实验一、调用win32api操作注册表

一、实验内容

使用windows操作系统提供的DLL，实现对注册表的操作：

- 创建子项
- 打开子项
- 删除项
- 设置键值
- 删除键值

二、实验思路

Windows系统下与注册表相关的系统dll文件为advapi32.dll。

在本次实验中主要利用如下几个函数：

- RegCreateKey() 创建子项（key）
- RegDeleteKey() 删除子项（key）
- RegOpenKeyEx() 打开子项（key）
- RegCloseKey() 关闭子项（key）
- RegSetValueEx() 设置键值（value）
- RegDeleteKeyValue() 删除键值

在项目开头导入InteropServices

```
using System.Runtime.InteropServices;
```

之后通过dllimport导入上述函数，如下：

```
        [DllImport("advapi32.dll", EntryPoint = "RegCreateKey")]//创建
```

```
        public static extern int RegCreate(IntPtr hkey, string lpSubKey,out
IntPtr phkResult);

        [DllImport("advapi32.dll", EntryPoint = "RegDeleteKey")]//删除
        public static extern int RegDelete(IntPtr hkey, string lpSubKey);

        [DllImport("advapi32.dll", EntryPoint = "RegOpenKeyEx")]//打开
        public static extern int RegOpen(IntPtr h, string subkey, uint opt, int
desired,out IntPtr res);

        [DllImport("advapi32.dll", EntryPoint = "RegCloseKey")]//关闭
        public static extern int RegClose(IntPtr h);



        //设置Key值
        [DllImport("Advapi32.dll", EntryPoint = "RegSetValueEx")]
        private static extern int RegSetValue(IntPtr hKey, string lpValueName,
uint unReserved, uint unType, byte[] lpData, uint dataCount);

        //删除键
        [DllImport("Advapi32.dll", EntryPoint = "RegDeleteKeyValue")]
        private static extern int RegDeleteKeyValue(IntPtr hKey, string
lpSubKey, string lpValueName);
```

数据类型变换参考如下表格：

Unmanaged type in Wtypes.h	Unmanaged C language type	Managed class name	Description
HANDLE	void*	System.IntPtr	32 bits on 32-bit OS
, 64 bits on 64-bit OS.			
BYTE	unsigned char	System.Byte	8 bits
SHORT	short	System.Int16	16 bits
WORD	unsigned short	System.UInt16	16 bits
INT	int	System.Int32	32 bits
UINT	unsigned int	System.UInt32	32 bits
LONG	long	System.Int32	32 bits
BOOL	long	System.Int32	32 bits
DWORD	unsigned long	System.UInt32	32 bits
ULONG	unsigned long	System.UInt32	32 bits
CHAR	char	System.Char	Decorate with ANSI.
WCHAR	wchar_t	System.Char	Decorate with Unicode.
LPSTR	char*	System.String	Decorate with ANSI.
LPCSTR	Const char*	System.String	Decorate with ANSI.
LPWSTR	wchar_t*	System.String	Decorate with Unicode.
LPCWSTR	Const wchar_t*	System.String	Decorate with Unicode.
FLOAT	Float	System.Single	32 bits
DOUBLE	Double	System.Double	64 bits

2.1 创建子项事件处理

```
            IntPtr phk = IntPtr.Zero;
            int ret = RegCreate(currenthKey, textBox2.Text,out phk);
            if (ret == 0)
            {
                MessageBox.Show("创建成功！");

            }
            else
            {
                MessageBox.Show("创建失败！");
            }
            RegClose(phk);
```

2.2 打开子项事件处理

```
            int ret = RegOpen(currenthKey, textBox2.Text, 0, KEY_ALL_ACCESS, out
IntPtr phkResult);
            if (ret == 0)
            {
                MessageBox.Show("打开项成功！");

            }
            else
            {
                MessageBox.Show("没有找到子项！");
            }
```

2.3 删除项事件处理

```
            if (0 == RegDelete(currenthKey, textBox2.Text))
            {
                MessageBox.Show("删除项成功！");
            }
            else
            {
                MessageBox.Show("删除项失败！");
            }
```

2.4 设置键值事件处理

```
            IntPtr pHKey = IntPtr.Zero;
            int lpdwDisposition = 0;
            int ret = RegCreate(currenthKey, textBox2.Text, out pHKey);
            uint REG_SZ = 1;
            byte[] data = Encoding.Unicode.GetBytes(textBox3.Text);
            int success = RegSetValue(pHKey, textBox1.Text, 0, REG_SZ, data,
(uint)data.Length);
            MessageBox.Show("创建键成功！");
            RegClose(pHKey);
```

