

Network Programming



Chương 2: Lập trình C

ÔN TẬP LẬP TRÌNH C

6.2.1. Tập ký tự

- Tập ký tự trong C
 - 26 chữ cái hoa: A B C ... X Y Z
 - 26 chữ cái thường: a b c ... x y z.
 - 10 chữ số: 0 1 2 3 4 5 6 7 8 9.
 - Các kí hiệu toán học: + - * / = < >
 - Các dấu ngăn cách: . ; , : space tab
 - Các dấu ngoặc: () [] { }
 - Các kí hiệu đặc biệt: _ ? \$ & # ^ \ ! ' " ~
.v.v.

6.2.2. Từ khóa

- Từ khóa hay dùng trong Turbo C

break	case	char	const	continue	default
do	double	else	enum	float	for
goto	if	int	interrupt	long	return
short	signed	sizeof	static	struct	switch
typedef	union	unsigned	void	while	

6.2.3. Định danh

- Quy tắc đặt tên định danh trong C
 - Các kí tự được sử dụng: chữ cái, chữ số và dấu gạch dưới “_” (*underscore*)
 - Bắt đầu của định danh phải là chữ cái hoặc dấu gạch dưới “_”, không được bắt đầu định danh bằng chữ số.
 - Định danh do người lập trình đặt không được trùng với các từ khóa của C
- Chú ý: *C là ngôn ngữ có phân biệt chữ hoa và chữ thường*

6.2.5. Hằng số

- Biểu diễn hằng xâu ký tự:
 - Hằng xâu kí tự được biểu diễn bởi dãy các kí tự thành phần có trong xâu đó và được đặt trong cặp dấu nháy kép.
- Ví dụ:
 - “ngon ngu lap trinh C”
 - “Tin hoc dai cuong”
 - “Dai hoc Bach Khoa Ha Noi”

6.2.5. Hằng số

- Định nghĩa:
 - hằng (*constant*) là đại lượng có giá trị không đổi trong chương trình.
- Biểu diễn hằng số nguyên: trong C, một hằng số nguyên có thể biểu diễn dưới 3 dạng
 - Dạng thập phân
 - Dạng thập lục phân
 - Dạng bát phân

Giá trị thập phân	Giá trị thập lục phân	Giá trị bát phân
2007	0x7D7	03727
396	0x18C	0614

6.2.5. Hằng số

- Biểu diễn hằng số thực: trong C, một hằng số thực có thể biểu diễn dưới 2 dạng
 - Dạng số thực dấu phẩy tĩnh
 - Dạng số thực dấu phẩy động
- Ví dụ

Số thực dấu phẩy tĩnh	Số thực dấu phẩy động
3.14159	31.4159 E-1
123.456	12.3456 E+1 hoặc 1.23456 E+2

6.2.6. Biến

- Định nghĩa:
 - Biến (*variable*) là đại lượng mà giá trị có thể thay đổi trong chương trình.
- Chú ý:
 - Hằng số và biến được sử dụng để lưu trữ dữ liệu trong chương trình
 - Hằng số và biến phải thuộc một kiểu dữ liệu nào đó
 - Hằng số và biến đều phải đặt tên theo quy tắc

6.2.7. Hàm

- Mô tả:
 - Hàm (function) là một chương trình con có chức năng nhận dữ liệu đầu vào (các tham số đầu vào), thực hiện một chức năng nào đó và đưa ra các kết quả.

Hàm	Ý nghĩa	Ký hiệu toán học	Ví dụ
$\text{Pow}(x,y)$	X mũ y	X^y	$\text{Pow}(2,3)=8$
$\text{Sin}(x)$	Sin của x	$\text{Sin}x$	$\text{Sin}(0)=0$
$\text{Cos}(x)$	Cos của x	$\text{Cos}x$	$\text{Cos}(0)=1$

6.2.8. Biểu thức

- Định nghĩa:
 - Biểu thức là sự ghép nối các toán tử (*operator*) và các toán hạng (*operand*) theo một quy tắc xác định.
 - Các toán hạng có thể là biến, hằng
 - Các toán tử rất đa dạng: cộng, trừ, nhân, chia..
- Ví dụ: biểu thức tính thể tích hình chữ nhật
chieu_dai * chieu_rong * chieu_cao
 - chieu_dai, chieu_rong, chieu_cao là các hằng hoặc biến số đóng vai trò toán hạng
 - Phép * đóng vai trò toán tử

