实验背景

在C/C++编程过程中，我们经常调用到一些库函数，以实现自己需要的功能，例如：字符串的比较，求长度，拷贝等，这些库函数是怎么生成的呢？我们可不可自己定义一些功能，并将它写成库函数呢？

函数库包含一些常用到的函数，供程序员进行调用。调用的时候把它所在的文件名用#include<>加到里面就可以了，函数库有静态函数库和动态函数库之分，静态函数库它在用户程序进行链接时，将各个目标文件(.obj)、运行时函数库(.lib)以及已编译的资源文件链接到一起，形成一个可执行文件(.exe)。

实验目的

1. 掌握静态函数库调用基本原理。
2. 掌握静态函数库的创建。
3. 掌握静态函数库的调用。

实验分析

如何通过调用静态函数库的方法，实现字符串的求长度，比较，拷贝，连接功能？

实验步骤

一、创建静态函数库文件

1. 创建Win32控制台的应用程序
   * 1. 设置项目名称StringFunction。
     2. 给出文件的路径：

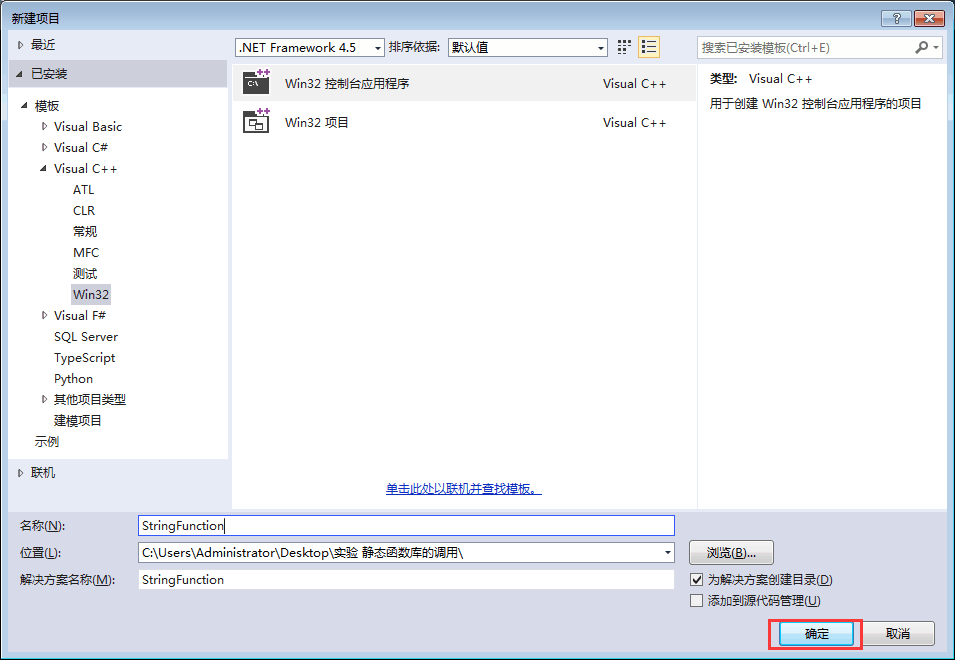


图1-1 创建Win32控制台的应用程序

1. 应用程序设置为静态库
   * 1. 使用Win32应用程序向导，点击下一步：



图1-2 使用程序向导

* + 1. 在对话框中，只选中静态库即可。

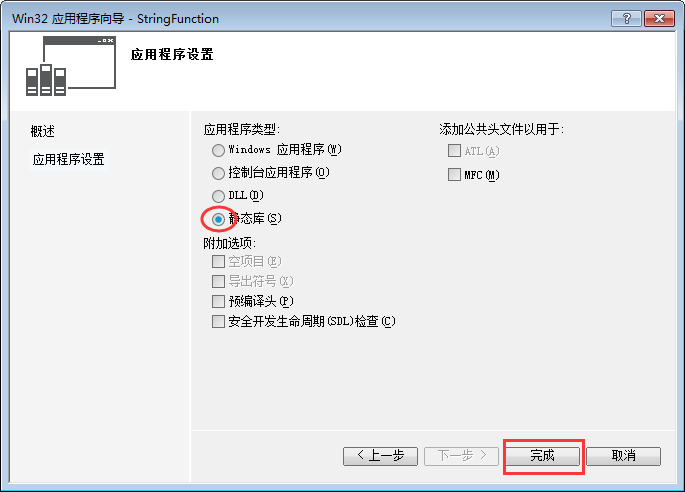


图1-3 静态函数库创建

1. 添加StringFunction.c文件和StringFunction.h文件
   * 1. 添加新建项StringFunction.c文件

