HÀM VÀ CHƯƠNG TRÌNH CON

Khoa: Công nghệ thông tin 1

Nội dung

1. Khái niệm và Cú pháp
2. Định nghĩa hàm
3. Tham số và lời gọi hàm
4. Đệ quy

Khái niệm hàm

- Tại sao phải dùng chương trình con:
  - Có công việc cần phải được thực hiện tại nhiều nơi trong chương trình → tách công việc đó thành chương trình con.
  - Để thuận tiện trong quản lý, trình bày và phát triển.
- Trong C, một chương trình con gọi là **hàm**: có tên, đầu vào và đầu ra.
- Có chức năng giải quyết một số vấn đề chuyên biệt cho chương trình chính.

Khái niệm hàm

- Hàm trong C có thể trả về kết quả thông qua tên hàm hay có thể không trả về kết quả.
- Một hàm khi được định nghĩa thì có thể được gọi trong chương trình.
- Được gọi nhiều lần với các tham số khác nhau.
- Trong C, hàm main() được gọi thực hiện đầu tiên.
- Hàm có hai loại: hàm chuẩn (hàm được trình biên dịch C viết sẵn) và hàm tự định nghĩa bởi người sử dụng.

Hàm thư viện/hàm chuẩn

- Hàm thư viện là những hàm đã được định nghĩa sẵn trong một thư viện nào đó.
- Muốn sử dụng các hàm thư viện thì phải khai báo thư viện trước khi sử dụng bằng lệnh

#include <tên thư viện.h>

Hàm thư viện/hàm chuẩn

- Ý nghĩa của một số thư viện thường dùng:

1. stdio.h:

- Thư viện chứa các hàm vào/ ra chuẩn (standard input/output).
- Gồm các hàm printf(), scanf(), getc(), putc(), gets(), puts(), fflush(), fopen(), fclose(), fread(), fwrite(), getchar(), putchar(), getw(), putw()…

2. conio.h :

- Thư viện chứa các hàm vào ra trong chế độ DOS (DOS console).
- Gồm các hàm clrscr(), getch(), getche(), getpass(), cgets(), cputs(), putch(), clreol(),…

Hàm thư viện/hàm chuẩn

3. math.h:

- Thư viện chứa các hàm tính toán.
- Gồm các hàm abs(), sqrt(), log(). log10(), sin(), cos(), tan(), acos(), asin(), atan(), pow(), exp(),…

4. alloc.h:

- Thư viện chứa các hàm liên quan đến việc quản lý bộ nhớ.
- Gồm các hàm calloc(), realloc(), malloc(), free(), farmalloc(), farcalloc(), farfree(), …

Hàm thư viện/hàm chuẩn

5. io.h:

- Thư viện chứa các hàm vào ra cấp thấp.
- Gồm các hàm: open(), _open(), read(), _read(), close(), _close(), creat(), _creat(), creatnew(), eof(), filelength(), lock(),…

6. graphics.h:

- Thư viện chứa các hàm liên quan đến đồ họa.
- Gồm initgraph(), line(), circle(), putpixel(), getpixel(), setcolor(), …

