28TECH
Become A Better Developer

PROGRAM IN C++

Hướng dẫn

Các bạn đọc slide này kèm với thực hành luôn bài tập phần Làm quen OJ và Cấu trúc rẽ nhánh trên website OJ tại địa chỉ :
http://oj.28tech.com.vn/problems/?category=2&point_start=&point_end=&order=code
http://oj.28tech.com.vn/problems/?category=3&point_start=&point_end=&order=code

NỘI DUNG BÀI HỌC

01 Cấu trúc chương trình trong C++

...

02 Kiểu dữ liệu

...

03 Biến, cin, cout

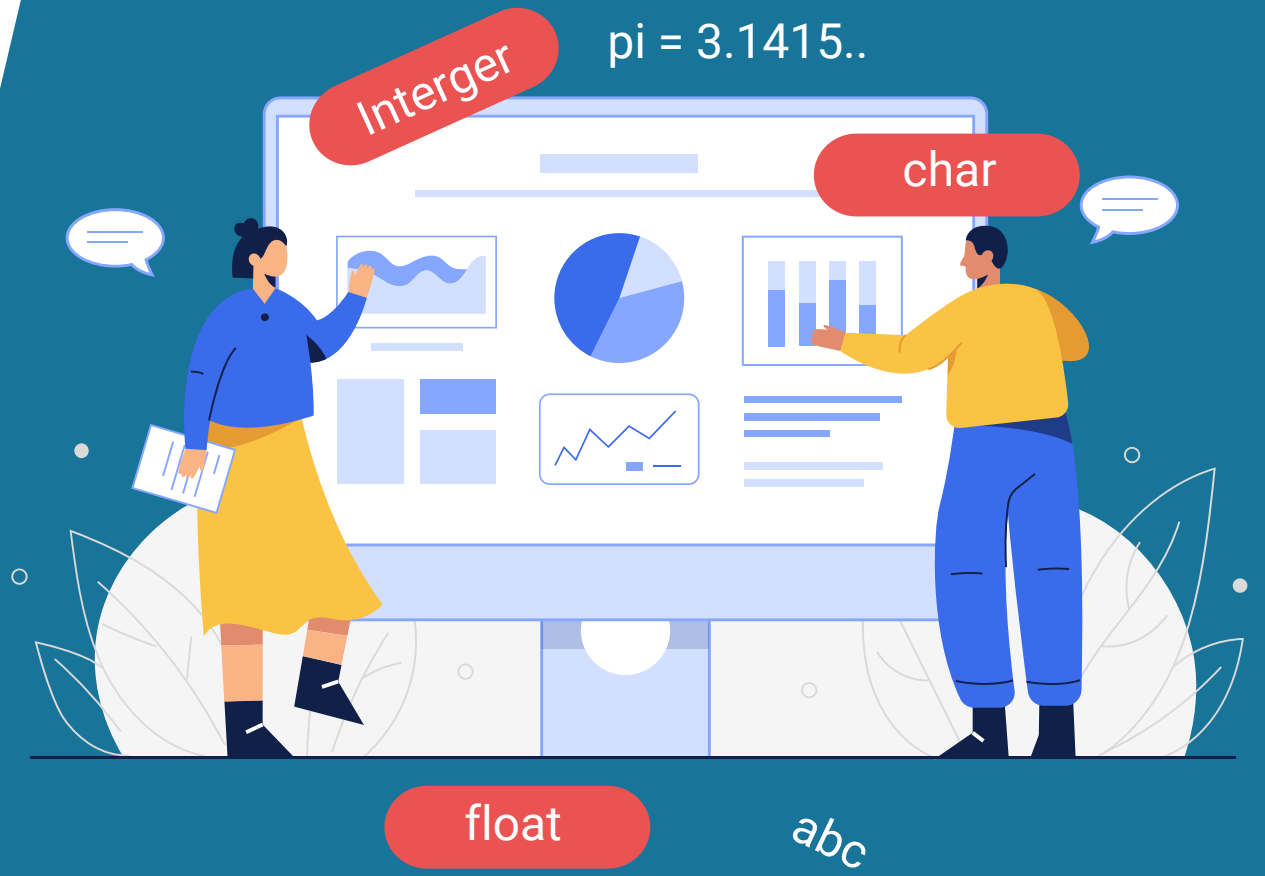
...