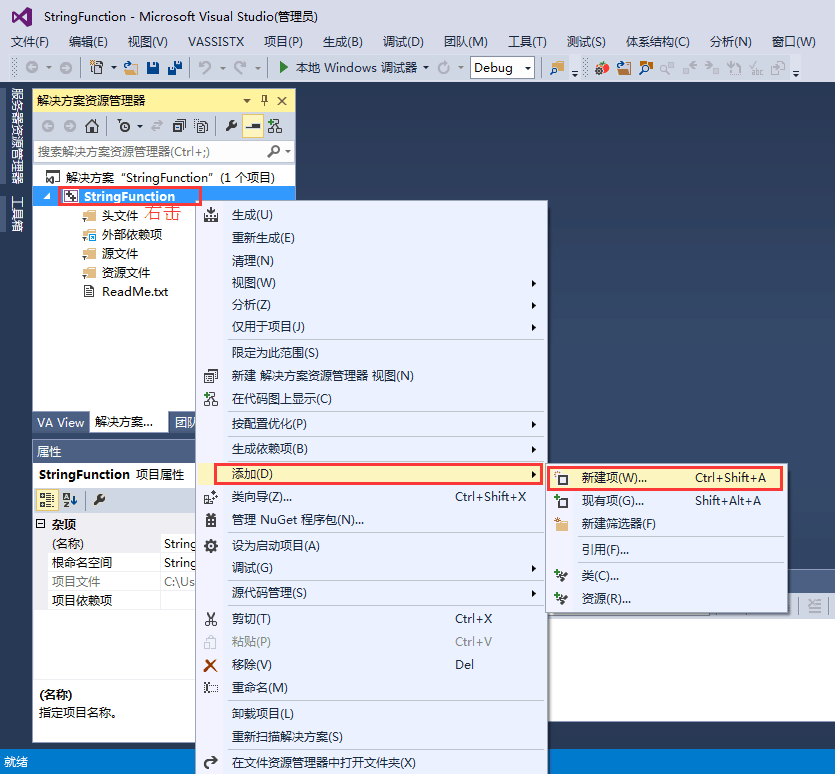


图1-4 选择添加新建项

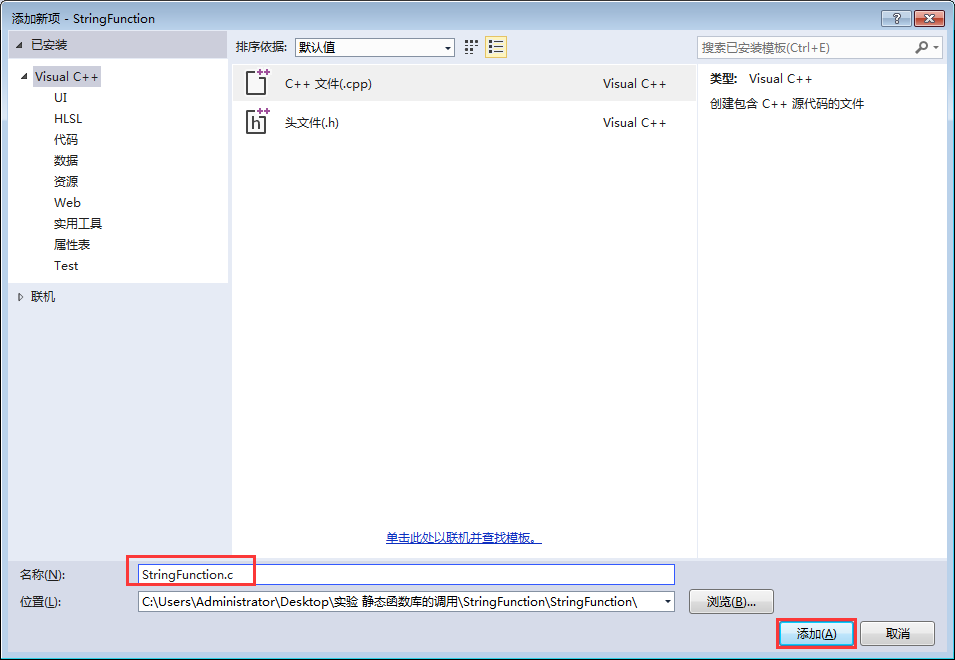


图1-5 添加StringFunction.c文件

* + 1. 添加新建项StringFunction.h文件

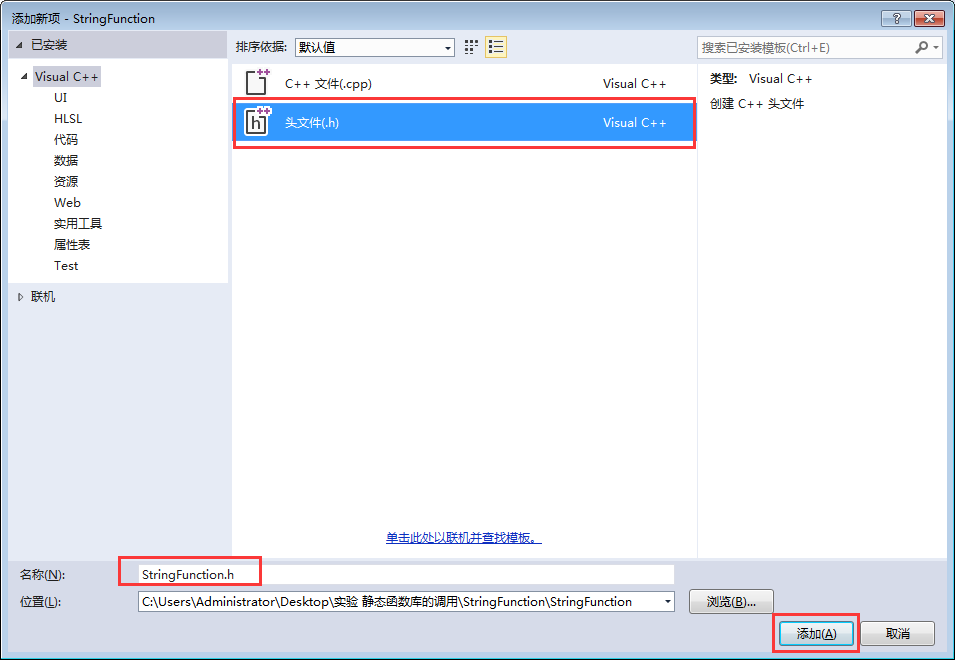


图1-6 添加StringFunction.h文件

1. 编写完成StringFunction.c文件和StringFunction.h文件，添加可以实现字符串的求长度，比较，拷贝，连接功能的函数。
   * 1. 编写新建项StringFunction.c文件，需要包含头文件StringFunction.h，并编写长度，比较，拷贝，连接功能函数的定义部分。