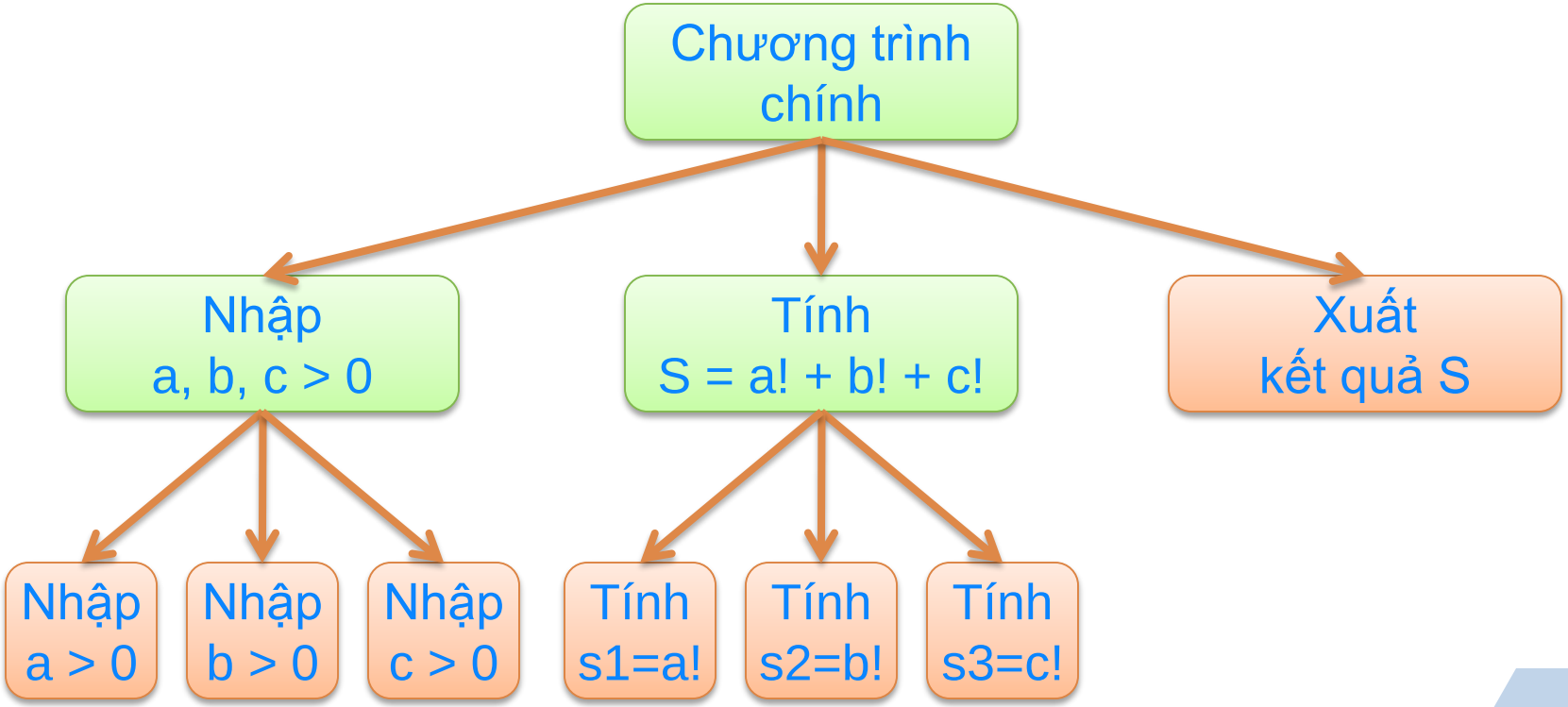
…

Muốn sử dụng các hàm thư viện thì ta phải xem cú pháp của các hàm và sử dụng theo đúng cú pháp (xem trong phần trợ giúp của Turbo C).

Hàm tự định nghĩa

Đặt vấn đề

- Viết chương trình tính S = a! + b! + c! với a, b, c là 3 số nguyên dương nhập từ bàn phím.



Đặt vấn đề

- 3 đoạn lệnh nhập a, b, c > 0

```
do {
      printf("Nhap mot so nguyen duong: ");
      scanf("%d", &a);
} while (a <= 0);

do {
      printf("Nhap mot so nguyen duong: ");
      scanf("%d", &b);
} while (b <= 0);

do {
      printf("Nhap mot so nguyen duong: ");
      scanf("%d", &c);
} while (c <= 0);
```

Đặt vấn đề

- 3 đoạn lệnh tính s1 = a!, s2 = b!, s3 = c!

```
{ Tính s1 = a! = 1 * 2 * … * a }
s1 = 1;
for (i = 2; i <= a ; i++)
      s1 = s1 * i;

{ Tính s2 = b! = 1 * 2 * … * b }
s2 = 1;
for (i = 2; i <= b ; i++)
      s2 = s2 * i;

{ Tính s3 = c! = 1 * 2 * … * c }
s3 = 1;
for (i = 2; i <= c ; i++)
      s3 = s3 * i;
```

Đặt vấn đề

- Giải pháp => Viết 1 lần và sử dụng nhiều lần
  - Đoạn lệnh nhập tổng quát, với n = a, b, c

```
do {
        printf("Nhap mot so nguyen duong: ");
        scanf("%d", &n);
} while (n <= 0);
```

  - Đoạn lệnh tính giai thừa tổng quát, n = a, b, c

```
{ Tính s = n! = 1 * 2 * … * n }
s = 1;
for (i = 2; i <= n ; i++)
        s = s * i;
```

Hàm

Cú pháp

```
<kiểu trả về> <tên hàm>([danh sách tham số])
{
        <các câu lệnh>
        [return <giá trị>;]
}
```

- Trong đó
  - <kiểu trả về> : kiểu bất kỳ của C (char, int, long, float,…). Nếu không trả về thì là void.
  - <tên hàm>: theo quy tắc đặt tên định danh.
  - <danh sách tham số> : tham số hình thức đầu vào giống khai báo biến, cách nhau bằng dấu ,
  - <giá trị> : trả về cho hàm qua lệnh return.

