# 中国矿业大学计算机学院

# 2021 级本科生课程设计报告

课程名称 程序设计实验

报告时间 2022 年 9 月

学生姓名 杨学通

学　　号 08213129

专　　业 人工智能

任课教师 杨小冬

# 《程序设计综合实践》课程报告评分表

| 序号 | 课程教学目标 | 考查方式与考查点 | 占比 | 得分 |
|---|---|---|---|---|
| 1 | **目标 1：**掌握一门计算机高级语言，并能使用特定的软件开发工具，设计、开发、调试及运行应用程序。 | 使用程序设计集成开发工具设计开发、调试应用程序，考察计算机工程基础知识 | 10% | |
| 2 | **目标 2：**针对具体的应用问题，进行功能需求分析，确定设计目标，并能绘制算法流程图。 | 系统需求、系统流程图、软件界面设计、关键类图及软件扩展描述，考察问题分析能力 | 40% | |
| 3 | **目标 3：**在进行需求分析的基础上，设计软件运行界面、关键类、编写代码，调试并正确运行满足需求的应用程序。 | 软件代码编写、调试、运行演示、系统功能扩展，考察计算机工程实践能力 | 50% | |
| 总分 | | | 100% | |

评阅人:

年　　月　　日

# 实验一 简单计算器

## 1 系统概述

本次实验基于 C++，python 和 Java 语言其中的一种设计一款支持加、减、乘、除、乘方、开方的 Windows 窗体应用。开发过程包括项目分析、界面设计、代码编写和运行调试。

通过本次实验，我们可以更直观地认识到 Windows 窗体项目开发的主要流程，为以后的软件设计夯实基础。

## 2 系统设计

### 2.1 设计目标

基本目标：

设计开发一个支持连续计算的简易计算器，其中包含加、减、乘、除、乘方、开方、退格、清空等功能。

拓展目标：

1. 设计开发一个支持代数式运算的代数式计算器，实现整个代数式的计算；
2. 设计开发一个支持计算贷款利息的计算器，包括“等额本息”和“等额本金”两种还款方式；
3. 实现三种计算器之间任意切换。

### 2.2 设计分析与算法流程

计算器的主要功能是满足对两个数之间的加减乘除，同时需要支持多步的连续运算如“1+1=”，“1+1+1+1……”等。本次选用 C 语言进行开发，开发软件为 Visual Studio 2022，开发项目为 Windows 窗体应用。开发者需要熟悉 VS 支持的窗体设计工具栏的各项工具的具体功能以及属性栏中对各种工具功能的控制。本次实验用到的工具有 textBox，和 button。同时值得注意的是我设置 Keypreview 的值为 True，开启了键盘控制计算器的数值和运算符号的输入权限，并通过编写代码实现了键盘键位和 button 的一一对应关系。对于 textBox 和 button 工具还应有以下认识：

textBox 作为 System.Windows.Forms 中的一个类，由它定义的对象的支持多种函数如 textBox.Clear（）完成 textBox 的清屏工作，textBox 类还进行了运算符的重载，可以通过“=”、“+”等对类内私有成员 Text 进行赋值或者拷贝等。

textBox 中的 Text 的数据类型是 String 型，当我们从界面获取要计算的数值时需要 Convert.ToDouble（）函数将 String 类型的数据强制转换为等值的

double 类型进行加减乘除运算，同时需要 Convert.ToString（）函数将运算得到 double 类型的数据强制转换为等效的 String 类型，传到 textBox 显示框中。

同时，打开 Keypreview 的权限，编码实现 button 与键盘键位的一一对应关系，实现键盘对计算器的控制。

相关流程图如下：

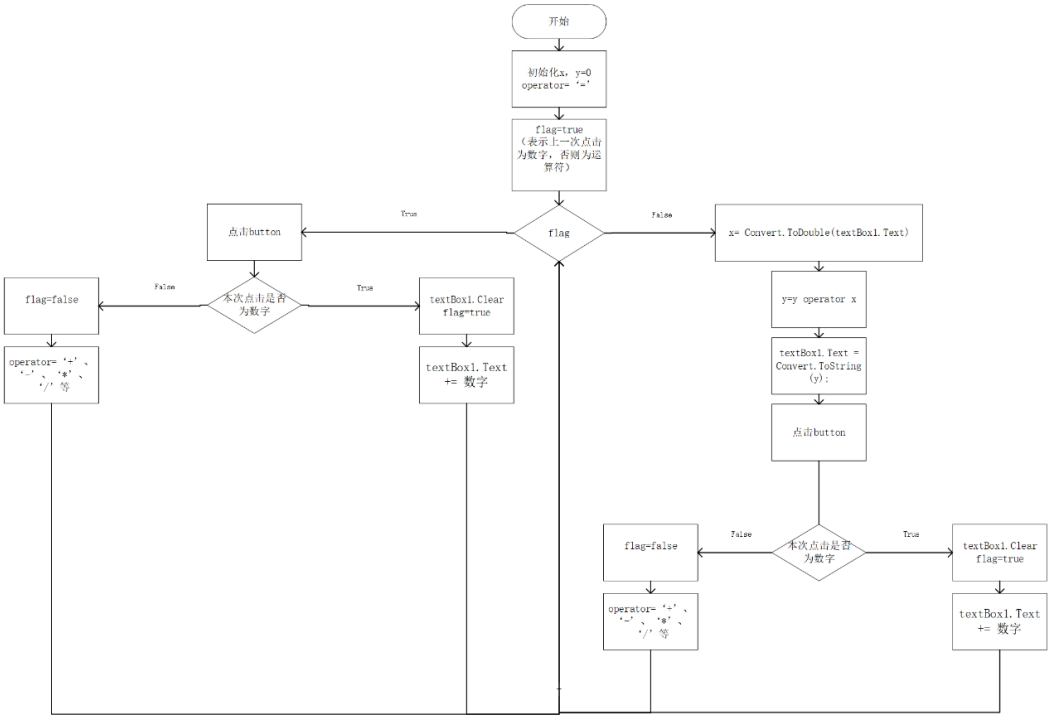


图 1 简单计算器流程图

## 2.3 界面设计

为方便用户的点击将各种按键做了放大处理，并按照点击习惯将各个按键进行了排版；为了用户更好的区别各种按键，特意将运算符和数字做了颜色改变，同时按键大小做了不同的处理缓解用户的视觉疲劳。

