NHẬP MÔN LẬP TRÌNH

CHƯƠNG 2. CÁC THÀNH PHẦN CƠ BẢN CỦA NNLT C/C++

Mục tiêu

- Hiểu và vận dụng các kiểu dữ liệu cơ bản, hằng, biến của NNLT C.
- Hiểu và vận dụng các phép toán cơ bản của NNLT C.
- Biết viết các dạng biểu thức toán học, biểu thức quan hệ, biểu thức logic dưới dạng NNLT C.
- Hiểu ý nghĩa và vận dụng đúng cách các toán tử tăng, giảm, biểu thức điều kiện, độ ưu tiên các toán tử.

Nội dung

1. Danh hiệu
2. Biến
3. Các kiểu dữ liệu
4. Hằng số
5. Biểu thức
6. Các phép toán
7. Bài tập

1. Danh hiệu

Có 4 loại danh hiệu khác nhau: ký hiệu, từ khóa, tên và chú thích.

Ký hiệu: là tập kí tự hợp lệ trong ngôn ngữ C bao gồm:

- 52 kí tự chữ: A, B, ..., Z và a, b, ..., z.
- 10 kí tự số, các kí hiệu toán học: +, -, *, /, =,<,>,(,)
- Các kí tự đặc biệt như: ., ; : [ ] {} ? ! \ & \ | # $ " ' @ ^…
- Kí tự gạch nối _
- Dấu cách (khoảng trắng) dùng để phân cách giữa các từ

1. Định danh

Từ khóa: là các từ sử dụng để dành riêng trong ngôn ngữ lập trình C

Các từ khóa không được sử dụng làm các biến, hằng.

Không được định nghĩa lại các từ khoá.

Bảng liệt kê các từ khoá:

auto	break	case	char	continue	default	do	double	
else	extern	float	for	goto	if	int	long	register
return	short	sizeof	static	struct	switch	typedef	union	unsigned
void	while							
_cs	_ds	_es	_ss	_AH	_AL	_AX	_BH	_BL
_BX	_CH	_CL	_CX	_DH	_DL	_DX	_BP	_DI
_SI	_SP							

1. Danh hiệu

Tên: là một dãy các ký tự liền nhau bắt đầu bằng chữ cái hoặc ký tự gạch dưới.

Quy tắc đặt tên:

- Tên là một dãy các ký tự liền nhau bắt đầu bằng chữ cái hoặc ký tự gạch dưới theo sau là chữ cái, dấu gạch dưới, chữ số.
- Một tên không được chứa các kí tự đặc biệt như dấu cách, dấu chấm câu,…
- Theo tiêu chuẩn C các kí tự chữ thường và hoa thì xem như khác nhau.

Ví dụ: biến ADD, add và Add là khác nhau.

- Tên một biến nên có ý nghĩa, gợi tả và mô tả rõ kiểu dữ liệu của nó.

Ví dụ: nếu tìm tổng của 2 số thì tên biến lưu trữ tổng nên đặt là sum (tổng).

1. Danh hiệu

Chú thích: là những dòng mô tả ý nghĩa câu lệnh đang dùng, giải thích ý nghĩa của một hàm nào đó.

Phần nội dung ghi chú này khi biên dịch sẽ được bỏ qua. Trong ngôn ngữ lập trình C, nội dung chú thích phải được viết trong cặp dấu /* và */ hoặc sau dấu "//" nếu chú thích nằm trên một dòng.

Ví dụ:

```
#include <stdio.h>
#include <conio.h>
int main ()
{  int a, b;  //Khai báo 2 biến số nguyên
   /* Nhập vào 2 số nguyên a và b */
   printf("\n Nhap vao hai so nguyen a, b: \t");
   scanf("%d %d",&a, &b);
   return (a+b);
}
```

2. Biến

- **Biến** là một tên đại diện cho một giá trị dữ liệu cần lưu trữ.
- Khi biến được tạo sẽ xuất hiện một vùng nhớ để lưu trữ giá trị của biến.
- Kiểu dữ liệu quyết định tổng số bộ nhớ được chỉ định. Những tên được gán cho biến giúp chúng ta sử dụng lại dữ liệu khi cần đến.