Các bước viết hàm

- Cần xác định các thông tin sau đây:
  - Tên hàm.
  - Hàm sẽ thực hiện công việc gì.
  - Các đầu vào (nếu có).
  - Đầu ra (nếu có).

Hàm

- Ví dụ 1
  - Tên hàm: XuatTong
  - Công việc: tính và xuất tổng 2 số nguyên
  - Đầu vào: hai số nguyên x và y
  - Đầu ra: không có

```
void XuatTong(int x, int y)
{
        int s;
        s = x + y;
        printf("%d cong %d bang %d", x, y, s);
}
```

Hàm

- Ví dụ 2
  - Tên hàm: TinhTong
  - Công việc: tính và trả về tổng 2 số nguyên
  - Đầu vào: hai số nguyên x và y
  - Đầu ra: một số nguyên có giá trị x + y

```
int TinhTong(int x, int y)
{
        int s;
        s = x + y;
        return s;
}
```

Hàm

- Ví dụ 3
  - Tên hàm: NhapXuatTong
  - Công việc: nhập và xuất tổng 2 số nguyên
  - Đầu vào: không có
  - Đầu ra: không có

```
void NhapXuatTong()
{
        int x, y;
        printf("Nhap 2 so nguyen: ");
        scanf("%d%d", &x, &y);
        printf("%d cong %d bang %d", x, y, x + y);
}
```

Tầm vực

- Khái niệm
  - Là phạm vi hiệu quả của biến và hàm.
  - Biến:
    - Toàn cục: khai báo trong ngoài tất cả các hàm (kể cả hàm main) và có tác dụng lên toàn bộ chương trình.
    - Cục bộ: khai báo trong hàm hoặc khối { } và chỉ có tác dụng trong bản thân hàm hoặc khối đó (kể cả khối con nó). Biến cục bộ sẽ bị xóa khỏi bộ nhớ khi kết thúc khối khai báo nó.

Tầm vực

```
int a;
        int Ham1()
        {
                int a1;
        }

        int Ham2()
        {
                int a2;
                {
                        int a21;
                }
        }

        void main()
        {
                int a3;
        }
```

Một số lưu ý

- Thông thường người ta thường đặt phần tiêu đề hàm/nguyên mẫu hàm (prototype) trên hàm main và phần định nghĩa hàm dưới hàm main.

```
void XuatTong(int x, int y);  // prototype

void main()
{
        …
}

void XuatTong(int x, int y)
{
        printf("%d cong %d bang %d", x, y, x + y);
}
```

Các cách truyền tham số

Ví dụ: Viết chương trình hoán vị 2 phần tử

```
#include<stdio.h>
1  // Truyền bằng tham trị
2  void Swap1 (int x, int y)
3  {
4      int temp = x;
5      x = y;
6      y = temp;
   }
7  // Truyền bằng tham biến (con trỏ)
8  void Swap2 (int *x, int *y)
9  {
10     int temp = *x;
11     *x = *y;
       *y = temp;
12 }
13 // Truyền bằng tham chiếu
14 void Swap3 (int &x, int &y)
15 {
16     int temp = x;
17     x = y;
18     y = temp;
   }
```

```
int main()
{     int m=12; n=28;
      Swap1(m,n);
      printf("m=%d n=%d\n",m,n");
      Swap2(&m,&n);
      printf("m=%d n=%d\n",m,n");
      Swap3(m,n);
      printf("m=%d n=%d\n",m,n");
      return 0;
}
```

Các cách truyền tham số

- Truyền bằng Giá trị (Call by Value) (tham trị)
  - Truyền đối số cho hàm ở dạng giá trị.
  - Có thể truyền hằng, biến, biểu thức nhưng hàm chỉ sẽ nhận giá trị.
  - Được sử dụng khi không có nhu cầu thay đổi giá trị của tham số sau khi thực hiện hàm.

```
void TruyenGiaTri(int x)
{
        …
        x++;
}
```

Các cách truyền tham số

Mặc nhiên, việc truyền tham số cho hàm trong C là truyền theo giá trị; nghĩa là các giá trị thực (tham số thực) không bị thay đổi giá trị khi truyền cho các tham số hình thức.