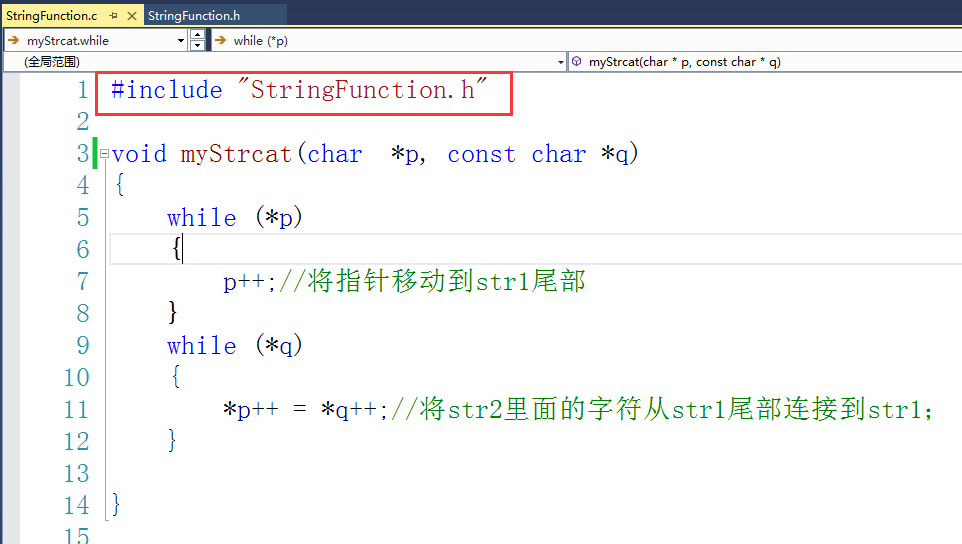


图1-7 编写StringFunction.c文件

* + 1. 在StringFunction.h文件中，添加函数的声明部分。

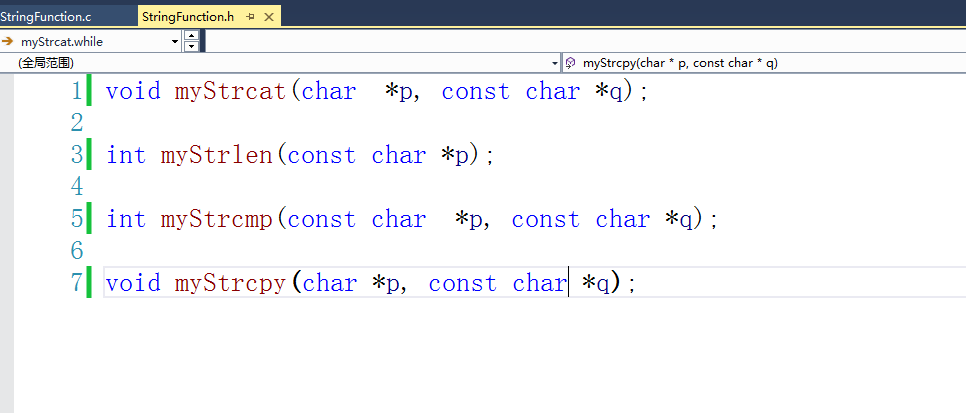


图1-8 编写StringFunction.h文件

1. 编译即可生Debug文件夹下生成静态函数库StringFunction.lib文件

