TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

TIN HỌC ĐẠI CƯƠNG

Phần 2. Lập trình

Chương 2. Kiểu dữ liệu và biểu thức trong C

Nội dung

2.1. Các kiểu dữ liệu chuẩn trong C

Kiểu dữ liệu	**Ý nghĩa**	**Kích thước**	**Miền dữ liệu**
unsigned char	Kí tự	1 byte	0 ÷ 255
char	Kí tự	1 byte	-128 ÷ 127
unsigned int	Số nguyên không dấu	2 bytes (hoặc 4 bytes)	0÷65,535
short int	Số nguyên có dấu	2 bytes	-32,768÷32,767
int	Số nguyên có dấu	***2 bytes*** (hoặc 4 bytes)	-32,768÷32,767

2.1. Các kiểu dữ liệu chuẩn trong C

Kiểu dữ liệu	Ý nghĩa	Kích thước	Miền dữ liệu
unsigned long	Số nguyên không dấu	4 bytes	0 ÷ 4,294,967,295
long	Số nguyên có dấu	4 bytes	-2,147,483,648 ÷ 2,147,483,647
float	Số thực dấu phẩy động, độ chính xác đơn	4 bytes	± 3.4E-38 ÷ ± 3.4E+38
double	Số thực dấu phẩy động, độ chính xác kép	8 bytes	± 1.7E-308 ÷ ± 1.7E+308
long double	Số thực dấu phẩy động, độ chính xác kép mở rộng	16 bytes	± 3.4E-4932 ÷ ± 1.1E+4932

Nội dung

2.1. Các kiểu dữ liệu chuẩn trong C

2.2. Khai báo và khởi tạo biến, hằng

2.3. Biểu thức trong C

2.4. Các phép toán trong C

2.5. Một số toán tử đặc trưng

2.6. Các lệnh vào ra dữ liệu với các biến

2.2.1. Khai báo và khởi tạo biến

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

```
kieu_du_lieu ten_bien;
kieu_du_lieu ten_bien_1, …, 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 số thực 4 byte (float) như sau:

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

2.2.1. Khai báo và khởi tạo biến (2)

Kết hợp khai báo và khởi tạo

- ***Cú pháp:***

```
kieu_du_lieu ten_bien = gia_tri_ban_dau;
kieu_du_lieu bien_1 = gia_tri_1,  bien_N =
  gia_tri_N;
```

- ***Ví dụ:***

```
int a = 3;// sau lenh nay bien a se co
  gia tri bang 3
float x = 5.0, y = 7.6; // sau lenh nay x
  co gia tri 5.0, y co gia tri 7.6
```

2.2.2. 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
```

2.2.2. 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;
```

2.2.2. Khai báo hằng

- ***Chú ý:***
  - Giá trị của các hằng phải được xác định ngay khi khai báo.
  - Trong chương trình, **KHÔNG thể thay đổi** được giá trị của hằng.
  - **`#define`** là chỉ thị tiền xử lý (preprocessing directive)
    - Dễ đọc, dễ thay đổi
    - Dễ chuyển đổi giữa các nền tảng phần cứng hơn
    - Tốc độ nhanh hơn

Nội dung

2.1. Các kiểu dữ liệu chuẩn trong C

2.2. Khai báo và khởi tạo biến, hằng

2.3. Biểu thức trong C

2.4. Các phép toán trong C

2.5. Một số toán tử đặc trưng

2.6. Các lệnh vào ra dữ liệu với các biến

2.3.1. Các loại biểu thức

a. Biểu thức số học:

- Là biểu thức mà giá trị của nó là các 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:

```
3 * 3.7
8 + 6 / 3
a + b - c  // Với a, b, c là các biến thuộc một kiểu dữ liệu số nào đó
```

2.3.1. 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 **!**

2.3.1. Các loại biểu thức

- Ví dụ về biểu thức logic:

```
(5 > 7) && (9 != 10)   // FALSE
0 || 1                 // TRUE
(5 > 7) || (9 != 10)   // TRUE
0                      // FALSE
!0                     // TRUE
3                      // TRUE
!3                     // FALSE
(a > b) && (a < b)     // FALSE
```

2.3.1. 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.

2.3.1. Các loại biểu thức

- Ví dụ về biểu thức quan hệ:

```
5 > 7           // FALSE
9 != 10         // TRUE
2 >= 2          // TRUE
a > b           // ???
a + 1 > a       // TRUE
```

Sử dụng biểu thức

- Làm vế phải của lệnh gán.

  x = (y + 1) / z

- Làm toán hạng trong các biểu thức khác
- Làm tham số thực trong lời gọi hàm
- Làm chỉ số trong các cấu trúc lặp **`for`**, **`while`**, **`do while`**.
- Làm biểu thức kiểm tra trong các cấu trúc rẽ nhánh **`if`**, **`switch`**.

Nội dung

2.1. Các kiểu dữ liệu chuẩn trong C

2.2. Khai báo và khởi tạo biến, hằng

2.3. Biểu thức trong C

2.4. Các phép toán trong C

2.5. Một số toán tử đặc trưng

2.6. Các lệnh vào ra dữ liệu với các biến

2.4. Các phép toán trong C

- Bao gồm:
  - Nhóm các phép toán số học
  - Nhóm các phép toán quan hệ
  - Nhóm các phép toán logic
  - Nhóm các phép toán thao tác trên bit
- Ngoài ra C còn cung cấp một số phép toán khác như phép gán, phép lấy địa chỉ …

2.4.1. Phép toán số học

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

2.4.2. 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)

2.4.2. Phép toán trên bit

Toán tử	Ý nghĩa	Kiểu dữ liệu của toán hạng	Ví dụ
^	Phép HOẶC LOẠI TRỪ 2 số nhị phân TRỪ 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 0) 101 ^ 110(có giá trị 011)
<<	Phép DỊCH TRÁI nhị phân	Số nhị phân	a << n (có giá trị $a*2^n$) 101 << 2(có giá trị 10100)
>>	Phép DỊCH PHẢI nhị phân	Số nhị phân	a >> n (có giá trị $a/2^n$) 101 >> 2 (có giá trị 1)
~	Phép ĐẢO BIT Phép lấy BÙ 1	Số nhị phân	~ 0 (có giá trị 1) ~ 1 (có giá trị 0) ~ 110 (có giá trị 001)

2.4.3. 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)

2.4.4. 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 >= 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)

2.4.5. Phép toán gán

- Cú pháp:

```
tên_biến = biểu_thức;
```

- Lấy giá trị của ***biểu_thức*** gán cho ***tên_biến***
- Ví dụ:

```
int a, b, c;
a = 3;
b = a + 5;
c = a * b;
```

2.4.5. Phép toán gán

- Biểu thức gán là biểu thức nên nó cũng có giá trị.
- Giá trị của biểu thức gán bằng giá trị của biểu_thức:

  → Có thể gán giá trị của biểu thức gán cho một biến khác hoặc sử dụng như một biểu thức bình thường
- Ví dụ:

```
int a, b, c;
a = b = 2007;
c = (a = 20) * (b = 30); // c = ?
```

2.4.5. Phép toán gán

- Phép toán gán thu gọn:

  `x = x + y;` giống như

  `x += y;`

- Dạng lệnh gán thu gọn này còn áp dụng được với các phép toán khác: +, -, *, /, %, >>, <<, &, |, ^

2.4.6. 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	& (AND bit)	----->
9	^ (XOR bit)	----->
10	| (OR bit)	----->
11	&&	----->
12	||	----->
13	?:	<-----
14	= += -=	<-----

Ví dụ

Ví dụ 1:

```
a < 10 && 2 * b < c
→  ( a < 10 ) && ( ( 2 * b ) < c )
```

Ví dụ 2:

```
int a = 35, b = 14, c = 5, d = 6;
int e;
e = (a + b) * c / d;
e = ((a + b) * c) / d;
e = (a + b) * (c / d);
e = a + (b * c) / d;
```

Ví dụ 3:

```
int a = 10, b = 15;
int c = a & 0x0F + b | 0x0E;
```

Nội dung

2.1. Các kiểu dữ liệu chuẩn trong C

2.2. Khai báo và khởi tạo biến, hằng

2.3. Biểu thức trong C

2.4. Các phép toán trong C

2.5. Một số toán tử đặc trưng

2.6. Các lệnh vào ra dữ liệu với các biến

2.5.1. Các phép toán tăng giảm một đơn vị

- Tăng hoặc giảm một đơn vị cho biến:

```
<tên biến> = <tên biến> + 1;
        → <tên biến>++;
<tên biến> = <tên biến> - 1;
        → <tên biến>--;
```

Ví dụ:

```
int a = 5;
float x = 10;
a++; // tương đương với a = a + 1;
x--; // tương đương với x = x – 1;
```

Tiền tố và hậu tố

- **Tiền tố:** Thay đổi giá trị của biến trước khi sử dụng
- **Hậu tố:** Tính toán giá trị của biểu thức bằng giá trị ban đầu của biến, sau đó mới thay đổi giá trị của biến
- Ví dụ:

```
int a, b, c;
a = 3;          // a bằng 3
b = a++;        // Dạng hậu tố
                // b bằng 3, a bằng 4
c = ++b;        // Dạng tiền tố
                // b bằng 4, c bằng 4
```

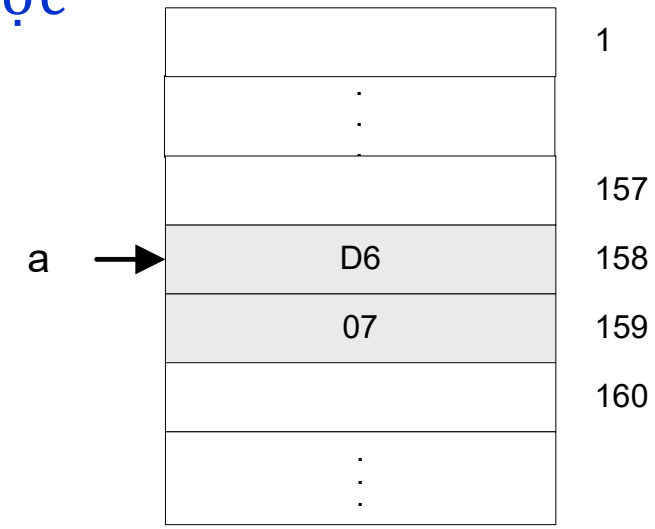
2.5.2. Phép toán lấy địa chỉ biến (&)

- Biến thực chất là một vùng nhớ được đặt tên (là tên của biến) trên bộ nhớ của máy tính.
- Mọi ô nhớ trên bộ nhớ máy tính đều được đánh địa chỉ. Do đó mọi biến đều có địa chỉ.
- Cú pháp:

```
&<tên biến>;
```

- Ví dụ:

```
int a = 2006;
→ &a; // co gia tri la 158 hay 9E
```



2.5.3. Phép toán chuyển đổi kiểu bắt buộc

- Chương trình dịch sẽ tự động chuyển đổi kiểu

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

- Ngược lại
  - Số nguyên **long int** 50,000 không phải là một số nguyên kiểu **int** vì phạm vi biểu diễn của kiểu **int** là từ (-32,768 đến 32,767).

  → Phải ép kiểu

- Cú pháp:

  `(<kiểu dữ liệu mới>) <biểu thức>;`

2.5.3. Phép toán chuyển đổi kiểu bắt buộc (2)

- Ví dụ:

```
#include <stdio.h>
#include <conio.h>
void main()
{
  long int li; int i; float f;
  clrscr();
  li = 0x123456; f = 123.456;
  i = (int) li;
  printf("\n li = %ld; i = %d", li, i);
  i = (int) f;
  printf("\n f = %f; i = %d", f, i);
  getch();
}
```

Kết quả

li = 1193046; i = 13398

f = 123.456000; i = 123

2.5.4. Biểu thức điều kiện

- Cú pháp

  biểu_thức_1 ? biểu_thức_2 : biểu_thức_3

  Giá trị của biểu thức điều kiện
  - Giá trị của biểu_thức_2 nếu biểu_thức_1 có giá trị khác 0 (tương ứng với giá trị logic ĐÚNG),
  - Ngược lại: Giá trị của biểu_thức_3 nếu biểu_thức_1 có giá trị bằng 0 (tương ứng với giá trị logic SAI).
- Ví dụ:

```
float x, y, z;      // khai báo biến
x = 3.8; y = 7.6;   // gán giá trị cho các biến x, y
z = (x<y) ? x : y; // z sẽ có giá trị bằng giá trị
                    // nhỏ nhất trong 2 số x và y
```