**武汉大学计算机学院**

**本科生实验报告**

专 业 名 称 ：计算机科学与技术

课 程 名 称 ：Windows原理与应用

指 导 教 师 ：刘敏忠

学 号 ：2021302111169

姓 名 ：牛锴鹏

二○二三年十一月

**郑 重 声 明**

本人呈交的实验报告，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，本实验报告不包含他人享有著作权的内容。对本实验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。本实验报告的知识产权归属于培养单位。

本人签名： 牛锴鹏 日期： 2023.11.18

目录

[课 程 名 称 ：Windows原理与应用 1](#_Toc6166)

[摘要 4](#_Toc30744)

[实验内容 5](#_Toc19886)

[1. 动态链接库DLL创建和调用 6](#_Toc30278)

[1.1 实验目的 6](#_Toc8674)

[1.2实验环境 6](#_Toc2709)

[1.3实验内容及步骤 6](#_Toc15065)

[1.3.1创建DLL（项目名：CreateNewDLL） 6](#_Toc9256)

[1.3.2. 创建WinForm应用程序（项目名：CallDLL） 8](#_Toc3675)

[1.4实验结果 10](#_Toc24591)

[1.5结论 11](#_Toc17780)

[2. 创建和调用自定义COM控件 12](#_Toc16959)

[2.1 实验目的 12](#_Toc20798)

[2.2实验环境 12](#_Toc15184)

[2.3 实验步骤与实现 12](#_Toc603)

[2.3.1 定义COM组件接口 IExpress 12](#_Toc32295)

[2.3.2 实现COM组件接口 IExpress 12](#_Toc13799)

[2.3.3 生成和注册COM组件 13](#_Toc29548)

[2.3.4 创建WinForm窗口程序并调用COM组件 13](#_Toc6655)

[2.4实验结果 15](#_Toc8336)

[2.5结论 16](#_Toc5871)

[教师评语评分 17](#_Toc27335)

## 摘要

本报告详细阐述了两个关键实验：创建和调用动态链接库（DLL）以及创建和调用自定义COM控件。这些实验重点关注在不同编程环境下的模块化编程实践，突出了跨语言编程的实用性和灵活性。我们综合总结了两个核心实验：动态链接库（DLL）的创建与调用，以及自定义COM控件的创建与使用。这些实验深入探讨了在C++和C#环境下进行的模块化编程实践，强调了跨语言编程的实用性和灵活性。首先，实验涉及了在C++中创建DLL，并在C#环境中通过WinForm应用程序调用它的过程，包括设计和实现了两个关键的功能：计算阶乘和计算两数之差。随后，实验转向了创建自定义COM控件，包括定义COM组件接口、实现该接口，并在WinForm程序中调用这些接口。这一系列实验不仅提升了对编程语言和软件开发工具的理解，还加强了跨语言集成在现代软件开发中的重要性认识。这些实验展现了动态链接库和组件对象模型的应用，以及在高级软件开发中运用这些技术的能力，强调了代码的模块化、接口的重要性和异常处理的必要性，为处理复杂软件系统开发中的挑战提供了宝贵的经验和见解。

**关键词：**

动态链接库 (DLL)

COM控件

跨语言编程

组件化编程

.NET Framework

接口实现

异常处理

## 实验内容

| **实验名称** | **实验内容** | **实验要求** |
| --- | --- | --- |
| 实验三：动态链接库DLL创建和调用，COM控件创建和调用 | 1、动态链接库DLL创建和调用  使用C++创建DLL实现简单的功能，并在C#环境下调用该DLL  （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。 | 1、业务逻辑要求：  （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。 |
|  | 2、COM控件创建和调用  创建自定义COM对象，并在C#环境下调用该COM对象 | 2、功能性要求：  （1）定义COM组件接口IExpress，包括抽象方法；  public interface IExpress  {  string minus(int a, int b);//返回值形如“9 = 23 - 14”  string divide(int a, int b);//若b为零，则返回“除零错误”；若b不为0，则返回整除表达式，形如“4 = 33 / 8”  }  （2）定义类实现COM组件接口IExpress；  （3）WinForm窗口程序调用COM组件，显示结果信息。 |

