**Windows原理与应用课程实验报告**

**实验三 动态链接库DLL创建和调用**

二○二二年十月

**郑 重 声 明**

本人呈交的设计报告，是在指导老师的指导下，独立进行实验工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本设计报告不包含他人享有著作权的内容。对本设计报告做出贡献的其他个人和集体，均已在文中以明确的方式标明。本设计报告的知识产权归属于培养单位。

本人签名： 日期：

**目录**

[1 实验内容 1](#_Toc17476)

[2 相关理论 1](#_Toc30234)

[2.1 动态链接库的作用 1](#_Toc323)

[2.2 动态链接库的运行机制 2](#_Toc13046)

[2.3 函数参数与返回值 3](#_Toc17605)

[3 实验步骤 4](#_Toc31056)

[3.1 实现创建DLL程序A 4](#_Toc16824)

[3.2 实现winform窗体应用程序B 5](#_Toc2714)

[4 实验代码 6](#_Toc3350)

[5 实验结果 8](#_Toc25579)

[教师评语评分 9](#_Toc9084)

# 实验内容

使用C++创建DLL实现简单的功能，并在C#环境下调用该DLL。

实验步骤：

（1）准备工作：新建解决方案，在解决方案下新建两个项目.其中第一个项目为CreateDLL，其输出类型为类库；第二个项目为CallDLL，其输出类型为windows窗体应用(.NET Framework)或WPF应用(.NET Framework)；

（2）实现创建DLL程序A：创建C++自定义函数源文件和头文件，创建def文件，生成DLL动态链接库；

（3）实现winform窗体应用程序B：引用CreateDLL，实现winform窗体对C++动态链接库DLL中函数的调用。

实验要求：

（1）DLL功能一：计算输入数据的阶乘，需要判断输入的合法性；

（2）DLL功能二：计算2个输入数据a和b的差(a-b)，若a<b则需先交换a,b的值再计算；

（3）按下“调用DLL”按钮后，窗体winform程序调用DLL中的自定义函数，接收信息并显示内容。

# 相关理论

## 2.1 动态链接库的作用

动态链接库(Dynamic Link Library,DLL)是 Windows系统中非常有效的运行机制,通用功能的函数和数据聚集在DLL文件中。DLL减少程序文件的大小和对内存空间的需求,又使公共函数在多个应用程序之间共享。DLL文件是独立编译和测试的,多种编程语言和编译器都支持生成和使用DLL,不同编程语言生成的DLL函数还可互相调用。商业软件产品发布时产品由大量DLL文件组成,用户使用时不能知晓代码实现,可以保护软件开发者利益。

## 2.2 动态链接库的运行机制

DLL文件中的内容由全局数据、服务函数和资源组成,程序运行时使用LoadLibrary或者LoadlibraryEx函数加载DLL模块,Windows实现地址映射工作,例如一个 DLL文件被加载后在物理内存中只占一个固定区域,有多个进程使用同一个 DLL文件，Windows将这个DLL的内存地址空间通过地址映射后提供给各个进程,进程代码地址与 DLL映射后地址构成的是进程的虚地址空间,进程在自己的虚地址空间中好像是自己独自在使用这个DIL文件,使用DLL中的函数与程序自身的函数没有区别。DLL有自己的数据段,没有自己的堆栈,使用与调用它的应用程序相同的堆栈模式,它在运行时需要分配的内存是属于调用它的进程的,不同程序即使调用相同函数所分配的内存互相也不会影响，DLL函数中的代码所创建的任何对象(包括变量)都归调用它的线程或进程所有。

在 Windows环境中,每个进程都复制了自己的读/写全局变量,如果想要与其他进程共享内存,可以使用内存映射文件或者声明一个共享数据段。一个DLL在内存中只有一个实例,系统为每个DLL维护一个线程级的引用计数，一旦一个线程载入了该DLL,引用计数将会加1。而程序终止引用计数变为0(仅指运行时动态链接库),系统会释放DLL占用的虚地址空间。不指定DLL文件完整路径时,Windows将遵循下面的搜索顺序定位DLL:

1. 包含EXE文件的目录。
2. 进程的当前工作目录。
3. Windows系统目录。
4. Windows目录。
5. Path环境变量指定的目录。
6. ActiveX控件或COM组件的 DLL通过注册后使用。

程序调用动态链接库的方式有静态调用和运行时的动态调用两种方式。静态调用由编译系统完成对DLL的加载和卸载的编码,动态调用方式是运行时加载或卸载DLL,能有效地使用内存,是大型应用程序主要使用方式。使用静态链接方式生成的程序包含全部库代码,编译生成的文件体积大,少量的代码更改都会重新链接所有库文件。动态链接方式生成的程序代码短小,每次代码的变动只影响所在模块的编译和链接。动态调用时程序使用LoadLibrary( Windows平台的系统API)显式调用DLL,GetProcAddress方法获取函数指针,通过外部函数声明与程序其他函数一样使用, FreeLibrary释放动态链接库。程序启动时找不到目标DLL文件时,LoadLibrary函数返回空指针,系统会给出错误信息并终止程序运行。

