

LẬP TRÌNH CƠ SỞ

*Trần Tuấn Vinh – Nguyễn Thị Loan
Viện CNTT – ĐHSP Hà Nội 2*

Tài liệu tham khảo

- # Kỹ thuật lập trình C – Phạm Văn Ât
- # Giáo trình C – Aptech
- # Giáo trình C – ĐHQG Hà Nội
- # Giáo trình Ngôn ngữ lập trình C – Tiêu Kim Cương
- # Giáo trình Ngôn ngữ lập trình C – Nguyễn Hữu Tuấn

Chương 1

Tổng quan thuật toán và
lập trình, giới thiệu về
ngôn ngữ lập trình C

Bài toán và thuật toán

- Để hiểu rõ được bài toán cần định nghĩa được nó.
- Định nghĩa được bài toán giúp xác định rõ được vấn đề cần giải quyết và mục tiêu cần đạt.
- Bài toán trong tin học là công việc nào đó muốn máy tính thực hiện. Để giải quyết bài toán trong tin học cần xác định rõ đầu vào (input), đầu ra (output).
- Ví dụ: Bài toán tìm nghiệm của phương trình bậc 2: $ax^2 + bx + c = 0$ ($a \neq 0$)
 - Input: Ba số thực a, b, c ($a \neq 0$)
 - Output: Tất cả các số thực x là nghiệm của chương trình

Bài toán và thuật toán

- Để giải quyết một bài toán khi đã có input, output ta cần có thuật toán.
- Thuật toán là một dãy hữu hạn các thao tác được sắp xếp theo một trình tự xác định sao cho sau khi thực hiện dãy thao tác đó, từ input của bài toán ta nhận được output cần tìm.

Bài toán và thuật toán

- Ví dụ: Thuật toán giải phương trình bậc 2: $ax^2 + bx + c = 0$ ($a \neq 0$)
 - Bước 1: Nhập các giá trị a, b, c.
 - Bước 2: Nếu $a = 0$, không thỏa mãn điều kiện của bài toán ($a \neq 0$), chuyển đến bước 4.
 - Bước 3: Nếu $a \neq 0$
 - Tính giá trị Delta $D = b^2 - 4ac$.
 - Nếu $D > 0$ thì phương trình có 2 nghiệm phân biệt x_1, x_2 được tính theo công thức $x_1 = \frac{-b+\sqrt{D}}{2a}; x_2 = \frac{-b-\sqrt{D}}{2a}$, chuyển đến bước 4.
 - Nếu $D = 0$, phương trình có nghiệm kép $x_0 = \frac{-b}{2a}$, chuyển đến bước 4.
 - Nếu $D < 0$, phương trình vô nghiệm, chuyển đến bước 4.
 - Bước 4: Kết thúc.

Bài toán và thuật toán

- Tính chất của thuật toán
 - Tính xác định
 - Tính dừng
 - Tính phổ dụng
 - Tính đúng
 - Tính khả thi

Bài toán và thuật toán

- Các phương pháp biểu diễn thuật toán
 - Biểu diễn bằng ngôn ngữ tự nhiên
 - Biểu diễn bằng sơ đồ khối
 - Biểu diễn bằng ngôn ngữ lập trình

Quy trình giải bài toán trên máy tính điện tử

- Bước 1: Xác định bài toán (xác định Input – Output)
- Bước 2: Tìm ra thuật toán để giải quyết bài toán (process)
- Bước 3: Lập trình giải bài toán
- Bước 4: Chạy thử chương trình.
- Bước 5: Kiểm chứng và hoàn thiện chương trình (thử nghiệm bằng nhiều số liệu và đánh giá).
- *BT: Xác định Input, Output, Process của các bài toán.*

Ngôn ngữ lập trình C

Lịch sử hình thành và phát triển

- Xuất hiện đầu những năm 70, TK 20.
- Tác giả là Brian W.Kernighan và Dennis M.Ritchie, phòng thí nghiệm Bell (Mỹ).
- Mục đích là phát triển hệ điều hành UNIX
- C chịu ảnh hưởng của ngôn ngữ B do Ken Thompson viết năm 70
- C có những điểm mạnh trong xử lý các vấn đề hiện đại của tin học
- 1978, ra đời giáo trình dạy lập trình C “The C programming language”
- 1983, đề xuất chuẩn cho ngôn ngữ C
- 1988, chuẩn ANSI C chính thức được ban hành
- Các hệ chương trình dịch: MSC, VC, Lattice C, Turbo C, Borland C
- Phát triển lập trình hướng đối tượng.

Các tính chất đặc trưng của ngôn ngữ

- Là ngôn ngữ lập trình vận năng
- Có mức độ thích nghi cao
- Viết các chương trình “khả chuyển” – có thể chạy ngay cả khi có sự thay đổi phần cứng
- Sử dụng trong các lĩnh vực chuyên nghiệp
- Sử dụng thư viện hàm
- Cung cấp con trỏ có khả năng định địa chỉ số học
- Cho phép gọi hàm đệ quy
- Các định nghĩa hàm không được lồng nhau

Cấu trúc cơ bản của một chương trình C (1)

- **Ví dụ 1**

```
/*Chương trình hiển thị dòng chữ “Welcome to C  
programming language!” trên màn hình */  
  
#include <stdio.h> //Thư viện vào/ra chuẩn  
#include <conio.h> /*Thư viện hỗ trợ các thao tác I/O từ  
màn hình */  
  
void main() {  
    printf(“Welcome to C programming language!\n”);  
    getch();  
}
```

Cấu trúc cơ bản của một chương trình C (2)

- Ví dụ 2

```
/* Chương trình nhập và in ra màn hình giá trị biến*/
#include <stdio.h>
#include <conio.h>
void main() {
    int i;
    printf("Nhập vào một số: ");
    scanf("%d", &i);
    printf("Số bạn vừa nhập là: %d.\n", i);
    getch();
}
```

Cấu trúc cơ bản của một chương trình C (3)

- **Ví dụ 3:** Viết chương trình nhập vào 2 số nguyên a và b, in ra màn hình tổng của 2 số nguyên đó.

