**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。

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。

不要返回局部对象的引用或指针。

顶层const不能重载，底层const是可以重载的。

**2，参数传递**

在C++语言中，建议使用引用类型的形参替代指针。

如果函数无需改变引用形参的值，最好将其声明为常量引用。

当用实参初始化形参时会忽略掉顶层const，形参的顶层const被忽略掉了。

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，特殊用途语言特性**

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

调用函数一般比求等价表达式的值要慢一些，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。

内联说明只是向编译器发出一个请求，编译器可以选择忽略这个请求。

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

constexpr是指能用于常量表达式的函数。该函数的返回类型及所有形参的类型都得是字面值类型，而且函数体中必须有且只有一条return语句。

为了能在编译过程中随时展开，constexpr被隐式的指定为内联函数。

constexpr size\_t scale(size\_t cnt){return new\_sz() \* cnt;} scale(2)是常量表达式；

int i = 2;scale(i)不是常量表达式。constexpr函数不一定返回常量表达式。

内联函数和constexpr函数通常定义在头文件中。

**4，调试帮助**

assert是一种预处理宏，所谓预处理宏其实是一个预处理变量，它的行为类似于内联函数。

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。

如果定义了NDEBUG，则assert什么都不做。

\_\_func\_\_输出当前调试函数的名字。\_\_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。

**5，函数指针**

声明一个函数指针，只需要用指针替换函数名即可。

函数类型bool(const string &, const string &)的指针bool (\*pf)(const string &, const string &);

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。

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

将decltype作用于某个函数时，它返回函数类型而非指针类型。