← C++ 存储类　　C++ 循环 →

C++ 运算符

运算符是一种告诉编译器执行特定的数学或逻辑操作的符号。C++ 内置了丰富的运算符，并提供了以下类型的运算符：

- 算术运算符
- 关系运算符
- 逻辑运算符
- 位运算符
- 赋值运算符
- 杂项运算符

本章将逐一介绍算术运算符、关系运算符、逻辑运算符、位运算符、赋值运算符和其他运算符。

算术运算符

下表显示了 C++ 支持的算术运算符。

假设变量 A 的值为 10，变量 B 的值为 20，则：

运算符	描述	实例
+	把两个操作数相加	A + B 将得到 30
-	从第一个操作数中减去第二个操作数	A - B 将得到 -10
*	把两个操作数相乘	A * B 将得到 200
/	分子除以分母	B / A 将得到 2
%	取模运算符，整除后的余数	B % A 将得到 0
++	自增运算符，整数值增加 1	A++ 将得到 11
--	自减运算符，整数值减少 1	A-- 将得到 9

实例

请看下面的实例，了解 C++ 中可用的算术运算符。

复制并粘贴下面的 C++ 程序到 test.cpp 文件中，编译并运行程序。

实例

分类导航

HTML / CSS
JavaScript
服务端
数据库
移动端
XML 教程
ASP.NET
Web Service
开发工具
网站建设

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

int main()
{
   int a = 21;
   int b = 10;
   int c;

   c = a + b;
   cout << "Line 1 - c 的值是 " << c << endl ;
   c = a - b;
   cout << "Line 2 - c 的值是 " << c << endl ;
   c = a * b;
   cout << "Line 3 - c 的值是 " << c << endl ;
   c = a / b;
   cout << "Line 4 - c 的值是 " << c << endl ;
   c = a % b;
   cout << "Line 5 - c 的值是 " << c << endl ;

   int d = 10;   //  测试自增、自减
   c = d++;
   cout << "Line 6 - c 的值是 " << c << endl ;

   d = 10;    // 重新赋值
   c = d--;
   cout << "Line 7 - c 的值是 " << c << endl ;
   return 0;
}
```

当上面的代码被编译和执行时，它会产生以下结果：

```
Line 1 - c 的值是 31
Line 2 - c 的值是 11
Line 3 - c 的值是 210
Line 4 - c 的值是 2
Line 5 - c 的值是 1
Line 6 - c 的值是 10
Line 7 - c 的值是 10
```

关系运算符

下表显示了 C++ 支持的关系运算符。

假设变量 A 的值为 10，变量 B 的值为 20，则：

运算符	描述	实例
==	检查两个操作数的值是否相等，如果相等则条件为真。	(A == B) 不为真。
!=	检查两个操作数的值是否相等，如果不相等则条件为真。	(A != B) 为真。
>	检查左操作数的值是否大于右操作数的值，如果是则条件为真。	(A > B) 不为真。

<	检查左操作数的值是否小于右操作数的值，如果是则条件为真。	(A < B) 为真。
>=	检查左操作数的值是否大于或等于右操作数的值，如果是则条件为真。	(A >= B) 不为真。
<=	检查左操作数的值是否小于或等于右操作数的值，如果是则条件为真。	(A <= B) 为真。

实例

请看下面的实例，了解 C++ 中可用的关系运算符。

复制并黏贴下面的 C++ 程序到 test.cpp 文件中，编译并运行程序。

实例

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

int main()
{
   int a = 21;
   int b = 10;
   int c ;

   if( a == b )
   {
      cout << "Line 1 - a 等于 b" << endl ;
   }
   else
   {
      cout << "Line 1 - a 不等于 b" << endl ;
   }
   if ( a < b )
   {
      cout << "Line 2 - a 小于 b" << endl ;
   }
   else
   {
      cout << "Line 2 - a 不小于 b" << endl ;
   }
   if ( a > b )
   {
      cout << "Line 3 - a 大于 b" << endl ;
   }
   else
   {
      cout << "Line 3 - a 不大于 b" << endl ;
   }
   /* 改变 a 和 b 的值 */
   a = 5;
   b = 20;
   if ( a <= b )
   {
      cout << "Line 4 - a 小于或等于 b" << endl ;
   }
   if ( b >= a )
   {
      cout << "Line 5 - b 大于或等于 a" << endl ;
   }
```

```
   return 0;
}
```