Cấu trúc cơ bản của một chương trình C (4)

- Thực hiện chương trình
 - Tạo tệp chương trình nguồn
 - Biên dịch chương trình
 - Chạy chương trình
- Định nghĩa hàm
 - Một chương trình C có ít nhất một hàm
 - Hàm main() có một và chỉ một
- Khai báo tệp tiêu đề
- Dòng chú thích
- Dấu “;” kết thúc câu lệnh
- Hàm printf(), scanf()
- Các thư viện trong C

Cấu trúc cơ bản của một chương trình C (5)

- 1: [các bao hàm tệp]
- 2: [các khai báo nguyên mẫu hàm của người dùng]
- 3: [các định nghĩa kiểu]
- 4: [các định nghĩa macro]
- 5: [các định nghĩa biến, hằng]
- 6: <kiểu_hàm> main ([khai báo tham số])
- 7: {
- 8: < thân hàm main>
- 9: }
- 10: [các định nghĩa hàm của người dùng]

Chương 2

Các yếu tố cơ bản của ngôn ngữ C

Tập ký tự, tên gọi

- Tập ký tự dùng trong C
 - Các chữ cái: A,..,Z, a,..,z
 - Các chữ số: 0,..,9
 - Các dấu phép toán số học: +,-,*,/,...
 - Các dấu ngoặc: (,), [,],...
- Tên (identifier):
 - Tên là xâu kí tự chỉ có thể gồm: các chữ cái, chữ số, dấu gạch nối
 - Bắt đầu bằng chữ cái hoặc dấu gạch nối
 - Độ dài cực đại mặc định là 32
 - Phân biệt chữ hoa và chữ thường

Từ khóa

- Từ khóa
 - asm, break, case, char, const, continue, default, do, double, else, enum, extern, far, float, for, goto, if, int, interrupt, long, near, register, return, short, signed, sizeof, static, struct, switch, typedef, union, unsigned, void, while, ...
- Không được đặt tên trùng với từ khóa

Các kiểu dữ liệu cơ sở

Kiểu	Kích thước	Phạm vi
unsigned char	1 byte	0 ÷ 255
char	1 byte	-128 ÷ 127
unsigned int	2 byte	0 ÷ 65535
short int	2 byte	-32768 ÷ 32767
int	2 byte	-32768 ÷ 32767
unsigned long	4 byte	0 ÷ 4294967295
long	4 byte	-2147483648 ÷ 2147483647
float	4 byte	±3.4E-38 ÷ ±3.4E+38
double	8 byte	±1.7E-308 ÷ ±1.7E+308
long double	10 byte	±3.4E-4932 ÷ ±1.1E+4932

Khai báo biến

- Quy tắc khai báo

- <Kiểu dữ liệu> <Tên biến>;

- Ví dụ:

```
int var1, var2; /* biến có kiểu nguyên */
```

```
float f; /* biến nhận giá trị thực */
```

```
char ch; /* biến ký tự hay biến nguyên */
```

```
int n=0; /* khai báo biến và khởi tạo giá trị */
```

- Có thể khai báo biến đồng thời khởi tạo giá trị cho biến.

Khai báo hằng

- Dùng **#define** định nghĩa hằng tượng trưng
 - `#define <tên hằng> <xâu ký tự>`
- Dùng từ khóa **const**
 - `const <tên kiểu> <tên biến hằng>=<giá trị>;`
- Cách biểu diễn hằng số
 - Hằng nguyên: Hệ 10, hệ 8 (thêm 0 ở đầu), hệ 16 (thêm 0x).
 - Hằng nguyên dài: như hằng nguyên nhưng thêm ký tự l hay L ở cuối.
 - Hằng dấu phẩy động: viết thông thường và ký pháp khoa học.
 - Hằng ký tự:
 - Bằng ký hiệu trong bảng mã ASCII
 - Bằng cặp bao gồm ký tự ‘\’ và số thứ tự của ký tự trong bảng mã ASCII
 - Hằng xâu ký tự: đặt trong 2 dấu nháy kép (“”)

Biểu thức

- Là kết quả ghép nối các toán tử và các toán hạng để diễn đạt một công thức nào đó
- Các toán hạng: hằng, biến, lời gọi hàm, biểu thức con
- Số ngôi của một toán tử xác định số toán hạng kết hợp với toán tử đó
- Biểu thức được dùng trong
 - Vết phải của lệnh gán
 - Làm tham số thực của các hàm
 - Làm chỉ số
 - Các câu lệnh if, for, while, do while
 - Các biểu thức phức tạp hơn

Biểu thức (tiếp)

- Các phép toán số học: +, -, *, /, %, đổi dấu (-)
- Các phép toán quan hệ: >, >=, <, <=, ==, !=
- Các phép toán logic: &&, ||, !
- Các phép toán thao tác trên bit: &, |, ^, <<, >>, ~
Không dùng cho dữ liệu kiểu float hay double
- Phép toán lấy địa chỉ biến (&)
- Phép toán chuyển đổi kiểu bắt buộc: (<Kiểu>) <biểu thức>

Biểu thức (tiếp)

- Biểu thức gán
 - <Biến>=<Biểu thức>
 - Nếu thêm dấu chấm phẩy vào cuối ta được câu lệnh gán.
 - Có thể sử dụng trong các phép toán và các câu lệnh như các biểu thức thông thường
 - $a=b=10; x=(a=5)*(b=10);$
 - Một số ký pháp đặc biệt của phép gán
 - $+ =, - =, * =, / =, \% =, \& =, | =, ^ =, << =, >> =$

Biểu thức (tiếp)