04 Chú thích, Ép kiểu trong C++

05 Toán tử trong C++

06 Thư viện và các hàm toán học

1. CẤU TRÚC CỦA CHƯƠNG TRÌNH TRONG C++



1. Cấu trúc chương trình trong C++

Khuôn mẫu chung của một chương trình

INPUT → 

→ OUTPUT

1. Cấu trúc chương trình trong C++

Chương trình C++ đầu tiên

```
#include <iostream>

using namespace std;

int main () {
    cout << "Hello world !" << endl;
    return 0;
}
```

Khai báo thư viện cần thiết

Thêm namespace

Hàm main: là nơi bắt đầu thực thi của chương trình

1. Cấu trúc chương trình trong C++

Những phần chính của một chương trình C++:

/01 Thư viện mà chương trình sử dụng.

/02 Namespace.

/03 Chương trình chính với mã nguồn.

1. Cấu trúc chương trình trong C++

▶▶▶ Những chú ý khi viết chương trình C++

Các câu lệnh kết thúc bằng dấu " ; "

Không hợp lệ

```
cout << “Hello 28tech” <<
```

Hợp lệ

```
cout << “Hello 28tech” << ;
```

1. Cấu trúc chương trình trong C++

- Các câu lệnh kết thúc bằng dấu " ; "
- Luôn thụt lề các câu lệnh so với hàm main

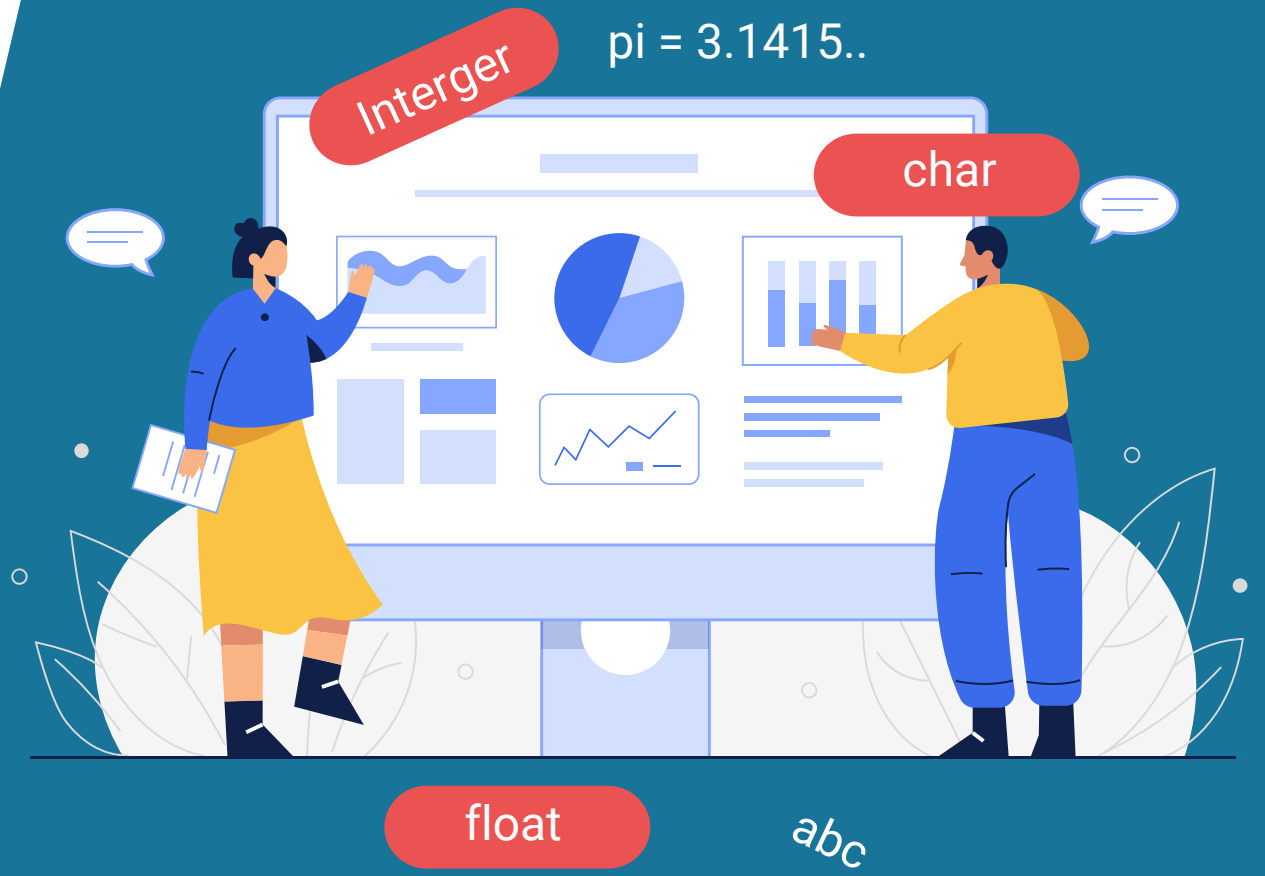
Không chấp nhận

```
int main () {
cout << "Hello world !" << endl;
return 0;
}
```