当上面的代码被编译和执行时，它会产生以下结果：

```
Line 1 - a 不等于 b
Line 2 - a 不小于 b
Line 3 - a 大于 b
Line 4 - a 小于或等于 b
Line 5 - b 大于或等于 a
```

逻辑运算符

下表显示了 C++ 支持的关系逻辑运算符。

假设变量 A 的值为 1，变量 B 的值为 0，则：

运算符	描述	实例
&&	称为逻辑与运算符。如果两个操作数都非零，则条件为真。	(A && B) 为假。
||	称为逻辑或运算符。如果两个操作数中有任意一个非零，则条件为真。	(A || B) 为真。
!	称为逻辑非运算符。用来逆转操作数的逻辑状态。如果条件为真则逻辑非运算符将使其为假。	!(A && B) 为真。

实例

请看下面的实例，了解 C++ 中可用的逻辑运算符。

复制并黏贴下面的 C++ 程序到 test.cpp 文件中，编译并运行程序。

实例

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

int main()
{
   int a = 5;
   int b = 20;
   int c ;

   if ( a && b )
   {
      cout << "Line 1 - 条件为真"<< endl ;
   }
   if ( a || b )
   {
      cout << "Line 2 - 条件为真"<< endl ;
   }
   /* 改变 a 和 b 的值 */
   a = 0;
   b = 10;
   if ( a && b )
   {
```

```
      cout << "Line 3 - 条件为真"<< endl ;
   }
   else
   {
      cout << "Line 4 - 条件不为真"<< endl ;
   }
   if ( !(a && b) )
   {
      cout << "Line 5 - 条件为真"<< endl ;
   }
   return 0;
}
```

当上面的代码被编译和执行时，它会产生以下结果：

```
Line 1 - 条件为真
Line 2 - 条件为真
Line 4 - 条件不为真
Line 5 - 条件为真
```

位运算符

位运算符作用于位，并逐位执行操作。&、 | 和 ^ 的真值表如下所示：

p	q	p & q	p | q	p ^ q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

假设如果 A = 60，且 B = 13，现在以二进制格式表示，它们如下所示：

A = 0011 1100

B = 0000 1101

A&B = 0000 1100

A|B = 0011 1101

A^B = 0011 0001

~A = 1100 0011

下表显示了 C++ 支持的位运算符。假设变量 A 的值为 60，变量 B 的值为 13，则：

运算符	描述	实例
&	如果同时存在于两个操作数中，二进制 AND 运算符复制一位到结果中。	(A & B) 将得到 12，即为 0000 1100
|	如果存在于任一操作数中，二进制 OR 运算符复制一位到结果中。	(A | B) 将得到 61，即为 0011 1101
^	如果存在于其中一个操作数中但不同时存在于两个操作数中，二进制异或运算符复制一	(A ^ B) 将得到 49，即为 0011 0001

	位到结果中。	
~	二进制补码运算符是一元运算符，具有"翻转"位效果，即0变成1，1变成0。	(~A) 将得到 -61，即为 1100 0011，一个有符号二进制数的补码形式。
<<	二进制左移运算符。左操作数的值向左移动右操作数指定的位数。	A << 2 将得到 240，即为 1111 0000
>>	二进制右移运算符。左操作数的值向右移动右操作数指定的位数。	A >> 2 将得到 15，即为 0000 1111

实例

请看下面的实例，了解 C++ 中可用的位运算符。

复制并黏贴下面的 C++ 程序到 test.cpp 文件中，编译并运行程序。

实例

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

int main()
{
   unsigned int a = 60;      // 60 = 0011 1100
   unsigned int b = 13;      // 13 = 0000 1101
   int c = 0;

   c = a & b;             // 12 = 0000 1100
   cout << "Line 1 - c 的值是 " << c << endl ;

   c = a | b;             // 61 = 0011 1101
   cout << "Line 2 - c 的值是 " << c << endl ;

   c = a ^ b;             // 49 = 0011 0001
   cout << "Line 3 - c 的值是 " << c << endl ;

   c = ~a;                // -61 = 1100 0011
   cout << "Line 4 - c 的值是 " << c << endl ;

   c = a << 2;            // 240 = 1111 0000
   cout << "Line 5 - c 的值是 " << c << endl ;

   c = a >> 2;            // 15 = 0000 1111
   cout << "Line 6 - c 的值是 " << c << endl ;

   return 0;
}
```