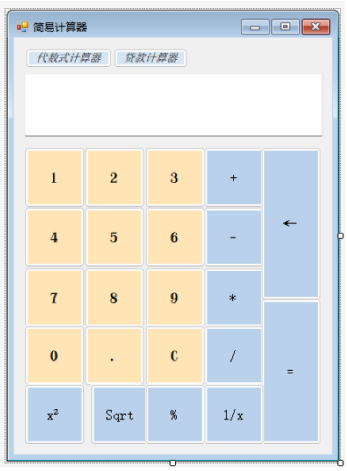


图 2 简易计算机主界面

## 2.4 关键类图

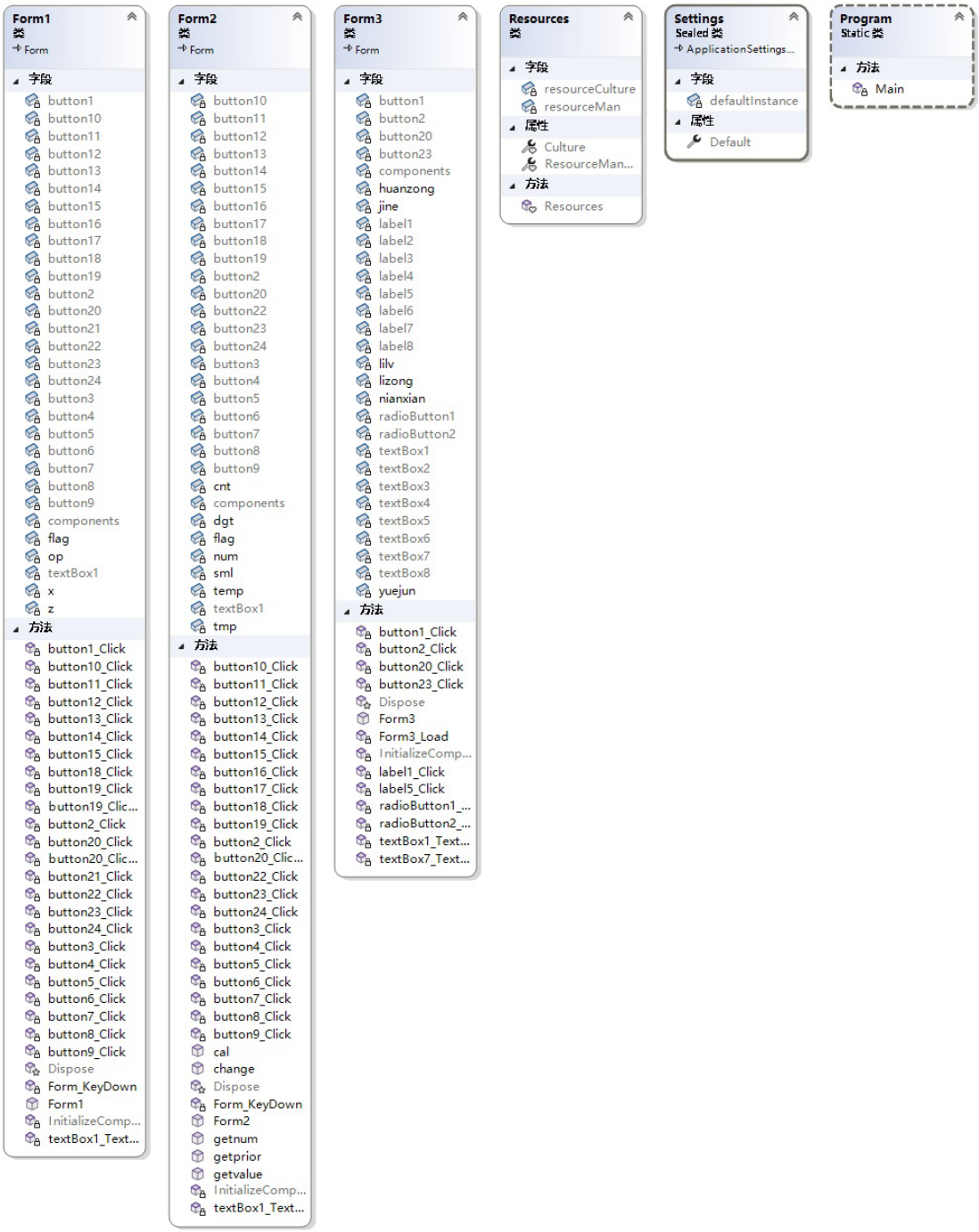


图 33 关键类图

## 2.5 关键功能代码

```
namespace 计算器
{
    public partial class Form1 : Form
```

```
{
    public Form1()
    {
        InitializeComponent();
    }
    static bool flag = true;
    static double x, z = 0;
    static char op = '=';

    private void button20_Click(object sender, EventArgs e)
    {
        textBox1.Clear();
        flag = true;
        z = 0;
        x = 0;
    }

    private void button19_Click(object sender, EventArgs e)
    {
        if (textBox1.Text != "")
        {
            textBox1.Text += ".";
        }
        else
        {
            textBox1.Text += "0.";
            flag = false;
        }

    }

    private void button15_Click(object sender, EventArgs e)
    {
        if(textBox1.Text != "")
        x = Convert.ToDouble(textBox1.Text);
        switch (op)
        {
            case '=':
                {
                    z = x;
                    break;
                }
            case '+':
                {
```

```
                z = z + x;
                break;
            }
        case '-':
            {
                z = z - x;
                break;
            }
        case '*':
            {
                z = z * x;
                break;
            }
        case '/':
            {
                z = z / x;
                break;
            }
    }
    op = '=';
    textBox1.Text = Convert.ToString(z);
    flag = true;

}

private void button14_Click(object sender, EventArgs e)
{
    if (textBox1.Text != "")
        x = Convert.ToDouble(textBox1.Text);
    switch (op)
    {
        case '=':
            {
                z = x;
                break;
            }
        case '+':
            {
                z = z + x;
                break;
            }
        case '-':
            {
                z = z - x;
```

```
                break;
            }
        case '*':
            {
                z = z * x;
                break;
            }
        case '/':
            {
                z = z / x;
                break;
            }
    }
    op = '/';
    textBox1.Text = Convert.ToString(z);
    flag = true;
}

private void button13_Click(object sender, EventArgs e)
{
    if (textBox1.Text != "")
        x = Convert.ToDouble(textBox1.Text);
    switch (op)
    {
        case '=':
            {
                z = x;
                break;
            }
        case '+':
            {
                z = z + x;
                break;
            }
        case '-':
            {
                z = z - x;
                break;
            }
        case '*':
            {
                z = z * x;
                break;
            }
```

```
            case '/':
                {
                    z = z / x;
                    break;
                }
        }
        op = '*';
        textBox1.Text = Convert.ToString(z);
        flag = true;
    }

    private void button12_Click(object sender, EventArgs e)
    {
        if (textBox1.Text != "")
            x = Convert.ToDouble(textBox1.Text);
        switch (op)
        {
            case '=':
                {
                    z = x;
                    break;
                }
            case '+':
                {
                    z = z + x;
                    break;
                }
            case '-':
                {
                    z = z - x;
                    break;
                }
            case '*':
                {
                    z = z * x;
                    break;
                }
            case '/':
                {
                    z = z / x;
                    break;
                }
        }
        op = '-';
```

```
    textBox1.Text = Convert.ToString(z);
    flag = true;
}

private void button11_Click(object sender, EventArgs e)
{
    if (textBox1.Text != "")
        x = Convert.ToDouble(textBox1.Text);
    switch (op)
    {
        case '=':
            {
                z = x;
                break;
            }
        case '+':
            {
                z = z + x;
                break;
            }
        case '-':
            {
                z = z - x;
                break;
            }
        case '*':
            {
                z = z * x;
                break;
            }
        case '/':
            {
                z = z / x;
                break;
            }
    }
    op = '+';
    textBox1.Text = Convert.ToString(z);
    flag = true;
}

private void button10_Click(object sender, EventArgs e)
{
    if (flag)
```

```
    {
        textBox1.Clear();
        flag = false;
    }
    textBox1.Text += "0";
}

private void button9_Click(object sender, EventArgs e)
{
    if (flag)
    {
        textBox1.Clear();
        flag = false;
    }
    textBox1.Text += "9";
}

private void button8_Click(object sender, EventArgs e)
{
    if (flag)
    {
        textBox1.Clear();
        flag = false;
    }
    textBox1.Text += "8";
}

private void button7_Click(object sender, EventArgs e)
{
    if (flag)
    {
        textBox1.Clear();
        flag = false;
    }
    textBox1.Text += "7";
}
private void button6_Click(object sender, EventArgs e)
{
    if (flag)
    {
        textBox1.Clear();
        flag = false;
    }
    textBox1.Text += "6";
```

```
        }

        private void button5_Click(object sender, EventArgs e)
        {
            if (flag)
            {
                textBox1.Clear();
                flag = false;
            }
            textBox1.Text += "5";
        }

        private void button4_Click(object sender, EventArgs e)
        {
            if (flag)
            {
                textBox1.Clear();
                flag = false;
            }
            textBox1.Text += "4";
        }

        private void button18_Click(object sender, EventArgs e)
        {
            if(textBox1.Text!="")
            textBox1.Text = textBox1.Text.Substring(0, textBox1.Text.Length -
1);
        }

        private void button3_Click(object sender, EventArgs e)
        {
            if (flag)
            {
                textBox1.Clear();
                flag = false;
            }
            textBox1.Text += "3";
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (flag)
            {
                textBox1.Clear();
```

```
        flag = false;
    }
    textBox1.Text += "2";
}


private void button1_Click(object sender, EventArgs e)
{
    if (flag)
    {
        textBox1.Clear();
        flag = false;
    }
    textBox1.Text += "1";
}

private void button19_Click_1(object sender, EventArgs e)
{
    if (textBox1.Text != "")
    {
        x = Convert.ToDouble(textBox1.Text);
        z = Math.Sqrt(x);
        textBox1.Text = Convert.ToString(z);
    }
}

private void button22_Click(object sender, EventArgs e)
{
    if (textBox1.Text != "")
    {
        x = Convert.ToDouble(textBox1.Text);
        z = x * x;
        textBox1.Text = Convert.ToString(z);
    }
}

private void button20_Click_1(object sender, EventArgs e)
{
    if (textBox1.Text != "")
    {
        x = Convert.ToDouble(textBox1.Text);
        z = x*100;
        textBox1.Text = Convert.ToString(z);
        textBox1.Text +="%";
```

```
            }
        }
        private void button21_Click(object sender, EventArgs e)
        {
            if (textBox1.Text != "")
            {
                x = Convert.ToDouble(textBox1.Text);
                z = 1/x;
                textBox1.Text = Convert.ToString(z);

            }
            }
private void Form_KeyDown(object sender, KeyEventArgs e)
        {
            switch (e.KeyCode)
            {
                case Keys.NumPad0:
                    {
                        button10_Click(sender, e);
                        break;
                    }
                case Keys.NumPad1:
                    {
                        button1_Click(sender, e);
                        break;
                    }
                case Keys.NumPad2:
                    {
                        button2_Click(sender, e);
                        break;
                    }
                case Keys.NumPad3:
                    {
                        button3_Click(sender, e);
                        break;
                    }
                case Keys.NumPad4:
                    {
                        button4_Click(sender, e);
                        break;
                    }
                case Keys.NumPad5:
                    {
                        button5_Click(sender, e);
```

```
            break;
        }
    case Keys.NumPad6:
        {
            button6_Click(sender, e);
            break;
        }
    case Keys.NumPad7:
        {
            button7_Click(sender, e);
            break;
        }
    case Keys.NumPad8:
        {
            button8_Click(sender, e);
            break;
        }
    case Keys.NumPad9:
        {
            button9_Click(sender, e);
            break;
        }
    case Keys.Add:
        {
            button11_Click(sender, e);
            break;
        }
    case Keys.Subtract:
        {
            button12_Click(sender, e);
            break;
        }
    case Keys.Multiply:
        {
            button13_Click(sender, e);
            break;
        }
    case Keys.Divide:
        {
            button14_Click(sender, e);
            break;
        }
    case Keys.Decimal:
        {
```

```
                button19_Click(sender, e);
                break;
            }
        case Keys.Back:
            {
                button18_Click(sender, e);
                break;
            }
    }
}
```

## 3 系统实现（运行调试）

本部分调试内容主要包括对加减乘除的单次运算进行调试，以及混合多步运算的测试以及用户错误输入时对系统优化。

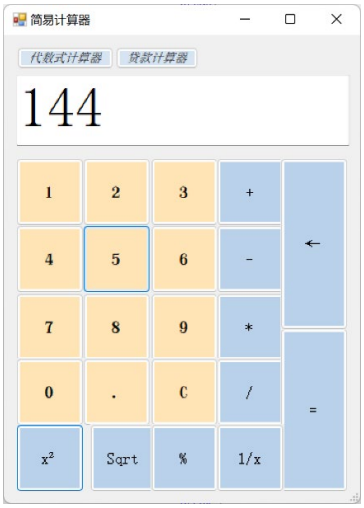


图 4 12 的平方

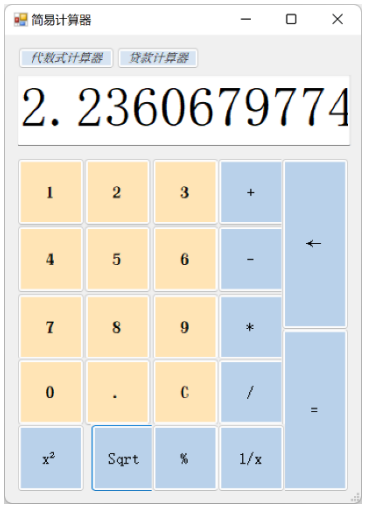


图 5 5 的开方

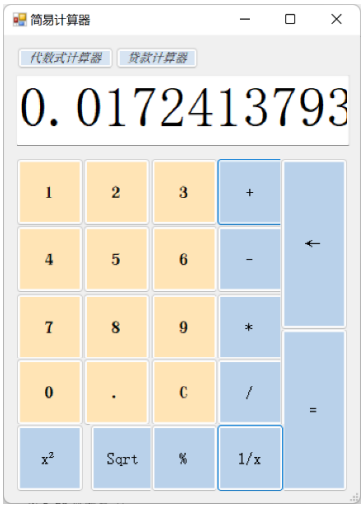


图 6 58 取倒数

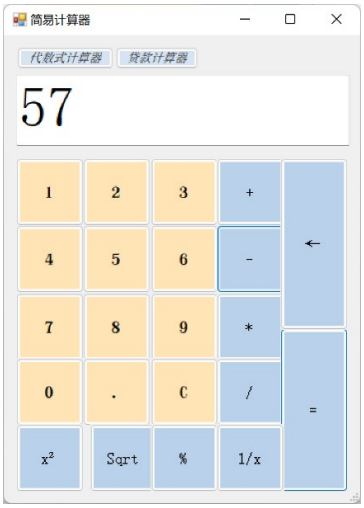


图 7 “12+5-8*57/9”
（仅代表按键顺序）

# 4 系统扩展

## 4.1 代数式计算器

### 4.1.1 算法设计

代数式计算器的难点主要在于运算符的优先级问题，解决这个问题的最好方法是借助栈将中缀表达式转化成后缀表达式，计算后缀表达式的值。

我们申请如下栈：

```
Stack<char> tmp = new Stack<char>();        //储存后缀表达式的栈
Stack<char> sml = new Stack<char>();        //储存符号的栈
Stack<double> dgt = new Stack<double>();    //储存数的栈
Stack<char> temp = new Stack<char>();       //转化栈， 将字符串转化为数值
```

我们定义如下函数：

```
void getnum(bool flag);                 //将字符串转化为数值的函数
void change(string exp);                //将中缀表达式转化为后缀表达式的函数
int getvalue();                         //计算后缀表达式的函数
int getprior(char ch);                  //获取符号优先级的函数；
void cal(double num1 , double num2 , char op);
                                        //计算函数
```

