

中国矿业大学计算机学院

2017 级本科生课程设计报告

课程名称：程序设计综合实践

报告时间：2018-10-30

学生姓名：李治远

学 号：07172757

专 业：计算机科学与技术

任课教师：王荣存

成绩考核

编号	课程教学目标	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。	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。	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。	50%	
总成绩			
指导教师		评阅日期	

目录

个人感悟	0
实验一 简单计算器	1
1. 计算器概述	1
2. 计算器设计	1
2.1 设计目标	1
1) 简单计算器的实现.....	1
2) 贷款计算器的实现.....	1
2.2 设计分析与算法流程	3
2.3 界面设计	4
2.4 关键类图	5
3. 计算器实现（运行调试）	5
4. 计算器扩展	6
5. 总结	7
1) Stack（栈）的用法总结.....	7
2) DialogResult 方法	7
3) 主要代码与分析	7
实验二 简单文本编辑器	15
1. 简单文本编辑器概述	15
2. 简单文本编辑器设计	15
2.1 设计目标	15
2.2 设计分析与算法流程	16
2.3 界面设计	16
2.4 关键类图	17
3. 简单文本编辑器实现（运行调试）	19
4. 简单文本编辑器扩展	19
5. 总结	20
1) 富文本框控件 RichTextBox 知识.....	20

2) C#中几种对话框	21
3) 主要代码与分析	25
实验三 学生通讯录	36
1. 学生通讯录概述	36
2. 学生通讯录设计	36
2.1 设计目标	36
2.2 设计分析与算法流程	38
2.3 界面设计	39
1) 主窗体设计	40
2) 添加窗体设计	40
3) 编辑窗体设计	41
4) 查找窗体设计	41
2.4 关键类图	42
3. 学生通讯录现（运行调试）	42
4. 学生通讯录扩展	43
5. 总结	43
1) xml 基础	43
2) xml 文件操作方法	44
3) Combox 控件	45
4) 主要代码与分析	47
实验四 拼图游戏	59
1. 拼图游戏概述	59
2. 拼图游戏设计	59
2.1 设计目标	59
2.2 设计分析与算法流程	60
2.3 界面设计	61
2.4 关键类图	61
3. 拼图游戏实现（运行调试）	62

4. 拼图游戏扩展	62
1) 切割块数	62
2) 计时、计步	62
5. 总结	63
1) Timer 控件总结	63
2) PictureBox 控件	63
3) 主要代码与分析	64

个人感悟

在这几周的 C# 课程中，我学到很多东西，也掌握了 C# 可视化设计的方法，它有助于开发者更好的设计 UI，使应用软件更加人性化，但是这仅仅是 UI 视觉上的设计，并不能说自己会使用 C# 开发程序。

所以，除了视觉方面，我还在代码编写的过程中花费更多的时间，这不仅锻炼了我的思维，我的逻辑设计以及 BUG 查找与修复，还加深了我对一些控件的理解等等很多方面。

比如在计算器设计中，我可以了解并掌握 Stack 栈的使用方法；在学生通讯录中，我可以学到如何设计文件路径，以及如何检索 xml 文件中的属性与内容。虽然，我以后使用 C# 的时候并不多，但是通过这四个程序的开发与设计，还是大大的拓宽了我的知识面，使我的编程开发能力有所提高。

在这个学习的过程中，我也了解到一些学习编程的方法，比如，学习数据结构会使我的编程思路更加清晰，改正我自己那臃肿繁琐的算法，使用简单高效的算法；经常逛编程论坛，遇到问题自己可以在上面进行搜索或者提问，慢慢的自己的水平也会提高。。。

虽然结束了这门课程，但是我的学习之路并没有停止，我会吸取这次学习的经验为我今后的发展做铺垫。

实验一 简单计算器

基本功能：四则运算、开方、平方等

特色功能：贷款计算器、连续计算等

1. 计算器概述

主要内容是设计开发一个支持连续计算的简单计算器，其过程包括设计目标、设计分析与算法流程、界面设计、关键类图、运行调试与项目扩展。通过该项目的开发，使自己进一步了解基于图形用户界面的 Windows 应用程序的开发过程。

2. 计算器设计

2.1 设计目标

1) 简单计算器的实现

该项目是一个支持连续计算的四则运算计算器，通过点击相应的数字按钮和运算按钮，输入并完成如 $4*6+3$ 、 $\cos(x)$ 或者取倒数等等类似的计算，并将运算结果显示输出在文本框中，同时此计算器也具备清空、退格等其他功能，其运行界面如图 1-1 所示。

计算器是一个简单的单窗口桌面程序，在主窗体上放置了按钮、文本框等控件，通过改变窗体与控件的属性，使界面更加美观与人性化，并为不同的控件添加了相应的事件和可执行的代码，能够完成简单的加减乘除等操作。

2) 贷款计算器的实现

首先为计算器增加开平方根、平方、求倒数等功能，使其更加接近科学计算器。

然后在主窗体上增加一个按钮控件，通过点击可以打开贷款计算器的窗口。



图 1-1

Mortgage

还款方式: ☒ 等额本息 ☐ 等额本金

贷款年限(年):

贷款金额(万元):

贷款利率(年/%) :

月均还款(元):

利息总额(元):

还款总额(元):

图 1-2 贷款计算器

2.2 设计分析与算法流程

计算器的主要功能是完成加减乘除四则运算，而且支持连续运算，如 $4+4-7$ 或 $4\times 7-6$ 等运算。连续运算的实现借助于数据结构中的 Stack（栈），如图 1-3 所示：首先是输入数字 A 入栈，然后输入运算符，如果是开方、平方运算则直接输出结果替换数字 A，否则将运算符入栈，接着输入数字 B 入栈；然后，如果输入“=”，则通过出栈计算 $A \text{ OP } B$ ，得到计算结果，否则，继续输入其他运算符，重复上述的过程，来实现连续运算。

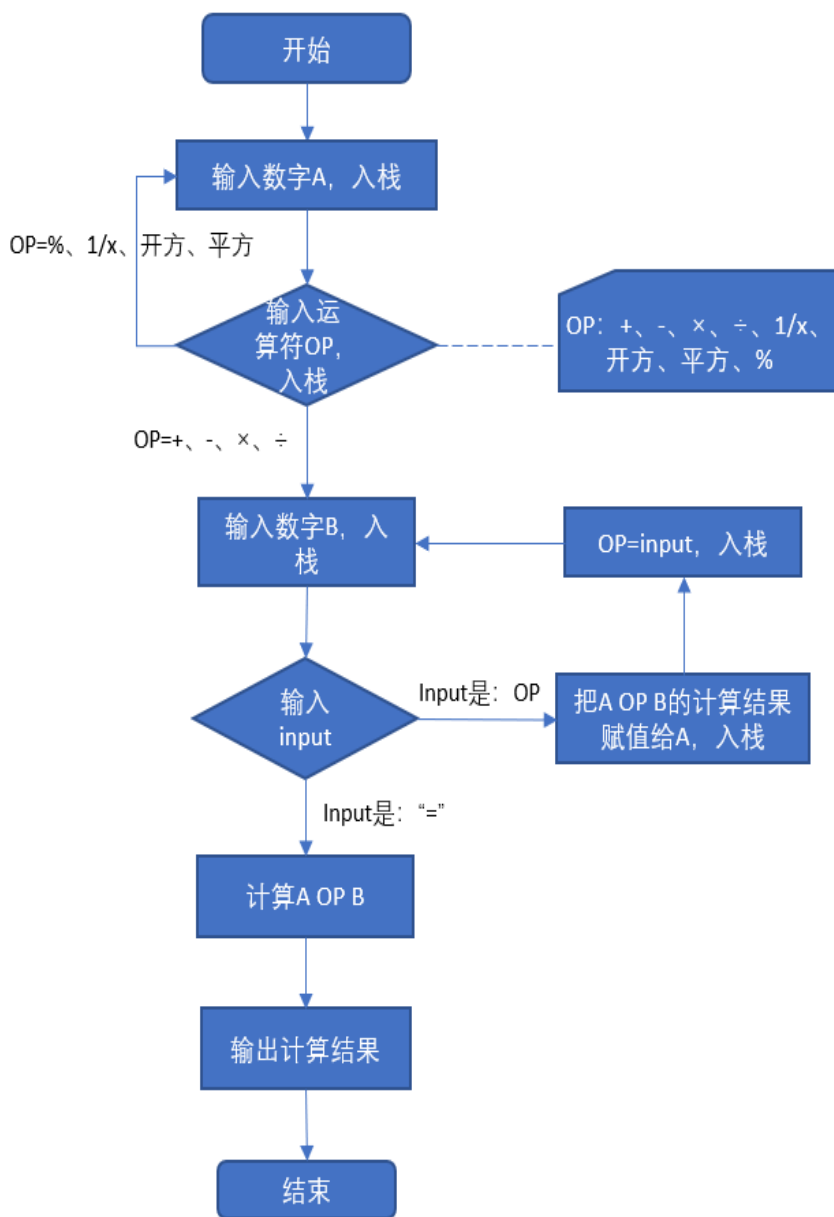


图 1-3 简单计算器的算法路程图

2.3 界面设计

进入“工具箱”中，展开“公共控件”，把计算器所需要的控件拖到 Form_Main 窗体上，并设置控件的属性。

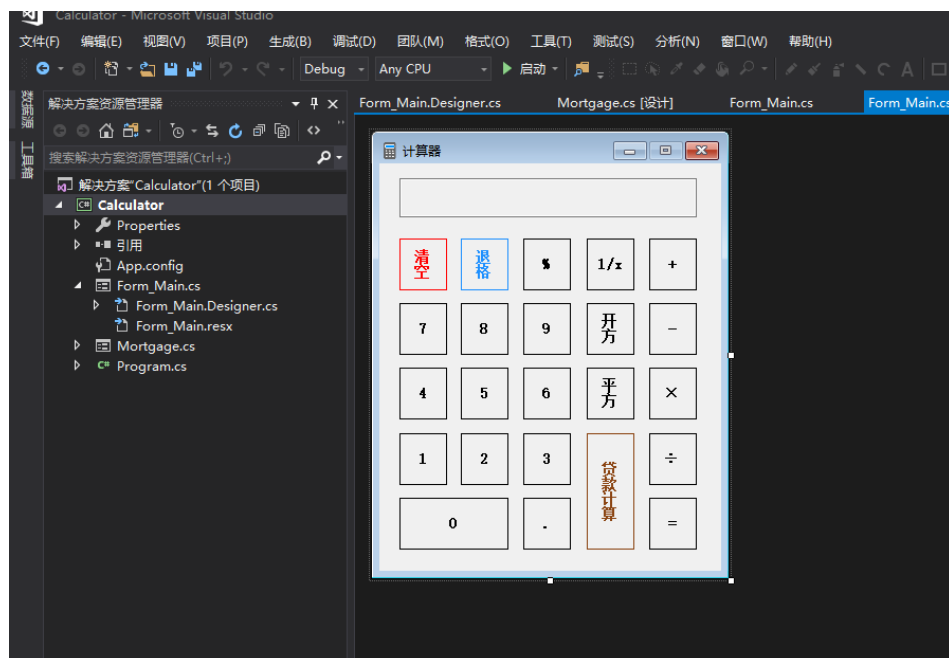


图 1-5

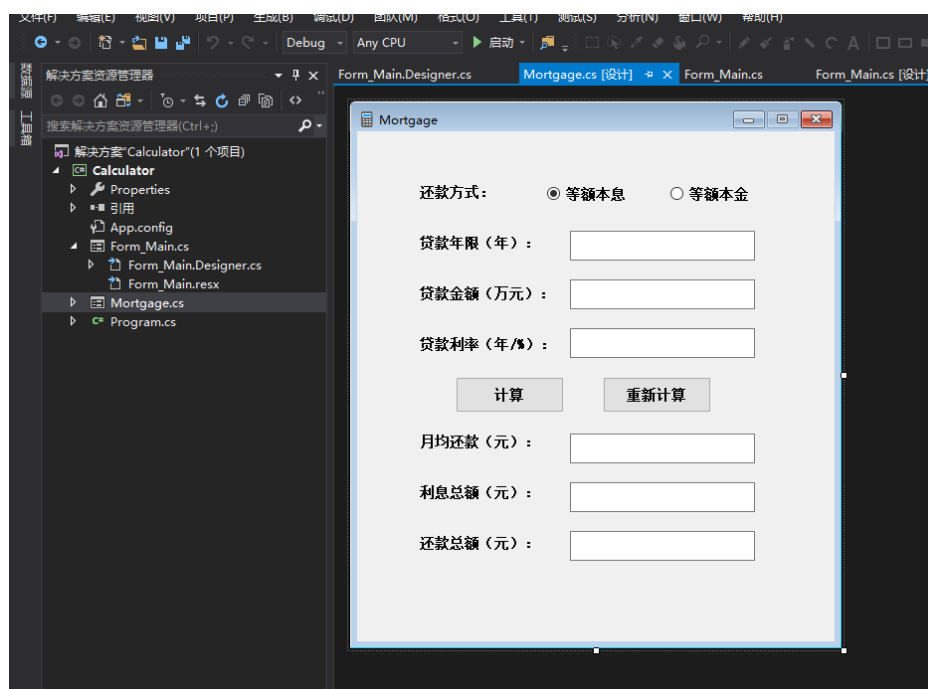


图 1-6

2.4 关键类图

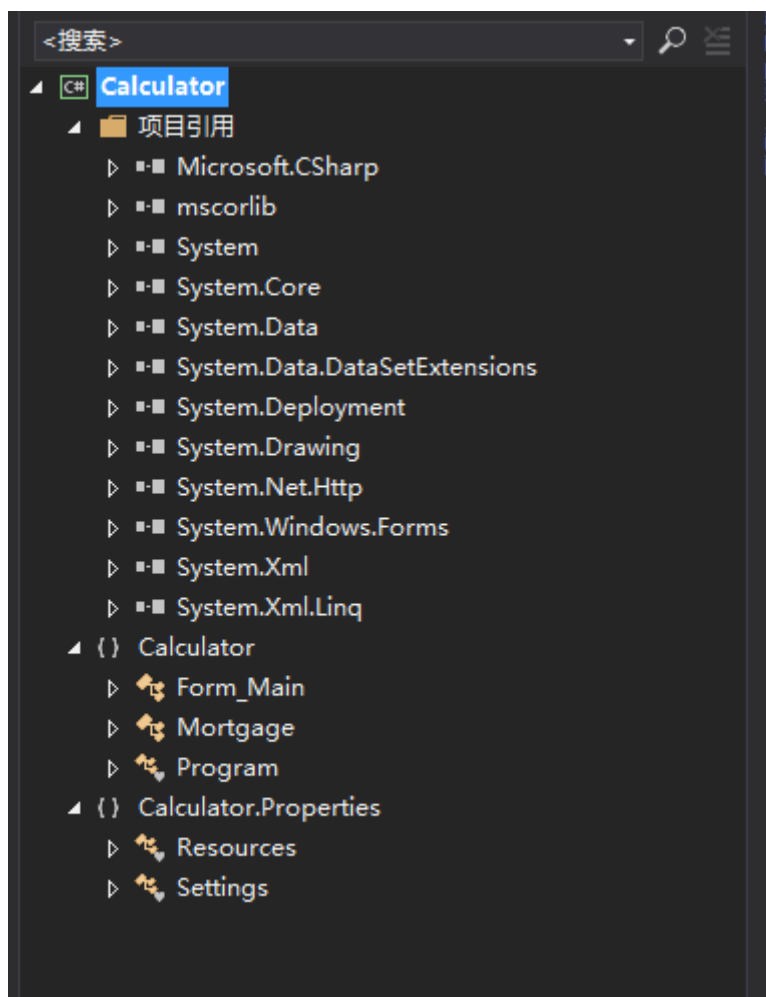


图 1-7

btn_A_Click()	按下数字按钮将值显示在文本框上
btn_OP_Click()	按下操作符将文本框内容与操作入栈
btn_backspace_Click()	将文本框中的内容从右消除一个
btn_equal_Click	计算器的算式计算与结果输出

