Hàm – Con trỏ

Thời lượng: 6 tiết

Người soạn: Lương Thái Lê.

Hàm (chương trình con)

Khái niệm Hàm

- Hàm là một đoạn chương trình (có cấu trúc nhất định) thực hiện một phần việc nào đó (hay còn gọi là modul công việc) trong một chương trình lớn trọn vẹn.
- Hàm có thể được xây dựng sẵn (trong các thư viện), hoặc được xây dựng trực tiếp ngay trong chương trình.
- Ví dụ 1: (Hàm có sẵn)

```
#include <stdio.h>
#include <math.h>
main()
{
    float a;
   printf("Nhap a ="); scanf("%f",&a);
   printf("Can bac ba cua a = %.1f",pow(a,1.0/3));
}
```

Ví dụ về Hàm

- #include <stdio.h>

```
/* Dinh nghia ham max3s */
float max3s(float a, float b, float c)
{
float max;
max = a>b?a:b;
return (max>c?max:c); /* Gia tri tra ve */
}


main()
{
float x, y, z;
printf("\nNhap ba so thuc: ");
scanf("%f%f%f", &x, &y, &z);
printf("\nx = %.2f\ny = %.2f\nz= %.2f\nmax = %0.2f",
x, y, z, max3s(x, y, z));
}
```

Một số quy tắc về Hàm

- Mỗi hàm có 1 tên. Trong 1 chương trình, các hàm khác nhau phải có tên khác nhau.
- Hàm có thể có một vài đối hoặc không có đối
  - max3s(a,b,c);
  - main();
- Hàm thường trả cho ta một giá trị nào đó hoặc không có giá trị trả về:
  - giá trị trả về đó có thể có kiểu *int, float,* …
  - nếu không có giá trị trả về thì kiểu trả về của hàm là *void*
- Hàm là một đơn vị độc lập. Không được xây dựng 1 hàm trong 1 hàm khác

Định nghĩa Hàm trong C

Cấu trúc của một hàm trong C như sau:

```
kiểu_trả_về    tên_hàm(danh sách tham số hình thức)
{
        các_câu_lệnh
        return [Biểu_thức_có_kiểu_là_kiểu_trả_về];
}
```

- danh sách tham số hình thức (của N biến) được viết như sau:

  kiểu_dữ_liệu var1, **kiểu_dữ_liệu** var2, ..., **kiểu_dữ_liệu** varN

- Nếu **kiểu_trả_về** là **void** thì câu lệnh trả về sẽ là **return;** hoặc không có câu lệnh **return;**
- *Chú ý:* ***tham số hình thức*** *còn được gọi là* ***đối số***

Ví dụ về định nghĩa Hàm

```
#include <stdio.h>
 /* Dinh nghia ham max3s */
float max3s(float a, float b, float c)
{
float max;
max = a>b?a:b;
return (max>c?max:c); /* Gia tri tra ve */
}

main()
{
float x, y, z;
printf("\nNhap ba so thuc: ");
scanf("%f%f%f", &x, &y, &z);
printf("\nx = %.2f\ny = %.2f\nz= %.2f\nmax = %0.2f",
x, y, z, max3s(x, y, z));
}
```

Sử dụng Hàm (gọi hàm)

- Hàm sau khi được định nghĩa được sử dụng bằng cách gọi hàm:

  tên_hàm ([danh_sách_các_tham_số_thực_ sự])

- ***Tham số thực sự*** : có thể là các giá trị hoặc các biến có giá trị tương ứng với các ***tham số hình thức*** trong định nghĩa của hàm.
  - Mỗi tham số thực sự được phân cách nhau bằng dấu phẩy.
  - Có bao nhiêu tham số hình thức thì có bấy nhiêu tham số thực sự tương ứng.
- *Chú ý*:
  - **Hàm phải được định nghĩa trước khi sử dụng.**
  - ***Tham số thực sự*** còn được gọi là ***đối số thực sự***

Ví dụ về sử dụng Hàm

```
#include <stdio.h>
/* Dinh nghia ham max3s */
float max3s(float a, float b, float c)
{
float max;
max = a>b?a:b;
return (max>c?max:c); /* Gia tri tra ve */
}

main()
{
float x, y, z;
printf("\nNhap ba so thuc: ");
scanf("%f%f%f", &x, &y, &z);
printf("\nx = %.2f\ny = %.2f\nz= %.2f\nmax = %0.2f",
x, y, z, max3s(x, y, z));
}
```

Cách hoạt động của hàm:

- 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 xoá các đối, các biến cục bộ (giải phóng bộ nhớ của các đối, biến cục bộ) và thoát khỏi hàm.