### 4.1.2 界面设计

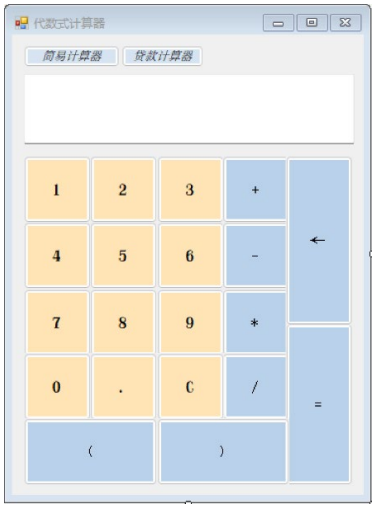


图 8 代数式计算机主界面

### 4.1.3 关键功能代码

```
namespace 计算器
{

    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }
        Stack<char> tmp = new Stack<char>();   //储存后缀表达式的栈
        Stack<char> sml = new Stack<char>();   //储存符号的栈
        Stack<double> dgt = new Stack<double>();   //储存数的栈
        Stack<char> temp = new Stack<char>();   //转化栈， 将字符串转化为数
值
        double[] num = new double[1000];
        int cnt = 0;
        bool flag = false;
        public int getprior(char ch)
        {
            if (ch == '-' || ch == '+')
                return 2;
            else if (ch == '*' || ch == '/' || ch == '%')
                return 3;
            else if (ch == '^')
                return 4;
            else if (ch == '(')
                return 5;
            else if (ch == ')')
                return 1;
            else return 0;
        }
        public void getnum(bool x)
        {
            double ans = 0;
            double t = 0;
            int counted = 0;   //计算小数点后有几位
            int index = 0;     //整数部分的下标
            while (temp.Count != 0)
            {
                if (x)    //如果有小数点
                {
                    while (temp.Peek() != '.')
```

```
            {
                int ij = temp.Peek() - '0';
                t = t + ij * Math.Pow(10, counted);
                counted++;
                temp.Pop();
            }
            temp.Pop();   //将小数点出栈
            t = t * Math.Pow(10, -counted);
            flag = false;   //小数点已出栈
        }
        int i = temp.Peek() - '0';
        ans = ans + i * Math.Pow(10, index);
        index++;
        temp.Pop();
    }
    ans += t;   //将小数部分和整数部分相加
    num[cnt] = ans;
}
public void change(string exp)
{
    int len = exp.Length;
    for (int i = 0; i < len; i++)
    {
        if (char.IsDigit(exp[i]) || exp[i] == '.')
        {
            if (exp[i] == '.')
                flag = true;
            temp.Push(exp[i]);
        }
        else
        {
            if (temp.Count != 0)
            {
                cnt++;
                getnum(flag);
                char t = '1';  //标记是数字
                tmp.Push(t);
            }
            if (sml.Count == 0 || exp[i] == '(')
                sml.Push(exp[i]);
            else
            {
                if (exp[i] == ')')
                {
```

```
                            while (sml.Peek() != '(')
                            {
                                char ch = sml.Peek();
                                tmp.Push(ch);
                                sml.Pop();
                            }
                            sml.Pop();   //将'('出栈
                        }
                        else
                        {
                            char ch = sml.Peek();
                            while (getprior(ch) >= getprior(exp[i]) && ch !=
'(')
//如果栈顶符号的优先级大于这个符号，则压入后缀栈，等于因为在前面也压入后缀栈
                            {
                                tmp.Push(ch);
                                sml.Pop();
                                if (sml.Count != 0) ch = sml.Peek();
                                else break;
                            }
                            sml.Push(exp[i]);
                        }
                    }
                }
            }
            if (temp.Count != 0)
            {
                cnt++;
                getnum(flag);
                char t = '1';
                tmp.Push(t);
            }
            while (sml.Count != 0)
            {
                char c = sml.Peek();
                tmp.Push(c);
                sml.Pop();
            }
        }
        public double getvalue()
        {
            Stack<char> s = new Stack<char>();
            while (tmp.Count != 0)
            {
```

```
                char c = tmp.Peek();
                s.Push(c);
                tmp.Pop();
            }
            int index = 1;
            while (s.Count != 0)
            {
                char c = s.Peek();
                if (char.IsDigit(c))
                {
                    dgt.Push(num[index]);
                    index++;
                    s.Pop();
                }
                else
                {
                    double num2 = dgt.Peek();
                    dgt.Pop();
                    double num1 = dgt.Peek();  //取出数值栈中的值
                    dgt.Pop();
                    cal(num1, num2, c);
                    s.Pop();   //将符号出栈
                }
            }
            return dgt.Peek();
        }
        public void cal(double num1, double num2, char op)
        {
            switch (op)
            {
                case '+': dgt.Push(num1 + num2); break;
                case '-': dgt.Push(num1 - num2); break;
                case '*': dgt.Push(num1 * num2); break;
                case '/': dgt.Push(num1 / num2); break;
                case '%': dgt.Push((int)num1 % (int)num2); break;
                case '^': dgt.Push(Math.Pow((int)num1, (int)num2)); break;
            }
        }
private void button17_Click(object sender, EventArgs e)
        {
            textBox1.Clear();
            sm1.Clear();
            tmp.Clear();
            dgt.Clear();
```

```
            temp.Clear();
            cnt = 0;
            flag = false;
        }

        private void button15_Click(object sender, EventArgs e)
        {
            string b;
            b= textBox1.Text;
            change(b);
            double result=getvalue();
            textBox1.Text = Convert.ToString(result);
            sml.Clear();
            tmp.Clear();
            dgt.Clear();
            temp.Clear();
            cnt = 0;
            flag = false;
    }
        private void button23_Click(object sender, EventArgs e)
        {
            Hide();
            Form1 form1 = new Form1();
            form1.ShowDialog();
            this.Close();
        }

        private void button20_Click_1(object sender, EventArgs e)
        {
            Hide();
            Form3 form3 = new Form3();
            form3.ShowDialog();
            this.Close();
            }
     }
}
```

### 4.1.4 运行调试

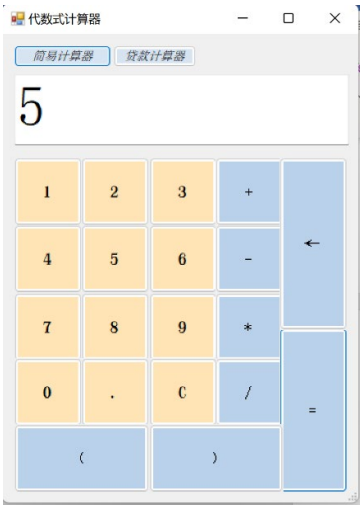


图 9 1+(2*3-4/2)

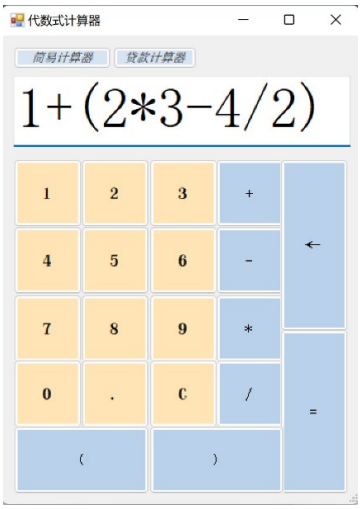


图 10 运算结果

## 4.2 贷款计算器

### 4.2.1 算法设计

贷款计算器主要根据 textBox 获得的贷款数据(贷款金额、贷款年限、利率）按照用户选择的还款方式（等额本金或等额本息）计算用户的月均还款，利息总额，还款总额等。

等额本息计算公式如下：

贷款本金×月利率×（1+月利率）^还款月数]÷［（1+月利率）^还款月数－1]；

等额本金计算公式如下：

每月月供额=(贷款本金÷还款月数)+(贷款本金-已归还本金累计额)×月利率。

### 4.2.2 界面设计

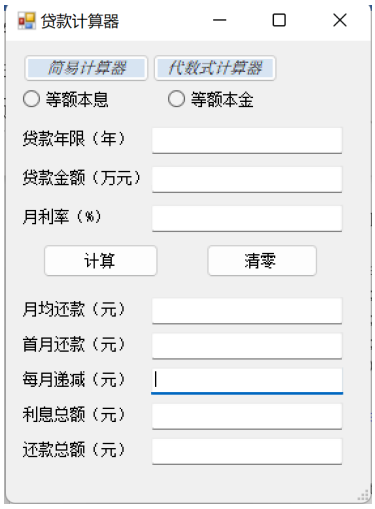


图 11 贷款计算器界面

### 4.2.3 关键功能代码

```
namespace 计算器
{

    public partial class Form3 : Form
    {
        public Form3()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            double yuejun, lizong, huanzong;
            double nianxian, jine, lilv;
            nianxian  = Convert.ToDouble(textBox1.Text)*12;
            jine = Convert.ToDouble(textBox2.Text) * 10000;
            lilv=Convert.ToDouble(textBox3.Text)/100;
            if(radioButton2.Checked)
            {
                double gg;
                gg = Math.Pow(lilv + 1, nianxian);
                yuejun = (jine * lilv *gg)/(gg-1);
                huanzong = yuejun * nianxian;
                lizong = huanzong - jine;
                textBox4.Text = Convert.ToString(yuejun);
```

```
            textBox5.Text = Convert.ToString(lizong);
            textBox6.Text = Convert.ToString(huanzong);
            textBox7.Text = Convert.ToString(yuejun);
            textBox8.Text = "0";
        }
        if (radioButton1.Checked)
        {
            double jianmian=0;
            double shouyue;
            shouyue=jine* lilv +jine / nianxian;
            for (int i=1;i<nianxian;i++)
            {
                jianmian+=i;
            }
            jianmian = jianmian * jine / nianxian * lilv;
            huanzong = shouyue*nianxian-jianmian;
            lizong = huanzong - jine;
            textBox4.Text = "***";
            textBox5.Text = Convert.ToString(lizong);
            textBox6.Text = Convert.ToString(huanzong);
            textBox7.Text = Convert.ToString(shouyue);
            textBox8.Text = Convert.ToString(jine/nianxian*lilv);
          }
    }

    private void button23_Click(object sender, EventArgs e)
    {
        Hide();
        Form1 form1 = new Form1();
        form1.ShowDialog();
        this.Close();
    }

    private void button20_Click(object sender, EventArgs e)
    {
        Hide();
        Form2 form2 = new Form2();
        form2.ShowDialog();
        this.Close();
    }

    private void button2_Click(object sender, EventArgs e)
    {
        textBox1.Clear();
```

```
            textBox2.Clear();
            textBox3.Clear();
            textBox4.Clear();
            textBox5.Clear();
            textBox6.Clear();
            textBox7.Clear();
            textBox8.Clear();
        }
    }
    }
```

### 4.2.4 运行调试


图 12 等额本息计算结果

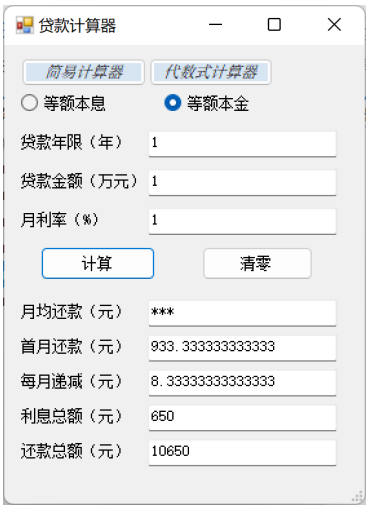


图 13 等额本金计算结果

## 5 总结

C++基于对象的开发过程极大程度上方便了开发者的开发工作，为开发者省去了基础控件开发的时间。本次实验即复习到了 C++的基本语法，同时学到了 textBox，button 等控件的基本操作，实现了不同界面之间的自由切换。同时将数据结构中利用栈实现代数式的四则运算应用到 winform 中，实现计算器更强大的功能。当然，本次实验还存在一些不足需要改进，如代数式运算时对特殊情况的负数计算还需仔细斟酌。总体来说本次的实验收获很大，希望自己能在以后的学习中再接再厉。

# 中国矿业大学计算机学院

# 2021 级本科生课程设计报告

课程名称 程序设计实验

报告时间 2022 年 9 月

学生姓名 杨学通

学　　号 08213129

专　　业 人工智能

任课教师 杨小冬

# 《程序设计综合实践》课程报告评分表

| 序号 | 课程教学目标 | 考查方式与考查点 | 占比 | 得分 |
|---|---|---|---|---|
| 1 | **目标 1：**掌握一门计算机高级语言，并能使用特定的软件开发工具，设计、开发、调试及运行应用程序。 | 使用程序设计集成开发工具设计开发、调试应用程序，考察计算机工程基础知识 | 10% | |
| 2 | **目标 2：**针对具体的应用问题，进行功能需求分析，确定设计目标，并能绘制算法流程图。 | 系统需求、系统流程图、软件界面设计、关键类图及软件扩展描述，考察问题分析能力 | 40% | |
| 3 | **目标 3：**在进行需求分析的基础上，设计软件运行界面、关键类、编写代码，调试并正确运行满足需求的应用程序。 | 软件代码编写、调试、运行演示、系统功能扩展，考察计算机工程实践能力 | 50% | |
| 总分 | | | 100% | |

评阅人:

年　　月　　日

# 实验二 拼图游戏

## 1 系统概述

本次实验基于C#语言其中的一种设计一款拼图游戏。完成图片加载，图片切割，图片重新排序，以及图片的拖动等基本功能。

通过本次实验，可以加强我们对Picturebox、Button、Label等控件的认识和了解，同时可以学会如何为一个程序加载图片，按钮等资源，为程序的UI设计打下基础。

## 2 系统设计

### 2.1 设计目标

基本目标：设计开发一个拼图游戏，可以完成图片切割，图片重新排列，图片的拖动等功能。

项目还可以支持玩家加载自己喜欢的图片，支持查看原图的功能。

拓展目标：

让游戏具备挑战模式，可以让玩家自行选择图片的切割份数如3×3、5×5等，让游戏更具有挑战性。

### 2.2 设计分析与算法流程

拼图游戏的运行过程大致如下：首先将图片加载至Picturebox，完成图片的切割；然后将切割好的图片，按照顺序编号，这一步骤相当于将一张图片的各个部分排列到了一个平面坐标轴上，而图片的编号，即为图片的坐标；打乱分割好的图片，图片背后的编号呈乱序状态；玩家通过拖动鼠标改变任意两张图片，互换其位置，当所有的图片按照打乱前的顺序排列时，即为完成拼图，系统提示“成功!”字样。

