Ngôn ngữ lập trình C

B9: Hàm và thủ tục

Khoa Công nghệ thông tin

Chủ đề

- Giới thiệu về hàm
- Các hàm toán học
- Các hàm người dùng định nghĩa
- Truyền tham số cho hàm
- Phạm vi biến
- Đệ quy
- Bài tập thực hành

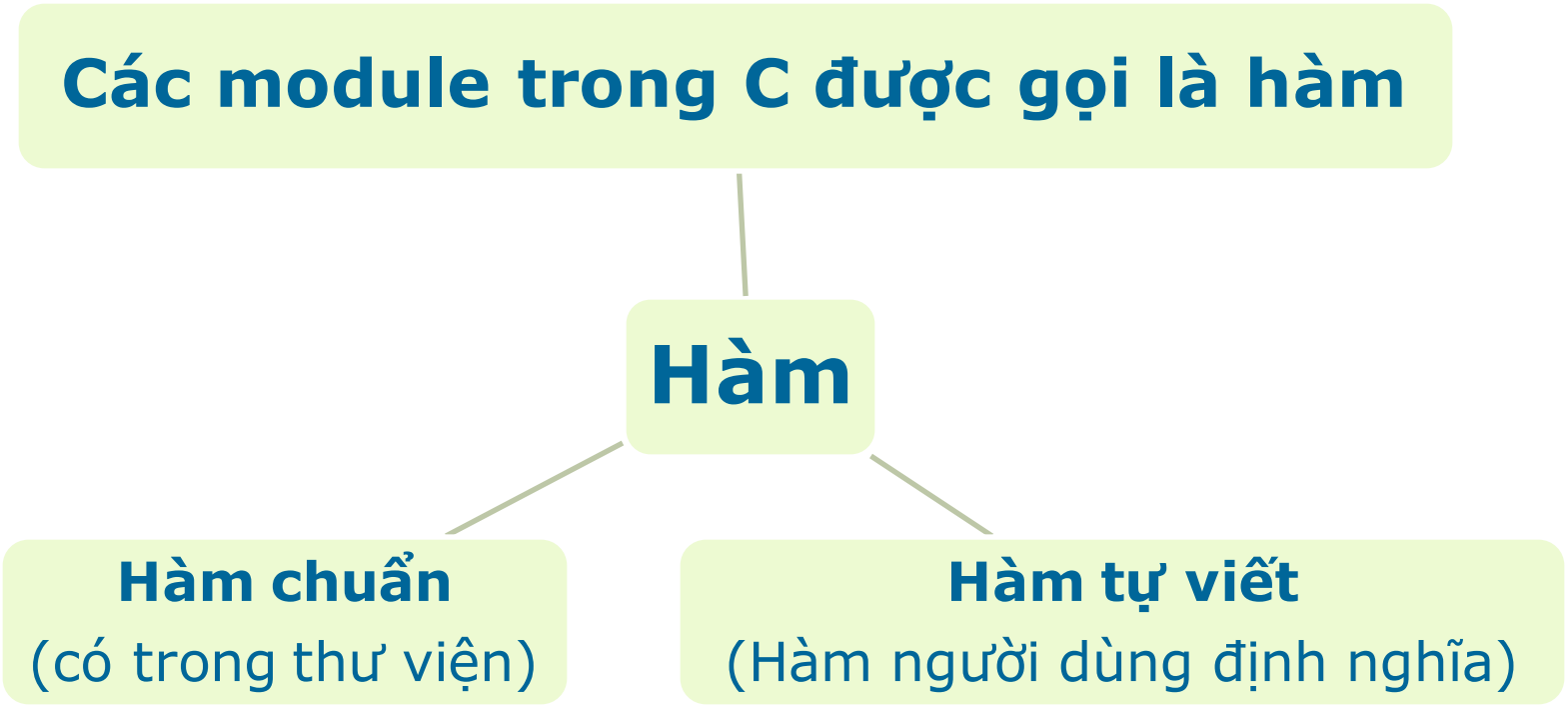
Giới thiệu về hàm

- Có những đoạn chương trình được thực hiện lặp đi lặp lại nhiều lần, tuy dữ liệu có khác nhưng bản chất các công việc lại giống nhau → Viết gộp những đoạn chương trình đó lại thành một chương trình con mà khi cần chỉ việc truyền dữ liệu cho nó?
- Tư tưởng đó cũng dẫn chúng ta tới việc chia một chương trình lớn thành nhiều phần nhỏ rồi giải quyết từng phần; sau đó sẽ ráp nối chúng lại là sẽ hoàn tất một chương trình lớn. Các chương trình nhỏ này trong C chính là các ***hàm (function)***.
- Như vậy một chương trình sẽ là một tập hợp các định nghĩa hàm riêng biệt.

Giới thiệu về hàm

- Các hàm được gọi để thực hiện bằng các lời gọi hàm.
- Các lời gọi hàm xác định tên của hàm và cung cấp các thông tin (hay còn gọi là các tham số) mà hàm được gọi theo đúng trình tự khi khai báo hàm đó.
- Khi viết chương trình chúng ta luôn định nghĩa một hàm `main`.
- Trong hàm `main` có thể gọi các hàm khác và trong các hàm đó cũng có thể gọi đến các hàm khác nữa.

Giới thiệu về hàm



Hàm: viết một lần, dùng lại nhiều lần, ở nhiều nơi.

Các hàm toán học trong C

- Khai báo tệp tiêu đề `#include<math.h>`
