Chương VI

HÀM

CBGD: ThS.Trần Anh Dũng

1

KHÁI NIỆM HÀM

- Bất kỳ ngôn ngữ lập trình nào cũng đều có khái niệm chương trình con (subroutine), mỗi chương trình con như vậy sẽ đảm nhận thực hiện một thao tác nhất định.
- Đối với C, chương trình con chỉ ở một dạng là hàm (**function**), không có khái niệm thủ tục (procedure).
- Nếu các ngôn ngữ khác, như Pascal, sẽ gọi hàm trong chương trình chính và sử dụng hàm thì đối với C, chương trình chính cũng là một hàm, đó là hàm main (). Hàm main () là hàm đặc biệt của C.
- Việc sử dụng hàm trong C sẽ làm cho chương trình trở nên rất dễ quản lý, dễ sửa sai
- Tất cả các hàm trong C đều ngang cấp nhau, các hàm đều có thể gọi lẫn nhau

CBGD: ThS.Trần Anh Dũng

2

KHÁI NIỆM HÀM

Ví dụ: Ví dụ 6.1/p148

Chương trình 1 → **Không dùng hàm**

Chương trình 2 → **Dùng hàm**

3

KHÁI NIỆM HÀM

Chương trình 1

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
main ()
{
    double a, b, c, delta, n1, n2;
    clrscr();
    printf ("Nhap 3 he so phuong trinh bac hai; ");
    scanf ("%lf %lf %lf", &a, &b, &c);
if (a ==0)
    /* phuong trinh suy bien ve bac nhat */
    {
        printf ("Phuong trinh suy bien ve bac nhat va ");
        if (b == 0)
            if (c == 0)
                printf ("vo so nghiem\n");
            else /* c != 0 */
                printf ("vo nghiem\n");
        else / * b != 0 */
```


4

KHÁI NIỆM HÀM

Chương trình 1

```
        {
            n1 = -c/b;
            printf ("co 1 nghiem: = %5.2f \n", n1);
        }
    }
else /* a != 0 */
    {
        printf ("Phuong trinh bac hai va ");
        delta = b*b - 4*a*c;
        if (delta < 0)
            printf ("vo nghiem thuc\n");
        else if (delta == 0)
            {
                n1 = n2 = -b/2/a;
                printf ("co nghiem kep x1 = x2 = %5.2f \n" ,n1);
            }
        else /* delta > 0 */
            {
                n1 = (-b + sqrt(delta))/2/a;
                n2 = (-b - sqrt(delta))/2/a;
                printf ("co hai nghiem phan biet; \n");
                printf ("x1 = %5.2f \n", n1);
            printf ( x2 = %5.2f \n", n2);
            }
        }
    getch();
}
```


KHÁI NIỆM HÀM

Chương trình 2

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
void gptb1 (double a, double b);
void gptb2 (double a, double b, double c);
void gptb1 (double a, double b)
    {
        printf ("Phuong trinh suy bien ve bac nhat va ");
        if (a == 0)
            if (b == 0)
                printf ("vo so nghiem\n");
            else /* b != 0 */
                printf ("vo nghiem\n");
        else
            printf ("co 1 nghiem: x = %5,2f \n", b/a);
    }
```

KHÁI NIỆM HÀM

Chương trình 2

```
void gptb2 (double a,double b,double c)
    {
        double delta, x1, x2;
        printf ("Phuong trinh bac hai va ");
        delta = b*b - 4*a*c;
        if (delta < 0)
            printf ("vo nghiem thuc\n");
        else if (delta == 0)
            printf ("co nghiem kep x1 = x2 = %5.2f \n", -b/2/a);
        else /* delta > 0 */
            {
                x1 = (-b + sqrt(delta))/2/a;
                x2 = (-b - sqrt(delta))/2/a;
                printf ("co hai nghiem phan biet: \n");
                printf ("x1 = %5.2f \n ", x1);
                printf ("x2 = %5.2f \n" , x2);
            }
        }
```