Cú pháp: khai báo biến

<Kiểu dữ liệu> <Tên biến> [= <Giá trị>];

Ví dụ:

```
void main()
{   int sum;          //khai báo biến số nguyên sum
    sum = 10 + 25 + 50;
    printf("\n 10 + 25 + 50 = %d", sum);
}
```

3. Các kiểu dữ liệu

Các loại dữ liệu khác nhau được lưu trữ trong biến là :

- Số (Numbers)
  - Các số nguyên.
    Ví dụ: 10 hay 178993455.
  - Các số thực.
    Ví dụ: 15.22 hay 15463452.25.
  - Các số dương.
  - Các số âm.
- Tên

  Ví dụ: John.
- Giá trị luận lý

  Ví dụ: đúng hay sai (true – false)

3. Các kiểu dữ liệu

C có 5 kiểu dữ liệu cơ bản. Tất cả những kiểu dữ liệu khác dựa vào một trong số những kiểu này. 5 kiểu dữ liệu đó là:

- **int:** là một số nguyên, về cơ bản nó biểu thị kích cỡ tự nhiên của các số nguyên **(integers)**.
- **float** và **double:** dùng cho các số có dấu chấm động. Kiểu **float** (số thực) chiếm 4 byte và có thể biểu diễn 6 số phần thập phân, trong khi **double** chiếm 8 byte và có thể biểu diễn 10 số phần thập phân.
- **char:** chiếm 1 byte và có khả năng lưu một ký tự đơn.
- **void:** được dùng điển hình để khai báo một hàm không trả về giá trị.

3. Các kiểu dữ liệu

a. Kiểu char:

Kiểu dữ liệu **char** được dùng để lưu trữ một ký tự đơn.

Một kiểu dữ liệu **char** có thể lưu một ký tự đơn được đặt trong hai dấu nháy đơn (‘’). Khi truy xuất giá trị của một biến kiểu **char** ta dùng kí hiệu đại diện %c.

Cú pháp: **char <tên biến> [= ‘kí tự’];**

Ví dụ:

```
#include <stdio.h>
void main()
{   char c = ‘a’;
    printf(“\nXuat du lieu bien c = %c”, c);
    printf(“\nNhap vao ki tu c = ”);  scanf(“%c”, &c);
    printf(“\nXuat ky tu c vua nhap:  %c”);
}
```

3. Các kiểu dữ liệu

b. Kiểu int:

Kiểu dữ liệu **int** được dùng để lưu trữ một số nguyên có miền giá trị từ -2^{15} đến $2^{15} - 1$.

Khi truy xuất giá trị của một biến kiểu **int** ta dùng kí hiệu đại diện %d.

Cú pháp: **int <Tên biến>;**

Ví dụ:

```
#include <stdio.h>
void main()
{   int a = 12345;
    printf("\nXuat gia tri bien a = %d", a);
    printf("\Nhap vao so nguyen a = ");      scanf("%d", &a);
    printf("\nXuat gia tri a vua nhap: %d", a);
}
```

3. Các kiểu dữ liệu

c. Kiểu float và double:

- Một số thực kiểu ***float*** có giá trị trong khoảng 1.2E-38 ÷ 3.4E+38 với độ chính xác khoảng 6 số phần thập phân.
- Một số thực kiểu ***double*** có giá trị trong khoảng 2.2E – 308 ÷1.8E + 308 với độ chính xác khoảng 10 số phần thập phân.
- Khi truy xuất giá trị của một biến kiểu số thực ta dùng kí hiệu đại diện **%f**.
- Nếu cách hiển thị một số thực là **%.nf** khi đó giá trị số hiển thị **n** kí tự cho phần thập phân. Ví dụ như %.5f thì 5 kí tự cho phần thập phân của số hiển thị.

Cú pháp: **float <Tên biến>;**

double <Tên biến>;

3. Các kiểu dữ liệu

c. Kiểu float và double:

Ví dụ:

```
#include<stdio.h>
void main()
{
    float a;
    printf("nhap gia tri cua a: ");
    scanf("%f", &a);
    printf("Ket qua: %.1f\n",a);
    printf("Ket qua: %.3f\n",a);
    printf("Ket qua: %.6f\n",a);
}
```

3. Các kiểu dữ liệu

d.Kiểu void:

C có một kiểu dữ liệu đặc biệt gọi là **void**. Trong C, các hàm số thường trả về dữ liệu thuộc một kiểu nào đó. Tuy nhiên, khi một hàm không trả về kết quả thì kiểu trả về của hàm sẽ là kiểu **void**.

