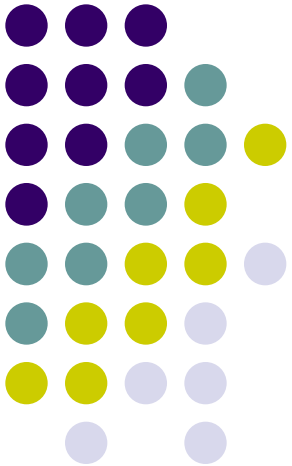
LẬP TRÌNH CĂN BẢN

Phần 2 - Chương 5

CHƯƠNG TRÌNH CON



Nội dung chương này

- Ví dụ
- Khái niệm về hàm trong C
- Xây dựng một hàm
- Truyền tham số cho hàm
- Hàm đệ qui

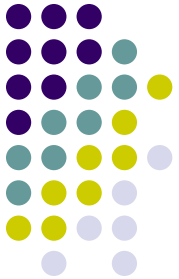
Ví dụ (1)

- In ra 50 ký tự ‘*’ và 50 ký tự ‘+’

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

int main(){
	int i;
	char ch='*';
	for(i=1;i<=50;i++)
		printf("%c",ch);
	printf("\n");

	ch='+';
	for(i=1;i<=50;i++)
		printf("%c",ch);
	printf("\n");
	getch();
	return 0;
}
```

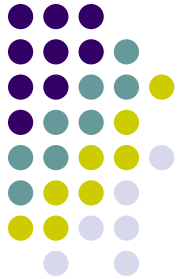
```
#include <stdio.h>
#include <conio.h>

void InKT(char ch)
{
	for(int i=1;i<=50;i++)
		printf("%c",ch);
	printf("\n");
}

int main()
{
	InKT('*'); // In ra 50 dau *
	InKT('+'); // In ra 50 dau +

	char c = 'A';
	InKT(c);
	return 0;
}
```

Ví dụ (2)



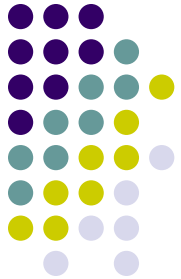
- Đâu là ưu điểm của việc dùng hàm?

Khái niệm về hàm trong C (1)



- Để tránh rườm rà và mất thời gian khi viết chương trình, những đoạn chương trình lặp đi lặp lại nhiều lần được viết trong 1 module.
- Chia chương trình thành nhiều module, mỗi module giải quyết 1 công việc nào đó.
- Mỗi module như trên được gọi là 1 chương trình con.
- Các module dễ dàng được kiểm tra tính đúng đắn trước khi được ráp nối vào chương trình.

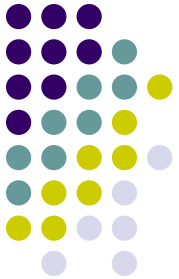
Khái niệm về hàm trong C (2)



- Ví dụ: Tìm số lớn nhất trong 3 số a, b, và c.

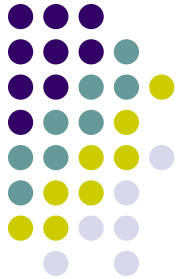
```
#include <stdio.h>
#include <conio.h>
int max(int a, int b){
    return (a>b) ? a:b;
}
int main()
{
   int a, b, c;
   printf("\n Nhap vao 3 so a, b,c  ");
   scanf("%d%d%d",&a,&b,&c);
   printf("\n So lon la %d",max(a, max(b,c)));
   getch();
   return 0;
}
```

Khái niệm về hàm trong C (3)



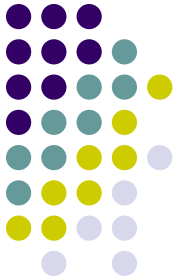
- Có 2 loại hàm:
  - Hàm chuẩn
  - Hàm tự định nghĩa

Hàm chuẩn (hàm thư viện)



- Được định nghĩa sẵn bởi ngôn ngữ lập trình và được chứa vào các thư viện.
- Muốn sử dụng phải khai báo ***#include <tên thư viện.h>***
- Một số thư viện thường dùng trong C:
  - **stdio.h** **:** Thư viện chứa các hàm vào/ ra chuẩn (**st**an**d**ard **i**nput/**o**utput): **printf(), scanf(), getc(), putc(), gets(), puts(), fflush(), fopen(), fclose(), fread(), fwrite(), getchar(), putchar(), getw(), putw(), …**
  - **conio.h :** Thư viện chứa các hàm vào ra trong chế độ DOS (DOS console): **clrscr(), getch(), getche(), getpass(), cgets(), cputs(), putch(), clreol(), …**
  - **math.h:** Thư viện chứa các hàm tính toán: **abs(), sqrt(), log(). log10(), sin(), cos(), tan(), acos(), asin(), atan(), pow(), exp(), …**
  - **alloc.h:** Thư viện chứa các hàm liên quan đến việc quản lý bộ nhớ: **calloc(), realloc(), malloc(), free(), farmalloc(), farcalloc(), farfree(), …**
  - **io.h:** Thư viện chứa các hàm vào ra cấp thấp: **open(), _open(), read(), _read(), close(), _close(), creat(), _creat(), creatnew(), eof(), filelength(), lock(), …**
  - **graphics.h:** Thư viện chứa các hàm liên quan đến đồ họa:**initgraph(), line(), circle(), putpixel(), getpixel(), setcolor(), …**

Hàm tự định nghĩa (hàm người dùng) (1)

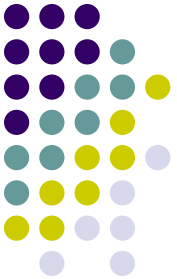


- Do người lập trình tự tạo ra nhằm đáp ứng nhu cầu xử lý của mình.

- **Cấu trúc của một hàm tự thiết kế:**

