

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							

一、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。

二、实验内容

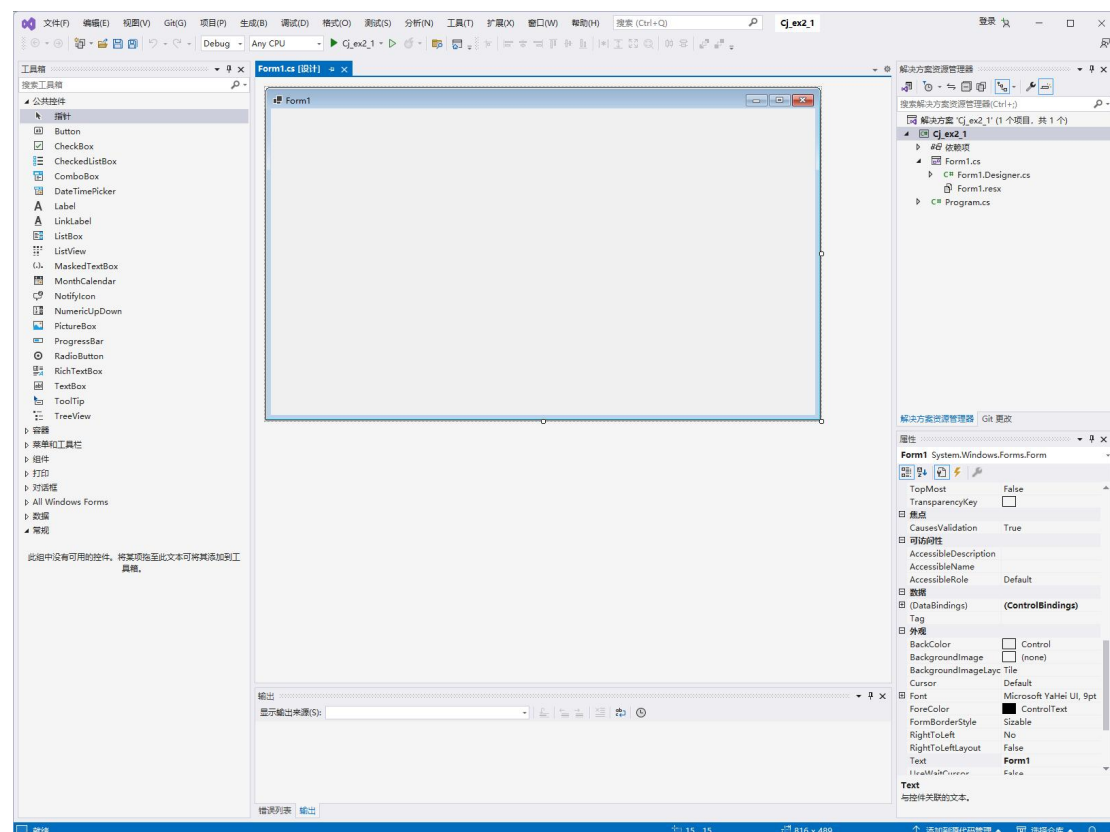
- 编写一个窗体程序，实现运算符（+ - * /）的重载，实现复数的运算的功能。
- 设计一个字符串类，并用 Windows 程序测试。要求如下：
 - 定义三个私有字段，string 用于存放字符串的值，length 用于记录字符串长度，color 用于记录字体颜色；
 - 构建 3 个构造函数，第一个函数不带参数，能默认生成 100 个大写字母；第二个函数带一个长度参数，可以生成用户指定个数的混合字符串；第三个构造函数带两个参数（第一个为长度，第二个为类别），可以生成用户指定个数的大写、小写、混合的字符串；
 - 采用属性对 length、color 两个私有字段进行控制，其中控制 length 的属性是只读，控制 color 的属性是只写；

- (4) 采用索引器输入或返回字符串中的某个字符；
- (5) 设计一方法可截取字符串某个区间内的子字符串；
- (6) 设计一方法可以将字符串大小写互换；
- (7) 设计一方法可以根据字符 ASCII 码值进行排序；

