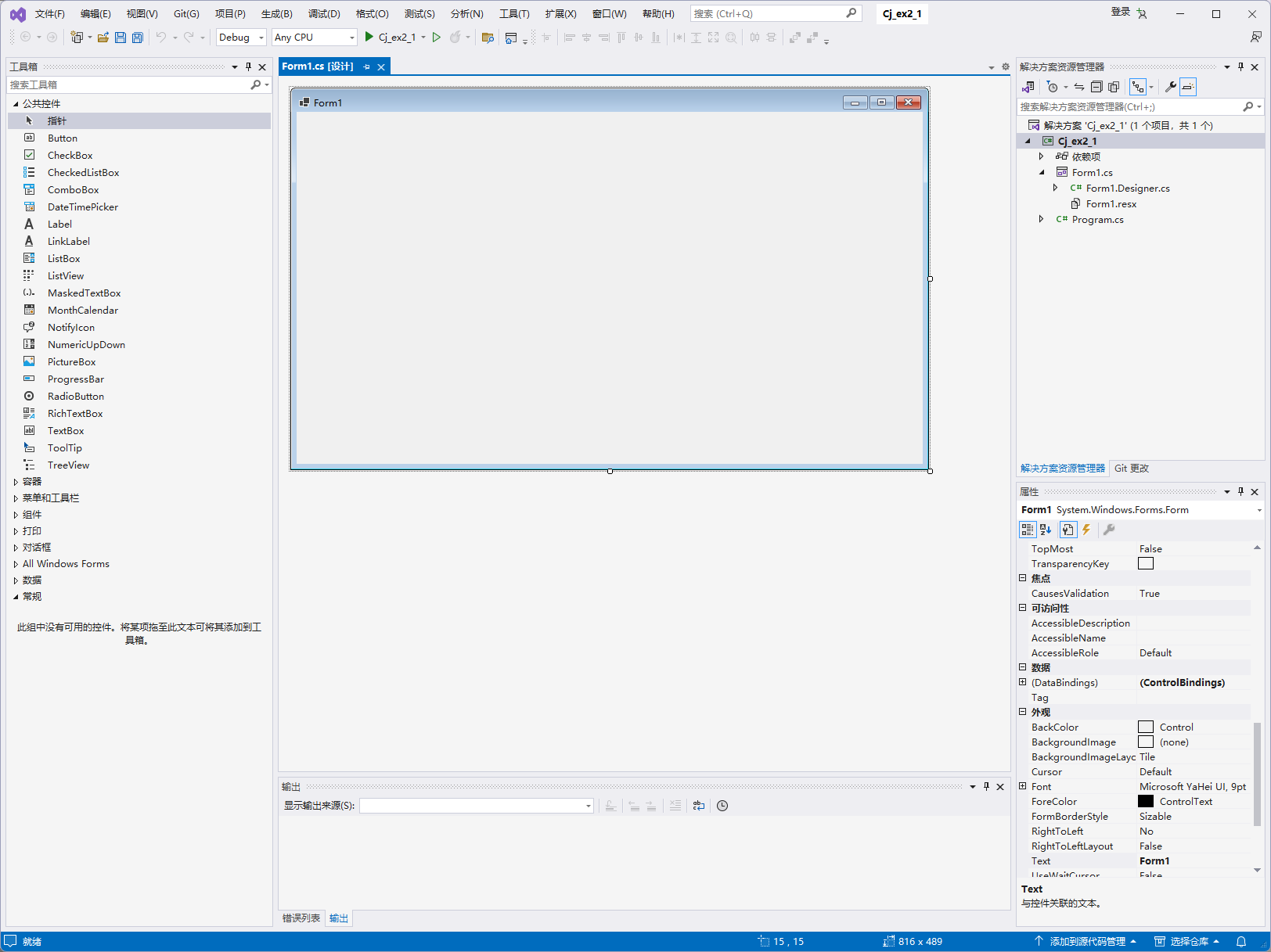
**实验报告**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **课程**  **名称** | 面向对象程序设计语言 | | | | **实验名称** | 复数计算器、字符串类设计 | | |
| **系部** | 地理信息科学系 | | **班级** | 地信班 | **姓名** | 许愿 | **学号** | 109092023XXX |
| **成绩** | |  | | | | | | |

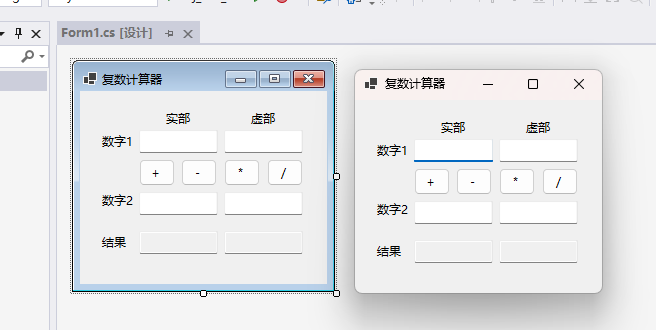
1. **实验目的**

通过设计复数运算计算器和字符串类，进一步加深对C#程序设计过程的认识。通过本实验来深入了解C#的语法和功能，在此前基础上进一步提升使用C#程序设计来解决生活中的实际问题的能力。