Chấp nhận

```
int main () {
    cout << "Hello world !" << endl;
    return 0;
}
```

2. KIỂU DỮ LIỆU (DATA TYPE)



2. Kiểu Dữ Liệu

KIỂU DỮ LIỆU SỐ NGUYÊN

1 byte = 8 bit

Đối với số nguyên ta chia làm số nguyên không dấu và số nguyên có dấu, từ số byte lưu trữ ta có thể suy ra số bit cần để biểu diễn số nguyên đó, quy tắc xác định giá trị của 1 số nguyên.

Giả sử số nguyên có K bit

-Số nguyên có dấu : -2^{K-1} tới $2^{K-1} - 1$

-Số nguyên không dấu : 0 tới $2^{K} - 1$

2. Kiểu Dữ Liệu

KIỂU DỮ LIỆU SỐ NGUYÊN

Kiểu dữ liệu	Ý nghĩa	Số Byte	Giải giá trị có thể lưu
short	Số nguyên có dấu	2 byte	-32768 đến 32767
unsigned short	Số nguyên không dấu	2 byte	0 đến 65535
int = long	Số nguyên có dấu	4 byte	-2147483648 đến 2147483647
unsigned int	Số nguyên không dấu	4 byte	0 đến 4294967295
long long	Số nguyên	8 byte	-9223372036854775808 đến 9223372036854775807
unsigned long long	Số nguyên không dấu	8 byte	0 đến 2^64 - 1

2. Kiểu Dữ Liệu

Lưu ý về số nguyên

1. Nếu lưu số nguyên cỡ từ âm 2 tỷ tới 2 tỷ thì dung int là đủ, còn ngoài khoảng đó tức là nhỏ hơn âm 2 tỷ hoặc lớn hơn 2 tỷ thì dung long long
2. Nếu số cần lưu vượt giới hạn của số long long thì cần dùng chuỗi (sẽ học sau)
3. Số long long có thể lưu được số int nhưng ngược lại thì không, tuy nhiên số long long sẽ chậm hơn int trong lúc nhập dữ liệu, tính toán nên việc luôn dùng long long là không được khuyến khích.

2. Kiểu Dữ Liệu

KIỂU DỮ LIỆU SỐ THỰC

Kiểu dữ liệu	Ý nghĩa	Số Byte	Giải giá trị có thể lưu
float	Số thực độ chính xác đơn	4 byte	1.17549e-038 đến 3.40282e+038 Độ chính xác : 7 chữ số sau dấu phẩy
double	Số thực độ chính xác kép	8 byte	2.22507e-308 đến 1.79769e+308 Độ chính xác : 15 chữ số sau dấu phẩy

Ưu tiên sử dụng double vì double có độ chính xác cao hơn float

2. Kiểu Dữ Liệu

Lưu ý về số thực

1. Ưu tiên sử dụng double thay cho float để có độ chính xác cao hơn, có kiểu dữ liệu là long double độ chính xác cao hơn double
2. Không sử dụng số thực để lưu bài toán có dữ liệu là số nguyên sẽ bị sai số khi tính toán và việc tính toán cũng chậm hơn số nguyên
3. Số thực khi lưu trữ sẽ không thể lưu chính xác 100% mà luôn có sự sai số.

