【C++】Day twelve(2)

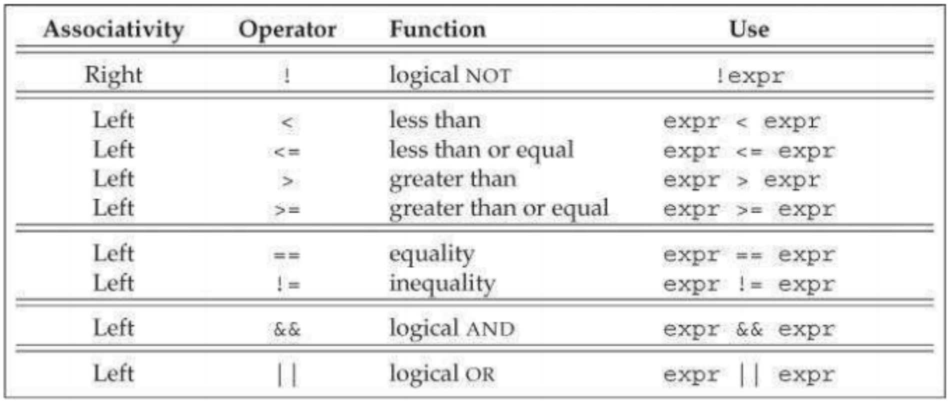
Class	C++
Date	@November 30, 2021
Material	
Series Number	
Summary	

【Ch4】Operators

4.3 Logical and Relational Operators

The relational operators take operands of arithmetic or pointer type; the logical operators take operands of any type that can be converted to bool

Table 4.2. Logical and Relational Operators



Associativity	Operator	Function	Use
Right	!	logical NOT	!expr
Left	<	less than	expr < expr
Left	<=	less than or equal	expr <= expr
Left	>	greater than	expr > expr
Left	>=	greater than or equal	expr >= expr
Left	==	equality	expr == expr
Left	!=	inequality	expr != expr
Left	&&	logical AND	expr && expr
Left	||	logical OR	expr || expr

Warning: It is usually a bad idea to use the boolean literals true and false as operands in a comparison. These literals should be used only to compare to an object of type bool.

4.4 Assignment Operators

The left-hand operand of an assignment operator must be a modifiable lvalue.

For example, given

```
int i = 0, j = 0, k = 0; //initialization, not assignment
const int ci = i; //initialization, not assignment
```

Each of these assignments is illegal

```
1024 = k; //error: literals are rvalues
i + j = k; //error: arithmetic expressions are rvalues
ci = l; //error: ci is a const lvalue
```

The result of an assignment is its left-hand operand, which is an lvalue. The type of the result is the type of the left-hand operand. If the types of the left and right operands differ, the right-hand operand is converted to the type of the left:

```
k = 0; //result: type int, value 0
k = 3.14; //result: type int, value 3
```

Assignment Is Right Associative

Unlike the other binary operators, assignment is right associative:

```
int ival, jval;
ival = jval = 0; //ok: each assigned 0
```

Because assignment is right associative, the right-most assignment, `jval = 0`, is the right-hand operand of the left-most assignment oeprator. Because assignment returns its left-hand operand, the result of the right-most assignment is assigned to ival.

Each object in a multiple assignment must have the same type as its right-hand neighbor or a type to which that neighbor can be converted.

Note: Because assignment has lower precedence than the relational operators, parentheses are usually needed around assignments in conditions.

4.5 Increment and Decrement Operators

Advice Use Postfix Operators only When Necessary

The postfix operator must store the original value so that it can return the unincremented value as its result. If we don't need the unincremented value, there's no need for the extra work done by the postfix operator.

As one example, we can use postfix increment to write a loop to print the values in a vector up to but not including the first negative value:

```
auto pbeg = v.begin();
while(pbeg != v.end() && *beg >= 0)
  cout << *pbeg++ << endl; //print the current value and advance pbeg
```

The precedence of postfix increment is higher than that of the dereference operator, so `*pbeg++` is equivalent to `*(pbeg++)`. The subexpression `pbeg++` increments pbeg and yields a copy of the previous value of pbeg as its result. Accordingly, the operand of `*` is the unincremented value of `pbeg`. Thus, the statement prints the element to which pbeg originally pointed and increments pbeg.