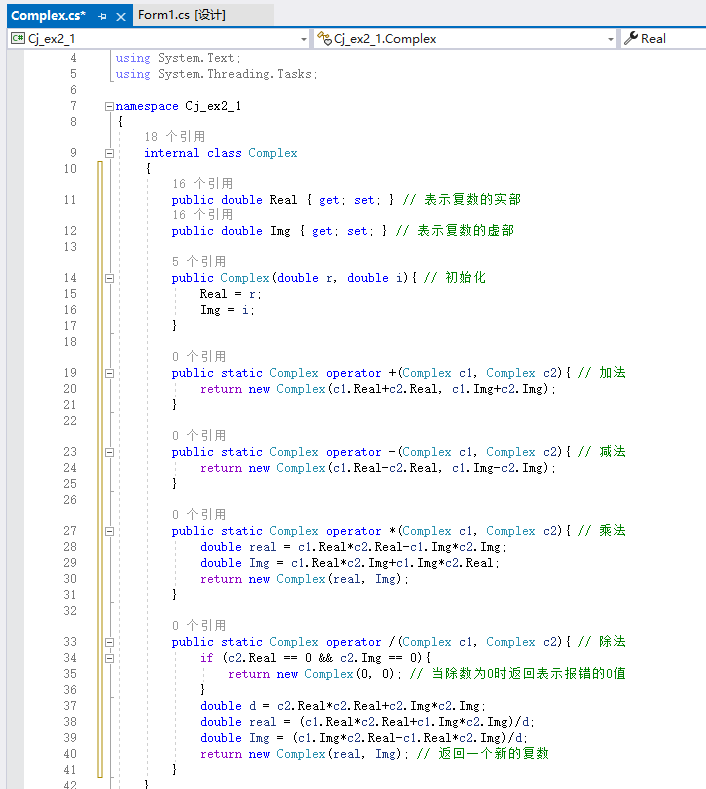
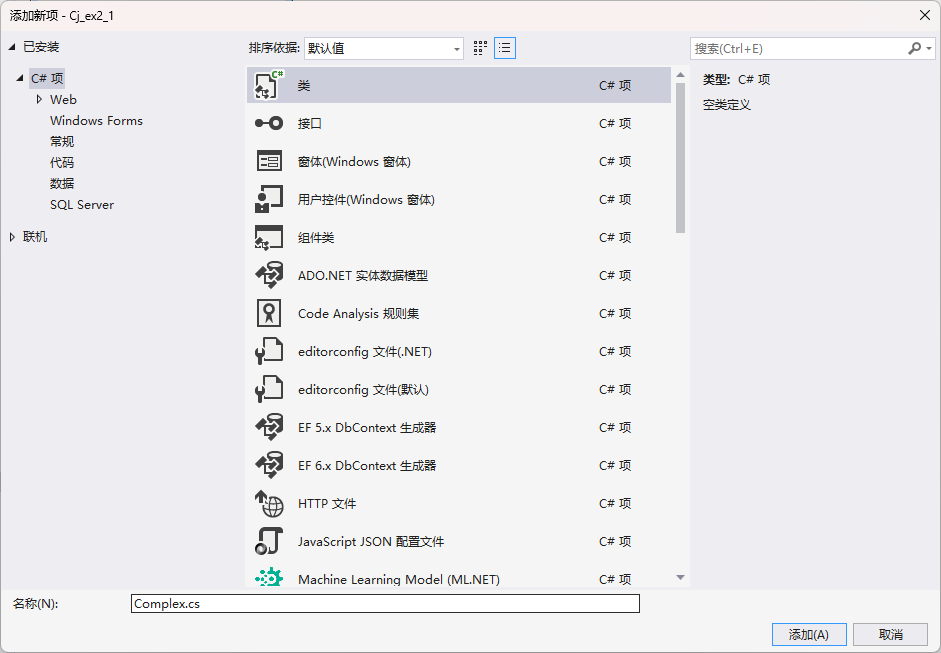
1. **实验内容**
2. 编写一个窗体程序，实现运算符（+ - \* /）的重载，实现复数的运算的功能。
3. 设计一个字符串类，并用Windows程序测试。要求如下：
4. 定义三个私有字段，string用于存放字符串的值，length用于记录字符串长度，color用于记录字体颜色；
5. 构建3个构造函数，第一个函数不带参数，能默认生成100个大写字母；第二个函数带一个长度参数，可以生成用户指定个数的混合字符串；第三个构造函数带两个参数（第一个为长度，第二个为类别），可以生成用户指定个数的大写、小写、混合的字符串；
6. 采用属性对length、color两个私有字段进行控制，其中控制length的属性是只读，控制color的属性是只写；
7. 采用索引器输入或返回字符串中的某个字符；
8. 设计一方法可截取字符串某个区间内的子字符串；
9. 设计一方法可以将字符串大小写互换；
10. 设计一方法可以根据字符ASCII码值进行排序；
11. **实验步骤**
12. 设计复数运算计算器
13. 新建Windows 窗体应用项目，命名为Cj\_ex2\_1。