当上面的代码被编译和执行时，它会产生以下结果：

```
Line 1 - c 的值是 12
Line 2 - c 的值是 61
Line 3 - c 的值是 49
Line 4 - c 的值是 -61
Line 5 - c 的值是 240
Line 6 - c 的值是 15
```

赋值运算符

下表列出了 C++ 支持的赋值运算符：

运算符	描述	实例
=	简单的赋值运算符，把右边操作数的值赋给左边操作数	C = A + B 将把 A + B 的值赋给 C
+=	加且赋值运算符，把右边操作数加上左边操作数的结果赋值给左边操作数	C += A 相当于 C = C + A
-=	减且赋值运算符，把左边操作数减去右边操作数的结果赋值给左边操作数	C -= A 相当于 C = C - A
*=	乘且赋值运算符，把右边操作数乘以左边操作数的结果赋值给左边操作数	C *= A 相当于 C = C * A
/=	除且赋值运算符，把左边操作数除以右边操作数的结果赋值给左边操作数	C /= A 相当于 C = C / A
%=	求模且赋值运算符，求两个操作数的模赋值给左边操作数	C %= A 相当于 C = C % A
<<=	左移且赋值运算符	C <<= 2 等同于 C = C << 2
>>=	右移且赋值运算符	C >>= 2 等同于 C = C >> 2
&=	按位与且赋值运算符	C &= 2 等同于 C = C & 2
^=	按位异或且赋值运算符	C ^= 2 等同于 C = C ^ 2
|=	按位或且赋值运算符	C |= 2 等同于 C = C | 2

实例

请看下面的实例，了解 C++ 中可用的赋值运算符。

复制并黏贴下面的 C++ 程序到 test.cpp 文件中，编译并运行程序。

实例

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

int main()
{
   int a = 21;
   int c ;

   c =  a;
   cout << "Line 1 - =  运算符实例，c 的值 = : " <<c<< endl ;

   c +=  a;
   cout << "Line 2 - += 运算符实例，c 的值 = : " <<c<< endl ;

   c -=  a;
```

```
   cout << "Line 3 - -= 运算符实例，c 的值 = : " <<c<< endl ;

   c *=  a;
   cout << "Line 4 - *= 运算符实例，c 的值 = : " <<c<< endl ;

   c /=  a;
   cout << "Line 5 - /= 运算符实例，c 的值 = : " <<c<< endl ;

   c  = 200;
   c %=  a;
   cout << "Line 6 - %= 运算符实例，c 的值 = : " <<c<< endl ;

   c <<=  2;
   cout << "Line 7 - <<= 运算符实例，c 的值 = : " <<c<< endl ;

   c >>=  2;
   cout << "Line 8 - >>= 运算符实例，c 的值 = : " <<c<< endl ;

   c &=  2;
   cout << "Line 9 - &= 运算符实例，c 的值 = : " <<c<< endl ;

   c ^=  2;
   cout << "Line 10 - ^= 运算符实例，c 的值 = : " <<c<< endl ;

   c |=  2;
   cout << "Line 11 - |= 运算符实例，c 的值 = : " <<c<< endl ;

