*	*0*
z_2	z_2-*1*	z_2+*1*	z_2-*2*	z_2+*2*

nghĩa là phương án thay đổi z_2 ít hơn sẽ được chọn trước.

c) Ở bước 5, nếu kiểm nghiệm độ bền tiếp xúc không đạt yêu cầu, tức là tính ra $\sigma_H > [\sigma_H]$ hoặc σ_H quá thừa bền, tức là $\sigma_H < k.[\sigma_H]$ thì có thể thay đổi các thông số theo một trong các phương án sau đây:

- Thay đổi môđun (do đó a_w sẽ thay đổi theo).
- Thay đổi hệ số đường kính q (do đó a_w sẽ thay đổi theo).
- Thay đổi khoảng cách trục a_w (do đó m và q sẽ thay đổi theo).
- Thay đổi số răng bánh vít z_2.

Muốn vậy hãy dùng lệnh lựa chọn *Case* với các biến i, j và k trong chương trình con sau đây để thực hiện ý tưởng trên, trong đó để biểu thị ng suất tiếp xúc cho phép ta sử dụng ký hiệu σ_{HP} thay cho $[\sigma_H]$:

```
Procedure lap;
begin
 z2:= round (u*z1) ;
 j:=1 ; jj:= 0; js:= 0 ; jk:= 0;
 repeat
   case pa of
    1: begin
        mt[j]:=m+1.5*jj;
        trate(2,1,mt[j],'b46.mq',m);          {để tra m tiêu chuẩn}
        aw:=round (m*(q+z2)/2) /5)*5; {để có aw tận cùng bằng 0 hoặc 5}
       x:= aw/m -(q+z2) /2;
      end;
    2: begin
        qt[j ]:=q+1.5*js;
        trate(2, 2, qt[j ],' b46.mq', q);     {để tra q tiêu chuẩn }
        aw:=round (m*(q+z2)/2) /5)*5;
       x:= aw/m -(q+z2) /2;
      end;
    3: begin
        at[j ]:= aw+5*jk;
        aw:= at[j ];
        x:= aw/m -(q+z2) /2;
       end;
    4:begin
       chonz;                          {gọi tới procedure chonz ở mục 2.3.3.b)}
      end;
   end;
   σH[j ] : = 170/z2*sqrt(exp (3*ln(z2+q))*T2*KH/q);
   if σH [j ] > σHP[j ] then
   begin
     write(' pa = ');
     readln(pa);
     case pa of
       1: jj:= jj +1; 2: js:= js+1; 3: jk:= jk+1;
     end;
     j:=j+1;
    end;
  until (σH [j ] <= σHP[j ] ) and (-0.7≤x ≤0.7);
end;
```

d) Chú ý rằng khi sử dụng các giải pháp khác nhau để xử lý, có thể xảy ra tình huống một số giải pháp nào đó đều thỏa mãn điều kiện bền tiếp xúc, điều kiện $-0,7 \leq x \leq 0,7$ và sai lệch tỉ số truyền dưới 4%. Khi đó cần tiến hành so sánh chúng với nhau nhằm chọn ra phương án tốt nhất để dùng (bước chọn pa[j] trong lưu đồ thuật giải trên hình 2.7). Các thông số được dùng làm căn cứ để lựa chọn là: khoảng cách trục a_w nhỏ hơn, sai lệch tỉ số truyền ít hơn, chênh lệch giữa σ_H và $[\sigma_H]$ ít hơn.

e) Ngoài các phương án liên quan đến các cách xử lý khác nhau nhưng sử dụng cùng một loại vật liệu bánh vít, khi cần thiết có thể tiến hành thiết kế bộ truyền ứng với các phương án sử dụng vật liệu khác nhau để chế tạo bánh vít, sau đó cũng tiến hành so sánh với các phương án khác nhằm chọn ra phương án thích hợp nhất để dùng. Ngoài yêu cầu về kích thước gọn, kết cấu hợp lý, khi lựa chọn phương án còn cần lưu ý đến hiệu suất bộ truyền (xem thí dụ ở chương 4, mục 4.3.3).

g) Sau khi hoàn thành bước tính toán thứ bảy, tức là xác định lần cuối các thông số cơ bản và kích thước bộ truyền theo các công thức cho trong bảng 2.9, cần ghi các số liệu phục vụ các bước thiết kế tiếp theo vào các *File* tương ứng trên đĩa cũng như tổng hợp các số liệu cần thiết để ghi ra màn hình và in các kết quả tính toán thiết kế ứng với dữ liệu đã cho.

2.4. XÂY DỰNG PHẦN MỀM TÍNH TOÁN THIẾT KẾ TRUYỀN ĐỘNG ĐAI

Truyền động đai được dùng để truyền chuyển động giữa các trục xa nhau, bao gồm:

- Truyền động đai dẹt có tiết diện là hình chữ nhật với các thông số tiêu chuẩn là chiều dày đai δ, chiều rông đai b.

- Truyền động đai hình thang, gồm hai loại: đai thang thường với các tiết diện tiêu chuẩn là Z, O, A, B, C, D (theo TCVN) và O, A, Б, B, Г, Д, E (theo tiêu chuẩn Nga) và đai thang hẹp với các tiết diện tiêu chuẩn SPZ, SPA , SPB (theo TCVN) và УО, УА, УБ (theo tiêu chuẩn Nga).

- Truyền động đai hình lược với các tiết diện tiêu chuẩn K, Л, M (theo tiêu chuẩn Nga).

Ngoài tiết diện tiêu chuẩn, ở đai hình thang và đai hình lược còn tiêu chuẩn về chiều dài đai l.

Mỗi một loại đai sẽ được mắc lên các bánh đai tương ứng có đường kính, chiều rộng,... được tiêu chuẩn hoá.

2.4.1. Các quan hệ hình học và công thức tính toán khả năng tải

Các công thức dùng để xác định các kích thước và thông số của đai và bộ truyền đai cho trong bảng 2.11.

Bảng 2.11 . Các thông số của đai và bộ truyền đai

Tên thông số	*Công thức tính*	*Số công thức*
Góc ôm	$\alpha = 180 - (d_2 - d_1)57 / a$	(2.89)
Chiều dài đai	$l = 2a + \pi(d_1 + d_2)/2 + (d_2 - d_1)^2 /(4a)$	(2.90)
Tỉ số truyền	$u = \dfrac{d_2}{d_1(1-\varepsilon)}$	(2.91)
Số lần uốn của đai trong 1 giây	$i = v / l \le [i]$	(2.92)
Chiều rộng đai dẹt	$b = \dfrac{1000 P_1 K_d}{v\delta[\sigma_F]_o C_\alpha C_a C_v}$	(2.93)
Chiều rộng bánh đai dẹt	$B = 1{,}1b + (10 \sim 15)\,mm$	(2.94)
Số đai hình thang	$z = \dfrac{P_1 K_d}{[P_0] C_\alpha C_l C_u C_z}$	(2.95)
Số chêm hình lược	$z = \dfrac{P_1 K_d}{[P_0] C_\alpha C_l C_u}$	(2.96)
Chiều rộng bánh đai hình thang và hình lược	$B = (z-1)t + 2e$ *với t, e - xem bảng 8.5 [2] đối với đai hình thang hoặc bảng 8.6[2] đối với đai hình lược*	(2.97)
Lực tác dụng lên trục	$F_r = 2F_0 \sin(\alpha_1 / 2)$	(2.98)

2.4.2. Những vấn đề cần lưu ý khi xây dựng phần mềm thiết kế truyền động đai

Chỉ tiêu thiết kế truyền động đai là khả năng kéo và tuổi thọ của đai. Với các bộ truyền đai thông dụng thường tiến hành thiết kế truyền động

đai theo khả năng kéo và tiến hành kiểm nghiệm đai về tuổi thọ. Chỉ tiêu tuổi thọ cũng được quan tâm khi lựa chọn một số thông số của bộ truyền đai như đường kính bánh đai, góc ôm của đai, chiều dài đai và khoảng cách trục, v.v... Như vậy thiết kế truyền động đai được thực hiện theo các bước sau đây:

2.4.2.1. Đối với truyền động đai dẹt:

1. Chọn loại đai.

2. Xác định đường kính bánh đai nhỏ theo công thức thực nghiệm

$$d_1 = (1100 \div 1300)\sqrt[3]{P_1 / n_1}$$

trong đó P_1, n_1 lần lượt là công suất (kW) và số vòng quay (vg/ph) của bánh đai 1, lấy giá trị d_1 tiêu chuẩn và tiến hành kiểm nghiệm về vận tốc theo công thức

$$v = \pi d_1 n_1 / 60000 \quad \text{m/s} \qquad (2.99)$$

và nghiệm điều kiện $v \le 25 \div 30$ m/s, nếu không đảm bảo thì nên chọn trong dãy trị số tiêu chuẩn giá trị d_1 nhỏ hơn.

3. Tính đường kính bánh đai lớn d_2 theo d_1 và tỉ số truyền u

$$d_2 = u d_1 (1 - \varepsilon) \qquad (2.100)$$

với hệ số trượt $\varepsilon = 0{,}01 \div 0{,}02$ và lấy d_2 tiêu chuẩn. Chú ý chọn d_1 và d_2 sao cho sai lệch tỉ số truyền u_t tính theo (2.91) và u cho trước không vượt quá 4%, tức là

$$\Delta u = abs((u_t - u)/u_t) \le 0{,}04. \qquad (2.101)$$

4. Xác định khoảng cách trục a và chiều dài đai l:

- Trường hợp không cho trước khoảng cách trục, có thể chọn a theo chiều dài tối thiểu l_{min} tính từ số vòng chạy cho phép của đai trong một giây

$$i = v/l \le 3 \div 5 \ \text{1/s, tức là } l_{min} = v/(3 \div 5)$$

- Tính a theo l_{min} theo công thức

$$a = \frac{2l - \pi(d_1 + d_2) + \sqrt{[2l - \pi(d_1 + d_2)]^2 - 8(d_2 - d_1)^2}}{8} \qquad (2.102)$$

và kiểm nghiệm điều kiện $a \ge 2(d_1+d_2)$ nhằm giảm tần số thay đổi ứng suất trong đai, như thế sẽ có lợi về tuổi thọ đồng thời đảm bảo cho góc ôm không quá nhỏ. Nếu $a < 2(d_1+d_2)$ thì lấy $a = 2(d_1+d_2)$ và từ a tính lại

chiều dài đai theo công thức (2.90) và tăng l thêm 100 ÷ 400 mm tuỳ theo cách nối đai.

5. Tính góc ôm α theo công thức (2.89) với yêu cầu $\alpha \geq 150^0$.

6. Chọn chiều dày δ của đai theo đường kính bánh đai nhỏ d_1 sao cho tỉ số $\delta / d_1 \geq 1/40$; 1/35 ; 1/30 tương ứng với đai vải cao su, đai da và đai sợi bông nhằm hạn chế sinh ra ứng suất uốn quá lớn tác dụng lên đai. Trị số δ tính được cần lấy theo tiêu chuẩn (bảng 8.1[2] đối với đai vải cao su).

7. Xác định chiều rộng đai tính toán b_t theo công thức (2.93), trong đó

K_d – tra bảng 4.7 [2];

$C_\alpha = 1 - 0{,}003(180 - \alpha_1)$;

$C_v = 1{,}04 - 0{,}0004v^2$ đối với đai vai cao su, đai da và đai sợi bông,

$C_v = 1{,}01 - 0{,}0001v^2$ đối với đai sợi tổng hợp;

$C_0 = 1$ khi $\beta \leq 60^0$ hoặc bộ truyền có bộ phận tự căng đảm bảo lực căng không đổi,

$C_0 = 0{,}9$ khi $60^0 < \beta \leq 80^0$,

$C_0 = 0{,}8$ khi $80^0 < \beta \leq 90^0$.

Sau khi tính được b_t lấy b_t theo giá trị b tiêu chuẩn (bảng 8.1 [2]).

Nhằm đảm bảo cho đai không bị quá tải hoặc non tải nhiều, ảnh hưởng đến chất lượng làm việc của đai, cần hạn chế sai lệch tương đối giữa trị số tính toán b_t và trị số tiêu chuẩn b của chiều rộng đai dẹt theo điều kiện sau:

$$\Delta b = abs((b_t - b)/b_t) \leq 0{,}1 \qquad (2.103)$$

Nếu điều kiện (2.103) không thoả mãn cần thay đổi đường kính bánh đai và tính lại b_t.

8. Tính chiều rộng bánh đai theo công thức (2.94).

9. Tính lực tác dụng lên trục theo công thức (2.98), với $F_0 = \sigma_0 b\delta$, trong đó σ_0 - bảng 8.8 [2].

2.4.2.2. Đối với truyền động đai hình thang:

1. Chọn loại đai theo điều kiện làm việc và kích thước khuôn khổ mong muốn. Thông thường với v < 25 m/s dùng đai thang thường và với

$v \geq 25$ m/s dùng đai thang hẹp. Tuy nhiên nếu muốn có kích thước nhỏ gọn thì nên dùng đai thang hẹp.

Để chọn được loại tiết diện đai thang thường của Nga có thể dựa vào công suất trên bánh chủ động P_1 và số vòng quay n_1 của bánh chủ động trên đồ thị hình 4.1 [2].

2. Chọn đường kính bánh đai nhỏ. Trong bảng 4.13 [2] cho khoảng đường kính bánh đai nhỏ phụ thuộc loại tiết diện đai, còn trong bảng 4.21 [2] cho các giá trị tiêu chuẩn của đường kính. Muốn có kích thước nhỏ gọn nên dùng giá trị d_1 lớn hơn d_{min} một bậc. Tính vận tốc v theo (2.99) và nghiệm điều kiện $v \leq v_{max}$, với $v_{max} = 25$ m/s, nếu không đảm bảo thì nên chọn trong dãy trị số tiêu chuẩn giá trị d_1 nhỏ hơn hoặc dùng đai thang hẹp thay cho đai thang thường.

3. Tính đường kính bánh đai lớn d_2 theo công thức (2. 100), với hệ số trượt $\varepsilon = 0{,}01 \div 0{,}02$ và lấy d_2 sao cho sai lệch tỉ số truyền không vượt quá 4%.

4. Xác định chiều dài đai và khoảng cách trục. Trường hợp không cho trước khoảng cách trục a thì dựa vào tỉ số truyền u để chọn tỉ số a / d_2 theo dãy số dưới đây, biết d_2 sẽ suy ra a:

u	1	2	3	4	5	≥ 6
a/d_2	1,5	1,2	1	0,95	0,9	0,85

Trị số a tính được nên nằm trong khoảng sau (với h là chiều cao tiết diện đai hình thang):

$$0{,}55\,(d_1+d_2) + h \leq a \leq 2\,(d_1+d_2)$$

Từ a xác định được l theo (2.90) và chọn l theo tiêu chuẩn (bảng 4.13 [2]), sau đó từ l tiêu chuẩn tính lại khoảng cách trục a theo (2.102).

5. Tính góc ôm α theo công thức (2.89) với yêu cầu $\alpha \geq 120^0$.

6. Xác định số đai tính toán z_1 theo công thức (2.95) với chú ý là số đai phải là số nguyên z và z không nên quá 6 nhằm giảm bớt sự phân bố không đều tải trọng cho các đai, trong đó:

K_d - tra bảng4.7 [2] ;

C_α - tra bảng 4.15 [2] ; C_l - tra bảng 4.16 [2] ; C_u - tra bảng 4.17 [2] ;

C_z - tra bảng 4.18 [2] ;

$[P_0]$ - tra bảng 4.19 [2] đối với đai thang thường và tra bảng 4.20 [2] đối với đai thang hẹp.

Nhằm đảm bảo cho sai lệch tương đối giữa z_t và z không quá lớn làm cho đai bị quá tải hoặc non tải nhiều, ảnh hưởng đến chất lượng làm việc của đai, cần lặp lại việc xác định z cho tới khi thoả mãn điều kiện sau:

$$\Delta z = abs\left((z_t - z)/z_t\right) \le 0{,}1 \qquad (2.104)$$

Muốn vậy có thể xử lý theo một trong hai cách sau đây:

- Thay đổi đường kính bánh đai bằng cách sử dụng công thức

$$d_t = d_1 + jc \qquad (2.105)$$

Lưu ý rằng các khoảng chênh lệch giữa các giá trị tiêu chuẩn của đường kính rất khác nhau nên cần dựa vào dãy trị số tiêu chuẩn này (bảng 4.13 [2]) để định ra hệ số c trong (2.105) thích hợp cho từng khoảng đường kính. Đồng thời còn phải gán dấu thích hợp cho j [chọn j = +1 khi cần giảm số đai tính toán z tính theo (2.95) và j = – 1 khi cần tăng z], nhờ đó đảm bảo được điều kiện (2.104).

- Chọn loại tiết diện khác để dùng.

7. Tính chiều rộng bánh đai theo (2.97), trong đó t, e - xem bảng 8.5[2] và tính đường kính ngoài của các bánh đai.

8. Tính lực tác dụng lên trục theo (2.98), trong đó lực căng F_0 trên z đai được tính theo công thức

$$F_0 = 780P_1K_d/(vC_\alpha) + F_v$$

trong đó F_v – lực căng do lực ly tâm sinh ra; $F_v = 0$ khi bộ truyền có khả năng tự động điều chỉnh lực căng; $F_v = q_m . m^2$ nếu định kỳ điều chỉnh lực căng, với q_m là khối lượng 1 mét chiều dài đai, tra theo bảng 4.22 [2].

Trên hình 2.8 là lưu đồ thuật giải thiết kế truyền động đai hình thang theo các bước trình bày ở trên, trong đó quan tâm đến việc lựa chọn tối ưu các thông số của bộ truyền và chọn cách xử lý khi không đảm bảo sai lệch tương đối về tỉ số truyền, về số đai tính toán và số đai nguyên trong phạm vi cho phép.

2.4.2.3. Đối với truyền động đai hình lược:

Về cơ bản tiến hành như đối với đai hình thang nhưng có những điểm khác biệt sau đây:

1. Chọn loại tiết diện đai K, Л, M (theo tiêu chuẩn Nga) tương ứng với công suất tăng dần.

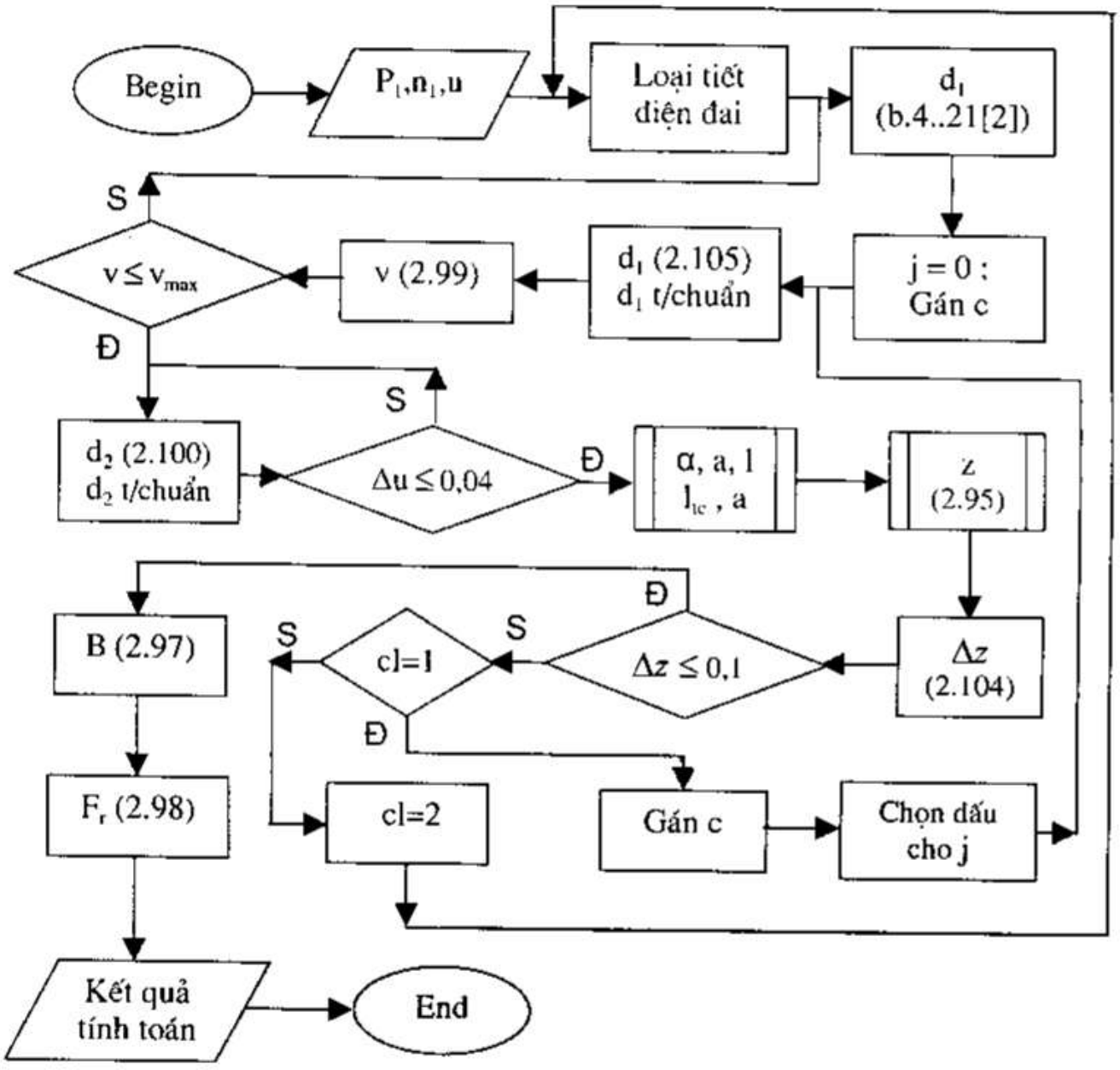
2. Tính đường kính bánh đai nhỏ theo công thức thực nghiệm sau (với T_1 – Nmm):

$$d_1 = 3\sqrt[3]{T_1} \text{ mm} \qquad (2.106)$$

Đường kính d_1 tính theo (2.106) cần lấy theo tiêu chuẩn trong dãy số sau:

25 ; 40 ; 45 ; 50 ;56 ; 63 ;71 ; 80 ; 90 ;100 ;112 ;125 ; 140 ;160 ;180 ; 200; 224 ; 250 ; 280 ; 315 ;355 ; 400 ; 450 ;500 ; 560 ; 630; 710 ; 800; 900 ; 1000

và nên lớn hơn d_{min} cho trong bảng 4. 23 [2].



Hình 2.8. Lưu đồ thuật giải thiết kế truyền động đai hình thang

3. Số chêm được xác định theo (2.96) với các hệ số được xác định như đối với đai hình thang, riêng công suất cho phép được tra trong bảng

4.24 [2]. Số chêm của đai hình lược nên là số chẵn, trong phạm vi nên dùng và không vượt quá số chêm cho phép ghi trong bảng 4.23 [2].

4. Lực tác dụng lên trục và lực căng đai với z chêm cũng được tính toán như đối với đai hình thang, chỉ khác ở chỗ lực căng do lực ly tâm sinh ra, ứng với trường hợp lực căng được điều chỉnh định kỳ, được tính theo công thức

$$F_v = 0{,}1 q_{m10} z v^2 \qquad (2.107)$$

trong đó q_{m10} -- khối lượng 1 mét đai có 10 chêm, tra trong bảng 4.23 [2].

2.5. XÂY DỰNG PHẦN MỀM TÍNH TOÁN THIẾT KẾ TRUYỀN ĐỘNG XÍCH

2.5.1. Các quan hệ hình học và công thức thiết kế truyền động xích

Cũng như truyền động đai, truyền động xích được dùng để truyền chuyển động và mômen xoắn giữa các trục xa nhau nhưng nhờ sự ăn khớp của các má xích và răng đĩa. Về cấu tạo, xích truyền động được chia thành ba loại: xích ống, xích con lăn và xích răng.

Công thức xác định các thông số của xích và của bộ truyền cho trong bảng 2.12.

2.5.2. Những vấn đề cần lưu ý khi xây dựng phần mềm thiết kế truyền động xích

1. Dạng hỏng chủ yếu của truyền động xích là mòn bản lề, do vậy tính toán xích theo chỉ tiêu về độ bền mòn là phần cơ bản trong tính toán thiết kế truyền động xích. Phép tính được bắt đầu bằng việc lựa chọn loại xích với chú ý rằng so với xích ống, xích con lăn có độ bền mòn cao hơn, chế tạo xích con lăn không phức tạp và đắt bằng xích răng nên với vận tốc dưới 10÷15 m/s nên dùng xích con lăn; khi cần truyền động với tải trọng và vận tốc lớn hơn nên dùng xích răng.

2. Tính xích ống và xích con lăn theo độ bền mòn nhằm xác định bước xích p trong khi đối với xích răng đó là nhằm xác định chiều rộng xích B sau khi đã chọn trước bước xích.

Sau khi xác định được các thông số khác của bộ truyền như khoảng cách trục và số má xích, đường kính các đĩa xích, còn tiến hành kiểm nghiệm xích về độ bền và kiểm nghiệm răng đĩa xích về độ bền tiếp xúc và tính lực tác dụng lên trục.

3. Tuỳ thuộc vào loại xích đã chọn, tính toán thiết kế truyền động xích được tiến hành theo trình tự khác nhau.

Bảng 2.12 . Các thông số và công thức tính toán thiết kế truyền động xích

Thông số	*Công thức tính*	*№ công thức*
Tỉ số truyền	$u = \frac{n_1}{n_2} = \frac{z_2}{z_1}$	(2.108)
Số mắt xích	$x = 2a/p + 0{,}5(z_1 + z_2) + (z_2 - z_1)^2 p/(4\pi^2 a)$	(2.109)
Khoảng cách trục	$a = \frac{p}{4}\left[\lambda + \sqrt{\lambda^2 - 2(\frac{z_2 - z_1}{\pi})^2}\right]$	(2.110)
	trong đó $\lambda = x_c - 0{,}5(z_2 + z_1)$	
Đường kính đĩa xích	$d_i = p/\sin(\pi/z_i)$, với i = 1 ; 2	(2.111)
Công suất tính toán	$P_t = P_1 k k_Z k_n / k_d \le [P_0]$	(2.112)
	trong đó $k_Z = z_{01}/z_1$	(2.113)
	$k_n = n_{01}/n_1$	(2.114)
	$[P_0]$ –xem bảng 5.5 [2] ;	
	k -xem bảng 5.6 và 5.7 [2] ;	
Hệ số an toàn	$s = Q/(k_d F_t + F_0 + F_v) \ge [s]$	(2.115)
Ứng suất tiếp xúc	$\sigma_H = 0{,}47\sqrt{k_r(F_t K_d + F_{vd})E/(Ak_{day})} \le [\sigma_H]$	(2.116)
Lực tác dụng lên trục	$F_r = 6.10^7 k_x P/(zpn)$	(2.117)

Đối với xích ống và xích con lăn:

Tính toán được thực hiện theo các bước sau đây:

a) Chọn số răng đĩa xích: số răng đĩa nhỏ z_1 được xác định theo công thức gần đúng sau

$$z_1 = 29 - 2u, \qquad (2.118)$$

lấy z_1 nguyên và nên dùng số răng lẻ. Từ z_1 và tỉ số truyền u cho trước để chọn được z_2 nguyên lẻ với sai số ít nhất về tỉ số truyền có thể sử dụng các câu lệnh sau:

$z_2 := trunc(u^* z_1)$;
if $(z_2 \bmod 2) = 0$ *then* $z_2 := z_2 + 1$;

Dùng số răng lẻ để ăn khớp với số mắt xích chẵn, xích sẽ mòn đều hơn.

b) Xác định bước xích p:

Trước hết dựa vào điều kiện sử dụng thực tế để xác định các hệ số thành phần trong công thức tính hệ số sử dụng k (bảng 5.6[2]), dựa vào z_1 đã chọn và z_{01}= 25 để tính k_Z theo (2.113), chọn n_{01} trong bảng 5.5 [2] gần nhất với n_1 đã biết để tính k_n theo (2.114) và bước đầu chọn số dãy bằng 1. Với k, k_z, k_n và công suất P_1 tính được P_t theo (2.112), dùng bảng 5.5 [2] ứng với n_{01} đã chọn sẽ xác định được công suất $[P_0] \geq P_t$, từ đó gióng sang cột 1 của bảng 5.5 [2] sẽ tìm được giá trị của bước xích p tương ứng. Trị số p tìm được phải thoả mãn điều kiện $p < p_{max}$, với p_{max} cho trong bảng 5.8 [2].

Cần lưu ý rằng khi dựa vào công suất tính toán P_t tính theo (2.112) để tra bước xích tiêu chuẩn p có thể phải xử lý hai tình huống sau:

- Mặc dù $p < p_{max}$ nhưng có thể bước xích vẫn còn tương đối lớn (thí dụ p > 38,1 mm) do đó đường kính đĩa xích lớn có thể khá lớn, bộ truyền sẽ khá cồng kềnh. Vì vậy để có bước xích nhỏ hơn có thể dùng xích nhiều dãy, khi đó tương ứng với số dãy xích đã chọn có thể suy ra hệ số k_{day} trong dãy số sau:

Số dãy xích	1	2	3	4
Hệ số k_{day}	1	1,7	2,5	3

Đem thay k_{day} vào (2.112) sẽ xác định được bước xích nhỏ hơn.

- Công suất cho phép $[P_0]$ lớn hơn nhiều so với P_t, tức là quá dư thừa khả năng tải, xích làm việc quá nhẹ nhàng trong khi kích thước bộ truyền có thể khá lớn. Để khỏi lãng phí và thu được bộ truyền gọn nhẹ hơn, có thể chọn các giải pháp xử lý sau đây:

* Cách 1:

Tăng hệ số k_Z để tăng P_t bằng cách giảm dần số răng đĩa nhỏ theo công thức $z_1 = z_1 + 2j$, với j = – 1; – 2;... cho tới khi thoả mãn điều kiện

$$\Delta P = ([P_0] - P_t)/P_t \leq [\Delta P] \qquad (2.119)$$
$$z_1 \geq z_{1\min}$$

trong đó với xích con lăn z_{1min}= 17 khi v ≥ 2 m/s và z_{1min}= 15 khi v < 2 m/s; sai lệch tương đối giữa công suất cho phép và công suất tính toán có thể lấy bằng 10%.

* Cách 2:

Giảm hệ số k_Z để giảm P_t bằng cách tăng dần số răng đĩa xích nhỏ theo công thức $z_1 = z_1 + 2j$, với j = 1; 2;... cho tới khi thoả mãn điều kiện (2.119), khi đó nhờ giảm được P_t có khả năng chọn được bước xích nhỏ hơn. Trường hợp bước xích không thay đổi cần chú ý không để kích thước bộ truyền tăng quá lớn, tức là cần đảm bảo điều kiên $d_2 \le d_{max}$, với d_{max} có thể chọn trong khoảng $d_{max} = 650 \div 700$ mm;

* Cách 3:

Tăng hệ số sử dụng k để tăng P_t bằng cách:

+ Chọn khoảng cách trục nhỏ hơn (a < 30p) nhưng phải đảm bảo điều kiện góc ôm $\alpha \ge 120^0$, do đó k_a tăng.

+ Dùng phương pháp điều chỉnh vị trí trục bằng con lăn căng xích thay cho trường hợp điều chỉnh lực căng bằng cách di chuyển một trong hai đĩa xích, hoặc dùng bộ truyền không điều chỉnh được lực căng, như thế k_{dc} sẽ tăng lên,...

* Cách 4:

Tăng số dãy xích và sử dụng thêm cách 1 hoặc cách 3 để giảm P_t cho tới khi thoả mãn điều kiện (2.119). Lúc này nhờ giảm được công suất tính toán P_t tương đối nhiều nên có thể chọn được bước xích nhỏ hơn và nhờ đó dễ đảm bảo điều kiên $d_2 \le d_{max}$.

Lưu đồ thuật giải chọn bước xích p của xích con lăn đảm bảo chỉ tiêu về độ bền mòn của bản lề xích cùng các hướng xử lý khác nhau cho trên hình 2.9.

c) Chọn sơ bộ khoảng cách trục $a = (30 \div 50)p$, tính số mắt xích theo (2.110), lấy số mắt xích chẵn x_c, từ số mắt xích chẵn tính lại a theo (2.111) và với bộ truyền đặt nằm ngang hoặc chếch một góc $\beta \le 70^0$ cần giảm bớt a một lượng $\Delta a = (0{,}002 \div 0{,}004)a$ để xích không quá căng.

d) Kiểm nghiệm số lần va đập của xích trong một giây theo công thức

$$i = z_1 n_1 / (15x) \le [i] \qquad (2.120)$$

trong đó [i] – số lần va đập cho phép trong một giây, 1/s, xem bảng 5.9[2].

e) Kiểm nghiệm xích về độ bền theo (2.115), trong đó

Q - tải trọng phá hỏng, tra bảng 5.2 hoặc 5.3 [2].

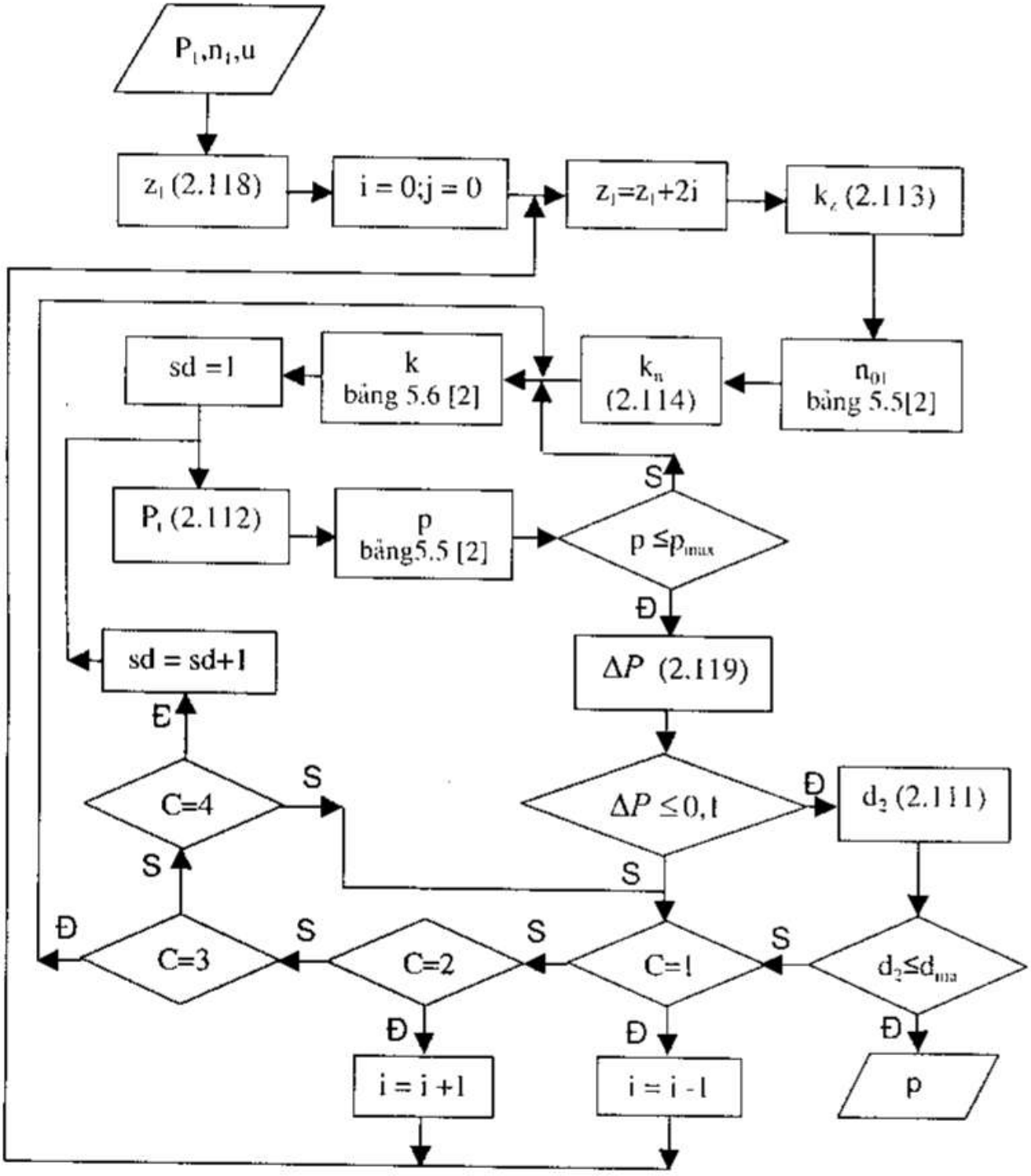
k_d – hệ số tải trọng động ; k_d = 1,2 ; 1,7 và 2 ứng với chế độ làm việc trung bình, nặng và rất nặng với tải trọng mở máy bằng 150, 200 và

300% so với tải trọng danh nghĩa.

$F_t = 1000P/v$ – lực vòng.

$F_v = qv^2$ – lực ly tâm do lực căng sinh ra với q là khối lượng 1 mét xích, tra bảng 5.2 hoặc 5.3 [2] phụ thuộc loại xích và bước xích và v – vận tốc xích, được xác định theo công thức

$$v = z_1 p n_1 / 60000 \ m/s \qquad (2.121)$$



Hình 2.9 : Lưu đồ thuật giải xác định bước xích của xích con lăn

$F_0 = 9{,}81 k_f q\, a$ – lực căng do trọng lượng nhánh xích bị động sinh ra, với a – khoảng cách trục, tính bằng mét; $k_f = 6; 4; 2$ ứng với độ võng của dây xích $f = (0{,}02 \div 0{,}02)a$.

[s] – hệ số an toàn cho phép, bảng 5.10 [2].

g) Tính đường kính các đĩa xích theo (2.111).

h) Kiểm nghiệm răng đĩa xích về độ bền tiếp xúc theo (2.116), với:

$[\sigma_H]$ - ứng suất tiếp xúc cho phép, bảng 5.11 [2].

$F_{vd} = 13.10^{-7} n_1 p^3 m$ – lực va đập trên m dãy xích.

K_d – hệ số tải trọng động, bảng 5.6 [2].

k_{day} – hệ số phân bố không đều tải trọng cho các dãy xích.

k_r – hệ số kể đến ảnh hưởng của số răng đĩa xích, chọn theo dãy số sau:

z	15	20	30	40	50	60
k_r	0,59	0,48	0,36	0,29	0,24	0,22

k) Xác định lực tác dụng lên trục theo (2.117).

Đối với xích răng:

Tính toán được thực hiện theo các bước sau đây:

a) Dựa vào tỉ số truyền của bộ truyền xích, chọn số răng đĩa xích nhỏ theo dãy số sau:

u	1.. 2	2.. 3	3.. 4	4.. 5	5.. 6	6
z_1	35.. 32	32.. 30	30.. 27	27.. 23	23.. 19	19.. 17

b) Dựa vào số vòng quay đĩa nhỏ n_1 (vg/ph) để xác định bước xích lớn nhất cho phép p_{max} theo dãy số sau đây rồi chọn p sao cho $p < p_{max}$:

n_1 (vg/ph)	3300	2650	2000	1650	1320
p_{max} (mm)	12,7	15,875	19,05	25,4	31,75

Đương nhiên muốn cho bộ truyền có kích thước gọn nên chọn p khá nhỏ so với p_{max} .

c) Xác định các hệ số thành phần trong công thức tính hệ số sử dụng k (bảng 5.6[2]), xác định vận tốc xích theo (2.121) và tính hệ số vận tốc k_v theo công thức

$$k_v = 1 + q_m v / P_1 \qquad (2.122)$$

trong đó:

P_1 – công suất cần truyền, kW.

v – vận tốc xích tính theo (2.121), m/s.

q_m – khối lượng một mét xích trên một đơn vị chiều rộng xích, kg/mm, được xác định theo bước xích p theo dãy số sau:

p mm	12,7	15,875	19,05	25,4	31,75
q_m kg/mm	0,058	0,072	0,086	0,114	0,145

d) Xác định chiều rộng cần thiết của xích răng theo công thức sau:

$$B = 250P_t\, k\, k_v / (pv^{2/3}) \qquad (2.123)$$

Chú ý rằng, ứng với một giá trị của bước xích p đã chọn có thể có một loạt giá trị của chiều rộng xích tiêu chuẩn B_t (bảng5.3[2]), do đó cần dựa vào giá trị của B tính theo (2.123) để chọn B_t gần nhất lớn hơn.

Các bước tính còn lại, bao gồm việc xác định khoảng cách trục và số má xích, tính kiểm nghiệm số lần va đập của xích trong một giây, kiểm nghiệm độ bền của xích, xác định đường kính của các đĩa xích và tính lực tác dụng lên trục, tiến hành tương tự như đối với xích con lăn.

Chương 3

XÂY DỰNG CHƯƠNG TRÌNH TÍNH TOÁN THIẾT KẾ TRỤC VÀ Ổ TRỤC

Máy móc, thiết bị được tạo thành nhờ liên kết các chi tiết máy và bộ phận máy với nhau. Đó có thể là các liên kết động (có sự chuyển động tương đối) hoặc liên kết cố định, tuỳ thuộc vào sơ đồ động của máy hoặc các yêu cầu về kết cấu và công nghệ. Chẳng hạn, muốn truyền được năng lượng, các chi tiết quay của truyền động cơ khí như: bánh đai, bánh răng, bánh vít, đĩa xích, v.v... cần được lắp cố định lên trục, bản thân trục cần có chuyển động tương đối với ổ trục và qua gối đỡ liên kết với khung và bệ máy. Dưới đây sẽ trình bày những vấn đề liên quan đến việc xây dựng phần mềm thiết kế trục và ổ trục.

3.1. XÂY DỰNG PHẦN MỀM TÍNH TOÁN THIẾT KẾ TRỤC

3.1.1. Chỉ tiêu tính toán và các công thức dùng trong thiết kế trục

Trục dùng để đỡ các chi tiết quay, để truyền mômen xoắn hoặc đồng thời thực hiện hai nhiệm vụ này. Thực tế sử dụng cho thấy rằng phần lớn các trục, đặc biệt là các trục quay nhanh, gẫy hỏng vì mỏi là dạng hỏng chủ yếu nhất. Vì vậy khi thiết kế trục, trước tiên cần xuất phát từ chỉ tiêu quan trọng nhất là độ bền mỏi để tính toán thiết kế trục, sau đó tuỳ theo yêu cầu và điều kiện làm việc cụ thể mà tiến hành kiểm nghiệm thêm về độ cứng và độ ổn định dao động của trục.

Tính trục về độ bền mỏi bao gồm hai bước cơ bản sau:

- Tính thiết kế trục theo độ bền tĩnh nhằm xác định gần đúng đường kính trục, chiều dài các đoạn trục và thiết kế sơ bộ kết cấu trục;

- Tính kiểm nghiệm trục về độ bền mỏi bằng cách xác định hệ số an toàn ở một số tiết diện nguy hiểm của trục, kết hợp với việc chọn ổ lăn hoặc thiết kế ổ trượt và kiểm nghiệm độ bền dập của then hoặc then hoa để quyết định lần cuối kết cấu trục, đáp ứng ba yêu cầu cơ bản trong thiết kế trục, đó là độ bền, lắp ghép và công nghệ.

Các công thức tính lực từ bánh răng, bánh vít, bánh đai, đĩa xích và nối trục di động tác dụng lên trục cho trong bảng 3.1.

Các công thức chủ yếu dùng để tính thiết kế và tính kiểm nghiệm trục cho trong bảng 3.2.

Bảng 3.1: Các công thức tính lực từ các chi tiết quay tác dụng lên trục

Tên gọi	*Công thức tính*	*Nọ công thức*
Lực từ bánh răng trụ chủ động	$F_{xki} = \frac{r_{kt}}{\lvert r_{ki}\rvert} cq_k cb_{ki} F_{tki}$	(3.1)
	$F_{yki} = -\frac{r_{ki}}{\lvert r_{ki}\rvert} F_{tki} tg\alpha_{tw} / \cos\beta$	(3.2)
	$F_{zki} = cq_k hr_{ki} cb_{ki} F_{tki} tg\beta$	(3.3)
	trong đó $F_{tki} = 2T_k / d_{Wki}$; $r_{ki} = d_{Wki}/2$, $r_{ki} < 0$ khi điiểm đặt lực nằm phía trên trục 0z và $r_{ki} > 0$ khi ngược lại.	
Lực từ bánh răng côn chủ động	$F_{xki} = \frac{r_{ki}}{\lvert r_{ki}\rvert} cq_k cb_{ki} F_{tki}$	(3.4)
	$F_{yki} = -\frac{r_{ki} F_{tki}}{\lvert r_{ki}\rvert \cos\beta_m} (tg\alpha_n \cos\delta_{ki} - cq_k hr_{ki} cb_{ki} \sin\beta_m \lvert\sin\delta_{ki}\rvert)$	(3.5)
	$F_{zki} = \frac{F_{tki}}{\cos\beta_m} (tg\alpha_n \sin\delta_{ki} + \frac{\sin\delta_{ki}}{\lvert\sin\delta_{ki}\rvert} cq_k hr_{ki} cb_{ki} \sin\beta_m \cos\delta_{ki})$	(3.6)
	trong đó $F_{tki} = 2T_k / d_{mki}$; $r_{ki} = d_{mki}/2$; $\delta_{ki} < 0$ khi đỉnh côn nằm bên phải đáy côn ; $\delta_{ki} > 0$ khi đỉnh côn nằm bên trái đáy côn .	
Lực từ trục vít	$F_{xki} = \frac{r_{ki}}{\lvert r_{ki}\rvert} cq_k cb_{ki} F_{aki} tg(\gamma \pm \varphi')$	(3.7)
	$F_{yki} = -\frac{r_{ki}}{\lvert r_{ki}\rvert} \frac{F_{aki} \cos\varphi'}{\cos(\gamma \pm \varphi')} tg\alpha \cos\gamma$	(3.8)
	$F_{zki} = cq_k hr_{ki} cb_{ki} F_{aki}$	(3.9)
	trong đó $F_{aki} = 2T_k \eta u / d_{k+1,i}$; φ' - góc ma sát thay thế ; $d_{k+1,i}$- đường kính vòng chia bánh vít i lắp trên trục k +1	
Lực từ khớp nối	$F_{xki} = (0{,}2...0{,}3) F_{tki}$ trong đó $F_{tki} = 2T_k / D_k$ với D_k - đường kính vòng tròn qua tâm các chốt (của nối trục vòng đàn hồi – xem bảng 16-10a [2])	(3.10)

Bảng 3.2: Các công thức dùng trong thiết kế trục

Tên thông số	*Công thức tính*	*Nº công thức*
Đường kính trục sơ bộ	$d = \sqrt[3]{T_k / (0,2[\tau])}$ mm, với $[\tau]$ =15...30 MPa	(3.11)
Đường kính trục gần đúng	$d_{kj} = \sqrt[3]{M_{Ekj} / W_{kj}}$ mm *với trục tiết diện tròn* $W_{kj} = 0,1d_{kj}^3$; *với trục khoét lỗ hoặc có rãnh then – xem bảng 10-6[2]*	(3.12)
Mômen uốn tại trục k, tiết diên j	$M_{kj} = \sqrt{M_{ykj}^2 + M_{xkj}^2}$ Nmm	
Mômen tương đương tại trục k, tiết diên j	$M_{Ekj} = \sqrt{M_{kj}^2 + 0,75T_{kj}^2}$ Nmm	(3.13)
Hệ số an toàn chung tại trục k, tiết diện j	$s_{kj} = \dfrac{s_{\sigma kj} s_{\tau kj}}{\sqrt{s_{\sigma kj}^2 + s_{\tau kj}^2}} \geq [s]$ *trong đó* [s] = 2 ÷ 3 – *hệ số an toàn cho phép*	(3.14)
Hệ số an toàn chỉ xét riêng ứng suất pháp tại trục k, tiết diện j	$s_{\sigma kj} = \dfrac{\sigma_{-1}}{K_{\sigma dkj}\sigma_{akj} + \psi_{\sigma k}\sigma_{mkj}}$	(3.15)
Hệ số an toàn chỉ xét riêng ứng suất tiếp tại trục k, tiết diện j	$s_{\tau kj} = \dfrac{\tau_{-1}}{K_{\tau dkj}\tau_{akj} + \psi_{\tau k}\tau_{mkj}}$ *trong đó* $\psi_{\sigma k}, \psi_{\tau k}$ – xem bảng 10-7 [2] ; σ_{akj}, τ_{akj} -biên độ ứng suất uốnvà ứng suất xoắn σ_{mkj}, τ_{mkj} – giá trị trung bình của ứng suất uốn và ứng suất xoắn ; $K_{\sigma dkj} = (K_{\sigma kj} / \varepsilon_{\sigma kj} + K_{xkj} - 1) / K_{ykj}$ $K_{\tau dkj} = (K_{\tau kj} / \varepsilon_{\tau kj} + K_{xkj} - 1) / K_{ykj}$ *với* K_{xkj} , K_{ykj} – *xem bảng 10-8 và 10-9 [2].*	(3.16)
Hệ số kích thước	$\varepsilon_{\sigma k}, \varepsilon_{\tau k}$ – bảng 10-10 [2] ;	
Hệ số tập trung ứng suất khi uốn và khi xoắn	$K_{\sigma kj}, K_{\tau kj}$ – *bảng 10-12 [2] đối với trục có rãnh then hoặc 10-13 [2] đối với góc lượn và chân ren trục vít.* *Đối với bề mặt trục lắp có độ dôi tra trực tiếp tỉ số* $K_{\sigma kj} / \varepsilon_{\sigma kj}$ *và* $K_{\tau kj} / \varepsilon_{\tau kj}$ *trong bảng 10-11 [2]*	

3.1.2. Những vấn đề cần lưu ý khi xây dựng phần mềm thiết kế trục

Dữ liệu thiết kế trục thường là công suất P (kW), số vòng quay n (vg/ph), chiều quay (cùng chiều kim đồng hồ hoặc ngược lại) và số chiều quay, số lực từ các chi tiết quay tác dụng lên trục. Dữ liệu có thể được nhập từ bàn phím nhưng tốt nhất là nên gọi từ *File dữ liệu* đã được nạp sẵn trên đĩa, như thế sau mỗi lần sửa chữa sai sót hoặc gỡ rối, không phải nạp lại dữ liệu ban đầu.

Thiết kế trục được tiến hành theo các bước sau đây:

1. Chọn vật liệu chế tạo trục:

Thông thường hay dùng thép 45 tôi cải thiện để chế tạo trục. Trường hợp trục chịu ứng suất không lớn có thể dùng thép CT5 không nhiệt luyện còn khi trục làm việc trong các máy quan trọng và chịu ứng suất lớn hơn có thể dùng thép hợp kim như 40Cr, 40CrNi, v.v... tôi cải thiện hoặc tôi bề mặt bằng dòng điện tần số cao. Với các trục quay nhanh lắp ổ trượt, trục được chế tạo bằng thép 20, 20Cr,... thấm than rồi tôi để nâng cao độ chịu mòn của ngõng trục. Sau khi chọn vật liệu, tra giới hạn bền σ_b ở bảng 10.5 [2].

2. Xác định sơ bộ đường kính trục theo (3.12) với chú ý là nếu dùng khớp nối để nối với trục động cơ điện thì để dễ lắp, chênh lệch giữa hai đường kính của các trục được nối không nên quá lớn, nên chọn $d = (0,8 \div 1,2)d_{dc}$, trong đó d_{dc} là đường kính trục động cơ điện.

3. Xác định khoảng cách giữa các gối đỡ và điểm đặt lực. Muốn vậy trước hết hãy dùng bảng 10.2 [2] để chọn sơ bộ chiều rộng ổ lăn, tính chiều dài mayơ chi tiết quay theo các công thức sau:

Chiều dài mayơ bánh răng trụ, bánh đai, đĩa xích	$l_m = (1,2 \div 1,5)\, d$
Chiều dài mayơ bánh vít	$l_m = (1,2 \div 1,8)\, d$
Chiều dài mayơ bánh răng côn	$l_m = (1,2 \div 1,4)\, d$
Chiều dài mayơ nửa khớp nối	$l_m = (1,2 \div 1,8)\, d$

Sau đó tuỳ theo sơ đồ bố trí trục trong thiết bị để xác định khoảng cách giữa các gối đỡ và điểm đặt lực. Chẳng hạn, nếu là trục của hộp giảm tốc bánh răng trụ một cấp thì sử dụng hình 10.6 [2], hộp giảm tốc bánh răng trụ hai cấp - hình 10.7 [2], hộp giảm tốc phân đôi - hình 10.8 [2], hộp giảm tốc đồng trục - hình 10.9 [2], hộp giảm tốc côn trụ - hình 10.10 [2], hộp giảm tốc trục vít - hình 10.11 [2] kết hợp với bảng10.4 [2].

4. Xác định tải trọng từ các chi tiết quay tác dụng lên trục: lực từ bánh răng trụ – theo (3.1) đến (3.3); từ bánh răng côn - theo (3.4) đến

(3.6); từ trục vít - theo (3.7) đến (3.9); từ nối trục di động - theo (3.10), từ bánh đai - theo (2.98) và từ đĩa xích - theo (2.117).

Khi sử dụng các công thức từ (3.1) đến (3.9) để viết các chương trình con xác định trị số và chiều của các lực từ các chi tiết quay tác dụng lên trục cũng như để tính phản lực và vẽ biểu đồ mômen trong các mặt phẳng nằm ngang và thẳng đứng, cần lưu ý một số điểm sau đây:

a) Nên chọn thống nhất một hệ tọa độ 0xyz với quy ước về chiều dương cho các trục: ở đây chọn chiều hướng ra đối với trục x, hướng xuống đối với trục y và hướng từ trái qua phải đối với trục z làm chiều dương.

b) Vì lực hướng tâm luôn hướng vào tâm trục nên trong các công thức tính F_r cần nhân thêm tỉ số của bán kính r_{ki} và trị số tuyệt đối của nó $(-r_{ki}/|r_{ki}|)$ với r_{ki} lấy dấu – khi điểm đặt lực nằm phía trên trục z và lấy dấu + khi điểm đặt lực nằm phía dưới trục z. Vì lực vòng phụ thuộc chiều quay và bánh răng là chủ động hay bị động nên cần nhân biểu thức tính lực vòng với các biến cq_k và cb_{ki} ; trong khi đó ngoài hai biến này, lực dọc trục còn phụ thuộc vào hướng răng phải hay trái, vì vậy còn cần nhân thêm biến hr_{ki} vào biểu thức tính lực dọc trục. Như vậy để máy tính có thể xác định được chiều của các lực từ bánh răng, trục vít, bánh vít tại vị trí i của trục k, cần sử dụng các biến với quy ước dấu sau đây:

cb_{ki} = 1 khi chi tiết quay là chủ động, cb_{ki} = -1 khi chi tiết quay là bị động;

hr_{ki} = 1 khi răng hướng phải, hr_{ki} = -1 khi răng hướng trái;

cq_k = 1 khi trục quay cùng chiều kim đồng hồ (nhìn từ mút trục bên trái), ngược lại: cq_k = -1.

c) Khi xác định chiều của các lực từ bánh răng côn tác dụng lên trục cần chú ý:

- Dấu của cb_{ki} , hr_{ki} và cq_k vẫn chọn như đối với bánh răng trụ nhưng với chú ý là khi xác định chiều quay cần nhìn từ phía đỉnh côn.

- Với bánh côn răng nghiêng và răng cung tròn, trong khi chiều của lực hướng tâm chỉ phụ thuộc vào chiều quay, hướng răng và bánh răng là chủ động hay bị động thì lực dọc trục không những phụ thuộc vào các yếu tố đó mà còn phụ thuộc vào vị trí đỉnh côn. Vì thế cần nhân thêm tỉ số $\sin\delta_{ki}/|\sin\delta_{ki}|$ vào số hạng thứ hai của công thức tính F_{zki} và quy ước dấu xác định vị trí đỉnh côn: $\delta_{ki} < 0$ nếu đỉnh côn nằm bên phải và $\delta_{ki} > 0$ nếu đỉnh côn nằm bên trái đáy côn, đồng thời dùng trị tuyệt đối cho $\sin\delta_{ki}$ ở biểu thức tính lực hướng tâm [xem công thức (3.6) và (3.5)].

Như vậy nhờ dùng thêm biến δ_{ki} cùng các biến cb_{ki} , hr_{ki} và cq_k, công thức (3.5) và (3.6) có thể dùng chung cho mọi trường hợp tính toán. Tuy nhiên khi thiết kế cần chú ý sao cho lực dọc trục hướng về đáy côn của bánh nhỏ, cụ thể là nếu bánh nhỏ là chủ động thì nên chọn hr_{ki} và cq_k cùng dấu, như thế sẽ không tạo ra lực hướng tâm lớn hơn và mômen uốn lớn hơn (do F_{yki} và F_{zki} gây nên mômen uốn cùng chiều) tác dụng lên trục và ổ của bánh côn nhỏ. Trái lại nếu bánh nhỏ là bị động (bộ truyền tăng tốc) thì nên chọn hr_{ki} và cq_k khác dấu.

- Các công thức từ (3.4) đến (3.6) dùng chung cho các loại răng nhưng với răng thẳng do $\beta_m = 0$ nên số hạng thứ hai trong hai biểu thức tính F_{yki} và F_{zki} sẽ bằng không. Khi đó chiều của lực dọc trục chỉ còn phụ thuộc vào vị trí đỉnh côn thông qua biến δ_{ki}.

d) Khi sử dụng các công thức từ (3.1) đến (3.9) để xác định lực ăn khớp cho một hộp truyền động từ các chi tiết quay ở vị trí i_k trên trục k tác dụng lên trục, nên tiến hành tính toán cho bánh chủ động với $cb_{ki} = 1$, còn lực từ bánh bị động ở vị trí i_{k+1} tác dụng lên trục k+1 được suy từ kết quả tính lực đối với bánh chủ động. Cụ thể là nếu bỏ qua ảnh hưởng của lực ma sát thì:

- Các lực trên bánh răng trụ bị động bằng và ngược chiều từng đôi với các lực trên bánh trụ chủ động.

- Lực vòng trên hai bánh côn bằng nhau về trị số và ngược chiều nhưng lực hướng tâm trên bánh côn bị động bằng về trị số và ngược với chiều của lực dọc trục trên bánh côn chủ động, lực dọc trục trên bánh côn bị động bằng về trị số và ngược với chiều của lực hướng tâm của bánh côn chủ động.

- Lực hướng tâm trên bánh vít và trục vít bằng về trị số và ngược chiều nhau nhưng lực vòng trên bánh vít bằng về trị số và ngược với chiều của lực dọc trục trên trục vít, lực dọc trục trên bánh vít bằng về trị số và ngược với chiều của lực vòng trên trục vít.

e) Với bộ truyền đai và bộ truyền xích, trị số của lực F_{rki} từ bánh đai hoặc đĩa xích tác dụng lên trục được tính theo (2.98) hoặc (2.117), chiều của các lực này hướng từ tâm bánh đai (hoặc đĩa xích) đang xét đến tâm bánh kia. Tuy nhiên khi đường nối tâm các bánh đai (hoặc đĩa xích) chủ động và bị động tạo với phương nằm ngang một góc α thì cần phân F_{rki} thành hai thành phần thẳng đứng và nằm ngang sau đây để tính toán với chú ý là góc α được tính từ chiều dương của các trục:

$$F_{yki} = F_{rki} \sin\alpha$$
$$F_{xki} = F_{rki} \cos\alpha$$

g) Riêng lực từ khớp nối di động tác dụng lên trục có thể có chiều bất kỳ phụ thuộc vào sai số ngẫu nhiên khi lắp ghép nối trục [1]. Tuy nhiên trong tính toán cần phải chọn sao cho chiều của lực này làm tăng ứng suất và biến dạng do các lực từ chi tiết máy quay khác gây nên trên các tiết diện của trục. Chẳng hạn, trường hợp bánh răng được lắp lên trục tại vị trí bất kỳ trong khoảng giữa hai ổ thì lực F_{xk} từ khớp nối tác dụng lên trục được chọn ngược với chiều của lực vòng từ bánh răng này tác dụng lên trục.

5. Vẽ sơ đồ trục, đặt các lực tác dụng lên trục, xác định phản lực và mômen uốn trong các mặt phẳng z0y, z0x và vẽ biểu đồ mômen uốn và mômen xoắn.

Để máy tính thực hiện được bước tính này cần lưu ý những điểm sau:

a) Trước khi vẽ sơ đồ trục, chi tiết quay và các lực cần khởi tạo đồ họa;

b) Để loại trừ các thành phần không phù hợp trong phương trình tính toán mômen cần soạn thảo các *Function* riêng. Chẳng hạn để tính mômen uốn tại một tiết diện của trục cần tính tổng mômen của các lực đặt trước tiết diện đang xét và loại trừ mômen của các lực đặt sau tiết diện đó. Muốn vậy hãy sử dụng hàm "uốn" sau đây:

```
Function uon ;
begin
  If lc[j] - lk[i] >= 0 then uon:=1 else uon:= 0 ;
end ;
```

Cũng thế, để tính momen xoắn ta sử dụng hàm "xoắn" sau đây:

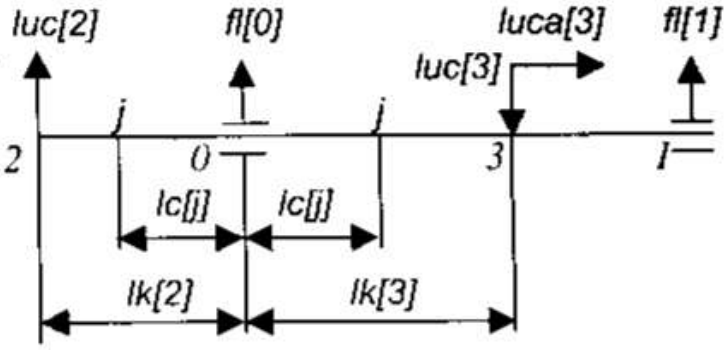
```
Function xoan ;
begin
  If lc[j] - lk[i] = 0 then xoan:=1 else xoan:= 0 ;
end ;
```

trong đó lc[j] - khoảng cách từ gốc đến tiết diện j đang xét; lk[i] - khoảng cách từ gốc đến điểm đặt lực i (xem hình 3.1).

Khi đó có thể tính mômen uốn theo chương trình con dưới đây với:

sodc – số điểm chia trên chiều dài trục;
soluc – số lực từ các chi tiết quay tác dụng lên trục (được đánh số theo thứ tự từ 2 trở đi);
mu0[i], mul[j] – lần lượt là giá trị của mômen uốn khi không tính và có tính đến giá trị của mômen uốn do lực dọc trục *luca[j]* sinh ra;
matc : array[0..5] of real;
matd : array[0..n] of real;

```
Procedure momenuon (luc,luca,fl: matc; var mu0, mua, mul, mu, mut:matd);
var tb, t0 : real;
begin
for j:=0 to sodc do
begin
  i: =0 ; t0: =0 ; tb: =0 ;
  while i<=soluc do
  begin
    if i<=1 then
    begin
     luc[i]:=fl[i] ; luca[i]:=0
    end;
    t0:= t0+luc[i]*(lc[j]
        - lk[i])*uon + luca[i]*r[i]*uon;
    tb:= tb+luca[i]*r[i]*xoan;
  end
  mu0[j]: = t0 ; mua[j]: = tb;
  mul[j]: = mu0[j] - mua[j];
  if abs(mu0[j])> mul[j] then  mu[j]: = mu0[j]) else mu[j]: = mul[j]);
 end;
end;
```



Hình 3.1: Sơ đồ tính mômen uốn

c) Trường hợp trục được thiết kế để quay 2 chiều (số chiều quay bằng 2), chiều của lực dọc trục từ bánh răng trụ, trục vít, bánh vít cũng như chiều của lực dọc trục và lực hướng tâm từ bánh răng côn răng nghiêng và răng cung tròn tác dụng lên trục sẽ thay đổi, do đó làm thay đổi giá trị của phản lực và biểu đồ mômen uốn. Vì vậy trong trường hợp này cần tiến hành xác định phản lực và vẽ biểu đồ mômen uốn ứng với hai chiều quay ngược nhau rồi sử dụng giá trị M_{kj} lớn hơn để xác định đường kính trục cũng như lấy giá trị lớn hơn của phản lực ứng với hai chiều quay ngược nhau trên mỗi ổ để làm căn cứ chọn ổ lăn.

d) Để vẽ biểu đồ mômen uốn và mômen xoắn hãy dùng biến phụ kèm theo biến điều khiển j tương tự như khi vẽ các đường cong thay đổi bất kỳ. Chẳng hạn, muốn vẽ biểu đồ mômen uốn hoặc mômen xoắn kể cả trường hợp có bước nhảy của mômen ứng với tiết diện thứ j trên trục k, có thể sử dụng chương trình con dưới đây với các ký hiệu sau:

h3 – hệ số đưa vào để biểu diễn các đồ thị mômen uốn đứng, mômen xoắn theo trục y (ứng với s = 1) và mômen uốn ngang theo đường phân giác của góc vuông (ứng với s = 2) ;

cl, cm – lần lượt là hệ số tỉ lệ khi biểu diễn khoảng cách và mômen;

xo – tọa độ ban đầu của x;

yo – tung độ của trục hoành;

matl : array[0..4,0..n] of real;

ts, tsa, tsl – lần lượt là trị số của mômen uốn do lực ngang *luc[k,i]*, lực dọc *luca[k,i]* và do tổng của hai lực này sinh ra.

```
Procedure vedothi(mau,s,yo: integer; ts, tsa, tsl: matl);
var  h3,cm : real; xx, rx, ry, xl, yl, lch: array[0.. 4, 0.. 24] of integer;
begin
  h3:=sqrt(s)*cm/s;
  for j:=0 to sodc[k] do
  begin
    xx[k,j]:=round(xo+cl*lch[k,j]);
    rx[k,j]:=round(xx[k,j]-(s-1)*h3*ts[k,j]);
    xl[k,j]:=round(xx[k,j]-(s-1)*h3*tsl[k,j]);
    ry[k,j]:=round(yo+h3*ts[k,j]);
    yl[k,j]:=round(yo+h3*tsl[k,j]);
  end;
  setcolor(7);
  line(g,yo,g+200,yo);
  setcolor(10);
  for j:=0 to sodc[k] do line(xx[k,j],yo,rx[k,j],ry[k,j]);
  setcolor(mau);
  for j:=0 to sodc[k]-1 do if tsa[k,j+1]=0 then
  line(rx[k,j],ry[k,j],rx[k,j+1],ry[k,j+1]) else
  line(rx[k,j],ry[k,j],xl[k,j+1],yl[k,j+1]);
  for j:=0 to sodc[k] do if tsa[k,j]<>0 then
  line(xl[k,j],yl[k,j],rx[k,j],ry[k,j]);
end;
```

e) Sau khi kết thúc phần đồ họa và muốn tiếp tục tính toán cần chuyển từ chế độ màn hình đồ họa về chế độ văn bản nhờ lệnh Textmode.

6. Tính đường kính các đoạn trục theo (3.13), trên cơ sở đó thiết kế sơ bộ kết cấu trục thoả mãn ba yêu cầu cơ bản: độ bền, lắp ghép và công nghệ.

Khi thiết kế kết cấu trục cần dựa vào đặc điểm kết cấu của các chi tiết quay, cụ thể là:

- Trục vít thường được chế tạo liền trục và do yêu cầu nâng cao độ cứng thường có đường kính khá lớn so với đường kính trục, do đó cần chú ý đến các tiết diện chuyển tiếp từ chân ren trục vít đến trục, không nên có sự chênh lệch nhiều về hai đường kính này.

- Các bánh răng lớn, bánh vít, bánh đai, đĩa xích thường được chế tạo rời rồi lắp lên trục, tuy nhiên bánh răng trụ nhỏ hoặc bánh răng côn nhỏ chỉ nên chế tạo rời rồi lắp lên trục nếu khoảng cách x từ rãnh then đến đường kính đáy răng bánh răng đủ lớn, trái lại nếu $x \le 2,5.m$ đối với bánh răng trụ và $x \le 1,6.m_{tc}$ đối với bánh răng côn thì nên chế tạo bánh răng

liền trục.

- Trong các trường hợp chế tạo bánh răng liền trục cần lưu ý chọn vật liệu trục và bánh răng như nhau.

- Chọn hệ lắp và kiểu lắp thích hợp để lắp các chi tiết quay lên trục: thường lắp các chi tiết quay lên trục theo hệ lỗ với cấp chính xác của lỗ thấp hơn một bậc so với cấp chính xác của trục (thí dụ H7/ k6); kiểu lắp của trục phụ thuộc vào độ lớn của tải trọng và yêu cầu đảm bảo độ đồng tâm giữa các chi tiết quay và trục.

7. Tính toán trục về độ bền mỏi bằng cách sử dụng các công thức (3.14), (3.15) và (3.16) để kiểm nghiệm hệ số an toàn cho một số tiết diện nguy hiểm của trục, trong đó biên độ và giá trị trung bình của ứng suất uốn và ứng suất xoắn được xác định như sau:

$\sigma_{akj} = M_{kj} / W_{kj}$; $\sigma_m = 0$ khi trục quay (1 hoặc 2 chiều);

$\tau_{akj} = \tau_{mkj} = T_{kj} / W_{0kj}$ khi trục quay một chiều;

$\tau_{akj} = T_{kj} /(2.W_{0kj})$; $\tau_{mkj} = 0$ khi trục quay hai chiều;

trong đó, với trục tiết diện tròn $W_{kj} = 0,1\, d_{kj}^3$, $W_{0kj} = 0,2\, d_{kj}^3$. Với tiết diện có rãnh then, W_{kj} và W_{0kj} tính theo các công thức trong bảng 10.6 [2].

Tiết diện nguy hiểm được hiểu là những tiết diện chịu mômen uốn và mômen xoắn lớn, tại đó có những yếu tố gây tập trung ứng suất làm giảm độ bền mỏi của trục. Chọn bao nhiêu tiết diện để kiểm nghiệm hệ số an toàn tuỳ thuộc vào loại trục (trục vào, trục ra hay trục trung gian,...) và số lực ứng với số chi tiết quay lắp trên trục. Do đó chương trình con tính hệ số an toàn cần được soạn thảo sao cho có khả năng dừng hoặc tiếp tục theo ý muốn.

Khi thực hiện bước tính này có thể xảy ra các tình huống sau:

a) Hệ số an toàn tại tiết diện đang xét nhỏ hơn trị số cho phép, tức là $s_j < [s]$, với $[s] = 1,5 \div 2,5$. Trường hợp này có thể xử lý theo các hướng sau đây:

- Tìm các biện pháp giảm tập trung ứng suất như dùng dao phay đĩa thay cho dao phay ngón để gia công rãnh then, tăng bán kính góc lượn hoặc dùng các biện pháp tăng bền bề mặt trục.

- Tăng đường kính trục và lặp lại bước tính này cho đến khi thoả mãn điều kiện $s_j \geq [s_j]$. Tuy nhiên khi thay đổi đường kính của tiết diện đang xét cần quan tâm đến các tiết diện bên cạnh nhằm đảm bảo các yêu cầu lắp ghép và công nghệ cũng như đảm bảo khe hở cần thiết giữa các đoạn trục của trục k đang xét với các chi tiết quay lắp trên trục k-1.

b) Hệ số an toàn tại tiết diện j nào đó lớn hơn nhiều so với trị số cho phép, lúc này cần xem xét tương quan giữa tiết diện đang xét với các tiết diện khác. Nếu vì lý do lắp ghép hoặc yêu cầu công nghệ mà đường kính trục của tiết diện đang xét phải lớn thì cần chấp nhận mà không xử lý, trái lại có thể giảm bớt đường kính trục này và lặp lại phép tính cho đến khi nhận được hệ số an toàn có thể chấp nhận được.

8. Kiểm nghiệm độ bền dập của then hoặc răng then hoa theo công thức

$$\sigma_d = \frac{2T_{kj}}{d_{kj} A z \varphi} \le [\sigma_d] \qquad (3.17)$$

trong đó

z – số răng then hoa, đối với then z =1;

$\varphi = 0,7 \div 0,8$ - hệ số phân bố không đều tải trọng cho các răng then hoa, đối với then $\varphi = 1$;

A – diện tích chiếu, đối với then A = 0,5 hl, với h – chiều cao của then, tra theo đường kính tiết diện trục lắp then (bảng 9.1a [2] đối với then bằng và bảng 9.2 [2] đối với then bán nguyệt);

l – chiều dài then, thường lấy $l = (0,8 \div 0,9)\, l_m$;

$[\sigma_d]$ - ứng suất dập cho phép, MPa, với mayơ bằng thép lấy ứng suất dập cho phép $[\sigma_d]$=190; 100 và 50 MPa, ứng với tải trọng tĩnh, va đập nhẹ và va đập nặng.

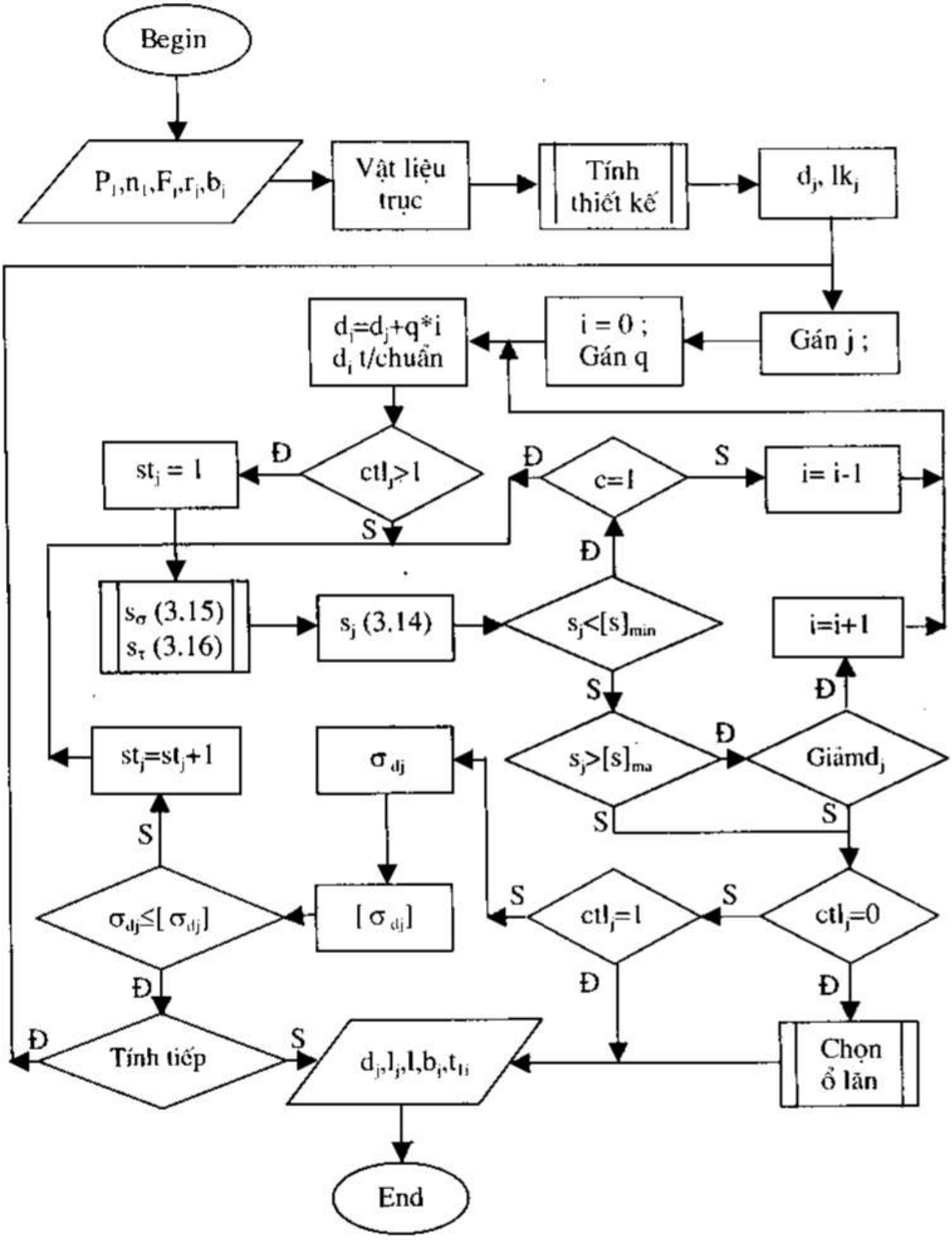
9. Kết hợp với tính toán ổ trượt hoặc ổ lăn để quyết định lần cuối kết cấu trục, khi cần thiết thì kiểm nghiệm trục về quá tải, về độ cứng và dao động.

Trên hình 3.2 là lưu đồ thuật giải tính trục về độ bền mỏi (kiểm nghiệm trục về hệ số an toàn sau khi đã tiến hành xong bước tính thiết kế trục), ở đó sử dụng một số biến sau đây:

lk_j - khoảng cách từ gốc 0 đến các tiết diện j ;

ctl[j] – tiết diện trục j được lắp với chi tiết khác nhau. Vì các chi tiết lắp liên quan đến tiết diện trục (có lắp then hay không?) và việc chọn chiều dài may ơ của các chi tiết lắp nên cần quy ước ký hiệu ctl[j], chẳng hạn như:

ctl[j] = 0 : ổ lăn; ctl[j] = 1 : trục liền bánh răng hoặc trục vít;
ctl[j] = 2 : bánh răng trụ; ctl[j] = 3 : bánh răng côn;
ctl[j] = 4 : bánh vít; ctl[j] = 5 : bánh đai;
ctl[j] = 6 : đĩa xích; ctl[j] = 7 : khớp nối .



Hình 3.2 : Lưu đồ thuật giải tính trục về độ bền

st[j] – số then được dùng để lắp chi tiết quay lên tiết diện j của trục;
$[s]_{min}$ – hệ số an toàn cho phép nhỏ nhất, $[s]_{min} = 1,5 \div 2$;
$[s]_{max}$ – hệ số an toàn cho phép lớn nhất, $[s]_{max} = 2,5 \div 3$.
q – được gán giá trị tương ứng với chênh lệch giữa hai giá trị liên tiếp của đường kính trục tiêu chuẩn đối với bề mặt trục lắp chi tiết quay:

10 ; 10,5 ; 11 ; 11,5 ; 12 ; 13 ; 14 ; 15 ; 16 ; 17 ; 18 ; 19 ; 20 ; 21 ; 22 ; 24 ; 25 ; 26 ; 28 ; 30 ; 32 ; 34 ; 36 ; 38 ; 40 ; 42 ; 45 ; 48 ; 50 ; 52 ; 55 ; 60 ; 63 ; 65 ; 70 ; 80 ; 85 ; 95 ; 100 ; 105 ; 110 ; 120 ; 125 ; 130 ; 140 ; 150 ; 160;

Với các giá trị d tiêu chuẩn ở trên, có thể chọn q = 1; 2 hoặc 3 đối với các đường kính trục từ 65 mm trở xuống. Riêng với đường kính ngõng trục từ 20 mm trở lên lắp với ổ lăn, q lấy bằng 5 để tương ứng với đường kính trong của ổ tiêu chuẩn.

c – cách, dùng để xử lý khi kiểm nghiệm hệ số an toàn không đảm bảo, tức là khi $s_j < [s]_{min}$, với c = 1: dùng các biện pháp kết cấu và công nghệ để tăng độ bền mỏi cho trục, còn c = 2: tăng đường kính trục tại tiết diện đang xét (tất nhiên khi dùng biện pháp này phải quan tâm đến đường kính trục của các tiết liền kề như đã trình bày ở trên).