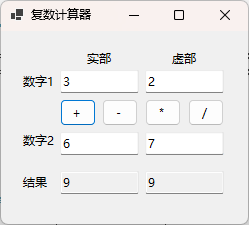
1. 设计界面，为窗口添加控件。此处需要添加两个文本框用于复数1的实部、虚部；两个文本框用于复数2的实部、虚部；两个文本框用于结果的实部、虚部（设置为只读）；四个按钮分别用于加减乘除。最终效果如图所示。



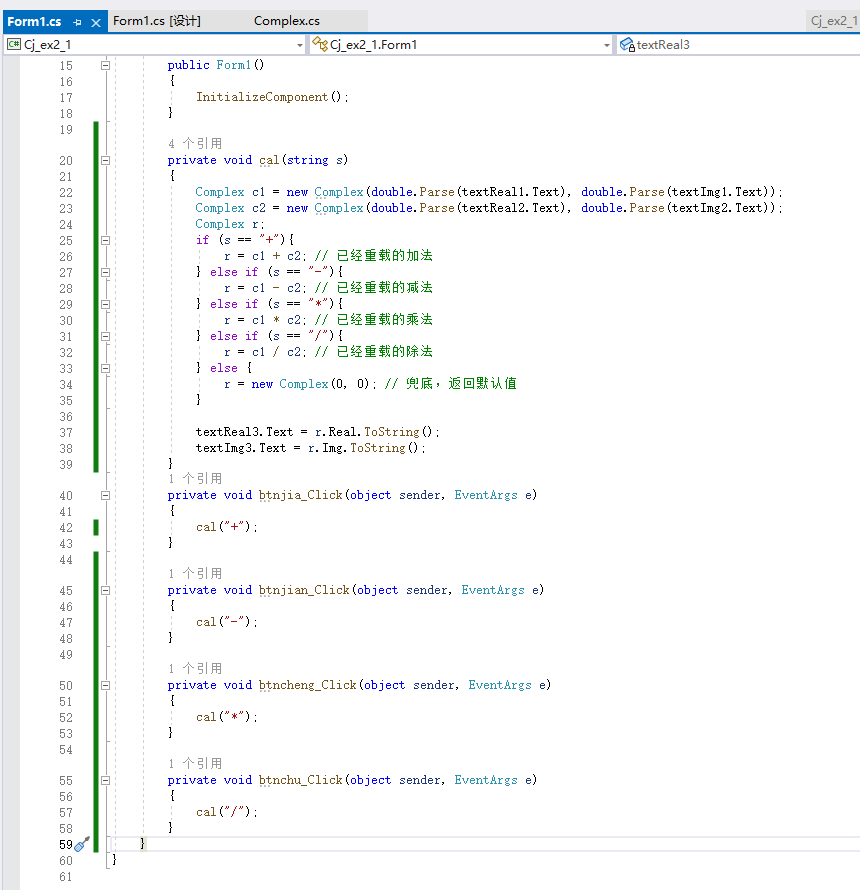
1. 创建复数类，命名为Complex.cs。为该类编写代码（初始化、加减乘除运算符的重载）。



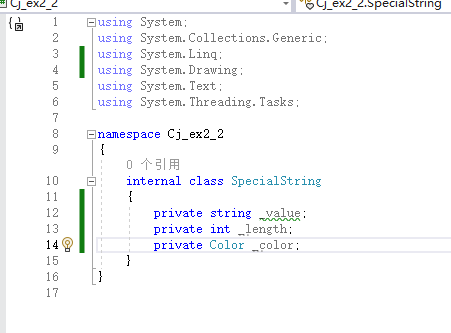
1. 双击窗体中的各个运算按钮，为按钮分别编写加法、减法、乘法、除法函数。编写完成后测试基本功能，可以正常运行。



1. 此时该计算器大体上已经可以正常使用，但代码依然比较冗长。四个按键的部分可以进行合并，这样代码就不会存在多次重复的情况，在有扩充或者修改的需要时更为便捷。最终代码如下。



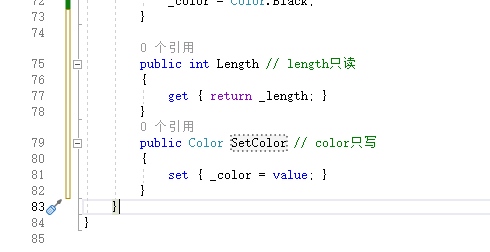
1. 设计字符串类
2. 新建Windows 窗体应用项目，命名为Cj\_ex2\_2。
3. 创建字符串类，命名为SpecialString.cs。
4. 满足第一个要求：定义三个私有字段。