Ví dụ:

```
void Giai_PT(int a, int b);
```

3. Các kiểu dữ liệu

e. Các kiểu dữ liệu bổ sung

Các kiểu dữ liệu bổ sung bao gồm **unsigned, signed, short, long**.

Các kiểu dữ liệu bổ sung kết hợp với các dữ liệu cơ sở làm thay đổi miền giá trị của chúng.

Thứ tự kết hợp là: kiểu dữ liệu bổ sung → kiểu dữ liệu cơ sở.

*Các kiểu có dấu (**signed**) và không dấu (**unsigned**)*

Kiểu **signed** chỉ sử dụng với kiểu dữ liệu **char** vì **char** là kiểu mặt định không dấu.

Kiểu **unsigned** chỉ rõ rằng một biến chỉ nhận giá trị dương.

Ví dụ: ***unsigned*** *int varNum = 50000;*

Ý nghĩa: biến varNum vẫn được cấp phát vùng nhớ 2 byte. Tuy nhiên, giá trị mà mà varNum nhận được nằm trong khoảng từ 0 đến 65535, thay vì từ -32768 tới 32767 như kiểu ***int*** *quy định.*

3. Các kiểu dữ liệu

*Các kiểu có dấu (**signed**) và không dấu (**unsigned**)*

Kiểu **signed** chỉ sử dụng với kiểu dữ liệu **char** vì **char** là kiểu mặt định không dấu.

Kiểu **unsigned** chỉ rõ rằng một biến chỉ nhận giá trị dương.

Ví dụ:

unsigned *int varNum = 50000;*

Ý nghĩa: biến *varNum* vẫn được cấp phát vùng nhớ 2 byte. Tuy nhiên, giá trị mà *varNum* nhận được nằm trong khoảng từ 0 đến 65535, thay vì từ -32768 tới 32767 như kiểu **int** quy định.

3. Các kiểu dữ liệu

*Các **long** và **short**:*

Kiểu **short** được kết hợp với kiểu dữ liệu cơ sở khi chiều dài yêu cầu ngắn hơn chiều dài bình thường của kiểu dữ liệu cơ sở.

Ví dụ: kiểu **int** chiếm 2 byte vùng nhớ trong khi đó kiểu **short int** chỉ chiếm 1 byte.

int a; // giá trị a nằm trong khoảng -32768 ... 32767

short int b; // giá trị b nằm trong khoảng -128 ... 127

Kiểu **long** được dùng khi chiều dài yêu cầu dài hơn chiều dài bình thường của kiểu dữ liệu cơ sở.

Ví dụ: kiểu **int** chiếm 2 byte trong khi kiểu **long int** chiếm 4 byte.

kiểu **double** chiếm 8 byte thì **long double** chiếm 16 byte.

4. Hằng số

Hằng số là đại lượng cố định trong chương trình.

Muốn sử dụng hằng ta phải khai báo trước với từ khóa ***const*** hoặc ***define***.

Cú pháp:

const <kiểu dữ liệu><Tên hằng> = <Giá trị hằng>;

Hoặc:

#define <Tên hằng> <Giá trị hằng>

Ví dụ:

```
const int a= 32767 ;
const float b = 3.14;
const int a= 3, b = 4, C = 5;
#define a 32767
#define b 3.14
```

5. Biểu thức

Biểu thức là một công thức tính toán, bao gồm 2 phần: toán tử và toán hạng. Toán tử là các phép toán, toán hạng có thể là hằng, biến hay hàm nào đó.

Có 2 loại toán tử: một ngôi (lấy đối số, tăng giảm một đơn vị) và 2 ngôi (cộng, trừ, nhân, chia, lấy dư và lũy thừa).

Ví dụ: 2 + 3*(++a) + sum(a, b) – 2^3

Có 3 loại biểu thức:

Biểu thức số học: trả về giá trị số

Biểu thức logic: trả về kết quả đúng, sai

Biểu thức quan hệ: liên quan đến các phép toán so sánh (>, <, !=, ==, ...).