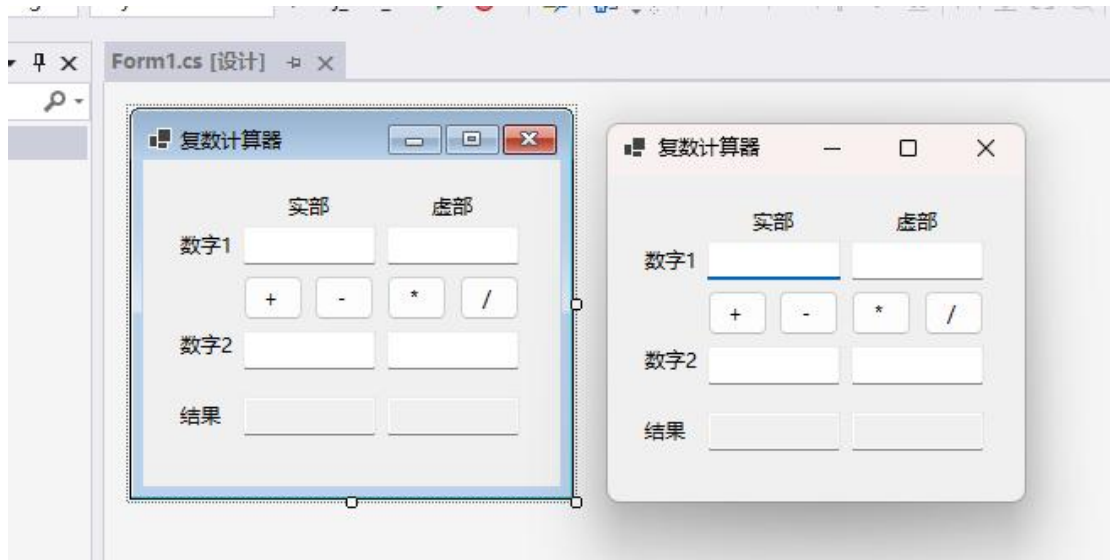
三、实验步骤

1. 设计复数运算计算器

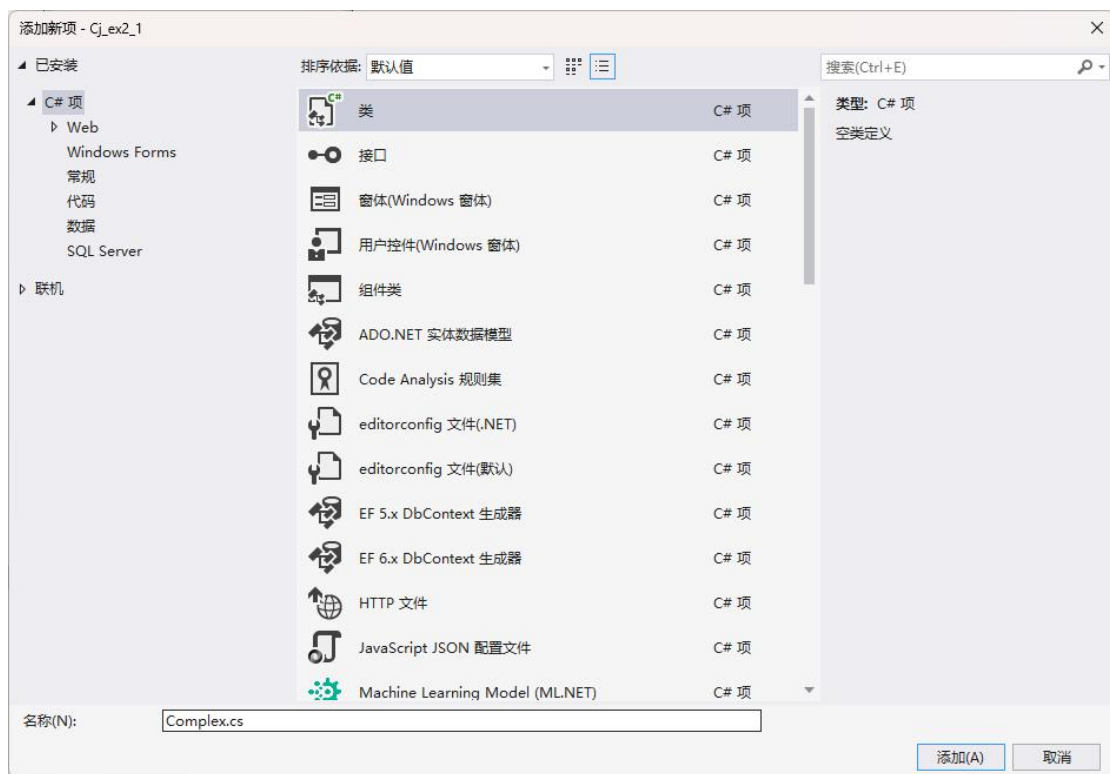
(1) 新建 Windows 窗体应用项目，命名为 Cj_ex2_1。



(2) 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。



(3) 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）。



```
Complex.cs*  Form1.cs [设计]
Cj_ex2_1  Cj_ex2_1.Complex  Real

4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace Cj_ex2_1
8  {
9      18 个引用
10     internal class Complex
11     {
12         16 个引用
13         public double Real { get; set; } // 表示复数的实部
14         16 个引用
15         public double Img { get; set; } // 表示复数的虚部
16
17         5 个引用
18         public Complex(double r, double i){ // 初始化
19             Real = r;
20             Img = i;
21         }
22
23         0 个引用
24         public static Complex operator +(Complex c1, Complex c2){ // 加法
25             return new Complex(c1.Real+c2.Real, c1.Img+c2.Img);
26         }
27
28         0 个引用
29         public static Complex operator -(Complex c1, Complex c2){ // 减法
30             return new Complex(c1.Real-c2.Real, c1.Img-c2.Img);
31         }
32
33         0 个引用
34         public static Complex operator *(Complex c1, Complex c2){ // 乘法
35             double real = c1.Real*c2.Real-c1.Img*c2.Img;
36             double Img = c1.Real*c2.Img+c1.Img*c2.Real;
37             return new Complex(real, Img);
38         }
39
40         0 个引用
41         public static Complex operator /(Complex c1, Complex c2){ // 除法
42             if (c2.Real == 0 && c2.Img == 0){
43                 return new Complex(0, 0); // 当除数为0时返回表示报错的0值
44             }
45             double d = c2.Real*c2.Real+c2.Img*c2.Img;
46             double real = (c1.Real*c2.Real+c1.Img*c2.Img)/d;
47             double Img = (c1.Img*c2.Real-c1.Real*c2.Img)/d;
48             return new Complex(real, Img); // 返回一个新的复数
49         }
50     }
51 }
```

