CHƯƠNG 4 : HÀM VÀ THỦ TỤC

Mã chương: MH15-04.
Mục tiêu :

- Trình bày được khái niệm hàm, thủ tục;
- Trình bày được qui tắc xây dựng hàm, thủ tục và vận dụng được khi thiết kế xây dựng chương trình;
- Phân biệt được cách sử dụng tham số, tham biến;
- Sử dụng được các lệnh kết thúc và lấy giá trị trả về của hàm.
- Thực hiện các thao tác an toàn với máy tính.

Nội dung :

1. Khái niệm chương trình con

***Mục tiêu**: Trình bày được khái niệm hàm, thủ tục.*

Trong chương trình, có những đoạn cần phải lập đi, lập lại nhiều lần ở những chỗ khác nhau. Để tránh phải viết lại các đoạn đó người ta thường phân chương trình ra thành nhiều module, mỗi module giải quyết một công việc nào đó, các module như vậy là những chương trình con (subprogram).

Một tiện lợi khác của việc sử dụng module là ta có thể dễ dàng kiểm tra tính đúng đắn của nó trước khi ráp nối vào chương trình chính. Do đó việc xác định sai sót và tiến hành điều chỉnh trong chương trình sẽ thuận lợi hơn.

Trong Pascal chương trình con được viết dưới dạng hàm (FUNCTION) hoặc thủ tục (PROCEDURE). Hàm và thủ tục đều là những chương trình con, nhưng hàm khác thủ tục ở chỗ hàm trả về một giá trị cho lệnh gọi thông qua tên hàm còn thủ tục thì không. Do đó ta chỉ dùng hàm khi thoả mãn các yêu cầu sau :

- Ta muốn nhận một kết quả và chỉ một mà thôi.
- Ta cần dùng tên chương trình con (chứa kết quả đó) để viết trong các biểu thức.

Nếu không thỏa hai yêu cầu trên thì ta dùng thủ tục.

2. Cấu trúc chương trình có sử dụng chương trình con

***Mục tiêu**: Trình bày được cấu trúc một chương trình có sử dụng chương trình con.*

Trong một chương trình các chương trình con được bố trí ngay sau phần khai báo biến. Cấu trúc tổng quát một chương trình trong ngôn ngữ Pascal như sau:

PROGRAM tên_chương_trình;
USES tên các Unit (*Khai báo các đơn vị chương trình cần thiết*)

```
LABEL          (*Khai báo nhãn*)
CONST          (*Khai báo hằng*)
TYPE           (*Định nghĩa kiểu dữ liệu mới*)
VAR            (*Khai báo biến*)
PROCEDURE Tên_CTC1 (danh sách các tham số hình thức);
Begin
……..  (*thân thủ tục thứ nhất*)
End;

PROCEDURE Tên_CTC2 (danh sách các tham số hình thức);
Begin
……..  (*thân thủ tục thứ hai*)
End;

FUNCTION Tên_Hàm1(danh sách các tham số hình thức);
Begin
……..  (*thân hàm thứ nhất*)
End;
FUNCTION Tên_Hàm2(danh sách các tham số hình thức);
Begin
……..  (*thân hàm thứ nhất*)
End;

……..
BEGIN
……..  (*chương trình chính*)
END.
```

3. Các hàm và thủ tục trong ngôn ngữ lập trình

***Mục tiêu**: Trình bày được qui tắc xây dựng hàm, thủ tục và vận dụng được khi thiết kế xây dựng chương trình.*

3.1 Hàm

3.1.1 Cấu trúc một hàm

Hàm là một chương trình con tính toán trả về cho ta một giá trị kiểu vô hướng. Cấu trúc hàm như sau:

FUNCTION <Tên hàm>[(<Th.số>:<Kiểu>[;<Th.số>: <Kiểu>])]: <KiểuKQ>;	(Header)
[VAR <Biến>:<Kiểu>[;<Biến>: <Kiểu>]]	Khai báo các

	biến cục bộ nếu có.
BEGIN **<các câu lệnh>** END**;**	Thân hàm

Bảng 1.7. Cấu trúc của một hàm

- Tên hàm là một danh biểu, phải tuân thủ theo qui tắc đặt danh biểu đã đề cập ở chương I.
- Một hàm có thể không có hoặc có một hoặc nhiều tham số. Trong trường hợp có nhiều tham số có cùng một kiểu dữ liệu thì ta có thể viết chúng cách nhau bởi dấu **,** (phẩy). Ngược lại, các tham số hình thức khác kiểu nhau thì phải cách nhau dấu **;** (chấm phẩy).
- KiểuKQ là một kiểu vô hướng, nó phản ảnh kiểu của giá trị mà hàm trả về lại sau khi chạy xong. Ví dụ, ta khai báo hàm như sau:

FUNCTION TEST(x,y:Integer; z:Real): Real;

Đây là một hàm có tên là TEST, với 3 tham số, x và y thuộc kiểu Integer, z thuộc kiểu real, hàm trả về một kết quả kiểu real.

