



2.2 Biến, kiểu dữ liệu và các toán tử toán học

Nội dung

- Biến
- Hằng
- Các kiểu dữ liệu cơ bản trong C
- Biểu thức toán học
- Một số hàm toán học trong C
- Toán tử logic và toán tử trên bit
- Độ ưu tiên của các toán tử

2.2 Biến, hằng, kiểu dữ liệu

- **Biến (variable)** là đại lượng mà giá trị có thể thay đổi trong chương trình.
- Biến phải được khai báo trước khi sử dụng.
- **Tên biến** được đặt theo quy tắc **định danh**.
- Quy tắc đặt **định danh (identifier)**:
 - Gồm có: chữ cái, chữ số và dấu gạch dưới “_”
 - 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 từ khóa, và các định danh khác.

2.2 Biến, hằng, kiểu dữ liệu

Định danh hợp lệ	Định danh không hợp lệ
wiggles	\$Z]**
cat2	2cat
Hot_Tub	Hot-Tub
taxRate	tax rate
_kcab	don't
	int

Ngôn ngữ C phân biệt chữ hoa và chữ thường !

sum, Sum, sUm, suM, SUM là các tên biến khác nhau

2.2 Biến, hằng, kiểu dữ liệu

- Các kiểu số nguyên khác: **char**, **short**, **long**, **long long**
- Kích thước lưu trữ

Type	Macintosh Metrowerks CW (Default)	Linux on a PC	IBM PC Windows XP Windows NT	ANSI C Minimum
char	8	8	8	8
int	32	32	32	16
short	16	16	16	16
long	32	32	32	32
long long	64	64	64	64

- Để chỉ rõ hằng kiểu **long** ta dùng thêm ký hiệu **L** hoặc **l** ở sau

2.2 Biến, hằng, kiểu dữ liệu

- Số nguyên có dấu và số nguyên không dấu: **signed** và **unsigned**
 - Mặc định các kiểu số nguyên là **signed**
 - Khai báo số nguyên không dấu: **unsigned int**, **unsigned long**,...
 - Để chỉ ra rõ một hằng số nguyên là không dấu, dùng **u** hoặc **U** ở cuối
243U, 34u, 343454UL

2.2 Biến, hằng, kiểu dữ liệu

- Kiểu số thực : **float**
 - **3.**, **125.8**, và **-.0001** là các hằng số thực
 - **3.5e+2**, **.12e3** là các hằng số thực dưới dạng ký pháp khoa học
 - Kích thước biểu diễn : 32 bit
 - In ra số thực
 - **%f** dưới dạng dấu phẩy tính
 - **%e** dưới dạng ký pháp khoa học
 - **%g** tự điều chỉnh cho dễ nhìn
- ```
printf("%f %e %g", .00000012, 5.12, 50000.12);
```

## 2.2 Biến, hằng, kiểu dữ liệu

- Kiểu số thực mở rộng : **double**
  - Sử dụng 64 bit
  - Độ chính xác gấp đôi so với float
  - Để phân biệt 1 hằng số thực là float thì thêm ký hiệu **f** hoặc **F** ở cuối .34f 45.56F

## 2.2 Biến, hằng, kiểu dữ liệu

### ■ Kiểu ký tự : char

- 'A', 'v', '0', '\n' là các hằng ký tự
- In ra bằng %c
- Có thể dùng như một giá trị nguyên (chính là mã ASCII của ký tự đó)

```
printf("%c %d", 'A', 'A');
```

## 2.2 Biến, hằng, kiểu dữ liệu

### ■ Một số hằng ký tự

| Ký hiệu | Ý nghĩa         |
|---------|-----------------|
| \a      | Audible alert   |
| \b      | Backspace       |
| \f      | Form feed       |
| \n      | Newline         |
| \r      | Carriage return |
| \t      | Horizontal tab  |
| \v      | Vertical tab    |
| \\      | Backslash       |
| \"      | Double quote    |
| \'      | Single quote    |
| \?      | Question mark   |

