通过调用运算符（“()”，一对圆括号）调用函数，作用于一个表达式，该表达式是函数或指向函数的指针。

函数的调用完成两项工作：

1.用实参初始化函数对应的形参。

2.将控制权转移给被调用函数。

调用函数后，主调函数被中断，被调函数开始执行。

函数执行的第一步：隐式地定义并初始化它的形参。实参是形参的初始值，第一个实参初始化第一个形参。

return 语句完成两项工作：

1.返回 return 语句中的值（如果有的话）

2.将控制权从被调函数转移回主调函数。

ps：返回类型不能是数组或函数类型，但是可以是数组指针或函数指针。

局部对象：在语句块内定义的变量，仅在作用域内可见。局部变量会隐藏作用域外的同名变量。

自动对象：只在块执行期间的对象。块执行结束后自动对象的值变为未定义，即被销毁。

局部静态对象：在程序的执行路径第一次经过对象定义语句时初始化，生命周期持续到程序终止。

ps：局部静态变量是 static 型而不是 const 型。

函数的名字必须在使用前声明，并且可以只声明无定义，声明无需形参的名字（因为声明不包含函数体）。

函数的声明和定义的区别在于声明不需要函数体，用一个分号代替。

可以在头文件中对函数进行声明，这样的好处：

1.可以确保所有同样的函数声明一致。

2.便于修改函数声明。

C++支持分离编译，在编译时编译器把所有的源文件链接在一起进行编译（编译过程见《深入理解计算机》系统第二章）

函数的参数传递：

传值参数：若形参为非引用，传入的实参被拷贝给自动对象，函数内自动对象不会影响传入的实参；若形参为指针，执行指针的拷贝操作时，拷贝的是指针的值，两个指针是不同的指针但指向同一片地址。

传引用参数：引用形参在传递实参时直接传入对象。最好不要传引用参数，一是可能被其他地方修改，二是可能会拷贝一些大的容器对象比如数组，这样效率很低。如果函数不需要改变引用形参的值，最好声明为常量引用。

ps：

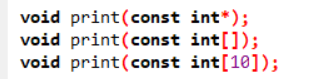
1.可以使用字面值初始化常量引用。常量引用是底层const，顶层const会被忽略。

2.不能把 const 对象、字面值或需要类型转换的对象传递给普通的引用形参。但是可以传递给常量引用形参。

数组形参：

ps：如果传递的形参是多维数组，从第二维开始后面的数组大小都为类型的一部分，不能省略，因为编译器需要这些信息来正确地计算内存地址和访问数组元素。

因为数组不能被直接拷贝，而且使用数组时会被转换为指针：



实际上这三个函数声明是一模一样的，编译器只检查传入的实参是否为 const int\* 类型。

使用数组做形参确保数组访问不越界的方法：

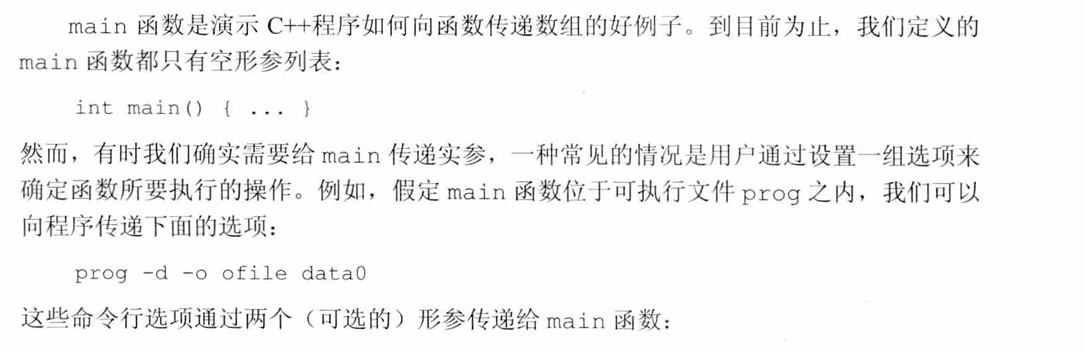
1.使用一个结束标记指定数组已结束，典型代表为 C 风格字符串（例如c风格字符串中字符数组以“\0”结尾）