1. 满足第二个要求，构建构造函数。使用Random函数来随机生成字符。生成时注意前闭后开。



1. 采用属性对两个私有字段分别设置只读、只写。



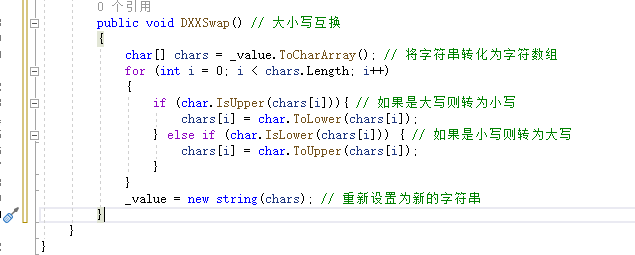
1. 采用索引器输入或返回字符串中某字符。此处若索引超出规定范围则报错并返回空白信息。



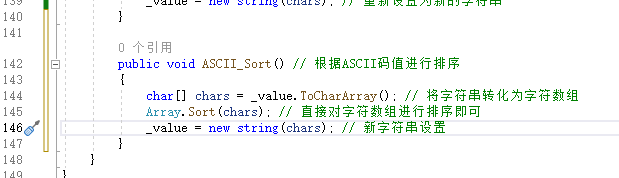
1. 设计可以截取字符串某区间内子字符串的方法。此处还需要单独增加一个直接赋值的构造函数，不然截取子字符串的方法没办法返回SpecialString类。



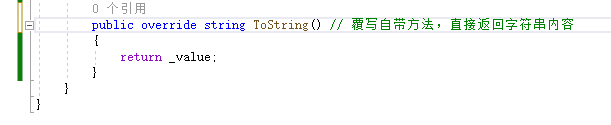
1. 设计可以互换字符串大小写的方法。若字符不为大写或小写字母，则不做改动。



1. 设计可以根据字符的ASCII码值进行排序的方法。Array类中自带相关的排序函数，直接使用即可。



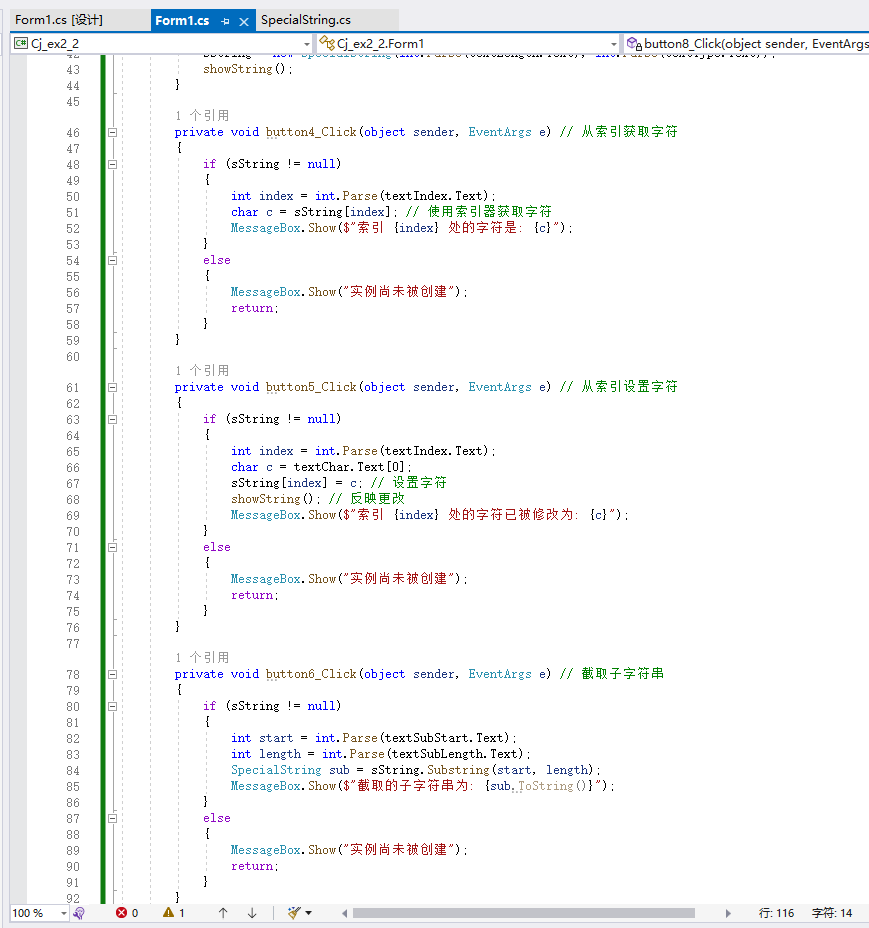
1. 考虑到需要在程序窗口中调试以展示字符串，还需要覆写ToString方法以进行字符内容的返回。

