第17章 基于范围的for循环

《现代C++语言核心特性解析》 谢丙堃

烦琐的容器遍历

- 例子

```
std::map<int, std::string> index_map{ {1, "hello"}, {2, "world"}, {3, "!"} };

std::map<int, std::string>::iterator it = index_map.begin();
for (; it != index_map.end(); ++it) {
    std::cout << "key=" << (*it).first
    << ", value=" << (*it).second << std::endl;
}
```

基于范围的for循环语法

- 范围声明和一个范围表达式

```
for ( range_declaration : range_expression ) loop_statement
```

- 范围声明是一个变量的声明，其类型是范围表达式中元素的类型或者元素类型的引用。

- 范围表达式可以是数组或者是对象，对于对象必须满足以下2个条件中的任意一个：
  - 对象类型定义了begin和end成员函数。
  - 定义了以对象类型为参数的begin和end普通函数。

基于范围的for循环语法

- 例子

```
std::map<int, std::string> index_map{ {1, "hello"}, {2, "world"}, {3, "!"} };
int int_array[] = { 0, 1, 2, 3, 4, 5 };

int main()
{
    for (const auto &e : index_map) {
        std::cout << "key=" << e.first << ", value=" << e.second << std::endl;
    }

    for (auto e : int_array) {
        std::cout << e << std::endl;
    }
}
```

begin和end函数不必返回相同类型

```
{// C++11
    auto && __range = range_expression;
    for (auto __begin = begin_expr, __end = end_expr; __begin != __end; ++__begin) {
        range_declaration = *__begin;
        loop_statement
    }
}

{// C++17
    auto && __range = range_expression;
    auto __begin = begin_expr;
    auto __end = end_expr;
    for (; __begin != __end; ++__begin) {
        range_declaration = *__begin;
        loop_statement
    }
}
```

临时范围表达式的陷阱

- 例子

```
class T {
  std::vector<int> data_;
public:
  std::vector<int>& items() { return data_; }
  // ...
};
T foo()
{
    T t;
    return t;
}
for (auto& x : foo().items()) {} // 未定义行为
```

临时范围表达式的陷阱

- 解决方案

```
// 拷贝
T thing = foo();
for (auto & x :thing.items()) {}

// C++20
for (T thing = foo(); auto & x :thing.items()) {}
```

实现一个支持基于范围的for循环的类

- 条件：
  - 该类型必须有一组和其类型相关的begin和end函数，它们可以是类型的成员函数也可以是独立函数。
  - begin和end函数需要返回一组类似迭代器的对象，并且这组对象必须支持operator *，operator !=和operator ++运算符函数。

感谢聆听
欢迎关注