Cần lưu ý rằng số lượng tiết diện trục cần kiểm nghiệm về hệ số an toàn tuỳ thuộc vào sơ đồ chịu lực và kết cấu cụ thể của trục được thiết kế, do đó trong lưu đồ thuật giải cần gán j = 0 ; 1;... tương ứng với số thứ tự của tiết diện cần kiểm nghiệm. Như vậy sau mỗi lần kiểm nghiệm xong một tiết diện của trục, nếu tiếp tục tính thì lại gán j cho tiết diện mới, nếu không thì chuyển qua bước thiết kế kết cấu trục, bao gồm việc xác định lần cuối đường kính của các tiết diện trục, chiều dài các đoạn trục l_j, chiều dài mayơ của các chi tiết quay lm_j và kết cấu rãnh then.

Ngoài ra đối với tiết diện trục lắp ổ lăn, trên lưu đồ thuật giải có phần tính toán chọn ổ lăn trước khi quyết định lần cuối kết cấu trục.

3.2. XÂY DỰNG CHƯƠNG TRÌNH TÍNH TOÁN CHỌN Ổ LĂN

Ổ trục dùng để đỡ trục, giữ cho trục có một vị trí xác định trong không gian, tiếp nhận tải trọng và truyền đến bệ máy. Tuỳ theo ma sát trên bề mặt tiếp xúc giữa ngõng trục và ổ trục, người ta phân ra: ổ trượt (ma sát ở đây là ma sát trượt) và ổ lăn (ma sát ở đây là ma sát lăn). Trong nền sản suất hiện đại, việc sản xuất ổ lăn được cơ khí hoá và tự động hoá rất cao. Tiêu chuẩn về kích thước ổ lăn và phương pháp chọn ổ của các nước về cơ bản là giống nhau, chỉ khác nhau về phạm vi kích thước ổ, số chủng loại và giá trị của các hệ số thực nghiệm nhằm xác định khả năng tải của ổ [4, 6, 8, 13]. Dưới đây trình bày những vấn đề liên quan đến việc xây dựng chương trình tính toán lựa chọn ổ lăn trên cơ sở sử dụng các

nguồn tài liệu và tiêu chuẩn ổ lăn của Nga.

3.2.1. Chỉ tiêu chọn ổ lăn và các công thức tính toán

Ổ lăn là một chi tiết tiêu chuẩn, khi thiết kế máy chỉ tiến hành tính toán nhằm chọn được loại ổ và kích thước ổ lăn thích hợp để dùng. Với ổ lăn làm việc lâu dài, chịu tải trọng và số vòng quay trung bình trở lên, tróc rỗ bề mặt các vòng ổ và con lăn là dạng hỏng chủ yếu, vì vậy chỉ tiêu tính toán lựa chọn ổ lăn làm việc trong điều kiện này là *khả năng tải động*. Với ổ lăn chịu tải trọng lớn, quay chậm hoặc không quay thì biến dạng dư lớp bề mặt là dạng hỏng nguy hiểm nhất, do đó *khả năng tải tĩnh* được dùng làm căn cứ để chọn ổ cho trường hợp này đồng thời còn dùng khả năng tải tĩnh để kiểm nghiệm độ bền tĩnh cho các ổ lăn đã được chọn theo khả năng tải động.

Dữ liệu tính toán ổ lăn bao gồm: tải trọng, số vòng quay, đường kính ngõng trục và điều kiện làm việc của ổ. Số vòng quay của ổ được dùng làm căn cứ để quyết định phương án tính toán lựa chọn ổ lăn:

- Với $n \geq 10$ vg/ph : chọn ổ theo khả năng tải động;
- Với $n < 1$ vg/ph : chọn ổ theo khả năng tải tĩnh;
- Với $1 \leq n < 10$ vg/ph lấy $n = 10$ vg/ph và tiến hành chọn ổ theo khả năng tải động.

Chọn ổ lăn theo khả năng tải động gồm các nội dung chính sau đây:

1. Chọn loại ổ lăn.
2. Chọn sơ bộ kích thước ổ.
3. Kiểm nghiệm ổ về khả năng tải động.
4. Kiểm nghiệm ổ về khả năng tải tĩnh.

Các công thức dùng trong tính toán ổ lăn ghi trong bảng 3.3.

3.2.2. Những vấn đề cần lưu ý khi xây dựng phần mềm chọn ổ lăn

1. Khi lập trình chọn ổ lăn cần sử dụng các ký hiệu sau đây theo TCVN 3776 – 83:

- Loại ổ : 0 – ổ bi đỡ một dãy; 1 – ổ bi đỡ lòng cầu hai dãy;
  2 – ổ đũa trụ ngắn đỡ; 3 – ổ đũa đỡ lòng cầu hai dãy;
  4 – ổ kim; 5 – ổ đũa trụ xoắn;
  6 – ổ bi đỡ – chặn; 7 – ổ đũa côn; 8 – ổ bi chặn;
- Cỡ ổ : 2;5 – cỡ nhẹ; 3;6 – cỡ trung; 4 – cỡ nặng;
  1;7 – cỡ đặc biệt nhẹ; 8;9 – cỡ siêu nhẹ;

Bảng 3.3: Các công thức dùng trong tính toán ổ lăn

Tên thông số	*Công thức tính, bảng*	*№ công thức*
Khả năng tải động tính toán	$C_t = QL^{1/m}$	(3.18)
Tuổi thọ (triệu vòng)	$C_t = Q_E L^{1/m}$	(3.19)
Tuổi thọ (giờ)	$L = 60.10^{-6} nL_h$ với L_h tính bằng giờ	(3.20)
Tải trọng quy ước đối với :		
Ổ bi đỡ, đỡ - chặn và đũa côn	$Q = (XVF_r + YF_a)K_d K_t$	(3.21)
Ổ chặn – đỡ	$Q = (XF_r + YF_a)K_d K_t$	(3.22)
Ổ chặn	$Q = F_a K_d K_t$	(3.23)
Ổ đũa trụ ngắn đỡ	$Q = VF_r K_d K_t$	(3.24)
Ổ đỡ – chặn kép	$Q = (0,5XVF_r + YF_a)K_d K_t$	(3.25)
Tải trọng tương đương	$Q_E = \sqrt[m]{\Sigma(Q_i^m L_i)/\Sigma L_i}$	(3.26)
Hệ số tải trọng hướng tâm	X – bảng 11.4 [2]	
Hệ số tải trọng dọc trục	Y – bảng 11.4 [2]	
Lực dọc trục do lực hướng tâm sinh ra trong:		
Ổ bi đỡ – chặn	$F_{sj} = eF_{rj}$ với j = 0; 1	(3.27)
Ổ đũa côn	$F_{sj} = 0,83eF_{rj}$	(3.28)
Hệ số thực nghiệm	e – bảng 11.4 [2]	
Hệ số đặc tính tải trọng	K_d – bảng 11.3 [2]	
Hệ số ảnh hưởng của nhiệt độ:		
Khi t trong ổ t = 105° ÷ 250°	$K_t = (108 + 0,4t)/150$	(3.29)
Khi t trong ổ t < 105°	$K_t = 1$	
Lực dọc trục tổng tác dụng vào ổ đỡ - chặn	$\Sigma F_{zj} = F_{sk} \pm F_{at}$ (khi j = 0 : k = 1; khi j = 1: k = 0)	(3.30)
Khả năng tải tĩnh đối với :		
Ổ đỡ và ổ đỡ – chặn	$Q_t = X_0 F_r + Y_0 F_a$ nhưng $Q_t \geq F_r$	(3.31)
Ổ chặn và chặn - đỡ	$Q_t = F_a - 2,3F_r tg\alpha$ với X_0 , Y_0 – bảng 11.5 [2]	(3.32)

Ngoài ra đối với ổ bi đỡ – chặn dùng thêm các con số ký hiệu kiểu ổ:

36 : ổ bi đỡ – chặn có góc tiếp xúc 12^0;
46 : ổ bi đỡ – chặn có góc tiếp xúc 26^0;
66 : ổ bi đỡ – chặn có góc tiếp xúc 36^0.

- Các yếu tố kích thước như đường kính trong d, đường kính ngoài D, chiều rộng ổ,... cũng như khả năng tải động C và khả năng tải tĩnh C_0 tương ứng với mỗi loại ổ, cỡ ổ và kiểu ổ cho trong các bảng ổ lăn (xem bảng P2.7 đến 2.14 [2]).

2. Tải trọng là một trong số dữ liệu đầu vào của chương trình tính toán ổ lăn. Lực hướng tâm tác dụng lên ổ chính là phản lực tại các ổ được xác định khi tính trục. Nó phụ thuộc vào mômen xoắn, các lực tập trung từ các chi tiết quay tác dụng lên trục và vào chiều quay của trục.

Cần lưu ý rằng khi dùng nối trục di động, các phản lực hướng tâm này đã được xác định theo chiều của lực tác dụng từ nối trục F_k làm tăng mômen uốn tại tiết diện nguy hiểm nhưng lại làm giảm trị số của phản lực tại ổ đặt gần nối trục. Vì vậy cần theo chiều ngược với chiều của F_k khi tính trục để xác định lại phản lực của ổ đặt gần nối trục, như thế sẽ nhận được trị số lớn hơn. Trái lại, giá trị lớn hơn của lực hướng tâm tại ổ thứ hai lại chính là giá trị của phản lực tại ổ đó khi tính trục. Nói cách khác cần xuất phát từ các chiều ngược nhau của F_k để tìm ra phản lực lớn hơn tại hai gối, phục vụ tính toán chọn ổ.

Lực dọc trục ngoài F_{at} là tổng của các lực dọc trục từ các chi tiết quay tác dụng lên trục. Các lực dọc trục này phụ thuộc vào chiều quay của trục (cùng chiều kim đồng hồ hoặc ngược lại), trong tính toán cũng cần được xác định theo chiều quay nào gây nên phản lực lớn hơn tại các ổ. Trị số của F_{at} được cho trong phần dữ liệu đầu vào phải kèm theo dấu (lấy dấu + nếu F_{at} hướng theo chiều dương từ trái qua phải của trục z, ngược lại: lấy dấu –), điều này là cần thiết vì nó liên quan đến việc xác định lực dọc trục F_{a0} và F_{a1} tác dụng vào ổ 0 và ổ 1 (xem công thức 3.30).

3. Cơ sở để chọn loại ổ lăn là đặc điểm làm việc của ổ, thể hiện ở khả năng tiếp nhận tải trọng lớn hay nhỏ, chịu được tải trọng hướng tâm F_r, tải trọng dọc trục F_a hay cả hai, độ cứng và giá thành ổ, khả năng quay nhanh cũng như yêu cầu đơn giản về kết cấu và chế tạo gối đỡ ổ lăn. Nhìn chung nhờ các con lăn và vòng ổ có diện tích tiếp xúc lớn hơn, ổ đũa chịu được tải trọng lớn hơn (do đó khi cùng kích thước, C và C_0 của ổ đũa lớn hơn), chịu va đập tốt hơn nhưng khả năng quay nhanh kém hơn.

Trong tính toán có thể dựa vào tỉ số của lực dọc trục và lực hướng tâm tác dụng vào ổ

$$g = F_a / F_r \qquad (3.33)$$

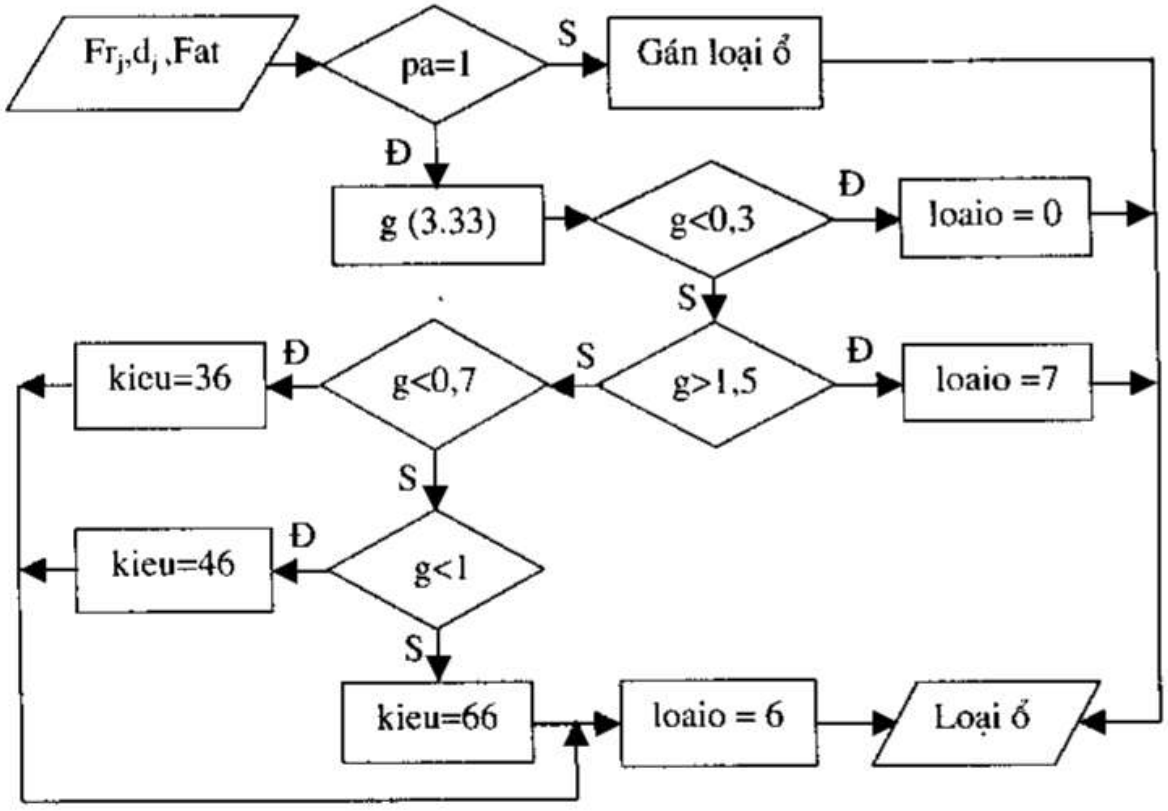
để chọn loại ổ:

a) Khi $F_a / F_r \leq 0,3$ và không yêu cầu nâng cao độ cứng và cố định chính xác vị trí ổ theo phương dọc trục thì ưu tiên dùng ổ bi đỡ một dãy để có kết cấu đơn giản nhất, giá thành rẻ nhất.

b) Khi $F_a / F_r > 0,3$ dùng ổ đỡ – chặn, cho tới khi $F_a / F_r < 1,5$ thì nên dùng ổ bi đỡ – chặn có góc tiếp xúc α lớn hơn khi F_a lớn hơn ($\alpha = 12^0$; 26^0 và 36^0). Khi $F_a / F_r > 1,5$ nên dùng ổ đũa côn.

c) Tuy nhiên ngay cả khi F_a / F_r nhỏ nhưng yêu cầu cao về độ cứng vẫn nên dùng ổ đũa côn vì ổ này đắt hơn không nhiều so với ổ bi đỡ một dãy nhưng có độ cứng cao hơn, lắp ghép và điều chỉnh ổ theo phương dọc trục thuận lợi hơn. Ngoài ra, việc chọn loại ổ cần được gắn với phương pháp cố định ổ, với chú ý rằng chỉ có *ổ cố định* mới tiếp nhận tải trọng dọc trục, mặt khác chỉ có thể dùng : ổ bi đỡ một dãy, ổ bi đỡ lòng cầu hai dãy và ổ đũa trụ ngắn đỡ làm *ổ tuỳ động*.

Như vậy khi soạn thảo chương trình con để chọn loại ổ lăn có thể sử dụng một trong hai phương pháp: dựa vào tỉ số F_a / F_r (phương án 1) và tự chọn (phương án 2). Lưu đồ thuật giải chọn loại ổ lăn thể hiện trên hình 3.3, trong đó đối với ổ bi đỡ – chặn còn chọn thêm kiểu ổ.



Hình 3.3: Lưu đồ thuật giải chọn loại ổ lăn

4. Trước khi tiến hành tính toán chọn kích thước ổ lăn còn phải quyết định phương án bố trí ổ trên hai gối 0 và 1. Có thể chọn một trong ba phương án sau đây:

pa =1 : dùng ổ cùng loại và cùng đường kính ngõng trục trên hai gối;

pa =2 : dùng ổ kép, tức là bố trí hai ổ trên một gối còn trên gối kia bố trí một ổ;

pa =3 : dùng ổ có đường kính ngõng trục khác nhau trên hai gối.

Thông thường nên dùng phương án 1 vì như thế chế tạo trục và lắp ghép ổ lên trục sẽ thuận lợi hơn, trừ trường hợp do yêu cầu kết cấu phải chế tạo hai đường kính ngõng trục khác nhau, khi đó dùng phương án 3. Phương án 2 được sử dụng khi cần cố định một đầu trục về cả hai phía, còn đầu kia cho phép trục di chuyển tự do về cả hai phía ứng với các trường hợp: trục dài, lực dọc trục lớn hoặc dãn nở vì nhiệt nhiều do nhiệt độ làm việc cao (thí dụ ổ của truc vít). Khi dùng phương án 1, kích thước ổ dùng cho cả hai gối được chọn theo gối chịu lực lớn hơn, còn ở phương án 2 và 3 sẽ tiến hành chọn kích thước ổ cho từng gối.

5. Chọn ổ lăn theo khả năng tải động theo các bước sau đây:

a) Chọn sơ bộ cỡ ổ và dựa vào đường kính ngõng trục để tra kích thước ổ, C và C_0 trong các bảng ổ lăn (bảng P2.7 đến P2.14 [2]).

b) Tra hệ số thực nghiệm e: Với mối loại ổ chỉ có một giá trị của e (bảng 11. 4 [2]), trừ ổ bi đỡ một dãy và ổ bi đỡ – chặn với góc tiếp xúc $\alpha = 12^0$, có một loạt giá trị của e, do đó cần xác định tỉ số iF_a / C_0 với i – số dãy bi và dựa vào iF_a / C_0 để tra hệ số e.

c) Xác định tải trọng dọc trục:

Đối với ổ bi đỡ một dãy, ổ bi đỡ lòng cầu hai dãy và ổ đũa đỡ lòng cầu hai dãy, tải trọng dọc trục là lực dọc ngoài tổng F_{at}. Một trong hai ổ sẽ chịu F_{at} khi chiều của lực này hướng vào ổ đó.

Đối với ổ đỡ – chặn tiến hành như sau:

– Tính F_{sj} theo (3.27) đối với ổ bi đỡ – chặn hoặc theo (3.28) đối với ổ đũa côn, với j = 1; 2.

– Từ công thức (3. 30):

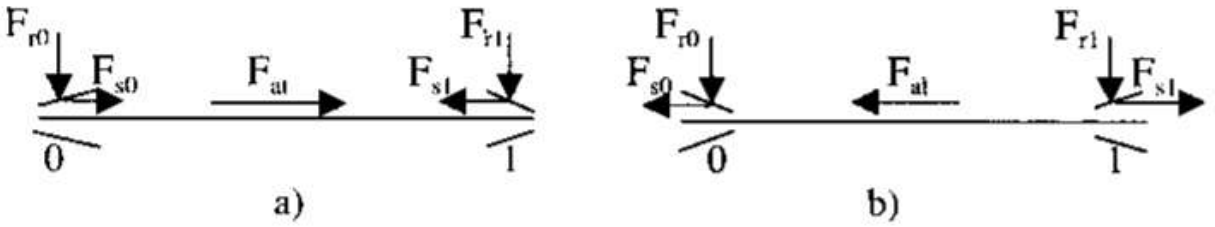
$$\Sigma F_{zj} = F_{sk} \pm F_{at}$$

có thể nhận được hai công thức xác định tổng lực dọc trục Fd_0 và Fd_1 tác dụng vào ổ 0 và ổ 1 với một dấu duy nhất dùng chung cho sơ đồ bố trí ổ theo kiểu chữ O (hình 3.4a) hoặc chữ X (hình 3.4b):

$$Fd_0 = F_{s1} + i.F_{at} \qquad (3.\ 34\ a)$$

$$Fd_i = F_{s0} - i.F_{at} \qquad (3.34\ b)$$

trong đó i = – 1 khi bố trí ổ theo sơ đồ 1 (chữ O - hình 3.4a) và i = 1 khi bố trí ổ theo sơ đồ 2 (chữ X - hình 3.4b). Chú ý rằng F_{at} ở hình 3.4a lấy dấu (+), còn F_{at} ở hình 3. 4b lấy dấu (–). Cách sử dụng biến i kèm theo dấu rõ ràng rất thuận lợi cho việc soạn thảo chương trình con xác định lực dọc trục tác dụng vào ổ đỡ – chặn trên hai gối đỡ.



Hình 3.4: Sơ đồ bố trí ổ đỡ – chặn

– So sánh Fd_i với F_{si} nếu $Fd_i \le F_{si}$ thì lấy $F_{ai} = F_{si}$, ngược lại, lấy $F_{ai} = Fd_i$.

– Xác định hệ số tải trọng hướng tâm X và hệ số tải trọng dọc trục Y: với ổ bi đỡ một dãy và tất cả các loại ổ đỡ – chặn một dãy khi $F_a / (VF_r) \le e$ thì X = 1 và Y = 0, khi $F_a / (VF_r) > e$, tra X và Y trong bảng 11.4 [2].

d) Tính tải trọng quy ước Q tác dụng vào từng ổ theo một trong các công thức từ (3.21) đến (3.25) tuỳ thuộc loại ổ và kết cấu gối đỡ ứng với tải trọng ngoài là tải trọng tĩnh. Trường hợp ổ chịu tải trọng động thì sau khi xác định được Q_0 và Q_1, chọn giá trị lớn hơn để tính tải trọng tương đương Q_E theo (3.26) và dùng nó để tính khả năng tải động tính toán của ổ lăn theo phương án 1 (hai ổ cùng loại và cùng đường kính ngõng trục). Với phương án 2 và 3 cần tính cả Q_{E0} và Q_{E1}.

e) Xác định khả năng tải động tính toán: khi ổ chịu tải trọng tĩnh, C_t được tính theo (3.18) với Q là giá trị lớn hơn trong hai giá trị Q_0 và Q_1, L là thời hạn sử dụng tính bằng triệu vòng quay, tính theo (3.20). Trường hợp ổ chịu tải trọng thay đổi, trước hết sử dụng công thức (3.26) để tính Q_E theo giá trị lớn hơn trong hai giá trị Q_0 và Q_1 ứng với tải trọng tĩnh, sau đó tính C_t theo công thức (3.19).

g) Kiểm nghiệm khả năng tải động của ổ lăn: với đường kính trong của ổ bằng đường kính ngõng trục và biết khả năng tải động C của ổ ứng với loại ổ, cỡ ổ và kiểu ổ (trong các bảng từ P2.7 đến P2.14 [2]), ổ đã chọn đảm bảo được khả năng tải động nếu đảm bảo điều kiện:

$$C_t \le C \qquad (3.35)$$

Trường hợp điều kiện (3.35) được thoả mãn, tiến hành tra kích thước ổ lăn gồm: đường kính ngoài D, chiều rộng ổ b, bán kính góc lượn, v.v... Trái lại nếu điều kiện (3. 35) không được thực hiện, tức là có thể rơi vào một trong hai khả năng sau:

- Với đường kính trong của ổ bằng đường kính ngõng trục mà $C_t > C$, lúc này có thể xử lý theo mấy hướng sau đây:

+ Chọn cỡ ổ lớn hơn khi đó C sẽ tăng lên;

+ Tăng đường kính ngõng trục (và tương ứng thay đổi đường kính các đoạn trục khác) nếu kết cấu cho phép (chẳng hạn không gây nên sự va chạm giữa bánh răng lắp trên trục trước với trục đang xét do đường kính tăng lên);

+ Dùng loại ổ khác, chẳng hạn dùng ổ đũa thay ổ bi nếu số vòng quay của trục không quá lớn;

+ Giảm thời hạn sử dụng ổ cho đến kỳ đại tu hoặc trung tu máy sẽ thay ổ, lúc này sẽ giảm được C_t.

- C_t nhỏ hơn nhiều so với C nhưng không thể xử lý bằng cách giảm đường kính ngõng trục vì nó được xác định từ độ bền mỏi của trục hoặc do yêu cầu thuận tiện trong lắp ghép. Biện pháp xử lý ở đây là dùng ổ cỡ nhẹ hơn hoặc dùng ổ bi thay ổ đũa.

6. Kiểm nghiệm khả năng tải tĩnh của ổ: ổ đã chọn theo khả năng tải động sẽ bảo đảm được khả năng tải tĩnh nếu thoả mãn điều kiện sau đây:

$$Q_t \leq C_0 \qquad (3.36)$$

trong đó Q_t được tính theo (3.31) đối với ổ đỡ và ổ đỡ – chặn hoặc theo (3.32) đối với ổ chặn và ổ chặn - đỡ.

3.3. XÂY DỰNG CHƯƠNG TRÌNH TÍNH TOÁN THIẾT KẾ Ổ TRƯỢT

3.3.1. Chỉ tiêu và các bước tính toán ổ trượt

Dữ liệu tính toán ổ trượt bao gồm: tải trọng (chính là phản lực F_r trên gối đỡ, được xác định khi tính trục), số vòng quay, đường kính ngõng trục và nhiệt độ dầu ở cửa vào (miệng lỗ cho dầu). Để tránh các dạng hỏng mòn, dính và mỏi rỗ lớp bề mặt lót ổ, thường tiến hành tính toán, lựa chọn các thông số đảm bảo cho ổ làm việc trong chế độ bôi trơn ma sát ướt. Tuy nhiên không phải lúc nào cũng tạo được điều kiện để ổ làm việc trong chế độ ma sát ướt, hơn nữa dù ổ làm việc ở chế độ bôi trơn ma sát ướt nhưng khi mở máy, dừng và đóng máy vẫn xảy ra ma sát nửa ướt, do đó còn phải tính ổ theo áp suất p và tích số pv.

Các công thức tính toán ổ trượt cho trong bảng 3.4.

Bảng 3.4. Các công thức tính toán thiết kế ổ trượt

Thông số	*Công thức tính*	*№ công thức*
Vận tốc vòng,m/s	$v = \pi dn/60000$	(3.37)
Độ hở tương đối	$\psi = 0{,}8.10^{-3}v^{0.25}$	(3.38)
Chiều dày lót ổ, mm	$\delta = (0{,}035 \div 0{,}05)d + 2{,}5$ mm	(3.39)
Áp suất, MPa	$p = F_r/(l.d) \le [p]$	(3.40)
Tích số pv, MPa. m/s	$pv = F_r n/(19100l) \le [pv]$	(3.41)
	trong đó	
	[p] và [pv] – tra bảng 12.1[2]	
Hệ số khả năng tải của ổ	$C_F = p\psi^2/(\mu\omega)$,	(3.42)
	trong đó $\omega = \pi n/30$	
	μ – độ nhớt động lực của dầu, cP, tra bảng 12.2[2]	
Chiều dày nhỏ nhất của lớp dầu bôi trơn	$h_{min} = 0{,}5S_{lb}(1-\chi)$	(3.43)
Hệ số an toàn	$s = h_{min}/(R_{z1} + R_{z2}) \ge [s] = 2{,}5$	(3.44)
Chênh lệch giữa nhiệt độ ra t_r và nhiệt độ vào t_v	$\Delta t = fF_t v/1000[(C\gamma Q + K_T \pi d)]$	(3.45)
	trong đó Q – lưu lượng dầu chảy qua ổ, được xác định từ tỉ số $Q/(\psi\omega ld^2)$ theo χ và l/d – hình 12.2[2]	
	C – nhiệt dung riêng của dầu, $C = 1{,}7 \div 2{,}1$ KJ/(kg °C)	
	γ – khối lượng riêng của dầu $\gamma = 850 \div 900$ kg/m^3	
	K_T – hệ số thoát nhiệt, $K_T = (0{,}04 \div 0{,}08)$ kW/(m^2 °C)	

Tính toán ổ trượt được tiến hành theo các bước sau đây:

1) Chọn vật liệu lót ổ, từ đó xác định được [p] và [pv] (bảng 11.2[2]).

2) Chọn các thông số của ổ, bao gồm:

- Tỉ số l/d = 0,5 ÷ 0,9. Biết d từ đó xác định được chiều dài lót ổ l;
- Chọn độ hở tương đối ψ theo công thức thực nghiệm (3.38);
- Chọn loại dầu (bảng12.2[2]), từ đó tra được độ nhớt động lực μ.

3) Kiểm nghiệm p và tích số pv.

4) Tính ổ trượt bôi trơn ma sát ướt:

- Tính hệ số khả năng tải của ổ C_F theo (3.42) rồi dựa vào C_F và l/d để tra ra độ lệch tâm tương đối χ theo bảng 12.3[2] ;
- Tính chiều dày nhỏ nhất của lớp dầu bôi trơn h_{min} theo (3.43);
- Chọn độ nhám bề mặt lót ổ và ngõng trục, suy ra chiều cao mấp mô trung bình của bề mặt ngõng trục và lót ổ R_{z1} và R_{z2} , từ đó xác định được hệ số an toàn đảm bảo cho ổ làm việc trong chế độ ma sát ướt.

5) Chọn phương pháp bôi trơn và tính kiểm nghiệm ổ trượt về nhiệt.

3.3.2. Những điểm cần lưu ý khi xây dựng phần mềm tính toán ổ trượt

1) Khác với ổ lăn, ở đó vòng trong của ổ được lắp với ngõng trục theo các kiểu lắp trung gian có độ dôi, lót ổ trong ổ trượt được lắp có độ hở với ngõng trục. Độ hở này có vai trò hết sức quan trọng trong việc hình thành chế độ bôi trơn ma sát ướt và đảm bảo khả năng tải của ổ trượt. Sau khi tính được độ hở tương đối ψ theo công thức thực nghiệm (3.38), sẽ tính được độ hở $S = \psi d$, dựa vào S chọn kiểu lắp (xem bảng P4.1 [2] và P4.2 [2]) sẽ tra được độ hở lớn nhất S_{max} và độ hở nhỏ nhất S_{min}, biết S_{max} và S_{min} sẽ xác định được độ hở trung bình S_{tb} và độ hở tương đối $\psi = S_{tb}/d$ và ψ được dùng làm căn cứ để tính khả năng tải của ổ trượt.

2) Khi tính ổ trượt, người ta tiến hành lựa chọn các thông số kết cấu (tỉ số l/d, độ hở tương đối ψ), thông số công nghệ (phương pháp gia công đạt độ nhám R_{Z1}, R_{Z2})(*) và chọn độ nhớt μ của dầu bôi trơn ổ, từ đó tiến

---

(*) Chẳng hạn, đối với ngõng trục, khi tiện tinh có thể đạt độ nhám $R_{Z1} = 6,3 \div 1,6\mu m$, khi mài đạt độ nhám $R_{Z1} = 3,2 \div 0,4$ µm, còn khi đánh bóng: $R_{Z1} = 6,3 \div 1,6$ µm; đối với lót ổ, khi chuốt hoặc doa có thể đạt độ nhám $R_{Z2} = 10 \div 1,6$ µm còn khi tiện trong có thể đạt $R_{Z2} = 6,3 \div 1,6$ µm.

hành kiểm nghiệm điều kiện bôi trơn ma sát ướt (3.44). Trường hợp điều kiện (3.44) được thoả mãn, có nghĩa là với các thông số đã chọn, ổ trượt làm việc trong chế độ bôi trơn ma sát ướt.

3) Tuy nhiên khi kiểm nghiệm điều kiện (3.44) có thể xảy ra hai khả năng:

a) $s < [s]$, tức là với các thông số đã chọn, không đảm bảo chế độ bôi trơn ma sát ướt cho ổ. Lúc này có thể sử dụng lênh lựa chọn *case* với các cách (c) sau đây:

- $c = 1$: chọn phương pháp gia công làm giảm độ nhám bề mặt;
- $c = 2$: chọn dầu có độ nhớt μ lớn hơn;
- $c = 3$: giảm độ hở tương đối ψ;
- $c = 4$: tăng tỉ số l/d,

với chú ý rằng các cách 2, 3 và 4 đều nhằm giảm độ lệch tâm tương đối χ, nhờ đó có thể tăng được h_{min} tức là tăng được s, còn hai phương án 3 và 4 phụ thuộc vào khả năng thiết bị và công nghệ của cơ sở chế tạo.

b) $s >> [s]$: trường hợp này cần áp dụng các biện pháp xử lý ngược với hướng xử lý ở trường hợp a).

Chương 4

XÂY DỰNG CHƯƠNG TRÌNH THIẾT KẾ HỆ DẪN ĐỘNG CƠ KHÍ

Trong các thiết bị và dây chuyền công nghệ sử dụng nhiều hệ dẫn động cơ khí, bao gồm các loại hộp giảm tốc khác nhau, truyền động đai, truyền động xích và khớp nối. Các hệ dẫn động cơ khí này được dùng để truyền cơ năng từ động cơ đến các bộ phận làm việc của máy và các thiết bị công nghiệp.

Dữ liệu thiết kế hệ dẫn động cơ khí thường là: sơ đồ của hệ dẫn động và sơ đồ tải trọng, công suất P_{lv} và số vòng quay n_{lv} của trục làm việc (hoặc lực kéo F và vận tốc vòng v của trục làm việc); đường kính (tang quay, thùng trộn,...) lắp trên trục làm việc hoặc bước xích p và số răng đĩa xích nhỏ z_1 của xích tải, đặc tính tải trọng, chế độ làm việc và thời hạn sử dụng.

4.1. NỘI DUNG TÍNH TOÁN THIẾT KẾ

Thiết kế hệ dẫn động cơ khí bao gồm các nội dung sau đây:

1. Tính toán động học, chọn động cơ điện và phân phối tỉ số truyền của hệ cho các cấp.
2. Tính toán thiết kế các bộ truyền.
3. Tính toán thiết kế trục và chọn ổ lăn hoặc thiết kế ổ trượt.
4. Tính toán các yếu tố của vỏ hộp giảm tốc và thiết kế hộp.
5. Tính toán thiết kế khung và bệ máy.
6. Thiết kế bản vẽ chung toàn hệ thống.
7. Lập thuyết minh, hướng dẫn sử dụng, chế độ vận hành và bảo dưỡng thiết bị.

Như vậy ngoài phần tính toán thiết kế các bộ truyền được bố trí ở bên trong hoặc ở ngoài hộp, thiết kế trục và chọn ổ đã trình bày ở các chương 2 và 3, ở đây còn phải tiến hành tính toán chọn động cơ điện và phân phối tỉ số truyền của hệ cho các cấp, tính toán các yếu tố của vỏ hộp giảm tốc, khung và bệ máy; phần còn lại liên quan đến vấn đề thiết kế tự động các bản vẽ kỹ thuật.

4.2. NHỮNG VẤN ĐỀ CẦN LƯU Ý KHI LẬP TRÌNH TÍNH TOÁN THIẾT KẾ HỆ DẪN ĐỘNG CƠ KHÍ

4.2.1. Chọn sơ đồ động

Như đã biết, để dẫn động một thiết bị hoặc một dây chuyền công nghệ có thể sử dụng nhiều phương án dẫn động, mỗi một phương án sẽ là tiền đề cho tính toán động học và phân phối tỉ số truyền của hệ cho các cấp, đồng thời liên quan chặt chẽ đến khuôn khổ, kích thước và kết cấu của hệ.Vì vậy, nếu trong dữ liệu thiết kế không cho sơ đồ của hệ dẫn động cần thiết kế thì việc đầu tiên cần tiến hành là chọn sơ đồ động, sử dụng các loại hộp giảm tốc khác nhau với số cấp truyền động khác nhau và sử dụng các phương án khác nhau để nối ghép các trục của hộp giảm tốc với động cơ điện và cơ cấu chấp hành.

Muốn chọn được một sơ đồ động thích hợp để dùng, một mặt cần dựa vào dữ liệu thiết kế (công suất, vận tốc,v.v...), mặt khác cần nắm vững đặc điểm làm việc của các loại truyền động (khả năng tải, tỉ số truyền, khả năng công nghệ, độ tin cậy làm việc,...) cũng như ưu nhược điểm và phạm vi sử dụng của các loại hộp giảm tốc.

4.2.2. Xác định tải trọng tác dụng vào các bộ phận máy và chi tiết máy

Như đã biết, tải trọng cho trước là lực kéo F trên băng tải, xích tải, v.v... Nó có thể là tải trọng tĩnh (không thay đổi) hoặc tải trọng thay đổi theo một sơ đồ tải xác định. Từ tải trọng này và vận tốc v sẽ xác định được công suất lớn nhất trong chu trình làm việc của hệ dẫn động:

$$P_{lv} = Fv/1000 \qquad kW \tag{4.1}$$

trong đó F tính bằng N và v tính bằng m/s.

Số vòng quay trên trục làm việc n_{lv} được xác định theo vận tốc làm việc của băng tải:

$$n_{lv} = 6.10^4\, v/(\pi d) \quad vg/ph \tag{4.2}$$

hoặc xích tải:

$$n_{lv} = 6.10^4\, v/(zp) \quad vg/ph \tag{4.3}$$

trong đó d - đường kính của tang quay, mm; p – bước xích tải, mm; z – số răng của đĩa xích tải chủ động.

Cùng với số vòng quay n_{lv}, công suất P_{lv} tính theo (4.1) được dùng làm căn cứ để tính toán động học cho hệ dẫn động cũng như để xác định công suất và mômen xoắn tác dụng lên các trục trong hệ. Còn đặc tính

thay đổi của tải trọng sẽ được tính đến khi tiến hành tính toán khả năng tải của các chi tiết máy (bánh răng, trục vít - bánh vít, ổ lăn) hoặc bộ phận máy (động cơ điện).

4.2.3. Chọn động cơ điện

Nhờ có nhiều ưu điểm như: kết cấu đơn giản, giá thành hạ, có thể mắc trực tiếp vào lưới điện ba pha, dễ bảo quản, làm việc tin cậy nên động cơ ba pha không đồng bộ kiểu rôto ngắn mạch thường được sử dụng để dẫn động các thiết bị cơ khí làm việc lâu dài. Khi chọn loại và kích thước động cơ này, cần biết công suất cần thiết và số vòng quay đồng bộ (số vòng quay khi không tải) của động cơ.

Công suất cần thiết trên trục động cơ điện được xác định theo công thức sau:

$$P_{ct} = \beta P_{lv} / \eta \qquad (4.4)$$

trong đó P_{lv} xác định theo (4.1),

$$\beta = \sqrt{\Sigma (P_i / P_1)^2 t_i / \Sigma t_i} \qquad (4.5)$$

$$\eta = \prod_{i=1}^{n} \eta_i \qquad (4.6)$$

với η_i – hiệu suất của các bộ truyền và các cặp ổ trong hệ dẫn động (bảng 2.3 [2]); P_i , t_i – công suất và thời gian làm việc ứng với chế độ i; P_1 – công suất lớn nhất trong các công suất tác dụng lâu dài trên trục máy công tác. Rõ ràng rằng với tải trọng tĩnh, $\beta = 1$.

Số vòng quay đồng bộ của động cơ được xác định khi động cơ chạy không tải theo công thức:

$$n_{db} = 60f/p \qquad (4.7)$$

trong đó f – tần số của dòng điện xoay chiều, 1/s ; p – số đôi cực.

Thường hay sử dụng f = 50 1/s , do đó số vòng quay đồng bộ của động cơ sẽ là 3000; 1500; 1000; 750 vg/ph; v.v... tương ứng với số đôi cực p = 1 ; 2 ; 3 ; 4 ; v.v... Rõ ràng rằng khi số vòng quay đồng bộ của động cơ càng tăng thì số đôi cực càng giảm, do đó khuôn khổ, kích thước, khối lượng và giá thành động cơ càng giảm, trong khi đó hệ số công suất (cosφ) và hiệu suất càng tăng. Vì vậy người sử dụng mong muốn chọn động cơ có số vòng quay cao.

Tuy nhiên nếu vận tốc làm việc thấp mà sử dụng động cơ với số vòng quay cao (số đôi cực ít) sẽ phải giảm tốc nhiều lần hơn, tức là phải sử dụng hệ dẫn động với tỉ số truyền lớn hơn, dẫn đến kích thước và giá

thành các bộ truyền tăng lên. Vì vậy trong thiết kế cần phối hợp hai yếu tố vừa nêu, đồng thời cần căn cứ vào sơ đồ của hệ dẫn động để chọn số vòng quay thích hợp cho động cơ.

Như vậy có thể xác định số vòng quay sơ bộ của động cơ theo công thức sau:

$$n_{sb} = n_{lv} \prod_{i=1}^{n} u_i$$

với u_i – tỉ số truyền của các bộ truyền trong hệ n cấp, chọn theo bảng 2.4 [2].

Từ số vòng quay sơ bộ của động cơ sẽ suy ra số đôi cực p của động cơ được sử dụng.

Sau khi chọn được loại động cơ (Việt – Hung: bảng P1.1[2], Điện cơ Hà Nội: bảng P1.2[2] hay động cơ của Nga: bảng P1.3 [2]), dựa vào công suất cần thiết và số đôi cực p được gán sẽ tra được nhãn động cơ từ các *File* dữ liệu trên đĩa với các thông số cơ bản gồm công suất, số vòng quay n_{dc}, hiệu suất và khối lượng động cơ. Đường kính trục động cơ và các kích thước lắp đặt của động cơ được tra trong các bảng P1.4[2]; P1.5[2]; P1.6[2] hoặc P1.7[2] tuỳ thuộc vào loại động cơ và phương pháp lắp đặt động cơ, phù hợp với hệ dẫn động.

4.2.4. Tính tỉ số truyền của hệ u_t và phân phối u_h cho các cấp

Từ số vòng quay động cơ n_{dc} và số vòng quay của trục làm việc n_{lv} xác định được tỉ số truyền của hệ

$$u_t = n_{dc} / n_{lv}$$

Từ u_t sau khi chọn tỉ số truyền của bộ truyền đai u_d hoặc tỉ số truyền của bộ truyền xích u_x (xem bảng 2.4 [2]) sẽ xác định được tỉ số truyền của hộp giảm tốc

$$u_h = u_t / u_n$$

trong đó u_n – tỉ số truyền của bộ truyền ngoài hộp ($u_n = u_d$ hoặc $u_n = u_x$). Chú ý rằng đường kính các bánh đai của bộ truyền đai là những giá trị tiêu chuẩn, do đó để tránh cho sai lệch của tỉ số truyền không quá lớn nên chọn u_d theo dãy số sau (tương ứng với dãy đường kính bánh đai tiêu chuẩn):

$$2 ; 2{,}24 ; 2{,}5 ; 2{,}8 ; 3{,}15 ; 3{,}56 ; 4 ; 5 ; 6$$

Tương tự như vậy, đối với bộ truyền trục vít, vì số mối ren trục vít chỉ bằng $1 \div 4$ do đó cũng không nên chọn tỉ số truyền của bộ truyền trục vít là số lẻ bất kỳ mà u_{lv} chỉ nên là số nguyên hoặc lẻ 0,5.

Biết u_h và dựa vào mục tiêu phân phối tỉ số truyền (theo khối lượng nhỏ nhất, theo yêu cầu bôi trơn bằng cách nhúng các bánh lớn trong hộp dầu hay theo đa mục tiêu,...[2,11]) tiến hành phân phối tỉ số truyền cho các cấp trong hộp. Cần lưu ý rằng, bên cạnh vật liệu và phương pháp nhiệt luyện dùng để chế tạo bánh răng, trục vít,... và điều kiện chịu tải, việc phân phối tỉ số truyền cho các cấp trong hộp giảm tốc có ảnh hưởng rất lớn đến kích thước và khối lượng của hộp. Vì vậy người thiết kế cần cân nhắc để quyết định chọn phương án phân phối tỉ số truyền nào trong các phương án (*fa*) sau đây khi phân phối tỉ số truyền của hộp cho các cấp:

fa = 1 : theo công thức thực nghiệm [2];
fa = 2 : theo các đồ thị tương ứng [2];
fa = 3 : theo khối lượng nhỏ nhất đối với hộp giảm tốc bánh răng trụ (3.12) [2];
fa = 4 : theo phương pháp đa mục tiêu – bảng 3.1 [2].

Ứng với mỗi phương án đã chọn cần gán *fa* và đưa vào chương trình tính toán các công thức và bảng tra tương ứng.

4.2.5. Xác định công suất, số vòng quay và mômen xoắn trên các trục

Chú ý rằng trong trường hợp tải trọng làm việc thay đổi, công suất cần thiết của động cơ được xác định không phải từ công suất làm việc P_{lv} mà từ công suất tương đương thông qua hệ số $\beta<1$. Vì vậy khi thiết kế không xuất phát từ công suất cần thiết của động cơ mà từ công suất làm việc P_{lv} để tính toán công suất trên các trục của hệ, phục vụ cho việc thiết kế các bộ truyền, tính trục và chọn ổ lăn (xem thí dụ xác định công suất trên các trục của sơ đồ hộp giảm tốc trên trang 49 [2]). Đặc tính thay đổi của tải trọng sẽ được tính đến khi thiết kế từng chi tiết máy cụ thể (xem phần tính hệ số tuổi thọ khi tính bánh răng, bánh vít và tính tải trọng tương đương khi chọn ổ lăn).

Từ số vòng quay của trục động cơ và tỉ số truyền của các cấp sẽ xác định được số vòng quay của các trục và từ công suất và số vòng quay trên trục i sẽ xác định được mômen xoắn trên trục i theo công thức sau:

$$T_i = 9{,}55.10^6 P_i / n_i \quad \text{(Nmm)} \qquad (4.8)$$

trong đó giá trị của mômen xoắn trên trục trung gian của hộp giảm tốc bánh răng phân đôi chỉ bằng một nửa giá trị của mômen xoắn tính từ công suất làm việc P_{lv}.

Cần chú ý sử dụng đúng giá trị của mômen xoắn trên các trục mang bộ truyền đang được thiết kế. Thông thường khi tính truyền động bánh răng, người ta sử dụng mômen xoắn trên trục mang bánh răng chủ động

(bánh nhỏ), còn khi tính truyền động trục vít - bánh vít lại sử dụng mômen xoắn trên trục bánh vít.

Mặt khác, khi tính bánh răng trong hộp giảm tốc bánh răng phân đôi, cần lưu ý sử dụng đúng giá trị của mômen xoắn tại các tiết diện trục mang bánh răng.

Ở hộp giảm tốc phân đôi cấp nhanh, công suất và mômen xoắn do hai cặp răng truyền đi, do đó khi thiết kế bộ truyền cấp nhanh chỉ được dùng một nửa công suất hoặc một nửa mômen xoắn trên đầu vào của trục để tính toán, trái lại ở cấp chậm, bộ truyền bánh răng không phân đôi phải chịu toàn bộ tải trọng làm việc.

Ở hộp giảm tốc phân đôi cấp chậm, mômen xoắn trên các tiết diện mang bánh răng chủ động của cấp chậm chính là mômen xoắn trên các tiết diện của trục trung gian (đã được chia đôi), trái lại ở cấp nhanh, bộ truyền không phân đôi phải chịu toàn bộ tải trọng làm việc.

Tất nhiên để tải trọng phân bố đều hơn trên chiều rộng vành răng của hai cặp bánh răng phân đôi, thường chọn hệ số chiều rộng răng nhỏ hơn, còn ở cặp cặp bánh răng không phân đôi, nhờ bánh răng được bố trí đối xứng đối với hai ổ nên có thể chọn hệ số chiều rộng răng lớn hơn [2].

4.2.6. Mở rộng các biến tổng thể

Nghiên cứu các loại sơ đồ hộp giảm tốc có thể thấy rằng chúng khác nhau chủ yếu ở việc bố trí các loại bộ truyền khác nhau trên trục vào, trục ra và các trục của hộp giảm tốc. Vì vậy để có thể sử dụng các phần mềm đã được soạn thảo cho từng loại bộ truyền, trục và ổ riêng biệt, đồng thời cũng nhằm tiết kiệm bộ nhớ của máy tính, cần mở rộng các biến được định nghĩa cho từng phần mềm thiết kế chi tiết máy thành các biến dùng chung cho các bộ truyền và chi tiết máy trong hệ. Có thể nêu ra một số biến sau đây làm thí dụ:

Công suất P, số vòng quay vq, tỉ số truyền u từ biến thực (real) được chuyển thành mảng một chiều cuả biến thực:

P, vq, u : Array[1..n] of real;

Đường kính chia d, đường kính đỉnh d_a và đường kính đáy răng d_f từ mảng một chiều của biến thực chuyển thành mảng hai chiều cuả biến thực trong hệ dẫn động:

d, d_a, d_f : Array[1..2,1..n] of real;

Các biến tổng thể này nên được đặt ở *Unit trabang* vì *Unit* này được sử dụng chung cho các phần mềm thiết kế chi tiết máy và bộ phận

máy tạo thành hệ và là *Unit* đầu tiên của chương trình thiết kế hệ dẫn động cơ khí.

4.2.7. Chuyển các program thành các Units và Units Overlays

Các chương trình thiết kế máy và hệ dẫn động cơ khí rất dài, chiếm bộ nhớ rất lớn, do đó cần được chia thành nhiều phần, viết *Program* cho từng phần sau đó chuyển chúng thành *Unit* hoặc *Unit Overlay* rồi gắn vào chương trình chính. Để có thể sử dụng các phần mềm thiết kế chi tiết máy đã được soạn thảo ở chương 2 và 3, trước khi chuyển các *Program* này thành *Unit* hoặc *Unit Overlay* phục vụ cho chương trình thiết kế máy và hệ dẫn động cơ khí cần lưu ý mở rộng các biến được sử dụng riêng thành các biến tổng thể. Bằng các biện pháp này có thể tiết kiệm được bộ nhớ khá nhiều khi thực hiện chương trình, đồng thời làm cho chương trình chính có cấu trúc gọn và rõ ràng hơn.

4.2.8. Thiết kế các bộ truyền

Sau khi đã gọi dữ liệu tính toán từ các *file* nạp sẵn trên đĩa và chọn được sơ đồ hệ dẫn động, thiết kế hệ dẫn động cơ khí bắt đầu bằng việc chọn động cơ và tính toán động học. Trình tự này không thay đổi đối với mọi sơ đồ hệ dẫn động. Tuy nhiên khi xuất phát từ các chỉ tiêu chủ yếu về khả năng làm việc của chi tiết máy để tiến hành thiết kế các bộ truyền, tính trục và chọn ổ lăn, cần quan tâm đến trình tự tính toán và mối quan hệ giữa các phần tính toán với nhau.

1. Thường thiết kế các bộ truyền trong và ngoài hộp theo trình tự từ cấp nhanh đến cấp chậm. Bộ truyền đai nhờ làm việc êm, thích hợp với vận tốc lớn và có thể đề phòng được quá tải cho thiết bị nên thường được đặt giữa động cơ và hộp giảm tốc, do đó sẽ được tính trước tiên, như thế sẽ thuận lợi trong việc chọn các đường kính tiêu chuẩn cho bánh đai. Bước tính tiếp theo được thực hiện đối với các bộ truyền trong hộp và cuối cùng là các bộ truyền bánh răng để hở hoặc bộ truyền xích. Tuy nhiên với các hệ dẫn động có bộ truyền trục vít thì do hiệu suất của bộ truyền này thay đổi (phụ thuộc vào các thông số đã chọn khi thiết kế) sẽ làm thay đổi mômen xoắn tác dụng vào các bộ truyền khác trong hộp, do đó bộ truyền trục vít cần được thiết kế trước.

2. Vì trong hộp giảm tốc đồng trục khoảng cách trục của cấp nhanh và cấp chậm bằng nhau, trong khi đó mômen xoắn trên cấp chậm lớn hơn ($T_c \approx uT_n$) nên thường thiết kế cấp chậm trước, lấy $a_n = a_c$ và để tận dụng khả năng tải không dùng hết của cấp nhanh thường chọn hệ số chiều rộng răng của cấp nhanh nhỏ hơn: $\psi_{ban} = 0,6\psi_{buc}$ và chọn $u_n > u_c$ (theo đồ thị trên hình 3.20 [2]) rồi tiến hành xác định các thông số ăn khớp và tính kiểm nghiệm. Để tận dụng khả năng tải không dùng hết của cấp nhanh,

đôi khi cũng dùng phương án chọn vật liệu chế tạo bánh răng có cơ tính kém hơn, tuy nhiên phương án này lại không đáp ứng được yêu cầu thống nhất hoá trong thiết kế (xem mục 4.2.10).

3. Ngoài tỉ số truyền, hệ số chiều rộng vành răng ψ_{ba} cũng ảnh hưởng đáng kể đến kích thước và khối lượng của hộp giảm tốc. Có thể tham khảo bảng 6.6[2] để chọn . Tuy nhiên để giảm bớt sự phân bố không đều tải trọng cho hai cặp răng đồng thời ăn khớp của hộp giảm tốc phân đôi, thường chọn hệ số chiều rộng răng ψ_{ba} của cấp phân đôi nhỏ hơn, chẳng hạn có thể chọn $\psi_{ba} = 0{,}2 \div 0{,}25$, trái lại ở cấp không phân đôi, nhờ cặp bánh răng này được bố trí đối xứng với hai ổ cho nên có thể chọn ψ_{ba} lớn hơn, chẳng hạn có thể chọn $\psi_{ba} = 0{,}45 \div 0{,}5$.

4. Để thu được kết quả tin cậy khi thiết kế các bộ truyền cần lưu ý:

a) Kiểm tra số liệu đầu vào bằng cách hiển thị trên màn hình các dữ liệu hoặc kết quả tính toán đã được thực hiện. Muốn vậy ở cuối mỗi một phần tính toán cần tạo *Procedure* ghi kết quả và nạp chúng vào các *File* tương ứng trên đĩa, đồng thời sử dụng các *Procedure* gọi dữ liệu hoặc gọi kết quả từ các *File* trên đĩa trước khi bắt đầu một phần tính toán mới, ở đây là phần thiết kế các bộ truyền.

b) Chuyển sai số giữa tỉ số truyền thực tế và tỉ số truyền tính toán của các bộ truyền được tính trước vào tỉ số truyền của bộ truyền ở cấp tính cuối cùng, theo công thức sau đây nhằm giảm bớt sai số của tỉ số truyền toàn hệ:

$$u_c = u_h / (u_{t1}\, u_{t2} \ldots)$$

trong đó u_{t1}, u_{t2},... là tỉ số truyền thực của các bộ truyền đã được thiết kế.

c) Xác định lại công suất, số vòng quay và mômen xoắn trên các trục do sự thay đổi tỉ số truyền thực tế của các bộ truyền và hiệu suất thực tế của bộ truyền trục vít.

4.2.9. Thiết kế trục và chọn ổ lăn

Phần mềm thiết kế trục và chọn ổ lăn rất dài, gồm phần tính toán (các thông số và kích thước cơ bản) và phần đồ họa (phân tích lực và vẽ biểu đồ mômen) được bố trí xen kẽ nhau. Khi xây dựng phần mềm này cần lưu ý những điểm sau:

1. Gọi dữ liệu tính toán từ các *File* kết quả đã được nạp vào đĩa sau phần chọn động cơ và tính toán các bộ truyền để kiểm tra trước khi tiến hành tính toán.

2. Dựa vào sơ đồ hệ dẫn động ứng với công suất và chiều quay xác định của trục làm việc để:

a) Xác định trị số và chiều của các thành phần lực từ các bộ truyền và khớp nối tác dụng lên các trục của hộp giảm tốc [xem các công thức từ (3.1) đến (3.9)]. Chú ý rằng khi tính lực từ cặp bánh răng phân đôi tác dụng lên trục chỉ được dùng một nửa công suất hoặc một nửa mômen xoắn để tính toán.

b) Xác định khoảng cách giữa các điểm đặt lực, khoảng cách giữa các gối đỡ và chiều dài các trục của hộp giảm tốc, trong đó cần lưu ý đảm bảo trị số tối thiểu cho khe hở giữa các chi tiết quay và chi tiết cố định.

3. Phần mềm thiết kế trục gồm hai bước: tính thiết kế và tính kiểm nghiệm trục về độ bền mỏi, cần được soạn thảo để có thể tiến hành tính toán cho tiết diện j bất kỳ của trục và cho trục k bất kỳ của hệ trong đó k thay đổi từ $k = 1$ đến $k = sotruc$, với *sotruc* là một biến kiểu *Byte* và được dùng để chỉ số trục của hệ.

4. Vì kết cấu của mỗi một trục trong hệ chỉ có thể được quyết định lần cuối sau khi đã tiến hành chọn và kiểm nghiệm độ bền dập của then hoặc then hoa cũng như chọn ổ lăn cho từng trục, cho nên khi thiết kế hệ dẫn động cơ khí, sau bước tính thiết kế trục, cần phối hợp bước tính kiểm nghiệm trục với phần chọn loại và kích thước ổ lăn cũng như phương pháp cố định ổ theo phương dọc trục (dùng ổ cố định hay ổ tuỳ động).

4.2.10. Thực hiện yêu cầu thống nhất hoá trong thiết kế

Sự thống nhất hoá trong thiết kế đem lại nhiều lợi ích như giảm thời gian và giá thành thiết kế và chế tạo sản phẩm, giảm giá thành sử dụng và sửa chữa, v.v... cho nên trong thiết kế cần lưu ý:

a) Sử dụng với số lượng tối đa các chi tiết máy có cùng quy cách kích thước như sử dụng môđun như nhau cho các bộ truyền bánh răng trong hệ (do đó giảm được danh mục dao cắt), thống nhất hoá bề mặt lắp ghép (sẽ giảm được danh mục calip kiểm tra), thống nhất hoá các chi tiết ghép có ren (sẽ giảm được bộ chìa vặn), v.v...Tất nhiên các thông số và kích thước được chọn như nhau này phải đảm bảo các chỉ tiêu chủ yếu về khả năng làm việc của chi tiết máy.

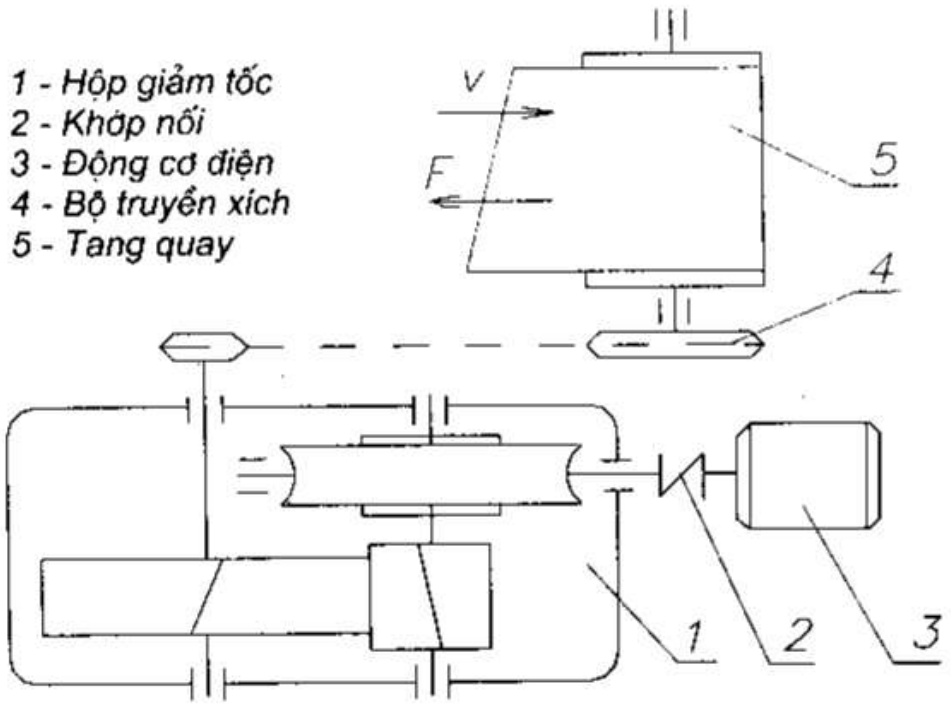
b) Sử dụng cùng một loại phôi, cùng một loại vật liệu và chế độ nhiệt luyện để chế tạo bánh răng nhỏ của các bộ truyền bánh răng trong hệ và tương tự như thế đối với bánh răng lớn của hệ. Cũng nên chọn cùng một loại vật liệu và chế độ nhiệt luyện để chế tạo tất cả các trục của hộp giảm tốc và của hệ dẫn động, đồng thời cũng nên cố gắng giảm bớt chủng loại ổ lăn dùng để đỡ các trục, v.v...

4.3. THÍ DỤ

Soạn thảo phần mềm thiết kế hệ dẫn động cơ khí và ứng dụng nó để tiến hành tính toán động học, thiết kế các bộ truyền, tính trục và chọn ổ lăn cho hệ dẫn động băng tải có sơ đồ vẽ trên hình 4.1.

Cho biết dữ liệu tính toán như sau:

Lực kéo băng tải	:	F = 18500 N
Vận tốc băng tải	:	v = 0,14 m/s
Đường kính tang quay	:	D = 300 mm
Thời hạn sử dụng	:	L_h = 14000 giờ
Số chiều quay	:	scq = 1
Sai lệch của tỉ số truyền	:	Δu = 4%
Đặc tính tải trọng	:	tải trọng tĩnh
Chế độ làm việc của thiết bị	:	làm việc 1 ca
Đặc điểm bố trí thiết bị	:	bộ truyền xích đặt nghiêng một góc 30° so với phương nằm ngang.



Hình 4.1: Sơ đồ hệ dẫn động băng tải sử dụng hộp giảm tốc trục vít – bánh răng

Thực hiện :

Vì đây là một phần mềm chiếm bộ nhớ rất lớn nên chương trình chính sẽ bao gồm nhiều *Unit* và *Unit Overlay*. Dưới đây chỉ trình bày thân của chương trình chính và một đơn thể (*Program botruyen*) của chương trình này, sau đó

tiến hành tính toán theo phần mềm đã được soạn thảo và cho kết quả tính toán theo dữ liệu cho trước cuả đầu bài.

4.3.1. Thân chương trình chính thiết kế hệ dẫn động cơ khí:

```
Program hecokhi;
{$M 4000,0,0}
uses crt, dos;
var i: byte;
{-------------------------------------}
begin
 clrscr;
 write(' i = ');
 readln(i);
 case i of
   1 : exec('c:\hedd\chondco. exe',");
   2 : exec('c:\hedd\botruyen. exe',");
   3 : exec('c:\hedd\trucolan. exe',");
 end;
end.
```

Chú thích

1. Ba phần chính của *Propram hecokhi* là các *Propram chondc*, *Propram botruyen* và *Propram trucolan*. Đó là các đơn thể được thiết kế, biên dịch và kiểm tra độc lập nhưng có liên hệ với nhau thông qua các *File* dữ liệu. Mỗi chương trình bao gồm nhiều *Unit* và *Unit Overlay*, vì vậy để có thể kết nối được các chương trình này, ở đây sử dụng thủ tục *Exec* của *Dos Unit* (xem chương 1, mục 1.2.11).

2. Đưa dẫn hướng đặc biệt ·{$M 4000,0,0} vào dòng đầu của chương trình để hạn chế sử dụng bộ nhớ.

3. Kiểm tra lỗi *Overlay* được thực hiện ở đầu mỗi chương trình (đơn thể) trước khi tiến hành các bước tính toán. Để rõ hơn, trong mục 4.3.2 sẽ cho thí dụ về cách viết một trong ba đơn thể của *Program hecokhi*.

4.3.2. Chương trình thiết kế các bộ truyền trong hệ dẫn động:

```
{$O+,F+}
Program botruyen;
uses Overlay, crt, tb, kht, Uscp, Ubr, Utv, Uxi, solict;
     {$O Uscp} {$O Ubr} {$O Utv} {$O Uxi}
type ten=string[10];
     matx=array[0.. 4] of real;
var  uh, utg, ndc, ux: real;
     he, cp, pp, sotruc: byte;
```

```
      texthe, textpp, tentep : ten;
      vg, vq, u, ut, T1  :  matx;
{---------------------------------------}
Procedure tstt;
begin
 case cp of
  1:begin
      vg[1]:=vq[1];vg[2]:=vg[1]/u[1];
    end;
  2:begin
      vg[1]:=vq[1]/ut[1];T1[2]:=T1[2]*ut[1]/u[1];
      vg[2]:=vg[1]/u[2];
    end;
  3:begin
      u[3]:=utg/(ut[1]*ut[2]);
      vg[1]:=vq[1]/(ut[1]*ut[2]);
      vg[2]:=vg[1]/u[3]; vq[3]:=vg[1];
      T1[3]:=T1[2]*ut[2];
    end;
 end;
end;
{-----------------------------------------------}
Procedure tinhbt;
var du: real;
begin
  tdtv(1);
  tdbr(2);
  vg[1]:=ndc/ut[1]/ut[2];
  ux:=utg/(ut[1]*ut[2]);
  tdxich;
  uh:=1;for cp:=1 to sotruc-1 do  uh:=uh*ut[cp];
  du:=(utg-(ux*uh))*100/utg;
end;
{-------------------------------------------------}
Begin
   clrscr;
   OvrInit('botruyen.OVR');
   if OvrResult<>OvrOk then
   begin
     writeln(' Loi Overlay ');
     Halt(1);
   end;
   write(' Tiep tuc tinh toan he = '); read(he);
   str(he,texthe);
```

```
    write(' So doi cuc o dong co dien = '); read(pp);
    str(pp,textpp);
    tentep:='he'+texthe+'.'+textpp+'1';
    goikq1(tentep);
    tstt;
    tinhbt;
    tentep:='he'+texthe+'.'+textpp+'2'; kq2(tentep);
    tentep:='he'+texthe+'.'+textpp+'3'; kq3(tentep);
    kqhgtbrtv;slhgtbrtv('hvt.'+texthe++textpp+'');
End.
```

Chú thích

1. Trong chương trình sử dụng các biến sau đây:

u, ut – tỉ số truyền tính toán và tỉ số tryền thực;

vq – số vòng quay các trục sau khi chọn động cơ và tính toán động học;

vg -- số vòng quay thực ứng với tỉ số truyền thực, được xác định sau khi đã thiết kế xong một bộ truyền;

he – hệ dẫn động gồm các bộ truyền trục vít, bộ truyền bánh răng và bộ truyền xích ứng với sơ đồ động cho trong đề thí dụ;

pp – số đôi cực của động cơ điện đã được chọn.

2. Chương trình *botruyen* gồm các *Unit*: Overlay, crt , tb, kht, solict và các *Unit Overlay*: Uscp, Ubr, Utv, Uxi.

3. Trước khi tính toán bộ truyền cần gọi kết quả tính toán chọn động cơ điện thông qua chương trình con *goikq1(tentep)* được đặt trong *Unit solict*, với *tentep* là một biến xâu ký tự, được dùng để biểu thị số sơ đồ của hệ dẫn động là he = 21, số đôi cực động cơ điện được chọn pp = 2 và là kết quả thứ nhất:

he21.21

Tương tự như vậy, sau khi tính xong các bộ truyền trong hệ, cũng cần ghi ra các *file* kết quả 2 và 3 với các chương trình con *kq2(tentep)* và *kq3(tentep)* để phục vụ chương trình *trucolan* ở phần dưới. Ngoài ra trong *program hecokhi* còn gọi tới hai chương trình con đặt trong *Unit solict*, đó là *kqhgtbrtv* và *slhgtbrtv('hvt.'+texthe++textpp+'')* ghi các thông số và kích thước của các bộ truyền trục vít và bánh răng phục vụ thiết kế tự động bản vẽ lắp hộp trục vít – bánh răng (xem chương 5).

4. Vì các bộ truyền trục vít, bộ truyền bánh răng và bộ truyền xích được đặt ở các cấp khác nhau ứng với mômen xoắn thực và số vòng quay thực khác nhau nên ở đây dùng thêm biến ut[cp] và biến vg[1] và vg[2] để biểu thị số vòng quay của trục 1 và trục 2 của mỗi cấp, dùng làm căn cứ tính toán mômen xoắn trên trục chủ động T1[cp] của mỗi bộ truyền nhằm tránh nhầm lẫn và điều chỉnh lại số liệu tính toán sau khi đã tính xong một bộ truyền.

5. Chương trình con *tinbt* gọi tới các *Procedure* sau đây:

- *Procedure tdtv(1)* đặt trong *Unit Overlay* Utv;
- *Procedure tdbr(2)* đặt trong *Unit Overlay* Ubr và
- *Procedure tdxich* đặt trong *Unit Overlay* Uxi.

4.3.3. Kết quả tính toán

4.3.3.1. Chọn động cơ và phân phối tỉ số truyền:

Công suất cần thiết của động cơ	P_{ct} = 3,77 kW
Số vòng quay của trục làm việc	n_{lv} = 8,91 vg/ph
Số đôi cực của động cơ điện	2p = 2

Số liệu động cơ điện

Loại động cơ	Việt –Hung
Nhãn động cơ	K132S4
Công suất động cơ	P_{dc} = 4 kW
Số vòng quay động cơ	n_{dc} = 1445 vg/ph
Hệ số quá tải	K_{ql} = 2
Khối lượng động cơ	G = 58 kg
Đường kính trục động cơ	d_{dc} = 32 mm

Phân phối tỉ số truyền

Tỉ số truyền của hệ	u_t = 162,13
Tỉ số truyền của bộ truyền xích	u_x = 2,5
Tỉ số truyền của hộp giảm tốc	u_h = 64,85
Tỉ số truyền của bộ truyền trục vít và bộ truyền bánh răng	
theo thực nghiệm	u_1 = 20 u_2 = 3,24
theo kết cấu và bôi trơn	u_1 = 17,5 u_2 = 3,712

4.3.3.2. Công suất, số vòng quay và mômen xoắn trên các trục

Thông số \ Trục	I	II	III
Công suất P (kW)	3,73	2,88	2,76
Tỉ số truyền u	17,5	3,712	2,5
Số vòng quay n (vg/ph)	1445	82,57	22, 24
Mômen xoắn T (Nmm)	24643	333 011	1186421

4.3.3.3. Thiết kế các bộ truyền

Tính toán lần lượt cho 3 bộ truyền: bộ truyền trục vít, bộ truyền bánh răng trụ răng thẳng và bộ truyền xích. Kết quả như sau:

1. Các kích thước và thông số cơ bản của bộ truyền trục vít

Thông số	*Ký hiệu*	*Phương án I*	*Phương án II*
Vật liệu trục vít		Thép 45	Thép 45
Nhiệt luyện trục vít		tôi đạt độ rắn 45 HRC	tôi đạt độ rắn 45 HRC
Vật liệu bánh vít		БрАЖН 10-4-4	БрАЖ 9-4
Cách đúc		đúc trong khuôn kim loại	đúc trong khuôn cát
Khoảng cách trục (mm)	a_{w1}	135	170
Môđun (mm)	m_1	6,3	8
Hệ số đường kính	q	8	8
Số mối ren trục vít và số răng bánh vít	z_{11}/z_{12}	2/35	2/35
Hệ số dịch chỉnh	x	- 0,071	- 0,25
Hiệu suất	η	0,81	0,82
Tỉ số truyền thực	u_{tt}	17,5	17,5
Ứng suất tiếp xúc:			
+ tính toán (MPa)	σ_H	196,3	138,9
+ cho phép (MPa)	$[\sigma_H]$	201,7	146
Chiều dài phần cắt ren trục vít (mm)	b_1	83	105
Chiều rộng bánh vít (mm)	b_2	47	60
Góc vít	γ	0,249	0,261
Đường kính chia (mm)	d_{11}/d_{12}	50,4/220,5	93/345
Đường kính đỉnh (mm)	d_{11}/d_{a12}	63/223,10	99,41/352,56
Đường kính đáy (mm)	d_{f11}/d_{a12}	35,28/204,48	85,94/339,08
Đường kính ngoài bánh vít (mm)	d_{aM2}	242,55	304

Cả hai phương án đều đảm bảo các chỉ tiêu về độ bền tiếp xúc và độ bền uốn. Tuy nhiên phương án I có nhiều ưu điểm như: kết cấu khuôn khổ bộ truyền nhỏ gọn hơn, tương quan về kết cấu giữa trục vít liền trục, ngõng trục và gối đỡ hợp lý hơn, tuy hiệu suất có kém hơn chút ít. Vì vậy ta quyết định chọn phương án I.

2. Thiết kế bộ truyền bánh răng trụ răng thẳng

a) Vật liệu và nhiệt luyện

Bánh răng nhỏ	Thép 45 ; tôi cải thiện đạt độ rắn 255 HB
Bánh răng lớn	Thép 45 ; tôi cải thiện đạt độ rắn 235 HB

b) Thực hiện các vòng lặp.

Thông số	*Ký hiệu*	Lần 1	Lần 2	Lần 3
Khoảng cách trục sơ bộ (mm)	a_{sb}	229,8		
Khoảng cách trục tính toán (mm)	a_{W1}	228	226,5	220
Môđun (mm)	m_2	3	3	3
Số răng bánh răng	z_{21}/z_{22}	33/122	33/122	31/115
Hệ số dịch chỉnh	x_{21}/x_{22}	0	0	0,07/0,27
Góc nghiêng của răng	β	0	0	0
Ứng suất tiếp xúc:				
+ tính toán (MPa)	σ_{H1}	416,9	434,4	449,5
+ cho phép (MPa)	$[\sigma_{H1}]$	463,4	463,4	463,4

c) Các kích thước và thông số cơ bản của bộ truyền bánh răng

Khoảng cách trục	(mm)	a_{W2}	220
Môđun	(mm)	m_2	3
Chiều rộng vành răng	(mm)	b_W	88
Góc nghiêng của răng	(°)	β	0
Số răng bánh răng		z_{21}/z_{22}	31/115
Hệ số dịch chỉnh		x_{21}/x_{22}	0,07/0,27
Đường kính chia	(mm)	d_{21}/d_{22}	93/345
Đường kính đỉnh	(mm)	d_{a21}/d_{a22}	99,41/352,56
Đường kính đáy	(mm)	d_{f21}/d_{a22}	85,94/339,05
Tỉ số truyền thực tế		u_{t2}	3,709

3. Các kích thước và thông số cơ bản của bộ truyền xích

Sau khi thiết kế xong bộ truyền trục vít và bộ truyền bánh răng với các tỉ số truyền thực tế là 17,5 và 3,709, tỉ số truyền của bộ truyền xích cần thiết kế sẽ là: 162,13/(17,5.3,709) = 2, 497. Bằng cách thực hiện các vòng lặp khi tính toán sẽ thu được các kết quả sau đây:

Thông số	Ký hiệu	Lần 1	Lần 2	Lần 3	Lần 4
Số vòng quay của đĩa xích nhỏ, vg/ph	n_1	22,3			
Hệ số vòng quay	k_n	2,248			
Số răng đĩa xích nhỏ	z_1	23	21	23	21
Số răng đĩa xích lớn	z_2	57	53	57	53
Hệ số răng	k_z	1,09	1,19	1,09	1,19
Hệ số sử dụng	k	1,3	1,3	1,3	1,3
Số dãy xích	sd	1	1	2	2
Công suất tính toán, kW	P_t	8,73	9,56	5,13	5,47
Công suất cho phép, kW	[P]	10,5	10,5	5,83	5,83
Sai lệch công suất	ΔP	16,87%	8,95%	11,93%	3,54%
Bước xích, mm	p	38,1	38,1	31,75	31,75
Đường kính đĩa xích lớn, mm	d_2	691,62	643,14	576,35	535,95
Sai lệch tỉ số truyền	Δu		- 0,95 %		- 0,95 %
Lực tác dụng lên trục, N	F_r		10654		12786
Đường kính đĩa xích nhỏ, mm	d_1		255,63		233,17
Khoảng cách trục, mm	a		1526		1272
Chiều dài xích, mm	l		4496		3747
Số má xích	x		118		118
Chiều rộng xích, mm	b		36,2		82
Tỉ số truyền thực	u_t		2,48		2,48

Như vậy trong 4 lần lặp, do chênh lệch giữa công suất tính toán và công suất cho phép ở lần 1 và lần 3 vượt quá 10% nên chúng bị loại. So sánh kết quả của hai lần lặp còn lại, ta chọn phương án 4 vì ở phương án này bước xích nhỏ hơn (do đó va đập của bộ truyền giảm), đường kính các đĩa xích, khoảng cách trục và chiều dài xích nhỏ hơn tuy rằng chiều rộng đĩa xích và lực tác dụng lên trục lớn hơn.

4.3.3.4. Thiết kế trục và chọn ổ lăn

Các kích thước và thông số cơ bản của các trục và ổ lăn

	Trục I
Chiều dài trục (mm)	l_1 = 372 mm
Đường kính các đoạn trục (mm)	Φ28 Φ30 Φ36 Φ30 Φ36 Φ30
Ren đầu trục (dùng đai ốc hãm và đệm cánh để cố định ổ bi đỡ)	M27
Chiều dài các đoạn trục tính từ mút trục vào, mm	87 137 142 18

Chiều dài các đoạn trục tính từ mút đầu kia, mm	30 35 102
Kích thước then tại tiết diện lắp nối truc b x h , mm	8 x7
Chiều dài then l_t và ứng suất dập σ_d tương ứng	40 mm, σ_d = 14,6 MPa<$[\sigma_d]$
Ứng suất dập cho phép	$[\sigma_d]$ =150MPa
Số lượng then	1
Hệ số an toàn tại:	
+ *Tiết diện lắp ổ lăn*	25,6
+ *Tiết diện chân ren trục vít*	21,4 > [s] = 2 ÷ 3

Sử dụng ổ đũa côn kép cho gối đỡ gần khớp nối và ổ bi đỡ một dãy cho gối đỡ bên kia, có đường kính trong d, đường kính ngoài D và chiều rộng ổ b như sau:

Ổ đũa côn kép 7606	d = 30 mm	D_n = 72 mm	b = 27 mm
Ổ bi đỡ một dãy 206	d = 30 mm	D_n = 62 mm	b = 16 mm

. Trục II

Chiều dài trục, mm	l_1 = 234 mm
Đường kính các đoạn trục, mm	Φ35 Φ40 Φ45 Φ40 Φ35
Chiều dài các đoạn trục tính từ mút trái, mm	41 127
Chiều dài các đoạn trục tính từ mút phải, mm	41 92
Kích thước then b x h, mm:	
+ *Tại tiết diện lắp bánh vít*	12 x 8
+ *Tại tiết diện lắp bánh răng*	12 x 8
Chiều dài then l_t và ứng suất dập σ_d tương ứng:	
+ *Tại tiết diện lắp bánh vít*	42 mm, σ_d = 79 MPa
+ *Tiết diện lắp bánh răng*	40 mm, σ_d = 131,7 MPa
Ứng suất dập cho phép	$[\sigma_d]$ = 150MPa
Số lượng then tại:	
+ *Tại tiết diện lắp bánh vít*	1
+ *Tại tiết diện lắp bánhrăng*	1

Hệ số an toàn tại:

+ *Tại tiết diện lắp bánh vít* 3,3
+ *Tại tiết diện lắp bánh răng* 2,17 > [s] = 2÷3

Dùng ổ bi đỡ một dãy 307 cho hai gối với các thông số d = 35 mm D_n = 80 mm b = 21mm

Trục III

Chiều dài trục, mm	l_1 = 345
Đường kính các đoạn trục, mm	Φ45 Φ50 Φ55 Φ60 Φ65 Φ60 Φ55
Chiều dài các đoạn trục tính từ mút trái, mm	39 99
Chiều dài các đoạn trục tính từ mút phải, mm (đầu trục ra)	82 111 151 238
Kích thước then b x h, mm:	
+ *Tại tiết diện lắp bánh răng*	12 x 8
+ *Tiết diện lắp đĩa xích*	12 x 8
Chiều dài then l_t và ứng suất dập σ_d tương ứng:	
+ *Tại tiết diện lắp bánh răng*	79 mm, σ_d = 142,3 MPa
+ *Tiết diện lắp đĩa xích*	74 mm, σ_d = 101,2 MPa
Ứng suất dập cho phép	$[\sigma_d]$ =150MPa
Số lượng then dùng để:	
+ *Lắp bánh răng*	1
+ *L ắp đĩa xích*	2
Hệ số an toàn tại các tiết diện:	
+ *Lắp bánh răng*	2,18
+ *L ắp ổ lăn*	1,98
+ *L ắp đĩa xích*	2,22 > [s] = 2÷3

Dùng ổ đũa côn 7211 cho hai gối với các thông số d = 55 mm D_n = 100 mm b = 21 mm

Chương 5

XÂY DỰNG CHƯƠNG TRÌNH THIẾT KẾ TỰ ĐỘNG BẢN VẼ

5.1. NỘI DUNG VÀ PHƯƠNG PHÁP SOẠN THẢO PHẦN MỀM THIẾT KẾ TỰ ĐỘNG BẢN VẼ

Các kết cấu cơ khí có thể được vẽ trực tiếp trong môi trường AutoCAD bằng đối thoại trực tiếp giữa người và máy. Cách vẽ này không những không kết nối được với các phần tính toán thiết kế tự động các chi tiết máy và bộ phận máy mà còn tốn rất nhiều thời gian để xây dựng dữ liệu thiết kế, thiết lập bản vẽ, ghi kích thước, v.v... ứng với một số liệu đầu vào xác định. Sử dụng phần mềm AutoCAD với các tập tin đáp bản (*Script File*) hoặc sử dụng phần mềm AutoLISP trong môi trường AutoCAD có thể tạo ra các phần mềm thiết kế tự động bản vẽ kỹ thuật và dễ dàng thay đổi kết cấu theo yêu cầu.