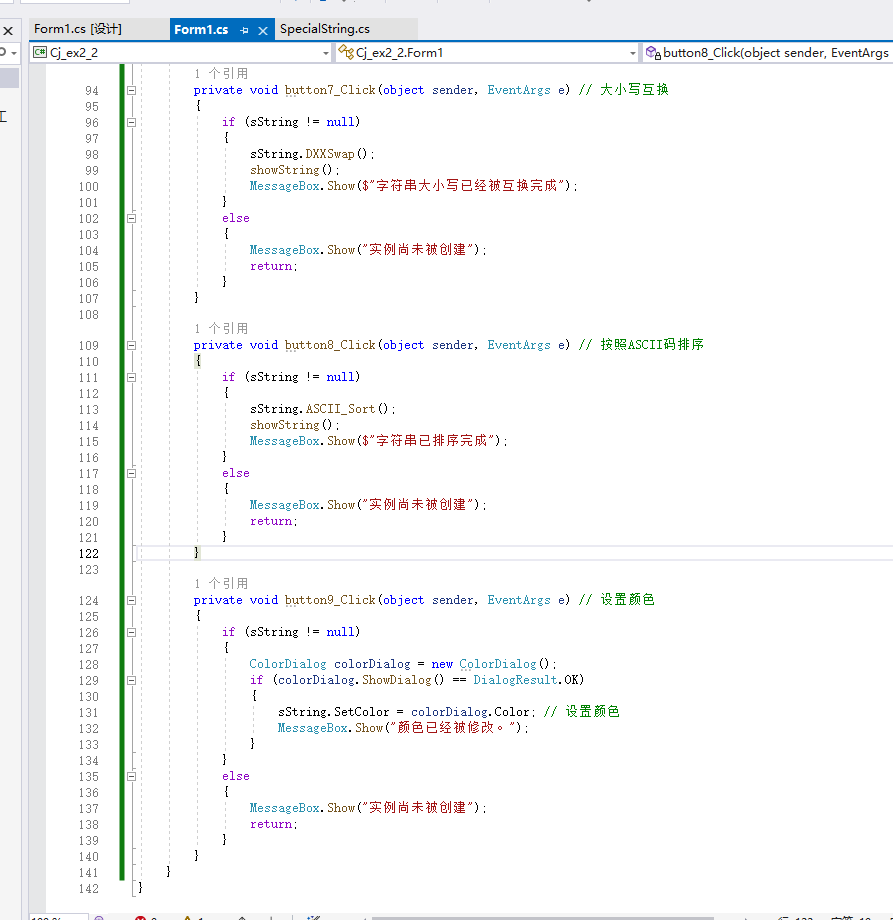


1. 设计测试字符串类的窗口。

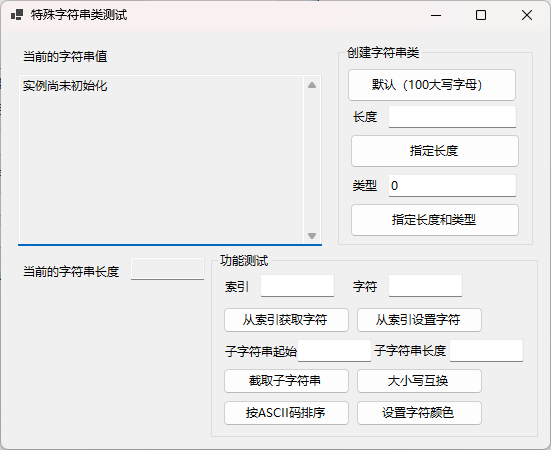


1. 设计用于测试代码正常运行的功能。

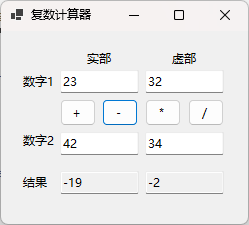


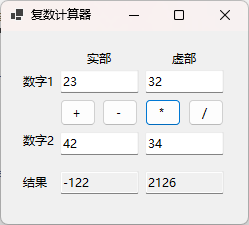


1. 运行程序，当前效果如下。

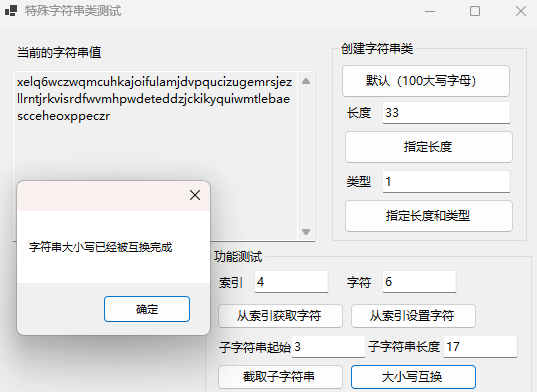
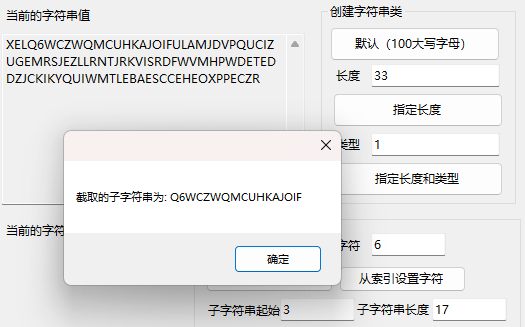
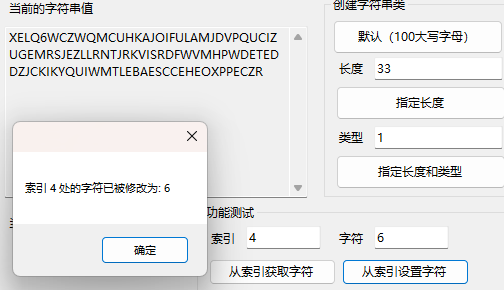
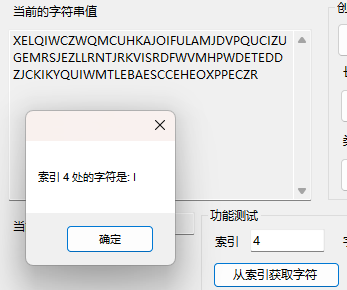
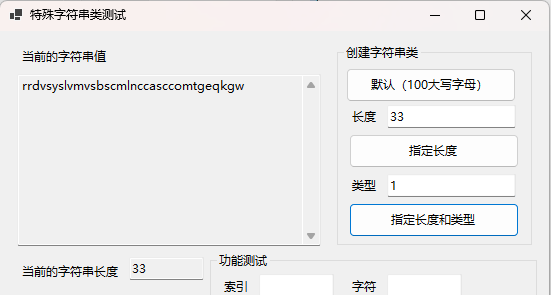
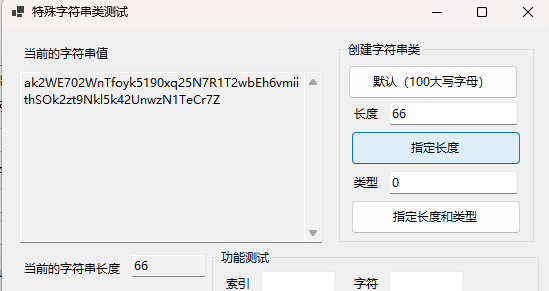
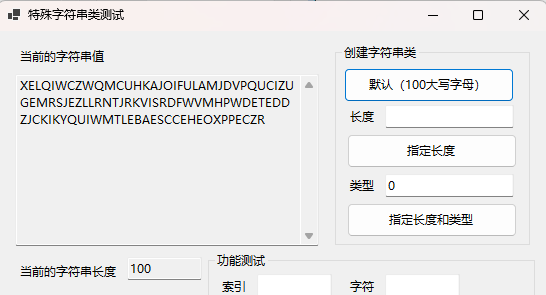


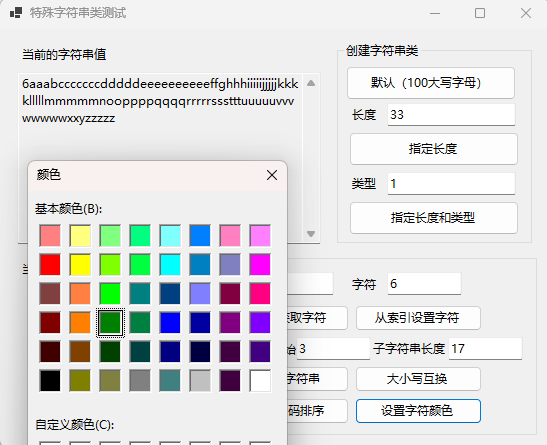
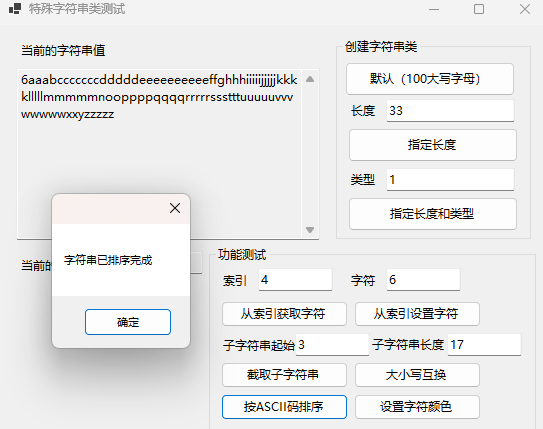
1. 对程序进行测试。运行结果在下一部分中。
2. **程序运行结果截图**
3. 复数运算计算器





1. 设计字符串类





由于specialString类的Color是只写，无法对其进行读取，也就无法显示在字符串值中。

1. **程序源代码**
2. 复数运算计算器
3. 窗口代码

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Cj\_ex2\_1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void cal(string s)

{

Complex c1 = new Complex(double.Parse(textReal1.Text), double.Parse(textImg1.Text));

Complex c2 = new Complex(double.Parse(textReal2.Text), double.Parse(textImg2.Text));

Complex r;

if (s == "+"){

r = c1 + c2; // 已经重载的加法

} else if (s == "-"){

r = c1 - c2; // 已经重载的减法

} else if (s == "\*"){

r = c1 \* c2; // 已经重载的乘法

} else if (s == "/"){

r = c1 / c2; // 已经重载的除法

} else {

r = new Complex(0, 0); // 兜底，返回默认值

}

textReal3.Text = r.Real.ToString();

textImg3.Text = r.Img.ToString();

}

private void btnjia\_Click(object sender, EventArgs e)

{

cal("+");

}

private void btnjian\_Click(object sender, EventArgs e)

{

cal("-");

}

private void btncheng\_Click(object sender, EventArgs e)

