



# **CHƯƠNG 7**

## **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

### **CHƯƠNG 7**

## **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

7.1 Danh hiệu

7.2 Các kiểu dữ liệu chuẩn của C

7.3 Hằng (*constant*)

7.4 Biến (*variable*)

7.5 Biểu thức

7.6 Các phép toán của C

7.7 Cấu trúc tổng quát của một chương trình C

Bài tập cuối chương



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.1 DANH HIỆU*

Danh hiệu là tên của hằng, biến, hàm... hoặc các ký hiệu đã được quy định đặc trưng cho một thao tác nào đó. Danh hiệu có hai loại: ký hiệu và danh hiệu.



# *CHƯƠNG 7*

## *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.1 DANH HIỆU*

**Ký hiệu** (symbol) là các dấu đã được C quy định để biểu diễn cho một thao tác nào đó.

-Nếu dùng **một dấu** để biểu diễn cho một thao tác thì ta có **ký hiệu đơn** (single symbol).

Ví dụ: +, -, \*, /, %, =, >, <



# *CHƯƠNG 7*

## *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.1 DANH HIỆU*

**Ký hiệu** (symbol) là các dấu đã được C quy định để biểu diễn cho một thao tác nào đó.

-Nếu dùng **hai dấu** trở lên biểu diễn cho một thao tác thì ta có ký hiệu kép (compound symbol).

Ví dụ: ==, >=, <=, /\*, \*/, ++, --, &&, ||, ...



# **CHƯƠNG 7**

## **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

### **7.1 DANH HIỆU**

**Danh hiệu** (Identifier) là các từ khóa của ngôn ngữ hoặc tên của các hằng, biến, hàm trong C. Danh hiệu bao hàm từ khóa và danh hiệu.

**Từ khóa** (keyword) là các danh hiệu mà C đã định nghĩa sẵn cho lập trình viên sử dụng để thiết kế chương trình, tập các từ khóa của C sẽ được liệt kê trong phần phụ lục.

Ví dụ: if, for, while...



# *CHƯƠNG 7*

## *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.1 DANH HIỆU*

**Danh hiệu** (Identifier) là các từ khóa của ngôn ngữ hoặc tên của các hằng, biến, hàm trong C. Danh hiệu bao hàm từ khóa và danh hiệu.

**Danh hiệu** là tên của các hằng, biến, hàm...

-Nếu các hằng, biến, hàm ... này do C đã khai báo và thiết kế sẵn thì các danh hiệu có được gọi là các danh hiệu chuẩn.

Ví dụ: main, scanf, printf...



# **CHƯƠNG 7**

## **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

### **7.1 DANH HIỆU**

**Danh hiệu** (Identifier) là các từ khóa của ngôn ngữ hoặc tên của các hằng, biến, hàm trong C. Danh hiệu bao hàm từ khóa và danh hiệu.

**Danh hiệu** là tên của các hằng, biến, hàm...

-Nếu các hằng, biến, hàm ... này do lập trình viên khai báo và định nghĩa trong quá trình thiết kế chương trình thì các danh hiệu đó được gọi là các **danh hiệu không chuẩn**.

-Ví dụ: a, b, delta



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.1 DANH HIỆU*

**Chú ý** rằng C là một ngôn ngữ nhạy cảm với sự phân biệt giữa ký tự hoa và ký tự thường, do đó khi viết **While** sẽ hoàn toàn phân biệt với **while**. Các từ khóa của C đều ở dạng chữ thường.





# *CHƯƠNG 7*

## *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.1 DANH HIỆU*

Như vậy danh hiệu không chuẩn là tên của các hằng, biến, hàm ... do lập trình viên tự đặt, do đó nguyên tắc đặt tên của danh hiệu không chuẩn cũng cần phải được nêu cụ thể:

- Danh hiệu không chuẩn **không** trùng với từ khóa
- Danh hiệu không chuẩn **không** trùng với danh hiệu chuẩn

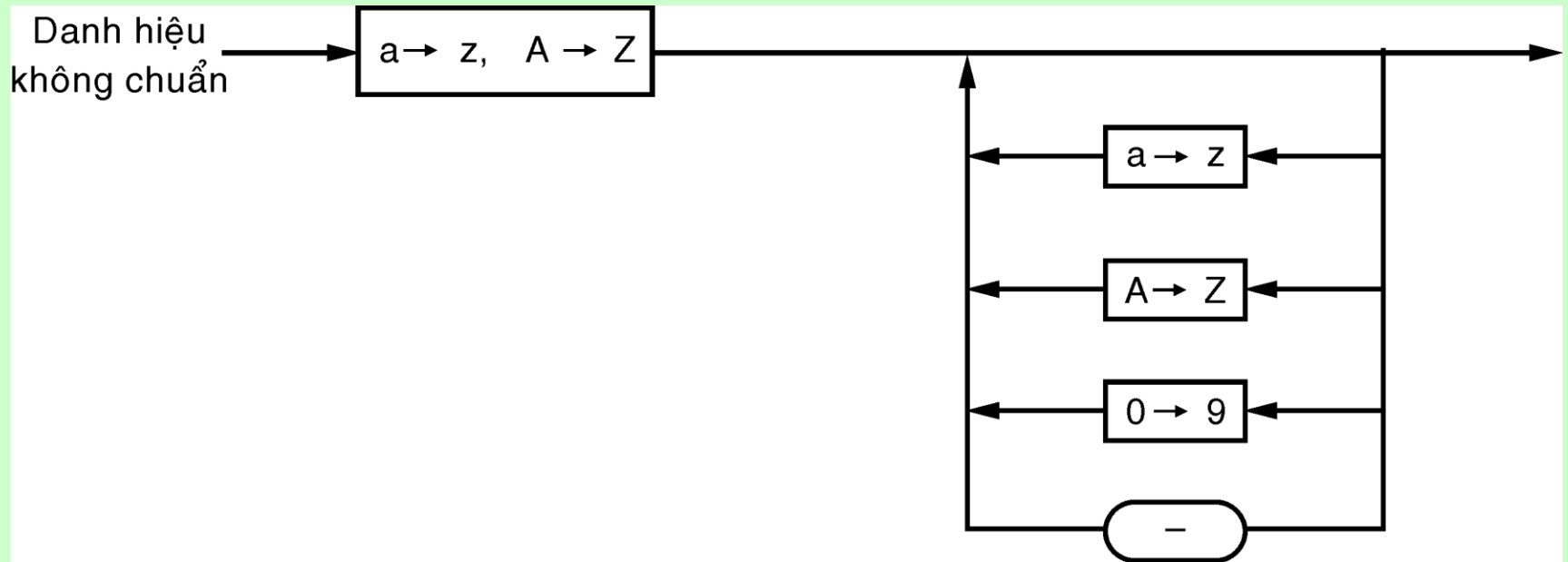


# CHƯƠNG 7

## CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C

### 7.1 DANH HIỆU

- Khi đặt tên cho danh hiệu không chuẩn cần phải theo sơ đồ cú pháp sau:





## ***CHƯƠNG 7***

# ***CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C***

### ***7.1 DANH HIỆU***

*Chú ý:*

- Đối với ô vuông khi đi ngang qua ta cần phải lấy một phần tử trong nó.
- Đối với ô tròn khi đi ngang qua ta phải lấy phần tử trong nó.
- Một danh hiệu có thể được bắt đầu bằng dấu gạch dưới



# *CHƯƠNG 7*

## *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.1 DANH HIỆU*

*Ví dụ:*

*Main ?*

*-batdau ?*

*\_batdau ?*

*2thang9 ?*

*ket thuc ?*



# CHƯƠNG 7

## CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C

### 7.1 DANH HIỆU

Ví dụ:

*Main ?*

*-batdau ?*

*\_batdau ?*

*2thang9 ?*

*ket thuc ?*

*/\* màu đỏ: danh hiệu sai\* /*



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.1 DANH HIỆU*

Chiều dài một danh hiệu không bị hạn chế, mỗi bộ dịch C sẽ có quy định về chiều dài danh hiệu khác nhau, đối với các bộ dịch C/C++ thì danh hiệu có thể dài tùy ý, tuy nhiên trong các bộ dịch Borland C/C++ có quy định một giá trị xác định số ký tự đầu có nghĩa để phân biệt sự giống nhau và khác nhau giữa hai danh hiệu. Trong **Turbo C 2.0**, giá trị này là **31**, trong **Borland C++ 5.02**, giá trị này là **55**.



