§17. CHƯƠNG TRÌNH CON VÀ PHÂN LOẠI

1. Khái niệm chương trình con

Các chương trình giải các bài toán phức tạp thường rất dài, có thể gồm hàng trăm, hàng nghìn lệnh. Khi đọc những chương trình dài, rất khó nhận biết được chương trình thực hiện các công việc gì và việc hiệu chỉnh chương trình cũng khó khăn. Vì vậy, vấn đề đặt ra là phải cấu trúc chương trình như thế nào để cho chương trình dễ đọc, dễ hiệu chỉnh, dễ nâng cấp.

Mặt khác, việc giải quyết một bài toán phức tạp thường đòi hỏi và nói chung có thể phân thành các bài toán con.

Xét bài toán tính tổng bốn luỹ thừa:

$$TLuythua = a^n + b^m + c^p + d^q$$

Bài toán trên bao gồm bốn bài toán con tính a^n, b^m, c^p, d^q, có thể giao cho bốn người, mỗi người thực hiện một bài. Giá trị *TLuythua* là tổng kết quả của bốn bài toán con đó. Với những bài toán phức tạp hơn, mỗi bài toán con lại có thể được phân chia thành các bài toán con nhỏ hơn. Quá trình phân rã làm "mịn" dần bài toán như vậy được gọi là cách thiết kế từ trên xuống.

Tương tự, khi lập trình để giải bài toán trên máy tính có thể phân chia chương trình (gọi là chương trình chính) thành các khối (môđun), mỗi khối bao gồm các lệnh giải một bài toán con nào đó. Mỗi khối lệnh sẽ được xây dựng thành một *chương trình con*. Sau đó, chương trình chính sẽ được xây dựng từ các chương trình con này. Chương trình con cũng có thể được xây dựng từ các chương trình con khác.

Cách lập trình như vậy dựa trên phương pháp lập trình có cấu trúc và chương trình được xây dựng gọi là chương trình có cấu trúc.

Chương trình con là một dãy lệnh mô tả một số thao tác nhất định và có thể được thực hiện (được gọi) từ nhiều vị trí trong chương trình.

Ví dụ, chương trình nhập dữ liệu từ bàn phím, tính và đưa ra màn hình giá trị *TLuythua* được mô tả như trên với a, b, c, d có kiểu thực và m, n, p, q có kiểu nguyên có thể viết bằng Pascal như sau:

```
program tinh_tong;
var TLuythua, Luythua1, Luythua2, Luythua3, Luythua4: real;
   a,b,c,d: real;
   i,n,m,p,q:integer;
begin
   write('Hay nhap du lieu theo thu tu a,b,c,d,m,n,p,q');
   readln(a,b,c,d,m,n,p,q);
   Luythua1:=1.0;
      for i:=1 to n do
         Luythua1:=Luythua1*a;
   Luythua2:=1.0;
      for i:=1 to m do
         Luythua2:=Luythua2*b;
   Luythua3:=1.0;
      for i:=1 to p do
         Luythua3:=Luythua3*c;
   Luythua4:=1.0;
      for i:=1 to q do
         Luythua4:=Luythua4*d;
   TLuythua:= Luythua1+Luythua2+Luythua3+Luythua4;
   writeln('Tong luy thua = ', TLuythua:8:4);
   readln
end.
```

Trong chương trình trên có bốn đoạn lệnh tương tự nhau, việc lặp lại những đoạn lệnh tương tự nhau làm cho chương trình vừa dài vừa khó theo dõi. Để nâng cao hiệu quả lập trình, các ngôn ngữ lập trình bậc cao đều cung cấp khả năng xây dựng chương trình con dạng tổng quát "đại diện" cho nhiều đoạn lệnh tương tự nhau, chẳng hạn tính luỹ thừa $Luythua = x^k$, trong đó *Luythua* và x là giá trị kiểu thực còn k thuộc kiểu nguyên:

```
var j: integer;
Tich:=1.0;
   for j:=1 to k do
      Tich:=Tich*x;
```

Ta có thể đặt tên cho chương trình con này là *Luythua* và tên các biến chứa dữ liệu vào của nó là x và k. Khi cần tính luỹ thừa của những giá trị cụ thể ta chỉ

cần viết tên gọi chương trình con và thay thế (x, k) bằng giá trị cụ thể tương ứng. Chẳng hạn để tính a^n, b^m, c^p, d^q ta viết *Luythua*(a, n), *Luythua*(b, m), *Luythua*(c, p), *Luythua*(d, q).