   return 0;
}
```

当上面的代码被编译和执行时，它会产生以下结果：

```
Line 1 - =  运算符实例，c 的值 = 21
Line 2 - += 运算符实例，c 的值 = 42
Line 3 - -= 运算符实例，c 的值 = 21
Line 4 - *= 运算符实例，c 的值 = 441
Line 5 - /= 运算符实例，c 的值 = 21
Line 6 - %= 运算符实例，c 的值 = 11
Line 7 - <<= 运算符实例，c 的值 = 44
Line 8 - >>= 运算符实例，c 的值 = 11
Line 9 - &= 运算符实例，c 的值 = 2
Line 10 - ^= 运算符实例，c 的值 = 0
Line 11 - |= 运算符实例，c 的值 = 2
```

杂项运算符

下表列出了 C++ 支持的其他一些重要的运算符。

运算符	描述
sizeof	sizeof 运算符返回变量的大小。例如，sizeof(a) 将返回 4，其中 a 是整数。
Condition ? X : Y	条件运算符。如果 Condition 为真 ? 则值为 X : 否则值为 Y。
,	逗号运算符会顺序执行一系列运算。整个逗号表达式的值

	是以逗号分隔的列表中的最后一个表达式的值。
.（点）和 ->（箭头）	成员运算符用于引用类、结构和共用体的成员。
Cast	强制转换运算符把一种数据类型转换为另一种数据类型。例如，int(2.2000) 将返回 2。
&	指针运算符 & 返回变量的地址。例如 &a; 将给出变量的实际地址。
*	指针运算符 * 指向一个变量。例如，*var; 将指向变量 var。

C++ 中的运算符优先级

运算符的优先级确定表达式中项的组合。这会影响到一个表达式如何计算。某些运算符比其他运算符有更高的优先级，例如，乘除运算符具有比加减运算符更高的优先级。

例如 x = 7 + 3 * 2，在这里，x 被赋值为 13，而不是 20，因为运算符 * 具有比 + 更高的优先级，所以首先计算乘法 3*2，然后再加上 7。

下表将按运算符优先级从高到低列出各个运算符，具有较高优先级的运算符出现在表格的上面，具有较低优先级的运算符出现在表格的下面。在表达式中，较高优先级的运算符会优先被计算。

类别	运算符	结合性
后缀	() [] -> . ++ - -	从左到右
一元	+ - ! ~ ++ - - (type)* & sizeof	从右到左
乘除	* / %	从左到右
加减	+ -	从左到右
移位	<< >>	从左到右
关系	< <= > >=	从左到右
相等	== !=	从左到右
位与 AND	&	从左到右
位异或 XOR	^	从左到右
位或 OR	|	从左到右
逻辑与 AND	&&	从左到右
逻辑或 OR	||	从左到右
条件	?:	从右到左

赋值	= += -= *= /= %=>>= <<= &= ^= |=	从右到左
逗号	,	从左到右

实例

请看下面的实例，了解 C++ 中运算符的优先级。

复制并黏贴下面的 C++ 程序到 test.cpp 文件中，编译并运行程序。

对比有括号和没有括号时的区别，这将产生不同的结果。因为 ()、 /、 * 和 + 有不同的优先级，高优先级的操作符将优先计算。

实例

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

int main()
{
   int a = 20;
   int b = 10;
   int c = 15;
   int d = 5;
   int e;

   e = (a + b) * c / d;      // ( 30 * 15 ) / 5
   cout << "(a + b) * c / d 的值是 " << e << endl ;

   e = ((a + b) * c) / d;    // (30 * 15 ) / 5
   cout << "((a + b) * c) / d 的值是 " << e << endl ;

   e = (a + b) * (c / d);   // (30) * (15/5)
   cout << "(a + b) * (c / d) 的值是 " << e << endl ;

   e = a + (b * c) / d;     //  20 + (150/5)
   cout << "a + (b * c) / d 的值是 " << e << endl ;

   return 0;
}
```

当上面的代码被编译和执行时，它会产生以下结果：

```
(a + b) * c / d 的值是 90
((a + b) * c) / d 的值是 90
(a + b) * (c / d) 的值是 90
a + (b * c) / d 的值是 50
```

← C++ 存储类　　C++ 循环 →

在线实例

· HTML 实例
· CSS 实例
· JavaScript 实例
· Ajax 实例
· jQuery 实例
· XML 实例
· Java 实例

字符集&工具

· HTML 字符集设置
· HTML ASCII 字符集
· HTML ISO-8859-1
· HTML 实体符号
· HTML 拾色器
· JSON 格式化工具

最新更新

· Python redis 使...
· Windows10 MYSQ...
· Docke 镜像加速
· Debian Docker 安装
· C 库函数 –...
· Linux groupadd ...
· CSS var() 函数

站点信息

· 意见反馈
· 合作联系
· 免责声明
· 关于我们
· 文章归档

关注微信

备案号：闽ICP备15012807号-1

反馈/建议