2.5 删除键值事件处理

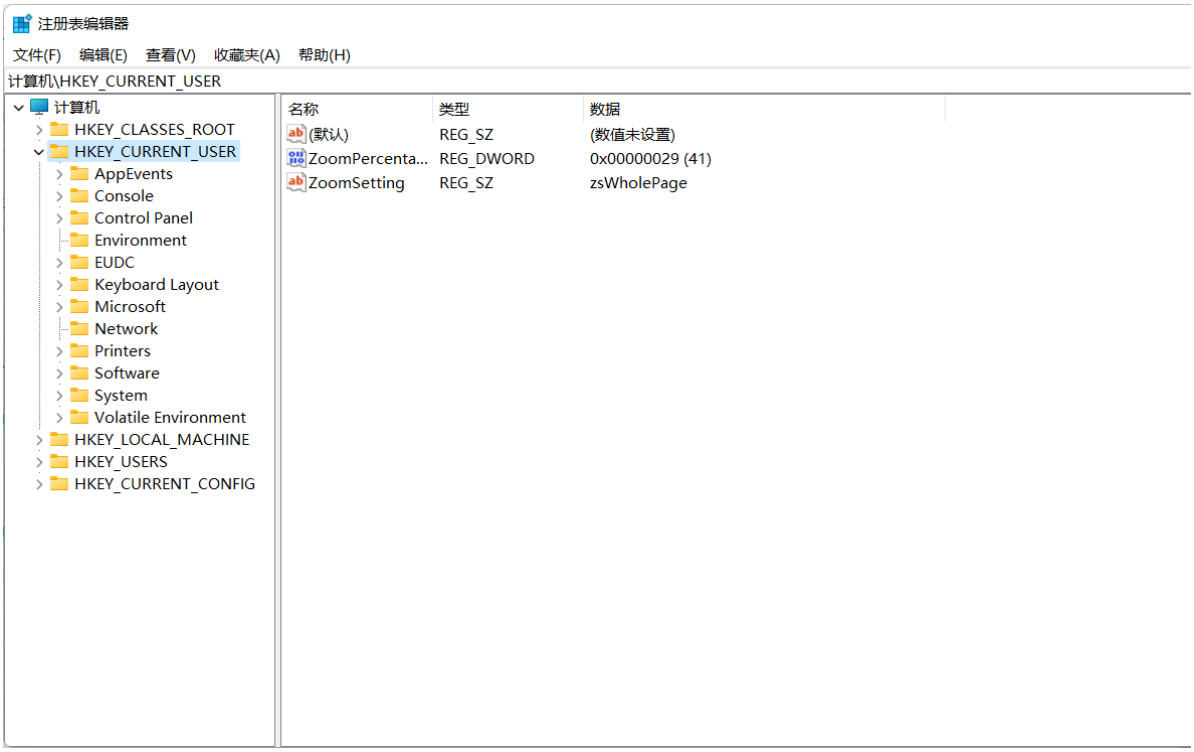
```
            if (0 == RegDeleteKeyValue(currenthKey, textBox2.Text,
textBox1.Text))
            {
                MessageBox.Show("删除键成功！");
            }
            else
            {
                MessageBox.Show("删除键失败！");
            }
```