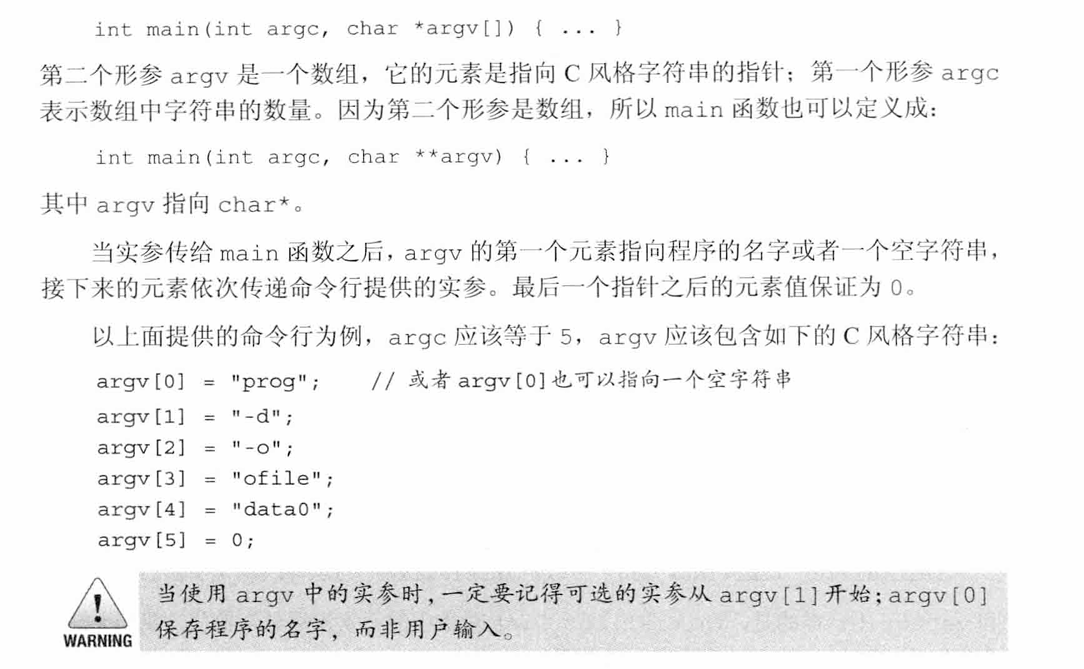
2.传递指向数组首元素和尾后元素的指针（arr.end()）

3.专门定义一个表示数组大小的形参。

数组引用形参：

image.png





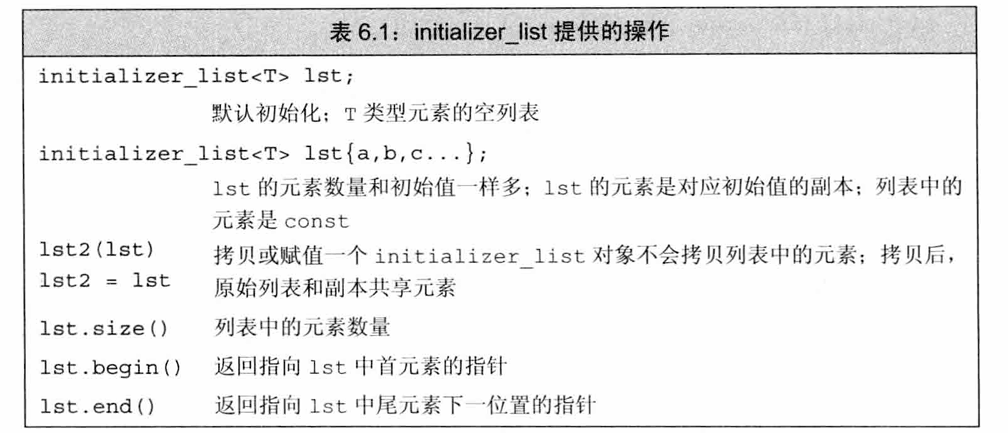
可变参数：

处理不同数量实参的主要方法：

1.如果所有实参类型相同但数量未知，传递一个initializer\_list类型。

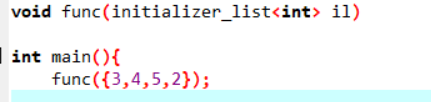
2.使用省略符形参，它可以传递可变数量的实参，注意它一般仅用于与 C 函数交互的接口程序，不会对可变数量的实参进行类型检查。