(4) 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。

```

19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

```

```

1 个引用
private void btnjia_Click(object sender, EventArgs e)
{
    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 = c1 + c2; // 已经重载的加法
    textReal3.Text = r.Real.ToString();
    textImg3.Text = r.Img.ToString();
}

1 个引用
private void btnjian_Click(object sender, EventArgs e)
{
    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 = c1 - c2; // 已经重载的减法
    textReal3.Text = r.Real.ToString();
    textImg3.Text = r.Img.ToString();
}

1 个引用
private void btncheng_Click(object sender, EventArgs e)
{
    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 = c1 * c2; // 已经重载的乘法
    textReal3.Text = r.Real.ToString();
    textImg3.Text = r.Img.ToString();
}

1 个引用
private void btnchu_Click(object sender, EventArgs e)
{
    Complex c1 = new Complex(double.Parse(textReal1.Text), double.Parse(textImg1.Text));
    Complex c2 = new Complex(double.Parse(textReal2.Text), double.Parse(textImg2.Text));

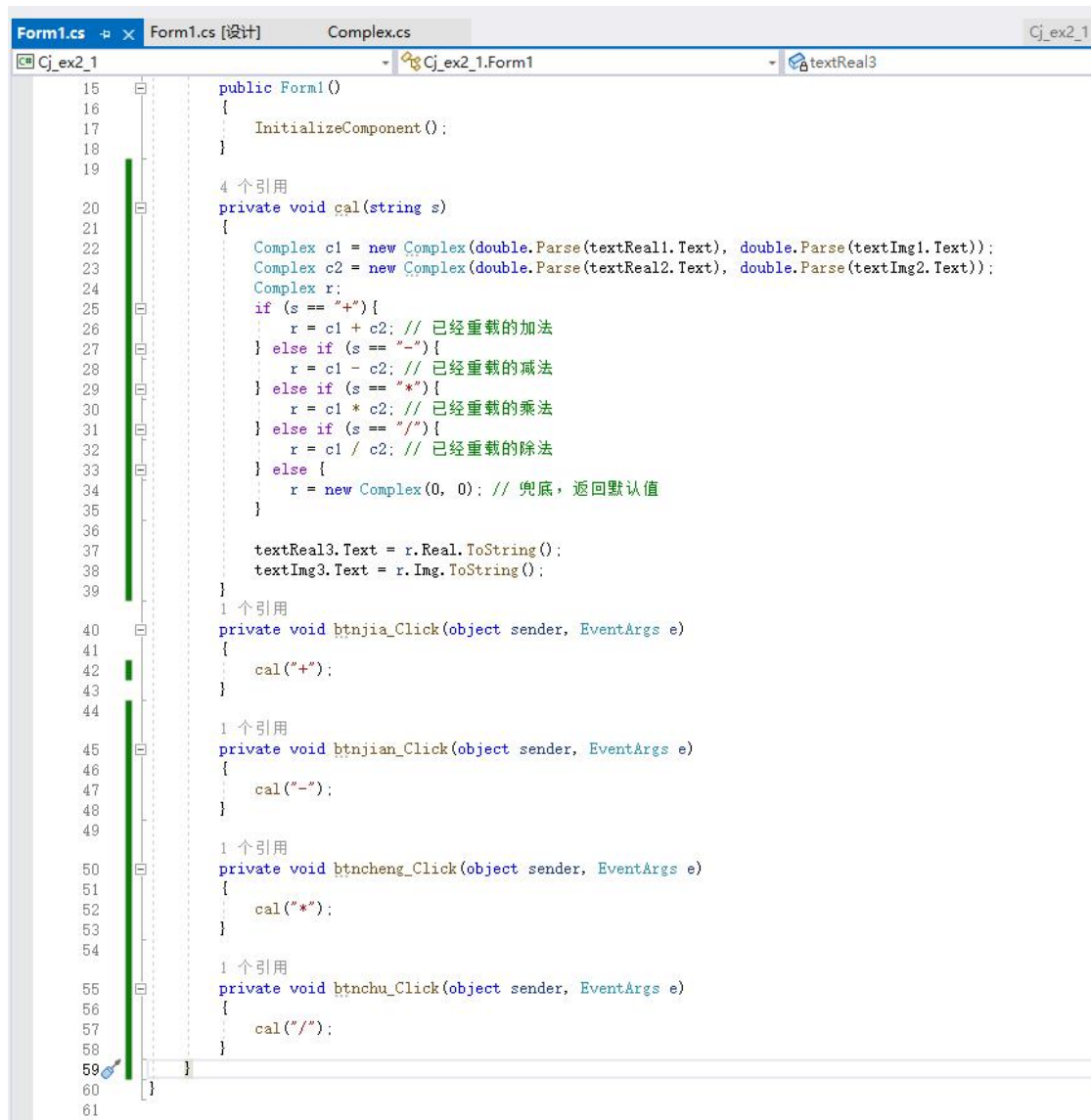
    Complex r = c1 / c2; // 已经重载的除法
    textReal3.Text = r.Real.ToString();
    textImg3.Text = r.Img.ToString();
}