5.1.1. Nội dung của các chương trình thiết kế tự động bản vẽ

Chương trình thiết kế tự động bản vẽ bao gồm các nội dung sau đây:

1) Các tập tin văn bản (*File text*) dùng để gọi kết quả tính toán từ các *file dữ liệu* đã được nạp vào đĩa khi hoàn thành các chương trình tính toán thiết kế trước đó của chi tiết máy, bộ phận máy và hệ.

2) *Unit* tra bảng. Thông qua *Unit* này, các dữ liệu khác đã được lưu trữ trên đĩa cần cho thiết kế bản vẽ như: các giá trị tiêu chuẩn của các thông số và kích thước, sai lệch giới hạn của các kích thước, dung sai hình dáng và vị trí, các trị số nhám bề mặt v.v..., sẽ được gọi đến khi thực hiện chương trình vẽ tự động.

3) Các chương trình con với nội dung:

- Xác định thêm các thông số, kích thước cần thiết khác cũng như các tọa độ gốc và tọa độ tương đối phục vụ vẽ tự động.

- Tạo ra những đoạn tin sử dụng các lệnh vẽ của AutoCAD để vẽ kết cấu chi tiết máy, cụm chi tiết cũng như các kết cấu khác trên 3 hình chiếu chính và các hình cắt.

- Thiết lập cách ghi kích thước, dung sai lắp ghép, ghi nhám bề mặt cũng như ghi sai lệch hình dáng và vị trí (đối với bản vẽ chế tạo), đánh số chi tiết và ghi vào bảng kê (đối với bản vẽ lắp), ghi điều kiện kỹ thuật, vẽ và điền vào khung tên theo TCVN.

4) Thân chương trình chính thực chất là tập tin đáp bản hoặc tập tin AutoLISP chứa các chương trình con ở trên cùng các lệnh vẽ và các lệnh hiệu chỉnh khác.

5.1.2. Phương pháp xây dựng phần mềm thiết kế tự động bản vẽ

Để có thể thiết kế tự động bản vẽ, người thiết kế cần:

1) Tạo ra các tập tin đáp bản với tên *File* và phần mở rộng *.scr* hoặc tập tin AutoLISP với phần mở rộng *.lsp* .

2) Nắm vững các lệnh vẽ thông dụng (*Pline, Arc, Rectangle...*), các lệnh vẽ nhanh (*Copy, Mirror...*) cũng như các lệnh hiệu chỉnh *(Move, Rotate, Break,...)* để có thể chuyển nó thành các đoạn tin của tập tin đáp bản hoặc tập tin AutoLISP một cách chính xác, ít phải sửa chữa do đó tiết kiệm được thời gian lập trình.

3) Tìm hiểu kỹ kết cấu của thiết bị cần thể hiện, các yêu cầu về vẽ kỹ thuật, nắm vững và sử dụng tốt các tiêu chuẩn nhà nước và tiêu chuẩn ngành để tiến hành các bản vẽ nhằm thực hiện có kết quả các ý đồ thiết kế.

4) Tạo ra các *Unit* có mặt ở phần lớn các bản vẽ cơ khí để có thể gọi ra khi cần thiết như: khung tên, bảng thông số ăn khớp, điều kiện kỹ thuật, đánh số chi tiết và ghi vào bảng kê, ghi kích thước, dung sai lắp ghép, sai lệch hình dáng và vị trí cũng như các kết cấu điển hình như lỗ chi tiết quay với mặt cắt rãnh then, mặt cắt trục có rãnh then, lỗ tâm, v.v...

5) Xác định các tọa độ trên bản vẽ với số biến cần thiết tạo điều kiện thuận lợi cho việc thay đổi hoặc hiệu chỉnh kích thước sau này. Nên sử dụng các tọa độ sau:

- Tọa độ gốc xg, yg. Các tọa độ này sẽ được thay đổi cho phù hợp với kích thước và khuôn khổ của từng bản vẽ;

- Tọa độ cơ bản xác định các đường tâm trục xa[i], ya[i] trên 3 hình chiếu, được tính theo tọa độ gốc và được dùng làm cơ sở để xác định vị trí các điểm nối ghép các chi tiết hoặc cụm chi tiết với các kết cấu cơ bản như thành máy, vỏ hộp v.v...;

- Tọa độ các điểm nối ghép, được xác định theo tọa độ gốc hoặc tọa độ cơ bản (tọa độ các đường tâm trục) và phụ thuộc vào kích thước của các chi tiết tạo thành kết cấu máy.

6) Chia chương trình phần mềm đối với các bản vẽ của những máy móc và thiết bị phức tạp thành nhiều phần, viết *Program* cho từng phần sau đó chuyển thành *Unit* hoặc *Unit Overlay* rồi gắn vào chương trình chính, như thế sẽ tránh được tình trạng tràn bộ nhớ.

7) Với mỗi loại chi tiết, đặc biệt là chi tiết tiêu chuẩn, nên lập thành một *Procedure* riêng với các *tham biến và tham trị* để *truyền dữ liệu* phục vụ cho nhiều trường hợp vẽ khác nhau và sẽ gọi đến khi cần thiết. Chẳng hạn để vẽ ổ lăn có thể viết chương trình con với cách truyền dữ liệu như sau:

Procedure veolan(kieu, loaio: byte; xt, yt, dt, dn, bo: real);

trong đó

kieu =1 : vẽ theo phương đứng , kieu=2: vẽ theo phương ngang;

loaio : ký hiệu bằng con số theo TCVN (thí dụ 0: ổ bi đỡ một dãy);

xt, yt : tọa độ giao điểm của đường tâm trục và mặt mút vòng ngoài của ổ;

dt, dn, bo: đường kính trong, đường kính ngoài và chiều rộng ổ lăn;

8) Vẽ hoàn chỉnh từng kết cấu chi tiết (kể cả gạch mặt cắt), từng cụm chi tiết (kể cả việc kẻ đường tâm, chèn vào vị trí xác định trên bản vẽ và ghi kích thước lắp ghép) rồi mới chuyển sang cụm khác trên từng hình chiếu cần thể hiện.

9) Vẽ các chi tiết theo một thứ tự hợp lý: trước tiên nên vẽ các chi tiết được chọn làm cơ sở cho bản vẽ (chẳng hạn đối với bản vẽ lắp hộp truyền động đó là vỏ hộp ở hình chiếu bằng hoặc hình chiếu đứng), rồi đến các chi tiết ở bên trong hoặc các chi tiết riêng rẽ không ảnh hưởng đến các yếu tố khác, tiếp đến là các chi tiết bên ngoài, các hình cắt trích và cuối cùng là ghi kích thước khuôn khổ, kích thước liên kết, ghi điều kiện kỹ thuật, đánh số chi tiết và vẽ khung tên.

10) Chọn vị trí thuận lợi trên màn hình để vẽ chi tiết hoặc cụm chi tiết, sau đó dùng lệnh *Move* để chèn vào vị trí chính xác của kết cấu. Như thế sẽ tránh được nhầm lẫn, điều này đặc biệt cần thiết đối với các bản vẽ phức tạp.

5.2. TẬP TIN ĐÁP BẢN

Nhờ khả năng đọc được các lệnh từ một tập tin văn bản (*Text File*), AutoCAD cho phép thực hiện một chuỗi lệnh định trước thông qua nội dung đối thoại ứng với mỗi lệnh, tức là thông qua các chuỗi đối thoại

được gọi là tập tin đáp bản (*Script File*). Tập tin đáp bản được viết bằng ngôn ngữ Pascal chính là thân chương trình chính của các phần mềm thiết kế tự động bản vẽ.

5.2.1. Nội dung

Tập tin đáp bản bao gồm các nội dung sau:

1. Phần mở *File* với các *biến tệp* và *tên tệp* để ghi các số liệu vào *File*, chẳng hạn

```
Assign(fv,'c:\'+tenfile);
Rewrite(fv);
```

trong đó *fv* là một *File Text* ; *tenfile* là một tên riêng và phần mở rộng *.scr*

2. Các chuỗi đối thoại, nhằm thực hiện các nội dung sau đây:

a) Định khổ bản vẽ. Điều này phụ thuộc vào :

- Kích thước của kết cấu như chiều dài trục (khi vẽ chế tạo trục), đường kính (khi vẽ chế tạo bánh răng) hoặc kích thước khuôn khổ L*B*H (khi vẽ bản lắp);

- Khổ giấy mà người thiết kế định vẽ. Theo TCVN 2-74, khổ giấy được quy định như sau:

A0 : 1189 x 841 ; A1 : 841 x 594 ; A2 : 594 x 420 ;

A3 : 420 x 297 ; A4 : 297 x 210.

Chẳng hạn sử dụng các dòng lệnh

```
Writeln(fv,'Limits ','1089,841');
Writeln(fv,'Units ','2 2 3 2 ');
Writeln(fv,'Zoom a ');
```

để vẽ bản vẽ trên khổ giấy A0.

b) Tạo ra các lớp sẽ được sử dụng thống nhất cho toàn bộ bản vẽ với màu sắc, loại nét và độ dày của nét khác nhau nhờ lệnh *Layer*. Làm như vậy không những tránh được sự nhầm lẫn của các nét (thí dụ, nét liền vẽ trước chuyển thành thành nét chấm gạch khi thực hiện lệnh vẽ đường tâm) mà còn dễ hiệu chỉnh nét khi in bản vẽ. Thí dụ, tạo cho bản vẽ nét liền *LI* và nét chấm gạch *CG* bằng cách dùng câu lệnh sau:

```
Writeln(fv,'Layer N LI CG ');
```

c) Vẽ các cụm kết cấu, các chi tiết ghép nối ở hình chiếu chính và các hình cắt trích, ghi kích thước tự do, ghi kích thước lắp ghép với hệ lắp, kiểu lắp và cấp chính xác (đối với bản vẽ lắp), ghi sai lệch hình dáng

và vị trí, ghi nhám bề mặt (đối với bản vẽ chế tạo), ghi điều kiện kỹ thuật, v.v... Tất cả các lệnh này được bố trí một phần ở chương trình chính nhưng phần lớn được đặt trong các *Unit* và sẽ được gọi đến khi thực hiện bản vẽ. Bản vẽ càng phức tạp thì số lượng *Unit* sẽ càng lớn.

3. Lệnh đóng *File*, chẳng hạn

Close(fv);

5.2.2. Những vấn đề cần lưu ý khi soạn thảo tập tin đáp bản trong thiết kế tự động

1. Chuỗi đối thoại được bắt đầu bằng từ *Writeln* và dấu ngoặc đơn để đặt trong đó tên tệp, sau đó dùng 2 dấu nháy để đặt nội dung đáp lại các lệnh, đặt ký tự trắng, con số chỉ tọa độ, ký hiệu của tọa độ tương đối @ và dấu phẩy ngăn cách các biến, cuối cùng kết thúc bằng dấu chấm phẩy. Chẳng hạn để vẽ vòng tròn với tâm có tọa độ xt,yt và bán kính r, dùng câu lệnh sau:

Writeln(fv,'Circle ',xt:a:b,',',b:a:b,' ',r:a:b);

2. Trước khi kết thúc một chuỗi đối thoại, phía trước dấu ")" phải có dấu trắng, trừ các chuỗi đối thoại ứng với các lệnh *Circle*, *Arc* (khi dùng riêng biệt), *Rectangle, Break, Exi, Redraw*...

3. Sau các biến, thí dụ, x,y và r phải cho trường, ở đây là a và b.

4. Mỗi một ký tự trắng tương ứng với một lần nhấn *Enter* khi vẽ trực tiếp trong môi trường AutoCAD. Các ký tự trắng có thể đặt liên tiếp nhau trong cùng một cặp dấu ngoặc đơn (khác với các chuỗi lệnh viết bằng *File AutoLISP*, ở đó mỗi một ký tự trắng phải đặt trong một cặp dấu ngoặc kép).

5. Ngoài các nội dung chính như trình bày ở mục 5.2.1, ở phần thân chương trình nên bổ sung thêm các nội dung nhằm nâng cao chất lượng bản vẽ và tạo thuận lợi cho việc lập trình vẽ tự động và tiến hành sửa chữa bản vẽ, cụ thể là:

a) Sau khi mở *File*, nên dùng lệnh *Erase* để làm sạch màn hình trước khi bắt đầu thực hiện các lệnh vẽ, chẳng hạn

Writeln(fv,'Erase all ');

b) Nên dùng lệnh *Viewres* để làm tăng độ mịn của tất cả các đường cong và đường tròn trên bản vẽ, chẳng hạn

Writeln(fv,'Viewres 10000');

c) Trước khi đóng *File*, nên dùng lệnh *Redraw* sau đây để xoá các nét thừa

Writeln(fv,'Redraw');

6. Để AutoCAD tiến hành vẽ tự động bản vẽ đã được soạn thảo bằng ngôn ngữ Pascal ở tập tin đáp bản hoặc ở phần mềm thiết kế tự động, dùng chuột bấm lần lượt:

Tool, Run Script, C:\ACAD2002\ <tên bản vẽ>

5.3. SỬ DỤNG CÁC LỆNH CỦA AUTOCAD ĐỂ LẬP TRÌNH THIẾT KẾ TỰ ĐỘNG BẢN VẼ

5.3.1. Tạo các lớp cho bản vẽ

Để tạo ra các lớp cần thiết cho bản vẽ với màu sắc, loại nét và độ dày của nét khác nhau, phục vụ việc vẽ và in sau này, cần sử dụng lệnh *Layer* của AutoCAD. Chú ý rằng, trên bản vẽ kỹ thuật cần phân biệt các nét liền (có chiều dày nét là *b* để vẽ các kết cấu chính) và nét mảnh (có chiều dày nét là *b*/3 để gạch mặt cắt, ghi kích thước, đường gióng, đường cắt trích, nét chấm gạch, v.v...), trong đó *b* có thể chọn trong khoảng 0,6 đến 1,2 mm. Có thể chỉ định tên một hoặc nhiều lớp, chẳng hạn dùng câu lệnh:

Writeln(fv,'Layer N LI MA ');

để tạo ra các lớp vẽ nét liền và nét mảnh cho bản vẽ hoặc dùng câu lệnh

Writeln(fv,'Layer N CG C red CG Lt Dashdot CG ');

để tạo ra lớp vẽ nét chấm gạch (*Dashdot*) với màu đỏ.

Khi gọi các lớp LI và MA ra để trở thành lớp hiện hành sẽ cho màu, chẳng hạn dùng các dòng lệnh

Writeln(fv,'Layer on LI M LI C 2 ');

Writeln(fv,'Layer on MA M MA C 7 ');

để vẽ các nét liền với màu vàng, các nét mảnh với màu trắng, còn khi gọi lớp chấm gạch chỉ cần ghi

Writeln(fv,'Layer S CG ');

trong đó *S (Set)* có tác dụng gọi một lớp sẽ chỉ định tên (ở đây là *CG*) ra màn hình để trở thành lớp hiện hành; *M (Make)* phối hợp cả hai mục *New* và *Set* để tạo ra lớp mới đồng thời chỉ định luôn làm lớp hiện hành.

5.3.2. Cho tọa độ hai điểm

Từ 6 phương pháp nhập tọa độ điểm của AutoCAD, trừ phương pháp dùng phím chọn (PICK) của chuột không thích hợp trong lập trình thiết kế tự động, còn lại đều có thể sử dụng tuỳ từng trường hợp cụ thể. Đó là:

a) Dùng tọa độ tuyệt đối x1,y1; x2,y2.

b) Dùng tọa độ cực tuyệt đối (D < α) với D là khoảng cách từ điểm đang xét đến gốc tọa độ (0,0) hoặc đến tọa độ của điểm trước đó và góc nghiêng α so với đường chuẩn.

c) Dùng tọa độ tương đối: cho tọa độ điểm theo điểm cuối cùng vừa vẽ với ký hiệu @kx,ky, trong đó @: *Atsign*, có nghĩa là *Last Point* (điểm cuối cùng đã được xác định trên bản vẽ) ; kx,ky - khoảng cách theo các phương x và y.

d) Dùng tọa độ cực tương đối @ D<α.

e) Cho khoảng cách trực tiếp.

Như vậy trong *File Text*, muốn vẽ đường qua 2 điểm có thể sử dụng một trong các dòng lệnh sau đây:

```
Writeln(fv,'Line ',x1:a:b,',',y1:a:b,' ', x2:a:b,',',y2:a:b,' ');
Writeln(fv,'Line ',x1:a:b,',',y1:a:b,' @0,',ky:a:b,' ');
Writeln(fv,'Line ',x1:a:b,',',y1:a:b,' @',kx:a:b,',5 ');
Writeln(fv,'Line ',x1:a:b,',',y1:a:b,' @5,-3 ');
Writeln(fv,'Line ',x1:a:b,',',y1:a:b,' @100<-90 ');
```

trong đó dòng lệnh cuối cùng tương đương với

```
Writeln(fv,'Line ',x1:a:b,',',y1:a:b,' @0,-100 ');
```

5.3.3. Sử dụng các lệnh vẽ thông dụng trong thiết kế tự động

5.3.3.1. Lệnh Line và Pline

Trong khi lệnh *Line* được dùng để vẽ một đoạn thẳng hoặc các đoạn thẳng liên tiếp thì dùng lệnh *Pline* có thể vẽ các đoạn thẳng và đường cong liên tiếp. Chẳng hạn, để vẽ các đường thẳng và cung lượn liên tiếp biểu diễn trục có đường kính tiết diện thay đổi theo chiều dài, có thể dùng câu lệnh sau:

```
for i:=1 to 3 do
writeln(fv,'pline ', x[i]:a:b,',',y[i]:a:b,' ', xl[i+1]:a:b,',',y[i]:a:b,' A ',
        x[i+1]:a:b,',',y[i+1]:a:b,' ');
```

trong đó

$xl[i+1] = x[i+1] - r$,

với $r = y[i+1] - y[i]$.

Tuy nhiên ngay cả khi chỉ vẽ các đoạn thẳng liên tiếp vẫn nên dùng lệnh *Pline* vì như thế sẽ thuận tiện hơn khi cần thực hiện các lệnh hiệu chỉnh như *Erase, Break, Copy, Move*, v.v...

5.3.3.2. Lệnh Arc

Lệnh *Arc* được sử dụng trong những trường hợp sau:

a) Để vẽ cung lượn bằng cách:

- Cho tọa độ tâm xt, yt và tọa độ 2 điểm x1, y1; x2, y2.

- Cho tọa độ 3 điểm của đường lượn.

- Cho tọa độ tâm xt, yt cùng tọa độ điểm đầu và góc ôm (lấy dấu + khi vẽ đường lượn ngược chiều kim đồng hồ, lấy dấu - khi vẽ đường lượn cùng chiều kim đồng hồ).

Thí dụ:

```
Writeln(fv,'Arc C ',xt:a:b,',',yt:a:b,' ',x1:a:b,y1:a:b,' A-90');
```

trong đó dùng chữ C để ký hiệu tâm (*Center*), chữ A để ký hiệu góc (*Angle*).

b) Kết hợp với L trong vẽ đa tuyến khi vẽ đường lượn và đường thẳng liên tiếp. Thí dụ, vẽ đường thẳng từ tọa độ x1,y1 tới x2,y2, đường lượn tới x3,y3 và đường thẳng tới x4, y4:

```
Writeln(fv,'Pline ',x1:a:b,',',y1:a:b,' ',x2:a:b,',',y2:a:b,' A ',
        x3:a:b,',',y3:a:b, ' L ',x4:a:b,',',y4:a:b,' ');
```

hoặc từ x2, y2, dùng tọa độ tương đối để xác định tâm của cung lượn bán kính r, vẽ cung lượn rồi xác định điểm cuối của cung lượn và vẽ đường thẳng tới vị trí x4,y4:

```
Writeln(fv,'Pline',x1:a:b,',',y1:a:b,' ',x2:a:b,',',y2:a:b,' A C @0,',r:a:b,
        ' @',r:a:b,',0 L ', x4:a:b,',',y4:a:b,' ');
```

Chú ý rằng đường lượn thường là cung lõm nên cần được vẽ theo chiều ngược với chiều kim đồng hồ. Trường hợp cần vẽ theo chiều kim đồng hồ thì dùng tọa độ tâm với góc ôm có dấu trừ (-).

b) Vẽ đường cắt trích bằng cách sử dụng từ 2 đến 3 đường cong liên tiếp. Thí dụ :

```
Writeln(fv,'Arc',x1:a:b,',',y1:a:b,' ',x2:a:b,',',y2:a:b,' @3,-4');
```

```
Writeln(fv,'Arc',x2+3:a:b,',',y2-4:a:b,' @ 2,-4 @4,-2');
```

d) Vẽ cung tròn bị cắt một phần (chẳng hạn khi vẽ mặt cắt trục tại tiết diện có rãnh then):

```
Writeln(fv,'Arc C ',xt:a:b,',',yt:a:b,' ',x1:a:b,',',y1:a:b,' ',x2:a:b,',',y2:a:b);
```

trong đó xt,yt là tọa độ tâm, cung tròn được vẽ ngược chiều kim đồng hồ từ tọa độ x1,y1 tới x2,y2 tương ứng với chiều rộng rãnh then trên trục.

5.3.3.3. Lệnh Hatch và Bhatch

Lệnh *Hatch* và *Bhatch* được dùng để gạch mặt cắt trong đường biên kín (*Boundary Hatch*). Để xác định vùng vẽ ký hiệu mặt cắt, có thể dùng một trong 2 cách:

a) Sau lệnh *Bhatch* viết chữ P (*Pick Point*) để xác định một điểm nằm bên trong đường biên cần gạch mặt cắt, viết chữ ANSI31 để ký hiệu mặt cắt có dạng là các đoạn thẳng song song, sau đó ghi giá trị của khoảng cách giữa các đường gạch và góc nghiêng của các đường này (ghi số 0 nếu đường gạch nghiêng một góc 45^0 hoặc ghi con số 90 nếu đường gạch nghiêng một góc 135^0 so với phương nằm ngang), cuối cùng là tọa độ x, y của điểm được chọn nằm bên trong đường biên. Thí dụ:

Writeln(fv,'Bhatch P ANSI31 1 0 ',x:a:b,',',y:a:b,' ');

hoặc thay cho ANSI31 dùng chữ U (*User Defined*) cũng dùng để chỉ mẫu mặt cắt *(Pattern Type)* có dạng các đoạn thẳng song song, tiếp đến là giá trị của góc (45^0 hoặc 135^0) và khoảng cách giữa các đoạn thẳng song song so với phương nằm ngang:

Writeln(fv,'Bhatch P U 45 3 ',x:a:b,',',y:a:b,' ');

Ở đây cần lưu ý thứ tự ghi góc và khoảng cách giữa các đoạn thẳng song song so với phương nằm ngang cũng như giá trị của khoảng cách giữa các đường gạch và góc nghiêng của các đường gạch song song trong hai trường hợp là khác nhau. Đồng thời nếu vẫn còn sử dụng AutoCAD R14 thì giá trị của khoảng cách giữa các đường gạch cần lấy khá lớn, chẳng hạn, thay cho 1 ở ACAD 2000 hoặc ACAD 2002 khoảng cách này ở AutoCAD R14 là 25.

b) Sau lệnh *Hatch* viết chữ U rồi ghi giá trị của góc và khoảng cách của các đoạn thẳng song song so với phương nằm ngang, tiếp theo để đáp lại dòng nhắc *Double Hatch Area*, ghi chữ N nếu gạch theo một phương (để ký hiệu các chi tiết bằng kim loại) và chữ Y nếu gạch theo hai phương vuông góc (để ký hiệu vật liệu không kim loại), cuối cùng là chọn đối tượng xác định đường biên kín theo một trong 3 cách sau đây:

- Dùng chữ *W* ký hiệu cửa sổ với các tọa độ x1,y1; x2,y2. Thí dụ:

Writeln(fv,'Hatch U 45 3 N W ',x1:a:b,',',y1:a:b,' ', x2:a:b,',',y2:a:b,' ');

- Dùng tọa độ một điểm trên đường biên kín, với chú ý là đường biên kín phải được vẽ bằng *Circle*, *Pline* hoặc *Rectangle*;

- Dùng đa tuyến và sau khi gạch xong mặt cắt có thể giữ lại hoặc loại bỏ đường biên là đa tuyến vừa được tạo thành. Lúc này để đáp lại dòng nhắc *'Retain Polyline'* (có giữ lại đa tuyến hay không?) cần viết chữ Y hoặc N.

Thí dụ, để gạch mặt cắt lỗ tâm trên bản vẽ chế tạo trục (xem thí dụ 2), dùng đường biên kín từ điểm x1,y1 đến điểm xn,yn rồi trở về x1,y1 bằng câu lệnh sau:

Writeln(fv,'Hatch U 45 3 N N ',x1:a:b,',',y1:a:b,' ', x2:a:b,',',y2:a:b,' ', x3:a:b,',',y3:a:b,' ', ...,' ',xn:a:b,',',yn:a:b,' C ');

Chọn lệnh *Bhatch* hay *Hatch* với đối tượng nào để xác định đường biên kín tùy thuộc vào dạng của đường biên kín. Chẳng hạn, khi gạch mặt cắt cho bản vẽ bánh răng gồm mayơ và vành răng được chế tạo từ cùng một vật liệu và được khoét lỗ, nếu dùng lệnh *Bhatch* thì máy thực hiện việc gạch mặt cắt 4 lần cho 4 đường biên kín, còn nếu dùng lệnh *Hatch* với cửa sổ bao cả 4 đường biên kín thì máy chỉ gạch mặt cắt một lần, do đó tốc độ nhanh hơn nhiều. Trái lại, nếu đường biên kín được vẽ gắn với các kết cấu khác (chẳng hạn khi cắt trích một phần vỏ máy có kết cấu phức tạp) thì dùng tọa độ một điểm trên đường biên kín sẽ rất thuận lợi.

5.3.4. Lệnh vẽ nhanh

Để tăng tốc độ thiết kế bản vẽ có thể dùng các lệnh vẽ nhanh sau đây:

a) Lệnh *Fillet* để vẽ nối tiếp 2 đối tượng bằng một cung tròn. Hai đối tượng có thể là hai đường thẳng cắt nhau hoặc song song, hai đường cong, hai đường tròn. Lúc này cần lưu ý:

- Khi đối tượng là hai đường giao nhau hoặc hai đường tròn (thí dụ vẽ các má xích con lăn), cần chọn bán kính nối tiếp R;

- Để nối 2 đường song song, không cần nhập bán kính R vì khi đó bán kính chính là nửa khoảng cách giữa 2 đường song song (thí dụ, vẽ rãnh then trên trục);

- Có thể dùng lệnh *Fillet* với R = 0 để kéo dài hoặc xén các đối tượng giao nhau.

b) Lệnh *Offset* để tạo các đối tượng mới song song theo hướng vuông góc với đối tượng được chọn, thí dụ, vẽ hai vòng tròn đồng tâm, hai đường thẳng song song, hai đường cong song song.

c) Lệnh *Chamfer* để vát mép các cạnh.

d) Lệnh *Mirror* để tạo ra các đối tượng mới đối xứng với các đối tượng được chọn qua trục đối xứng xác định bởi 2 điểm.

Thí dụ dùng cửa sổ *W* với các tọa độ x1, y1 và x2, y2 bao lấy kết cấu cần lấy đối xứng qua một đường thẳng có các tọa độ xt1, yt1 và xt2, yt2, dùng câu lệnh sau:

```
Writeln(fv,'Mirror W ',x1:3:3,',',y1:3:3,' ',x2:3:3,',',y2:3:3,'
    ',xt1:3:3,',',yt1:3:3,' ',xt2:3:3,',',yt2:3:3,' ');
```

e) Lệnh *Copy* để sao chép các đối tượng đến các vị trí xác định trên bản vẽ, còn lệnh *Array* dùng để sao chép các đối tượng:

- Thành dãy chữ nhật (*Rechtangular Array*) hoặc

- Quanh tâm (*Polar Array*).

Thí dụ, muốn sao chép một đối tượng thành dãy *(R)* với 2 hàng cách nhau *kh* và 3 cột cách nhau *kc*, có thể sử dụng câu lệnh sau (dùng cửa sổ *W* có các tọa độ *x1,y1* và *x2,y2* để chọn nhóm đối tượng):

```
Writeln(fv,'Array W ',x1:3:3,',',y1:3:3,' ',x2:3:3,',',y2:3:3,' R 2 3 ',
    kh:3:3,' ', kc:3:3);
```

Khi dùng lệnh *Copy* hoặc lệnh *Array* nên sử dụng lựa chọn M (*Multiple*) để có thể sao chép nhiều bản từ nhóm các đối tượng được chọn. Để không làm rối hình, nên vẽ nhóm các đối tượng ở một vị trí thuận lợi, sau khi sao chép đến những vị trí xác định trên bản vẽ, sẽ xoá các nhóm đối tượng này bằng lệnh *Erase* với cùng một tọa độ cửa sổ như ở lệnh *Copy* .

Chẳng hạn muốn vẽ nhiều mặt cắt bulông và lỗ bulông phân bố trên hình vẽ, có thể tiến hành lần lượt như sau:

- Dùng lệnh *Circle* vẽ vòng tròn chỉ mặt cắt ngang của thân bulông tại một vị trí thích hợp trên bản vẽ;

- Dùng lệnh *Hatch* để gạch mặt cắt với đối tượng xác định đường biên là W;

- Dùng lệnh *Circle* hoặc lệnh *Offset* để vẽ vòng tròn chỉ lỗ bulông;

- Vẽ 2 đường tâm bulông;

- Dùng lệnh *Copy* để sao chép hình vẽ mặt cắt bulông và lỗ bulông tại điểm xt,yt đến hai điểm xd1, yd1 và xd2, yd2 (vẫn sử dụng cửa sổ *W*) bằng câu lệnh sau:

```
Writeln(fv,'Copy W ',x1: a:b,',',y1: a: b,' ', x2: a: b,',',y2: a: b,' M ',
    xt:a:b,',',yt: a: b,' ', xd1: a: b,',',yd1: a: b,' ', xd2: a: b,',',yd2: a:b,' ');
```

- Dùng lệnh *Erase* để xoá nhóm đối tượng ban đầu;

5.3.5. Lệnh hiệu chỉnh

Như đã biết, AutoCAD cung cấp cho người thiết kế các lệnh hiệu chỉnh sau đây:

- Lệnh *Break* để xén một phần đối tượng nằm giữa hai điểm chọn.
- Lệnh *Trim* để xén một phần đối tượng nằm giữa hai đối tượng hoặc
- Lệnh *Extrim* để xoá cùng một lúc nhiều đối tượng được giới hạn bởi một cạnh cắt.
- Lệnh *Stretch* để rời và kéo dãn các đối tượng.
- Lệnh *Extend* để kéo dài đối tượng và lệnh *Lenghthen* để thay đổi chiều dài đối tượng.
- Lệnh *Move* để *rời* các đối tượng.
- Lệnh *Rotate* để *quay* các đối tượng.
- Lệnh *Align* để *rời và quay* các đối tượng.

Khi lập trình thiết kế tự động bản vẽ, nhất là các bản vẽ phức tạp, thường rất hay sử dụng các lệnh hiệu chỉnh như *Break, Trim, Move* và *Rotate.*

Thật vậy, khi vẽ một cụm kết cấu gồm nhiều chi tiết lắp ghép với nhau như trục, ổ lăn, cốc lót, bộ đệm điều chỉnh, nắp ổ và vỏ hộp giảm tốc, công việc sẽ trở nên đơn giản và hình không bị rối nếu tiến hành vẽ xong từng chi tiết ở một vị trí thuận lợi trên bản vẽ rồi dùng lệnh *Move* để chèn chúng vào các vị trí xác định trên bản vẽ. Cũng thế, mỗi kết cấu của chi tiết máy hoặc bộ phận máy, thường phải được thể hiện trên bản vẽ bằng 3 hình chiếu chính và đôi khi cả ở hình cắt. Nếu với mỗi một vị trí trên bản vẽ, lại gọi đến một chương trình con với các tham trị và tham biến để thể hiện kết cấu sẽ tốn nhiều bộ nhớ và thời gian. Dùng lệnh hiệu chỉnh, người thiết kế chỉ cần vẽ kết cấu ở một vị trí trên một hình chiếu và nhanh chóng có được kết cấu ở vị trí khác hoặc trên hình chiếu khác.

Chẳng hạn, để vẽ các đầu bulông hoặc đầu vít, chỉ cần vẽ một lần ở vị trí ngang với kiểu 1 (vẽ 3 mặt) hoặc kiểu 2 (vẽ 2 mặt), rồi dùng lệnh *Rotate* quay một góc xác định để biểu diễn chúng ở vị trí khác. Thí dụ câu lệnh

```
Writeln(fv,'Rotate W ',x1:3:3,',',y13:3,' ', x2:3:3,',',y2:3:3,' ',
        xc:3:3,',',yc:3:3,' 90 ');
```

có nghĩa: quay một đối tượng được xác định bởi hai tọa độ x1,y1 và x2,y2 của cửa sổ (*W*), sau khi kết thúc việc lựa chọn đối tượng (thêm 1 dấu trắng sau dấu trắng ứng với tọa độ x2,y2), chọn tiếp điểm chuẩn (*base point*) với tọa độ xc, yc mà các đối tượng quay xung quanh, cuối cùng ghi

giá trị góc quay là 90°. Nếu muốn vẽ nhiều đối tượng vừa quay đến những vị trí xác định, dùng lệnh *Copy* vừa trình bày ở mục trên.

5.3.6. Ghi kích thước trên bản vẽ

5.3.6.1. Quy tắc ghi kích thước

Ghi kích thước trên bản vẽ chế tạo cần được thực hiện theo những quy tắc sau đây:

1) Số lượng kích thước trên bản vẽ phải ít nhất nhưng đủ để chế tạo và kiểm tra chi tiết máy.

2) Ghi kích thước phải thể hiện được quá trình chế tạo chi tiết, tức là phải tương ứng với các công đoạn khác nhau và các nguyên công khác nhau khi gia công cơ khí (thí dụ, tiện bề mặt trục, phay rãnh then trên trục, v.v...).

3) Ghi kích thước phải đảm bảo độ chính xác của các yếu tố quan trọng nhất của chi tiết. Muốn vậy tuỳ theo trường hợp cụ thể cần lựa chọn một trong 3 cách sau đây để ghi ghi kích thước:

- Phương pháp chuỗi để ghi liên tiếp các kích thước, đảm bảo độ chính xác của kích thước sau so với kích thước trước, thí dụ ghi trị số của các khoảng cách trục $a_w[k]$ của cấp k và trị số dung sai của $a_w[k]$ trong bản vẽ lắp hộp giảm tốc bánh răng n cấp.

- Phương pháp tọa độ để ghi kích thước theo một chuẩn A nào đó, đảm bảo độ chính xác về khoảng cách của tất cả các yếu tố so với chuẩn.

- Phương pháp liên hợp, phối hợp hai phương pháp chuỗi và phương pháp tọa độ nhằm giảm sai số ở những kích thước yêu cầu chính xác nhất, thí dụ dùng mặt mút của trục làm chuẩn để ghi kích thước đoạn trục lắp với đĩa xích và ghi kích thước xác định vị trí rãnh then trên trục (phương pháp tọa độ) sau đó ghi trị số của chiều dài rãnh then (phương pháp chuỗi) .

4) Không ghi kích thước ở khâu khép kín trừ trường hợp đó là kích thước tham khảo, khi đó phải đánh dấu sao (*) ở số mũ của trị số kích thước. Thí dụ, ghi chiều dài côn R_e với dấu (*) trên bản vẽ chế tạo bánh răng côn.

5) Ghi kích thước phải kèm theo sai lệch giới hạn trừ trường hợp đó là kích thước phân định các miền có cấp chính xác khác nhau, kích thước vát, góc lượn, chiều dài phần cắt ren của bulông và những yếu tố yêu cầu thấp về độ chính xác chế tạo.

Sai lệch giới hạn của kích thước lắp ghép phụ thuộc kiểu lắp và cấp chính xác được chọn trong bảng P4.1 và P4.2 của tài liệu [2], còn trị số

dung sai từ IT9 đến IT14 cho trong bảng P4.10 [2].

5.3.6.2. Những vấn đề cần lưu ý khi soạn thảo phần mềm ghi kích thước

Để sử dụng các lệnh vẽ của AutoCAD với yêu cầu các kích thước được ghi phải theo quy định của TCVN và không làm rối hình vẽ, cần lưu ý những vấn đề sau đây:

1) Đặt lệnh trạng thái:

Trước khi ghi kích thước phải đặt lệnh trạng thái, chẳng hạn dùng câu lệnh sau đây:

Writeln(fv,'Dim Dimexo 0 Dimexe 2 Dimasz 3 Dimtsz 0 Dimcen -3 Dimtad 1 Dimtih 0 Dimgap 1 Dimscale 1 Dimdec 0 Dimtdec 2 ');

trong đó

Dimexo = 0 : khoảng cách từ đầu đường gióng đến gốc đường kích thước bằng 0;
Dimexe = 2 : đường gióng vượt quá đường kích thước 2 mm;
Dimcen =-3 : ký hiệu tâm bằng dấu tâm hoặc đường tâm với phần vượt quá bán kính là 3 mm;
Dimtad = 1 : con số và chữ ghi kích thước và kiểu lắp nằm phía trên đường ghi kích thước (theo TCVN) (*)
Dimtih = 0 : chữ và con số ghi đường kính nằm phía trên đường ghi kích thước;
Dimgap =1 : khoảng cách từ đoạn thẳng ghi kích thước đến dòng ghi giá trị kích thước;
Dimtsz = 0 : dùng mũi tên cho đường ghi kích thước;
Dimasz = 3 : độ dài mũi tên bằng 3 mm;
Dimdec = 0 : kích thước có trị nguyên (số số lẻ cần ghi là 0);
Dimtdec= 2 : số số lẻ của trị số dung sai cần ghi là 2.

Trường hợp kích thước cần ghi là một số thập phân (số con số sau dấu phẩy > 0), thí dụ, d-t_1 hoặc t_1 tại tiết diện trục có rãnh then thì trước khi ghi kích thước cần đặt lại lệnh *Dimdec*, chẳng hạn, khi giá trị kích thước có 1 dấu phẩy dùng lệnh

Writeln(fv,'Dim Dimdec 1 Dimtdec 2 ');

2) Soạn thảo chương trình con ghi kích thước:

Nên soạn thảo một số chương trình con, tạo thuận lợi cho việc ghi kích thước trên bản vẽ lắp và bản vẽ chế tạo, bao gồm nội dung ghi kích

(*) Trường hợp gán Dimtad = 0 thay cho 1 thì chữ, con số và kiểu lắp nằm cùng đòng và ở giữa đường ghi kích thước

thước tự do với số số lẻ bất kỳ sau dấu phẩy, ghi kích thước lắp ghép kèm theo ký hiệu của AutoCAD, đó chính là:

kh=' %%C' : ký hiệu đường kính (chữ ∅) ;
kh=' %%p' : ký hiệu dung sai đối xứng (±) ;
kh=' %%d' : ký hiệu độ (°)

và cuối cùng là tiếp vĩ thể hiện các nội dung khác nhau như : hệ lắp, kiểu lắp và cấp chính xác, trị số của các sai lệch giới hạn.

Chẳng hạn, có thể sử dụng các *Procedure* ghi kích thước sau đây trong một *Unit* ghi kích thước:

```
Procedure dimfree(kg:string[8];x1,y1,x2,y2,xg,yg,ts:real);
 var hx:byte;
 begin
    if ts<10 then hx:=1 else hx:=3;
    writeln(fv,'Dim ',kg,' ',x1:3:3,',',y1:3:3,' ',x2:3:3,',',y2:3:3,' ',
    xg:3:3,',',yg:3:3,' ',ts:hx:0);
 end;
 {-----------------------------------------}
Procedure dimpost(kg:st4;x1,y1,x2,y2,xg,yg,ts:real;kh:string[4];
                  tv:st8;ds:real);
begin
  writeln(fv,'Dim ',kg,' ',x1:3:3,',',y1:3:3,' ', x2:3:3,',',y2:3:3,' ',
  xg:3:3,',',yg:3:3,' ', kh, ts:hx:0,tv,ds:3:2);
end;
{-------------------------------------------------------------------------------}
Procedure dimtext(kg:st4;x1,y1,x2,y2,xg,yg:real; kh:st4;ts:real;tv:st8);
begin
  writeln(fv,'Dim ',kg,' ',x1:3:3,',',y1:3:3,' ', x2:3:3,',',y2:3:3,' ',
  xg:3:3,',',yg:3:3, ' ', kh,  ts:3:0,tv,'' );
end;
{-------------------------------------------------}
Procedure dimver(x1,y1,x2,y2,kx,tp,tm:real);
begin
  writeln(fv,'Dim Dimtol on Dimtp ',tp:1:2,' Dimtm ',tm:1:2,' ',ver',
  x1:3:3,',',y1:3:3,' ',x2:3:3,',',y2:3:3,' ', x1+kx:3:3,',',y1:3:3,' ' );
 end;
{-------------------------------------------------}
Procedure dimhor(x1,y1,x2,y2,ky,tp,tm:real);
begin
   writeln(fv,'Dim Dimtol on Dimtp ',tp:1:2,' Dimtm ',tm:1:2,' ',hor',
  x1:3:3,',',y1:3:3, ' ',x2:3:3,',',y2:3:3,' ', x2:3:3,',',y2+ky:3:3,' ' );
 end;
```

Chú thích :

a) *Procedure dimfree*: ghi kích thước tự do;

Procedure dimtext: ghi kích thước và gán tiếp vĩ cho kích thước;

Procedure dimpost: ghi kích thước, gán tiếp vĩ và trị số dung sai cho kích thước;

Procedure dimver : ghi kích thước theo phương đứng kèm theo trị số dung sai;

Procedure dimhor: ghi kích thước theo phương ngang kèm theo trị số dung sai.

b) *kg* : kiểu ghi *('hor'* hoặc *'ver')*; st4 : string[4] ; st8 : string[8]; x1,y1, x2, y2: tọa độ giới hạn khoảng kích thước cần ghi ; xg, yg : tọa độ vị trí ghi kích thước; ts: ký hiệu thông số cần ghi kích thước; ds: trị số dung sai.

c) Thí dụ, muốn ghi đường kính tại các tiết diện i của trục theo phương đứng với liểu lắp k6, chỉ cần gọi tới *Procedure dimtext* ở trên :

dimtext('ver',x1[i],y1[i], x2[i],y2[i], xy[i],yx[i],'%%c',d[i],'k6');

còn khi cần ghi trị số khoảng cách trục với dung sai đối xứng, chỉ cần gọi tới

dimpost('hor',x[1],ya,x[2],ya,x[1],ya-80,' ',aw,' %%p',fa);

trong đó *fa* là trị số dung sai của khoảng cách trục, phụ thuộc vào dạng tiếp xúc và khoảng cách trục aw (xem bảng 21.21 [2]).

d) Cả 5 *procedure* ghi kích thước ở trên đều dùng lệnh "*Dim*", vì vậy sau khi sử dụng một trong các *procedure* này cũng như sau khi sử dụng các lệnh ghi kích thước "*Dim*" của AutoCAD, phải dùng lệnh *Exi*, nếu không các lệnh tiếp theo của tập tin sẽ không được thực hiện, muốn vậy hãy dùng câu lệnh sau:

Writeln(fv,'Exi');

Như vậy để giảm số lần phải dùng câu lệnh này, nên tập trung ghi kich thước theo từng cụm chi tiết trên bản vẽ.

3) Ghi kích thước theo đường dẫn:

Sử dụng lệnh ghi kích thước theo đường dẫn *(Leader)* để ghi bán kính góc lượn. Chẳng hạn từ điểm x1, y1 trên cung lượn với mũi tên và đường dẫn đến điểm x2, y2 sẽ ghi được bán kính góc lượn R=3 khi dùng câu lệnh sau :

Writeln(fv,'Leader ', x1:3:3,',',y1:3:3,' ', x2:3:3,',',y2:3:3,' R3 ');

4) Ghi kích thước góc:

Muốn ghi kích thước góc có thể sử dụng câu lệnh sau:

writeln(fv,'Dimangular ', xd:3:3,',',yd:3:3,' ', x1:3:3,',',y1:3:3,' ',
x2:3:3,',',y2:3:3,' ', xg:3:3,',',yg:3:3);

trong đó

xd, yd : tọa độ của đỉnh góc;

x1, y1 : tọa độ điểm cuối cạnh thứ nhất;

x2, y2 : tọa độ điểm cuối cạnh thứ hai;

xg, yg : tọa độ vị trí ghi kích thước góc.

5.3.7. Ghi sai lệch hình dáng và vị trí

Như đã biết, sai lệch hình dáng và vị trí ảnh hưởng đến chất lượng và độ tin cậy làm việc của chi tiết máy vì nếu sai lệch vượt quá trị số cho phép sẽ:

- Làm giảm độ cứng tiếp xúc, làm thay đổi đặc tính lắp ghép, gây va đập mép và mòn cục bộ;

- Làm tăng sự phân bố không đều tải trọng trên đường tiếp xúc của các răng ăn khớp;

- Gây tải trọng động phụ (khi độ đảo lớn) và làm giảm độ chính xác động học của bánh răng.

Vì vậy trên bản vẽ chế tạo cần phải ghi dung sai hình dáng và vị trí.

Khi tiến hành ghi dung sai hình dáng và vị trí trên bản vẽ chế tạo cần lưu ý những vấn đề sau đây:

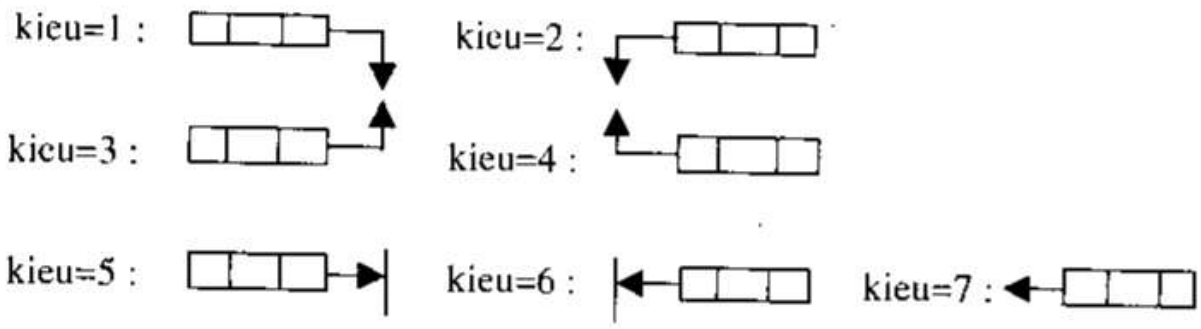
a) Vì có nhiều loại sai lệch hình dáng và vị trí bề mặt cho nên người thiết kế cần phân tích điều kiện làm việc cụ thể của chi tiết máy để chọn loại và trị số dung sai hình dáng và vị trí thích hợp. Chẳng hạn, đối với bản vẽ chế tạo trục của hộp giảm tốc thường yêu cầu ghi độ đảo hướng tâm và độ đảo mặt đầu của các bề mặt trục lắp với các chi tiết quay, dung sai độ trụ của mặt lắp ổ lăn,dung sai độ chéo và độ không đối xứng của rãnh then trên trục. Đối với bản vẽ chế tạo bánh răng cần ghi độ đảo hướng tâm của mặt trụ đỉnh và độ đảo mặt mút của vành răng và mayơ đối với đường tâm lỗ bánh răng cũng như dung sai độ chéo và độ không đối xứng của rãnh then trên mayơ bánh răng.

b) Ký hiệu và trị số dung sai hình dáng và vị trí được ghi trong khung, gồm hai hoặc ba phần, từ trái qua phải lần lượt là ký hiệu quy ước của dung sai, thí dụ dùng mũi tên để chỉ dung sai về độ đảo, hai vòng tròn đồng tâm để chỉ dung sai đối xứng, v.v... (xem bảng 20.14 [2]), tiếp đến

là trị số dung sai tính bằng mm và cuối cùng là ký hiệu bằng chữ cái in hoa biểu thị mặt chuẩn hoặc một mặt nào đó. Khung được đặt ở một vị trí thích hợp trên bản vẽ và được nối với mặt cần ghi dung sai bằng nét mảnh và mũi tên chỉ vào đường biên của mặt này. Chẳng hạn, có thể soạn thảo *Procedure* ghi độ đảo sau đây:

Procedure ghidodao(kieu:byte;x1,y1,kc:real;v1,v2:string[4]);

trong đó :



ở đây các kiểu 5, 6 và 7 được dùng để ghi độ đảo mặt đầu, các kiểu còn lại để ghi độ đảo hướng tâm tuỳ theo vị trí điểm đo.

x1,y1 : tọa độ của điểm trên bề mặt cần ghi độ đảo .

kc : khoảng cách từ điểm đo đến khung độ đảo .

v1 : trị số dung sai độ đảo.

v2 : ký hiệu mặt chuẩn.

c) Chuẩn được ký hiệu bằng tam giác đều tô đen, đáy tam giác nằm trên biên mặt chuẩn hoặc trên đường gióng từ biên mặt chuẩn, từ một đỉnh tam giác dùng một đoạn thẳng nối đến khung chữ nhật, bên trong ghi ký hiệu mặt chuẩn. Như vậy để vẽ mặt chuẩn hãy soạn thảo một *Procedure matchuan*, trong đó để tô đen tam giác có cạnh a = 3 mm, tương ứng chiều cao tam giác h = 2,65 mm, hãy sử dụng lệnh *Solid* trong câu lệnh sau đây:

Writeln(fv,'Solid ',xc:3:3,',',yc:3:3,' @-1.5, 2.65 @3, 0 ');

sau đó vẽ đoạn thẳng nối đến khung chữ nhật và ghi ký hiệu mặt chuẩn theo các lệnh vẽ thông dụng của AutoCAD.

5.3.8. Ghi độ nhám bề mặt

Như đã biết, nhiều chỉ tiêu quan trọng của chi tiết máy như độ bền mòn, độ cứng tiếp xúc, độ bền của mối ghép có độ dôi, v.v... phụ thuộc vào độ nhám bề mặt. Độ nhám lớn tác động xấu đến độ bền mỏi của chi tiết máy, nhất là tại những nơi có tập trung ứng suất như góc lượn của

trục, bề mặt lắp ghép bánh răng, rãnh then, chân ren, v.v... Nhám bề mặt cũng ảnh hưởng đến độ đồng đều và chất lượng thấm cacbon, thấm nitơ và thấm xianuya. Vì vậy khi thiết kế cần chọn độ nhám thích hợp và ghi đầy đủ trên bản vẽ.

5.3.8.1. Ký hiệu độ nhám

Sử dụng một trong 3 ký hiệu sau đây để biểu thị độ nhám:

Dấu $\surd$ dùng để ký hiệu độ nhám bề mặt mà không chỉ dẫn phương pháp gia công bề mặt đó. Đây là cách ký hiệu ưu tiên.

Dấu $\surd$ với vòng tròn nội tiếp với hai cạnh tam giác dùng để ký hiệu độ nhám bề mặt gia công không phoi (đúc, dập, cán, v.v...);

Dấu $\surd$ để ký hiệu độ nhám bề mặt gia công có phoi (tiện, phay, bào, v.v...);

Ngoài ký hiệu nhám còn phải ghi trị số nhám,với R_a (là sai lệch trung bình số học của prôfin) chỉ cần ghi trị số của nó, còn khi dùng R_z (là chiều cao nhấp nhô bề mặt tính trên 10 điểm) phải ghi cả ký hiệu R_z và trị số nhám.

Trị số nhám ứng với các phương pháp gia công khác nhau xem trong bảng 21.2 [2], trị số độ nhám mặt răng, mặt trụ đỉnh và mặt mút chuẩn - xem bảng 21.3 [2], còn độ nhám của bề mặt trục và lỗ hộp lắp ổ lăn - xem bảng 21.4 [2].

5.3.8.2. Vị trí ghi ký hiệu độ nhám

- Để dễ đọc bản vẽ các ký hiệu độ nhám cần được ghi trên các đường biên hoặc đường gióng gần chỗ ghi kích thước. Đỉnh của ký hiệu độ nhám phải hướng về bề mặt chi tiết.

- Quy ước ghi độ nhám mặt răng trên đường chia của bản vẽ bánh răng.

- Ngoài độ nhám được ghi trực tiếp trên các phần bề mặt khác nhau, độ nhám như nhau của các bề mặt còn lại (độ nhám của vai trục, đoạn vát, v.v...) được ghi ở góc bên phải của bản vẽ chế tạo bằng ký hiệu quy định và được bổ sung thêm dấu ($\surd$).

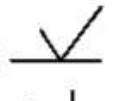
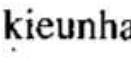
Như vậy để thuận tiện cho việc ghi độ nhám trên bản vẽ chế tạo nên soạn thảo các *Procedure* sau đây:

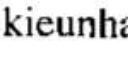
Procedure nham1(kieunham,cachviet:byte;xd,yd,tsnham:real);

Procedure nham2(kieunham,cachviet:byte;xd,yd,tsnham:string[5]);

Procedure nhamconlai(tsnham:string[5]);

trong đó :

kieunham = 1:  ; kieunham = 2 : 

kieunham = 3:  ; kieunham = 4 : 

cachviet =1 : ghi trị số nhám theo phương ngang;
cachviet =2 : ghi trị số nhám theo phương đứng;
xd,yd : tọa độ đỉnh nhám;
tsnham : ở *Procedure nham1* là trị số nhám theo R_a (chỉ cần ghi trị số của R_a mà không ghi ký hiệu R_a), còn ở *Procedure nham2* là trị số nhám theo R_z.

Thí dụ ghi:

nham2(2,1,150,200,' Rz20 ');

5.3.9. Ghi điều kiện kỹ thuật trên bản vẽ

Khi thực hiện các bản vẽ kỹ thuật nên sử dụng phông (*Font*) chữ *Simplex. shx*, nghiêng 15° để ghi kích thước, lắp ghép và dung sai.

Điều kiện kỹ thuật được ghi ở phần cuối bản vẽ lần lượt từng dòng theo kiểu chữ của AutoCAD. Mỗi kiểu chữ (*Text Style*) có thể sử dụng *Font* chữ AutoCAD như *SHX Font* hoặc *TTF (True Type Font)*. Các kiểu chữ khác được định bằng lệnh *Style*. Để nhập dấu tiếng Việt cho *Font* chữ .SHX cần kết hợp một *Font* chữ của AutoCAD và *File* dấu VN.SHX.

Khi ghi bảng thông số cho các bản vẽ chế tạo bánh răng, bánh vít hoặc trục vít, cũng như ghi khung tên, thống kê chi tiết và điều kiện kỹ thuật nếu dùng *Vntime* hoặc *VntimeH* cũng như *VnArial* hoặc *VnArialH* sẽ không phù hợp mà nên dùng các phông chữ tiếng Việt khác, phù hợp hơn với phông chữ *Simplex* như : *Vntxt.shx* để viết chữ thường và *Vnsimli.shx* để viết chữ hoa

Khi dùng các phông chữ tiếng Việt này cần sử dụng các con số sau đây ngay sau chữ cần có dấu [16]:

Số	*Biểu thị*	*Thí dụ*	*Sẽ xuất hiện*
1	dấu sắc	ha1i	hái
2	dấu huyền	ha2i	hài
3	dấu hỏi	ha3i	hải
4	dấu ngã	ha4i	hãi
5	dấu nặng	ha5i	hại
6	dấu mũ	be6n	bên
7	dấu móc	chu7o7ng	chương
8	dấu á	na8m	năm
9	chữ đ	d9o65	độ

Chú ý rằng trước khi sử dụng dấu tiếng Việt để soạn thảo phần mềm thiết kế tự động bản vẽ trong môi trường AutoCAD, cần kích chuột phải để hiện cửa sổ *Vietkey* và dùng *Vni* cho mục *tiếng Việt* (*) .

Chẳng hạn để viết nội dung của điều kiện kỹ thuật theo kiểu chữ Vntxt, chiều cao chữ 3,5 ứng với khổ bản vẽ A3 và A4, nét chữ là 1, nghiêng 15 độ có thể dùng *Procedure* sau đây:

```
Procedure dkkt (sodong,v1,u1:byte);
var v: array[0..8] of integer; dong:array[0..8] of string[60];
begin
  for i:=1 to sodong do v[i+1]:=v1-7;
  writeln(fv,'Style Vntxt 3.5 1 15  ');
  for i:=1 to sodong do writeln(fv,'Text ',u[1]:2,' 0 ',dong[i],' ');
end;
```

Lúc này nếu cần ghi 3 dòng tại vị trí u1 = 60 , v1 = 70 với nội dung:

```
dong[1]='Độ rắn bề mặt HB = 260';
dong[2]='Sai lệch giới hạn không chỉ dẫn của các kích thước: bao - H14 ');
dong[3]='bị bao - h14,còn lại : ±IT12/2 ');
```

với chú ý là dùng các con số từ 1 ÷ 9 để biểu thị dấu và chữ đ tiếng Việt trong hai dấu nháy.

Khi đó trong phần điều kiện kỹ thuật chỉ cần ghi:

```
dkkt(3,60,70);
```

5.4. LẬP TRÌNH THIẾT KẾ TỰ ĐỘNG BẢN VẼ CHẾ TẠO CHI TIẾT MÁY

5.4.1. Các bước tiến hành

Chương trình vẽ tự động bản vẽ chế tạo chi tiết máy được thực hiện theo các bước sau đây:

1. Nhập từ bàn phím hoặc gọi số liệu từ các *File* kết quả ở chương trình tính toán thiết kế chi tiết máy.

2. Tính toán bổ sung các yếu tố kết cấu của chi tiết máy như:

- Kết cấu và kích thước lỗ tâm của trục – bảng 13.3 [2];

- Kết cấu của bánh răng trụ và bánh răng côn chế tạo rời - hình 14.1 và 14.8 [2];

(*) Cũng có thể dùng *Telex* ở cửa sổ *Vietkey* và soạn thảo tiếng Việt cho bản vẽ như phương pháp thông thường (không dùng con số mà dùng chữ để biểu thị dấu tiếng Việt).

- Kết cấu của bánh răng trụ và bánh răng côn liền trục - hình 14.5, 14.6 và 14.7 [2];

- Kết cấu của bánh vít – hình 14.14 [2];

- Kết cấu của bánh đai – hình 14.16 [2];

- Kết cấu của đĩa xích – hình 14.17 [2], bảng 14.4a và 14.4b [2];

3. Định khổ bản vẽ. Chú ý rằng bản vẽ chế tạo chi tiết máy thường được vẽ trên khổ giấy A4 hoặc A3 theo tỷ lệ 1:1. Trường hợp với khổ giấy A3 mà kích thước quá lớn thì có thể dùng tỷ lệ 1:2. Khi chọn khổ bản vẽ có thể dựa vào chiều dài (đối với bản vẽ chế tạo trục) hoặc đường kính (đối với bản vẽ chế tạo bánh răng hoặc các chi tiết quay khác).

4. Dựa vào các số liệu từ *File* kết quả, xác định lại các thông số và kích thước (nhân với hệ số tỷ lệ đã chọn) và tính toán bổ sung các kích thước khác như: đường kính và chiều dài mayơ của chi tiết quay, các yếu tố của phần khoét và lỗ khoét trên thân bánh răng, v.v... Cho các tọa độ gốc và thiết lập các biểu thức tính toán các tọa độ tương đối của các yếu tố trên bản vẽ.

5. Vẽ các kết cấu chính và các mặt cắt. Chẳng hạn đối với bản vẽ chế tạo bánh răng có thể tiến hành lần lượt như sau:

- Tính và vẽ vành răng và mayơ bánh răng (vẽ 1/2 kết cấu);

- Lấy đối xứng qua trục nằm ngang;

- Khoét rãnh trên mayơ;

- Gạch mặt cắt;

- Vẽ các đường biên và các vết;

- Vẽ các đường đỉnh răng bao gồm cả đoạn vát mép đỉnh răng;

- Vẽ lỗ bánh răng và rãnh then trên lỗ.

6. Ghi kích thước chiều dài (chiều dài các đoạn trục, chiều dài mayơ chi tiết quay, chiều dài rãnh then trên trục và trên lỗ chi tiết quay v.v...), đường kính danh nghĩa kèm theo dung sai (đường kính các đoạn trục, đường kính đỉnh bánh răng v.v...). Chú ý rằng kích thước danh nghĩa là kích thước thực, do đó khi ghi kích thước, các giá trị biểu thị kích thước cần được chia cho hệ số tỷ lệ đã dùng.

7. Ghi sai lệch hình dáng và vị trí.

8. Ghi nhám bề mặt và nhám còn lại.

9. Kẻ và điền vào bảng thông số (đối với bản vẽ chế tạo bánh răng, bánh vít, trục vít). Bảng thông số đặt ở góc phải, phía trên bản vẽ gồm 3 phần: các số liệu cơ bản (để chế tạo) gồm có: môđun m, số răng z, góc

nghiêng β, hệ số dịch chỉnh x, prôfin gốc, cấp chính xác theo TCVN; các số liệu kiểm tra và các số liệu tham khảo.

Để kiểm tra chiều dày răng có thể sử dụng một trong 3 thông số sau đây:

- Khoảng pháp tuyến chung W. Công thức xác định W tra ở bảng P4.13 và P4.15 của tài liệu [2];

- Dây cung chiều dày răng $\overline{s_c}$ và chiều cao đo $\overline{h_c}$ đến dây cung này. Trị số của chúng có thể tra trong bảng P4.14 [2];

- Khoảng kích thước con lăn M.

Bộ số liệu kiểm tra bao gồm các yếu tố đặc trưng cho mức chính xác động học, mức làm việc êm, vết tiếp xúc và khe hở cạnh răng. Các số liệu kiểm tra này phụ thuộc cấp chính xác và đường kính bánh răng có thể tra trong TCVN 1067-84. Tuy nhiên với các bánh răng của bộ truyền cấp chính xác 7 và thấp hơn cho phép không ghi các số liệu kiểm tra trong bảng mà ghi ở phần điều kiện kỹ thuật.

10. Ghi điều kiện kỹ thuật:

Điều kiện kỹ thuật có thể bao gồm các nội dung sau:

- Cơ sở chế tạo chọn các số liệu kiểm tra theo TCVN 1067-84 (đối với bản vẽ chế tạo bánh răng có cấp chính xác từ 7 trở xuống).
- Phương pháp nhiệt luyện và độ rắn của vật liệu chế tạo chi tiết.
- Sai lệch giới hạn của các kích thước không chỉ dẫn trên bản vẽ.
- Dung sai độ không đối xứng và độ chéo rãnh then trên trục hoặc trên mayơ của chi tiết quay.
- v.v...

11. Kẻ và điền vào khung tên:

Khung tên dùng chung cho tất cả các bản vẽ chế tạo chi tiết máy vì vậy nên soạn thảo một *Procedure* khung tên theo TCVN và sẽ gọi đến ở mỗi bản vẽ. Chẳng hạn, có thể sử dụng các biến theo phương pháp truyền tham số ở chương trình con sau đây:

Procedure khungten(tenct,kyhieu,hdan,duyet,sv,lop,nhanvl:string[12]);

trong đó tên chi tiết (*tenct*) và nhãn vật liệu (*nhanvl*) có thể gán vào chương trình vẽ, các biến còn lại gồm tên người hướng dẫn (*hdan*), người duyệt (*duyet*), tên sinh viên (*sv*), tên đơn vị hoặc lớp (*lop*) sẽ được cho trực tiếp khi thực hiện bản vẽ.

5.4.2. Soạn thảo các Unit dùng chung

Ngoài một số chương trình con dùng chung cho các bản vẽ như các *Procedure* ghi kích thước, ghi nhám bề mặt, bảng thông số ăn khớp răng, khung tên đã trình bày ở trên, trong các bản vẽ chế tạo còn có một số câu lệnh lặp lại nhiều lần trong chương trình thiết kế bản vẽ cũng như có một số kết cấu dùng chung cho nhiều bản vẽ như mặt cắt trục tại tiết diện có rãnh then, lỗ tâm ở đầu trục, lỗ chi tiết quay v.v... Khi thiết kế tự động bản vẽ chế tạo nên tạo ra các *Unit* bao gồm các nội dung trên đây. Chúng sẽ được gọi đến khi thực hiện từng chương trình vẽ cụ thể.

Dưới đây là một vài thí dụ.

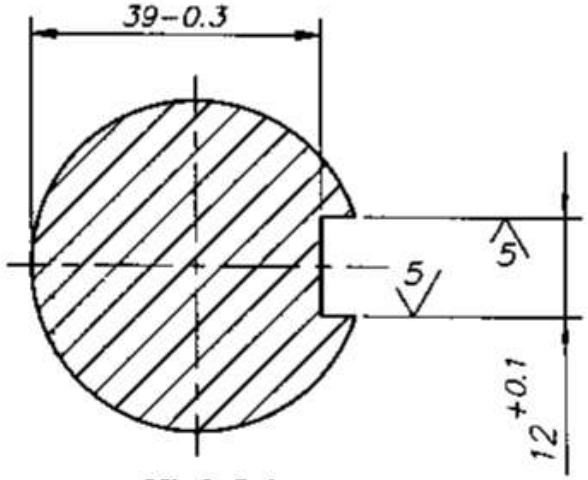
Thí dụ 1

Soạn thảo chương trình vẽ mặt cắt trục, bao gồm phần xác định tọa độ các điểm của rãnh then trên mặt cắt ngang của trục, vẽ vòng tròn bán kính d/2 bị cắt một đoạn tương ứng với chiều rộng then b, gạch mặt cắt, ghi kích thước danh nghĩa của chiều rộng b và chiều sâu rãnh then t1 với sai lệch giới hạn của chúng, ghi nhám bề mặt rãnh then và cuối cùng là kẻ hai đường tâm.

Thực hiện

```
Program mctruc;
Uses crt,tcc;
var  xcp,xct,yct,ycd,x1,x2:real;
{--------------------------------}
Procedure khoitao;
begin
   writeln(fv,'Limits ','297,210');(1)
   writeln(fv,'Units ','2 2 3 2  ');
   writeln(fv,'Zoom  a '
    writeln(fv,'Layer N  LI  MA ');
   writeln(fv,'Layer N  CG C red CG Lt Dashdot CG ');
end;
{-------------------------------------------------------}
Procedure trangthai; (2)
begin
  writeln(fv,'Layer  on MA M MA C7  ');
  writeln(fv,'Dim Dimexo 0 Dimexe 2 Dimtext 3 Dimasz 3 Dimtsz 0
        Dimcen 3 ');
  writeln(fv,'Dim Dimtad 1 Dimtih 0 Dimgap 1 Dimscale 1 Dimdec 0
        Dimtdec 2 ');
end;
{-----------------------------------------------------------------}
Procedure solieu;(3)
begin
```

```
    xcp:=xt + sqrt(sqr(d/2) - sqr(b/2));
    xct:=xt + d/2 - t1 ;
    yct:=yt + b/2;
    ycd:=yt - b/2;
    x2:=xt + d/2 +17;
  end;
{--------------------------------------}
Procedure vemct(xt,yt,d,b,t1:real);
begin
    writeln(fv,'Layer  on LI M LI C5  ');
    writeln(fv,'ARC CE ',xt:3:3,',',yt:3:3,' ',
            xcp:3:3,',',yct:3:3,' ', xcp:3:3,',',ycd:3:3);
    writeln(fv,'Pline ',xcp:3:3,',',yct:3:3,' ', xct:3:3,',',yct:3:3,' ',
            xct:3:3,',', ycd:3:3,' ', xcp:3:3,',',ycd:3:3,' ');
    writeln(fv,'Layer  on MA M MA C7  ');
    writeln(fv,'Bhatch P U 45 4 ', xct-4:3:3,',', yct+2:3:3,' ');
    trangthai;(2)
    writeln(fv,'Style KT Simplex 3.5 1 15  ');
    dimtext ('hor ',xt-d/2,yt,xct,yct,xct,yt+d/2+12,' ',t1,1,ss);(4)
    dimver(xcp,yct,xcp,ycd,25,slt,sld);(4)
    writeln(fv,'Exi');
    nham1(1,1,x1,ycd,5);
    nham2(2,1,x2,yct,5);
    writeln(fv,'Layer S CG ');
    writeln(fv,'Line ', xt-d/2-5:3:3,',',yt:3:3,' ', xt+d/2+5:3:3,',',yt:3:3,' ');
    writeln(fv,'Line ', xt:3:3,',',yt+d/2+5:3:3,' ', xt:3:3,',',yt-d/2-5:3:3,' ');
end;
{--------------------------}
Begin
  clrscr;
  tenfile:='mct.scr';
  assign(fv,'c:\'+tenfile);
  rewrite(fv);
  writeln(fv,'Erase all ');
  khoitao;
  solieu(160,125,40,12,5);
  vemct(160,125,40,12,5);
  close(fv);
End;
```



Hình 5.1
Mặt cắt trục tại tiết diện rãnh then

Chú thích

(1) Sử dụng khổ bản vẽ A4, tương ứng với khổ giấy có kích thước 297 x 210;

(2) Trước khi ghi kích thước sẽ gọi tới *Procedure trangthai* để chỉ định lớp nét mảnh MA là lớp hiện hành và quy định cách ghi kích thước.

(3) Trong chương trình đã sử dụng các ký hiệu sau đây:

xt, yt : tọa độ tâm của mặt cắt trục;
d : đường kính trục tại tiết diện có rãnh then;
b : chiều rộng then;
t1 : chiều sâu rãnh then trên trục;
xcp, xct : hoành độ của các điểm bên phải và bên trái của rãnh then;
yct, ycd : tung độ các điểm trên và dưới của rãnh then trên mặt cắt ngang của trục;
ss : dung sai của chiều sâu rãnh then trên trục (tra bảng 20.6 [2]);
slt, sld : sai lệch trên và sai lệch dưới của chiều rộng rãnh then (bảng 20.6 [2]);
fv : *Text File*;
dimtext : ghi kích thước kèm theo sai lệch giới hạn cho chiều sâu rãnh then t1 có số số lẻ bằng 1;
dimver : ghi kích thước theo phương đứng kèm sai lệch giới hạn cho chiều rộng then b.

(4) Các *procedure* ghi kích thước (xem trang 133) được đặt trong *Unit* tcc.

Thí dụ 2

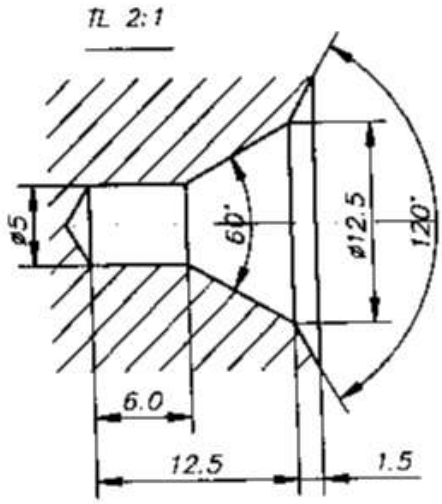
Soạn thảo chương trình thiết kế tự động lỗ tâm dùng cho các trục được gia công trên máy tiện.

Thực hiện

Nội dung cần soạn thảo ở đây bao gồm phần xác định kích thước lỗ tâm, vẽ kết cấu lỗ tâm, gạch mặt cắt, ghi kích thước danh nghĩa của chiều dài và đường kính lỗ tâm, ghi kích thước góc và cuối cùng là kẻ đường tâm. Chú ý rằng lỗ tâm là hình cắt trích ở đầu trục thường được biểu diễn với tỷ lệ lớn hơn so với tỷ lệ của bản vẽ chế tạo trục do đó trị số thực của các kích thước lỗ tâm cần được nhân với hệ số tỷ lệ. Bản vẽ lỗ tâm do máy tính thực hiện tự động theo *Program lotam* dưới đây, cho trên hình 5.2.

```
Program lotam;
Uses srt,tcc;
var  lf,x,y,xc,yc,yk : array[1..4] of real;
{-----------------------------------------}
Procedure duong(x1,y1,x2,y2:real);
begin
  writeln(fv,'Line ', x1:3:3,',',y1:3:3,' ', x2:3:3,',',y2:3:3,' ');
end;
{-------------------------------------------------}
Procedure tyler(xv,yv,kv:real;dv:string[8]);
Begin
   writeln(fv,'Style KT Simplex 2 1 15  ');
   writeln(fv,'Text ', vx:3:3,',',vy:3:3,' ',dv,' ');
   duong(xv,yv,xv+kv,yv);
end;
```

```
Procedure ltam(d,xt,yt:real;tl:byte);
{---------------------------------------}
Procedure kichthuoc;
begin
 if d<20 then (1)
 begin
  l[1]:=4.8;l[2]:=10;l[3]:=1.2;
  f[1]:=4; f[2]:=10; f[3]:=14.16;
 end else
 begin
  l[1]:=6;l[2]:=12.5;l[3]:=1.5;
  f[1]:=5; f[2]:=12.5; f[3]:=18.19;
 end;
 for i:=1 to 3 do
 begin
  l[i]:=l[i]*tl; f[i]:=f[i]*tl;
 end;
 x[0]:=xt; for i:=1 to 2 do x[i]:=xt+l[i];
 x[3]:=x[2]+l[3];
 for i:=0 to 3 do yk[i]:=yt-(16+8*i);(2)
 for i:=1 to 2 do y[i]:=yt- f[i]/2;
 xc[2]:=x[2]-f[2]*cos(π/6)/sin(π/6)/2; (3)
 xc[3]:=x[3]-f[2]*cos(π/6)/sin(π/3)/2;
 xc[4]:=x[3];
 if d<20 then
 xc[1]:=xc[2]+2/5*f[2]*cos(π/6)/sin(π/6) else
 xc[1]:=xc[2]+f[2]*cos(π/6)/sin(π/6)/2+3;
end;
{---------------------------------------}
begin
  kichthuoc;
  writeln(fv,'Layer  on LI M LI C5  ');
   for i:=0 to 2 do duong(x[i],y[i], x[i+1],y[i+1]);
   duong(xt-3,yt, xt,y[1]);
   writeln(fv,'Mirror w ',xt-25:3:3,',',yt-25:3:3,' ', x[3]+25:3:3,',',
          y[3]+25:3:3,' ', xt-25:3:3,',',yt:3:3,' ', x[3]+25:3:3,',',y[3]:3:3,' ');
   f[0]:=f[1];for i:=0 to 3 do duong(x[i],y[i], x[i],y[i]+f[i]);
   trangthai;(4)
   writeln(fv,'Layer  on MA M MA C7  ');
   writeln(fv,'Style KT Simplex 1.2 1 15  ');
   for i:=1 to 2 do dimfree('hor',xt,y[1],x[i],y[i],x[i],yk[i],f[i]/tl,1);(5)
   dimfree('hor',x[2],y[2],x[3],y[3],x[3],yk[2],f[3]/tl,1);(5)
   dimtext('ver',x[0],y[1]+f[1], x[0],y[1], x[0]-10,yt,' %%c',f[1]/tl,'');(5)
```



Hình 5.2 : Lỗ tâm

```
  dimtext('ver',x[2],y[2]+f[2], x[2],y[2], x[2]+10,y[2],' %%c' f[2]/tl,'');[5]
  writeln(fv,'Exi');
  writeln(fv,'Dimangular  ',xc[2]:3:3,',',yt:3:3,' ', xc[1]:3:3,',',yc[1]:3:3,' ',
          xc[1]:3:3,',',yc[2]:3:3,' ', x[3]-8:3:3,',',yt:3:3);
  writeln(fv,'Dimangular  ',xc[3]:3:3,',',yt:3:3,' ', xc[4]:3:3,',',yc[3]:3:3,' ',
          xc[4]:3:3,',',yc[4]:3:3,' ', x[3]+15:3:3,',',yt:3:3);
  if tl=2 then tile:= 'TL 2:1' else tile:='TL 1:1';
  tyler(xt-3,yt+25,11,tile);
  writeln(fv,'Hatch U 45 3 N  N ',xt-63:3:3,',',yt:3:3,' ',x[0]:3:3,',',
          y[0]:3:3,' x[1]:3:3,',',y[1]:3:3,' ', x[2]:3:3,',',y[2]:3:3,' ',
          x[3]- 60:3:3,',',y[3]:3:3, ' @',-(x[3]-xt-5):3:3,',0 @0,', f[3]:3:3,
          ' @',x[3]-xt+5:3:3,',0 ', x[2]:3:3,',',y[2]+f[2]:3:3,' ',
          x[1]:3:3,',',y[1]+f[1]:3:3,' ',x[0]:3:3,',',y[0]+f[1]:3:3,' C ');
  writeln(fv,'Layer S CG ');
  duong (xt-15,yt, x[3]+15,yt);
  writeln(fv,'Change L ', xt-15:3:3,',',yt:3:3,' P Lt Dashdot ');
  writeln(fv,'Ltscale 20  ');
end;
{-------------------------------------------------}
Begin
  clrscr;
  tenfile:='lt.scr';
  assign(fv,'c:\'+tenfile);
  rewrite(fv);
  writeln(fv,'Erase all ');
  khoitao;[4]
  ltam(25,150,100,2);
  writeln(fv,'viewres  10000');
  writeln(fv,'redraw');
  close(fv);
End;
```

Chú thích

[1] Suy từ bảng 13.3 [2] ;

[2] yk[i] : tung độ các vị trí ghi kích thước theo chiều dài;

[3] xc[i], yc[i]: tọa độ dùng để ghi kích thước góc qua đỉnh góc và 2 điểm trên 2 cạnh góc (bạn đọc tự tính yc[i];

[4] Sử dụng các *procedure* này như trong thí dụ 1;

[5] Các *procedure* ghi kích thước (xem trang 133) được đặt trong *Unit* tcc.

5.5. LẬP TRÌNH THIẾT KẾ TỰ ĐỘNG BẢN VẼ LẮP

5.5.1. Các nguyên tắc và giải pháp trong thiết kế

Trước khi tiến hành thiết kế bản vẽ một bộ phận máy, máy hoặc thiết bị cơ khí, người thiết kế cần nắm vững các yêu cầu sau đây:

Thiết bị cần có kích thước khuôn khổ nhỏ gọn, kết cấu hợp lý, có sự hài hoà về kích thước của các chi tiết máy và bộ phận máy, có hệ số an toàn hợp lý và đảm bảo độ tin cậy làm việc...

