## 中国矿业大学计算机学院

**18 级本科生课程设计报告**

课程名称 程序设计综合实验

报告时间 2019年12月16日

学生姓名 聂鹏远

学 号 08182871

专 业 计算机科学与技术

任课教师 闫秋艳

## 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。 | | | 20% | |  |
| 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。 | | | 20% | |  |
| 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。 | | | 60% | |  |
| 总成绩 | | | | | |  |
| 指导教师 | |  | 评阅日期 | |  | |

## 目 录

**实验一 简单计算器…………………………………………………1**

**系统概述……………………………………………………………1**

**系统设计……………………………………………………………1**

**系统实现……………………………………………………………2**

**系统拓展……………………………………………………………3**

**总结…………………………………………………………………8**

**实验二 多文档编辑器 ……………………………………………9**

**系统概述……………………………………………………………9**

**系统设计……………………………………………………………9**

**系统实现……………………………………………………………11**

**系统拓展……………………………………………………………12**

**总结…………………………………………………………………15**

**实验三 学生通讯录 ………………………………………………16**

**系统概述……………………………………………………………16**

**系统设计……………………………………………………………16**

**系统实现……………………………………………………………21**

**系统拓展……………………………………………………………23**

**总结…………………………………………………………………26**

**实验四 拼图游戏 …………………………………………………27**

**系统概述……………………………………………………………27**

**系统设计……………………………………………………………27**

**系统实现……………………………………………………………31**

**系统拓展……………………………………………………………32**

**总结…………………………………………………………………34**

**实验一 简单计算器**

1 系统概述

开发一个支持连续计算的简单计算器，其过程包括项目分析、界面设计、代码编写和运行调试。

2 系统设计

2.1 设计目标

一个支持连续计算的四则运算计算器，其中加减乘除等基本运算，并将运算结果显示在输出文本栏，同时也具备清空、后退等功能。

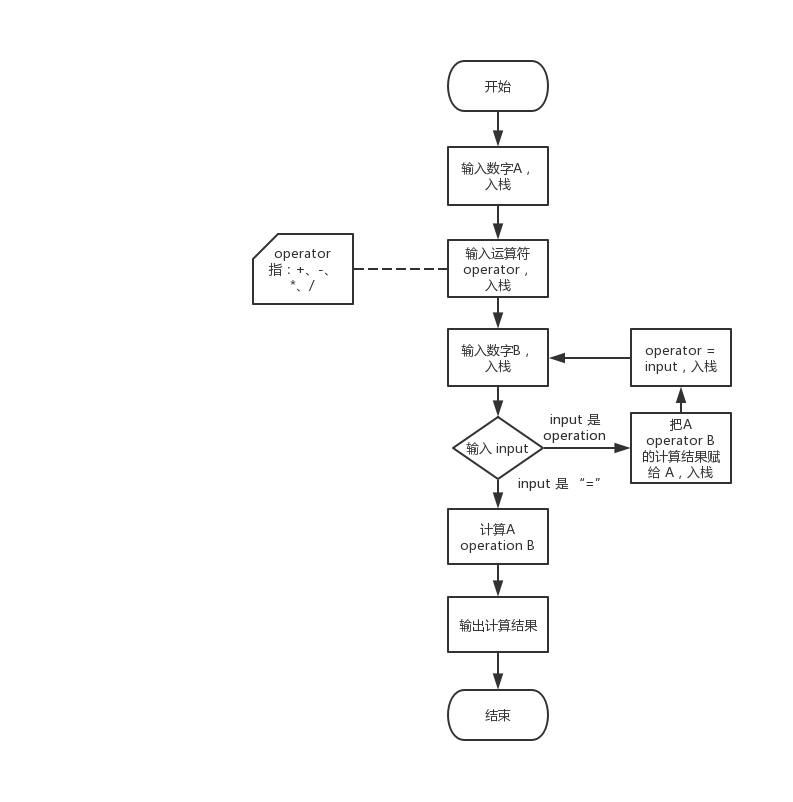
2.2 设计分析与算法流程（如图1所示）

图 1

2.3 界面设计（如图2所示）

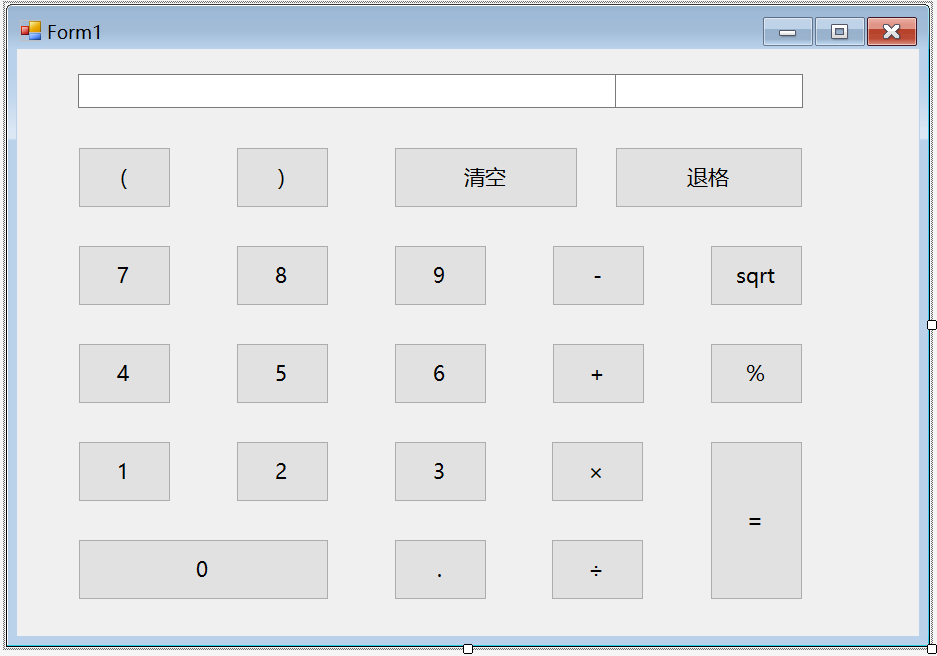


图 2

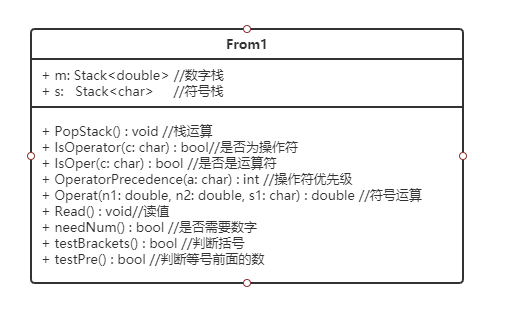
2.4 关键类图（如图3所示）

图 3

3 系统实现（运行调试）

简单计算过程（如图4所示）

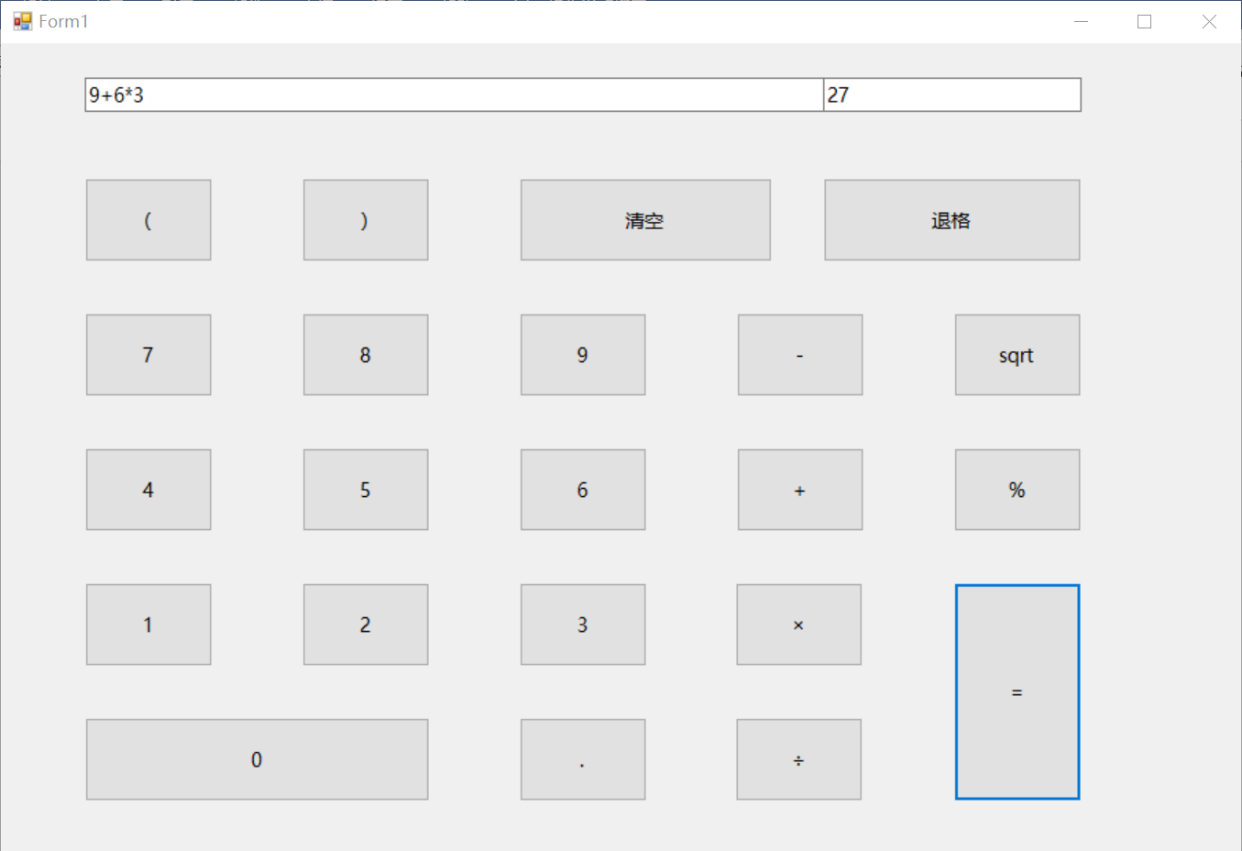


图 4

4 系统扩展

（1）添加了之前内容的可视化，实现了格式错误的提示及友情提示。

格式判断（如图5所示）

图 5

（2）实现了带括号的优先级，同时增填了开方和取模运算。

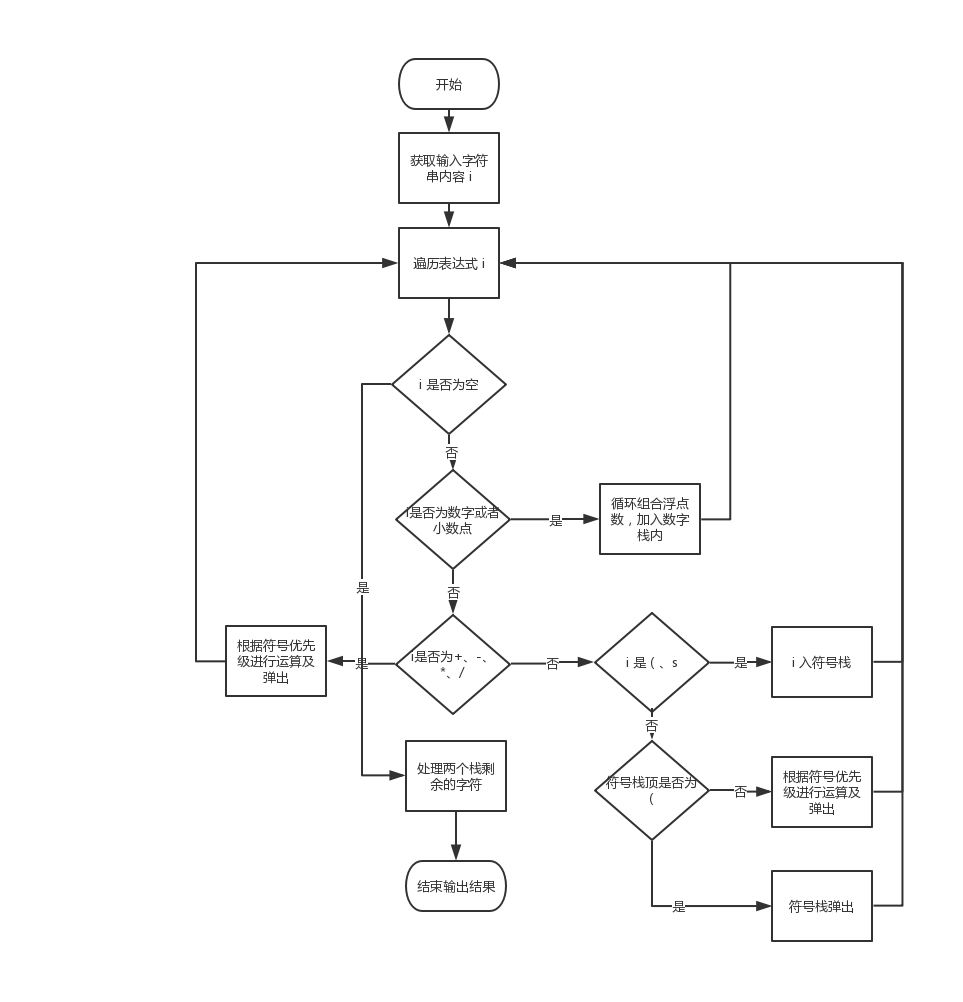
栈优先级实现流程图（如图6所示）

图 6

优先级调试（如图7所示）

图 7

优先级实现相关代码：

public void Read() //读值 实现优先级

{

string text = textDisplay.Text;