initializer\_list形参：定义在同名的头文件中



1.initializer\_list中的所有元素都是常量值

2.如果向 initailizer\_list 形参中传递一个值的序列，必须把序列放在花括号里。



省略符形参：

ps：

1.省略符形参仅用于 C 和 C++ 通用的类型，大多数类类型的对象传递给省略符形参都无法正确拷贝。

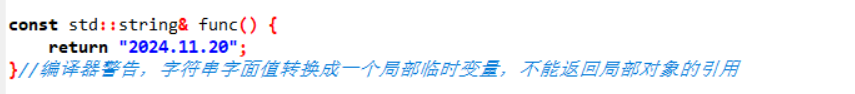
2.省略符形参只能出现于形参列表的最后一个位置。

image.png

返回类型和return语句：

返回一个值和初始化一个变量或形参的方式一样，返回的值用于初始化调用点的变量。

不能返回一个对局部变量的引用或指针



对于返回引用的函数，得到的必定是左值（因为需要确保引用必须指向一个有效的对象，避免未定义行为，右值在函数结束后会被销毁）；其他类型返回值为右值。

ps：可以为返回类型是非常量引用的函数的结果赋值。

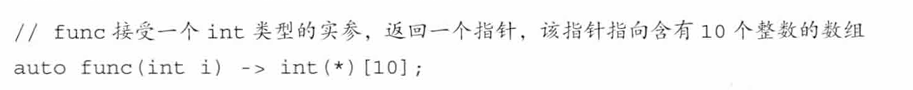
main函数的返回值：返回0代表正常退出，返回其他代表异常，cstdlib头文件定义了两个预处理变量分别表示成功和失败，EXIT\_FAILURE（失败）、EXIT\_SUCCESS（成功）