- Biểu thức điều kiện
 - Expr1 ? Expr2 : Expr3
 - Expr1, Expr2, Expr3 là các biểu thức
 - Nếu Expr1 có giá trị khác 0 thì biểu thức điều kiện có giá trị cho bởi Expr2, ngược lại là Expr3.
 - Ví dụ:
 - $\text{MAX}=(a>b)?a:b;$
 - $\text{MIN}=a<b?(a<c?a:c):(b<c?b:c);$
- Các phép toán tăng giảm một đơn vị
 - Tăng 1 đơn vị: $++n$ hay $n++$
 - Giảm 1 đơn vị: $--n$ hay $n--$

Vào - ra

- Thư viện vào-ra chuẩn: stdio.h và conio.h
- Hàm printf(): in dữ liệu ra màn hình
 - `printf(<format>[,<arguments>, ...]);`
 - *format* là xâu định dạng xác định cách thức hiển thị dữ liệu
 - Danh sách *arguments* chứa các giá trị được đưa ra
 - Một số định dạng
 - %d, %i, %o, %u, %x, %e, %E, %f, %lf, %g, %G, %c, %s
- Hàm scanf(): nhập dữ liệu từ bàn phím
 - `scanf(<format>, {<address>, ...});`
 - Một số định dạng
 - %d, %o, %x, %c, %s, %f, %ld, %lf
- Hàm clrscr() – xóa màn hình; getch() – chờ nhận 1 ký tự từ bàn phím

Vào - ra

- **Hàm getch(), getche(): nhập 1 ký tự**

- Cú pháp:

`int getch();`

`int getche();`

- Chức năng: Hai hàm này thực hiện đợi người dùng nhập một ký tự từ bàn phím và trả về một số nguyên là mã của kí tự được bấm, ví dụ bạn gõ phím ‘a’ thì hàm sẽ trả về 97.
- Sự khác nhau giữa hai hàm là hàm getche hiện kí tự được nhập lên màn hình, còn getch thì không.

Vào - ra

- **Hàm gets()**
 - Cú pháp: **char * gets(char * s);**
 - Chức năng của hàm gets() là nhập một xâu kí tự từ bàn phím, khác với hàm scanf() với đặc tả "%s" kết thúc một xâu khi gặp dấu cách hoặc enter, tức là xâu không thể có dấu cách, hàm gets chỉ kết thúc khi gặp enter (kí tự '\n').
 - Xâu kí tự được ghi vào s (với s là mảng các kí tự hoặc con trỏ kí tự), dấu kết thúc xâu ('\0' - kí tự có mã 0) được tự động thêm vào cuối xâu.

Vào - ra

- **Hàm putch()**
 - Cú pháp: **int putch(int ch);**
 - Chức năng: Hàm này in kí tự có mã là ch ra màn hình tại vị trí hiện tại của con trỏ, chuyển con trỏ sang phải 1 ký tự, hàm trả về số nguyên chính là mã kí tự in ra.

Vào - ra

- **Hàm puts()**
 - Cú pháp: **int puts(char * s);**
 - Chức năng: Hàm này in xâu kí tự s ra màn hình tại vị trí hiện tại của con trỏ, sau đó tự động chuyển con trỏ sang dòng mới. Trong trường hợp in thành công hàm trả về số nguyên dương, ngược lại trả về -1.

Một số định dạng nhập, xuất

Chuỗi định dạng	Đại diện cho kiểu ký tự	Ý nghĩa	Ví dụ
%c	char	Xuất ra một ký tự	"%c"
%s	char *	Xuất ra một chuỗi ký tự	"%8s", "%-10s"
%d	int, short	Xuất ra một số nguyên dưới dạng thập phân	"%-2d", "%03d"
%u	unsigned int, unsigned short	Xuất ra một số nguyên dưới dạng thập phân không dấu	"%2u", "%02u"
%x	int, short, unsigned int, unsigned short	Xuất ra một số nguyên dưới dạng thập lục phân	%04x"
%o	int, short, unsigned int, unsigned short	Xuất ra một số nguyên dưới dạng bát phân	"%06o", "%03o"
%f	float	Xuất ra một số thực	"%5.2f"

Một số định dạng nhập, xuất

%e	float	Xuất ra một số thực dưới dạng số mũ	"%5.3e"
%g	float	Xuất ra một số thực dưới dạng phù hợp nhất	"%g"
%ld	long	xuất ra số nguyên chính xác kép ở dạng thập phân	"%-10ld"
%lu	unsigned long	xuất ra số nguyên chính xác kép ở dạng thập phân không dấu	"%10lu"
%lo	long, unsigned long	Xuất ra số nguyên chính xác kép trong hệ bát phân	"%12lo"
%lx	long, unsigned long	xuất ra số nguyên chính xác kép ở hệ thập lục phân	"%08lx"
%lf	double, unsigned long	xuất ra số thực chính xác gấp đôi	"%8.3lf"
%a	double	Xuất ra một số thực chính xác kép thập lục phân	"%a"

Các lệnh điều khiển chương trình

- Lệnh đơn
- Lệnh ghép
- Lệnh if [else]
- Lệnh while
- Lệnh do-while
- Lệnh for
- Lệnh break và continue
- Lệnh switch
- Lệnh goto

Lệnh đơn và lệnh ghép

- Lệnh đơn
 - Một biểu thức lệnh (gán, tăng giá trị) kết thúc bởi dấu “;” là một câu lệnh đơn.
 - Một lời gọi hàm đi sau bằng một dấu “;” cũng là một lệnh đơn.
- Lệnh ghép
 - Là tập hợp các câu lệnh được bao bởi hai dấu “{“ và “}”
 - Các lệnh thành phần bên trong câu lệnh ghép sẽ được thực hiện một cách tuần tự.