Tips:

A 为 0-9 十个数字；

OP 为 + - × ÷ 等。

3 . 计算器实现（运行调试）

按下数字 6，再按下开方结果如下图所示：



图 1-8

4. 计算器扩展

计算_Click()	贷款计算输出

图 1-9

根据等额本息与等额本金的还款公式，编写一个贷款计算器。

5 . 总结

1) Stack（栈）的用法总结

为了使计算器支持连续计算，在项目中使用了 Stack 类，Stack 类表示对象的简单后进先出非泛型集合，它位于 System.Collections 命名空间中，它的特点是：后进先出的集合，Stack 能接受空引用作为有效值并且允许重复的元素。

Stack 常用的方法及属性如下：

方法名称	说明
public virtual void Clear();	从 Stack 中移除所有的元素
public virtual bool Contains(object obj);	判断某个元素是否在 Stack 中
public virtual object Peek();	返回在 Stack 的顶部的对象，但不移除它
public virtual object Pop();	移除并返回在 Stack 的顶部的对象
public virtual void Push(object obj);	向 Stack 的顶部添加一个对象

2) DialogResult 方法

方法名称	说明
AbortRetryIgnore	提供“中止”、“重试”和“忽略”三个按钮
OK	提供“确定按钮”
OKCancel	提供“确定”和“取消”两个按钮
RetryCancel	提供“重试”和“取消”两个按钮
YesNo	提供“是”和“否”两个按钮
YesNoCancel	提供“是”、“否”和“取消”三个按钮

3) 主要代码与分析

①计算器

```
using System;
using System.Collections;
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;
```

变量

```
Stack st = new Stack();
double num1, num2, tempresult;
string Operator;
bool flag = true, flag2 = false;
```

方法

// 按下等于, 求结果

```
private void btn_equal_Click(object sender, EventArgs e)
{
    try
    {
        st.Push(txtDisplay.Text);
        if (st.Count >= 3)
        {
            num1 = Convert.ToDouble(st.Pop());
            Operator = Convert.ToString(st.Pop());
            num2 = Convert.ToDouble(st.Pop());
            switch (Operator)
            {
                case "+":
                    tempresult = num2 + num1;
                    st.Push(tempresult);
                    break;
                case "-":
                    tempresult = num2 - num1;
                    st.Push(tempresult);
                    break;
                case "x":
                    tempresult = num2 * num1;
                    st.Push(tempresult);
                    break;
                case "÷":
```

```

        tempresult = num2 / num1;
        st.Push(tempresult);
        break;
    }
    txtDisplay.Text = tempresult.ToString();
}
st.Clear();
flag2 = true;
}
catch (Exception)
{
    MessageBox.Show("先输入数字才能进行此运算!", "Error", MessageBoxButtons.OK);
}
}

```

上面的代码是按下等于后的计算，首先将 txtDisplay 中显示的数字字符串 Push 进栈中，如果栈中的项大于或等于三个就可以进行运算。之后再用 switch 语句来判断操作符 OP 是+、-、×、÷中的哪个，然后在进行运算。

注意：num1 是后进入的数字，num2 是先进入的数字。如，5-6，num1 即为 6，num2 即为 5。

// 按下数字 7 按钮

```

private void btn_7_Click(object sender, EventArgs e)
{
    if (flag2)
    {
        flag2 = false;
        txtDisplay.Text = "7";
    }
    else if (flag)
    {
        txtDisplay.Text += "7";
    }
    else
    {
        flag = true;
        txtDisplay.Text = "7";
    }
}

// 求百分值
private void btn_per_Click(object sender, EventArgs e)
{
    double num;
    try

```

```
{
    st.Push(txtDisplay.Text);

    num = Convert.ToDouble(st.Pop());
    st.Push(num * 0.01);
    txtDisplay.Text = (num * 0.01).ToString();
    flag2 = true;
}
catch (Exception)
{
    MessageBox.Show("请先输入一个数字! ", "Error", MessageBoxButtons.OK);
}
}
// 求开方值
private void btn_sqrt_Click(object sender, EventArgs e)
{
    double num;
    try
    {
        st.Push(txtDisplay.Text);

        num = Convert.ToDouble(st.Pop());
        st.Push(Math.Sqrt(num));
        txtDisplay.Text = Math.Sqrt(num).ToString();
        flag2 = true;
    }
    catch (Exception)
    {
        MessageBox.Show("请先输入一个数字! ", "Error", MessageBoxButtons.OK);
    }
}
// 平方
private void btn_ping_Click(object sender, EventArgs e)
{
    double num;
    try
    {
        st.Push(txtDisplay.Text);

        num = Convert.ToDouble(st.Pop());
        st.Push(Math.Pow(num, 2));
        txtDisplay.Text = Math.Pow(num, 2).ToString();
        flag2 = true;
    }
}
```



```
        catch (Exception)
        {
            MessageBox.Show("请先输入一个数字! ", "Error", MessageBoxButtons.OK);
        }
    }

    // 打开贷款计算器
    private void btn_dai_Click(object sender, EventArgs e)
    {
        var fm = new Mortgage();
        fm.ShowDialog();
    }

    // 求倒数
    private void btn_re_Click(object sender, EventArgs e)
    {
        double num;
        try
        {
            st.Push(txtDisplay.Text);

            num = Convert.ToDouble(st.Pop());
            st.Push(1 / num);
            txtDisplay.Text = (1 / num).ToString();
            flag2 = true;
        }
        catch (Exception)
        {
            MessageBox.Show("请先输入一个数字! ", "Error", MessageBoxButtons.OK);
        }
    }

    // 退格
    private void btn_backspace_Click(object sender, EventArgs e)
    {
        try
        {
            txtDisplay.Text = txtDisplay.Text.Substring(0, txtDisplay.Text.Length - 1);
        }
        catch (Exception)
        {
        }
    }

    // 清空编辑框
```

```
private void btn_clear_Click(object sender, EventArgs e)
{
    txtDisplay.Text = "";
    st.Clear();
    flag = true; flag2 = false;
}
```

tip: 上面的 flag2 是用来判断是否计算出结果, 如果得出结果, 那么可以利用这个 flag2 来判断下次按下数字按键的时候是否会清空 txtDisplay 内容, 并加入按下的数字。

②贷款计算器

```
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 Calculator
{
    public partial class Mortgage : Form
    {
        public Mortgage()
        {
            InitializeComponent();
        }

        private void Mortgage_Load(object sender, EventArgs e)
        {
        }

        private void Mortgage_FormClosed(object sender, FormClosedEventArgs e)
        {
        }

        private void 计算_Click(object sender, EventArgs e)
        {
            if(string.IsNullOrEmpty(贷款年限.Text)||string.IsNullOrEmpty(贷款金额.Text)||
string.IsNullOrEmpty(贷款利率.Text))
            {
                MessageBox.Show("请填写完整数据! ", "Error", MessageBoxButtons.OK);
            }
        }
    }
}
```

```

    }
    else
    {
        double yearM, money, rate;
        double moHuan, moLi, moZli;
        yearM = Convert.ToDouble(贷款年限.Text) * 12;
        money = Convert.ToDouble(贷款金额.Text) * 10000;
        rate = Convert.ToDouble(贷款利率.Text) / 1200;
        if (等额本息.Checked)
        {
            moHuan = money * rate * Math.Pow(1 + rate, yearM) / (Math.Pow(1 +
rate, yearM) - 1);
            moLi = moHuan * yearM - money;
            moZli = moHuan * yearM;
            月均还款.Text = moHuan.ToString("0.00");
            利息总额.Text = moLi.ToString("0.00");
            还款总额.Text = moZli.ToString("0.00");
        }
        else
        {
            moHuan = (money / yearM) + (money - money) * rate;
            moLi = (money / yearM + money * rate + money / yearM * (1 + rate)) / 2
* yearM - money;
            moZli = (money / yearM + money * rate + money / yearM * (1 + rate)) / 2
* yearM;
            月均还款.Text = moHuan.ToString("0.00");
            利息总额.Text = moLi.ToString("0.00");
            还款总额.Text = moZli.ToString("0.00");
        }
    }
}

private void 重新计算_Click(object sender, EventArgs e)
{
    贷款年限.Text = "";
    贷款金额.Text = "";
    贷款利率.Text = "";
    月均还款.Text = "";
    利息总额.Text = "";
    还款总额.Text = "";
}
}
}

```

贷款计算器主要是公式的运用，比如计算结果是 double 型，最后要将计算

结果显示在文本框中。

```
if (等额本息.Checked)
{
    moHuan = money * rate * Math.Pow(1 + rate, yearM) / (Math.Pow(1 + rate, yearM) - 1);
    moLi = moHuan * yearM - money;
    moZli = moHuan * yearM;
    月均还款.Text = moHuan.ToString("0.00");
    利息总额.Text = moLi.ToString("0.00");
    还款总额.Text = moZli.ToString("0.00");
}
```

图 1-8

首先判断等额本息被选中，然后调用 Math 中的方法加以简化算式，使得代码更加简洁。

实验二 简单文本编辑器

基本功能：文本的编辑、保存、打开、改变字形、对齐方式等

特色功能：插入图片、改变字体颜色、判断文件是否已被打开与是否保存

1. 简单文本编辑器概述

主要内容是设计开发一个多文档文本编辑器。在文本编辑器、图像处理器这样的应用软件中，通常需要同时处理一个或着多个文档，每个文档独立地执行软件所需的功能。这种需要在一个窗体中同时包含多个子窗体的应用程序通常称为多文档应用程序，子窗体之间可以进行数据交互，也可以互不相干。

2. 简单文本编辑器设计

2.1 设计目标

设计开发一个简单的多文档文本编辑器，具有新建、打开、保存一个文本文件，设置字体、颜色、下划线等功能，也具有复制、粘贴、插入图片等扩展功能，并且可以判断文件是否被重复打开。其主界面如下图所示：

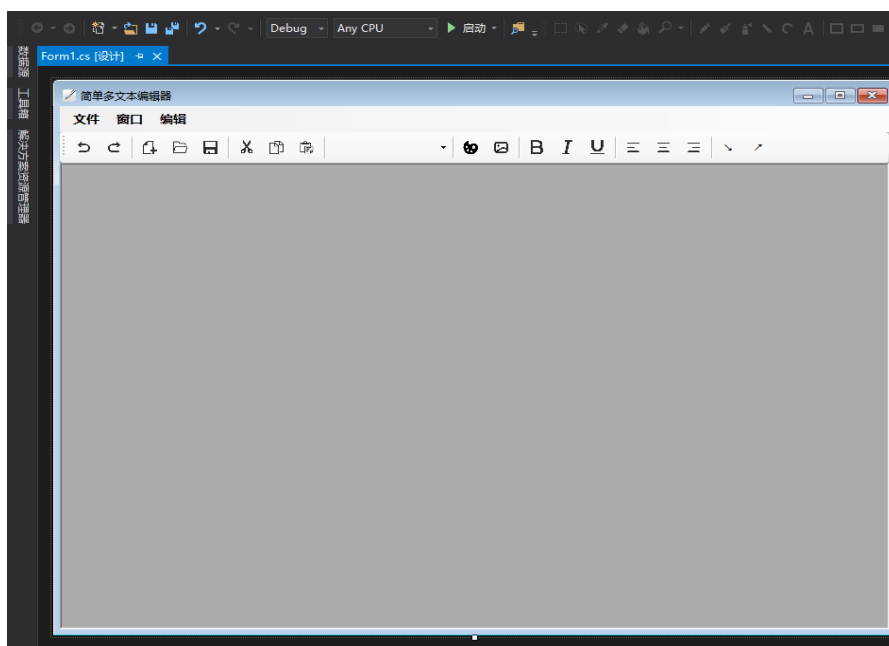


图 2-1

2.2 设计与算法流程

简单的文本编辑器主要实现多文档功能、简单的文本编辑功能，如打开、保存、字体等设置功能。流程图如下：



图 2-2

各功能模块具体功能描述如下：

打开已存在文档：可以打开以 .txt 结尾的文本文件，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）。

2.3 界面设计

在本程序中有两个窗体，一个是作为多文档容器的主窗体，另一个是子窗体。

主窗体：

- (1) 进入“工具箱”中，展开“菜单和工具栏”，把控件 MenuDtrip、ToolDtrip 和

ToolTip 拖到 Form1 窗体上。

- (2) 编辑快捷工具栏: 选中 toolScript1, 选择最右端的小三角形, 打开“ToolScript 任务”, 选择“插入标准项”。将不需要的快捷按钮选中后删除, 然后点击工具栏上的下拉按钮, 新建一个 DropDownButton 和三个 Button, 分别是字体下拉框, 粗体按钮, 斜体按钮和下划线按钮, Name 设置为 tSCbBoxFontName、粗体 ToolScriptButton、斜体 ToolScriptButton、下划线 ToolScriptButton。
- (3) 编辑菜单栏: 直接在窗体菜单区编辑菜单内容。其中属性“Image”的设置可以打开“Form1.Designer.cs”修改代码, 在 InitializeComponent() 方法中添加。