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

{

if (!IsOperator(text[i])) //数字和小数点

{

string s = null;

while (i < text.Length && !IsOperator(text[i]))

{

s += text[i];

i++;

}

i--;

double mm = Convert.ToDouble(s);

m.Push(mm);

///MessageBox.Show(mm.ToString());///////////

}

else if (IsOper(text[i])) //+ - \* /

{

if (s.Count.Equals(0) || s.Peek().Equals('(') || s.Peek().Equals('s'))

{

s.Push(text[i]);

/// MessageBox.Show(text[i].ToString());//////////

}

else if (OperatorPrecedence(text[i]) > OperatorPrecedence(s.Peek()))

{

s.Push(text[i]);

/// MessageBox.Show(text[i].ToString());////////////////

}

else if (s.Count != 0 && m.Count >= 2)

{

double n1, n2;

char s1;

n2 = m.Pop();

n1 = m.Pop();

s1 = s.Pop();

double sum = Operat(n1, n2, s1);

m.Push(sum);

/// MessageBox.Show(sum.ToString());////////////////

s.Push(text[i]);

/// MessageBox.Show(text[i].ToString());////////////////

}

}

else // s、（ 和 ）

{

if (text[i].Equals('(') || text[i].Equals('s'))

{

s.Push(text[i]);

///MessageBox.Show(text[i].ToString());///////////////

if (text[i].Equals('s'))

{

i += 4;

}

}

else if (text[i].Equals(')'))

{

while (!s.Peek().Equals('(') && !s.Peek().Equals('s'))

{

double n1, n2;

char s1;

n2 = m.Pop();

n1 = m.Pop();

s1 = s.Pop();

double sum = Operat(n1, n2, s1);

m.Push(sum);

/// MessageBox.Show(sum.ToString());////////////////

}

if (s.Peek().Equals('s'))

{

double use = m.Pop();

use = Math.Pow(use, 0.5);

m.Push(use);

///MessageBox.Show(use.ToString());////////////////

s.Pop();

}

else

{

s.Pop();

}

}

}

}

}

5 总结

本项目较为简单，仅在实现优先级是稍为麻烦，很适合上手，增添对 C# 的相关语句进行了基础了解，对两个栈判定并实现优先级进行了细致的运用，感觉跟 VB 差不多，算了重温了 VB 的相关知识。

**实验二 多文档编辑器**

1 系统概述

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多个文档，每个文档独立地执行软件所需要的功能。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成为多文档应用程序，子窗体之间可以进行数据交互，也可以互不相干。