## 动态链接库DLL创建和调用

### 1.1 实验目的

* 创建一个C++动态链接库（DLL），实现特定的函数功能。
* 在C#环境中通过WinForm应用程序调用这个DLL。

### 1.2实验环境

* 开发环境：Visual Studio
* 编程语言：C++（用于DLL），C#（用于WinForm应用程序）
* 目标框架：.NET Framework

### 1.3实验内容及步骤

#### 1.3.1创建DLL（项目名：CreateNewDLL）

##### 1.3.1.1准备工作

创建新解决方案：在Visual Studio中，首先需要创建一个新的解决方案。这是一个包含所有相关项目的容器。

添加C++类库项目：在这个解决方案中，添加一个动态链接库（DLL）项目。这个项目将被生成之后会输出一个DLL文件，即动态链接库。

##### 1.3.1.2实现DDL：

创建C++函数源文件和头文件：需要创建两个C++文件：一个源文件（.cpp）和一个头文件（.h）。头文件声明了将在源文件中定义的函数。

头文件CreateNewDLL.h内容如下：

|  |
| --- |
| extern "C" \_declspec(dllexport) int \_\_stdcall Factorial(int number); extern "C" \_declspec(dllexport) int \_\_stdcall CalculateDifference(int a, int b); |

CreateNewDLL.h 的内容为DLL定义了可从外部访问的接口，并规定了其他程序如何与这些函数互动。这是创建可在不同编程语言之间互操作的DLL的关键步骤。

定义两个函数：**int \_\_stdcall Factorial(int number)**：这个函数接受一个整数作为输入，并返回其阶乘的结果。

代码如下：

|  |
| --- |
| int \_\_stdcall Factorial(int number)  {   if (number < 0)   {   throw std::invalid\_argument("负数不能进行阶乘");   }   if (number == 0 || number == 1)   {   return 1;   }     // 检查乘法前是否会溢出   int result = Factorial(number - 1);   if (result > INT\_MAX / number)   {   throw std::overflow\_error("阶乘结果超出了int的范围");   }     return number \* result;  } |

**int \_\_stdcall CalculateDifference(int a, int b)**：这个函数接受两个整数作为输入，并返回它们的差值。如果a < b，则在计算之前先交换这两个值。

代码如下：

|  |
| --- |
| int \_\_stdcall CalculateDifference(int a, int b) {  if (a < b)  {  int temp = a;  a = b;  b = temp;  }  int result = a - b;  return result; } |

**创建def文件：**.def 文件用于明确指定哪些函数是要从DLL中导出的。这确保了在DLL中定义的函数可以被外部程序调用。

代码如下：

|  |
| --- |
| LIBRARY CreateNewDLL EXPORTS  Factorial  CalculateDifference |

##### 1.3.1.3生成DLL：

一旦完成了代码的编写和配置，就可以构建项目，这将生成一个DLL文件。这个文件包含了您定义的函数，可以被其他程序引用和调用。

#### 1.3.2. 创建WinForm应用程序（项目名：CallDLL）

**准备工作：**

添加WinForm应用程序项目：在相同的解决方案中，添加一个新的C# Windows窗体应用程序。这个应用程序将用于调用之前创建的DLL。

**引用CreateNewDLL:**

将刚刚创建的DLL文件作为引用添加到WinForm应用程序中。这样，应用程序就可以访问并调用DLL中定义的函数了。

代码如下：

|  |
| --- |
| public static class NativeMethods {  [DllImport(@"E:\study\C\_Sharp\Win\_DLL\x64\Debug\CreateNewDLL.dll")]  public extern static int Factorial(int number);   [DllImport(@"E:\study\C\_Sharp\Win\_DLL\x64\Debug\CreateNewDLL.dll")]  public extern static int CalculateDifference(int a, int b); } |

通过 NativeMethods 类和 DllImport 属性，C#程序能够调用存储在 CreateNewDLL.dll 中的函数。这种方法提供了一种在托管代码（C#）和非托管代码（C++ DLL）之间进行互操作的有效方式。在这个特定的示例中，通过直接指定DLL的路径，C#应用程序可以访问并执行C++ DLL中定义的 Factorial 和 CalculateDifference 函数。