```



(5) 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。



```
Form1.cs  Form1.cs [设计]  Complex.cs  Cj_ex2_1  Cj_ex2_1  textReal3
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61

public Form1()
{
    InitializeComponent();
}

4 个引用
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();
}

1 个引用
private void btnjia_Click(object sender, EventArgs e)
{
    cal("+");
}

1 个引用
private void btnjian_Click(object sender, EventArgs e)
{
    cal("-");
}

1 个引用
private void btncheng_Click(object sender, EventArgs e)
{
    cal("*");
}

1 个引用
private void btncchu_Click(object sender, EventArgs e)
{
    cal("/");
}
```

2. 设计字符串类

- (1) 新建 Windows 窗体应用项目，命名为 Cj_ex2_2。
- (2) 创建字符串类，命名为 SpecialString.cs。
- (3) 满足第一个要求：定义三个私有字段。


```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Drawing;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace Cj_ex2_2
9  {
10     0 个引用
11     internal class SpecialString
12     {
13         private string _value;
14         private int _length;
15         private Color _color;
16     }
17

```

(4) 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。

```

26
27
28     0 个引用
29     public SpecialString(int length) // 带一个长度参数的构造函数
30     {
31         if (length < 0) length = 0; // 若长度为负数，则不处理了
32         string s = "";
33         Random rand = new Random();
34         for (int i = 0; i < length; i++) {
35             int charType = rand.Next(3); // 0大写, 1小写, 2数字
36             if (charType == 0) {
37                 s += (char)rand.Next('A', 'Z' + 1);
38             } else if (charType == 1) {
39                 s += (char)rand.Next('a', 'z' + 1);
40             } else {
41                 s += (char)rand.Next('0', '9' + 1);
42             }
43         }
44         _value = s;
45         _length = _value.Length;
46         _color = Color.Black;
47     }
48
49     0 个引用
50     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，
51     {
52         if (length < 0) length = 0; // 若长度为负数，则不处理了
53         string s = "";
54         Random rand = new Random();
55         for (int i = 0; i < length; i++) {
56             if (type == 1) { // 小写
57                 s += (char)rand.Next('a', 'z' + 1);
58             } else if (type == 2) { // 混合
59                 int charType = rand.Next(3); // 0大写, 1小写, 2数字
60                 if (charType == 0) {
61                     s += (char)rand.Next('A', 'Z' + 1);
62                 } else if (charType == 1) {
63                     s += (char)rand.Next('a', 'z' + 1);
64                 } else {
65                     s += (char)rand.Next('0', '9' + 1);
66                 }
67             } else { // 大写
68                 s += (char)rand.Next('A', 'Z' + 1);
69             }
70         }
71         _value = s;
72         _length = _value.Length;
73         _color = Color.Black;
74     }
75
76

```

(5) 采用属性对两个私有字段分别设置只读、只写。



```
72         _color = Color.Black,
73     }
74
75     0 个引用
76     public int Length // length只读
77     {
78         get { return _length; }
79     }
80     0 个引用
81     public Color SetColor // color只写
82     {
83         set { _color = value; }
84     }
85 }
```

(6) 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。



```
81         set { _color = value; }
82     }
83
84     0 个引用
85     public char this[int index] // 重载
86     {
87         get {
88             if (index < 0 || index >= _value.Length) {
89                 MessageBox.Show("索引超出范围!");
90                 return ' ';
91             }
92             return _value[index];
93         }
94         set {
95             if (index < 0 || index >= _value.Length) {
96                 MessageBox.Show("索引超出范围!");
97                 return;
98             }
99             char[] chars = _value.ToCharArray(); // 将字符串转化为字符数组
100             chars[index] = value; // 修改其中内容
101             _value = new string(chars); // 重新设置为新的字符串
102         }
103     }
104 }
```

(7) 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。


```

104
105 0 个引用
106 public SpecialString Substring(int start, int length)
107 {
108     if (start < 0 || start >= _value.Length) {
109         MessageBox.Show("索引超出范围!");
110         return new SpecialString(); // 返回空
111     } else if (length < 0) {
112         MessageBox.Show("长度不能为负数!");
113         return new SpecialString(); // 返回空
114     } else if (start + length > _value.Length) {
115         MessageBox.Show("索引超出范围!");
116         return new SpecialString(); // 返回空
117     }
118     return new SpecialString(_value.Substring(start, length), _color);
119 }
120
121 1 个引用
122 public SpecialString(string value, Color color) // 专门用于截取子字符串的构造函数
123 {
124     _value = value;
125     _length = value.Length;
126     _color = color;
127 }
128
129

```

(8) 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。

```

130
131 0 个引用
132 public void DXSwap() // 大小写互换
133 {
134     char[] chars = _value.ToCharArray(); // 将字符串转化为字符数组
135     for (int i = 0; i < chars.Length; i++)
136     {
137         if (char.IsUpper(chars[i])) { // 如果是大写则转为小写
138             chars[i] = char.ToLower(chars[i]);
139         } else if (char.IsLower(chars[i])) { // 如果是小写则转为大写
140             chars[i] = char.ToUpper(chars[i]);
141         }
142     }
143     _value = new string(chars); // 重新设置为新的字符串
144 }
145
146