{

cal("\*");

}

private void btnchu\_Click(object sender, EventArgs e)

{

cal("/");

}

}

}

1. 类代码

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Cj\_ex2\_1

{

internal class Complex

{

public double Real { get; set; } // 表示复数的实部

public double Img { get; set; } // 表示复数的虚部

public Complex(double r, double i){ // 初始化

Real = r;

Img = i;

}

public static Complex operator +(Complex c1, Complex c2){ // 加法

return new Complex(c1.Real+c2.Real, c1.Img+c2.Img);

}

public static Complex operator -(Complex c1, Complex c2){ // 减法

return new Complex(c1.Real-c2.Real, c1.Img-c2.Img);

}

public static Complex operator \*(Complex c1, Complex c2){ // 乘法

double real = c1.Real\*c2.Real-c1.Img\*c2.Img;

double Img = c1.Real\*c2.Img+c1.Img\*c2.Real;

return new Complex(real, Img);

}

public static Complex operator /(Complex c1, Complex c2){ // 除法

if (c2.Real == 0 && c2.Img == 0){

return new Complex(0, 0); // 当除数为0时返回表示报错的0值

}

double d = c2.Real\*c2.Real+c2.Img\*c2.Img;

double real = (c1.Real\*c2.Real+c1.Img\*c2.Img)/d;

double Img = (c1.Img\*c2.Real-c1.Real\*c2.Img)/d;

return new Complex(real, Img); // 返回一个新的复数

}

}

}

1. 设计字符串类
2. 窗口代码

using System;

namespace Cj\_ex2\_2

{

public partial class Form1 : Form

{

private SpecialString sString;

public Form1()

{

InitializeComponent();

showString();

}

private void showString() // 自定义函数，用来显示当前的值和长度

{

if (sString != null)

{

textStringValue.Text = sString.ToString();

textStringLength.Text = sString.Length.ToString();

}

else

{

textStringValue.Text = "实例尚未初始化";

textStringLength.Text = "";

}

}

private void button1\_Click(object sender, EventArgs e) // 创建默认的sString

{

sString = new SpecialString();

showString();

}

private void button2\_Click(object sender, EventArgs e) // 创建指定长度的sString

{

sString = new SpecialString(int.Parse(textLength.Text));

showString();

}

private void button3\_Click(object sender, EventArgs e) // 创建指定长度和类型的sString

{

sString = new SpecialString(int.Parse(textLength.Text), int.Parse(textType.Text));

showString();

}

private void button4\_Click(object sender, EventArgs e) // 从索引获取字符

{

if (sString != null)

{

int index = int.Parse(textIndex.Text);

char c = sString[index]; // 使用索引器获取字符

MessageBox.Show($"索引 {index} 处的字符是: {c}");

}

else

{

MessageBox.Show("实例尚未被创建");

return;

}

}

private void button5\_Click(object sender, EventArgs e) // 从索引设置字符

{

if (sString != null)

{

int index = int.Parse(textIndex.Text);

char c = textChar.Text[0];

sString[index] = c; // 设置字符

showString(); // 反映更改

MessageBox.Show($"索引 {index} 处的字符已被修改为: {c}");

}

else

{

MessageBox.Show("实例尚未被创建");

return;

}

}

private void button6\_Click(object sender, EventArgs e) // 截取子字符串

{

if (sString != null)

{

int start = int.Parse(textSubStart.Text);

int length = int.Parse(textSubLength.Text);

SpecialString sub = sString.Substring(start, length);

MessageBox.Show($"截取的子字符串为: {sub.ToString()}");

}

else

{

MessageBox.Show("实例尚未被创建");

return;

}

}

private void button7\_Click(object sender, EventArgs e) // 大小写互换

{

if (sString != null)

{

sString.DXXSwap();

showString();

MessageBox.Show($"字符串大小写已经被互换完成");

}

else

{

MessageBox.Show("实例尚未被创建");

return;

}

}

private void button8\_Click(object sender, EventArgs e) // 按照ASCII码排序

{

if (sString != null)

{

sString.ASCII\_Sort();

showString();

MessageBox.Show($"字符串已排序完成");

}

else

{

MessageBox.Show("实例尚未被创建");

return;

}

}

private void button9\_Click(object sender, EventArgs e) // 设置颜色

{

if (sString != null)

{

ColorDialog colorDialog = new ColorDialog();

if (colorDialog.ShowDialog() == DialogResult.OK)

{

sString.SetColor = colorDialog.Color; // 设置颜色

MessageBox.Show("颜色已经被修改。");

}

}

else

{

MessageBox.Show("实例尚未被创建");

return;

}

}

}

}

1. 类代码

using System;

using System.Collections.Generic;

using System.Linq;

using System.Drawing;

using System.Text;

using System.Threading.Tasks;

using static System.Runtime.InteropServices.JavaScript.JSType;

namespace Cj\_ex2\_2

{

internal class SpecialString

{

private string \_value;

private int \_length;

private Color \_color;

public SpecialString() // 默认参数的构造函数

{

string s = "";