- Trong hàm, ta có thể sử dụng các hằng, kiểu, biến dùng riêng trong nội bộ hàm.
- Thông thường mục đích sử dụng hàm là để lấy trị trả về do đó cần lưu ý gán kết quả cho tên hàm trong thân hàm.

3.1.2 Phương pháp gọi hàm

Lời gọi hàm trong chương trình chính đươch thực hiện như sau:

Tên_hàm (danh sách các tham số thực)

3.1.3 Ví dụ

Ví dụ 1: Ta xây dựng hàm **DT** truyền tham số vào là bán kính của hình tròn, hàm này sẽ trả về diện tích của hình tròn đó.

```
Program TinhDienTich;
Uses Crt;
VAR  BanKinh: real; Ch: Char;
{-----------------------------------------}
Function DT(Radius:Real):Real;
Begin
      DT := PI * Radius* Radius;
End;
```

Phép gán để trả về giá trị cho tên hàm.

```
{----------------------------------------}

Begin
        Clrscr;
        Repeat
                Write('Nhập bán kính: '); Readln(BanKinh);
                Writeln('Diện tích hinh tron tuong ung: ' ,DT(Bankinh):0:2);
                Writeln;
                Write('Tiếp tục (C/K)? ');
                        Repeat
                                ch:=readkey;
                        Until Upcase(ch) in ['C','K'];
        Until UpCase(Ch) = 'K';  {Lưu ý: 'K' in hoa}
End.
```

Ví dụ 2:

```
Program TinhGiaithua;
USES CRT;
Var Num:longint; Ch:char; X,Y:byte;
{----------------------------------------}
Function GiaiThua(m: longint): longint;
Var Tam, Dem:Longint;
BEGIN
IF (M<0) THEN
  Begin
          Write('Khong tinh duoc'); HALT(1);
        End
ELSE
        Begin
     Tam:=1;
     For Dem:=1 to m do Tam:=Tam*Dem;
     GiaiThua:=Tam;
        End;
 END;
{------------- Chương trình chính ------------------------}
BEGIN
        Writeln('CHUONG TRINH TINH GIAI THUA.');
REPEAT
                Write('Cho so nguyen muon tinh giai thua. M= ');
                X:=WhereX; Y:=WhereY;
                REPEAT
Gotoxy(X,Y); CLREOL; Readln(Num);
                        UNTIL (Num>=0);
                Writeln(M,'! = ',GiaiThua(Num));
                REPEAT
                   Write('Tinh nua khong ? (C/K) :'); CH:=READKEY;
```

```
            UNTIL Upcase(Ch) in ['C','K'];
        Writeln(Ch);
UNTIL Upcase(Ch)='K';
Readln
END.
```

3.2 Thủ tục

3.2.1 Cấu trúc một thủ tục

Cấu trúc của một thủ tục như sau:

PROCEDURE <Tên>(<Th.số>:<Kiểu>[;<Th.số>: <Kiểu>]): <Kiểu>;	(Header)
[VAR <Biến>:<Kiểu>[;<Biến>: <Kiểu>]	Khai báo các biến cục bộ nếu có.
BEGIN **<các câu lệnh>** END**;**	Thân thủ tục.

Bảng 1.8. Cấu trúc của thủ tục

Như vậy cấu trúc của một thủ tục cũng tương tự như cấu trúc của một hàm. Chỉ có hai điều khác:

- Header bắt đầu bằng từ khóa Procedure thay vì Function.
- Không có câu lệnh gán <Tenham:=GiaTri;> trong thân Procedure.

3.2.2 Phương pháp gọi thủ tục

Lời gọi thủ tục trong chương trình chính được thực hiện như sau:

Tên_thủ tục (danh sách các tham số thực)

3.2.3 Ví dụ:

Thủ tục INSO sau sẽ in các số từ 1 đến giá trị biến truyền vào. Với ***n*** là tham số thực tế, ***So*** là tham số hình thức.

```
Program TEST;
```

```
Var n: Integer;
{---------------------------------------}
Procedure INSO(So: Integer);
Var  i: Integer;
Begin
        For i := 1 to So do
                Write( i:10 );
End;
{------------ Chương trình chính ----------------------}
Begin
        Write('Nhập một số bất kỳ lớn hơn không: '); Readln(n);
        INSO( n );
        Readln;
End.
```

3.3 Bài tập

Nội dung

- Cách thức xây dựng thủ tục trong chương trình
- Việc gọi thủ tục thực hiện trong chương trình chính
- Cách thức xây dựng hàm trong chương trình
- Việc gọi hàm thực hiện trong chương trình chính
- Cách truyền tham số cho hàm

3.3.1 Xây dựng thủ tục cho các bài tập sau

Bài 1: Tìm lỗi sai:

```
Var i, a: integer;
procedure thutuc1(int a);
void main()
{
        a:=4;
        for i:=0 to 5 do
                Ham1(a);
}
Procedure thutuc1(int a)
{
        for i:=0 to 5 do
                writeln('a= ',a);
}
```

Bài 2: Viết thủ tục kiểm tra số nguyên n có phải là **số nguyên tố** hay không?