- Khi dùng chỉ việc viết lời gọi hàm
- Để sử dụng được thư viên toán khi biên dịch cần them option: -lm vào lệnh biên dịch. Ví dụ:

```
gcc -o r baitap.c -lm
```

- Ví dụ: viết hàm tính và in ra căn bậc 2 của 900.0 là: `printf("%.2f", sqrt(900.0));`
  - `sqrt` là hàm khai căn bậc 2
  - số 900.0 là tham số của hàm `sqrt`
  - hàm `sqrt` nhận tham số thực và trả về giá trị kiểu thực
  - Câu lệnh sẽ in ra 30.00

Các hàm toán học trong C (Thông dụng)

Hàm	Mô tả	Ví dụ
sqrt(x)	Căn bậc 2 của x	sqrt(9.00) bằng 3.0
exp(x)	hàm mũ e^x	exp(1.0) bằng 2.718282
log(x)	logarithm tự nhiên (cơ số e) của x	log(2.718282) bằng 1.0
log10(x)	logarithm thập phân (cơ số 10) của x	log10(1.0) bằng 0.0 log10(10.0) bằng 1.0
fabs(x)	giá trị tuyệt đối của x	nếu x≥0 thì fabs(x) bằng x nếu x<0 thì fabs(x) bằng -x
ceil(x)	làm tròn thành số nguyên nhỏ nhất lớn hơn x	ceil(9.2) bằng 10.0 ceil(-9.8) bằng -9.0

Các hàm toán học trong C (Thông dụng)

floor(x)	làm tròn thành số nguyên lớn nhất nhỏ hơn x	floor(9.2) bằng 9.0 floor(-9.8) bằng -10.0
pow(x,y)	x mũ y (x^y)	pow(2,7) bằng 128
fmod(x,y)	phần dư của phép chia x cho y	fmod(13.657,2.333) bằng 1.992
sin(x)	sin của x (x theo radian)	sin(0.0) bằng 0.0
cos(x)	cos của x (x theo radian)	cos(0.0) bằng 1.0
tan(x)	tan của x (x theo radian)	tan(0.0) bằng 0.0

9

Hàm người dùng định nghĩa
Ví dụ 9.1

```
double square(double a)
{
   return a * a;
}
```