动态链接具有下列优点:节约内存及减少文件操作；节省磁盘空间；方便软件升级。DLL带来的缺点包括会造成应用程序非独立性,程序受DLL模块的牵制,错误和危险的使用方式可能破坏 Windows的正常运行。

DLL文件的编制应符合调用方式的约定,包括参数进入堆栈的顺序,是函数还是应用程序负责清理堆栈以及寄存器中是否传递了任何参数,不同编程语言生成的DLL函数可实现兼容相互调用。

## 2.3 函数参数与返回值

函数设计时要考虑3个问题:输入参数、输出和函数功能,参数个数可以是0个或多个,还可在函数形参定义时采用params关键字定义可变数目参数。函数在定义时使用的参数声明称为形式参数,调用函数时使用具有真实值的实参。

获得函数运算结果有return方式和参数引用方式, return方式只能在函数结束后返回一个值。引用方式的形式参数添加ref或out关键字修饰,在函数内对参数变量的赋值在函数调用完成后仍然保留,调用有些Windows的内核API函数必须遵守引用参数(out)的方式返回运行结果。

C#规定函数参数的使用有3种:a.传值, b. ref和 c. out,函数在声明形式参数时无修饰符属传值方式,变量的值在函数执行后不受影响。传值的使用方式相当于实参的值被复制到函数中,不影响实参变量,引用方式中函数体内对实参的赋值被保留。传值方式和ref引用方式要求函数调用前实参必须已赋初值,out方式不要求函数调用前变量值初始化,变量的值在函数执行过程中被设置。

# 实验步骤

## 3.1 实现创建DLL程序A

使用VS2022新建动态链接库项目，项目名为CreateDLL。在“解决方案资源管理器”视图中，添加名为“myDLL.h”（文件名可以任意）的头文件（右键单击“头文件-添加-新建项-头文件”），输入代码如下：

#ifndef \_MY\_DLL\_FUNC\_H\_

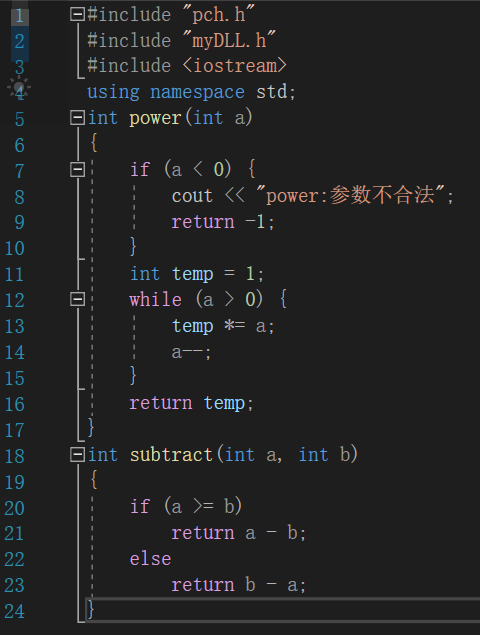
#define \_MY\_DLL\_FUNC\_H\_

int power(int a);

int subtract(int a, int b);

#endif

在“解决方案资源管理器”视图中，添加名为“myDLL.cpp”的C++文件（右键单击“源文件-添加-新建项-C++文件”），输入如下代码，其中power函数实现了求阶乘的功能，subtract函数实现了求差的功能。



在CreatDLL项目的“解决方案资源管理器”视图中，添加模块定义文件“Source.def”，输入如下代码：

LIBRARY

EXPORTS

power @1

subtract @2

**说明**：

.def文件的规则为：

　　(1)LIBRARY语句说明.def文件相应的DLL；

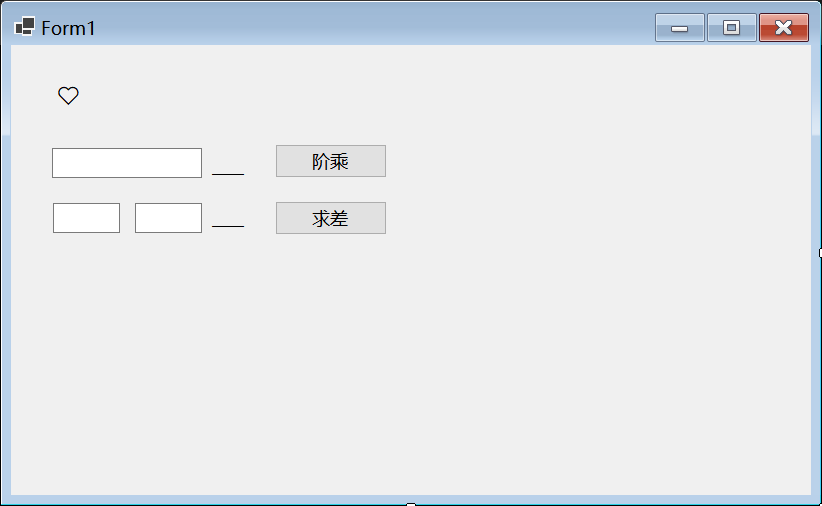
　　(2)EXPORTS语句后列出要导出函数的名称。可以在.def文件中的导出函数名后加@n，表示要导出函数的序号为n（在进行函数调用时，这个序号将发挥其作用）；