返回数组指针：

声明一个返回数组指针的函数：

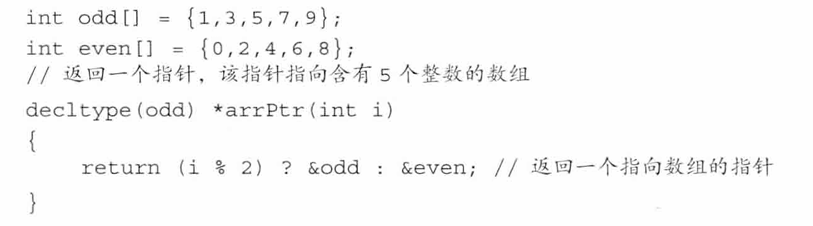
image.png

使用尾置返回类型：



尾置返回类型在形参后使用箭头标识符，然后加上实际的返回类型，在原本放返回类型的地方使用auto

使用decltype：



函数重载：

函数重载的条件：

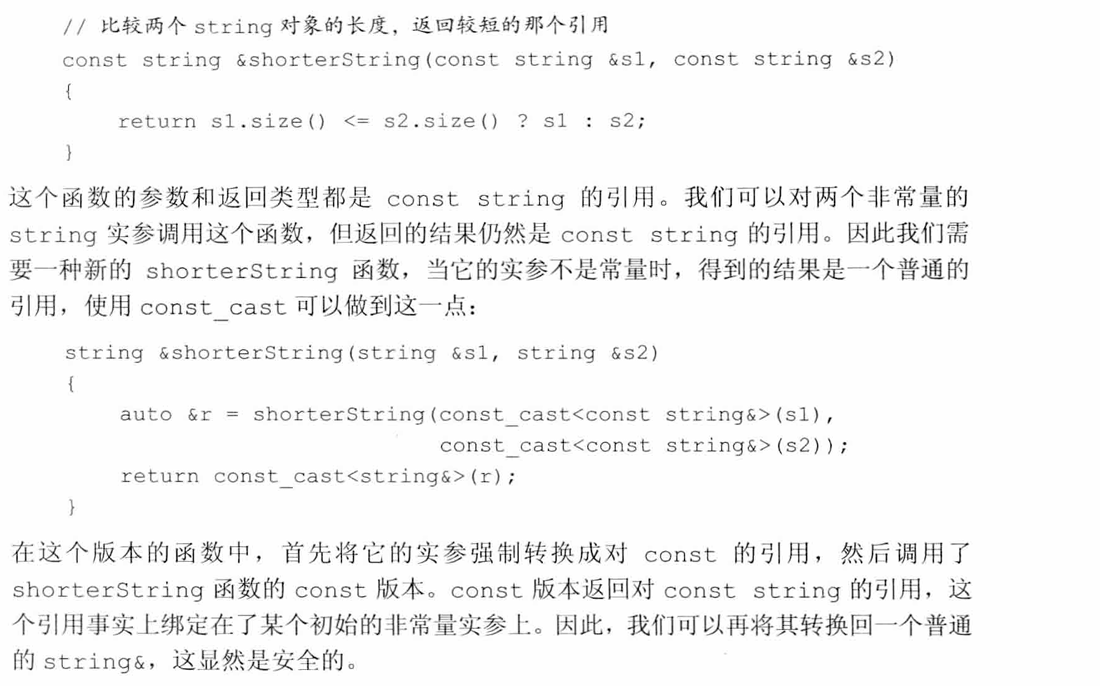
1、定义在同一个作用域（不同作用域间无法重载，外层作用域中声明的同名实体会被隐藏）；

2、函数名称相同；

3、函数参数类型不同或者个数不同或者顺序不同

ps：顶层const因为会被忽略，所以并不能区分重载。

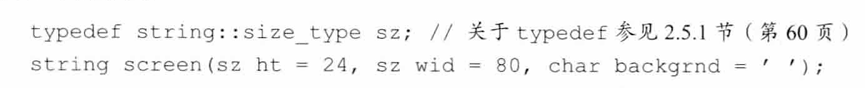
const\_cast显式转换在函数重载中的作用：



当要重载常量引用与非常量引用的版本时，在非常量引用的版本中可以通过 const\_cast 将参数和返回值从常量引用转换为非常量引用，以实现对常量引用版本的调用。