Lệnh đơn và lệnh ghép

- Ví dụ

```
d = b*b - 4*a*c; // Lệnh đơn
if(d>=0) {
    x1 = (-b - sqrt(d))/(2*a);
    x2 = (-b + sqrt(d))/(2*a);
    printf("nghiem x1 = %4.2f, x2 = %4.2f",x1,x2);
} //Lệnh ghép
```

Lời chú thích

- Lời giải thích, không ảnh hưởng đến chương trình
- Chú thích được đặt giữa cặp /* và */, có thể trên một hoặc nhiều dòng.
- Với các chương trình dịch của C++ có thể sử dụng // để ghi một chú thích trên một dòng.

Lệnh if [else]

- Dùng để lựa chọn quyết định

- Cú pháp

if (expr)

<lệnh 1>;

[else <lệnh 2>]

- Phần lệnh sau if hay else có thể là lệnh đơn, lệnh ghép hay cấu trúc điều khiển chương trình khác

- Ví dụ:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
void main()
{ float a,b,c, p,q,s;
    printf("Nhập số đo 3 cạnh của tam giác ");
    printf("\na = "); scanf("%f", &a);
    printf("\nb = "); scanf("%f", &b);
    printf("\nc = "); scanf("%f", &c);
    if(a+b>c) && (a+c>b) && (b+c>a)
    {
        p = a+b+c;    q = p/2;
        s = sqrt(q*(q-a)*(q-b)*(q-c));
        printf("\n\nChu vi là %.1f,\n
                diện tích là %.2f ",p,s);
    }
else
    printf("\nBộ số da cho là ba cạnh tam giác");
getch(); }
```

Lệnh for

- Cú pháp

for ([expr 1]; [expr 2]; [expr 3])
 <lệnh>

- Có thể có nhiều biến điều khiển
- Một trong 3 biểu thức thành phần của vòng for có thể bỏ qua, nhưng phải có dấu “;”
- Khi expr 2 không có mặt, thì biểu thức điều kiện luôn đúng.
- Ví dụ

Lệnh for

```
1 #include <stdio.h>
2 #include <conio.h>
3 void main(){
4     int n,i;
5     printf("Nhập n:"); scanf("%d",&n);
6     for(i=1; i<=n; i++)
7         printf(" %d",i);
8 }
```

Lệnh while

- Cú pháp

while (expr)

`<lệnh>;`

- Ví dụ

```
#include <stdio.h>
#include <conio.h>
const int Max =10; // gioi han
void main() {
    int n, k ;  float s, pt;
    printf("\nNhap n = ");
    scanf("%d", &n);
    while((n<=0) || (n>Max)) {
        printf("\nNhap lai n (0<n<=%d) : ", Max);
        scanf("%d", &n);
    }
}
```

Lệnh do-while

- Cú pháp

```
do      {  
    <lệnh>  
}while (expr);
```

- Ví dụ

Lệnh do-while

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
const int ESC =27; // ma phim ESC
void main() {
    int ch;
    do{
        printf("\n Hay nhap mot ki tu : ")
        ch = getch()
        printf("\nKi tu %c co ma %d", ch, ch);
    }while(ch !=ESC)
}
```

Lệnh break

- Lệnh break: Cho phép chương trình thoát ngay khỏi vòng lặp trong cùng chứa nó.

- Ví dụ

```
1 #include <stdio.h>
2
3 int main () {
4     int a = 10;
5
6     while( a < 20 ) {
7         printf("Gia tri cua a: %d\n", a);
8         a++;
9
10        if( a > 15) {
11            /* ket thuc vong lap khi a lon hon 15 */
12            break;
13        }
14    }
15
16    return 0;
17 }
```

Lệnh continue

- Lệnh continue: Khi gặp lệnh continue chương trình không kết thúc nốt vòng lặp chứa nó mà bỏ qua các câu lệnh trong thân vòng lặp để bắt đầu một lần lặp mới.

- Ví dụ

```
#include <stdio.h>
#include <conio.h>
void main() {
    int n,i, tonguoc=0;
    do{
        printf("Nhập số n : ");
        scanf("%d", &n);
    }while(n<2);
    printf("\nCác uoc cua %d là\n", n);
    for(i=1;i<n/2; i++)
    {   if(n%i)
        continue;
        printf("%d, ", i);
        tonguoc+=i;
    }
    printf("tổng các uoc là %d", tonguoc);
}
```

Lệnh switch

- Cú pháp

```
switch (expr) {  
    case <value>: <lệnh 1>; [break;]  
    case <value>: <lệnh 2>; [break;]  
    .....  
    case <value>: <lệnh n>; [break;]  
    [default: <lệnh n+1>;]  
}
```

- Ví dụ

```
#include <stdio.h>
#include <conio.h>
void main()
{
int a,b;
char tt; // dấu toán tử
printf("\nnhap bieu thuc don gian :");
scanf("%d%c%d", &a, &tt, &b);
switch(tt)
{
case '+': printf("\n %d %c %d = %d ",a,tt,b, a+b);
break;
case '-': printf("\n %d %c %d = %d ",a,tt,b, a-b);
break;
case 'x':
case '*': printf("\n %d %c %d = %d ",a,tt,b, a*b);
break;
case ':':
case '/': if(b!=0)
                printf("\n %d %c %d = %d ",a,tt,b, a/b);
            else
                printf("loi chia cho 0");
            break;
default: printf("\n\nkhong hieu phep toan %c",tt);
}
getch();
}
```

Chương 3

HÀM

Giới thiệu

- Tổng quát về chương trình con
 - Mô đun hóa chương trình
 - Truyền tham số
- Các môđun chương trình trong C
 - Các môđun trong C gọi là hàm
 - Hàm trong thư viện chuẩn và hàm người dùng tự định nghĩa
 - Cách gọi hàm

Ưu điểm

- Sử dụng chương trình con khiến code sáng sủa và gọn gàng hơn.
- Dễ dàng quản lý, nâng cấp và tìm lỗi chương trình.
- Viết 1 lần và gọi được ở nhiều nơi: Khi bạn dùng hàm thì chỉ phải viết một lần và gọi tới nó bất cứ khi nào muốn. Cũng có thể đóng gói các hàm đó để sử dụng cho các chương trình khác

Hoạt động của hàm trong C

- Chương trình của sẽ nhảy tới nơi định nghĩa hàm và thực thi các lệnh từ trên xuống dưới ở trong hàm.
- Khi hàm thực hiện xong, chương trình tiếp tục quay về thực hiện các lệnh phía sau lời gọi hàm.