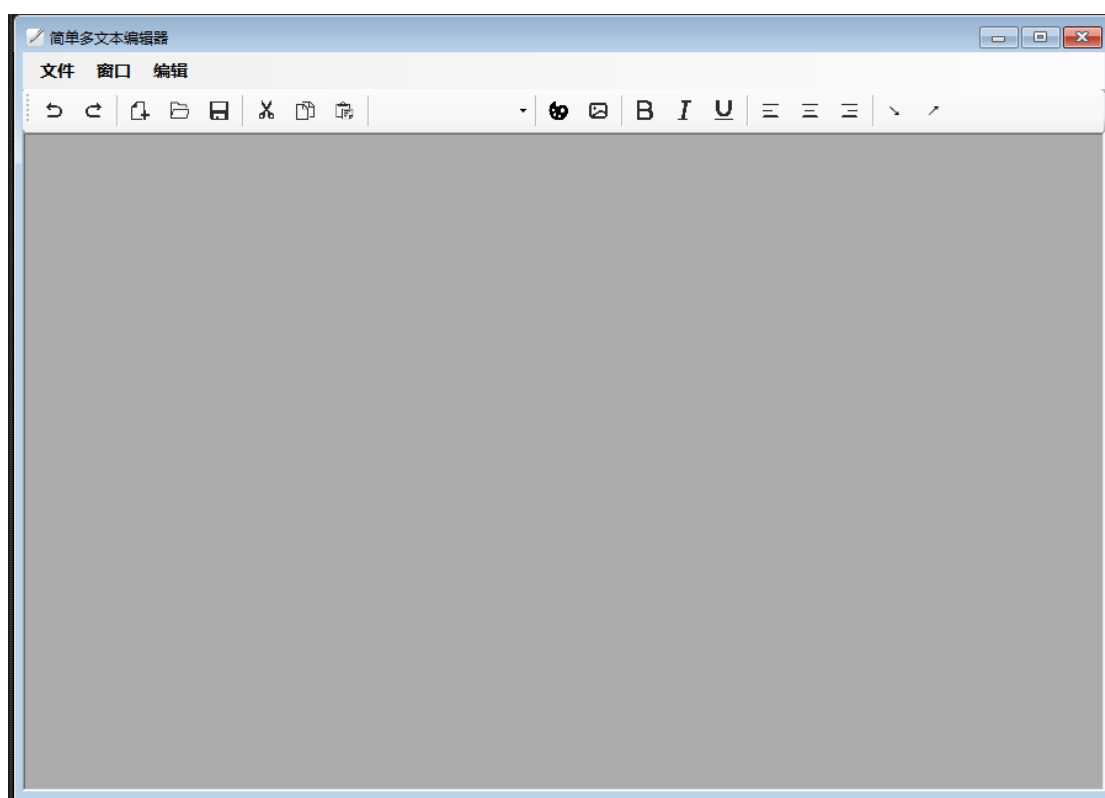


图 2-3

2.4 关键类图

checkFile()	检查文件是否重复打开
NewDoc()	新建一个文件
打开 OToolStripMenuItem_Click()	打开文件
保存 SToolStripMenuItem_Click()	保存文件

窗口层叠 ToolStripMenuItem_Click()	窗口层叠样式
closeFile()	关闭文件并检查是否保存
tSCbBoxFontName_TextChanged()	改变字体
ChangeRTBFontStyle()	改变字体样式
img_Click()	插入图片

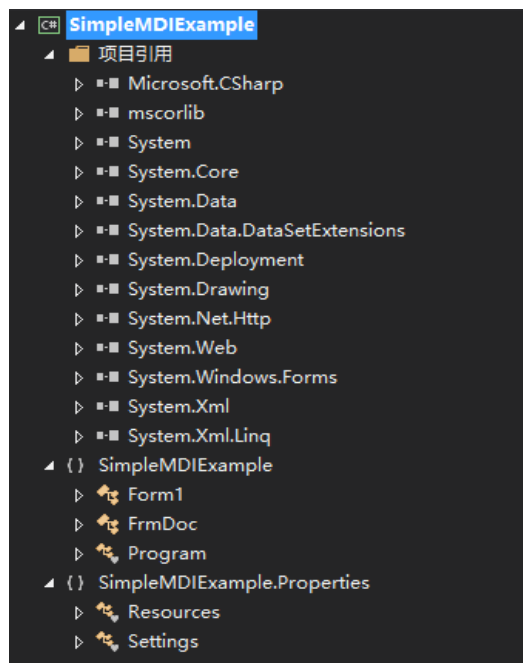


图 2-4

3. 简单文本编辑器实现（运行调试）

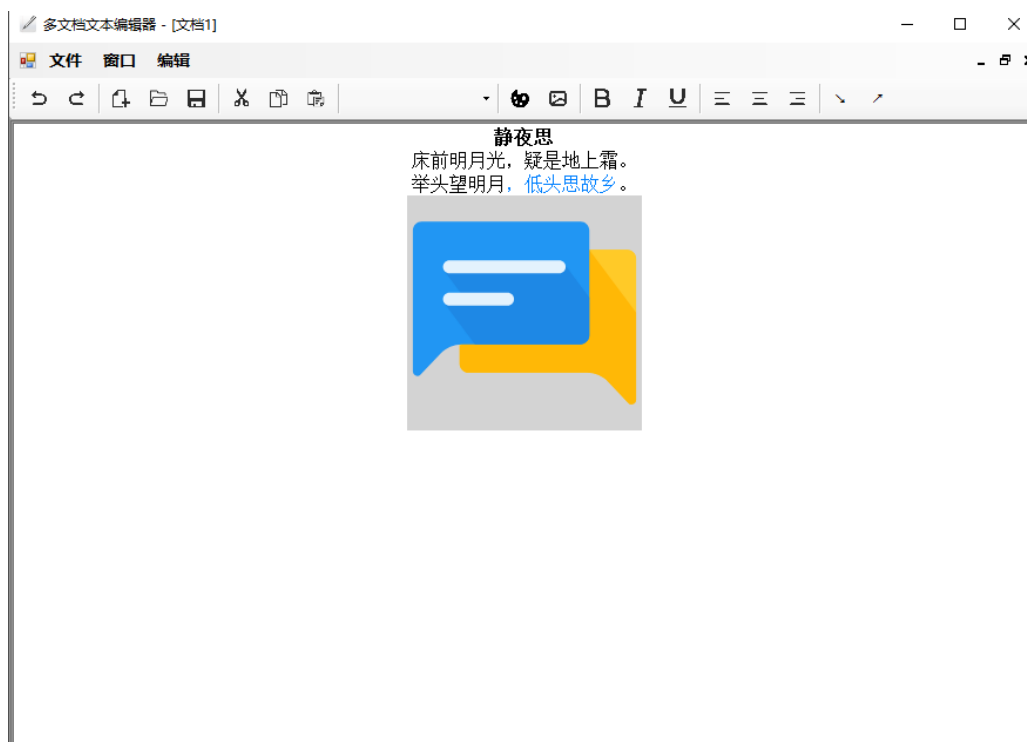


图 2-5

4. 简单文本编辑器扩展

多文本编辑器一般还具有撤销、重做、设置文字对齐方式、复制、粘贴、剪切、查找、替换等功能。实现这些扩展功能时，可以利用 RichTextBox 的属性和方法来实现。

对齐方式使用 HorizontalAlignment 枚举：指定控件中的对象或文本如何相对于控件元素水平对齐。HorizontalAlignment 枚举有三个成员，见下表：

属性	说明
Left	对象或文本与控件元素的左侧对齐
Right	对象或文本与控件元素的右侧对齐
Center	对象或文本与控件元素的中心对齐

对于每一个文档设置一个标识是新建文件还是打开的文件，保存时有区别，并进行提示。

5 . 总结

1) 富文本框控件 RichTextBox 知识

RichTextBox 是一种既可以输入，又可以编辑文本的文字处理控件，与 TextBox 控件相比，RichTextBox 控制的文字处理功能更加丰富，不仅可以设定文字的颜色、字体，还有字符串检索功能。还可以打开、编辑和储存 .rtf 格式文件、ASCII 文本格式文件及 Unicode 编码格式的文件。

名 称	描 述
CanRedo	如果上一个被撤销的操作可以使用 Redo 重复，这个属性就是 true
CanUndo	如果可以在 RichTextBox 上撤销上一个操作，这个属性就是 true，注意，CanUndo 在 TextBoxBase 中定义，所以也可以用于 TextBox 控件
RedoActionName	这个属性包含通过 Redo 方法执行的操作名称
DetectUrls	把这个属性设置为 true，可以使控件检测 URL，并格式化它们
Rtf	它对应于 Text 属性，但包含 RTF 格式的文本
SelectedRtf	使用这个属性可以获取或设置控件中被选中的 RTF 格式文本。如果把这些文本复制到另一个应用程序中，例如 Word，该文本会保留所有的格式化信息
SelectedText	与 SelectedRtf 一样，可以使用这个属性获取或设置被选中的文本。但与该属性的 RTF 版本不同，所有的格式化信息都会丢失
SelectionAlignment	它表示选中文本的对齐方式，可以是 Center, Left 或 Right
SelectionBullet	使用这个属性可以确定选中的文本是否格式化为项目符号的格式，或使用

	它插入或删除项目符号
BulletIndent	使用这个属性可以指定项目符号的缩进像素值
SelectionColor	这个属性可以修改选中文本的颜色
SelectionFont	这个属性可以修改选中文本的字体
SelectionLength	使用这个属性可以设置或获取选中文本的长度
SelectionType	这个属性包含了选中文本的信息。 它可以确定是选择了一个或多个 OLE 对象，还是仅选择了文本
ShowSelectionMargin	如果把这个属性设置为 true，在 RichTextBox 的左边就会出现一个 页边距，这将使用户更易于选择文本
UndoActionName	如果用户选择撤销某个动作，该属 性将获取该动作的名称
SelectionProtected	把这个属性设置为 true，可以指定不 修改文本的某些部分

RichTextBox 常用事件为 TextChanged，在 Text 属性值改变时触发。

2) C#中几种对话框

① OpenFileDialog 控件

属性	说明
Title	用来获取或设置对话框标题，默认值为空字符串（""）。如果标题为空字符串，则系统将使用默认标题：“打开”
Filter	用来获取或设置当前文件名筛选器字符串，该字符串决定对话框的【另存为文件类型】或【文件类型】框中出现的选项内容。对于每个筛选选项，筛选器字符串都包含筛选器说明、垂直线条（ ）和筛选器模式。不同筛选选项的字符串由垂直线条隔开，例

	如：“文本文件 (*.txt) *.txt 所有文件 (*.*) *.*” 。还可以通过用分号来分隔各种文件类型，可以将多个筛选器模式添加到筛选器中，例如“图像文件 (*.BMP;*.JPG;*.GIF) *.BMP;*.JPG;*.GIF 所有文件 (*.*) *.*”
FilterIndex	用来获取或设置文件对话框中当前选定筛选器的索引。第一个筛选器的索引为 1，默认值为 1
FileName	用来获取在打开文件对话框中选定的文件名的字符串。文件名既包含文件路径也包含扩展名。如果未选定文件，该属性将返回空字符串 (“”)
InitialDirectory	用来获取或设置文件对话框显示的初始目录，默认值为空字符串 (“”)
ShowReadOnly	用来获取或设置一个值，该值指示对话框是否包含只读复选框。如果对话框包含只读复选框，则属性值为 true，否则属性值为 false。默认值为 false
ReadOnlyChecked	用来获取或设置一个值，该值指示是否选定只读复选框。如果选中了只读复选框，则属性值为 true，反之，属性值为 false。默认值为 false
Multiselect	用来获取或设置一个值，该值指示对话框是否允许选择多个文件。如果对话框允许同时选定多个文件，则该属性值为 true，反之，属性值为 false。默认值为 false
FileNames	用来获取对话框中所有选定文件的文件名。每个文件名都既包含文件路径又包含文件扩展名。如果未选定文件，该方法将返回空数组
RestoreDirectory	用来获取或设置一个值，该值指示对话框在关闭前是否还原当前目录。假设用户在搜索文件的过程中更改了目录，且该属性值为 true，那么，对话框会

	<p>将当前目录还原为初始值，若该属性值为 false，则不还原成初始值。默认值为 false。</p> <p>OpenFileDialog 控件的常用方法有两个：OpenFile 和 ShowDialog 方法，本节只介绍 ShowDialog 方法，该方法的作用是显示通用对话框，其一般调用形式如下：通用对话框对象名.ShowDialog();通用对话框运行时，如果单击对话框中的【确定】按钮，则返回值为 DialogResult.OK；否则返回值为 DialogResult.Cancel</p>
--	---

OpenFileDialog 控件最主要的常用方法是 ShowDialog 方法，该方法的作用是显示通用对话框，其一般调用形式是：

对象名.ShowDialog();

通用对话框运行时，如果单击对话框中的“确定”按钮，则返回值为 DialogResult.OK；否则返回值为 DialogResult.Cancel。

② SaveFileDialog 控件

SaveFileDialog 又称保存文件对话框，主要用来弹出 Windows 中标准的“保存文件”对话框。

SaveFileDialog 控件也具有 FileName、Filter、FilterIndex、InitialDirectory、Title 等属性，这些属性的作用与 OpenFileDialog 对话框的一样，但要实现打开或保存程序，需要自己设计。

③ FontDialog 控件

FontDialog 又称字体对话框，主要用来弹出 Windows 中标准的“字体”对话框。字体对话框的作用是显示当前安装在系统中的字体列表，供用户进行选择。

FontDialog 主要属性如下表所示：

AllowVectorFonts	指示对话框是否允许选择矢量字体
AllowScriptChange	指示用户能否更改“脚本”组合框中指定的字符集，以显示除了当前所显示字符集以外的字符集，如果用户能更改“脚本”组合框中指定的字符集，则为 true

FixedPitchOnly	表示指定对话框是否只允许选择固定间距的字体
Color	获取或设置指定字体的颜色
Font	获取或设置选定的字体
FontMustExist	指示当前用户选择不存在的字体或样式时，对话框是否报告错误信息
MaxSize	指示可选择字体的最大磅值
MinSize	指示可选择字体的最小磅值
ShowApply	指示对话框是否包含“应用”按钮
ShowColor	指示对话框是否显示颜色选择
ShowEffects	指示对话框是否包含允许用户指定删除线、下划线和文本颜色选项的控件

FontDialog 的事件 Apply 当用户单击字体对话框中的“应用”按钮时发生。

④ ColorDialog 控件

ColorDialog 又称颜色对话框，主要用来弹出 Windows 中标准的“颜色”对话框。颜色对话框的作用是供用户选择一种颜色，并用 Color 属性记录用户选择的颜色值。属性表如下：

属性值	说明
AllowFullOpen	指示用户是否可以使用该对话框定义自定义颜色
AnyColor	指示对话框是否显示基本颜色集中可用的所有颜色
Color	获取或设置用户选定的颜色
SolidColorOnly	获取或设置用户选定的颜色
FullOpen	获取或设置用户选定的颜色
方法名	说明

ShowDialog	用默认的所有者运行通用对话框。该方法的返回值类型为 DialogResult，如果用户在对话框中单击“确定”按钮，则为 DialogResult.OK，否则为 DialogResult.Cancel
方法名	说明
ShowDialog	用默认的所有者运行通用对话框。该方法的返回值类型为 DialogResult，如果用户在对话框中单击“确定”按钮，则为 DialogResult.OK，否则为 DialogResult.Cancel
ShowDialog	用默认的所有者运行通用对话框。该方法的返回值类型为 DialogResult，如果用户在对话框中单击“确定”按钮，则为 DialogResult.OK，否则为 DialogResult.Cancel

3) 主要代码与分析

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Text;
using System.Web;
using System.Web.UI; //项目 添加引用
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Collections;

namespace SimpleMDIExample
{
    public partial class Form1 : Form
    {
        private int _Num = 1;
        //Stack<string> list = new Stack<string>();
        List<string> dinosaurs = new List<string>();
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)

```