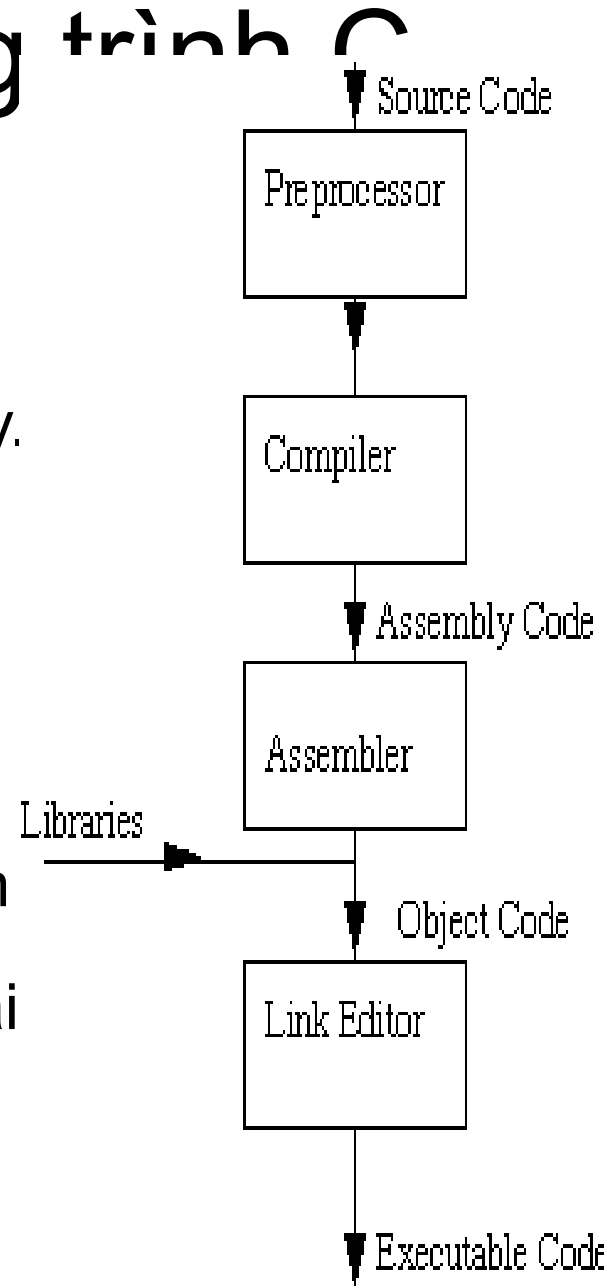
6.3. Cấu trúc cơ bản của chương trình C

- **Gồm 6 phần có thứ tự như sau:**

Phần 1: Khai báo tệp tiêu đề: <code>#include</code>
Phần 2: Định nghĩa kiểu dữ liệu mới: <code>typedef ...</code>
Phần 3: Khai báo các hàm nguyên mẫu
Phần 4: Khai báo các biến toàn cục
Phần 5: Hàm main()
Phần 6: Nội dung các hàm đã khai báo

6.4. Biên dịch chương trình

- **Preprocessor**
 - Loại bỏ các chú thích
 - Dịch các chỉ thị tiền xử lý bắt đầu là #
- **C Compiler**
 - Biên dịch mã nguồn thành mã assembly.
- **Assembler**
 - Tạo ra mã object.
 - Trên UNIX → file .o
 - Trên MS-DOS → file.OBJ
- **Link Editor**
 - Nếu tệp nguồn tham chiếu đến các hàm thư viện/hàm được định nghĩa thì *Link editor* kết hợp các hàm này với hàm mai để tạo ra tệp có thể thực thi được
 - Trong MS-DOS là file .exe



7.1. Các kiểu dữ liệu chuẩn trong C

Kiểu dữ liệu	Ý nghĩa	Kích thước	Miền dữ liệu
unsigned char	Kí tự không dấu	1 byte	$0 \div 255$
char	Kí tự có dấu	1 byte	$-128 \div 127$
unsigned int	Số nguyên không dấu	2 byte	$0 \div 65,535$
int	Số nguyên có dấu	2 byte	$-32,768 \div 32,767$

7.1. Các kiểu dữ liệu chuẩn trong C

Kiểu dữ liệu	Ý nghĩa	Kích thước	Miền dữ liệu
unsigned long	Số nguyên không dấu	4 byte	0 ÷ 4,294,967,295
long	Số nguyên có dấu	4 byte	-2,147,483,648 ÷ 2,147,483,647
float	Số thực dấu phẩy động, độ chính xác đơn	4 byte	$\pm 3.4E-38 \div$ $\pm 3.4E+38$
double	Số thực dấu phẩy động, độ chính xác kép	8 byte	$\pm 1.7E-308 \div \pm$ $1.7E+308$

7.2.1. Khai báo và khởi tạo biến

- Một biến trước khi sử dụng phải được khai báo
- Cú pháp khai báo:

kieu_du_lieu ten_bien;

Hoặc:

kieu_du_lieu ten_bien₁, ..., ten_bien_N;

- Ví dụ: Khai báo một biến x thuộc kiểu số nguyên 2 byte có dấu (int), biến y, z, t thuộc kiểu thực 4 byte (float) như sau:

```
int          x;  
float        y, z, t;  
x = 3;  y = x + 1;
```


7.2.2. Khai báo hằng

- Cách 1: Dùng từ khóa **#define**:

- Cú pháp:

define ten_hang gia_tri

- Ví dụ:

```
#define MAX_SINH_VIEN 50
```

```
#define CNTT "Cong nghe thong tin"
```

```
#define DIEM_CHUAN 23.5
```

7.2.2. Khai báo hằng

- Cách 2: Dùng từ khóa **const** :

- Cú pháp:

const kieu_du_lieu ten_hang = gia_tri;

- Ví dụ:

```
const int MAX_SINH_VIEN = 50;
```

```
const char CNTT[20] = "Cong nghe thong tin";
```

```
const float DIEM_CHUAN = 23.5;
```