# *CHƯƠNG 7*

## *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.1 DANH HIỆU*

Ví dụ:

ket\_thuc\_vong\_lap\_in\_ra\_ky\_tu\_khoang\_trang

ket\_thuc\_vong\_lap\_in\_ra\_k



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.2 CÁC KIỂU DỮ LIỆU CỦA C**

C có bốn kiểu dữ liệu chuẩn: char, int, float và double, mỗi kiểu sẽ có yêu cầu về bộ nhớ và tầm trị như sau:

KIỂU	KÍCH THƯỚC	TẦM TRỊ BIỂU DIỄN
char	8 bit	-128 .. + 127
int	16 bit	- 32768 .. + 32767
float	32 bit	- 3.4E37 .. 3.4E+38
double	64 bit	- 1.7E307.. 1.7E+308





# *CHƯƠNG 7*

## *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.2 CÁC KIỂU DỮ LIỆU CỦA C*

#### **1- Kiểu char**

**char** là kiểu nguyên một byte, kiểu này có thể được sử dụng để khai báo biến, biến đó sẽ chiếm kích thước trong bộ nhớ là **1 byte** và có thể giữ một **ký tự** hoặc một **giá trị 8 bit**. Mỗi bộ dịch C sẽ có quy định khác nhau về tầm trị của kiểu **char**, đối với bộ dịch TURBO C VERSION 2.0 kiểu **char** là kiểu có dấu.



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.2 CÁC KIỂU DỮ LIỆU CỦA C**

### **1- Kiểu char**

**Ví dụ:** Biến kiểu char lưu trữ hằng ký tự

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

```
main()
```

```
{
```

```
    char d;
```

```
    d = 'a';
```

```
    printf ("Ký tự trong biến d là %c ", d);
```

```
}
```



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.2 CÁC KIỂU DỮ LIỆU CỦA C**

### **1- Kiểu char**

**Ví dụ:** Biến kiểu char lưu trữ số nguyên

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

```
main()
```

```
{
```

```
    char c;
```

```
    c = 89;
```

```
    printf ("Tri trong bien c la %d ", c);
```

```
}
```



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.2 CÁC KIỂU DỮ LIỆU CỦA C*

### **2- Kiểu int**

Kiểu **int** là một kiểu số nguyên, có thể được sử dụng để khai báo biến, biến đó có kích thước trong bộ nhớ là kích thước của số nguyên mà máy quy định, đối với máy PC và bộ dịch Borland C/C++ thì chiều dài của kiểu **int** là 16 bit có dấu, như vậy một biến hay hằng thuộc kiểu này có tầm trị biểu diễn từ **-32768** đến **32767** (tức từ  $-2^{15}$  đến  $2^{15} - 1$ ).



# **CHƯƠNG 7**

## **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

### **7.2 CÁC KIỂU DỮ LIỆU CỦA C**

#### **2- Kiểu int**

**Ví dụ :**

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

```
main()
```

```
{    int i;  
    i = 1234;  
    i = i + 123;  
    printf ("Trị trong biến i là %d ", i);  
}
```



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.2 CÁC KIỂU DỮ LIỆU CỦA C**

### **3- Kiểu float và double**

**float** là kiểu số thực dấu chấm động, có độ chính xác đơn (7 ký số sau dấu chấm thập phân), **double** là kiểu số thực, dấu chấm động, có độ chính xác kép (15 ký số sau dấu chấm thập phân).

Kiểu **double** còn có thể được khai báo là **long float**, do đó khi khai báo *double b*; thì cũng hoàn toàn tương đương với *long float b*;



# **CHƯƠNG 7**

## **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

### **7.2 CÁC KIỂU DỮ LIỆU CỦA C**

#### **3- Kiểu float và double**

Để xuất nhập cho hằng, biến, biểu thức **float** chuỗi định dạng được sử dụng là **"%f"** đối với kiểu **double** thì chuỗi định dạng là **"%lf"** cho các hàm **printf** và **scanf**.

**Ví dụ:**

```
float a;  
double b;
```



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.2 CÁC KIỂU DỮ LIỆU CỦA C*

### **3- Kiểu float và double**

**Ví dụ:**

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
main()
{
    double x, y, luy_thua;
    clrscr(); printf ("Moi nhap 2 so:");
    scanf ("%lf %lf", &x, &y);
    if (x < 0 && (y - (int)y != 0)) printf ("Ban da nhap sai tri");
    else    { luy_thua = pow (x, y);
              printf ("Luy thua cua %5.2lf voi %5. 2lf la %5.2lf", x, y,
luy_thua); }
}
```





## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.2 CÁC KIỂU DỮ LIỆU CỦA C**

Ngoài ra, ANSI (*American National Standards Institute* – *ANSI*) còn đưa thêm một kiểu dữ liệu nữa là **void**. Đây là kiểu không trị, chỉ dùng để biểu diễn kết quả trả về của hàm và khai báo pointer không trỏ đến một kiểu dữ liệu xác định nào cả. Kiểu này sẽ được nói chi tiết hơn ở các phần sau.



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.2 CÁC KIỂU DỮ LIỆU CỦA C*

Để bổ sung cho bốn kiểu dữ liệu cơ bản C còn đưa ra các dạng bổ sung **signed**, **unsigned**, **short**, **long** :

- **signed** xác định kiểu có dấu.
- **unsigned** xác định kiểu không dấu.
- **short** xác định kiểu ngắn của kiểu cơ bản.
- **long** xác định kiểu dài của kiểu cơ bản.



# **CHƯƠNG 7**

## **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

### **7.2 CÁC KIỂU DỮ LIỆU CỦA C**

<b>Dạng</b> <b>Kiểu</b>	<b>signed</b>	<b>unsigned</b>	<b>short</b>	<b>long</b>
char	signed char → char	unsigned char	x	x
int	signed int → int	unsigned int → unsigned	short int → int/short	long int → long
float	x	x	x	long float → double
double	x	x	x	long double



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.2 CÁC KIỂU DỮ LIỆU CỦA C*

Chú ý rằng trong bộ nhớ máy tính, các giá trị đều được lưu trữ dưới dạng mã nhị phân có nhiều bit (8, 16 hoặc 32 bit tùy theo kiểu của biến hoặc giá trị), trong đó số thứ tự của các bit được đánh số từ phải sang trái bắt đầu từ 0, số hiệu này được gọi là vị trí của bit. Mỗi bit như vậy có một trọng số là **2 mũ n**, với n là vị trí của bit đó.



# CHƯƠNG 7

## CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C

### 7.2 CÁC KIỂU DỮ LIỆU CỦA C

Do đó, một số mà nhị phân là **10011110** sẽ được biểu diễn như sau:

7	6	5	4	3	2	1	0	← Vị trí
1	0	0	1	1	1	1	0	
↑	↑	↑	↑	↑	↑	↑	↑	
$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	← Trọng số (Weight hay Significance)



## *CHƯƠNG 7* *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.2 CÁC KIỂU DỮ LIỆU CỦA C*

Ví dụ: một giá trị kiểu integer (có dấu)

0 0 0 0 0 0 1 0 1 0 0 0 1 0 1 0

$$2^9 + 2^7 + 2^3 + 2^1 = 512 + 128 + 8 + 2 = +650$$



## *CHƯƠNG 7* *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ* *LIỆU CỦA C*

### *7.2 CÁC KIỂU DỮ LIỆU CỦA C*

Ví dụ: một giá trị kiểu integer (có dấu)

1 0 0 0 0 0 1 0 1 0 0 0 1 0 1 0

$$= 2^9 + 2^7 + 2^3 + 2^1 - 2^{15}$$

$$= 512 + 128 + 8 + 2 - 32768 = 650 - 32768$$

$$= -32118$$



## *CHƯƠNG 7* *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.2 CÁC KIỂU DỮ LIỆU CỦA C*

Ví dụ: một giá trị kiểu **unsigned integer** (không có dấu)

1 0 0 0 0 0 1 0 1 0 0 0 1 0 1 0

$$= 2^9 + 2^7 + 2^3 + 2^1 + 2^{15}$$

$$= 512 + 128 + 8 + 2 + 32768 = 650 + 32768$$

$$= 33418$$





## CHƯƠNG 7

# CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C

## 7.2 CÁC KIỂU DỮ LIỆU CỦA C

Ví dụ :

khi cần khai báo một biến n có kiểu là unsigned int ta chỉ cần viết: unsigned int n;

hoặc gọn hơn unsigned n;

hoặc chỉ cần viết: long p; là đủ để khai báo cho biến p có kiểu là *signed long int*.



## ***CHƯƠNG 7*** ***CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C***

### ***7.2 CÁC KIỂU DỮ LIỆU CỦA C***

Ta cần chú ý là khi muốn khai báo đây là một char có hay không có bit dấu thì nên khai báo đầy đủ:

signed char c;

hay

unsigned char c;



## ***CHƯƠNG 7*** ***CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C***

### ***7.2 CÁC KIỂU DỮ LIỆU CỦA C***

Mỗi kiểu dữ liệu chỉ biểu diễn được các giá trị nằm trong một giới hạn nhất định. Giới hạn này phụ thuộc vào số bit mà kiểu dữ liệu đó quy định khi khai báo biến và do đó còn tùy thuộc vào loại máy. Giới hạn này ta gọi là tầm trị của kiểu.



# CHƯƠNG 7

## CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C

### 7.2 CÁC KIỂU DỮ LIỆU CỦA C

KIỂU	KÍCH THƯỚC	TẦM TRỊ BIỂU DIỄN
unsigned char	8 bit	0..255
char	8 bit	-128..+ 127
unsigned short	16 bit	0..65535
short	16 bit	-32768..+ 32767
unsigned	16 bit	0..65535
<u>int</u>	16 bit	-32768..+ 32767
unsigned long	32 bit	0.. 4294967295
long	32 bit	-2147483648..+ 2147483647
float	32 bit	-3.4 E 37..3.4 E + 38
double	64 bit	-1.7 E 307..1.7 E + 308
long double	80 bit	-3.4 E 4932..1.1 E + 4932



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.3 HẲNG (CONSTANT)*

Hằng là những giá trị cố định có trị hoàn toàn xác định và không thể thay đổi được chúng trong quá trình thực thi chương trình. Trong C, mỗi hằng đều có một kiểu dữ liệu riêng mà căn cứ vào kiểu dữ liệu ta có các loại hằng sau:

- Hằng số
- Hằng ký tự
- Chuỗi ký tự
- Biểu thức hằng



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.3 HẲNG (CONSTANT)*

#### **1- Hằng số**

Hằng số là các trị số đã xác định, một hằng số có thể là số nguyên hoặc số thực.



## CHƯƠNG 7

# CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C

### 7.3 HẰNG (CONSTANT)

#### 1- Hằng số

##### *Hằng số nguyên*

Trong ngôn ngữ C, hằng số nguyên có thể thuộc một trong hai kiểu là *integer* hoặc *long integer*. Ứng với mỗi kiểu, hằng số có thể được biểu diễn ở dạng thập phân, bát phân(05) hay thập lục phân(0x6).



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.3 HẰNG (CONSTANT)*

### **1- Hằng số**

#### *Hằng số nguyên*

+ Hằng số nguyên được viết một cách bình thường và thường chiếm một từ (word) trong bộ nhớ, do đó nó có giá trị đi từ -32768 đến 32767, có nghĩa là **MSB** của dạng lưu trữ nhị phân của một số nguyên luôn là **bit dấu**.

Các hằng số nguyên 10    -4    -23456





## CHƯƠNG 7

# CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C

### 7.3 HẰNG (CONSTANT)

#### 1- Hằng số

##### *Hằng số nguyên*

Ta cần lưu ý khi sử dụng hằng số nguyên vượt quá tầm quy định.

**Ví dụ:** Xét chương trình sau:

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