Ví dụ 1: Giả sử muốn in ra các, mỗi dòng gồm 50 ký tự nào đó. Để đơn giản ta viết hàm, hàm này sẽ in ra trên một dòng 50 ký tự cho trước.

Các cách truyền tham số

```
#include <stdio.h>
#include <conio.h>
void InKT(char ch)
{  int i;
       for(i=1;i<=50;i++) printf("%c",ch);
                  printf("\n");
}
void  main()
{
    char c = 'A';
    InKT('*'); /* In ra 50 dau * */
    InKT('+');
    InKT(c);
}
```

Lưu ý:

- Trong hàm InKT, biến ch gọi là tham số hình thức được truyền bằng giá trị (gọi là ***tham trị*** của hàm). Các tham trị của hàm coi như là một biến cục bộ trong hàm và chúng được sử dụng như là dữ liệu đầu vào của hàm.
- Khi chương trình con được gọi để thi hành, tham trị được ***cấp ô nhớ*** và nhận giá trị là bản sao giá trị của tham số thực.
-Việc thay đổi giá trị của chúng không có ý nghĩa gì đối với bên ngoài hàm, không ảnh hưởng đến chương trình chính, nghĩa là không làm ảnh hưởng đến tham số thực tương ứng.

Các cách truyền tham số

- Truyền Địa chỉ (Call by Address)
  - Truyền đối số cho hàm ở dạng địa chỉ (con trỏ).
  - Không được truyền giá trị cho tham số này.
  - Được sử dụng khi có nhu cầu thay đổi giá trị của tham số sau khi thực hiện hàm.

```
void TruyenDiaChi(int *x)
{
        …
        *x++;
}
```

Các cách truyền tham số

- Truyền Tham chiếu (Call by Reference) (C++)
  - Truyền đối số cho hàm ở dạng địa chỉ (con trỏ). Được bắt đầu bằng & trong khai báo.
  - Không được truyền giá trị cho tham số này.
  - Được sử dụng khi có nhu cầu thay đổi giá trị của tham số sau khi thực hiện hàm.

```
void TruyenThamChieu(int &x)
{
        …
        x++;
}
```

Các cách truyền tham số

- Lưu ý
  - Trong một hàm, các tham số có thể truyền theo nhiều cách.

```
void HonHop(int x, int &y)
{
        …
        x++;
        y++;
}
```

Các cách truyền tham số

- Lưu ý
  - Sử dụng tham chiếu là một cách để trả về giá trị cho chương trình.

```
int TinhTong(int x, int y)
{
	return x + y;
}
void TinhTong(int x, int y, int &tong)
{
	tong = x + y;
}
void TinhTongHieu(int x, int y, int &tong, int &hieu)
{
	tong = x + y; hieu = x - y;
}
```

Các cách truyền tham số - Lưu ý

Ví dụ: Viết chương trình hoán vị 2 phần tử

```
#include<stdio.h>
1  // Truyền bằng tham trị
2  void Swap1 (int x, int y)
3  {
4      int temp = x;
5      x = y;
6      y = temp;
   }
7  // Truyền bằng tham biến (con trỏ)
8  void Swap2 (int *x, int *y)
9  {
10     int temp = *x;
11     *x = *y;
       *y = temp;
12 }
13 // Truyền bằng tham chiếu
14 void Swap3 (int &x, int &y)
15 {
16     int temp = x;
17     x = y;
18     y = temp;
   }
```

```
int main()
{     int m=12; n=28;
      Swap1(m,n);
      printf("m=%d n=%d\n",m,n");
      Swap2(&m,&n);
      printf("m=%d n=%d\n",m,n");
      Swap3(m,n);
      printf("m=%d n=%d\n",m,n");
      return 0;
}
```

?