Biến toàn cục và Biến cục bộ

- ***Biến toàn cục*** : là các biến được khai báo ở đầu chương trình (không nằm trong hàm nào), nó có thể sử dụng ở vị trí bất kỳ trong chương trình.

- ***Biến cục bộ (biến địa phương)***: là các biến được khai báo trong các hàm, tại đầu mỗi khối lệnh.
  - Phạm vi hoạt động của các biến này là trong thân hàm hay khối lệnh chứa nó.
  - Tham số thực sự và tham số hình thức của một hàm được coi là biến cục bộ của hàm đó

- Biến toàn cục và biến cục bộ có thể có cùng tên, nhưng khi ở trong một hàm thì biến cục bộ được ưu tiên hơn.

Ví dụ về sử dụng Hàm

```
#include <stdio.h>
 float k,x=5;

/* Dinh nghia ham max3s */
float max3s(float a, float b, float c)
{
float max;
max = a>b?a:b;
return (max>c?max:c); /* Gia tri tra ve */
}


main()
{
float x, y, z;
printf("\nNhap ba so thuc: ");
scanf("%f%f%f", &x, &y, &z);
printf("\nx = %.2f\ny = %.2f\nz= %.2f\nmax = %0.2f",
x, y, z, max3s(x, y, z));
}
```

Biến toàn cục

Tham số hình thức (Đối số)

Biến cục bộ

Biến cục bộ

Con trỏ (kiểu của tham số)

Khái niệm Con trỏ

- ***Con trỏ là biến dùng để chứa địa chỉ***
  - giá trị của con trỏ là địa chỉ của 1 biến nào đó
  - địa chỉ của biến là số thứ tự của byte đầu tiên của vùng nhớ của biến.

  => con trỏ cung cấp cách thức truy cập dữ liệu gián tiếp thông qua địa chỉ biến
- ***Kiểu của con trỏ***: tương ứng với kiểu của biến mà nó trỏ tới:
  - int, float…

Khai báo Con trỏ

- Khai báo con trỏ:

kiểu_con_trỏ * tên_con_trỏ

- Ví dụ:

```
int   x, y, *px, *py;
float  f, *pf;
```

- Khi đó ta có thể gán giá trị cho con trỏ:

tên_con_trỏ = &tên_biến

```
px = &x;  =>gán địa chỉ của x cho con trỏ px
pf = &f;   =>gán địa chỉ của f cho con trỏ pf
```

- Không được gán địa chỉ của biến nguyên cho con trỏ kiểu thực.

Sử dụng Con trỏ

- tên_con_trỏ chỉ giá trị của con trỏ (tức là địa chỉ của biến mà con trỏ trỏ tới)
- &tên_con_trỏ chỉ địa chỉ của biến con trỏ
- * tên_con_trỏ chỉ giá trị của biến mà con trỏ trỏ tới

- Ví dụ:

```
int x = 5,y;
int *px = &x, *py;
```

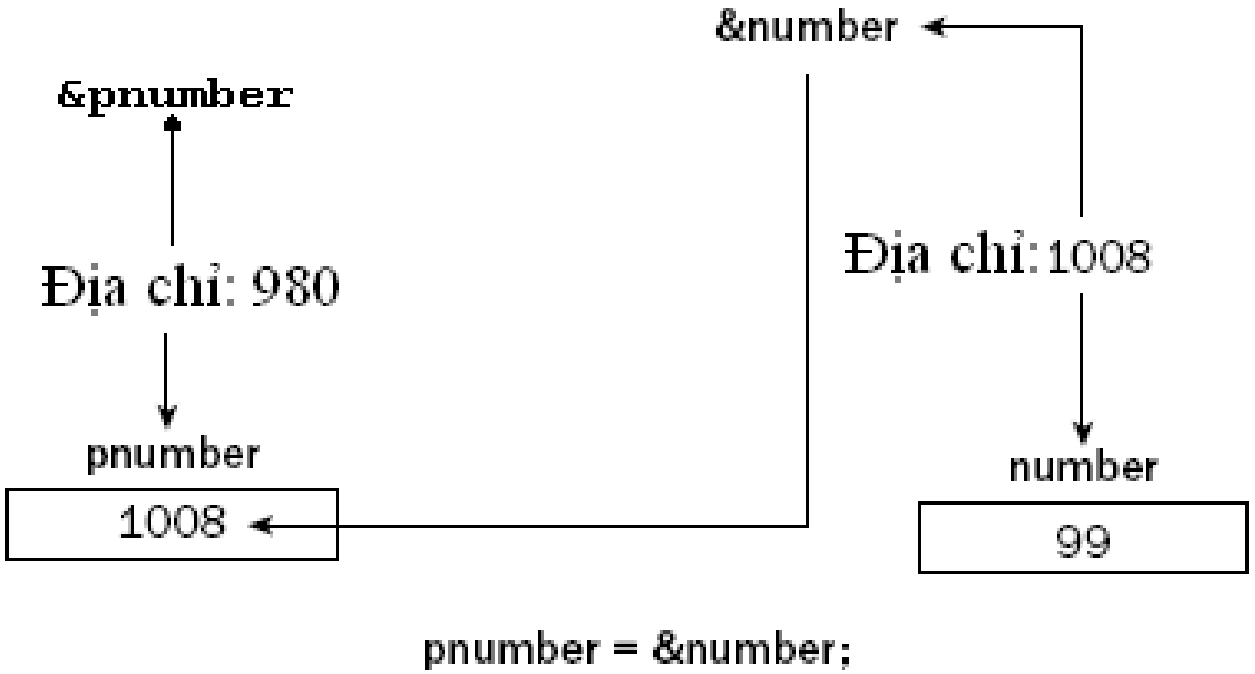
Khi đó:
- Khi viết px thì tương đương với &x
- Khi viết *px thì tương đương với x.