1) Có tính công nghệ, nghĩa là trong điều kiện sản suất hiện có, các chi tiết và bộ phận máy phải dễ chế tạo đảm bảo độ chính xác cần thiết, ít tốn nguyên vật liệu và thời gian.

2) Tháo lắp đơn giản, dễ điều chỉnh khi lắp và trong quá trình sử dụng, chọn chế độ phù hợp để bôi trơn chỗ ăn khớp và các ổ, chăm sóc, bảo dưỡng đơn giản và thuận lợi.

3) Hình dáng đẹp.

Các yêu cầu trên đây sẽ được đảm bảo nếu trong quá trình thiết kế, các nguyên tắc và giải pháp thiết kế sau đây được quán triệt:

- Thực hiện đúng nhiệm vụ thiết kế (các số liệu đầu vào). Các số liệu kỹ thuật phải được tuân thủ triệt để;

- Bố trí hợp lý các cụm kết cấu, chọn loại truyền động, khớp nối thích hợp, có thể chọn tuổi thọ làm việc như nhau hoặc khác nhau đối với các chi tiết trong máy, sử dụng cơ cấu điều khiển, định vị và điều chỉnh đơn giản;

- Lựa chọn có căn cứ vật liệu và phương pháp nhiệt luyện để chế tạo chi tiết máy và bộ phận máy, đảm bảo giảm khối lượng sản phẩm, giảm chi phí đối với các vật liệu đắt tiền và giảm giá thành kết cấu;

- Lựa chọn hợp lý khe hở giữa các chi tiết quay và đứng yên, vị trí tương đối của các chi tiết trong không gian;

- Chọn dạng công nghệ gia công các chi tiết có xét tới quy mô sản suất, phương pháp chế tạo phôi và gia công cơ;

- Lựa chọn có căn cứ các kiểu lắp, dung sai, cấp chính xác và cấp độ nhám bề mặt chi tiết có xét tới khả năng thực tế của nơi chế tạo;

- Sử dụng rộng rãi các tiêu chuẩn nhà nước, tiêu chuẩn ngành, địa phương và tiêu chuẩn xí nghiệp;

- Thực hiện sự thống nhất hoá trong thiết kế, tức là sử dụng với số lượng tối đa có thể các chi tiết máy và bộ phận máy có cùng quy cách kích thước, cùng loại vật liệu và loại phôi, nhờ đó sẽ giảm được thời gian và giá thành thiết kế chế tạo sản phẩm.

Trên cơ sở quán triệt các yêu cầu về kết cấu, người thiết kế nên phác thảo một số phương án kết cấu, so sánh và chọn ra phương án thích hợp, sau đó mới tiến hành lập trình thiết kế tự động bản vẽ lắp.

5.5.2. Các bước tiến hành

Lập trình thiết kế tự động bản vẽ lắp được tiến hành theo các bước

sau đây:

1. Gọi số liệu từ các *File* kết quả ở chương trình tính toán thiết kế các chi tiết máy và bộ phận máy. Đối với hộp giảm tốc đó là các thông số như: khoảng cách trục hoặc chiều dài côn, kích thước các bánh răng, bánh đai, đĩa xích, đường kính và chiều dài các đoạn trục, loại ổ lăn và kích thước ổ.

2. Dựa vào các số liệu từ *File* kết quả và tỷ lệ đã chọn, xác định lại các thông số và kích thước đã có (nhân với hệ số tỷ lệ), sau đó tính toán bổ sung kích thước của các yếu tố kết cấu bánh răng, bánh vít, bánh đai, đĩa xích (xem mục 5.4.1) và xác định các tọa độ gốc và tọa độ tương đối trên bản vẽ.

3. Định khổ bản vẽ với chú ý là bản vẽ lắp thường được vẽ trên khổ giấy A0 hoặc A1 theo một tỉ lệ xác định phụ thuộc vào kích thước của kết cấu cần thể hiện. Ở đây chính là các kích thước khuôn khổ của bộ phận máy gồm chiều dài L, chiều rộng B và chiều cao H được xác định theo các thông số cơ bản, đối với hộp giảm tốc đó là khoảng cách trục hoặc chiều dài côn, đường kính các bánh răng, bánh đai, đĩa xích và chiều dài các trục.

4. Soạn thảo các chương trình con tính toán và vẽ kết cấu các chi tiết, chẳng hạn, đối với hộp giảm tốc bao gồm các *Procedure* vẽ thân và nắp hộp, vẽ bánh răng, trục và rãnh then trên trục, ổ lăn, bạc chặn và kết cấu các chi tiết lắp vào vỏ hộp như: nắp ổ và lót kín, bulông vòng, các loại bulông và đai ốc, nút tháo dầu, thăm dầu, nắp kiểm tra, nút thông hơi. Các chương trình con này nên được xây dựng theo phương pháp truyền tham số ứng với một số kiểu kết cấu xác định, khi gọi ra để sử dụng, kiểu sẽ được chọn và các tham trị sẽ được thay đổi cho phù hợp với từng trường hợp thiết kế cụ thể. Vì chương trình vẽ tự động bản vẽ lắp rất dài, chiếm bộ nhớ lớn hơn bộ nhớ tối thiểu của máy tính, do đó sau khi gỡ rối hãy biến các chương trình con thành các *Unit* hoặc *Unit Overlay* để sử dụng khi tiến hành vẽ từng cụm kết cấu trong chương trình chính.

5. Soạn thảo các *Unit* dùng chung cho các bản vẽ lắp gồm:

- Khung tên theo TCVN 3821-83 ;
- Đánh số thứ tự chi tiết trên các hình chiếu và trên hình cắt, với đường gióng bằng nét mảnh, nối từ vị trí đánh số chi tiết tới các đoạn gạch đậm (nét liền) được bố trí theo những cột đứng hoặc nằm ngang, trên đó ghi các số thứ tự chi tiết (chú ý rằng con số phải được viết bằng khổ lớn hơn và nét đậm hơn so với con số ghi kích thước lắp ghép và kích thước khuôn khổ);
- Kẻ bảng thống kê chi tiết;

- Ghi đặc tính kỹ thuật của kết cấu và ghi ký hiệu các mặt cắt.

6. Gọi tới các *Unit* kết cấu đã được soạn thảo (mục 3.5.2.4) để vẽ lần lượt các chi tiết trong từng cụm kết cấu tại những vị trí xác định trên hai hoặc ba hình chiếu cùng các hình cắt trích, nếu cần thể hiện thêm các kết cấu phức tạp. Cần lưu ý chọn hình chiếu chính để biểu diễn các kết cấu bên trong: đối với hộp truyền động có các trục được phân bố trên cùng một mặt phẳng nên chọn hình chiếu bằng làm hình chiếu chính, còn đối với các hộp có trục chéo nhau như hộp giảm tốc trục vít, nên chọn hình chiếu đứng làm hình chiếu chính. Các hình chiếu còn lại được dùng để thể hiện kết cấu và vị trí tương quan trong không gian của các chi tiết và cụm kết cấu. Như vậy ngoài việc phải xác định chính xác vị trí cần thể hiện kết cấu của các chi tiết và cụm kết cấu, còn phải viết bổ sung các đoạn chương trình nhằm xác định kích thước và thể hiện các yếu tố của thân máy hoặc vỏ hộp ghép với các kết cấu này trên các hình chiếu của bản vẽ.

Sau khi vẽ xong từng cụm kết cấu, nên tiến hành ngay việc ghi kích thước lắp ghép cũng như kẻ đường tâm cho cụm kết cấu đó với chú ý rằng, kích thước danh nghĩa phải là kích thước thực, tức là phải chia cho hệ số tỷ lệ đã chọn khi xác định các thông số và kích thước để vẽ.

7. Ghi kích thước khuôn khổ, kích thước lắp đặt (kích thước liên kết bản vẽ lắp với khung hoặc bệ máy) và các trị số dung sai cần thiết (thí dụ, dung sai khoảng cách trục trong hộp giảm tốc bánh răng và hộp giảm tốc trục vít). Chú ý rằng, các kích thước danh nghĩa này đều là kích thước thực và tất cả các con số biểu thị kích thước phải có cùng độ cao, cùng chiều dày nét và phải nhỏ hơn con số chỉ thứ tự các chi tiết trên bản vẽ.

8. Gọi đến các *Unit* dùng chung để tiến hành đánh số thứ tự các chi tiết trong bản vẽ lắp, vẽ và điền vào khung tên, kẻ và điền vào bảng thống kê chi tiết, ghi ký hiệu các mặt cắt và đặc tính kỹ thuật của kết cấu.

9. Trong trường hợp cần thiết, trên bản vẽ lắp còn ghi điều kiện kỹ thuật của kết cấu. Chẳng hạn, đối với hộp giảm tốc bánh răng côn - trụ hai cấp dùng trong máy nổ và toa xe điện có thể ghi các nội dung sau đây:

- Khe hở cạnh răng khi ăn khớp: đối với cặp bánh răng côn – không nhỏ hơn 0,2 mm, đối với cặp bánh răng trụ - không nhỏ hơn 0,15 mm;
- Vết tiếp xúc : đối với răng bánh răng côn – theo chiều dài răng và theo chiều cao răng không nhỏ hơn 60% ; đối với răng bánh răng trụ – theo chiều dài răng không nhỏ hơn 60%, theo chiều cao răng không nhỏ hơn 45%;
- Các ổ đũa côn được điều chỉnh khi lắp nhờ đai ốc và căn đệm;
- Bôi một lớp sơn mỏng hoặc một lớp thủy tinh nước lên bề mặt ghép

giữa nắp và thân hộp;

- Sau khi lắp xong, cho hộp giảm tốc chạy rà cho tới khi các bộ truyền bánh răng làm việc êm, tiếng ồn đều (trung bình khoảng 2 giờ).

10. Đặt tên *File* và ghi vào *File* các số liệu phục vụ thiết kế bản vẽ chung, bao gồm các số liệu liên quan đến kích thước khuôn khổ (chiều dài L, chiều cao H và chiều rộng B của hộp), các kích thước liên kết (chiều dày đế hộp, khoảng cách giữa các lỗ bulông, chiều cao từ mặt dưới đế hộp đến tâm trục) và kích thước lắp ghép (đường kính danh nghĩa và kiểu lắp của đầu trục) cũng như các thông số của bộ truyền ngoài gồm đường kính và chiều rộng bánh đai, đường kính và số răng đĩa xích.

5.5.3. Thí dụ

Lập trình thiết kế tự động bản vẽ lắp hộp giảm tốc trục vít - bánh răng trong hệ dẫn động băng tải có sơ đồ vẽ trên hình 4.1. Dữ liệu của đề thiết kế lấy từ *File* kết quả (khoảng cách trục a_{w1}= 135 mm; a_{w2}= 220 mm v.v...), đã được ghi vào *File* khi thực hiện chương trình tính toán ở chương 4, mục 4.3 ứng với các số liệu cho trong thí dụ này.

Thực hiện

```
{$O+,F+}
Program velap;
Uses Overlay,Crt,Graph,tb,kht, solict,tcb,ctg,cth,vtr,vbr,hcd,hcc;(1)
{$O ctg}  {$O cth} {$O vtr} {$O vbr}
{---------------------------------------}
Procedure KiemtraOverlay;
begin
  OvrInit ('velap . OVR');
  If OvrResult < > OvrOk then
  begin
    writeln(' Loi Overlay');
    halt;
  end;
end;
{---------------------------------}
Begin
  KiemtraOverlay;
  Clrscr;
  slvehop (2);
  assign(fv,'C:\'+tenfile);
  rewrite(fv);
  writeln(fv,'Erase all ');
  khobanve(Lh);(3)
  napkin(2,1,xa[2],dn[2],d[2,0],hp[2],hv[2]);
```

```
    goiolan;
    lobulong (db[3],db[2]);
    truc;
    banhrang;
    vohop;
    khungten(tensv,tenlop,'HOP GIAM TOC TV-BR', '01-05-03',' ');
    danhsoct;
    bangke;
    writeln(fv,'redraw');
    writeln(fv,' viewres  10000');
    writeln(' Ten ban ve lap  thv= ', tenfile);
  close(fv);
End.
```

Chú thích

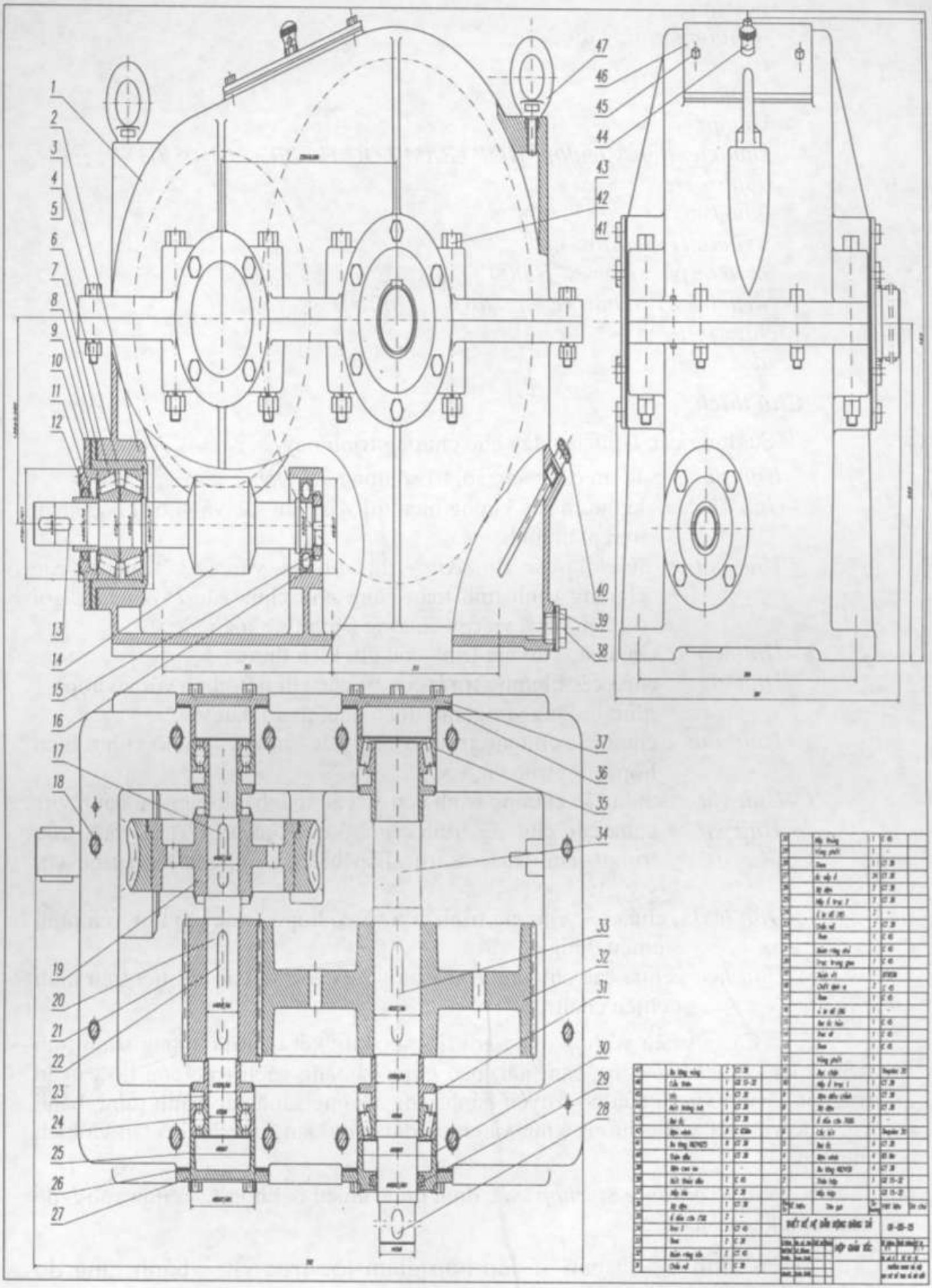
(1) Sử dụng các *Unit* sau đây cho chương trình này :

- *Unit th* : để tra các bảng số, trị số dung sai, v.v...
- *Unit kht* : để tạo ra các khung hiển thị với màu sắc và vị trí khác nhau trên màn hình;
- *Unit solict*: để chứa các *Procedure* ghi dữ liệu và các kết quả từ các chương trình tính toán cũng như chứa các *Procedure* gọi dữ liệu phục vụ các chương trình tính toán và vẽ;
- *Unit tcb* : chứa các chương trình con ghi kích thước;
- *Unit ctg* : chứa các chương trình con vẽ các chi tiết ghép với vỏ hộp như: bulông vòng, nút tháo dầu, thăm dầu v.v...
- *Unit cth* : chứa các chương trình con vẽ các chi tiết của hộp như: bích hộp, cụm trục vít, v.v...
- *Unit vbr* : chứa các chương trình con vẽ các loại bánh răng và bánh vít;
- *Unit vtr* : chứa các chương trình con vẽ các loại trục (trục vào, trục trung gian, trục ra, trục liền bánh răng, trục liền trục vít, v.v...);
- *Unit hcd* : chứa các chương trình con vẽ vỏ hộp và các chi tiết trên hình chiếu đứng;
- *Unit hcc* : chứa các chương trình con vẽ vỏ hộp và các chi tiết trên hình chiếu cạnh;

(2) Các số liệu vẽ hộp được gọi từ các *File* kết quả ở chương trình tính toán thiết kế hệ dẫn động băng tải, bao gồm: khoảng cách trục của bộ truyền trục vít - bánh vít và của bộ truyền bánh răng; đường kính các bánh răng, bánh vít, trục vít, đĩa xích; đường kính và chiều dài các đoạn trục; loại ổ lăn và kích thước ổ.

(3) Từ *Procedure slvehop* xác định được chiều dài hộp *L* và dựa vào *L* để xác định khổ bản vẽ.

Trên hình 5.3 là bản vẽ lắp hộp giảm tốc trục vít – bánh răng do máy tính thực hiện theo chương trình vẽ tự động (*Program velap*) ở trên.



Hình 5.3 : Bản vẽ lắp hộp giảm tốc trục vít – bánh răng

5.6. LẬP TRÌNH THIẾT KẾ TỰ ĐỘNG BẢN VẼ CHUNG

Bản vẽ chung được hoàn thành vào giai đoạn cuối cùng của quá trình thiết kế cơ khí nhằm biểu diễn toàn bộ hệ thống thiết bị, vị trí tương đối giữa các bộ phận máy trong hệ thống và quan hệ giữa chúng với nhau.

5.6.1. Một số vấn đề cần lưu ý khi thiết kế bản vẽ chung

Ngoài việc thể hiện hình dáng bên ngoài, tương quan giữa các bộ phận máy và các yếu tố kết cấu cũng như nguyên tắc làm việc của sản phẩm, bản vẽ chung còn cần thể hiện nguyên tắc lắp đặt và các đặc tính kỹ thuật của thiết bị. Muốn vậy khi thiết kế bản vẽ chung cần đảm bảo những nội dung và yêu cầu sau đây:

1. Bản vẽ chung cần được biểu diễn trên hai hoặc ba hình chiếu với tỷ lệ thu nhỏ 1:2; 1:2,5; 1:4; 1:5; 1:10, đồng thời khi cần thiết, dùng thêm các hình cắt trích cũng như biểu diễn sơ đồ động của hệ thống. Các yếu tố của bản vẽ chung không yêu cầu thể hiện quá tỉ mỉ nhưng cần rõ ràng và chính xác. Chẳng hạn không cần biểu diễn các bu lông ghép nắp và thân hộp giảm tốc nhưng các bu lông dùng để cố định các bộ phận máy trên nền thì cần được thể hiện rõ ràng. Chú ý chọn vị trí tương quan hợp lý nhất để biểu diễn động cơ điện, hộp giảm tốc hoặc hộp tốc độ và các bộ phận máy khác đồng thời tìm vị trí thuận lợi để liên kết tốt nhất với khung hoặc bệ máy.

2. Bản vẽ chung cần đảm bảo yêu cầu lắp ghép, tức là một mặt cần đảm bảo khả năng lắp ghép chính xác động cơ và các bộ phận máy với khung hoặc bệ máy, chỗ nối giữa các đầu trục, v.v...Tuy nhiên không yêu cầu mô tả kỹ về kết cấu cũng như cách bố trí mặt ghép nối hoặc đường bao chỗ ghép với chuẩn đo.

3. Trên bản vẽ chung cần ghi những kích thước sau đây:

- Kích thước khuôn khổ, liên quan đến nhà xưởng phục vụ lắp đặt thiết bị.

- Kích thước lắp đặt và kích thước kiểm tra độ chính xác lắp đặt, bao gồm các kích thước phục vụ việc lắp ghép các đơn vị lắp, khe hở cần thiết giữa hai nửa khớp nối (dùng để kiểm tra vị trí tương đối giữa các đơn vị lắp theo phương dọc trục).

- Kích thước liên kết, xác định vị trí của các bộ phận lắp và mối liên hệ giữa chúng với nhau, bao gồm chiều cao tâm trục tính đến mặt tựa của khung hoặc bệ máy, tọa độ phân bố các lỗ bulông ghép các bộ phận máy vào khung, tọa độ các lỗ để cố định giá đỡ trục, các thông số của bộ truyền ngoài (số răng đĩa xích và bước xích, đường kính và chiều rộng bánh đai), tức là các kích thước liên quan đến khung và bệ máy. Ngoài ra,

để có cơ sở thiết kế móng cần ghi đường viền trên mặt bằng, khoảng cách từ bệ máy đến nền, tọa độ các lỗ bulông nền, chiều dày đế máy (để xác định chiều dài bulông nền) và kích thước phần bê tông đặt bulông nền.

4. Trên bản vẽ chung cần đánh số các đơn vị lắp và ghi vào bảng kê đặt phía trên khung tên, trong đó ghi: số lượng, phương thức cung ứng, chẳng hạn đối vớichi tiết máy - ghi "vật liệu", đối với động cơ - ghi chữ "mua", đối với hộp truyền động - ghi chữ "lắp sẵn", đồng thời cần ghi đặc tính kỹ thuật của thiết bị, bao gồm công suất và số vòng quay của động cơ, lực kéo hoặc mômen xoắn và số vòng quay trên trục ra, đặc tính truyền động (đường kính đĩa xích, góc nghiêng của xích so với mặt bằng, v.v...)

5.6.2. Các bước tiến hành

Lập trình thiết kế tự động bản vẽ chung được tiến hành theo các bước sau đây:

1. Gọi số liệu từ *File* dữ liệu chứa các kết quả thu được ở chương trình thiết kế bản vẽ lắp các bộ phận máy. Đó là các số liệu liên quan đến kích thước khuôn khổ (chẳng hạn chiều dài L, chiều cao H và chiều rộng B của hộp truyền động), các kích thước liên kết (chiều dày đế hộp, khoảng cách giữa các lỗ bu lông, chiều cao từ mặt dưới đế hộp đến tâm trục), kích thước lắp ghép (đường kính danh nghĩa và kiểu lắp của đầu trục) và các thông số của bộ truyền ngoài được xác định khi tính toán thiết kế các bộ truyền, gồm đường kính và chiều rộng bánh đai, đường kính và số răng đĩa xích.

2. Sử dụng các *Unit* tra bảng để tra các số liệu cần cho thiết kế bản vẽ chung và đã được nạp vào đĩa như: các kích thước lắp đặt động cơ, các thông số và kích thước của khớp nối, quy cách và các kích thước của thép cán định hình phục vụ thiết kế khung máy, kích thước của các bulông, đai ốc, đệm vênh và đệm xiên ghép các bộ phận máy vào khung thép định hình, kích thước của bu lông nền, v.v...

3. Định khổ bản vẽ với chú ý là bản vẽ chung thường được vẽ trên khổ giấy A0 theo một tỷ lệ thu nhỏ (1:2; 1:2,5; 1:4; 1:5; 1:10), phụ thuộc vào kích thước của thiết bị cần thể hiện.

4. Dựa vào các số liệu đã gọi từ *File* dữ liệu, các kích thước tra được từ các bảng và tỷ lệ đã chọn, xác định lại các thông số và kích thước đã có (nhân với hệ số tỷ lệ), sau đó tiến hành tính toán bổ sung các thông số kết cấu, các tọa độ gốc và tọa độ tương đối.

5. Soạn thảo các chương trình con để vẽ tự động kết cấu của các bộ phận máy và chi tiết máy trên các hình chiếu, bao gồm: động cơ điện,

khớp nối, hộp giảm tốc hoặc hộp tốc độ, các bộ truyền ngoài (truyền động đai, truyền động xích), gối đỡ ở các trục truyền, tang quay, khung và bệ máy, các chi tiết kẹp chặt, v.v...

6. Ghi kích thước khuôn khổ, kích thước lắp đặt và kích thước liên kết. Chú ý rằng, các kích thước danh nghĩa này đều là kích thước thực, tức là chúng cần được chia cho hệ số tỷ lệ (đã chọn khi vẽ kết cấu) trước khi tiến hành ghi tự động các loại kích thước trên bản vẽ.

7. Gọi đến các *Unit* dùng chung để tiến hành đánh số thứ tự các chi tiết trên bản vẽ chung, vẽ và điền vào khung tên, kẻ và điền vào bảng thống kê các bộ phận máy và chi tiết máy, ghi ký hiệu các mặt cắt, ghi vào bảng số liệu biểu thị đặc tính kỹ thuật của thiết bị.

8. Trường hợp cần thiết, có thể soạn thảo các chương trình con để vẽ tự động sơ đồ động của hệ thống và sơ đồ thiết bị điều khiển trên bản vẽ chung.

5.6.3. Thí dụ

Lập trình thiết kế tự động bản vẽ chung hệ dẫn động băng tải có sơ đồ động được vẽ trên hình 4.1 và bản vẽ lắp hộp giảm tốc trục vít – bánh răng được thể hiện trên hình 5.3. Dữ liệu của để thiết kế lấy từ các số liệu đã được ghi vào *File* kết quả khi thực hiện chương trình tính toán ở thí dụ 5.5.3.

Thực hiện :

```
{$O+,F+}
Program bvchung;
Uses Overlay,Crt,graph,tb,kht, solict,tcb,kht,hcb,hcd;(1)
{$O hcb}  {$O hcd}
{------------------------------}
procedure KiemtraOverlay;
begin
  OvrInit ('bvchung OVR');
  If OvrResult < > OvrOk then
  begin
    writeln(' Loi Overlay');
    halt;
  end;
end;
{------------------------------}
Begin
  kiemtraOverlay;
  slvehedd(2);
  tenfile:='bvchung.scr';
```

```
    assign(fv,'C:\'+tenfile);
    rewrite(fv);
    writeln(fv,'Erase all ');(3) ;
    khobanve(L);(4)
    khung;
    hop;
    dongco;
    khopnoi;
    tangquay;
    otruot;
    btxich;
    writeln(fv,'redraw');
    Close(fv);
End;
```

Chú thích

(1) Ngoài các *Unit* có sẵn như *Crt, Graph và Overlay*, trong chương trình còn sử dụng các *Unit tb, kht, tcb, kht, solict, hcb, hcd*, trong đó *Unit hcb* và *Unit hcd* chứa các chương trình con dùng để:

- Vẽ toàn bộ các kết cấu của động cơ điện, khớp nối, tang quay, ổ trượt, bộ truyền xích con lăn;

- Ghi các kích thước, kẻ khung tên và bảng kê các chi tiết; ghi điều kiện kỹ thuật, đặc tính của thiết bị cũng như đánh số chi tiết;

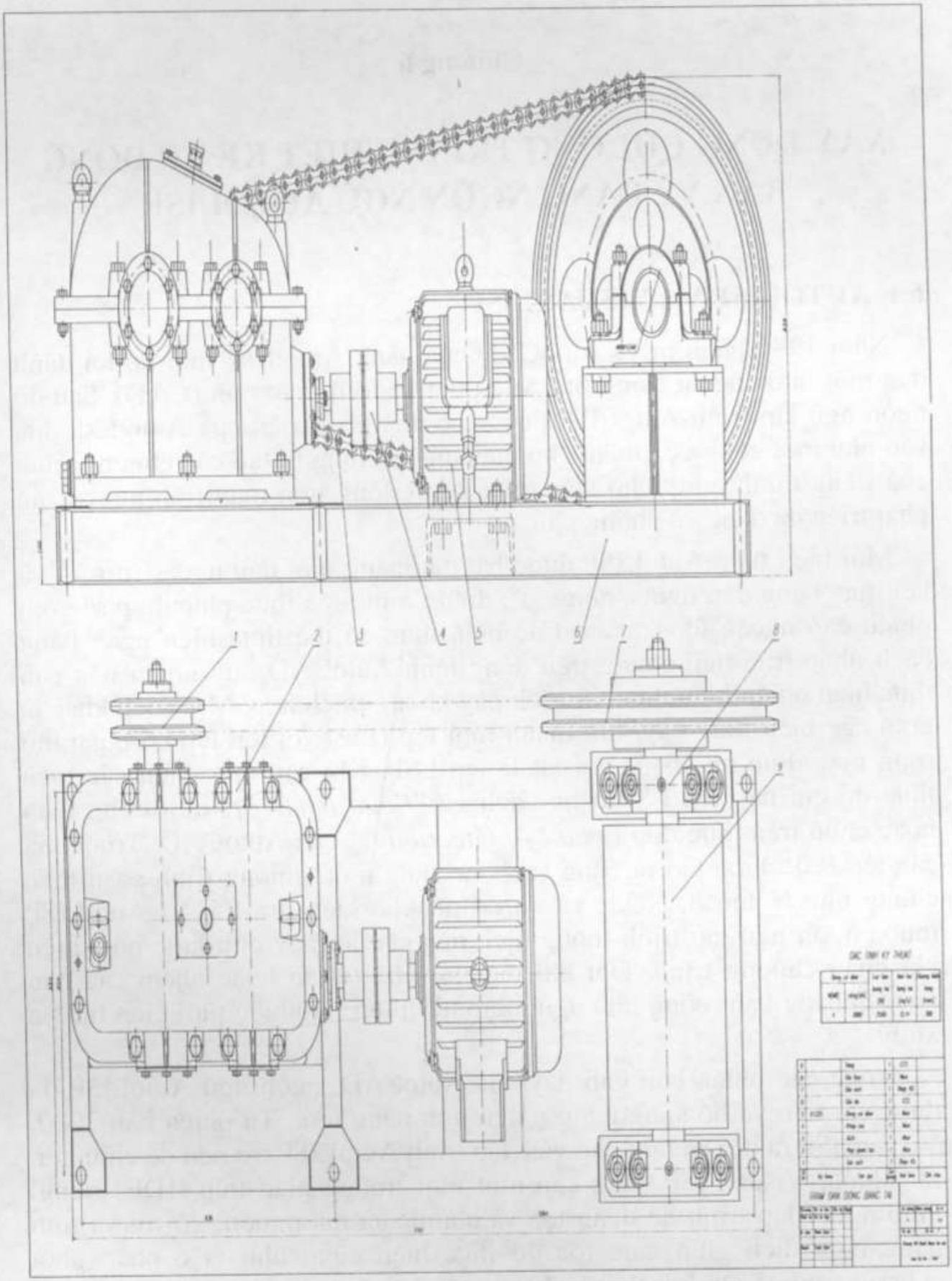
Các nội dung này được thực hiện trên hai hình chiếu bằng và hình chiếu đứng. Đó là hai *Unit* có dung lượng rất lớn, vì vậy chúng được chuyển thành các *Unit Overlay* do đó ở dòng thứ tư của chương trình còn ghi thêm các ký hiệu *{$O hcb} {$O hcd}*.

(2) *Slvehedd* là chương trình con gọi từ *Unit solict* liên quan đến các kích thước khuôn khổ của hộp truyền động (được xác định khi thiết kế bản vẽ lắp) cũng như của động cơ điện, khớp nối, tang quay, ổ trượt và gối đỡ ổ trượt, các kích thước liên kết và phần tính toán bổ sung các kích thước cần thiết khác.

(3) Xoá sạch màn hình trước khi tiến hành vẽ.

(4) Dựa theo kích thước chiều dài của động cơ điện, khớp nối, hộp giảm tốc và các bộ phận máy khác sẽ tính được chiều dài của thiết bị *L* và dựa vào *L* để xác định khổ bản vẽ và tỉ lệ của bản vẽ.

Trên hình 5.4 là bản vẽ chung của hệ dẫn động băng tải do máy tính thực hiện theo chương trình vẽ tự động bản vẽ chung đã được soạn thảo trên đây (*Program bvchung*).



Hình 5.4 : Bản vẽ chung hệ dẫn động băng tải sử dụng hộp giảm tốc trục vít – bánh răng

Chương 6

XÂY DỰNG CHƯƠNG TRÌNH THIẾT KẾ TỰ ĐỘNG BẢN VẼ BẰNG NGÔN NGỮ AUTOLISP

6.1. AUTOLISP VÀ VISUALLISP

Năm 1982 phần mềm AutoCAD của hãng Autodesk Inc. ra đời đánh dấu một bước ngoặt trong lĩnh vực thiết kế trên máy tính (CAD). Sau đó ngôn ngữ lập trình AutoLISP được các nhà phát triển của Autodesk đưa vào như một công cụ chuẩn cho phép người dùng tự tạo các chương trình của riêng mình, giúp cho việc thiết kế tự động trên máy tính ngày càng phát triển đa dạng và phong phú.

Mỗi biểu thức AutoLISP được bắt đầu bằng một dấu ngoặc mở "(" và kết thúc bằng dấu ngoặc đóng ")", do đó một biểu thức phức hợp sẽ gồm nhiều cặp ngoặc lồng nhau. Các biểu thức có thể thực hiện ngay bằng cách nhập trực tiếp chúng trên dòng lệnh AutoCAD, tuy nhiên nếu cần thực hiện nhiều biểu thức thì cách này không thích hợp. Một cách khác là gom các biểu thức này, lưu thành một tệp (file) với tên tệp có phần mở rộng mặc định là ".lsp", gọi tắt là tệp LSP. Khi cần thực hiện các biểu thức đó chỉ nạp tệp LSP bằng cách gõ "*(load tên_tệp)*" trên dòng lệnh hoặc chọn trên thực đơn *Load / Application /...* của AutoCAD. Trước kia các tệp LSP được tạo ra bằng cách sử dụng một chương trình soạn thảo chung như Notepad, NC,... và do đó rất khó kiểm tra lỗi. Vì AutoLISP thuộc ngôn ngữ lập trình thông dịch nên các lỗi này chỉ được phát hiện khi chạy chương trình. Đôi khi một lỗi nhỏ (thiếu hoặc nhầm các dấu ngoặc chẳng hạn) cũng làm người lập trình tốn rất nhiều thời gian tìm và xử lý.

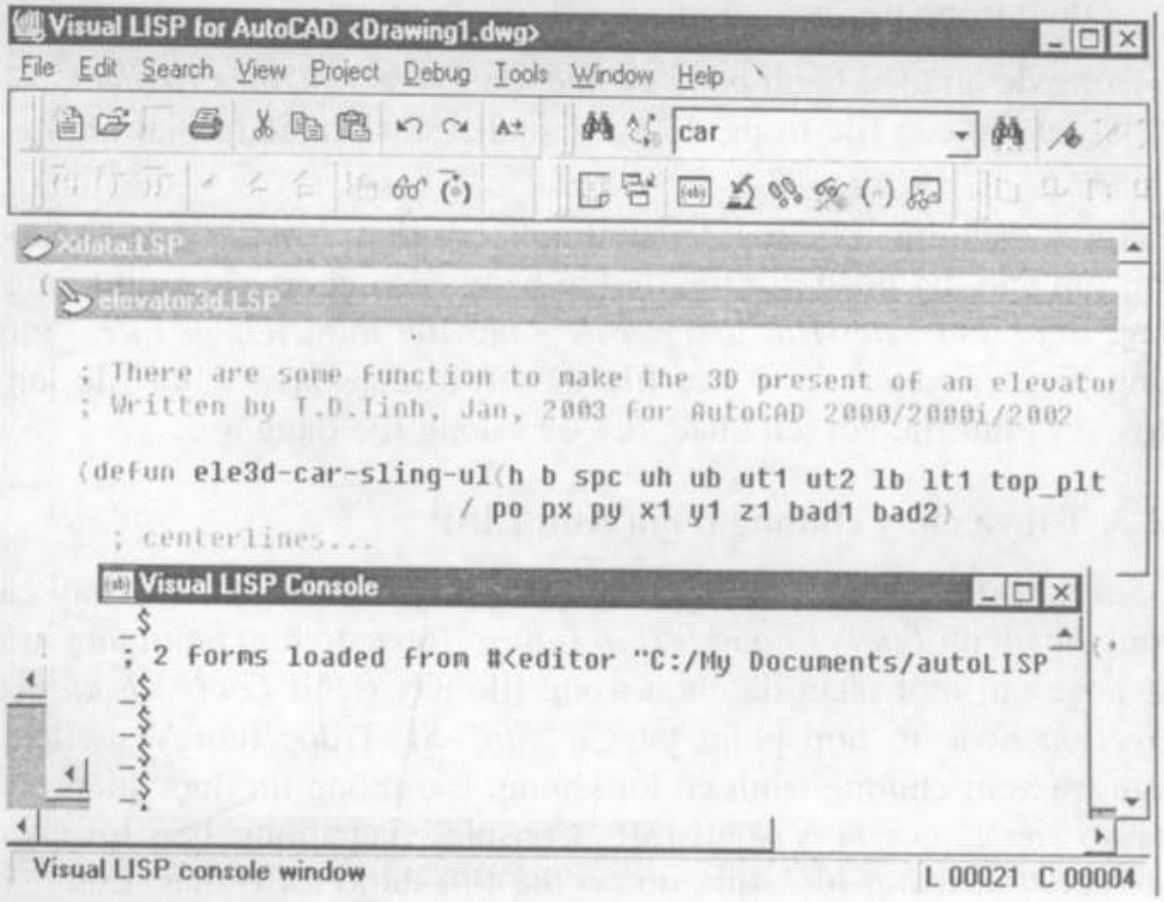
Trong các phiên bản gần đây của AutoCAD, ngôn ngữ AutoLISP đã được cải tiến và bổ sung thêm nhiều tính năng mới. Từ phiên bản 2000, VisualLISP ra đời đã làm cho việc lập trình AutoLISP trở nên dễ chịu hơn rất nhiều. VisualLISP cung cấp một môi trường giao tiếp (IDE) riêng, giúp người lập trình dễ dàng tạo và chỉnh sửa mã nguồn, gỡ rối và tính năng biên dịch giúp tăng tốc độ thực hiện cũng như việc phân phối chương trình thuận lợi, thống nhất và tránh mất mát mã nguồn. Việc nạp chương trình cũng có thể thực hiện ngay trong môi trường VisualLISP làm cho việc kiểm tra chương trình nhanh hơn. Dưới đây giới thiệu một số điểm cơ bản về môi trường VisualLISP.

6.1.1. Khởi động VisualLISP

VisualLISP được khởi động từ AutoCAD bằng một trong các cách sau:

- Gõ lệnh "*VLIDE*" hoặc lệnh "*VLISP*" trên dòng lệnh AutoCAD.
- Chọn từ menu *Tools / AutoLISP / VisualLISP Editor* của AutoCAD.

Môi trường giao tiếp VisualLISP hiện ra như trên hình 6.1.



Hình 6.1: Màn hình VisualLISP

6.1.2. Quản lý chương trình VisualLISP

Để bắt đầu một đề án mới hãy chọn thực đơn *Project / New Project*, sau đó lựa chọn các file cần thiết cho đề án này. Để mở một đề án đã có hãy chọn *Project / Open Project*. Các chương trình AutoLISP nhìn chung chứa nhiều file và nhiều dòng mã lệnh. Sau khi thực hiện việc biên dịch sẽ xuất hiện một loạt file FAS. Khi có sự thay đổi mã nguồn, việc chọn các file nào để biên dịch lại là một thách thức với người lập trình. AutoLISP cung cấp một phương tiện đơn giản và hiệu quả giúp quản lý các chương trình lớn gọi là đề án (project). Một đề án VisualLISP chứa danh sách các file nguồn và cách thức biên dịch các file này. Việc quản lý các file qua đề án sẽ đảm bảo tính toàn vẹn của chương trình, thể hiện qua các điểm sau:

- Kiểm tra các file nguồn LSP trong đề án xem chúng có thay đổi không, cho phép chỉ biên dịch lại các file thay đổi.
- Việc truy cập vào các file trong đề án đơn giản hơn, thông qua một lần kích chuột vào tên file trong bảng kê các file trong đề án.
- Có thể tìm kiếm một biểu thức trong các file khác nhau trong đề án.
- Tối ưu mã biên dịch bằng cách liên kết trực tiếp với thành phần cần thiết trong đề án.

Đóng đề án hiện hành bằng cách chọn *Project / Close Project*. Nếu có sự thay đổi ở các file trong đề án VisualLISP sẽ nhắc lưu lại những thay đổi này.

Ngoài cách quản lý qua đề án như đã nói trên, các file đơn lẻ được tạo mới, lưu vào đĩa hoặc mở file đã lưu bằng cách chọn các lệnh tương ứng trong thực đơn *File* (*File /New File* - tạo file mới; */Open File* - mở file trong đĩa; */Reopen* - mở các file đã mở trước; */Save* - lưu file lên đĩa; */Save As* - lưu file với tên khác; */Close* - đóng file đang mở...)

6.1.3. Tải và chạy chương trình AutoLISP

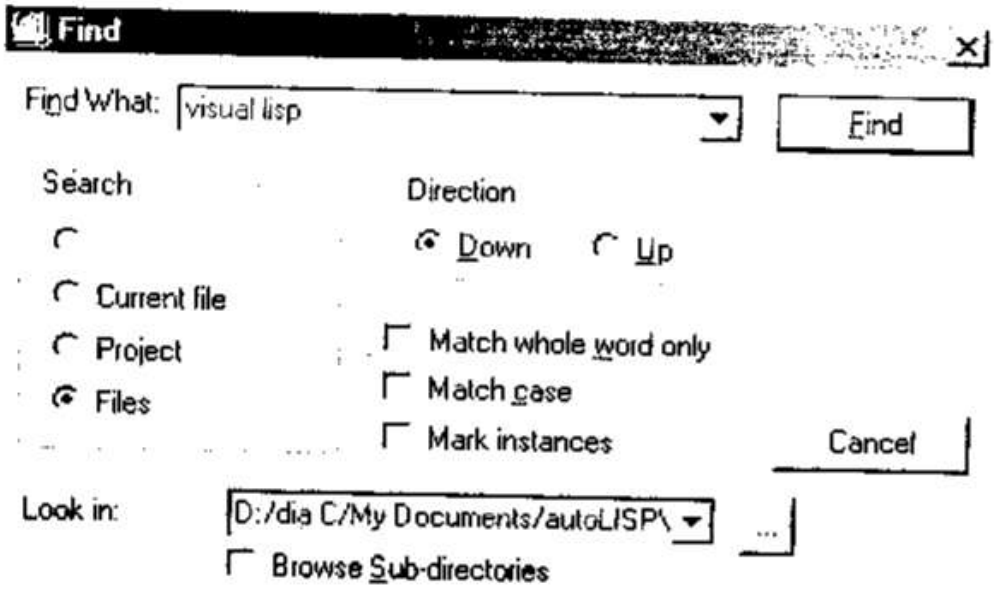
Sau khi đã mở, file chương trình AutoLISP có thể tải và chạy cả file thông qua lệnh *Tools / Load text in Editor* (hoặc tổ hợp phím tắt *Ctrl-Alt-E*) hoặc chỉ một phần đã chọn trong file này (lệnh *Tools / Load text in Selection* hoặc tổ hợp phím tắt *Ctrl-Shift-S*). Trước tiên VisualLISP sẽ kiểm tra xem chương trình có lỗi không, các thông tin được hiển thị trên cửa sổ riêng, gọi là VisualLISP Console. Các thông báo lỗi có dạng *";error: tình trạng lỗi"*, qua đó có thể tìm được lỗi ở đâu. Lưu ý rằng, VisualLISP chỉ phát hiện các lỗi cú pháp, còn các lỗi khác như thuật toán sai, các thông số gửi qua lệnh AutoCAD không đúng trình tự, hoặc không đủ,... sẽ chỉ được phát hiện khi chạy chương trình.

6.1.4. Các chức năng khác của môi trường VisualLISP

6.1.4.1. Tìm kiếm và thay thế

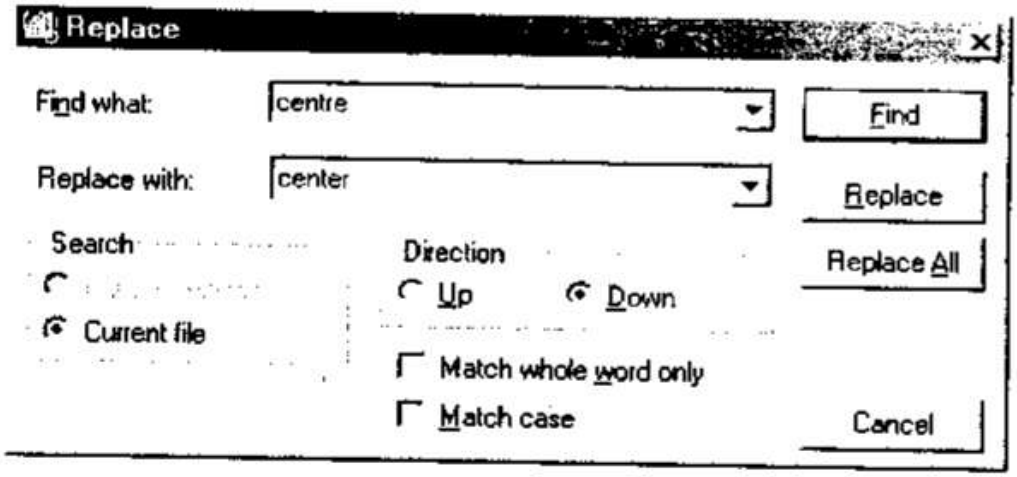
Để tìm kiếm một chuỗi văn bản nào đó, hãy chọn thực đơn *Search / Find*. Hộp thoại Find hiện ra như hình 6.2. Gõ nội dung cần tìm vào hộp *Find what* (ví dụ, cần tìm chuỗi "visual lisp"). Trong mục *Direction* chọn chiều tìm kiếm (Down - tìm từ vị trí con trỏ về cuối; Up - tìm từ vị trí con trỏ về đầu). Mục *Search* để chọn vùng cần tìm (Current selection - tìm trong phần đã chọn (bôi đen); Curent file - tìm trong file hiện hành; Project - tìm trong đề án; Files - tìm trong các file khác - khi đó chọn mục Look in để chỉ định các file sẽ tiến hành việc tìm kiếm). Các lựa

chọn khác thể hiện cách tìm kiếm văn bản: đánh dấu vào hộp *Match whole word only* để tìm kiếm cả từ (chẳng hạn, nếu văn bản cần tìm là *lisp* thì các chữ như vlisp, visuallisp... sẽ không được xét đến, trong khi nếu ô này không được chọn thì các chữ trên cũng được thể hiện trong kết quả). Hộp *Match case* chỉ định việc tìm kiếm có phân biệt chữ hoa hay chữ thường (nếu hộp này được đánh dấu thì các từ Lisp và lisp sẽ là 2 từ khác nhau, ngược lại nếu hộp không được đánh dấu thì VisualLISP sẽ coi hai từ này như nhau). Hộp *Mark instances* đánh dấu vị trí văn bản được tìm thấy, cho phép quay lại vị trí này một cách dễ dàng. Sau khi khai báo xong, nhấn nút *Find* để thực hiện tìm kiếm.



Hình 6.2: Tìm kiếm văn bản

Trường hợp tìm kiếm trong vùng chọn hoặc trong file hiện hành con trỏ văn bản sẽ di chuyển đến vị trí nơi văn bản được tìm thấy. Để di chuyển đến vị trí tìm thấy kế tiếp hãy nhấn phím F3.



Hình 6.3: Thay thế văn bản

Để thay thế chuỗi văn bản bằng chuỗi mới - chọn *Search / Replace*. Hộp thoại *Replace* hiện ra như hình 6.3. Gõ nội dung cần thay thế vào hộp *Find what*, nội dung mới vào hộp *Replace with*. Ví dụ, trên hình 6.3 sẽ thay thế chuỗi ký tự "centre" bằng chuỗi mới "center". Các lựa chọn khác trong mục Search, Direction,... giống như việc tìm kiếm văn bản. Sau khi khai báo xong, kích vào nút *Find* để tìm kiếm, *Replace* để thay thế văn bản vừa tìm thấy, *Replace all* để tự động thay thế tất cả các văn bản tìm được bằng nội dung mới.

6.1.4.2. Mã màu VisualLISP

Trong cửa sổ soạn thảo chương trình VisualLISP, các thành phần khác nhau của ngôn ngữ được gán các màu riêng. Điều này cho phép dễ dàng phát hiện lỗi (gõ sai tên hàm chuẩn hoặc thiếu dấu ngoặc kép trong chuỗi ký tự...). Màu mặc định của các thành phần ngôn ngữ như sau:

Màu	***Thành phần ngôn ngữ***
Xanh nước biển	các hàm và các ký hiệu chuẩn của VisualLISP
Đỏ tím	các chuỗi ký tự (String - đặt trong ngoặc kép)
Xanh lục	các số
Tím trên nền xám	các đoạn ghi chú (Comments)
Đỏ	các dấu ngoặc đơn
Đen	thành phần khác (các biến, hàm người dùng...)

Sơ đồ màu này có thể thay đổi theo ý người dùng bằng cách chọn lệnh *Tools / Window Attributes / Configure Current*.

6.1.4.3. Định dạng mã nguồn VisualLISP

VisualLISP cho phép định dạng mã nguồn theo một cách thức thống nhất. Nhờ định dạng thích hợp có thể dễ dàng phát hiện một lỗi cú pháp như lỗi dấu ngoặc đơn (rất hay gặp trong lập trình AutoLISP). Các kiểu định dạng được thiết lập qua lệnh *Tools / Environment Options / VisualLISP Format Options*. Hộp thoại hiện ra cho phép thiết lập các thông số định dạng:

- ***Các kiểu ghi chú:*** kiểu *"; single semicolon"* được đặt tại vị trí khai báo trên dòng *Single semicolun comment indentation* trong hộp thoại. Kiểu *";; curent-column"* xuất hiện trên đầu dòng mới, được đặt cùng mức với đầu dòng cuối trong đoạn mã chương trình . Kiểu *";;; 0-column"* xuất hiện ngay đầu dòng mới; loại này thường được dùng để ghi chú các tiêu đề. Các kiểu ghi chú trên được thể hiện từ vị trí ghi chú đến hết dòng,

còn kiểu *;/ inline /;* dùng để thể hiện ghi chú trong một đoạn bất kỳ (giữa dòng hoặc nhiều dòng).

- ***Kiểu định dạng các phần tử trong danh sách nhiều thành phần***: Narrow-style, Wide-style, Column-style.

Ví dụ, dòng lệnh *(autoload "appload" '("appload"))* định dạng theo các kiểu khác nhau sẽ có dạng tương ứng như sau:

Kiểu Narrow-style - các thành phần được lùi đầu dòng 1 khoảng hẹp:

```
(autoload
 "appload"
 '("appload")
)
```

Kiểu Wide-style - các thành phần được lùi đầu dòng ứng với thành phần đầu của dòng trên:

```
(autoload "appload"
          '("appload")
                    )
```

Kiểu Column-style - các thành phần cách nhau theo dấu *tab:*

```
(autoload"appload"          '("appload")
```

- ***Các dấu ngoặc đóng ")" kết thúc danh sách***: có thể có các tuỳ chọn (1) *Close at the same line* - đặt dấu ngoặc đóng trên dòng cuối của danh sách; (2) *Close at the new line with inner indentation* - đưa dấu ngoặc ")" xuống dòng dưới và đóng bằng các thành phần trong khối; (3) *Close at the new line with outer indentation* - đưa dấu ngoặc ")" xuống dòng dưới và đóng bằng thành phần đầu khối. Các ví dụ sau minh họa cho các kiểu định dạng trên:

Trước khi định dạng
```
(setq a 100 b (list 1 2 "abc" 10) c '(1 2 3))
```

Định dạng kiểu (1)
```
(setq a 100
      b (list 1 2 "abc" 10)
      c '(1 2 3))
```

Định dạng kiểu (2)
```
(setq a 100
      b (list 1 2 "abc" 10)
      c '(1 2 3)
      )
```

Định dạng kiểu (3)
```
(setq a 100
      b (list 1 2 "abc" 10)
      c '(1 2 3)
)
```

Sau khi thiết lập các thông số ưng ý có thể tiến hành định dạng toàn bộ file bằng cách chọn lệnh *Tools / Format code in Editor (Ctrl-Alt-F)* hoặc lệnh *Tools / Format code in Selection (Ctrl-Shift-F)* để định dạng phần mã nguồn đã chọn.

Ngoài ra còn có thể sử dụng tính năng *Check* để tìm các lỗi cú pháp. Chọn lệnh *Tools / Check text in Editor (Ctrl-Alt-C)* để dò lỗi cho file đang mở hoặc chọn *Tools / Check text in Selection (Ctrl-Shift-C)* để dò lỗi cho phần mã nguồn đã chọn trước.

6.2. CÁC KHÁI NIỆM CƠ BẢN VỀ NGÔN NGỮ AUTOLISP

6.2.1. Danh sách và biểu thức

AutoLISP được xây dựng trên cơ sở các danh sách. Một danh sách (List) gồm một hay nhiều phần tử, hoặc không có phần tử nào (danh sách rỗng), chứa trong một cặp ngoặc đơn. Các phần tử trong danh sách có thể gồm các kiểu dữ liệu khác nhau: số nguyên, số thực, xâu ký tự... hoặc các danh sách khác. Mỗi đối tượng (thực thể - Entity) vẽ trong AutoCAD được quản lý như một danh sách gồm nhiều phần tử là các danh sách con. Bản vẽ AutoCAD thực chất được quản lý như cơ sở dữ liệu mà các bản ghi là các danh sách.

Một dạng danh sách đặc biệt gọi là Dot-pair gồm 2 phần tử, liên kết với nhau bởi dấu chấm (Dot). Loại danh sách này thường được dùng để biểu thị các mã nhóm đối tượng (thực thể - Entity) trong cơ sở dữ liệu bản vẽ. Cần phân biệt giữa dấu chấm trong danh sách với dấu chấm thập phân.Ví dụ dưới đây hiển thị mã của đối tượng vòng tròn màu đỏ vẽ trên lớp L123 trong AutoCAD:

Ví dụ 6.1: *Mã đối tượng*

```
(                                   ; bắt đầu danh sách
 (-1 . <Entity name: 14f9d88>)      ; tên đối tượng
 (0 . "CIRCLE")                     ; loại đối tượng "CIRCLE"
 (330 . <Entity name: 14f9cf8>)     ; con trỏ
 (5 . "41")                         ; handle
 (100 . "AcDbEntity")               ; mặt nạ
 (67 . 0)                           ; model space
 (410 . "Model")                    ; tên tab hiển thị đối tượng
 (8 . "L123")                       ; lớp
 (62 . 1)                           ; màu
 (100 . "AcDbCircle")               ; mặt nạ
 (10 10.0 20.0 0.0)                 ; tâm
 (40 . 5.0)                         ; bán kính
```

```
(210 0.0 0.0 1.0)                ; vectơ pháp tuyến của mặt phẳng
                                 ; chứa đối tượng
)                                ; kết thúc danh sách
```

Một biểu thức (câu lệnh) AutoLISP được bao trong một cặp dấu ngoặc đơn, trong đó gồm các phần tử. Điều khác biệt so với danh sách là phần tử đầu của biểu thức phải là một hàm, các phần tử sau là các tham số của hàm đó. Cũng cần lưu ý rằng trong AutoLISP không có khái niệm về các toán tử, chúng cũng được coi là hàm. Trong một biểu thức cũng có thể chứa các biểu thức con. Ví dụ sau đây thể hiện một biểu thức AutoLISP:

(setq var1 (- (+ 10 5 2.0) (setq var2 (/ pi 2.0))))

Biểu thức này bắt đầu bằng hàm "setq", dùng để gán giá trị cho biến var1. Biến này là hiệu từ kết quả xử lý hai hàm khác: hàm "+" tính tổng các số sau nó (10, 5 và 2.0) và giá trị var2 từ hàm "/" tính giá trị π/2. Hãy cẩn thận với các dấu ngoặc khi viết các biểu thức AutoLISP phức tạp. Nhiều khi chỉ một sai sót nhỏ như thiếu hay sai vị trí các ngoặc này có thể dẫn đến kết quả sai.

6.2.2. Các lời chú thích

AutoLISP sử dụng dấu chấm phẩy ";" để ghi chú thích trên một dòng. Chú thích đoạn hoặc giữa dòng (inline) được đặt trong cặp dấu ";|" và "|;". Loại chú thích này được dùng khi cần ghi một đoạn chú thích dài trên nhiều dòng hoặc giữa một biểu thức. Trong ví dụ sau đây các chữ in nghiêng thể hiện lời chú thích:

Ví dụ 6.2: *Các lời chú thích*

```
(setq a 10); gán giá trị a = 10. Chú thích này có giá trị đến hết dòng
(setq ;| gán giá trị cho biến |; a 10)      ; chú thích trong biểu thức
```

;| Còn đây là một chú thích dài được ghi trên nhiều dòng để giải thích công dụng một đoạn chương trình.
Đoạn chương trình này dùng chuyển đổi giá trị một góc đo từ độ sang radian. Thông số truyền vào hàm là giá trị góc đo bằng độ. Hàm trả về giá trị đổi ra radian. Gọi hàm theo cú pháp (do2rad do).
|;

```
(defun do2rad (do / ppi)          ; khai báo hàm
   (setq ppi 3.14159)
   (* do (/ ppi 180.0))
) ; hết hàm do2rad
```

6.2.3. Biến, hằng và các kiểu dữ liệu

Các biến trong AutoLISP không phân biệt chữ hoa và chữ thường, kiểu dữ liệu của biến không cần phải khai báo trước. Một biến có thể được gán các kiểu dữ liệu khác nhau và kiểu dữ liệu được xác định qua kiểu của giá trị được gán. Hàm *setq* dùng để gán giá trị cho biến có cú pháp như sau:

(setq bien gia_tri)

trong đó *setq* - từ khoá (tên hàm) thể hiện phép gán.

bien - tên biến cần gán giá trị.

gia_tri - giá trị cần gán cho biến (hằng số hoặc biểu thức).

Ví dụ 6.3: *Phép gán setq*

```
(setq var 10)          ; biến var nhận kiểu dữ liệu nguyên
(setq var 10.0)        ; biến var nhận kiểu dữ liệu số thực
(setq var "Abc")       ; biến var nhận kiểu dữ liệu xâu ký tự
```

Trong một hàm AutoLISP có hai loại biến: các biến tham số cần truyền qua hàm và các biến cục bộ chỉ có tác dụng trong bản thân hàm. Các tham số cần truyền được khai báo ngay đầu hàm, còn các biến cục bộ được khai báo sau dấu gạch chéo "/". Cần có khoảng trắng trước và sau dấu gạch chéo. Trong ví dụ 6.2, biến *do* là biến cần truyền qua hàm, còn biến *ppi* là biến cục bộ. Các biến cục bộ nếu không được khai báo có thể sẽ làm thay đổi các biến ngoài hàm, gây nên những rắc rối không lường trước.

VisualLISP dùng một số ký hiệu dành riêng cho hằng, nên lưu ý tránh đặt tên biến trùng tên với các hằng này:

PI - hằng số pi = 3.14159

NIL - hằng số lôgic false

T - hằng số lôgic true (non-NIL)

PAUSE - sử dụng với hàm *command* nhằm tạm dừng chương trình chờ người dùng nhập dữ liệu.

Ngoài các kiểu dữ liệu thông dụng trong các ngôn ngữ lập trình khác như số nguyên (Integer), số thực (Real), chuỗi ký tự (String), trong AutoLISP còn có một số kiểu dữ liệu với những đặc thù riêng. Kiểu dữ liệu số nguyên cũng hơi đặc biệt: trong chương trình có thể gán một số nguyên 32-bit (LongInteger - có giá trị từ -2147483647 đến +2147483647 với AutoCAD 2000, trong khi với AutoCAD 14, số nguyên bé nhất lại là -2147483648), nhưng hàm nhập liệu từ bàn phím GetInt lại chỉ chấp nhận số nguyên 16-bit. Khi gán một biến giá trị số nguyên vượt quá 32-bit, số này được chuyển thành số thực, nhưng các phép tính trên

số nguyên sẽ cho kết quả nguyên, nên cần chú ý đặc biệt đến điều này nếu không có thể cho kết quả sai. Ví dụ, các biểu thức:

(setq int (/ 10 4)) gán cho biến int giá trị nguyên 2.

((setq rea (/ 10 4.0)) hoặc (setq rea (/ 10.0 4)) gán cho rea giá trị 2,5.

(setq max 2147483647) gán max là một số nguyên.

(setq max 2147483648) gán cho max số thực 2.14748e+009.

(setq mis (+ 2147483646 3)) gán cho mis số nguyên -2147483647 (sai do vượt quá 32 bit).

(setq ok (+ 2147483648 2)) gán cho ok giá trị số thực 2.14748e+010.

Một kiểu dữ liệu đặc thù của AutoLISP là kiểu danh sách List. Các phần tử trong danh sách có thể là các kiểu dữ liệu khác nhau: số nguyên, số thực, xâu ký tự, danh sách, ký hiệu (tên hàm, tên biến,...). Có 2 cách xác định một danh sách: sử dụng hàm *quote* (hoặc dấu nháy đơn ') trước danh sách hoặc hàm *list* trước các thành phần của danh sách. Hàm *quote* trả về danh sách mà không xử lý các biểu thức, trong khi hàm *list* cho ra các thành phần là các kết quả của các biểu thức này. Trong AutoLISP không có dữ liệu dạng mảng (array), nên danh sách cũng được dùng để xử lý dữ liệu dạng mảng. Các ví dụ sau tạo ra cùng một danh sách:

```
'(1  2 "abc" (0 . "CIRCLE"))                ; danh sách 4 phần tử
(quote (1 2 "abc" (0 . "CIRCLE")))          ; tạo bằng hàm quote
(list 1 (+ 1 1) "abc" (cons 0 "CIRCLE"))    ; tạo bằng hàm list
```

nhưng biểu thức

```
(quote (1 (+ 1 1) "abc" (cons 0 "CIRCLE")))
```

sẽ trả về danh sách gồm các phần tử nguyên dạng đã khai báo 1, (+ 1 1), "abc" và (cons 0 "CIRCLE") mà không xử lý các biểu thức này.

Hai kiểu dữ liệu đặc thù khác của AutoLISP là kiểu EName và PickSet (SelectionSet) quản lý các đối tượng trong bản vẽ. Kiểu EName cho tên đối tượng trên bản vẽ AutoCAD, qua đó có thể truy cập đến cơ sở dữ liệu của chúng và xử lý theo các cách khác nhau. Kiểu PickSet quản lý một tập hợp các đối tượng được chọn để cùng xử lý theo một cách như nhau. Chúng tương tự như nhóm đối tượng được chọn sau câu nhắc *Select Objects* trong AutoCAD. Có thể thêm các đối tượng vào hoặc loại bớt khỏi tập hợp chọn.

AutoLISP cũng sử dụng các file để lưu dữ liệu. Tuy nhiên các file ở đây chỉ sử dụng kiểu file văn bản (text) truy cập tuần tự, không có kiểu truy cập ngẫu nhiên. Dữ liệu trong file thường được ghi vào và đọc ra theo dòng, lần lượt từ dòng thứ nhất đến dòng cuối cùng. Để làm việc với

dữ liệu dạng bảng nhiều dòng và cột lưu trong file, bảng thường được xử lý thành một danh sách gồm các danh sách con tạo từ các dòng dữ liệu, sau đó sử dụng các hàm thao tác với danh sách khác (xem mục 6.2.4.3) để truy cập đến các ô của bảng.

6.2.4. Cấu trúc chương trình AutoLISP

Chương trình AutoLISP có một cấu trúc rất tự do: các biến không cần khai báo trước, cũng như không cần khai báo kiểu dữ liệu của chúng; các hàm con cũng có thể khai báo bất kỳ ở đâu trong chương trình. Chính vì vậy cần rất chú ý khi lập trình để tránh các sai lầm không đáng có. Đoạn chương trình sau không sai nhưng cách viết như vậy sẽ rất khó theo dõi:

Ví dụ 6.4: *Cấu trúc tự do của AutoLISP*

```
;;; Chương trình vẽ vòng tròn, ghi ra các thông số chu vi và diện tích
;;; AutoCAD nhắc nhập tâm và bán kính vòng tròn

;;; Đây là hàm con thứ nhất - viết ở đầu chương trình chính để
;;; tính thông số chu vi và diện tích vòng tròn qua bán kính đã cho
(defun tinh-tso(bk / cv dt dsach)
 (setq cv (* pi bk 2.0))     ; chu vi vòng tròn
 (setq dt (* pi bk bk))      ; diện tích
 (setq dsach (list cv dt))   ; kết quả trả về là danh sách
 )

;;; Bắt đầu chương trình chính
(setq tamvt (getpoint "Nhập tâm vòng tròn: "))

;;; Tạm dừng chương trình chính để khai báo hàm con thứ hai,
;;; hàm này viết ở giữa chương trình
(defun hienthi(tam bk tso / t1 t2 txt)
 (command ".CIRCLE" tam bk)
 (setq t1 (rtos (car tso) 2 3)
  t2 (rtos (cadr tso) 2 3))
 (setq txt (strcat "Chuvi: " t1 " Dientich: " t2))
 (command ".-TEXT" tam "" "" txt)
 )

;;; Tiếp tục chương trình chính
(setq bkvt (getdist tamvt "Nhập bán kính vòng tròn: "))
(setq tso (tinh-tso bkvt))   ; gọi hàm tinh-tso
(hienthi tamvt bkvt tso)     ; gọi hàm hiển thị vòng tròn và ghi thông số.
```

6.2.4.1. Hàm người dùng

Hàm người dùng được định nghĩa với cú pháp sau:

```
(defun fun-name([arg] [/ var])
    (expr...)
    (last-expr)
    )
```

trong đó *fun-name* là tên hàm, *arg* là các tham số cần truyền qua hàm, *var* là các biến cục bộ. Trước và sau dấu phân cách "/" cần có khoảng trắng. Thân hàm chứa các biểu thức *expr*. Giá trị trả về của hàm là kết quả xử lý biểu thức cuối cùng (*last-expr*) trong thân hàm. Để hàm không trả về giá trị nào (tức là trả về NIL), thường sử dụng biểu thức cuối cùng là lời gọi hàm *(princ)*.

Gọi hàm đã định nghĩa theo cú pháp:

```
(fun-name arg1 arg2 ...)
```

trong đó *arg1*, *arg2*, ... là các tham số truyền cho hàm. Lưu ý khi gọi hàm cần truyền đủ tham số với kiểu dữ liệu thích hợp.

Kiểu hàm đặc biệt có tiền tố "*C:*" đặt trước tên hàm thường được dùng để tạo thêm các lệnh AutoCAD. Sau khi nạp chương trình, hàm được gọi trong AutoCAD như các lệnh bình thường khác. Loại hàm này cần khai báo không có tham số truyền, nhưng nên khai báo các biến cục bộ sử dụng trong hàm.

AutoLISP cũng cho phép định nghĩa các hàm đệ quy, nghĩa là có thể gọi chính nó. Ví dụ, hàm tính giai thừa của một số nguyên (n!) được viết kiểu đệ quy như sau:

Ví dụ 6.5: *Gọi hàm kiểu đệ quy - tính giai thừa n!*

```
(defun giaithua(n / i a)
  ;; hàm này tính giai thừa của một số nguyên
  ;; như một ví dụ về cách gọi hàm kiểu đệ quy (recursive)
  (if (= n  0)
    1                          ; giá trị trả về   0! = 1 (khi n=0)
    (* n (giaithua (1- n)))    ; còn không thì tính n! = n * (n-1)!
  )
)
   ;; và tạo một lệnh AutoCAD có tên "n!" để tính giai thừa

(defun C:n! (/ c n)
  ;; không cho phép người sử dụng nhập số âm hoặc nhấn ngay Enter
  ;; và nhắc người dùng nhập một số nguyên không âm
     (initget 5)
     (setq n (getint "Hãy nhập một số nguyên không âm: "))
  ;; gọi hàm tính giai thừa và in kết quả
```

```
    (setq c (giaithua n))
    (princ (strcat (itoa n) "! = " (itoa c)))
  ;; rút lui im lặng !
    (princ)
  )
```

Lưu ý đến giới hạn 32-bít của số nguyên và phép tính trên số nguyên cho kết quả nguyên nên khi nhập số quá lớn kết quả trả về không còn đúng nữa. Để tính giai thừa cho các số lớn hơn có thể thay đổi một chút mã chương trình để các phép tính thực hiện trên số thực.

6.2.4.2. Các hàm cơ sở trong AutoLISP

AutoLISP cung cấp một loạt các hàm cơ sở. Dưới đây giới thiệu một số hàm được dùng nhiều trong lập trình AutoLISP.

* ***Các hàm số học*** thực hiện các phép tính trên số nguyên hoặc số thực và kết quả trả về là một số.

- *Hàm cộng :* *(+ n1 n2 ...)*

Đây thực chất là phép cộng trong các ngôn ngữ lập trình khác. Các tham số *n1*, *n2*,... là các số nguyên hoặc số thực. Kết quả trả về là tổng các số này. Kiểu dữ liệu trả về phụ thuộc các tham số: nếu chỉ gồm số nguyên, kết quả cũng là số nguyên; nếu có số thực thì kết quả là số thực; nếu không có tham số, hàm trả về số 0 (nguyên). Lưu ý dấu cách giữa các thành phần trong biểu thức.

- *Hàm trừ :* *(- n1 n2 ...)*

Hàm này thực hiện phép trừ n1 – n2 – ... Các tham số *n1*, *n2*,... là các số nguyên hoặc số thực. Nếu không có tham số hàm trả về số 0 (nguyên).

- *Hàm đếm tăng (1+ n)*

Hàm trả về giá trị n+1, trong đó n có thể là số nguyên hay số thực, nhưng nhất thiết phải có. Lưu ý "1+" là tên hàm, và vì thế số 1 và dấu + viết liền nhau.

- *Hàm đếm giảm (1- n)*

Hàm trả về giá trị n-1, trong đó n có thể là số nguyên hay số thực, nhưng nhất thiết phải có.

- *Hàm nhân :* *(* n1 n2 ...)*

Hàm này thực hiện phép nhân n1*n2*... Các tham số *n1*, *n2*,... là các số nguyên hoặc số thực. Nếu không có tham số, hàm trả về số 0 (nguyên).

- *Hàm chia :* *(/ n1 n2 ...)*

Hàm này thực hiện phép chia kép (n1/n2)/... Các tham số *n1, n2,...* là các số nguyên hoặc số thực. Nếu không có tham số hàm trả về số 0 (nguyên). Nếu chỉ có một tham số khác NIL, sẽ trả về chính số đó. Nếu có các tham số n2, n3...., chúng cần phải khác 0.

- *Hàm lấy giá trị tuyệt đối : (abs n)*

Hàm trả về giá trị tuyệt đối của số n, là số nguyên hay số thực, nhưng nhất thiết phải có.

- *Hàm lấy phần nguyên : (fix n)*

Hàm trả về giá trị nguyên, sau khi đã bỏ phần thập phân của số n. Số này có thể là số âm hoặc dương. Lưu ý rằng đây không phải là hàm làm tròn số, cũng không phải là phần nguyên của số theo định nghĩa quen thuộc trong số học. Xem các ví dụ sau:

(int 12.1) trả về 12 (nguyên)
(int 12.9) trả về 12 (nguyên)
(int -12.4) trả về -12 (nguyên)

- *Tìm giá trị lớn nhất và nhỏ nhất*

(max n1 n2 ...) trả về giá trị lớn nhất trong các số n1, n2, ...
(min n1 n2 ...) trả về giá trị bé nhất trong các số n1, n2, ...

- *Hàm mũ và lôgarit*

(exp x) - hàm mũ cơ số e (e^x).
Ví dụ: (exp 1.0) cho kết quả 2.71828 (chính là số e)
(expt a x) - hàm mũ cơ số bất kỳ (a^x).
Ví dụ: (expt 2 (/ 1.0 3)) cho kết quả 2.0 ($= 8^{1/3}$)
(sqrt x^+) - hàm căn bậc 2 của số dương.
Ví dụ: (sqrt 4) trả về 2.0
(log x^+) - hàm lôgarit tự nhiên (cơ số e) của số dương.
Ví dụ: (log 2.71828) trả về 0.999999 *; lưu ý: ln(e) = 1*

- *Hàm lượng giác*

(cos rad) trả về giá trị côsin của góc. Tham số góc tính bằng radian.
(sin rad) trả về giá trị sin của góc. Tham số góc tính bằng radian.
(atan n1 [n2]) tính arctg của 1 số. Kết quả cho góc đo bằng radian. Nếu chỉ có tham số n1, hàm trả về arctg(n1), còn nếu có cả n2, giá trị trả về là arctg(n1/n2). Ví dụ:

(atan 1) → 0.785398 (π/4)

(atan 1 0) → 1.5708 (π/2) nhưng lưu ý

(atan -1 0) → -1.5708 (-π/2)

* ***Các hàm so sánh*** - các hàm này trả về một trong hai giá trị T (đúng) hoặc NIL (sai) sau khi đánh giá các biểu thức trong hàm.

- *So sánh "bằng"* *(= ns1 ns2 ...)* - so sánh các tham số ns1, ns2,... xem chúng có bằng nhau hay không. Các tham số này có thể là số hoặc chuỗi ký tự string. Nếu tất cả các tham số bằng nhau hoặc chỉ có 1 tham số, hàm trả về T, còn không hàm sẽ trả về NIL. Ví dụ:

(= 1 1.0) → T ; *chuyển về cùng kiểu real trước khi so sánh*

(= 1 1 2) → NIL ; *các số không bằng nhau*

(= "ABC" "Abc") → NIL ; *phân biệt chữ hoa và chữ thường*

(= NIL) → T ; *chỉ có 1 tham số*

- *So sánh "khác"* *(/= ns1 ns2 ...)* - so sánh các tham số ns1, ns2,... xem chúng có khác nhau không. Nếu chỉ có 1 tham số hàm sẽ trả về T, còn nếu nhiều tham số, kết quả là T nếu các số đứng cạnh nhau không giống nhau, còn ngược lại là NIL. Ví dụ:

(/= 10 20 10 20) → T nhưng (/= 10 20 20 10) → NIL

- *So sánh "gần bằng"* *(equal exp1 exp2 fuzz)* - so sánh gần đúng giá trị hai biểu thức *exp1*, *exp2* với *fuzz* là số thực chỉ giá trị của sai số. Nếu bỏ qua *fuzz*, hàm này thực hiện như hàm so sánh "=". Ví dụ:

(setq a 1.123456 b 1.123457); *gán a = 1.123456, b = 1.123467*

(equal a b) → NIL ; *vì hai số a và b khác nhau*

(equal a b 0.000001) → T ; *vì abs(a-b) ≤fuzz = 0.000001*

- *So sánh "bé hơn"* *(< ns1 ns2 ...)* - trả về T nếu các tham số ns bé hơn tham số bên phải nó, nghĩa là các tham số phải theo thứ tự tăng dần. Ngược lại, hàm trả về NIL. Tương tự với các hàm ">" (lớn hơn), "<=" (bé hơn hoặc bằng, ">=" (lớn hơn hoặc bằng).

* ***Các hàm lôgic*** trả về các giá trị T hoặc NIL. Tham số của hàm là các biểu thức lôgic. Loại hàm này được dùng để liên kết các biểu thức lôgic, phục vụ các cấu trúc rẽ nhánh và vòng lặp. Dưới đây giới thiệu các hàm này:

- *(or expr1 expr2 ...)* trả về NIL nếu tất cả các biểu thức expr được đánh giá là NIL, hàm trả về giá trị T trong trường hợp ngược lại. Danh

sách rỗng '() được đánh giá là NIL.

- *(and expr1 expr2 ...)* trả về T nếu tất cả các biểu thức expr được đánh giá là T, hàm trả về giá trị NIL trong trường hợp ngược lại.

- *(not expr)* trả về NIL nếu expr được đánh giá là T, ngược lại nếu expr là NIL hàm sẽ trả về T. Để kiểm tra xem một biểu thức có phải NIL hay không, có thể sử dụng hàm *(null expr)* - hàm này trả về T nếu expr là NIL, ngược lại trả về T. Thường hàm *null* thường được dùng để đánh giá danh sách có rỗng hay không.

- *(numbp item)* trả về T nếu item là một số (nguyên hoặc thực), ngược lại, hàm trả về NIL. Hàm này thường được dùng để đánh giá xem giá trị một biến hoặc một biểu thức có phải là số hay không.

* ***Các hàm xử lý chuỗi ký tự*** sử dụng với dữ liệu dạng chuỗi ký tự (String): nối các chuỗi ký tự, lấy một phần, thay đổi chữ hoa/thường,...

- *(read [str])* lấy thành phần đầu của chuỗi ký tự *str*, ngăn cách bởi dấu trắng (Space), dòng mới, dấu Tab hoặc ngoặc đơn. Giá trị trả về của hàm là thành phần nhận được, chuyển đổi về dạng dữ liệu thích hợp. Nếu tham số *str* không có, hàm trả về NIL. Ví dụ:

```
(read "abc")        trả về ký hiệu (symbol) ABC
(read "abc def 123") cũng trả về ABC, nhưng
(read "\"abc\" def") trả về chuỗi ký tự "abc"
(read "123 abc def") sẽ trả về số nguyên 123 và
(read "(a b c)(d e f)) sẽ trả về danh sách (A B C)
```

Hàm này rất hữu ích khi cần đọc dữ liệu từ file theo dòng và chuyển cả dòng thành dữ liệu dạng danh sách list.