**设计WinForm界面：**

设计一个界面，包括输入框供用户输入数据，以及一个按钮来触发DLL函数的调用。此外，还需要一个文本标签或类似控件来显示调用结果。

**实现DLL函数调用：**

在WinForm应用程序的代码中，编写函数来处理按钮点击事件。这些函数将调用DLL中的Factorial和Difference函数，并将结果显示在界面上。

代码如下：

|  |
| --- |
| private void button1\_Click(object sender, EventArgs e)  {   try   {   if (button1.Text == "计算阶乘")   {   int number = Convert.ToInt32(textBox1.Text); // 确保输入能转换为整数   int result = NativeMethods.Factorial(number);   labelResult.Text = $"阶乘结果: {result}";   }   else if (button1.Text == "计算数据差值")   {   int a = Convert.ToInt32(textBox1.Text); // 确保输入能转换为整数   int b = Convert.ToInt32(textBox2.Text); // 确保输入能转换为整数     int c\_result = NativeMethods.CalculateDifference(a, b);   labelResult.Text = $"数据差值: {c\_result}";   }   }   catch (FormatException fe)   {   MessageBox.Show($"输入格式错误: {fe.Message}");   }   catch (OverflowException oe)   {   MessageBox.Show($"输入数值过大或过小: {oe.Message}");   }   catch (Exception ex)   {   MessageBox.Show($"发生错误: {ex.Message}");   }  } |

在代码段中，实现了一个具有高鲁棒性的WinForm应用程序界面，允许用户通过文本框输入数据，并通过点击按钮触发DLL中的`Factorial`和`CalculateDifference`函数。界面设计简洁直观，包含用于输入的文本框、用于触发计算的按钮，以及用于显示结果的文本标签。程序能够根据`button1.Text`的值判断用户想要执行的操作类型，无论是进行阶乘计算还是差值计算。

数据输入部分包含了重要的验证步骤，通过`Convert.ToInt32`方法将用户的文本输入转换为整数。这个转换过程中自然包含了对输入的验证，如果输入不是有效的整数，则会抛出`FormatException`异常。此外，为了防止数据溢出，如果输入的数字超出了`int`类型的范围，程序会抛出`OverflowException`异常。一旦确定了操作类型和验证了输入数据，相应的DLL函数被调用，计算结果通过`labelResult.Text`显示在界面上。这一过程中的任何异常，无论是格式异常、数据溢出异常还是其他类型的异常，都被程序妥善处理，并通过消息框向用户报告错误信息。

这种细致入微的设计不仅确保了应用程序能够正确执行用户请求的计算任务，还能在遇到不合法的输入或其他潜在的运行时错误时提供适当的反馈，极大地增强了程序的用户友好性和鲁棒性。

#### 1.4实验结果

**（1）阶乘函数（Factorial Function）**

阶乘函数的功能是计算输入数字的阶乘。当用户输入一个整数时，程序将计算并输出这个数字的阶乘值。例如，在C++编写的程序中，当用户输入数字5时，程序会计算出5的阶乘（5!），即120，并在屏幕上显示结果。这一过程通常在控制台环境中展示，为用户提供了直观的数值计算结果。

初始化界面及运行结果如下：

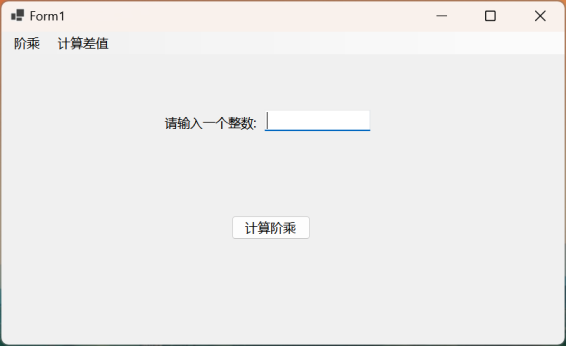
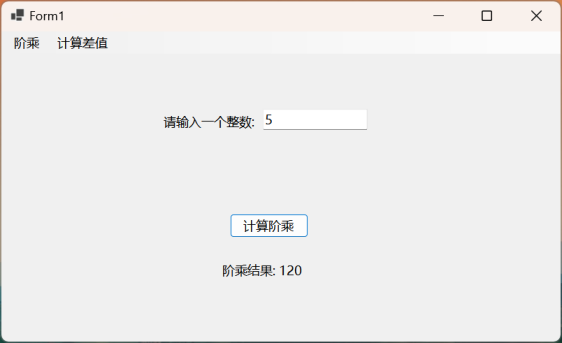
 