```
<kiểu kết quả> Tên hàm ([<kiểu t số> <tham số>][,<kiểu t số><tham số>][…]){
        [Khai báo biến cục bộ ]
        [Các câu lệnh thực hiện hàm]
        [return [<Biểu thức>];]
}
```

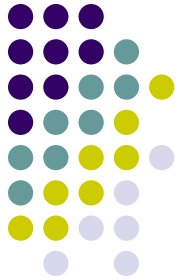
Hàm tự định nghĩa (hàm người dùng) (2)



- **Cú pháp gọi hàm:**<Tên hàm>([*Danh sách các tham số*])
- **Ví dụ:** Tìm UCLN của 2 số tự nhiên:

```
#include<stdio.h>
unsigned int ucln(unsigned  int a, unsigned  int b){
   unsigned int u;
   if (a<b)
         u=a;
   else
         u=b;
   while ((a%u !=0) || (b%u!=0))
         u--;
   return u;
}
int main(){
   unsigned int a, b, UC;
   printf("Nhap a,b: ");scanf("%d%d",&a,&b);
   UC = ucln(a,b);
   printf("Uoc chung lon nhat la: ", UC);
   return 0;
}
```

Nguyên tắc hoạt động của hàm



- Trong chương trình, khi gặp một lời gọi hàm thì các bước sau được thực hiện:
  - Nếu hàm có tham số, trước tiên các tham số sẽ được *gán giá trị thực tương ứng*.
  - Chương trình sẽ thực hiện tiếp các câu lệnh trong thân hàm bắt đầu từ lệnh đầu tiên đến câu lệnh cuối cùng.
  - Khi gặp lệnh return hoặc dấu } cuối cùng trong thân hàm, chương trình sẽ thoát khỏi hàm để trở về chương trình gọi nó.
  - Thực hiện tiếp tục những câu lệnh của chương trình.

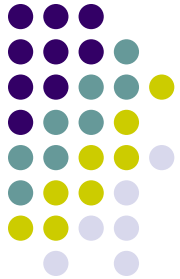
Truyền tham số cho hàm (1)

- **Ví dụ:** Hoán đổi nội dung của 2 biến

```
#include <stdio.h>
#include <conio.h>
int  hoanvi(int a, int b){
      int t;
      t=a; /*Đoạn này hoán vị giá trị của 2 biến a, b*/
      a=b;
      b=t;
      printf("\Ben trong ham a=%d  , b=%d",a,b);
      return 0;
}
int main(){
      int a, b;
      clrscr();
      printf("\n Nhap vao 2 so nguyen a, b:");
      scanf("%d%d",&a,&b);
      printf("\n Truoc khi goi ham hoan vi a=%d  ,b=%d",a,b);
      hoanvi(a,b);
      printf("\n Sau khi goi ham hoan vi a=%d  ,b=%d",a,b);
      getch();
      return 0;
}
```

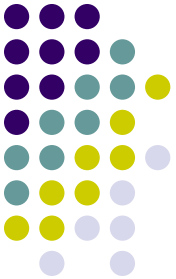
```
Nhap vao 2 so nguyen a, b:6 5