- *(strcase str [mode])* chuyển toàn bộ các chữ trong chuỗi ký tự *str* thành chữ in (khi bỏ qua mode hoặc mode = NIL) hoặc chữ thường (khi mode khác NIL). Ví dụ:

```
(strcase "AutoCAD")  trả về chuỗi "AUTOCAD"
(strcase "AutoCAD" NIL) cũng trả về chuỗi "AUTOCAD"
(strcase "AutoCAD" 1) trả về chuỗi "autocad"
```

- *(strcat [str1 [str2]...])* nối các chuỗi ký tự str1, str2... thành một chuỗi chung. Ví dụ:

```
(strcat "AutoCAD" "va" "AutoLISP") sẽ trả về chuỗi lớn "AutoCAD va
AutoLISP".
```

- *(strlen [str])* cho chiều dài (số lượng các ký tự) trong chuỗi *str*

(gồm cả dấu cách). Ví dụ:

(strlen "ab") trả về số nguyên 2

(strlen "AutoCAD &") trả về 9

- *(substr str start [length])* trả về chuỗi ký tự con, lấy ra từ chuỗi *str*, bắt đầu từ vị trí *start* với số ký tự (độ dài) *length*. Nếu *length* được bỏ qua, chuỗi mới sẽ được lấy từ vị trí *start* đến hết chuỗi *str*. Ví dụ:

(substr "abcde" 2) cho chuỗi con "bcde" (chuỗi con bắt đầu từ vị trí 2 đến hết chuỗi gốc)

(substr "abcde" 2 1) cho chuỗi con "b" (chuỗi con dài 1 ký tự, bắt đầu từ vị trí thứ 2)

(substr "abcde" 3 2) cho chuỗi con "cd" (chuỗi con dài 2 ký tự, bắt đầu từ vị trí thứ 3)

6.2.4.3. Các hàm xử lý danh sách

Danh sách là một kiểu dữ liệu đặc thù trong AutoLISP. Một danh sách được bắt đầu bằng dấu ngoặc đơn mở "(" và kết thúc bằng một dấu ngoặc đơn đóng ")". Danh sách có thể chứa các dữ liệu bất kỳ: số, ký tự, danh sách con,... Cơ sở dữ liệu chứa các đối tượng AutoCAD cũng được tổ chức dưới dạng danh sách. Vì AutoLISP không có kiểu dữ liệu mảng nên danh sách được dùng để thay thế. Dưới đây là một số hàm thông dụng xử lý danh sách:

- ***(quote expr)*** trả về biểu thức *expr* mà không xử lý giá trị của nó. Có thể sử dụng hàm này để tạo danh sách từ các phần tử là hằng số. Ví dụ, để chỉ định 1 điểm như một danh sách gồm 3 tọa độ của nó (x = 1, y = 1, z = 0), có thể sử dụng biểu thức sau:

(setq pt1 (quote (1 1 0)) *; biến pt1 được gán giá trị là danh sách (1 1 0)*

Lưu ý một cách viết khác của hàm *quote* là đặt trước biểu thức *expr* một dấu nháy đơn. Ví dụ, biểu thức sau sẽ tương đương với biểu thức nói trên:

(setq pt1 '(1 1 0)) *; dấu nháy đơn thay cho hàm quote*

Biểu thức dưới đây tạo ra một danh sách rỗng (không chứa phần tử nào), và gán nó cho biến lst:

(setq lst '())

Danh sách rỗng thường được dùng để khởi tạo danh sách mới, sau đó các phần tử sẽ được thêm vào sau bằng hàm *append*.

- *(list [expr ...])* xử lý các biểu thức *expr* và liên kết chúng thành một danh sách. Hàm trả về danh sách, ngoại trừ trường hợp không có

tham số, hàm sẽ trả về NIL. Khác với hàm *quote*, hàm này có thể tạo danh sách không những từ các hằng số, mà có thể từ giá trị các biến hoặc các biểu thức. Ví dụ, với đoạn chương trình:

```
(setq n1 10)
(setq n2 20)
(setq lst (list n1 (+ n1 n2) 'a '(1 2)))
```

Kết quả biến *lst* sẽ được gán dữ liệu là danh sách (10 30 A (1 2)), trong đó thành phần đầu của danh sách là giá trị của biến *n1* (10), thành phần thứ 2 (30) là kết quả xử lý hàm *(+ n1 n2)*, thành phần thứ 3 (A) là hằng số (thuộc kiểu dữ liệu Symbol) và phần tử cuối cùng (1 2) là một hằng số kiểu danh sách List.

- ***(cons el1 list-atom)*** thêm thành phần *el1* vào đầu danh sách hoặc tạo danh sách dạng Dot-pair tuỳ theo tham số thứ hai *list-atom* là danh sách hay là giá trị. Giá trị trả về của hàm tuỳ thuộc vào kiểu của tham số thứ 2. Nếu là danh sách, kết quả trả về là danh sách mới sau khi đã thêm thành phần *el1* vào đầu danh sách đã cho, còn nếu là giá trị, kết quả trả về sẽ là danh sách Dot-pair liên kết 2 phần tử là các tham số trên. Xem các ví dụ minh họa sau:

```
(cons 10 '(1 2 3))          trả về danh sách gồm 4 phần tử (10 1 2 3)
(cons 10 (list 1 2 3))      cũng cho kết quả như trên
(cons 10 '())               trả về danh sách gồm 1 phần tử (10)
(cons '(10 20) '(1 2 3))    cho kết quả danh sách ((10 20) 1 2 3)
(cons 10 1)                 cho danh sách liên kết (10 . 1)
(cons '(10 20) 'a)          cho danh sách liên kết ((10 20) . A)
(cons 10 "AutoLISP")        cho danh sách liên kết (10 . "AutoLISP").
```

- ***(append [lst ...])*** liên kết các danh sách *lst* và tạo ra một danh sách mới. Nếu không có tham số *lst* nào, hàm trả về NIL. Ví dụ:

```
(append '(a b) '(1 2) '("a" "b"))  trả về danh sách (A B 1 2 "a" "b")
(append (list '(a b)) (list '(1 2)) (list '("a" "b")))  trả về danh sách
        ((A B) (1 2) ("a" "b")).
```

Lưu ý rằng trong các ví dụ trên *'(a b)* là danh sách chứa hai *ký hiệu (Symbol)* A và B, chứ không phải là ký tự *a* và *b*.

- ***(reverse lst)*** đảo ngược thứ tự các phần tử trong danh sách *lst* và trả về danh sách đã đảo ngược này. Ví dụ:

```
(setq lst '(1 2 3 4))              ; gán danh sách (1 2 3 4) cho biến lst
(setq newlst (reverse lst))        ; tạo danh sách mới newlst (4 3 2 1)
```

- ***(subst new-item old-item lst)*** tìm kiếm phần tử *old-item* trong danh sách *lst* và thay thế bằng phần tử mới *new-item*. Nếu tìm thấy

old-item hàm trả về danh sách mới, còn nếu không tìm thấy, kết quả của hàm sẽ là danh sách *lst* ban đầu. Ví dụ:

```
(setq lst '(1 2 3 4))                ; gán danh sách (1 2 3 4) cho biến lst
(setq newlst (subst "Ab" 3 lst))     ; gán cho biến newlst danh sách mới
                                     ; sau khi thay thế phần tử có giá trị 3
                                     ; trong danh sách lst bằng phần
                                     ; tử mới "Ab", kết quả newlst
                                     ; sẽ là danh sách (1 2 "Ab" 4).
```

- ***(length lst)*** cho biết số phần tử có mặt trong danh sách *lst*. Ví dụ:
  (length '(1 2 3 4)) sẽ trả về số nguyên 4,
  (length '((10 11) (20 21) 3 (40 41)) cũng trả về 4, còn
  (length '()) sẽ trả về số nguyên 0.

Để truy cập và lấy giá trị của một phần tử trong danh sách có thể sử dụng một trong các hàm sau đây:

- Hàm ***(last lst)*** cho phần tử cuối cùng của danh sách *lst*. Ví dụ:
  (last '(1 2 3 4)) sẽ trả về số nguyên 4, còn
  (last '((10 11) (20 21) 3 (40 41)) trả về danh sách con (40 41).

- Hàm ***(car lst)*** cho phần tử đầu tiên của danh sách *lst*. Ví dụ:
  (car '(1 2 3 4)) sẽ trả về 1, còn
  (car '((10 11) (20 21) 3 (40 41)) trả về danh sách con (10 11).

- Hàm ***(nth n lst)*** trả về phần tử thứ *n* của danh sách *lst*. Lưu ý rằng tham số *n* phải là một số nguyên không âm và các phần tử trong danh sách được đánh số từ 0, do đó biểu thức *(nth 0 lst)* và *(car lst)* sẽ trả về cùng một kết quả là phần tử đầu trong danh sách. Tương tự, hai biểu thức *(nth (1- (length lst)) lst)* và *(last lst)* là tương đương, cùng trả về phần tử cuối của danh sách.

- Hàm ***(cdr lst)*** trả về danh sách được trích ra từ danh sách *lst* (tham số thứ hai của hàm) sau khi đã bỏ đi phần tử đầu. Ví dụ:

```
(setq 3DP '(100 10 0))          ; gán cho biến 3DP danh sách
                                ; gồm 3 phần tử (tọa độ điểm)
(setq YZ-lst (cdr 3DP))         ; YZ-lst sẽ nhận giá trị là danh
                                ; sách 2 phần tử (10 0)
(setq Z-lst (cdr YZ-lst))       ; Z-lst sẽ nhận giá trị là danh
                                ; sách gồm 1 phần tử (0)
```

```
(setq NONE (cdr Z-lst))          ; NONE sẽ nhận giá trị NIL -
                                 ; không còn gì để bỏ đi nữa!
```

Để lấy phần tử thứ 2 và thứ 3 của danh sách này (số 10 và số 0 trong ví dụ trên), có thể kết hợp hai hàm *car* và *cdr* như cách viết sau đây:

```
(setq  YCord  (car (cdr 3DP))          ; trả về 10
(setq  ZCord  (car (cdr (cdr 3DP)))    ; trả về 0
```

Rõ ràng, cách viết như vậy rất khó xem, dễ gây nhầm lẫn. AutoLISP cung cấp một cách viết khác để kết hợp các hàm này với nhau, tạo nên một loạt các hàm khác để truy cập phần tử trong danh sách. Tên các hàm này được bắt đầu bằng chữ "c", kết thúc bằng chữ "r" và giữa chúng là các chữ cái "a" và "d" thay thế một lần gọi hàm *car* hoặc *cdr* tương ứng. Bảng 6.1 mô tả các hàm này (cột kết quả ứng với giá trị '((11 21 31) (12 22 32) (13 23 33)) của biến *lst*).

Các hàm này rất tiện lợi để xử lý tọa độ các điểm. Các tọa độ x, y, z của một điểm *pt* trong không gian 3 chiều (3D) được truy cập qua các biểu thức gán và gọi hàm tương ứng sau:

```
(setq   x  ( car   pt )      ; gán tọa độ X của điểm cho biến x
        y  ( cadr  pt )      ; gán tọa độ Y của điểm cho biến y
        z  ( caddr pt ) )    ; gán tọa độ Z của điểm cho biến z
```

Lưu ý rằng với các điểm 2D (chỉ có 2 tọa độ X và Y), các lời gọi hàm này sẽ thể trả về các kết quả không như mong muốn.

Bảng 6.1. Các hàm car và cdr

Biểu thức	*Biểu thức tương đương*	*Kết quả trả về*	*Kiểu dữ liệu kết quả*
(**caar** lst)	(**car** (**car** lst))	11	số
(**cadr** lst)	(**car** (**cdr** lst))	(12 22 32)	danh sách
(**cddr** lst)	(**cdr** (**cdr** lst))	(13 23 33)	danh sách
(**cadar** lst)	(**car** (**cdr** (**car** lst)))	21	số
(**cddar** lst)	(**cdr** (**cdr** (**car** lst)))	(31)	danh sách
...			
Ghi chú: Biến lst là danh sách: '((11 21 31) (12 22 32) (13 23 33))			

- Để truy cập tới dữ liệu dạng danh sách liên kết chứa các danh sách con hoặc Dot-pair, AutoLISP cung cấp một hàm *assoc* với cú pháp

chung như sau:

(assoc item asslst)

trong đó

item - khoá cần tìm kiếm trong danh sách liên kết hoặc Dot-pair (thành phần đầu của danh sách con cần tìm).

asslst - danh sách liên kết nơi sẽ tiến hành tìm kiếm. Thực chất đây là một danh sách gồm các danh sách con.

Nếu việc tìm kiếm cho kết quả, hàm trả về danh sách con hoặc Dot-pair chứa khoá cần tìm. Nếu không tìm thấy, hàm trả về giá trị NIL. Ví dụ sau minh họa điều này.

Ví dụ 6.6: *Sử dụng assoc để tìm kiếm danh sách liên kết*

```
(setq al '(          ; khai báo danh sách liên kết nơi sẽ tìm kiếm
          (410 . "Model")
          (8 . "L123")
          (62 . 1)
          (100 . "AcDbCircle")
          (10 10.0 20.0 0.0)
          (40 . 5.0)
          ))         ; hết setq
(setq space (assoc 410 al))      ; trả về danh sách Dot-pair
                                 ; (410 . "Model")
(setq layer (assoc 8 al))        ; trả về danh sách dot-pair (8. "L123")
(setq center (assoc 10 al))      ; trả về danh sách (10 10.0 20.0 0.0)
(setq its8 (assoc "L123" al))    ; trả về NIL (từ khoá phải là
                                 ; thành phần đầu của list)
(setq none (assoc "Some" al))    ; trả về NIL (không tìm thấy)
```

Hàm *assoc* thường được dùng để truy cập CSDL AutoCAD nhằm tìm kiếm một loại đối tượng nhất định nào đó, thông qua mã đối tượng (mã nhóm - GroupCode) của chúng. Trong CSDL chúng được thể hiện thông qua danh sách liên kết gồm các danh sách con và Dot-pair. Danh sách liên kết trong ví dụ trên là một phần trong mã đối tượng đồ họa được AutoCAD quản lý. Đối tượng là vòng tròn (mã 100), màu đỏ (mã 62), được thể hiện trên lớp L123 (mã 8) trong không gian mô hình (mã 410). Tâm vòng tròn có tọa độ (10.0 20.0 0.0) và bán kính 5.0 thể hiện trong danh sách này tương ứng với mã 10 và mã 40.

6.2.4.4. Cấu trúc rẽ nhánh

Hai cấu trúc rẽ nhánh trong AutoLISP gồm cấu trúc ***if*** (rẽ 2 nhánh - tuỳ thuộc vào giá trị trả về của một biểu thức được đánh giá) và cấu trúc

cond (rẽ nhiều nhánh - theo sự đánh giá từng biểu thức trong hàm).

- Cú pháp chung của hàm *if* như sau:

  (if testexpr then expr [elseexpr])

trong đó

testexpr - biểu thức được đánh giá, trả về NIL hoặc không NIL.

thenexpr - biểu thức sẽ thực hiện khi biểu thức được đánh giá trả về giá trị khác NIL.

elseexpr - biểu thức sẽ thực hiện khi biểu thức được đánh giá trả về giá trị NIL.

Giá trị trả về của hàm *if* là giá trị của biểu thức được thực hiện. Nếu bỏ qua biểu thức *elseexpr* và *testexpr* là NIL thì hàm trả về giá trị NIL.

Cần lưu ý rằng *thenexpr* và *elseexpr* chỉ được phép là một biểu thức. Nếu muốn thực hiện nhiều biểu thức cần gom chúng lại thành khối bằng hàm *progn*, tương tự cú pháp *begin...end* trong ngôn ngữ Pascal hoặc khối {...} trong ngôn ngữ C. Hàm này có cú pháp như sau:

```
(progn
        (expr1)
        (expr2)
        (...)
        (exprn)
)
```

Khi gặp khối này AutoLISP sẽ thực hiện lần lượt các biểu thức expr1, epxpr2, ..., exprn và trả trả về giá trị biểu thức cuối cùng.

Ví dụ 6.7: *Sử dụng hàm progn*

```
(if (= 1 3)
    (setq ans "YES")
    (progn
        (alert "Quá sai!")
        (setq ans "no")
    ) ; kết thúc progn
); kết thúc hàm if.
```

Khi thực hiện đoạn chương trình, hàm *if* sẽ đánh giá biểu thức *(= 1 3)*, tức là so sánh xem 1 có bằng 3 không. Biểu thức này rõ ràng là sai, do đó trả về NIL, và vì thế AutoCAD sẽ hiện hộp thoại thông báo "Quá sai!" (thông qua hàm *alert*), đồng thời biến *ans* được gán giá trị *"no"* và hàm *if* cũng trả về chuỗi ký tự *"no"*.

- Hàm *cond* có cú pháp giống như Case hoặc Select Case trong các ngôn ngữ lập trình khác. Cú pháp chung như sau:

```
(cond
   (test1 result1)
   (test2  result2)
   ...
   (testn  resultn)
   (T      elsewise)
)
```

trong đó

test1, *test2*,... - biểu thức được đánh giá, trả về NIL hoặc T.

result1, *result2*,... - biểu thức tương ứng thực hiện khi biểu thức *test* được đánh giá khác NIL.

T - trường hợp còn lại (khi tất cả các biểu thức *test* đều trả về NIL, hàm *cond* sẽ thực hiện biểu thức *elsewise*).

Giá trị trả về của hàm *cond* là giá trị của biểu thức được thực hiện. Tương tự hàm *if*, ở đây *test* và *result* chỉ được phép là một biểu thức. Nếu muốn thực hiện nhiều biểu thức cần gom chúng lại bằng hàm *progn*. Ví dụ 6.8 là cấu trúc rẽ nhánh trong thuật toán giải phương trình bậc 2:

Ví dụ 6.8: *Giải phương trình bậc 2*

```
;;; các hệ số a, b, c trong phương trình bậc hai aX² + bX + c = 0
;;; đã được nhập trước. Yêu cầu giải và biện luận.
;;; các biểu thức trong chương trình bắt đầu từ đây

(if (= a 0)
        (setq   reply "Đây không phải là phương trình bậc 2"
                x1 NIL)
        ; khi a = 0, gán reply bằng câu trả lời "Đây không phải..."
        ; nếu a ≠0 bắt đầu đoạn chương trình trong khối progn
        (progn
           (setq delta (- (* b b) (* 4 * a c)))        ; tính  Δ = b² - 4ac
           (cond
                ;; trường hợp Δ < 0
                  ((< delta 0) (setq reply "Vô nghiệm"
                                     x1  NIL
                               ))
                ;; trường hợp Δ = 0
                  ((= delta 0)  (setq reply "Nghiệm kép"
                                     x1 (/ (- b) (* 2 a))
```

```
                                    x2 x1
                                ))
                ;; trường hợp còn lại
                  (T    (setq reply "Hai nghiệm: ")
                              x1 (/ (- (sqrt delta) b) (* 2.0 a))
                              x2 (/ (+ (sqrt delta) b) (* -2.0 a))
                              ))
            ) ; kết thúc cond
        ) ; kết thúc progn
) ; kết thúc if
;;; xuất kết qủa
(if (= x1 NIL)
    (setq result reply)
    (setq result
        (strcat reply "\n" "X1 = " (rtos x1) "\n" "X2 = " (rtos x2))))
        ; hàm strcat ghép các chuỗi ký tự con thành chuỗi result
        ; "\n" - ký tự điều khiển, ngắt dòng (sang dòng mới)
        ; hàm rtos chuyển kiểu dữ liệu số thực sang chuỗi ký tự
(alert result)      ; hiện chuỗi ký tự result trong hộp thoại alert
```

6.2.4.5. Vòng lặp

Một cấu trúc cơ bản khác trong lập trình là các vòng lặp. AutoLISP cung cấp các hàm *repeat*, *while*, *foreach* và *mapcar* cho mục đích này:

- Hàm *repeat* có cú pháp sau:

  (Repeat n [expr...])

trong đó

n - số lần lặp (nhất thiết là một số nguyên dương).

expr... - một hoặc nhiều biểu thức.

Hàm *repeat* thực hiện lặp *n* lần các biểu thức *expr* và trả về giá trị biểu thức thực hiện sau cùng. Nếu trong hàm không có biểu thức nào giá trị NIL sẽ được trả về. Ví dụ sau đây tính tổng các số từ 1 đến 10:

Ví dụ 6.9: *Sử dụng vòng lặp repeat*

```
(setq tong 0)                   ; gán giá trị ban đầu 0 cho tổng (tong = 0)
(setq sohang 1)                 ; gán giá trị biến đếm sohang = 1
(repeat 10                      ; lặp 10 lần các biểu thức trong vòng lặp
  (setq tong (+ tong sohang))
  (setq sohang (1+ sohang))
) ; hết repeat.
```

Kết quả sau vòng lặp này *tong* nhận giá trị 55, *sohang* nhận giá trị 11 và hàm *repeat* cũng nhận giá trị 11.

- Hàm *while* có cú pháp sau:

(while testexpr [expr...])

trong đó

testexpr - biểu thức đánh giá điều kiện lặp.

expr... - một hoặc nhiều biểu thức thực hiện.

Hàm *while* trước tiên đánh giá biểu thức *testexpr*, nếu trả về giá trị khác NIL, các biểu thức *expr* sẽ được thực hiện và trả về giá trị biểu thức thực hiện sau cùng. Vòng lặp kết thúc khi *testexpr* trả về giá trị NIL. Đoạn chương trình tính tổng các số từ 1 đến 10 trên đây có thể viết lại như sau:

Ví dụ 6.10: *Sử dụng vòng lặp while*

```
(setq tong 0)              ; gán giá trị ban đầu 0 cho tổng (tong = 0)
(setq sohang 1)            ; gán giá trị biến đếm sohang = 1
(while (<= sohang 10)      ; kiểm tra xem số hạng có lớn hơn 10
  (setq tong (+ tong sohang))
  (setq sohang (1+ sohang))
) ; while
```

Kết quả trả về cũng như ví dụ 6.9.

Để tạo ***vòng lặp vô tận*** có thể sử dụng cú pháp sau:

```
(while T          ; điều kiện luôn đúng; có thể thay bằng (while 1 ...)
  (...)
  (...)
)
```

khi đó để thoát khỏi vòng lặp cần tác dụng từ bên ngoài (nhấn Ctrl-Break)

- Hàm *foreach* có cú pháp sau:

(foreach ele lst [expr...])

trong đó

ele- tên biến, được gán lần lượt giá trị các phần tử trong danh sách *lst*.

lst - danh sách (hằng, biến hoặc biểu thức cho kết quả là một danh sách.

expr... - một hoặc nhiều biểu thức thực hiện.

Hàm *foreach* gán lần lượt giá trị các phần tử trong danh sách *lst* cho biến *ele* và thực hiện các biểu thức *expr*. Quá trình lặp kết thúc khi thực hiện xong với phần tử cuối cùng của danh sách đã cho. Hàm trả về giá trị mới nhất của biểu thức cuối cùng. Nếu không có biểu thức nào, hàm trả về giá trị NIL.Ví dụ tính tổng 10 số đầu tiên ở trên có thể thực hiện qua hàm *foreach* như sau:

Ví dụ 6.11: *Sử dụng vòng lặp foreach*

```
; tạo danh sách gồm 10 phần tử các số hạng cần tính tổng
(setq lst '(1 2 3 4 5 6 7 8 9 10))
(setq tong 0)
(foreach sohang lst
        (setq tong (+ tong sohang))
) ; het foreach
```

Hàm *foreach* thường dùng để xử lý các phần tử trong danh sách, ví dụ, truy cập và xử lý cơ sở dữ liệu của đối tượng bản vẽ.

- Hàm *mapcar* có cú pháp sau:

  (mapcar 'func lst1 ... lstn])

trong đó

func- tên một hàm AutoLISP nào đó.

lst1,..., *lstn* - danh sách, số lượng phải thích hợp với số lượng tham số của hàm *func*.

Hàm *mapcar* thực hiện hàm *func* với các đối số là các danh sách *lst*, hàm trả về một danh sách mới từ kết quả thực hiện hàm trên.

Thực chất thì hàm *mapcar* không phải là vòng lặp, tuy nhiên có thể sử dụng để xử lý các danh sách tương tự như *foreach*. Ví dụ sau minh họa việc xử lý danh sách các điểm cho trong *plist* và tạo ra 2 danh sách mới *xlist* và *ylist* chứa tương ứng các tọa độ x và y của các điểm này, sau đó các điểm được thêm tọa độ z trong danh sách *zlist*. Kết quả có thể dùng để sắp xếp các điểm đã cho theo một trật tự nào đó hoặc biến điểm trong mặt phẳng thành điểm trong không gian.

Ví dụ 6.12: *Sử dụng hàm mapcar*

```
(setq plist '((1 10) (2 20) (3 30) (4 40)))     ; danh sách các điểm plist
(setq xlist (mapcar 'car plist))        ; lấy thành phần đầu (tọa độ x) của các
                                        ; điểm và trả về dưới dạng danh
                                        ; sách list '(1 2 3 4)
(setq ylist (mapcar 'cadr'plist))       ; lấy thành phần thứ 2 (tọa độ y)
```

```
                                        ; của các điểm và trả về dưới dạng
                                        ; danh sách ylist '(10 20 30 40)
(setq zlist '(100 200 300 400))         ; tọa độ z
(setq newpl (mapcar 'list xlist ylist zlist))  ; thêm tọa độ z cho cácđiểm,
        ; trả về danh sách các điểm mới newpl trong không gian 3 chiều
        ; '((1 10 100) (2 20 200) (3 30 300) (4 40 400))
(princ xlist)                           ; xuất kết quả
(princ ylist)
(princ newpl)
```

Trong ví dụ trên các hàm được *mapcar* thực hiện gồm *car* - lấy thành phần đầu của danh sách, *cadr* - lấy thành phần thứ 2 và *list* - tạo danh sách từ các phần tử.

6.3. LẬP TRÌNH ỨNG DỤNG BẰNG AUTOLISP

Nhìn chung ngôn ngữ AutoLISP có thể thực hiện được hầu hết các chức năng như các ngôn ngữ lập trình khác, tuy nhiên do cách viết tương đối rắc rối và chạy trong nền AutoCAD nên thường chỉ được sử dụng trong thiết kế tự động, nhất là trong việc xây dựng các bản vẽ thiết kế. Việc tổ chức nhập và xuất dữ liệu là bước cơ bản đầu tiên trong lập trình.

6.3.1. Nhập xuất dữ liệu

Dữ liệu có thể được nhập qua hai cách: qua đối thoại hoặc nhập từ file đã được chuẩn bị trước. Với AutoLISP, đối thoại được thực hiện qua hai phương thức: trực tiếp trên dòng lệnh command của AutoCAD qua các lời nhắc hoặc thông qua hộp thoại (Dialog), tương tự như các lệnh của AutoCAD.

6.3.1.1. Thông báo

Thông báo cũng được đưa ra thông qua hai cách: trên dòng lệnh hoặc qua hộp thoại. Ngoài các thông báo kèm với các hàm nhập dữ liệu (getX), AutoLISP còn sử dụng các hàm sau:

- ***(prompt msg)***
- ***(alert msg)***

Hàm *prompt* đưa thông báo ra dòng lệnh command, còn hàm *alert* đưa thông báo ra hộp thoại chuẩn của AutoCAD. Trong cả hai hàm trên *msg* là chuỗi ký tự (String), thể hiện nội dung cần thông báo. Nếu muốn thể hiện trên nhiều dòng, hãy thêm ký hiệu \n để chuyển sang dòng mới. Các hàm này đều trả về giá trị NIL.

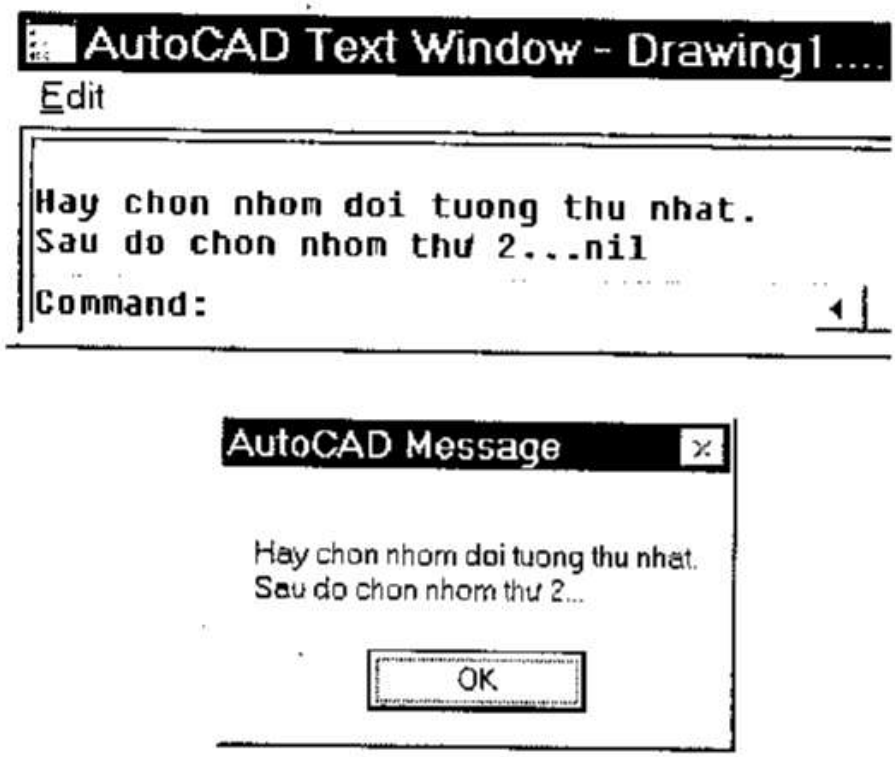
Ví dụ (hình 6.4):

(prompt "\nHay chon nhom doi tuong thu nhat.\nSau do chon nhom thu 2...")

cho hai dòng thông báo trên dòng lệnh.

(alert "\nHay chon nhom doi tuong thu nhat.\nSau do chon nhom thu 2...")

cho hai dòng thông báo trên hộp thoại AutoCAD Message.



Hình 6.4. Thông báo người dùng

6.3.1.2. Nhập dữ liệu trực tiếp trên dòng lệnh

Các hàm sử dụng để nhập dữ liệu trực tiếp trên dòng lệnh command thường có cú pháp chung như sau:

(getX [msg] [...])

trong đó

getX - tên hàm dùng nhập dữ liệu, X thường là kiểu của dữ liệu

msg - chuỗi ký tự thể hiện lời nhắc, hiện trên dòng lệnh AutoCAD khi chạy chương trình

[...] - các thông số khác của hàm

Nếu thực hiện thành công các hàm này sẽ trả về giá trị đã nhập.

Sau đây là một số hàm nhập liệu thường dùng.

- ***(getint [msg])*** dùng để nhập số nguyên (từ -32768 đến +32767). Hàm xuất hiện lời nhắc *msg* (nếu có) trên dòng lệnh, sau đó đợi người

dùng nhập một số nguyên bằng cách gõ các chữ số từ bàn phím và nhấn phím Enter hoặc phím Space để kết thúc. Nếu nhập đúng, hàm trả về số nguyên đã nhập. Nếu dãy ký tự gõ vào không phải là số nguyên AutoCAD sẽ thông báo lỗi và nhắc nhập lại. Nếu người dùng không nhập gì mà nhấn ngay Enter hoặc Space, hàm sẽ trả về NIL. Để hạn chế dữ liệu nhập (không cho nhập ngay Enter, không cho nhập số âm...) cần gọi hàm *initget* với các thông số thích hợp trước khi gọi hàm *getint*. Hàm *initget* sẽ được giới thiệu kỹ ở cuối phần này.

Ví dụ, câu lệnh *(setq inum (getint "\nHãy nhập một số nguyên: "))* khi thực hiện sẽ hiện lời nhắc *"Hãy nhập một số nguyên: "* trên dòng lệnh và chờ người dùng nhập số cần thiết, kết quả của hàm được gán cho biến *inum*. Như vậy *inum* sẽ là một số nguyên hoặc có giá trị NIL. Hai ký tự "*\n*" đưa vào để lời nhắc được thể hiện trên dòng mới.

- ***(getreal [msg])*** dùng để nhập một số thực từ bàn phím. Lời nhắc *msg* giúp người dùng biết phải nhập dữ liệu nào. Cũng như hàm *getint*, hàm trả về NIL nếu người dùng nhấn ngay phím Enter hoặc Space mà không nhập gì, còn nếu chuỗi ký tự gõ vào không phải là số thì AutoCAD sẽ thông báo lỗi và yêu cầu nhập lại.

Câu lệnh *(setq rnum (getreal "\nHãy nhập một số thực: "))*.khi thực hiện sẽ hiện lời nhắc *"Hãy nhập một số thực: "* trên dòng lệnh và chờ người dùng nhập số cần thiết; kết quả của hàm được gán cho biến *rnum*. Lưu ý rằng người dùng có thể gõ nhập số nguyên (ví dụ, 11), nhưng kết quả trả về vẫn là số thực (11.0).

- ***(getstring [cr] [msg])*** dùng để nhập chuỗi ký tự, trong đó các tham số có ý nghĩa như sau:

cr - nếu có và khác NIL, hàm cho phép nhập chuỗi ký tự gồm cả các dấu trắng (gõ bằng phím Space). Khi đó cần nhấn Enter để kết thúc chuỗi ký tự cần nhập. Ngược lại, hàm không cho phép nhập dấu trắng. Phím Space trong trường hợp này có tác dụng như phím Enter.

msg - lời nhắc, xuất hiện trên dòng lệnh command.

Khi nhấn ngay Enter mà không nhập gì, hàm trả về NIL. Ngược lại hàm trả về chuỗi ký tự đã nhập. Ví dụ:

```
(setq sHoten (getstring "\nHọ và tên: "))
```

trên dòng lệnh sẽ hiện lời nhắc "Họ và tên: ". Nếu người dùng nhập Tran và dấu cách (Space) thì hàm nhập liệu kết thúc và kết quả "Tran" được gán cho biến *sHoten*. Nếu muốn nhập đầy đủ họ tên (chẳng hạn, Tran Van Binh), cần viết lại câu lệnh trên:

```
(setq sHoten (getstring T "\nHọ và tên: "))
```

sau đó nhập Tran Van Binh rồi nhấn Enter để kết thúc.

Để đưa thêm các ký tự đặc biệt vào chuỗi (như các dấu " - nháy kép, - xổ chéo trong đường dẫn file,...) cần thêm vào trước chúng dấu xổ chéo.

Ví dụ, khi thực hiện biểu thức:

```
(setq sTenfile (getstring "\nNhập tên file kèm đường dẫn: "))
```

trên dòng lệnh command của AutoCAD sẽ xuất hiện câu nhắc:

```
Nhập tên file kèm đường dẫn:
```

Nếu muốn nhập đường dẫn "C:\AutoCAD2000\VLISP\TFILE.LSP" cần gõ chuỗi ký tự sau:

```
C:\\AutoCAD2000\\VLISP\\TFILE.LSP và kết thúc bằng nhấn Enter.
```

Một số ký tự đặc biệt khác:

\n	dòng mới
\r	Enter (đưa con trỏ về đầu dòng)
\"	nháy kép
\t	khoảngTtab (bảng)
\e	Escape
\nnn	hiện ký tự có mã ASCII nnn (cơ số 8)

- ***(getpoint [pt] [msg])*** nhập một điểm bằng cách kích chuột hoặc gõ tọa độ như trong các câu lệnh AutoCAD, trong đó các tham số có ý nghĩa như sau:

  pt - điểm tham chiếu, nếu có thì trên màn hình AutoCAD sẽ xuất hiện một đoạn thẳng (Rubber Band) nối điểm này với vị trí con trỏ chuột.

  msg - lời nhắc, xuất hiện trên dòng lệnh command.

Hàm trả về điểm vừa nhập dưới dạng một danh sách (List) gồm 3 phần tử thể hiện các tọa độ x, y và z của nó. Các điểm này được sử dụng cùng hàm *command* để tạo các đối tượng hình học qua các lệnh AutoCAD thông dụng hoặc có thể dùng các hàm với danh sách để xử lý riêng từng tọa độ. Ví dụ, đoạn chương trình sau vẽ vòng tròn qua hai điểm mút đường kính do người dùng nhập:

```
(setq   pt1 (getpoint "\nNhập điểm đầu: ")    ; nhắc nhập điểm đầu
        pt2 (getpoint pt1 "\nNhập điểm đối diện: ") ; điểm đối diện
        )                                     ; với điểm tham chiếu là điểm đầu
; gọi lệnh Circle của AutoCAD để vẽ vòng tròn.
(command "._Circle" "2P" pt1 pt2)
```

- ***(getdist [pt] [msg])*** nhập khoảng cách. Giá trị trả về của hàm là số thực hoặc NIL nếu người dùng nhấn ngay Enter. Các tham số trong hàm

gồm:

pt - điểm tham chiếu, nếu có thì trên màn hình AutoCAD sẽ xuất hiện một đoạn thẳng nối điểm này với vị trí con trỏ chuột.

msg - lời nhắc, xuất hiện trên dòng lệnh command.

Sau khi xuất hiện lời nhắc trên dòng lệnh, có thể nhập khoảng cách theo các cách sau:

- Gõ một số thể hiện khoảng cách cần nhập và nhấn Enter.
- Nhập 1 điểm (khi thông số *pt* được đưa trong lời gọi hàm) - giá trị trả về của hàm sẽ là khoảng cách từ điểm đã nhập đến điểm tham chiếu *pt*.
- Nhập 2 điểm (khi thông số *pt* được bỏ qua) - giá trị trả về của hàm sẽ là khoảng cách giữa hai điểm đã nhập.

Đoạn chương trình sau yêu cầu nhập tâm vòng tròn và bán kính, sau đó vẽ vòng tròn bằng lệnh AutoCAD:

```
(setq   tam (getpoint "\nNhập tâm: ")     ; nhắc nhập tâm vòng tròn
        bk (getdist pt1 "\nNhập bán kính: ") ; nhắc nhập bán kính
        )                                 ; có thể kích chuột tại điểm trên vòng tròn
                  ; gọi lệnh Circle của AutoCAD để vẽ vòng tròn.
(command ".Circle" tam bk)
```

- ***(getangle [pt] [msg])*** nhập góc. Giá trị trả về của hàm là số thực thể hiện góc đã nhập, đo bằng radian hoặc NIL nếu người dùng nhấn ngay Enter. Các tham số trong hàm gồm:

pt - điểm tham chiếu, nếu có thì trên màn hình AutoCAD sẽ xuất hiện một đoạn thẳng nối điểm này với vị trí con trỏ chuột.

msg - lời nhắc, xuất hiện trên dòng lệnh command.

Sau khi xuất hiện lời nhắc trên dòng lệnh, góc được nhập theo các cách sau:

- Gõ giá trị góc cần nhập theo đơn vị mặc định, được thiết lập trước bởi lệnh *Unit* của AutoCAD (thường là độ) và nhấn Enter.
- Nhập 1 điểm (khi thông số *pt* được đưa trong lời gọi hàm) - giá trị trả về của hàm sẽ là góc tạo bởi đường nối điểm đã nhập đến điểm tham chiếu *pt* và trục X.
- Nhập 2 điểm (khi thông số *pt* được bỏ qua) - giá trị trả về của hàm sẽ là góc tạo bởi đường nối hai điểm đã nhập với trục X.

Lưu ý rằng, giá trị trả về của hàm *getangle* luôn là radian và có giá trị từ 0 đến 2π. Ví dụ, đoạn chương trình sau:

```
(setq  pt  '(0 0)        . ; điểm tham chiếu là gốc tọa độ
       a (getangle pt "Hãy nhập giá trị góc xoay: ")
       )
```

Khi chạy đoạn chương trình này, trên dòng lệnh sẽ xuất hiện lời nhắc yêu cầu người dùng nhập giá trị góc xoay. Nếu nhập 180, hàm sẽ trả về 3.14159, nhập 360 - trả về 0, nhập 540 (= 360+180) - hàm trả về 3.14159, nhập -90, hàm trả về 4.71238 ($3\pi/2$ chứ không phải $-\pi/2$). Nếu nhập điểm (1 .0,1.0) - hàm sẽ trả về 0.785399 ($\pi/4$), nếu nhập điểm (1.0, -1.0) - trả về giá trị 5.44779 ($7\pi/4$ chứ không phải $-\pi/4$).

- ***(getkword [msg])*** nhập từ khoá (Keyword). Giá trị trả về của hàm là chuỗi ký tự ứng với từ khoá đã được định nghĩa bởi hàm *initget* trước đó. Từ khoá có thể nhập bằng cách gõ đầy đủ hoặc gõ các ký tự viết tắt cho từ khoá. Biến *msg* thể hiện một thông báo trên dòng lệnh để người sử dụng biết cần phải nhập gì. Nếu nhập từ khoá sai, sẽ xuất hiện báo lỗi và nhắc lại thông báo *msg* để người dùng nhập lại. Ví dụ, đoạn chương trình sau yêu cầu người dung nhập từ khoá T (Tiep) hoặc K (Khong):

```
(initget 1 "Tiep Khong")  ; định nghĩa từ khoá T và K, bitcode 1 có
                          ; nghĩa là không cho phép nhấn ngay Enter
                          ; mà không nhập gì.
; Hiện thông báo "Có tiếp tục..." và đợi người dùng nhập từ khoá
(setq traloi (getkword "Có tiếp tục không (Co/Khong) ?"))
```

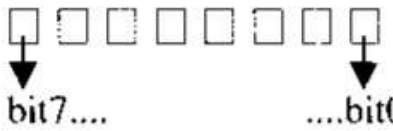
Trong ví dụ này, nếu nhập "K" hoặc "Khong", hàm *getkword* sẽ trả về giá trị "Khong" và gán cho biến *traloi*, còn nếu nhập "T" hoặc "Tiep", biến *traloi* sẽ được gán giá trị "Tiep". Ngược lại, hàm *getkword* sẽ báo lỗi và yêu cầu nhập lại. Lưu ý hàm *getkword* không phân biệt chữ hoa hay chữ thường khi nhập từ khoá. Chẳng hạn nhập "T" hay "t" sẽ cho kết quả như nhau là "Tiep".

Các phương pháp khai báo chuỗi từ khoá xem chi tiết hàm *initget*.

- ***(initget [bits] [str])*** thiết lập từ khoá và hạn chế loại dữ liệu được nhập cho hàm getX theo sau nó. Ngoại trừ hàm *getstring* không bị ảnh hưởng của *initget*, các hàm nhập liệu khác đều bị hàm này chi phối. Hàm *initget* trả về giá trị NIL.

Các tham số của hàm có ý nghĩa như sau:

- ***bits*** là một số nguyên (8-bit). Số nguyên 8-bit được viết dưới dạng cơ số 2 thành một chuỗi 8 ô (các bit) chứa các chữ số 0 và 1. Giá trị của các bit này tương ứng với việc cho phép hay không cho phép nhập một kiểu dữ liệu nào đó. Ý nghĩa của một số bit cho trong bảng 6.2.



Bảng 6.2. Các bit trong hàm initget

Bits	*Ý nghĩa*	*Ghi chú*
1	Không cho phép nhấn ngay Enter (NIL)	bit0 có giá trị 1
2	Không cho phép nhập giá trị 0 (zero)	bit1 có giá trị 1
4	Không cho phép nhập số âm	bit2 có giá trị 1
8	Cho phép nhập điểm nằm ngoài vùng giới hạn vẽ, xác định bởi lệnh LIMIT	bit3 có giá trị 1

Nếu muốn cấm nhập cả NIL và số 0 thì cần đặt các bit0 và bit1 cùng giá trị 1, khi đó bits sẽ có giá trị là 3 (= 2+1), hoặc nếu không muốn người dùng nhập số âm và số 0 (cho phép nhấn ngay Enter - nhập NIL) thì bit0 có giá trị 0, bit1 và bit2 có giá trị 1, khi đó giá trị bits sẽ là 6 (= 2+4). Các câu gọi hàm sau thực hiện việc hạn chế nhập liệu này:

```
(initget 3)   ; hoặc (initget (+ 1 2)) - không cho phép nhập NIL và 0
(initget 6)   ; hoặc (initget (+ 2 4)) - không cho phép nhập số âm và 0
```

- ***str*** là chuỗi ký tự thể hiện từ khoá, dùng với hàm *getkword*. Chuỗi ký tự này cần được thiết lập theo quy tắc sau:

+ Các từ khoá được phân cách nhau bởi một hay nhiều dấu cách (dấu trắng Space), ví dụ chuỗi sau là hợp lệ: "LType LIne eXit toP".

+ Các từ khoá chỉ có thể chứa các chữ cái, chữ số và dấu gạch ngang.

+ Có 2 phương pháp thể hiện cụm từ viết tắt của từ khoá:

- Viết hoa một hay một số ký tự liền nhau trong chuỗi từ khoá. Cụm từ viết tắt có thể đặt ở vị trí bất kỳ trong chuỗi từ khoá. Trong ví dụ trên các cụm viết tắt là LT (thay cho LType), LI (thay cho LIne), X (cho eXit) và P (thay cho toP). Thường hay sử dụng 1 chữ cái để thể hiện cụm từ viết tắt, nhưng khi cần phân biệt 2 từ khoá khác nhau có thể cần dùng nhiều từ hơn (ví dụ LT và LI).
- Các từ khoá được viết hoa, liền theo sau là cụm từ viết tắt của từ khoá tương ứng. Trong trường hợp này, cụm từ viết tắt phải chứa chữ cái đầu tiên của từ khoá. Ví dụ, chuỗi ký tự "LTYPE,LT LINE,LI" là hợp lệ, nhưng "EXIT,X" là không hợp lệ.

Lưu ý rằng, việc sử dụng từ khoá không chỉ dùng cho hàm *getkword*, mà còn có thể sử dụng cho các hàm getX khác. Ví dụ sau đây sử dụng hàm *initget* và *getreal* để nhập giá trị góc từ dòng lệnh command:

Ví dụ 6.13: Sử dụng initget

```
(initget 4 "Pi Two-pi thRee-pi")
        ; không cho phép nhập số âm, đặt các
        ; từ khoá Pi, Two-pi và thRee-pi
(setq ang (getreal "Nhập góc Pi/Two-pi/thRee-pi <Pi> "))
(if (= ang NIL) (setq ang 3.14159))
        ; nếu người dùng nhấn ngay Enter, gán ang = π
(if (= ang "Pi") (setq ang 3.14159))
        ; nếu người dùng nhập P (hoặc Pi), gán ang = π
(if (= ang "Two-pi") (setq ang 6.28318))
        ; nếu người dùng nhập T, gán ang = 2π
(if (= ang "thRee-pi") (setq ang 9.42477))
        ; nếu người dùng nhập R, gán ang = 3π
```

Trong đoạn chương trình này, dòng lệnh thứ 3 xử lý trường hợp người dùng nhấn ngay Enter mà không nhập gì. Khi đó chương trình sẽ gán cho biến *ang* giá trị mặc định π. Các dòng 4,5,6 xử lý các trường hợp người dùng nhập từ khoá thay cho giá trị. Biến *ang* sẽ được gán các giá trị tương ứng. Điều này rất hữu ích khi muốn người dùng nhập các giá trị định trước nào đó mà không cần gõ nhiều. Nếu người dùng nhập một số nào đó không âm, biến *ang* sẽ nhận giá trị số vừa nhập. Trường hợp người dùng nhập các giá trị khác (số âm hoặc chuỗi ký tự không đúng với từ khoá, chẳng hạn nhập 2pi hoặc -3.14) thì AutoCAD sẽ yêu cầu nhập lại.

Một cách thức nhập liệu khác được thực hiện qua giao tiếp hộp thoại (Dialog) sẽ được đề cập chi tiết tại phần 6.4.

6.3.1.3. Chuyển đổi kiểu dữ liệu

Một nhu cầu không thể thiếu trong lập trình là chuyển dữ liệu từ kiểu này sang kiểu khác. Ví dụ, cần thông báo chu vi của vòng tròn đã chọn lên hộp Alert, trước tiên cần tìm bán kính vòng tròn (qua CSDL AutoCAD), tính chu vi và chuyển giá trị tính được từ kiểu số thực (Real) sang kiểu chuỗi ký tự (String). Ngược lại, khi nhập số liệu bằng hộp thoại DCL, giá trị nhận được là một chuỗi ký tự chứa các số, do vậy cần chuyển chúng thành kiểu dữ liệu số để có thể tính toán.

AutoLISP cung cấp một loạt các hàm chuyển đổi dữ liệu. Cú pháp chung của các hàm này thường có dạng StoD, trong đó S là kiểu nguồn, D là kiểu cần chuyển tới. Dưới đây giới thiệu một số hàm thông dụng:

- ***(atof str)*** chuyển đổi chuỗi ký tự số *str* sang số thực. Ví dụ:

(atof "97.1") trả về số thực 97.1

(atof "97") trả về số thực 97.0

Tham số *str* phải là một chuỗi String, nếu là số thì hàm *atof* sẽ báo lỗi. Lưu ý rằng, nếu chuỗi *str* chứa cả các ký tự không hợp lệ (không phải là các chữ số và dấu thập phân) hàm *atof* vẫn thực hiện. Kết quả trả về số thực 0.0 nếu chuỗi *str* không hợp lệ ngay từ chữ đầu tiên, còn không sẽ trả về số tương ứng với các chữ hợp lệ tính từ trái sang của chuỗi *str*. Ví dụ:

(atof "123.4a15") trả về số 123.4

(atof "a.123") hoặc (atof "") trả về 0.0

- ***(atoi str)*** chuyển đổi chuỗi ký tự sang số nguyên. Cách thức thực hiện giống như hàm *atof*. Ví dụ:

(atoi "123") trả về số nguyên 123

(atoi "123.987") trả về số nguyên 123

(atoi "123abc") trả về số nguyên 123

(atoi "0123") trả về số nguyên 123

(atoi "") hoặc (atoi "xyz") trả về số nguyên 0

Lưu ý rằng, giá trị trả về của hàm là số nguyên dài (LongInt) 32-bit. Nếu chuỗi ký tự ứng với số lớn hơn giới hạn 32-bit, kết quả sẽ là một số khác. Ví dụ, biểu thức *(atoi "1234567889990")* sẽ trả về số nguyên 2147483647.

- ***(distof str [mode])*** chuyển đổi chuỗi ký tự dạng số đo khoảng cách sang số thực. Các tham số có ý nghĩa như sau:
  - *str:* chuỗi ký tự cần chuyển đổi (cách thể hiện xem hàm *rtos* trình bày ở phần sau).
  - *mode:* kiểu thể hiện số thực trong chuỗi ký tự (tương tự hàm *rtos* trình bày ở phần sau).

Nếu chuỗi hợp lệ theo *mode* đã cho hàm sẽ trả về số thực, ngược lại hàm trả về NIL.

Ví dụ:

(distof "17.5" 2) trả về số thực 17.5

(distof "1.7500E+01" 1) trả về giá trị 17.5

(distof "1'-5.5000\"" 3) trả về 17.5, nhưng

(distof "1'-5.5000\"" 2) trả về giá trị NIL do chuỗi "1'-5.5000\"" không hợp lệ khi thể hiện số kiểu thập phân (mode = 2).

- ***(angtof str [mode])*** chuyển đổi chuỗi ký tự dạng số đo góc sang

số thực thể hiện giá trị của góc này bằng radian (nằm trong khoảng 0 đến 2π). Các tham số có ý nghĩa như sau:

- *str:* chuỗi ký tự (cách thể hiện xem hàm *angtos* trình bày ở phần sau).
- *mode:* kiểu thể hiện góc trong chuỗi ký tự (tương tự hàm *angtos* trình bày ở phần sau).

Ví dụ:

(angtof "45.0000" 0) trả về số thực 0.785398 (π/4): chuỗi ký tự "45.0000" thể hiện góc 45 độ (mode = 0), chuyển sang radian được số thực π/4, nhưng

(angtof "-45.0000" 0) trả về giá trị 5.49779 (7π/4 = 2π - π/4)

(angtof "45.0000" 3) trả về số thực 1.0177: chuỗi ký tự "45.0000" thể hiện góc 45 radian (mode = 3), sau khi loại bớt các góc tròn 2π, còn lại giá trị 1.0177 radian

(angtof "45r" 3) cũng cho kết quả tương tự (1.0177)

Nếu chuỗi *str* không hợp lệ, hàm trả về NIL (ví dụ, biểu thức (angtof "45.00x") trả về NIL).

• ***(ascii str)*** trả về số nguyên ứng với mã ASCII của chữ cái đầu trong chuỗi *str*. Ví dụ, biểu thức *(ascii "ABC")* trả về số nguyên 65.

• ***(chr int)*** ngược lại với hàm *ascii*, hàm này trả về chuỗi chứa chữ cái có mã tương ứng với giá trị mã ASCII bằng *int* đã cho. Ví dụ, biểu thức *(chr 65)* trả về chuỗi ký tự "A", nhưng *(chr 12)* lại trả về chuỗi "\014", thể hiện số 12 trong hệ đếm cơ số 8.

• ***(rtos num [mode [precission]])*** chuyển đổi số num sang chuỗi ký tự. Các tham số của hàm như sau:

- *num:* số cần chuyển đổi.
- *mode:* số nguyên dương thể hiện kiểu thể hiện chuỗi chữ số, giá trị và ý nghĩa cho trong bảng 6.3.
- *precission:* số nguyên dương, thể hiện độ chính xác chuyển đổi (số chữ số sau dấu thập phân).

Trong các tham số này, hai tham số *mode* và *precision* không nhất thiết phải có. Trong trường hợp này kiểu thể hiện và độ chính xác được lấy theo mặc định, xác lập qua các biến LUNITS và LUPREC của AutoCAD.

• ***(itoa int)*** trả về chuỗi ký tự thể hiện số nguyên *int* trong tham số của hàm. Nói cách khác hàm *itoa* chuyển dữ liệu từ kiểu số nguyên sang

chuỗi ký tự. Ví dụ:

(itoa 123) trả về chuỗi "123", nhưng

(itoa 123456767889900) hàm không thực hiện do tham số vượt quá giới hạn 32-bit.

(itoa 123.456) cũng không thực hiện do không phải số nguyên.

Bảng 6.3. Ý nghĩa của tham số mode (hàm rtos)

Giá trị mode	*Kiểu thể hiện*	*Ví dụ*
1	Khoa học	(setq num 17.5) (rtos num 1 4) trả về chuỗi "1.7500E+01"
2	Thập phân	(rtos num 2 4) trả về chuỗi "17.5000"
3	Kỹ thuật (Anh)	(rtos 17.5 3 4) trả về chuỗi "1'-5.5000\"" *có nghĩa 1 ft 5.5 in - lưu ý 1 ft =12in*
4	Kiến trúc (Anh)	(rtos 17.5 4) trả về chuỗi "1'-5 1/2\"" *có nghĩa 1 ft 5 1/2 in (viết kiểu phân số)*
5	Phân số	(rtos 17.5 5) trả về chuỗi "17 1/2"

- ***(angtos ang [mode [precission]])*** chuyển đổi góc *ang* sang chuỗi ký tự. Các tham số của hàm như sau:

- *ang:* góc cần chuyển đổi (đo bằng radian).
- *mode:* số nguyên dương thể hiện kiểu thể hiện chuỗi chữ số, giá trị và ý nghĩa như trong bảng 6.4.
- *precission:* số nguyên dương, thể hiện độ chính xác chuyển đổi (số chữ số sau dấu thập phân).

Bảng 6.4. Ý nghĩa của tham số mode (hàm angtos)

Giá trị mode	*Kiểu thể hiện*	*Ví dụ, với góc ang = 0.785398 (π/4)*
0	Độ	(angtos ang 0 4) trả về chuỗi "45.0000"
1	Độ/phút/giây dạng XdY'Z"	(angtos ang 1) trả về chuỗi "45d0'0\"" *45 độ, 0 phút, 0 giây*
2	Gradian (1 g = 100 độ)	(angtos ang 2 4) trả về chuỗi "50.0000g" *50.0000 gradian*
3	Radian	(angtos ang 3 4) trả về chuỗi "0.7854r"
4	Surveyor's	(angtos ang 3 4) trả về chuỗi "N 45d0'0\" E" *45 độ, hướng Đông - Bắc (North-East)* (angtos -0.785398 4 4) trả về "S 45d0'0\" E" *45 độ, hướng Đông - Nam (South - East)*

Các tham số *mode* và *precision* không nhất thiết phải có. Khi đó, kiểu thể hiện và độ chính xác được lấy theo mặc định, xác lập qua các biến AUNITS và AUPREC của AutoCAD.

6.3.1.4. Xuất dữ liệu ra màn hình

Có thể xuất dữ liệu ra màn hình thông qua 2 hàm thông báo *prompt* và *alert* ở mục 6.1.1, nhưng dữ liệu cần phải chuyển đổi sang dạng chuỗi ký tự, và kết quả trả về của các hàm này là NIL. Các hàm sau đây xuất trực tiếp dữ liệu lên màn hình văn bản của AutoCAD. Dữ liệu không nhất thiết phải là dạng chuỗi ký tự, mà có thể thuộc dạng khác như số nguyên, số thực, danh sách,... hoặc là kết quả của một biểu thức AutoLISP.

(princ expr [file]) xuất kết quả ra mà hình mà không kèm theo dấu nháy kép " (đối với dữ liệu dạng văn bản).

(prin1 expr [file]) xuất nguyên kết quả ra màn hình (gồm cả ký tự nháy kép " với dữ liệu dạng văn bản)

(print expr [file]) xuất kết quả nguyên bản ra màn hình (gồm cả ký tự nháy kép " với dữ liệu dạng văn bản), nhưng bắt đầu ở dòng mới và có thêm dấu cách sau dữ liệu.

Với các hàm này, chỉ được phép duy nhất 1 biểu thức (hoặc tham số) *expr*, chẳng hạn phát biểu (princ 10 15) là không hợp lệ. Tham số *file* là con trỏ file, nhận được từ hàm mở file *open* được giới thiệu ở mục 6.3.1.5. Nếu có tham số này, dữ liệu sẽ được lưu vào file, còn không, dữ liệu được in ra màn hình. Các hàm này đều trả về giá trị của biểu thức (hoặc giá trị tham số) *expr*. Ví dụ, với chuỗi *str* có nội dung: *The "allowable" tolerance is 0.25in*, khi in ra bằng các hàm này sẽ cho kết quả tương ứng như sau:

(princ str)	in ra nội dung	*The "allowable" tolerance is 0.25in*
	và trả về chuỗi ký tự	"The \"allowable\" tolerance is 0.25in"
(prin1 str)	in ra nội dung	*"The \"allowable\" tolerance is 0.25in"*
	và trả về chuỗi ký tự	"The \"allowable\" tolerance is 0.25in"
(print str)	in ra nội dung	*<xuống dòng mới>*
		"The \"allowable\" tolerance is 0.25in"
		<dấu cách>
	và trả về chuỗi ký tự	"The \"allowable\" tolerance is 0.25in"

Khi sử dụng các hàm này với kiểu dữ liệu không phải chuỗi ký tự, hàm *princ* và *prin1* cho kết quả như nhau, còn *print* sẽ in kết quả ra dòng mới và thêm dấu cách vào sau kết quả. Ví dụ:

(princ (+ 2.5 2.4)) in ra số *4.9* và trả về số thực 4.9

(prin1 (+ 2.5 2.4)) in ra số *4.9* và trả về số thực 4.9

(print (+ 2.5 2.4)) thực hiện việc xuống dòng, in số *4.9 <dấu cách>* và trả về số thực 4.9

Khi gọi hàm AutoLISP, kết quả biểu thức cuối cùng của hàm được in ra màn hình. Để tránh việc này, thường thêm lệnh *princ* (không tham số) ở cuối hàm. Hàm sẽ trả về giá trị NIL và không in kết quả ra dòng lệnh command. Cách thức này trong AutoLISP được gọi là "rút lui im lặng!".

Hai hàm khác thường sử dụng với file là ***write-char*** và ***write-line*** cũng có thể dùng để ghi dữ liệu ra màn hình văn bản của AutoCAD. Hàm ***(write-char int)*** in ra màn hình ký tự có mã ASCII bằng số nguyên *int*, còn hàm ***(write-line str)*** in ra màn hình chuỗi ký tự *str* (không kèm theo dấu nháy kép " bao chuỗi ký tự) và sau đó sang dòng mới. Ví dụ, đoạn chương trình sau:

```
(write-char 65)
(write-char 66)
```

khi thực hiện sẽ in ra màn hình các ký tự AB (viết liền nhau trên cùng 1 dòng), còn nếu thực hiện các biểu thức:

```
(write-line "Abc")
(write-line "Def")
```

sẽ in ra màn hình 2 dòng với nội dung sau:

```
Abc
Def
```

Như vậy, có thể nói rằng hàm *(write-line str)* thay thế lời gọi hai hàm *(princ str) (print)*. Để viết các dòng Abc và Def trên hai dòng như ví dụ trên, nếu dùng *princ* và *print* cần viết lại đoạn chương trình như sau:

```
(princ "Abc")
(print)
(princ "Def")
(print)
```

6.3.1.5. Làm việc với file dữ liệu

File dữ liệu trong AutoLISP thuộc dạng văn bản, truy cập tuần tự. Đóng / mở file được thực hiện thông qua các hàm *open* và *close*. Dữ liệu được lưu và file qua các hàm *write-char* và *write-line*, còn việc đọc dữ liệu được thực hiện mỗi lần 1 dòng với hàm *read-line*. Ngoài ra AutoLISP còn cung cấp một số hàm khác như hàm *findfile* tìm xem file có tồn tại trong các đường dẫn khai báo trong mục Reference của AutoCAD hay không và hàm *getfiled* cho phép chọn file thông qua việc truy tìm trên cây

thư mục (Folder Tree) theo giao thức hộp thoại FileOpen hoặc FileSaveAs như với các ứng dụng Windows khác. VisualLISP còn cung cấp thêm các tính năng về file khác như đổi tên, xoá, sao chép file,...

Dưới đây giới thiệu chi tiết một số hàm thường dùng với file dữ liệu:

- ***Mở file*** được thực hiện thông qua hàm ***open***. Cú pháp chung của hàm như sau:

(open filename mode)

trong đó:

- *filename* - chuỗi ký tự (String) thể hiện tên file cần mở. Nếu không ghi đầy đủ đường dẫn, file được tìm trong Folder (thư mục) khởi động của AutoCAD.
- *mode* - kiểu mở, chuỗi ký tự, không phân biệt viết hoa hay viết thường, thể hiện mục đích mở file, bao gồm các lựa chọn:

"r" - mở file để đọc dữ liệu. File có tên đã cho cần phải tồn tại.

"w"- mở file để ghi dữ liệu. Nếu file chưa tồn tại, file mới sẽ được tạo ra, còn nếu đã tồn tại, nội dung cũ sẽ bị xoá để ghi dữ liệu mới. Thực chất, dữ liệu chỉ được ghi lên file trên đĩa sau khi thực hiện hàm *close*.

"a"- mở file ghi thêm dữ liệu. Nếu file chưa tồn tại, file mới sẽ được tạo ra, còn nếu đã tồn tại, dữ liệu mới sẽ được ghi tiếp sau dữ liệu đã có trong file.

Nếu hàm thực hiện thành công, kết quả trả về con trỏ file, được sử dụng như tham số với các hàm đọc và ghi dữ liệu. Nếu không thành công, hàm trả về NIL.

- ***Đóng file*** đã mở được thực hiện thông qua hàm ***close***. Cú pháp của hàm rất đơn giản:

(close file)

Trong đó *file* là con trỏ file, nhận được từ hàm *open* đã nói ở phần trên. Nếu tham số *file* hợp lệ, thực hiện thành công thì hàm trả về NIL, ngược lại sẽ xuất hiện thông báo lỗi.

Ví dụ sau thể hiện việc mở file có tên VIDU6-14.TXT, đọc dữ liệu và đóng file khi đọc xong.

Ví dụ 6.14: *Làm việc với file*

```
(setq fname "VIDU6-14.TXT")  ; tên file
(setq fil (open fname "r"))  ; mở file để đọc, gán con trỏ file cho biến fil
(setq  ct 0)                 ; đếm
```

```
(while (read-line fil)      ; vòng lặp while đọc từng dòng dữ liệu.
                            ; cho đến khi không đọc được nữa (cuối file)
        (setq ct (1+ ct))           ; đếm số dòng đã đọc
)                                   ; kết thúc vòng lặp
(close fil)                         ; đóng file
(princ "File chứa ")        ; thông báo số dòng dữ liệu có trong file
(princ ct)
(princ "dòng dữ liệu")
(princ)                     ; rút lui im lặng.
```

- ***Ghi dữ liệu vào file*** thường được thực hiện qua 2 hàm *write-char* và *write-line*. Hàm thứ nhất ghi từng ký tự vào file đã mở bằng hàm *open* với kiểu mở "w" hoặc "a", còn hàm thứ 2 ghi vào file dòng dữ liệu dạng văn bản. Các hàm này đã giới thiệu qua ở phần trên khi ghi dữ liệu ra màn hình, khi sử dụng để ghi dữ liệu vào file, chúng có cú pháp như sau:

(write-char int file)

(write-line str file)

trong đó

int - số nguyên thể hiện mã ASCII của ký tự cần ghi vào file.

str - chuỗi ký tự cần ghi vào file.

file - con trỏ file, nhận được từ kết quả hàm *open*. Nếu bỏ qua tham số này, kết quả sẽ được ghi lên màn hình văn bản của AutoCAD như đã trình bày trong mục 6.3.1.4.

Lưu ý rằng dữ liệu chỉ được thực sự ghi vào file trên đĩa sau khi thực hiện hàm *close* để đóng file đã mở.

Ví dụ sau đây lưu dữ liệu vào file VIDU6-15.TXT trong folder VLISP của ổ đĩa D. Dữ liệu gồm 3 dòng: dòng thứ nhất ghi các ký tự có mã ASCII 65, 66, 67 (ABC), dòng thứ 2 ghi nội dung "write-line: dong 2", còn dòng 3 có nội dung "write-line: dong 3".

Ví dụ 6.15: *Ghi dữ liệu vào file*

```
(setq fname "D:/VLISP/VIDU6-15.TXT")        ; khai báo tên file
; hoặc có thể thay bằng "D:\\VLIST\\VIDU2.TXT"
(setq fil (open fname "w"))     ; mở file để ghi dữ liệu
(if fil                         ; nếu mở file thành công
   (progn
      (write-char 65 fil)       ; ghi các ký tự A, B, C trên dòng 1
      (write-char 66 fil)
      (write-char 67 fil)
```

```
        (write-line "" fil)          ; sang dòng mới
        (write-line "write-line: dong 2" fil)      ; ghi dòng thứ 2
        (write-line "write-line: dong 3" fil)      ; ghi dòng 3
        (close fil)                                ; đóng file.
    )
    (prompt "Không mở được file...")  ; thông báo không mở được file
) ; hết hàm if
```

Sau khi đoạn chương trình thực hiện, nếu mở file trên, sẽ thấy nội dung sau:

```
ABC
write-line: dong 2
write-line: dong 3
<xuống dòng>
```

Có thể thấy rằng nếu folder VLISP chưa tồn tại trên ổ đĩa D thì trên dòng lệnh AutoCAD sẽ thấy thông báo lỗi: *Không mở được file...; error: bad argument type: streamp nil* do không mở được file.

Như đã thấy, các hàm này chỉ lưu dữ liệu dạng ký tự hoặc văn bản. Nếu muốn ghi trực tiếp các dữ liệu kiểu khác như danh sách, số nguyên, số thực,... cách tốt hơn là sử dụng các hàm *princ*, *prin1* và *print* đã giới thiệu ở mục 6.3.1.4.

- ***Đọc dữ liệu từ file*** được thực hiện qua các hàm *read-char* và *read-line*. Cú pháp chung của các hàm này như sau:

(read-char [file])

(read-line [file])

trong đó, tham số *file* là con trỏ file, nhận được từ kết quả hàm *open*. Nếu bỏ qua tham số *file*, các hàm này sẽ đọc dữ liệu từ bộ đệm bàn phím.

Hàm *read-char* đọc từng ký tự và trả về số nguyên là mã ASCII của ký tự đó. Nếu gọi tiếp hàm một lần nữa, hàm sẽ đọc ký tự tiếp theo. Khi kết thúc dòng, hàm trả về số 10, tương ứng với ký tự hết dòng và chuyển con trỏ xuống dòng kế tiếp. Ví dụ, nếu gõ trên dòng lệnh AutoCAD biểu thức *(read-char)*, AutoCAD sẽ chờ nhập dữ liệu. Nếu người dùng gõ AB và nhấn Enter, hàm trả về số nguyên 65, ứng với ký tự A. Nếu gọi tiếp hàm *(read-char)* một lần nữa, sẽ nhận được ngay số nguyên 66 (ứng với ký tự B trong bộ đệm bàn phím) mà không cần chờ người dùng nhập gì cả. Gọi thêm hàm *(read-char)* một lần nữa sẽ nhận được số 10, ứng với ký tự xuống dòng.

Hàm *read-line* đọc dòng văn bản từ vị trí con trỏ và trả về chuỗi ký tự

ứng với dòng đã được đọc, không kèm theo ký tự xuống dòng. Con trỏ được rời xuống dòng tiếp theo. Khi hết file, hàm trả về giá trị NIL.

Đoạn chương trình sau đây minh họa việc sử dụng hai hàm này để đọc dữ liệu từ file VIDU6-15.TXT đã được tạo ra từ ví dụ trong phần lưu dữ liệu. Nội dung chứa trong file, như đã trình bày, gồm 4 dòng:

```
ABC
write-line: dong 2
write-line: dong 3
<xuống dòng>
```

Nội dung đoạn chương trình, ý nghĩa và kết quả trả về của các biểu thức như sau:

```
(setq fname "D:/VLISP/VIDU6-15.TXT")  ; khai báo tên file và gán
                                      ; cho biến fname
(setq fil (open fname "r"))           ; mở file để đọc dữ liệu
(if fil                               ; nếu mở file thành công
    (progn
        (read-char fil)   ; trả về 65, ứng với ký tự "A" trên dòng 1
        (read-char fil)   ; trả về 66, ứng với ký tự "B" trên dòng 1
        (read-char fil)   ; trả về 67, ứng với ký tự "C" trên dòng 1
        (read-char fil)   ; trả về 10, ứng với ký tự hết dòng 1
        (read-char fil)   ; trả về 119, ứng với ký tự "w" trên dòng 2
        (read-line fil)   ; trả về "rite-line: dong 2" (thiếu chữ "w")
        (read-line fil)   ; trả về "write-line: dong 3" (đầy đủ dòng)
        (read-line fil)   ; trả về NIL (chỉ có ký tự dòng mới)
        (read-line fil)   ; trả về NIL (hết file - không còn gì để đọc)
        (close fil)       ; đóng file
        )       ; kết thúc progn
    )           ; kết thúc if
```

- ***Hàm findfile*** tìm xem file hay folder có tồn tại trong các đường dẫn khai báo trong mục Reference của AutoCAD hay không. Hàm này thường được dùng để kiểm tra xem file hay folder có tên chỉ định đã tồn tại hay chưa, rất có lợi khi sử dụng cùng hàm *open* đã nói ở trên. Cú pháp chung của hàm này như sau:

(findfile fname)

trong đó, *fname* là tên file hoặc folder cần tìm. Nếu tham số này không kèm theo đường dẫn, hàm sẽ tìm trong các folder khai báo trong mục

Reference của AutoCAD, ngược lại hàm sẽ chỉ tìm kiếm trong folder với tên đã cho. Nếu tìm thấy file hay folder đã chỉ định thì hàm *findfile* sẽ trả về tên đầy đủ của file hay folder cần tìm, còn nếu không tìm thấy, hàm sẽ trả về NIL. Các ví dụ sau đây minh họa điều đó:

(setq fname (findfile "vidu2.txt")) sẽ tìm kiếm file có tên *vidu2.txt* trong các folder AutoCAD. Không có file như vậy nên hàm sẽ trả về NIL và gán giá trị NIL này cho biến fname.

(setq fname (findfile "d:/vlisp/vidu2.txt")) sẽ tìm file cùng tên *vidu2.txt*, nhưng chỉ trong folder *d:/vlisp* đã chỉ định mà thôi. Do file đã tồn tại, hàm trả về giá trị *"D:\\vlisp\\vidu2.txt"* và gán nó cho biến fname.

Nếu folder hiện thời "/AutoCAD 2004" có chứa file "abc.lsp", biểu thức sau đây sẽ trả về tên đầy đủ của file này:

(findfile "abc.lsp") cho kết quả "C:\\Program Files\\AutoCAD 2004\\abc.lsp"

Bảng 6.5. Ý nghĩa của tham số flag (hàm getfiled)

Bit	*Flags*	*Ý nghĩa*
0	1	Hiện hộp thoại tương tự SaveAs tiêu chuẩn của AutoCAD với mục đích nhập tên file để lưu dữ liệu (dùng với hàm open, kiểu "w") Nếu file đã tồn tại, sẽ xuất hiện cảnh báo, tránh bị ghi đè dữ liệu *Nếu bit này có giá trị 0 sẽ xuất hiện hộp thoại Open với mục đích nhập tên file để đọc dữ liệu (dùng với hàm open, kiểu "r")* *Nếu nhập tên file chưa tồn tại, sẽ xuất hiện thông báo nhắc nhở.*
1	2	Không sử dụng
2	4	Cho phép nhập tên file không có phần đuôi hoặc có phần đuôi khác mặc định *ext* đã khai báo trong hàm. *Nếu bit này có giá trị 0, tên file sẽ được gán đuôi mặc định. Phần đuôi mặc định này sẽ được tự động gắn thêm vào nếu người dùng chưa nhập trong hộp File name của hộp thoại.*

- ***Hàm getfiled*** yêu cầu người dùng nhập tên file qua hộp thoại file tiêu chuẩn của AutoCAD và trả về tên đầy đủ của file. Cú pháp chung của hàm này như sau:

(getfiled title default ext flags)

trong đó,

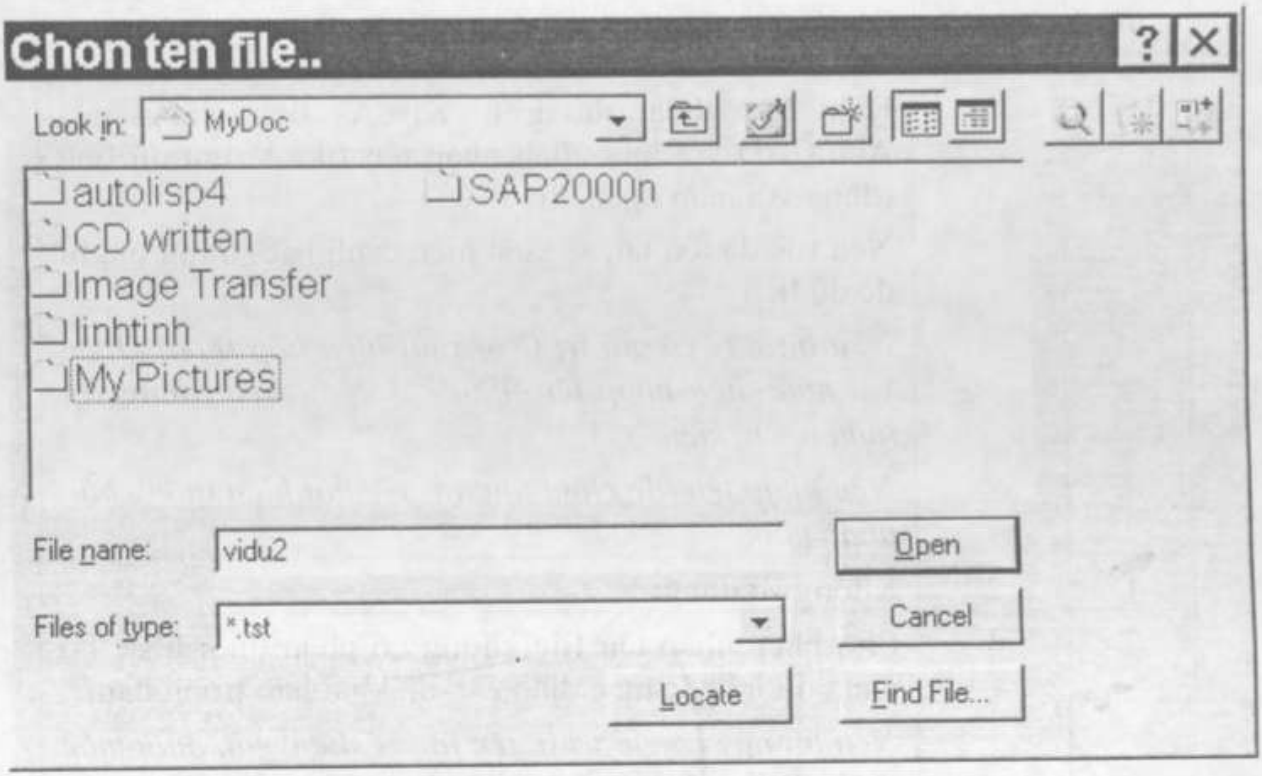
title - chuỗi ký tự, hiển thị trên thanh tiêu đề của hộp thoại.

default - chuỗi ký tự, thể hiện tên file mặc định (có thể sử dụng chuỗi rỗng ("") nếu không muốn thể hiện tên mặc định nào cả).

ext - chuỗi ký tự, thể hiện phần mở rộng (phần đuôi) của tên file. Nếu *ext* được gán chuỗi rỗng, nó được coi như chuỗi "*" (file có phần tên mở rộng bất kỳ).

flags - số nguyên, điều khiển việc thể hiện hộp thoại theo một cách thức nhất định. Giá trị của *flags* có thể tính được thông qua giá trị các bit, tương tự như trong hàm *initget* đã trình bày ở phần nhập dữ liệu. Ý nghĩa một số bít (khi có giá trị 1) cho trong bảng 6.5.

Ví dụ, biểu thức *(getfiled "Chon ten file.." "vidu2" "tst" 0)* sẽ làm bật hộp thoại FileOpen như trên hình 6.5. Nếu nhập tên file hoặc chọn trên hộp thoại, hàm này sẽ trả về tên đầy đủ (kèm đường dẫn), còn không (chẳng hạn, người dùng nhấn nút Cancel), hàm trả về NIL.



Hình 6.5: Sử dụng hàm getfiled để nhập tên file

Sau đây là một số ví dụ khác minh họa việc nhập dữ liệu và thao tác với file. Hàm *pts2file* yêu cầu người dùng nhập các điểm, kết quả sẽ được lưu vào file thành các dòng, mỗi dòng là một danh sách chứa 3 tọa độ các điểm đã nhập, hàm *pts2file2* là một biến thể của hàm này, mỗi dòng sẽ không phải là danh sách mà là các tọa độ điểm tương ứng. Hàm *file2list* thực hiện công việc ngược lại: đọc dữ liệu từ file và chuyển mỗi dòng thành một danh sách con, sau đó gộp tất cả các danh sách con này

thành một danh sách chung. Các dòng trong file dữ liệu, nếu được bắt đầu bằng ký tự chấm phẩy (;) sẽ được coi như chú thích và bỏ qua không đưa vào danh sách.

