6 函数

1，函数基础

函数的返回类型不能是数组类型或函数类型，但可以是指向数组或函数的指针。

定义成static的局部变量的生命周期贯穿函数调用及之后的时间。

函数只能定义一次，但可以声明多次。函数的声明可以省略参数名字。

函数应该在头文件中声明，在源文件中定义。

定义函数的源文件应该把含有函数声明的头文件包含进来，编译器负责验证函数的定义和声明是否匹配。

2，参数传递

熟悉C语言的程序员常常使用指针类型的形参访问函数外部的对象。在C++语言中，建议使用引用类型的形参替代指针。

如果函数无需改变应用形参的值，最好将其声明为常量引用。

数组的两个特殊性质：不允许拷贝数组，数组使用时会被转换成指针。

当我们为函数传递一个数组时，实际上传递的是指向数组首元素的指针。

为了编写能处理不同数量实参的函数的三种方法：

1. 如果实参类型相同，使用initializer\_list；
2. 实参类型不同，使用过可变参数模版；

3，使用省略符(这种功能一般只用于与C函数交互的接口程序)

3 返回类型和return语句

不要返回局部对象的引用或指针。

要想定义一个返回数组的指针或引用的函数比较烦琐，但是有一些办法可以简化这一任务，其中最直接的方法是使用类型别名typedef。

4 函数重载

顶层const的形参无法和没有顶层const的形参区分开来，但是底层const是可以重载的。

5 特殊用途语言特性

默认实参

默认实参，一旦某个形参被赋予了默认值，它后面的所有形参都必须有默认值。

内联函数

调用函数比求等价表达式的值慢一些，内联函数可避免函数调用的开销，它会在调用点上“内联地”展开。

在函数的返回类型前面加上关键字inline，这样就可以将它声明为内联函数。

一般来说，内联机制用于优化规模较小，流程直接，频繁调用的函数。

constexpr函数是指能用于常量表达式的函数。

constexpr函数不一定返回常量表达式。

调试帮助

assert是一种预处理宏，assert(expr)首先对expr求值，如果表达式为假，assert输出信息并终止程序的执行。如果表达式为真（即非0），assert什么也不做。

assert的行为依赖于一个名为NDEBUG的预处理变量的状态。如果定义了NDEBUG，则assert什么都不做。

#define NDEBUG加到每个源文件中,但这条语句必须放在#include <assert.h>之前。

使用\_\_func\_\_输出当前调试的函数的名字。它是一个静态数组。

预处理器还定义了另外4个对于程序测试很有用的名字：

\_\_FILE\_\_存放文件名的字符串字面值；\_\_LINE\_\_存放当前行的整型字面值；\_\_TIME\_\_存放文件编译时间的字符串字面值；\_\_DATA\_\_存放编译日期。

6 函数匹配

7 函数指针

要想声明一个可以指向该函数的指针，只需要用指针替换函数名即可。

当我们把函数名作为一个值使用时，该函数自动转换成指针。

pf = lengthCompare和pf = &lengthCompare等价。

我们还能直接使用指向函数的指针调用该函数，无须提前解引用指针。

pf("Hello","Hi")等价于(\*pf)("Hello","Hi")

可以直接把函数作为实参使用，此时它会自动转换成指针。

直接使用函数指针类型显得冗长而烦琐，类型别名和decltype能让我们简化使用函数指针的代码。

typedef bool Func(const string&, const string&)

typedef decltype(lengthCompare) Func2

typedef bool(\*FuncP)(const string&, const string&)

typedef decltype(lengthCompare) \*FuncP2

以上都是等价的

要想声明一个返回函数指针的函数，最简单的办法是使用类型别名：using PF = int(\*)(int\*, int);

将decltype作用于某个函数时，它返回函数类型而非指针类型。