Truoc khi goi ham hoan vi a=6  ,b=5
Ben trong ham a=5  , b=6
Sau khi goi ham hoan vi a=6  ,b=5
```

Truyền tham số cho hàm (2)



- Ta vẫn chưa hoán vị được!
- **Tại sao?**
  - 2 tham số a và b của hoanvi là **tham số hình thức** được **truyền bằng giá trị (tham trị).**
    - 1 tham trị được coi như 1 biến cục bộ của hàm, chứa dữ liệu đầu vào cho hàm.
  - Còn 2 tham số a,b của hoanvi trong lời gọi hàm trong main() là **tham số thực**.
  - 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**.
  - Do đó, **mọi sự thay đổi trên tham trị không ảnh hưởng gì đến tham số thực tương ứng.**

Truyền tham số cho hàm (3)

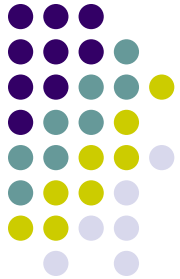


- Hãy xem chương trình sau

```
#include <stdio.h>
#include <conio.h>
int hoanvi(int *a, int *b){
      int  t;
      t=*a;      /*gán nội dung của x cho t*/
      *a=*b;     /*Gán nội dung của b cho a*/
      *b=t;      /*Gán nội dung của t cho b*/
      printf("\n Ben trong ham a=%d  , b=%d",*a,*b);
      return 0;
}
int main(){
      int a, b;
      clrscr();
      printf("\n Nhap vao 2 so nguyen a, b:");
      scanf("%d%d",&a,&b);
      printf("\n Truoc khi goi ham hoan vi a=%d  ,b=%d",a,b);
      hoanvi(&a,&b); // Phải là địa chỉ của a và b
      printf("\n Sau khi goi ham hoan vi a=%d  ,b=%d",a,b);
      getch();
      return 0;
}
```

```
Nhap vao 2 so nguyen a, b: 5 6

Truoc khi goi ham hoan vi a=5  ,b=6
Ben trong ham a=6  , b=5
Sau khi goi ham hoan vi a=6  ,b=5
```

Truyền tham số cho hàm (4)



- Tại sao ta đã hoán vị được?
  - 2 tham số a và b của hoanvi là **tham số hình thức** được **truyền bằng địa chỉ (tham biến) – con trỏ.**
  - Khi chương trình con (ctc) được gọi để thi hành, **tham biến chứa địa chỉ tham số thực, ô nhớ của tham số thực được dùng trực tiếp trong ctc qua biến con trỏ**.
  - Do đó, **mọi sự thay đổi trên tham biến đều ảnh hưởng đến tham số thực tương ứng.**

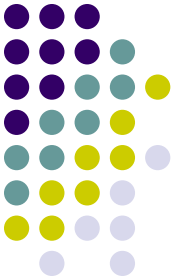
Hàm đệ quy

- Một hàm được gọi là đệ quy nếu bên trong thân hàm có lệnh gọi đến chính nó.
- Ví dụ:

$$n! = \begin{cases} 1 \text{ nếu } n=0 \\ n*(n-1)! \text{ nếu } n\#0 \end{cases}$$

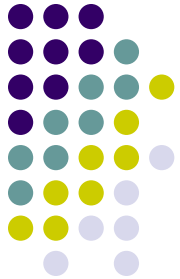
```
unsigned int giaithua_dequy(int n)
{
   if (n==0)
      return 1;
   else
      return n*giaithua_dequy(n-1);
}
```

Đặc điểm cần lưu ý khi viết hàm đệ quy



- Hàm đệ quy phải có 2 phần:
  - **Phần dừng:** là trường hợp nguyên tố.
    - **Ví dụ:** n=0 trong tính n!
  - **Phần đệ quy:** là phần có gọi lại hàm đang được định nghĩa.
    - **Ví dụ:** nếu n>0 thì n! = n * (n-1)!

Ưu và khuyết điểm của đệ quy



- Làm chương trình dễ đọc, dễ hiểu và vấn đề được nêu bật rõ ràng hơn.
- Đệ quy tốn bộ nhớ nhiều hơn và tốc độ thực hiện chương trình chậm hơn không đệ quy.
- Tùy từng bài cụ thể mà ta quyết định có nên dùng đệ quy hay không.
- Có những trường hợp không dùng đệ quy thì không giải quyết được bài toán.


Hết chương