Ví dụ 6.16a: *Lưu trực tiếp dữ liệu vào file bằng lệnh print.*

```
(defun pts2file (fname / p fil fname)      ; khai báo hàm người dùng
 (setq fname "d:/vlisp/vidu6-16a.txt")     ; tên file
 (setq fil (open fname "w"))               ; mở file để lưu dữ liệu
 (if fil                                   ; nếu mở file thành công
   (progn
     (princ ";;; Tọa độ các điểm ;;;" fil)     ; ghi dòng tiêu đề vào file
     (setq p (getpoint "\nNhập điểm đầu: ")) ; nhập điểm đầu
     (while p                                  ; vòng lặp
        (print p fil)          ; lưu trực tiếp dữ liệu vừa nhập vào file
        (setq p (getpoint p "\nNhập điểm tiếp theo: ")) ; nhập điểm tiếp theo
         )                     ; nhấn Enter (không nhập gì) để kết thúc
      (close fil)              ; đóng file
      )
    (prompt "\nKhông mở được  file...") ; thông báo nếu không mở được file
    ) ; hết if
 ) ; kết thúc defun
```

Trong ví dụ này, nếu kích chuột để nhập một số điểm trên màn hình AutoCAD, các điểm này sẽ được trả về như một danh sách chứa 3 phần tử, ứng với các tọa độ X, Y và Z của điểm đã nhập. File kết quả (vidu6-16a.txt) sẽ chứa dữ liệu có dạng như sau:

```
;;; Tọa độ các điểm ;;;
(1.0 1.0 0.0)
(2.5 1.30 0.0)
(4.0 2.40 0.0)
(5.0 3.80 0.0)
```

Nếu không muốn các dấu ngoặc đơn được lưu vào file, mà chỉ có các số thể hiện tọa độ các điểm, có thể sửa lại một số lệnh trong ví dụ trên như sau:

Ví dụ 6.16b: *Lưu trực tiếp dữ liệu vào file bằng lệnh print.*

```
(defun pts2file2 (fname / p fil)           ; khai báo hàm người dùng
 (setq fname "d:/vlisp/vidu6-16b.txt")     ; tên file
 (setq fil (open fname "w"))               ; mở file để lưu dữ liệu
 (if fil                                   ; nếu mở file thành công
    (progn
       (princ ";;; Tọa độ các điểm ;;;" fil)    ; ghi dòng tiêu đề vào file
```

```
    (setq p (getpoint "\nNhập điểm đầu: "))        ; nhập điểm đầu
    (while p                                       ; vòng lặp
          (print (car p)  fil)                     ; lưu tọa độ X điểm vừa nhập
          (princ (cadr p) fil)                     ; tọa độ Y
          (princ " ")                              ; dấu cách
          (princ (last p) fil)                     ; tọa độ Z
          (setq p (getpoint p "\nNhập điểm tiếp theo: ")) ; nhập tiếp...
          )                    ; nhấn Enter (không nhập gì) để kết thúc
     (close fil)               ; đóng file
     )
   (prompt "\nKhông mở được  file...") ; thông báo (nếu không mở được file)
   ) ; hết if
)    ; kết thúc defun
```

Khi chạy hàm này và nhập các điểm, kết quả lưu trong file vidu4-16b.txt sẽ có dạng như sau:

```
;;; Tọa độ các điểm ;;;
1.0 1.0 0.0
2.5 1.30 0.0
4.0 2.40 0.0
5.0 3.80 0.0
```

Ví dụ 6.17: *Đọc dữ liệu từ file và chuyển thành danh sách.*

```
(defun  file2list ( fname  /  f  line line1 lst char1  res) ; khai báo hàm
    (setq f (open fname "r"))          ; mở file để đọc dữ liệu
    (if (not f) (progn
          (alert (strcat "Không mở được file " fname " để đọc..."))
          (exit))            ; nếu không mở được, thoát chương trình
          )
    (setq res '())                     ; tạo một danh sách trắng
    (while  (setq line (read-line f))  ; đọc các dòng dữ liệu trong file
          (setq char1 (substr line 1 1)) ; lấy ký tự đầu của dòng vừa đọc
          (if (/= ";" char1)   ; kiểm tra xem có phải là dòng chú thích
            (progn             ; nếu không phải, xử lý tiếp
                (if (/= "(" char1)  ; thêm dấu "(" nếu chưa có để tạo
                                    ; chuỗi ký tự dạng danh sách
                    (setq line (strcat "(" line)))
                (if (/= ")"  (substr line (strlen line) 1))
                    ; thêm dấu ")" nếu chưa có
                (setq line (strcat line ")")))
                (setq lst (read line))      ; đọc dòng dữ liệu mới
                                            ; và trả về một danh sách
                (setq res (append res (list lst)))
                                    ; thêm danh sách lst vào kết quả res
```

```
        ))          ; kết thúc progn và if
    )               ; kết thúc vòng lặp while (khi hết file)
    (close f)       ; đóng file dữ liệu
    res             ; trả kết quả về
)
```

Chẳng hạn, nếu muốn xử lý dữ liệu trong file vidu6-16b.txt ở ví dụ trên có thể gọi hàm này như sau:

```
(file2list "d:/vlisp/vidu6-16b.txt")
```

Kết quả trả về danh sách liên kết sau đây:

```
((1.0 1.0 0.0) (2.5 1.30 0.0) (4.0 2.40 0.0) (5.0 3.80 0.0))
```

6.3.2. Tạo các đối tượng AutoCAD

Với AutoLISP có thể tạo các đối tượng AutoCAD (các đối tượng đồ họa như các đoạn thẳng, cung tròn,... và các đối tượng khác như khối - block, lớp - layer,...) thông qua 2 cách: sử dụng hàm command gửi các lệnh và thông số tương ứng để AutoCAD thực hiện, hoặc trực tiếp can thiệp vào CSDL AutoCAD để thêm các đối tượng mới hoặc sửa đổi các đối tượng đã tồn tại.

6.3.2.1. Sử dụng hàm command và lệnh AutoCAD

Cú pháp chung của hàm này như sau:

(command [arguments] ...)

trong đó, *arguments* là các tham số của hàm, gồm tên lệnh AutoCAD, các lựa chọn con và dữ liệu cần nhập như khi thực hiện vẽ trong AutoCAD. Các tham số này có thể là chuỗi ký tự (String), số thực (Real), số nguyên (Integer), dữ liệu dạng điểm (Point) hoặc nhóm đối tượng (SelectionSet). Chuỗi ký tự trắng ("") được hiểu như việc nhập Enter hoặc Space, còn hàm *command* không tham số sẽ tương đương như việc nhấn phím Esc để thoát các lệnh trong AutoCAD.

Khi gặp biểu thức này, AutoLISP sẽ gửi tới AutoCAD và các thông số sẽ được xử lý lần lượt theo thứ tự của lệnh tương ứng. Nếu thông số đưa vào không thích hợp, hàm sẽ xử lý sai.

Hàm *command* luôn trả về giá trị NIL.

Để tránh việc lệnh gốc của AutoCAD đã bị định nghĩa chồng, hãy thêm dấu chấm (.) vào trước tên lệnh để gọi lệnh gốc, còn việc thêm dấu gạch dưới (_) vào trước tên lệnh nhằm gọi đúng lệnh ở phiên bản tiếng Anh nếu AutoCAD được cài đặt ở ngôn ngữ khác. Ví dụ, biểu thức: *(command "Line" pt1 pt2 pt3 "c")* sẽ không thực hiện vẽ 3 cạnh tam giác qua 3 đỉnh pt1 pt2 pt3 nếu AutoCAD được cài đặt bản tiếng Pháp (khi đó

lệnh vẽ đoạn thẳng sẽ là Ligne chứ không phải Line) hoặc trước đó lệnh Line đã được định nghĩa chồng qua việc khai báo hàm C:Line. Để biểu thức này luôn thực hiện đúng nên đặt thêm các tiền tố đã nói trên như sau: *(command "_.Line" pt1 pt2 pt3 "_c")* hoặc *(command "._Line" pt1 pt2 pt3 "_c")*.

Nếu không muốn các lời nhắc của lệnh hiện ra trên màn hình AutoCAD hãy đặt lại biến hệ thống *Cmdecho* về giá trị 0. Một số biến hệ thống khác như *Osmode* (OSnap), *Orthomode*... gây ảnh hưởng tới việc thực thi chương trình cũng nên đặt lại giá trị thích hợp. Việc lấy giá trị và cài đặt lại biến hệ thống được thực hiện bằng các hàm *getvar* và *setvar* sẽ được đề cập ở phần sau.

Lưu ý rằng một số lệnh AutoCAD không thể sử dụng trong AutoLISP. Ví dụ, lệnh *Script* chạy file đáp bản chỉ được phép gọi sau cùng trong đoạn chương trình AutoLISP, còn lệnh *Sketch* đọc dữ liệu trực tiếp từ thiết bị Digitizer nên cũng không được dùng. Ngoài ra nếu lệnh AutoCAD liên quan đến việc chọn đối tượng (như *Break* hoặc *Trim*) thì việc chọn đối tượng được truyền qua danh sách *entsel* chứ không phải qua điểm nằm trên đối tượng như khi thao tác trong AutoCAD. Ví dụ, đoạn chương trình sau minh họa điều đó:

```
(command "_.Circle" "5,5" "2")          ; vẽ vòng tròn
(command "_.Line" "3.5" "7,5" "")       ; vẽ đoạn thẳng
(setq el  (entlast))                    ; chọn đối tượng vừa vẽ sau cùng (đoạn thẳng)
(setq pt  '(5 7))                       ; định nghĩa điểm trên đối tượng nơi sẽ cắt đi
(command "_.Trim" el "" pt "")          ; thực hiện việc Trim đoạn thẳng
                                        ; (có biên là vòng tròn)
```

Nhìn chung, đây là một cách rất hữu hiệu để tạo các đối tượng AutoCAD, nhưng yêu cầu người lập trình phải thành thạo với AutoCAD, nắm chắc trình tự các bước một lệnh AutoCAD thực hiện như thế nào ở phương án dòng lệnh của nó, bao gồm các lựa chọn con (nếu có) trong lời nhắc của lệnh, các dữ liệu cần nhập,... Ví dụ, để vẽ một vòng tròn, cần nắm được các yếu tố sau:

- Tên lệnh AutoCAD : *Circle*
- Các lời nhắc tiếp theo: *Specify center point for circle or [3P/2P/Ttr (tan tan radius)]*. Cần hiểu ý nghĩa các lời nhắc này để nhập phương án đúng.
- Các dữ liệu cần nhập tương ứng với phương án đã chọn. Trong ví dụ trên, nếu chọn phương án mặc định, dữ liệu cần nhập là một điểm (tâm vòng tròn) và một khoảng cách (bán kính), còn nếu muốn vẽ vòng tròn qua 3 điểm, cần nhập phương án 3P và nhập vào 3 điểm...

Đoạn chương trình sau đây minh họa các lưu ý này:

```
(setq pt1 '(0 0 0) pt2 '(1 0 0) pt3 '(0 1 0) ; khai báo tọa độ cho 3 điểm
      bk 10)        ; gán cho biến bk giá trị 10
(command "_.Circle" pt1 bk) ; vẽ vòng tròn tâm pt1 và bán kính bk
(command "_.Circle" "3P" pt1 pt2 pt3)   ; vẽ vòng tròn qua 3 điểm
```

Một cách rất thông dụng và đơn giản để tạo các đối tượng đồ họa phức tạp (ví dụ như vẽ bánh răng, trục, ổ hay đơn giản hơn như bulông, đai ốc,...) là tính trước các điểm đặc trưng hoặc khoảng cách và góc cần thiết, sau đó dùng hàm *command* với các lệnh AutoCAD như Line, PLine, Circle,... để xây dựng đối tượng. Thông thường, các điểm trên một đối tượng phức tạp liên kết với nhau qua kích thước và hướng của các phần tử, hay nói khác đi, chúng có thể tính được nếu biết trước các điểm khác.

Việc tính tọa độ các điểm có thể thực hiện được qua các hàm số học như cộng, trừ, nhân, chia,... và các hàm xử lý danh sách. Ví dụ, để vẽ một hình thoi có đỉnh dưới cùng do người dùng nhập trực tiếp, chiều dài trục đứng là a và tỷ số chiều dài trục ngang và trục đứng là k có thể thực hiện thông qua các biểu thức AutoLISP sau:

Ví dụ 6.18: *Vẽ hình thoi đứng biết tọa độ một đỉnh, độ dài trục cho trước và tỷ số độ dài các trục là k.*

```
;;; Nhập dữ liệu người dùng trực tiếp từ AutoCAD
(setq p0 (getpoint "\nNhập đỉnh hình thoi: ")
      a  (getdist p0 "\nNhập chiều dài trục đứng: " ))
(initget 6)                  ; không cho phép nhập số 0 và số âm
(setq k (getreal  "\nNhập tỷ lệ trục ngang/ trục đứng k <1> : "))
(if (null k) (setq k 1.0))  ; lấy giá trị mặc định k=1 nếu nhấn ngay Enter
(setq b2 (*  a  k  0.5))    ; tính chiều dài nửa trục ngang
;;; Tính các điểm cần thiết...
;;; các điểm p0 p1 p2 p3 là các đỉnh của hình thoi
(setq    x0 (car p0)        ; các tọa độ của điểm p0 do người dùng nhập
         y0 (cadr p0)
         z0 (last p0))
(setq    y2 (+  a  y0)              ; đỉnh đối diện
         p2 (list  x0 y2 z0))
(setq    y1 (+  (* a 0.5)  y0)      ; đỉnh trái
         x1 (-  x0  b2)
         p1 (list  x1 y1 z0))
(setq    x3 (+  x0  b2)             ; đỉnh phải
         p3 (list  x3 y1 z0))
;;; Vẽ hình thoi như một đường PLine kín...
(command "._PLine" p0 p1 p2 p3 "_c")
(princ)                             ; rút lui im lặng
```

Rõ ràng, việc tính toán tọa độ theo kiểu này khá vất vả. Nếu phải vẽ hình thoi qua 2 đỉnh đối diện bất kỳ do người dùng nhập và tỷ số chiều dài các trục k thì việc tính toán còn phức tạp hơn nhiều do phải sử dụng thêm các hàm lượng giác để tính góc, độ lớn các trục và tọa độ các đỉnh của hình thoi.

Ngôn ngữ AutoLISP cung cấp ***các hàm hình học*** để phục vụ việc này giúp cho việc tính toán góc, khoảng cách và tọa độ điểm đơn giản đi rất nhiều. Sau đây là một số hàm như vậy:

- Hàm ***polar*** trả về một điểm, xác định thông qua tọa độ tương đối với một điểm khác. Cú pháp chung của hàm như sau:

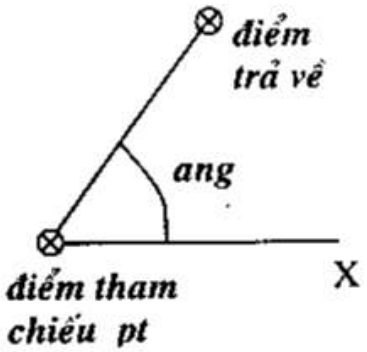
(polar pt ang dist)

trong đó (xem hình minh họa):

pt - một điểm (điểm tham chiếu, đóng vai trò gốc tọa độ tạm thời).

ang - góc tạo bởi đường nối các điểm pt và điểm mới với trục X của hệ tọa độ chuẩn World, đo bằng radian. Lưu ý, chiều dương của góc ngược chiều kim đồng hồ.

dist - khoảng cách giữa 2 điểm.



- Hàm (***distance pt1 pt2***) trả về khoảng cách giữa 2 điểm *pt1* và *pt2*. Các điểm này có thể là 2D (chứa 2 tọa độ trong mặt phẳng XY) hoặc 3D (3 tọa độ trong không gian XYZ). Việc tính khoảng cách giữa hai điểm cũng có thể thực hiện qua việc lấy căn bậc 2 của tổng bình phương của hiệu các tọa độ, nhưng rõ ràng là rắc rối hơn nhiều.

- Hàm ***(angle pt1 pt2)*** trả về giá trị của góc tạo bởi trục X của hệ tọa độ hiện hành với đường nối các điểm *pt1* và *pt2*. Giá trị góc trả về là radian, trong khoảng 0 đến 2π. Các điểm có thể là 2D hoặc 3D. Trường hợp 3D, các điểm này được chiếu xuống mặt phẳng XY hiện hành.

- Hàm ***(inters pt1 pt2 pt3 pt4 [onseg])*** trả về giao điểm của hai đoạn thẳng nối các điểm *pt1-pt2* và *pt3-pt4*. Nếu bỏ qua tham số *onseg* hoặc tham số này có giá trị khác *NIL*, hàm sẽ chỉ tìm giao điểm thực sự của hai đoạn thẳng và trả về *NIL* nếu chúng không cắt nhau. Còn nếu *onseg* là *NIL* hàm sẽ tìm giao điểm cả trên đường kéo dài của hai đoạn thẳng này. Các điểm *pt1*, *pt2*, *pt3* và *pt4* trong tham số của hàm có thể là 2D hoặc 3D. Nếu tất cả chúng là các điểm 3D, hàm sẽ tìm giao điểm của chúng trong không gian, còn nếu có ít nhất 1 điểm 2D, các điểm khác sẽ

được chiếu xuống mặt phẳng XY của hệ tọa độ hiện hành (UCS) và tìm giao điểm trong mặt phẳng này. Ví dụ:

```
(setq a '(1.0 1.0) b '(9.0 9.0))    ; khai báo các điểm 2D
(setq c '(4.0 1.0) d '(4.0 2.0))
(setq intpt (inters a b c d))       ; tìm giao điểm của các đoạn ab và cd.
                                    ; kết quả gán cho biến intpt. Trường hợp này hàm inter
                                    ; trả về NIL và intpt nhận giá trị NIL.
(setq intpt2 (inters a b c d NIL))  ; hàm inter sẽ tìm giao điểm cả ở
                                    ; phần kéo dài của các đoạn trên và tìm được giao điểm
                                    ; có tọa độ (4.0 4.0)
```

- Hàm ***(osnap pt mode)*** trả về điểm 3D từ kết quả của việc truy bắt điểm đặc trưng trên đối tượng bằng Object Snap (OSnap) tương tự như khi sử dụng tính năng này để vẽ trong AutoCAD. Trong hàm này tham số *pt* là điểm nơi sẽ tìm điểm OSNAP, còn *mode* là chuỗi ký tự thể hiện đặc trưng của OSNAP như *mid* (điểm giữa của đối tượng), *cen* (tâm đối tượng), *end* (điểm cuối),... Khi cần đưa nhiều đặc trưng, các đặc trưng này cần cách nhau bởi dấu phẩy (,). Kết quả hàm trả về là một điểm nếu truy bắt được, còn nếu không, hàm trả về *NIL*. Ví dụ, biểu thức *(setq ospt (osnap '(1 1 0) "_int,_end"))* sẽ truy tìm giao điểm hoặc điểm cuối của đối tượng vẽ trong vùng quanh điểm có tọa độ (1 1 0).

Cần lưu ý rằng, việc truy bắt được hay không phụ thuộc không những vào điểm truy bắt *pt* mà còn vào độ lớn của vùng truy bắt, được xác lập qua biến hệ thống *Aperture* của AutoCAD. Nhìn chung việc truy bắt điểm kiểu này rất ít được sử dụng trong lập trình thiết kế tự động do có thể trả về những điểm không mong muốn, nhất là khi sử dụng nhiều đặc trưng trong tham số *mode*, hoặc khi được, khi không do phụ thuộc biến hệ thống *Aperture*. Điều này làm chương trình chạy không ổn định.

Để minh họa cho việc sử dụng các hàm này, dưới đây trình bày đoạn chương trình vẽ hình thoi qua 2 đỉnh đối diện do người dùng nhập và tỷ số giữa độ lớn các trục như đã trình bày ở trên.

Ví dụ 6.19: *Vẽ hình thoi biết 2 đỉnh đối diện và tỷ số độ dài các trục là k.*

```
;;; Nhập dữ liệu người dùng trực tiếp từ AutoCAD
(setq p0 (getpoint "\nNhập đỉnh hình thoi: ")
      p2  (getdist p0 "\nNhập đỉnh đối diện: " ))
(initget 6)                    ; không cho phép nhập số 0 và số âm
(setq k (getreal  "\nNhập tỷ lệ trục ngang/ trục đứng k <1> : "))
(if (null k) (setq k 1.0))  ; lấy giá trị mặc định k=1 nếu nhấn ngay Enter
;;; Tính các điểm cần thiết...
;;; các điểm p0 p1 p2 p3 là các đỉnh của hình thoi
```

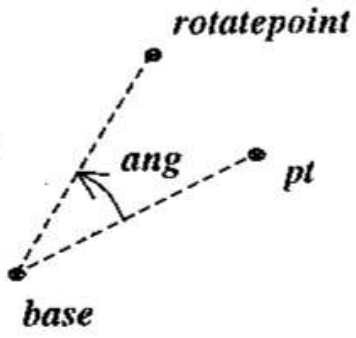
```
(setq g90 (/ pi 2.0))          ; đổi góc 90 độ ra radian
(setq goc (angle p0 p2))       ; góc tạo bởi hai điểm do người dùng nhập
(setq a (distance p0 p2)       ; tính chiều dài trục thứ nhất
      b2 (* a k 0.5))          ; tính chiều dài trục thứ hai
(setq tam (polar p0 (+ goc g90) (* 0.5 a))   ; tính tọa độ tâm hình thoi
      p1 (polar tam (+ goc g90) b2)
      p3 (polar tam (- goc g90) b2))
;;; Vẽ hình thoi như PLine kín....
(command "._PLine" p0 p1 p2 p3 "_c")
(princ)                                ; rút lui im lặng
```

Ví dụ 6.20 dưới đây giới thiệu một số hàm người dùng, có thể sử dụng để xử lý điểm rất hữu ích.

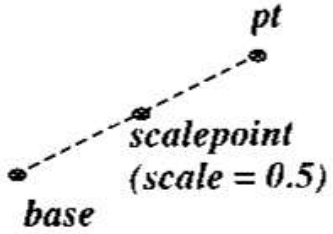
- Hàm (*RotatePoint pt base ang*) trả về một điểm sau khi quay điểm *pt* quanh điểm *base* một góc *ang* (tính bằng radian).
- Hàm (*ScalePoint pt base scale*) trả về một điểm nằm trên đường nối các điểm *base* và *pt*, cách điểm gốc *base* một khoảng bằng tích khoảng cách hai điểm *base* và *pt* với hệ số tỷ lệ *scale*. Ví dụ, với *scale* bằng 1/2 hàm này sẽ trả về điểm giữa 2 điểm đã cho.
- Hàm (*MirrorPoint pt pt1 pt2*) trả về điểm đối xứng của điểm *pt* qua đường thẳng nối các điểm *pt1* và *pt2*. Lưu ý rằng, điểm sử dụng với các hàm này là các điểm 2D (nằm trong mặt phẳng XY của hệ tọa độ UCS). Có thể sửa đổi thêm để sử dụng với điểm 3D.

Ví dụ 6.20: *Một số hàm xử lý điểm*

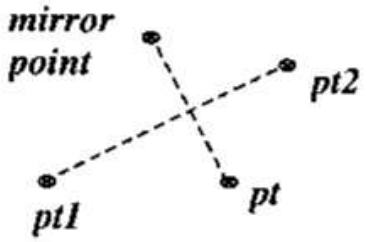
```
;;; Rotate a point around base with
;;; an angle in radians
   (defun RotatePoint(pt base ang / dist ang1 res)
    (setq dist (distance base pt)
          ang1 (+ ang (angle base pt)))
    (setq res (polar base ang1 dist))
    )      ; end of RotatePoint
```




```
;;; Scale a point from base
   (defun ScalePoint(pt base scale / dist ang res)
    (setq dist (distance base pt)
          ang (angle base pt))
    (setq res (polar base ang (* scale dist)))
    )      ; end of ScalePoint
```



```
;;; Mirror a point over a line
(defun MirrorPoint(pt pt1 pt2 / ang res)
 (setq ang (- (angle pt1 pt) (angle pt1 pt2))
       ang (* ang -2.0))
 (setq res (RotatePoint pt pt1 ang))
 )      ; end of MirrorPoint
```



Một lưu ý khác liên quan đến sử dụng hàm *command* là các biến hệ thống như *OSmode*, *Orthomode*,... có thể ảnh hưởng đến việc các điểm đã cho bị rời sang vị trí khác. Vì vậy, trước khi chạy chương trình thường cần đặt lại các biến này về giá trị 0, sau đó chúng được cài đặt lại các giá trị ban đầu, khôi phục trạng thái trước đó cho người dùng. Để thực hiện điều này có thể áp dụng các hàm sau:

- Hàm ***(getvar varname)*** trả về giá trị hiện thời của biến hệ thống *varname*. Tham số *varname* là chuỗi ký tự hoặc ký hiệu thể hiện biến hệ thống mà AutoCAD quản lý. Nếu tham số *varname* không đúng, hàm trả về kết quả NIL. Có thể tham khảo *Command Reference / System Variables* trong phần trợ giúp của AutoCAD để biết chi tiết các biến hệ thống này. Hàm này được sử dụng để tìm giá trị hiện thời của biến hệ thống nào đó, nhằm mục đích lưu giữ lại và trả lại chúng sau khi thực hiện chương trình.

- Hàm ***(setvar varname value)*** đặt lại giá trị cho biến hệ thống *varname* giá trị mới *value*. Tham số *value* có thể là giá trị (số nguyên, số thực, chuỗi ký tự,...) hay biểu thức trả về giá trị thích hợp. Nếu thành công, hàm trả về giá trị của tham số *value*, còn không, ví dụ, viết sai tên biến hệ thống, hàm cho kết quả NIL. Lưu ý rằng, không phải biến hệ thống nào cũng có thể thay đổi giá trị của nó. Một số biến hệ thống chỉ cho phép đọc, không cho phép sửa, do vậy không thể đặt lại giá trị khác. Trong trợ giúp *Command Reference / System Variables* các biến này được ghi chú là chỉ được đọc (*read only*), kiểu dữ liệu (*type*) và giá trị mặc định (*initial value*) của các biến hệ thống cũng được chỉ rõ.

Các hàm con trong ví dụ dưới đây minh họa việc đọc, lưu và khôi phục lại một số biến hệ thống hay sử dụng trong khi vẽ các đối tượng đồ họa AutoCAD.

Ví dụ 6.21: *Thao tác với biến hệ thống*

```
;;; VarGetting ()
;;; Hàm này đọc giá trị các biến hệ thống hiện hành như chế
;;; độ OSnap, Ortho, Echo,...
;;; Kết quả trả về của hàm là danh sách (list) chứa các giá trị này.
```

```
;;; VarSetting ()
;;; Hàm này đặt các biến hệ thống như chế độ OSnap, Ortho, Echo,...
;;; về 0 để tránh sai sót khi nhập tọa độ điểm, vẽ các đối
;;;  tượng AutoCAD...
;;; Kết quả trả về của hàm là NIL.

;;; VarResetting (varlist)
;;; Hàm này đặt lại các biến hệ thống như chế độ OSnap, Ortho,
;;; Echo,... về giá trị ban đầu đã được lưu trong danh sách varlist
;;; Kết quả trả về của hàm là NIL.

;;; Cách sử dụng: Các hàm này được thực hiện theo thứ tự sau:
;;; 1. Gọi hàm VarGetting() để lấy giá trị các biến hệ thống và lưu vào
;;;    một biến nào đó
;;;    Ví dụ sử dụng biểu thức: (setq varlist (VarGetting))
;;; 2. Gọi hàm VarSetting() để đặt các biến hệ thống về giá trị 0
;;; 3. Thực hiện các biểu thức, hàm... vẽ đối tượng
;;; 4. Gọi hàm VarResetting(varlist) để khôi phục lại các biến
;;;    hệ thống như cũ

;;; Bắt đầu nội dung các hàm
(defun VarGetting( / varlist os ortho echo clay cunit cunitmode
                     cblip cangbase)
                                        ; lấy giá trị các biến hệ thống
    (setq os (getvar "osmode")          ; chế độ OSnap
          ortho (getvar "orthomode")    ; chế độ Ortho
          echo (getvar "cmdecho")       ; Command Echo
          clay (getvar "clayer")        ; lớp hiện hành
          cunit (getvar "AUnits") ; số đo góc
          cunitmode (getvar "Unitmode")  ; kiểu thể hiện số
          cblip (getvar "Blipmode")   ; Blip (vết chấm trên màn hình)
          cangbase (getvar "Angbase"))    ; hướng mặc định của trục X
    ;gom tất cả vào một danh sách và là kết quả trả về của hàm
    (setq varlist
          (list os ortho echo clay cunit cunitmode cblip cangbase))
    ) ; kết thúc hàm


(defun VarSetting()
    (setvar "osmode" 0)
    (setvar "orthomode" 0)
    (setvar "cmdecho" 0) ; không hiện các biểu thức và kết quả trung
                         ; gian trong khi vẽ, giúp chương trình chạy nhanh hơn.
```

```
    (setvar "Aunits" 0)
    (setvar "Unitmode" 0)
    (setvar "Blipmode" 0)
    (setvar "Angbase" 0)
    (princ)
    )      ; kết thúc hàm

(defun VarResetting(varlist)
    (setvar "osmode" (nth 0 varlist))
    (setvar "orthomode" (nth 1 varlist))
    (setvar "cmdecho" (nth 2 varlist))
    (setvar "clayer" (nth 3 varlist))
    (setvar "Aunits" (nth 4 varlist))
    (setvar "Unitmode" (nth 5 varlist))
    (setvar "Blipmode" (nth 6 varlist))
    (setvar "Angbase" (nth 7 varlist))
    (princ)
    )      ; kết thúc hàm
```

Một lưu ý khác là hàm *command* còn gửi giá trị đến dòng lệnh AutoCAD để xử lý. Ví dụ, để gửi tọa độ một điểm *pt* sang AutoCAD có thể sử dụng biểu thức *(command pt)*. Điều này có thể được sử dụng để vẽ các đối tượng qua các điểm đã nhập từ trước hoặc từ việc tính toán bằng các phần mềm khác và lưu sẵn trong file dữ liệu.

Đoạn chương trình trong ví dụ 6.22 minh họa việc vẽ các đường Spline qua các điểm đọc từ file dữ liệu, tạo nên các đường đồng mức như thể hiện trong các bản đồ. Các điểm cùng mức được lưu trên một dòng dữ liệu trong file vidu6-22.txt theo cấu trúc sau:

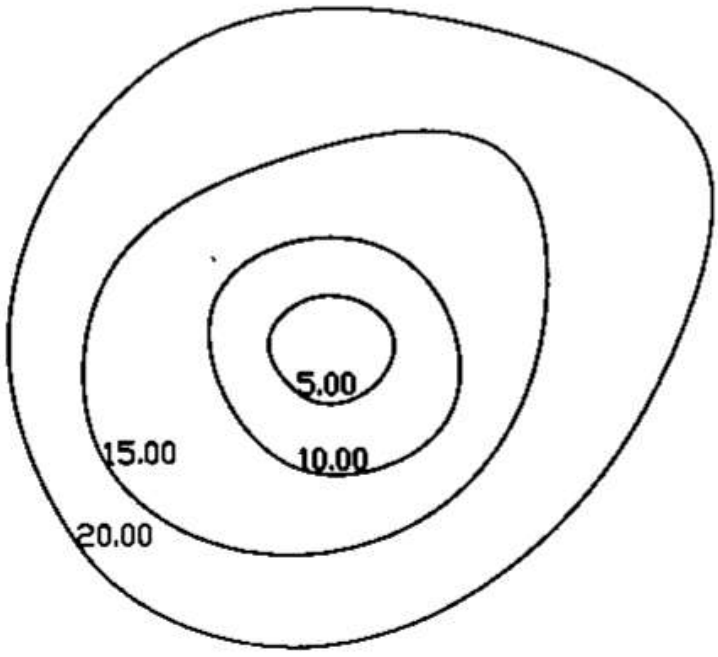
```
5 (20.5 7.2) (23.7 7.2) (25.2 9.9) (21.7 12.0) (19.02 9.65)
                                        ; mức Z = 5, có 5 điểm
10 (20.5 3.5) (27.9 6.1) (25.2 14.12) (16.4 11.8)
                                        ; mức Z = 10, có 4 điểm
15 (10.9 3.9) (10.9 12.6) (19.7 18.7) (31.0 18.7) (32.6 10.5) (26.9 1.3)
   (16.4 -0.15)                         ; mức Z = 15, có 7 điểm
20 (9.5 -0.1) (19.8 -5.2) (32.4 0.0) (39.1 9.6) (40.4 19.0) (30.4 25.0)
   (16.6 25.0) (8.0 16.6) (6.3 8.4)     ; mức Z = 20, có 9 điểm
```

Ví dụ 6.22: *Vẽ các đường đồng mức*

```
(defun  mpline (fname / llst lst z pt n)   ; định nghĩa hàm người dùng
;;; đọc dữ liệu từ file. Sử dụng hàm file2list đã nói ở ví dụ 6.17
```

```
(setq llst (file2list fname))        ; lấy dữ liệu và đưa vào danh sách
(setq n 0)
(while (setq lst (nth n llst))
    (setq z (car lst)                ; lấy mức Z
        lst (cdr lst))               ; còn lại là các điểm trên mức đã cho
    (command "._Spline")
    (foreach  pt lst (command pt))       ; gửi lần lượt các điểm
                                         ;cho lệnh Spline
    (command "_c" "" )                   ; hoàn tất lệnh Spline
    (command "._Text" (car lst) "" "" (rtos z 2 2) "")   ; ghi cao độ Z
    (setq n (1+ n))                  ; tiếp tục sang dòng dữ liệu khác
    )                                ; kết thúc vòng lặp
(princ)          ; rút lui im lặng
)        ; kết thúc hàm mpline
```

Kết quả khi gọi hàm này với file dữ liệu chứa các dòng với nội dung trên là các đường cong Spline như trên hình 6.6.



Hình 6.6. Kết quả ví dụ 6.22

Cách truyền dữ liệu qua hàm *command* như minh họa trong ví dụ 6.22 tỏ ra rất hữu ích khi thao tác với các lệnh vẽ có cấu trúc dữ liệu không cố định như các lệnh Line, PLine, Spline,... Với các lệnh này số điểm cần nhập không hạn chế và lệnh chỉ kết thúc khi người dùng nhấn Enter mà không nhập gì cả.

6.3.2.2. Mã đối tượng và tạo đối tượng qua CSDL AutoCAD

Một cách khác để tạo đối tượng là thêm trực tiếp các bản ghi thể hiện mã của đối tượng cần thiết vào cơ sở dữ liệu AutoCAD. Với phương pháp này chương trình sẽ xử lý nhanh hơn, nhưng đòi hỏi người lập trình phải nắm được cách tổ chức CSDL chứa các đối tượng và cách thức thao tác với chúng.

Đối tượng vẽ trong CSDL thường được truy cập qua tên của chúng. Tên đối tượng được AutoCAD đặt riêng, tự thay đổi mỗi khi mở bản vẽ và đối với người dùng chúng không mang ý nghĩa gì cả, vì vậy thường tên này được gán qua biến và sau đó các thao tác được thực hiện qua biến này. AutoLISP cung cấp các hàm (*entlast*) để lấy tên đối tượng vừa vẽ sau cùng và hàm *(entnext [ename])* trả về tên đối tượng liền sau đối tượng *ename* trong CSDL. Muốn lấy đối tượng đầu trong CSDL chỉ cần gọi hàm (*entnext*) không kèm theo tham số *ename*. Mã đối tượng có tên *ename* được truy cập qua hàm (*entget ename*). Chẳng hạn, nếu vẽ đoạn thẳng nối hai điểm có tọa độ (1,2) và (6,5) trong mặt phẳng XY, sau đó gọi lần lượt các biểu thức AutoLISP sau:

```
(setq ename (entlast))
        ; lấy tên đối tượng vừa vẽ gán cho biến ename
(setq ecode (entget ename))
        ; lấy mã đối tượng này gán cho biến ecode
```

thì kết quả biến *ename* sẽ nhận giá trị *<Entity name: 7ef4fea0>* , còn biến *ecode* sẽ được gán giá trị là một danh sách liên kết:

((-1 . <Entity name: 7ef4fea0>) (0 . "LINE") (330 . <Entity name: 1bbd0c8>) (5 . "6A") (100 . "AcDbEntity") (67 . 0) (410 . "Model") (8 . "0") (100 . "AcDbLine") (10 1.0 2.0 0.0) (11 6.0 5.0 0.0) (210 0.0 0.0 1.0))

Danh sách này thể hiện mã đối tượng được quản lý trong CSDL (mã DXF). Nó chứa các danh sách con dạng Dot-pair hoặc nhiều phần tử, trong đó phần tử đầu gọi là mã nhóm (Group Code) của đối tượng, còn thành phần sau của Dot-pair hoặc các phần tử còn lại là các giá trị ứng với mã nhóm đã cho của đối tượng vẽ. Để thao tác trực tiếp với dữ liệu dạng này, cần nắm được ý nghĩa của các phần tử trong các danh sách con của mã đối tượng. Có thể tham khảo phần *DXF Reference* trong chức năng trợ giúp trực tuyến (Help) của AutoCAD để biết thêm chi tiết về các mã đối tượng đơn (mục */Entities Section*) hoặc các đối tượng phức hợp như khối (mục */Blocks Section*) hoặc các loại đối tượng phi hình học như lớp, hướng nhìn, khuông nhìn (mục */Tables Section*),...

Phần đầu trong danh sách này thể hiện những tính chất chung của đối tượng như kiểu đối tượng (Line, Circle,...), lớp chứa đối tượng, màu sắc,... Mã nhóm chung của các đối tượng đồ họa trong CSDL của AutoCAD được mô tả trong bảng 6.6.

Bảng 6.6. Mã nhóm chung

Mã nhóm	*Ý nghĩa*
-1	Tên đối tượng (entity) do AutoCAD tự đặt, cập nhật mỗi khi mở file
0	Kiểu đối tượng đồ họa (entity type)
5	Handle
102	Bắt đầu định nghĩa nhóm đối tượng "{application_name"
102	Kết thúc định nghĩa nhóm "}"
100	Subclass marker (AcDbEntity)
67	Nếu bỏ qua hoặc có giá trị 0 đối tượng được thể hiện trên không gian mô hình (Model space). Nếu bằng 1 – thể hiện trong không gian giấy (Paper space). Giá trị mặc định là 0.
410	Tên layout tab thể hiện đối tượng
8	Tên lớp thể hiện đối tượng
6	Tên loại nét vẽ (linetype) nếu khác BYLAYER. Ngược lại, mã này sẽ bị bỏ qua. Mặc định: BYLAYER
62	Màu đối tượng (nếu khác BYLAYER). Giá trị 0 tương ứng với BYBLOCK, còn giá trị 256 ứng với BYLAYER. Giá trị âm thể hiện lớp chứa đối tượng này đã bị tắt (OFF). Giá trị mặc định là BYLAYER.
48	Tỷ lệ co giãn nét vẽ (linetype scale). Giá trị mặc định: 1.0
60	Khả năng thấy/ không thấy đối tượng (tuỳ chọn, không bắt buộc). Giá trị 0 ứng với đối tượng thấy được, giá trị 1 - đối tượng ẩn
92	Số byte của hình ảnh (tuỳ chọn). Không có giá trị mặc định
310	Các dòng ký tự, chứa dữ liệu hình ảnh xem trước (preview) – tuỳ chọn. Không có giá trị mặc định.

Phần tiếp theo trong danh sách liên kết thể hiện các đặc trưng riêng của từng loại đối tượng (với đoạn thẳng: các mã này thể hiện tọa độ các điểm cuối, với đường tròn - thể hiện tâm và bán kính,...). Các bảng 6.7.1 đến 6.7.8 trình bày mã riêng cho một số đối tượng đồ họa thông dụng, được dùng trong AutoLISP.

Để dễ dàng hơn cho việc theo dõi mã đối tượng, nên ghi mỗi danh sách con lên một dòng. Hàm người dùng *c:lstDXF* trong đoạn chương trình ở ví dụ 6.23 nhằm thực hiện mục đích này.

Bảng 6.7.1. Mã nhóm đối tượng cung tròn (Arc)

Mã nhóm	*Ý nghĩa*
100	Subclass marker (AcDbCircle)
39	Chiều dày trượt (thickness) – tuỳ chọn. Mặc định bằng 0
10	Tâm (trong hệ tọa độ OCS)
40	Bán kính
100	Subclass marker (AcDbArc)
50	Góc bắt đầu (start angle), radians với AutoLISP hoặc ARX
51	Góc kết thúc (end angle)
210	Hướng trượt (vectơ pháp của mặt phẳng chứa đối tượng) – tuỳ chọn. Mặc định bằng 0, 0, 1

Bảng 6.7.2. Mã nhóm đối tượng đường tròn (Circle)

Mã nhóm	*Ý nghĩa*
100	Subclass marker (AcDbCircle)
39	Chiều dày trượt (thickness) – tuỳ chọn. Mặc định bằng 0
10	Tâm (trong hệ tọa độ OCS)
40	Bán kính
100	Subclass marker (AcDbArc)
210	Hướng trượt (vectơ pháp của mặt phẳng chứa đối tượng) – tuỳ chọn. Mặc định bằng 0, 0, 1

Bảng 6.7.3. Mã nhóm đối tượng đoạn thẳng (Line)

Mã nhóm	*Ý nghĩa*
100	Subclass marker (AcDbLine)
39	Chiều dày trượt (thickness) – tuỳ chọn. Mặc định bằng 0
10	Điểm đầu (trong hệ tọa độ WCS) Với AutoLISP hoặc ARX: 3 tọa độ X, Y, Z
11	Điểm cuối
210	Hướng trượt (vectơ pháp của mặt phẳng chứa đối tượng) – tuỳ chọn. Mặc định bằng 0, 0, 1

Bảng 6.7.4. Mã nhóm đối tượng đa tuyến (Polyline)

Mã nhóm	*Ý nghĩa*
100	Subclass marker (AcDb2DPolyline) hoặc (AcDb3dPolyline)
10	Điểm. Tọa độ X, Y luôn bằng 0, Z - cao độ (elevation). Hệ tọa độ OCS với 2Dpolyline, hoặc WCS với 3Dpolyline
39	Chiều dày trượt (thickness), mặc định 0
70	Kiểu bitcode: giá trị 0 = đa tuyến hở, 1 = kín, 2 = curve-fit, 4 = spline-fit, 8 = 3Dpolyline, 16 = 3Dpolygon mesh (lưới), 32 = 3D polygon mesh, kín theo phương N, 64 = polyface mesh (mặt lưới). Giá trị mặc định bằng 0
40	Chiều rộng đường tại điểm đầu đoạn. Không sử dụng khi đặt mã 43
41	Chiều rộng đường tại điểm cuối đoạn. Không sử dụng khi đặt mã 43
71	Số điểm chia lưới theo phương M
72	Số điểm chia lưới theo phương N
73	Mật độ trơn theo phương M
74	Mật độ trơn theo phương N
75	Kiểu trơn (bitcode). Giá trị 0 = không làm trơn; 5 = thay bằng B-spline bậc 4; 6 = B-spline bậc 3; 8 = mặt Bezier
210	Hướng trượt (tuỳ chọn). Mặc định bằng 0, 0, 1

Bảng 6.7.5. Mã nhóm đối tượng đa tuyến phẳng (LWPolyline)

Mã nhóm	*Ý nghĩa*
100	Subclass marker (AcDbPolyline)
90	Số đỉnh đa tuyến
70	Kiểu: giá trị 0 - đa tuyến hở, 1 - kín. Giá trị mặc định bằng 0
43	Độ rộng đường (khi là hằng số). Mặc định bằng 0. Không dùng khi chiều rộng thay đổi (mã nhóm 40, 41)
38	Cao độ Z (elevation), mặc định bằng 0
39	Chiều dày trượt (thickness), mặc định 0
10	Tọa độ đỉnh đa tuyến (trong hệ tọa độ OCS)
40	Chiều rộng đường tại điểm đầu đoạn. Không sử dụng khi đặt mã 43
41	Chiều rộng đường tại điểm cuối đoạn. Không sử dụng khi đặt mã 43
42	Bulge (độ cong của đoạn, giá trị thay đổi theo từng đoạn trong đa tuyến). Mặc định bằng 0 (đoạn đa tuyến thẳng)
210	Hướng trượt (vectơ pháp của mặt phẳng chứa đối tượng) – tuỳ chọn. Mặc định bằng 0, 0, 1

Bảng 6.7.6. Mã nhóm đối tượng đường êlip (Ellipse)

Mã nhóm	*Ý nghĩa*
100	Subclass marker (AcDbEllipse)
10	Tâm (trong hệ tọa độ WCS)
11	Điểm cuối trục chính, đo tương đối với tâm
100	Subclass marker (AcDbArc)
210	Hướng trượt (vectơ pháp của mặt phẳng chứa đối tượng) – tuỳ chọn. Mặc định bằng 0, 0, 1
40	Tỷ số độ dài trục phụ / trục chính
41	Thông số góc đầu của cung êlip. Bằng 0 nếu vẽ cả êlip
42	Thông số góc cuối của cung êlip. Bằng 2π nếu vẽ cả êlip
Lưu ý:	*Ý nghĩa các thông số mã 41, 42 xem phần trợ giúp trực tuyến để xem cách AutoCAD xây dựng một cung êlip như thế nào.*

Bảng 6.7.7. Mã nhóm đối tượng văn bản dạng khối (Mtext)

Mã nhóm	*Ý nghĩa*
100	Subclass marker (AcDbMtext)
10	Điểm chèn
40	Chiều cao chữ đã định
41	Chiều rộng vùng văn bản
71	Kiểu căn lề: 1 = Top left; 2 = Top center; 3 = Top right; 4 = Middle left; 5 = Middle center; 6 = Middle right; 7 = Bottom left; 8 = Bottom center; 9 = Bottom righ
72	Hướng văn bản: 1 = Từ trái qua phải 3 = Từ trên xuống 5 = Theo định dạng (textstyle) mặc định
1	Chuỗi ký tự Nếu dưới 250 ký tự tất cả được ghi trong một nhóm (nhóm 1). Nếu dài hơn, chia thành các nhóm bé hơn kèm theo mã nhóm 3. Trong trường hợp này, nhóm cuối cùng là nhóm 1 và luôn có độ dài bé hơn 250
3	Chuỗi ký tự (luôn có độ dài 250 ký tự)
7	Tên định dạng (text style)
210	Hướng trượt (tuỳ chọn). Mặc định bằng 0, 0, 1
11	Hướng trục X, thể hiện qua điểm nằm trên trục X (WCS)
42	Chiều dài dòng văn bản (bé hơn hoặc bằng giá trị trong mã 41). Mã này chỉ cho phép đọc
43	Chiều cao vùng văn bản (chỉ đọc)
50	Góc nghiêng dòng văn bản (radian)
73	Kiểu cách dòng trong Mtext: 1 = at least (dòng cách dòng tối thiểu, tự giãn dòng khi gặp chữ cao hơn) 2 = exact (ấn định chính xác, không tự giãn dòng)
44	Tỷ lệ giãn dòng. Giá trị cho phép từ 0.24 đến 4 so với mặc định

Bảng 6.7.8. Mã nhóm đối tượng văn bản dòng đơn (Text)

Mã nhóm	*Ý nghĩa*
100	Subclass marker (AcDbText)
39	Chiều dày trượt (thickness)
10	Điểm chèn
40	Chiều cao chữ
1	Nội dung dòng văn bản
50	Góc nghiêng dòng văn bản (radian)
41	Độ giãn ngang của chữ, mặc định bằng 1
51	Góc nghiêng chữ (obligue angle)
7	Tên định dạng văn bản (textstyle)
71	Kiểu chữ, mặc định bằng 0. Các giá trị khác có ý nghĩa sau: 2 = backward; 4 = upsidedown
72	Căn dòng văn bản theo chiều ngang. Số nguyên, nhận một trong các giá trị sau: 0 = Căn trái (Left) - giá trị mặc định; 1 = Căn giữa (Center); 2 = Căn phải (Right); 3 = Align; 4 = Middle; 5 = Fit.
11	Điểm dóng hàng (tuỳ chọn)
210	Hướng trượt (tuỳ chọn). Mặc định bằng 0, 0, 1
100	Subclass marker (AcDbText)
73	Căn dòng văn bản theo chiều dọc. Mặc định = 0 (Baseline). Các giá trị khác: 1 = Bottom; 2 = Middle; 3 = Top

Ví dụ 6.23: *In mã DXF đối tượng vừa vẽ*

```
(defun c:lstDXF ( / ename ecode cnt)
    (setq ename (entlast))          ; lấy tên đối tượng vẽ sau cùng
    (setq ecode (entget ename))     ; lấy DXF code của đối tượng
    (setq cnt 0)
    (textpage)                      ; chuyển sang màn hình text
    (princ "\nDXF codes cua doi tuong ve sau cung...")
    (repeat (length ecode)          ; vòng lặp
        (print (nth cnt ecode))     ; in mỗi dòng một danh sách con
        (setq cnt (1+ cnt))
        )                           ; hết vòng lặp
    (princ)                         ; rút lui im lặng
    )                               ; kết thúc defun
```

Kết quả đoạn thẳng vừa vẽ nối các điểm có tọa độ (1 2) và (6 5) trong mặt phẳng XY ở ví dụ trên sẽ được trình bày trên màn hình văn bản của AutoCAD dưới dạng sau:

```
(-1 . <Entity name: 7ef4fea0>)
(0 . "LINE")
(330 . <Entity name: 7ef4fcf8>)
(5 . "6A")
(100 . "AcDbEntity")
(67 . 0)
(410 . "Model")
(8 . "0")
(100 . "AcDbLine")
(10 1.0 2.0 0.0)
(11 6.0 5.0 0.0)
(210 0.0 0.0 1.0)
```

Việc thêm một bản ghi vào CSDL AutoCAD (tạo đối tượng mới) được thực hiện qua hàm ***(entmake [elist])***, trong đó *elist* là danh sách liên kết thể hiện mã đối tượng. Nếu thành công, hàm trả về giá trị *elist*, ngược lại, nếu không tạo được đối tượng mới, hàm trả về NIL.

Ví dụ dưới đây tạo lớp "MYLAYER" và vẽ đối tượng vòng tròn có tâm tại điểm (5.0 7.0 0.0) và bán kính 1.0 trên lớp vừa tạo.

Ví dụ 6.24: *Tạo đối tượng mới sử dụng mã đối tượng*

```
(entmake '(
    (0 . "CIRCLE")          ; loại đối tượng
    (8 . "MYLAYER")         ; lớp (Layer)
    (10 5.0 7.0 0.0)        ; tâm vòng tròn
    (40 . 1.0) )            ; bán kính
)
```

Sau đó, nếu sử dụng chương trình ở ví dụ 6.23, kết quả trả về như sau:

```
(-1 . <Entity name: 7ef4feb8>)
(0 . "CIRCLE")
(330 . <Entity name: 7ef4fcf8>)
(5 . "8F")
(100 . "AcDbEntity")
(67 . 0)
(410 . "Model")
(8 . "MYLAYER")
(100 . "AcDbCircle")
(10 5.0 7.0 0.0)
(40 . 1.0)
(210 0.0 0.0 1.0)
```

Có thể thấy rằng AutoCAD đã tự thêm các mã nhóm chung vào mã của đối tượng vừa tạo như tên đối tượng, màu,... theo giá trị mặc định.

6.3.3. Sửa đổi các đối tượng AutoCAD

6.3.3.1. Sử dụng Entity và SelectionSet truy cập CSDL AutoCAD

Các lệnh sửa đổi trong AutoCAD nói chung có liên quan đến việc chọn đối tượng hay một nhóm đối tượng trên bản vẽ. Thông thường để yêu cầu người dùng thực hiện việc này AutoCAD đưa lời nhắc *Select Object(s)*... lên dòng lệnh command. Trong lập trình thiết kế, người lập trình cần phải truyền dữ liệu cho lệnh sửa đổi một cách tự động, không qua thao tác chọn trực tiếp, vì vậy AutoLISP cung cấp một dạng dữ liệu kiểu nhóm chọn gọi là *Selection Set* hay *Pick Set* như đã giới thiệu ở phần đầu. Dưới đây giới thiệu một số hàm thông dụng để thao tác với kiểu dữ liệu này:

- ***(ssget [sel-method] [pt1 [pt2]] [pt-list] [filter-list])***

Hàm này dùng để chọn nhóm đối tượng như khi thao tác trực tiếp với AutoCAD. Các tham số của hàm như sau:

sel-method - chuỗi ký tự thể hiện cách chọn nhóm đối tượng. Có thể sử dụng các giá trị sau:

"C" - chọn theo phương pháp Crossing Windows. Các đối tượng nằm trong hoặc cắt biên một vùng chữ nhật xác định bởi các đỉnh đối diện *pt1* và *pt2*.

"CP" - chọn theo Crossing Polygon (đa giác). Các đỉnh của đa giác cho trong tham số *pt-list*.

"F" - chọn theo Fence: các đối tượng cắt qua một đường gấp khúc, xác định bởi các đỉnh trong danh sách *pt-list*.

"L" - chọn đối tượng cuối cùng trong CSDL (đối tượng vẽ sau cùng, nhìn thấy được).

"P" - chọn lại nhóm đối tượng vừa chọn trước đó (Previous) .

"W" - chọn theo kiểu Windows: chỉ chọn các đối tượng nằm lọt trong vùng chữ nhật, xác định bởi các điểm *pt1* và *pt2*.

"WP" - tương tự CP, nhưng chỉ chọn các đối tượng nằm gọn trong đa giác, không cắt biên.

"X" - chọn tất cả các đối tượng trong CSDL AutoCAD, bao gồm cả các đối tượng trên các lớp OFF, FROZEN và nằm ngoài vùng nhìn thấy.

pt1 - điểm xác định vùng sẽ chọn các đối tượng.

pt2 - điểm xác định vùng sẽ chọn các đối tượng.

pt-list - danh sách các điểm, xác định đường gấp khúc hoặc đa giác liên quan đến vùng chọn.

filter-list - danh sách liên kết thể hiện việc chọn loại trừ. Chỉ những đối tượng thoả mãn điều kiện trong danh sách này mới được đưa vào nhóm chọn.

Dưới đây là một số ví dụ về cách thể hiện danh sách này trong việc chọn đối tượng:

Ví dụ 6.25: *Chọn nhóm đối tượng*

Gọi hàm	**Kết quả**
(setq ss1 (ssget '((0 . "TEXT"))))	Thể hiện lời nhắc chung *Select Objects...* để người dùng chọn, nhưng chỉ đưa vào nhóm chọn các đối tượng là văn bản Text.
(setq ss1 (ssget "P" '((0 . "LINE"))))	Chọn các đối tượng là đoạn thẳng Line trong nhóm đối tượng đã chọn trước đó.
(setq ss1 (ssget "W" pt1 pt2 '((8 . "FLOOR9"))))	Chọn các đối tượng nằm trong vùng chữ nhật xác định bởi các đỉnh pt1 và pt2, nhưng chỉ đưa vào kết quả các đối tượng trên lớp có tên là FLOOR9.
(setq ss1 (ssget "X" '((0 . "CIRCLE"))))	Chọn tất cả các đối tượng vòng tròn (Circle) trong CSDL.
(ssget "X" '((0 . "LINE") (62 . 5)))	Chọn tất cả các đối tượng Line được vẽ với màu 5 (Xanh - Blue). Lưu ý rằng các đối tượng cùng màu này nhưng với đặc tính là ByLayer hoặc ByBlock sẽ không được chọn.

Nếu không chỉ rõ phương pháp chọn, AutoLISP sẽ chọn đối tượng đi qua điểm đã chỉ định. Còn nếu không có tham số, hàm sẽ hiện lời nhắc *Select Object(s)...* lên dòng lệnh command để yêu cầu người dùng chọn trực tiếp trên bản vẽ AutoCAD.

Kết quả trả về của hàm là tên nhóm chọn nếu việc lựa chọn thành công, ngược lại, nếu không chọn được đối tượng nào, hàm *ssget* sẽ trả về giá trị NIL.

- ***(ssadd [ename [ss]])***

Hàm này dùng để thêm đối tượng (Entity) vào nhóm chọn. Nếu không có tham số, hàm sẽ tạo một nhóm rỗng. Nếu chỉ có tham số *ename*, hàm

sẽ tạo nhóm đối tượng mới gồm một thành phần, ngược lại, nếu đầy đủ tham số, hàm sẽ thêm đối tượng có tên *ename* vào nhóm *ss*. Kết quả trả về của hàm là tên nhóm chọn *ss*. Các tham số của hàm có ý nghĩa như sau:

ename - tên đối tượng (entity). Có thể lấy tên đối tượng qua các hàm đã trình bày ở mục 3.2.2 hoặc qua hàm *(ssname ss index)* trong phần này.

ss - tên nhóm đối tượng cần thêm vào.

Ví dụ dưới đây minh họa cách sử dụng hàm này.

Ví dụ 6.26: *Thêm đối tượng vào nhóm*

```
(setq ss1 (ssadd))                  ; tạo nhóm đối tượng mới (rỗng), gán
                                    ; kết quả cho biến ss1
(setq ss2 (ssadd (entlast)))        ; tạo nhóm đối tượng gồm phần tử vừa
                                    ; vẽ, kết quả gán cho biến ss2
(ssadd (entlast) ss1)               ; thêm đối tượng vừa vẽ vào nhóm ss1

;;; Đoạn chương trình sau yêu cầu người dùng chọn nhóm đối tượng
;;; và sau đó các đối tượng đã chọn được thêm vào nhóm ss1:
(setq ss (ssget))                   ; chọn nhóm đối tượng, gán kết
                                    ; quả cho biến ss
(setq ind 0                         ; index
      num (sslengh ss)              ; số phần tử trong nhóm đã chọn
      )
(repeat num                         ; lặp num lần
      (setq ent (ssname ss ind))    ; lấy tên phần tử thứ ind
      (ssadd ent ss1)               ; thêm vào nhóm ss1 phần tử trên
      (setq ind (1+ ind))           ; phần tử tiếp theo trong ss
      )
```

- Hàm ***(ssdel ename ss)*** làm công việc ngược lại với hàm ***ssadd***. Công dụng là để loại bớt đối tượng khỏi nhóm chọn (giống như nhập "R" trong khi AutoCAD yêu cầu chọn đối tượng). Nếu *ename* là phần tử trong nhóm chọn *ss* thì hàm sẽ loại khỏi nhóm *ss* đối tượng đã chỉ định và trả về tên nhóm chọn *ss*. Ngược lại, khi *ename* không thuộc nhóm chọn *ss* hàm trả về NIL.

- Hàm ***(sslength ss)*** trả về số nguyên thể hiện số lượng các đối tượng (entity) có trong nhóm chọn *ss*. Xem ví dụ vừa trình bày trên đây để tham khảo cách sử dụng hàm.

- Hàm ***(ssname ss index)*** trả về tên đối tượng thứ *index* trong nhóm chọn *ss*. Lưu ý rằng các đối tượng trong nhóm được đánh số từ 0, tức là

đối tượng thứ nhất có *index* là 0, còn đối tượng có *index bằng (n-1)* là đối tượng cuối cùng (với n là số đối tượng trong nhóm, trả về bởi hàm *sslength*). Xem ví dụ 6.26.

- Hàm ***(ssmemb ename ss)*** dùng để kiểm tra xem đối tượng nào đó có thuộc nhóm chọn hay không. Nếu đối tượng có tên *ename* thuộc nhóm chọn *ss*, hàm trả về tên đối tượng *ename*, còn không, hàm sẽ trả về NIL.

6.3.3.2. Sửa đổi các đối tượng AutoCAD

Một bản vẽ nói chung không thể thiếu các thao tác sửa chữa đối tượng. AutoCAD cung cấp khá nhiều lệnh sử dụng cho mục đích này như Erase để xoá nhóm đối tượng, Copy - sao chép, Rotate - quay, Mirror - lấy đối xứng gương, Trim - xoá một phần đối tượng,... Trong chương trình AutoLISP các thao tác sửa đổi này có thể thực hiện tự động thông qua cách gọi hàm *command* cùng với các lệnh này. Cú pháp và cách sử dụng hàm *command* trong AutoLISP xem chi tiết phần 6.3.2.1, ở đây chỉ lưu ý thêm là các lệnh sửa đổi thường được áp dụng với dữ liệu kiểu nhóm chọn *Selection Set*.

Các hàm AutoLISP dưới đây minh họa việc sử dụng lệnh sửa đổi của AutoCAD với nhóm đối tượng. Nhóm đối tượng mới được thêm vào bản vẽ bằng cách lấy đối xứng, sao chép hoặc quay nhóm đối tượng đã chọn. Kết quả trả về của hàm là nhóm đối tượng vừa được tạo ra. Bằng cách này, người lập trình có thể quản lý không những các đối tượng đã có, mà còn quản lý được các đối tượng vừa tạo.

Ví dụ 6.27: *Một số hàm thao tác với nhóm đối tượng*

```
; Dưới đây là 3 hàm dùng để thực hiện sửa đổi nhóm đối tượng.
; Phần in đậm là tên hàm. Công dụng chung của các hàm này được mô tả
; vắn tắt ở các dòng chú thích phía dưới.

; (defun ss-mirror (ss p1 p2 flag)
; Hàm trả về nhóm đối tượng mới sau khi lấy đối xứng nhóm ss đã chọn
; qua đường thẳng nối hai điểm p1 và p2
; Flag: kiểu đối xứng: 0 - xoá bản gốc khi xong;  1 - giữ bản gốc

; (defun ss-copy (ss p1 p2)
; Hàm trả về nhóm đối tượng mới bằng cách sao chép nhóm ss đã chọn
; từ điểm p1 đến điểm p2.

; (defun ss-rcopy (ss pt ang)
; Hàm thực hiện việc sao chép tại chỗ nhóm đối tượng ss và quay nhóm đối
; tượng mới quanh điểm pt một góc ang (đo bằng độ).
; Trả về nhóm đối tượng mới.
```

```
(defun ss-mirror (ss p1 p2 flag / ent ss1 num ind flag1)
  (if (null ss)
   (setq ss1 NIL)      ; nếu nhóm ss rỗng (null object), trả về NIL
   (progn              ; nếu ngược lại, thực hiện lấy đối xứng
    (if (= flag 0)
     (progn
      (command "._Mirror" ss "" p1 p2 "y")
      (setq ss1 ss)
      )
     (progn
      (setq ss1 (ssadd))
      (setq num (sslength ss))
      (setq ind 0)
      (while (< ind num)
             (setq ent (ssname ss ind))
             (command "._Mirror" ent "" p1 p2 flag1)
             (ssadd (entlast) ss1)
             (setq ind (1+ ind))
             )          ; while
      )
     )          ; if
    )           ; progn
   )            ; if ss not null
  ss1           ; kết quả trả về
  )             ; defun

(defun ss-copy (ss p1 p2 / ss1 ent num ind)
  (if (null ss)
   (setq ss1 NIL)      ; nếu ss rỗng (null object), trả về NIL
   (progn              ; ngược lại...
    (setq ss1 (ssadd))
    (setq num (sslength ss))
    (setq ind 0)
    (while (< ind num)
     (setq ent (ssname ss ind))
     (command "._Copy" ent "" p1 p2)
     (ssadd (entlast) ss1)
     (setq ind (1+ ind))
     )          ; while
    )           ; progn
   )            ; if
  ss1           ; kết quả trả về
  )             ; defun
```

```
(defun ss-rcopy (ss pt ang / ss1 ent num ind)
  (if (null ss)
  (setq ss1 NIL)      ; nếu ss rỗng (null object), trả về NIL
  (progn              ; ngược lại...
   (setq ss1 (ssadd))
   (setq num (sslength ss))
   (setq ind 0)
   (while (< ind num)
    (setq ent (ssname ss ind))
    (command "._Copy" ent "" pt pt)
    (command "._Rotate" (entlast) "" pt ang)
    (ssadd (entlast) ss1)
    (setq ind (1+ ind))
    )          ; while
   )           ; progn
  )            ; if
 ss1           ; kết quả trả về
 )             ; defun
```

Một cách khác để thay đổi đối tượng là truy cập trực tiếp vào CSDL AutoCAD, sửa đổi chúng và cập nhật lại đối tượng đã thay đổi.

Các tính chất của đối tượng có thể thay đổi bằng cách thay trực tiếp mã đối tượng qua hàm ***(entmod elist)***. sau đó hình ảnh mới của đối tượng được cập nhật lại qua hàm ***(entupd ename)***. Trong các hàm này, *elist* là danh sách liên kết thể hiện mã đối tượng, còn *ename* là tên đối tượng cần cập nhật.

Giả sử vừa gọi lệnh AutoLISP sau đây (xem ví dụ 6.24) để vẽ vòng tròn bán kính 1.0, tâm có tọa độ (5.0 7.0 0.0) trên lớp MYLAYER:

```
(entmake
   '(                          ; bắt đầu danh sách liên kết
      (0 . "CIRCLE")           ; loại đối tượng
      (8 . "MYLAYER")          ; lớp (Layer) chứa đối tượng
      (10 5.0 7.0 0.0)         ; tâm vòng tròn
      (40 . 1.0)               ; bán kính vòng tròn
    )                          ; kết thúc danh sách liên kết
  )
```

sau đó cần thay đổi bán kính vòng tròn vừa tạo ra thành 10.0 và đổi màu đường tròn (BYLAYER) thành màu đỏ. Để thực hiện điều này cần truy cập tới CSDL, lấy các thông tin về đối tượng vừa vẽ, thay đổi các thông số tương ứng (bán kính - mã 10; tên lớp - mã 62). Lưu ý rằng, mã 62 chưa có trong mã nhóm đối tượng (do lớp đặt mặc định BYLAYER).

Ví dụ 6.28: *Sửa đổi đối tượng bằng entmod và entupt*

```
(setq newrad (cons 40 10.0))      ; bán kính mới
(setq newcolor (cons 62 1))       ; màu mới (1 = đỏ)
(setq ent (entlast))              ; lấy tên đối tượng vừa vẽ
(setq ed (entget ent))            ; lấy thông tin mã đối tượng
                                  ; thay màu mới và bán kính mới
(if (setq color (assoc 62 ed))    ; nếu đã được đặt màu (tồn tại mã 62)
        (setq ed (subst newcolor color ed ))      ; thì thay mới
        (setq ed (append ed (list newcolor)))     ; nếu không - thêm vào
        )
(setq ed (subst newrad (assoc 40 ed) ed))
(entmod ed)                       ; cập nhật dữ liệu
(entupd ent)                      ; cập nhật đối tượng
```

Trong ví dụ này đã sử dụng các hàm thao tác với danh sách: *assoc* - tìm kiếm và *subst* - thay thế phần tử trong danh sách.

Trong thực tế, nhiều khi cần kết hợp cả 2 phương pháp sửa đổi tượng: qua hàm *command* kết hợp với truy cập CSDL. Ví dụ sau đây yêu cầu người dùng chọn nhóm đối tượng, sau đó chỉ đưa vào nhóm chọn các đối tượng vòng tròn và vẽ thêm các đường tâm cho các vòng tròn này.

Ví dụ 6.29: *Thêm đường tâm cho vòng tròn*

```
(setq ss (ssget '((0 . "CIRCLE"))))  ;yêu cầu người dùng chọn đối tượng
                                      ; chỉ đưa các đường tròn vào nhóm
(setq ind 0                           : index
      num (sslength ss))              ; số phần tử trong nhóm đã chọn
(repeat num                           ; vòng lặp xử lý từng đối tượng
      (setq ent (ssname ss ind))      ; lấy tên phần tử thứ ind
      (setq ecode (entget ent))       : lấy mã đối tượng
      (setq cen (cdr (assoc 10 ecode)))  ; lấy tâm đường tròn
      (setq rad (cdr (assoc 40 ecode)))  ; lấy bán kính đường tròn
      ;; vẽ 2 đường tâm có độ dài 1.1 đường kính qua tâm vòng tròn
      (setq c1 (polar cen 0.0 (* 1.1 rad))
            c2 (polar cen pi (* 1.1 rad))
            c3 (polar cen (* pi 0.5) (* 1.1 rad))
            c4 (polar cen (* pi 1.5) (* 1.1 rad)))
      (command "._Line" c1 c2 "")
      (command "._Line" c3 c4 "")
      (setq ind (1+ ind))
) ; kết thúc vòng lặp repeat...
```

6.4. THIẾT KẾ VÀ ĐIỀU KHIỂN HỘP THOẠI DCL

Một phương thức giao tiếp người - máy khác hay được sử dụng trong lập trình là hộp thoại (Dialog Box). So với cách giao tiếp trực tiếp trên dòng lệnh, hộp thoại có nhiều ưu điểm hơn. Đối với hộp thoại, người sử dụng chương trình không nhất thiết phải nhập số liệu tuần tự. Mặt khác các dữ liệu tiêu chuẩn có thể chọn thay vì gõ trực tiếp nên tránh được sai sót và hạn chế nhập sai dữ liệu. Một điều nữa, giao diện hộp thoại trông thân thiện hơn là kiểu nhắc nhập trực tiếp trên dòng lệnh, giúp người dùng thoải mái hơn khi sử dụng.

Hộp thoại gồm các nút điều khiển, gọi là *tile*. Có thể tạo một tile phức hợp (gọi là *subassembly*) gồm nhiều nút, sắp xếp trong một dòng hoặc cột, bao hoặc không bao bởi một đường viền. Bản thân các dòng và cột chứa các tile được gọi là *cluster*. Các tile phức hợp này có thể được sử dụng cho nhiều hộp thoại khác nhau. Ví dụ, các nút OK, Cancel, Help được gom thành nhóm và được sử dụng cho nhiều hộp thoại khác nhau trong AutoCAD. Các *subassembly* được coi như một *tile* đơn. Các tile trong nhóm *subassembly* được gọi là *children*.

6.4.1. Thiết kế hộp thoại DCL

Hộp thoại được tổ chức theo cấu trúc cây, với phần gốc là tile với tên gọi *dialog*, định nghĩa bản thân hộp thoại, tiếp đến là các tile, các cluster, subassembly,... Trên hình 6.7 trình bày cấu trúc cây của hộp thoại nhập thông số vít 6 cạnh tiêu chuẩn, còn trên hình 6.8 là bản thân giao diện của hộp thoại này.

Hộp thoại được định nghĩa qua file văn bản ASCII, viết theo cú pháp của ngôn ngữ DCL (Dialog Control Language). File chứa định nghĩa các tile, cluster, subassembly, cách xắp xếp và độ lớn của chúng được lưu trong file có phần mở rộng *.DCL. Một file DCL có thể chứa nhiều hộp thoại, cũng như các định nghĩa tile và subassembly khác nhau. Hộp thoại và các tile của chúng được điều khiển thông qua các hàm AutoLISP. Môi trường VisualLISP IDE cung cấp tính năng *Tools / Interface tools / Preview DCL /...* để xem trước kết quả hộp thoại đang thiết kế.

Một số lưu ý khi thiết kế hộp thoại:

- Thuận tiện cho người dùng:

Vẻ ngoài của hộp thoại rất quan trọng: nếu trình bày quá nhiều tile trên một hộp thoại sẽ rất khó cho người sử dụng khi thao tác. Mặt khác, việc sắp xếp các tile nên khoa học, dễ tiếp cận, các tile hay sử dụng (ví dụ, OK, Cancel...) nên đặt ở vị trí thống nhất trong các hộp thoại.

- Thống nhất với các ứng dụng liên quan:

Một hộp thoại trình bày thống nhất với các hộp thoại khác sẽ làm người sử dụng dễ chịu hơn là trình bày tuỳ ý. Ví dụ, các nút OK, Cancel, Help thường được đặt phía đáy hộp thoại, nếu khác đi, người dùng sẽ mất công tìm kiếm xem chúng ở đâu và do vậy, gây thêm những phiền toái không cần thiết. Một cách tốt nhất để thống nhất hoá cách trình bày là sử dụng các hộp thoại có sẵn, sửa đổi lại nội dung cho phù hợp với yêu cầu của mình. Có thể tham khảo các file định nghĩa hộp thoại *base.dcl* và *acad.dcl* có sẵn trong folder *Support* của AutoCAD.

- Sử dụng ngôn ngữ rõ ràng:

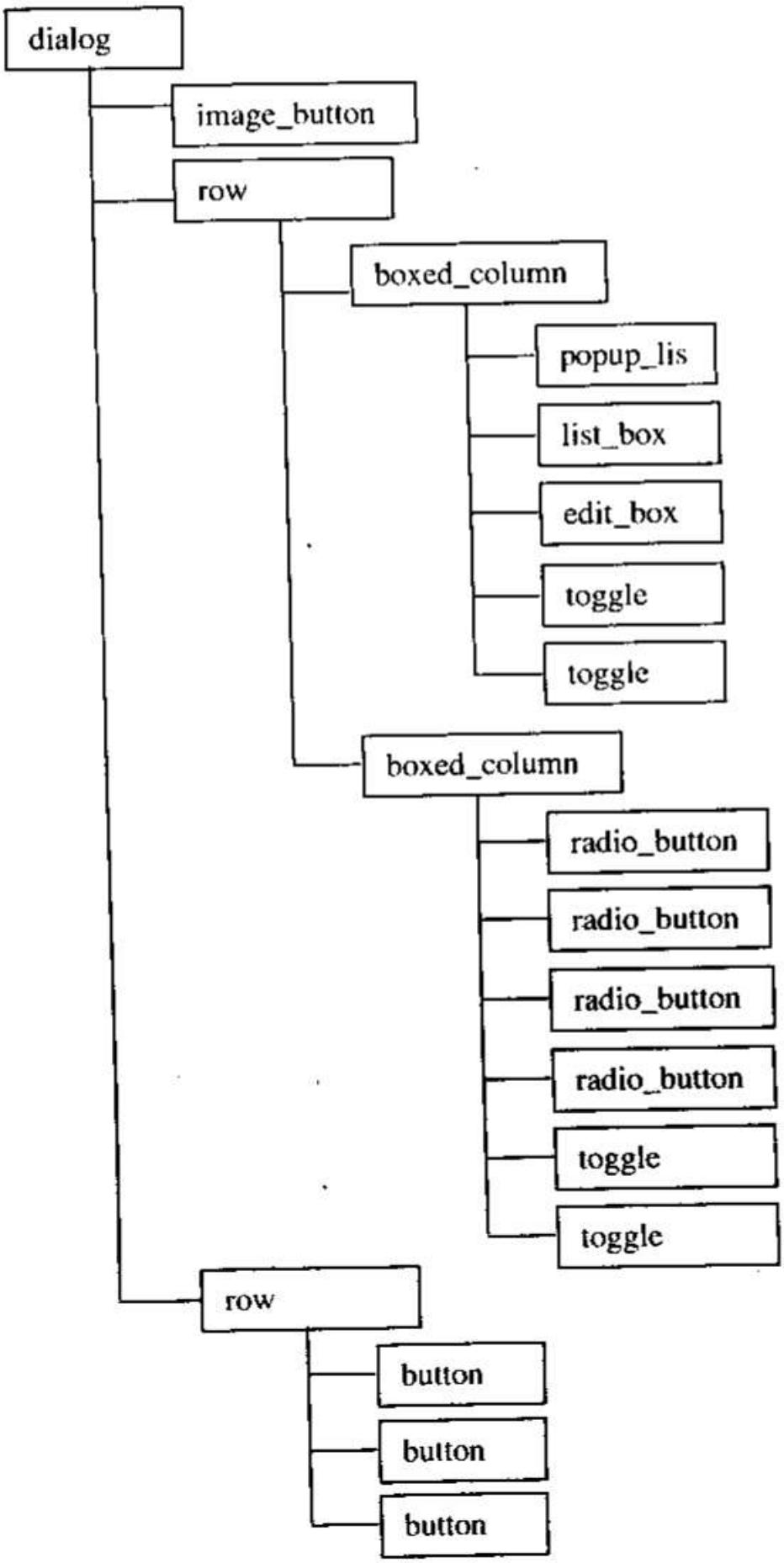
Các văn bản kèm theo trong hộp thoại (tên hộp thoại, các tile, các dòng chú thích,...) cần thể hiện ngắn gọn, nhưng thông tin cần đầy đủ, rõ ràng và chính xác, tránh dùng tiếng lóng hoặc viết tắt làm người dùng không thể hiểu. Một hình vẽ minh họa các dữ liệu cần nhập sẽ rất hữu ích và có thể thay thế cho những câu văn dài.

- Hạn chế người dùng nhập sai dữ liệu:

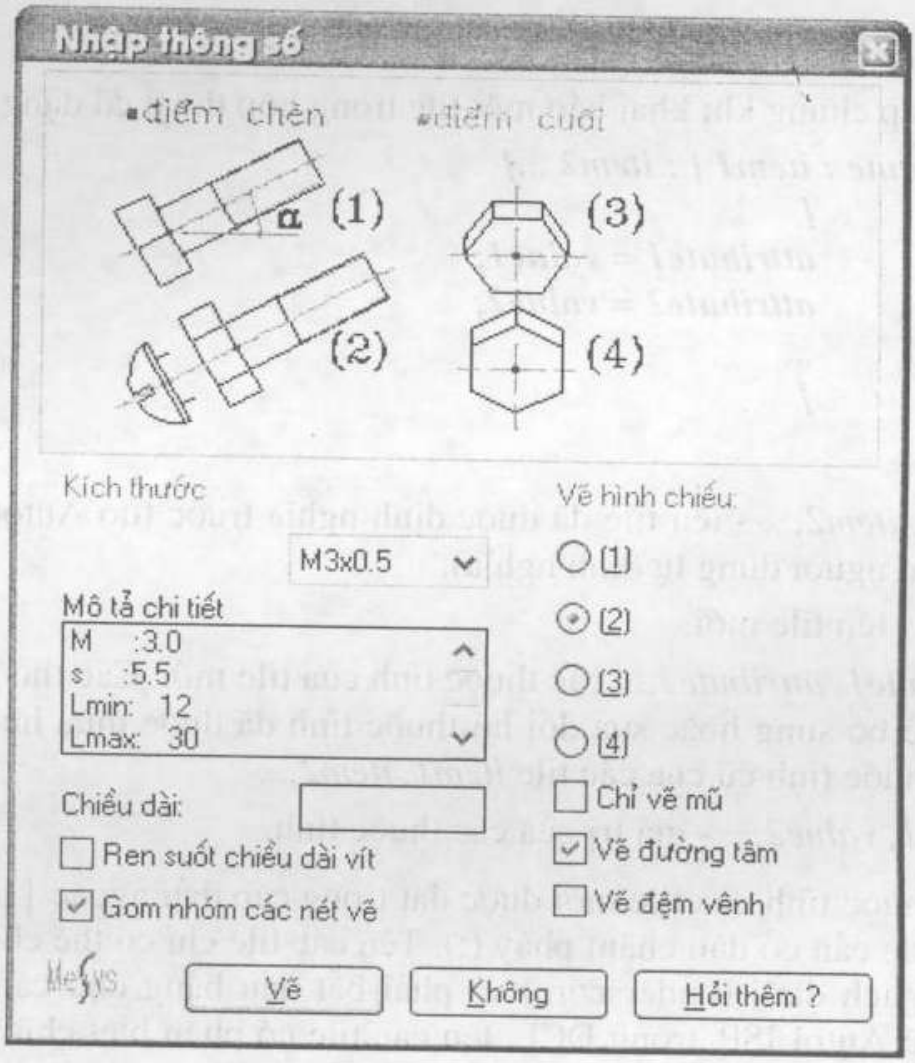
Những dữ liệu tiêu chuẩn (đường kính bulông, môđun bánh răng,...) nên thiết kế thành danh sách (popup_list hoặc list_box) để người dùng chọn thay vì gõ trực tiếp qua hộp text_box. Điều này sẽ tránh cho người dùng nhập sai thông số hoặc thông số phi tiêu chuẩn. Khi dữ liệu không được phép sửa hoặc không cần nhập cho trường hợp cụ thể nào đó thì các tile tương ứng này nên tắt đi để người dùng không thể truy cập. Khi có các giá trị hay dùng, ít thay đổi hoặc dùng chung cho nhiều bộ dữ liệu khác nhau thì nên đặt giá trị mặc định cho các tile tương ứng. Việc này sẽ làm người dùng nhập dữ liệu nhanh hơn. Một cách khác cũng hay được sử dụng là lấy giá trị các tile ở lần gọi hộp thoại trước đó để đặt mặc định cho lần gọi sau, việc này rất có lợi khi cần tính toán, thiết kế với nhiều phương án khác nhau.

- Sử dụng phím tắt:

Đôi khi người dùng không thích sử dụng thiết bị chuột hoặc con chuột bị hỏng, khi đó để chương trình hoạt động bình thường nên cung cấp cho người dùng chức năng truy cập các tile từ bàn phím. Thông thường phím Tab được dùng để di chuyển từ tile này đến tile khác, phím Space để thay đổi giá trị các nút kiểm (toggle), phím Enter dùng để nhập các giá trị trên hộp thoại, còn Esc sẽ bỏ qua các giá trị đã nhập và thoát hộp thoại. Nếu muốn di chuyển nhanh đến một tile nào đó mà không cần dùng phím Tab duyệt qua các tile khác, tile này cần được gán phím tắt (gọi là *mnemonic*). Trong ví dụ trên hình 6.8, việc chọn hình chiếu vẽ vít tiêu chuẩn được thể hiện qua các nút radio (1) (2) (3) (4) đã được gán các phím tắt. Người



Hình 6.7: Cấu trúc cây của hộp thoại



Hình 6.8: Hộp thoại nhập thông số vít tiêu chuẩn

dùng chỉ cần nhấn các phím 1, 2, 3 hoặc 4 trên bàn phím để chọn hình chiếu tương ứng. Thông thường phím tắt được lấy từ chữ cái trong tên của tile, nếu được đặt *mnemonic* chữ cái này sẽ được gạch chân (các chữ cái V, K, H trong các nút *Vẽ*, *Không*, *Hỏi thêm* ở ví dụ trên hình 6.8).