7.6.1. Các loại biểu thức

- a. Biểu thức số học:
 - Là biểu thức mà giá trị của nó là cái đại lượng số học (số nguyên, số thực).
 - Các toán tử là các phép toán số học (cộng, trừ, nhân, chia...), các toán hạng là các đại lượng số học (số, biến, hằng).
 - Ví dụ:
 - a, b, c là các biến thuộc một kiểu dữ liệu số nào đó.
 - $3 * 3.7$
 - $8 + 6/3$
 - $a + b - c...$

7.6.1. Các loại biểu thức

- b. Biểu thức logic:
 - Là biểu thức mà giá trị của nó là các giá trị logic, tức là một trong hai giá trị: Đúng (*TRUE*) hoặc Sai (*FALSE*).
 - Giá trị nguyên khác 0: Đúng (*TRUE*),
 - Giá trị 0: Sai (*FALSE*).
 - Các phép toán logic gồm có
 - AND: VÀ logic, kí hiệu là **&&**
 - OR: HOẶC logic, kí hiệu là **| |**
 - NOT: PHỦ ĐỊNH, kí hiệu là **!**

7.6.1. Các loại biểu thức

- c. Biểu thức quan hệ:
 - Là những biểu thức trong đó có sử dụng các toán tử quan hệ so sánh như lớn hơn, nhỏ hơn, bằng nhau, khác nhau...
 - Chỉ có thể nhận giá trị là một trong 2 giá trị Đúng (TRUE) hoặc Sai (FALSE)
 - Biểu thức quan hệ là một trường hợp riêng của biểu thức logic.

7.6.1. Các loại biểu thức

- Ví dụ về biểu thức quan hệ:
 - $5 > 7$
 - $9 \neq 10$
 - $2 \geq 2$
 - $a > b$
 - $a+1 > a$
 - $//$ có giá trị logic là sai, FALSE
 - $//$ có giá trị logic là đúng, TRUE
 - $//$ có giá trị logic là đúng, TRUE
 - $//$ giả sử a, b là 2 biến kiểu `int`
 - $//$ có giá trị đúng, TRUE

7.6.1. Các loại biểu thức

- Ví dụ về biểu thức logic:

- `(5 > 7) && (9 != 10)` • `// có giá trị logic là sai, FALSE`
- `0 || 1` • `// có giá trị logic là đúng, TRUE`
- `(5 > 7) || (9 != 10)` • `// có giá trị logic là đúng, TRUE`
- `0` • `// có giá trị logic là sai, FALSE`
- `!0` • `// phủ định của 0, có giá trị logic là đúng, TRUE`
- `3` • `// có giá trị logic là đúng, TRUE`
- `!3` • `// phủ định của 3, có giá trị logic là sai, FALSE`
- `(a > b) && (a < b)` • `// Có giá trị sai, FALSE. Giả sử a, b là 2 biến kiểu int`

7.4.1. Phép toán số học

Toán tử	Ý nghĩa	Kiểu dữ liệu của toán hạng	Ví dụ
-	Phép đổi dấu	Số thực hoặc số nguyên	<code>int a, b;</code> <code>-12; -a; -25.6;</code>
+	Phép toán cộng	Số thực hoặc số nguyên	<code>float x, y;</code> <code>5 + 8; a + x;</code> <code>3.6 + 2.9;</code>
-	Phép toán trừ	Số thực hoặc số nguyên	<code>3 - 1.6; a - 5;</code>
*	Phép toán nhân	Số thực hoặc số nguyên	<code>a * b; b * y;</code> <code>2.6 * 1.7;</code>
/	Phép toán chia	Số thực hoặc số nguyên	<code>10.0/3.0; (bằng 3.33...)</code> <code>10/3.0; (bằng 3.33...)</code> <code>10.0/3; (bằng 3.33...)</code>
/	Phép chia lấy phần nguyên	Giữa 2 số nguyên	<code>10/3; (bằng 3)</code>
%	Phép chia lấy phần dư	Giữa 2 số nguyên	<code>10%3; (bằng 1)</code>

7.4.7. Phép toán trên bit

Toán tử	Ý nghĩa	Kiểu dữ liệu của toán hạng	Ví dụ	
&	Phép VÀ nhị phân	2 số nhị phân	0 & 0	(có giá trị 0)
			0 & 1	(có giá trị 0)
			1 & 0	(có giá trị 0)
			1 & 1	(có giá trị 1)
			101 & 110	(có giá trị 100)
	Phép HOẶC nhị phân	2 số nhị phân	0 0	(có giá trị 0)
			0 1	(có giá trị 1)
			1 0	(có giá trị 1)
			1 1	(có giá trị 1)
			101 110	(có giá trị 111)