2.6 UI设计

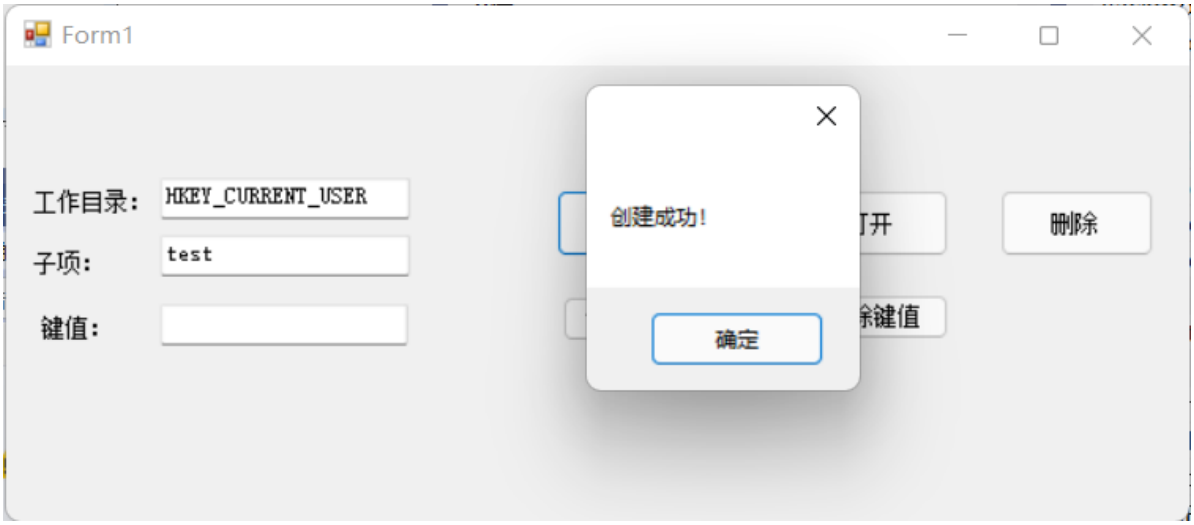
三、实验结果

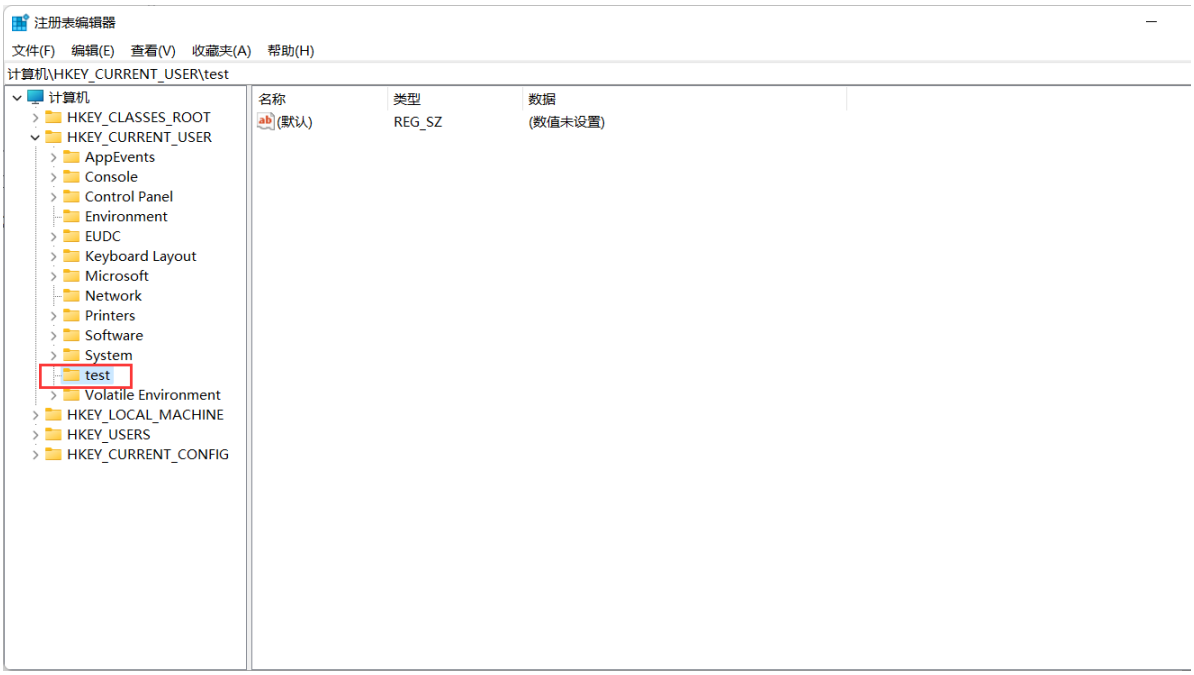
实验主要在HKEY_CURRENT_USER下操作。

实验初始状态：

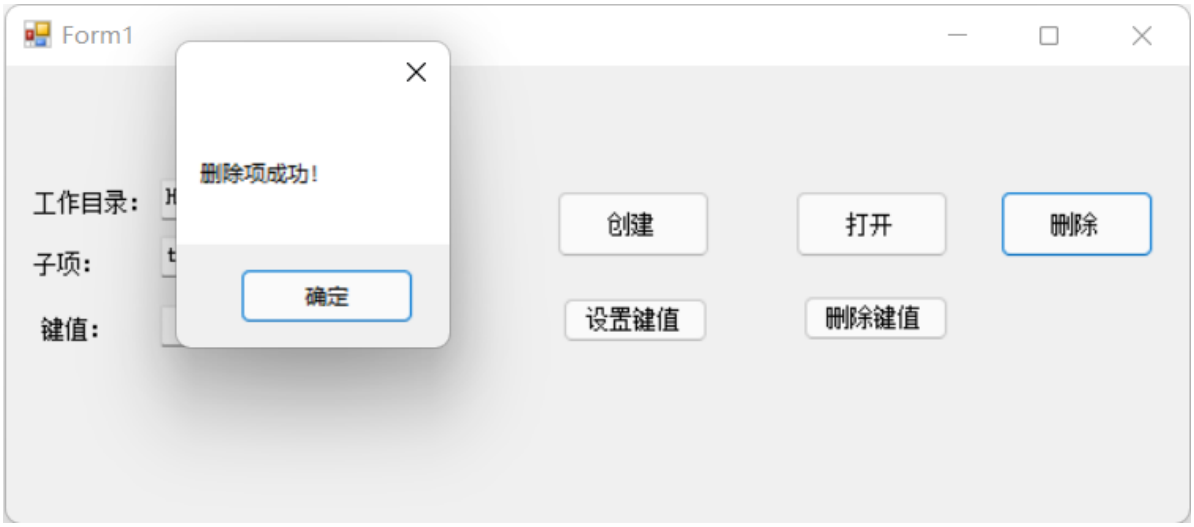


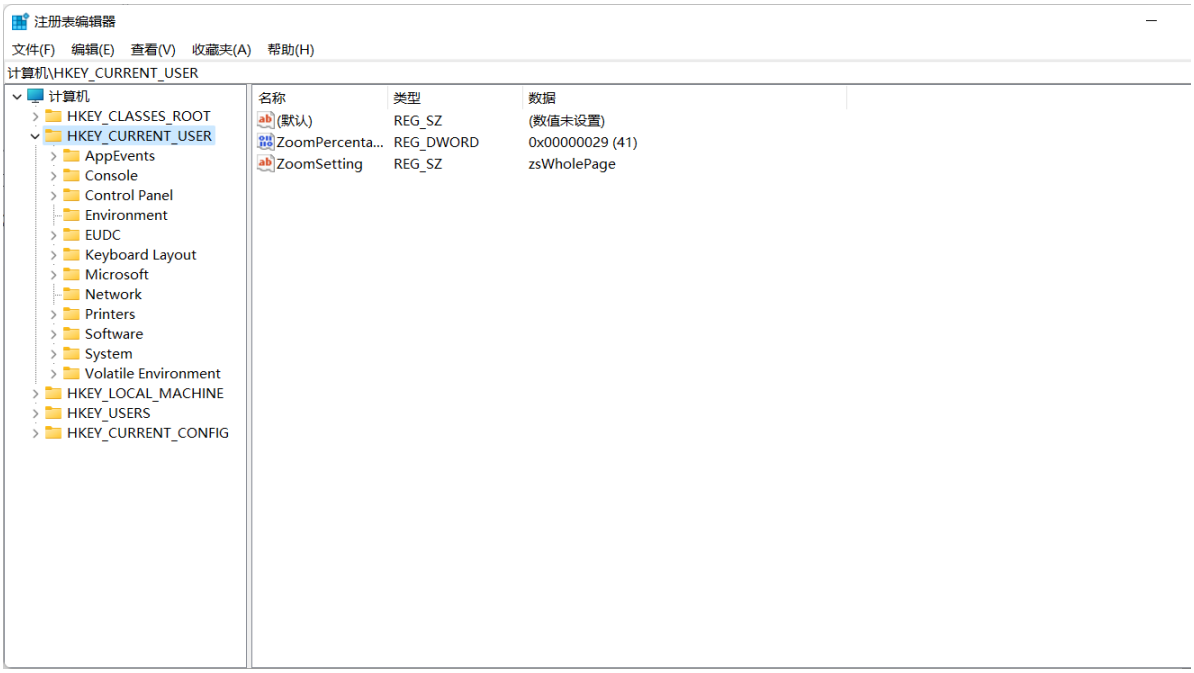
点击创建，如下：


Form1
工作目录:
HKEY_CURRENT_USER
子项:
test
键值:
创建成功!
确定
打开
删除
除键值

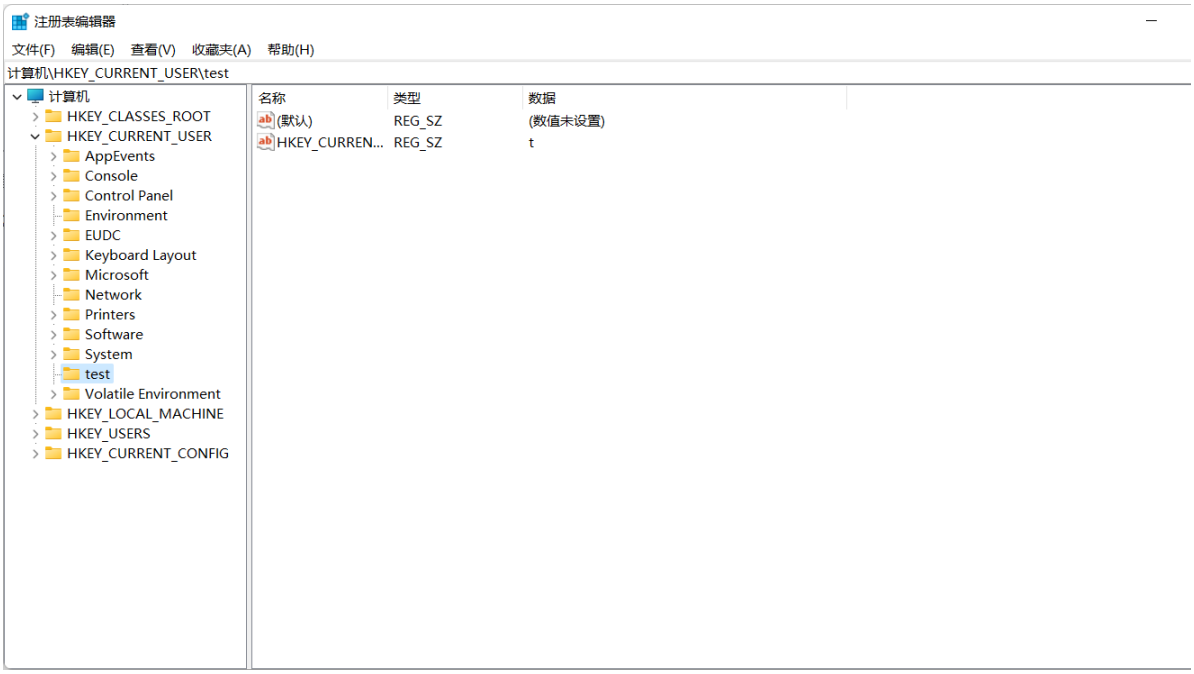

注册表编辑器
文件(F) 编辑(E) 查看(V) 收藏夹(A) 帮助(H)
计算机\HKEY_CURRENT_USER\test
计算机
HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
AppEvents
Console
Control Panel
Environment
EUDC
Keyboard Layout
Microsoft
Network
Printers
Software
System
test
Volatile Environment
HKEY_LOCAL_MACHINE
HKEY_USERS
HKEY_CURRENT_CONFIG
名称
类型
数据
(默认)
REG_SZ
(数值未设置)

点击删除，如下:

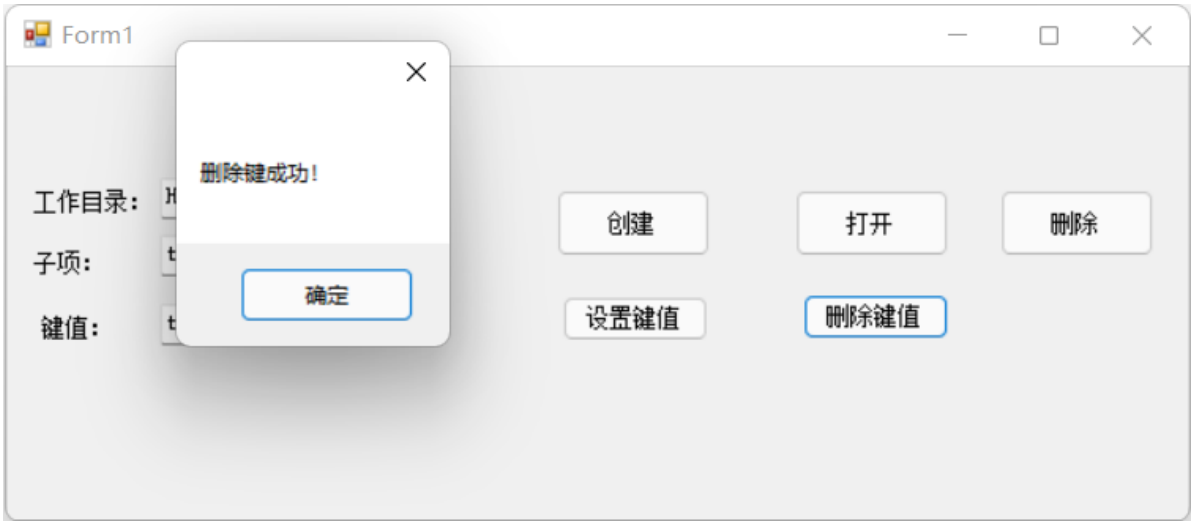

Form1
工作目录:
子项:
键值:
删除项成功!
确定
创建
打开
删除
设置键值
删除键值

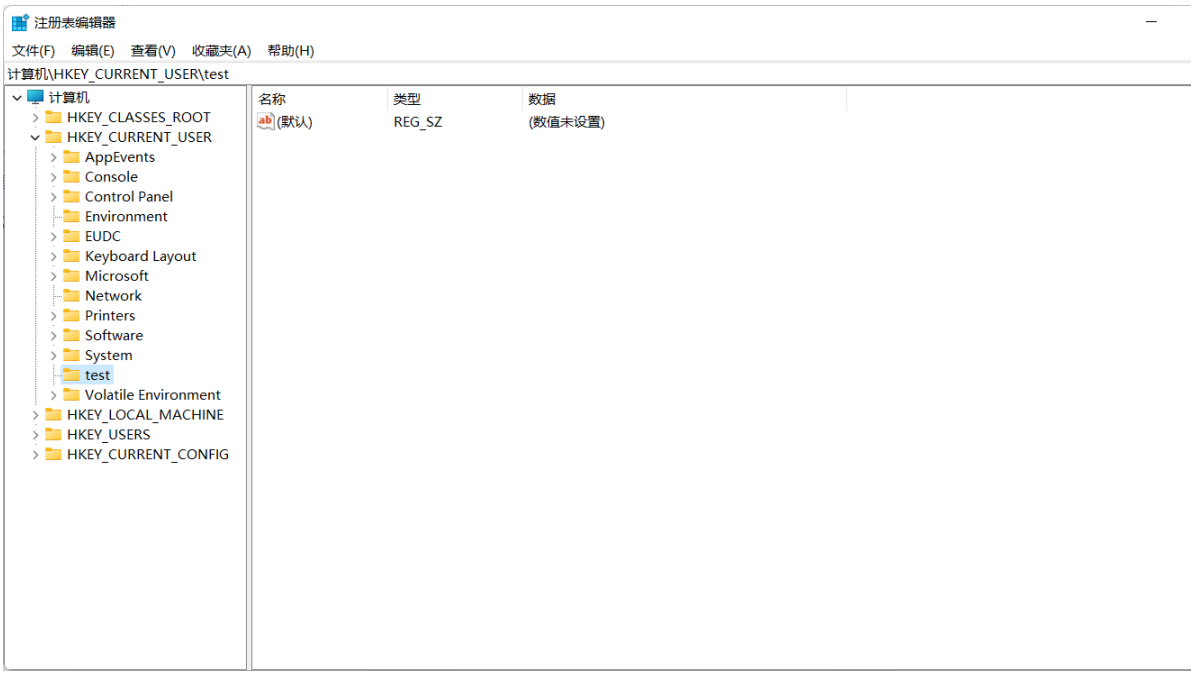


点击设置键值，如下：



点击删除键值，如下：





实验二、c++DLL调用

一、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。

二、实验思路

首先使用c++创建DLL，通过点击新建项目创建一个动态链接库，如下图：

之后添加对应的头文件和源文件，在头文件中编写如下代码：

```
#pragma once

extern "C" _declspec(dllexport) int _stdcall test01(int a);