```
main()
```

```
    {                               printf ("%d %d %d", 32767, 32767 + 1,  
32767 + 2);  
    }
```

32767

-32768

-32767



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

### **7.3 HẲNG (CONSTANT)**

#### **1- Hằng số**

##### ***Hằng số nguyên***

+ Hằng số nguyên dạng **long integer** lại được lưu trữ trong bộ nhớ với chiều dài **32 bit**, có nghĩa là nó có thể có trị nằm trong khoảng -2147483648 đến +2147483647, và khi viết các hằng số nguyên dạng này ta cần phải thêm l hay L ngay sau số cần làm việc.



## **CHƯƠNG 7**

# ***CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C***

### ***7.3 HẰNG (CONSTANT)***

#### **1- Hằng số**

#### ***Hằng số nguyên***

#### **Ví dụ:**

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

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

```
main()
```

```
{
```

```
    clrscr();
```

```
    printf ("%ld %ld %ld", 32767L, 32768L,
```

```
    32769L);
```

```
    getch();
```

```
}
```



## CHƯƠNG 7

# CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C

### 7.3 HẰNG (CONSTANT)

#### 1- Hằng số

##### *Hằng số nguyên*

Hằng số nguyên có thể ở dạng unsigned, khi đó ta sẽ thêm u hoặc U vào ngay sau số đang làm việc (số đó có thể đang ở kiểu integer hoặc long integer).

**Ví dụ:** Các hằng số sau đây ở dạng *unsigned*

123U

234u

24UL



# CHƯƠNG 7

## CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C

### 7.3 HẲNG (CONSTANT)

#### 1- Hằng số

*Hằng số nguyên*

Hằng nguyên	Dạng biểu diễn	Tương đương thập phân
12347	decimal	12347
-153	decimal	-153
034	octal	28 ( $3 \cdot 8^1 + 4 \cdot 8^0$ )
0xa5	hex	165 ( $10 \cdot 16^1 + 5 \cdot 16^0$ )
0Xa2	hex	162 ( $10 \cdot 16^1 + 2 \cdot 16^0$ )
123L	decimal (dạng long)	123
034L	octal (dạng long)	28
0xa2L	hex (dạng long)	162



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.3 HẲNG (CONSTANT)*

#### **1- Hằng số**

##### *Hằng thực*

Trong ngôn ngữ C, số thực có thể ở dạng dấu chấm tĩnh hoặc dấu chấm động.

**Ví dụ:** các số thực sau ở dạng dấu chấm tĩnh

1.4

-2.34

-10.0234



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.3 HẲNG (CONSTANT)*

### **1- Hằng số**

#### *Hằng thực*

Một hằng thực ở dạng số dấu chấm động có thể có các thành phần sau:

- **phần nguyên**: phần này là tùy yêu cầu.
- **dấu chấm thập phân**: bắt buộc phải có.
- **phần lẻ**: tùy yêu cầu.
- các ký tự "**e**" hoặc "**E**" và một số mũ. Số mũ bản thân nó có thể âm.



## CHƯƠNG 7

# CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C

### 7.3 HẲNG (CONSTANT)

#### 1- Hằng số

##### *Hằng thực*

Hằng thực dấu chấm động	Hằng thực tương đương
2.1415e4	21415.0
0.2344e−4	0.00002344
.2344e3	234.4





## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

### **7.3 HẲNG (CONSTANT)**

#### **1- Hằng số**

*Cần lưu ý:*

- Các hằng số được viết không có dấu thập phân và số mũ, sẽ được hiểu là nguyên và được lưu trữ theo kiểu int, ngược lại sẽ được lưu trữ theo kiểu double.
- Các hằng số nguyên lớn hơn khả năng một int được tự động lưu trữ theo kiểu long.
- Các hằng số nguyên lớn hơn một long được lưu trữ theo kiểu double.



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

### **7.3 HẲNG (CONSTANT)**

#### **2- Hằng ký tự**

Hằng ký tự biểu diễn một giá trị ký tự đơn, ký tự này phải được viết giữa cặp dấu nháy đơn (' '), mỗi ký tự có một mã số tương ứng trong bảng mã ký tự của máy, bình thường là mã ASCII.

**Ví dụ:**

'a' '/' '9'

'A' có mã là 65 trong bảng mã ASCII.

'0' có mã là 48 (0x30) trong bảng mã ASCII.



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.3 HẰNG (CONSTANT)*

#### **2- Hằng ký tự**

**Ví dụ 1.26:**

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