Hàm định nghĩa bên ngoài hàm main

```
int main(void)
{
   double num = 0.0, sqr = 0.0;

   printf("enter a number\n");
   scanf("%lf",&num);

   sqr = square(num);

   printf("square of %g is %g\n", num, sqr);

   return 0;
}
```

gọi sử dụng hàm

Hàm người dùng định nghĩa

- Khuôn dạng của hàm

```
kiểu tên-hàm (danh-sách-tham-số)
{
 Các khai báo
 ...
 Các câu lệnh
 [return [bieu_thuc];]
}
```

Hàm người dùng định nghĩa

- **Giải thích về khuôn dạng của hàm:**
  - Ở đây kiểu là kiểu dữ liệu của kết quả của hàm, tên-hàm là một cái tên do người lập trình tự đặt tuân theo quy tắc đặt tên trong C;
  - Danh-sách-tham-số gồm danh sách các tham số hàm nhận được khi nó được gọi và được ngăn cách nhau bởi dấu phẩy.
  - Các khai báo và các câu lệnh trong ngoặc là phần thân của hàm.

Hàm người dùng định nghĩa

- Hàm có thể có giá trị trả về hoặc không có giá trị trả về:
  - Nếu có giá trị trả về, trong thân hàm có ít nhất một lệnh `return`.
  - Nếu không có giá trị trả về cần khai báo cho hàm đó có kiểu trả về là `void`. Với kiểu này thường được gọi là các thủ tục.

Hàm người dùng định nghĩa

- Ví dụ 9.2: viết hàm tính giai thừa

```
int giai_thua(int a)
{
  int ket_qua;
  int i;

  ket_qua = 1;
  for(i = 1;i<=a;i++)
    ket_qua = ket_qua * i;
  if(a < 0) ket_qua = -1;
  if(a == 0) ket_qua = 1;
  return ket_qua;
}
```

Dong dau ham

Cac khai bao

Cac cau lenh

Hàm người dùng định nghĩa

- Ví dụ 9.3: Xây dựng hàm maximum trả về số nguyên lớn nhất trong ba số nguyên.

```
#include<stdio.h>
int maximum(int a, int b, int c){
 int max;
 max = a;
 if(max<b) max = b;
 if(max<c) max = c;
 return max;
}

void main(){
 int x, y, z;
 printf("Moi ban nhap vao ba so nguyen: ");
 scanf("%d %d %d",&x,&y,&z);
 printf("So nguyen lon nhat la: %d\n",maximum(x,y,z));
}
```

Truyền tham số cho hàm

- Các tham số của hàm có thể là các hằng, biến hay các biểu thức.
- Ví dụ:

nếu c1 = 13.0, d = 3.0 thì câu lệnh

```
printf("%.2f", sqrt(c1 + d * 4.0));
```

sẽ tính và in ra căn bậc 2 của 13.0 + 3.0 * 4.0 = 25.0 là 5.00

Truyền tham số cho hàm

- Có hai cách để truyền các tham số cho hàm:
  - *Khi các tham số được truyền theo trị, một bản sao giá trị của tham số được tạo ra và truyền cho hàm. Vì vậy mọi sự thay đổi trong hàm trên bản sao sẽ không ảnh hưởng đến giá trị ban đầu của biến nằm trong hàm gọi.*
  - *Khi một tham số được truyền theo con trỏ, hàm gọi sẽ truyền trực tiếp tham số đó cho hàm bị gọi thay để đổi giá trị nguyên thuỷ của biến truyền vào tham số.*
- Trong C, tất cả các tham số được truyền đều là truyền theo trị.
- Ta có thể mô phỏng cách truyền tham số theo tham chiếu bằng cách truyền địa chỉ của các biến vào thamsố. Với cách này ta có thể gọi là truyền tham biến.