图1：初始化界面 图2：运行结果

输入错误或溢出的出错如下：

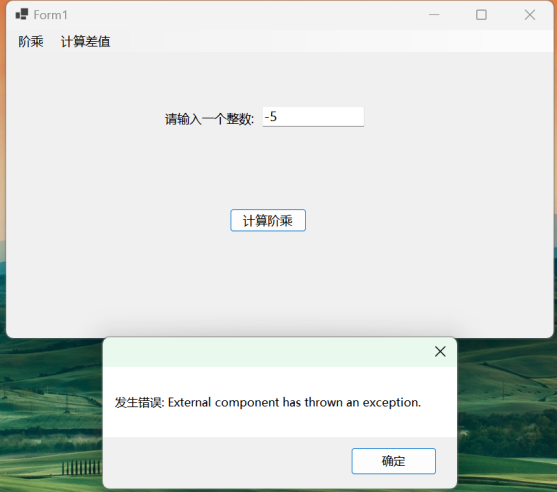
 

图3：输入不符合要求 图4：输入数值过大

**（2）差值函数（Difference Function）**

差值函数的目的是计算两个输入数字的差。在C#编写的WinForm应用程序中，用户通过图形界面输入两个整数a和b。程序将计算这两个数的差值，并在界面上显示结果。如果输入的第一个数a小于第二个数b，程序会先交换这两个数，然后再计算差值。例如，用户输入8和3，程序将显示两数之差为5。

初始化界面及运行结果如下：

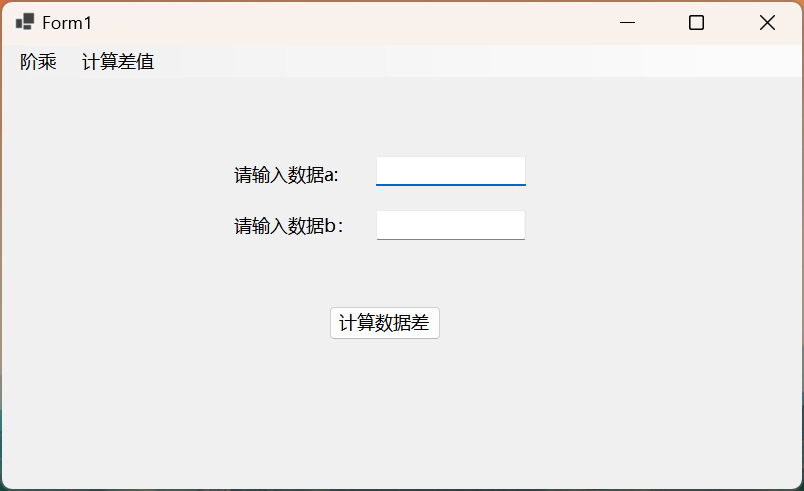
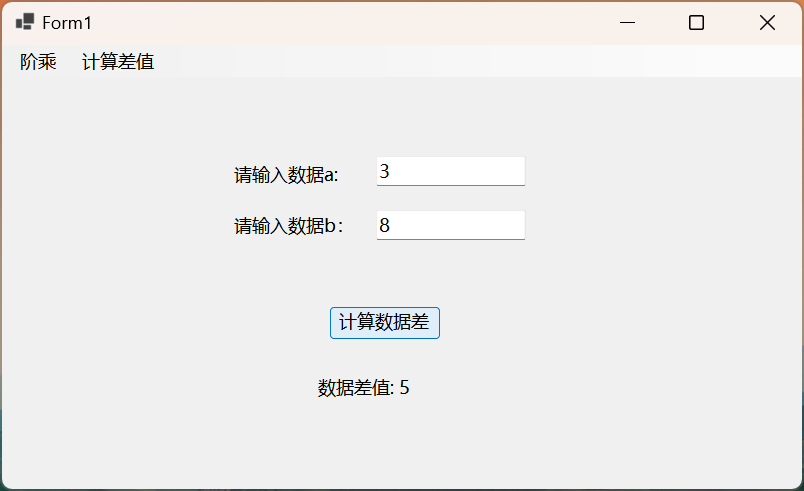
 