- Kế thừa các hộp thoại hoặc tile:

Nhiều hộp thoại và tile cùng được sử dụng cho các ứng dụng khác nhau. Để thống nhất và tiết kiệm thời gian, file DCL mới có thể kế thừa các định nghĩa trong một file DCL đã có bằng cách tham chiếu đến chúng thông qua lời gọi *@include dclfile*, đặt trên đầu file mới. Nếu tên file tham chiếu đến (*dclfile*) không ghi kèm đường dẫn, AutoLISP sẽ tìm nó trước tiên trong folder hiện hành, sau đó mới tìm trong folder chứa file DCL mới, còn nếu kèm đường dẫn đầy đủ, nó chỉ được tìm trong folder đã được chỉ định. Lưu ý rằng, không thể tham chiếu tới file acad.dcl, nên việc viết *@include "acad.dcl* sẽ làm xuất hiện lỗi. Tuy vậy, nếu muốn sử dụng các định nghĩa hộp thoại trong file acad.dcl, có thể mở file này và

sao chép phần cần thiết sang file của mình.

Cú pháp chung khi khai báo một tile trong hộp thoại có dạng sau:

```
name : item1 [ : item2 ...]
        {
        attribute1 = value1;
        attribute2 = value2;
        ...
        }
```

trong đó:

item1, *item2*,... - tên tile đã được định nghĩa trước (do AutoCAD hoặc do người dùng tự định nghĩa).

name - tên tile mới.

attribute1, *attribute2*,... các thuộc tính của tile mới. Các thuộc tính này sẽ bổ sung hoặc sửa đổi lại thuộc tính đã được thừa hưởng từ các thuộc tính cũ của các tile *item1*, *item2*,...

value1, *value2*,... - giá trị của các thuộc tính.

Các thuộc tính của tile mới được đặt trong cặp dấu ngoặc { } và sau giá trị mỗi tile cần có dấu chấm phẩy (;). Tên các tile chỉ có thể chứa chữ cái, số, dấu gạch dưới (underscore) và phải bắt đầu bằng chữ cái. Khác với ngôn ngữ AutoLISP, trong DCL, tên các tile có phân biệt chữ hoa và chữ thường, chẳng hạn BigButton và BigBUTTON là không như nhau.

Trên hộp thoại ở hình 6.8, nút "*Hỏi thêm* ?" được định nghĩa như sau:

```
help1 : button {
        label="&Hỏi thêm ?";
        key="help";
        width=13;
        fixed_width=true;
        alignment=centered;
    } // end help
```

trong đó:

help1 - tên tile mới.

button - tên tile được AutoCAD định nghĩa, với 2 thuộc tính được đặt trước là chiều cao không thay đổi và mặc định dừng Tab:

```
button : tile {
            fixed_height = true;
            is_tab_stop = true;
          }
```

Các thuộc tính (attribute) mới do người dùng đưa thêm gồm:

label - nhãn của tile (Hỏi thêm ?)

key - khoá tham chiếu tới tile (help)

width - chiều rộng tile (13)

fixed-width - chiều rộng tile cố định (true = không được thay đổi).

alignment - dóng hàng (centered = căn giữa).

Hai dấu xổ chéo (//) dùng để đặt trước các chú thích. Các ký tự sau 2 dấu này được AutoLISP bỏ qua không xử lý khi chạy chương trình. Nếu muốn ghi chú thích kiểu *inline*, cần thể hiện chú thích như sau: /* *Đây là chú thích...* */

Để tham chiếu tới tile (đưa tile vào trong cluster hoặc hộp thoại) có thể sử dụng một trong các cú pháp sau:

name;

hoặc

```
: name {
        attribute = value;
        ...
        }
```

trong đó *name* là tên tile đã định nghĩa. Với cú pháp thứ nhất, tất cả các thuộc tính của tile được giữ nguyên theo định nghĩa, còn với cú pháp thứ 2, có thể thêm hoặc sửa đổi một số thuộc tính của tile. Cần lưu ý rằng, cú pháp thứ 2 chỉ áp dụng được cho tile đơn, không áp dụng cho nhóm subassembly.

Ví dụ 6.30 trình bày nội dung đầy đủ của file DCL tương ứng với hộp thoại vẽ vít tiêu chuẩn trên hình 6.8.

Ví dụ 6.30: *Ngôn ngữ DCL*

```
dcl_settings : default_dcl_settings { audit_level = 3; } // đặt mức kiểm tra lỗi
InputData : dialog {                    // tên hộp thoại là InputData
  label = "  Nhập thông số ";           // tiêu đề hộp thoại
  : image_button {                      // nút ảnh minh họa
     width=35;
     aspect_ratio=2.0;
     height=12;
     color= 0;
     key="hv_image";
     allow_accept=true;
     }
```

```
: row {                                  // cluster để gom 2 cột trên 1 dòng
  : boxed_column{                        // cột thứ nhất
    label="Kích thước";                  // tiêu đề của cột này
    : popup_list {                       // danh sách popup chứa đường kính
        key="kichthuoc";                 // danh nghĩa của vít tiêu chuẩn
        edit_width=10;
        }
     : list_box {
        label="Mô tả chi tiết";          // hộp danh sách thể hiện các
        key="motachitiet";               // kích thước khác của vít đã chọn
        width=4;
        height=6;
        }
     : edit_box {
        label="Chiều dài:";              // hộp text nhập chiều dài vít
        key="l";
        edit_width=10;
        }
     : toggle {
        label="Ren suốt chiều dài vít"; // hộp kiểm (toggle, checkbox)
        value="0";                       // mặc định ô này không được kiểm
        key="allren";                    // ren chỉ cắt trên phần đầu vít
        }
     : toggle {
        label="Gom nhóm các nét vẽ";
        value="1";                       // mặc định ô này được đánh dấu kiểm
        key="group";                     // nhằm gom nhóm các nét vẽ
        }
     }                                   // kết thúc cột thứ nhất
  : boxed_column{                        // sang cột thứ 2
        label="Vẽ hình chiếu:"; // chọn cách thể hiện
        : radio_button{
            label="(&1)";    // kiểu 1
            key="hcc";
            //value="";
            }
        : radio_button{
            label="(&2)";    // kiểu 2
            key="hcd";
            value="1";
            }
```

```
    : radio_button{
            label="(&3)";   // kiểu 3
            key="hcbc";
            }
    : radio_button{
            label="(&4)";   // kiểu 4
            key="hcbd";
            }
    : toggle {
            label="Chỉ vẽ mũ";       // chỉ vẽ mũ hay toàn bộ vít
            value="0";                // mặc định: không
            key="mu";                 // tức là vẽ toàn bộ vít
            }
    : toggle {
            label="Vẽ đường tâm"; // có vẽ đường tâm không
            value="1";                // mặc định: có
            key="tam";
            }
    : toggle {
            label="Vẽ đệm vênh";  // có vẽ đệm vênh không
            value="0";                // mặc định: không
            key="demvenh";
            }
      }                       // kết thúc cột 2
                              // kết thúcdòng 1
}
                              // sang dòng 2
: row{                        // gọi các tile định nghĩa ở phần dưới
    mesys11;
    ok1;
    cancel1;
    help1;
    }
}                             // kết thúc định nghĩa hộp thoại InputData

// Dưới đây là các tile tự định nghĩa
// mesys11 : nút ảnh thể hiện lôgô chương trình
// ok1 : nút Vẽ ;  cancel1 : nút Không;  help1 : nút Hỏi thêm ?

mesys11 : image{
            key="mesys11";
            width=8;
            height=2;
            fixed_width=true;
            color=dialog_background;
            }
```

```
ok1      : button {
                  label="&Vẽ";
                  key="accept";
                  width=13;
                  fixed_width=true;
                  is_default=true;
                  alignment=centered;
                  }

cancel1 : button {
                  label="&Không";
                  key="cancel";
                  is_cancel=true;
                  width=13;
                  fixed_width=true;
                  alignment=centered;
                  }

help1 : button {
                  label="&Hỏi thêm ?";
                  key="help";
                  width=13;
                  fixed_width=true;
                  alignment=centered;
                  }
```

Mức kiểm tra lỗi *dcl_setting : default_dcl_setting {audit_level = value;}* gồm 4 mức 0, 1, 2 và 3 được đặt qua giá trị *value* ở trong file DCL nhằm mục đích kiểm tra xem file DCL có lỗi hay thừa dòng lệnh nào không. Việc kiểm tra được tiến hành khi chương trình được thực hiện. Khi thiết kế hộp thoại nên đặt mức 3, và sau khi đã thực hiện trôi chảy, hãy chuyển về mức 0.

Phần tiếp theo sau đây sẽ giới thiệu chi tiết một số tile thông dụng trong ngôn ngữ DCL và ý nghĩa các thuộc tính của chúng.

6.4.2. Các tile trong DCL và thuộc tính của chúng

AutoCAD định nghĩa sẵn một số tile, các tile này được sử dụng làm cơ sở để tạo các tile mới hoặc các subassembly. Phần định nghĩa của chúng và các chú thích cho trong file base.dcl.

Các tile hoạt động gồm:

button: sử dụng tạo các nút ấn như OK, Cancel,...

image_button: nút ảnh, thường dùng để tạo các hình minh họa.

edit_box: hộp văn bản, sử dụng để người dùng nhập dữ liệu số, văn bản,... từ bàn phím.

popup_list: danh sách thu gọn, sử dụng chọn các dữ liệu theo dãy tiêu chuẩn. Khi kích vào tile này, các lựa chọn sẽ hiện ra.

list_box: hộp danh sách. Khác với popup_list, hộp danh sách thể hiện tất cả các lựa chọn trong một hộp chữ nhật. Nếu kích thước hộp không đủ chứa toàn bộ danh sách, sẽ xuất hiện các thanh trượt (scroll bar).

radio_button: nút radio, thường dùng khi có nhu cầu chỉ chọn một trong số ít các lựa chọn cho trước.

toggle: nút kiểm (checkbox), sử dụng để đánh dấu khẳng định hoặc không với các lựa chọn có 2 khả năng trả lời - có hoặc không.

slider: thanh trượt, sử dụng khi muốn chọn một số nằm trong vùng cho trước.

Các *cluster* dùng để gom nhóm các tile bao gồm:

dialog: gom các tile, subassembly, cluster để tạo hộp thoại.

row: gom các tile thành hàng.

boxed_row: gom các tile thành hàng và vẽ đường viền quanh nhóm tile trong khối.

column: gom các tile thành cột.

boxed_column: gom các tile thành cột và vẽ đường viền quanh nhóm tile trong khối.

radio_row: gom các nút radio thành nhóm theo hàng.

boxed_radio_row: gom các nút radio thành hàng và vẽ đường viền quanh nhóm tile.

radio_column: gom các nút radio thành nhóm theo cột.

radio_boxed_column: gom các nút radio thành cột và vẽ đường viền quanh nhóm tile.

Các tile trang trí dùng để sắp xếp tile, tạo hình minh họa hoặc ghi chú thích. Các tile này không nhận các tác động từ người dùng, nghĩa là khi nhấp chuột vào chúng, không nhận được phản ứng gì. Loại này bao gồm:

image: hiển thị hình ảnh lên hộp thoại. Khác với *image_button* có thể nhận tác động từ người dùng như một nút ấn, *image* chỉ hiện hình ảnh mà không nhận một tác động nào.

text: hiển thị văn bản lên hộp thoại. Khác với *text_box* dùng để nhập dữ liệu, *text* chỉ hiển thị dữ liệu.

spacer: bình thường khoảng cách các các tile được tự động tính toán, *spacer* dùng để thay đổi lại khoảng cách các tile theo ý người thiết kế hộp thoại.

Kèm theo mỗi tile là các thuộc tính đặc trưng cho việc thể hiện chúng trên hộp thoại. Bảng 6.8 tóm tắt một số thuộc tính thông dụng.

Bảng 6.8. Một số thuộc tính thông dụng của tile

Tên thuộc tính	*Áp dụng cho loại tile*	*Kiểu dữ liệu và ý nghĩa của thuộc tính*
action	tất cả các tile hoạt động	Chuỗi ký tự thể hiện biểu thức AutoLISP sẽ thực hiện khi tile được chọn. Không thể sử dụng hàm *command* trong biểu thức này.
alignment	tất cả các tile	Cách dóng các tile theo chiều ngang hoặc dọc trong nhóm cluster. Với cluster dạng *column*, các giá trị gồm *left* (căn trái, mặc định), *right* (căn phải), *centered* (căn giữa). Còn với cluster dạng *row*, các giá trị gồm *top* (căn từ trên xuống), *bottom* (dưới lên) hoặc *centered* (căn giữa).
color	image, image_button	Màu nền của nút ảnh. Giá trị có thể là số nguyên thể hiện màu (1=đỏ, 2=vàng,...) hoặc các từ khoá như *dialog_foreground* (màu văn bản trên hộp thoại hiện hành), *dialog_background* (màu nền hộp thoại), *red* (đỏ), *blue* (xanh dương),...
edit_width	edit_box, popup_list	Chiều rộng của các hộp nhập dữ liệu (số nguyên hoặc số thực thể hiện số ký tự có thể xuất hiện trong hộp).
height	tất cả các tile	Số thực hoặc số nguyên thể hiện chiều cao của tile (cao bằng bao nhiêu ký tự).
is_cancel	button	Giá trị gồm một trong 2 từ khoá: *true* hoặc *false*. Mặc định là *false*. Khi đặt giá trị *true*, việc nhấn phím Esc trên bàn phím sẽ có tác động tương đương với việc chọn tile này. Trong hộp thoại chỉ 1 tile duy nhất được đặt thuộc tính này bằng *true*.

Tiếp bảng 6.8

Tên thuộc tính	*Áp dụng cho loại tile*	*Kiểu dữ liệu và ý nghĩa của thuộc tính*
is_default	button	Giá trị gồm một trong 2 từ khoá: *true* hoặc *false*. Mặc định là *false*. Khi đặt giá trị *true*, việc nhấn phím Enter trên bàn phím sẽ có tác động tương đương với việc chọn tile này. Trong hộp thoại chỉ có 1 tile duy nhất được đặt thuộc tính này bằng *true*.
is_enable	tất cả các tile hoạt động	Cho phép tile có thể truy cập hay không ngay khi khởi tạo. Giá trị mặc định *true* cho phép truy cập đến tile. Nếu muốn tile không được sử dụng ngay, hãy đặt giá trị *false*. Thuộc tính này có thể thay đổi khi chạy chương trình qua hàm mode_tile của AutoLISP.
key	tất cả các tile hoạt động	Chuỗi ký tự thể hiện khoá tham chiếu đến tile. Phân biệt chữ hoa, chữ thường.
label	button, toggle, text, edit_box, list_box, popup_list, radio_button, boxed_column, boxed_khác,...	Chuỗi ký tự thể hiện tên nút button, dòng văn bản sau dấu kiểm, nút radio, tiêu đề của nhóm chọn list_box, pupup_list, hộp nhập dữ liệu edit_box, tiêu đề nhóm,... Để gán phím tắt cho các tile hoạt động, có thể thêm dấu & vào trước ký tự thể hiện phím tắt này.
list	list_box, popup_list	Đặt mặc định các lựa chọn sẽ được dùng trong danh sách ngay khi khởi tạo hộp thoại. Các lựa chọn này cho dưới dạng chuỗi ký tự, trong đó các phần tử được cách nhau bởi ký tự dòng mới "\n". Ví dụ: list = "Red\nBlue\nGreen"
mnemonic	tất cả các tile hoạt động	Đặt phím tắt cho tile. Giá trị của thuộc tính này là chuỗi, thể hiện ký tự phím tắt. Ký tự này cần thuộc một trong các ký tự trong label (tên) của tile. Phím tắt cũng có thể được đặt luôn bằng thuộc tính label đã nói ở trên.
value	text và các tile hoạt động (trừ button và image_button)	Chuỗi ký tự thể hiện giá trị đặt ban đầu cho tile (khi khởi tạo). Giá trị này có thể thay đổi khi chạy chương trình qua hàm set_tile của AutoLISP.
width	tất cả các tile	Số thực hoặc số nguyên thể hiện chiều rộng của tile (đo bằng số lượng các ký tự).

6.4.3. Điều khiển hộp thoại bằng AutoLISP

Hộp thoại viết bằng ngôn ngữ DCL được điều khiển bằng AutoLISP. Các hàm AutoLISP cho phép điều khiển việc nạp, hiện, đóng,... hộp thoại tổng thể, cũng như khả năng điều khiển các tile riêng lẻ và lấy dữ liệu đã nhập trên hộp thoại.

Giả sử nội dung trong ví dụ 6.30 trên đây được lưu vào file vidu6-30.dcl. Để sử dụng hộp thoại này trong chương trình AutoLISP cần thực hiện các bước sau:

1. Sử dụng hàm ***(load_dialog tên_file_dcl)*** để nạp file DCL vào bộ nhớ. Ví dụ, sử dụng biểu thức AutoLISP sau đây:

(setq dcl_id (load_dialog "vidu6-30.dcl"))

Hàm này sẽ trả về số hiệu file DCL, số hiệu này được gán cho biến dcl_id. Nếu không thành công hàm trả về số âm. Biến này được sử dụng cho các thao tác tiếp theo.

2. Gọi hàm ***new_dialog*** với các tham số tên hộp thoại muốn hiện ra và số hiệu DCL vừa nhận được trong hàm trên:

(new_dialog "InputData" dcl_id)

Nếu thành công, hàm trả về giá trị T, ngược lại hàm trả về NIL.

3. Khởi tạo hộp thoại bằng việc đặt các thông số ban đầu cho các tile, danh sách, hình ảnh,... qua các hàm xử lý tile sẽ được trình bày ở cuối phần này.

4. Gọi hàm ***start_dialog*** gửi các điều khiển và làm hiện hộp thoại trên màn hình AutoCAD cho người dùng:

(start_dialog)

Hàm này sẽ hoạt động cho đến khi người dùng chọn một tile (thường là một button) đã được gán biểu thức AutoLISP có chứa hàm ***done_dialog***. Hàm start_dialog trả về giá trị *status* trong biểu thức gọi hàm done_dialog, còn nếu bỏ qua tham số *status*, hàm trả về 1 khi người dùng chọn tile được gán is_default bằng true, trả về 0 khi chọn tile có is_cancel là true.

5. Gọi hàm ***unload_dialog*** để giải phóng hộp thoại khỏi bộ nhớ sau khi sử dụng:

(unload_dialog dcl_id)

Hàm này được dùng khi muốn tiết kiệm bộ nhớ hoặc khi có nhu cầu định nghĩa lại hộp thoại từ một file DCL khác. Tham số *dcl_id* là số hiệu nhận được từ hàm load_dialog. Hàm luôn trả về giá trị NIL.

Cần lưu ý rằng trong khi hộp thoại đang có hiệu lực, nghĩa là khi hàm start_dialog đang hoạt động, AutoLISP không cho phép gọi các hàm liên quan đến nhập xuất dữ liệu trên dòng lệnh command. Ví dụ, các hàm princ, print, prin1, getint,... sẽ không thực hiện khi hộp thoại đang mở.

Hàm ***done_dialog*** có tác dụng đóng hộp thoại. Cú pháp chung của hàm này như sau:

(done_dialog [status])

trong đó số nguyên dương *status* là giá trị muốn trả về của hàm start_dialog. Các giá trị 0, 1 được dùng mặc định khi người dùng chọn Cancel hoặc OK. Các giá trị khác do người dùng tự đặt nhằm mục đích riêng trong chương trình AutoLISP.

Hàm trả về tọa độ 2D của điểm góc hộp thoại trên màn hình AutoCAD.

Lưu ý rằng hàm này chỉ có thể gọi được trong biểu thức liên quan tới việc chọn tile (sử dụng như tham số của hàm action_tile).

Như đã trình bày ở phần trên, quá nhiều tile trên một hộp thoại sẽ làm người dùng khó định hướng nên chương trình sẽ mạch lạc hơn nếu thiết kế các hộp thoại nhiều cấp. Ví dụ, trong hộp thoại dùng để nhập dữ liệu vẽ bản vẽ chi tiết bánh vít, ngoài các thông số chính liên quan đến chi tiết cần vẽ, còn cần thêm các thông tin ở phần khung tên (người vẽ, thiết kế, kiểm, duyệt...) Việc nhập các thông số này nên trình bày trên một hộp thoại riêng, liên kết với một tile nào đó (chẳng hạn, có tên "Khung tên" với key "khungten") trong hộp thoại chính. Khi cần thay đổi các thông số này, người dùng sẽ chọn tile tương ứng, làm xuất hiện hộp thoại con và sau khi nhập xong sẽ quay lại hộp thoại chính. Cách đơn giản nhất là lập một hàm con (ví dụ, với tên "goi_khungten") chứa các hàm tương ứng theo 5 bước đã trình bày cho hộp thoại con, sau đó, ở phần hộp thoại chính, việc gọi hộp thoại con thực hiện qua hàm action_tile như sau:

(action_tile "khungten" "(goi_khungten)")

Phần dưới đây giới thiệu chi tiết hàm action_tile và một số hàm thao tác với tile trong hộp thoại DCL:

- Hàm ***action_tile*** gán các biểu thức cần thiết khi người dùng chọn một tile nào đó trong hộp thoại. Cú pháp chung của hàm như sau:

(action_tile key action-expr)

trong đó:

key - khoá tham chiếu đến tile trên hộp thoại (chuỗi ký tự được định nghĩa trong phần thuộc tính của tile trong ngôn ngữ DCL).

action-expr - chuỗi ký tự thể hiện các tác vụ xảy ra khi người dùng lựa chọn tile này.

Chẳng hạn, với hộp thoại DCL trên ví dụ 6.30, biểu thức *(action_tile "kichthuoc" "(setq size $value)")* sẽ trả về số chuỗi ký tự thể hiện thứ tự phần tử đã chọn trong hộp popup_list khi người dùng tác động vào tile này. Giá trị này được gán cho biến size. Thứ tự các phần tử trong danh sách được đánh số từ 0, vì vậy nếu người dùng chọn phần tử đầu tiên trong danh sách, biến size sẽ nhận giá trị là chuỗi ký tự "0". Còn biểu thức *(action_tile "l" "(setq chieudai (atof (get_tile \"l\")))")* sẽ thực hiện việc đọc giá trị trong hộp edit_box do người dùng vừa nhập (hàm get_tile), sau đó chuyển sang số thực và gán giá trị thu được cho biến *chieudai*. Lưu ý rằng tham số *action_expr* là một chuỗi ký tự, do vậy, khi cần thực hiện nhiều tác vụ với một tile, nếu viết chung thành một chuỗi sẽ rất khó xem vì có thể sẽ phải thêm rất nhiều các ký tự xổ chéo (\ ) vào trước các dấu ngoặc kép (") để thể hiện chúng. Trong trường hợp như vậy, việc tạo ra một hàm con sẽ làm cho biểu thức sáng sủa lên rất nhiều. Trong ví dụ 6.31, khi người dùng đã chọn kích thước vít, các thông số khác cần được tra bảng và cập nhật lại trên tile motachitiet. Cần nhiều biểu thức để thực hiện việc này, chúng được gom lại trong hàm con vit::trasolieu và đưa lời gọi hàm vào biểu thức thể hiện tác vụ.

Biểu thức *$value* được gọi là call-back. Các call-back hay được dùng gồm *$key* - trả về giá trị thuộc tính key của tile đã chọn, và *$value*, như đã trình bày trong ví dụ trên, trả về giá trị của tile. Giá trị trả về này là chuỗi ký tự, ý nghĩa của chúng tuỳ thuộc loại tile: với edit_box, chuỗi này thể hiện nội dung trong hộp text_box, với toogle, giá trị trả về là "0" hoặc "1", tuỳ theo tile này được kiểm hay không, còn với list_box hoặc popup_list, nếu người dùng đã chọn 1 phần tử, sẽ trả về thứ tự của nó ("0", "1",...), còn nếu không chọn phần tử nào, *$value* trả về giá trị NIL.

- ***Đặt lại giá trị cho tile hoặc đặc tính enable*** (cho phép hay không cho phép chọn tile) được thực hiện thông qua các hàm ***set_tile*** và ***mode_tile***. Các đặc tính này của tile trên hộp thoại có thể được đặt trước ngay khi thiết kế thông qua các thuộc tính *value* hoặc *is_enable* đã nói ở phần trên. Các hàm nói ở phần này thực hiện công việc tương tự khi chương trình đang làm việc.

+ Hàm ***mode_tile*** có cú pháp chung như sau:

(mode_tile key mode)

trong đó:

key - khoá tham chiếu tới tile muốn thay đổi đặc tính.

mode - số nguyên, thể hiện đặc tính mới của tile.

Ý nghĩa của giá trị *mode* như sau:

0 cho phép chọn tile (is_enable = true).

1 không cho phép chọn tile (tile bị mờ đi trên hộp thoại).

2 đặt focus tới tile này (tương tự như khi dùng Tab để di chuyển đến nó).

3 chọn nội dung văn bản trong hộp edit_box.

4 bật / tắt hình ảnh trên tile (image, image_button).

Hàm mode_tile luôn trả về giá trị NIL.

+ Hàm ***set_tile*** có cú pháp chung như sau:

(set_tile key value)

trong đó:

key - khoá tham chiếu tới tile muốn thay đổi đặc tính.

value - chuỗi ký tự, thể hiện giá trị mới của tile.

Hàm trả về giá trị *value* vừa đặt lại cho tile.

Thông thường hàm này hay được sử dụng để đưa các kết quả tính toán lên trên hộp thoại. Ví dụ, khi biết các thông số chính của bánh vít như môđun, số răng, hệ số dịch chỉnh, đường kính trục, qua đó tính toán được các thông số khác như các đường kính, thông số mayơ,... và đưa lên hộp thoại để người dùng tham khảo, nếu cần thiết thì sửa đổi trước khi thực hiện vẽ chi tiết.

• ***Lấy giá trị hiện thời của tile*** được thực hiện qua hàm ***get_tile*** với cú pháp chung như sau:

(get_tile key)

với *key* là khoá tham chiếu tới tile. Lưu ý phân biệt chữ hoa và chữ thường.

Giá trị trả về của hàm là một chuỗi ký tự thể hiện giá trị hiện thời của tile tương ứng.

• ***Các thao tác với danh sách trong pupup_list và list_box*** được thực hiện qua các hàm ***start_list, add_list*** và ***end_list***. Các thao tác này luôn bắt đầu bằng hàm start_list và kết thúc bằng end_list.

+ Hàm (***start_list key [operation [index]]***) thực hiện việc tạo mới, thêm hoặc sửa đổi danh sách, trong đó *key* là khoá tham chiếu tới tile (list_box hoặc popup_list), *operation* nhận một trong các giá trị 1, 2 hoặc 3 biểu thị mục đích sử dụng hàm, còn *index* là một số nguyên, chỉ được sử dụng khi cần thay đổi danh sách, thể hiện thành phần thứ mấy trong

danh sách cần thay đổi. Số thứ tự các thành phần trong danh sách được bắt đầu từ 0.

Ý nghĩa của tham số *operation* như sau:

1 thay đổi nội dung danh sách.
2 thêm thành phần mới vào danh sách.
3 tạo danh sách mới (mặc định).

Nếu danh sách đã tồn tại, khi tham số *operation* được gán giá trị 3 (hoặc bỏ qua), hàm sẽ thực hiện việc xoá danh sách cũ và tạo mới danh sách bằng việc thêm các thành phần qua hàm *add_list*. Hàm *start_list* trả về tên danh sách tham chiếu tới.

+ Hàm ***(add_list elem)*** sẽ thêm hoặc thay đổi một thành phần trong danh sách trả về bởi hàm *start_list*, trong đó tham số *elem* là chuỗi ký tự thể hiện thành phần mới của danh sách. Nếu công việc thành công, hàm trả về chuỗi ký tự vừa thêm vào, còn không, hàm trả về NIL.

+ Hàm ***(end_list)*** kết thúc các thao tác với danh sách popup_list hoặc list_box.

Sau đây là một số ví dụ:

```
;;; lấy các phần tử trong danh sách AutoLISP và đưa vào danh sách
;;; popup_list hoặc list_box có key là " longlist"
        (setq llist (list "first line" "second line" "third line"))
        (start_list "longlist")          ; tạo danh sách mới longlist
        (mapcar 'add_list llist)         ; hoặc (foreach str llist (add_list str))
        (end_list)                       ; hoàn tất công việc
;;; đoạn mã sau đây thay thế thành phần "second line" trong danh
;;; sách thành "2nd line"
        (start_list "longlist" 1 1)      ; thay đổi danh sách
        (add_list "2nd line")
        (end_list)
;;; thêm vào cuối danh sách 2 thành phần mới "line 4" và "line 5"
        (start_list "longlist" 2) ; thay đổi danh sách
        (add_list "line 4")
        (add_list "line 5")
        (end_list)
```

- ***Thao tác với các tile ảnh (image, image_button)*** được thực hiện qua các hàm sau đây:

+ Hàm ***(start_image key)*** bắt đầu các thao tác tile có thuộc tính *key* đã chỉ định. Sau đó để thực hiện việc chèn ảnh vào tile, cần gọi tiếp một trong các hàm *vector_image*, *fill_image* hoặc *slide_image*. Kết thúc các thao tác này bằng hàm *(end_image)*.

+ Hàm ***(vector_image x1 y1 x2 y2 color)*** vẽ một đoạn thẳng lên tile. Các tham số của hàm có ý nghĩa như sau:

x1, y1: tọa độ của điểm đầu đoạn thẳng

x2, y2: tọa độ điểm cuối đoạn thẳng

color: màu đoạn thẳng cần vẽ

Để xác định tọa độ các điểm cần biết kích thước của tile. Tọa độ (0,0) ứng với góc trái trên của tile. Tọa độ X, Y góc đối diện của tile cho bởi các hàm *(dimx_tile key)* và *(dimy_tile key)*.

Màu của đoạn thẳng là một số nguyên, tương ứng với số hiệu màu trong AutoCAD (1 = đỏ, 2 = vàng...) hoặc các giá trị sau:

-2 (BGLCOLOR) - màu màn hình AutoCAD hiện hành

-15 (DBGLCOLOR) - màu nền hộp thoại hiện hành

-16 (DFGLCOLOR) - màu văn bản trên hộp thoại hiện hành

-18 (LINELCOLOR) - màu đường viền hộp thoại hiện hành

Ví dụ, vẽ các đường chéo màu đỏ qua các đỉnh của tile "vecimage":

```
(setq color 1
      w (dimx_tile "vecimage")
      h (dimy_tile "vecimage")
      )
(start_image "vecimage")
(vector_image 0 0 w h color)
(vector_image 0 h w 0 color)
(end_image)
```

+ Hàm ***(fill_image x1 y1 width height color)*** tô màu có số hiệu *color* lên một vùng chữ nhật của tile, xác định bởi tọa độ góc trái trên của vùng (*x1*, *y1*) và độ lớn (*width* - chiều ngang, *height* - chiều cao).

Ví dụ, tô màu xanh cho toàn bộ tile "fillimage":

```
(setq color 1
      w (dimx_tile "fillimage")
      h (dimy_tile "fillimage")
      )
(start_image "fillimage")
(fill_image 0 0 w h color)
(end_image)
```

+ Hàm ***(slide_image x1 y1 width height sldname)*** gắn ảnh slide vào một vùng của tile. Vùng này được xác định như với hàm *fill_image*. Tham số *sldname* là chuỗi ký tự, thể hiện tên file slide. Nếu ảnh slide

được lưu trong file thư viện ảnh (*.slb), *sldname* được cho dưới dạng *libname(sldname)*.

File ảnh slide (có phần mở rộng "*.sld") được tạo ra từ các hình ảnh trên vùng làm việc của màn hình AutoCAD qua lệnh MSLIDE. Còn file thư viện ảnh (có phần mở rộng (*.slb) được tạo ra từ chương trình ứng dụng (DOS) SLIDELIB.EXE có trong folder gốc của AutoCAD theo trình tự sau:

- Sử dụng một trình soạn thảo văn bản ASCII soạn tên các file sld cần đưa vào thư viện. Nội dung file này có dạng:

```
entrance.sld
hall.sld
stairs.sld
study.sld
balcony.sld
```

- Lưu nội dung soạn thảo vào file với phần mở rộng (.txt), ví dụ, "home.txt".

- Chuyển sang môi trường DOS, chuyển tới folder gốc của AutoCAD.

- Gõ lệnh "SLIDELIB slbhome < home.txt" và nhấn Enter để tạo file thư viện ảnh với tên "slbhome.slb".

Cuối cùng, đoạn chương trình tổng hợp sau đây thực hiện các thao tác với hộp thoại trong ví dụ 6.30 (hình 6.8).

Ví dụ 6.31: *Điều khiển hộp thoại DCL*

```
(setq dcl_id (load_dialog "vidu6-30.dcl"))
(if (< dcl_id  0) (exit))   ; nếu không nạp được file thì dừng chương trình
(if (not (new_dialog "InputData" my_dcl)) (exit)) ; nếu không nạp được
                                  ; hộp thoại thì thoát chương trình, còn
                                  ; nếu thành công, sẽ tiếp tục...
(start_ima "hv_image" "vits")     ; hiện ảnh minh họa
(start_ima "mesys11" "mesys")     ; hiện lôgô
(setq CHC 2)                      ; gán giá trị 2 (lựa chọn mặc định) cho
                                        ; biến CHC (chọn hình chiếu)
(action_tile "hcbc" "(setq CHC 11)      ; nếu chọn kiểu 3, CHC = 11
        (mode_tile \"demvenh\" 1)       ; làm mất hiệu lực các tile
        (mode_tile \"mu\" 1)            ; đệm vênh, mũ, ren suốt.
        (mode_tile \"allren\" 1)")
(action_tile "hcbd" "(setq CHC 12)      ; nếu chọn kiểu 4, CHC = 12
        (mode_tile \"demvenh\" 1)       ; làm mất hiệu lực các tile
        (mode_tile \"mu\" 1)            ; đệm vênh, mũ, ren suốt.
        (mode_tile \"allren\" 1)")
```

```
(action_tile "hcd" "(setq CHC 2)          ; nếu chọn kiểu 2. CHC = 2
        (mode_tile \"demvenh\" 0)         ; cho phép các tile đệm vênh,
        (mode_tile \"mu\" 0)              ; mũ, ren suốt hoạt động.
        (mode_tile \"allren\" 0)")
(action_tile "hcc" "(setq CHC 1)          ; nếu chọn kiểu 1, CHC = 1
        (mode_tile \"demvenh\" 0)         ; cho phép các tile đệm vênh,
        (mode_tile \"mu\" 0)              ; mũ, ren suốt hoạt động.
        (mode_tile \"allren\" 0)")

;; nếu tác động vào các tile Chỉ vẽ mũ, Vẽ đường tâm,... thì cập nhật
;; giá trị trạng thái của tile, sau đó đổi sang số nguyên và gán cho các
;; biến tương ứng
(action_tile "mu" "(setq mu (atoi (get_tile \"mu\")))")
(action_tile "tam" "(setq tam (atoi (get_tile \"tam\")))")
(action_tile  "allren"  "(setq allren (atoi (get_tile \"allren\")))")
(action_tile "demvenh" "(setq demvenh (atoi (get_tile \"demvenh\")))")
(action_tile "group" "(setq group (atoi (get_tile \"group\")))")

;; nếu tác động vào tile Chiều dài... thì cập nhật giá trị vừa nhập
(action_tile "l" "(setq chieudai (atof (get_tile \"l\")))")

;; gọi hàm vit:kichthuoc để lấy các kích thước vít và tạo danh sách
;; chứa các kích thước vít tiêu chuẩn. Danh sách này sẽ được dùng cho
;; popup_list "kichthuoc". Nội dung hàm xem ở phần sau.
(vit::kichthuoc)
(setq size "0")              ; khởi tạo M3 như giá trị chọn mặc định...
                             ; gán cho biến size giá trị "0" - thành
                             ; phần đầu của popup_list
(vit::trasolieu size)        ; gọi hàm tra các số liệu khácủa vít
                             ; và hiện lên trên hộp list_box Mô tả chi tiết...

;; nếu tác động vào popup_list "kichthuoc", cập nhật lại biến size và
;; tra lại các số liệu khác của vít...
(action_tile  "kichthuoc"  "(setq size $value) (vit::trasolieu size)" )

;; nếu người dùng chọn các tile Vẽ, Không, Hỏi thêm...
(action_tile "accept" "(done_dialog) (vit::ve)")   ; thoát hộp thoại và
                                                   ; gọi hàm vẽ
(action_tile "cancel" "(setq CHC 0) (done_dialog)") ; thoát hộp thoại
(action_tile "help" "(itsme)")      ; gọi hàm itsme
(start_dialog)                ; hiện hộp thoại
(unload_dialog dcl_id)  ; giải phóng file DCL khỏi bộ nhớ
```

```
;; Các hàm khác...

(defun vit::kichthuoc( / alist)
        ;; mở file chứa dữ liệu vít (vit1.dat), đọc các dòng dữ liệu
        ;; tạo danh sách và lấy thành phần đầu tạo danh sách popup
        (setq (findfile "vit1.dat"))          ; tìm xem file có tồn tại
        (if f1
            (progn
                (setq mlist (mk_list f1)  ; đọc dữ liệu và tạo danh sách
                      alist (mk_key_list mlist 0)
                      )
                )
            )
        ; đưa các phần tử trong danh sách alist vào popup
        (start_list "kichthuoc")
        (mapcar 'add_list alist)
        (end_list)
        )          ; kết thúc hàm vit::kichthuoc


(defun vit::trasolieu (size / alist num d s tt lmin lmax c1 c2 c3 c4 c5)
        ;; truy cập tới bản ghi số liệu vít có đường kính đã định
        (setq num   (atoi size)
              alist (nth num mlist))
        ;; tra các thông số...
        (setq    d  (nth 1 alist)   ; đường kính danh nghĩa
                 s  (nth 2 alist)   ; kích thước S (cờ lê)
                 tt (nth 5 alist)   ; bước
                 lmin (nth 6 alist)         ; chiều dài tối thiểu
                 lmax (nth 7 alist)         ; chiều dài tối đa
                 )
        ;; chuyển các kích thước sang chuỗi ký tự
        (setq c1 (strcat "M    :" (rtos d 2 1)))
        (setq c2 (strcat "s    :" (rtos s 2 1)))
        (setq c4 (strcat "Lmin:  " (rtos lmin 2 0)))
        (setq c5 (strcat "Lmax:  " (rtos lmax 2 0)))
        (setq c3 (strcat "t    :" (rtos tt 2 1)))
        ;; đưa vào danh sách cho list_box
        (start_list "motachitiet")
        (add_list c1)
        (add_list c2)
        (add_list c4)
        (add_list c5)
        (add_list c3)
        (end_list)
    )      ;; kết thúc hàm vit::trasolieu
```

6.5. LIÊN KẾT VỚI CÁC CHƯƠNG TRÌNH KHÁC

Nhược điểm của ngôn ngữ AutoLISP là cách thức tổ chức chương trình lỏng lẻo và cách viết mã lệnh tương đối phức tạp. Vì vậy, nếu dùng AutoLISP viết các chương trình tính toán lớn sẽ rất khó và mất nhiều thời gian. Một giải pháp khắc phục là viết chương trình tính toán trên ngôn ngữ khác, lưu kết quả vào file và truyền kết quả này qua AutoLISP để thể hiện bản vẽ.

Từ AutoLISP có thể gọi một chương trình thực thi (đã dịch) qua hàm ***startapp*** với cú pháp chung như sau:

(startapp appcmd [file])

trong đó, *appcmd* - tên chương trình ứng dụng cần thực hiện.

file - tên file cần mở bởi chương trình *appcmd* (nếu có).

Ví dụ, để xem nội dung file "acad.lsp" bằng ứng dụng Notepad của Windows có thể gọi từ AutoLISP biểu thức sau:

(startapp "notepad" "acad.lsp")

Các phiên bản gần đây của AutoCAD cung cấp thêm một khả năng phát triển ứng dụng mới dành cho người sử dụng đã quen thuộc với VisualBasic. Đó là AutoCAD VBA (VisualBasic for Application), được nhúng ngay trong ứng dụng AutoCAD. Chương trình viết trong môi trường này được gọi là *VBA macro*, các macro này có thể đứng độc lập hoặc được gắn (embeded) vào bản vẽ AutoCAD.

Các macro thường được quản lý trong một *modul* nào đó của một đề án *(VBA Project)*. Macro có thể thực hiện từ AutoLISP thông qua hàm *command* với lệnh "-VBARUN" (phương án dòng lệnh của lệnh VBARUN). Để chạy một VBA macro, cần gọi biểu thức AutoLISP dưới dạng sau:

(command "-VBARUN" "Project1.Modul1.Macro1")

trong đó:

Project1 - tên đề án VBA chứa macro cần chạy.

Modul1 - tên modul chứa macro cần chạy.

Macro1 - tên macro cần chạy.

Ngược lại VBA cũng có thể gọi được hàm AutoLISP thông qua method *SendCommand*. Các tham số truyền qua method này tương tự như các dòng chương trình trong kiểu file đáp bản (script file), được sử dụng khá rộng rãi trước đây.

Ví dụ sau đây minh họa việc sử dụng VBA liên kết với chương trình

viết bằng AutoLISP. Các lệnh nạp và chạy chương trình được gửi qua dòng lệnh command của AutoCAD thông qua method *SendCommand* viết trong macro RPline. File AutoLISP có tên "RedirectPline.lsp" được soạn thảo trước và lưu trong folder của AutoCAD. Trong file này có chứa các hàm con C:RPline(), nhằm tạo một lệnh RPline thực hiện như các lệnh AutoCAD khác. Các ghi chú kèm theo ghi rõ mục đích từng lệnh trong macro.

```
Sub RPline()                    'Định nghĩa macro
Dim SendStr As String           'Khai báo biến cục bộ
    'Khai báo dòng (Load "RedirectPline.lsp") sẽ đưa sang AutoCAD
    SendStr="(Load "+String(1,34)+"RedirectPline.lsp"+String(1,34)+")"
    'Gửi dòng trên sang AutoCAD để nạp file AutoLISP có tên RedirectPline
    ThisDrawing.SendCommand SendStr & vbCr
    'Chạy hàm C:RPline định nghĩa trong file AutoLISP
    ThisDrawing.SendCommand "RPline" & vbCr
End Sub                         'Kết thúc macro..
```

Ngoài ra, nhiều ngôn ngữ lập trình khác cũng được sử dụng để phát triển các ứng dụng trong AutoCAD. Một trong những ngôn ngữ được sử dụng sớm nhất là C++, dùng tạo các ứng dụng ARX. Hiện nay, VC, Delphi, Java, VisualBasic và các VBA khác như MSWord VBA, MSExcel VBA,... cũng được sử dụng thông qua công nghệ AutoCAD ActiveX Technology. Điều này làm cho AutoCAD trở nên một ứng dụng mang tính mở rất cao.

Chương 7

THIẾT KẾ ỨNG DỤNG

Trên cơ sở mục tiêu, nội dung và phương pháp của thiết kế tự động, được đề cập trong các chương trước, chương này trình bày tóm tắt mục đích, nội dung và yêu cầu của thiết kế ứng dụng, quan hệ giữa công việc của mỗi thành viên trong nhóm thiết kế và nội dung được giao của cả nhóm, hướng dẫn viết thuyết minh khi thiết kế hoàn thành.

Bạn đọc có cơ hội chứng tỏ khả năng của mình khi biết ứng dụng một cách độc lập và sáng tạo những hiểu biết đã thu được để tiến hành tính toán thiết kế tự động một chi tiết máy cụ thể và tham gia thiết kế tự động các bộ phận máy được giao.

7.1. MỤC ĐÍCH, NỘI DUNG VÀ YÊU CẦU CỦA THIẾT KẾ ỨNG DỤNG

7.1.1. Mục đích

Thông qua công cụ tin học và nội dung thiết kế được giao, đồ án "Tự động hoá thiết kế cơ khí" nhằm mục đích:

1. Củng cố và nâng cao trình độ tính toán thiết kế các chi tiết máy và bộ phận máy có công dụng chung, bổ sung và hoàn thiện các kiến thức về kết cấu cơ khí đã được giới thiệu, phân tích, tổng hợp và đi sâu trong các chương trước.

2. Nâng cao một bước kỹ năng lập trình thông qua các vấn đề cụ thể cần đi sâu giải quyết khi xây dựng hoàn chỉnh các chương trình phần mềm được giao, nhằm thiết kế ra những chi tiết máy và bộ phận máy có hình dáng và kích thước hợp lý, đáp ứng các yêu cầu chủ yếu về kinh tế kỹ thuật.

3. Bồi dưỡng khả năng thiết kế tự động các bản vẽ kỹ thuật nhờ vận dụng thành thạo phần mềm AutoCAD và quan tâm đầy đủ đến các vấn đề cần lưu ý khi lập trình thiết kế tự động các bản vẽ chế tạo chi tiết, bản vẽ lắp các bộ phận máy và bản vẽ chung của toàn hệ thống.

4. Tìm hiểu để nắm vững các chương trình phần mềm về thiết kế tự động đã được soạn thảo, từ đó có thể khai thác chúng phục vụ các nhiệm vụ thiết kế ứng dụng được giao.

Để đạt được mục đích trên đây, một mặt bạn đọc cần nắm vững và ứng dụng có kết quả các ngôn ngữ lập trình bậc cao cũng như ngôn ngữ AutoLISP và phần mềm AutoCAD trong soạn thảo phần mềm tính toán thiết kế cũng như phần mềm thiết kế tự động bản vẽ, mặt khác cần tìm hiểu kỹ các chỉ tiêu tính toán thiết kế chi tiết máy và bộ phận máy có công dụng chung cũng như các đặc điểm về nội dung và phương pháp lập trình thiết kế cơ khí, đã được trình bày trong các chương từ 1 đến 6.

7.1.2. Nội dung

Xuất phát từ các mục tiêu trên đây và với thời lượng của một đồ án thiết kế ứng dụng là hai học trình, nội dung của đồ án "Tự động hoá thiết kế cơ khí" bao gồm hai phần chính sau đây:

1. Xây dựng phần mềm tính toán thiết kế hoàn chỉnh cho một trong các bộ truyền cơ khí: bộ truyền bánh răng trụ hoặc bánh răng côn, bộ truyền trục vít – bánh vít, bộ truyền đai, bộ truyền xích hoặc thiết kế trục và chọn ổ lăn, bao gồm các nội dung sau đây:

a) Xây dựng cơ sở dữ liệu cho chương trình phần mềm sẽ được soạn thảo, gồm các bảng tra cơ tính của vật liệu, các giá trị của ứng suất giới hạn và công suất cho phép, các giá trị tiêu chuẩn của các thông số và kích thước chi tiết máy, các giá trị của các hệ số dùng trong tính toán cũng như các bảng tra dùng cho thiết kế bản vẽ như sai lệch giới hạn và dung sai, trị số nhám, v.v...

b) Lập trình tính toán thiết kế một chi tiết máy được chỉ định nhằm xác định các thông số và kích thước của nó. Các thông số và kích thước này một mặt phải thoả mãn các chỉ tiêu chủ yếu về khả năng làm việc của chi tiết máy, mặt khác phải được lựa chọn tối ưu, đáp ứng các yêu cầu: kết cấu hợp lý, khối lượng nhỏ, dễ chế tạo, trong điều kiện sản suất hiện có ít tốn nguyên vật liệu và thời gian.

c) Lập trình thiết kế tự động bản vẽ chế tạo chi tiết máy như: bánh răng trụ hoặc bánh răng côn, trục vít, bánh vít, các loại bánh đai, đĩa xích và các loại trục với dữ liệu thiết kế là các thông số và kích thước thu được khi chạy chương trình phần mềm thiết kế chi tiết máy đã được soạn thảo ở trên.

2. Tham gia vào một nhóm thiết kế gồm từ 5 đến 7 người để cùng thiết kế tự động một hệ dẫn động cơ khí, bao gồm các nội dung:

a) Trên cơ sở chọn sơ đồ động thích hợp cho hệ, tiến hành lập trình tính toán chọn động cơ điện có công suất và số vòng quay phù hợp để cung cấp năng lượng cho hệ dẫn động đã chọn, đảm bảo cho thiết bị có kết cấu và kích thước gọn nhẹ, làm việc lâu dài và ổn định, đáp ứng các chỉ tiêu kinh tế kỹ thuật.

b) Tính toán phân phối tỉ số truyền của hệ cho các cấp truyền động, xuất phát từ các chỉ tiêu: kích thước và khối lượng nhỏ, hiệu suất cao, thuận tiện cho việc bôi trơn các bộ truyền bánh răng trong hộp (xem các tài liệu [2], [8], [9], [10] và [11]).

Các thông số và kích thước động cơ cũng như các giá trị của công suất, số vòng quay, tỉ số truyền và mômen xoắn trên các trục cần được ghi vào một *File* kết quả phục vụ các bước tính tiếp theo cũng như để thiết kế bản vẽ chung của hệ.

c) Lập trình tính toán thiết kế các bộ truyền cơ khí trong hệ theo một trình tự hợp lý, tuỳ thuộc vào loại sơ đồ động và loại truyền động nhằm thu được các thông số và kích thước tối ưu cho các bộ truyền và chi tiết quay (bánh răng, bánh đai, trục vít, bánh vít, đĩa xích) cũng như giảm thiểu sai số của số vòng quay của trục công tác hoặc tỉ số truyền của hệ (xem chương 4, mục 4.2.8). Các thông số và kích thước của các bộ truyền được ghi vào một *File* kết quả phục vụ các bước tính tiếp theo cũng như để thiết kế bản vẽ lắp và bản vẽ chung của hệ.

d) Lập trình tính toán thiết kế trục và chọn ổ lăn cho các trục và các gối đỡ trục trong hệ (xem chương 3). Riêng phần thiết kế trục, ngoài phần tính toán còn sử dụng đồ họa để vẽ các biểu đồ mômen uốn và mômen xoắn phục vụ việc thiết kế kết cấu trục, đảm bảo sự tương thích về vị trí của các trục với các chi tiết lắp trên trục và các gối đỡ của các trục trong hệ. Các thông số và kích thước của các trục và ổ lăn được ghi vào một *File* kết quả phục vụ các bước tính tiếp theo cũng như để thiết kế bản vẽ lắp và bản vẽ chung của hệ.

e) Từ các *File* kết quả thu được ở các phần thiết kế trước, tiến hành tính toán bổ sung các số liệu về kích thước các bộ phận của hộp giảm tốc và các chi tiết lắp với hộp, trên cơ sở đó tiến hành lập trình vẽ tự động kết cấu của các chi tiết và cụm chi tiết của bản vẽ lắp trên ba hình chiếu chính và khi cần thiết có thêm các hình cắt trích. Các thông số và kích thước của hộp giảm tốc được ghi vào một *File* kết quả phục vụ tính toán và thiết kế tự động bản vẽ chung của hệ.

g) Tính toán bổ sung các số liệu về kích thước của các khớp nối, khung và bệ máy, v.v... và tiến hành lập trình thiết kế tự động bản vẽ chung của toàn hệ thống.

Các nội dung 1 và 2 có liên quan chặt chẽ với nhau. Người được giao lập trình tính toán thiết kế hoàn chỉnh một chi tiết máy xác định, bao gồm cả phần tính toán và phần thiết kế bản vẽ chế tạo chi tiết, khi tham gia nhóm thiết kế hệ dẫn động, sẽ được đảm nhiệm phần thiết kế kết cấu chi tiết đó của hệ. Sự khác nhau ở đây chỉ là khi thực hiện nội dung thứ nhất, bạn có thể tuỳ ý lựa chọn ngôn ngữ lập trình để sử dụng, tuỳ ý ký hiệu

các thông số, kích thước và tọa độ trong tính toán cũng như tuỳ ý lựa chọn màu sắc và đường nét trên các bản vẽ. Tuy nhiên khi thực hiện nội dung thứ hai, công việc chỉ có thể được hoàn thành thuận lợi nếu mỗi thành viên trong nhóm tuân thủ những nguyên tắc sau đây:

- Sử dụng thống nhất một ngôn ngữ lập trình để tiến hành tính toán thiết kế tất cả các chi tiết máy và bộ phận máy của toàn hệ thống.

- Sử dụng ký hiệu thống nhất cho tất cả các biến biểu thị kích thước và thông số của các chi tiết máy và bộ phận máy trong hệ cũng như các biến xác định tọa độ vị trí của từng yếu tố trong hệ. Nhờ đó có thể tránh được nhầm lẫn về kết quả tính toán cũng như tránh được sai lệch về vị trí tương quan trên các bản vẽ.

- Nắm vững và hoàn thành tốt nội dung được giao trong nhóm thiết kế, đồng thời cần tìm hiểu để nắm được nội dung và phương pháp hoàn thành các phần việc khác của bản thiết kế do nhóm tiến hành. Có như vậy mới có sự phối hợp tốt nhất nhằm đảm bảo chất lượng và tiến độ thực hiện các nội dung thiết kế tự động của toàn nhóm.

7.1.3. Yêu cầu

Các chương trình phần mềm được soạn thảo phải đảm bảo các yêu cầu sau đây:

1. Chương trình phải có cấu trúc rõ ràng, thể hiện đúng lưu đồ thuật giải đã được vạch ra, phù hợp với các bước tính toán thiết kế từng chi tiết máy cụ thể. Các thông số và kích thước của chi tiết máy phải được xác định trên cơ sở các chỉ tiêu chủ yếu về khả năng làm việc của chi tiết máy và bộ phận máy, đồng thời đáp ứng yêu cầu tối ưu trong thiết kế.

2. Chương trình phải đảm bảo *độ chính xác* của các số liệu đầu ra, tức là có khả năng kiểm chứng sự đúng đắn của các công thức được sử dụng trong lập trình tính toán độ bền và xác định các quan hệ động học. Mặt khác, các kết quả tính toán chỉ có thể *chính xác* nếu các số liệu và hệ số dùng trong tính toán được tra một cách chính xác, ứng với những điều kiện tra bảng khác nhau.

3. Chương trình phải có *tính tổng quát*, tức là phần mềm đã được soạn thảo có thể sử dụng để thiết kế các chi tiết máy và bộ phận máy ứng với một phạm vi đủ rộng của các dữ liệu đầu vào cũng như của cơ tính vật liệu được chọn để thiết kế. Yêu cầu này không dễ thực hiện, nhất là đối với phần thiết kế tự động các bản vẽ, do khuôn khổ kích thước của chúng phụ thuộc rất nhiều vào công suất tính toán, trong khi đó có nhiều số liệu lại được chọn theo kinh nghiệm và chỉ đúng trong một phạm vi hẹp. Vì vậy trước hết người lập trình cần nạp vào *File* một bộ dữ liệu xác định để tiến hành chạy thử toàn bộ chương trình đang được soạn thảo, sau đó mới

mở rộng dần các dữ liệu thiết kế và tiếp tục kiểm chứng nhằm loại trừ các sai sót có thể phát sinh.

4. Chương trình phải có *tính hiệu quả*, tức là có khả năng liên kết được các phần tính toán với phần thiết kế bản vẽ, nhờ đó có thể đưa ra những thay đổi và hiệu chỉnh cần thiết trước khi kết cấu được quyết định lần cuối. Tính hiệu quả còn thể hiện ở chỗ trong phần mềm có nhiều giải pháp thiết kế, nhờ đó có thể đánh giá, so sánh và lựa chọn được phương án tốt nhất để dùng. Mặt khác hiệu quả của chương trình sẽ được tăng thêm nếu có sự phối hợp tốt giữa phần đối thoại giữa người và máy với phần máy tính thực hiện tự động theo những đoạn chương trình được soạn thảo nhằm đạt được *tốc độ tính toán nhanh, tiết kiệm tối đa thời gian thiết kế*, đồng thời đảm bảo khả năng kiểm tra các kết quả tính toán khi cần thiết.

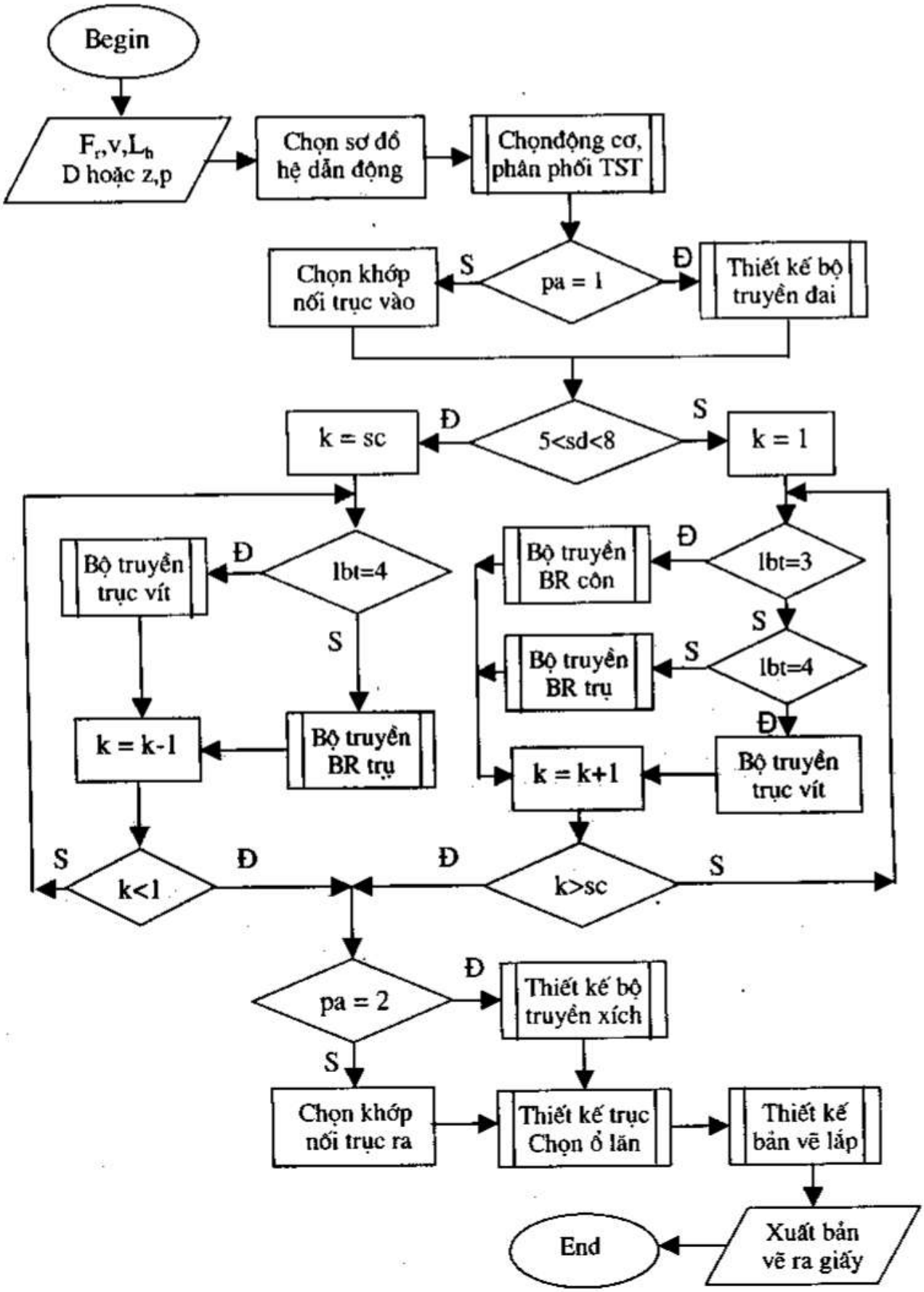
5. Các thông số và kích thước của sản phẩm, được thiết kế theo chương trình phần mềm, phải được *lựa chọn tối ưu* trên cơ sở sử dụng nhiều phương án và giải pháp thiết kế cũng như dựa trên các tiêu chí và căn cứ xác đáng để có thể quyết định đúng đắn phương án sản phẩm cùng với kết cấu và kích thước chi tiết, đáp ứng đầy đủ các chỉ tiêu kinh tế kỹ thuật.

6. Chương trình thiết kế tự động phải đảm bảo cho bản vẽ được thực hiện theo một trình tự hợp lý, tránh xảy ra sự chồng chéo của các nét, nhằm thể hiện rõ ràng và chính xác các kết cấu cơ khí phức tạp. Đồng thời phải có sự phân biệt rõ ràng về đường nét, mặt cắt trên các hình chiếu chính và hình cắt trích cũng như có sự khác biệt về độ cao và chiều dày nét chữ thể hiện số thứ tự chi tiết, các kích thước liên kết, kích thước khuôn khổ và kích thước lắp ghép hoặc dung sai.

Để thực hiện đầy đủ các nội dung và yêu cầu trên đây, một mặt người thiết kế cần nghiên cứu nắm vững và vận dụng một cách sáng tạo các kiến thức về tin học và kết cấu cơ khí để hoàn thành đúng tiến độ phần mềm thiết kế tự động của bản thân, mặt khác cần chủ động phối hợp với các thành viên trong nhóm hoàn thành đúng hạn phần việc được giao, cùng cả nhóm thực hiện tốt nhiệm vụ thiết kế.

7.2. KHAI THÁC PHẦN MỀM HEDD

Nếu quỹ thời gian bạn dùng cho thiết kế ứng dụng bị hạn chế (khoảng 100 đến 150 giờ), người thiết kế khó có thể hoàn thành một khối lượng khá lớn của đồ án "Tự động hoá thiết kế cơ khí". Trong trường hợp này, bạn đọc có thể được trợ giúp một phần nhờ tìm hiểu và khai thác có hiệu quả phần mềm HEDD được xây dựng theo lưu đồ thuật giải trên hình 7.1 (ở đó TST viết tắt cho tỉ số truyền) và được giới thiệu và chú thích trong phần phụ lục (xem phụ lục 2).



Hình 7..1. Lưu đồ thuật giải thiết kế hệ dẫn động cơ khí

7.3. VIẾT THUYẾT MINH

7.3.1. Nội dung viết thuyết minh

Thuyết minh là một phần tài liệu không thể thiếu trong hồ sơ thiết kế của bất kỳ một đề tài thiết kế nào. Viết thuyết minh là phần việc cuối cùng của quá trình làm đồ án môn học chuyên ngành "Tự động hoá thiết kế cơ khí".

Ngoài phần nhiệm vụ thiết kế ở trang đầu và phần mục lục ở trang cuối, thuyết minh cần bao gồm những nội dung sau đây:

1. Xây dựng cơ sở dữ liệu trong thiết kế cơ khí:

- Các loại dữ liệu.
- Các phương pháp tạo *File* dữ liệu và *File* kết quả.
- Vào, ra và kiểm tra các số liệu được ghi vào bảng.
- Nhận xét.
- In bảng số liệu đã được ghi vào bảng.

2. Xây dựng chương trình tính toán thiết kế:

- Nội dung thiết kế được giao.
- Chỉ tiêu tính toán và các bước thiết kế.
- Lưu đồ thuật giải và một số thuật giải được sử dụng trong chương trình.
- Kết quả tính toán với phần mềm đã được soạn thảo.
- Phương pháp xây dựng phần mềm thiết kế tự động bản vẽ chế tạo.
- In bản vẽ chế tạo trên khổ giấy A4.
- Các biện pháp đã được sử dụng để khắc phục các khó khăn xuất hiện khi soạn thảo phần mềm và nhận xét.

3. Lập trình thiết kế tự động bản vẽ lắp cụm kết cấu:

- Nhiệm vụ được giao trong nhóm thiết kế hệ dẫn động cơ khí.
- Các khó khăn và biện pháp đã giải quyết khi khai thác phần mềm HEDD để tính toán thiết kế các chi tiết máy trong hệ dẫn động cơ khí.
- Phương pháp xây dựng phần mềm thiết kế tự động bản vẽ lắp cụm kết cấu cơ khí.
- In bản vẽ lắp cụm kết cấu của từng thành viên trong nhóm trên khổ giấy A4.

- Các biện pháp đã được sử dụng nhằm khắc phục khó khăn khi kết hợp các thành viên trong nhóm để soạn thảo phần mềm thiết kế hệ dẫn động cơ khí.
- Nhận xét chung.
- In bản vẽ lắp của toàn nhóm trên khổ giấy A0.

4. Tài liệu tham khảo.

5. Một phần thư viện chương trình.

7.3.2. Yêu cầu của bản thuyết minh

Các nội dung trên đây của bản thuyết minh cần được hoàn thành theo đúng các yêu cầu sau:

1. Thuyết minh được đánh máy vi tính trên khổ giáy A4, được trình bày ngắn gọn, bảng số và hình vẽ rõ ràng.

2. Nội dung cần được trình bày súc tích, các nhận xét, phân tích và so sánh cần rõ ràng và có căn cứ, dựa trên những kiến thức về kết cấu cơ khí và tin học đã thu nhận được trong quá trình làm đồ án. Trích dẫn tài liệu tham khảo cần chính xác, số thứ tự của tài liệu tham khảo cần được đặt trong ngoặc vuông.

3. Phần phụ lục về thư viện chương trình được in ở phần cuối thuyết minh với khổ chữ thu nhỏ.

4. Thuyết minh bao gồm phần chính, các bản vẽ và phần phụ lục cần được đóng bằng bìa cứng và nạp đúng hạn.

PHỤ LỤC 1

CHƯƠNG TRÌNH AUTOLISP VẼ BÁNH VÍT

Thiết lập bản vẽ chế tạo chi tiết là một phần quan trọng trong hồ sơ thiết kế. Khác với bản vẽ lắp, trên bản vẽ chi tiết cần ghi đầy đủ các kích thước, các thông số và yêu cầu kỹ thuật, v..v, phục vụ việc chế tạo chi tiết. Công việc này đòi hỏi phải tính toán, lựa chọn trước kích thước kết cấu và các số liệu liên quan. Chẳng hạn, với bánh răng, bánh vít,..., ngoài các thông số cho trước như môđun, số răng, chiều rộng vành răng, đường kính trục,... cần phải lựa chọn các kích thước khác như đường kính mayơ, chiều dày vành răng,... và cũng cần kiểm tra các điều kiện khác như có đủ kích thước để lắp rời hay phải làm liền trục,...

Tuy nhiên, do các kết cấu này tương đối thống nhất nên có thể lập các chương trình máy tính để tự động thực hiện công việc này. Người dùng chỉ cần nhập các thông số cơ bản đầu vào của chi tiết, việc kiểm tra điều kiện kết cấu, tính toán kích thước được chương trình thực hiện và đưa kết quả ra, qua đó người dùng có thể sửa đổi lại theo ý mình và sau đó chương trình sẽ thực hiện vẽ, điền thêm các yêu cầu kỹ thuật, khung tên,...

Phụ lục này giới thiệu chương trình thiết lập bản vẽ chế tạo chi tiết bánh vít. Chương trình được viết bằng ngôn ngữ AutoLISP và DCL. Giao diện hộp thoại của chương trình như trên hình PL1.1.

Sau khi nạp chương trình, AutoLISP tạo ra một lệnh BVT trong AutoCAD. Sau đó, mỗi khi muốn vẽ chi tiết bánh vít người dùng chỉ cần gõ vào dòng lệnh command của AutoCAD câu lệnh BVT và nhấn Enter.

Trên hộp thoại, phía góc trái trên là hình minh họa kết quả sau khi chạy chương trình và qua đó người dùng có thể thấy sơ bộ kết cấu.

Phía góc phải trên là ***các lựa chọn chính (các button)***, công dụng của chúng như sau:

+ Nút ***Đọc DL*** - đọc dữ liệu đã lưu trước vào file. Khi kích chuột vào nút này, một hộp thoại tương tự như hộp thoại chuẩn *SelectFile* của AutoCAD hiện ra cho phép chọn file dữ liệu cần mở (hình PL1.2). Dữ liệu đọc từ file được đưa lên các hộp văn bản (edit_box) để người dùng có thể thay đổi nếu thấy cần thiết.

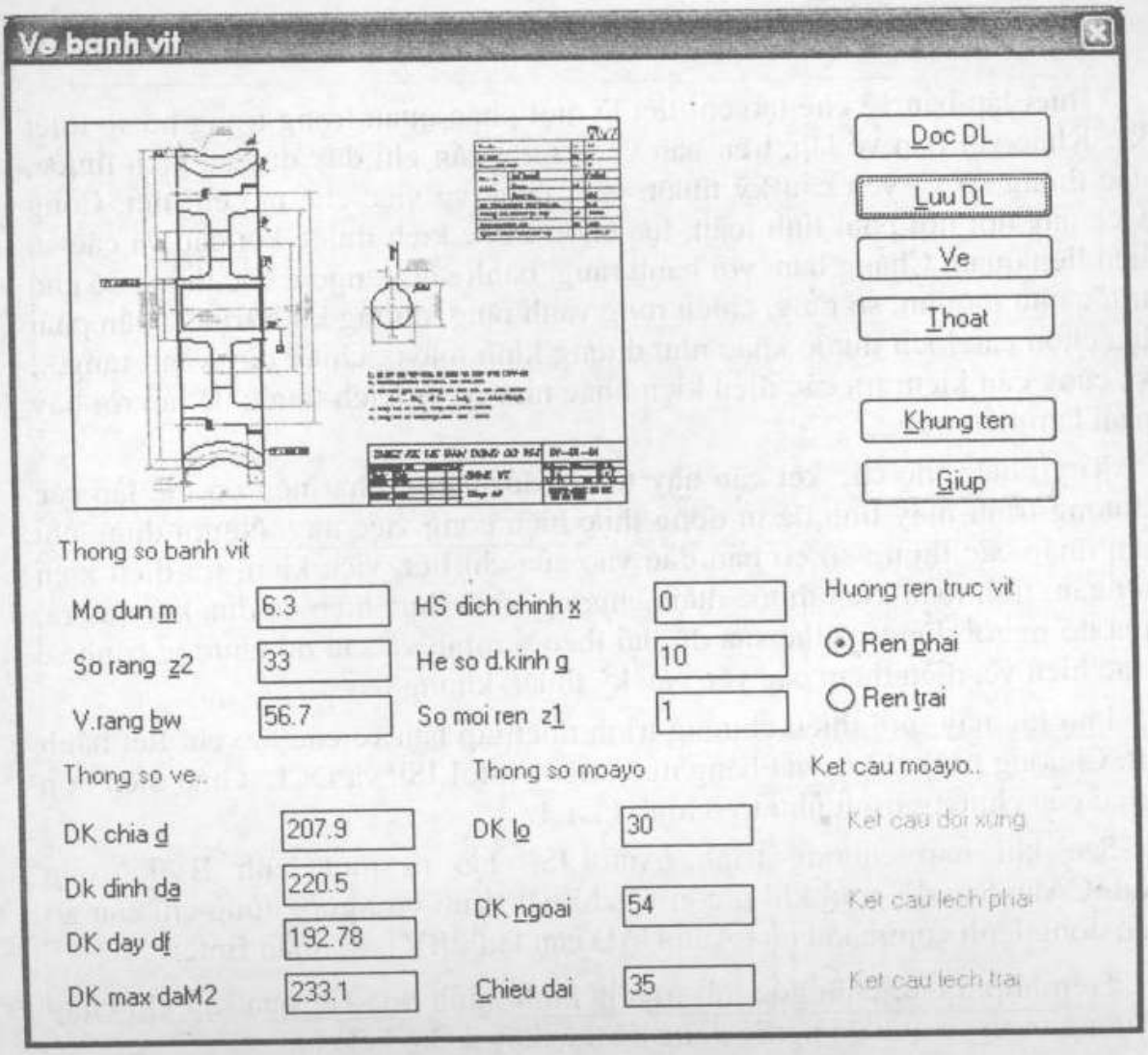
+ Nút ***Lưu DL*** - lưu dữ liệu trên hộp thoại vào file. Khi kích chuột vào nút này, một hộp thoại tương tự như hộp thoại chuẩn *SaveAs* của AutoCAD hiện ra cho phép chọn folder và file để lưu dữ liệu vào.

Lưu ý rằng các file dữ liệu ở đây chỉ chứa các thông tin trên hộp thoại, chứ không phải file bản vẽ nên kích thước của chúng rất bé. Khi cần thiết, người dùng chỉ việc gọi file này và kích vào nút *Ve* để tạo bản vẽ tương ứng.

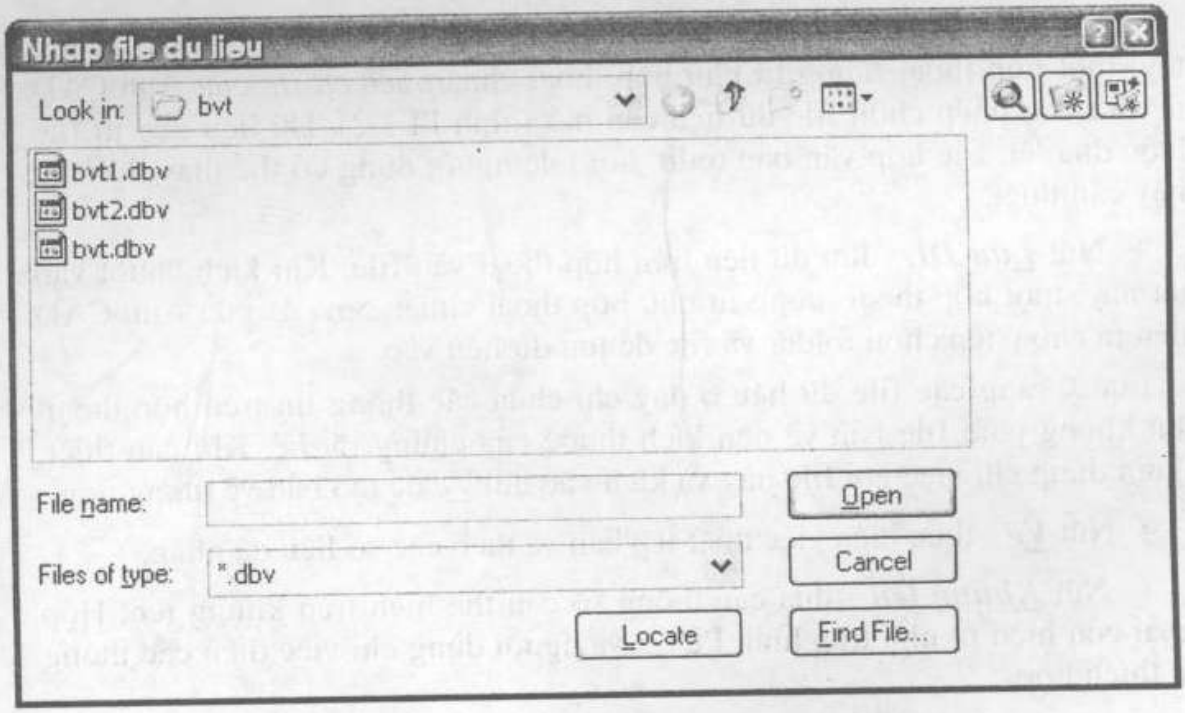
+ Nút ***Vẽ*** - thực hiện việc thiết lập bản vẽ theo các số liệu đã nhập.

+ Nút ***Khung tên*** - đưa các thông số cần thể hiện trên khung tên. Hộp thoại con hiện ra như trên hình PL1.3 và người dùng chỉ việc điền các thông tin thích hợp.

+ Nút ***Giúp*** - hiện một vài trợ giúp như giá trị mô đun tiêu chuẩn, hệ số đường kính q... hoặc cách nhập dữ liệu trong các ô.

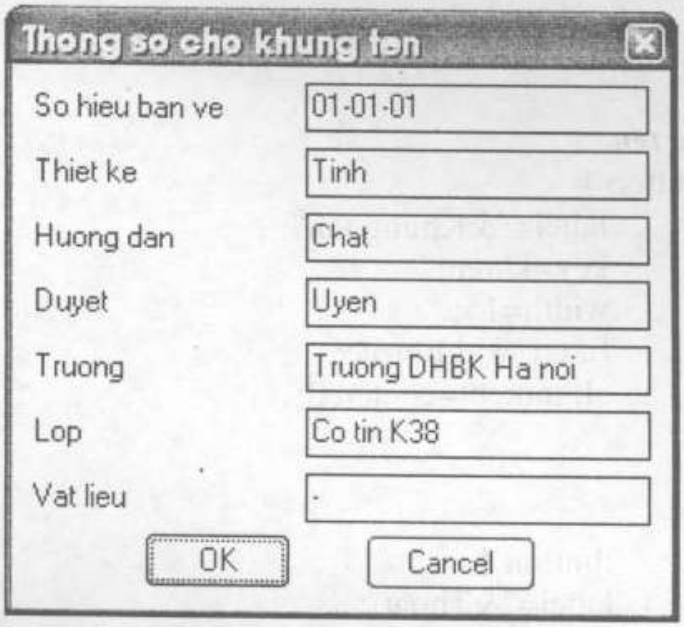


Hình PL1.1. Giao diện chính chương trình vẽ bánh vít



Hình PL1.2. Hộp thoại chọn file dữ liệu

Bạn đọc có thể thay thế phần mã nguồn để hiện các thông số tiêu chuẩn dưới dạng danh sách popup_list. Cách thể hiện này sẽ giúp người dùng không nhập dữ liệu phi tiêu chuẩn.



Hình PL1.3. Nhập thông số khung tên

Phần ***Thông số bánh vít*** là các thông số chính của bánh vít do người dùng nhập (bắt buộc), có thể từ file hoặc trực tiếp từ bàn phím. Ngoài ra, người dùng cần nhập thêm đường kính trục lắp bánh vít.

Các thông số khác do chương trình tính ra sẽ hiện ra trong các hộp edit_box ở phần ***Thông số vẽ...*** và phần ***Thông số mayơ***. Nếu muốn, người dùng có thể kích chuột vào các ô này để chỉnh lại theo ý mình.

Do bản vẽ chế tạo các chi tiết loại này có nhiều điểm chung như phần khung tên, các yêu cầu kỹ thuật, bảng thông số,... Chỉ số liệu ghi vào là khác nhau, còn cấu trúc của chúng thống nhất. Vì vậy, trong ví dụ ở phụ lục này, các thành phần này được chèn vào bằng cách sử dụng khối (block) tạo sẵn với các thuộc tính (attributes) thay đổi. Phần lỗ mayơ cũng như nhau, vì thế nên lập một chương trình con, dùng chung cho cả bánh răng, bánh vít, đĩa xích,... Việc ghi kích thước, độ nhám,... cũng được viết riêng thành các chương trình con.

Sau khi thực hiện chương trình, kết quả thu được bản vẽ chế tạo sẽ như minh họa trên hình PL1.4. Còn dưới đây là mã nguồn của chương trình.