```
main()
```

```
{
```

```
    printf ("Ký tự: %c %c %c \n", 'A', '$', '1' );
```

```
    printf ("Mã ASCII (Octal): %o %o %o \n", 'A', '$', '1'
```

```
);
```

```
    printf ("Mã ASCII (Decimal): %d %d %d \n", 'A', '$',
```

```
'1' );
```

```
}
```



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.3 HẰNG (CONSTANT)*

#### **2- Hằng ký tự**

**Ví dụ 1.26:**

Chương trình in ra màn hình kết quả sau:

Ký tự: A \$ 1

Mã ASCII (Octal): 101      44      61

Mã ASCII (Decimal):      65      36      49



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.3 HẰNG (CONSTANT)*

#### **2- Hằng ký tự**

##### **Ví dụ:**

'\n' là ký tự xuống dòng (line feed)

'\45' là ký tự ASCII có mã octal là 45 hay 37 decimal

'\0' là ký tự NUL



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.3 HẲNG (CONSTANT)*

#### **3- Chuỗi ký tự**

Trong ngôn ngữ C, **một chuỗi ký tự** là một loạt các ký tự nằm trong cặp dấu **nháy kép** (“ ”); các ký tự này có thể là ký tự được biểu diễn bằng chuỗi thoát.

**Ví dụ:**

"Một chuỗi ký tự"

"Chuỗi ký tự có chuỗi thoát: i can\'t go to school \n\a"



# *CHƯƠNG 7*

## *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.3 HẲNG (CONSTANT)*

#### **3- Chuỗi ký tự**

Ví dụ: “string”

s	t	r	i	n	g	\0
---	---	---	---	---	---	----



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

### **7.3 HẲNG (CONSTANT)**

#### **4- Biểu thức hằng**

Một biểu thức được xem là một biểu thức hằng nếu giá trị của biểu thức hoàn toàn xác định, như vậy một biểu thức toán học là một biểu thức hằng khi trong biểu thức đó các toán hạng đều là những hằng số hoặc hằng ký tự.

**Ví dụ:** Xét các biểu thức hằng sau

$10 - 13 \% 3$  sẽ được tính trước và được ghi là 9

'a' - 'A' sẽ được tính trước và được ghi là 32

$1 < 8$  sẽ được tính trước và được ghi là 1 (true)





## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.4 BIẾN (VARIABLE)*

### **7.4.1 Khai báo biến**

Tất cả các biến được sử dụng trong một chương trình C đều phải được khai báo trước.

Khi muốn sử dụng biến ta chỉ cần gọi tên biến, dĩ nhiên tên biến phải là một danh hiệu không chuẩn hợp lệ



## CHƯƠNG 7

# CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C

## 7.4 BIẾN (VARIABLE)

### 7.4.1 Khai báo biến

Ví dụ:

bat_dau	}	<i>đều là các danh hiệu hợp lệ</i>
lf		
_hoten		
Thu_thang_1_68		

printf	}	<i>đều là các danh hiệu không hợp lệ</i>
pow		
31_thang_12		
ket thuc		



## **CHƯƠNG 7**

# ***CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C***

## ***7.4 BIẾN (VARIABLE)***

### **7.4.1 Khai báo biến**

Ví dụ:

BIEN  
BiEN  
Bien  
biEN  
bien

*đều là những biến khác nhau*



# CHƯƠNG 7

## CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C

### 7.4 BIẾN (VARIABLE)

#### 7.4.1 Khai báo biến

Cú pháp khai báo biến

**kiểu** *dsach\_tenbien*;

Giải thích:

*kiểu* kiểu của các biến cần khai báo

*dsach\_tenbien* danh sách liệt kê các tên biến cần khai báo, các biến cách nhau bằng dấu ",",

Ví dụ:

**int** lap, count, max;

**double** he\_so\_1, he\_so\_2, delta;



# *CHƯƠNG 7*

## *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.4 BIẾN (VARIABLE)*

#### **7.4.1 Khai báo biến**

**Ví dụ:**

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

```
main()
```

```
{    int thu;
```

```
    thu = 65;
```

```
    printf ("Trị trong biến thu là %d \n", thu);
```

```
    thu = thu + 10;
```

```
    printf ("Sau khi gán, trị trong biến thu là %d
```

```
    \n", thu);
```

```
}
```



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.4 BIẾN (VARIABLE)**

### **7.4.1 Khai báo biến**

Biến của một chương trình C có thể được khai báo ở một trong ba vị trí sau:

- Ngoài tất cả các hàm khi đó ta có biến toàn cục.
- Đầu phần thân của một hàm hoặc một khối lệnh khi đó ta có biến cục bộ.
- Trong phần định nghĩa đối số của hàm (gọi là đối số hàm hoặc tham số hàm).



# CHƯƠNG 7

## CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C

### 7.4 BIẾN (VARIABLE)

#### 7.4.1 Khai báo biến

Ví dụ:

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

```
int a, b;
```

← *khai báo biến ngoài*

```
main()
```

```
{ ...
```

```
    a = 10;
```

```
    b = a + 24;
```

```
    ... }
```



# CHƯƠNG 7

## CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C

### 7.4 BIẾN (VARIABLE)

#### 7.4.1 Khai báo biến

Ví dụ:

```
int tong()  
{  
    int i, tam;    ← khai báo biến trong  
    ...  
    for (i = 10; ...  
    ...  
}
```





## CHƯƠNG 7

# CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C

## 7.4 BIẾN (VARIABLE)

### 7.4.1 Khai báo biến

Ví dụ:

```
int luy_thua (int n, char ket_qua) ← đối số hàm
{
    ...
    for (i =1; ...
        ...
    }
}
```



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.4 BIẾN (VARIABLE)*

### **7.4.1 Khai báo biến**

Một biến có thể được khởi động trị ngay sau khi khai báo bằng phép gán với giá trị tương ứng, khi đó cú pháp khai báo biến như sau:

**kiểu biến1 = tri1, biến2 = tri2;**

với các tri1 và tri2 phải là những giá trị đã xác định. **Ví dụ:**

```
double he_so_1 = 20.37;
```

```
char ky_tu = 'A', ky_tu_moi = ky_tu;
```



## CHƯƠNG 7

# CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C

### 7.4 BIẾN (VARIABLE)

#### 7.4.2 Các bổ túc kiểu const và volatile

**Từ khóa const:** khi được khai báo cho biến thì nó xác định rằng biến sẽ **không bị thay đổi trị** trong suốt quá trình thực thi chương trình, **mọi sự thay đổi trị** đều gây ra **lỗi**, biến đó ta gọi là biến hằng.

Cú pháp:

**const** kiểu tên\_biến [ = trị được thay thế];

**Ví dụ:**           const double bat\_dau = 3.1415;  
                  const int max = 100;



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

### **7.4 BIẾN (VARIABLE)**

#### **7.4.2 Các bổ túc kiểu const và volatile**

Nếu **kiểu** của biến hằng không nêu cụ thể thì biến hằng đó sẽ thuộc loại **int**, ngay cả nếu trị được thay thế là một trị khác int thì chỉ phần nguyên được sử dụng và lưu vào biến hằng mà thôi.

**Ví dụ :**

