第7章 lambda表达式

《现代C++语言核心特性解析》 谢丙堃

lambda表达式语法

- 语法定义

```
[ captures ] ( params ) specifiers exception -> ret { body }
```

- 示例

```
int x = 5;
auto foo = [x](int y)->int { return x * y; };
std::cout << foo(8) << std::endl;
```

捕获列表

- 捕获列表的作用域

```
int x = 0;
int main()
{
    int y = 0;
    static int z = 0;
    auto foo = [x, y, z] {};
}
```

- 捕获值和捕获引用

```
int x = 5, y = 8;
auto foo1 = [x, y] { return x * y; };
auto foo2 = [&x, &y] { return x * y; };
```

捕获列表

- 特殊的捕获方法
  - [this] - 捕获this指针，捕获this指针可以让我们使用this类型的成员变量和函数。
  - [=] - 捕获lambda表达式定义作用域的全部变量的值，包括this。
  - [&] - 捕获lambda表达式定义作用域的全部变量的引用，包括this。

无状态lambda表达式

- 隐式转换为函数指针

```
void f(void(*)()) {}
void g() { f([] {}); } // 编译成功
```

在STL中使用lambda表达式

```
#include <iostream>
#include <vector>
#include <algorithm>
int main()
{
    std::vector<int> x = {1, 2, 3, 4, 5};
    std::cout << *std::find_if(x.cbegin(), x.cend(),
                 [](int i) { return (i % 3) == 0; }) << std::endl;
}
```

广义捕获

- 简单捕获
- 初始化捕获

```
int main()
{
    int x = 5;
    auto foo = [r = x + 1]{ return r; };
}
```

泛型lambda表达式

```
int main()
{
    auto foo = [](auto a) { return a; };
    int three = foo(3);
    char const* hello = foo("hello");
}
```

其他特性

- 捕获[*this]和捕获[=, this]
  - 拷贝this对象
  - [=, this]为了区分[=, *this]

- 模板语法的泛型lambda表达式

```
auto f = []<typename T>(std::vector<T> vector) { // ...
};
```

- 可构造和可赋值的无状态lambda表达式

```
auto greater = [](auto x, auto y) { return x > y; };
std::map<std::string, int, decltype(greater)> mymap;
```

感谢聆听
欢迎关注