Bài 27. Hàm trong C

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

```
void functionName()
{
    ...
}

int main()
{
    ...
    functionName();
    ...
}
```

Khai báo và định nghĩa hàm

- Khai báo hàm (function prototype – nguyên mẫu)
 - [Kiểu giá trị trả về] tên hàm ([danh sách tham số]);
- Định nghĩa hàm
 - [Kiểu giá trị trả về] tên hàm ([danh sách tham số]) {
 - Các khai báo
 - ...
 - Các câu lệnh
 - }
- Kiểu giá trị trả về có thể là các kiểu dữ liệu hợp lệ của C, **trừ một biến mảng**.
- Kiểu giá trị là void thì hàm không trả về giá trị nào cả.
- Nếu không xác định kiểu giá trị trả về thì trình biên dịch ngầm định là int.
- Danh sách tham số mô tả kiểu dữ liệu cùng thứ tự các tham số hàm nhận được khi nó được gọi.
- Nếu hàm không nhận tham số nào thì danh sách tham số là void
- Khai báo và cài đặt hàm không được nằm trong hàm khác.
- Hàm có kiểu giá trị trả về là void kết thúc khi gặp dấu đóng ngoặc “}” cuối cùng hay câu lệnh **return**;
- Nếu hàm trả lại kết quả thì câu lệnh **return biểu thức**; sẽ trả lại giá trị tính được của biểu thức cho hàm gọi nó.

Lệnh return trong C

- Câu lệnh return được sử dụng để trả về một giá trị nào đó hoặc chỉ đơn giản là chuyển điều khiển cho hàm gọi. Đôi khi câu lệnh này cũng được sử dụng như một hành động kết thúc cho một đoạn chương trình đang được thực thi.
- Câu lệnh return có thể được sử dụng theo hai cách sau:
 - Dạng đầu tiên của câu lệnh **return** được sử dụng để kết thúc hàm và chuyển điều khiển cho hàm đang gọi. Không có giá trị nào từ hàm được gọi được trả về khi dạng **return** này được sử dụng.
 - Dạng thứ hai của câu lệnh trả về được sử dụng để trả về các giá trị từ một hàm. Biểu thức trả về sau có thể là bất kỳ lệnh gọi hàm, biến, hằng số, mảng, con trỏ...

return 0 và return 1 trong hàm main()

- Hai giá trị trả về này của hàm main() có ý nghĩa như sau:
 - *return 0* để kết thúc chương trình theo cách bình thường. Khi đó dù chương trình có lỗi hay không thì C vẫn ngầm định là chương trình kết thúc mà không có lỗi.
 - *return 1* để kết thúc chương trình theo cách bất thường. Khi chương trình xảy ra lỗi thì lỗi này sẽ được trả về khi kết thúc chương trình.

Hàm nguyên mẫu

- Hàm nguyên mẫu thông báo cho chương trình biên dịch biết kiểu dữ liệu hàm trả lại, số lượng, kiểu và thứ tự của các tham số được truyền cho hàm.
- Trình biên dịch dùng hàm nguyên mẫu để kiểm tra lời gọi hàm.
- *Nếu không có hàm nguyên mẫu, trình biên dịch tự động tạo ra hàm nguyên mẫu khi gấp hàm lần đầu tiên thông qua định nghĩa hàm hoặc lời gọi hàm.*
 - Nếu xây dựng hàm sau hàm main(), cần khai báo nguyên mẫu.
 - Các tệp tiêu đề (header file)

Các loại hàm trong C

- Hàm không có tham số, không có giá trị trả về. Ví dụ: *void checkPrimeNumber()*, trong hàm không có lệnh return.
- Hàm không có tham số, có giá trị trả về: *void checkPrimeNumber()*, trong hàm có lệnh return.
- Hàm có tham số, không có giá trị trả về. Ví dụ: *void checkPrimeAndDisplay(int n)*, trong hàm không có lệnh return.
- Hàm có tham số, có giá trị trả về. Ví dụ: *void checkPrimeAndDisplay(int n)*, trong hàm có lệnh return.

Tham số trong lời gọi hàm

- Khái niệm
 - Tham số hình thức
 - Tham số thực sự
 - Truyền tham số theo trị và truyền theo tham biến.
- Truyền tham số trong C
 - Tất cả các tham số được truyền theo trị
- Tham số hình thức của hàm là con trỏ
 - Tham số thực sự tương ứng phải là một địa chỉ của một biến hay tên của một mảng.
 - Có thể thay đổi giá trị của biến với địa chỉ được truyền.
- Tham số của hàm main()
 - Sử dụng khi viết các chương trình có tham số dòng lệnh

```
void main(int argc, char *argv[]) {...}
```
 - Trong đó argc là số lượng các tham số dòng lệnh (cả tên chương trình), argv là mảng các xâu ký tự tương ứng với các tham số dòng lệnh.