2. Kiểu Dữ Liệu

KIỂU DỮ LIỆU ĐÚNG SAI

Kiểu dữ liệu	Ý nghĩa	Số Byte	Giải giá trị có thể lưu
bool	Kiểu dữ liệu luận lý	1 byte	true hoặc false true : Đúng false : Sai

2. Kiểu Dữ Liệu

KIỂU DỮ LIỆU KÝ TỰ

Kiểu dữ liệu	Ý nghĩa	Số Byte	Giải giá trị có thể lưu
char	Kiểu dữ liệu ký tự	1 byte	-128 tới 127

2. Kiểu Dữ Liệu

Tổng quan về kiểu dữ liệu:

/01 Kiểu số nguyên có int và long long.

>> Sử dụng long long cho các kết quả là số lớn

/02 Kiểu số thực có float và double.

>> Sử dụng double vì có độ chính xác cao

/03 Kiểu đúng sai có bool.

/04 Kiểu ký tự có char.

3. BIẾN (VARIABLE)

3. Biến, Cin và Cout

BIẾN (VARIABLE)

Khái niệm

▶▶▶ Biến được sử dụng để lưu các giá trị trong quá trình tính toán của chương trình. Tùy theo kiểu dữ liệu của biến, một ô nhớ trong bộ nhớ sẽ được cấp phát để lưu trữ giá trị của biến này.

Cú pháp

▶▶▶ [Kiểu dữ liệu] [Tên biến];

Ví dụ: int x; long long b; char ki_tu; bool check; double dienTich; ...

3. Biến, Cin và Cout

BIẾN (VARIABLE)

Quy tắc đặt tên biến:	Ví dụ cách đặt sai:
01 Không đặt tên biến bắt đầu bằng chữ số	1dientich, 2chuvi, 222bankinh, 9a,...
02 Tên biến không được chứa dấu cách và các kí tự đặc biệt	ban kinh, dien#tich, chu@vi,...
03 Tên biến không được trùng với tên từ khóa trong C++	int, main, for, while, ...
04 Tên biến trong C++ là phân biệt hoa thường(case sensitive)	banKinh và BanKinh là 2 biến khác nhau
05 Không được đặt 2 biến có cùng tên trong cùng một phạm vi	int a; float a;

MỘT SỐ VÍ DỤ

3. Biến, Cin và Cout

Khai báo biến

```
#include <iostream>
using namespace std;

int main(){
    int a;   // biến a có kiểu int
    int x, y, z; // khai báo 3 biến cùng kiểu int
    long long dien_tich;  // biến dien_tich có kiểu ll
    bool check;  // biến check kiểu đúng sai
    char kitu;  // biến kitu kiểu char
}
```

CIN VÀ COUT

3. Biến, Cin và Cout

Hiển thị giá trị của biến ra màn hình

Cú pháp: cout << var ;

Hiển thị giá trị của biến số thực var với độ chính xác k số sau dấu phẩy:

Cú pháp:

cout << fixed << setprecision(k) << var;

Phải khai báo thư viện iomanip !

```
#include <iostream>
#include <iomanip>
using namespace std;

int main(){
    int a = 28;   // biến a có kiểu int và có giá trị là 28
    cout << a << endl;  //xuat biến a ra màn hình
    int x = 100, int y = 200; //khởi tạo nhiều biến
    cout << x << ' ' << y << ' ' << endl; //xuất cùng lúc
    double res = 3.14;
    cout << fixed << setprecision(5) << res << endl;
   //output: 3.14000
}
```

3. Biến, Cin và Cout

Nhập giá trị cho biến từ bàn phím

Cú pháp: cin >> var ;

Chú ý : Khi nhập xong giá trị các bạn ấn enter để cin bắt đầu nhập giá trị cho biến, trong trường hợp bạn nhập giá trị cho nhiều số, có thể nhập từng số rồi enter hoặc nhập 1 lúc nhiều số rồi ấn enter 1 lần.

```
#include <iostream>
using namespace std;

int main(){
    int a;   // biến a có kiểu int
    cin >> a; //nhập giá trị cho a
    cout << a << endl; //xuất ra giá trị vừa nhập
    int x, y, z; // khai báo 3 biến cùng kiểu int
    cin >> x >> y >> z;
    cout << x << " " << y << " " << z << " " << endl;
}
```

Bài tập áp dụng : http://oj.28tech.com.vn/problem/oj02

4.ÉP KIỂU & CHÚ THÍCH TRONG C++

// /* */

CHÚ THÍCH

4. Ép kiểu và chú thích

Khái niệm

Chú thích (Comment) là một giải pháp bổ sung thông tin vào code của bạn, nhằm làm rõ nội dung, giải thích câu lệnh, mục đích của code...