## 2.2 Biến, hằng, kiểu dữ liệu

| Type                   | Constant Examples     | Printf char      |
|------------------------|-----------------------|------------------|
| char                   | 'a', '\n'             | %c               |
| short int              |                       | %hi, %hx, %ho    |
| unsigned short int     |                       | %hi, %hx, %ho    |
| int                    | 12, -97, 0xFFE0, 0177 | %i, %x, %o       |
| unsigned int           | 12u, 100U, 0xFFu      | %u, %x, %o       |
| long int               | 12L, -2001, 0xffffL   | %li, %lx, %lo    |
| unsigned long int      | 12UL, 100uL, 0xffeeUL | %lu, %lx, %lo    |
| long long int          | 0xe5e5e5e5LL, 50011   | %lli, %llx, %llo |
| unsigned long long int | 12ull, 0xffeeULL      | %llu, %llx, %llo |
| float                  | 12.34f, 3.1e-5f       | %f, %e, %g       |
| double                 | 12.34, 3.1e-5         | %f, %e, %g       |
| long double            | 12.341, 3.1e-5l       | %Lf, %Le, %Lg    |

## 2.2 Biến, hằng, kiểu dữ liệu

```
/* Example 2.2.1
 * kích thước các kiểu dữ liệu cơ bản */
#include <stdio.h>

int main(void)
{
 printf("Kiểu int %d bytes.\n", sizeof(int));
 printf("Kiểu char %d bytes.\n", sizeof(char));
 printf("Kiểu long %i bytes.\n", sizeof(long));
 printf("Kiểu double %u bytes.\n", sizeof(double));
 return 0;
}
```

## 2.2 Biến, hằng, kiểu dữ liệu

### ■ Khai báo biến

```
Kiểu_dữ_liệu tên_biến ;
Kiểu_dữ_liệu tên_biến = giá_trị_ban_đầu ;
```

```
int a;
float b, c, diem_thi;
double g, pi=3.1415, rad = 3.14;
```

### ■ Khai báo hằng dưới dạng biểu tượng (symbolic constant)

```
#define TÊN_HẰNG giá_trị
const kiểu_dữ_liệu TÊN_HẰNG = giá_trị ;
```

## 2.2 Biến, hằng, kiểu dữ liệu

```
/* Example 2.2.2 */
#include <stdio.h>

#define lai_xuat 0.013

int main(void)
{
 const int so_thang = 12;
 float tien_gui = 10e6;

 printf("Tien lai 1 nam : %g\n", tien_gui*so_thang*lai_xuat);

 return 0;
}
```

## Kiểu dữ liệu logic – Boolean

### ■ Kiểu\_Bool : kiểu logic chỉ có từ bản C99

### ■ Kiểu logic trong C :

- False tương ứng với 0
- True tương ứng với giá trị ≠ 0

### ■ Kiểu\_Bool : được định nghĩa trong <stdbool.h>

- Hai giá trị **true**, **false** được định nghĩa
- Có thể dùng **%i** để in ra biến kiểu\_Bool

```
_Bool sam = true;
if(sam) printf("TRUE");
else printf("FALSE");
```

## Biểu thức trong C

- Biểu thức toán học
- Các loại toán tử
- Độ ưu tiên của các toán tử
- Thay đổi độ ưu tiên của toán tử

## 2.2 Biến, hằng, kiểu dữ liệu

- **Biểu thức toán học** : gồm các toán tử và toán hạng
- Toán hạng có thể là **biến** hoặc là **hằng số**
- **Toán tử** :
  - Toán tử 1 ngôi: -, +, !
  - Toán tử hai ngôi: +, -, \*, /, %, ...
- Trình tự thực hiện theo thứ tự ưu tiên của các toán tử