Cấp lưu trữ và phạm vi của các đối tượng

- Từ khóa **auto**
 - Dùng để khai báo các biến cục bộ
 - Ít khi sử dụng vì các biến cục bộ đã mặc định là auto.
- Từ khóa **register**
 - Đặt trước khai báo các biến tự động để yêu cầu trình biên dịch duy trì giá trị của biến đó trong thanh ghi của máy tính.
 - Dùng cho các biến kiểu int và char
- Từ khóa **static**
 - Đi với biến cục bộ thông báo cho trình biên dịch cung cấp một vùng nhớ thường xuyên cho biến này.
 - Giá trị của biến tĩnh cục bộ được duy trì giữa các lần gọi hàm.
- Từ khóa **extern**
 - Cho phép sử dụng các biến hàm trên phạm vi nhiều tệp

Ví dụ

```
#include <stdio.h>
#include <conio.h>
void nhapmang(int a[], int n);
void inmang(int a[], int n);
void tong(int A[], int B[], int C[], int n);
void main() {
    clrscr();
    const int max = 20; //
    int A[max], B[max], C[max];
    int n;
    do{
        printf("\nNhap so phan tu mang = ");
        scanf("%d", &n);
    } while(n<1 || n>max);
    printf("\nNhap A \n");
    nhapmang(A, n);
    printf("\nNhap B \n");
    nhapmang(B, n);
    tong(A, B, C, n);
    printf("\nmang A: ");
    inmang(A, n);
    printf("\nmang B: ");
    inmang(B, n);
    printf("\nmang C: ");
    inmang(C, n);
    getch();
}
```

Ví dụ

```
void nhapmang(int a[], int n) {
    int i;
    printf("\nNhập mảng có %d phần tử\n", n);
    for(i=0; i<n; i++)
    {
        printf("Nhập phần tử thứ [%d] = ", i);
        scanf("%d", &a[i]);
    }
}
void inmang(int a[], int n) {
    int i;
    for(i=0; i<n; i++)
    printf("%d ", a[i]);
}
void tong(int A[], int B[], int C[], int n) {
    int i;
    for (i = 0; i<n; i++) C[i]=A[i]+B[i];
}
```

Con trỏ hàm

- Khái niệm
 - Con trỏ hàm là một loại biến con trỏ dùng để chứa vị trí của hàm trong bộ nhớ.
 - Con trỏ hàm có thể dùng thay cho tên hàm
 - Sử dụng con trỏ hàm cho phép các hàm được truyền như là tham số cho các hàm khác.
- Khai báo

[Kiểu giá trị] (*tên biến con trỏ hàm)([danh sách tham số]);

- Tham số hình thức của hàm là con trỏ hàm
- Ví dụ

- Hàm hoán đổi giá trị của 2 biến a và b

- Nếu viết hàm đổi chỗ như sau:

```
void doicho(int x, int y) {  
    int tg;  
    tg = x;  
    x=y;  
    y=tg;  
}
```

- Trong hàm main() có các lệnh như sau:

```
int a = 4;  
int b = 6;  
printf ("\ntruoc khi goi ham doi cho a=%d, b=%d",a,b);  
doicho(a,b);  
printf ("\nsau khi goi ham doi cho a=%d, b=%d",a,b);
```

- Thì kết quả sẽ là

trước khi gọi hàm doi cho a=4,b=6

sau khi gọi hàm doi cho a=4,b=6

- Khi đó hàm doi chưa thực hiện được mục tiêu của nó
- Để khắc phục điều cần định nghĩa hàm với tham số là con trỏ và khi gọi các bạn hãy truyền cho nó địa chỉ của tham số thực sự.

- Hàm doicho2

```
void doicho2(int * x, int *y) {  
    int tg;  
    tg = *x;  
    *x = *y;  
    *y = tg;  
}
```

- Lệnh trong hàm main()

```
int a = 4;  
int b = 6;  
printf ("\ntruoc khi goi ham doi cho a=%d, b=%d",a,b);  
doicho(&a,&b);  
printf ("\nsau khi goi ham doi cho a=%d, b=%d",a,b);
```

- Thị kết quả sẽ là

truoc khi goi ham doi cho a = 4, b = 6
sau khi goi ham doi cho a = 6 , b = 4

Đệ quy

- Hàm đệ quy là hàm gọi đến chính nó trực tiếp hay gián tiếp thông qua các hàm khác.
- Hàm đệ quy chỉ giải bài toán trong trường hợp đơn giản nhất, gọi là trường hợp cơ sở.
- Muốn xây dựng hàm đệ quy phải đưa ra được trường hợp cơ sở (còn gọi là điểm dừng của chương trình đệ quy)
- Ví dụ

Đệ quy

```
#include <stdio.h>

int factorial(int n) {
    if (n == 1)
        return 1;
    else
        return (n * factorial(n - 1));
}

int main() {
    printf("Giai thua cua 5 la: %d", factorial(5));
    return 0;
}
```

Chương 4

Con trỏ và mảng

Mảng

- Mảng là một dãy liên tiếp các phần tử trong bộ nhớ, mỗi phần tử chứa dữ liệu cùng một kiểu.
- Kích thước của mảng chính là số phần tử trong mảng
- Mảng có thể có một hay nhiều chiều.

Khai báo mảng

- Mảng 1 chiều

`<type> <array_name> [size];`

- *type* là kiểu dữ liệu
- *array_name* là tên của mảng
- *size* là số thành phần trong mảng, giá trị này phải là một hằng số.
- Ví dụ: int a[100];

- Truy nhập mảng

- Các phần tử được truy nhập thông qua tên biến mảng và chỉ số tương ứng
- Phần tử đầu tiên có chỉ số là 0.
- Chỉ số có thể dùng các đại lượng với giá trị là số
- Ví dụ: a[2]=7;

Khai báo mảng (tiếp)

- Mảng nhiều chiều

- Một mảng nhiều chiều là mảng một chiều trong đó mỗi phần tử là một mảng với số chiều không ít hơn 1.
- Trong khai báo phải mô tả kích thước của tất cả các chiều đặt trong dấu ngoặc “[]”
- Ví dụ: float a[10][20];
int b[2][50];