相关流程图如下：

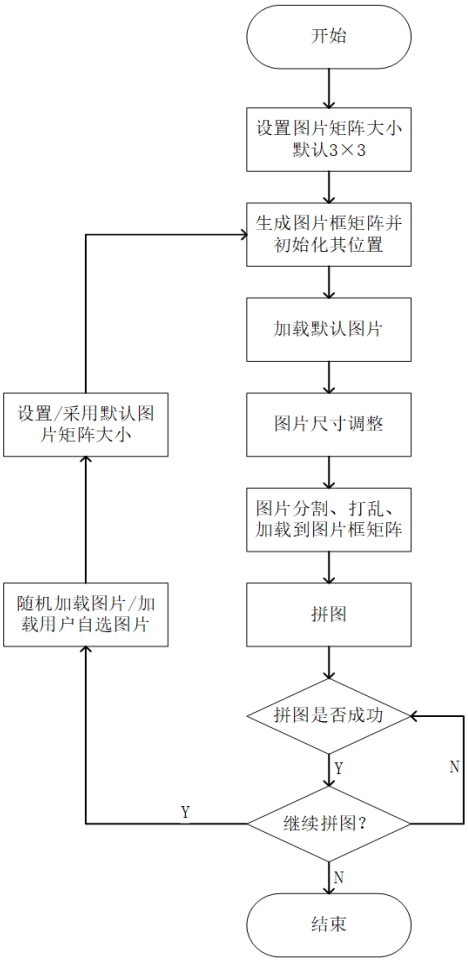


图 4 拼图游戏流程图

## 2.3 界面设计

本实验的界面按左右顺序排列，左边是 Picturebox，右边是四个按钮，分别对应“查看原图”，“试玩新图”，“切换图片”，“图片重排”功能。左右两边通过 Splitcontainer 分隔开。


图 5 拼图游戏主界面

## 2.4 关键类图

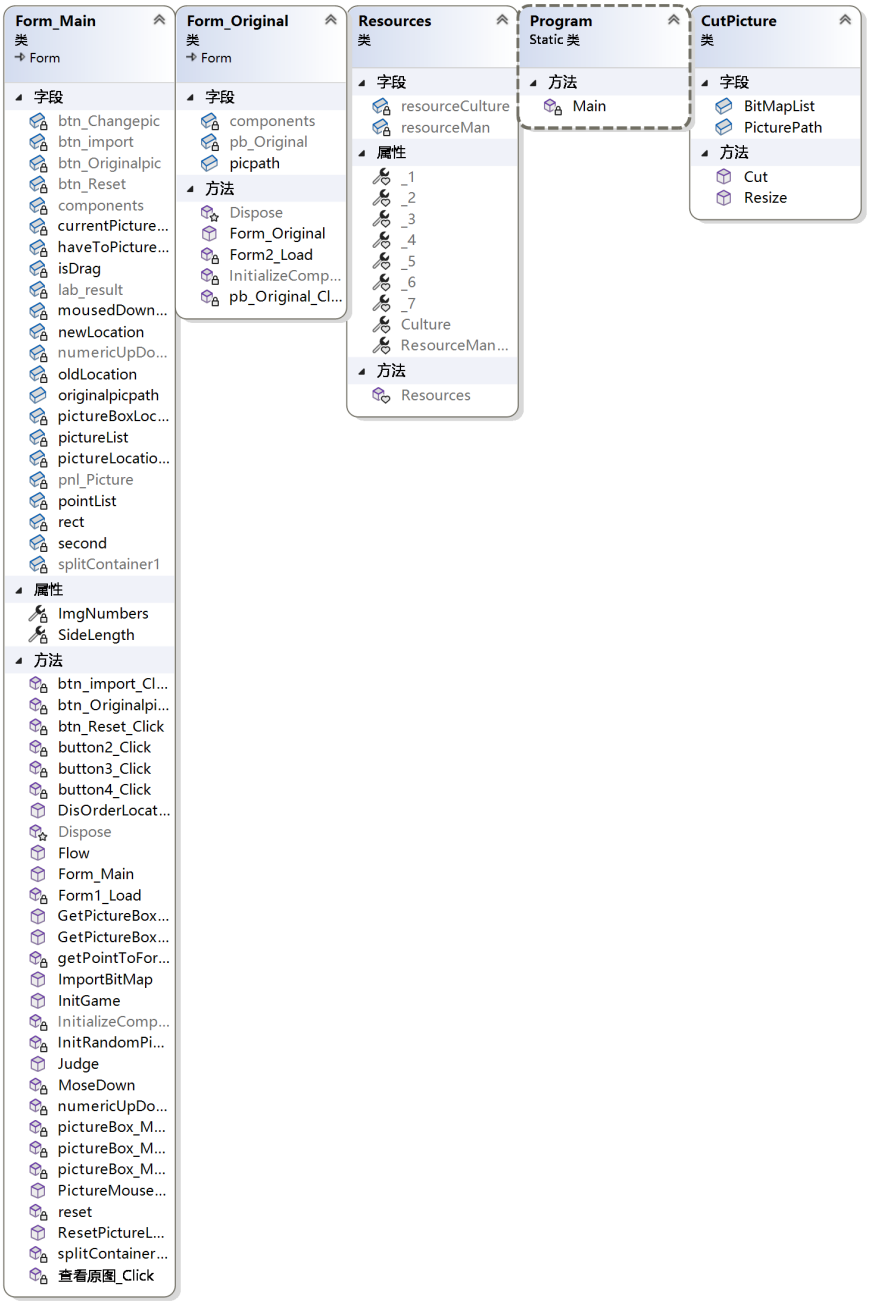


图 36 关键类图

## 2.5 关键功能代码

```
namespace 拼图游戏
{
    public partial class Form_Main : Form
    {
```

```
        public Form_Main()
        {
            InitializeComponent();
            InitGame();
        }

        PictureBox[] pictureList = null;
        SortedDictionary<string, Bitmap> pictureLocationDict = new
SortedDictionary<string, Bitmap>();
        Point[] pointList = null;//Location List
        SortedDictionary<string, PictureBox> pictureBoxLocationDict = new
SortedDictionary<string, PictureBox>();//图片控件字典
        int second = 0;//拼图时间
        PictureBox currentPictureBox = null;//所拖拽的图片
        PictureBox haveToPictureBox = null;//被迫需要移动的图片
        Point oldLocation = Point.Empty;//原位置
        Point newLocation = Point.Empty;//新位置
        Point mousedDownPoint = Point.Empty;//鼠标按下坐标（control 控件
的相对坐标）
        Rectangle rect = Rectangle.Empty; //显示拖动效果的矩形
        bool isDrag = false;//是否正在拖拽
        public string originalpicpath;//图片位置
        /// <summary>
        /// 每个方向上需要切割成的块数
        /// </summary>
        private int ImgNumbers
        {
            get
            {
                return (int)this.numericUpDown1.Value;
            }
        }
        /// <summary>
        /// 要切割的正方形图片边长
```

```
        /// </summary>
        private int SideLength
        {
            get
            {
                return 600 / this.ImgNumbers;
            }
        }
        private void InitRandomPictureBox()
        {
            pnl_Picture.Controls.Clear();
            pictureList = new PictureBox[ImgNumbers * ImgNumbers];
            pointList = new Point[ImgNumbers * ImgNumbers];
            for (int i = 0; i < this.ImgNumbers; i++)
            {
                for (int j = 0; j < this.ImgNumbers; j++)
                {
                    PictureBox pic = new PictureBox();
                    pic.Name = "pictureBox" + (j + i * ImgNumbers +
1).ToString();
                    pic.Location = new Point(j * SideLength, i *
SideLength);
                    pic.Size = new Size(SideLength, SideLength);
                    pic.Visible = true;
                    pic.BorderStyle = BorderStyle.FixedSingle;
                    pic.MouseDown += new
MouseEventHandler(pictureBox_MouseDown);
                    pic.MouseMove += new
MouseEventHandler(pictureBox_MouseMove);
                    pic.MouseUp += new
MouseEventHandler(pictureBox_MouseUp);
                    pnl_Picture.Controls.Add(pic);
                    pictureList[j + i * ImgNumbers] = pic;
                    pointList[j+i*ImgNumbers]=new Point(j*SideLength,
```

```
i * SideLength);
                    }
                }
            }
            public void Flow(string path, bool disorder)
            {
                InitRandomPictureBox();
                Image bm = CutPicture.Resize(path, 600, 600);
                CutPicture.BitMapList = new List<Bitmap>();
                for (int y = 0; y < 600; y += SideLength)
                {
                    for (int x = 0; x < 600; x += SideLength)
                    {
                        Bitmap temp = CutPicture.Cut(bm, x, y, SideLength,
SideLength);
                        CutPicture.BitMapList.Add(temp);
                    }
                }
                ImportBitMap(disorder);
            }
            public Point[] DisOrderLocation()
            {
                Point[] tempArray = (Point[])pointList.Clone();
                for (int i = tempArray.Length - 1; i > 0; i--)
                {
                    Random rand = new Random();
                    int p = rand.Next(i);
                    Point temp = tempArray[p];
                    tempArray[p] = tempArray[i];
                    tempArray[i] = temp;
                }
                return tempArray;

            }
```

```
        public void ResetPictureLocation()
        {
            Point[] temp = DisOrderLocation();
            int i = 0;
            foreach (PictureBox item in pictureList)
            {
                item.Location = temp[i];
                i++;
            }
        }
        public void ImportBitMap(bool disorder)
        {
            try
            {
                int i = 0;
                foreach (PictureBox item in pictureList)
                {
                    Bitmap temp = CutPicture.BitMapList[i];
                    item.Image = temp;
                    i++;
                }
                if (disorder)
                    ResetPictureLocation();
            }
            catch (Exception exp)
            {
                Console.WriteLine(exp.Message);
            }
        }
        public void InitGame()
        {
            if (!Directory.Exists(Application.StartupPath.ToString() +
"\\Picture"))
            {
```

```
Directory.CreateDirectory(Application.StartupPath.ToString() + "\\Picture");

Properties.Resources._1.Save(Application.StartupPath.ToString() +
"\\Picture\\1.jpg");

Properties.Resources._2.Save(Application.StartupPath.ToString() +
"\\Picture\\2.jpg");

Properties.Resources._3.Save(Application.StartupPath.ToString() +
"\\Picture\\3.jpg");

Properties.Resources._4.Save(Application.StartupPath.ToString() +
"\\Picture\\4.jpg");

Properties.Resources._5.Save(Application.StartupPath.ToString() +
"\\Picture\\5.jpg");

Properties.Resources._6.Save(Application.StartupPath.ToString() +
"\\Picture\\6.jpg");

Properties.Resources._7.Save(Application.StartupPath.ToString() +
"\\Picture\\7.jpg");
                }
                Random r = new Random();
                int i = r.Next(7);
                originalpicpath = Application.StartupPath.ToString() +
"\\Picture\\" + i.ToString() + ".jpg";
                Flow(originalpicpath, true);
            }
            public PictureBox GetPictureBoxByLocation(int x, int y)
            {
                PictureBox pic = null;
                foreach (PictureBox item in pictureList)
```

```
                {
                    if (x > item.Location.X && y > item.Location.Y &&
item.Location.X + SideLength > x && item.Location.Y + SideLength > y)
                    {
                        pic = item;
                    }
                }
                return pic;
            }

            public PictureBox GetPictureBoxByHashCode(string hascode)
            {
                PictureBox pic = null;
                foreach (PictureBox item in pictureList)
                {
                    if (hascode == item.GetHashCode().ToString())
                    {
                        pic = item;
                    }
                }
                return pic;
            }

            private void pictureBox_MouseDown(object sender, MouseEventArgs e)
            {
                oldLocation = new Point(e.X, e.Y);
                currentPictureBox =
GetPictureBoxByHashCode(sender.GetHashCode().ToString());
                MoseDown(currentPictureBox, sender, e);

            }

            private void MoseDown(PictureBox pic, object sender,
MouseEventArgs e)
```

```
        {
            if (e.Button == MouseButtons.Left)
            {
                oldLocation = e.Location;
                rect = pic.Bounds;
            }
        }

        private void pictureBox_MouseMove(object sender,MouseEventArgs e)
        {
            if(e.Button == MouseButtons.Left)
            {
                isDrag= true;
                rect.Location = getPointToForm(new Point(e.Location.X -
oldLocation.X, e.Location.Y - oldLocation.Y));
                this.Refresh();
            }
        }

        private Point getPointToForm(Point p)
        {
            return this.PointToClient(pictureList[0].PointToScreen(p));
        }

        private void reset()
        {
            mousedDownPoint=Point.Empty;
            rect = Rectangle.Empty;
            isDrag = false;
        }

        private void pictureBox_MouseUp(object sender,MouseEventArgs e)
        {
            oldLocation =new Point(currentPictureBox.Location.X,
```

```
currentPictureBox.Location.Y);
                if(oldLocation.X +e.X>600||oldLocation.Y
+e.Y>600||oldLocation.X+e.X<0||oldLocation.Y+e.Y<0)
                {
                    return;
                }
                haveToPictureBox = GetPictureBoxByLocation(oldLocation.X +
e.X, oldLocation.Y + e.Y);
                newLocation = new Point(haveToPictureBox.Location.X,
haveToPictureBox.Location.Y);
                haveToPictureBox.Location = oldLocation;
                PictureMouseUp(currentPictureBox, sender, e);
                if(Judge())
                {
                    lab_result.Text = "成功！";
                }
            }

            public void PictureMouseUp(PictureBox pic,object
sender,MouseEventArgs e)
            {
                if(e.Button==MouseButtons.Left)
                {
                    if(isDrag)
                    {
                        isDrag=false;
                        pic.Location = newLocation;
                        this.Refresh();
                    }
                    reset();
                }
            }

            public bool Judge()
```

```
        {
            bool result = true;
            int i = 0;
            foreach(PictureBox item in pictureList)
            {
                if (item.Location != pointList[i])
                {
                    result = false;
                }
                i++;
            }
            return result;
        }

        private void button3_Click(object sender, EventArgs e)
        {
            Random r=new Random();
            int i=r.Next(7);
            originalpicpath = Application.StartupPath.ToString() +
"\\Picture\\" + i.ToString() + ".jpg";
            Flow(originalpicpath, true);
        }

        private void btn_import_Click(object sender, EventArgs e)
        {
            OpenFileDialog new_picture = new OpenFileDialog();
            if (new_picture.ShowDialog()==DialogResult.OK)
            {
                lab_result.Text = "";
                originalpicpath = new_picture.FileName;
                CutPicture.PicturePath = new_picture.FileName;
                Flow(CutPicture.PicturePath, true);
            }
        }
```

```
        private void btn_Reset_Click(object sender, EventArgs e)
        {
            Flow(originalpicpath, true);
        }
        private void btn_Originalpic_Click(object sender, EventArgs e)
        {
            Form_Original original = new Form_Original();
            original.picpath = originalpicpath;
            original.ShowDialog();
        }
    }
```