Các câu lệnh sau tương đương nhau:

```
y = 3*x + 1;
*py = 3*x + 1;
*py = 3*(*px) + 1;
```

Ví dụ minh họa

```
int *pnumber, number = 99;
pnumber = &number;              /*lưu địa chỉ của number trong pnumber*/
```



***pnumber = 99**

Mảng và Con trỏ

- Tên mảng là địa chỉ phần tử đầu tiên của mảng
- Ví dụ: `int a[100];`

  $\Rightarrow$a = &a[0]; a+i = &a[i]; *(a+i) = a[i]
- Do vậy có thể gán tên mảng cho 1 con trỏ:

```
int *p;
p=a;
```

Khi đó :

- a[5]=p[5]=*(a+5)=*(p+5) đều biểu diễn phần tử thứ 5 của mảng
- a[0] = *a = *p đều biểu diễn phần tử đầu tiên của mảng

Ví dụ về mảng và con trỏ

```c
#include<stdio.h>
main()
{
float a[4];
int i;
for (i=0;i<4;i++)
    {
    printf("\na[%d] = ",i);
    scanf("%f",a+i);
    }
}
```

```c
#include<stdio.h>
main()
{
float a[4];
int i;
for (i=0;i<4;i++)
    {
    printf("\na[%d] = ",i);
    scanf("%f",&a[i]);
    }
}
```

```c
#include<stdio.h>
main()
{
float a[4],s,*pa;
int i;
pa=a;
for (i=0;i<4;i++)
    {
    printf("\na[%d] = ",i);
    scanf("%f",&pa[i]);
    }
s=0;
for (i=0;i<4;i++)
    s + = pa[i]);
printf("\nTong = %.2f",s);
}
```

```c
#include<stdio.h>
main()
{
float a[4],s,*pa;
int i;
pa=a;
for (i=0;i<4;i++)
    {
    printf("\na[%d] = ",i);
    scanf("%f",pa+i);
    }
s=0;
for (i=0;i<4;i++)
    s + = *(pa+i);
printf("\nTong = %.2f",s);
}
```

Hàm có đối là Con trỏ

Thường được sử dụng trong các trường hợp sau:

1. Sử dụng hàm để thay đổi giá trị của một biến
2. Truyền một mảng vào cho hàm
3. Trả về nhiều kết quả

Sử dụng hàm để thay đổi giá trị của một biến

```
#include <stdio.h>
void hoan_vi(float *px, float *py)
{
float z;
z = *px;
*px = *py;
*py = z;
}
void main()
{
float a = 7.6, b = 13.5;
hoan_vi(&a, &b);
printf("\na = %0.2f b = %0.2f", a, b);
}
```

Đối của hàm là con trỏ

Địa chỉ của biến a được gán cho con trỏ px : px=&a
Địa chỉ của biến b được gán cho con trỏ py: py=&b

Chú ý:
- Nếu không sử dụng đối là con trỏ thì sự thay đổi giá trị của đối ở bên trong hàm không làm thay đổi giá trị các biến truyền vào cho hàm. Thử code với trường hợp không dùng đối là con trỏ để so sánh

Truyền mảng 1 chiều vào cho hàm

- *Bài toán*: Xây dựng hàm để nhập dữ liệu cho mảng một chiều, và truyền dữ liệu vừa nhập ra ngoài hàm.