KHÁI NIỆM HÀM

Chương trình 2

```
main()
    {
        double a, b, c;
        clrscr();
        printf ("Nhap 3 he so phuong trinh bac hai: ");
        scant ("%lf %lf %lf", &a, &b, &c);
        if (a == 0)
            /* phuong trinh suy bien ve bac nhat */
            gptb1 (b, c);
        else /* a != 0 */
            gptb2 (a, b, c);
        getch();
    }
```

KHAI BÁO HÀM

Khai báo một hàm có nghĩa:

- Chỉ ra rõ rằng trả về vị trí kiểu gì
- Đối số đưa vào cho hàm có bao nhiêu đối số, mỗi đối số có kiểu như thế nào
- Các lệnh bên trong thân hàm xác định thao tác của hàm.

Có hai loại hàm:

- Hàm trong thư viện của C
- Hàm do lập trình viên tự định nghĩa.

- Nếu hàm sử dụng là hàm chuẩn trong thư viện thì việc khai báo hàm chỉ đơn giản là khai báo prototype của hàm, các prototype này đã được phân loại và ở trong các file .h, lập trình viên cần ra lệnh **#include** bao hàm các file này vào chương trình hoặc module chương trình sử dụng nó.

- Nếu các hàm sử dụng là do lập trình viên tự định nghĩa thì việc khai báo hàm bao gồm hai việc: khai báo prototype của hàm đầu chương trình và định nghĩa các lệnh bên trong thân hàm (hay thường được gọi tắt là định nghĩa hàm).

9

KHAI BÁO HÀM

Dạng 1:

```
kiểu tên_hàm (danh_sách_khai_báo_đối_số)      → đầu hàm
    {
        khai_báo_biến_cục_bộ                   → thân hàm
        lệnh
    }
```

Dạng 2

```
kiểu tên_hàm (danh_sách_đối_số)
khai_báo_đối_số
    {
        khai_báo_biến_cục_bộ
        lệnh
    }
```


10

KHAI BÁO HÀM

Ví dụ Xét hai dạng của hàm so sánh hai số sau:

Dạng 1:

```
int so_sanh (int a, int b)      → phần đầu hàm
{
    int ket_qua;
    if (a >b)
        ket_qua = 1:
    else if (a == b)
        ket_qua = 0;
    else if (a < b)
        ket_qua = -1;
    return ket_qua;
}
```

→ phần thân hàm

Dạng 2:

```
int so_sanh (a, b)              → phần đầu hàm
int a, b;
{
    int ket_qua:
    if (a >b)
        ket_qua = 1;
    else if (a == b)
        ket_qua = 0;
    else if (a < b)
        ket_qua = -1
    return ket_qua;
}
```

→ phần thân hàm

KHAI BÁO HÀM

Ví dụ

```
#include <stdio.h>
#include <conio.h>
int so_sanh (int a, intb);  →  prototype của hàm so_sa
main()
    {
        int a, b, ket_qua;
        clrscr();
        printf ("Moi nhap hai so ");
        scanf ("%d %d" , &a, &b);
        ket_qua = so_sanh (a, b);  →  gọi hàm
        switch (ket_qua)
            {
                case -1:
                    printf ("So %d nho hon so %d \n" , a, b);
                    break;
                case 0:
                    printf ("So %d bang so %d \n", a, b);
                    break;
                case 1:
                    printf ("So %d lon hon so %d \n" , a, b);
                    break;
            }
        getch();
    }
```

KHAI BÁO HÀM

```
int so_sanh (int a, int b)
    {
        int ket_qua:
        if (a >b)
            ket_qua = 1;
        else if (a == b)
            ket_qua = 0;
        else if (a < b)
            ket_qua = -1;
        return ket_qua;
    }
```