Giúp người đọc code có thể dễ nắm bắt nội dung code và thuận lợi trong việc bảo trì code

Các chú thích sẽ không được coi là câu lệnh và sẽ được loại bỏ khi chương trình thực thi

CHÚ THÍCH

4. Ép kiểu và chú thích

Cách chú thích

Để chú thích trên 1 dòng ta dùng //

Ví dụ :

```
//Đây là chú thích
//Chú thích giúp code rõ ràng hơn
```

Để chú thích trên nhiều dòng ta dùng /* ND cần chú thích */

```
/*
Đây là
chú thích trên nhiều dòng
*/
```

ÉP KIỂU DỮ LIỆU (TYPE CASTING)

4. Ép kiểu và chú thích

- Ép kiểu là hình thức gán giá trị của 1 biến, giá trị thuộc kiểu dữ liệu này cho biến thuộc kiểu dữ liệu khác, ví dụ gán giá trị số thực vào biến int, gán biến kiểu char vào số int...

-Có 2 kiểu ép kiểu là

+Ép kiểu ngầm định (Implicit conversion) hay ép kiểu tự động, trong cách ép kiểu này bạn chỉ cần gán mà không cần bổ sung thêm chi tiết nào cả

+Ép kiểu rõ ràng (Explicit convertion) bạn cần chỉ rõ kiểu dữ liệu cần ép sang

Cú pháp : (type) expression

4. Ép kiểu và chú thích

Một số ví dụ về ép kiểu tự động

Biểu thức	Giải thích
1. int n = 100; long long m = n	m sẽ lưu được giá trị của n là 100
2. int n = 3.5	n = 3 vì chỉ có thể lưu số nguyên
3. int n = true, m = false	n = 1, m = 0
4. bool x = 3, y = 0	x = true, y = false
5. int n = ‘A’	n = 65, mã ASCII của ký tự A
6. int n = 5; float m = n;	m = 5.0
7. int x = 3 + 2.5	x = 5

4. Ép kiểu và chú thích

Một số ví dụ về ép kiểu rõ ràng

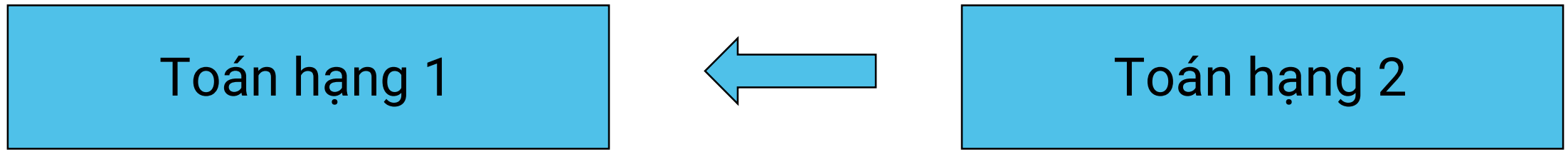
Biểu thức	Giải thích
1. int n = (int)3.2	n = 3
2. long long m = (long long)100	m sẽ là số 100 ở kiểu long long
3. int n = (int)10.5 / 3	n = 3
4. int n = (int)true	n = 1
5. int n = (int)false	n = 0
6. int n = 10; double k = (double)n	k = 10.0
7. double k = (double)'A'	k = 65.0

5. TOÁN TỬ (OPERATOR)

TOÁN TỬ GÁN: (ASSIGNMENT OPERATOR)

5. Toán tử

Cú pháp: [Toán hạng 1] = [Toán hạng 2]



Ý nghĩa:Gán giá trị của toán hạng 2 cho toán hạng 1.

VÍ DỤ:

```
ban_kinh = 100;   // Gán giá trị 100 cho biến ban_kinh
Chu_vi = 1000;   // Gán giá trị của chuvi là 1000
```

5. Toán tử

TOÁN TỬ TOÁN HỌC: (ARITHMETIC OPERATOR)

Toán tử	Ý nghĩa	Ví dụ
+	Cộng 2 toán hạng	X = Y + Z X = 100 + 200; // 300
-	Trừ 2 toán hạng	X = Y - Z X = 200 - 100; // 100
*	Nhân 2 toán hạng	X = Y * Z X = 10 * 50; // 500
/	Chia 2 toán hạng	X = Y / Z X = 300 / 200; // 1
%	Chia dư 2 toán hạng	X = Y % Z X = 300 % 200; // 100