函数的默认实参：

下面的函数是含有默认实参的函数



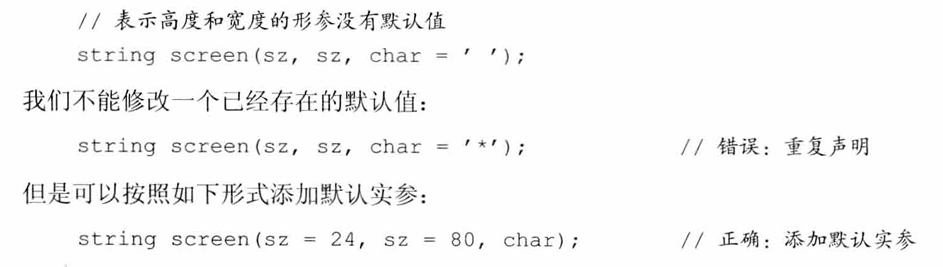
1.设置默认值的形参必须都放在没有默认值的形参后面。

2.使用默认实参时省略该实参。

3.通常应该在函数声明中指定默认实参，并将该声明放在合适的头文件中。

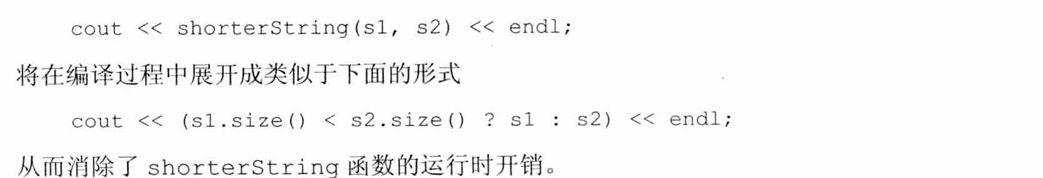
4.局部变量不能作为默认实参，全局变量和字面值都可以。

ps：通常应该在函数声明中指定默认实参，也就是说如果不这么做，在后续声明中仍能指定默认实参，但只能给未指定默认实参的形参添加默认实参，并不能改变某一已指定默认实参的形参的默认实参

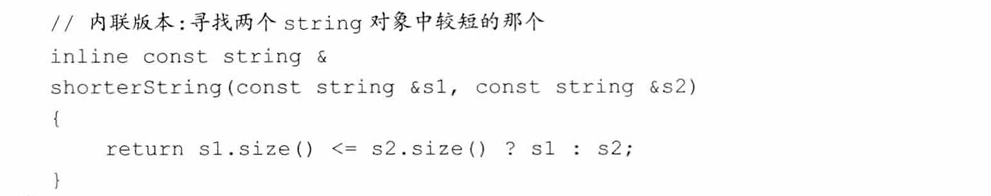


内联函数：

将函数指定为内联函数能够避免函数调用带来的开销（因为调用函数前要先保存寄存器，可能有拷贝实参的开销，可能有指针在内存中跳转的开销，最后返回时恢复寄存器也需要开销），内联函数相当于在编译时把函数展开在原位置，避免了这种开销。类似于书上的例子：



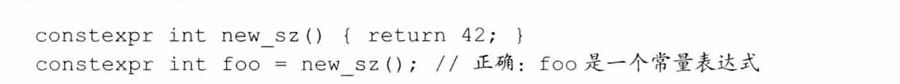
定义内联函数的方式：在函数的返回类型前加上inline关键字



ps：内联机制用于优化规模小、流程直接、调用频繁的函数。

constexpr函数：

constexpr函数指能用于常量表达式的函数（能参与常量表达式的计算），函数的返回类型、所有形参的类型都必须是字面值类型（可以不是常量但必须是常量表达式），且函数体有且只有一个出口。constexpr函数内同样可以包含其他语句，但是这些语句在运行时不能执行任何操作（例如空语句、类型别名、using声明）。