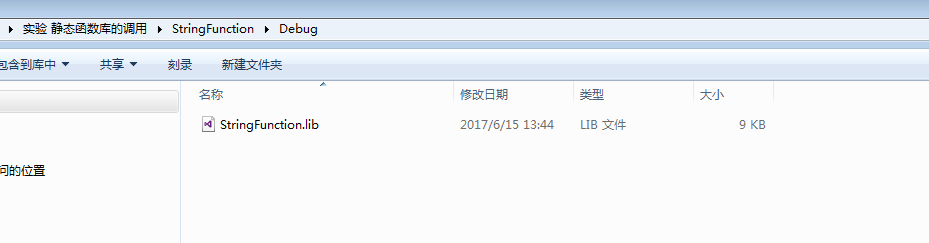


图1-9 lib文件生成

二、调用静态函数库文件

**目前，静态函数库文件创建完毕，接下来是如何调用静态函数库文件？**

1. 创建Win32控制台应用程序，并修改名称和添加路径。

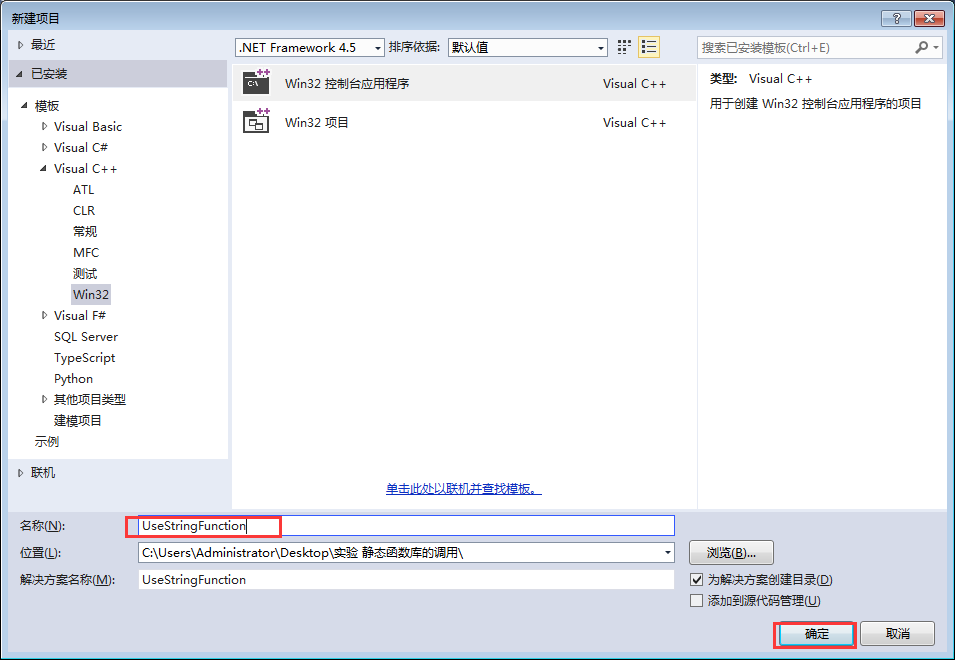


图2-1 创建调用函数库项目

1. 使用应用程序向导，单击“下一步”



