TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

TIN HỌC ĐẠI CƯƠNG
Bài 13. Hàm

Nội dung

1. Khái niệm hàm
2. Khai báo và sử dụng hàm
3. Phạm vi của biến

Nội dung

1. Khái niệm hàm
   1.1. Khái niệm chương trình con
   1.2. Phân loại chương trình con
2. Khai báo và sử dụng hàm
3. Phạm vi của biến

3

Một ví dụ

```
#include <stdio.h>
#include <conio.h>
int giaiThua(int);//Khai báo nguyên mẫu hàm
int main(){
   //Khai báo n, k và nhập thông tin
   //...
   toHop=giaiThua(n)/(giaiThua(k)*giaiThua(n-k));
   //In kết quả
}
//Khai báo nội dung hàm
int giaiThua(int n){
   int i,ketQua = 1;
   for(i = 1;i <= n; i++) ketQua = ketQua*i;
   return ketQua;
}
```

4

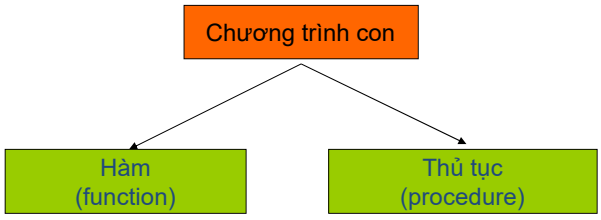
1.1. Khái niệm chương trình con

- Khái niệm
  - Là một chương trình nằm trong một chương trình lớn hơn nhằm thực hiện một nhiệm vụ cụ thể
- Vai trò
  - Chia nhỏ chương trình ra thành từng phần để quản lý => Phương pháp lập trình có cấu trúc
  - Có thể sử dụng lại nhiều lần: printf, scanf…
  - Chương trình dễ dàng đọc và bảo trì hơn

5

1.2. Phân loại chương trình con

- Phân loại chương trình con

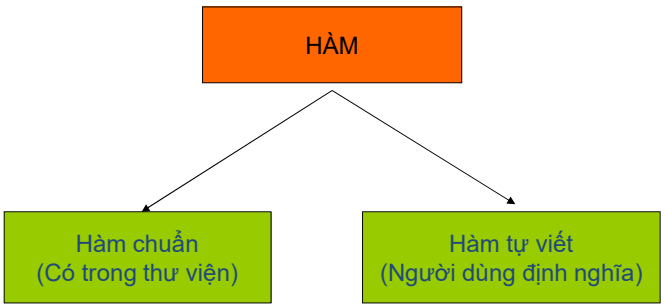


  - Hàm: trả về giá trị trong khi thủ tục thì không
  - Trong C:
    - Chỉ cho phép khai báo chương trình con là hàm.
    - Sử dụng kiểu “void” với ý nghĩa “không là kiểu dữ liệu nào cả” để chuyển thủ tục về dạng hàm

6

1.2. Phân loại chương trình con

- Phân loại hàm



7

Nội dung

8

2.1. Khai báo hàm

- Trong chương trình lớn có nhiều chương trình con, điểm bắt đầu thực hiện chương trình sẽ thuộc chương trình con nào?
- *main* là một chương trình con?
- Khai báo các chương trình con độc lập nhau/lồng lẫn nhau?
- Muốn "lắp ráp" các công việc khác nhau để cùng thực hiện, cần phải đưa ra "lời gọi" hàm. "Lời gọi" cần cung cấp những gì?

9

2.1. Khai báo hàm

- Ví dụ:
  - Chương trình in ra bình phương của các số tự nhiên từ 1 đến 10
  - Gồm 2 hàm:
    - Hàm *binhPhuong(int x)*: trả về bình phương của x
    - Hàm *main():* với mỗi số nguyên từ 1 đến 10, gọi hàm *binhPhuong* với một giá trị đầu vào và hiển thị kết quả.

10

2.1. Khai báo hàm

Khai báo hàm

Gọi hàm

```
#include<stdio.h>
#include<conio.h>
int binhPhuong(int x){
   int y;
   y =  x * x;
   return y;
}
int main(){
   int i;
   for (i=0; i<= 10; i++)
     printf("%d  ", binhPhuong(i));
   getch();
   return 0;
}
```

11

2.1. Khai báo hàm

```
KieuDuLieu tenHam (danh_sách_tham_số)
{
        [<Các_khai_báo>]
        [<Các_câu_lệnh>]
 }
```

- Dòng đầu hàm
  - Là thông tin trao đổi giữa các hàm. Phân biệt giữa các hàm với nhau.
  - KieuDuLieu: kiểu dữ liệu giá trị trả về của hàm
  - tenHam: là tên hợp lệ, trong C tên hàm là duy nhất

12

2.1. Khai báo hàm

- Danh sách tham số
  - Cho biết những tham số giả định cung cấp hoạt động cho hàm => các tham số hình thức
  - Tham số cung cấp dữ liệu cho hàm lúc hoạt động: tham số thực
- Ví dụ: int max(int a, int b, int c)