```

{
    tSCbBoxFontName.DropDownItems.Clear();
    InstalledFontCollection ifc = new InstalledFontCollection();
    FontFamily[] ffs = ifc.Families;
    foreach (FontFamily ff in ffs)
    {
        tSCbBoxFontName.DropDownItems.Add(ff.GetName(1));
    }
    LayoutMdi(MdiLayout.Cascade);
    Text = "多文档文本编辑器";
    WindowState = FormWindowState.Maximized;
}
// 判断文件是否已经被打开
private bool checkFile(string fromText)
{
    if (dinosaurs.Exists(o => o == fromText))
    {
        return true;
    }
    else
    {
        return false;
    }
}
// 新建一个文件
private void NewDoc()
{
    FrmDoc fd = new FrmDoc();
    fd.MdiParent = this;
    fd.Text = "文档" + _Num;
    fd.WindowState = FormWindowState.Maximized;
    fd.Show();
    fd.Activate();
    _Num++;
}

private void 新建 NToolStripMenuItem_Click(object sender, EventArgs e)
{
    NewDoc();
}
// 如果文件已经被打开，警告
private void 打开 OToolStripMenuItem_Click(object sender, EventArgs e)
{

```



```

        OpenFileDialog openFileDialog1 = new OpenFileDialog();
        openFileDialog1.Filter = "RTF 格式|*.rtf|文本文件|*.txt|所有文件|*.*";
        openFileDialog1.Multiselect = false;
        if (openFileDialog1.ShowDialog() == DialogResult.OK)
        {
            try
            {
                NewDoc();

                ((FrmDoc)this.ActiveMdiChild).rTBDoc.LoadFile(openFileDialog1.FileName,
                    RichTextBoxStreamType.RichText);

                //((FrmDoc)this.ActivateMdiChild).rTBDoc.LoadFile(openFileDialog1.FileName,
                RichTextBoxStreamType.RichText);

                ((FrmDoc)this.ActiveMdiChild).Text = openFileDialog1.FileName;
                if (checkFile(openFileDialog1.FileName))
                {
                    Console.WriteLine("111");
                    ((FrmDoc)this.ActiveMdiChild).Close();
                    _Num--;
                    MessageBox.Show("此文件已经打开！ ", "Error",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                else
                {
                    dinosaurs.Add(openFileDialog1.FileName);
                }
            }
            catch
            {
                MessageBox.Show("打开失败！ ", "Error", MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
            }
        }
        openFileDialog1.Dispose();
    }

    private void 保存 SToolStripMenuItem_Click(object sender, EventArgs e)
    {
        if (this.MdiChildren.Count() > 0)
        {
            SaveFileDialog saveFileDialog1 = new SaveFileDialog();

```

```

        savefiledialog1.Filter = "RTF 格式|*.rtf|文本文件|*.txt";
        if (savefiledialog1.ShowDialog() == DialogResult.OK)
        {
            try
            {
                if (savefiledialog1.FilterIndex == 1)
                {
                    ((FrmDoc)this.ActiveMdiChild).rTBDoc.SaveFile(savefiledialog1.FileName,
                        RichTextBoxStreamType.RichText);
                }
                else
                {
                    ((FrmDoc)this.ActiveMdiChild).rTBDoc.SaveFile(savefiledialog1.FileName,
                        RichTextBoxStreamType.PlainText);
                }
                MessageBox.Show("保存成功! ", "", MessageBoxButtons.OK,
                    MessageBoxIcon.None);
            }
            catch
            {
                MessageBox.Show("保存失败! ", "Error", MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
            }
        }
        savefiledialog1.Dispose();
    }
}

private void 窗口层叠 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    LayoutMdi(MdiLayout.Cascade);
    this.窗口层叠 ToolStripMenuItem.Checked = true;
    this.水平平铺 ToolStripMenuItem.Checked = false;
    this.垂直平铺 ToolStripMenuItem.Checked = false;
}

private void 水平平铺 ToolStripMenuItem_Click(object sender, EventArgs e)
{
    LayoutMdi(MdiLayout.TileHorizontal);
    this.窗口层叠 ToolStripMenuItem.Checked = false;
    this.水平平铺 ToolStripMenuItem.Checked = true;
    this.垂直平铺 ToolStripMenuItem.Checked = false;
}

```

```

    }
    private void 垂直平铺 ToolStripMenuItem_Click(object sender, EventArgs e)
    {
        LayoutMdi(MdiLayout.TileVertical);
        this.窗口层叠 ToolStripMenuItem.Checked = false;
        this.水平平铺 ToolStripMenuItem.Checked = false;
        this.垂直平铺 ToolStripMenuItem.Checked = true;
    }
    public void closeFile()
    {
        if (this.MdiChildren.Count() > 0)
        {
            if (MessageBox.Show("你确定要关闭当前文档吗？", "Tip",
                MessageBoxButtons.OKCancel, MessageBoxIcon.Information) == DialogResult.OK)
            {
                for (int i = 0; i < dinosaurs.Count; i++)
                {
                    if (dinosaurs[i] == ((FrmDoc)this.ActiveMdiChild).Text)
                        dinosaurs.Remove(dinosaurs[i]);
                }
                ((FrmDoc)this.ActiveMdiChild).Close();
                _Num--;
            }
        }
    }
    public void 关闭 CToolStripMenuItem_Click(object sender, EventArgs e)
    {
        closeFile();
    }
    // 判断文件是否被保存，是则关闭文件，否则提示
    private void 退出 EToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Console.WriteLine(_Num);
        if (_Num > 1)
        {
            MessageBox.Show("请关闭全部文件再退出！", "Tip",
                MessageBoxButtons.OKCancel, MessageBoxIcon.Information);
        }
        else
        {
            if (MessageBox.Show("确定要退出吗？", "Tip", MessageBoxButtons.OKCancel,
                MessageBoxIcon.Information) == DialogResult.OK)
            {

```

```

        foreach (FrmDoc fd in this.MdiChildren)
            fd.Close();
        Application.Exit();
    }
}

// 下拉选择字体
private void tSCbBoxFontName_DropDownItemClicked(object sender,
ToolStripItemClickedEventArgs e)
{
    tSCbBoxFontName.Text = e.ClickedItem.ToString();
}
private void tSCbBoxFontName_Click(object sender, EventArgs e)
{
}
private void tSCbBoxFontName_TextChanged(object sender, EventArgs e)
{
    if (this.MdiChildren.Count() > 0)
    {
        RichTextBox tempRTB = new RichTextBox();
        int RtbStart = ((FrmDoc)this.ActiveMdiChild).rTBDoc.SelectionStart;
        int len = ((FrmDoc)this.ActiveMdiChild).rTBDoc.SelectionLength;
        int tempRtbStart = 0;
        Font font = ((FrmDoc)this.ActiveMdiChild).rTBDoc.SelectionFont;
        if (len <= 0 && null != font)
        {
            ((FrmDoc)this.ActiveMdiChild).rTBDoc.Font = new
Font(tSCbBoxFontName.Text, Font.Size, Font.Style);
            return;
        }
        tempRTB.Rtf = ((FrmDoc)this.ActiveMdiChild).rTBDoc.SelectedRtf;
        for(int i = 0; i < len; i++)
        {
            tempRTB.Select(tempRtbStart + i, 1);
            tempRTB.SelectionFont = new Font(tSCbBoxFontName.Text,
tempRTB.SelectionFont.Size, tempRTB.SelectionFont.Style);
        }
        tempRTB.Select(tempRtbStart, len);
        ((FrmDoc)this.ActiveMdiChild).rTBDoc.SelectedRtf = tempRTB.SelectedRtf;
        ((FrmDoc)this.ActiveMdiChild).rTBDoc.Select(RtbStart, len);
        ((FrmDoc)this.ActiveMdiChild).rTBDoc.Focus();
    }
}

```

```

        tempRTB.Dispose();
    }
}
// 改变字体样式
private void ChangeRTBFontStyle(RichTextBox rtb, FontStyle style)
{
    if (style != FontStyle.Bold && style != FontStyle.Italic && style !=
FontStyle.Underline)
    {
        throw new System.InvalidProgramException("字体格式错误! ");
    }
    RichTextBox tempRTB = new RichTextBox();
    int curRtbStart = rtb.SelectionStart;
    int len = rtb.SelectionLength;
    int tempRtbStart = 0;
    Font font = rtb.SelectionFont;
    if (len <= 0 && font != null)
    {
        if (style == FontStyle.Bold && font.Bold || style == FontStyle.Italic &&
font.Italic || style == FontStyle.Underline && font.Underline)
        {
            rtb.Font = new Font(font, Font.Style ^ style);
        }
        else if (style == FontStyle.Bold && !font.Bold || style == FontStyle.Italic
&& !font.Italic || style == FontStyle.Underline && !font.Underline)
        {
            rtb.Font = new Font(font, 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++)
    {
        tempRTB.Select(tempRtbStart + i, 1);
        if (style == FontStyle.Bold && tempFont.Bold || style == FontStyle.Italic &&
tempFont.Italic || 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)

```

```

        {
            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 粗体 ToolStripButton_Click(object sender, EventArgs e)
{
    if (this.MdiChildren.Count() > 0)
    {
        ChangeRTBFontStyle(((FrmDoc)this.ActiveMdiChild).rTBDoc, FontStyle.Bold);
    }
}

private void 斜体 ToolStripButton_Click(object sender, EventArgs e)
{
    if (this.MdiChildren.Count() > 0)
    {
        ChangeRTBFontStyle(((FrmDoc)this.ActiveMdiChild).rTBDoc, FontStyle.Italic);
    }
}

private void 下划线 ToolStripButton_Click(object sender, EventArgs e)
{
    if (this.MdiChildren.Count() > 0)
    {
        ChangeRTBFontStyle(((FrmDoc)this.ActiveMdiChild).rTBDoc,
FontStyle.Underline);
    }
}

private void 剪切 UToolStripButton_Click(object sender, EventArgs e)
{
    ((FrmDoc)this.ActiveMdiChild).rTBDoc.Cut();
}

private void 复制 CToolStripButton_Click(object sender, EventArgs e)
{

```

```
((FrmDoc)this.ActiveMdiChild).rTBDoc.Copy();
}

private void 粘贴_PToolStripButton_Click(object sender, EventArgs e)
{
    ((FrmDoc)this.ActiveMdiChild).rTBDoc.Paste();
}
// 左侧对齐
private void left_Click(object sender, EventArgs e)
{
    ((FrmDoc)this.ActiveMdiChild).rTBDoc.SelectionAlignment =
HorizontalAlignment.Left;
}
// 中心对齐
private void center_Click(object sender, EventArgs e)
{
    ((FrmDoc)this.ActiveMdiChild).rTBDoc.SelectionAlignment =
HorizontalAlignment.Center;
}
// 右侧对齐
private void right_Click(object sender, EventArgs e)
{
    ((FrmDoc)this.ActiveMdiChild).rTBDoc.SelectionAlignment =
HorizontalAlignment.Right;
}

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

private void redo_Click(object sender, EventArgs e)
{
    ((FrmDoc)this.ActiveMdiChild).rTBDoc.Redo();
}

private void color_Click(object sender, EventArgs e)
{
    if (colorDialog1.ShowDialog() == DialogResult.OK)
        ((FrmDoc)this.ActiveMdiChild).rTBDoc.SelectionColor = colorDialog1.Color;
}
// 插入图片
private void img_Click(object sender, EventArgs e)
{

```

```

        Bitmap bmp;
        OpenFileDialog openImageDialog = new OpenFileDialog();
        openImageDialog.Filter = "PNG 图片|*.png|JPG 图片|*.jpg|BMP 图片|*.bmp|所有类
型|*.*;";

        openImageDialog.Multiselect = true;
        if (openImageDialog.ShowDialog() == DialogResult.OK)
        {
            string filename = openImageDialog.FileName;
            try
            {
                bmp = new Bitmap(filename);
                Clipboard.SetDataObject(bmp);
                DataFormats.Format dft = DataFormats.GetFormat(DataFormats.Bitmap);
                if (((FrmDoc)this.ActiveMdiChild).rTBDoc.CanPaste(dft))
                    ((FrmDoc)this.ActiveMdiChild).rTBDoc.Paste(dft);
            }
            catch (Exception ex)
            {
                MessageBox.Show("图片插入失败" + ex.Message, "提示信息",
                MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }
        openImageDialog.Dispose();
    }

    private void fonts_Click(object sender, EventArgs e)
    {
        if (((FrmDoc)this.ActiveMdiChild).rTBDoc.SelectionCharOffset < 0)
        {
            ((FrmDoc)this.ActiveMdiChild).rTBDoc.SelectionCharOffset = 0;
        }
        else
        {
            ((FrmDoc)this.ActiveMdiChild).rTBDoc.SelectionCharOffset = -5;
        }
        ((FrmDoc)this.ActiveMdiChild).rTBDoc.Focus();
    }

    private void fontb_Click(object sender, EventArgs e)
    {
        if (((FrmDoc)this.ActiveMdiChild).rTBDoc.SelectionCharOffset > 0)
        {
            ((FrmDoc)this.ActiveMdiChild).rTBDoc.SelectionCharOffset = 0;
        }
    }

```



