Chương 01.3: Khai báo. Biểu thức. Khối lệnh

I. Các khai báo

II. Biểu thức

III. Khối lệnh

I. Các khai báo

1. Khai báo sử dụng thư viện hàm
2. Khai báo hằng
3. Khai báo biến

1. Khai báo sử dụng thư viện hàm

- Các trình biên dịch C++ có sẵn rất nhiều chương trình con (gọi là hàm), các hàm này để ở các thư viên chương trình con khác nhau. Muốn sử dụng hàm nào ta phải khai báo sử dụng thư viện chương trình chứa hàm đó.
- Cú pháp khai báo như sau:

```
#include<tên tệp header>
#include “tên tệp header”
```

Tên tệp header của thư viện chương trình có đuôi .h

Ví dụ: #include<iostream> //Khai báo sử dụng các chương trình vào/ra

2. Khai báo hằng

- Khai báo hằng là việc đặt tên cho các hằng
- Cú pháp khai báo hằng:

  #define Tên_hằng Giá_trị_của_hằng

  const kiểu tên_hằng=giá_trị;

Ví dụ: #define PI 3.141593

  const float PI=3.141593;

- Khai báo hằng có thể đặt bất kỳ đâu trong chương trình. Khi biên dịch chương trình, tất cả tên hằng được sử dụng sau dòng khai báo nó sẽ được thay bằng giá trị của tên hằng.

3. Khai báo biến

- Biến là tên của một ô nhớ trong bộ nhớ trong (RAM) dùng để chứa dữ liệu.
- Khai báo biến là đặt tên cho ô nhớ. Khai báo biến có thể để bất kỳ đâu trong chương trình. Vị trí khai báo của một biến sẽ quyết định phạm vi hoạt động của biến. Vấn đề này sẽ được nói kỹ hơn trong phần Khối lệnh.
- Cú pháp: Tên_kiểu_dl Tên_biến;

  Ví dụ: int a; //biến tên là a, có kiểu số nguyên int

  - Nếu có nhiều biến cùng kiểu thì có thể khai báo cùng nhau, giữa các tên biến phân tách nhau bởi dấu phẩy.

  Ví dụ: float a,b,c;

3. Khai báo biến (tiếp)

- Biến có kiểu nào thì chỉ chứa được giá trị của kiểu đó.
- Khi khai báo biến có thể khởi tạo giá trị ban đầu cho biến bằng đặt dấu bằng và một giá trị nào đó cách ngay sau tên biến.

  Ví dụ: int a,b=20,c,d=35;

II. Biểu thức

1. Biểu thức
2. Phép toán số học
3. Phép toán quan hệ và logic
4. Phép toán tăng giảm
5. Thứ tự ưu tiên của các phép toán
6. Các hàm số học
7. Câu lệnh gán và biểu thức gán
8. Biểu thức điều kiện
9. Chuyển đổi kiểu giá trị

1. Biểu thức

- Biểu thức là một sự kết hợp giữa các phép toán và các toán hạng để diễn đạt một công thức toán học nào đó, để có được một giá trị mới. Toán hạng có thể xem là một đại lượng có giá trị. Toán hạng có thể là hằng, biến, hàm.
- Khi viết biểu thức có thể dùng dấu ngoặc tròn để thể hiện đúng trình tự tính toán trong biểu thức.
- Mỗi biểu thức sẽ có một giá trị và nói chung cái gì có giá trị đều được xem là biểu thức.

1. Biểu thức (tiếp)

- Có hai loại biểu thức:
  - Biểu thức số: có giá trị là nguyên hoặc thực
  - Biểu thức logic: có giá trị là đúng (giá trị khác 0) hoặc sai (giá trị bằng 0)
- *Ví dụ:*

  (a+b+c)/2 (-b-sqrt(delta))/(2*a)

  (a+b) > 2*c

2. Phép toán số học

- Phép toán hai ngôi: + - * / %
  - % là phép lấy phần dư, ví dụ: 11%2 = 1
  - Phép chia hai số nguyên chỉ giữ lại phần nguyên

    Ví dụ: 11/2 = 5
- Phép toán một ngôi: dấu âm –

  Ví dụ -(a+b)
- Các phép toán số học tác động trên tất cả các kiểu dữ liệu cơ bản.