Random rand = new Random();

for (int i = 0; i < 100; i++){

s += (char)rand.Next('A', 'Z' + 1); // 随机生成大写字母，前闭后开

}

\_value = s;

\_length = \_value.Length;

\_color = Color.Black;

}

public SpecialString(int length) // 带一个长度参数的构造函数

{

if (length < 0) length = 0; // 若长度为负数，则不处理了

string s = "";

Random rand = new Random();

for (int i = 0; i < length; i++) {

int charType = rand.Next(3); // 0大写, 1小写, 2数字

if (charType == 0) {

s += (char)rand.Next('A', 'Z' + 1);

} else if (charType == 1) {

s += (char)rand.Next('a', 'z' + 1);

} else {

s += (char)rand.Next('0', '9' + 1);

}

}

\_value = s;

\_length = \_value.Length;

\_color = Color.Black;

}

public SpecialString(int length, int type = 0) // 带两个参数的构造函数，类别中0为大写（默认），1为小写，2为混合

{

if (length < 0) length = 0; // 若长度为负数，则不处理了

string s = "";

Random rand = new Random();

for (int i = 0; i < length; i++) {

if (type == 1) { // 小写

s += (char)rand.Next('a', 'z' + 1);

} else if (type == 2) { // 混合

int charType = rand.Next(3); // 0大写, 1小写, 2数字

if (charType == 0) {

s += (char)rand.Next('A', 'Z' + 1);

} else if (charType == 1) {

s += (char)rand.Next('a', 'z' + 1);

} else {

s += (char)rand.Next('0', '9' + 1);

}

} else { // 大写

s += (char)rand.Next('A', 'Z' + 1);

}

}

\_value = s;

\_length = \_value.Length;

\_color = Color.Black;

}

public int Length // length只读

{

get { return \_length; }

}

public Color SetColor // color只写

{

set { \_color = value; }

}

public char this[int index] // 重载

{

get {

if (index < 0 || index >= \_value.Length) {

MessageBox.Show("索引超出范围！");

return ' ';

}

return \_value[index];

}

set {

if (index < 0 || index >= \_value.Length) {

MessageBox.Show("索引超出范围！");

return;

}

char[] chars = \_value.ToCharArray(); // 将字符串转化为字符数组

chars[index] = value; // 修改其中内容

\_value = new string(chars); // 重新设置为新的字符串

}

}

public SpecialString Substring(int start, int length)

{

if (start < 0 || start >= \_value.Length) {

MessageBox.Show("索引超出范围！");

return new SpecialString(); // 返回空

} else if (length < 0) {

MessageBox.Show("长度不能为负数！");

return new SpecialString(); // 返回空

} else if (start + length > \_value.Length) {

MessageBox.Show("索引长度超出范围！");

return new SpecialString(); // 返回空

}

return new SpecialString(\_value.Substring(start, length), \_color);

}

public SpecialString(string value, Color color) // 专门用于截取子字符串的构造函数

{

\_value = value;

\_length = value.Length;

\_color = color;

}

public void DXXSwap() // 大小写互换

{

char[] chars = \_value.ToCharArray(); // 将字符串转化为字符数组

for (int i = 0; i < chars.Length; i++)

{

if (char.IsUpper(chars[i])){ // 如果是大写则转为小写

chars[i] = char.ToLower(chars[i]);

} else if (char.IsLower(chars[i])) { // 如果是小写则转为大写

chars[i] = char.ToUpper(chars[i]);

}

}

\_value = new string(chars); // 重新设置为新的字符串

}

public void ASCII\_Sort() // 根据ASCII码值进行排序

{

char[] chars = \_value.ToCharArray(); // 将字符串转化为字符数组

Array.Sort(chars); // 直接对字符数组进行排序即可

\_value = new string(chars); // 新字符串设置

}

public override string ToString() // 覆写自带方法，直接返回字符串内容

{

return \_value;

}

}

}

1. **收获，体会及问题**

本次实验通过设计复数计算器和自定义字符串类，让我对C#面向对象程序设计的核心概念有了更深入的理解和实践。

通过设计复数计算器，我学习了如何利用运算符重载简化数学运算的表达，体会到将运算逻辑封装可以提升代码的复用性的优势。但依然存在一些局限：double类型在处理大数据时可能溢出从而导致计算错误；对于空白文本和除数为零的情况在实际使用时返回0不足以应对，依然缺少更完善的处理机制。

在字符串类设计过程中，我通过定义私有字段、构建构造函数、设置只读只写属性、实现索引器和设计字符串操作方法来更加系统地掌握了C#中类的各种用法。但在实践中，也再次暴露出一些问题。用户如果输入非数字值（想起一个经典例子：在酒吧点炒饭），直接的int.Parse方法会引发异常导致程序崩溃，与复数计算器面临的问题类似，更体现输入验证和异常处理的重要性。color属性因为只写而无法在用户界面上直观展示当前字符串颜色，总感觉有些不足。

总而言之，此次实验不仅使我巩固了此前课堂上学习的理论知识，还让我发现了在实际开发中可能遇到的一些问题，对如何编写更好的C#应用程序有了一些思考。