图5：初始化界面 图6：运行结果

### 1.5结论

在C#应用程序中调用C++ DLL的过程展示了跨语言编程的实用性和灵活性。通过这个过程，不仅成功实现了在C#环境下调用C++动态链接库的目标，而且加深了对动态链接库（DLL）在软件开发中的应用的理解。这种跨语言集成提供了一种有效的方式来利用不同编程语言的优势，实现了功能的复用和模块化编程，同时也展示了现代软件开发中语言间互操作性的重要性。此外，通过这个实验，也进一步验证了动态链接库在实现代码重用和分布式部署方面的有效性，这对于复杂软件系统的开发尤为关键。

### 创建和调用自定义COM控件

#### 2.1 实验目的

本实验的主要目的是学习如何在C#环境下创建自定义的COM控件，并在WinForm应用程序中调用该控件。通过这个过程，我们将深入理解COM控件的创建、接口的定义与实现，以及在客户端应用程序中如何利用这些控件。

#### 2.2实验环境

* 开发工具：Microsoft Visual Studio
* 编程语言：C#
* 应用类型：Class Library（用于COM控件）和WinForms（用于客户端应用程序）
* 目标框架：.NET Framework

#### 2.3 实验步骤与实现

##### 2.3.1 定义COM组件接口 IExpress

首先，我们创建了一个名为 ExpressCom 的类库项目，并在其中定义了一个 IExpress 接口。这个接口包含了两个抽象方法 minus 和 divide，它们分别用于执行减法和除法操作，并返回特定格式的字符串。

|  |
| --- |
| public interface IExpress {  string Minus(int a, int b);  string Divide(int a, int b); } |

##### 2.3.2 实现COM组件接口 IExpress

接下来，我们在同一项目中定义了一个类 ExpressImpl，该类实现了 IExpress 接口。在这个类中，我们为接口声明的两个方法提供了具体的实现。Minus 方法返回两数相减的结果，而 Divide 方法则执行除法运算，并处理了除以零的情况。

|  |
| --- |
| public class ExpressImpl : IExpress {  public string Minus(int a, int b)  {  return $"{a - b} = {a} - {b}";  }   public string Divide(int a, int b)  {  if (b == 0) return "除零错误";  return $"{a / b} = {a} / {b}";  } } |