extern "C" _declspec(dllexport) int _stdcall test02(int a, int b);
```

在源文件编写如下代码：

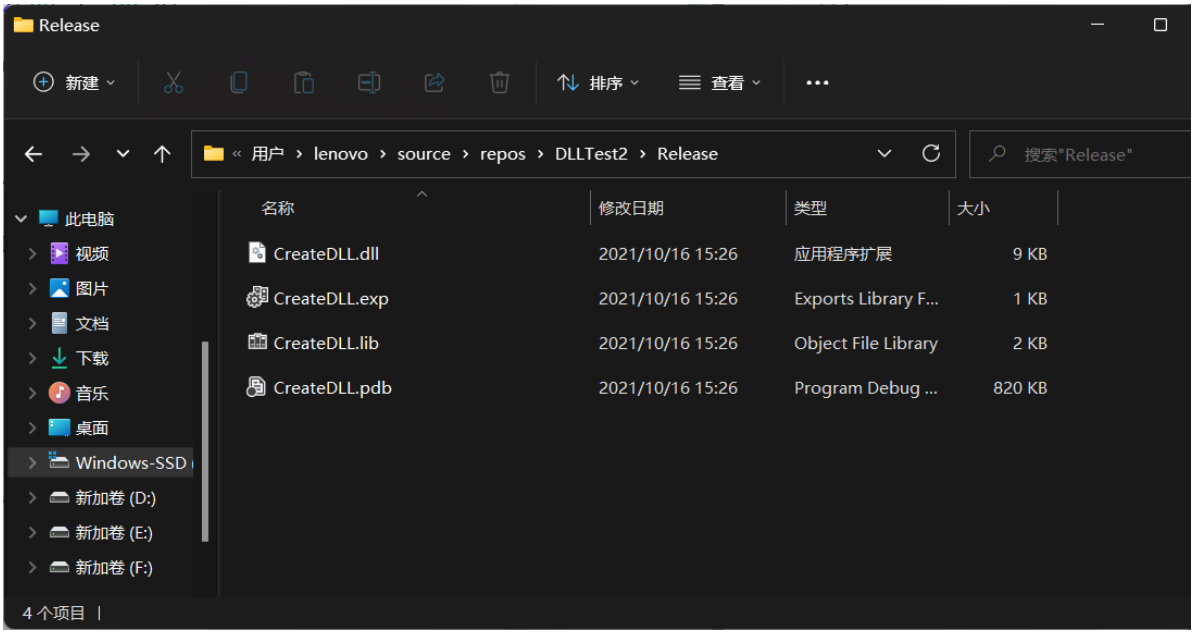
```
#include"Test.h"
#include"pch.h"
int _stdcall test01(int a)
{
    if (a < 0)  return -1;
    else if (a > 31)return 0;
    else if (a == 0)return 1;
    else return(a * test01(a - 1));
}
int _stdcall test02(int a, int b)
{
    if (a >= b)return (a - b);
    else return (b - a);
}
```