Lợi ích của việc sử dụng chương trình con

- *Tránh được việc phải viết lặp đi lặp lại cùng một dãy lệnh* nào đó tương tự như trong ví dụ tính *TLuythua* ở trên. Ngôn ngữ lập trình cho phép tổ chức dãy lệnh đó thành một chương trình con. Sau đó, mỗi khi chương trình chính cần đến dãy lệnh này chỉ cần gọi thực hiện chương trình con đó.
- *Hỗ trợ việc thực hiện các chương trình lớn:* Khi phải viết chương trình lớn hàng nghìn, hàng vạn lệnh, cần huy động nhiều người tham gia, có thể giao cho mỗi người (hoặc mỗi nhóm) viết một chương trình con, rồi sau đó lắp ghép chúng lại thành chương trình chính. Ví dụ, với các bài toán mà việc tổ chức dữ liệu vào và ra không đơn giản thường người ta chia bài toán thành ba bài toán con như nhập, xử lí và xuất dữ liệu, rồi viết các chương trình con tương ứng.
- *Phục vụ cho quá trình trừu tượng hoá:* Người lập trình có thể sử dụng các kết quả được thực hiện bởi chương trình con mà không phải quan tâm đến việc các chương trình con đó được cài đặt như thế nào. Trừu tượng hoá là tư tưởng chủ đạo để xây dựng chương trình nói chung và chương trình có cấu trúc nói riêng.
- *Mở rộng khả năng ngôn ngữ:* Các ngôn ngữ lập trình thường cung cấp phương thức đóng gói các chương trình con nhằm cung cấp như một câu lệnh mới (tương tự như các lệnh gọi thực hiện các hàm và thủ tục chuẩn) cho người lập trình sử dụng mà không cần biết mã nguồn của nó như thế nào. Hiện nay, ngày càng có nhiều thiết bị kĩ thuật số tiện ích như máy quay phim, máy ảnh, máy ghi âm, các thiết bị âm thanh, màn hình màu độ phân giải cao,... có thể được kết nối với máy tính. Việc thiết kế những chương trình con thực hiện các giao tiếp cơ bản với các thiết bị như vậy là rất cần thiết và giúp mở rộng khả năng ứng dụng của ngôn ngữ.
- *Thuận tiện cho phát triển, nâng cấp chương trình:* Do chương trình được tạo thành từ các chương trình con nên chương trình dễ đọc, dễ hiểu, dễ kiểm tra và hiệu chỉnh. Việc nâng cấp, phát triển chương trình con nào đó, thậm chí bổ sung thêm các chương trình con mới nói chung không gây ảnh hưởng đến các chương trình con khác.

2. Phân loại và cấu trúc của chương trình con

a) Phân loại

Trong nhiều ngôn ngữ lập trình, chương trình con thường gồm hai loại:

- *Hàm* (function) là chương trình con thực hiện một số thao tác nào đó và trả về một giá trị qua tên của nó. Ví dụ hàm toán học hay hàm xử lí xâu:

  `sin(x)` nhận giá trị thực x và trả về giá trị $\sin x$,
  `sqrt(x)` nhận giá trị x và trả về giá trị căn bậc hai của x,
  `length(x)` nhận xâu x và trả về độ dài của xâu x,...

- *Thủ tục* (procedure) là chương trình con thực hiện các thao tác nhất định nhưng không trả về giá trị nào qua tên của nó. Ví dụ các thủ tục vào/ra chuẩn hay thủ tục xử lí xâu:

  `writeln`, `readln`, `delete`, `insert`,...

b) Cấu trúc chương trình con

Chương trình con có cấu trúc tương tự chương trình, nhưng nhất thiết phải có tên và phần đầu dùng để khai báo tên, nếu là hàm phải khai báo kiểu dữ liệu cho giá trị trả về của hàm:

<phần đầu>
[*<phần khai báo>*]
<phần thân>

Phần khai báo

Phần khai báo có thể có khai báo biến cho dữ liệu vào và ra, các hằng và biến dùng trong chương trình con.

Phần thân

Phần thân của chương trình con là dãy câu lệnh thực hiện để từ những dữ liệu vào ta nhận được dữ liệu ra hay kết quả mong muốn.

Tham số hình thức

Các biến được khai báo cho dữ liệu vào/ra được gọi là *tham số hình thức* của chương trình con. Các biến được khai báo để dùng riêng trong chương trình con được gọi là *biến cục bộ*.

Ví dụ, trong chương trình con *Luythua(x, k)* ở phần 1 thì x, k là các tham số hình thức và j là biến cục bộ.

Nói chung, chương trình chính và các chương trình con khác không thể sử dụng được các biến cục bộ của một chương trình con, nhưng mọi chương trình con đều sử dụng được các biến của chương trình chính. Do vậy, các biến của chương trình chính được gọi là *biến toàn cục*. Ví dụ, biến *TLuythua* khai báo trong chương trình chính ở ví dụ trên là biến toàn cục.

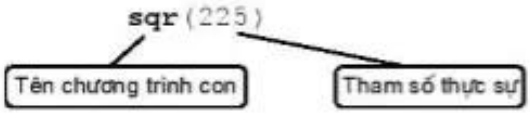
Một chương trình con có thể có hoặc không có tham số hình thức, có thể có hoặc không có biến cục bộ.

c) *Thực hiện chương trình con*

Tham số thực sự

Để thực hiện (gọi) một chương trình con, ta cần phải có lệnh gọi nó tương tự lệnh gọi hàm hay thủ tục chuẩn, bao gồm tên chương trình con với tham số (nếu có) là các hằng và biến chứa dữ liệu vào và ra tương ứng với các tham số hình thức đặt trong cặp ngoặc (và). Các hằng và biến này được gọi là các *tham số thực sự*.

Ví dụ



Khi thực hiện chương trình con, các tham số hình thức dùng để nhập dữ liệu vào sẽ nhận giá trị của tham số thực sự tương ứng, còn các tham số hình thức dùng để lưu trữ dữ liệu ra sẽ trả giá trị đó cho tham số thực sự tương ứng.

Ví dụ, khi thực hiện tính *TLuythua* cần bốn lần gọi chương trình con *Luythua(x, k)* với các tham số (a, n), (b, m), (c, p), (d, q) và các tham số này là tham số thực sự tương ứng với tham số hình thức (x, k).

Sau khi chương trình con kết thúc, lệnh tiếp theo lệnh gọi chương trình con sẽ được thực hiện.