##### 2.3.3 生成和注册COM组件

·右键点击工程->属性->应用程序->程序集信息->“使程序集COM可见（M）”打上勾，点 击【确定】

·生成->“为COM互操作注册（C）”打上勾，保存

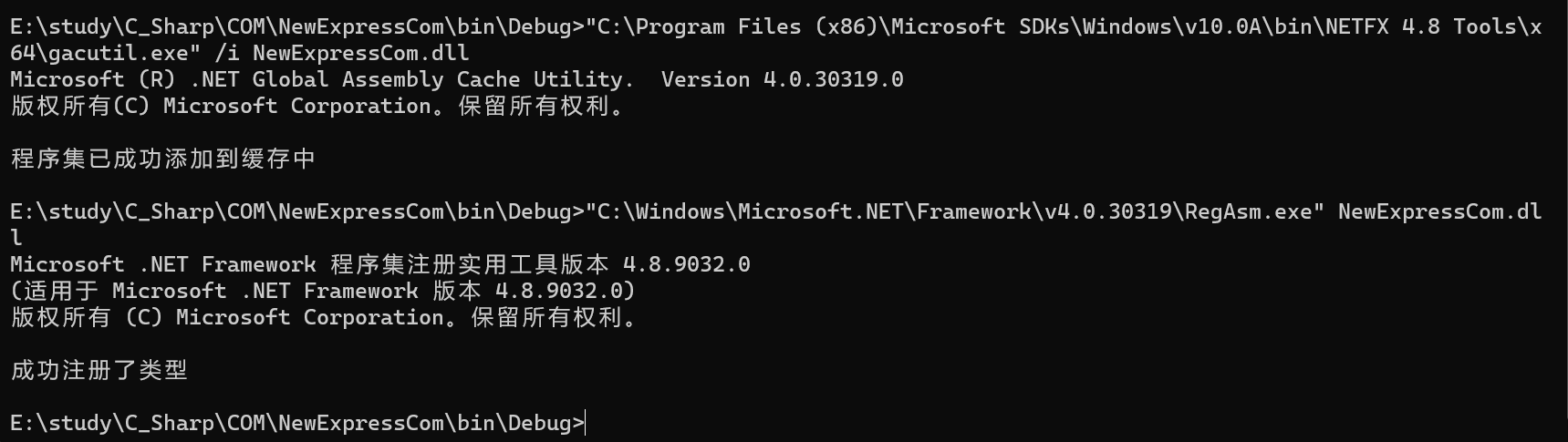
·生成dll文件

·注册COM组件至系统:

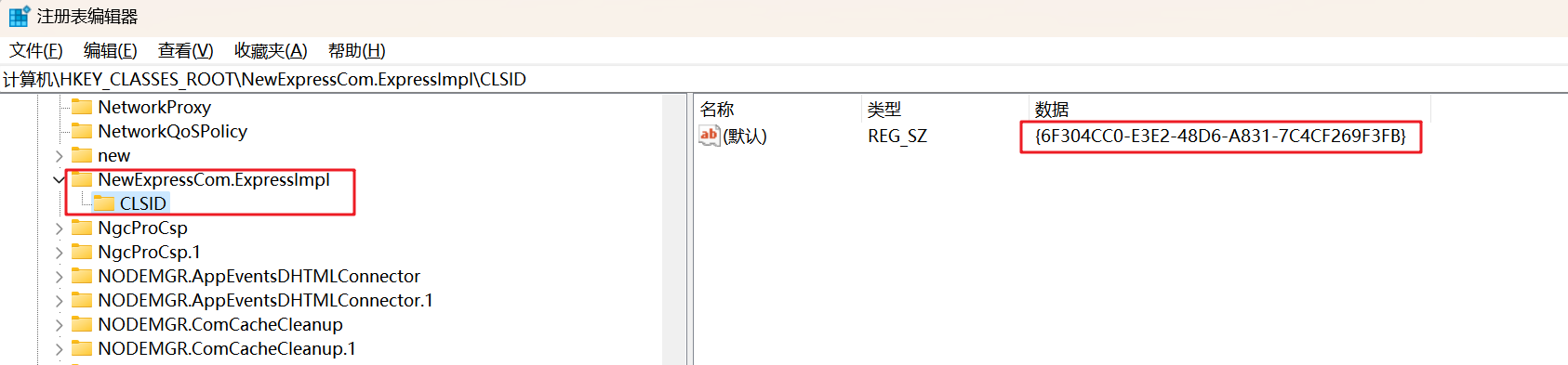
开始菜单->打开VS 2015自带CMD命令窗口（管理员权限）->定位至MyComToJava.dll 文件夹下 。

·执行：gacutil /i NewExpressCom.dll 添加dll至全局缓存 ；

·执行：regasm NewExpressCom.dll 注册dll至系统 。



·查看注册表regedit，HKEY\_CLASSES\_ROOT中是否有NewExpressCom.ExpressImpl项。如果有，则说明注册COM成功；如果没有，请重新注册



##### 2.3.4 创建WinForm窗口程序并调用COM组件

为了测试我们创建的 COM 控件，我们开发了一个名为 ExpressComClient 的 WinForm 应用程序。此应用程序具有直观的用户界面，包含以下元素：

* **两个文本框** (txtNumber1 和 txtNumber2)：用于输入要进行运算的数字。
* **两个按钮** (btnMinus 和 btnDivide)：分别用于执行减法和除法运算。
* **一个标签** (lblResult)：用于显示运算结果。