两个测试函数分别是做阶乘和求两数之差。

之后添加定义文件，添加如下代码：

```
LIBRARY"CreateDLL"
EXPORTS
test01 @1
test02 @2
```

最后在Release版本下生成解决方案，得到如下：



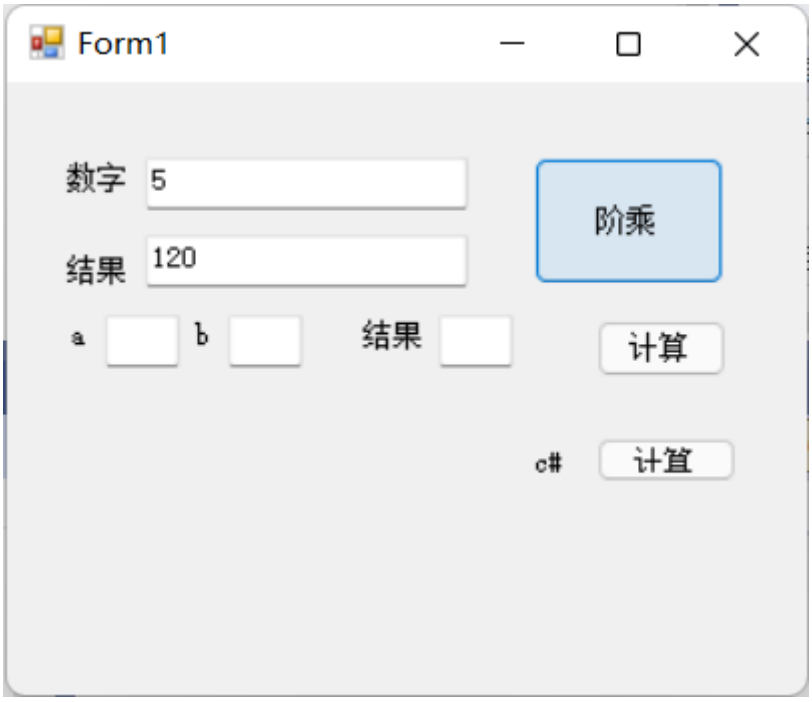
其次，创建C#工程对其进行调用。本次实验使用winform。

使用dllimport对dll导入，如下代码：

```
            [DllImport(@"../../../Release/CreateDLL.dll", EntryPoint = "test01",
SetLastError = true, CharSet = CharSet.Ansi, ExactSpelling = false,
CallingConvention = CallingConvention.StdCall)]
            public static extern int test01(int a);

            [DllImport(@"../../../Release/CreateDLL.dll", EntryPoint = "test02",
SetLastError = true, CharSet = CharSet.Ansi, ExactSpelling = false,
CallingConvention = CallingConvention.StdCall)]
            public static extern int test02(int a, int b);
```

