【C++】Day six(3)

Class	C++
Date	@November 23, 2021
Material	
Series Number	
Summary	const continue(2)

【Ch2】Const Continue(2)

2.4.4 constexpr and Constant Expression

A constant expression is an expression whose value cannot change and that can be evaluated at compile time.

A literal is a constant expression. A const object that is initialized from a constant expression is also a constant expression.

```
const int max_files = 20; //max_files is a constant expression
const int limit = max_files + 1; //limit is a constant expression
int staff_size = 27;
const int sz = get_size(); //sz is not a constant expression
```

Even though sz is a const, the value of its initializer is not known until run time. Hence, sz is not a constant expression.

We might define a const variable with an initializer that we think is a constant expression. However, when we use that variable in a context that requires a constant expression we may discover that the initializer was not a constant expression.

We can ask the compiler to verify that a variable is a constant expression by declaring the variable in a `constexpr` declaration. Variables declared as constexpr are implicitly const and must be initialized by const expression.

```
constexpr int mf = 20; //20 is a constant expression
constexpr int limit = mf + 1; //mf + 1 is a constant expression
constexpr int sz = size(); //ok only if size is a constexpr function
```

Although we cannot use an ordinary function as an initializer for a constexpr variable, we'll see that the new standard lets us define certain functions as constexpr. We can use constexpr functions in the initializer of a constexpr variable.

Note: Generally, it is a good idea to use constexpr for variables that you intend to use as constant expression.