```
        else
        {
            ((FrmDoc)this.ActiveMdiChild).rTBDoc.SelectionCharOffset = 5;
        }
        ((FrmDoc)this.ActiveMdiChild).rTBDoc.Focus();
    }
}
}
```

实验三 学生通讯录

基本功能：xml 文件打开与保存、学生信息的添加、更新与删除

特色功能：判断学生信息是否已经被添加过、快捷导航、备份与恢复

1. 学生通讯录概述

主要内容是开发一个学生通讯录软件，采用 xml 格式储存数据，具备学生信息添加、编辑、删除、查找等功能。其过程包括设计目标、设计分析与算法流程、界面设计、关键类图、运行调试与项目扩展。通过该程序的开发，使我了解 xml 的基础知识，并掌握操作 xml 文件的基本方法。

2. 学生通讯录设计

2.1 设计目标

1) 学生通讯录

开发一个采用 xml 格式储存数据的学生通讯录 myContracts，具备学生信息添加、编辑、删除、查找等功能，运行界面如下图所示：



图 3-1

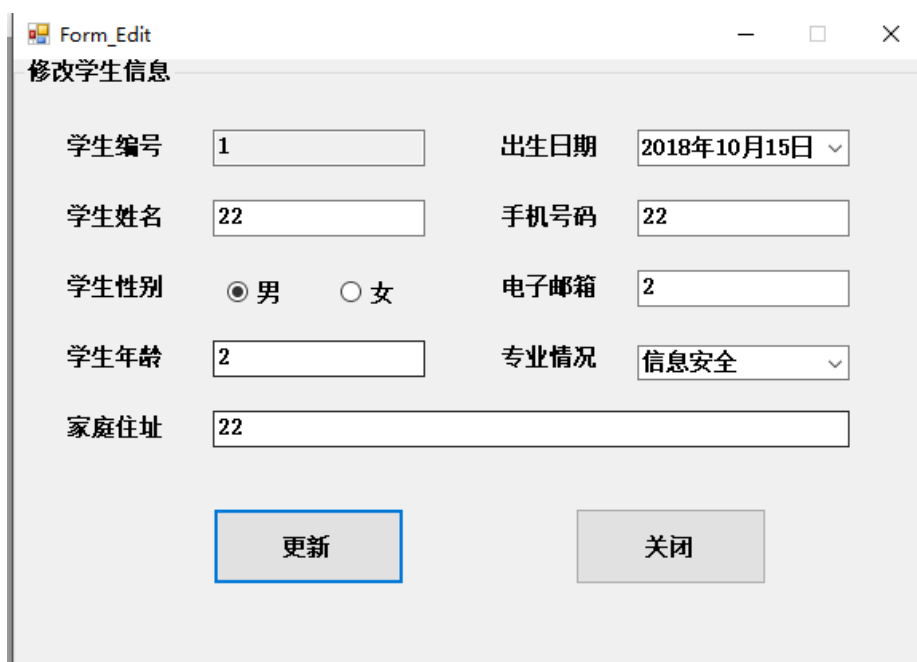
学生通讯录 myContracts 是一个多窗体桌面程序，不同窗体上分别布置了工具条、按钮、数据列表等控件，便于操作学生信息。通过鼠标单击工具条上的“添加”按钮，弹出添加学生信息的窗体，如下图所示：



The screenshot shows a Windows-style dialog box titled "Form_Add" with the subtitle "添加学生信息". It contains several input fields for student data: "学生编号" (Student ID), "学生姓名" (Student Name), "学生性别" (Student Gender) with radio buttons for "男" (Male) and "女" (Female), "学生年龄" (Student Age), "出生日期" (Date of Birth) with a date picker set to "2018年10月28日", "手机号码" (Mobile Number), "电子邮箱" (Email), "专业情况" (Major Situation) with a dropdown menu set to "选择专业", and "家庭住址" (Home Address). At the bottom, there are two buttons: "添加" (Add) and "关闭" (Close).

图 3-2

选中一条学生信息后，单击工具条上的“编辑”按钮，弹出修改学生信息的窗体，如下图所示：



The screenshot shows a Windows-style dialog box titled "Form_Edit" with the subtitle "修改学生信息". It contains the same input fields as Figure 3-2, but with pre-filled data: "学生编号" is "1", "学生姓名" is "22", "学生性别" has "男" selected, "学生年龄" is "2", "出生日期" is "2018年10月15日", "手机号码" is "22", "电子邮箱" is "2", "专业情况" is "信息安全", and "家庭住址" is "22". At the bottom, there are two buttons: "更新" (Update) and "关闭" (Close). The "更新" button is highlighted with a blue border.

图 3-3

点击工具条上的“查找”按钮，弹出查找学生信息的窗口，如下图：

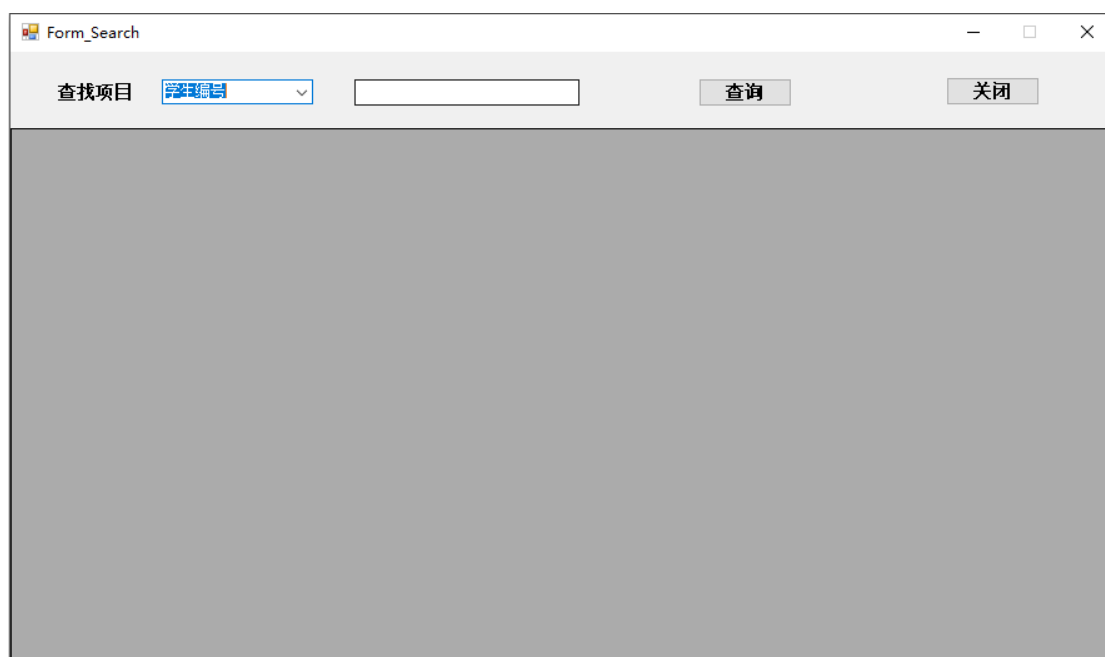


图 3-4

2) 项目扩展

在学生通讯录 myContracts 开发基础上，可以从不同的角度进一步扩展项目的功能和使用范围：

扩展学生通讯录 myContracts 的功能，在主界面的左侧增加分类窗口，在分类窗口中添加不同的专业类别，便于分类查找。增加学生通信数据的备份和恢复功能，以保证数据的安全。

2.2 设计分析与算法流程

学生通讯录 myContracts 中的学生信息以 xml 格式文件保存在文件中，对学生信息的添加、编辑、删除、查找操作都是通过操作 xml 文件实现的，其功能模块见下图：

各模块的功能描述如下：

学生信息添加：添加学生信息，包括姓名、性别、电话、邮箱等信息，不允许添加重复信息，若有重复信息，系统会进行提示。

学生信息编辑：对存在的信息进行编辑、更新。

学生信息删除：删除学生通讯录的学生信息，删除后无法恢复。

学生信息查找：可根据学生姓名等信息查询学生通讯录中的学生信息，便于检索。

学生信息分类：可根据预先设置的类别标签，对学生进行分类，便于管理，该功能属于扩展功能。

学生信息备份：对保存学生信息的 xml 文件进行备份，该功能属于扩展功能。

学生信息恢复：恢复学生信息文件，该功能属于扩展功能。

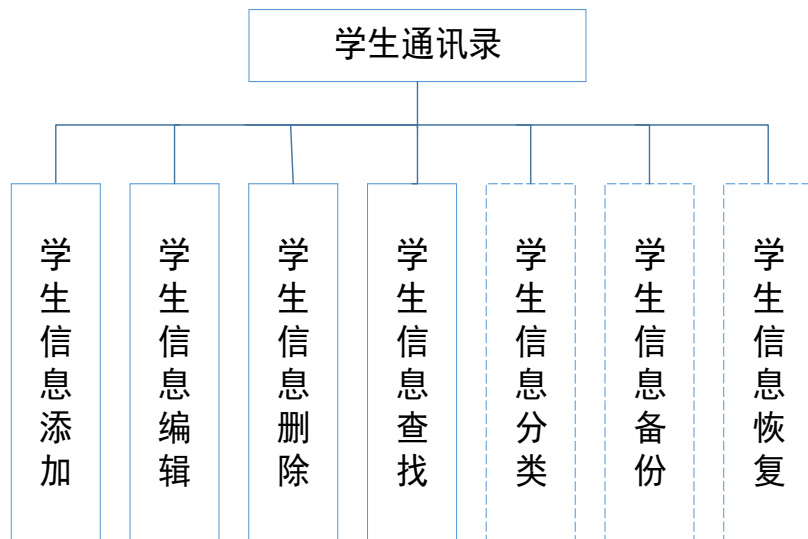


图 3-5

2.3 界面设计

myContracts 项目有多个窗体，项目启动运行后显示的是主窗体，通过主窗体上的工具条按钮中的按钮控件能够打开其他窗体，窗体间的调用关系如下图所示：

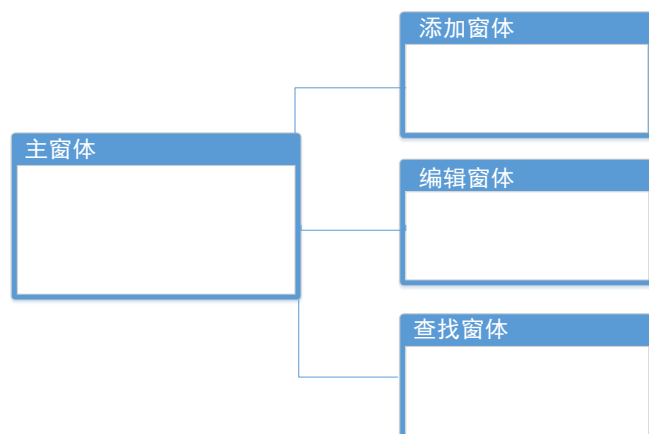


图 3-6

1) 主窗体设计

主窗体采用上、下结构布局，上半部分放置工具条，工具条中放置多个按钮，工具条中的按钮使用图标通过项目 Resources.res 导入；窗体的下半部分放置用于显示数据的控件。进入“工具箱”中，展开“公共控件”，将项目所需要的控件拖到 Form_Main 窗口上，并设置控件的属性。



图 3-7

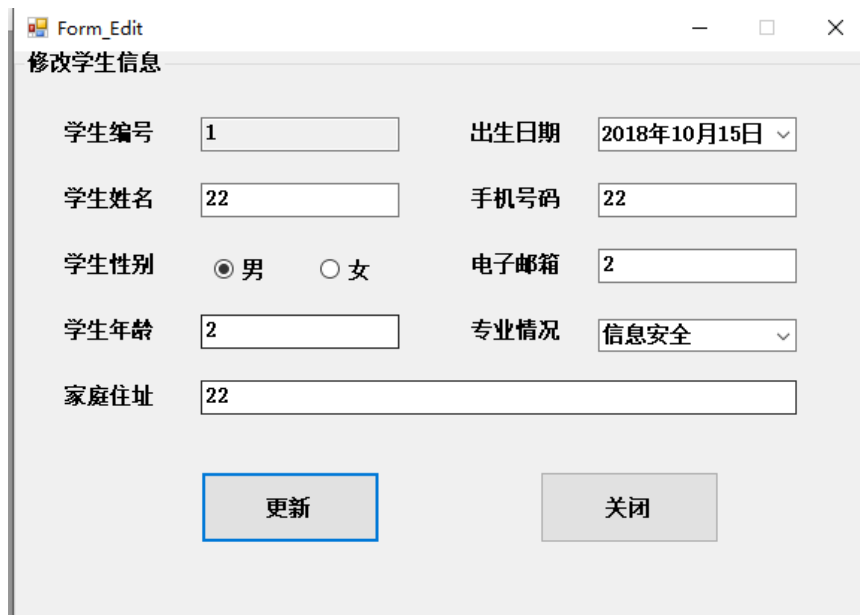
2) 添加窗体设计

进入“工具箱”中，展开“公共控件”，把项目所需要的控件拖到 Form_Add 窗体上，并设置控件的属性。

图 3-8

3) 编辑窗体设计

进入“工具箱”中，展开“公共控件”，把所需要的控件拖到 Form_Edit 窗体上，并设置控件的属性。

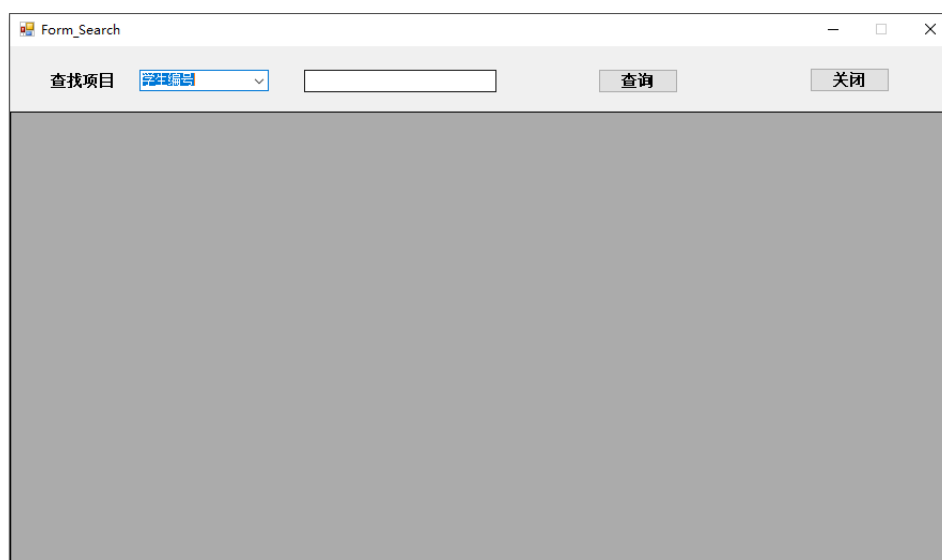


The image shows a Windows form titled "Form_Edit" with the subtitle "修改学生信息" (Modify Student Information). The form contains several input fields and buttons. The fields are arranged in two columns. The first column contains: "学生编号" (Student ID) with value "1", "学生姓名" (Student Name) with value "22", "学生性别" (Student Gender) with radio buttons for "男" (Male) and "女" (Female), "学生年龄" (Student Age) with value "2", and "家庭住址" (Home Address) with value "22". The second column contains: "出生日期" (Date of Birth) with a date picker showing "2018年10月15日", "手机号码" (Mobile Number) with value "22", "电子邮箱" (Email) with value "2", "专业情况" (Major Situation) with a dropdown menu showing "信息安全", and a large empty text area. At the bottom, there are two buttons: "更新" (Update) and "关闭" (Close).

图 3-9

4) 查找窗体设计

进入“工具箱”中，展开“公共控件”，把所需要的控件拖到 Form_Search 窗体上，并设置控件的属性。



The image shows a Windows form titled "Form_Search". The form has a search bar at the top with a dropdown menu labeled "查找项目" (Search Item) and a text input field. To the right of the input field are two buttons: "查询" (Search) and "关闭" (Close). The main area of the form is a large, empty gray rectangle.

图 3-10

2.4 关键类图

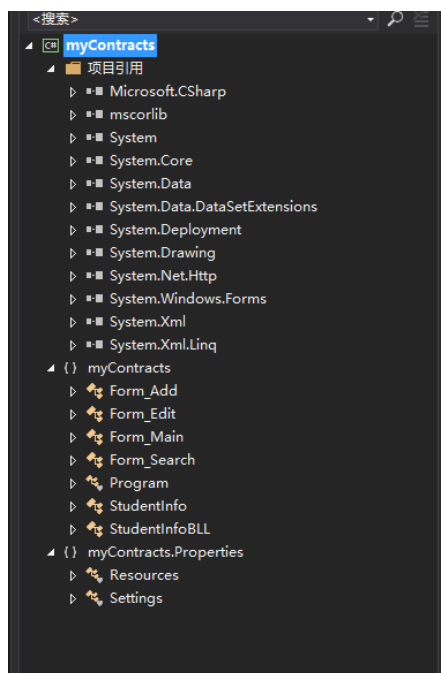


图 3-11

3. 学生通讯录现（运行调试）

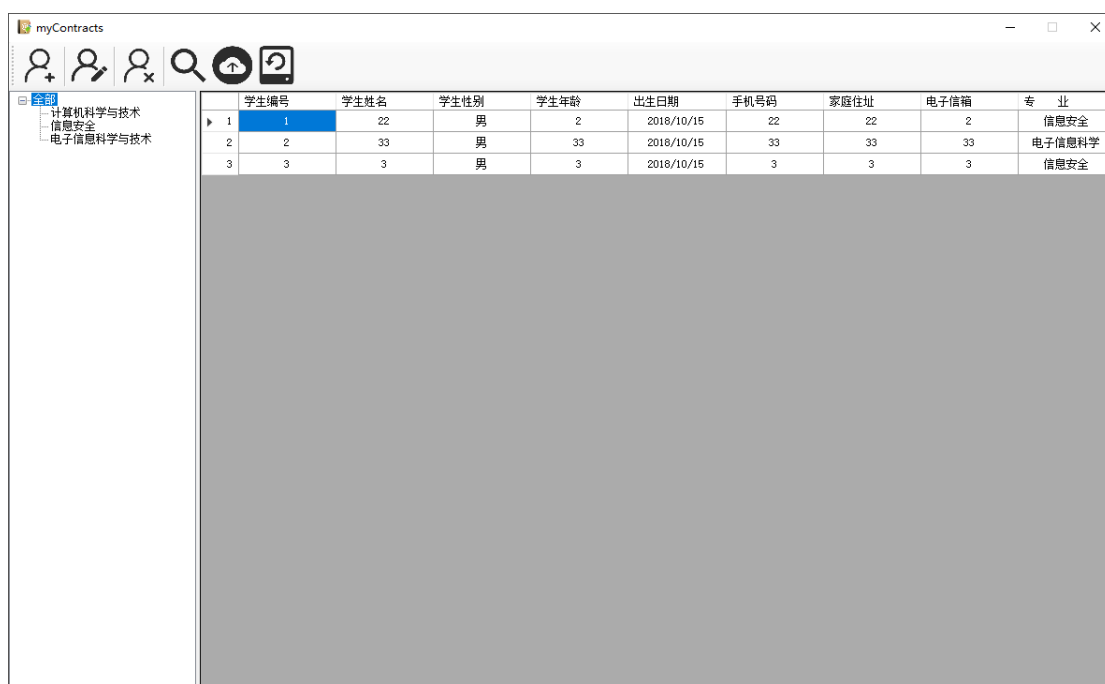


图 3-12

4 . 学生通讯录扩展

1) 在主页面的左侧增加分类窗口, 在分类窗口中添加不同的专业类别, 以便于分类查找。

①在主窗体上添加 SplitContainer 控件, 将工具条下的窗体部分分为左、右两部分, 左半部分放置树形控件 TreeView, 在树形控件中手工或通过代码添加学生专业的层次分类; 在右半部分放置 DataGridView 控件, 显示学生信息。

②选择窗体左侧的 TreeView 控件, 为其添加鼠标单击事件, 根据树形控件选中的节点值查询学生信息。

2) 扩展学生通讯录 myContracts 功能, 增加学生通讯录数据的备份和恢复功能。

①在主窗体上的工具条按钮控件中, 增加“备份”、“恢复”按钮。

②单击“备份”按钮时, 把保存学生信息的 Students.xml 文件保存到其他位置; 单击“恢复”按钮时, 用备份的 Students.xml 文件覆盖程序中默认的 Students.xml 文件。

③“备份”和“恢复”按钮可以使用 OpenFileDialog 控件、SaveFileDialog 控件。

5 . 总结

1) xml 基础

XML: Extensible Markup Language (可扩展标记语言) 的缩写, 是用来定义其它语言的一种元语言, 其前身是 SGML (Standard Generalized

Markup Language, 标准通用标记语言)。它没有标签集 (tag set), 也没有语法规则 (grammatical rule), 但是它有句法规则 (syntax rule)。

任何 XML 文档对任何类型的应用以及正确的解析都必须是良构的 (well-formed), 即每一个打开的标签都必须有匹配的结束标签, 不得

含有次序颠倒的标签, 并且在语句构成上应符合技术规范的要求。XML 文档可以是有效的 (valid), 但并非一定要求有效。所谓有效文档是指其符合其文档类型定义 (DTD) 的文档。如果一个文档符合一个模式 (schema) 的规定,

那么这个文档是“模式有效的（schema valid）”。

主要模块为：

生成 xml 文件、遍历 xml 文件的节点信息、修改 xml 文件的节点信息、向 xml 文件添加节点信息、删除指定 xml 文件的节点信息

2) xml 文件操作方法