## 3 系统实现（运行调试）

本部分主要是运行程序并测试程序的各项功能是否正常。测试内容包括“查看原图”、“图片重排”、“切换图片”、“试玩新图”四项基本功能。下面是程序运行时的截图：

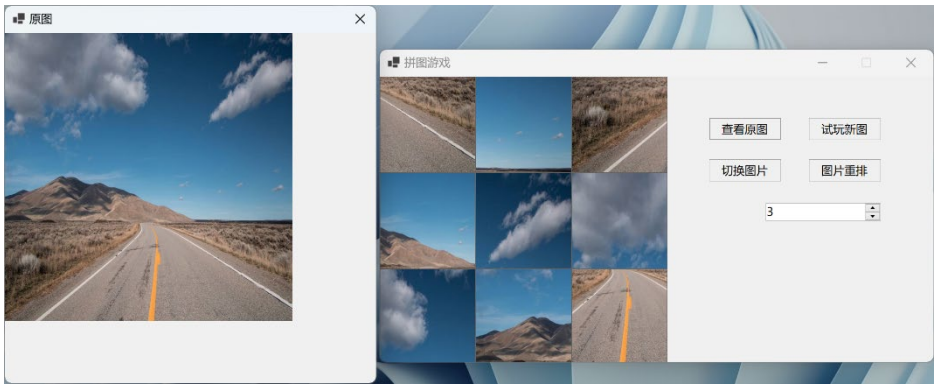


图 4 查看原图

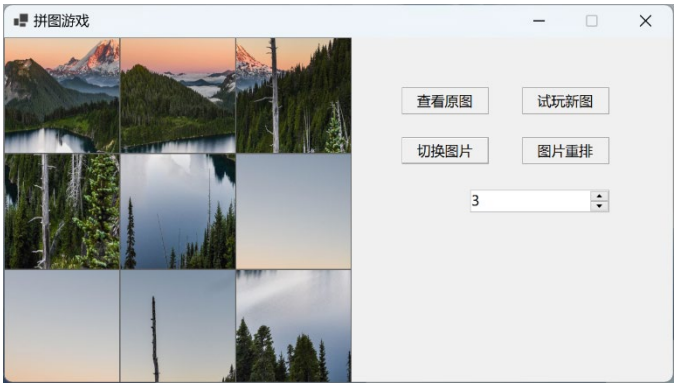


图 5 切换图片

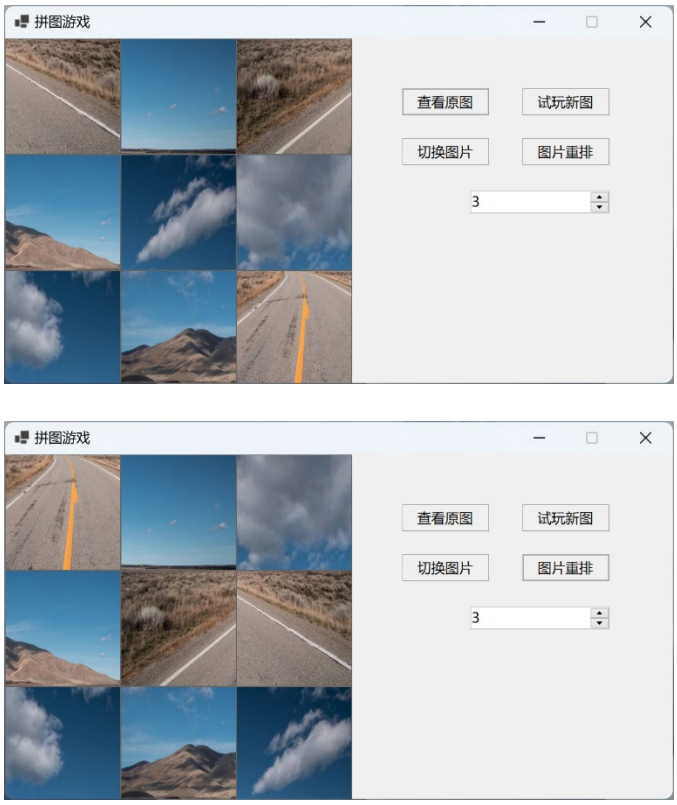


图 6 图片重排


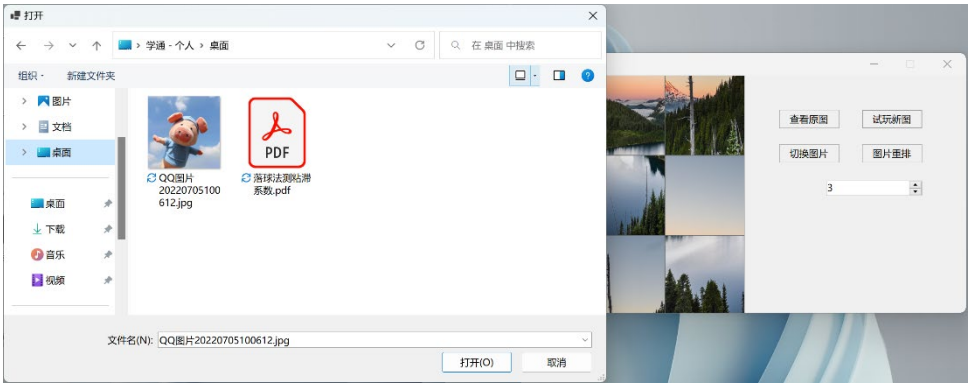


图 7 试玩新图

# 4 系统扩展

## 4.1 图片动态切割

### 4.1.1 算法设计

图片动态切割，实际就是更改 NumericUpDown 的值，重新加载到获取切割数函数 private int ImgNumbers（）中，在图片切割函数中，调用获取切割数函数，从而将图片切割成想要的份数。

### 4.1.2 界面设计


图 8 图片动态切割界面

### 4.1.3 关键功能代码

```
/// <summary>
/// 每个方向上需要切割成的块数
/// </summary>
 private int ImgNumbers
 {
     get
     {
         return (int)this.numericUpDown1.Value;
     }
 }
 /// <summary>
 /// 要切割的正方形图片边长
 /// </summary>
 private int SideLength
 {
```

```
            get
            {
                return 600 / this.ImgNumbers;
            }
        }
        private void InitRandomPictureBox()
        {
            pnl_Picture.Controls.Clear();
            pictureList = new PictureBox[ImgNumbers * ImgNumbers];
            pointList = new Point[ImgNumbers * ImgNumbers];
            for (int i = 0; i < this.ImgNumbers; i++)
            {
                for (int j = 0; j < this.ImgNumbers; j++)
                {
                    PictureBox pic = new PictureBox();
                    pic.Name = "pictureBox" + (j + i * ImgNumbers +
1).ToString();
                    pic.Location = new Point(j * SideLength, i * SideLength);
                    pic.Size = new Size(SideLength, SideLength);
                    pic.Visible = true;
                    pic.BorderStyle = BorderStyle.FixedSingle;
                    pic.MouseDown += new
MouseEventHandler(pictureBox_MouseDown);
                    pic.MouseMove += new
MouseEventHandler(pictureBox_MouseMove);
                    pic.MouseUp += new MouseEventHandler(pictureBox_MouseUp);
                    pnl_Picture.Controls.Add(pic);
                    pictureList[j + i * ImgNumbers] = pic;
                    pointList[j + i * ImgNumbers] = new Point(j * SideLength,
i * SideLength);
                }
            }
        }
        public void Flow(string path, bool disorder)
        {
            InitRandomPictureBox();
            Image bm = CutPicture.Resize(path, 600, 600);
            CutPicture.BitMapList = new List<Bitmap>();
            for (int y = 0; y < 600; y += SideLength)
            {
                for (int x = 0; x < 600; x += SideLength)
                {
                    Bitmap temp = CutPicture.Cut(bm, x, y, SideLength,
SideLength);
```

```
                CutPicture.BitMapList.Add(temp);
            }
        }
        ImportBitMap(disorder);
    }
    public Point[] DisOrderLocation()
    {
        Point[] tempArray = (Point[])pointList.Clone();
        for (int i = tempArray.Length - 1; i > 0; i--)
        {
            Random rand = new Random();
            int p = rand.Next(i);
            Point temp = tempArray[p];
            tempArray[p] = tempArray[i];
            tempArray[i] = temp;
        }
        return tempArray;

    }
    public void ResetPictureLocation()
    {
        Point[] temp = DisOrderLocation();
        int i = 0;
        foreach (PictureBox item in pictureList)
        {
            item.Location = temp[i];
            i++;
        }
    }
    public void ImportBitMap(bool disorder)
    {
        try
        {
            int i = 0;
            foreach (PictureBox item in pictureList)
            {
                Bitmap temp = CutPicture.BitMapList[i];
                item.Image = temp;
                i++;
            }
            if (disorder)
                ResetPictureLocation();
        }
        catch (Exception exp)
```

```
        {
            Console.WriteLine(exp.Message);
        }
    }
```

### 4.1.4 运行调试


图 9 3×3 切割


图 10 5×5 切割

## 5 总结

通过本次实验，我掌握了 Picturebox 控件的应用，通过该控件，我完成了图片的加载，通过编写各种函数，完成了图片的切割和图片的重新排列，通过 Resources 文件夹，为项目添加各种图片资源，让程序的功能复杂度更大，让程序更有价值。本次通过 Winform 制作一个拼图游戏，十分具有趣味性，对当前游戏的开发具有一定的了解，收获颇丰。