Truyền tham số cho hàm

- Có hai cách để truyền các tham số cho hàm:
  - *Khi các tham số được truyền theo trị, một bản sao giá trị của tham số được tạo ra và truyền cho hàm. Vì vậy mọi sự thay đổi trong hàm trên bản sao sẽ không ảnh hưởng đến giá trị ban đầu của biến nằm trong hàm gọi.*
  - *Khi một tham số được truyền theo con trỏ, hàm gọi sẽ truyền trực tiếp tham số đó cho hàm bị gọi. Do đó có thể thay đổi giá trị nguyên thuỷ của biến truyền vào tham số.*
- Trong C, tất cả các tham số được truyền đều là truyền theo trị.
- Ta có thể mô phỏng cách truyền tham số theo tham chiếu bằng cách truyền địa chỉ của các biến vào thamsố. Với cách này ta có thể gọi là truyền tham biến.

Ví dụ 9.4

- Viết hàm hoán đổi hai số nguyên.
- Hãy quan sát theo hai cách làm sau đây.

Ví dụ 9.4

- **Cách 1**: Viết hàm hoán đổi hai số nguyên. Trả lời câu hỏi xem việc hoán đổi có thực sự xảy ra?

```
#include<stdio.h>
void swap(int x, int y){
 int temp;
 printf("Hai so nguyen ban dau duoc truyen vao ham: %d, %d\n",x,y);
 printf("Ham thu hien viec trao doi hai so nguyen nay.\n");
 temp = x;
 x = y;
 y = temp;
 printf("Hai so nguyen sau khi duoc trao doi trong ham: %d, %d\n",x,y);
}

void main(){
 int x, y;
 printf("Moi ban nhap vao hai so nguyen:\n");
 printf("x = "); scanf("%d",&x);
 printf("y = "); scanf("%d",&y);
 printf("Hai so nguyen truoc khi duoc truyen vao ham swap: %d, %d\n",x,y);
 printf("Goi ham swap\n");
 swap(x,y);
 printf("Hai so nguyen sau khi thuc hien xong ham swap: %d, %d\n",x,y);
}
```

Ví dụ 9.4

- **Cách 2**: Viết hàm hoán đổi hai số nguyên. Trả lời câu hỏi xem việc hoán đổi có thực sự xảy ra?

```c
#include<stdio.h>
void swap(int* x, int* y){
 int temp;
 printf("Hai so nguyen ban dau duoc truyen vao ham: %d, %d\n",*x,*y);
 printf("Ham thu hien viec trao doi hai so nguyen nay.\n");
 temp = *x;
 *x = *y;
 *y = temp;
 printf("Hai so nguyen sau khi duoc trao doi trong ham: %d, %d\n",*x,*y);
}

void main(){
 int x, y;
 printf("Moi ban nhap vao hai so nguyen:\n");
 printf("x = "); scanf("%d",&x);
 printf("y = "); scanf("%d",&y);
 printf("Hai so nguyen truoc khi duoc truyen vao ham swap: %d, %d\n",x,y);
 printf("Goi ham swap\n");
 swap(&x, &y);
 printf("Hai so nguyen sau khi thuc hien xong ham swap: %d, %d\n",x,y);
}
```

Phạm vi biến

- Biến địa phương (Local Variable):
  - Là các biến được khai báo trong lệnh khối hoặc trong than chương trình con.
- Biến toàn cục (Global Variable):
  - Là biến được khai báo trong chương trình chính.
  - Vị trí khai báo của biến toàn cục là sau phần khai báo tệp tiêu đề và khai báo hàm nguyên mẫu.

Phạm vi biến

- Ví dụ 9.5: xét các chương trình sau

```c
#include<stdio.h>
void main(){
 {
  int x;
  x = 1;
  printf("x1 = %d\n",x);
  {
   int x;
   x = 2;
   printf("x2 = %d\n",x);
  }
  printf("x3 = %d\n",x);
 }
 {
  int x;
  x = 3;
  printf("x4 = %d\n",x);
 }
}
```

```c
#include<stdio.h>
void test(){
 int x;
 x = 1;
 printf("Bien x khai bao trong ham test: %d\n",x);
}

void main(){
 int x;
 x = 2;
 printf("Bien x khai bao trong ham main = %d\n",x);
 printf("Goi ham test co bien x khai bao trong do\n");
 test();
 printf("Bien x trong ham main sau khi goi ham test = %d\n",x);
}
```

Xem xét phạm vi của các biến?

