Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ thông tin
Bộ môn Tin học cơ sở

NHẬP MÔN LẬP TRÌNH

Đặng Bình Phương
dbphuong@fit.hcmuns.edu.vn

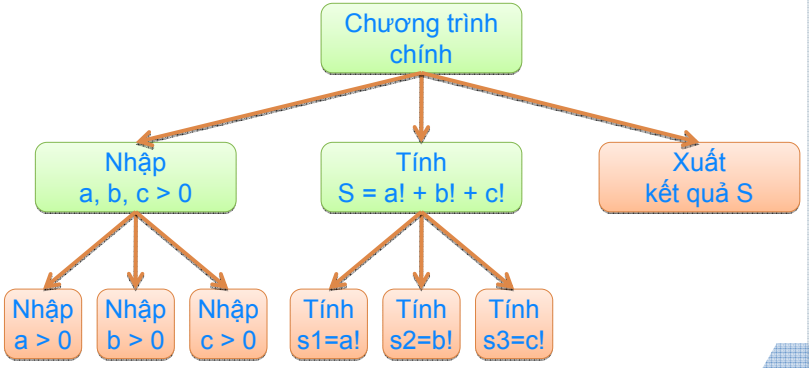
HÀM (FUNCTION)

Nội dung

1. Khái niệm và cú pháp
2. Tầm vực
3. Tham số và lời gọi hàm
4. Đệ quy

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

- Khái niệm
  - Một đoạn chương trình có tên, đầu vào và đầu ra.
  - Có chức năng giải quyết một số vấn đề chuyên biệt cho chương trình chính.
  - Được gọi nhiều lần với các tham số khác nhau.
  - Được sử dụng khi có nhu cầu:
    - Tái sử dụng.
    - Sửa lỗi và cải tiến.

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;
}
```


Chương trình con - Function

- 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 đối số

- Truyền Giá trị (Call by Value)
  - 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 đối 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 đối 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++;
}
```


Lưu ý khi truyền đối số

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

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


Lưu ý khi truyền đối 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;
}
```

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 chương trình con

- 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;
}
```


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

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;
}
```


Bài tập thực hành

5. Bài 4, 5, 6, 7, 8 trang 140-141 chương 8 (Câu lệnh điều kiện và rẽ nhánh)
   a. Viết hàm đổi một ký tự hoa sang ký tự thường.
   b. Viết thủ tục giải phương trình bậc nhất.
   c. Viết thủ tục giải phương trình bậc hai.
   d. Viết hàm trả về giá trị nhỏ nhất của 4 số nguyên.
   e. Viết thủ tục hoán vị hai số nguyên.
   f. Viết thủ tục sắp xếp 4 số nguyên tăng dần

Bài tập thực hành

6. Bài tập 3 trang 155 chương 9 (Câu lệnh lặp). Hàm nhận vào một số nguyên dương n và thực hiện:
   a. Trả về số đảo của số đó.
   b. Có phải là số đối xứng (Trả về True/False)
   c. Có phải là số chính phương.
   d. Có phải là số nguyên tố.
   e. Tổng các chữ số lẻ.
   f. Tổng các chữ số nguyên tố.
   g. Tổng các chữ số chính phương.

Bài tập thực hành

7. Bài tập 4 trang 156 chương 9 (Câu lệnh lặp). Hàm nhận vào một số nguyên dương n và thực hiện:
   a. $S = 1 + 2 + \ldots + n$
   b. $S = 1^2 + 2^2 + \ldots + n^2$
   c. $S = 1 + 1/2 + \ldots + 1/n$
   d. $S = 1 * 2 * \ldots * n$
   e. $S = 1! + 2! + \ldots + n!$
8. Hàm trả về USCLN của 2 số nguyên.
9. In ra n phần tử của dãy Fibonacy.