# 中国矿业大学计算机学院

# 2021 级本科生课程设计报告

课程名称 程序设计实验

报告时间 2022 年 9 月

学生姓名 杨学通

学　　号 08213129

专　　业 人工智能

任课教师 杨小冬

# 《程序设计综合实践》课程报告评分表

| 序号 | 课程教学目标 | 考查方式与考查点 | 占比 | 得分 |
|---|---|---|---|---|
| 1 | **目标 1：**掌握一门计算机高级语言，并能使用特定的软件开发工具，设计、开发、调试及运行应用程序。 | 使用程序设计集成开发工具设计开发、调试应用程序，考察计算机工程基础知识 | 10% | |
| 2 | **目标 2：**针对具体的应用问题，进行功能需求分析，确定设计目标，并能绘制算法流程图。 | 系统需求、系统流程图、软件界面设计、关键类图及软件扩展描述，考察问题分析能力 | 40% | |
| 3 | **目标 3：**在进行需求分析的基础上，设计软件运行界面、关键类、编写代码，调试并正确运行满足需求的应用程序。 | 软件代码编写、调试、运行演示、系统功能扩展，考察计算机工程实践能力 | 50% | |
| 总分 | | | 100% | |

评阅人:

年　　月　　日

# 实验三 多文本文档编辑器

## 1 系统概述

本次实验利用 VS 中的 Winform 设计一款类似于 office Word 功能的多文本文档编辑器。编辑器可以支持文档的新建、保存、打开等功能。同时，可以对加载到工作区的文档进行编辑，其中包括文档字体的调整（如字体、颜色、大小）、复制、粘贴、下划线、删除线、倾斜等。

实验项目可以同时打开多个文档，同时处理多个文档、为了便于用户操作，系统为用户提供了窗口的布局快捷键。

## 2 系统设计

### 2.1 设计目标

基本目标：

1. 设计一款多文档文本编辑器，实现文档的新建、保存、打开等功能；
2. 支持用户对文档编辑的基本功能，如字体调整、复制、粘贴等功能；
3. 多窗口、多文本同时编辑。

拓展目标：

1. 实现撤销、重做、设置文字对齐方式、复制、粘贴、剪切、查找、替换等功能；
2. 分情况确定是否需要弹出保存对话框进行保存。

### 2.2 设计分析与算法流程

打开已存在文档：可以打开以.txt 结尾的文本文档，读取文件内容并将其显示在文本框中。

新建文档：新建一个子窗体，其文本编辑框处于空白状态。

保存文档：将文本编辑框中的文本保存到.txt 文件中。

设置字体：对选中的文本设置字体。

设置粗体：选中的文本粗体显示。

设置斜体：选中的文本斜体显示。

设置下划线：对选中的文本设置下划线。

设置窗体排列方式：对子窗体设置排列方式（窗口层叠、水平平铺、垂直平铺）。

相关流程图如下：

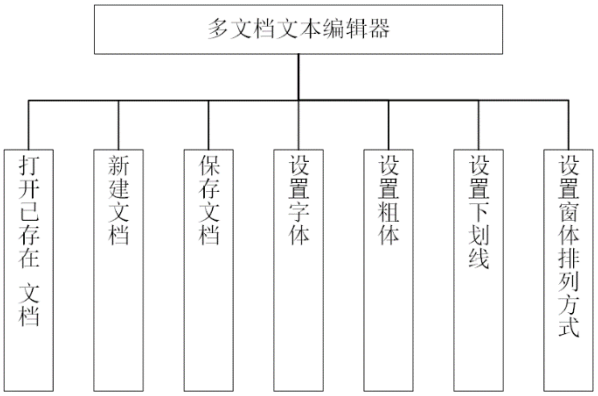


图 7 多文档文本编辑器流程图

## 2.3 界面设计

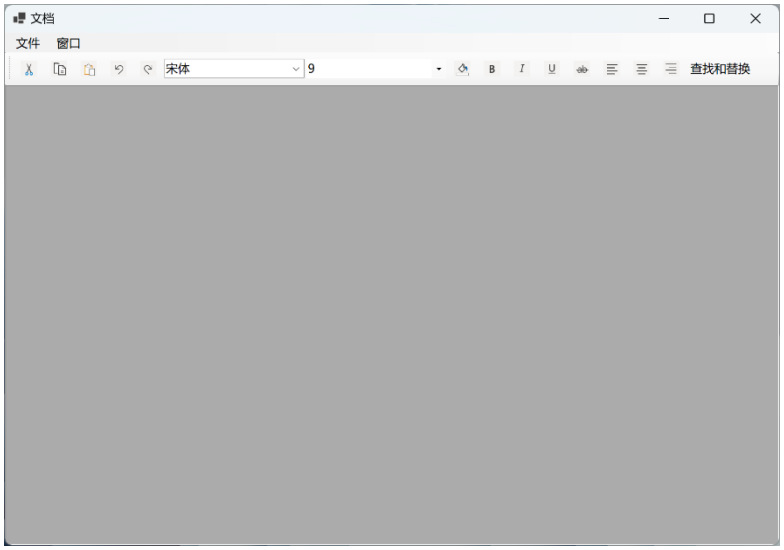


图 8 多文档文本编辑器主界面

多文档文本编辑器的界面设计参考于 office Word 的布局方式。界面通过添加 MenuStrip 添加了编辑器的属性栏，涵盖了两大主菜单，一个是对对文件操作的文件栏，另一个是对界面操作的窗口栏。项目通过添加 ToolStrip 添加了编辑器的工具栏，该部分大多数工具为 Button 和 SplitButton。