Bài 3: Viết thủ tục kiểm tra số nguyên n có phải là **số chính phương** hay không? ($25=5^2$, $16=4^2$, $9=3^2$; các số 25, 16, 9 là những số chính phương)

Bài 4: Viết thủ tục tính n! (n là số nguyên nhập vào từ bàn phím)

Bài 5: Viết thủ tục tìm **UCLN** và **BCNN** của hai số nhập vào từ bàn phím

Hướng dẫn

Bài 1: Sinh viên tự làm
Bài 2:
❖Thuật toán:
- **Viết thủ tục Procedure ktra_ngto(int x):** kiểm tra x có phải là số nguyên tố hay không
  - Khai báo biến i, ktra
  - Cho biến đếm i chạy từ 2 đến x
    - Nếu x mod i = 0 thì thoát
  - Kiểm tra nếu i=x thì x là số nguyên tố, ngược lại thì x không phải là số nguyên tố
- **Chương trình chính**:
  - Khai báo biến n, kq
  - Nhập số n
  - Gọi thủ tục với số giá trị n vừa nhập vào: **ktra_ngto(n)**

❖Chương trình:
Bài 5:
❖Thuật toán:
- **Viết thủ tục Procedure USCLN(int a, int b):** tìm ước số chung lớn nhất của hai số a và b
  - Lấy trị tuyệt đối hai số a và b
  - Chừng nào (a!=0 và b!=0) làm
    - Nếu a>b thì a=a-b
    - Ngược lại, b=b-a
  - Nếu a = = 0 thì b là USCLN
  - Ngược lại, a là USCLN
- **Viết thủ tục Procedure BSCNN(int a, int b):** tìm bội số chung nhỏ nhất của hai số a và b
  - BSCNN= (a*b)/USCLN(a,b)
- **Chương trình chính**:
  - Khai báo biến a, b, US, BS
  - Nhập hai số a,b
  - Gọi thủ tục:
    - **USCLN(a, b)**
    - **BSCNN(a, b)**
  - Xuất BS, US ra màn hình

3.3.2 Xây dựng hàm cho các bài tập sau:

Bài 1: Viết một **hàm tính tổng các chữ số** của một số nguyên. Viết chương trình nhập vào một số nguyên, dùng hàm trên **kiểm tra** xem số **đó có chia hết cho 3** không (Một số chia hết cho 3 khi tổng các chữ số của nó chia hết cho 3).
Bài 2: Viết hàm tìm tất cả các **ước số** của số nguyên n (n nhập vào từ bàn phím). In ra các ước số đó và đếm có bao nhiêu ước số
Bài 3: Viết chương trình tính tổng sau:
- $S=1+x+x^2+x^3+\ldots+x^n$

- S= 1 - x +x^2- x^3+ ... $(-1)^n x^n$
- S=1+ 1/2!+…+1/n!

Hướng dẫn

Bài 1:

❖Thuật toán:

- **Viết hàm (function) tong_cacchuso(int n):** Tính tổng các chữ số của số nguyên n
  - Khai báo biến S, t
  - Gán S=0
  - Chừng nào n>0 làm
    - t=t%10
    - n=n/10
    - S=S+t
  - Hàm tong_cacchuso(int n) trả về giá trị S
- **Chương trình chính**:
  - Khai báo biến n, tong
  - Nhập số nguyên n từ bàn phím
  - Gọi hàm:
    - tong = **tong_cacchuso(n)**
    - Nếu tong%3= =0 thì n chia hết cho 3

❖Chương trình:

Bài 2:

❖Thuật toán:

- **Viết hàm (function) tim_uoc_so(int n):** Tìm ước số của số nguyên n
  - Khai báo biến i, count
  - Gán count=0;
  - Cho i chạy từ 1 đến n
    - Nếu n%i= = 0 thì
      - in i ra màn hình
      - count=count+1
  - Xuất count ra màn hình
- **Chương trình chính**:
  - Khai báo biến n
  - Nhập số nguyên n từ bàn phím
  - Gọi hàm: **tim_uoc_so(n)**

