【C++】Day16(2)

Class	C++
Date	@December 6, 2021
Material	
Series Number	
Summary	

【Ch6】Function

6.2.3 const Parameters and Arguments

Top-level const on parameters are ignored. We can pass either a const or a nonconst object to a parameter that has a top-level const:

```
//func can read but not write i
void func(const int i) {}
```

In C++, we can define several different functions that have the same name. However, we can do so only if their parameter lists are sufficiently different.

```
void func(const int i) {}
void func(int i) {} //error
```

Because top-level consts are ignored, we will pass exactly the same types to either version of fcn. The second version of fcn is an error

Pointer or Reference Parameters and const

We can initialize an object with a low-level const from a nonconst object but not vice versa.

Use Reference to const When Possible

It is a common mistake to define parameters that a function does not change as (plain) references. Doing so gives the function's caller the misleading impression that the function might change its argument's value.

Moreoever, using a reference instead of a reference to const undly limits the type of arguments that can be used with the function.

For example, we cannot pass a const string if the function requires a string.

6.2.4 Array Parameters

Array have two special properties that affect how we define and use functions that operate on arrays: We cannot copy an array, and when we use an array it is usually converted to a pointer.