- Thân hàm
  - return
    - Gọi hàm thông qua tên hàm và các tham số thực cung cấp cho hàm.
    - Sau khi thực hiện xong, trở về điểm mà hàm được gọi thông qua câu lệnh return hoặc kết thúc hàm.
    - Cú pháp chung: *return biểu_thức;*

13

2.1. Khai báo hàm

Nguyên mẫu hàm (function prototype)

Định nghĩa hàm

```
#include<stdio.h>
#include<conio.h>
int binhPhuong(int );
int main(){
  int i;
  for (i=0; i<= 10; i++)
    printf("%d  ",binhPhuong(i));
  getch();
  return 0;
}
int binhPhuong(int x){
  int y;
  y =  x * x;
  return y;
}
```

2.1. Khai báo hàm

- Ý nghĩa của nguyên mẫu hàm
  - Cho phép định nghĩa sau khi sử dụng. Nhưng phải khai báo trước
  - Cho phép đưa ra lời gọi đến một hàm mà không cần biết định nghĩa
    - Ví dụ: khi gọi printf, scanf chúng ta chỉ cần quan tâm các tham số truyền cho hàm
    - Tệp stdio.h chứa nguyên mẫu hàm của printf và scanf

2.1. Khai báo hàm

- Các hàm thư viện
- Ngôn ngữ C cung cấp một số hàm thư viện như: xử lý vào ra, hàm toán học, hàm xử lý xâu…
- Để sử dụng các hàm này chúng ta chỉ cần khai báo nguyên mẫu của chúng trước khi sử dụng.
  - Khai báo thông qua chỉ thị #include<tệp_tiêu_đề>
  - tệp_tiêu_đề (.h) đã chứa các nguyên mẫu hàm

2.2. Sử dụng hàm

- Cú pháp:

  tên_hàm (danh_sách_tham_số);
- Ví dụ: binhphuong(0), binhphuong(1)…
- Lưu ý:
  - Nếu hàm nhận nhiều tham số thì các tham số ngăn cách nhau bởi dấu phẩy
  - Luôn luôn cần cặp dấu ngoặc đơn sau tên hàm
  - Các tham số của hàm sẽ nhận các giá trị từ tham số truyền vào
  - Thực hiện lần lượt các lệnh cho đến khi gặp lệnh return/kết thúc chương trình

17

Nội dung

18

3.1. Phạm vi của biến

- Phạm vi: khối lệnh, chương trình con, chương trình chính
- Biến khai báo trong phạm vi nào thì sử dụng trong phạm vi đó
- Trong cùng một phạm vi các biến có tên khác nhau.
- Tình huống
  - Trong hai phạm vi khác nhau có hai biến cùng tên. Trong đó một phạm vi này nằm trong phạm vi kia?

```
#include<stdio.h>
#include<conio.h>
int i;
int binhPhuong(int x){
   int y;
   y =  x * x;
   return y;
}
int main(){
  int y;
  for (i=0; i<= 10; i++){
     y = binhPhuong(i);
      printf("%d  ", y);
   }
  return 0;
}
```

19

3.2. Phân loại biến

- Phân loại biến
  - Biến toàn cục: biến được khai báo ngoài mọi hàm, được sử dụng ở các hàm đứng sau nó
  - Biến cục bộ: biến được khai báo trong lệnh khối hoặc chương trình con, được đặt trước các câu lệnh.
- Ghi nhớ
  - Hàm main() cũng là một chương trình con nhưng là nơi chương trình được bắt đầu cũng như kết thúc
  - Biến khai báo trong hàm main() cũng là biến cục bộ, chỉ có phạm vi trong hàm main().

20

3.3. Câu lệnh static và register

- Biến static
  - Xuất phát: biến cục bộ ra khỏi phạm vi thì bộ nhớ dành cho biến được giải phóng
  - Yêu cầu lưu trữ giá trị của biến cục bộ một cách lâu dài => sử dụng từ khóa *static*
  - So sánh với biến toàn cục?
  - Cú pháp:

    static KieuDuLieu tenBien;

21

3.3. Câu lệnh static và register

```
# include <stdio.h>
# include <conio.h>
void fct() {
   static int count =0;
   printf("\n Day la lan goi ham fct lan thu %2d",
   count++);
}
int main(){
   int i;
   for(i = 0;  i < 10; i++)  fct();
   getch();
   return 0;
}
```

22

3.3. Câu lệnh static và register

```
Day la lan goi ham fct lan thu  1
Day la lan goi ham fct lan thu  2
Day la lan goi ham fct lan thu  3
Day la lan goi ham fct lan thu  4
Day la lan goi ham fct lan thu  5
Day la lan goi ham fct lan thu  6
Day la lan goi ham fct lan thu  7
Day la lan goi ham fct lan thu  8
Day la lan goi ham fct lan thu  9
Day la lan goi ham fct lan thu 10
```

23

3.3. Câu lệnh static, register

- Biến register
  - Thanh ghi có tốc độ truy cập nhanh hơn RAM, bộ nhớ ngoài
  - Lưu biến trong thanh ghi sẽ tăng tốc độ thực hiện chương trình
  - Cú pháp

    register KieuDuLieu tenBien;
  - Lưu ý: số lượng biến register không nhiều và thường chỉ với kiểu dữ liệu nhỏ như int, char

24

Thảo luận

25