7.4.2. Phép toán trên bit

^	Phép HOẶC CÓ LOẠI TRỪ nhị phân	2 số nhị phân	$0 \wedge 0$	(có giá trị 0)
			$0 \wedge 1$	(có giá trị 1)
			$1 \wedge 0$	(có giá trị 1)
			$1 \wedge 1$	(có giá trị 0)
			$101 \wedge 110$	(có giá trị 011)
<<	Phép DỊCH TRÁI nhị phân	Số nhị phân	$a \ll n$	(có giá trị $a \cdot 2^n$)
			$101 \ll 2$	(có giá trị 10100)
>>	Phép DỊCH PHẢI nhị phân	Số nhị phân	$a \gg n$	(có giá trị $a / 2^n$)
			$101 \gg 2$	(có giá trị 1)
~	Phép ĐẢO BIT nhị phân (lấy Bù 1)	Số nhị phân	~ 0	(có giá trị 1)
			~ 1	(có giá trị 0)
			~ 110	(có giá trị 001)

7.4.3. Phép toán quan hệ

Toán tử	Ý nghĩa	Ví dụ
$>$	So sánh lớn hơn giữa 2 số nguyên hoặc thực.	$2 > 3$ (có giá trị 0) $6 > 4$ (có giá trị 1) $a > b$
$>=$	So sánh lớn hơn hoặc bằng giữa 2 số nguyên hoặc thực.	$6 >= 4$ (có giá trị 1) $x >= a$
$<$	So sánh nhỏ hơn giữa 2 số nguyên hoặc thực.	$5 < 3$ (có giá trị 0),
$<=$	So sánh nhỏ hơn hoặc bằng giữa 2 số nguyên hoặc thực.	$5 <= 5$ (có giá trị 1) $2 <= 9$ (có giá trị 1)
$==$	So sánh bằng nhau giữa 2 số nguyên hoặc thực.	$3 == 4$ (có giá trị 0) $a == b$
$!=$	So sánh không bằng (so sánh khác) giữa 2 số nguyên hoặc thực.	$5 != 6$ (có giá trị 1) $6 != 6$ (có giá trị 0)

7.4.4. Phép toán logic

Toán tử	Ý nghĩa	Kiểu dữ liệu của toán hạng	Ví dụ
&&	Phép VÀ LOGIC. Biểu thức VÀ LOGIC bằng 1 khi và chỉ khi cả 2 toán hạng đều bằng 1	Hai biểu thức logic	$3 < 5 \ \&\& \ 4 < 6$ (có giá trị 1) $2 < 1 \ \&\& \ 2 < 3$ (có giá trị 0) $a > b \ \&\& \ c < d$
	Phép HOẶC LOGIC. Biểu thức HOẶC LOGIC bằng 0 khi và chỉ khi cả 2 toán hạng bằng 0.	Hai biểu thức logic	$6 \ \ 0$ (có giá trị 1) $3 < 2 \ \ 3 < 3$ (có giá trị 0) $x \geq a \ \ x == 0$
!	Phép PHỦ ĐỊNH LOGIC một ngôi. Biểu thức PHỦ ĐỊNH LOGIC có giá trị bằng 1 nếu toán hạng bằng 0 và có giá trị bằng 0 nếu toán hạng bằng 1	Biểu thức logic	$!3$ (có giá trị 0) $!(2 > 5)$ (có giá trị 1)

7.4.6. Thứ tự ưu tiên các phép toán

Mức	Các toán tử	Trật tự kết hợp
1	() [] . -> ++ (hậu tố) -- hậu tố	----->
2	! ~ ++ (tiền tố) -- (tiền tố) - *	<-----
	& sizeof	
3	* / %	----->
4	+ -	----->
5	<< >>	----->
6	< <= > >=	----->
7	== !=	----->
8	&	----->
9	^	----->
10		----->
11	&&	----->
12		----->
13	?:	<-----
14	= += -=	<-----

7.5.1. Các phép toán tăng giảm một đơn vị

- Tăng hoặc giảm một đơn vị cho biến:
 - $\langle \text{tên biến} \rangle = \langle \text{tên biến} \rangle + 1;$
 $\rightarrow \langle \text{tên biến} \rangle ++;$
 - $\langle \text{tên biến} \rangle = \langle \text{tên biến} \rangle - 1;$
 $\rightarrow \langle \text{tên biến} \rangle --;$
 - Ví dụ:
 - `int a = 5;`
 - `float x = 10;`
 - `a ++; // tương đương với a = a + 1;`
 - `x --; // tương đương với x = x - 1;`

8.2.1. Cấu trúc if, if ... else

- Cú pháp cấu trúc **if**
`if (bieu_thuc_dieu_kien)`
`lenh;`
- Cú pháp cấu trúc **if ... else**
`if (bieu_thuc_dieu_kien)`
`lenh_1;`
`else`
`lenh_2;`

Kết hợp lệnh khối

```
if (bieu_thuc_dieu_kien)
{
    lenh_11;
    lenh_12;
}
if (bieu_thuc_dieu_kien)
{
    lenh_21;
    lenh_22;
}
else
{
    lenh_31;
    lenh_32;
}
```


8.2.1. Cấu trúc if, if ... else (2)

- Ví dụ: Bài toán tìm số lớn nhất trong 2 số thực a và b :

```
#include <conio.h>
#include <stdio.h>
void main()
{
    // khai bao bien
    float a, b;
    float max;
    printf(" Nhap gia tri a va b: ");
    scanf("%f %f", &a, &b);
```