$\Rightarrow$phải truyền mảng vào cho hàm qua đối số

$\Rightarrow$có thể dùng đối là con trỏ

- Đối con trỏ phải có cùng kiểu dữ liệu với mảng tương ứng

Ví dụ truyền mảng 1 chiều cho hàm

Sửa chương trình để:

- Dùng biến mảng để truyền vào cho hàm. So sánh với cách dùng biến con trỏ
- Nhập thêm dãy b chứa m phần tử. Tính tổng dãy a, tích dãy b.
- Viết thêm hàm sắp xếp dãy giảm dần

```c
#include<stdio.h>
void Nhapdayso(int *x, int n)
{
    int i;
    for (i=0;i<n;i++)
        {
            printf("Phan tu thu %d = ",i+1);
            scanf("%d",&x[i]);
        }
}
void Indayso(int *x, int n)
{
    int i;
    for (i=0;i<n;i++)
        printf ("%d   ",x[i]);
}
main()
{
int n,a[30],i;
printf("Nhap n= "); scanf("%d",&n);
Nhapdayso(a,n);
Indayso(a,n);
}
```

Truyền mảng nhiều chiều cho hàm

```c
#include<stdio.h>

void NhapMaTran(float a[50][50],int m,int n)
{
    int i,j;
    for (i=0;i<m;i++)
        for (j=0;j<n;j++)
            {
                printf("a[%d][%d] = ",i+1,j+1);
                scanf("%f",&a[i][j]);
            }
}
void InMaTran(float a[50][50],int m,int n)
{
    int i,j;
    for (i=0;i<m;i++)
        {
            for (j=0;j<n;j++)
                printf("%.2f  ",a[i][j]);
            printf("\n");
        }
}
```

```c
main()
{
    float a[50][50],tg;
    int n,m;
    printf("Nhap so hang va so cot cua ma tran m,n= ");
    scanf("%d%d",&m,&n);
    NhapMaTran(a,m,n);
    InMaTran(a,m,n);
}
```

Chú ý: Có thể dùng con trỏ để truyền mảng vào cho hàm

Trả về nhiều giá trị

- Bài toán: Giải phương trình bậc 2 sử dụng hàm.
- Khi đó, ta phải trả về:
  - số nghiệm của phương trình
  - giá trị của nghiệm

$\Rightarrow$dùng con trỏ để truyền tham số cho hàm

$\Rightarrow$lập trình

Ví dụ: Giải phương trình bậc 2

```c
#include<stdio.h>
#include<math.h>

int GPTB2(float a,float b, float c, float *px1, float *px2)
{
    float delta;
    delta = b*b-4*a*c;
    if (delta<0) return 0;
    else if (delta==0)
        {
            *px1=-b/(2*a);
            return 1;
        }
    else
        {
            *px1=-b-sqrt(delta)/(2*a);
            *px1=-b+sqrt(delta)/(2*a);
            return 2;
        }
}
```

Hàm đệ quy

Ví dụ: Viết chương trình tính n!

- Cách không đệ quy:

```
long giai_thua(int n)
{
long s=1;
int i;
for (i = 1; i <= n; i++)s*= i;
return s;
}
```

- Cách đệ quy:

  n! = 1 nếu n=0

  n! = (n-1)!*n nếu n>0

Cách xây dựng hàm đệ quy trong C

- Hàm đệ quy thường được viết theo thuật toán sau:

- Kiểu Tên-hàm(Danh sách tham số hình thức)

```
{
  if (trường hợp suy biến)
  {
  trình bày cách giải bài toán
  (giả định đã có cách giải)
  }
  else /* trường hợp tổng quát */
  {
  gọi đệ quy tới hàm (đang lập) với
  giá trị khác của tham số
  }

}
```

- Từ 1 vị trí trong thân 1 hàm gọi đến chính hàm đó
- Khi hàm gọi đệ quy đến chính nó thì mỗi lần gọi, máy sẽ tạo ra một tập các biến cục bộ mới hoàn toàn độc lập với các tập biến (cục bộ) đã được tạo ra trong các lần gọi trước.

Ví dụ: Tính n! sử dụng hàm đệ quy

```
#include <stdio.h>
long giai_thua(int n)
{
if (n==0) return 1;
else return n*giai_thua(n - 1);
}
void main()
{
printf("%d", giai_thua(5));
}
```

Dùng hàm đệ quy khi nào?

- Bài toán dễ dàng giải quyết trong một số trường hợp riêng ứng với các giá trị đặc biệt của tham số. Ta gọi đây là trường hợp suy biến.
- Trong trường hợp tổng quát, bài toán có thể quy về một bài toán cùng dạng nhưng giá trị tham số thay đổi. Và sau một số hữu hạn bước biến đổi đệ quy, sẽ dẫn tới trường hợp suy biến.

- Một số ví dụ:
  1. Tìm USCLN của 2 số nguyên dương a,b
  2. Tìm x^n với x là số thực, n là số nguyên dương.