## 2.2 Biến, hằng, kiểu dữ liệu

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

## 2.2 Biến, hằng, kiểu dữ liệu

```
#include <stdio.h> //Example 2.2.3
int main(void)
{
 int a = 100;
 int b = 2;
 int c = 25;
 int d = 4;
 int result;
 result = a - b; // trừ
 printf ("a - b = %i\n", result);
 result = b * c; // nhân
 printf ("b * c = %i\n", result);
 result = a / c; // chia
 printf ("a / c = %i\n", result);
 result = a + b * c; // thứ tự thực hiện
 printf ("a + b * c = %i\n", result);
 printf ("a * b + c * d = %i\n", a * b + c * d);
 return 0;
}
```

## Độ ưu tiên của toán tử

- Độ ưu tiên đề cập đến thứ tự thực thi các toán tử trong C
- Độ ưu tiên tạo nên cấu trúc phân cấp của loại toán tử này so với loại toán tử khác khi tính giá trị một biểu thức số học
- Độ ưu tiên của các toán tử này được thay đổi bởi các dấu ngoặc đơn trong biểu thức

| Loại toán tử | Toán tử                        | Tính kết hợp  |
|--------------|--------------------------------|---------------|
| Một ngôi     | -, ++, --                      | Phải đến trái |
| Hai ngôi     | ^ (pow(x,y) → x <sup>y</sup> ) | Trái đến phải |
| Hai ngôi     | *, /, %                        | Trái đến phải |
| Hai ngôi     | +, -                           | Trái đến phải |
| Hai ngôi     | =                              | Phải đến trái |

## 2.2 Biến, hằng, kiểu dữ liệu

### ■ Phép toán với số nguyên và toán tử một ngôi

```
#include <stdio.h> //Example 2.2.4
int main (void)
{
 int a = 25;
 int b = 2;
 float c = 25.0;
 float d = 2.0;
 printf ("6 + a / 5 * b = %i\n", 6 + a / 5 * b);
 printf ("a / b * b = %i\n", a / b * b);
 printf ("c / d * d = %f\n", c / d * d);
 printf ("-a = %i\n", -a);
 return 0;
}
```

## 2.2 Biến, hằng, kiểu dữ liệu

### ■ Phép chia module (%)

```
#include <stdio.h> //Example 2.2.5
int main (void)
{
 int a = 25, b = 5, c = 10, d = 7;
 printf ("a %% b = %i\n", a % b);
 printf ("a %% c = %i\n", a % c);
 printf ("a %% d = %i\n", a % d);
 printf ("a / d * d + a %% d = %i\n",
 a / d * d + a % d);
 return 0;
}
```

### ■ Chuyển đổi giữa số nguyên và số thực

```
// Example 2.2.6
#include <stdio.h>
int main (void)
{
 float f1 = 123.125, f2;
 int i1, i2 = -150;

 i1 = f1; // floating to integer conversion
 printf ("%f assigned to an int produces %i\n", f1, i1);

 f1 = i2; // integer to floating conversion
 printf ("%i assigned to a float produces %f\n", i2, f1);

 f1 = i2 / 100; // integer divided by integer
 printf ("%i divided by 100 produces %f\n", i2, f1);

 f2 = i2 / 100.0; // integer divided by a float
 printf ("%i divided by 100.0 produces %f\n", i2, f2);

 f2 = (float) i2 / 100; // type cast operator
 printf ("(float) %i divided by 100 produces %f\n", i2, f2);

 return 0;
}
```

## 2.2 Biến, hằng, kiểu dữ liệu

### ■ Toán tử chuyển kiểu (type cast operator)

`f2 = (float) i2 / 100;`

`(int) 29.55 + (int) 21.99`

Hoặc `int(29.55) + int(21.99)`

| Narrow conversion |        | Wide conversion |
|-------------------|--------|-----------------|
| lose<br>precision | ↑      | ↓               |
|                   | short  |                 |
|                   | int    |                 |
|                   | float  |                 |
|                   | double |                 |

