CHƯƠNG 4: HÀM

4.1. Ví dụ về một chương trình có hai hàm

Tính và đưa ra bình phương của 10 số nguyên dương bắt đầu từ 1:

```
#include <stdio.h>
#include <conio.h>
int  binhphuong ( int ) ;  //  nguyên mẫu hàm
void   main( )
{
    int i;
    clrscr();
    for(i = 1; i<=10; i++)
      printf("\n%d ", binhphuong(i));
      getch();
}
// Dinh nghia ham binhphuong
int binhphuong (int n)
{  return   n*n ; }
```

CHƯƠNG 4: HÀM

Chú ý:

- Không nhất thiết phải khai báo nguyên mẫu (prototype) của hàm. Nhưng nên có vì nó cho phép chương trình dịch phát hiện lỗi.

- Nguyên mẫu (prototype) của hàm thực chất là dòng đầu của hàm và thêm vào dấu chấm phẩy ";", tuy nhiên trong nguyên mẫu có thể bỏ tên các đối số.

4.2. Tổng quát về định nghĩa hàm

```
[ < kiểu giá trị trả về > ]   <tên-hàm> ( [ <danh sách tham số> ] )
{
      < các khai báo >
      .....
      < các câu lệnh >
}
```

CHƯƠNG 4: HÀM

Ví dụ:

float ham1 (float t[20]) { ..... }	Hàm trả về giá trị là một số thực. Tham số truyền cho hàm là một mảng 20 thành phần là các số thực.
float ham2 (float *t) { ..... }	Hàm trả về giá trị là một số thực. Tham số truyền cho hàm là một biến mảng thực hoặc một con trỏ thực.
float *ham3 (float *t) { ..... }	Hàm trả về giá trị là một con trỏ thực. Tham số truyền cho hàm là một biến mảng thực hoặc một con trỏ thực.
float ham4 (int a , int b) { ..... }	Hàm trả về giá trị là một số thực. Tham số truyền cho hàm là 2 số nguyên.

CHƯƠNG 4: HÀM

Quy tắc hoạt động của hàm:

- Lời gọi hàm có dạng sau:

```
tên-hàm ([ danh sách tham số thực ]);
```

- Chú ý là số tham số thực phải bằng số tham số hình thức (đối) và mỗi mỗi tham số thực phải có cùng kiểu giá trị với tham số hình thức tương ứng của nó.

Quá trình thực hiện hàm như sau:

- Cấp phát bộ nhớ cho các đối và các biến cục bộ.
- Gán giá trị của các tham số thực cho các đối tương ứng.
- Thực hiện các câu lệnh trong thân hàm.
- Khi gặp câu lệnh *return* hoặc dấu "}" cuối cùng của thân hàm thì máy sẽ xóa các đối, các biến cục bộ và thóat khỏi hàm.

CHƯƠNG 4: HÀM

4.3. Tham số trong lời gọi hàm

4.3.1. Một vài khái niệm

- **Tham số hình thức**: Là các tham số được khai báo trong phần *danh sách tham số* trong định nghĩa hàm.

- **Tham số thực sự**: Là các thông tin được truyền cho hàm trong các lời gọi hàm. Mỗi tham số thực tương ứng với một tham số hình thức.

- **Truyền theo tham trị**: một bản sao giá trị của tham số thực được tạo ra và gán cho các tham số hình thức của hàm.

- **Truyền theo tham biến**: hàm gọi sẽ truyền trực tiếp tham số đó (*bắt buộc phải là biến*) cho hàm được gọi.

CHƯƠNG 4: HÀM

4.3.2. Truyền tham số

- Việc truyền tham số trong C được thực hiện theo một kiểu duy nhất: truyền giá trị, nghĩa là giá trị của tham số thực sự trước và sau khi gọi hàm là không thay đổi.

4.3.3. Tham số hình thức của hàm là con trỏ

- Khi tham số hình thức của hàm là con trỏ thì tham số thực sự tương ứng phải là một địa chỉ, có thể là địa chỉ của một biến hoặc tên của một biến mảng.
- Khi truyền địa chỉ của biến, ta có thể thay đổi giá trị của biến.
- Khi truyền tên của mảng thì sẽ có 2 cái lợi so với việc khai báo tham số hình thức như một mảng:

+ Tiết kiệm thời gian.

+ Có thể thay đổi giá trị của các phần tử trong biến mảng.