```

(9) 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。

```

139 _value = new string(chars); // 重新设置为新的字符串
140 }
141
142 0 个引用
143 public void ASCII_Sort() // 根据ASCII码值进行排序
144 {
145     char[] chars = _value.ToCharArray(); // 将字符串转化为字符数组
146     Array.Sort(chars); // 直接对字符数组进行排序即可
147     _value = new string(chars); // 新字符串设置
148 }
149

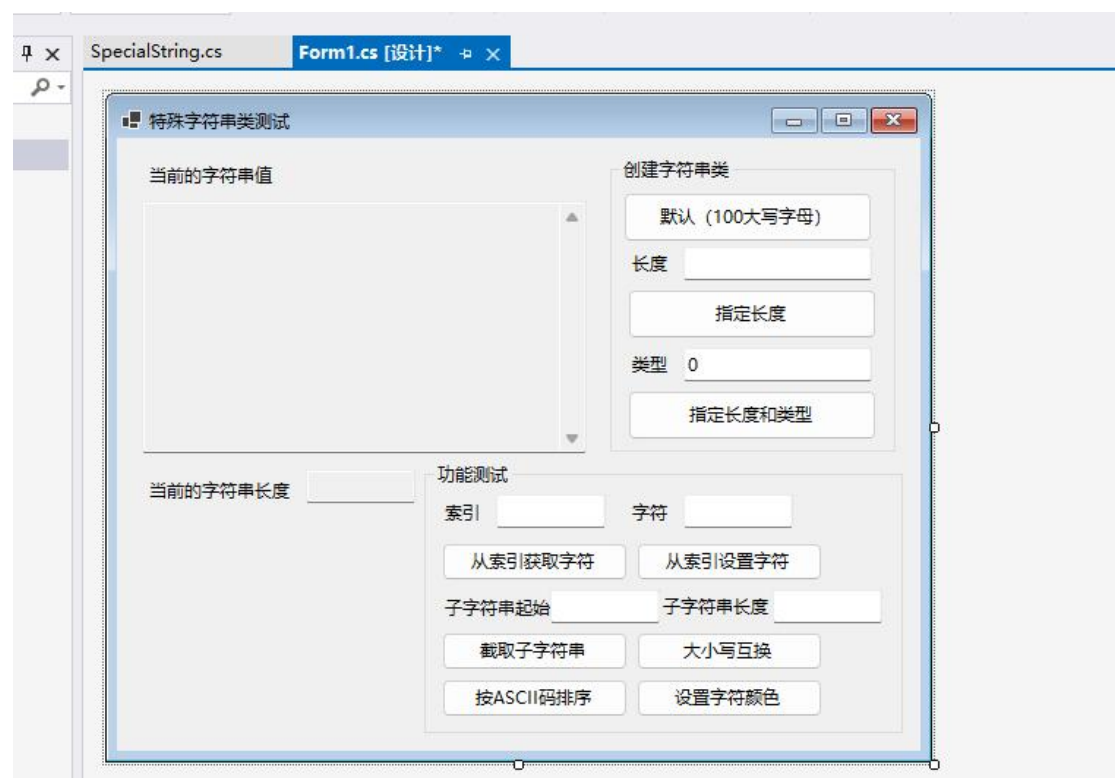
```

(10) 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。

```
0 个引用
public override string ToString() // 覆写自带方法，直接返回字符串内容
{
    return _value;
}
```