```
//所需要添加的命名空间
using System.Xml;

//初始化一个 xml 实例
XmlDocument xml=new XmlDocument();

//导入指定 xml 文件
xml.Load(“xml 文件路径 path”);

//指定一个节点
XmlNode root=xml.SelectSingleNode(“节点名称”);

//获取节点下所有直接子节点
XmlNodeList childlist=root.ChildNodes;

//判断该节点下是否有子节点
root.HasChildNodes;

//获取同名同级节点集合
XmlNodeList nodelist=xml.SelectNodes(“节点名称”);

//生成一个新节点
XmlElement node=xml.CreateElement(“节点名称”);

//将节点加到指定节点下，作为其子节点
root.AppendChild(node);

//将节点加到指定节点下某个子节点前
root.InsertBefore(node,root.ChildeNodes[i]);

//为指定节点的新建属性并赋值
node.SetAttribute(“id”,“11111”);

//为指定节点添加子节点
root.AppendChild(node);
```

```
//获取指定节点的指定属性值
string id=node.Attributes["id"].Value;
//获取指定节点中的文本
string content=node.InnerText;
//保存 XML 文件
xml.Save(“xml 文件存储的路径 path”);
```

3) Combox 控件

ComBox 类的主要成员函数

因为组合框是由编辑框和列表框组合而成的，所以组合框的操作和编辑框与列表框的操作有很多相似之处，同样的，ComBox 类的成员函数也和 Edit 类与 ListBox 类的成员函数有很多相似之处，不但功能相似，甚至函数名和参数也很相似。

```
int GetCount( ) const;
```

获取组合框控件的列表框中列表项的数量。

```
int GetCurSel( ) const;
```

获取组合框控件的列表框中选中项的索引，如果没有选中任何项，该函数返回 CB_ERR。

```
int SetCurSel(int nSelect);
```

在组合框控件的列表框中选择某项。nSelect 参数指定了要选择的列表项的索引，如果为-1 则列表框中当前选择项被取消选中，编辑框也被清空。

```
DWORD GetEditSel( ) const;
```

获取组合框控件的编辑框中当前选择范围的起始和终止字符的位置。该函数返回一个 32 位数，低 16 位存放起始位置，高 16 位存放选择范围后第一个非选择字符的位置。如果该函数用于下拉列表式组合框时，会返回 CB_ERR。

```
BOOL SetEditSel(int nStartChar,int nEndChar);
```

用于在组合框控件的编辑框中选择字符。nStartChar 参数指定起始位置，nEndChar 参数指定终止位置。

```
DWORD_PTR GetItemData(int nIndex) const;
```

获取组合框中指定项所关联的 32 位数据。nIndex 参数指定组合框控件的列

表框某项的索引（从 0 开始）。

```
int SetItemData(int nIndex, DWORD_PTR dwItemData);
```

为某个指定的组合框列表项设置一个关联的 32 位数。nIndex 参数指定要进行设置的列表项索引。dwItemData 参数指定要关联的新值。

```
void GetLBText(int nIndex, CString& rString) const;
```

从组合框控件的列表框中获取某项的字符串。nIndex 参数指定要获取字符串的列表项的索引，CString 参数用于接收取到的字符串。

```
int GetLBTextLen(int nIndex) const;
```

获取组合框控件的列表框中某项的字符串长度。nIndex 参数指定要获取字符串长度的列表项的索引。

```
int GetTopIndex( ) const;
```

获取组合框控件的列表框中第一个可见项的索引。

```
-int SetTopIndex(int nIndex);
```

将组合框控件的列表框中某个指定项设置为可见的。nIndex 参数指定了该列表项的索引。该函数成功则返回 0，有错误发生则返回 CB_ERR。

```
BOOL LimitText(int nMaxChars);
```

用于限制用户在组合框控件的编辑框中能够输入的最大字节长度。nMaxChars 参数指定了用户能够输入文字的最大字节长度，如果为 0 则长度被限制为 65535 个字节。

```
int AddString(LPCTSTR lpszString);
```

为组合框控件中的列表框添加新的列表项。lpszString 参数是指向要添加的字符串的指针。该函数的返回值如果大于等于 0，那么它就是新列表项的索引，而如果有错误发生则会返回 CB_ERR，如果没有足够的内存存放新字符串则返回 CB_ERRSPACE。

```
int DeleteString(UINT nIndex);
```

删除组合框中某指定位置的列表项。nIndex 参数指定了要删除的列表项的索引。该函数的返回值如果大于等于 0，那么它就是组合框中剩余列表项的数量。如果 nIndex 指定的索引超出了列表项的数量则返回 CB_ERR。

```
int FindString(int nStartAfter, LPCTSTR lpszString) const;
```

在 组合框控件的列表框中查找但不选中第一个包含指定前缀的列表项。
 nStartAfter 参数指定了第一个要查找的列表项之前的那个列表项的索引。
 lpszString 指向包含要查找的前缀的字符串。该函数的返回值如果大于等于 0，
 那么它是匹配列表项的索引，如果查找失败则返回 CB_ERR。

```
int InsertString(int nIndex, LPCTSTR lpszString);
```

向组合框控件的列表框中插入一个列表项。nIndex 参数指定了要插入列表项的位置，lpszString 参数则指定了要插入的字符串。该函数返回字符串 被插入的位置，如果有错误发生则会返回 CB_ERR，如果没有足够的内存存放新字符串则返回 CB_ERRSPACE。

```
int SelectString(int nStartAfter, LPCTSTR lpszString);
```

在组合框控件的列表框中查找一个字符串，如果查找到则选中它，并将其显示到编辑框中。参数同 FindString。如果字符串被查找到则返回此列表项的索引，如果查找失败则返回 CB_ERR，并且当前选择项不改变。

4) 主要代码与分析

①Form_Main.cs 代码

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml;
using System.Xml.Linq;
namespace myContracts
{
    public partial class Form_Main : Form
    {
        public Form_Main()
        {
            InitializeComponent();
        }
        private void setTitle()
```

```

{
    dataGridView1.Columns[0].HeaderText = "学生编号";
    dataGridView1.Columns[1].HeaderText = "学生姓名";
    dataGridView1.Columns[2].HeaderText = "学生性别";
    dataGridView1.Columns[3].HeaderText = "学生年龄";
    dataGridView1.Columns[4].HeaderText = "出生日期";
    dataGridView1.Columns[5].HeaderText = "手机号码";
    dataGridView1.Columns[6].HeaderText = "家庭住址";
    dataGridView1.Columns[7].HeaderText = "电子信箱";
    dataGridView1.Columns[8].HeaderText = "专    业";
}

private void Form_Main_Load(object sender, EventArgs e)
{
    if (File.Exists(System.Windows.Forms.Application.StartupPath + "/Students.xml"))
    {
        dataGridView1.DataSource = StudentInfoBLL.GetAllstudentInfo();
    }
    else
    {
        StudentInfoBLL.CreateStudentXml();
        dataGridView1.DataSource = StudentInfoBLL.GetAllstudentInfo();
    }

    setTitle();
}

public void intiContracts()
{
    dataGridView1.DataSource = StudentInfoBLL.GetAllstudentInfo();
    setTitle();
}

private void toolStrip_add_Click(object sender, EventArgs e)
{
    Form_Add formadd = new Form_Add();
    formadd.ShowDialog();
    intiContracts();
}

private void toolStrip_edit_Click(object sender, EventArgs e)
{
    if (dataGridView1.SelectedRows.Count == 1)
    {
        int selectrow = Int32.Parse(dataGridView1.SelectedCells[0].Value.ToString());
        Form_Edit formedit = new Form_Edit();
        formedit.studentid_edit = selectrow;
    }
}

```

```
        formedit.ShowDialog();
        intiContracts();
    }
    else
    {
        MessageBox.Show("请选中一行后再进行编辑！");
    }
}

private void toolStrip_delete_Click(object sender, EventArgs e)
{
    if (dataGridView1.SelectedRows.Count == 1)
    {
        if (MessageBox.Show("确定要删除此学生信息吗？", "确认信息",
        MessageBoxButtons.YesNo, MessageBoxIcon.Warning, MessageBoxDefaultButton.Button2) ==
        DialogResult.Yes)
        {
            int selectrow = Int32.Parse(dataGridView1.SelectedCells[0].Value.ToString());
            if (StudentInfoBLL.DeleteStudentInfo(selectrow))
                MessageBox.Show("删除学生信息成功！");
            else
                MessageBox.Show("删除学生信息失败，请检查是否选中学生信息！");
            intiContracts();
        }
        else
            MessageBox.Show("请选中一行后再点击删除按钮！");
    }
}

private void toolStrip_search_Click(object sender, EventArgs e)
{
    Form_Search formsearch = new Form_Search();
    formsearch.ShowDialog();
}

private void treeView1_AfterSelect(object sender, TreeViewEventArgs e)
{
    Console.WriteLine(e.Node.Text);
    switch (e.Node.Text)
    {
        case "全部":
            showData("全部");
            break;
        case "计算机科学与技术":
            showData("计算机科学与技术");
            break;
    }
}
```

```

        case "信息安全":
            showData("信息安全");
            break;
        case "电子信息科学与技术":
            showData("电子信息科学与技术");
            break;
    }
}

private void showData(string pro)
{
    List<StudentInfo> studentList = new List<StudentInfo>();
    XElement xml = XElement.Load(StudentInfoBLL._basePath);
    var studentVar = xml.Descendants("student");
    if (pro == "全部")
    {
        studentList = StudentInfoBLL.GetAllStudentInfo();
    }
    else
    {
        studentList = (from student in studentVar
                        where student.Element("pro").Value == pro
                        select new StudentInfo
                        {
                            StudentId = Int32.Parse(student.Attribute("studentid").Value),
                            Name = student.Element("name").Value,
                            Age = Int32.Parse(student.Element("age").Value),
                            Sex = student.Element("sex").Value,
                            BirthDate =
DateTime.Parse(student.Element("birthdate").Value),
                            Phone = student.Element("phone").Value,
                            Address = student.Element("address").Value,
                            Email = student.Element("email").Value,
                            Pro = student.Element("pro").Value
                        }).ToList();
    }

    dataGridView1.DataSource = studentList;
    setTitle();
}

private void toolStrip_backup_Click(object sender, EventArgs e)
{
    SaveFileDialog savefiledialog1 = new SaveFileDialog();
    savefiledialog1.Filter = "XML 文件|*.xml";
    savefiledialog1.FileName = "Student";
}