```
const max = 100;
```

```
const pi = 3.14; khi đó biến pi chỉ là 3 mà thôi
```



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.4 BIẾN (VARIABLE)**

### **7.4.2 Các bổ túc kiểu const và volatile**

**Từ khóa volatile:** chỉ ra rằng một biến có thể bị thay đổi trị từ một **tác nhân không nằm trong chương trình**. Từ khóa này làm cho biến của C có một tính linh động rất cao, ví dụ như biến của C có thể thay đổi theo đồng hồ hệ thống hay theo một chương trình nền nào đó.

Cú pháp:

**volatile <tên\_biến>;**



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

### **7.5 BIỂU THỨC**

Biểu thức là một sự kết hợp của các toán hạng là các biến, hằng hoặc phép gọi hàm bằng các toán tử xác định của C để tạo ra được một trị, trị này có thể được sử dụng hoặc không được sử dụng tùy nhu cầu của lập trình viên.

**Ví dụ:** Đối với C, một biểu thức như sau là hợp lệ:

$a = (x = 10) - (y = a + 1) * ((b += 1) > 12);$



## ***CHƯƠNG 7***

# ***CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C***

## ***7.6 CÁC PHÉP TOÁN CỦA C***

### **7.6.1 Toán tử số học**

C có các toán tử số học bình thường giữa 2 toán hạng, đó là

- Toán tử cộng (+) : thực hiện phép toán cộng
- Toán tử trừ (−) : thực hiện phép toán trừ
- Toán tử nhân (\*) : thực hiện phép nhân
- Toán tử chia (/) : thực hiện phép chia
- Toán tử modulo (%): thực hiện phép toán lấy số dư của phép chia nguyên



## *CHƯƠNG 7* *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.6 CÁC PHÉP TOÁN CỦA C*

#### **7.6.1 Toán tử số học**

Các phép toán này đều thực hiện được trên tất cả các **toán hạng** là hằng, biến hoặc biểu thức có kiểu dữ liệu char, int, long, float, double, trừ toán tử **modulo** chỉ thực hiện phép toán trên **các dữ liệu thuộc các kiểu nguyên** (char, int, long, unsigned).

Thứ tự kết hợp theo nguyên tắc toán học: **nhân, chia, modulo trước; cộng, trừ sau**, nếu các toán hạng đều ngang cấp thì kết hợp từ trái sang phải.





## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.6 CÁC PHÉP TOÁN CỦA C**

### **7.6.1 Toán tử số học**

**Ví dụ:**

float a, b, c;

double y; y = a \* b - c;

Phép toán cộng, trừ cũng có thể được dùng như phép toán đơn hạng (lấy dương và âm của một toán hạng).

**Ví dụ:**

b = - b; a = + b;



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.6 CÁC PHÉP TOÁN CỦA C*

### **7.6.1 Toán tử số học**

Phép nhân và chia các số nguyên sẽ chỉ cho **kết quả nguyên** (C tự động cắt bỏ phần thập phân nếu có).

**Ví dụ:**

```
int a = 10, b = 3, c;  
c = a/b; /*    c =3    */  
c = a % b; /*  c =1    */
```

```
double x = 10., y = 3, z;  
z = x/y; /*    z =3.3333 */
```



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.6 CÁC PHÉP TOÁN CỦA C*

### **7.6.1 Toán tử số học**

Một vấn đề đặt ra là nếu có nhiều toán hạng khác kiểu nhau thì C sẽ thực hiện việc tính toán biểu thức ra sao?

C sẽ thực hiện việc chuyển kiểu tự động theo quy luật sau: toán hạng thuộc kiểu có trị nhỏ hơn sẽ được chuyển sang kiểu có trị lớn hơn. **Ví dụ:**

```
double a = 10, c; int b = 2;  
c=a/b;
```

Trong quá trình tính b kiểu double, sau khi tính b thuộc kiểu int



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.6 CÁC PHÉP TOÁN CỦA C**

### **7.6.2 Toán tử quan hệ**

- Toán tử bằng (==)
- Toán tử khác (!=)
- Toán tử lớn hơn (>)
- Toán tử nhỏ hơn (<)
- Toán tử lớn hơn hoặc bằng (>=)
- Toán tử nhỏ hơn hoặc bằng (<=)



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.6 CÁC PHÉP TOÁN CỦA C*

### **7.6.2 Toán tử quan hệ**

Khi mỗi quan hệ giữa hai toán hạng theo toán tử quan hệ trong biểu thức là **ĐÚNG** thì biểu thức đó sẽ trả về một trị nguyên là **1**, còn ngược lại mỗi quan hệ đó là **SAI** thì biểu thức đó sẽ trả về một trị nguyên là **0**, đây là các trị nguyên bình thường mà ta có thể dùng nó để tính toán.



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.6 CÁC PHÉP TOÁN CỦA C*

### **7.6.2 Toán tử quan hệ**

**Ví dụ :**

...

if (delta > 0)

{

...

}

else

...



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.6 CÁC PHÉP TOÁN CỦA C*

### **7.6.2 Toán tử quan hệ**

**Ví dụ:**

```
int a, b, c;
```

```
char kt;
```

```
a = 1;
```

```
b = 2;
```

```
c = -3;
```

```
kt = (a >= 4) * 2 + (b < 3) - c;
```



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.6 CÁC PHÉP TOÁN CỦA C**

### **7.6.2 Toán tử quan hệ**

**Ví dụ:**

1/  $a = (b == 4);$   
toán tử quan hệ

2/  $a = (b = 4);$   
toán tử gán





## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.6 CÁC PHÉP TOÁN CỦA C*

### **7.6.3 Toán tử logic**

C có các toán tử logic not (!), and (&&), or (||). Các toán tử này cho phép lập trình viên liên kết các biểu thức quan hệ đơn lại với nhau để tạo các biểu thức phức tạp hơn.

Các toán tử này có độ ưu tiên là not, and, or; nếu trong biểu thức các toán tử đều ngang cấp nhau thì thứ tự tính toán từ trái sang phải.



## CHƯƠNG 7

# CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C

## 7.6 CÁC PHÉP TOÁN CỦA C

### 7.6.3 Toán tử logic

Toán hạng 1	Toán hạng 2	Kết quả		
A	B	! A	A && B	A    B
bằng 0	bằng 0	1	0	0
bằng 0	khác 0	1	0	1
khác 0	bằng 0	0	0	1
khác 0	khác 0	0	1	1



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.6 CÁC PHÉP TOÁN CỦA C**

### **7.6.3 Toán tử logic**

**Ví dụ:**

```
int a, b;
```

```
a = 4;
```

```
b = 5;
```

Biểu thức:  $(a > 5) \ \&\& \ (b < 3)$  có trị là false (0)



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.6 CÁC PHÉP TOÁN CỦA C**

### **7.6.3 Toán tử logic**

**Ví dụ:**

```
int a, b;
```

```
a = 4;
```

```
b = 5;
```

thì biểu thức  $(a > 5) \ \&\& \ (b > 3)$  có trị là true (1)



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.6 CÁC PHÉP TOÁN CỦA C**

### **7.6.3 Toán tử logic**

**Ví dụ:** trong mệnh đề if sau đây thay vì viết

```
if (delta == 0)      { ... }
```

ta có thể viết gọn hơn như sau:

```
if (!delta)         { ... }
```



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.6 CÁC PHÉP TOÁN CỦA C*

### **7.6.3 Toán tử logic**

Điều cần lưu ý là các phép toán and (&&) và or ( ) được thực hiện **từ trái sang phải** và việc tính toán sẽ **dừng lại** ngay khi giá trị của cả biểu thức logic đã **xác định trị** mà không cần tính tiếp các phần còn lại của biểu thức nữa.

**Ví dụ:**

```
(c >= 'A') && (c <= 'Z') || (c >= 'a') && (c <= 'z')
```



# *CHƯƠNG 7*

## *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.6 CÁC PHÉP TOÁN CỦA C*

#### **7.6.3 Toán tử logic**

**Ví dụ:**

```
scanf ("%d", &x);
```

```
if ( (x >= 10) && (x <= (y = 100)) )
```

```
{
```

```
...
```

```
}
```



## ***CHƯƠNG 7*** ***CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C***

### ***7.6 CÁC PHÉP TOÁN CỦA C***

#### **7.6.4 Toán tử trên bit**

C có sáu toán tử đặc biệt cho phép lập trình viên xử lý **dữ liệu kiểu nguyên** (như char, int, long, unsigned hoặc signed) ở **cấp độ bit**, các toán tử này được gọi là toán tử trên bit.





## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.6 CÁC PHÉP TOÁN CỦA C*

### **7.6.4 Toán tử trên bit**

- NOT:  $\sim$ , thực hiện việc **đảo bit**, từ bit 0 qua 1 và ngược lại
- AND:  $\&$ , thực hiện việc **and bit**
- OR:  $|$ , thực hiện việc **or bit**
- XOR:  $\wedge$ , thực hiện việc **xor bit**
- Dịch trái:  $<<$ , dịch các bit **sang trái**
- Dịch phải:  $>>$ , dịch các bit **sang phải**



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.6 CÁC PHÉP TOÁN CỦA C**

### **7.6.4 Toán tử trên bit**

Bit	Bit	Phép toán			
A	B	$\sim A$	$A \& B$	$A   B$	$A \wedge B$
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0



## **CHƯƠNG 7**

### **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.6 CÁC PHÉP TOÁN CỦA C**

### **7.6.4 Toán tử trên bit**

**Ví dụ :**

$a = 1030$  có mã nhị phân là 0000 0100 0000 0110

$b = 224$  có mã nhị phân là 0000 0000 1110 0000

$\sim a$	1111	1011	1111	1001
$a \& b$	0000	0000	0000	0000
$a   b$	0000	0100	1110	0110
$a \wedge b$	0000	0100	1110	0110



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.6 CÁC PHÉP TOÁN CỦA C*

### **7.6.4 Toán tử trên bit**

Phép toán AND (&) sẽ kết hợp với một trị mà ta gọi là **mặt nạ** che các bit dữ liệu không cần quan tâm, như vậy trong giá trị mặt nạ tại những vị trí cần che, bit sẽ có trị là 0, các bit còn lại là 1, và qua đó ta có thể đánh giá được dữ liệu đang làm việc một cách chính xác.



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.6 CÁC PHÉP TOÁN CỦA C**

### **7.6.4 Toán tử trên bit**

**Ví dụ:** Nhập vào một số nguyên, xét xem bit có vị trí là 9 có bằng 1 hay không

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

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