为了方便用户辨识各种工具的作用,我给各个按钮添加了 ToolTipText 即鼠标指针落在工具上，会给用户提示该工具的功能。

我还为按钮进行了简单的UI 设计,将相关的 png 图片加载进 Resources 文件夹中，让界面更简洁，便于辨识。

## 2.4 关键类图

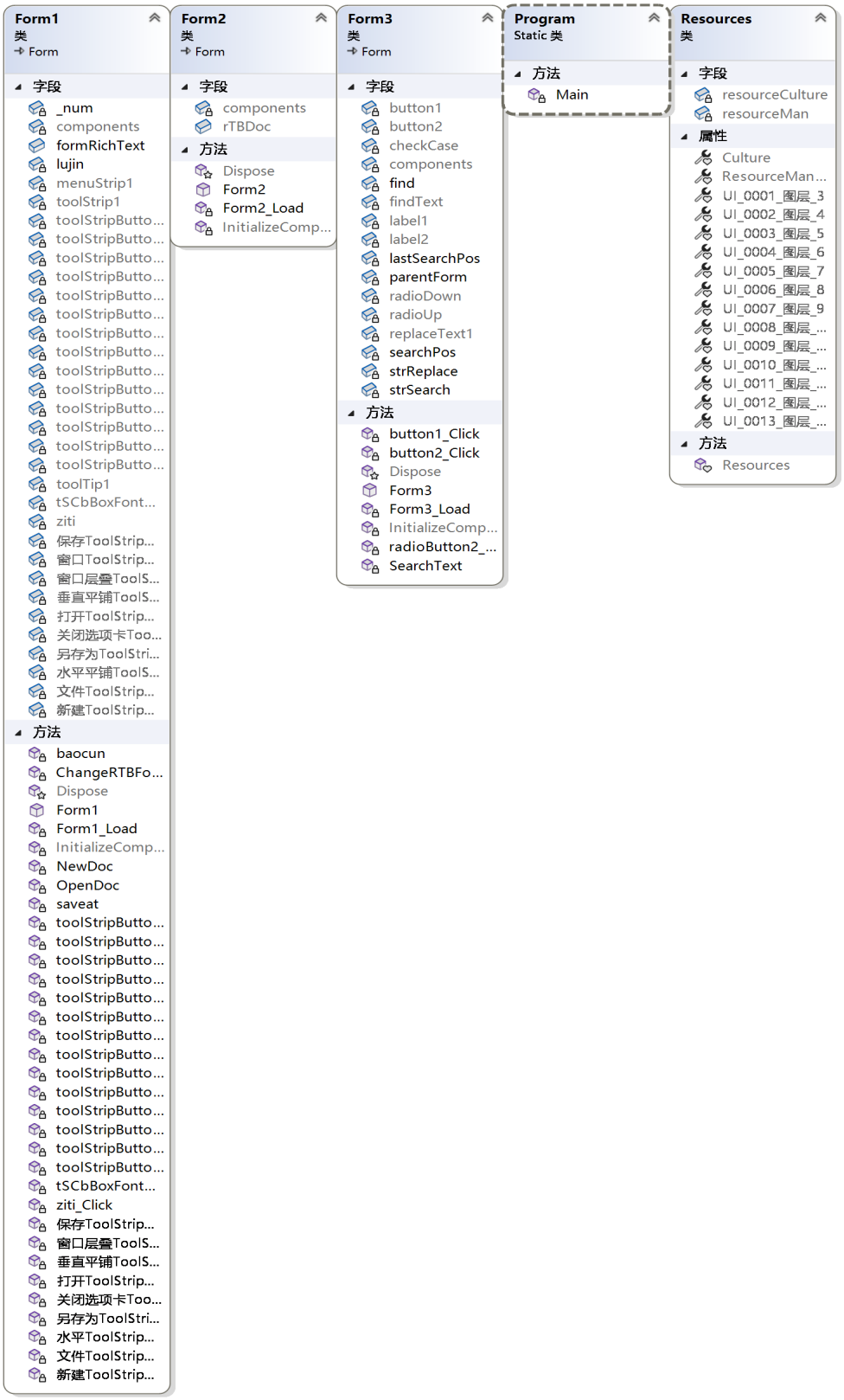


图 3 关键类图

## 2.5 关键功能代码

```
namespace 多文本文档编辑器
{
    public partial class Form1 : Form
    {
        private int _num = 1;
        string lujin = "untitle";
        public Form1()
        {
            InitializeComponent();
        }

        private void ChangeRTBFontStyle(RichTextBox rtb, FontStyle style)
        {
            if (style != FontStyle.Bold && style != FontStyle.Italic && style !=
FontStyle.Underline && style != FontStyle.Strikeout)
                throw new System.InvalidProgramException("字体格式错误");
            RichTextBox tempRTB = new RichTextBox();//保存选中文本的副本
            int curRtbStart = rtb.SelectionStart;//记录选取文本起始位置
            int len = rtb.SelectionLength;//记录选取文本长度
            int tempRtbStart = 0;
            Font font = rtb.SelectionFont;//记录当前字体
            if (len <= 0 && font != null)//若未选中文本
            {
                if (style == FontStyle.Bold && font.Bold || style ==
FontStyle.Italic && font.Italic || style == FontStyle.Underline &&
font.Underline || style == FontStyle.Strikeout && font.Strikeout)
                    rtb.SelectionFont = new Font(tSCbBoxFontName.Text,
Convert.ToSingle(ziti.Text), font.Style ^ style);
                else if (style == FontStyle.Bold && !font.Bold || style ==
FontStyle.Italic && !font.Italic || style == FontStyle.Underline
&& !font.Underline || style == FontStyle.Strikeout && !font.Strikeout)
                    rtb.SelectionFont = new Font(tSCbBoxFontName.Text,
Convert.ToSingle(ziti.Text), font.Style | style);
                return;
            }
            tempRTB.Rtf = rtb.SelectedRtf;
            tempRTB.Select(len - 1, 1);
            Font tempFont = (Font)tempRTB.SelectionFont.Clone();
            for (int i = 0; i < len; i++)//若选中文本
            {
                tempRTB.Select(tempRtbStart + i, 1);
                if (style == FontStyle.Bold && tempFont.Bold || style ==
```

```
FontStyle.Italic && tempFont.Italic || style == FontStyle.Strikeout &&
tempFont.Strikeout || style == FontStyle.Underline && tempFont.Underline)
                    tempRTB.SelectionFont = new Font(tempRTB.SelectionFont,
tempRTB.SelectionFont.Style ^ style);
                else if (style == FontStyle.Bold && !tempFont.Bold || style ==
FontStyle.Italic && !tempFont.Italic || style == FontStyle.Underline
&& !tempFont.Underline || style == FontStyle.Strikeout && !tempFont.Strikeout)
                    tempRTB.SelectionFont = new Font(tempRTB.SelectionFont,
tempRTB.SelectionFont.Style | style);
            }
            tempRTB.Select(tempRtbStart, len);
            rtb.SelectedRtf = tempRTB.SelectedRtf;
            rtb.Select(curRtbStart, len);
            rtb.Focus();
            tempRTB.Dispose();
        }
        private void 文件ToolStripMenuItem_Click(object sender, EventArgs e)
        {

        }

        private void Form1_Load(object sender, EventArgs e)
        {
            //目的为获取系统所有字体，并将字体名称显示在下拉框中
            tSCbBoxFontName.Items.Clear(); //清空下拉框
            InstalledFontCollection ifc = new InstalledFontCollection();//获
取系统的所有字体
            FontFamily[] ffs = ifc.Families;
            foreach (FontFamily ff in ffs)//将获取的字体放入下拉框
                tSCbBoxFontName.Items.Add(ff.GetName(1));
            LayoutMdi(MdiLayout.Cascade);
            WindowState = FormWindowState.Maximized;
        }

        private void tSCbBoxFontName_Click(object sender, EventArgs e)
        {
            RichTextBox tempRTB = new RichTextBox();//保存富文本副本
            int RtbStart =
((Form2)this.ActiveMdiChild).rTBDoc.SelectionStart;//获取选中文本起始位置
            int len = ((Form2)this.ActiveMdiChild).rTBDoc.SelectionLength;//
获取选中文本长度
            int tempRtbStart = 0;
            Font font = ((Form2)this.ActiveMdiChild).rTBDoc.SelectionFont;//
记录当前字体
```

```
            if (len <= 0 && null != font)//若未选中文本
            {
                ((Form2)this.ActiveMdiChild).rTBDoc.SelectionFont = new
Font(tSCbBoxFontName.Text,
                Convert.ToSingle(ziti.Text), tempRTB.SelectionFont.Style);
                return;
            }
            tempRTB.Rtf = ((Form2)this.ActiveMdiChild).rTBDoc.SelectedRtf;
            for (int i = 0; i < len; i++)//若选中文本
            {
                tempRTB.Select(tempRtbStart + i, 1);
                tempRTB.SelectionFont = new Font(tSCbBoxFontName.Text,
Convert.ToSingle(ziti.Text), tempRTB.SelectionFont.Style);
            }
            tempRTB.Select(tempRtbStart, len);
            ((Form2)this.ActiveMdiChild).rTBDoc.SelectedRtf =
tempRTB.SelectedRtf;
            ((Form2)this.ActiveMdiChild).rTBDoc.Select(RtbStart, len);
            ((Form2)this.ActiveMdiChild).rTBDoc.Focus();
            tempRTB.Dispose();//释放
        }

        private void ziti_Click(object sender, EventArgs e)
        {
            //该部分代码与上一个类似，相关注释就不再赘述
            if (this.MdiChildren.Count() > 0)
            {
                RichTextBox tempRTB = new RichTextBox();
                int RtbStart =
((Form2)this.ActiveMdiChild).rTBDoc.SelectionStart;
                int len =
((Form2)this.ActiveMdiChild).rTBDoc.SelectionLength;
                int tempRtbStart = 0;
                if (len <= 0)
                {
                    ((Form2)this.ActiveMdiChild).rTBDoc.SelectionFont = new
Font(tSCbBoxFontName.Text, Convert.ToSingle(ziti.Text),
tempRTB.SelectionFont.Style);
                    return;
                }
                tempRTB.Rtf =
((Form2)this.ActiveMdiChild).rTBDoc.SelectedRtf;
                for (int i = 0; i < len; i++)
                {
```

```
                    tempRTB.Select(tempRtbStart + i, 1);
                    tempRTB.SelectionFont = new Font(tSCbBoxFontName.Text,
Convert.ToSingle(ziti.Text), tempRTB.SelectionFont.Style);
                }
                tempRTB.Select(tempRtbStart, len);
                ((Form2)this.ActiveMdiChild).rTBDoc.SelectedRtf =
tempRTB.SelectedRtf;
                ((Form2)this.ActiveMdiChild).rTBDoc.Select(RtbStart, len);
                ((Form2)this.ActiveMdiChild).rTBDoc.Focus();
                tempRTB.Dispose();
            }
        }

        private void toolStripButton12_Click(object sender, EventArgs e)
        {
            ColorDialog ColorDlg = new ColorDialog();
            //禁止使用自定义颜色
            ColorDlg.AllowFullOpen = false;
            //提供自己给定的颜色
            ColorDlg.CustomColors = new int[] { 6916092, 15195440, 16107657,
1836924, 3758726, 12566463, 7526079, 7405793, 6945974, 241502, 2296476, 5130294,
3102017, 7324121, 14993507, 11730944 };//设置颜色
            ColorDlg.ShowHelp = true;
            ColorDlg.ShowDialog();//打开颜色选择窗口
            if (this.MdiChildren.Count() > 0)
            {
                RichTextBox tempRTB = new RichTextBox();
                int RtbStart =
((Form2)this.ActiveMdiChild).rTBDoc.SelectionStart;
                int len =
((Form2)this.ActiveMdiChild).rTBDoc.SelectionLength;
                int tempRtbStart = 0;
                if (len <= 0)
                {
                    ((Form2)this.ActiveMdiChild).rTBDoc.SelectionColor =
ColorDlg.Color;
                    return;
                }
                tempRTB.Rtf =
((Form2)this.ActiveMdiChild).rTBDoc.SelectedRtf;


                for (int i = 0; i < len; i++)
                {
```

```
                    tempRTB.Select(tempRtbStart + i, 1);
                    tempRTB.SelectionColor = ColorDlg.Color;
                }
                tempRTB.Select(tempRtbStart, len);
                ((Form2)this.ActiveMdiChild).rTBDoc.SelectedRtf =
tempRTB.SelectedRtf;
                ((Form2)this.ActiveMdiChild).rTBDoc.Select(RtbStart, len);
                ((Form2)this.ActiveMdiChild).rTBDoc.Focus();
                tempRTB.Dispose();
            }
        }

        private void toolStripButton8_Click(object sender, EventArgs e)
        {
            ChangeRTBFontStyle(((Form2)this.ActiveMdiChild).rTBDoc,
FontStyle.Underline);
        }

        private void toolStripButton6_Click(object sender, EventArgs e)
        {
            ChangeRTBFontStyle(((Form2)this.ActiveMdiChild).rTBDoc,
FontStyle.Bold);
        }

        private void toolStripButton7_Click(object sender, EventArgs e)
        {
            ChangeRTBFontStyle(((Form2)this.ActiveMdiChild).rTBDoc,
FontStyle.Italic);
        }

        private void toolStripButton9_Click(object sender, EventArgs e)
        {
            ChangeRTBFontStyle(((Form2)this.ActiveMdiChild).rTBDoc,
FontStyle.Strikeout);
        }

        private void toolStripButton4_Click(object sender, EventArgs e)
        {
            ((Form2)this.ActiveMdiChild).rTBDoc.Undo();
        }

        private void toolStripButton5_Click(object sender, EventArgs e)
        {
            ((Form2)this.ActiveMdiChild).rTBDoc.Redo();
```

```
        }

        private void toolStripButton1_Click(object sender, EventArgs e)
        {
            Clipboard.SetData(DataFormats.Rtf,
((Form2)this.ActiveMdiChild).rTBDoc.SelectedRtf);//复制RTF数据到剪贴板
            ((Form2)this.ActiveMdiChild).rTBDoc.SelectedRtf = "";//再把当前
选取的RTF内容清除掉
        }

        private void toolStripButton2_Click(object sender, EventArgs e)
        {
            Clipboard.SetData(DataFormats.Rtf,
((Form2)this.ActiveMdiChild).rTBDoc.SelectedRtf);//复制RTF数据到剪贴板
        }

        private void toolStripButton3_Click(object sender, EventArgs e)
        {
            ((Form2)this.ActiveMdiChild).rTBDoc.Paste();//把剪贴板上的数据
粘贴到目标
        }

        private void toolStripButton10_Click(object sender, EventArgs e)
        {
            ((Form2)this.ActiveMdiChild).rTBDoc.SelectionAlignment =
HorizontalAlignment.Left;
        }

        private void toolStripButton13_Click(object sender, EventArgs e)
        {
            ((Form2)this.ActiveMdiChild).rTBDoc.SelectionAlignment =
HorizontalAlignment.Center;
        }

        private void toolStripButton14_Click(object sender, EventArgs e)
        {
            ((Form2)this.ActiveMdiChild).rTBDoc.SelectionAlignment =
HorizontalAlignment.Right;
        }
        private void NewDoc()
        {
            Form2 fd = new Form2();//新建窗口
            fd.MdiParent = this;//该新建窗口的父亲是当前的主窗口
            fd.Text = "untitle" + "文档" + _num;//文件名
```

```
            fd.WindowState = FormWindowState.Maximized;
            fd.Show();//展示该文档
            fd.Activate();//激活窗体并给予焦点
            _num++;
        }
        private void OpenDoc()
        {
            OpenFileDialog openfiledialog1 = new OpenFileDialog();
            openfiledialog1.Filter = "RTF格式(*.rft)|*.rft|文本文件
(*.txt)|*.text|所有文件(*.*)|*.*";
            openfiledialog1.Multiselect = false;//禁止多选
            if (openfiledialog1.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    NewDoc();

((Form2)this.ActiveMdiChild).rTBDoc.LoadFile(openfiledialog1.FileName,
RichTextBoxStreamType.RichText);
                    string localFilePath =
openfiledialog1.FileName.ToString();
                    lujin = localFilePath;
                    ((Form2)this.ActiveMdiChild).Text =
localFilePath.Substring(localFilePath.LastIndexOf("\\") + 1);//设置文档名字
                }
                catch
                {
                    MessageBox.Show("打开失败！", "错误",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
            openfiledialog1.Dispose();
        }
        private void saveat()
        {
            SaveFileDialog savefiledialog1 = new SaveFileDialog();//打开另存
为的框框
            savefiledialog1.Filter = "RTF格式(*.rtf)|*.rtf|文本文件
(*.txt)|*text";
            if (savefiledialog1.ShowDialog() == DialogResult.OK)
            {
                try
                {
```

```
((Form2)this.ActiveMdiChild).rTBDoc.SaveFile(savefiledialog1.FileName,
RichTextBoxStreamType.RichText);
                    MessageBox.Show("保存成功！", "", MessageBoxButtons.OK,
MessageBoxIcon.None);
                }
                catch
                {
                    MessageBox.Show("保存失败！", "错误",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
            savefiledialog1.Dispose();
        }

        private void baocun()
        {
            string ext =
((Form2)this.ActiveMdiChild).Text.Substring(((Form2)this.ActiveMdiChild).rT
BDoc.Text.LastIndexOf(".") + 1);
            ext = ext.ToLower();//转小写
            ((Form2)this.ActiveMdiChild).rTBDoc.SaveFile(lujin,
RichTextBoxStreamType.RichText);
            ((Form2)this.ActiveMdiChild).rTBDoc.Modified = false;
            MessageBox.Show("保存成功", ext);
        }

        private void 新建ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            NewDoc();
        }
        private void 打开ToolStripMenuItem_Click(object sender, EventArgs e)
        {
           OpenDoc();
        }

        private void 保存ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (((Form2)this.ActiveMdiChild).Text.Length < 7)//如果名字长度
小于等于7，那就是“打开”，就执行保存
                baocun();
            else
            {
                if (((Form2)this.ActiveMdiChild).Text.Substring(0, 7) ==
"untitle")//如果名字前七个元素等于untitle，那就是“新建”，就执行另存为
```

```
                saveat();
            else
                baocun();
        }
    }
    private void 另存为ToolStripMenuItem_Click(object sender, EventArgs
e)
    {
        saveat();
    }

    private void 关闭选项卡ToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        if (MessageBox.Show("确定要关闭当前文档吗？", "提示",
MessageBoxButtons.OKCancel, MessageBoxIcon.Information) == DialogResult.OK)
            ((Form2)this.ActiveMdiChild).Close();
    }
    private void 窗口层叠ToolStripMenuItem_Click(object sender, EventArgs
e)
    {
        LayoutMdi(MdiLayout.Cascade);
        this.窗口层叠ToolStripMenuItem.Checked = true;
        this.垂直平铺ToolStripMenuItem.Checked = false;
        this.水平平铺ToolStripMenuItem.Checked = false;
    }
    private void 水平ToolStripMenuItem_Click(object sender, EventArgs e)
    {
        LayoutMdi(MdiLayout.TileHorizontal);
        this.窗口层叠ToolStripMenuItem.Checked = false;
        this.垂直平铺ToolStripMenuItem.Checked = false;
        this.水平平铺ToolStripMenuItem.Checked = true;
    }
    private void 垂直平铺ToolStripMenuItem_Click(object sender, EventArgs
e)
    {
        LayoutMdi(MdiLayout.TileVertical);
        this.窗口层叠ToolStripMenuItem.Checked = false;
        this.垂直平铺ToolStripMenuItem.Checked = true;
        this.水平平铺ToolStripMenuItem.Checked = false;
    }
    public Form2 formRichText;
    private void toolStripButton11_Click(object sender, EventArgs e)
    {
```

```
            formRichText = (Form2)this.ActiveMdiChild;
            Form3 formFind = new Form3(formRichText);
            formFind.Show();
        }
    }
}
```