当初始化时，constexpr函数被隐式地指定为内联函数，编译器把对constexpr函数的调用替换为了这个函数的结果值。

ps：内联函数和constexpr函数都可以在定义域内多次定义，但是多次定义必须完全一致，因此内联函数和constexpr函数通常会被声明和定义在头文件中。

const和constexpr的区别：

1.编译时和运行时

constexpr确保在编译时求值，用于编译时常量表达式。

const保证变量在初始化后不可变，但允许在运行时初始化。

2.用法和约束

constexpr适用于变量、函数和构造函数，以确保其在编译时计算。

const适用于更广泛的场景，用于变量、指针、引用和成员函数等，确保它们的不可变性。

3.编译时计算

constexpr强制表达式在编译时计算，语义更严格。

const允许表达式在运行时计算，提供更多灵活性。

assert 预处理宏（“断言”）：

用法：assert(expr) （定义在头文件 cassert 中）

如果表达式 expr 为假，assert 输出信息并终止程序，如果表达式为真，assert 什么也不做。assert 常用于检查“不能发生”的条件

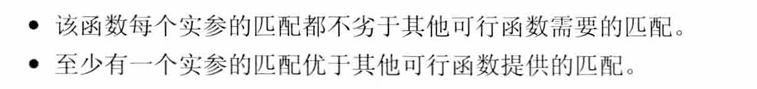
当有多个重载函数时，选出要调用的重载函数的流程：

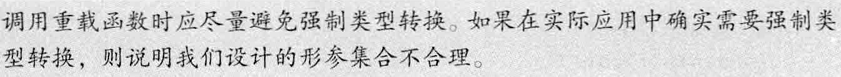
1、找出所有的候选函数——同名函数并且在调用点声明可见。

2、从候选函数中找出所有的可选函数——实参数量与形参数量相同且类型相同或能转换为形参的类型。

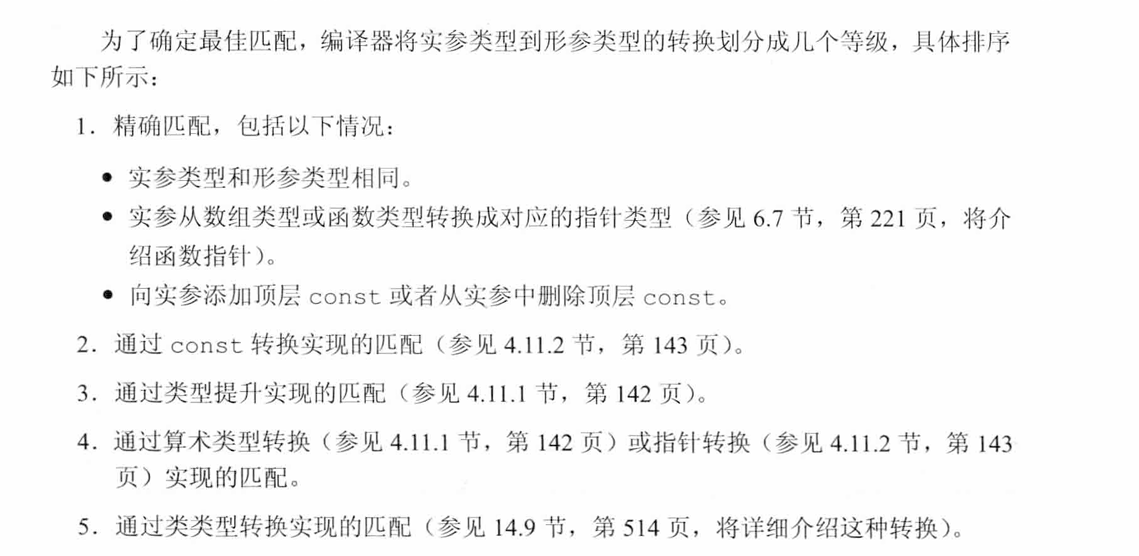
3、从可行函数中找出最佳匹配。如果不存在最佳匹配，会报错“二义性”。

ps：“最佳匹配”的定义：





寻找最佳匹配的原则：

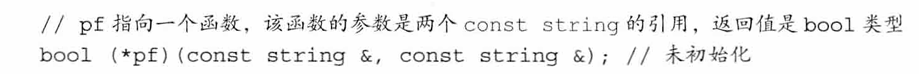


ps：2.将非常量实参转换为对应的底层const实参

3.隐式转换，范围小的类型提升为范围大的能完全包含原类型的类型

函数指针：





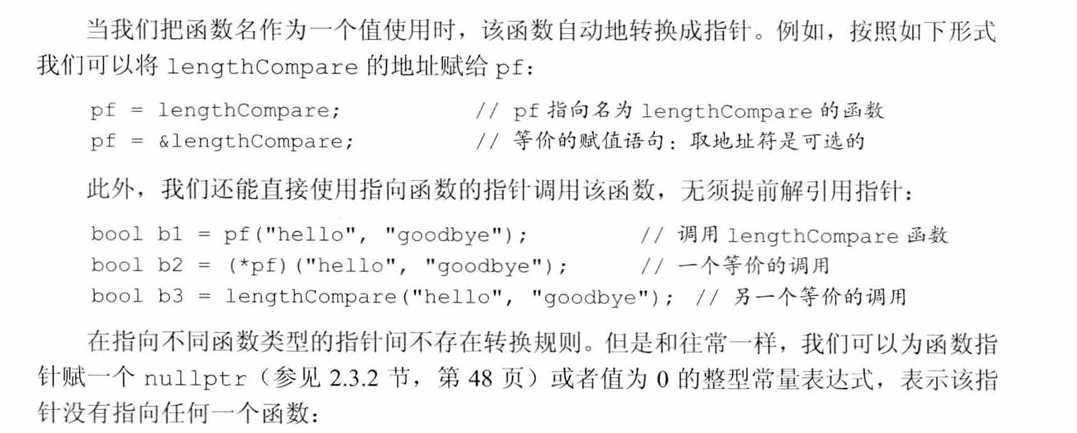
不可使用尾置类型：auto pf -> bool (\*)(const string&, const string&) ，函数指针的声明必须使用传统形式

