lambda *表达式捕获列表的变量必须是一个自动存储类型,也就是非静态的局部变量*

```
int static_test1;
auto foo = [static_test1] {};// 编译器报错,因为static_test1是一个全局变量
```

```
static int static_test2;
auto foo = [static_test2] {};// 编译器报错,因为static_test2是一个静态变量
```

值捕获

```
 int x = 1, y = 1;

 auto foo1 = [x, y]() {};//值捕获
 //auto foo1 = [x, y] { x++, y++ }; 编译器报错,因为是值捕获不允许修改
 auto foo2 = [x, y]()mutable { ++x, ++y; };
 //std::cout << x << " " << y << std::endl;加了mutable关键字使得x,y可以修改但是输出结果依旧为1,1是因为值捕获是把原变量拷贝了一份
```

引用捕获

```
 auto foo3 = [&x, &y] { ++x, ++y; };//引用捕获
```

this捕获

```
   struct A
    {
        static void print() { std::cout << __func__ << std::endl; }
        void test()
        {
            auto foo = [this] { print(); };
            //lambda 表达式通过this捕获使得可以在lambda表达式里直接访问成员
        }
    };
```

捕获定义域变量

```
 auto foo4 = [=] {};//捕获lambda表达式定义域的全部变量的值
 auto foo5 = [&] {};//捕获lambda表达式定义域的全部变量的引用
```

在捕获列表定义变量

```
auto foo6 = [i = 2] { std::cout << i; };
auto foo7 = [&i = x] { ++i; };
```