　　(3).def 文件中的注释由每个注释行开始处的分号 (;) 指定，且注释不能与语句共享一行。

点击项目的生成选项，通过提示的路径信息找到dll文件，之后复制到程序A的工作目录中备用。

## 3.2 实现winform窗体应用程序B

创建C# Windows窗体应用，添加所需控件如图：

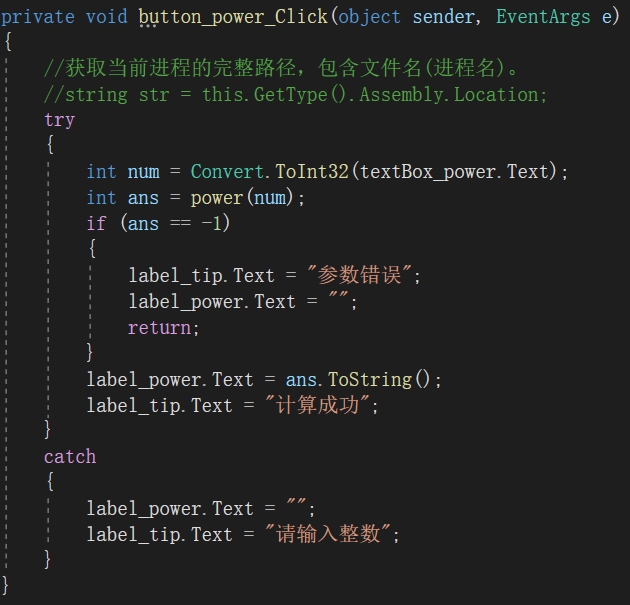


使用阶乘函数时先用如下语句引入对应函数（使用求差函数的方法与此类似），其中extern修饰符声明调用外部 API，使用 DllImport指定库文件和方法的参数格式。使用static关键字声明方法为静态。

[DllImport("CreateDLL.dll")]

public extern static int power(int a);

在相关事件中使用时先判断一下输入合法性，即数据类型是否正确，然后调用对应函数，更给界面显示结果。

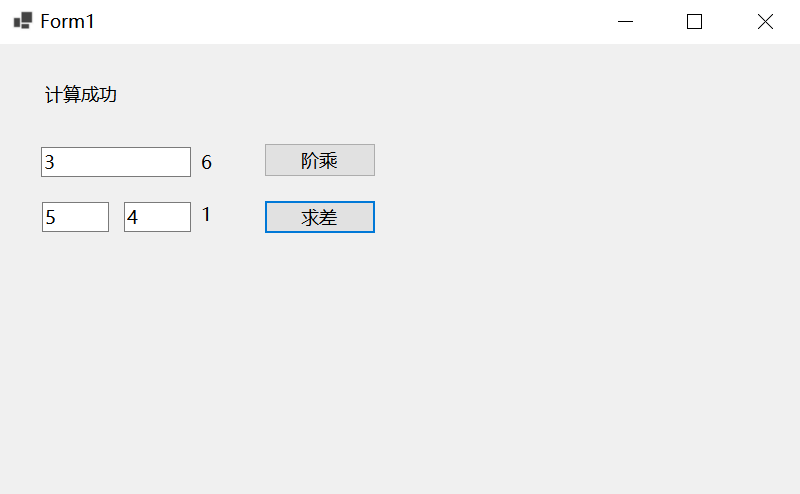


# 实验代码

窗体应用程序B部分代码:

|  |
| --- |
| using System.Runtime.InteropServices;  namespace CallDLL  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  }  [DllImport("CreateDLL.dll")]  public extern static int power(int a);  private void button\_power\_Click(object sender, EventArgs e)  {  //获取当前进程的完整路径，包含文件名(进程名)。  //string str = this.GetType().Assembly.Location;  try  {  int num = Convert.ToInt32(textBox\_power.Text);  int ans = power(num);  if (ans == -1)  {  label\_tip.Text = "参数错误";  label\_power.Text = "";  return;  }  label\_power.Text = ans.ToString();  label\_tip.Text = "计算成功";  }  catch  {  label\_power.Text = "";  label\_tip.Text = "请输入整数";  }  }  [DllImport(".\\CreateDLL.dll")]  public extern static int subtract(int a, int b);  private void button\_sub\_Click(object sender, EventArgs e)  {  try  {  int a = Convert.ToInt32(tb\_suba.Text);  int b = Convert.ToInt32(tb\_subb.Text);  label\_sub.Text = subtract(a, b).ToString();  label\_tip.Text = "计算成功";  }  catch  {  label\_sub.Text = "";  label\_tip.Text = "请输入整数";  }  }  }  } |

# 实验结果



点击按钮后，能够正常调用有关函数，显示对应结果。视频如下：



**教师评语评分**

评语：

评分：

评阅人： 年 月 日