```



```

savefiledialog1.AddExtension = true;
if (savefiledialog1.ShowDialog() == DialogResult.OK)
{
    try
    {
        string url = savefiledialog1.FileName.ToString();
        XDocument studentDoc = new XDocument();
        XDeclaration xDeclaration = new XDeclaration("1.0", "utf-8", "yes");
        studentDoc.Declaration = xDeclaration;
        XElement xElement = new XElement("studentcontract");
        studentDoc.Add(xElement);
        studentDoc.Save(url);
        XElement xml = XElement.Load(url);
        for (int i = 0; i < dataGridView1.Rows.Count; i++)
        {
            int StudentId =
Int32.Parse(dataGridView1.Rows[i].Cells[0].Value.ToString());
            string Name = dataGridView1.Rows[i].Cells[1].Value.ToString();
            string Sex = dataGridView1.Rows[i].Cells[2].Value.ToString();
            int Age = Int32.Parse(dataGridView1.Rows[i].Cells[3].Value.ToString());
            DateTime BirthDate =
DateTime.Parse(dataGridView1.Rows[i].Cells[4].Value.ToString());
            string Phone = dataGridView1.Rows[i].Cells[5].Value.ToString();
            string Email = dataGridView1.Rows[i].Cells[6].Value.ToString();
            string Address = dataGridView1.Rows[i].Cells[7].Value.ToString();
            string Pro = dataGridView1.Rows[i].Cells[8].Value.ToString();
            XElement studentXml = new XElement("student");
            studentXml.Add(new XAttribute("studentid", StudentId));
            studentXml.Add(new XElement("name", Name));
            studentXml.Add(new XElement("sex", Sex));
            studentXml.Add(new XElement("age", Age.ToString()));
            studentXml.Add(new XElement("birthdate",
BirthDate.ToString("yyyy-MM-dd")));
            studentXml.Add(new XElement("phone", Phone));
            studentXml.Add(new XElement("address", Address));
            studentXml.Add(new XElement("email", Email));
            studentXml.Add(new XElement("pro", Pro));
            xml.Add(studentXml);
        }
        xml.Save(url);
        MessageBox.Show("保存成功");
    }
    catch
    {

```

```

        MessageBox.Show("保存失败");
    }
}
private void toolStrip_rec_Click(object sender, EventArgs e)
{
    OpenFileDialog openFileDialog1 = new OpenFileDialog();
    openFileDialog1.Filter = "XML 文件|*.xml";
    openFileDialog1.Multiselect = false;
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        try
        {
            if (openFileDialog1.FilterIndex == 1)
            {
                string url = openFileDialog1.FileName.ToString();
                StudentInfoBLL_basePath = url;
                List<StudentInfo> studentList = new List<StudentInfo>();
                XElement xml = XElement.Load(url);
                var studentVar = xml.Descendants("student");
                studentList = (from student in studentVar
                               select new StudentInfo
                               {
                                   StudentId =
Int32.Parse(student.Attribute("studentid").Value),
                                   Name = student.Element("name").Value,
                                   Age = Int32.Parse(student.Element("age").Value),
                                   Sex = student.Element("sex").Value,
                                   BirthDate =
DateTime.Parse(student.Element("birthdate").Value),
                                   Phone = student.Element("phone").Value,
                                   Address = student.Element("address").Value,
                                   Email = student.Element("email").Value,
                                   Pro = student.Element("pro").Value
                               }).ToList();
                dataGridView1.DataSource = studentList;
                setTitle();
            }
            MessageBox.Show("恢复成功");
        }
        catch
        {
            MessageBox.Show("恢复失败");
        }
    }
}

```

```

    }
}
private void dataGridView1_RowPostPaint(object sender,
DataGridViewRowPostPaintEventArgs e)
{
    //自动编号，与数据无关
    Rectangle rectangle = new Rectangle(e.RowBounds.Location.X,
        e.RowBounds.Location.Y,
        dataGridView1.RowHeadersWidth - 4,
        e.RowBounds.Height);
    TextRenderer.DrawText(e.Graphics,
        (e.RowIndex + 1).ToString(),
        dataGridView1.RowHeadersDefaultCellStyle.Font,
        rectangle,
        dataGridView1.RowHeadersDefaultCellStyle.ForeColor,
        TextFormatFlags.VerticalCenter | TextFormatFlags.Right);
}
}
}

```

②Form_Edit.cs

```

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 myContracts
{
    public partial class Form_Edit : Form
    {
        public int studentid_edit = 0;
        public Form_Edit()
        {
            InitializeComponent();
        }
        private void Form_Edit_Load(object sender, EventArgs e)
        {
            initControl();
        }
        public void initControl()
        {

```

```

StudentInfo studentinfo = StudentInfoBLL.GetStudentInfo(studentid_edit);
if (studentinfo != null)
{
    stu_id.Text = studentinfo.StudentId.ToString();
    stu_name.Text = studentinfo.Name;
    if (studentinfo.Sex == "男")
    {
        stu_man.Checked = true;
        stu_woman.Checked = false;
    }
    else
    {
        stu_man.Checked = false;
        stu_woman.Checked = true;
    }
    stu_age.Text = studentinfo.Age.ToString();
    stu_bir.Text = studentinfo.BirthDate.ToString();
    stu_phone.Text = studentinfo.Phone;
    stu_email.Text = studentinfo.Email;
    stu_address.Text = studentinfo.Address;
    cb_pro.Text = studentinfo.Pro;
}
}

private void btn_update_Click(object sender, EventArgs e)
{
    StudentInfo studentinfo = StudentInfoBLL.GetStudentInfo(studentid_edit);
    studentinfo.StudentId = Int32.Parse(stu_id.Text);
    studentinfo.Name = stu_name.Text;
    if (stu_man.Checked)
        studentinfo.Sex = "男";
    else if (stu_woman.Checked)
        studentinfo.Sex = "女";
    studentinfo.Age = Int32.Parse(stu_age.Text);
    studentinfo.BirthDate = DateTime.Parse(stu_bir.Text);
    studentinfo.Phone = stu_phone.Text;
    studentinfo.Email = stu_email.Text;
    studentinfo.Address = stu_address.Text;
    studentinfo.Pro = cb_pro.Text;
    if (StudentInfoBLL.UpdateStudentInfo(studentinfo))
        if (MessageBox.Show("修改学生信息成功!", "Info!", MessageBoxButtons.OK) ==
DialogResult.OK)
        {
            this.Close();
        }
}

```

```

    }
    private void btn_close_Click(object sender, EventArgs e)
    {
        this.Close();
    }
}
}

```

③Form_Add.cs

```

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 myContracts
{
    public partial class Form_Add : Form
    {
        public Form_Add()
        {
            InitializeComponent();
        }
        private void btn_add_Click(object sender, EventArgs e)
        {
            StudentInfo studentinfo = new StudentInfo();
            bool flag = true;
            studentinfo.StudentId = Int32.Parse(stu_id.Text);
            studentinfo.Name = stu_name.Text;
            if (stu_man.Checked)
                studentinfo.Sex = "男";
            else if (stu_woman.Checked)
                studentinfo.Sex = "女";
            studentinfo.Age = Int32.Parse(stu_age.Text);
            studentinfo.BirthDate = DateTime.Parse(stu_bir.Text);
            studentinfo.Phone = stu_phone.Text;
            studentinfo.Email = stu_email.Text;
            studentinfo.Address = stu_address.Text;
            //studentinfo.Pro = stu_pro.Text;
            try
            {
                Console.WriteLine(cb_pro.SelectedItem.ToString());
            }
        }
    }
}

```

```

        flag = true;
    }
    catch
    {
        flag = false;
        MessageBox.Show("请选择您的专业!");
    }
    if(flag)
    {
        studentinfo.Pro = cb_pro.SelectedItem.ToString();
        if (!StudentInfoBLL.ConductStuID(stu_id.Text))
        {
            if (StudentInfoBLL.AddStudentInfo(studentinfo))
            {
                if (MessageBox.Show("添加学生信息成功!", "Info!", MessageBoxButtons.OK)
== DialogResult.OK)
                {
                    this.Close();
                }
            }
            else
            {
                MessageBox.Show("请填写其他的学生ID!");
            }
        }
    }
    private void btn_close_Click(object sender, EventArgs e)
    {
        this.Close();
    }
    private void Form_Add_Load(object sender, EventArgs e)
    {
    }
}

```

④ Form_Search.cs

```

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 myContracts
{
    public partial class Form_Search : Form
    {
        public Form_Search()
        {
            InitializeComponent();
        }

        private void Form_Search_Load(object sender, EventArgs e)
        {
            stu_searchitem.Text = "学生编号";
        }

        void InitHeadTitle()
        {
            dataGridView1.Columns[0].HeaderText = "学生编号";
            dataGridView1.Columns[1].HeaderText = "学生姓名";
            dataGridView1.Columns[2].HeaderText = "学生性别";
            dataGridView1.Columns[3].HeaderText = "学生年龄";
            dataGridView1.Columns[4].HeaderText = "出生日期";
            dataGridView1.Columns[5].HeaderText = "手机号码";
            dataGridView1.Columns[6].HeaderText = "家庭住址";
            dataGridView1.Columns[7].HeaderText = "电子信箱";
            dataGridView1.Columns[8].HeaderText = "专 业";
        }

        private void btn_search_Click(object sender, EventArgs e)
        {
            if (stu_search.Text != string.Empty)
            {
                StudentInfo studentsearch = new StudentInfo();
                switch (stu_searchitem.SelectedIndex)
                {
                    case 0:
                        studentsearch.StudentId = Int32.Parse(stu_search.Text);
                        break;
                    case 1:
                        studentsearch.Name = stu_search.Text;
                        break;
                }
                dataGridView1.DataSource = StudentInfoBLL.GetStudentInfoList(studentsearch);
                InitHeadTitle();
            }
            else
            {
                MessageBox.Show("请输入要查询的" + stu_searchitem.Text);
            }
        }
    }
}
```

```
        }  
    }  
    private void btn_close_Click(object sender, EventArgs e)  
    {  
        this.Close();  
    }  
}  
}
```


实验四 拼图游戏

基本功能：开始游戏、设置拼图块数、查看原图、选择图片

特色功能：记录步数、记录时间、挑战模式、按键移动一个空白块进行拼图

1. 拼图游戏概述

主要内容是设计开发一个趣味拼图游戏，其功能是对加载的图片进行分割（如分割成 3×3 矩阵或者其他类型矩阵）并随机加载到图片框矩阵中，用户使用按键移动一个空白元素进行拼图，系统能够自动判别拼图是否成功并进行提示。

2. 拼图游戏设计

2.1 设计目标

本项目目标是设计开发一个支持鼠标拖动拼图的游戏软件，软件能够自动对加载的图片进行分割，并打乱顺序后放置在不同的图片框中，用户使用鼠标拖动图片框中的图片进行拼图，拼图成功后，系统会自动进行提示，其运行效果如图所示。



图 4-1

其主要更能描述如下。

图片尺寸自适应：为方便用户能够对不同尺寸的图片进行拼图，软件加入了图片尺寸自动调整功能，能对不同尺寸的图片进行自动调整以满足拼图游戏软件的要求。

图片动态分割：能自动将图片分割为不同的大小，如 3×3 矩阵或其他类型的矩阵，便于控制拼图游戏的难易程度。

查看原图：为方便拼图，允许用户在拼图过程随时查看原图。

切换图片：可以另外选择用户自己喜欢的图片进行拼图，增加趣味性。

鼠标拖动拼图：选择相应图片框中的图片后，按下鼠标左键进行拖动，放置到适应位置即可，方便用户操作。

自动判断拼图成功：软件自动记忆图片切割后的原始状态，能够对用户拼图是否成功自动做出判断并进行提示。

2.2 设计分析与算法流程

拼图游戏的算法流程如图所示，其过程大致分为以下几个步骤：

软件启动后，依据预先设置的图片矩阵大小（默认 3×3 ），在主窗体上动态生成图片框矩阵，并初始化每个图片框的位置（窗体上的坐标轴方向如 X 轴、Y 轴），图片框在窗体上的排列顺序是从左到右，从上到下，如图所示（以 3×3 图片框矩阵为例）。

加载默认图片，并将图片调整为适合拼图游戏操作的大小，本软件默认将图片调整为 600×600 ，单位为像素。

对成功加载后的图片进行分割（分割为 3×3 的图片矩阵），将分割后的图片按照一一对应的原则加载到步骤（1）中生成的图片框矩阵中，并记录每张图片的原始位置。

随机打乱图片框中的图片的位置，并重新加载图片到图片框矩阵中。

使用鼠标拖动图片框中的图片开始进行拼图，用户每次把一个图片框中的图片拖动到新的位置，系统都会将图片矩阵的状态与步骤（3）中初始状态进行比较，如果一致，则显示拼图成功，否则继续进行拼图。

拼图成功后，用户可以选择随机加载系统自带的图片进行下一次拼图游戏，

或加载用户自己喜爱的图片进行拼图。

在拼图的过程中，用户可以随时查看原图作为提示或通过重置图片重新开始拼图。

2.3 界面设计

拼图游戏项目包括两个窗体：一个是进行拼图的主窗体，另一个是显示原图的窗体，下面分别说明。

主窗体设计

主窗体采用左右布局，左边用于显示图片框矩阵，同时也是用户进行拼图操作的位置；右边是功能区，放置控制程序的按钮和部分必要的文字输出显示。为方便控制，本项目使用 splitContainer 控件（其允许程序运行时调整左右两部分的大小）将窗体划分为左右两部分：左半部分是 splitContainer1.Panel1，右半部分是 splitContainer1.Panel2。