图2-2 使用应用程序向导

1. 建立控制台应用程序的空项目

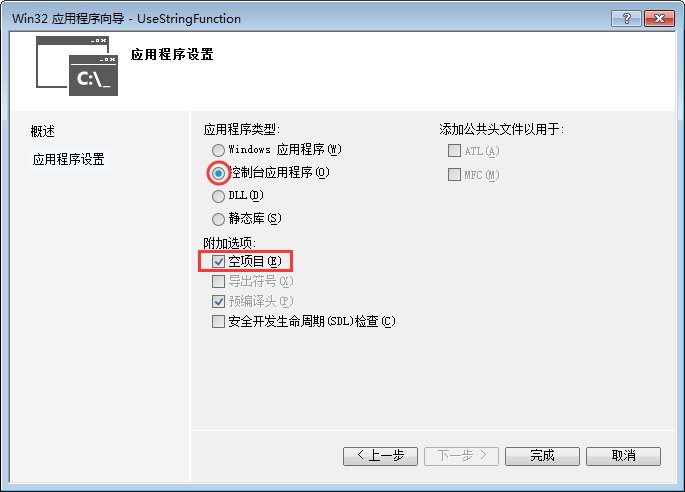


图2-3 控制台空项目

1. 右击项目名，添加新建项

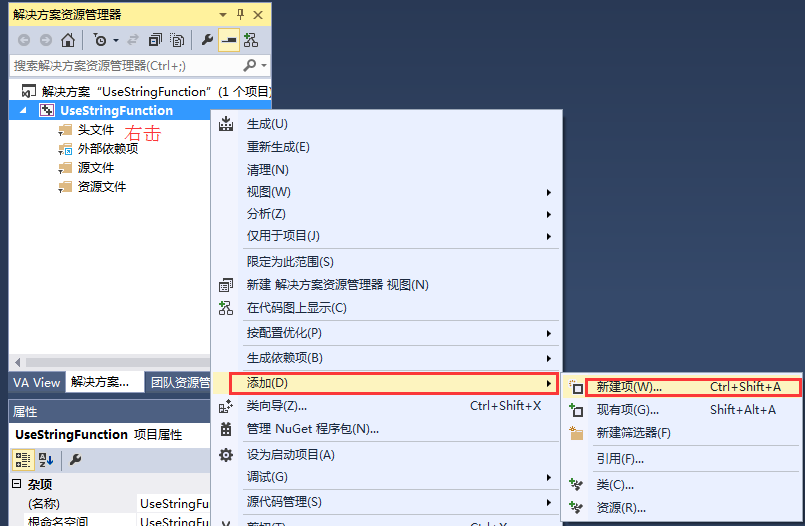


图2-4 添加新建项

1. 修改添加文件名，UseStringFunction.c

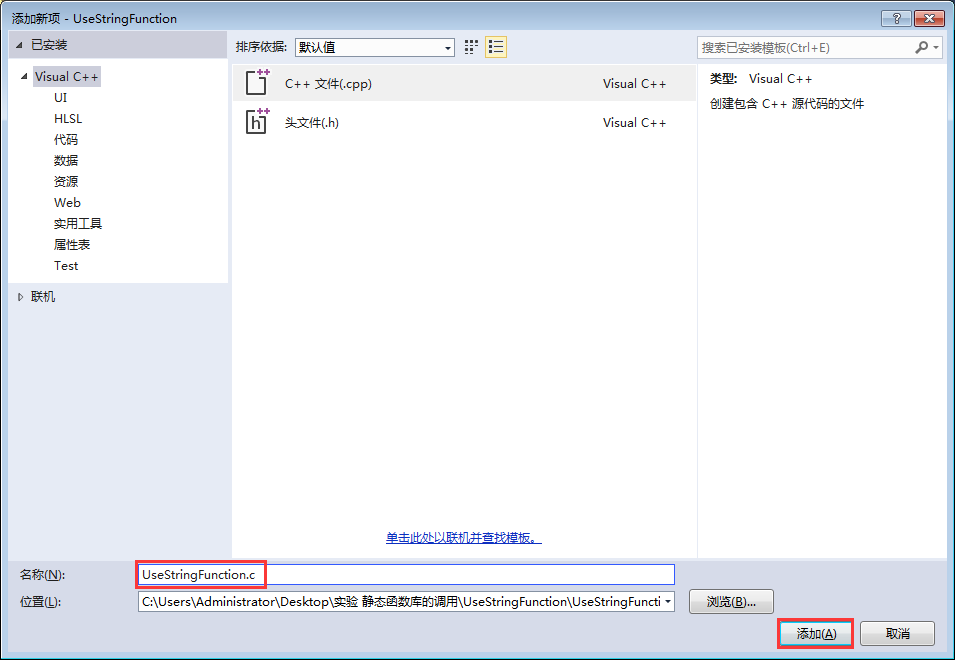


图2-5 添加UseStringFunction.c文件

1. 将生成的静态函数库文件（StringFunction.lib）和头文件（StringFunction.h）拷贝到UseStringFunction.c所在文件夹下；