5. Toán tử

TOÁN TỬ TOÁN HỌC: (ARITHMETIC OPERATOR)

Toán tử	Ý nghĩa	Ví dụ
+=	Toán tử gán cộng	X = 100; X += 10 X = 110
-=	Toán tử gán trừ	X = 100; X -= 10 X = 90
*=	Toán tử gán nhân	X = 100; X *= 2 X = 200
/=	Toán tử gán chia	X = 100; X /= 5 X = 20
%=	Toán tử gán chia dư	X = 100; X %= 9 X = 1

5. Toán tử

Lưu ý về toán tử toán học

1. Nếu biểu thức của bạn toàn là số nguyên kết quả sẽ luôn là số nguyên
2. Nếu biểu thức của bạn chứa ít nhất 1 số thực kết quả sẽ là số thực
3. Nếu biểu thức của bạn toàn số nguyên int thì kết quả cũng sẽ là kiểu int dù cho biểu thức có thể lớn hơn giới hạn của số
4. Nếu biểu thức của bạn toàn số nguyên nhưng có ít nhất 1 số long long thì kết quả sẽ lưu ở long long

TOÁN TỬ TOÁN HỌC: (ARITHMETIC OPERATOR)

5. Toán tử

Chú ý 1: Nếu bạn chia 2 số nguyên (int, long long) cho nhau thì phép chia ở trên sẽ là phép chia nguyên, tức là nó chỉ lấy phần nguyên và bỏ phần thập phân ở thương

- Nếu muốn kết quả ở số thập phân thì ít nhất 1 trong 2 số phải ở kiểu số thực và biến thương phải ở dạng số thực

+) Ép kiểu

+) Nhân thêm 1 số thực khi tính toán

```
int a = 100, b = 30;
float thuong = (float) a / b;
cout << fixed << setprecision(2) <<
thuong << endl; //3.33
```

Bài tập áp dụng : http://oj.28tech.com.vn/problem/ifelse08

TOÁN TỬ TOÁN HỌC: (ARITHMETIC OPERATOR)

5. Toán tử

Chú ý 1: Nếu bạn chia 2 số nguyên (int, long long) cho nhau thì phép chia ở trên sẽ là phép chia nguyên, tức là nó chỉ lấy phần nguyên và bỏ phần thập phân ở thương

- Nếu muốn kết quả ở số thập phân thì ít nhất 1 trong 2 số phải ở kiểu số thực và biến thương phải ở dạng số thực

+) Ép kiểu

+) Nhân thêm 1 số thực khi tính toán

```
int a = 100, b = 30;
float thuong = 1.0 * a / b;
cout << fixed << setprecision(2) <<
thuong << endl; //3.33
```

Bài tập áp dụng : http://oj.28tech.com.vn/problem/ifelse02

TOÁN TỬ TOÁN HỌC: (ARITHMETIC OPERATOR)

5. Toán tử

Chú ý 2: Nếu bạn nhân 2 số nguyên int với nhau mà kết quả của tích vượt giới hạn lưu của số int thì kết quả sẽ bị tràn, ngay cả khi bạn sử dụng biến long long để lưu biến tích. Việc cần xử lý ở đây là can thiệp vào phép nhân

Cách xử lí sai

```
int a = 1000000, b = 1000000;
long long tich = a * b; //sai
long long tich = (long long) (a  * b);
 // vẫn sai
cout << tich << endl; //tràn số
```

Cách xử đúng

```
int a = 1000000, b = 1000000;
// ép 1 trong 2 số thành ll
long long tich = (long long) a  * b;
long long tich = 1ll * a * b; //1ll ?
cout << tich << endl; //10^12
```

5. Toán tử

Một số chú ý về toán tử toán học

Chú ý	Câu lệnh, ví dụ
1. Lấy ra chữ số hàng đơn vị của số nguyên N	N % 10, Ví dụ N = 1234 thì N % 10 = 4
2. Giảm N đi 10 lần (xóa chữ số hàng đơn vị)	N = N / 10, Ví dụ N = 1234 thì N / 10 = 123
3. Khi chia dư cho K thì số dư sẽ từ 0 tới K -1	Chia dư cho 15 thì số dư sẽ từ 0 tới 14
4. Số lượng số nguyên từ a tới b	b – a + 1 số
5. Số lượng số nguyên dương chia hết cho k và <=n	n / k số
6. Giai thừa của N	N! = 1.2.3....N, 0! = 1
7. Tìm số lớn nhất <= N mà chia hết cho K	N / K * K, ví dụ 23 / 5 * 5 = 20
8. Thêm chữ số x vào cuối số N	N = N * 10 + x

TOÁN TỬ SO SÁNH: (COMPARISON OPERATOR)

5. Toán tử

Khi bạn sử dụng các toán tử so sánh để so sánh 2 toán hạng thì kết quả của phép so sánh sẽ trả về đúng hoặc sai.