image.png

image.png

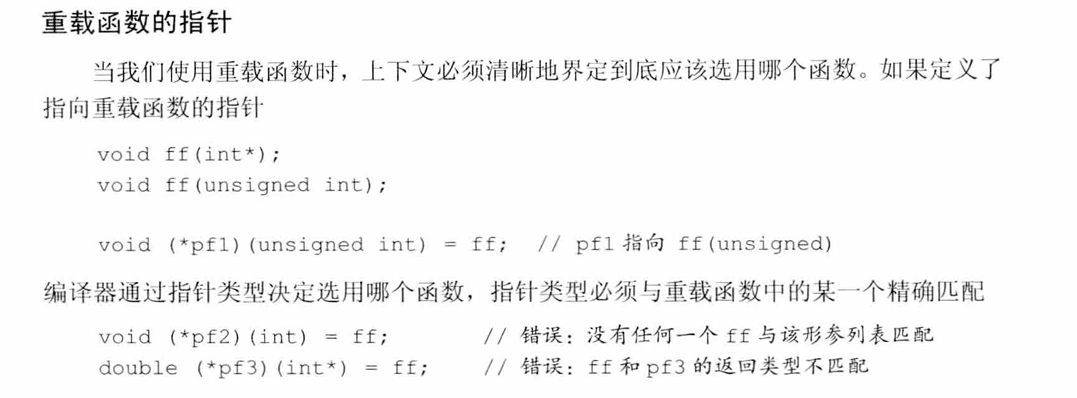
函数名必须用圆括号框起来后在里面加 \* ，理由同数组指针，如果不加则代表返回值是X类型的一个指针（上例中如果不加圆括号则代表返回值是一个bool类型指针）

如何使用函数指针：



重载函数的指针：

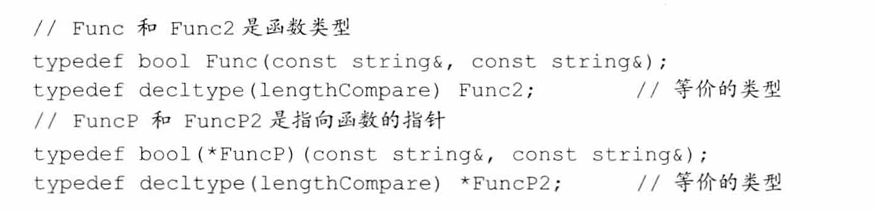
对于重载函数，声明的函数指针必须能够精确匹配这些重载函数中的一个。

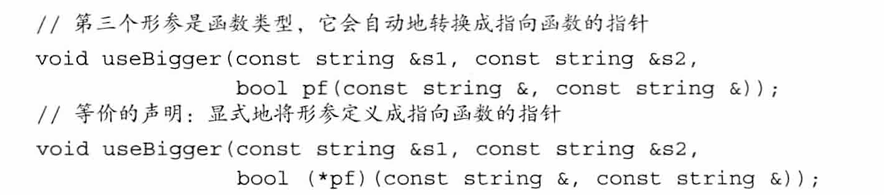


函数指针的别名：

函数指针作为形参：

函数不可以作为函数的形参，但是函数指针可以作为函数的形参（就应用场景上说，第一时间想到的是回调函数）可以通过下面书上例子一样的方法使用typedef定义别名使用。





函数指针作为返回值：

函数不可以作为函数的返回值，同样的函数指针也可以作为函数的返回值，不一样的是返回的函数可以使用尾置类型