❖Chương trình:

Bài 3a:

❖Thuật toán:

- **Viết hàm (function) tong_luythua(int x, int n):** Tìm ước số của số nguyên n
  - Khai báo biến i, S
  - Gán S=1
  - Cho i chạy từ 1 đến n
    - S=S+pow(x,i)

- Hàm tong_luythua(int x, int n) trả về giá trị S

- **Chương trình chính**:
  - Khai báo biến x, n, tongS
  - Nhập số nguyên x, n từ bàn phím
  - Gọi hàm: tongS = **tong_luythua(x, n)**
  - Xuất tongS ra màn hình

❖Chương trình: Sinh viên tự viết chương trình dựa trên thuật toán đưa ra

Bài 3b: Sinh viên tự làm (tham khảo câu a)

Bài 3c:

❖Thuật toán:

- **Viết hàm (function) giai_thua(int n):** Tính n giai thừa (n là số nguyên dương)
  - Khai báo biến i, gth
  - Gán gth=1
  - Cho i chạy từ 1 đến n
    - gth=gth*i
  - Hàm giai_thua(int n) trả về giá trị gth
- **Chương trình chính**:
  - Khai báo biến i, n, tong_gt
  - Gán tong_gt=0
  - Nhập số nguyên n từ bàn phím
  - Cho i chạy từ 1 đến n
    - Gọi hàm: tong_gt = tong_gt+1/**giai_thua(i)**
  - Xuất tong_gt ra màn hình

❖Chương trình: Sinh viên tự viết chương trình dựa trên thuật toán đưa ra

4. Tham trị và tham biến

***Mục tiêu**: Phân biệt được cách sử dụng tham trị, tham biến*;

Các tham số của chương trình con chính là các dữ liệu cần thiết nhập vào để xử lý các phép toán trong chương trình con sử dụng. Các tham số này được gọi là tham số hình thức, bởi nó chỉ mang danh nghĩa là các đối số của chương trình con, chứ về mặt bản chất dữ liệu nó lại mang thông tin của các biến trong chương trình chính. Các tham số này có 2 loại: Tham biến và Tham trị. Các chương trình con có thể có nhiều loại tham số hình thức khác nhau về kiểu tham số hay về kiểu dữ liệu của tham số.

4.1 Tham biến

Tham biến: Là loại tham số hình thức mà giá trị của nó có thể thay đổi được trong các phép xử lý tính toán của chương trình con. Có thể dữ liệu nạp vào chương trình con là A, nhưng sau khi ra khỏi chương trình con (kết quả sau khi thực hiện chương trình con) nó lại mang kết quả B. Tham biến là tham số hình thức được khai báo ở chương trình con và bắt buộc phải được khai báo với từ kháo khai báo VAR. Các chương trình con có thể có nhiều loại tham biến, và

cách khai báo các tham biến giống hệt như bạn khai báo biến trong chương trình chính.

4.2 Tham trị

Tham trị: Là loại tham số hình thức mà giá trị của nó không thể thay đổi được trong các phép xử lý tính toán của chương trình con. Dữ liệu nạp vào chương trình con là A, nhưng sau khi ra khỏi chương trình con (kết quả sau khi thực hiện chương trình con) nó vẫn phải là A. Chính vì vậy, trong chương trình con bạn không thể nào thực hiện 1 phép toán làm thay đổi giá trị của tham trị, nếu có máy sẽ báo lỗi.Tham trị là tham số hình thức được khai báo ở chương trình con và không bắt buộc phải được khai báo với từ kháo khai báo VAR. Các chương trình con có thể có nhiều loại tham trị, và cách khai báo các tham trị giống hệt như bạn khai báo biến trong chương trình chính.

4.3 Bài tập

1. Trong chương trình dưới đây, thủ tục TT có hai tham số a và b : a là tham số trị còn b là tham số biến. Hãy cho biết giá trị của hai biến x, y của chương trình chính trước và sau khi gọi thủ tục TT:

```
PROGRAM VIDU13_1;
Var x, y: Integer;
PROCEDURE TT( a : integer ; Var b:integer);
Begin
a:=a+6;
b:=b+8;
Writeln('a= ', a);
Writeln('b= ', b);
End;

BEGIN
x:=4;
y:=7;
TT(x,y); {14}
Writeln('x= ', x); {15}
Writeln('y= ', y); {16}
Readln;
END.
```

2. Chương trình sau nhập vào hai cạnh a, b của hình chữ nhật và sử dụng một thủ tụcđể tính cả diện tích và chu vi.

3. Viết chương trình hoán đổi giá trị hai số nguyên a,b.

Yêu cầu: Sử dụng theo hai cách như sau:

- Tham số hình thức là tham trị