【C++】Day twelve(3)

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

【Ch4】The Member Access Operators

4.6 The Member Access Operators

The dot and arrow operators provide for member access. The dot operator fetches a member from an object of class type; arrow is defined so that `ptr→mem` is a synonym for `(*ptr).mem`;

```
string s1 = "a string", *p = &s1;
auto n = s1.size(); // run the size member of the string s1
n = (*p).size(); //run size on the object to which p points
n = p->size(); //equivalent to (*p).size()
```

4.7 The Conditional Operator

Using a Conditional Operator in an Output Expression

The conditional operator has fairly low precedence. When we embed a conditional expression in a larger expression, we usually must parenthesize the conditional expression. An incompletely parenthesized conditional operator in an output expression can have surprising results:

```
cout << ((grade < 60) ? "fail" : "pass"); //print pass or fail
cout << (grade < 60) ? "fail" : "pass"; //print 1 or 0!
cout << grade < 60 ? "fail" : "pass"; //error: compares cout to 60
```

The second expression uses the comparison between grade and 60 as the operand to the << operator. The value 1 or 0 is printed, depending on whether grade < 60 is true or false.

The << operator returns cout, which is tested as the condition for the conditional operator. That is, the second expression is equivalent to:

```
cout << (grade < 60);
cout ? "fail" : "pass"; //test cout and then yield one of the two literals depending on whether cout is true or false.
```