Phạm vi biến

- Ví dụ 9.6: xét các chương trình sau

```c
#include<stdio.h>
//Khai bao bien toan cuc
float a,b;

float tinhtong(){
 printf("Gia tri cua hai bien a va b trong ham tinh tong: a = %f, b = %f\n",a,b);
 return a+b;
}

void main(){
 printf("Nhap vao gia tri cua a = "); scanf("%f",&a);
 printf("Nhap vao gia tri cua b = "); scanf("%f",&b);
 printf("Gia tri cua hai bien a va b trong ham main: a = %f, b = %f\n",a,b);
 printf("Goi ham tinh tong a va b\n");
 printf("Tong cua hai so a va b la: %f\n",tinhtong());
 printf("Gia tri cua hai bien a va b trong ham main sau khi tinh tong: a = %f, b = %f\n",a,b);
}
```

Xem xét phạm vi của các biến?

Đệ quy

- Trong C cho phép trong thân một hàm có thể gọi ngay đến chính nó, cơ chế này gọi là đệ qui.
- Có thể định nghĩa hàm đệ qui là hàm sẽ gọi đến chính nó trực tiếp hay gián tiếp thông qua các hàm khác.
- Cách tiến hành giải một bài toán đệ qui nhìn chung có những điểm chung sau:
  - Có trường hợp cơ sở
  - Nó gọi đến chính dạng của nó nhưng có sự chuyển biến
  - Sự chuyển biến ấy đến một lúc nào đó phải kết thúc

Đệ quy

- Ta xem xét ví dụ tính n giai thừa (**n!**).
  - Theo định nghĩa n! bằng tích của tất cả các số tự nhiên từ 1 đến n: n! = n * (n-1) ... 2 * 1
  - Ví dụ: 5! = 5 * 4 * 3 * 2 * 1 = 120, 0! được định nghĩa bằng 1.
  - Để tính n! có thể dùng vòng lặp như sau:

```
fact = 1;
for (i = n; i >= 1; i--)
fact *= i;
```

  - Tuy nhiên cũng có thể định nghĩa đệ qui hàm giai thừa như sau :

nếu n>0 thì n! = n * (n-1)!

nếu n=0 thì n! = 0! = 1

Đệ quy

- Ví dụ 9.7: đệ quy tính giai thừa

```
#include<stdio.h>
long giathuadq(long n){
 if(n==0) return 1;
 else return n*giathuadq(n-1);
}

void main(){
 long N;
 printf("Nhap vao N = "); scanf("%ld",&N);
 printf("%ld! = %ld\n", N, giathuadq(N));
}
```

Đệ quy

- Ví dụ 9.8: dẫy số Fibonaci gồm những số
  - 0, 1, 1, 2, 3, 5, 8, 13, 21 ...
  - Bắt đầu từ hai số 0 và 1, tiếp sau đó các số Fibonaci sau bằng tổng của 2 số Fibonaci trước nó.
  - Dãy Fibonaci có thể định nghĩa đệ qui như sau:

$$F(0) = 0;\ F(1) = 1;\ F(n) = F(n-1) + F(n-2)\ \forall n>1$$

```
#include<stdio.h>
long fibo(long n){
 if(n==0 || n==1) return n;
 else return fibo(n-1) + fibo(n-2);
}

void main(){
 long N, F;
 printf("Nhap vao N = "); scanf("%ld",&N);
 F = fibo(N);
 printf("So hang Fibonanci F(%ld) = %ld\n", N, F);
}
```

BÀI TẬP

THỰC HIỆN LẠI TOÀN BỘ BÀI TẬP BUỔI 8 SỬ DỤNG HÀM