Toán tử	Ý nghĩa	Ví dụ
>	Lớn hơn	10 > 50 : false
>=	Lớn hơn hoặc bằng	20>=10 : true
<	Nhỏ hơn	10 < 50 : true
<=	Nhỏ hơn hoặc bằng	20 <= 20 : true
!=	So sánh khác	10 != 20 : true
==	So sánh bằng	10 == 10 : true

5. Toán tử

TOÁN TỬ SO SÁNH: (COMPARISON OPERATOR)

VÍ DỤ:

Ví dụ	Kết quả
10 == 20	Sai
10 <= 20	Đúng
10 == 10	Đúng
50 != 50	Sai
100 <= 100	Đúng

TOÁN TỬ LOGIC: (LOGICAL OPERATOR)

5. Toán tử

Trong trường hợp bạn muốn kết hợp nhiều phép so sánh lại với nhau, ta sử dụng 3 cổng logic cơ bản của máy tính là AND, OR, NOT

Toán tử	Ký hiệu	Ví dụ
AND (VÀ)	&&	(x >= 10) && (x <= 50)
OR (HOẶC)	||	(a >= 10) || (a % 2 == 0)
NOT (PHỦ ĐỊNH)	!	!(a <= 10)

Để tính toán giá trị cuối cùng của biểu thức các bạn xác định giá trị của từng mệnh đề thành phần, sau đó áp dụng bảng chân lý của các cổng logic để tìm giá trị của biểu thức ban đầu.

5. Toán tử

TOÁN TỬ LOGIC: (LOGICAL OPERATOR)

Bảng chân lý của cổng AND

A	B	A && B
0	0	0
0	1	0
1	0	0
1	1	1

Chú ý: Cổng AND chỉ cho kết quả đúng khi mọi mệnh đề thành phần đều có giá trị đúng, sai trong các trường hợp còn lại.

5. Toán tử

Một số ví dụ về cổng AND

Ví dụ	Kết quả
1. (10 <= 10) && (5 != 10)	True && True = True
2. (5 == 5) && (3 >= 3) && (10 != 10)	True && True && False = False
3. n = 30, (n >= 10) && (n <= 50)	True && True = True
4. int n = 10, m = 3, (n / m == 3) && (n + m == 13)	True && True = True
5. int n = 35, k = 5, (n % k == 0) && (n / 10 == 3)	True && True = True
6. int n = 10, (n % 2 == 0) && (n % 3 == 0)	True && False = False
7. int n = 32, (n % 10 == 2) && (n % 4 != 0)	True && False = False

5. Toán tử

TOÁN TỬ LOGIC: (LOGICAL OPERATOR)

Bảng chân lý của cổng OR

A	B	A || B
0	0	0
0	1	1
1	0	1
1	1	1

Chú ý: Cổng OR chỉ cho kết quả sai khi mọi mệnh đề thành phần đều có giá trị sai, đúng khi chỉ cần ít nhất 1 mệnh đề thành phần có giá trị đúng

5. Toán tử

Một số ví dụ về cổng OR

Ví dụ	Kết quả
1. (10 <= 10) || (5 == 10)	True || False = False
2. (5 != 5) || ((3 >= 3) && (10 != 10))	False || (True && True) = False || True = True
3. n = 30, (n < 10) || (n > 50)	False || False = False
4. n = 10, m = 3, (n / m == 3) || (n % 2 == 5)	True || False = True
5. n = 35 (n >= 10) && ((n % 10 == 2) || (n % 10 ==5))	True && (False || True) = True && True = True
6. n = 10, (n % 2 == 0) || (n % 3 == 0)	True || False = True
7. n = 32, (n % 10 == 2) || (n % 4 != 8)	True || True = True

5. Toán tử

TOÁN TỬ LOGIC: (LOGICAL OPERATOR)

Bảng chân lý của cổng NOT

A	! A
0	1
1	0

TOÁN TỬ TĂNG GIẢM:

5. Toán tử (INCREMENT, DECREMENT OPERATOR)

Để tăng hoặc giảm giá trị của một biến đi 1 đơn vị ta có thể sử dụng toán tử tăng, giảm này sẽ thuật tiện hơn.