8.2.1. Cấu trúc if, if ... else (tiếp)

- Ví dụ (tiếp):

```
if (a < b)
    max = b;
else
    max = a;
printf("\n So lon nhat trong 2 so
        %.0f va %.0f la %.0f ", a, b, max);
getch();
} //ket thuc ham main()
```

- Kết quả:

Nhap vao 2 gia tri a va b: 23 247

So lon nhat trong hai so 23 va 247 la 247

8.2.2. Cấu trúc lựa chọn switch

- Cú pháp cấu trúc **switch**

```
switch (bieu_thuc)
{
    case gia_tri_1: lenh_1; [break];
    case gia_tri_2: lenh_2; [break];
    ...
    case gia_tri_n:    lenh_n; [break];
    [default: lenh_n+1; [break];]
}
```

8.2.2. Cấu trúc lựa chọn switch (2)

- Giá trị của biểu thức kiểm tra (bieu_thuc) phải là số nguyên:
 - Phải có kiểu dữ liệu là **char**, **int**, **long**.
- Tương ứng các giá trị sau **case** (gia_tri_1, gia_tri_2,...) cũng phải là số nguyên.

8.2.2. Cấu trúc lựa chọn switch (3)

- Ví dụ: Nhập vào số nguyên không âm, đưa ra ngày trong tuần tương ứng (theo số dư khi chia cho 7).

```
#include <conio.h>
#include <stdio.h>
void main()
{
    int a;
    printf("\nNhập một giá trị số nguyên không âm: ");
    scanf("%d", &a);
```

8.2.2. Cấu trúc lựa chọn switch (4)

- Ví dụ (tiếp):

```
switch(a % 7)
{
    case 0: printf(" Chu nhat"); break;
    case 1: printf(" Thu Hai"); break;
    case 2: printf(" Thu Ba"); break;
    case 3: printf(" Thu Tu"); break;
    case 4: printf(" Thu Nam"); break;
    case 5: printf(" Thu Sau"); break;
    case 6: printf(" Thu Bay"); break;
}
getch();
}
```

8.2.2. Cấu trúc lựa chọn switch (5)

- Bài tập:
 - Trong một năm các tháng có 30 ngày là 4, 6, 9, 11 còn các tháng có 31 ngày là 1, 3, 5, 7, 8, 10, 12. Riêng tháng hai có thể có 28 hoặc 29 ngày.
 - Hãy viết chương trình nhập vào 1 tháng, sau đó đưa ra kết luận tháng đó có bao nhiêu ngày.

8.2.2. Cấu trúc lựa chọn switch (6)

```
#include <conio.h>
#include <stdio.h>
void main ()
{
    int thang; clrscr();
    printf("\n Nhap vao thang trong nam ");
    scanf("%d",&thang);
    switch(thang)
    {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12:
            printf("\n Thang %d co 31 ngay ",thang);
            break;
```


8.2.2. Cấu trúc lựa chọn switch (7)

```
case 4:
case 6:
case 9:
case 11:
    printf("\n Tháng %d có 30 ngày ",thang);
    break;
case 2:
    printf ("\ Tháng 2 có 28 hoặc 29 ngày");
    break;
default :
    printf("\n Không có tháng %d", thang);
    break;
}
getch();
}
```

8.3.1. Vòng lặp for

- Mục đích
 - Dùng để thực hiện lặp đi lặp lại một công việc nào đó với số lần lặp xác định.

- Cú pháp:

```
for(bieu_thuc_1;bieu_thuc_2;bieu_thuc_3)
{
    day_cac_lenh;
}
```

- Trong đó:
 - **bieu_thuc_1**: Khởi tạo giá trị ban đầu cho vòng lặp
 - **bieu_thuc_2**: Điều kiện tiếp tục vòng lặp
 - **bieu_thuc_3**: Thực hiện bước tăng của vòng lặp
 - Chú ý các biểu thức 1, 2, 3 có thể có hoặc không

8.3.1. Vòng lặp for (2)

- Ví dụ: Đưa ra màn hình các số nguyên lẻ nhỏ hơn 100

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i;
    for(i = 1;i<100;i++)
    {
        if(i%2 == 1) printf("%5d",i);
        if((i+1)%20 ==0) printf("\n");
    }
    getch();
}
```

8.3.2. Vòng lặp while

- Mục đích:
 - Dùng để thực hiện lặp đi lặp lại một công việc nào đó với số lần lặp **không** xác định.
- Cú pháp: Có 2 dạng:

```
while (bieu_thuc)
{
    lenh;
}
```

hoặc

```
do
{
    lenh;
} while (bieu_thuc);
```

8.3.2. Vòng lặp while (2)

- **while** và **do{...} while**:
 - **while**:
 - Kiểm tra điều kiện vòng lặp (tức là giá trị của biểu thức) trước rồi mới thực hiện lệnh.
 - Các **lệnh** sau **while** có thể không được thực hiện lần nào.
 - **do{...} while**:
 - Thực hiện **lệnh** trước rồi mới kiểm tra **dieu_kien** của vòng lặp.
 - Các **lệnh** sau **while** được thực hiện ít nhất 1 lần dù **bieu_thuc** có giá trị như thế nào.

8.4. Các lệnh thay đổi cấu trúc lập trình

- Đối với các lệnh lặp:
 - **while**, **do{ . . . } while**, hoặc **for**
- Thay đổi việc thực hiện lệnh trong vòng lặp →
C cung cấp 2 lệnh:
 - **continue**;
 - **break**;

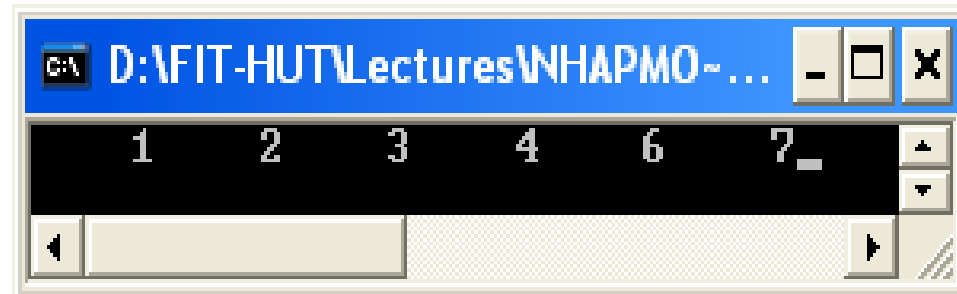
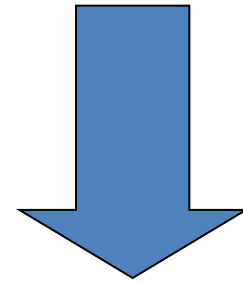
8.4. Các lệnh thay đổi cấu trúc lập trình

- **continue**
 - Bỏ qua việc thực hiện các câu lệnh nằm sau lệnh **continue** trong thân vòng lặp.
 - Chuyển sang thực hiện một vòng lặp mới
- **break**
 - Thoát khỏi vòng lặp ngay cả khi biểu thức điều kiện của vòng lặp vẫn còn được thỏa mãn.

8.4. Các lệnh thay đổi cấu trúc lập trình

- Ví dụ:

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int i;
    for(i = 1;i<=10;i++)
    {
        if(i == 5) continue;
        printf("%5d",i);
        if(i==7) break;
    }
    getch();
}
```



9.1.1. Khái niệm mảng

- Tập hợp hữu hạn các phần tử cùng kiểu, lưu trữ kế tiếp nhau trong bộ nhớ
- Các phần tử trong mảng có cùng tên (là tên mảng) nhưng phân biệt với nhau ở chỉ số cho biết vị trí của nó trong mảng
- Ví dụ:
 - Bảng điểm của sinh viên
 - Vector
 - Ma trận

9.1.2. Khai báo và sử dụng mảng

- Khai báo mảng (một chiều)

kiểu_dữ_liệu tên_mảng [kích_thước_mảng];

- Trong đó

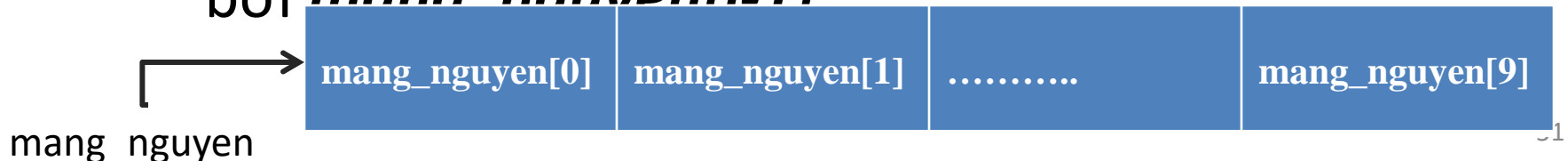
- kiểu_dữ_liệu: kiểu dữ liệu của các phần tử trong mảng
- tên_mảng: tên của mảng
- kích_thước_mảng: số phần tử trong mảng

- Ví dụ

```
int mang_nguyen[10]; // khai báo mảng 10 phần tử có  
kiểu dữ liệu int
```

9.1.2. Khai báo và sử dụng mảng

- Cấp phát bộ nhớ
 - Các phần tử trong mảng được cấp phát các ô nhớ kế tiếp nhau trong bộ nhớ
 - Biến mảng lưu trữ địa chỉ ô nhớ đầu tiên trong vùng nhớ được cấp phát
- Ngôn ngữ C đánh chỉ số các phần tử trong mảng bắt đầu từ 0
 - Phần tử thứ i trong ***mang_nguyen*** được xác định bởi ***mang_nguyen[i-1]***



9.1.2. Khai báo và sử dụng mảng

- Mảng một chiều và mảng nhiều chiều
 - Mỗi phần tử của mảng cũng là một mảng
=> mảng nhiều chiều
- Ví dụ
 - `int a[6][5];`
mảng a gồm 6 phần tử
mỗi phần tử là mảng gồm 5 số nguyên int
 - `int b[3][4][5];` // mảng b gồm 3 phần tử, mỗi phần tử là mảng hai chiều gồm 4 phần tử. Mỗi phần tử mảng hai chiều là mảng gồm 5 số nguyên int. b là mảng 3 chiều

9.1.2. Khai báo và sử dụng mảng

- Khai báo một biến mảng

kiểu_dữ_liệu tên_mảng[size₁][size₂]...[size_k];