### ■ Chuyển kiểu tự động

`char → int → long int → float → double → long double`

## Một số hàm toán học trong C

| Hàm                    | Ý nghĩa                                                          | Kí hiệu toán học    | Ví dụ                                                              |
|------------------------|------------------------------------------------------------------|---------------------|--------------------------------------------------------------------|
| <code>sqrt(x)</code>   | Căn bậc 2 của x                                                  | $\sqrt{x}$          | <code>sqrt(16.0)</code> bằng 4.0                                   |
| <code>pow(x, y)</code> | x mũ y                                                           | $x^y$               | <code>pow(2, 3)</code> bằng 8                                      |
| <code>exp(x)</code>    | e mũ x                                                           | $e^x$               | <code>exp(1.0)</code> bằng 2.718282                                |
| <code>log(x)</code>    | logarithm tự nhiên (cơ số e) của x                               | $\ln x$             | <code>log(2.718282)</code> bằng 1.0                                |
| <code>log10(x)</code>  | logarithm cơ số 10 của x                                         | $\log x$            | <code>log10(100)</code> bằng 2                                     |
| <code>sin(x)</code>    | sin của x                                                        | $\sin x$            | <code>sin(0.0)</code> bằng 0.0                                     |
| <code>cos(x)</code>    | cosin của x                                                      | $\cos x$            | <code>cos(0.0)</code> bằng 1.0                                     |
| <code>tan(x)</code>    | tang của x                                                       | $\tan x$            | <code>tan(0.0)</code> bằng 0.0                                     |
| <code>ceil(x)</code>   | phần nguyên giả của x, tức là số nguyên nhỏ nhất không nhỏ hơn x | $\lceil x \rceil$   | <code>ceil(2.5)</code> bằng 3<br><code>ceil(-2.5)</code> bằng -2   |
| <code>floor(x)</code>  | phần nguyên non của x, tức là số nguyên lớn nhất không lớn hơn x | $\lfloor x \rfloor$ | <code>floor(2.5)</code> bằng 2<br><code>floor(-2.5)</code> bằng -3 |

## Ví dụ

- VD1. Viết chương trình tính khoảng cách giữa hai điểm A(1, 3, 5) và B(0.5, 7, 6.5) trong không gian 3 chiều

- VD2. Viết chương trình tính

$$\log_3(56)$$

## Toán tử quan hệ và Logic

**Toán tử quan hệ** : Kiểm tra mối quan hệ giữa hai biến hay giữa một biến và một hằng

### Toán tử quan hệ

| Toán tử            | Ý nghĩa           |
|--------------------|-------------------|
| <code>&gt;</code>  | Lớn hơn           |
| <code>&gt;=</code> | Lớn hơn hoặc bằng |
| <code>&lt;</code>  | Nhỏ hơn           |
| <code>&lt;=</code> | Nhỏ hơn hoặc bằng |
| <code>==</code>    | Bằng              |
| <code>!=</code>    | Không bằng        |

## Toán tử quan hệ và Logic (tt.)

**Toán tử Logic**: là những ký hiệu dùng để kết hợp hay phủ định biểu thức chứa các toán tử quan hệ

| Toán tử                 | Ý nghĩa                                                                                    |
|-------------------------|--------------------------------------------------------------------------------------------|
| <code>&amp;&amp;</code> | <b>AND</b> : Kết quả là True khi cả 2 điều kiện đều đúng                                   |
| <code>  </code>         | <b>OR</b> : Kết quả là True khi chỉ một trong hai điều kiện là đúng                        |
| <code>!</code>          | <b>NOT</b> : Tác động trên các giá trị riêng lẻ, chuyển đổi True thành False và ngược lại. |

