**Lambda 表达式本质上是一种创建匿名函数对象的便捷方式。函数对象（也称为仿函数）是一个类的实例，这个类定义了 operator() 运算符，使得该对象可以像函数一样被调用。**

**Lambda 表达式**：就是一个表达式，它是源代码的一部分。

**闭包类**：编译器会为每个 Lambda 生成一个唯一的闭包类，闭包就是从这个类实例化而来的。

闭包，是实例化的对象。

编译时存在的 Lambda 表达式和闭包类，以及运行时存在的闭包。

不捕获外部变量，无状态lambda表达式

按引用捕获和按值捕获。默认按引用捕获可能会导致悬空引用。按值捕获较安全，但是捕获指针时，也有可能导致指针悬空。默认按值捕获成员变量时，其实捕获的是this指针。

将lambda表达式当做类，当有捕获时，会进行成员变量的初始化操作。

C++14 引入了初始化捕获（init capture）机制。通过=符号来捕获。“=” 左右两边的作用域不同，左边是闭包类的作用域，右边是 Lambda 表达式定义的作用域。

从 C++17 开始，如果条件允许，Lambda 表达式会隐式地成为常量表达式（constexpr）。建议使用 constexpr auto squared = [](auto val) constexpr { ... }; 的声明方式。（防止两个初始化顺序错误、

从 C++17 开始，你可以通过捕获 \*this 显式要求捕获当前对象的副本。以前是引用，可能有问题。

在 C++20 之前，无捕获的 Lambda 表达式既不可默认构造，也不可赋值。从 C++20 开始，无捕获的 Lambda 表达式具备了这两个特性。

泛型 Lambda 表达式

接受 auto 类型参数的 Lambda 表达式，这使得它可以使用任何类型进行调用。

C++20**使用 typename 替代 auto**：C++20 允许在泛型 Lambda 表达式的参数类型中使用 typename 语法，即 []<typename Val>(Val v) { return v + 1; }。