2 系统设计

2.1 设计目标

设计开发一个简单的多文档文本编辑器 SimpleMDIExample，具有新建、打开、保存一个文本文件，设置字体、字型功能。

2.2 设计分析与算法流程

简单的多文档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、字型等的设置功能。

流程如图8所示

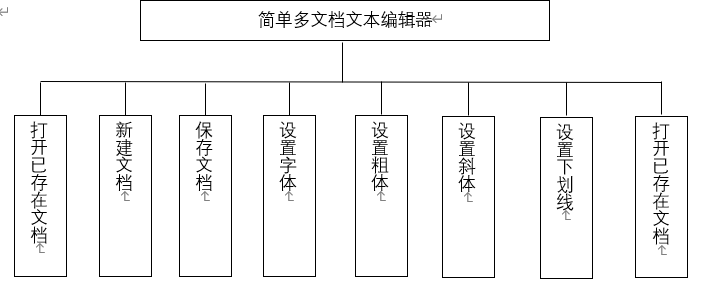


图 8

各功能模块具体功能描述：

打开已存在文档：可以打开以.txt为结尾的文本文档，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）。

2.3 界面设计

主要界面（如图9所示）

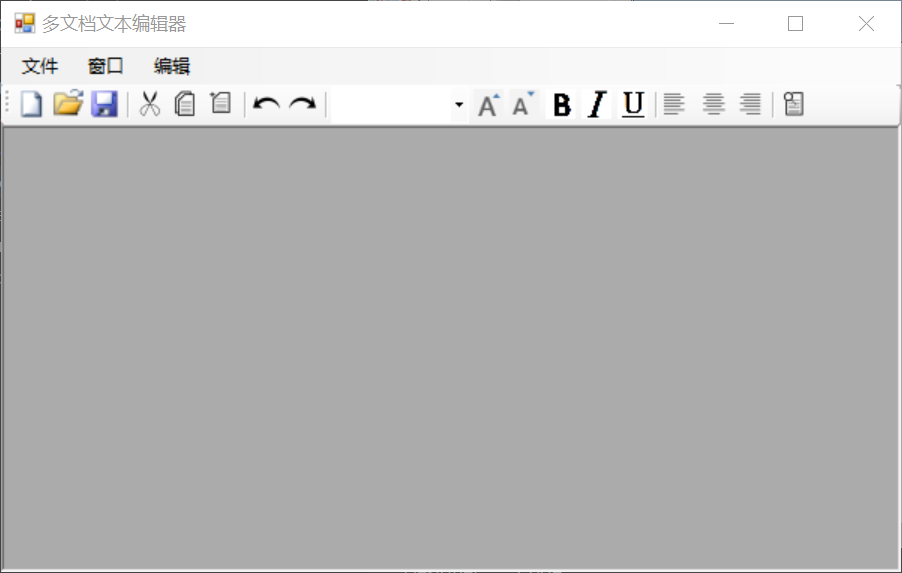


图 9

2.4 关键类图（如图10所示）

图 10

3 系统实现（运行调试）

文档横向排列（如图11所示）

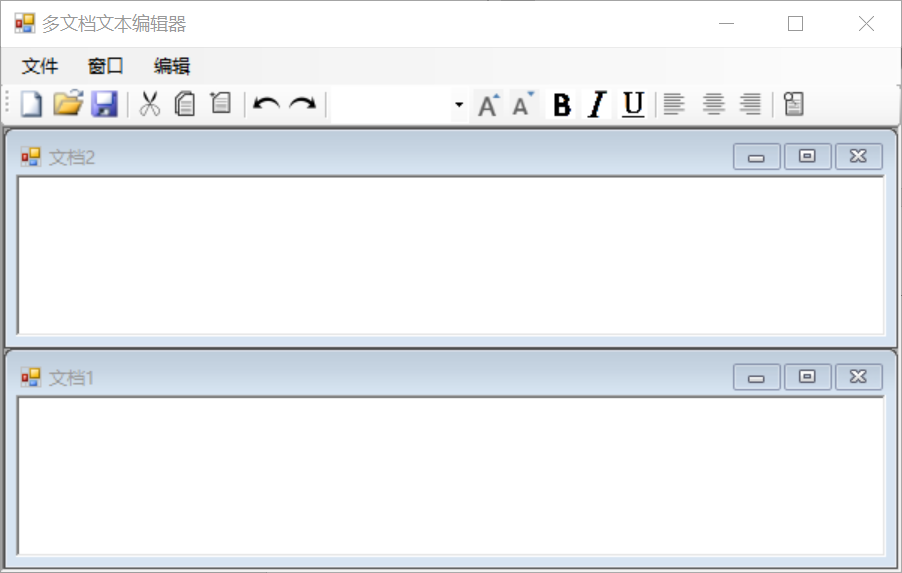


图 11

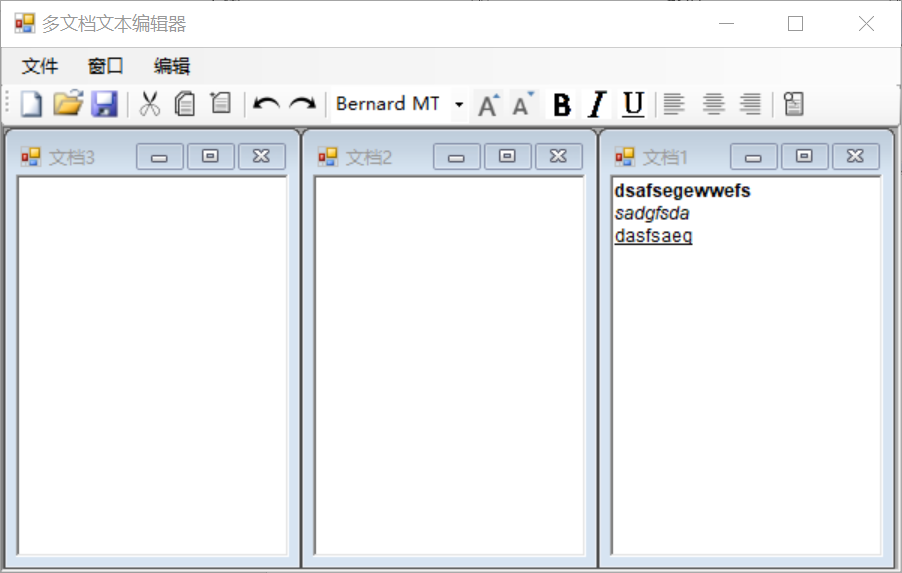
文档纵向排列+输入文本+加粗+斜体+下划线（如图12所示）

图 12

4 系统扩展

放大字体：对选中的文本进行放大。