3. Phép toán so sánh và logic

- Các phép toán quan hệ và logic cho ta giá trị đúng (có giá trị bằng 1) hoặc sai (có giá trị bằng 0).
- Các phép toán quan hệ gồm có:

Phép toán	Ý nghĩa
>	Lớn hơn
>=	Lớn hơn hoặc bằng
<	Nhỏ hơn
<=	Nhỏ hơn hoặc bằng
= =	Bằng (hai dấu bằng sát nhau)
!=	Khác nhau

3. Phép toán so sánh và logic (tiếp)

✧ Các phép toán logic gồm có:

Phép toán	Ý nghĩa
!	Phủ định (NOT)
&&	Và (AND)
||	Hoặc (OR)

4. Phép toán tăng giảm

- C++ có hai phép toán một ngôi để tăng và giảm giá trị **của các biến** (có kiểu nguyên hoặc thực). Toán tử tăng ++ cộng 1 vào toán hạng của nó, toán tử giảm -- trừ toán hạng của nó đi 1.

  Ví dụ: giả sử biến n đang có giá trị là 8, sau phép tính ++n làm cho n có giá trị là 9, sau phép tính --n làm cho n có giá trị là 7.

- Phép toán ++ và -- có thể đứng trước hoặc sau toán hạng. Nếu đứng trước thì toán hạng của nó sẽ được tăng/giảm trước khi nó được sử dụng, nếu đứng sau thì toán hạng của nó sẽ được tăng/giảm sau khi nó được sử dụng.

5. Thứ tự ưu tiên của các phép toán

- ✧ Khi trong một biểu thức có chứa nhiều phép toán thì các phép toán được thực hiện theo thứ tự ưu tiên: Các phép toán có mức ưu tiên cao thực hiện trước, các phép toán cùng mức ưu tiên được thực hiện từ trái qua phải hoặc từ phải qua trái.
- ✧ Bảng thứ tự ưu tiên các phép toán: Các phép toán cùng loại cùng mức ưu tiên. Các phép toán loại 1 có mức ưu tiên cao nhất, rồi đến các phép toán loại 2, 3,… Các phép toán loại 2 (phép toán một ngôi), 14 (phép toán điều kiện) và 15 (phép toán gán) kết hợp từ phải qua trái, các phép toán còn lại kết hợp từ trái qua phải.

5. Thứ tự ưu tiên của các phép toán (tiếp)

TT	Loại phép toán	Phép toán	Ý nghĩa
1	Cao nhất	( ) [ ] -> . ::	Lời gọi hàm, dấu ngoặc Truy nhập phần tử mảng Truy nhập gián tiếp Truy nhập trực tiếp Truy nhập tên miền
2	Phép toán 1 ngôi	! ~ + - ++ --	Phủ định (NOT) Đảo bit Dấu dương Dấu âm Toán tử tăng Toán tử giảm

5. Thứ tự ưu tiên của các phép toán (*tiếp*)

TT	Loại phép toán	Phép toán	Ý nghĩa
2	Phép toán 1 ngôi	& * sizeof new delete (Kiểu dl)	Lấy địa chỉ biến Truy nhập qua con trỏ Cho kích thước toán hạng Cấp phát bộ nhớ động Giải phóng bộ nhớ Phép ép kiểu dữ liệu
3	Phép toán truy nhập thành viên	.* ->*	
4	Phép toán nhân	* / %	Nhân Chia Chia lấy phần dư