(11) 设计测试字符串类的窗口。



(12) 设计用于测试代码正常运行的功能。

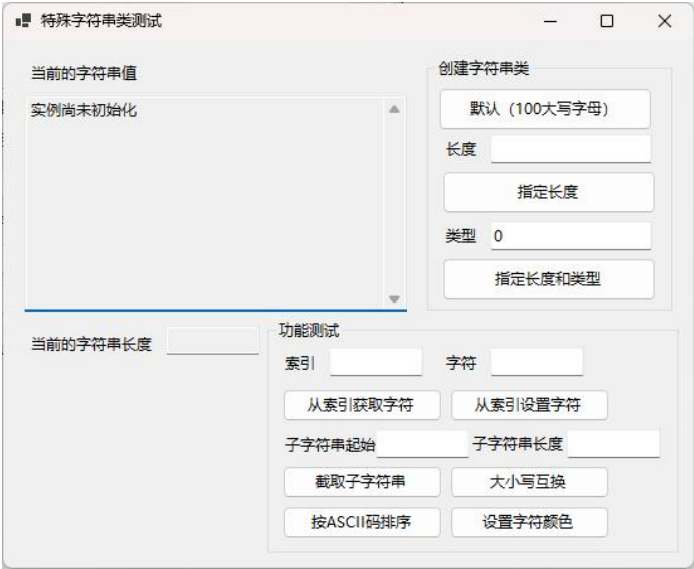
```
Form1.cs [设计] Form1.cs SpecialString.cs
Cj_ex2_2 Cj_ex2_2Form1 button8_Click(object sender, EventArgs)

43 showString();
44 }
45
46 1 个引用
47 private void button4_Click(object sender, EventArgs e) // 从索引获取字符
48 {
49     if (sString != null)
50     {
51         int index = int.Parse(textIndex.Text);
52         char c = sString[index]; // 使用索引器获取字符
53         MessageBox.Show($"索引 {index} 处的字符是: {c}");
54     }
55     else
56     {
57         MessageBox.Show("实例尚未被创建");
58         return;
59     }
60
61 1 个引用
62 private void button5_Click(object sender, EventArgs e) // 从索引设置字符
63 {
64     if (sString != null)
65     {
66         int index = int.Parse(textIndex.Text);
67         char c = textChar.Text[0];
68         sString[index] = c; // 设置字符
69         showString(); // 反映更改
70         MessageBox.Show($"索引 {index} 处的字符已被修改为: {c}");
71     }
72     else
73     {
74         MessageBox.Show("实例尚未被创建");
75         return;
76     }
77
78 1 个引用
79 private void button6_Click(object sender, EventArgs e) // 截取子字符串
80 {
81     if (sString != null)
82     {
83         int start = int.Parse(textSubStart.Text);
84         int length = int.Parse(textSubLength.Text);
85         SpecialString sub = sString.Substring(start, length);
86         MessageBox.Show($"截取的字字符串为: {sub.ToString()}");
87     }
88     else
89     {
90         MessageBox.Show("实例尚未被创建");
91         return;
92     }
93 }
94
100 % 行: 116 字符: 14
```

```
Form1.cs [设计] Form1.cs SpecialString.cs
Cj_ex2_2 Cj_ex2_2Form1 button8_Click(object sender, EventArgs)

94 1 个引用
95 private void button7_Click(object sender, EventArgs e) // 大小写互换
96 {
97     if (sString != null)
98     {
99         sString.DXSwap();
100         showString();
101         MessageBox.Show($"字符串大小写已经被互换完成");
102     }
103     else
104     {
105         MessageBox.Show("实例尚未被创建");
106         return;
107     }
108
109 1 个引用
110 private void button8_Click(object sender, EventArgs e) // 按照ASCII码排序
111 {
112     if (sString != null)
113     {
114         sString.ASCII_Sort();
115         showString();
116         MessageBox.Show($"字符串已排序完成");
117     }
118     else
119     {
120         MessageBox.Show("实例尚未被创建");
121         return;
122     }
123
124 1 个引用
125 private void button9_Click(object sender, EventArgs e) // 设置颜色
126 {
127     if (sString != null)
128     {
129         ColorDialog colorDialog = new ColorDialog();
130         if (colorDialog.ShowDialog() == DialogResult.OK)
131         {
132             sString.SetColor = colorDialog.Color; // 设置颜色
133             MessageBox.Show("颜色已经被修改");
134         }
135     }
136     else
137     {
138         MessageBox.Show("实例尚未被创建");
139         return;
140     }
141 }
142 }
```

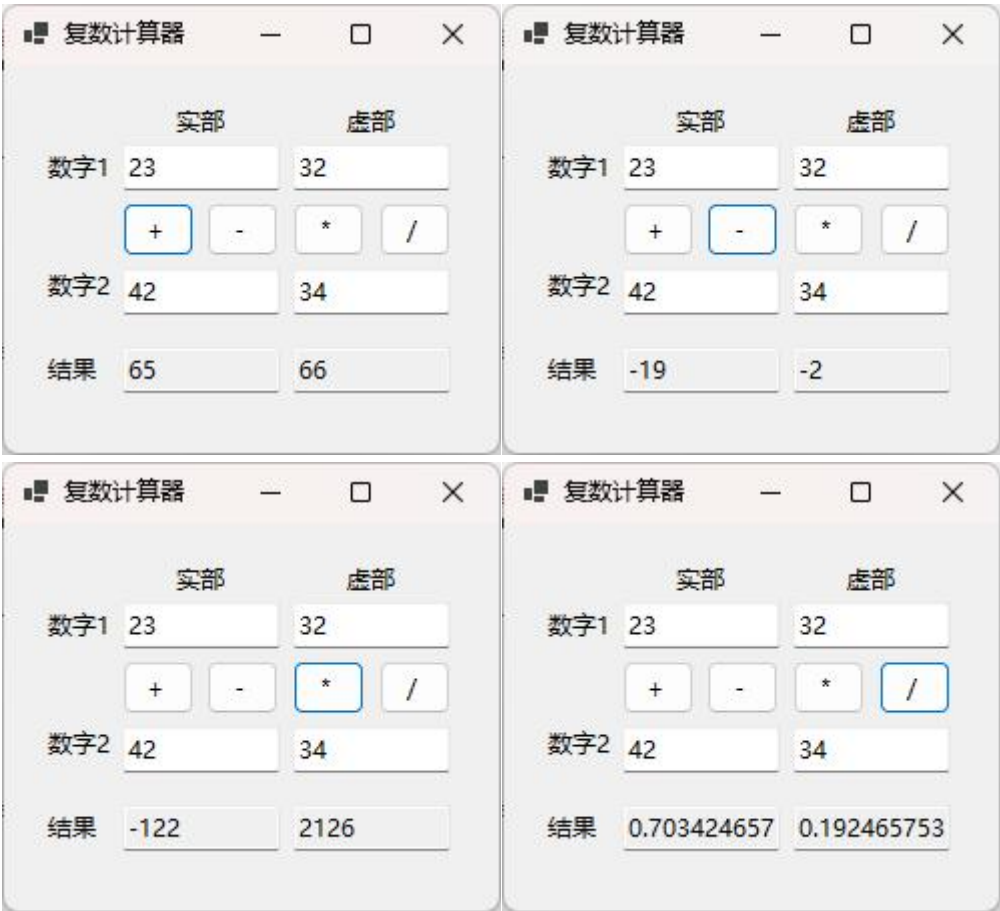
(13) 运行程序，当前效果如下。



(14) 对程序进行测试。运行结果在下一部分中。

四、程序运行结果截图

1. 复数运算计算器



2. 设计字符串类

特殊字符串类测试

当前的字符串值

XELQIW CZWQMCUHKAJOI FULAMJDVPQUCIZU
GEMRSJEZLLRNTJRKVISRDFWVMHPWDETEDD
ZJCKIKYQUIWMTLEBAESCEHEOXPECZR

创建字符串类

默认 (100大写字母)