Ví dụ: `(5>10) && (7<20)`

Những biểu thức dùng toán tử Logic trả về 0 thay cho false và 1 thay cho true



## Toán tử Logic nhị phân

**Toán tử logic nhị phân:** thực hiện giống toán tử logic nhưng trên các bit.

| Toán tử                                         | Mô tả                                                                             |
|-------------------------------------------------|-----------------------------------------------------------------------------------|
| <b>Bitwise AND</b><br>( $x \& y$ )              | Mỗi vị trí của bit trả về kết quả là 1 nếu bit của hai toán hạng là 1.            |
| <b>Bitwise OR</b><br>( $x   y$ )                | Mỗi vị trí của bit trả về kết quả là 1 nếu bit của một trong hai toán hạng là 1.  |
| <b>Bitwise NOT</b><br>( $\sim x$ )              | Đảo giá trị của bit (1 thành 0 và ngược lại).                                     |
| <b>Bitwise XOR</b><br>( $x \wedge y$ )          | Thực hiện XOR trên 2 bit (cho giá trị 0 nếu 2 bit bằng nhau, và 1 nếu ngược lại). |
| <b>&lt;&lt;, &gt;&gt; dịch bit trái và phải</b> |                                                                                   |

## Toán tử Logic nhị phân (tt.)

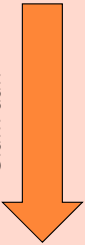
Ví dụ

- $10 \& 15 \rightarrow 1010 \& 1111 \rightarrow 1010 \rightarrow 10$
- $10 | 15 \rightarrow 1010 | 1111 \rightarrow 1111 \rightarrow 15$
- $10 \wedge 15 \rightarrow 1010 \wedge 1111 \rightarrow 0101 \rightarrow 5$
- $\sim 10 \rightarrow \sim 1010 \rightarrow 1...11110101 \rightarrow -11$
- $15 \ll 3 \rightarrow 1111 \ll 3 \rightarrow 1111000$
- $15 \gg 3 \rightarrow 1111 \gg 3 \rightarrow 1$

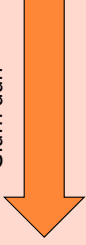
## Độ ưu tiên giữa các toán tử

Khi một biểu thức có nhiều loại toán tử thì độ ưu tiên giữa chúng phải được thiết lập.

| Thứ tự ưu tiên | Kiểu toán tử         |
|----------------|----------------------|
| 1              | Số học (Arithmetic)  |
| 2              | So sánh (Comparison) |
| 3              | Logic (Logical)      |

| Độ ưu tiên                                                                                        | Toán tử                                      | Trình tự kết hợp |
|---------------------------------------------------------------------------------------------------|----------------------------------------------|------------------|
| Giảm dần<br> | ( )<br>[ ]<br>.<br>-><br>++ -- (dạng hậu tố) | Trái sang phải   |
|                                                                                                   | ++ --<br>+ -<br>! ~<br>(type)<br>*           | Phải sang trái   |
|                                                                                                   | &<br>sizeof                                  |                  |
|                                                                                                   | * / %                                        | Trái sang phải   |
|                                                                                                   | + -                                          | Trái sang phải   |
|                                                                                                   | << >>                                        | Trái sang phải   |
|                                                                                                   | < <=<br>> >=                                 | Trái sang phải   |
|                                                                                                   |                                              |                  |
|                                                                                                   |                                              |                  |
|                                                                                                   |                                              |                  |

## Độ ưu tiên của các toán tử

| Độ ưu tiên                                                                                    | Toán tử                                     | Trình tự kết hợp |
|-----------------------------------------------------------------------------------------------|---------------------------------------------|------------------|
| Giảm dần<br> | == !=                                       | Trái sang phải   |
|                                                                                               | &                                           | Trái sang phải   |
|                                                                                               | ^                                           | Trái sang phải   |
|                                                                                               | !                                           | Trái sang phải   |
|                                                                                               | &&                                          | Trái sang phải   |
|                                                                                               |                                             | Trái sang phải   |
|                                                                                               | ?: (toán tử điều kiện)                      | Phải sang trái   |
|                                                                                               | =                                           | Phải sang trái   |
|                                                                                               | += -=<br>*= /=<br>%= &=<br>^=  =<br><<= >>= |                  |
|                                                                                               | , (ngăn cách biểu thức)                     | Trái sang phải   |