在左半部分的 splitContainer1.Panel1 控件上放置一个 Panel 控件，用于放置图片框矩阵。

在右半部分的 splitContainer1.Panel2 控件上放置需要操作的按钮。

主窗体的设计示意图如图。

进入“工具箱”中，展开“公共控件”，把拼图游戏项目所需要的控件拖到 From_Main 窗口上，并设置控件的属性。

2.4 关键类图

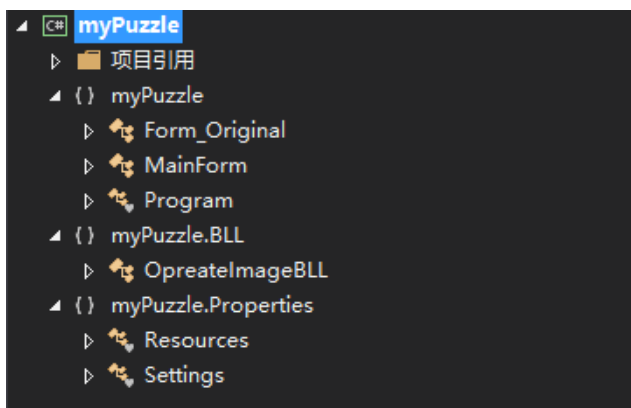


图 4-2

3 . 拼图游戏实现（运行调试）

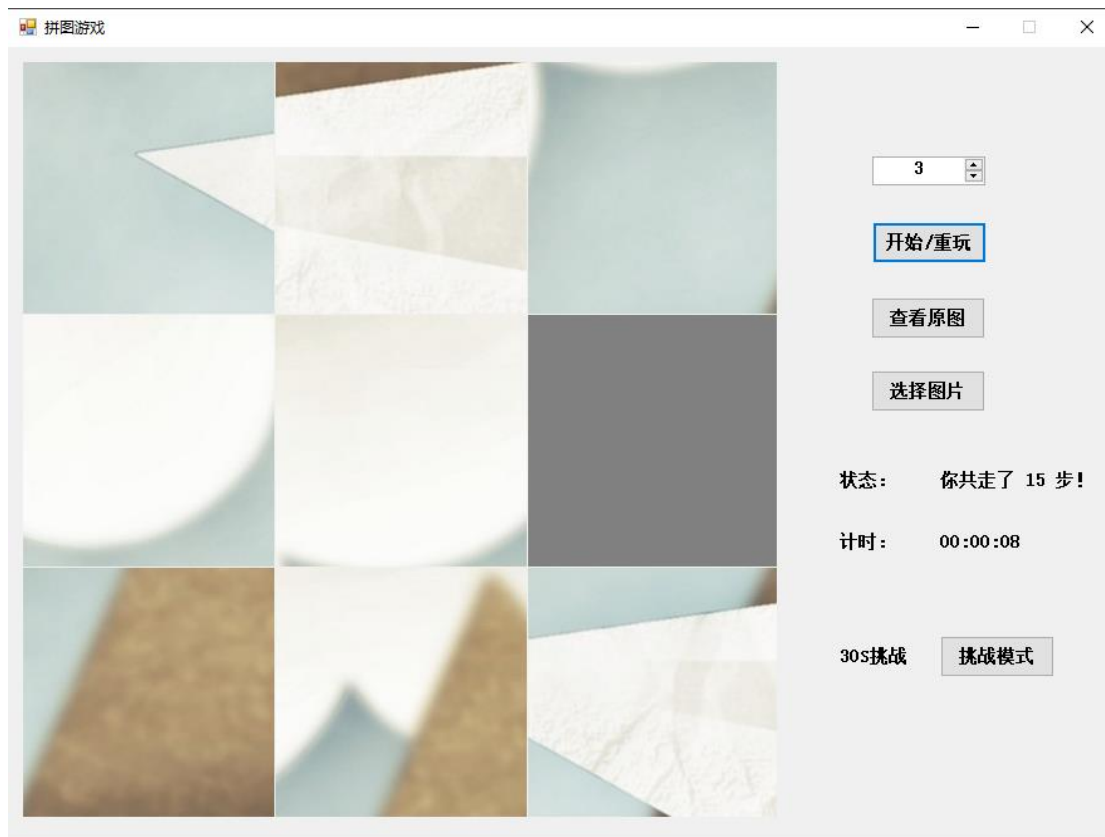


图 4-5

4 . 拼图游戏扩展

1) 切割块数

增加一个可以选择图片切割边数的控件，可以用此来增加游戏的难度。

2) 计时、计步

设置一个变量用来存储每次走的步数，用一个 Timer 控件来计时，并继续扩展一个倒计时的挑战模式。

5 . 总结

1) Timer 控件总结

该控件的主要作用是按一定的时间间隔周期性的触发一个名为 Tick 的事件，因此在该事件的代码中可以放置一些需要每隔一段时间重复执行的程序段，常用属性及方法如下：

Enabled：用来设定定时器是否正在运行。

Interval：用来设定定时器两次 Tick 事件发生的时间间隔，以 ms 为单位。

Start：用来启动定时器。

Stop：用来停止计时器。

2) PictureBox 控件

Width 和 Height 属性：

Width 和 Height 属性用来获取或设置图片框控件的宽度和高度，它们集成了 Size 属性，其度量单位由盛放此控件的容器来决定。如放在窗体上的是 PictureBox 控件，则其单位为像素。可以通过属性设计器或代码来设置图片框的大小。

Left 和 Top 属性：

Left 和 Top 属性来设置或获取图片框在容器内的位置，以容器的坐标系统表示。Left 用来设置或获取图片控件左边框相对于容器工作区左边框的距离（通常以像素为单位）；Top 用来设置或获取图片控件上边框相对于容器工作区顶部的距离（通常以像素为单位）。改变这两个属性的值时，控件的位置将发生变化。在属性设计器能通过 Location 的 X 与 Y 设定。

SizeMode 属性：

SizeMode 属性用来设置图片的显示模式，有 Normal、StretchImage、AutoSize、CenterImage 和 Zoom 5 种显示模式。

Normal 表示普通的显示模式，在此模式下，图片置于 PictureBox 的左上角，图片的大小由 PictureBox 控件的大小决定，当图片的尺寸大于 PictureBox 的尺寸时，多余的图像将被剪裁掉。

当采用 StretchImage 模式时, PictureBox 会根据自身尺寸的长宽调整图片的比例, 使图片在 PictureBox 中完整的显示出来, 图片形状可能会失真。

Zoom 模式表示将按图片的尺寸比较缩放图片使其完整地显示在 PictureBox 中。此种模式下的缩放图片形状不会失真。

AutoSize 模式表示图片框会根据图片的大小自动调整自身的大小以显示图片的全部内容。

在 CenterImage 模式下, 当图片尺寸小于 PictureBox 尺寸时, 使图片显示在 PictureBox 工作区的正中央, 当图片大于 PictureBox 大小时, 就显示图片的中央部分。

Image 属性:

Image 属性用来设置 PictureBox 中的图像。用户可以在属性设计器中指定图片, 在属性设计器中选择 Image 打开“选择资源”对话框。

ImageLocation 属性:

ImageLocation 属性用来获取或设置要在 PictureBox 中显示的图像的路径, 此功能为 .NET Framework 新增的功能。

3) 主要代码与分析

```
MainForm.cs
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Drawing.Imaging;
using System.IO;
namespace myPuzzle
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
        }
    }
}
```

```
System.DateTime TimeNow = new DateTime();
TimeSpan TimeCount = new TimeSpan();
int PointX = 0;
int PointY = 0;
int Imagecol = 0;
int Imagerow = 0;
int stepCount = 0;

bool flag = false;
bool challenge = false;
bool startGame = false;
string originalpicpath = "//Resources/tu.png";

Image[,] images = null;
private void MainForm_Load(object sender, EventArgs e)
{

    labStatus.Text = "未开始 ";

}

private void MainForm_KeyPress(object sender, KeyPressEventArgs e)
{
    if (startGame)
    {
        string key = (PointY * Imagecol + PointX).ToString();
        Control c = panel1.Controls.Find(key, true)[0];
        PictureBox pEmpty = null;
        if (c != null)
        {
            pEmpty = c as PictureBox;
        }
        if (e.KeyChar == 87 || e.KeyChar == 119)
        {
            if (PointY != (Imagerow - 1))
            {
                PointY++;
                stepCount++;
            }
        }

        if (e.KeyChar == 83 || e.KeyChar == 115)
        {
```

```

        if (PointY != 0)
        {
            PointY--;
            stepCount++;
        }
    }
    if (e.KeyChar == 65 || e.KeyChar == 97)
    {
        if (PointX != (Imagecol - 1))
        {
            PointX++;
            stepCount++;
        }
    }
    if (e.KeyChar == 68 || e.KeyChar == 100)
    {
        if (PointX != 0)
        {
            PointX--;
            stepCount++;
        }
    }
    string keynow = (PointY * Imagecol + PointX).ToString();
    Control conNow = panel1.Controls.Find(keynow, true)[0];
    PictureBox pNow = conNow as PictureBox;
    pEmpty.Image = pNow.Image;
    pNow.Image = null;
    pNow.BackColor = Color.Gray;
    if (BLL.OpreateImageBLL.CheckImageGamesComplate(panel1, Imagerow, Imagecol))
    {
        timer1.Stop();
        MessageBox.Show("恭喜完成! ");
        challenge = false;
        startGame = false;
    }
    labStatus.Text = "你共走了 " + stepCount.ToString() + " 步! ";
}

}

private void condect()
{
    PointX = 0;
    PointY = 0;
    BLL.OpreateImageBLL.DestoryPictureBox(panel1);

```



```

        Image img = global::myPuzzle.Properties.Resources.tu;
        flag = true;
        Imagecol = int.Parse(numericUpDown1.Value.ToString());
        Imagerow = int.Parse(numericUpDown1.Value.ToString());
        panel1.BackgroundImage = null;
        BLL.OpreateImageBLL.CreateImageBoxesByRowAndColumn(img, Imagecol, Imagerow,
panel1);
        images = BLL.OpreateImageBLL.images;
    }
    private void start()
    {
        startGame = true;
        stepCount = 0;
        condect();
        Dictionary<string, Image> dic = new Dictionary<string, Image>();
        Random random = new Random();
        int diccount = 0;
        int c = random.Next(0, Imagecol);
        int r = random.Next(0, Imagerow);
        while (dic.Count < (Imagecol * Imagerow - 1))
        {
            if (c == 0 && r == 0)
            {
                c = random.Next(0, Imagecol);
                r = random.Next(0, Imagerow);
                continue;
            }
            if (!dic.ContainsValue(images[r, c]))
            {
                dic.Add(diccount.ToString(), images[r, c]);
                diccount++;
            }
            c = random.Next(0, Imagecol);
            r = random.Next(0, Imagerow);
        }
        for (int i = 0; i < panel1.Controls.Count; i++)
        {
            PictureBox p = panel1.Controls[i] as PictureBox;
            p.Image = null;
            if (i != 0)
            {
                p.Image = dic[(i - 1).ToString()];
            }
        }
    }
}

```

```

        timer1.Start();
        TimeNow = DateTime.Now;
        labStatus.Text = "你共走了 " + stepCount.ToString() + " 步! ";
    }
    private void btnStart_Click(object sender, EventArgs e)
    {
        challenge = false;
        start();
    }
    private void timer1_Tick(object sender, EventArgs e)
    {
        TimeCount = DateTime.Now - TimeNow;
        labTime.Text = string.Format("{0:00}:{1:00}:{2:00}", TimeCount.Hours,
TimeCount.Minutes, TimeCount.Seconds);
        if (challenge)
        {
            if (TimeCount.Hours * 3600 + TimeCount.Minutes * 60 + TimeCount.Seconds > 30)
            {
                labStatus.Text = "挑战失败! ";
                challenge = false;
                startGame = false;
                start();
                timer1.Stop();
                timer1.Start();
            }
        }
    }
    private void btnOriginal_Click(object sender, EventArgs e)
    {
        Form_Original original = new Form_Original();
        original.picpath = originalpicpath;
        if (flag)
            original.flag = flag;
        else
            original.flag = flag;
        original.ShowDialog();
    }
    private void btnChoose_Click(object sender, EventArgs e)
    {
        if (new_picture.ShowDialog() == DialogResult.OK)
        {
            flag = false;
            originalpicpath = new_picture.FileName;
            PointX = 0;
        }
    }

```

```

    PointY = 0;
    BLL.OpreatelImageBLL.DestoryPictureBox(panel1);
    Image img = System.Drawing.Image.FromFile(originalpicpath);
    Imagecol = int.Parse(numericUpDown1.Value.ToString());
    Imagerow = int.Parse(numericUpDown1.Value.ToString());
    panel1.BackgroundImage = null;
    BLL.OpreatelImageBLL.CreateImageBoxesByRowAndColumn(img, Imagecol,
Imagerow, panel1);
    images = BLL.OpreatelImageBLL.images;
    stepCount = 0;
    Dictionary<string, Image> dic = new Dictionary<string, Image>();
    Random random = new Random();
    int diccount = 0;
    int c = random.Next(0, Imagecol);
    int r = random.Next(0, Imagerow);
    while (dic.Count < (Imagecol * Imagerow - 1))
    {
        if (c == 0 && r == 0)
        {
            c = random.Next(0, Imagecol);
            r = random.Next(0, Imagerow);
            continue;
        }
        if (!dic.ContainsValue(images[r, c]))
        {
            dic.Add(diccount.ToString(), images[r, c]);
            diccount++;
        }
        c = random.Next(0, Imagecol);
        r = random.Next(0, Imagerow);
    }
    for (int i = 0; i < panel1.Controls.Count; i++)
    {
        PictureBox p = panel1.Controls[i] as PictureBox;
        p.Image = null;
        if (i != 0)
        {
            p.Image = dic[(i - 1).ToString()];
        }
    }
    timer1.Start();
    TimeNow = DateTime.Now;
    labStatus.Text = "你共走了 " + stepCount.ToString() + " 步! ";
}

```



```

using System.Text;
using System.Drawing;
using System.Drawing.Imaging;
using System.IO;
using System.Windows.Forms;
namespace myPuzzle.BLL
{
    public class OpreateImageBLL
    {
        public static Image[,] images { get; set; }
        public static Image[,] GetImages(Image img, int col, int row, int width, int height)
        {
            Image[,] imgs = new Image[row, col];
            int iwidth = 600 / col;
            int iheight = 600 / row;
            for (int i = 0; i < row; i++)
            {
                for (int j = 0; j < col; j++)
                {
                    Bitmap bm = new Bitmap(iwidth, iheight, PixelFormat.Format24bppRgb);
                    Graphics g = Graphics.FromImage(bm);
                    g.DrawImage(img, new Rectangle(0, 0, iwidth, iheight), new Rectangle(j * iwidth,
i * iheight, iwidth, iheight), GraphicsUnit.Pixel);
                    imgs[i, j] = (Image)bm;
                    g.Dispose();
                }
            }
            images = imgs;
            return imgs;
        }
        public static void DestoryPictureBox(Panel panel)
        {
            while (panel.Controls.Count != 0)
            {
                panel.Controls.RemoveAt(0);
            }
        }
        public static void CreateImageBoxesByRowAndColumn(Image img, int col, int row, Panel
panel)
        {
            int width = 600 / col;
            int height = 600 / row;
            Image[,] images = GetImages(img, col, row, img.Width, img.Height);
            int cou = 0;

```

```

    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            PictureBox p = new PictureBox();
            p.Name = cou.ToString();
            p.Width = width;
            p.Height = height;
            p.Left = j * (width + 1);
            p.Top = i * (height + 1);
            panel.Controls.Add(p);
            cou++;
            if (i == 0 && j == 0)
            {
                p.BackColor = Color.Gray;
                continue;
            }
            p.Image = images[i, j];
        }
    }
}

public static bool CheckImageGamesComplate(Panel panel, int row, int col)
{
    bool win = false;
    try
    {
        int pStart = 0;
        for (int i = 0; i < row; i++)
        {
            for (int j = 0; j < col; j++)
            {
                PictureBox pic = panel.Controls.Find(pStart.ToString(), true)[0] as
PictureBox;

                if (i == 0 && j == 0)
                {
                    if (pic.Image != null)
                    {
                        return false;
                    }
                }
                else if (pic.Image == images[i, j])
                {
                    win = true;
                }
            }
        }
    }
}

```

```
        else
        {
            return false;
        }
        pStart++;
    }
}
}
catch
{
    return false;
}
return win;
}
}
}
```