长度

指定长度

类型 0

指定长度和类型

当前的字符串长度 100

功能测试

索引

字符

特殊字符串类测试

当前的字符串值

ak2WE702WnTfoyk5190xq25N7R1T2wbEh6vmii
thSO k2zt9Nkl5k42UnwzN1TeCr7Z

创建字符串类

默认 (100大写字母)

长度 66

指定长度

类型 0

指定长度和类型

当前的字符串长度 66

功能测试

索引

字符

特殊字符串类测试

当前的字符串值

rrdvsyslvmsbscmInccasccomtgeqkgw

创建字符串类

默认 (100大写字母)

长度 33

指定长度

类型 1

指定长度和类型

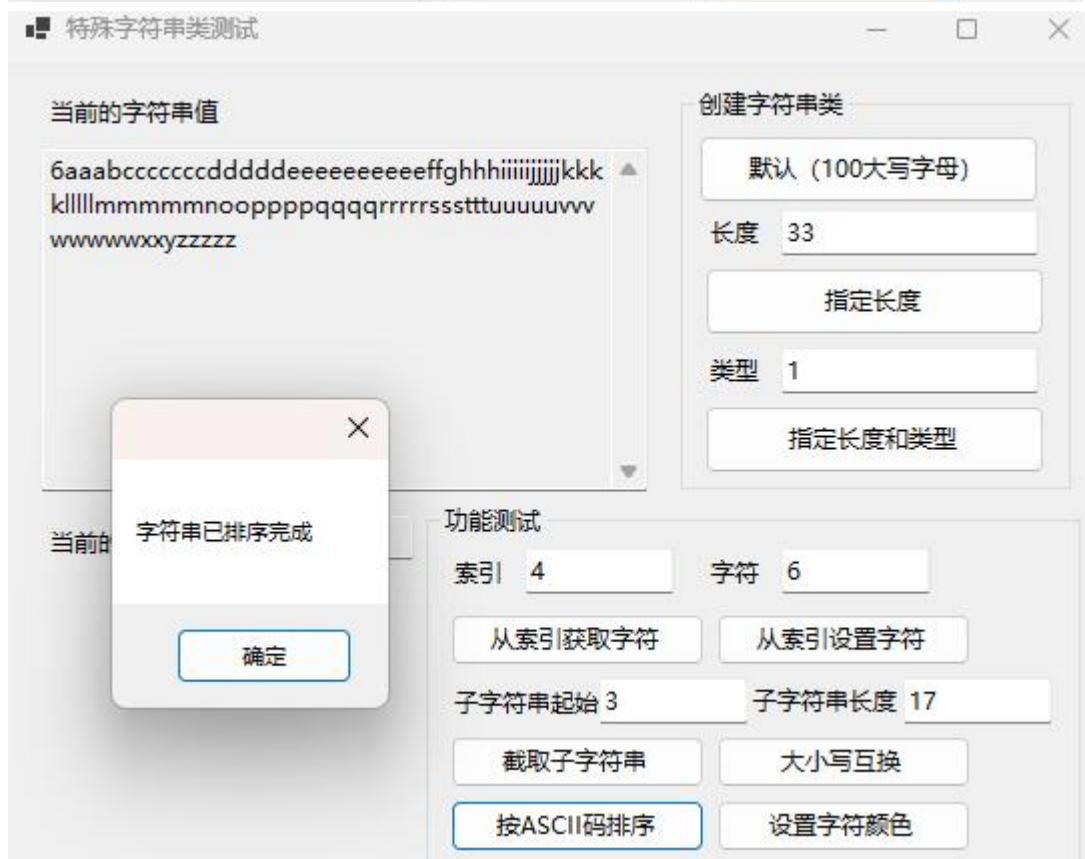
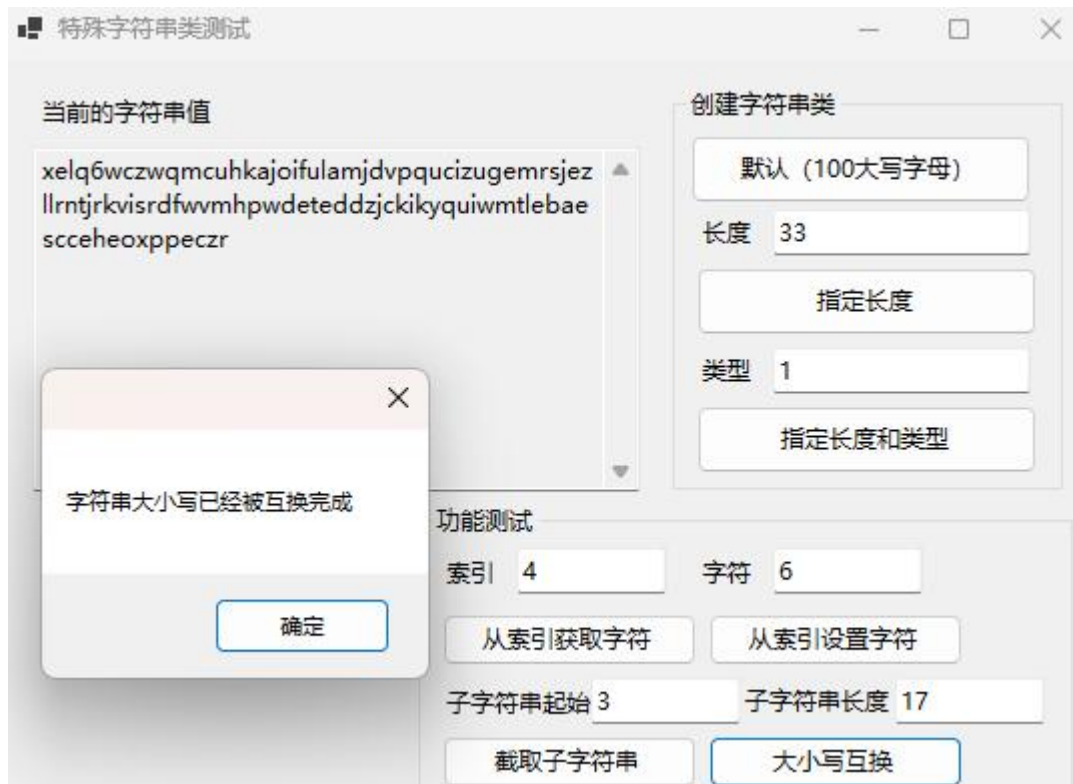
当前的字符串长度 33