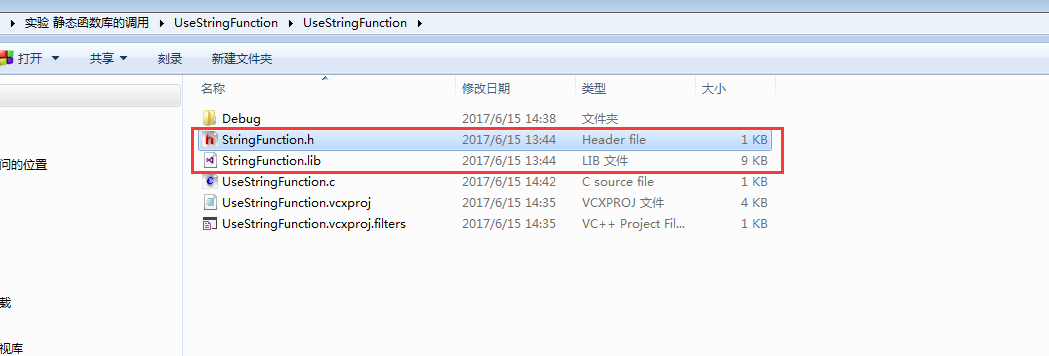


图2-6 拷贝静态函数库文件及其头文件

1. 调用库函数进行字符串 求长度，比较，拷贝，连接功能。
   * 1. 在UseStringFunction.c中编写调用静态函数库的代码，必须加入函数库的头文件和静态库文件，写法如下图：

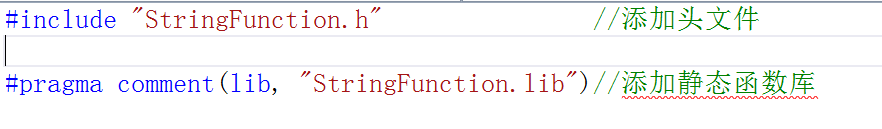


图2-7 添加静态函数库及其头文件

* + 1. 在main函数中，直接调用库函数即可。

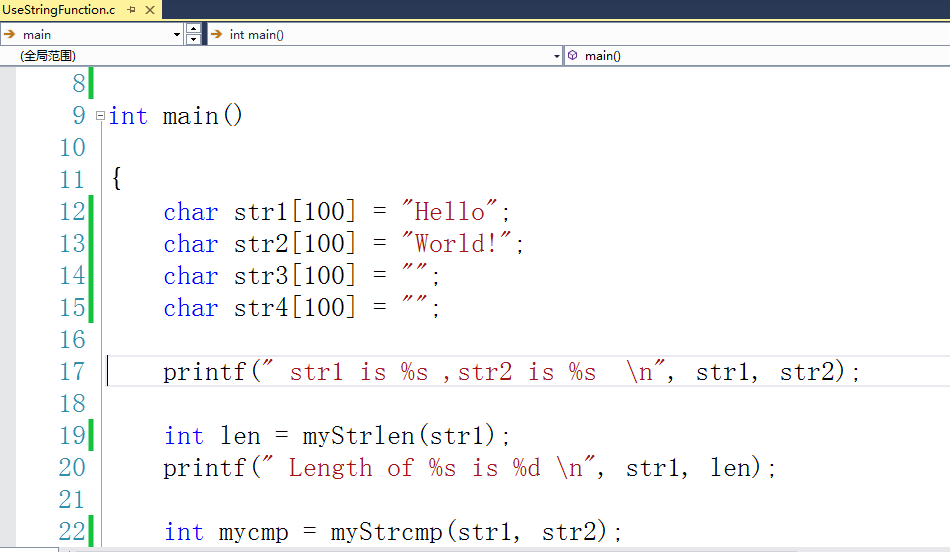


图2-8 调用库函数实现字符串运算

1. 编译并运行，得到结果如下：



图2-9 运行结果

扩展练习

1. 如何采用手动的方式添加函数库文件？
2. 静态函数库的名称可不可以改变，可不可以不放在调用它的函数文件夹下？如果改变了怎么调用？

实验思考

1. 在不知道静态函数库中的库函数的情况下，能调用库函数吗？
2. 静态调用函数库的项目，在生成\*.exe文件之后还需不需要函数库文件（lib）？