5. Biểu thức

Ví dụ: toán tử 1 ngôi

```
#include <stdio.h>
void main()
{
    int a = 5, b = 10;
    int c = -a;
    printf("\nGia tri c = %d", c); //kết quả c = -5
    printf("\nXuat a++ = %d", a++); //kết quả 5
    printf("\nXuat ++b = %d", ++b); //kết quả 11
}
```

Ghi chú: biểu thức $x = x + 1$ tương đương với $++x$ hay $x++$;

5. Biểu thức

Ví dụ: toán tử 2 ngôi

```
#include <stdio.h>
void main()
{
    int a = 5, b = 10;
    int c = (a*b/2)%4;
    int d = a^3;
    printf("\nGia tri c = %d", c); //kết quả c = 1
    printf("\nGia tri d = %d", d); //kết quả 125 = 5*5*5
}
```

5. Biểu thức

❑ Biểu thức phẩy:

Mỗi câu lệnh trong ngôn ngữ lập trình C được kết thúc bằng dấu chấm phẩy, tuy nhiên trong một biểu thức của ngôn ngữ lập trình C có thể có nhiều câu lệnh được cách nhau bởi dấu phẩy.

Ví dụ:

```
x = a*b, q = x + y, k = q / z;
    int n, m;
    char ho[50], ten[50];
```

5. Biểu thức

❑ Biểu thức điều kiện:

Cú pháp:

<Tên biến> = <Biểu thức điều kiện> ? <Biểu thức 1> : <Biểu thức 2>

Ví dụ:

```
m = a > b ? a : b  /* m = max(a,b) */
```

6. Các phép toán

a. Toán tử số học: cộng, trừ, nhân, chia, lấy dư và lũy thừa lần lượt tương ứng với các ký hiệu +, -, *, /, %, ^.

Ghi chú:

- Phép toán % không dùng cho kiểu dữ liệu **float** hay **double**.
- Phép chia(/) thực hiện theo kiểu của các toán hạng dù là phép chia số nguyên hay số thực.

Ví dụ:

```
#include <stdio.h>
void main()
{   int x = 10, y = 3;
    float z1, z2;
    z1 = x/y;  z2 = (float)x/y;
    printf("\n%d chia %d du: %d", x, y, x%y); //kết quả du 1
    printf("\n z1 = %.2f", z1); // kết quả z1 = 3.00
    printf("\nz2 = %.3f", z2); // kết quả z2 = 3.333
}
```

6. Các phép toán

b. Toán tử quan hệ: bao gồm

== ,!=, >=, >, <=, <

c. Toán tử logic: bao gồm && (phép AND), || (phép OR), ! (phép NOT).

A && B: đúng <=> A: đúng và B: đúng

A || B: sai <=> A: sai và B: sai

Nếu A: đúng thì !A: sai

6. Các phép toán

d. Toán tử trên bit: bao gồm phép AND (&), OR(|), XOR(^), phép dịch trái (<<), phép dịch phải (>>) và phép đảo bit (~).

Cách thực hiện các phép toán trên bit:

1 & 1 = 1 1 | 1 = 1 1 ^ 1 = 0
1 & 0 = 0 1 | 0 = 1 1 ^ 0 = 1
0 & 1 = 0 0 | 1 = 1 0 ^ 1 = 1
0 & 0 = 0 0 | 0 = 0 0 ^ 0 = 0

- x << M nghĩa là dịch sang trái số nguyên x đi M bit, tương đương với $x*2^M$
- x >> M nghĩa là dịch sang phải số nguyên x đi M bit, tương đương với phép chia $x/2^M$ (chia lấy phần nguyên).

Ví dụ: Ta có thể thay phép tính x*80 bằng cách:
x << 6 + x << 4 vì $80 = 2^6 + 2^4$

6. Các phép toán

e. Toán tử tăng giảm 1:

– i = i + 1 có thể được viết thành: i++(tăng sau) hoặc ++i(tăng trước).

– i = i – 1 có thể được viết thành: i-- (giảm sau) hoặc --i (giảm trước).