Trong đó

- size_i là kích thước chiều thứ i của mảng

9.1.2. Khai báo và sử dụng mảng

- Sử dụng mảng
 - Truy cập vào phần tử thông qua tên mảng và chỉ số của phần tử trong mảng
tên_mảng[chỉ_số_phần_tử]
 - Chú ý: chỉ số bắt đầu từ 0
- Ví dụ
 - `int a[4];`
 - phần tử đầu tiên (thứ nhất) của mảng: `a[0]`
 - phần tử cuối cùng (thứ tư) của mảng: `a[3]`
 - `a[i]`: là phần tử thứ $i+1$ của `a`

9.1.2. Khai báo và sử dụng mảng

- Ví dụ (tiếp)
 - `int b[3][4];`
 - phần tử đầu tiên của mảng: `b[0]` là một mảng một chiều
 - phần tử đầu tiên của mảng `b[0]`: `b[0][0]`
 - `b[i][j]`: là phần tử thứ $j+1$ của `b[i]`, `b[i]` là phần tử thứ $i+1$ của `b`

9.2.1. Khái niệm chuỗi ký tự

- Chuỗi ký tự (string) là một dãy các ký tự viết liên tiếp nhau
 - Độ dài chuỗi là số ký tự có trong chuỗi
 - Chuỗi rỗng là chuỗi không có ký tự nào
- Ví dụ: “Tin học”, “String”
- Lưu trữ: kết thúc chuỗi bằng ký tự ‘\0’ hay NUL (mã ASCII là 0)

'T'	'i'	'n'	' '	'h'	'o'	'c'	'\0'
-----	-----	-----	-----	-----	-----	-----	------

9.2.1. Khái niệm chuỗi ký tự

- So sánh
 - Chuỗi ký tự và mảng ký tự?
 - Tập hợp các ký tự viết liên tiếp nhau
 - Sự khác biệt: chuỗi ký tự có ký tự kết thúc chuỗi, mảng ký tự không có ký tự kết thúc chuỗi
 - Chuỗi ký tự "A" và ký tự 'A'?
 - 'A' là 1 ký tự
 - "A" là 1 chuỗi ký tự, ngoài ký tự 'A' còn có ký tự '\0' => gồm 2 ký tự

9.2.2. Khai báo và sử dụng xâu

a. Khai báo xâu

- Cú pháp

char tên_xâu [số_kí_tự_tối_đạ];

- Lưu ý:

- Để lưu trữ một xâu có n kí tự chúng ta cần một mảng có kích thước n+1

- Ví dụ

- Để lưu trữ xâu "Tin học" chúng ta phải khai báo xâu có số phần tử tối đa ít nhất là 8

char str [8];

9.2.2. Khai báo và sử dụng xâu

b. Truy cập vào một phần tử của xâu

- Cú pháp:

tên_xâu [chỉ_số_của_kí_tự]

- Ví dụ

```
char  quequan[10]="Ha noi";
```

Giả sử xâu này có nội dung là "Ha noi"

⇒ quequan[0]	lưu trữ	'H'
quequan[1]		'a'
quequan[5]		'i'
quequan[6]		'\0'

11.2.1. Khai báo hàm

```
[<kiểu_giá_trị_trả_về>] tên_hàm ([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.
 - Kiểu giá trị trả về: kiểu dữ liệu bất kì, không được là kiểu dữ liệu mảng.
 - Tên hàm: là tên hợp lệ, trong C tên hàm là duy nhất

11.2.1. Khai báo hàm

– 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)`

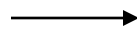
• Thân hàm

– 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;*

11.2.1. Khai báo hàm

Nguyên mẫu hàm
(function prototype)



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

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

```
int binhphuong(int x);
```

```
void main(){
```

```
    int i;
```

```
    for (i=0; i<= 10; i++)
```

```
        printf("%d ",
```

```
        binhphuong(i);
```

```
        getch();
```

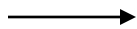
```
    }
```

```
int binhphuong(int x){
```

```
    int y; y = x * x; return y;
```

```
}
```

Định nghĩa hàm



11.3. Phạm vi của biến

- 11.3.1. Phạm vi của biến
- 11.3.2. Phân loại biến
- 11.3.3. Câu lệnh static và register

11.3.1. 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, y;
int binhphuong(int x){
    int y;
    y = x * x;
    return y;
}
void main(){
    int y;
    for (i=0; i<= 10; i++){
        y = binhphuong(i);
        printf("%d ", y);
    }
}
```


11.3.2. 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().