```
#define MASK 0x0200
```

```
main()
```

```
{ int a;
```

```
    clrscr();
```



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.6 CÁC PHÉP TOÁN CỦA C*

### **7.6.4 Toán tử trên bit**

```
printf ("Moi nhap mot so: ");  
scanf ("%d", &a);  
if (a & MASK)  
    printf ("Bit 9 bang 1 \n");  
else  
    printf ("Bit 9 bang 0 \n");  
getch();  
}
```



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.6 CÁC PHÉP TOÁN CỦA C**

### **7.6.4 Toán tử trên bit**

Phép toán OR (`|`) được dùng kết hợp với một mặt nạ để bật các bit cần thiết trong một giá trị lên một 1 và giữ nguyên các bit khác. Để tạo ra một trị mặt nạ, các bit tại các vị trí cần thiết sẽ được bật lên 1, còn lại giữ 0. Kết quả của phép OR thu được từ một giá trị và một mặt nạ là một giá trị mà trong đó các bit quan tâm sẽ được gán bằng 1.



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.6 CÁC PHÉP TOÁN CỦA C**

### **7.6.4 Toán tử trên bit**

**Ví dụ:** Nhập vào một số nguyên, xét xem bit có vị trí là 9 có bằng 1 hay không, nếu nó bằng 0 thì bật bit đó lên 1 và in ra kết quả.

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

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

```
#define MASK 0x0200
```

```
main()
```

```
{ int a;
```

```
    clrscr();
```

```
    printf ("Moi nhap mot so: ");
```





## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.6 CÁC PHÉP TOÁN CỦA C*

### **7.6.4 Toán tử trên bit**

```
scanf ("%d", &a);  
if (a & MASK)  
    {  
        printf ("Bit 9 bang 1 \n");  
        printf ("Tri cua bien a la %d \n", a);  
    }  
else  
    {  
        a =a | MASK;  
        printf ("Bit 9 bang 0 va da duoc bat len  
1\n");  
        printf ("Tri cua bien a la %d \n", a);  
    }  
getch();  
}
```



## ***CHƯƠNG 7***

# ***CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C***

## ***7.6 CÁC PHÉP TOÁN CỦA C***

### **7.6.4 Toán tử trên bit**

Phép toán XOR ( $\wedge$ ) kết hợp với một mặt nạ dùng để đảo các bit cần thiết trong một giá trị từ 0 lên 1 và từ 1 về 0, giữ nguyên các bit khác. Mặt nạ này có các bit tại các vị trí cần thiết sẽ được bật lên 1, các bit còn lại giữ nguyên.



## ***CHƯƠNG 7*** ***CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C***

### ***7.6 CÁC PHÉP TOÁN CỦA C***

#### **7.6.4 Toán tử trên bit**

**Ví dụ:** Nhập vào một số nguyên, xét xem bit có vị trí là 9 có bằng 1 hay không, nếu nó bằng 0 thì bật đó lên 1, nếu bằng 1 thì đưa về 0 và in ra kết quả.

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

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

```
#define MASK 0x0200
```

```
main()
```

```
{    int a;  
    clrscr();
```



## ***CHƯƠNG 7***

# ***CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C***

## ***7.6 CÁC PHÉP TOÁN CỦA C***

### **7.6.4 Toán tử trên bit**

```
printf ("Moi nhap mot so: ");
scanf ("%d", &a);
if (a & MASK) printf ("Bit 9 bang 1 \n");
else          printf ("Bit 9 bang 0 \n");
printf ("Tri cua bien a truoc khi dao bit 9 la %d \n", a);
a = a ^ MASK;
if (a & MASK) printf ("Bit 9 sau khi dao bang 1 \n");
else          printf ("Bit 9 sau khi dao bang 0 \n");
printf ("Tri cua bien a sau khi dao bit 9 la %d \n", a);
getch();
}
```



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.6 CÁC PHÉP TOÁN CỦA C**

### **7.6.4 Toán tử trên bit**

Toán tử NOT ( $\sim$ ) cho phép thực hiện việc đảo tất cả các bit của một giá trị từ 0 lên 1 và từ 1 xuống 0. Phép toán như vậy ta gọi là phép bù 1 giá trị hiện hành.



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.6 CÁC PHÉP TOÁN CỦA C*

### **7.6.4 Toán tử trên bit**

Toán tử dịch trái (<<) và dịch phải (>>) cho phép thực hiện việc dời các bit của toán hạng sang bên trái hoặc sang phải. Cú pháp như sau:

**biểu\_thức\_nguyên << số\_bit\_dời**

**biểu\_thức\_nguyên >> số\_bit\_dời**

Với biểu\_thức\_nguyên và số\_bit\_dời có thể là hằng, biến hoặc biểu thức kiểu nguyên.



## ***CHƯƠNG 7***

# ***CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C***

## ***7.6 CÁC PHÉP TOÁN CỦA C***

### **7.6.4 Toán tử trên bit**

Trong phép dịch trái, các bit ở bên phải của toán hạng sẽ được ghi vào các giá trị là 0



# CHƯƠNG 7

## CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C

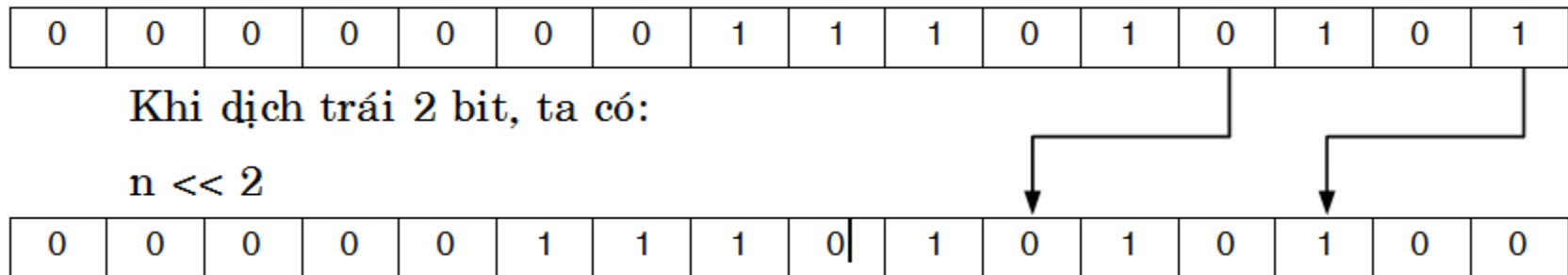
### 7.6 CÁC PHÉP TOÁN CỦA C

#### 7.6.4 Toán tử trên bit

Ví dụ: Xét biến `n` được khai báo