Toán tử	Ý nghĩa	Ví dụ
++	Tăng trước 1 đơn vị	++a
++	Tăng sau 1 đơn vị	a++
--	Giảm trước 1 đơn vị	--a
--	Giảm sau 1 đơn vị	a--

TOÁN TỬ TĂNG GIẢM:

5. Toán tử (INCREMENT, DECREMENT OPERATOR)

VÍ DỤ:

Ví dụ	Kết quả	Giải thích

int a = 100;
int b = a++;
cout << a << " " << b << endl;
	101 100	Đây là tăng sau, tức ban đầu câu lệnh sẽ gán giá trị của a là 100 cho b, sau đó mới tăng giá trị của a lên 101

int a = 100;
int b = ++a;
cout << a << " " << b << endl;
	101 101	Đây là tăng trước, giá trị của a sẽ được tăng ngay lập tức lên 101, sau đó mới lấy giá trị đó và gán cho b

5. Toán tử

TOÁN TỬ 3 NGÔI

Cú pháp

▶▶▶[Biểu thức so sánh] ? [Giá trị trả về khi biểu thức đúng] : [Giá trị trả về khi biểu thức sai]

Ví dụ	Kết quả	Giải thích
int x = 10 < 20 ? 10 : 20;	x = 10	Nếu 10 < 20 thì vế phải sẽ trả về 10, ngược lại sẽ trả về 20. Sau đó giá trị này được gán cho x
int y = (10 < 20) && (20 > 20) ? 5 : 10;	y = 10	Nếu 10 < 20 và 20 > 20 thì sẽ gán 5 cho y, ngược lại gán 10 cho y

5. Toán tử

CHÚ Ý Ở PHẦN TOÁN TỬ

Các toán tử sẽ có thứ tự ưu tiên nhất định, ví dụ như nhân chia trước, cộng trừ sau hoặc cùng mức độ ưu tiên thì thực thi từ trái qua phải

Nhưng dấu đóng mở ngoặc tròn luôn có độ ưu tiên cao nhất, vì thế khi viết biểu thức thì các bạn nên sử dụng dấu ngoặc để biểu thức được thực thi theo đúng mong muốn của mình.

6. THƯ VIỆN & CÁC HÀM TOÁN HỌC

Một số hàm toán học quan trọng

Hàm	Chức năng	Ví dụ
1. abs(x)	Trả về giá trị tuyệt đối của số nguyên x	cout << abs(-50); // 50
2. max(x, y)	Trả về số lớn hơn trong 2 số x, y	cout << max(10, 3); // 10
3. min(x, y)	Trả về số nhỏ hơn trong 2 số x, y	cout << min(10, 3); // 3
4. sqrt(n)	Trả về căn bậc 2 của n	cout << sqrt(10); // 3.162
5. cbrt(n)	Trả về căn bậc 3 của n	cout << cbrt(10); // 2.15443
7. pow(a, b)	Trả về a^b	cout << pow(2, 10); // 1024
8. ceil(n)	Làm tròn lên giá trị của số thực n	cout << ceil(3.2); // 4.0
9. floor(n)	Làm tròn xuống giá trị của số thực n	cout << floor(3.9); // 3

Lưu ý về các hàm toán học

1. Bạn có thể khai báo thư viện cmath hoặc math.h để sử dụng các hàm toán học này, một số hàm như abs, max, min là các hàm built-in nên có thể dung luôn mà không cần thư viện

2. Các hàm toán học phần lớn trả về giá trị ở số double, ví dụ hàm pow, sqrt, cbrt, ceil, floor, round đều trả về số thực. Trong trường hợp bạn gọi các hàm này với kết quả nhỏ nó sẽ chưa thể hiện rõ điều đó nhưng khi giá trị trả về của các hàm này lớn thì sẽ có sự khác biệt khi hiển thị.

Bài tập áp dụng : http://oj.28tech.com.vn/problem/oj04

KẾT THÚC BUỔI HỌC