Nhập dữ liệu cho biến mảng

- 2 cách
 - Sử dụng hàm scanf để nhập trực tiếp giá trị cho phần tử.
 - Sử dụng biến trung gian có cùng kiểu.
- Chú ý:
 - Áp dụng cách 2 cho các mảng số thực.
 - Ví dụ

```
#include <stdio.h>
#include <conio.h>
void main() {
    clrscr();
    const int max = 20;
    int A[max];
    int n, i;
    do{
        printf("\nNhap so phan tu mang = ");
        scanf("%d", &n);
    }while(n<1 || n>max); //nhập số pt mảng 1<=n<=max

    printf("\nNhap mang co %d phan tu \n", n);
    for(i=0; i<n; i++)
    {
        printf("A[%d] = ", i);
        scanf("%d", &A[i]);
    }
    printf("\nCac phan tu mang theo thu tu xuoi la \n");
    for(i=0; i<n; i++)
        printf("%d, ", A[i]);

    printf("\nCac phan tu mang theo thu tu nguoc la \n");
    for(i=n-1; i>=0; i--)
        printf("%d, ", A[i]);
    getch();
}
```

```
#include <stdio.h>
#include <conio.h>
void main() {
    clrscr(); // xoa man hinh
    const int max = 5; // kich thuoc toi da
    int A[max][max];
    int n,m,i,j;
    do{printf("\nNhap so dong cua mang = ");
        scanf("%d", &n);
        printf("\nNhap so cot cua mang = ");
        scanf("%d", &m);
    } while(n<1 || n>max || m<1 || m>max);
    printf("\nNhap mang co %d dong, %d cot \n", n, m);
    for(i=0; i<n; i++)
        for(j=0; j<m; j++)
    {
        printf("A[%d] [%d]= ", i, j);
        scanf("%d", &A[i][j]);
    }
    printf("\nCac phan tu mang la \n");
    for(i=0; i<n; i++)
    {
        printf("\n");
        for(j=0; j<m; j++)
            printf("%d ", A[i][j]);
    }
    getch();
}
```

Xâu ký tự

- Khai báo

- Một biến xâu ký tự được khai báo tương tự như một mảng một chiều chứa các ký tự.
- Ví dụ: char str[100];
- Trong xâu ký tự có ký tự kết thúc xâu (NUL hay '\0').
- Không tồn tại các phép toán gán, so sánh nội dung của xâu này với xâu khác.
- Cần phân biệt ký tự ‘A’ và xâu “A”.

- Nhập và in xâu

- Sử dụng scanf và printf
- Sử dụng gets và puts

- Ví dụ

Xâu ký tự

```
/* Chuong trinh nhap va in ra ten*/  
  
#include <stdio.h>  
#include <conio.h>  
  
void main(void)  
{  
    char cname[30];  
    printf("Cho biet ten cua ban: ");  
    scanf("%s", cname);  
    printf("Chao ban %s\n", cname);  
    getch();  
}
```

```
/* Chuong trinh nhap va in ra ten*/  
  
#include <stdio.h>  
#include <conio.h>  
  
void main(void)  
{  
    char cname[30];  
    puts("Cho biet ten cua ban: ");  
    gets(cname);  
    puts("Chao ban ");  
    puts(cname);  
    getch();  
}
```

Xâu ký tự

```
/* Chuong trinh nhap va in ra ten*/  
  
#include <stdio.h>  
#include <conio.h>  
  
void main(void)  
{  
    char cname[30];  
    char chao[] = "Chao ban";  
    printf("Cho biet ten cua ban: ");  
    gets(cname);  
    printf("%s %s.\n", chao, cname);  
    getch();  
}
```

Xâu ký tự

- Một số hàm trong xâu ký tự
 - strlen(): trả về độ dài của xâu ký tự (không bao gồm ký tự null '\0').
 - strcpy(): Sao chép nội dung của xâu ký tự nguồn vào xâu ký tự đích.
 - strcat(): Nối xâu ký tự nguồn vào xâu ký tự đích.
 - strcmp(): So sánh 2 xâu ký tự. Trả về giá trị âm nếu xâu ký tự thứ nhất nhỏ hơn xâu ký tự thứ 2, giá trị dương nếu ngược lại, giá trị 0 nếu chúng bằng nhau.
 - strncpy(): Sao chép tối đa n ký tự từ xâu ký tự nguồn vào xâu ký tự đích.
 - strstr(): tìm kiếm xâu ký tự con trong xâu ký tự chính trả về con trỏ đến vị trí đầu tiên xuất hiện.

Con trỏ (pointer)

- Khái niệm và cách khai báo

- Một con trỏ là một biến/hằng có giá trị (nội dung) là địa chỉ của một đối tượng khác;
- Đối tượng ở đây có thể là một biến hoặc một hàm, không thể là hằng.
- Khai báo biến con trỏ
 - **type *ptr_name;**
 - **type** là kiểu dữ liệu của biến mà con trỏ chứa địa chỉ.
 - **ptr_name** là tên biến con trỏ.
- Với khai báo trên, biến con trỏ có giá trị là NULL

- Toán tử & và *

- Toán tử & cho địa chỉ một biến
- Toán tử * áp dụng lên biến con trỏ cho nội dung của địa chỉ mà con trỏ đang giữ.
- Ví dụ: int n=110, *p;
thì p=&n; gán địa chỉ của n cho p và *p cho giá trị là 110

Các phép toán trên con trỏ

- Phép gán trên hai con trỏ cùng kiểu
 - Chỉ có thể gán địa chỉ của một biến nguyên cho một con trỏ kiểu nguyên.
 - Nếu muốn thực hiện phép gán giữa con trỏ với biến, hay con trỏ với con trỏ không cùng kiểu thì phải thực hiện ép kiểu
 - Ví dụ int m, *pi; float *pf;
thì pf=(float*) &m; pf=(float*)pi;
- Phép cộng con trỏ với số nguyên
 - Khi cộng hay trừ con trỏ với một số nguyên n, ta sẽ được một địa chỉ mới chỉ đến một biến khác nằm cách biến trước đó n vị trí.
- Phép trừ 2 con trỏ cùng kiểu
 - Cho kết quả là một số nguyên biểu thị “khoảng cách” giữa hai biến con trỏ.
- Các phép toán quan hệ trên các con trỏ cùng kiểu
 - Có thể so sánh các con trỏ cùng kiểu dựa trên vị trí ô nhớ tương ứng
 - Với hai con trỏ khác kiểu cần phải chuyển đổi kiểu.
- Ví dụ