```
int n;
```

đang có trị `n = 469` (tức `0x01d5`), biểu diễn trong bộ nhớ dưới dạng nhị phân là







## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.6 CÁC PHÉP TOÁN CỦA C**

### **7.6.4 Toán tử trên bit**

Trong phép dịch phải ta có hai trường hợp:

- Nếu toán hạng bên trái có dữ liệu thuộc dạng **unsigned** (unsigned int, unsigned long, unsigned char) thì phép dịch phải sẽ ghi **0** vào các bit bên trái của kết quả.
- Còn nếu toán hạng bên trái có dữ liệu thuộc dạng **signed** (int, long, char) thì phép dịch phải sẽ ghi **bit dấu** vào các bit bên trái của kết quả.



# CHƯƠNG 7

## CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C

### 7.6 CÁC PHÉP TOÁN CỦA C

#### 7.6.4 Toán tử trên bit

Ví dụ:

unsigned n;

Giả sử n đang lưu giá trị  $n = 42569$  (tức  $0xA649$ ), tức trong bộ nhớ ta có n

1	0	1	0	0	1	1	0	0	1	0	0	1	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Khi phép dịch phải 2 bit xảy ra ( $n \gg 2$ ), kết quả như sau:

0	0	1	0	1	0	0	1	1	0	0	1	0	0	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

0 →



# CHƯƠNG 7

## CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C

### 7.6 CÁC PHÉP TOÁN CỦA C

#### 7.6.4 Toán tử trên bit

Ví dụ:

```
int n;
```

Giả sử n đang lưu giá trị  $n = -32766$  (tức  $0x8002$ ), tức trong bộ nhớ ta có n

1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

bit dấu

Khi phép dịch phải 2 bit xảy ra ( $n \gg 2$ ), kết quả như sau:

1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

1 →



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.6 CÁC PHÉP TOÁN CỦA C**

### **7.6.5 Toán tử tăng giảm**

Trong C có hai toán tử đặc biệt gọi là toán tử tăng (++) và toán tử giảm (--) dùng để tăng hoặc giảm một biến nào đó đi 1. Cú pháp sử dụng:

**++ biến**

**biến ++**

**-- biến**

**biến --**

với biến có thể thuộc loại bất kỳ hoặc pointer.



## *CHƯƠNG 7* *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.6 CÁC PHÉP TOÁN CỦA C*

#### **7.6.5 Toán tử tăng giảm**

Ký hiệu ++ và -- có thể được đặt trước (gọi là **tiền tố**) hoặc sau (gọi là **hậu tố**) một tên biến.

**Ví dụ:**

++ a;

hoặc

a ++;

chính là

a = a + 1;



## ***CHƯƠNG 7*** ***CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C***

### ***7.6 CÁC PHÉP TOÁN CỦA C***

#### **7.6.5 Toán tử tăng giảm**

**Ví dụ:**

```
int i = 12;  
double a, b;  
a = ++i;  
b = i ++;
```



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.6 CÁC PHÉP TOÁN CỦA C**

### **7.6.5 Toán tử tăng giảm**

Cần lưu ý rằng cặp dấu ++ hoặc phải được viết liền nhau và cần phải rõ ràng, trong những trường hợp có thể nhầm lẫn, ví dụ như 2 biểu thức:

**a++ + b**

và

**a + ++b**

là khác nhau.



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.6 CÁC PHÉP TOÁN CỦA C*

### **7.6.5 Toán tử tăng giảm**

Các phép toán tăng giảm này **nhANH hơn nhiều** so với thao tác bình thường là cộng hoặc trừ biến đó với 1 rồi gán trở lại cho biến đó vì chúng rất gần với các phép toán tăng giảm của ngôn ngữ máy





## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.6 CÁC PHÉP TOÁN CỦA C**

### **7.6.5 Toán tử tăng giảm**

**Ví dụ:** Trong thân vòng while của chương trình tính tổng từ 1 tới n, ta có thể viết như sau

```
s = 0;  
so = 1;  
while (so <= n)  
{  
    s = s + so;  
    so++;  
}
```



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.6 CÁC PHÉP TOÁN CỦA C*

### **7.6.6 Toán tử gán**

Phép toán gán là phép toán cơ bản trong mỗi ngôn ngữ lập trình. Trong C phép gán có hai dạng theo cú pháp sau đây.

Gán đơn giản

**biến = trị**

Gán phức tạp

**biến op = trị**

Với **trị** có thể là hằng, biến hoặc là biểu thức

**op** có thể là \* / % + - hoặc << >> & ^ |



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.6 CÁC PHÉP TOÁN CỦA C*

### **7.6.6 Toán tử gán**

Phép gán trị phức tạp

biến **op** = tri

chính là

biến = biến **op** tri

**Ví dụ:**

```
int a, b = 2;
```

```
a = 4;
```

```
b *= a * 3;
```

→ a = 4

→ b = 24



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.6 CÁC PHÉP TOÁN CỦA C*

### **7.6.6 Toán tử gán**

*Trong phép gán đơn,*

Nếu hai toán hạng có cùng kiểu thì toán hạng bên phải sẽ được gán vào toán hạng bên trái.

Nếu không, trước khi gán toán hạng bên phải **sẽ được chuyển theo kiểu của toán hạng bên trái**, điều này có thể sẽ gây ra kết quả sai hoặc không chính xác, nếu như kiểu của toán hạng bên trái thấp hơn kiểu của toán hạng bên phải.



# CHƯƠNG 7

## CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C

### 7.6 CÁC PHÉP TOÁN CỦA C

#### 7.6.6 Toán tử gán

Kiểu toán hạng trái	Kiểu toán hạng phải	Trị có thể mất sau khi gán
signed char	unsigned char	Giá trị > 127, thành số âm
char	short int	Mất trị từ bit 8 trở đi
char	int	Mất trị từ bit 8 trở đi
char	long	Mất trị từ bit 8 trở đi
short int	long int	Mất 16 bit cao (một int)
int	float	Mất phần thập phân và phần trị lớn hơn một int
float	double	Độ chính xác do làm tròn



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.6 CÁC PHÉP TOÁN CỦA C**

### **7.6.6 Toán tử gán**

Đối với phép gán **phức**, việc chuyển kiểu được thực hiện theo việc chuyển kiểu tự động trong khi thực hiện việc tính toán biểu thức và việc chuyển kiểu của phép gán đơn giản.

Phép gán phức hợp này tỏ ra rất hiệu quả nhất là khi các toán hạng bên trái là những biến khá dài.



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.6 CÁC PHÉP TOÁN CỦA C*

### **7.6.6 Toán tử gán**

**Ví dụ:**

Thay vì viết:

$n = n * (x + 5) + n * (a + 8);$

ta chỉ cần viết:

$n *= x + 5 + a + 8;$

Hoặc phức tạp hơn

$a[i][j] -= b[i][j];$

thay vì phải viết dài dòng

$a[i][j] = a[i][j] - b[i][j];$



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.6 CÁC PHÉP TOÁN CỦA C*

### **7.6.6 Toán tử gán**

Đối với C, nếu một biểu thức gán được kết thúc bằng một **dấu ";"** thì ta có một lệnh gán, còn nếu biểu thức gán này được sử dụng trong một biểu thức phức hợp khác thì biểu thức gán sẽ có trị là trị của biến sau khi gán.

**Ví dụ:**

```
int a = 4, b = 3;
```

```
b += (a = 2 * b) + (a *= b);
```

Thực hiện  $(a = 2 * b)$ , nghĩa là  $a=6$ , sau đó tính tiếp.





## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.6 CÁC PHÉP TOÁN CỦA C**

### **7.6.7 Toán tử phẩy - Biểu thức phẩy**

Cú pháp:

**biểu\_thức\_1, biểu\_thức\_kết\_quả**

Với biểu\_thức\_1 và biểu\_thức\_kết\_quả là hai biểu thức bất kỳ.

**Ví dụ:**

**$m = (a = 2, t = a + 3);$**

sẽ cho  $a = 2, t = 5$  và  $m = t = 5$

hoặc  **$x = (t = 1, t + 4);$**

sẽ cho  $t = 1$  và  $x = 5$



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.6 CÁC PHÉP TOÁN CỦA C*

### **7.6.8 Toán tử điều kiện - biểu thức điều kiện**

Trong ngôn ngữ C có một toán tử khá đặc biệt gọi là toán tử điều kiện, ký hiệu của toán tử điều kiện là hai dấu "?" và ":" theo cú pháp sau:

**dieu-kien ? bieu-thuc1 : bieu-thuc2**

với **dieu-kien** là một biểu thức bất kỳ có kết quả thuộc kiểu chuẩn (scalar type)

**bieu-thuc1, bieu-thuc2** là hai biểu thức bất kỳ và dĩ nhiên có thể là một biểu thức điều kiện khác.



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.6 CÁC PHÉP TOÁN CỦA C**

### **7.6.8 Toán tử điều kiện - biểu thức điều kiện**

**Ví dụ:**

Thay vì phải viết dài dòng