在实现中，我们首先使用 Activator.CreateInstance 和 Type.GetTypeFromCLSID 方法动态创建 COM 组件的实例。然后在按钮的点击事件处理程序中调用 COM 组件的方法。

**实例化 COM 组件**

首先，我们在表单的构造函数中初始化 COM 组件：

|  |
| --- |
| public partial class MainForm : Form {  private dynamic expressComObject;   public MainForm()  {  InitializeComponent();  InitializeComObject();  }   private void InitializeComObject()  {  Guid clsid = new Guid("6F304CC0-E3E2-48D6-A831-7C4CF269F3FB"); // 替换为COM组件的CLSID  Type comType = Type.GetTypeFromCLSID(clsid);  expressComObject = Activator.CreateInstance(comType);  } } |

InitializeComObject 方法: 这个方法用于创建 COM 组件的实例。我们使用 COM 组件的 CLSID （类标识符）来获取其 Type 对象，然后通过 Activator.CreateInstance 方法创建其实例。创建的对象被存储在 expressComObject 变量中，这是一个 dynamic 类型的变量，它允许我们在运行时动态地调用 COM 组件的方法。

**按钮点击事件处理**

在按钮的点击事件处理程序中，我们从文本框中获取输入的数字，然后使用 COM 组件的方法来执行相应的运算。

* **减法运算**：

|  |
| --- |
| private void btnMinus\_Click(object sender, EventArgs e) {  int number1 = Convert.ToInt32(txtNumber1.Text);  int number2 = Convert.ToInt32(txtNumber2.Text);  lblResult.Text = expressComObject.Minus(number1, number2); } |

这个方法首先从文本框中读取输入的数字，然后调用 COM 组件的 Minus 方法执行减法运算。计算结果随后被赋值给 lblResult 标签，以便在界面上显示。

* **除法运算**：

|  |
| --- |
| private void btnDivide\_Click(object sender, EventArgs e) {  int number1 = Convert.ToInt32(txtNumber1.Text);  int number2 = Convert.ToInt32(txtNumber2.Text);  lblResult.Text = expressComObject.Divide(number1, number2); } |

这个方法的工作原理与 btnMinus\_Click 相似，区别在于它调用的是 Divide 方法 来执行除法运算。

通过这些步骤，我们在 WinForm 应用程序中成功地集成了 COM 组件，使得用户能够通过简洁的界面与 COM 组件交互，执行基本的数学运算。这不仅展示了 COM 组件的功能，还证明了它们在现代应用程序中的实用性。

#### 2.4实验结果

通过在WinForm客户端程序中输入不同的数字，我们成功地调用了COM组件的 Minus 和 Divide 方法，并在界面上展示了运算结果。我们输入第一个数字3和第二个数字8之后，分别运行结果如下所示：

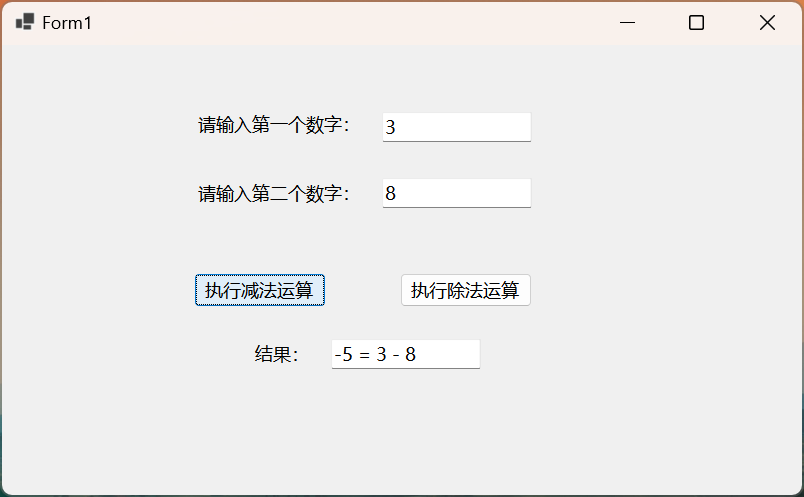
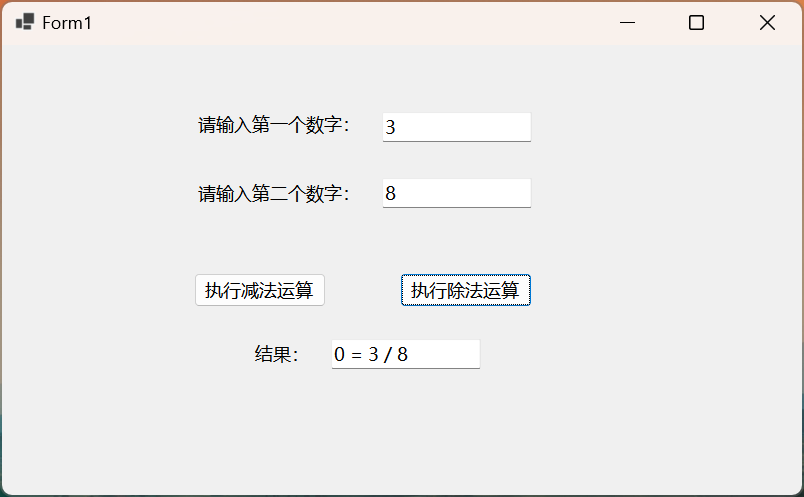
 