→ *định nghĩa hàm*

CBGD: ThS.Trần Anh Dũng

13

ĐỐI SỐ CỦA HÀM - ĐỐI SỐ LÀ THAM TRỊ

Một cách bình thường, khi gọi hàm thì đối số thật cần gởi cho hàm chỉ được gởi dưới dạng tham số trị, có nghĩa là các biến, trị hoặc biểu thức được gởi đến cho một hàm, qua đối số của nó, sẽ được lấy trị để tính toán trong thân hàm. Như vậy về khía cạnh biến có thể nói trị của biến thật bên ngoài khi gọi hàm đã được chép sang đối số giả, mà ta có thể xem như là biến cục bộ của hàm, và mọi việc tính toán chỉ được thực hiện trên biến cục bộ này mà thôi.

CBGD: ThS.Trần Anh Dũng

14

ĐỐI SỐ CỦA HÀM - ĐỐI SỐ LÀ THAM TRỊ

Ví dụ Viết chương trình tính lũy thừa n của x(x^n), với n nguyên và thực.

```
#include <stdio.h>
#include <conio.h>
  double luy_thua(double x,
  main()
      {
          int n;
          double x, xn;
          clrscr();
                                  printf ("Moi nhap so tinh luy thua: ");
                                  scant ("%lf", &x);
                                  printf ("Moi nhap so luy thua: ");
                                  scant ("%d", &n);
                                  xn =luy_thua (x, n);
                                  printf("Ket qua: %5.2f luy thua %d bang: %7.2f\n", x,n,xn);
                                  printf ("Tri cua so mu la %d", n);
                                  getch();
                              }
```


CBGD: ThS.Trần Anh Dũng

15

ĐỐI SỐ CỦA HÀM - ĐỐI SỐ LÀ THAM TRỊ

```
double luy_thua(double x, int n)
    {
        double t = 1;
        for ( ; n > 0; n--)
            t *= x;
        return t;
    }
```

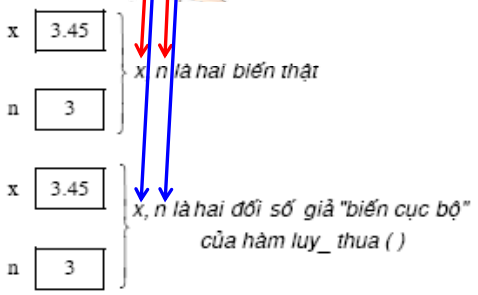

CBGD: ThS.Trần Anh Dũng

16

ĐỐI SỐ CỦA HÀM - ĐỐI SỐ LÀ THAM TRỊ

Lưu ý:

```
printf ("Moi nhap so tinh luy thua: ");
scant ("%lf", &x);
printf ("Moi nhap so luy thua: ");
scant ("%d", &n);
xn =luy_thua (x, n);
```



CBGD: ThS.Trần Anh Dũng

17

ĐỐI SỐ CỦA HÀM - ĐỐI SỐ LÀ THAM TRỊ

Ví dụ Viết chương trình sử dụng hàm nhập số liệu

```
#include <stdio.h>
#include <conio.h>
void nhap_tri (int a, int b);
main()
    {
        int a = 0, b = 0;
        clrscr();
        printf ("Truoc khi goi ham nhap_tri: a = %d, b = %d\n", a, b);
        nhap_tri (a, b);
        printf("Sau khi goi ham nhap_tri a = %d, b = %d\n", a, b);
        getch();
    }
void nhap_tri (int a, int b)
    {
        printf ("Moi nhap hai so: ");
        scanf ("%d %d", &a, &b);
    }
```

Nếu không có đối số cho hàm thì nên để là **void**

Chương trình này sẽ in ra màn hình các dòng sau:

Truoc khi goi ham nhap_tri: a = 0, b = 0

Moi nhap hai so: 5 4

Sau khi goi ham nhap_tri: a = 0, b = 0

CBGD: ThS.Trần Anh Dũng

18

ĐỐI SỐ CỦA HÀM - ĐỐI SỐ LÀ THAM TRỊ

Chương trình này sẽ in ra màn hình các dòng sau:

```
Truoc khi goi ham nhap_tri: a = 0, b = 0
Moi nhap hai so: 5 4
Sau khi goi ham nhap_tri: a = 0, b = 0
```

Như vậy thay vì thu được a = 5, b = 4 thì ta cũng chỉ nhận được a =0, b =0 mà thôi. Trong trường hợp này muốn đạt được kết quả như mong muốn ta không thể truyền theo giá trị mà cần truyền theo địa chỉ của biến mà ta thực sự muốn thay đổi trị. Cụ thể cách truyền như thế nào ta sẽ đề cập cụ thể trong chương 6, chương đề cập về con trỏ (pointer)

ĐỐI SỐ CỦA HÀM - ĐỐI SỐ LÀ THAM TRỊ

Ví dụ Thiết kế chương trình dùng hàm nhập mảng, tính tổng các phần tử và in ra màn hình kết quả.

```
#include <stdio.h>
#include <conio.h>
void nhap_tri (int a[], int n);
int tong (int a[], int n);
main()
    {
        int a[100], n;
        int sum;
        clrscr();
        printf ("Moi nhap so phan tu cua mang: ");
        scanf (%d , &n);
        nhap_tri (a, n);
        sum = tong (a, n);
        printf ("Tong cac phan tu cua mang la: %d \n", sum);
        getch();
    }
```

ĐỐI SỐ CỦA HÀM - ĐỐI SỐ LÀ THAM TRỊ

```
void nhap_tri (int a[], int n)
    {
        int i;
        printf ("Moi nhap cac phan tu cua mang: ");
        for (i = 0; i <n; i++)
        scanf ("%d", &a[i]);
    }
```

```
int tong (int a[], int n)
    {
        int i, s = 0;
        for (i = 0; i < n; s += a[i++])
                ;
        return s;
    }
```

Trong chương trình trên mảng a[] được truyền cho hàm với dạng tương tự như truyền theo kiểu tham trị, nhưng trị của mảng hoàn toàn có thể thay đổi được. Do đó khi làm việc trên mảng cần phải lưu ý về điều này.

CBGD: ThS.Trần Anh Dũng

21

KẾT QUẢ TRẢ VỀ CỦA HÀM – LỆNH RETURN

Đối với C không có sự phân biệt giữa thủ tục (procedure) và hàm (function), mà thủ tục cũng được xem là một hàm mà không trả về giá trị nào cả. Để khai báo kiểu trả về từ hàm như vậy C đưa ra kiểu **void**, tạm gọi là kiểu không kiểu.

CBGD: ThS.Trần Anh Dũng

22

KẾT QUẢ TRẢ VỀ CỦA HÀM – LỆNH RETURN

kiểu tên_hàm();

Ví dụ

```
#include <stdio.h>
#include <conio.h>
main()
        {
            int so_sanh ();   →  nơi khai báo hàm sử dụng
            int so1, so2;
            clrscr();
            printf ("Moi nhap hai so: ");
            scanf ("%d %d", &so1, &so2);
            so_sanh (so1, so2);
            getch();
        }
```


CBGD: ThS.Trần Anh Dũng

23

KẾT QUẢTRẢ VỀ CỦA HÀM – LỆNH RETURN

```
int so_sanh (int a, int b)
    {
        if (a > b)
            {
                printf ("So %d lon hon so %d", a, b);
                return 1;
            }
        else if (a == b)
            {
                printf ("So %d bangso %d", a, b);
                return 0;
            }
        else /* a < b */
            {
                printf ("So %d nho hon so %d", a, b)
                return -1;
            }
    }
```


CBGD: ThS.Trần Anh Dũng

24

PROTOTYPE CỦA MỘT HÀM

Như vậy để một hàm có thể sử dụng trong một hàm khác thì trong hàm sử dụng phải có khai báo hàm cần sử dụng

kiểu tên_hàm (danh_sách_khai_báo_đối_số);