Lời gọi hàm

- Cách thực hiện
  - Gọi tên của hàm đồng thời truyền các đối số (hằng, biến, biểu thức) cho các tham số theo đúng thứ tự đã được khai báo trong hàm.
  - Các biến hoặc trị này cách nhau bằng dấu ,
  - Các đối số này được được đặt trong cặp dấu ngoặc đơn ( )
  - <tên hàm> (<đối số 1>,… , <đối số n>);

Lời gọi hàm

- Ví dụ

```
{ Các hàm được khai báo ở đây }
void main()
{
        int n = 9;
        XuatTong(1, 2);
        XuatTong(1, n);
        TinhTong(1, 2);
        int tong = TinhTong(1, 2);
        TruyenGiaTri(1);
        TruyenGiaTri(n);
        TruyenDiaChi(1);
        TruyenDiaChi(&n);
        TruyenThamChieu(1);
        TruyenThamChieu(n);
}
```

Lời gọi hàm

- Ví dụ

```
void HoanVi(int &a, int &b);

void main()
{
        HoanVi(2912, 1706);
        int x = 2912, y = 1706;
        HoanVi(x, y);
}
void HoanVi(int &a, int &b)
{
        int tam = a;
        a = b;
        b = tam;
}
```

Các nguyên tắc xây dựng hàm

- Mỗi hàm chỉ thực hiện một công việc duy nhất.
- Độc lập với các hàm khác.
- Mỗi hàm nên che giấu thông tin.
- Tham số của hàm cần được chỉ rõ.
- Tính tái sử dụng càng cao tốt.
- Các dữ liệu mà hàm sử dụng:
  - Giữ cho các mỗi liên hệ càng đơn giản càng tốt. Tránh sử dụng biến toàn cục càng nhiều càng tốt.
  - Nếu sử dụng biến toàn cục, viết hướng dẫn chi tiết.

Hàm đệ quy

- Khái niệm
  - Một chương trình con có thể gọi một chương trình con khác.
  - Nếu gọi chính nó thì được gọi là sự đệ quy.
  - Số lần gọi này phải có giới hạn (điểm dừng)
- Ví dụ
  - Tính S(n) = n! = 1*2*…*(n-1)*n
  - Ta thấy S(n) = S(n-1)*n
  - Vậy thay vì tính S(n) ta sẽ đi tính S(n-1)
  - Tương tự tính S(n-2), …, S(2), S(1), S(0) = 1

Hàm đệ quy

- Ví dụ

```
int GiaiThua(int n)
{
	if (n == 0)
		return 1;
	else
		return GiaiThua(n - 1) * n;
}
int GiaiThua(int n)
{
	if (n > 0)
		return GiaiThua(n - 1) * n;
	else
		return 1;
}
```

Đặc điểm cần lưu ý khi viết hàm đệ quy

- Hàm đệ quy phải có 2 phần:

  + Phần dừng hay phải có trường hợp nguyên tố. Trong ví dụ ở trên thì trường hợp n=0 là trường hợp nguyên tố.

  + Phần đệ quy: là phần có gọi lại hàm đang được định nghĩa. Trong ví dụ trên thì phần đệ quy là n>0 thì n! = n * (n-1)!

- Sử dụng hàm đệ quy trong chương trình sẽ làm chương trình dễ đọc, dễ hiểu và vấn đề được nêu bật rõ ràng hơn. Tuy nhiên trong đa số trường hợp thì hàm đệ quy tốn bộ nhớ nhiều hơn và tốc độ thực hiện chương trình chậm hơn không đệ quy.
- Tùy từng bài có cụ thể mà người lập trình quyết định có nên dùng đệ quy hay không (có những trường hợp không dùng đệ quy thì không giải quyết được bài toán).

Tham số của hàm main()

- **Là 2 tham số : argc và argv**
- **Tham số argc là số nguyên chỉ tham số trên dòng lệnh, có giá trị nhỏ nhất =1, vì bản thân tên chương trình là tham số thứ nhất**
- **Tham số argv là mảng các con trỏ, trỏ đến các tham số trên dòng lệnh: char *argv[ ];**

argv[0]: chứa địa chỉ của tên chương trình

argv[1]: chứa địa chỉ của tham số thứ nhất

argv[2]: chứa địa chỉ của tham số thứ hai

Ví dụ: Chương trình sau đã được biên dịch thành MYPRO.EXE, nếu nhập trên dòng lệnh MYPRO thì có dòng nhắc nhở, nếu nhập MYPRO LAN thì Chao ban LAN

```
#include <stdio.h>
main(int argc, char *argv[])
{
    if (argc !=2) printf("Phai nhap Ten");
    else printf("Chao ban %s\n",argv[1]);
}
```