## 3 系统实现（运行调试）

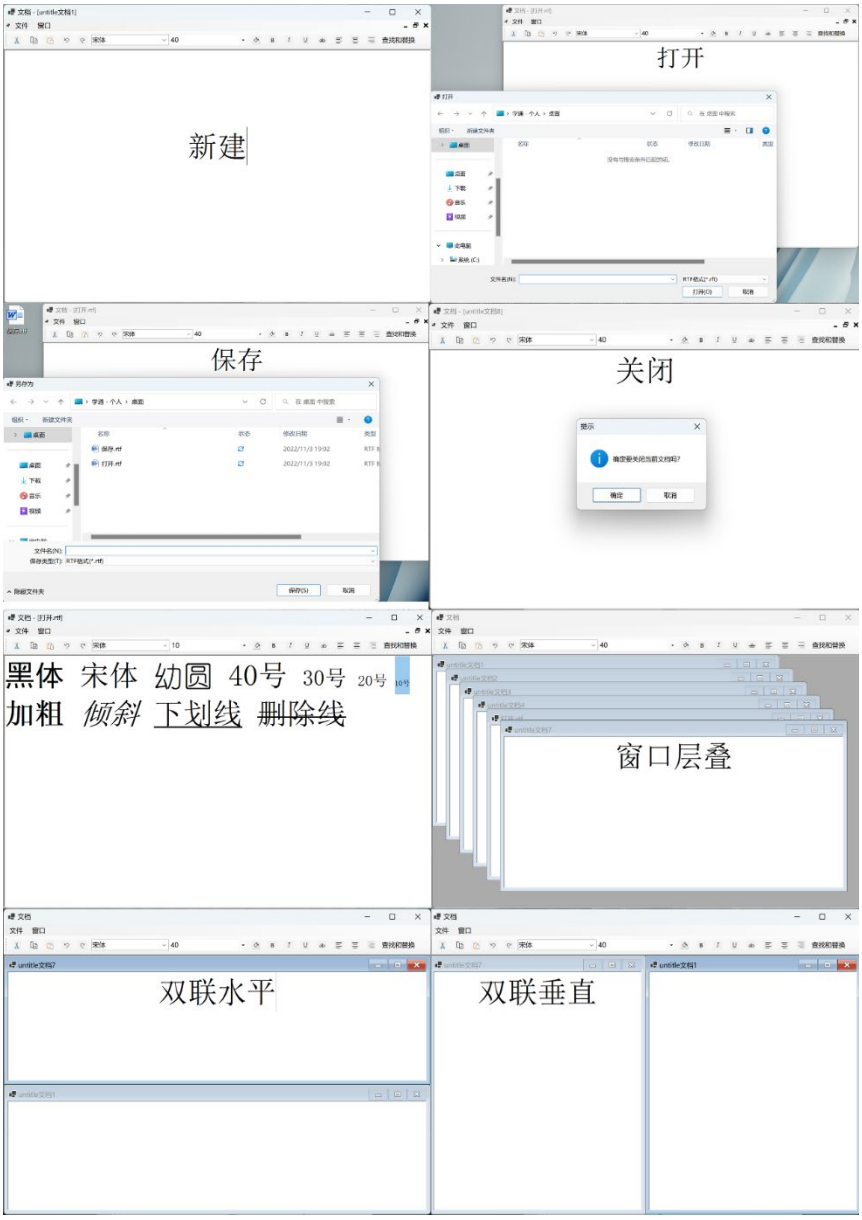


图 4 多文档文本编辑器各项功能调试

# 4 系统扩展

## 4.1 文本编辑器功能拓展

### 4.1.1 算法设计

多文档的文本编辑器一般还会具有撤销、重做、设置文字对齐方式、复制、粘贴、剪切、等功能。

该部分的功能实现主要靠调用 system 中的各种函数，如将数据加载到剪切板的函数，撤销函数，重做函数等。

查找替换功能的实现，通过编写查找函数实现。函数需要确定当前的所在文档的位置，确定向下还是向上查找文档，找到一个关键词后，更新当前位置为起始位置，如此下去。

替换函数是基于查找函数实现的，当前所在文档的位置加上关键词的长度，即为要替换字段的位置，通过赋值将新字段赋值给当前位置实现替换功能。

### 4.1.2 界面设计

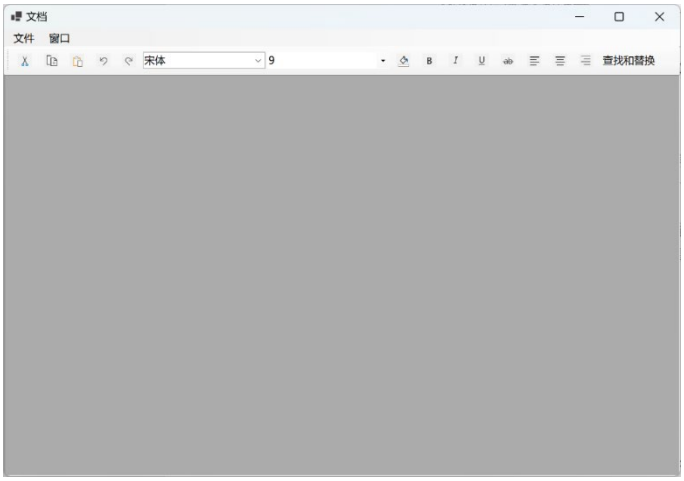


图 5 文本编辑器功能拓展主界面

### 4.1.3 关键功能代码

```
private void toolStripButton4_Click(object sender, EventArgs e)
 {
     ((Form2)this.ActiveMdiChild).rTBDoc.Undo();
 }
 private void toolStripButton5_Click(object sender, EventArgs e)
```

```
        {
            ((Form2)this.ActiveMdiChild).rTBDoc.Redo();
        }
        private void toolStripButton1_Click(object sender, EventArgs e)
        {
            Clipboard.SetData(DataFormats.Rtf,
((Form2)this.ActiveMdiChild).rTBDoc.SelectedRtf);//复制RTF数据到剪贴板
            ((Form2)this.ActiveMdiChild).rTBDoc.SelectedRtf = "";//再把当前
选取的RTF内容清除掉
        }
        private void toolStripButton2_Click(object sender, EventArgs e)
        {
            Clipboard.SetData(DataFormats.Rtf,
((Form2)this.ActiveMdiChild).rTBDoc.SelectedRtf);//复制RTF数据到剪贴板
        }
        private void toolStripButton3_Click(object sender, EventArgs e)
        {
            ((Form2)this.ActiveMdiChild).rTBDoc.Paste();//把剪贴板上的数据
粘贴到目标
        }
        private void toolStripButton10_Click(object sender, EventArgs e)
        {
            ((Form2)this.ActiveMdiChild).rTBDoc.SelectionAlignment =
HorizontalAlignment.Left;
        }
        private void toolStripButton13_Click(object sender, EventArgs e)
        {
            ((Form2)this.ActiveMdiChild).rTBDoc.SelectionAlignment =
HorizontalAlignment.Center;
        }

        private void toolStripButton14_Click(object sender, EventArgs e)
        {
            ((Form2)this.ActiveMdiChild).rTBDoc.SelectionAlignment =
HorizontalAlignment.Right;
            }
```

### 4.1.4 运行调试

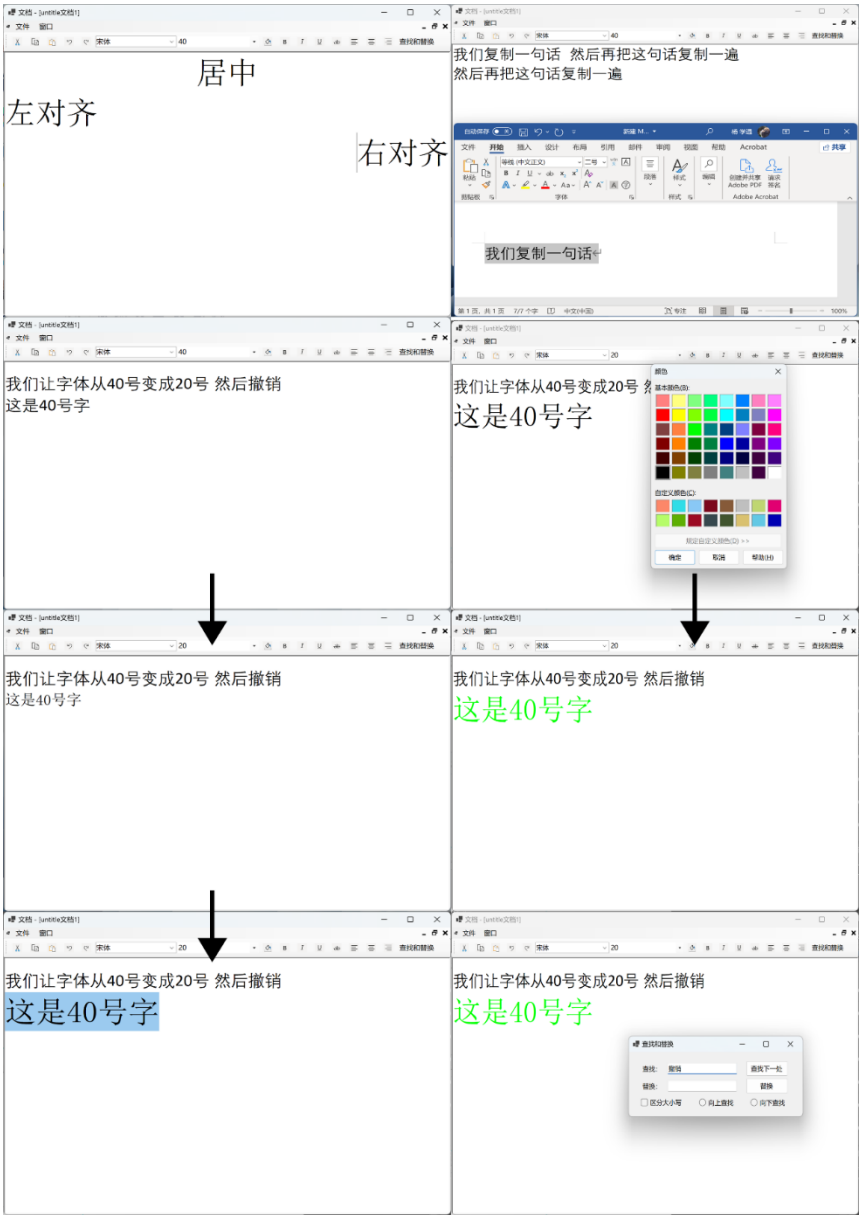


图 6 文档编辑器拓展功能调试

## 5 总结

好的程序不仅要有强大的功能，更要有美观、简洁、高效的 UI 设计。这让我理解了一个项目从开始到实现的基本过程，对以后程序开发有很大的帮助。本次多文本文档编辑器能得以成功实现，得益于 VS 提供的 system 中强大的函数功能，以及 VS 中各种控件支持的强大功能。这提醒我 C#中还有更多的功能等待我去探索，掌握。