Ví dụ: với i = 5 ; j = 10;

a/ i = ++ j ; i = j kết quả i = 11, j = 11

b/ i = j++ kết quả i = 10, j = 11

c/ j = --i + 2 kết quả i = 4, j = 6

d/ j = i-- + 2 kết quả i = 4, j = 7

6. Các phép toán

f. Toán tử gán: **<Tên biến> = <biểu thức>**

Có 3 loại phép gán khác nhau: phép gán đơn, phép gán kép và phép gán mở rộng.

Ví dụ:

Gán đơn:

i = 3;

i = i + 4 ;

Gán kép:

a= b = c = 5;

a= b + (c = 5);

Gán mở rộng:

x += y ⇔ x = x + y x -= y ⇔ x = x – y x *= y ⇔ x = x*y

x /= y ⇔ x = x / y x %= y ⇔ x = x % y x >>=y ⇔ x = x >> y

x <<=y ⇔ x = x << y x &= y ⇔ x = x & y x |= y ⇔ x = x | y

x ^= y ⇔ x = x ^ y

7. Độ ưu tiên của các phép toán

Thứ tự ưu tiên của các toán tử số học:

Đầu tiên ++, -- sau đó là *, /, % rồi mới đến +, -

Thứ tự ưu tiên của các toán tử quan hệ và logic:

Cao nhất: !

> >=<<=

== !=

&&

Thấp nhất: ||

7. Độ ưu tiên của các phép toán

Tổng kết về độ ưu tiên của các toán tử, với độ ưu tiên từ trên xuống dưới và từ trái qua phải.

Cao nhất	() [] ->
	sizeof() ! ~ ++ -- (Kiểu)* &
	* / %
	+ -
	<< >>
	< <= > >=
	== !=
	&
	^
	|

7. Độ ưu tiên của các phép toán

	&&
	||
	? :
	= += -= *= /= %= ^= | = << = >>
Thấp nhất	,

8. Câu lệnh

a. Câu lệnh đơn: là một câu lệnh không chứa các câu lệnh khác bên trong nó và kết thúc bằng một dấu chấm phẩy (;)

Ví dụ:

```
int x=5, y = 7;   //một câu lệnh đơn
x++;              //một câu lệnh đơn
x = x > y ? x : y ; //không là lệnh đơn
```

b. Câu lệnh phức: là một câu lệnh chứa câu lệnh khác bên trong nó hoặc một khối lệnh gồm nhiều câu lệnh như lệnh điều kiện (lệnh if), lệnh rẽ nhánh (lệnh switch), lệnh lặp (lệnh for, while, do … while).

Ví dụ:

```
x = x > y ? x : y ; //lệnh phức
```

Bài tập

1. Viết chương trình nhập vào 2 số thực. Tính và xuất kết quả tổng, hiệu, tích, thương của 2 số thực vừa nhập, kết quả lấy 2 số lẻ.
2. Viết chương trình đổi nhiệt độ từ đơn vị F (Ferarit) ra độ C (Celsius) theo công thức: C = 5/9 (F-32)
3. Viết chương trình tính giá trị F(x) và G(x), trong đó x là số nguyên nhập từ phím: F(x) = 5x2 + 6x + 1 và G(x) = 2x4 – 5x2 + 4x + 1
4. Viết chương trình tính giá trị của biểu thức, trong đó x là số nguyên nhập từ phím:

$$F(x) = \frac{1+x}{1-x} \qquad g(x) = \frac{3x^5 + 2x + \sqrt{x+1}}{5x^2 - 3}$$

Bài tập

5. Viết chương trình nhập vào chiều dài, chiều rộng của 1 hình chữ nhật. Tính và xuất kết quả chu vi, diện tích của hình chữ nhật trên.
6. Viết chương trình nhập vào bán kính của 1 hình tròn. Tính và xuất kết quả chu vi, diện tích của hình tròn trên.
7. Viết chương trình nhập vào chiều dài cạnh 1 hình vuông. Tính và xuất kết quả chu vi, diện tích, đường chéo của hình vuông trên.
8. Nhập vào 2 số nguyên a,b. Tìm số lớn nhất trong 2 số.
9. Nhập 3 số nguyên a, b, c. Xuất số lớn nhất và số nhỏ nhất của 3 số đó.

   Gợi ý: Sinh viên dùng biểu thức điều kiện.
10. Nhập vào 5 số nguyên. Tính trung bình cộng 5 số đó.


STOP
BLUEBUDDIES