5. Thứ tự ưu tiên của các phép toán (*tiếp*)

TT	Loại phép toán	Phép toán	Ý nghĩa
5	Phép toán cộng	+ -	Cộng Trừ
6	Phép toán dịch bit	>> <<	Dịch phải Dịch trái
7	Phép toán quan hệ	< <= > >=	Nhỏ hơn Nhỏ hơn hoặc bằng Lớn hơn Lớn hơn hoặc bằng
8	Phép toán so sánh bằng	== !=	Bằng Khác nhau

5. Thứ tự ưu tiên của các phép toán (*tiếp*)

TT	Loại phép toán	Phép toán	Ý nghĩa
9	Phép toán về bit	&	Phép AND bit
10	Phép toán về bit	^	Phép XOR bit
11	Phép toán về bit	|	Phép OR bit
12	Phép toán logic	**&&**	Phép AND logic
13	Phép toán logic	||	Phép OR logic
14	Phép toán điều kiện	**? :**	Ví dụ: a ? x : y //nếu a đúng thì bằng x, còn không bằng y

5. Thứ tự ưu tiên của các phép toán (*tiếp*)

TT	Loại phép toán	Phép toán	Ý nghĩa
15	Phép toán gán	=	Phép gán đơn giản
		*=	Phép gán nhân
		/=	Phép gán chia
		%=	Phép gán chia lấy phần dư
		+=	Phép gán cộng
		-=	Phép gán trừ
		&=	Phép gán AND bit
		^=	Phép gán XOR bit
		|=	Phép gán OR bit
		<<=	Phép gán dịch trái bit
		>>=	Phép gán dịch phải bit
16	Dấu phẩy	,	

6. Các hàm số học cơ bản

Các hàm số học nằm trong thư viện chương trình math, muốn sử dụng các hàm này ta phải khai báo: #include<math.h>

Dưới đây là một số hàm số học hay dùng:

Tên hàm	Ý nghĩa
cos(x)	Cho cos(x)
sin(x)	Cho sin(x)
acos(x)	Cho arccos(x)
asin(x)	Cho arcsin(x)

6. Các hàm số học cơ bản (tiếp)

Tên hàm	Ý nghĩa
tan(x)	Cho tgx
fabs(x)	Cho |x|
exp(x)	e^x
log(x)	Cho lnx
log10(x)	Cho $\log_{10}x$
pow(y,x)	Cho y^x
sqrt(x)	Cho căn bậc 2 của x

7. Câu lệnh gán và biểu thức gán

✧ Câu lệnh gán

- Để đưa giá trị vào các biến tại thời điểm lập trình ta sử dụng lệnh gán. Có lệnh gán đơn giản và lệnh gán phức hợp.
- Lệnh gán đơn giản có dạng: Biến = Biểu thức;

  Lệnh gán này đưa giá trị của biểu thức bên phải vào biến bên trái. Vế trái của phép gán chỉ có thể là biến và chỉ một mà thôi.

  Ví dụ: a = 2*x*x + 3*x + 1;

7. Câu lệnh gán và biểu thức gán (tiếp)

✧ Câu lệnh gán

- Lệnh gán phức hợp có dạng:

  Biến Phép_toán= Biểu thức;

  Phép toán để ngay trước dấu bằng, có thể là các phép toán số học hoặc các phép toán về bit.

  Ví dụ: a += 2;

  Lệnh gán này đem giá trị của biến kết hợp với giá trị của biểu thức theo phép toán rồi đưa kết quả vào biến, tức là thực hiện phép toán trước rồi mới gán.

  a *= 5; //lệnh này tương đương với lệnh a = a*5;

7. Câu lệnh gán và biểu thức gán (tiếp)

✧ Biểu thức gán

- Biểu thức gán là biểu thức có dạng:

  v = e

  (*Sau biểu thức gán không có dấu chấm phẩy*)

  trong đó v là một biến, e là một biểu thức.

- Biểu thức gán thực hiện gán e vào v. Giá trị của biểu thức gán là giá trị của biểu thức e, kiểu của biểu thức gán là kiểu của biến v. Biểu thức gán được sử dụng như bất kỳ biểu thức khác, chẳng hạn đem gán giá trị của nó vào biến.