Ví dụ

```
int so_sanh (int a, int b);
void gpth1 (double a, double b, doubbe c);
char kiem_tra (double n);
```


25

PROTOTYPE CỦA MỘT HÀM

Cần lưu ý rằng, đối với các hàm chuẩn trong thư viện C, prototype của chúng đã được C viết sẵn và để trong các file có phần mở rộng là .h, muốn lấy các prototype này vào chương trình ta cần ra chỉ thị bao hàm file .h chứa prototype của các hàm cần sử dụng vào đầu chương trình bằng lệnh tiền xử lý #include theo cú pháp sau:

 # include <file. h>

Ví dụ : Khi ra lệnh

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


26

HÀM ĐỆ QUY

C được gọi là một ngôn ngữ đệ quy vì C cho phép một hàm có thể gọi đến chính nó một cách trực tiếp, hoặc gián tiếp (tức là gọi qua trung gian một hàm khác)

khi đó ta nói hàm đó có tính đệ quy (recursive).

điều quan trọng là phải nắm được quá trình hàm gọi đệ quy, và điều kiện kết thúc đệ quy.

HÀM ĐỆ QUY

Ví dụ

```
#include <stdio.h>
#include <conio.h>
long factorial (long so);
int main()
    {
        long so, kq = 0;
        clrscr();
        printf ("Moi nhap mot so 0: ");
        scanf ("%ld", &so);
        kq = factorial (so);
        printf ("Ket qua %ld! la %ld \n", so, kq);
        getch();
        return 0;
    }
long factorial (long so)
    {
        if (so > 1)
            return (factorial(so - 1) * so);
        else
            return 1;
    }
```

BÀI TẬP

1. Thiết hàm giải phương trình $ax^2 + bx + c = 0$. Viết chương trình chính sử dụng hàm này.

2. Thiết hàm in ra màn hình chuỗi số Fibonaci, với thông số nhập là một số ngẫu nhiên trong số từ 1 đến 100.

Hướng dẫn: Sử dụng hàm random() và randomize() để tạo số ngẫu nhiên.

3. Thiết kế hàm tính các biểu thức sau đây:

$$T = \frac{1}{1!} + \frac{1+2}{2!} + ... + \frac{1+2+...+n}{n!}$$

$$S = (1)! + (1+2)! + ... + (1+...+n)!$$

BÀI TẬP

4. Thiết kế hàm vẽ ra màn hình hình sau:

n nhập

5. Thiết kế hàm và vẽ ra màn hình hình sau:

n nhập là số chẵn

BÀI TẬP

6. Viết chương trình thiết kế hai hàm max() và min() cho phép tìm số lớn nhất và nhỏ nhất trong loạt số đã nhập. Loạt số nhập này sẽ kết thúc việc nhập bằng việc nhấn phím <F6>

7. Viết một chương trình cho phép nhập một chuỗi. Hãy thiết kế một hàm cho phép cắt chuỗi đó ra làm nhiều từ và in ra màn hình mỗi từ trên một hàng.

8. Viết một hàm nhận một chuỗi và cho phép đảo chuỗi đó, in ra kết quả.

CBGD: ThS.Trần Anh Dũng

31

BÀI TẬP

9. Viết một hàm cho phép nhận một số nguyên dương. In ra màn hình ký số thứ n tính từ bên phải qua của số đó.

Ví dụ:

Nhập: 12345 4

Xuất: 2

10. Thiết kế một hàm đệ quy cho phép nhận một số nguyên dương, in ra màn hình số đó ở dạng nhị phân.

11. Viết hàm đệ quy tính x^n.

12. Viết một hàm nhận một số dương có phần lẻ và in ra màn hình phần nguyên và phần lẻ riêng biệt.

CBGD: ThS.Trần Anh Dũng

32