```
if ( i > 0 )
```

```
    n = 1;
```

```
else n = 0;
```

ta chỉ cần dùng biểu thức điều kiện

```
n =(i > 0) ? 1 : 0;
```



## ***CHƯƠNG 7*** ***CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C***

### ***7.6 CÁC PHÉP TOÁN CỦA C***

#### **7.6.8 Toán tử điều kiện - biểu thức điều kiện**

**Ví dụ:** Viết chương trình nhập một ký tự, đổi ký tự đó sang ký tự hoa nếu đó là ký tự thường.

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

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

```
main()
```

```
{
```

```
    char c;
```

```
    clrscr();
```

```
    printf("Nhap mot ky tu: ");
```



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.6 CÁC PHÉP TOÁN CỦA C**

### **7.6.8 Toán tử điều kiện - biểu thức điều kiện**

```
c = getchar();
```

(1)

```
c =( c >= 'a' && c <= 'z' ) ? c - 32 : c; (2)
```

```
printf ("Ky tu da duoc doi la: ");
```

```
putchar (c);
```

```
getch();
```

```
}
```



## ***CHƯƠNG 7***

# ***CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C***

## ***7.6 CÁC PHÉP TOÁN CỦA C***

### **7.6.9 Toán tử sizeof**

Đây là một toán tử cho ta kích thước của một biến hoặc một kiểu dữ liệu nào đó. Do phạm vi sử dụng của sizeof rất rộng và thường được dùng để lấy kích thước các kiểu dữ liệu phức hợp như struct, union... Việc sử dụng toán tử này cho phép ta không phải quan tâm đến chiều dài cụ thể của các biến.



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.6 CÁC PHÉP TOÁN CỦA C**

### **7.6.9 Toán tử sizeof**

Toán hạng của sizeof là một biến hoặc một kiểu dữ liệu bất kỳ nào đó đã định nghĩa.

*sizeof (biến)*

*sizeof biến*

*sizeof (kiểu)*

Kết quả của toán tử này là một giá trị nguyên chỉ kích thước (**tính bằng byte hoặc char**) của kiểu dữ liệu hoặc của biến đó. Biến hoặc kiểu này có thể là một biến hoặc một kiểu đơn giản hay phức hợp.



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.6 CÁC PHÉP TOÁN CỦA C*

### **7.6.9 Toán tử sizeof**

**Ví dụ:**

Nếu có một biến đã khai báo:

`double f;`

thì:

`sizeof (f);`

`sizeof f;`

hoặc:

`sizeof (double);`

sẽ cho ta kết quả là 8 (byte).





## *CHƯƠNG 7* *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

### *7.6 CÁC PHÉP TOÁN CỦA C*

#### **7.6.10 Độ ưu tiên của các toán tử**

Trong một biểu thức, các phép toán luôn được thực hiện theo một **mức độ ưu tiên** khác nhau, và nếu cùng một độ ưu tiên thì các phép toán sẽ được kết hợp với nhau **theo một trật tự nhất định**.



# CHƯƠNG 7

## CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C

### 7.6 CÁC PHÉP TOÁN CỦA C

#### 7.6.10 Độ ưu tiên của các toán tử

Độ ưu tiên	Phép toán	Thứ tự kết hợp
1	() [] ->.	Trái qua phải
2	! ~ ++ -+ (type) * & sizeof	Phải qua trái *
3	* / %	Trái qua phải
4	+ -	Trái qua phải
5	<< >>	Trái qua phải
6	< <= > >=	Trái qua phải
7	== !=	Trái qua phải
8	&	Trái qua phải
9	^^	Trái qua phải
10		Trái qua phải
11	&&	Trái qua phải
12		Trái qua phải
13	?:	Trái qua phải *
14	= += -= *= /= %= <<= >>= &=  = ^=	Phải qua trái *
15	,	Trái qua phải



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

### **7.7 CẤU TRÚC TỔNG QUÁT CỦA MỘT CHƯƠNG TRÌNH C**

Một chương trình C tổng quát bao hàm hai phần: phần khai báo đầu (header) và phần hàm (function).

*Phần khai báo đầu của một chương trình C bao gồm:*

- Các lệnh tiền xử lý: include, define ...
- Các khai báo hằng, biến ngoài ...
- Các prototype của các hàm được sử dụng trong chương trình



## *CHƯƠNG 7* *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ* *LIỆU CỦA C*

### *7.7 CẤU TRÚC TỔNG QUÁT CỦA MỘT CHƯƠNG* *TRÌNH C*

*Phần hàm của một chương trình C là phần định nghĩa các hàm sử dụng trong chương trình, trong các hàm này phải có hàm main().*



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

## **7.7 CẤU TRÚC TỔNG QUÁT CỦA MỘT CHƯƠNG TRÌNH C**

**Ví dụ:** Nhập một số kiểm tra số đó chẵn hay lẻ.

```
#include <stdio.h>
#include <conio.h>
int kiem_tra (int so);
/* ham kiem_tra nhan vao doi so
la mot so nguyen, tra ve tri
- 0 la so chan
- 1 la so le
*/
```

→ *phần khai báo*



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.7 CẤU TRÚC TỔNG QUÁT CỦA MỘT CHƯƠNG TRÌNH C*

```
main()  
{  
  
    int n;  
    clrscr();  
    printf("Nhap mot so: ");  
    scanf ("%d", &n);  
    if (kiem_tra(n))  
        printf("So da nhap la so le \n");  
}
```



## *CHƯƠNG 7*

# *CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C*

## *7.7 CẤU TRÚC TỔNG QUÁT CỦA MỘT CHƯƠNG TRÌNH C*

```
        else
            printf("So da nhap la so chan \n");
        getch();
    }
int kiem_tra (int so)
{
    return (so % 2 == 0)? 0:1;
}
```



# **CHƯƠNG 7**

## **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

### **BÀI TẬP CUỐI CHƯƠNG**

1. Viết chương trình in ra màn hình trị thập phân của các hằng sau đây

067

01234

0x1a1 0x89ad

0xfb

'h'

022

02365

2. Nhập ba số, tìm số lớn nhất và nhỏ nhất trong ba số đó.





## ***CHƯƠNG 7*** ***CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C***

### ***BÀI TẬP CUỐI CHƯƠNG***

**3. Nhập bốn số, sắp xếp theo thứ tự từ lớn tới nhỏ và từ nhỏ tới lớn theo menu sau:**

- 1. Từ lớn tới nhỏ**
- 2. Từ nhỏ tới lớn**
- 3. Kết thúc**

**Mời bạn chọn thao tác (1...3):**

**4. Nhập ba cạnh tam giác, kiểm tra ba cạnh đó có thỏa điều kiện hình thành tam giác không, in kết quả kiểm tra.**



## ***CHƯƠNG 7***

# ***CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C***

### ***BÀI TẬP CUỐI CHƯƠNG***

**5. Nhập ba cạnh tam giác, kiểm tra ba cạnh đó có thỏa điều kiện hình thành tam giác không, nếu thỏa in ra kết quả xem tam giác đó là tam giác gì (vuông thì vuông tại đâu, cân thì cân tại đâu ...)?**



## ***CHƯƠNG 7*** ***CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C***

### ***BÀI TẬP CUỐI CHƯƠNG***

**6.** Viết chương trình nhập một ký tự và xử lý theo yêu cầu sau:

- Nếu ký tự là hoa thì đổi sang thường, in kết quả đổi
- Nếu ký tự là thường thì không làm gì cả, in kết quả
- Nếu ký tự là ký số thì in ra màn hình câu: "Day la mot ky so".



## **CHƯƠNG 7**

# **CÁC THÀNH PHẦN CƠ BẢN VÀ CÁC KIỂU DỮ LIỆU CỦA C**

### **BÀI TẬP CUỐI CHƯƠNG**

7. Dùng hàm pow () để tính bình phương và lũy thừa ba của một số nhập từ bàn phím.

8. Viết chương trình đổi từ độ Fahrenheit (F) sang độ Celcius (C) theo công thức sau:

$$\frac{F - 32}{C} = \frac{9}{5}$$