  Ví dụ: sau lệnh a = b = 5; thì a và b sẽ bằng 5 vì biểu thức gán đưa 5 vào b còn lệnh gán đưa giá trị của biểu thức gán b=5 vào a.

8. Biểu thức điều kiện

- Biểu thức điều kiện là biểu thức có dạng:

  e1 ? e2 : e3

  trong đó e1, e2, e3 là các biểu thức nào đó.
- Giá trị của biểu thức điều kiện bằng giá trị của e2 nếu e1 đúng (có giá trị khác 0) và bằng giá trị của e3 nếu e1 sai (có giá trị bằng 0).
- Biểu thức điều kiện thực sự là một biểu thức, bởi vậy ta có thể sử dụng nó như bất kỳ một biểu thức nào khác.

  Ví dụ: biểu thức (a > b) ? a : b sẽ cho giá trị a nếu a lớn hơn b, còn không cho giá trị b.

9. Chuyển đổi kiểu giá trị

- Việc chuyển đổi kiểu giá trị thường diễn một cách tự động trong hai trường hợp sau:
  - Khi biểu thức có các toán hạng khác kiểu
  - Khi gán một giá trị kiểu này cho một biến kiểu khác.
- Chuyển đổi kiểu trong biểu thức: Khi hai toán hạng trong một phép toán có kiểu khác nhau thì kiểu thấp hơn sẽ được nâng thành kiểu cao hơn. Kết quả thu được một giá trị có kiểu cao hơn.

  Ví dụ: giữa int và long thì int chuyển thành long

  giữa int và float thì int chuyển thành float

9. Chuyển đổi kiểu giá trị (tiếp)

- Chuyển đổi kiểu khi gán: Giá trị của vế phải được chuyển sang kiểu của vế trái.
- Ta cũng có thể thực hiện chuyển đổi kiểu theo ý muốn bằng toán tử ép kiểu, có dạng: (Tên kiểu muốn ép) Biểu_thức

  Ví dụ: (int) a (float)(a+b)

III. Khối lệnh

- Nhiều lệnh đặt giữa dấu ngoặc { và } tạo thành một khối lệnh.

```
{
   a=2;
   b=3;
   cout<<a<<' '<<b;
}
```

- C++ coi một khối lệnh như một câu lệnh riêng lẻ. Bởi vậy chỗ nào viết được một câu lệnh thì chỗ đó viết cũng đặt được một khối lệnh. Sau dấu ngoặc } của khối lệnh không có dấu chấm phẩy.

III. Khối lệnh (tiếp)

- Bên trong một khối lệnh có thể chứa các khối lệnh khác. Sự lồng nhau này không bị hạn chế. Lưu ý rằng thân của một hàm cũng là một khối lệnh, đó là khối lệnh chứa các khối lệnh bên trong nó và không khối lệnh nào chứa nó.
- Các biến không chỉ khai báo ở đầu một hàm mà có thể khai báo ở đầu một khối lệnh. Biến được khai báo trong một khối lệnh thì chỉ có phạm vi hoạt động trong khối lệnh đó. Khi máy bắt đầu thực hiện khối lệnh thì các biến khai báo bên trong nó mới được hình thành và được cấp phát bộ nhớ. Các biến này chỉ tồn tại trong thời gian máy làm việc bên trong khối lệnh và chúng sẽ lập tức biến mất ngay sau khi máy ra khỏi khối lệnh.

III. Khối lệnh (tiếp)

- Nếu bên trong một khối lệnh ta khai báo một biến có tên là a thì tên biến này không ảnh hưởng tới một biến khác cũng có tên là a được dùng ở đâu đó ngoài khối lệnh.
- Nếu một biến được khai báo ở ngoài và trước một khối lệnh mà không trùng tên với các biến khai báo bên trong khối lệnh này thì biến đó có thể sử dụng cả bên ngoài và bên trong khối lệnh.