**** Phần định nghĩa hộp thoại DCL (file Chitiet.DCL)***

```
//
// Các tiles hay sử dụng..
//

// Ảnh lôgô
mesys1      :image{
            key="mss";
            width=8;
```

```
                height=2;
                alignment=right;
                fixed_width=true;
                color=dialog_background;
        }

    // Khung tên
    khten :button {
                label="&Khung ten";
                key=khten;
                width=15;
                fixed_width=true;
                alignment=centered;
    }

    // Thoát
    thoat       :button {
                label="&Thoat";
                key="thoat";
                width=15;
                fixed_width=true;
                is_cancel=true;
                alignment=centered;
    }

    // Giúp
    giup :button {
                label="&Giup";
                key="giup";
                width=15;
                fixed_width=true;
                alignment=centered;
    }

    // Vẽ
    ve1  :button {
                label="&Ve";
                key="ve1";
                width=15;
                fixed_width=true;
                alignment=centered;
    }

    // Hình minh họa
    image1      :image {
                key="image1";
                height=12;
                width=50;
                color=graphics_background;
    }
```

```
// Đọc dữ liệu từ file
file_dl  :button {
            label="&Doc DL";
            key="file";
            width=15;
            fixed_width=true;
            alignment=centered;
}

// Lưu dữ liệu vào file
luu_dl  :button {
            label="&Luu DL";
            key="luu";
            width=15;
            fixed_width=true;
            is_default=true;
            alignment=centered;
}

//
// Hộp thoại Vẽ bánh vít
//
ve_bvt : dialog {
      label="Ve banh vit";
     :row {    //row1
        :boxed_column {
            image1;
            }

       :boxed_column {
            file_dl;
            luu_dl;
            ve1;
            thoat;
            :spacer {height=1;}
            khten;
            giup;
            }
         }
       :boxed_row {
            label="Thong so banh vit";
            :column {
                  :edit_box {
                        label="  Modun &m";
                        key="m";
                        edit_width=8;
                        }
                  :edit_box {
                        label="  So rang  &z2";
```

```
                    key="z";
                    edit_width=8;
                    }
                :edit_box {
                    label="  V.rang &bw";
                    key="bw";
                    edit_width=8;
                    }
                }
            :column{
                :edit_box {
                    label=" HS dich chinh &x";
                    key="xx";
                    edit_width=8;
                    }
                :edit_box {
                    label=" Hc so d.kinh &q";
                    key="q";
                    edit_width=8;
                    }
                :edit_box {
                    label=" So moi ren  z&1";
                    key="dy";
                    edit_width=8;
                    }

                }
            :boxed_column {
                label="Huong ren truc vit";
                :radio_button {
                    label="Ren &phai";
                    key="hg1";
                    value=1;
                    }
                :radio_button {
                    label="Ren &trai";
                    key="hg2";
                    }
                }
            } //row

    :row {
       :boxed_column{
            label="Thong so ve..";
            :edit_box {
                label="DK chia &d";
                key="d";
                edit_width=8;
                }
```

```
                :edit_box {
                        label="Dk dinh d&a";
                        key="da";
                        edit_width=8;
                        }
                :edit_box {
                        label="DK day d&f";
                        key="df";
                        edit_width=8;
                        }
                :edit_box {
                        label="DK max daM2";
                        key="dm";
                        edit_width=8;
                        }
                }
                tsmyo1;
                tsmyo2;
        }//row..
} // hết hộp thoại chính

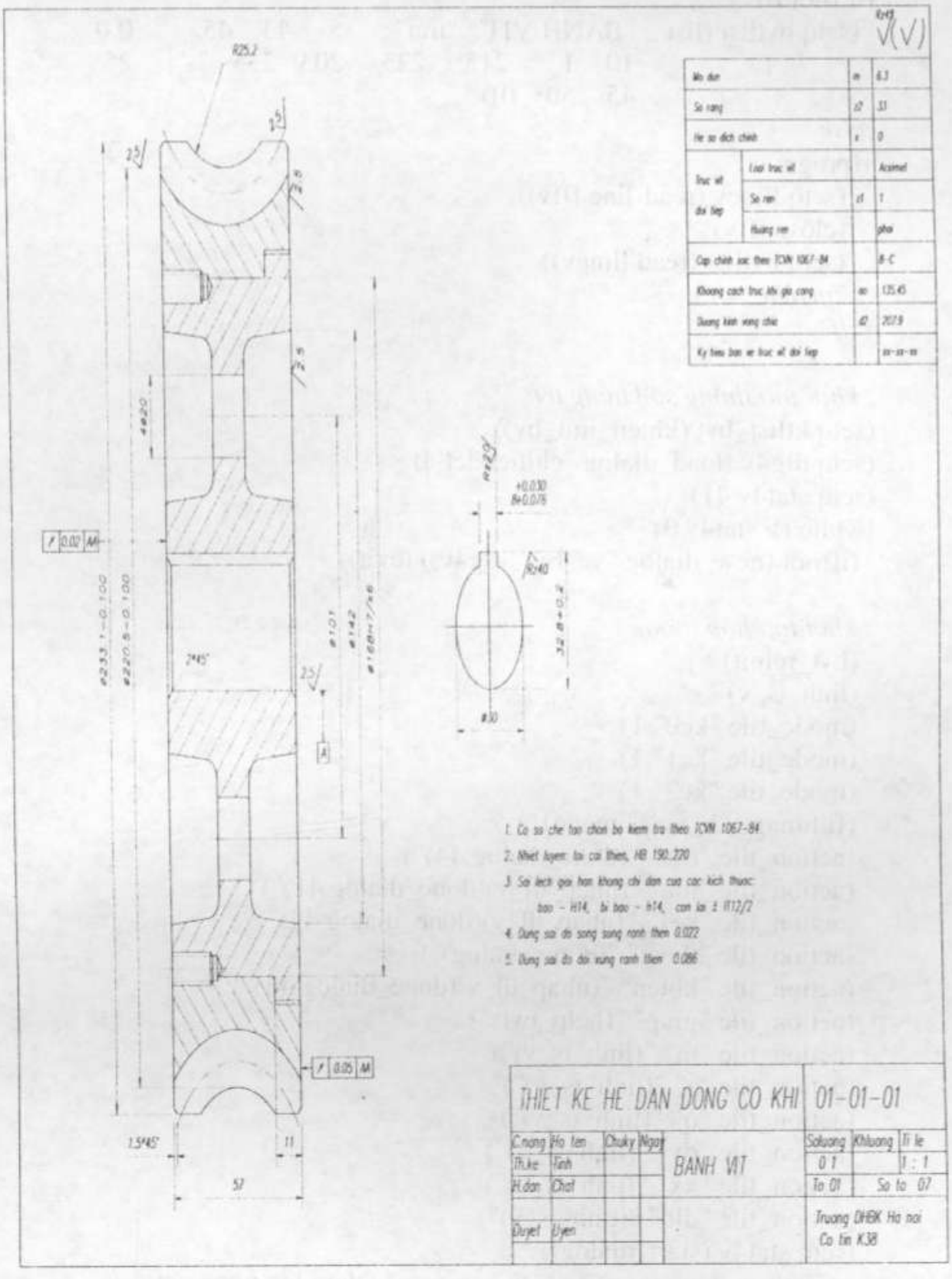
//
// Subassembly Thông số mayo
//
tsmyo1      :boxed_column {
                        label="Thong so moayo";
                        :edit_box {
                                label="DK l&o";
                                key="dlo";
                                edit_width=8;
                                }
                        :edit_box {
                                label="DK &ngoai";
                                key="dmyo";
                                edit_width=8;
                                }
                        :edit_box {
                                label="&Chieu dai";
                                key="lmyo";
                                edit_width=8;
                                }
                }

tsmyo2      :boxed_column{
                        label="Ket cau moayo..";
                        :radio_button {
                                label="Ket cau doi xung";
                                key="kc0";
                                value=1;
                                }
```

```
                :radio_button {
                        label="Ket cau lech phai";
                        key="kc1";
                        }
                :radio_button {
                        label="Ket cau lech trai";
                        key="kc2";
                        }
        }

//
// Hộp thoại Khung tên
//
khungten :dialog {
    label="Thong so cho khung ten";
     :edit_box {
            label="So hieu ban ve";
            key="sohieu";
            edit_width=20;
            }
     :edit_box {
            label="Thiet ke";
            key="thietke";
            edit_width=20;
            }
     :edit_box {
            label="Huong dan";
            key="huongdan";
            edit_width=20;
            }
     :edit_box {
            label="Duyet";
            key="duyet";
            edit_width=20;
            }
     :edit_box {
            label="Truong";
            key="truong";
            edit_width=20;
            }
     :edit_box {
            label="Lop";
            key="lop";
            edit_width=20;
            }
     :edit_box {
            label="Vat lieu";
            key="vatlieu";
            edit_width=20;
            }
```

```
:row {
        ok_cancel;
        }
} // kết thúc hộp thoại khung tên
```



Hình PLI.4. Bản vẽ chế tạo bánh vít

*** *Phần mã nguồn AutoLISP***

```
(defun c:bvt (/ osmod1 dlg4v stat4v ktlist_bvt bvtlist fffv llinev)

 ; khởi tạo thông số mặc định
 (setq osmod1 (getvar "osmode"))
 (setvar "osmode" 0)
 (setq fffv (open "c:/hedd/dwgdat/bvt.dat" "r"))
 (if (not fffv)
  (setq bvtlist (list   "BANH VIT" "trai"    5   43   45      0.0
                        10   1     215   225   203  235  "-"   25
                        45   50    0))
 ;else
  (progn
   (setq llinev (read-line fffv))
   (close fffv)
   (setq bvtlist (read llinev))
  ) ;progn
 ) ;if

 ; khởi tạo thông số khung tên
 (setq ktlist_bvt (khten_init_bv))
 (setq dlg4v (load_dialog "chitiet.dcl"))
 (setq stat4v 41)
 (while (> stat4v 0)
  (if(not (new_dialog "ve_bvt" dlg4v)) (exit))

 ; khởi tạo hộp thoại
  (bvt_reinit)
  (tinh_ts_v)
  (mode_tile "kc0" 1)
  (mode_tile "kc1" 1)
  (mode_tile "kc2" 1)
  (fillimage "b_vit" "image1")
  (action_tile "file" "(done_dialog 44)")
  (action_tile "luu" "(nhap_dl_v)(done_dialog 41)")
  (action_tile "ve1" "(nhap_dl_v)(done_dialog 42)")
  (action_tile "thoat" "(done_dialog)")
  (action_tile "khten" "(nhap_dl_v)(done_dialog 43)")
  (action_tile "giup" "(help_bvt)")
  (action_tile "m" "(tinh_ts_v)")
  (action_tile "z" "(tinh_ts_v)")
  (action_tile "q" "(tinh_ts_v)")
  (action_tile "dy" "(tinh_ts_v)")
  (action_tile "xx" "(tinh_ts_v)")
  (action_tile "dlo" "(tinh_ts_v)")
  (setq stat4v (start_dialog))
  ; tuỳ theo chọn nút nào, sẽ thực hiện các thao tác tương ứng
  (cond
   ((= stat4v 41) (luubvt1))          ;luu
```

```
    ((= stat4v 42) (vebvt1))          :ve
    ((= stat4v 43) (napkhten_v))      :thay doi khung ten
    ((= stat4v 44) (docbvt1))         :doc du lieu tu file
   )
  )
  (unload_dialog dlg4v)
  (setvar "osmode" osmod1)
) ;;; kết thúc chương trình chính

;;; chương trình con tính thông số vẽ bánh vít
(defun tinh_ts_v (/ m z xx q dy d0 da df dm z1 da1 bw bwnew dmnew tt)
 ; doc du lieu..
 (setq m  (get_tile "m")      m  (atof m)
       z  (get_tile "z")      z  (atoi z)
       xx (get_tile "xx")     xx (atof xx)
       q  (get_tile "q")      q  (atof q)
       z1 (get_tile "dy")     z1 (atoi z1)
       bw (get_tile "bw")     bw (atof bw)
       dm (get_tile "dm")     dm (atof dm)
       d  (get_tile "dlo")    d  (atof d))
 ; tính các kích thước vẽ
 (setq d0   (- (* m (+ q z xx xx)) (* q m))
       da   (* m (+ z 2 xx xx))
       df   (* m (- (+ z xx xx) 2.4))
       da1  (+ (* q m) m m)
       dmnew (+ da m m)
       bwnew (* 0.75 da1))
 (if (= z1 2)   (setq dmnew (+ da (* 1.5 m))))
 (if (= z1 4)   (setq dmnew   (+ da m)   bwnew (* 0.67 da1)))
 ; moayo..
 (setq dmyo (* d 1.8)          tt  (- dmyo d))
 (if (< tt 15)   (setq tt 15))
 (setq dmyo (+ d tt))
 ; hiện kết quả lên hộp thoại
 (set_tile "d" (rtos d0))
 (set_tile "da" (rtos da))
 (set_tile "df" (rtos df))
 (set_tile "dm" (rtos dmnew))
 (set_tile "bw" (rtos bwnew))
 (set_tile "dmyo" (rtos dmyo))
)

;;; chương trình con hiện hộp thoại khung tên
(defun napkhten_v (/   dlg5   stat5   kt1   sohieu
                   thietke huongdan duyet truong lop vatlieu)
 (setq dlg5 (load_dialog "chitiet.dcl"))
 (setq stat5 51)
```

```
  ; khởi tạo
  (setq sohieu (nth 5 ktlist_bvt)
      thietke   (nth 0 ktlist_bvt)
      huongdan (nth 1 ktlist_bvt)
      duyet     (nth 2 ktlist_bvt)
      truong    (nth 3 ktlist_bvt)
      lop       (nth 4 ktlist_bvt)
      vatlieu   (nth 7 ktlist_bvt)
  )
  (while (> stat5 0)
    (if(not (new_dialog "khungten" dlg5)) (exit))
    (set_tile "sohieu" sohieu)
    (set_tile "thietke" thietke)
    (set_tile "huongdan" huongdan)
    (set_tile "duyet" duyet)
    (set_tile "truong" truong)
    (set_tile "lop" lop)
    (set_tile "vatlieu" vatlieu)
    (action_tile "accept" "(dlkhten_v) (done_dialog)")
    (action_tile "cancel" "(done_dialog)")
    (setq stat5 (start_dialog))
  )
  (unload_dialog dlg5)
)

;;; nhập dữ liệu khung tên
(defun dlkhten_v ( / sohieu thietke huongdan duyet truong
                     lop vatlieu chitiet)
  (setq sohieu  (get_tile "sohieu")
      thietke   (get_tile "thietke")
      huongdan (get_tile "huongdan")
      duyet     (get_tile "duyet")
      truong    (get_tile "truong")
      lop       (get_tile "lop")
      vatlieu   (get_tile "vatlieu")
      chitiet   (nth 0 bvtlist))
  (setq ktlist_bvt       ; trả kết quả về dưới dạng danh sách
      (list thietke huongdan  duyet truong lop sohieu chitiet  vatlieu))
)

;;; dữ liệu khung tên mặc định
(defun khten_init_bv ( /   sohieu     thietke  huongdan  duyet
                     truong  lop vatlieu  chitiet  kt1 )
  (setq sohieu  "01-01-01"
      thietke   "Tinh"
      huongdan "Chat"
      duyet     "Uyen"
      truong    "Truong DHBK Ha Noi"
      lop       "Co tin K38"
      vatlieu   "-"
```

```
 )
 (setq chitiet "BANH VIT")
 (setq kt1 (list thietke huongdan duyet truonglop sohieu chitiet
                 vatlieu))
 kt1
)

;;; chương trình con để lưu dữ liệu vào file
(defun luubvt1 (/ default fname outf)
 (setq default "c:/hedd/dwgdat/bvt/bvt.dbv")
 (setq fname (getfiled "Luu file du lieu" default "dbv" 1))
 (setq outf (open fname "w"))
 (if (not outf)   (exit) ) ;else do next lines
 (prin1 bvtlist outf)
 (close outf)
)

;;; chương trình con đọc dữ liệu từ file
(defun docbvt1 (/ default fname inf line)
 (setq default "c:/hedd/dwgdat/bvt/")
 (setq fname (getfiled "Nhap file du lieu" default "dbv" 4))
 (setq inf (open fname "r"))
 (if (not inf)  (exit))
 (setq line (read-line inf))
 (close inf)
 (setq bvtlist (read line))
)

;;; chương trình con cập nhật các thông số bánh vít
(defun bvt_reinit ( /  loaibr  hgr  beta bw  m  d0  da df
                       z  kxi  vl dy  d  dmyo  lmyo kcmyo)
 (setq loaibr (nth 0 bvtlist)
       hgr   (nth 1 bvtlist)
       m     (rtos (nth 2 bvtlist))
       z     (itoa (nth 3 bvtlist))
       bw    (rtos (nth 4 bvtlist))
       kxi   (rtos (nth 5 bvtlist))
       q     (rtos (nth 6 bvtlist))
       dy    (itoa (nth 7 bvtlist))  ;z1
       d0    (rtos (nth 8 bvtlist))
       da    (rtos (nth 9 bvtlist))
       df    (rtos (nth 10 bvtlist))
       dm    (rtos (nth 11 bvtlist))
       vl    (nth 12 bvtlist)
       d     (rtos (nth 13 bvtlist))
       dmyo  (rtos (nth 14 bvtlist))
       lmyo  (rtos (nth 15 bvtlist))
       kcmyo (nth 16 bvtlist)
 )
 (if (= "phai" hgr)   (set_tile "hg1" "1") )
```

```
  (if (= "trai" hgr)   (set_tile "hg2" "1") )
  (if (= 0 kcmyo)   (set_tile "kc0" "1") )
  (if (= 1 kcmyo)   (set_tile "kc1" "1") )
  (if (= 2 kcmyo)   (set_tile "kc2" "1") )

  (set_tile "m" m)
  (set_tile "z" z)
  (set_tile "bw" bw)
  (set_tile "xx" kxi)
  (set_tile "q" q)
  (set_tile "dy" dy)
  (set_tile "d" d0)
  (set_tile "da" da)
  (set_tile "df" df)
  (set_tile "dm" dm)
  (set_tile "dlo" d)
  (set_tile "dmyo" dmyo)
  (set_tile "lmyo" lmyo)
)

;;; chương trình con nhập các dữ liệu phục vụ vẽ
(defun nhap_dl_v ( / loaibr   hgr  beta bw   m d0  da  df z   kxi   vl
                     dy d  dmyo  lmyo  kcmyo hg1 kc1  kc2 )
  (setq loaibr (nth 6 ktlist_bvt)
      hgr   -1
      m     (atof (get_tile "m"))
      z     (atoi (get_tile "z"))
      bw    (atof (get_tile "bw"))
      kxi   (atof (get_tile "xx"))
      q     (atof (get_tile "q"))
      dy    (atoi (get_tile "dy"))
      d0    (atof (get_tile "d"))
      da    (atof (get_tile "da"))
      df    (atof (get_tile "df"))
      dm    (atof (get_tile "dm"))
      vl    (nth 7 ktlist_bvt)
       ;thông số mayo
      d     (atof (get_tile "dlo"))
      dmyo  (atof (get_tile "dmyo"))
      lmyo  (atof (get_tile "lmyo"))
      kcmyo 0
  )
  (setq hg1 (get_tile "hg1"))
  (if (= hg1 "1")   (setq hgr 1) )

  (setq kc1 (get_tile "kc1")
        kc2 (get_tile "kc2") )
  (if (= kc1 "1")   (setq kcmyo 1) )
  (if (= kc2 "1")   (setq kcmyo 2) )
  (setq bvtlist (list loaibr hgr m z bw  kxi q dy d0 da df dm vl  d dmyo
```

```
                    lmyo kcmyo) )
)

;;; chương trình con chèn bảng thông số
(defun bangthso_v (loai pt / rzz mmm zz2 xxx loaitv zz1 hgg ccx aw aaa
                   ddd kyhieu)
 (cond
  ((= hgr 1) (setq hgg "phai"))
  ((= hgr 0) (setq hgg "-"))
  ((< hgr 0) (setq hgg "trai"))
 )
 (setq aw (* 0.5 m (+ q z kxi kxi)))
 (setq rzz    "Rz45"
       mmm    (rtos m 2 2)
       zz2    (itoa z)
       zz1    (itoa dy)
       xxx    (rtos kxi 2 2)
       loaitv "Acximet"
       ccx    "8-C"
       ddd    (rtos d0 2 2)
       aaa    (rtos aw 2 2)
       kyhieu "xx-xx-xx"
 )
 (if (<= htext 2) ; kho A4
  (command "insert"  loai  pt  "0.8" ""    ""   rzz  mmm   zz2  xxx
                loaitv zz1 hgg  ccx  aaa  ddd   kyhieu)
 ;else
  (command "insert"  loai pt "" ""  ""  rzz   mmm  zz2  xxx  loaitv zz1
                hgg   ccx   aaa   ddd   kyhieu ) )
)

;;; vẽ bánh vít
;;; gọi hàm con phía dưới đây
(defun vebvt1 ()
 (vebvt ktlist_bvt)
)

;;; chương trình con vẽ bánh vít
(defun vebvt (ktlist_bvt  /     loaibr hgr     m      z     bw
              kxi  q        dy     d0     da     df     dm    vl
              aw   d        dmyo   lmyo   kcmyo s      lientruc
              dayvr dkvr  dkmyo daynanhoa     loaikc vl    tlist1
              b    h   t1    t2  rth tb1     tb2a  tb2b   tt1   tt2)

 ; chuẩn bị vẽ
 (command "._erase" "all" "")

 ; lấy các thông số
 (setq loaibr (nth 0 bvtlist)
       hgr    (nth 1 bvtlist)
```

```
      m     (nth 2 bvtlist)
      z     (nth 3 bvtlist)
      bw    (nth 4 bvtlist)
      kxi   (nth 5 bvtlist)
      q     (nth 6 bvtlist)
      dy    (nth 7 bvtlist) ;z1
      d0    (nth 8 bvtlist)
      da    (nth 9 bvtlist)
      df    (nth 10 bvtlist)
      dm    (nth 11 bvtlist)
      vl    (nth 12 bvtlist)
      d     (nth 13 bvtlist)
      dmyo  (nth 14 bvtlist)
      lmyo  (nth 15 bvtlist)
      kcmyo (nth 16 bvtlist)
)
(setq aw (* 0.5 m (+ q z kxi kxi)))
; kiểm tra điều kiện liên trục
(setq lientruc 0)
(if (< df d)   (progn
   (alert "Kết cấu có thể không hợp lý..")     (exit)   ) )
; thông số then
(setq tlist1 (tinhthen d))
(setq         b    (nth 1 tlist1)
      h    (nth 2 tlist1)
      t1   (nth 3 tlist1)
      t2   (nth 4 tlist1)
      rth  (nth 5 tlist1)
      tb1  (nth 6 tlist1)
      tb2a (nth 7 tlist1)
      tb2b (nth 8 tlist1)
      tt1  (nth 9 tlist1)
      tt2  (nth 10 tlist1)
)
(setq s (- (/ (- df d) 2.0) t2))
(if (< s (* 2.4 m))   (setq lientruc 1)  )
(if (= lientruc 1)
 (progn
   (vebvtlt)     (exit)   ) )
; else
; vẽ bánh vít rời
(setq dayvr (* m 2.0))
(if (< dayvr 8)   (setq dayvr 8)  )
(setq dkmyo dmyo)
(setq daimyo lmyo)
(setq daynanhoa 0)
; kiểm tra điều kiện khoét lỗ, làm lõm
(setq dkvr (- df (* 4 dayvr)))
```

```
  (setq v1 (- dkvr dkmyo))
  (setq v1 (* 0.5 v1))
  (if (< v1 0)
   (setq loaikc 0) ; bánh vít liền khối
   (if(< v1 20)
    (setq loaikc 1) ; không lõm
    (if (< v1 40)
     (progn
      (setq loaikc 2) ; lõm không khoét lỗ
      (setq dktlo (/ (+ dkmyo dkvr) 2.0))        (setq dklo 0)
     )
     (progn
      (setq loaikc 3)
      (setq dktlo (/ (+ dkmyo dkvr) 2.0))        (setq dklo 20)
     )
    )
   )
  ) ; lõm, khoét lỗ trên nan hoa
  (if (> loaikc 1)
   (progn
    (setq daynanhoa (* 0.25 bw))     (setq rr 5)    )
  )
  (vebvit loaikc m z aw dm da d0 df bw daimyo d daynanhoa)
)

;;; đường kính vít ghép vành và thân
(defun  tradkvit (m / mv)
 (setq mv (* 1.25 m))
 (if (< mv 6)
  (setq mv 6)
  (if(< mv 8)
   (setq mv 8)
   (setq mv 10)
  )
 )
 mv
)

;; vẽ
(defun vebvit (loaikc  m z w  dm  da d0  df  bw daimyo d  daynanhoa  /
          hh ww kglist ss  ss2 c0  c00  c1  c2  c10  c20  p1  p2  p01  p02
          p011  p012  delta p3 p4 p5 p6 p01 p02  p  pp  u1 u2 mv  mv2)

 (setq dq  (- df dayvr dayvr)
     dq2 (/ dq 2.0)
     mv  (tradkvit m)
     mv2 (/ mv 2.0)
 )
 ; đặt lớp
 (command "._-layer" "m" "kt" "c" "white" "" "lt" "continuous" "" "")
```

```
(if (< daimyo bw)   (setq daimyo bw) )
(setq hh    df
    ww    daimyo
    kglist (khogiay hh ww ktlist_bvt) )
(setq htext (nth 0 kglist)
    c1    (nth 1 kglist) )
; khởi tạo kiểu ghi kích thước
(setdimvar1)
; tính tọa độ các điểm
(setq c2 (polar c1 0 daimyo))
(setq delta (- daimyo bw))
(if (> delta 0)   (setq delta (/ delta 2.0)) )
(setq c10 (polar c1 0 delta)
    c20 (polar c2 g180 delta) )
(setq delta (/ delta (cos (dr 5))))
(setq       p1       (polar c1 g90 (+ (/ d 2.0) t2))
    p2       (polar p1 0 daimyo)
    n01      (polar c1 g90 (/ dkmyo 2.0))
    n011     (polar n01 (dr 5) delta)
    n02      (polar c10 g90 (/ dkvr 2.0))
    n04      (polar n02 0 bw)
    n03      (polar n01 0 daimyo)
    n031     (polar n011 0 bw)
    q0       (polar c20 g90 dq2)
    q1       (polar q0 g90 (/ bw 10))
    q2       (polar q1 g180 (/ bw 5))
    q3       (polar q0 g180 (/ bw 5))
    q11      (polar q1 g90 1)
    q12      (polar q2 g90 1)
    q01a     (polar q0 g180 bw)
    lechvit  2.0
    q01      (polar q01a g270 lechvit)
    v1       (polar q01 g270 mv2)
    v2       (polar v1 g90 mv)
    v3       (polar v1 0 (* 0.3 bw))
    v4       (polar v3 g90 mv)
    v5       (polar v3 (dr 45) (sqrt 2))
    v6       (polar v4 (dr 315) (sqrt 2))
    v7       (polar v5 0 2)
    v8       (polar v6 0 2)
    v9       (polar q01 0 (+ (* 0.3 bw) 2 (sqrt 2) (* mv 0.2)))
    v9a      (polar q01a 0 (+ (* 0.3 bw) 2 (sqrt 2)))
    v01      (polar v1 g90 1)
    v02      (polar v2 g270 1)
)
(setq u001 (polar c1 g270 (/ d 2.0))
    u002 (polar c2 g270 (/ d 2.0))
    u1   (polar u001 0 2.0)
    u01  (polar u001 g270 2.0)
```

```
    u2  (polar u002 g180 2.0)
    u02  (polar u002 g270 2.0)
)
(setq m01 (polar n01 g270 dkmyo)
    m03 (polar n03 g270 dkmyo)
)
(setq p01  (polar u1 g90 d)
    p02  (polar u2 g90 d)
    p011 (polar u001 g90 (+ d 2.0))
    p021 (polar u002 g90 (+ d 2.0))
)
(setq c0   (polar c10 0 (/ bw 2.0))
    c00  (polar c0 g90 aw)
    cf2  (- aw (/ df 2.0))
    cf0  (sqrt (- (* cf2 cf2) (* bw bw 0.25)))
    f0   (polar c00 g270 cf0)
    f1   (polar f0 g180 (/ bw 2.0))
    f2   (polar c00 g270 cf2)
    f3   (polar f1 0 bw)
    ca2  (- aw (/ da 2.0))
    ca0  (- aw (/ dm 2.0))
    a0   (polar c00 g270 ca0)
    a0a1 (sqrt (- (* ca2 ca2) (* ca0 ca0)))
    a1   (polar a0 g180 a0a1)
    a3   (polar a0 0 a0a1)
    a2   (polar c0 g90 (/ da 2.0))
    a01  (polar c10 g90 (/ dm 2.0))
    a02  (polar c20 g90 (/ dm 2.0))
    ca12 (- aw (/ d0 2.0))
    a12  (polar c00 g270 ca12)
    a13  (polar c00 (dr 330) ca12)
    a11  (polar c00 (dr 210) ca12)
)
; các điểm chỉ sử dụng cho kết cấu kiểu  2,3
(setq ll1 (- bw daynanhoa)
    ll1 (/ ll1 2.0)
    ll1 (/ ll1 (cos (dr 5)))
)
(setq n1 (polar n011 (dr 5) ll1)
    n2 (polar n02 (dr 355) ll1)
    n3 (polar n1 0 daynanhoa)
    n4 (polar n2 0 daynanhoa)
)
; lỗ fi20
(setq nl0 (polar c10 0 (/ bw 2.0))
    nl0 (polar nl0 g90 (/ dktlo 2.0))
    nl1 (polar nl0 g180 (/ daynanhoa 2.0))
    nl1 (polar nl1 g270 (/ dklo 2.0))
    nl2 (polar nl1 g90 dklo)
```

```
    nl3 (polar nl1 0 daynanhoa)
    nl4 (polar nl2 0 daynanhoa)
)

; vẽ
(command "._fillet" "r" 2)
(command "._-layer" "m" "lien" "c" "white" "" "lt" "continuous" "" "")
(command "._zoom" "e")
(setq ss (ssadd))
(if (= loaikc 0)
 (progn
  (command "._-layer" "make" "lien" "")
  (command "._pline" f3 n031 n03 "")
  (ssadd (entlast) ss)
  (command "._mirror" "l" "" c1 c2 "n")
  (ssadd (entlast) ss)
  (command "._pline" n01 n011 f1 "")
  (ssadd (entlast) ss)
  (command "._mirror" "l" "" c1 c2 "n")
  (ssadd (entlast) ss)
  (command "._arc" f1 f2 f3)
  (ssadd (entlast) ss)
  (command "._mirror" ss "" c1 c2 "n")
  (ssadd (entlast) ss)
  (command "._pline" n01 p1 p2 n03 "")
  (ssadd (entlast) ss)
  (command "._pline" m01 u01 u1 u2 u02 m03 "")
  (ssadd (entlast) ss)
  ; gạch mặt cắt..
  (command "._-layer" "make" "hatch" "c" "yellow" "" "")
  (command "._hatch" "u" "45" "5" "n" ss "")
 )
) ; if loaikc = 0

; loại kết cấu 1
(if (= loaikc 1)
 (progn
   ; phần thân
  (command "._-layer" "make" "lien" "")
  (command "._pline" n03 n031 q1 q2 q3 v9 v7 v5 v3 v1 n011 n01 "")
  (ssadd (entlast) ss)
  (command "._mirror" "l" "" c1 c2 "n")
  (ssadd (entlast) ss)
  (command "._pline" n01 p1 p2 n03 "")
  (ssadd (entlast) ss)
  (command "._pline" m01 u01 u1 u2 u02 m03 "")
  (ssadd (entlast) ss)
   ; mặt cắt
  (command "._-layer" "make" "hatch" "c" "yellow" "" "")
  (command "._hatch" "u" "45" "5" "n" ss "")
  (setq ss nil          ss (ssadd) )
```

```
    ; vành răng
   (vevanh_bvt)
  )
) ; if loaikc = 1

; kết cấu kiểu 2,3
(if (>= loaikc 2)
 (progn
    ; phần thân
  (setq ss (ssadd))
  (command "._fillet" "r" 5)
  (command "._-layer" "make" "lien" "")
  (command "._pline" n03 n3 nl3 "")
  (command "._fillet" "p" "l")
  (ssadd (entlast) ss)
  (command "._mirror" "l" "" c1 c2 "n")
  (ssadd (entlast) ss)
  (command "._pline" n01 n1 nl1 "")
  (command "._fillet" "p" "l")
  (ssadd (entlast) ss)
  (command "._mirror" "l" "" c1 c2 "n")
  (ssadd (entlast) ss)
  (command "._pline" n02 n2 nl2 "")
  (command "._fillet" "p" "l")
  (ssadd (entlast) ss)
  (command "._mirror" "l" "" c1 c2 "n")
  (ssadd (entlast) ss)
  (command "._pline" n04 n4 nl4 "")
  (command "._fillet" "p" "l")
  (ssadd (entlast) ss)
  (command "._mirror" "l" "" c1 c2 "n")
  (ssadd (entlast) ss)
  (command "._pline" n04 q1 q2 q3 v9a v9 v7 v5 v3 v1 n02 "")
  (ssadd (entlast) ss)
  (command "._mirror" "l" "" c1 c2 "n")
  (ssadd (entlast) ss)
  (if (= loaikc 2)
   (progn
    (command "._pline" nl1 nl2 "")
    (ssadd (entlast) ss)
    (command "._mirror" "l" "" c1 c2 "n")
    (ssadd (entlast) ss)
    (command "._pline" nl3 nl4 "")
    (ssadd (entlast) ss)
    (command "._mirror" "l" "" c1 c2 "n")
    (ssadd (entlast) ss)
   )
  ; else
    (progn
     (command "._pline" nl1 nl3 "")
```

```
      (ssadd (entlast) ss)
      (command "._mirror" "l" "" c1 c2 "n")
      (ssadd (entlast) ss)
      (command "._pline" nl2 nl4 "")
      (ssadd (entlast) ss)
      (command "._mirror" "l" "" c1 c2 "n")
      (ssadd (entlast) ss)
     )
    )
    (command "._pline" n01 p1 p2 n03 "")
    (ssadd (entlast) ss)
    (command "._pline" m01 u01 u1 u2 u02 m03 "")
    (ssadd (entlast) ss)
     : mặt cắt
    (command "._-layer" "make" "hatch" "c" "yellow" "" "")
    (command "._hatch" "u" "45" "5" "n" ss "")
    (setq ss nil
        ss (ssadd)
    )

    (command "._-layer" "make" "lien" "")
    (command "._pline" n02 n011 "")
    (command "._mirror" "l" "" c1 c2 "n")
    (command "._pline" n04 n031 "")
    (command "._mirror" "l" "" c1 c2 "n")
    (vevanh_bvt)
   )
) ; if loaikc = 2,3

(if (= loaikc 3)
 (progn
  (command "._-layer" "make" "lien" "")
  (command "._pline" nl1 nl2 "")
  (command "._mirror" "l" "" c1 c2 "n")
  (command "._pline" nl3 nl4 "")
  (command "._mirror" "l" "" c1 c2 "n")
  (command "._-layer" "m" "cen" "c" "red" "" "lt" "center"  "" "")
  (command "._pline"  (polar nl0 g180 (+ (/ daynanhoa 2.0) 5))
                      (polar nl0 0 (+ (/ daynanhoa 2.0) 5)) "" )
  (command "._mirror" "l" "" c1 c2 "n")
 )
)

: vành răng và đường tâm
(setq ss nil)  (setq ss (ssadd))
(command "._-layer" "make" "lien" "")
(command "._pline" f1 a01 a1 "")
(command "._chamfer" "d" 1.5 1.5)
(command "._chamfer" "p" "l")  (ssadd (entlast) ss)
(command "._mirror" "l" "" c0 c00 "n")  (ssadd (entlast) ss)
```

```
(command "._arc" a1 a2 a3) (ssadd (entlast) ss)
(command "._-layer" "make" "cen" "c" "red" "" "lt" "center" "" "")
(command "._ltscale" 15)
(command "._arc" a11 a12 a13) (ssadd (entlast) ss)
(command "._mirror" ss "" c1 c2 "n")
(command "._pline" (polar c1 g180 5) (polar c2 0 5) "")
(setq ss nil)
; vát góc
(command "._-layer" "make" "lien" "")
(command "._pline" u01 p1 "")
(command "._pline" u02 p2 "")
(command "._pline" u1 p01 p011 "")
(command "._pline" u2 p02 p021 "")
(command "._pline" p01 p02 "")
; kích thước
(command "._-layer" "m" "dimtext" "c" 6 "" "lt" "continuous" "" "")
(setq u6 (polar a01 g270 dm)          u5 (polar u6 0 bw) )
(dimhor1 u5 u6 (polar u5 g270 20) (rtos bw 2 0))
(dimhor1 (polar u6 0 1.5) u6 (polar u6 g270 10)
      (strcat (rtos 1.5 2) "*45" "%%d") )
(rwrite (polar u6 (dr 245) 18) htext 0 "hai ben")
(setq p (polar c2 0 20))
(if (/= loaikc 0)
 (progn
  (setq q30 (polar q3 g270 dq))
  (dimhor1 q30 u5 (polar u6 g270 10) (rtos (/ bw 5) 2 0))
  (if (= loaikc 3)
   (progn
    (dimver1 nl0  (polar nl0 g270 dktlo) p
             (strcat "%%c" (rtos dktlo 2 0))  )
    (setq p (polar p 0 (* htext 3)))
    (dimver1 nl1 nl2 (polar n02 g180 10) "4%%c20")
   )
  )
  (if (>= loaikc 2)
   (progn
    (dimver1 n04
             (polar n04 g270 dkvr) p
             (strcat "%%c" (rtos dkvr 2 0)) )
    (setq p (polar p 0 (* htext 3)))
   )
  )
  (dimver1 q3 q30 p (strcat "%%c" (rtos dq 2 0) "H7/k6"))
  (setq p (polar p 0 (* htext 3)))
 )
)

(if (> delta 0)
 (progn
  (dimhor1 m01 m03 (polar m03 g270 15) (rtos daimyo 2 1))
```

```
   (dimver1 n01 m01 (polar c1 g180 10)
         (strcat "%%c" (rtos dkmyo 2 0)) )
  )
)
(dimver1 a2  (polar a2 g270 da) (polar c1 g180 (* htext 6))
      (strcat "%%c" (rtos da 2 2) "-0.100") )
(dimver1 a01  u6  (polar c1 g180 (* htext 9))
      (strcat "%%c" (rtos dm 2 2) "-0.100") )
(dimlead1 (polar c00 (dr 240) ca2)  c00 (strcat "R" (rtos ca2 2 2)) "")
(dimlead0 u1 (polar u1 (dr 45) 15) (strcat (rtos 2 2) "*45%%d")
            "hai ben" )
; nhám và dung sai khác
(command "._-layer" "make" "toler" "c" 8 "" "lt" "continuous" "" "")
(command "._pline" u2 (polar u2 0 18) "")
(setq pt (polar u2 0 14))
(matchuan pt (polar pt g270 10) 270 htext "A" )
; nhám
(nham (polar u2 0 10) 0 htext "2.5")
(setq pt (polar c1 g180 (* htext 3.5))  pt (polar pt g90 (/ da 2.0))  )
(nham pt 10 htext "2.5")
(nham a13 35 htext "2.5")
(if (> delta 0)
 (progn
  (command "pline" n03 (polar n03 g90 15) "")
  (nham (polar n03 g90 10) 270 htext "2.5") )
)
(if (/= loaikc 0)  (nham (polar n04 g270 10) 270 htext "2.5") )
(nham (polar f3 g270 5) 270 htext "2.5")
; dung sai vị trí
(setq pt (polar f3 g270 (* (distance c20 f3) 2.0))  pt (polar pt g90 3) )
(dimtol pt (polar pt 0 htext) 0 htext "dao" "0.05" "AA")
(if (or (> delta 0) (/= loaikc 0))
 (progn
  (setq pt (polar p1 g90 3))
  (dimtol pt  (polar pt g180 (* htext 10)) 180  htext  "dao"
       "0.02" "AA" ))
)
; vẽ hình chiếu bên (lỗ mayơ)
(command "._-layer" "make" "lien" "")
(if (> loaikc 1)
 (lo_moayo (polar p 0 (* htext 18)) d b t2 tb2a tb2b tt2 htext)
 (lo_moayo (polar p 0 (* htext 18)) d b t2 tb2a tb2b tt2 htext)
)
; chèn ghi chú và bảng thông số
(setq   ltext (list
       "1. Co so che tao chon bo kiem tra theo TCVN 1067-84"
       "2. Nhiet luyen: toi cai thien, HB 190..220"
       "3. Sai lech gioi han khong chi dan cua cac kich thuoc:"
       "     bao - H14,  bi bao - h14,  con lai %%p IT12/2"
       "4. Dung sai do song song ranh then 0.022"
```

```
        "5. Dung sai do doi xung ranh then  0.086")
  )
  (setq pt (nth 3 kglist))
  (brnote pt htext ltext)
  (setq pt (nth 4 kglist))
  (bangthso_v "bths-bv" pt)
)
;;; vẽ vành răng bánh vít
(defun vevanh_bvt ()
  ; vanh..
  (command "._-layer" "make" "lien" "")
  (command "._pline" q1 q11 "")       (ssadd (entlast) ss)
  (command "._pline" v1 v2 "")        (ssadd (entlast) ss)
  (command "._pline" v3 v4 "")        (ssadd (entlast) ss)
  (command "._pline" v5 v6 "")        (ssadd (entlast) ss)
  (command "._pline" v7 v8 "")        (ssadd (entlast) ss)
  (command "._mirror" ss "" c1 c2 "n")  (setq ss (ssadd))
  (command "._pline" f1 v2 v4 v6 v8 v9a q3 q2 q12 q11 f3 "")
  (ssadd (entlast) ss)
  (command "._mirror" "l" "" c1 c2 "n")  (ssadd (entlast) ss)
  (command "._arc" f1 f2 f3)             (ssadd (entlast) ss)
  (command "._mirror" ss "" c1 c2 "n")   (ssadd (entlast) ss)
  (command "._-layer" "make" "hatch" "")
  (command "._hatch" "u" "315" "5" "n" ss "")
  (setq ss nil)
  (command "._pline" v01 v5 "")
  (command "._mirror" "l" "" c1 c2 "n")
  (command "._pline" v02 v6 "")
  (command "._mirror" "l" "" c1 c2 "n")
  (command "._-layer" "make" "cen" "c" "red" "" "lt" "center" "" "")
  (command "._ltscale" 15)
  (command "._pline" (polar q01 g180 5) (polar v9 0 5) "")
  (command "._mirror" "l" "" c1 c2 "n")
)

(defun vebvtlt ()
  (alert  "Banh vit lien truc ???. Hay chay chuong trinh ve truc.." )
)

;;; Hộp alert hiện trợ giúp
(defun help_bvt ()
 (alert
   "* Modul tieu chuan m: (1,5) 2 2,5 (3) 3,15 (3,5) 4  5 (6) 6,3 (7) 8 10
      12,5 16 (18) 20
     \n* He so duong kinh q: (khong dung gia tri * khi m=2; **
      khi m=2,5)
     \n  6,3  (7,1) 8  (9) 10* (11,2) 12,5** (14) 16 (18) 20 (22,4) 25;
     \n* Cac so lieu: d, da, df, daM2 va bw duoc tu dong tinh lai,
      trong do daM2 va bw
     \n  duoc lay theo gia tri lon nhat cua chung.
```

```
        Hay nhap lai gia tri khac neu can.
        \n* De thay doi cac thong so ghi tren khung ten hay chon
        nut Khung ten." )
)

;;; Một số hàm xử lý danh sách

;;; Tạo danh sách liên kết từ các dòng trong file văn bản
(defun mk_list(fname / line f madelist char1 line1)
      (setq f (open fname "r"))
      (if (not f) (progn
                  (alert (strcat "Cannot open file " fname " to read."))
                  (exit))
                  )
      (setq madelist '())
      (while  (setq     line (read-line f))
                  (setq char1 (substr line 1 1))
                  (if (/= "/" char1)  (progn
                              (if (/= "(" char1) (setq line (strcat "(" line)))
                              (if (/= ")" (substr line (strlen line) 1))
                                          (setq line (strcat line ")")))
                              (setq     line1 (read line))
                              (setq madelist (append madelist (list line1))) ) )
      )
      (close f)
      madelist
)

;;; Tạo danh sách chứa số liệu ở trường khoá
;;; của một danh sách liên kết khác
(defun mk_key_list(llist key / ll lll newlist slist count keyitem)
      (if (not llist) (progn
                  ;(prompt "Nil list.")
                  (exit))  )
      (setq ll (length llist))
      (if (= ll 0) (progn
                  ;(prompt "Empty list.")
                  (exit))  )
      ; else...
      (setq slist (nth 0 llist))
      (setq lll (length slist))
      (if (or (>= key lll) (< key 0)) (progn
                  ;(prompt "Out of range.")
                  (exit))  )
      (setq count 0)
      (setq newlist '())
      (while (< count ll)
                  (setq slist (nth count llist))
                  (setq keyitem (nth key slist))
                  (setq newlist (append newlist (list keyitem)))
```

```
            (setq count (+ 1 count))
        )
        newlist
)

;;; Tìm kiếm danh sách
(defun scan_iteml (llist k1 ik10 ik11 k2 ik20 ik21 /
                   ik1 ik2 slist dk found ll key)
    (setq dk nil
          count 0
          found nil
          ll (length llist))
    (cond
          ((= nil llist) (progn      ;(prompt "Nil list.")
                (setq key nil)))
          ((= 0 ll) (progn           ;(prompt "Empty list.")
                (setq key nil)))
          (t  (progn
              (while (not dk)
                  (setq slist (nth count llist))
                  (setq ik1 (nth k1 slist))
                  (if (= k2 nil) (setq ik2 ik1)
                    (setq ik2 (nth k2 slist)))
                  (setq found (and (>= ik1 ik10) (<= ik1 ik11)
                                   (>= ik2 ik20) (<= ik2 ik21)))
                  (if (not found) (setq count (+ 1 count)))
                  (setq dk (or found (>= count ll)))
              ) ;end while
              (if (not found) (progn
                  ;(prompt "No such one in this list.")
                  (setq key nil))
                  ; else
                  (setq key count))
              ) ; progn T
          ) ; cond T
    ) ; cond
    key
)

;// Một số hàm xử lý DCL

;//  Chuyển tính năng is_enable của tile thành true
;//  Key của các tile truyền qua danh sách ltiles
(defun tilesON (ltiles / tile1)
    (foreach tile1 ltiles (mode_tile tile1 0))
)

;//  Chuyển tính năng is_enable của tile thành false
;//  Key của các tile truyền qua danh sách ltiles
```

```
(defun tilesOFF (ltiles / tile1)
      (foreach tile1 ltiles (mode_tile tile1 1))
)

;// Thao tác với Toogle
(defun togg(tilekey tlist / ans)
      (setq ans (get_tile tilekey))
      (if (= ans "1") (tilesON  tlist)
                      (tilesOFF tlist)    )
      ans
)

;// FillImage
;// Điền ảnh slide cho nút image và image_button
(defun fillimage(imagefile imagekey / x y)
      (setq x (dimx_tile imagekey)
            y (dimy_tile imagekey))
      (start_image imagekey)
      (fill_image 0 0 x y 0)
      (slide_image 0 0 x y imagefile)
      (end_image)
)

;// Khởi tạo danh sách cho popup_list hoặc list_box
;// Key là khóa liên kết với popup_list hoặc list_box
;// Các phần tử lấy từ cho trong llist
(defun mk_dlg_list ( key llist  / list1)
      (start_list key)
      (foreach list1 llist (add_list list1))
      (end_list)
)

;;; Một số hàm ghi kích thước, dung sai, văn bản,...
;;; và các hàm trợ giúp khác

;;; Kiểu định dạng kích thước
(defun setdimvar1()
      (command "-style" "dimtext" "romans" "" "" 15 "" "" "")
      (command "dimasz" htext)        ;arrow size
      (command "dimdec" 2)            ;decimal places
      (command "dimtdec" 3)           ;decimal places for tolerance
      (command "dimtzin" 0)
      (command "dimzin" 8)            ;suppress zero leading
      (command "dimtxt" htext)
      (command "dimtad" 1)
      (command "dimtix" 0)
      (command "dimgap" 2)            ;dimgap
      (command "dimexo" 2)
      (command "dimexe" 3)
```

```
        (command "dimtih" 0)
        (command "dimtolj" 0)
        (command "dimtxsty" "dimtext")
)

; Ghi kích thước ngang, không kèm dung sai
(defun dimhor1(p1 p2 p3 text)
        (command "dimtol" "off")
        (command "dim" "hor" p1 p2 p3 text "exit")
)

; Ghi kích thước dọc không kèm dung sai
(defun dimver1(p1 p2 p3 text)
        (command "dimtol" "off")
        (command "dim" "ver" p1 p2 p3 text "exit")
)

; Ghi kích thước ngang kèm dung sai
(defun dimhor2(p1 p2 p3 tp tm lead trail)
        (command "dimtol" "on")
        (command "dimtp" tp)
        (command "dimtm" tm)
        (command "dim" "hor" p1 p2 p3 (strcat lead "<>" trail) "exit")
)

; Ghi kích thước dọc kèm dung sai
(defun dimver2(p1 p2 p3 tp tm lead trail)
        (command "dimtol" "on")
        (command "dimtp" tp)
        (command "dimtm" tm)
        (command "dim" "ver" p1 p2 p3 (strcat lead "<>" trail) "exit")
)

; Ghi chú (đánh số chi tiết)
(defun dimlead1 (p1 p2 text text2)
        (command "dim" "lea" p1 p2 "" text "exit")
        (command "text" (polar p2 (dr 270) 5) htext "0" (strcat " " text2))
)
(defun dimlead0 (p1 p2 text text2 / dimasz1)
        (setq dimasz1 (getvar "dimasz"))
        (command "dimasz" 0)
        (dimlead1 p1 p2 text text2)
        (command "dimasz" dimasz1)
)

; Ghi văn bản lên bản vẽ
; justify left
(defun lwrite( p size ang text)
        (command "text" p size ang text)
```

```
)
; justify right
(defun rwrite(p size ang text)
     (command "text" "j" "r" p size ang text)
)
; justify center
(defun cwrite(p size ang text)
     (command "text" "j" "c" p size ang text)
)
;justify middle center
(defun mcwrite(p size ang text)
     (command "text" "j" "mc" p size ang text)
)
;justify middle left
(defun mlwrite(p size ang text)
     (command "text" "j" "ml" p size ang text)
)

; Ghi mặt chuẩn
(defun matchuan(p1 p2 alfa size mc / ang p size2)
     (setq size2 (* 2.0 size)   ang (angle p1 p2))
     (command "pline" p1 "w" 2 0 (polar p1 ang 1.5) p2 "")
     (command "pline" p2
              (setq p (polar p2 0 size))
              (setq p (polar p (dr 90) size))
              (setq p (polar p (dr 0) size2))
              (setq p (polar p (dr 270) size2))
              (setq p (polar p (dr 180) size2))
              (polar p (dr 90) size) "")
     (if (/= alfa) (command "rotate" "l" "" p2 alfa))
     (setq p (polar p2 (dr alfa) size2))
     (mcwrite p size 0 mc)
)

; Nhám
(defun nham(p ang size value / p1 ang1 size1 size2)
     ; góc xoay ang trong khoảng 0..360
     (setq size1 (* 1.5 size)
              size2 (* 3.0 size)
              p1 (polar p (dr (+ ang 140)) size1)
              ang1 (+ ang 90))
     (command "pline" p1 p (polar p (dr (+ ang 60)) size2) "")
     (setq p1 (polar p1 (dr ang1) (* size 0.7)))
     (setq ang1 ang)
     (if (> ang 180) (setq ang1 (- ang 180)))
     (if (<= ang1 90)  (mcwrite p1 size ang1 value)
                       (mcwrite p1 size (- ang1 180) value))
)
```

```
; Dung sai vị trí
(defun dimtol( p1 p2 alfa size loai gtri mc /
               ss ang size2 size3 size5 size07 size14 p p0 p01 p02)
      ; góc quay alfa = 0 hoặc 180..
      (setq size2 (* 2 size)
            size3 (* 3 size)
            size5 (* 5 size)
            size07 (* 0.7 size)
            size14 (* 1.4 size)
            ang (angle p1 p2)
            ss (ssadd))
      (command "pline" p1 "w" "0" "0.8" (polar p1 ang size)
                       "w" "0" "0" p2 "")
      (command "pline" p2
            (setq p (polar p2 0 size))
            (setq p (polar p (dr 90) size))
            (setq p01 (polar p 0 size3))
            (setq p02 (polar p01 0 size5))
            (setq p (polar p02 0 size3))
            (setq p (polar p (dr 270) size2))
            (setq p (polar p (dr 180) (+ size3 size5 size3)))
            (polar p (dr 90) size) "")
       (ssadd (entlast) ss)
       (command "pline" p01 (polar p01 (dr 270) size2) "")
       (ssadd (entlast) ss)
       (command "pline" p02 (polar p02 (dr 270) size2) "")
       (ssadd (entlast) ss)
       (if (/= alfa 0) (progn
            (command "rotate" ss "" p2 alfa)
            (setq p (polar p2 (dr alfa) (* size 10.5))) )
            ;else
            (setq p (polar p2 0 (* 2.5 size))) )
        (setq ss nil)
        (cond
            ((= loai "dao") (progn
                  (command "pline" (setq p0 (polar p (dr 45) size07))
                        "w" "0" "0.8" (polar p0 (dr 225) 1.5)
                        "w" "0" "0" (polar p (dr 225) size07) "")
                  ))
            ((= loai "vuonggoc") (progn
                  (command "pline" (setq p0 (polar p (dr 90) size07))
                        (setq p0 (polar p0 (dr 270) size)) "")
                  (command "pline" (polar p0 0 size07)
                        (polar p0 (dr 180) size07) "")
                  ))
             ((= loai "doixung") (progn
                  (setq p0 (polar p (dr 180) size))
                  (command "pline" p0 (polar p0 0 size2) "")
                  (setq p0 (polar p (dr 90) (* 0.5 size)))
                  (setq p0 (polar p0 (dr 180) (* size 0.5)))
```

```
                    (command "pline" p0 (polar p0 0 size) "")
                    (command "copy" "l" "" p0 (polar p0 (dr 270) size))
                    ))
            ((= loai "dongtam") (progn
                    (command "circle" p "d" size)
                    (command "circle" p "d" size14)
                    ))
            ((= loai "tru") (progn
                    (command "circle" p "d" size)
                    (setq p0 (polar p 0 size07))
                    (setq p0 (polar p0 (dr 60) size07))
                    (command "pline" p0
                            (polar p0 (dr 240) size14) "")
                    (command "copy" "l" ""
                            p0 (polar p0 (dr 180) size14))
                    ))
             ;.... các loại dung sai khác (tự thêm theo nhu cầu)
             )
    (mcwrite (polar p 0 (* size 4)) size 0 gtri)
    (mcwrite (polar p 0 (* size 8)) size 0 mc)
)

; Tra thông số then bằng
(defun tinhthen(d / fn key tlist tlist1)
    (setq fn (findfile "thenbang.dat"))
    (setq tlist (mk_list fn))
    (setq key (scan_iteml tlist 0 d 1000 0 0 1000))
    (setq tlist1 (nth key tlist))
    (setq tlist nil)
    tlist1
)

; Vẽ lỗ moayơ
(defun lo_moayo( p0 d b tt tpb tmb tpt htext /
                d2 dr5 b2 l1 l2 p1 p2 p3 p4 p5 c1 c2 c3 c4 ss)
    (setq d2 (/ d 2.0)
            d25 (+ d2 5)
            b2 (/ b 2.0)
            l1 (sqrt (- (* d2 d2) (* b2 b2)))
            l2 (+ d2 tt)
            ss (ssadd))
    (setq p1 (polar p0 (dr 90) l1))
    (setq p1 (polar p1 (dr 180) b2)
            p2 (polar p0 0 d2)
            p3 (polar p1 0 b)
            p4 (polar p0 (dr 90) l2)
            p4 (polar p4 0 b2)
            p5 (polar p4 (dr 180) b))
    (setq c1 (polar p0 (dr 90) l2)
```

```
            c3 (polar p0 (dr 180) d2)
            c4 (polar p0 0 d2))
    (command "._arc" p1 p2 p3)      (ssadd (entlast) ss)
    (command "._pline" p3 p4 p5 p1 "")      (ssadd (entlast) ss)

    ; gạch mặt cắt
    (if (< tt 0) (progn
            (command "._-layer" "make" "_mss.cat" "")
            (command "._hatch" "u" "45" "5" "n" ss "")
            ))
            (setq ss nil)

    ; đường tâm
    (command "._-layer" "make" "_mss.tam" "")
    (command "._ltscale" 15)
    (command "._pline"  (polar c1 (dr 90) 5)
                        (polar c2 (dr 270) 5) "")
    (command "._pline" (polar c3 (dr 180) 5) (polar c4 0 5) "")

    ; ghi kích thước
    (command "-layer" "make" "_mss.text" "")
    (dimhor1 c3 c4 (polar p0 (dr 270) (+ d2 10))
                (strcat "%%c" (rtos d 2 0)))
    (dimhor2 p5 p4 (polar p0 (dr 90) (+ 12 d2)) tpb (* -1 tmb) "" "")
    (if (> tt 0)
            (dimver1 (polar p0 (dr 270) d2)  p4 (polar p0 0 (+ d2 20))
                (strcat "<>+" (rtos tpt 2 3)))
            (dimver1 (polar p0 (dr 90) d2)   p4 (polar p0 0 (+ 20 d2))
                (strcat "<>+" (rtos tpt 2 3)))
            )

    ; nhám và dung sai khác
    (command "-layer" "make" "_mss.text" "")
    (command "pline" (polar p4 (dr 90) 10) (polar p4 (dr 90) 30) "")
    (nham (polar p4 (dr 90) 24) 90 htext "Rz20")
    (nham (polar p4 0 d2) 180 htext "Rz40")
)

; Viết các dòng ghi chú lên bản vẽ
(defun bmote(p0 size ltext / size3 p ltext ll i)
    (setq p p0)
    (setq size3 (* 2.5 size))
    (setq ll (length ltext))
    (setq i 0)
    (while (< i ll)
            (lwrite p size 0 (nth i ltext))
            (setq p (polar p (dr 270) size3))
            (setq i (+ 1 i))
            )
)
```

PHỤ LỤC 2

PHẦN MỀM HEDD

1. Phần mềm HEDD là một chương trình tính toán thiết hệ dẫn động cơ khí sử dụng các loại hộp giảm tốc khác nhau và kết nối khác nhau với động cơ điện và cơ cấu chấp hành, thể hiện bằng các biến sau đây:

a) *sd* được định nghĩa là một biến kiểu *Byte*, dùng để biểu thị các loại sơ đồ sau đây của hộp giảm tốc:

sd = 1: sơ đồ hộp giảm tốc bánh răng trụ một cấp ;

sd = 2: sơ đồ hộp giảm tốc bánh răng côn một cấp ;

sd = 3: sơ đồ hộp giảm tốc trục vít một cấp ;

sd = 4: sơ đồ hộp giảm tốc bánh răng côn - trụ hai cấp ;

sd = 5: sơ đồ hộp giảm tốc trục vít - bánh răng trụ ;

sd = 6: sơ đồ hộp giảm tốc bánh răng trụ - trục vít ;

sd = 7: sơ đồ hộp giảm tốc bánh răng trụ hai cấp đồng trục;

sd = 8: sơ đồ hộp giảm tốc bánh răng trụ hai cấp khai triển, có bộ truyền cấp nhanh đặt gần đầu vào;

sd = 9: sơ đồ hộp giảm tốc bánh răng trụ hai cấp khai triển, có bộ truyền cấp nhanh đặt xa đầu vào;

sd = 10: sơ đồ hộp giảm tốc bánh răng trụ hai cấp phân đôi cấp nhanh;

sd = 11: sơ đồ hộp giảm tốc bánh răng trụ hai cấp phân đôi cấp chậm;

sd = 12: sơ đồ hộp giảm tốc trục vít hai cấp;

sd = 13: sơ đồ hộp giảm tốc bánh răng trụ ba cấp ;

sd = 14: sơ đồ hộp giảm tốc bánh răng côn - trụ ba cấp ;

sd = 15: sơ đồ hộp giảm tốc bánh răng côn - trụ hai cấp có hai đầu ra, qua hai bộ truyền xích nối với hai cơ cấu chấp hành.

b) *pa* cũng được định nghĩa là một biến kiểu *Byte*, dùng để biểu thị phương án nối hộp giảm tốc với động cơ điện và trục công tác của máy:

pa = 1: nối động cơ điện với trục vào của hộp giảm tốc bằng bộ truyền đai, trục ra của hộp giảm tốc với trục làm việc bằng khớp nối.

pa = 2: nối động cơ điện với trục vào của hộp giảm tốc bằng khớp nối, trục ra của hộp giảm tốc với trục làm việc bằng bộ truyền xích.

pa = 3: dùng khớp nối để nối các trục của hộp giảm tốc với trục của động cơ điện và với trục làm việc.

2. Phần mềm HEDD bao gồm 4 chương trình lớn:

Program Chondco : tính toán động học, chọn động cơ điện và phân phối tỉ số truyền của hệ cho các cấp.

Program Botruyen: tính toán thiết kế các bộ truyền trong hộp và ngoài hộp (bộ truyền đai hoặc bộ truyền xích); chọn khớp nối để nối đầu trục vào với trục động cơ hoặc đầu trục ra với trục của cơ cấu chấp hành.

Program Trucolan: tính toán thiết kế trục và chọn ổ lăn cho các trục của hộp giảm tốc.

Program Banve : tính toán các yếu tố của vỏ hộp giảm tốc và thiết kế tự động bản vẽ lắp hộp giảm tốc.

Chương trình chính và các đơn thể (*Program*) sử dụng các *Unit* sau đây:

- *Unit tb* chứa các biến tổng thể và các cách tra bảng khác nhau sử dụng cho toàn bộ chương trình;

- *Unit kht* chứa các kiểu khung hiển thị và các giao diện dùng chung cho toàn bộ chương trình;

- *Unit solict* chứa các chương trình con nạp dữ liệu và gọi dữ liệu cũng như nạp và gọi kết quả tính toán cho các chương trình tính toán tiếp theo và các chương trình vẽ tự động các bản vẽ kỹ thuật.

Kèm theo phần mềm là toàn bộ các bảng tra cần cho tính toán và vẽ đã được nạp vào đĩa.

3. Để giảm thiểu dung lượng cho phần mềm HEDD, các chương trình lớn sử dụng nhiều chương trình con dưới dạng *Unit* và *Unit Overlay*.

a) Riêng *Program Chondco* sử dụng các *Unit* và *Units Overlay* sau đây:

- *dlieu* : nạp và gọi dữ liệu đầu vào cho các sơ đồ của hệ dẫn động;

- *sodohe* : sử dụng phần mềm đồ họa PASCAL để vẽ các loại sơ đồ của hệ dẫn động;

- *udc* : tính toán chọn động cơ; phân phối tỉ số truyền của hệ cho các cấp; tính công suất, số vòng quay và mômen xoắn trên các trục của hệ.

b) Riêng *Program Botruyen* sử dụng các *Unit Overlay* sau đây:

- *uscp* : chọn vật liệu và xác định ứng suất cho phép dùng chung cho các bộ truyền bánh răng trụ và bánh răng côn răng thẳng và răng không thẳng;

- *ubr, utv* : tính các bộ truyền trong hộp : bộ truyền bánh răng hoặc trục vít ;

- *udai, uxic*: tính toán các bộ truyền ngoài hộp (bộ truyền đai hoặc bộ truyền xích) ;

c) Riêng *Program Trucolan* sử dụng các *Unit Overlay* sau đây:

- *tktruc : tính toán thiết kế trục;*

- *olan : chọn ổ lăn ;*

4. Phần mềm HEDD, được xây dựng theo lưu đồ thuật giải vẽ trên hình 7.1, trên đó trình bày kỹ hơn về *Program Botruyen* . Đó là vì đối với các sơ đồ khác nhau của hệ dẫn động, có sự khác nhau về nội dung tính toán (do sử dụng loại bộ truyền khác nhau) và trình tự tính toán (nhằm giảm thiểu sai số của tỉ số truyền chung của hệ), trong khi đó đối với *Program Chondco* và *Program Trucolan*, chỉ có sự khác nhau ở các phần tính toán cụ thể, những nội dung này sẽ được chú thích hoặc hướng dẫn trực tiếp trên màn hình.

Ngoài các biến *sd*, *pa*, trên lưu đồ thuật giải còn sử dụng

- Các biến ở dữ liệu đầu vào :

F – lực kéo băng tải hoặc xích tải;

v – vận tốc băng tải hoặc xích tải;

L_h – thời hạn sử dụng thiết bị;

D – đường kính tang quay hoặc đường kính thùng trộn;

z,p – số răng đĩa xích nhỏ của xích tải và bước xích tải;

- *sc* : số cấp trong hộp giảm tốc (sc = 1; 2; 3;...);

- *lbt* : loại bộ truyền, với ký hiệu quy ước:

2 - bộ truyền bánh răng trụ;

3 - bộ truyền bánh răng côn;

4 - bộ truyền trục vít - bánh vít.

Với hộp giảm tốc đồng trục do $aw_1 = aw_2$, trong khi $T_2 \approx u_1 T_1$ nên thường tiến hành thiết kế cấp chậm trước để nhận được aw_2 và lấy $aw_1 = aw_2$ rồi tiến hành tính kiểm nghiệm bộ truyền cấp nhanh với hệ số chiều rộng răng được chọn nhỏ hơn so với cấp nhanh của sơ đồ khai triển.Với hộp giảm tốc bánh răng – trục vít do hiệu suất của bộ truyền trục vít phụ thuộc vào các thông số của bộ truyền này, từ đó ảnh hưởng đến công suất và mômen xoắn trên bộ truyền bánh răng trong hệ, do đó bộ truyền trục vít ở cấp chậm cũng cần được thiết kế trước. Vì vậy đối với hai sơ đồ 7 và 6, quá trình tính toán được lặp lại cho các bộ truyền trong hộp từ cấp chậm đến cấp nhanh, trái lại đối với các sơ đồ còn lại, quá trình tính toán lại được lặp lại cho các bộ truyền trong hộp từ cấp nhanh đến cấp chậm, như thể hiện trên hình 7.1.

5. Khi khai thác phần mềm HEDD cần lưu ý những vấn đề sau đây:

a) Chọn số đôi cực của động cơ :

Chọn số đôi cực của động cơ có ảnh hưởng lớn đến kích thước cũng như giá thành của các chi tiết máy và bộ phận máy trong hệ: cùng một công suất động cơ, số đôi cực càng tăng thì giá thành động cơ càng tăng nhưng số vòng quay đồng bộ của động cơ càng giảm, do đó với cùng một vận tốc của trục làm việc, tỉ số truyền chung của hệ sẽ giảm, dẫn đến kích thước và giá thành các bộ truyền và hộp giảm tốc sẽ giảm. Vì vậy để phối hợp hai yếu tố vừa nêu nhằm chọn được số vòng quay thích hợp cho động cơ, chương trình đã xuất phát từ sơ đồ của hệ dẫn động và khả năng thực hiện tỉ số truyền của từng bộ truyền để xác định số vòng quay sơ bộ của động cơ, làm căn cứ cho người sử dụng suy ra số đôi cực p của động cơ. Sau đó mới tiến hành tra các số liệu của loại động cơ được chọn (các bảng số liệu của ba loại động cơ

Việt – Hung , Điện cơ Hà Nội và động cơ của Nga đã được nạp vào đĩa), kèm theo phần mềm.

Tuy nhiên với cùng một bộ dữ liệu đầu vào, khi sử dụng phần mềm vẫn có thể chọn số đôi cực khác nhau để tính toán động học, thiết kế các bộ truyền, thiết kế trục và chọn ổ lăn cũng như thiết kế bản vẽ, cuối cùng có thể chọn được số đôi cực thích hợp hơn để dùng. Để tránh nhầm lẫn, trước khi bắt đầu thực hiện một trong 4 chương trình lớn đối với một số sơ đồ của hệ, phần mềm yêu cầu người sử dụng gán số đôi cực p.

b) Vấn đề phân phối tỉ số truyền:

- Do đường kính các bánh đai của bộ truyền đai là những giá trị tiêu chuẩn nên để giảm thiểu sai lệch cho tỉ số truyền, người thiết kế nên chọn u_d theo dãy số sau (tương ứng với dãy đường kính bánh đai tiêu chuẩn):

2 ; 2,24 ; 2,5 ; 2,8 ; 3,15 ; 3,56 ; 4 ; 5 ; 6

- Để tiến hành phân phối tỉ số truyền cho các cấp trong hộp giảm tốc, từ 4 phương án (*fa*) phân phối tỉ số truyền:

fa = 1 : theo công thức thực nghiệm [2] ;
fa = 2 : theo các đồ thị tương ứng [2] ;
fa = 3 : theo khối lượng nhỏ nhất đối với hộp giảm tốc bánh răng trụ, xem (3.12) [2];
fa = 4 : theo phương pháp đa mục tiêu – bảng 3.1 [2],

trong chương trình đã sử dụng các công thức và bảng tra ứng với các loại sơ đồ của hệ. Theo các hướng dẫn trên màn hình, người sử dụng cần tìm hiểu để quyết định chọn phương án nào, từ đó sẽ có được các giá trị của tỉ số truyền các cấp, phục vụ các bước tính tiếp theo.

c) Đối với *Program Botruyen*, ngoài những vấn đề đã nêu trong phần lưu đồ thuật giải, cần lưu ý thêm:

- Các bộ truyền đều được thiết kế xuất phát từ các chỉ tiêu chủ yếu về khả năng làm việc, kết hợp với việc lựa chọn tối ưu các thông số và kích thước, đồng thời ứng với mỗi dữ liệu đầu vào còn đưa ra nhiều phương án thiết kế. Người sử dụng có thể thiết kế theo một hoặc một số phương án, cuối cùng dựa vào các kết quả tính toán ứng với các phương án khác nhau để quyết định chọn phương án thích hợp nhất.

- Do số răng nguyên và đường kính bánh đai là những giá trị tiêu chuẩn nên tỉ số truyền thực tế của các bộ truyền đã được thiết kế sẽ khác với tỉ số truyền tính toán. Nhằm giảm thiểu sai số của tỉ số truyền chung toàn hệ, phần mềm HEDD đảm bảo khả năng thiết kế bộ truyền cuối cùng (là bộ truyền xích đối với phương án pa=2 và là bộ truyền bánh răng đối với pa=1 hoặc 3) với tỉ số truyền u_c, được xác định theo tỉ số truyền chung của hệ u_t và tỉ số truyền thực tế u_{tj} của các bộ truyền đã được thiết kế trước nó theo công thức sau:

$$u_c = \frac{u_t}{\prod_{j=1}^{n-1} u_{tj}}$$

$cl = 2$: dùng ổ đũa thay ổ bi;

$cl = 1$: tăng đường kính ngõng trục;

Riêng giải pháp sử dụng ổ với cỡ lớn hơn đã được kết hợp khi tra bảng thông số ổ lăn theo d và C_t.

e) Vấn đề đối thoại với máy tính :

Để tăng tốc độ tính toán, nhiều đoạn chương trình trong phần mềm HEDD đã được soạn thảo để máy tính tự động thực hiện. Tuy nhiên vẫn rất cần các đoạn đối thoại giữa người và máy nhằm thực hiện những nội dung sau đây:

- Chọn vật liệu để thiết kế chi tiết máy.

- Gán giá trị cho một thông số dựa trên một khoảng trị số được ghi trên màn hình.

- Thực hiện các nội dung hướng dẫn của phần mềm, phục vụ các bước tính tiếp theo.

- Trả lời các câu hỏi do chương trình đặt ra như : có chấp nhận kết quả tính toán hay không; có cần chọn phương án khác để tính toán nữa không; có cần giảm (hoặc tăng) đường kính trục không, v.v...

- Chọn giải pháp xử lý khi kiểm nghiệm không đạt yêu cầu.

- Quyết định phương án kết cấu, v.v...

Để giúp cho người khai thác phần mềm có thể đưa ra được những nội dung đối thoại phù hợp, trên màn hình có ghi những chú thích và hướng dẫn cần thiết. Tuy nhiên điều đó chỉ đáp ứng những nội dung đối thoại đơn giản. Muốn có những đối thoại xác đáng, người sử dụng phần mềm phải được trang bị đầy đủ nhứng kiến thức về kết cấu cơ khí và về tin học, những nội dung đã được trình bày chi tiết trong các chương trước.

TÀI LIỆU THAM KHẢO

[1]. Trịnh Chất:
Cơ sở thiết kế máy và Chi tiết máy.
Nhà xuất bản Khoa học và Kỹ thuật, In lần thứ ba, Hà nội 2001.

[2]. Trịnh Chất, Lê Văn Uyển:
Tính toán thiết kế hệ dẫn động cơ khí, tập I và II.
Nhà xuất bản Giáo dục, Tái bản lần thứ ba, Hà nội 2003.

[3]. Nguyễn Trọng Hiệp:
Chi tiết máy, tập I và II.
Nhà xuất bản Giáo dục, Hà nội 1994.

[4]. *Курсовое проектирование деталей машин*
Под. ред. Кудрявцева В.Н
Ленинград "Машиностроение", 1984

[5]. Чернавский и др.
Проектирование механических передач, 1984

[6]. Кузьмин А.В. и др.
Курсовое проектирование деталей машин,
Часть I и II, Минск, 1982

[7]. ISO 6336/1/2/3:
Calculation of load capacity of spur and helical gear
Part 1: Basic principles introduction and general influence factors
Part 2: Calculation of surface durability (pitting)
Part 3: Calculation of tooth bending strength.

[8]. W.Beitz, K.H.Küttner :
Dubbel – Taschenbuch für den Maschinen-bau, 1995.

[9] . Römhild L., Linke H.:
Gezielte Auslegung von Zahnradgetrieben mit minimale Masse auf der Basis neuer Berechnungsverfahren.
Konstruktion 44, 1992.

[10]. Trịnh Chất:
Tính toán phân phối tỉ số truyền cho các cấp trong hộp giảm tốc trục vít – bánh răng.
Tuyển tập công trình Hội nghị Cơ học toàn quốc lần thứ V, tập II, 1993.

[11]. Trịnh Chất:
Tính toán tối ưu tỉ số truyền trong hệ truyền động bánh răng.
Tuyển tập công trình khoa học nhân dịp 40 năm thành lập trường Đại học Bách khoa Hà Nội.

[12]. Trịnh Chất, Đỗ Đức Nam:
Thiết kế tối ưu hệ bánh răng côn – trụ.
Tuyển tập công trình Hội nghị Khoa học toàn quốc về Cơ học Kỹ thuật, Hà Nội 2001.

[13]. Niemann G., Winter H.:
Maschinenelement, Band I, II, III
Springer-Verlag Berlin, 1989.

[14]. *Turbo Pascal. Version 5.5*
Đỗ Phúc, Tạ Minh Châu, Trần Duy Thê *biên dịch,*
Nhà xuất bản Giáo dục.

[15]. Hoàng Hồng:
Lập trình Turbo Pascal 7.0
Nhà xuất bản Giao thông vận tải, 1999.

[16]. Nguyễn Hữu Lộc:
Sử dụng AutoCAD 2004, tập I.
Nhà xuất bản Thành phố Hồ Chí Minh, 2004.

[17]. Trần Công Nghị, Vũ Tiến Đạt:
Ngôn ngữ AutoLISP dùng trong AutoCAD.
Nhà xuất bản Thành phố Hồ Chí Minh, 1996.

[18]. *AutoCAD 2002, 2004:*
VisualLISP Developer's Guide
VisualLISP Reference Guide
ActiveX and VBA Developer's Guide
ActiveX and VBA Reference.

[19]. Trịnh Chất, Trịnh Đồng Tính, Lê Văn Uyển, Nguyễn Xuân Ngọc:
Tính toán tối ưu và thiết kế tự động hệ dẫn động cơ khí.
Tuyển tập công trình Hội nghị Khoa học toàn quốc về Cơ học Kỹ thuật, Hà Nội 2001.

[20]. Lê Văn Uyển, Trịnh Đồng Tính,Vũ Lê Huy, Đỗ Đức Nam:
Một phương pháp xây dựng và xử lý cơ sở dữ liệu trong bộ phần mềm "Tính toán thiết kế hệ dẫn động cơ khí".
Tuyển tập công trình Hội nghị Khoa học toàn quốc về Cơ học vật rắn biến dạng lần thứ 7, Đồ Sơn 2004.

MỤC LỤC

	Trang
Lời nói đầu	3
***Chương 1*: Những vấn đề cơ bản về thiết kế tự động**	5
1.1. Mục tiêu, nội dung và phương pháp thiết kế tự động	5
1.2. Phương pháp lập trình thiết kế cơ khí	9
1.3. Xây dựng dữ liệu trong thiết kế cơ khí	19
***Chương 2*: Xây dựng chương trình tính toán thiết kế truyền động cơ khí**	33
2.1. Xây dựng phần mềm tính toán thiết kế truyền động bánh răng	33
2.2. Xây dựng phần mềm tính toán thiết kế truyền động bánh răng hành tinh	46
2.3. Xây dựng phần mềm tính toán thiết kế truyền động trục vít - bánh vít	55
2.4. Xây dựng phần mềm tính toán thiết kế truyền động đai	63
2.5. Xây dựng phần mềm tính toán thiết kế truyền động xích	70
***Chương 3*: Xây dựng chương trình tính toán thiết kế trục và ổ trục**	77
3.1. Xây dựng phần mềm tính toán thiết kế trục	77
3.2. Xây dựng phần mềm tính toán chọn ổ lăn	89
3.3. Xây dựng phần mềm tính toán thiết kế ổ trượt	96
***Chương 4*: Xây dựng chương trình thiết kế hệ dẫn động cơ khí**	100
4.1. Nội dung tính toán thiết kế	100
4.2. Những vấn đề cần lưu ý khi lập trình tính toán thiết kế hệ dẫn động cơ khí	101
4.3. Thí dụ	109
***Chương 5*: Xây dựng chương trình thiết kế tự động bản vẽ**	119
5.1. Nội dung và phương pháp soạn thảo phần mềm thiết kế tự động bản vẽ	119
5.2. Tập tin đáp bản	121
5.3. Sử dụng các lệnh của AutoCAD để lập trình thiết kế tự động bản vẽ	124

Trang

5.4. Lập trình thiết kế tự động bản vẽ chế tạo chi tiết máy 139
5.5. Lập trình thiết kế tự động bản vẽ lắp 146
5.6. Lập trình thiết kế tự động bản vẽ chung 153

Chương 6: **Xây dựng chương trình thiết kế tự động bản vẽ bằng ngôn ngữ AutoLISP** 158

6.1. AutoLISP và VisualLISP 158
6.2. Các khái niệm cơ bản về ngôn ngữ AutoLISP 164
6.3. Lập trình ứng dụng bằng AutoLISP 184
6.4. Thiết kế và điều khiển hộp thoại DCL 230
6.5. Liên kết với các chương trình khác 251

Chương 7: **Thiết kế ứng dụng** 253

7.1. Mục đích, nội dung và yêu cầu thiết kế ứng dụng 253
7.2. Khai thác phần mềm HEDD 257
7.3. Viết thuyết minh 259

Phụ lục 1: **Chương trình AutoLISP vẽ bánh vít** 261

Phụ lục 2: **Phần mềm HEDD** 294

Tài liệu tham khảo 300

TỰ ĐỘNG HOÁ THIẾT KẾ CƠ KHÍ

Tác giả: PGS.TS. TRỊNH CHẤT
TS. TRỊNH ĐỒNG TÍNH

Chịu trách nhiệm xuất bản: PGS.TS. Tô Đăng Hải
Biên tập và sửa chế bản: Nguyễn Diệu Thuý
Trình bày và chế bản: Trịnh Chất, Trịnh Đồng Tính
Vẽ bìa: Hương Lan

Nhà xuất bản Khoa học và Kỹ thuật
Hà Nội 2005

In 800 cuốn 16 x 24 cm, tại Xí nghiệp in Thương mại.
Giấy phép số 6-400 do Cục xuất bản cấp ngày 5/1/2004.
In xong và nộp lưu chiểu 1/2005.

: động lực tk cơ khí
005012 600063
40.000 VND





Giá: 40.000d