图5：减法操作 图6：除法操作

在下边的示例中，进行除法运算，其中一个输入框中输入了“10”，另一个输入了“0”。结果标签正确地显示了“除零错误”，这表示当尝试以零作为除数进行除法运算时，应用程序能够正确地显示错误消息。

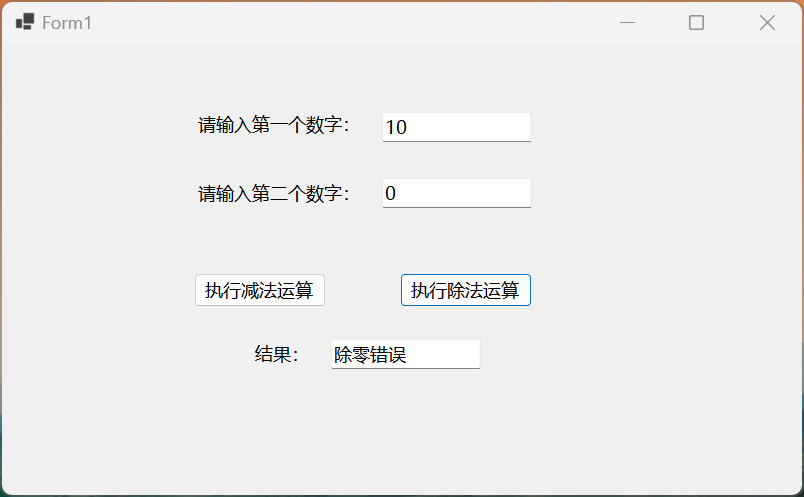


图7：除零错误

#### 2.5结论

在本实验中，我们成功地演示了如何在C#环境下创建和使用自定义COM控件，从而深入理解了组件对象模型（COM）的核心原理和应用。实验包括了COM接口的定义、实现，以及在WinForm应用程序中的集成和调用。通过定义 `IExpress` 接口并在 `ExpressImpl` 类中实现这一接口，我们展示了面向接口编程的重要性，强调了代码的模块化和可重用性。进一步地，通过使用 `regasm` 工具注册COM组件，并在WinForm客户端应用程序中引用这些组件，我们展现了COM组件在Windows系统中的部署和使用方法。WinForm应用程序 `ExpressComClient` 的开发不仅提供了与用户交互的界面，还展示了如何在实际的客户端应用中有效地集成和利用COM组件。此外，我们的实验还特别强调了异常处理在提升用户体验中的重要性，特别是在处理如除零错误等特殊情况时的准确性和稳健性。通过这一系列的步骤和方法，本实验不仅加深了我们对面向对象编程、接口定义、组件注册和用户界面设计的理解，还增强了我们在.NET环境中进行高级软件开发的能力，为深入掌握Windows平台下的组件化编程提供了宝贵的实践经验。

教师评语评分

评语：

评分：

评阅人：

年 月 日

（备注：对该实验报告给予优点和不足的评价，并给出百分之评分。）