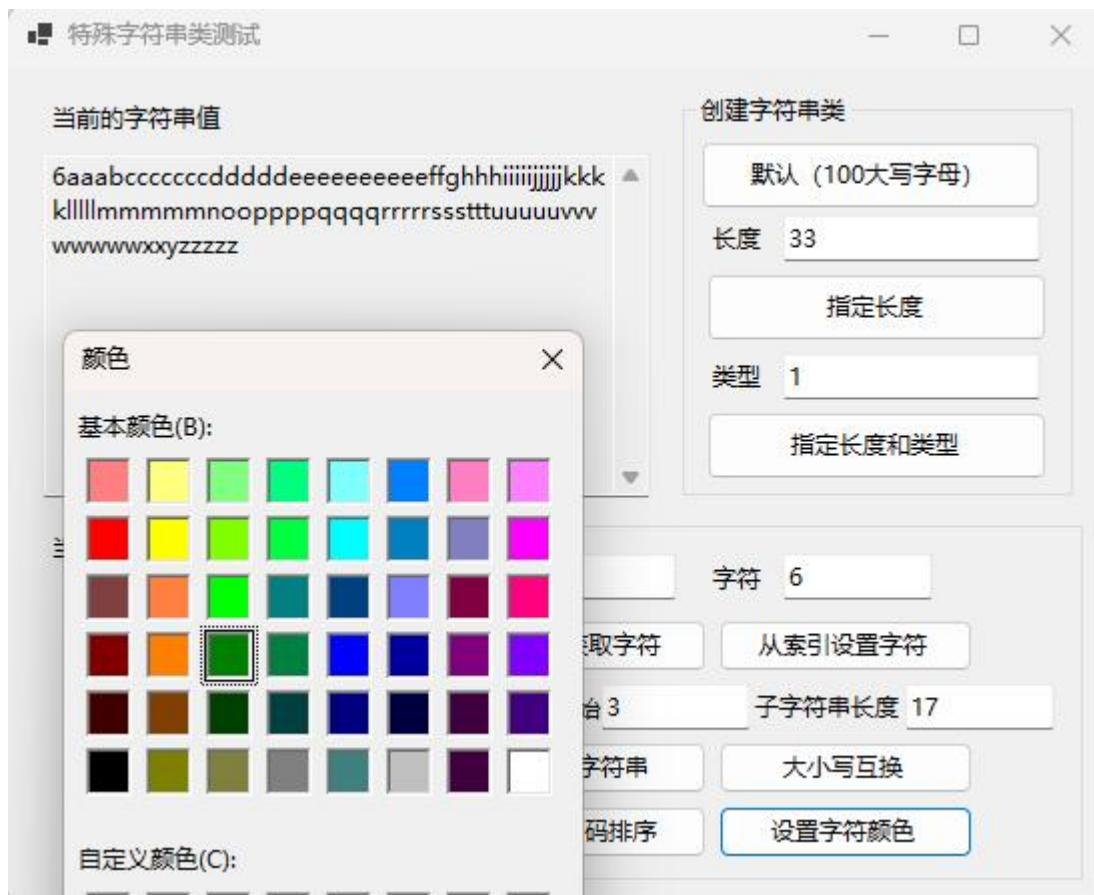
功能测试

索引

字符







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。

五、程序源代码

1. 复数运算计算器

(1) 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("/");
        }
    }
}

```

(2) 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
    }
}
}

```

2. 设计字符串类

(1) 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;
    }
}
}
}

```

(2) 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);
    }

```

数

```

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;  
    }  
}  
}
```

六、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。