三、实验结果

对于第一个测试函数，实现阶乘功能，如下图：



对于第二个测试函数，实现相差功能，如下图：

实验三、c#DLL调用

一、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。

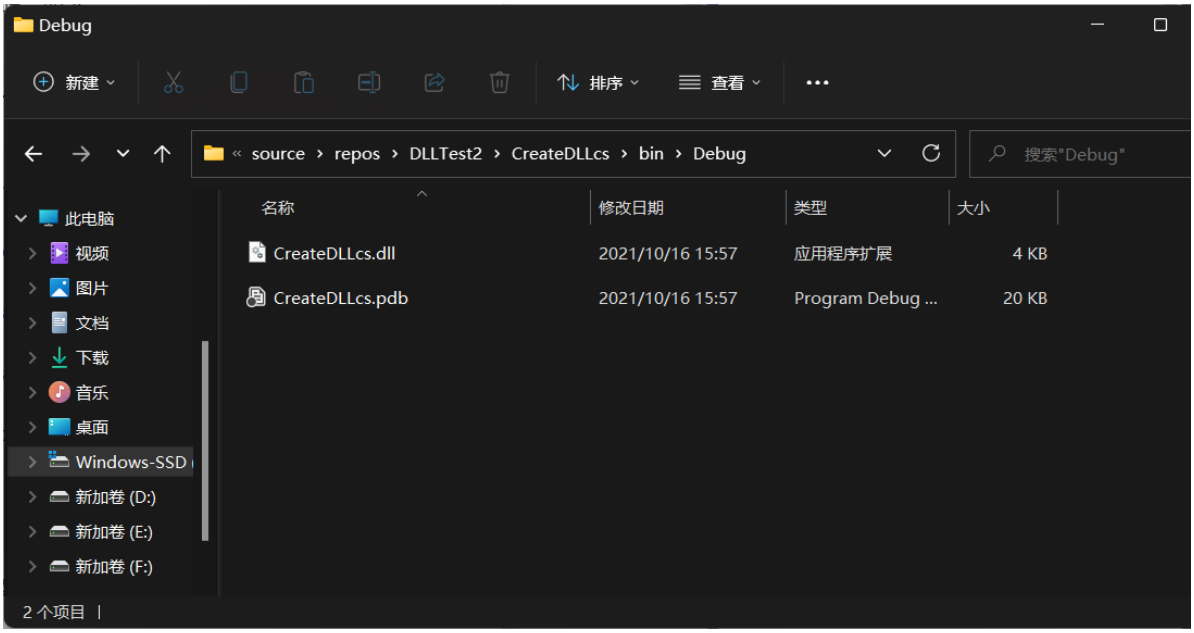
二、实验思路

首先使用C#编写dll类库。选择新建项目，如下图：

之后，在其中编写测试代码，如下：

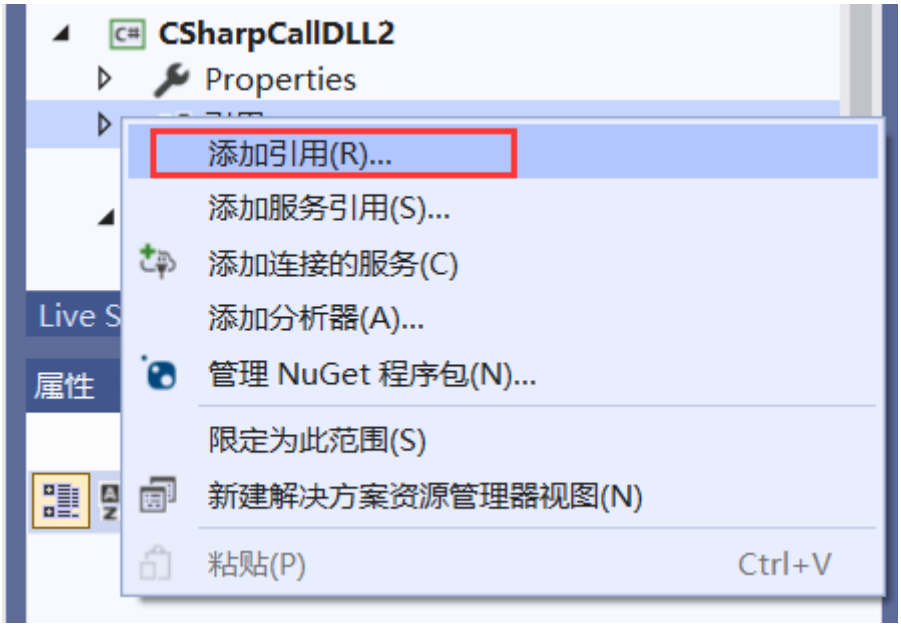
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CreateDLLcs
{
    public class Class1
    {
        public int add(int x, int y)
        {
            return x + y;
        }
    }
}
```

直接生成解决方案即可，如下图：



其次，在winform工程对其调用。

首先添加引用，如下



在浏览文件找到上一步生成的dll文件。

之后使用如下代码将其导入项目中：

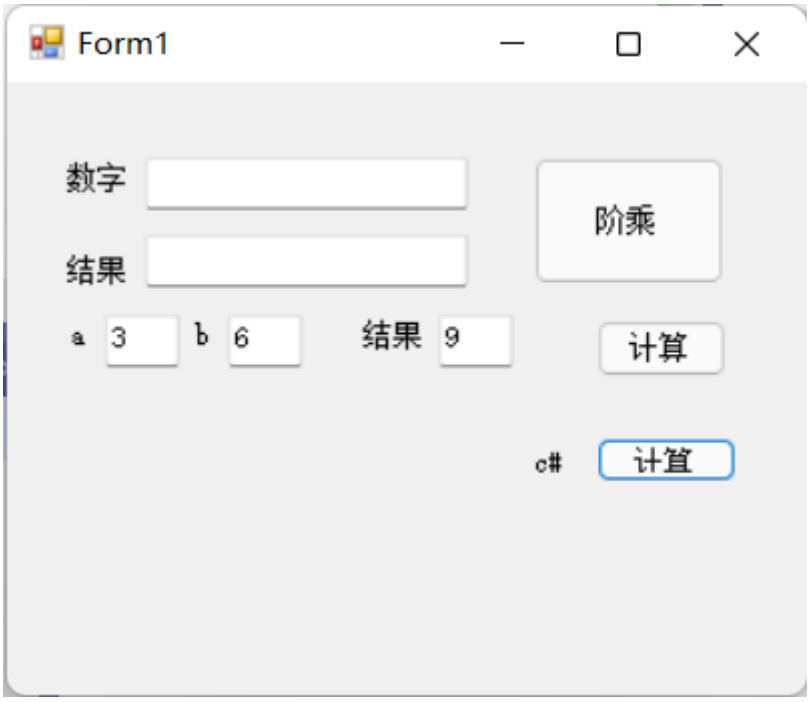
```
using CreateDLLcs;
```

在c#中调用代码如下：

```
            int a = int.Parse(textBox3.Text);
            int b = int.Parse(textBox4.Text);
            Class1 cl = new Class1();
            int res = cl.add(a, b);
            textBox5.Text = res.ToString();
```

三、实验结果

测试函数主要实现两数之和，如下：


Form1
数字
结果
阶乘
a 3
b 6
结果 9
计算
c#
计算