CHƯƠNG 4: HÀM

Ví dụ: Nhập, sắp xếp và in dãy số nguyên.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int  n, a[100];
    do {
       printf("Nhap so phan tu cua day so n = ");
       scanf("%d",&n); }
    while(n<=0 | |  n > 100);
    printf("Nhap vao day so\n");
    NhapDaySo(a,n);
    printf("\nDay so vua nhap:\n");
    InDaySo(a,n);
    SapXepDaySo(a,n);
    printf("\nDay so sau khi da sap xep:\n");
    InDaySo(a,n);
    printf("\nAn 1 phim bat ky de ket thuc...");
    getch( );
}
```

CHƯƠNG 4: HÀM

```
void NhapDaySo(int *d, int n)
{
   int i;
   for(i=0; i<n; i++)
      {
       printf("Phan tu thu %d: ",i+1);
       scanf("%d",&d[i]);
      }
}
//-----------------------------------
void InDaySo(int *d, int n)
{
   int i;
   for(i=0; i<n; i++)
       printf("%4d ",d[i]);
       printf("\n");
}
```

CHƯƠNG 4: HÀM

```
void SapXepDaySo(int *d, int n)
{
   int i, j, temp;
   for(i = 0; i < n-1; i++)
      for(j = i+1; j < n; j++)
        if(d[i]>d[j])
                    {
                           temp = d[i];
                           d[i] = d[j];
                           d[j] = temp;
                           }
}
```

CHƯƠNG 4: HÀM

4.3.4. Biến tòan cục, biến cục bộ

- Biến toàn cục là biến đươc sử dụng ở mọi nơi trong chương trình, biến toàn cục được khai báo ở đâu chương trình, nằm ngoài các hàm.

- Biến cục bộ (biến địa phương) là biến chỉ có giá trị trong thời gian hàm hoạt động, sau khi hàm kết thúc thì những biến khai báo bên trọng hàm đó cũng như các tham số của nó cũng kết thúc luôn.

- Biến cục bộ tĩnh static:

Cú pháp: static < khai báo biến > ;

Khác với biến cục bộ thông thường, biến tĩnh cục bộ *static* vẫn duy trì được giá trị của chúng giữa mỗi lần gọi hàm. Các biến tĩnh cục bộ *static* được sử dụng trong việc viết những hàm cần phải bảo toàn một số giá trị giữa mỗi lần gọi.

CHƯƠNG 4: HÀM

4.4. Con trỏ tới hàm

4.4.1. Cách khai báo con trỏ hàm và mảng con trỏ hàm

Hàm cũng có địa chỉ, biến chứa địa chỉ của hàm được gọi là con trỏ hàm. Khai báo con trỏ hàm như sau:

```
<kiểu dữ liệu> (*<tên trỏ hàm>) (<d/sách kiểu d/liệu tham số>);
```

4.4.2. Tác dụng của con trỏ hàm

Con trỏ hàm dùng để chứa địa chỉ của hàm. Có thể thực hiện phép gán tên hàm cho con trỏ hàm. Để phép gán có nghĩa thì kiểu hàm và kiểu con trỏ phải giống nhau. Sau phép gán, ta có thể dùng tên con trỏ hàm thay cho tên hàm.

CHƯƠNG 4: HÀM

4.4.3. Đối là con trỏ hàm

C cho phép thiết kế các hàm mà tham số thực sự trong lời gọi hàm lại là tên của một hàm khác. Khi đó tham số hình thức tương ứng phải là một con trỏ hàm.

Ví dụ: Lập hàm tính $\int_a^b f(x)\,dx$ theo phương pháp hình thang

bằng cách chia [a,b] thành 1000 khoảng có độ dài như nhau. Sau đó dùng hàm trên để tính:

$$S_1 = \int_0^{\pi/2} \sin x\,dx \qquad S_2 = \int_0^{\pi/2} \cos x\,dx$$

$$S_3 = \int_0^{1} e^x\,dx \qquad S_4 = g(x) = \int_{-1,2}^{3,5} \frac{e^x - 2\sin x}{1 + x^4}\,dx$$

CHƯƠNG 4: HÀM

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
// Hàm tính tích phân có đối là con trỏ hàm
double  tichphan (double (*f)(double), double a, double b)
{
    int   i , n = 1000 ;
    double  s , h = (b-a)/n ;
    s = (f(a)+f(b))/2. ;
    for (i=1; i<n; ++i) s += f(a + i*h) ;
    return  h*s;
}
// Hàm g(x)
double   g(double x)
{  return (exp(x) - 2.0*sin(x)) / (1.0 + pow(x,4)) ; }
```

CHƯƠNG 4: HÀM

```
// Hàm main
void main()
{
    printf("\nS1 = %f",tichphan(sin,0., M_PI/2.));    // M_PI
la pi
    printf("\nS2 = %f",tichphan(cos, 0., M_PI/2.));
    printf("\nS3 = %f",tichphan(exp, 0., 1.));
    printf("\nS4 = %f",tichphan(g, -1.2, 3.5));
    getch();
}
```