Các phép toán trên con trỏ

```
/* Cong hang so */

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

void main(void)
{
    int ix = 6, iy = 7;
    int *px, *py;
    printf("x = %d, y = %d\n", ix, iy);
    px = &ix;
    py = &iy;
    *px += 10;
    *py += 10;
    printf("x = %d, y = %d\n", ix, iy);
    getch();
}
```

Con trỏ void*

- Kiểu dữ liệu void
 - Là kiểu dữ liệu rỗng
 - Một biến kiểu void không tương ứng với bất cứ vùng nhớ nào.
- Con trỏ kiểu void*
 - Có thể nhận địa chỉ của bất kỳ vùng nhớ nào
 - Không thể thực hiện các phép tính số học trên con trỏ kiểu void*.
- Con trỏ đa cấp
 - Bản thân biến con trỏ cũng có địa chỉ, do đó có thể chứa địa chỉ của nó trong một đối tượng khác.
 - Đối tượng có giá trị là địa chỉ của một biến con trỏ là con trỏ đa cấp.
 - Ví dụ : int *pi; int **p=π

Liên hệ giữa con trỏ và mảng

- Con trỏ và mảng một chiều
 - Với khai báo `int a[10];`
 - Cấp phát **vùng nhớ 20 byte** cho 10 số nguyên $a[0]$, $a[1]$, ..., $a[9]$ có **địa chỉ xác định** bởi a .
 - Khi đó, với khai báo `int *p;` thì ta có thể gán $p=a$; và ta có thể sử dụng:
 - $p[0]$, $p[1]$, ..., $p[9]$ hay $*(p+i)$ với $i=0,..,9$
tương ứng là $a[0]$, $a[1]$, ..., $a[9]$.
 - Còn a , $a+1$, ..., $a+9$
tương ứng là **địa chỉ** của $a[0]$, $a[1]$, ..., $a[9]$.

Liên hệ giữa con trỏ và mảng

- Mảng các con trỏ
 - Là mảng chứa các phần tử có kiểu con trỏ.
 - Nếu các con trỏ thành phần chứa địa chỉ của các biến nào đó, thì chúng ta không cần quan tâm đến vị trí các biến đó có liên tiếp trong bộ nhớ hay không.
 - Nếu các con trỏ thành phần lại được gán địa chỉ của các mảng khác thì ta được một mảng của các mảng. Các mảng con này có thể nằm ở vị trí bất kỳ trong bộ nhớ.
- Ví dụ

Liên hệ giữa con trỏ và mảng

```
#include <stdio.h>

int main() {
    int i, arr[6],sum = 0;
    printf("Nhập 6 số nguyên:\n");
    for(i = 0; i < 6; ++i)
    {
        // (arr + i) là tuong duong voi &arr[i]
        scanf("%d", (arr + i));

        // *(arr + i) là tuong duong voi arr[i]
        sum += *(arr + i);
    }
    printf("Sum = %d", sum);
    return 0;
}
```

Cấp phát động

- Cho phép chương trình sử dụng đúng khối lượng bộ nhớ mà chương trình cần;
- Khi không cần dùng thì có thể giải phóng để cho các công việc khác có thể sử dụng.
- Vùng Heap dùng cho cấp phát động.
- Hàm malloc()
 - Cú pháp: malloc(size);
 - Dùng để cấp phát một vùng nhớ có kích thước size byte
 - Nếu thành công, trả về con trỏ trỏ tới vùng nhớ vừa được cấp phát, ngược lại trả về giá trị NULL
 - Sử dụng phép chuyển đổi kiểu để đảm bảo các yêu cầu của phép gán

Cấp phát động (tiếp)

- Hàm calloc()
 - Cú pháp: calloc(nitems, size);
 - Dùng để cấp phát một vùng nhớ có kích thước $nitems * size$ byte và xóa trắng vùng nhớ này.
 - Nếu thành công, trả về con trỏ trỏ tới vùng nhớ vừa được cấp phát, ngược lại trả về giá trị NULL
 - Sử dụng phép chuyển đổi kiểu để đảm bảo các yêu cầu của phép gán

Cấp phát động (tiếp)

- Hàm free()
 - Cú pháp: free(address);
 - Dùng để giải phóng một vùng nhớ đã được cấp phát trước đó bằng malloc() hoặc calloc().
 - Địa chỉ vùng nhớ giải phóng được truyền làm tham số của free().

- Viết chương trình nhập và in mảng 1 chiều gồm n phần tử là số nguyên ($n \leq 100$), sau đó:
 - Tìm phần tử lớn nhất, nhỏ nhất trong mảng
 - Thống kê số âm, số dương, số 0 có trong mảng;
 - Thống kê số nguyên tố có trong mảng;
 - Sắp xếp mảng theo thứ tự tăng dần của các phần tử;
 - Xoá phần tử ở vị trí thứ k trong mảng;
 - Chèn phần tử vào vị trí thứ k trong mảng.

- Viết chương trình nhập và in mảng 2 chiều chứa các số nguyên có kích thước mxn, sau đó:
 - Tìm GTLN, GTNN trong mảng;
 - Tìm các số hoàn hảo trong mảng;
 - Sắp xếp cột, hàng thứ k theo thứ tự tăng;
 - Xoá cột, hàng thứ k;
 - Chèn cột, hàng vào vị trí thứ k.

- Viết chương trình nhập và in 02 mảng 2 chiều chứa các số nguyên có kích thước mxn, sau đó kiểm tra và thực hiện phép toán trên ma trận:
 - Cộng, trừ 2 ma trận
 - Nhân 2 ma trận;
 - Chuyển vị ma trận;
 - In ra các phần tử trên đường chéo chính, chéo phụ;
 - In ma trận tam giác trên, ma trận tam giác dưới.