## Độ ưu tiên giữa các toán tử (tt.)

Ví dụ :

$$2*3+4/2 > 3 \text{ AND } 3<5 \text{ OR } 10<9$$

Việc tính toán như sau :

$$[2*3+4/2] > 3 \text{ AND } 3<5 \text{ OR } 10<9$$

Toán tử số học sẽ được tính trước

$$[(2*3)+[4/2]] > 3 \text{ AND } 3<5 \text{ OR } 10<9$$

$$[6+2] > 3 \text{ AND } 3<5 \text{ OR } 10<9$$

$$[8 > 3] \text{ AND } [3 < 5] \text{ OR } [10 < 9]$$

## Độ ưu tiên giữa các toán tử (tt.)

Kể đến là toán tử so sánh có cùng độ ưu tiên. Ta áp dụng quy tắc tính từ trái sang phải.

**True AND True OR False**

Cuối cùng là toán tử kiểu Logic.  
AND sẽ có độ ưu tiên cao hơn OR

**[True AND True] OR False**  
**True OR False**  
**True**

## Thay đổi độ ưu tiên

- ☐ Dấu ngoặc đơn ( ) có độ ưu tiên cao nhất
- ☐ Độ ưu tiên của các toán tử có thể được thay đổi bởi dấu ngoặc đơn
  - Toán tử có độ ưu tiên thấp hơn nếu đặt trong dấu ngoặc đơn sẽ được thực thi trước
  - Khi các cặp ngoặc đơn lồng nhau ( ( ( ) ) ), cặp ngoặc đơn trong cùng nhất sẽ được thực thi trước
- ☐ Nếu trong biểu thức có nhiều cặp ngoặc đơn thì việc thực thi sẽ theo thứ tự từ trái sang phải

### Thay đổi độ ưu tiên (tt.)

Ví dụ :

$$5+9*3^2-4 > 10 \text{ AND } (2+2^4-8/4 > 6 \text{ OR } (2<6 \text{ AND } 10>11))$$

Cách tính :

$$1) 5+9*3^2-4 > 10 \text{ AND } (2+2^4-8/4 > 6 \text{ OR } (\text{True AND False}))$$

Dấu ngoặc đơn bên trong sẽ được tính trước

$$2) 5+9*3^2-4 > 10 \text{ AND } (2+2^4-8/4 > 6 \text{ OR False})$$

### Thay đổi độ ưu tiên (tt.)

$$3) 5+9*3^2-4 > 10 \text{ AND } (2+16-8/4 > 6 \text{ OR False})$$

Kể đến dấu ngoặc đơn ở ngoài được tính đến

$$4) 5+9*3^2-4 > 10 \text{ AND } (2+16-2 > 6 \text{ OR False})$$

$$5) 5+9*3^2-4 > 10 \text{ AND } (18-2 > 6 \text{ OR False})$$

$$6) 5+9*3^2-4 > 10 \text{ AND } (16 > 6 \text{ OR False})$$

$$7) 5+9*3^2-4 > 10 \text{ AND } (\text{True OR False})$$

$$8) 5+9*3^2-4 > 10 \text{ AND True}$$

### Thay đổi độ ưu tiên (tt.)

$$9) 5+9*9-4>10 \text{ AND True}$$

Biểu thức bên trái được tính trước

$$10) 5+81-4>10 \text{ AND True}$$

$$11) 86-4>10 \text{ AND True}$$

$$12) 82>10 \text{ AND True}$$

$$13) \text{True AND True}$$

$$14) \text{True}$$