11.3.3. 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 <kiểu_dữ_liệu> tên_biến;`

11.3.3. Câu lệnh static và register

```
# include <stdio.h>
# include <conio.h>
void fct() {
    static int count = 1;
    printf("\n Day la lan goi ham fct lan thu
    %2d", count++);
}
void main() {
    int i;
    for(i = 0; i < 10; i++) fct();
    getch();
}
```

11.3.3. 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
```

Con trỏ trong C

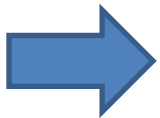
- Để lấy địa chỉ, dùng &

```
#include <stdio.h>

int main ()
{
    int var1;
    char var2[10];

    printf("Address of var1 variable: %x\n", &var1 );
    printf("Address of var2 variable: %x\n", &var2 );

    return 0;
}
```



```
Address of var1 variable: bff5a400
Address of var2 variable: bff5a3f6
```

- Con trỏ là 1 biến mà giá trị của nó là địa chỉ của biến khác.
- Cú pháp khai báo `type *var-name;`
- Ví dụ:

```
int    *ip;    /* pointer to an integer */
double *dp;    /* pointer to a double */
float  *fp;    /* pointer to a float */
char   *ch     /* pointer to a character */
```

```
#include <stdio.h>

int main ()
{
    int var = 20;    /* actual variable declaration */
    int *ip;         /* pointer variable declaration */

    ip = &var; /* store address of var in pointer variable*/

    printf("Address of var variable: %x\n", &var );

    /* address stored in pointer variable */
    printf("Address stored in ip variable: %x\n", ip );

    /* access the value using the pointer */
    printf("Value of *ip variable: %d\n", *ip );

    return 0;
}
```



```
Address of var variable: bffd8b3c
Address stored in ip variable: bffd8b3c
Value of *ip variable: 20
```

```

#include <stdio.h>

const int MAX = 3;

int main ()
{
    int var[] = {10, 100, 200};
    int i, *ptr;

    /* let us have array address in pointer */
    ptr = var;
    for ( i = 0; i < MAX; i++)
    {

        printf("Address of var[%d] = %x\n", i, ptr );
        printf("Value of var[%d] = %d\n", i, *ptr );

        /* move to the next location */
        ptr++;
    }
    return 0;
}

```



```

Address of var[0] = bf882b30
Value of var[0] = 10
Address of var[1] = bf882b34
Value of var[1] = 100
Address of var[2] = bf882b38
Value of var[2] = 200

```



```
#include <stdio.h>

int main ()
{
    int var;
    int *ptr;
    int **pptr;

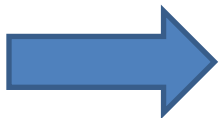
    var = 3000;

    /* take the address of var */
    ptr = &var;

    /* take the address of ptr using address of operator & */
    pptr = &ptr;

    /* take the value using pptr */
    printf("Value of var = %d\n", var );
    printf("Value available at *ptr = %d\n", *ptr );
    printf("Value available at **pptr = %d\n", **pptr);

    return 0;
}
```



```
Value of var = 3000
Value available at *ptr = 3000
Value available at **pptr = 3000
```

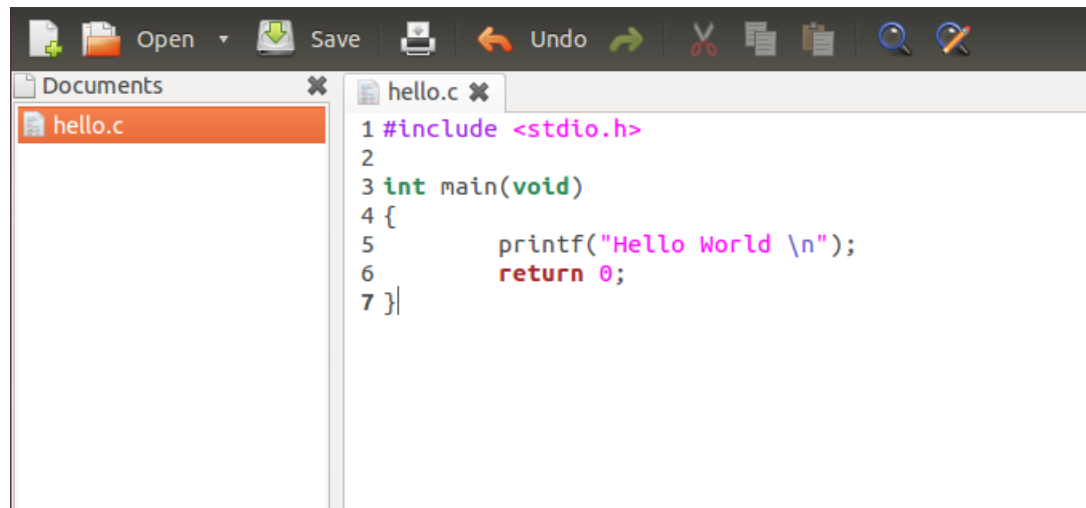
LẬP TRÌNH C TRÊN LINUX

Chuẩn bị

- 3 thứ cần thiết để lập trình C trên Linux
 - Text editor
 - Compiler
 - C standard library

Text editor

- Vi
- nano
- gedit
- V.V...



A screenshot of a text editor window. The window has a dark gray title bar with icons for file operations (Open, Save, Print) and editing (Undo, Cut, Copy, Paste, Find, Replace). Below the title bar is a tab bar with two tabs: 'Documents' and 'hello.c'. The 'hello.c' tab is active. The main editing area shows a C program with the following code:

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Hello World \n");
6     return 0;
7 }
```

Compiling & Running

- Cài đặt gcc

```
sudo apt-get install gcc
```

- Dịch

```
gcc -o hello hello.c
```

- Chạy

```
./hello
```

Make