缩小字体：对选中的文本进行缩小。

剪切：对选中的本文进行剪切操作。

复制：对选中的文本进行复制操作。

粘贴：对剪切或复制选中的文本进行粘贴操作。

撤消：撤消上一步操作。

左对齐：对选中的本文进行左对齐。

居中：对选中的本文进行居中。

右对齐：对选中的本文进行右对齐。

查找功能：查找输入文本，并支持查找到下一个，以及到末尾的提示。

运行调试：放大字体+查找第一个（如图13所示）

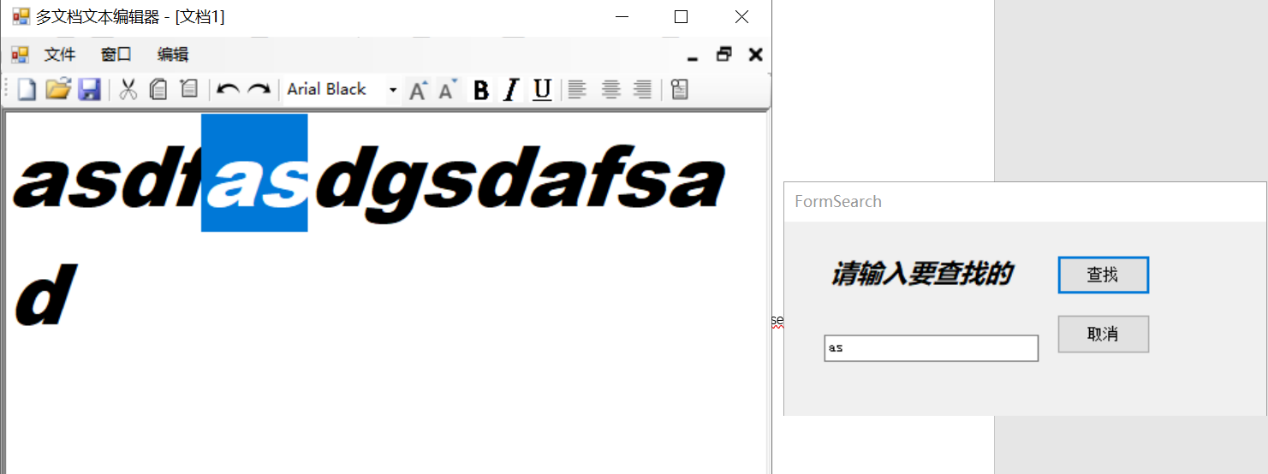


图 13

运行调试：放大字体+查找到最后（如图14所示）

图 14

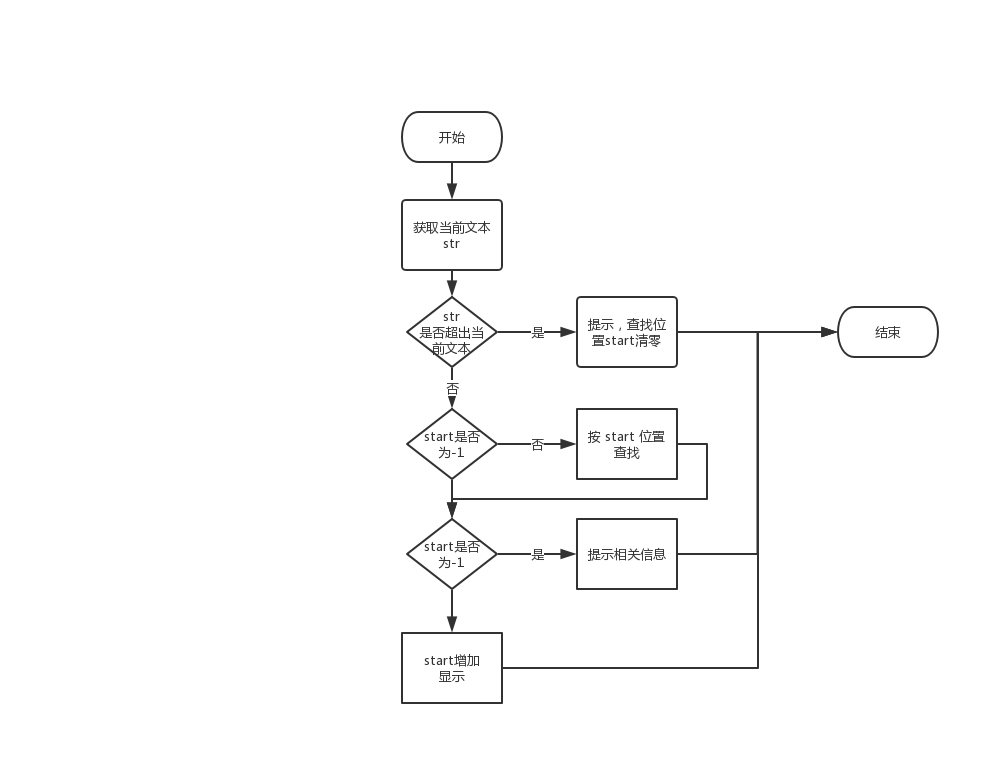
 查找流程图（如图15所示）

图 15

查找实现代码：

private void search(object sender, EventArgs e) //查找文本

{

string str = textSearch.Text;

if (start >= \_rtb.Text.Length)

{

MessageBox.Show("已查找到尾部");

start = 0;

}

else

{

if (start != -1)

{

start = \_rtb.Find(str, start, RichTextBoxFinds.MatchCase);

}

if (start == -1)

{

if (count == 0)

{

MessageBox.Show("没有该字符");

}

else

{

MessageBox.Show("已查找到尾部！");

start = 0;

count = 0;

}

}

else

{

start = start + str.Length;

count++;

\_rtb.Focus();

}

}

}

5 总结

比上一个实验多了面向对象的过程，稍有项目的样子，模式很简单，就是内容稍微繁琐，查找功能比较有意思且关键。

**实验三 学生通讯录**

1 系统概述

设计开发一个学生通讯录软件，采用xml格式存储数据，具备学生信息添加、编辑、删除、查找等功能。

2 系统设计

2.1 设计目标

开发一个采用xml格式储存数据的学生通信录myContracts，具备学生信息添加、编辑、删除、查找等功能。

学生通信录myContracts是一个多窗体桌面程序，不同窗体上面分别布置了工具条、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。

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。

点击工具条上的“查找”按钮，弹出查找学生信息的窗口。

2.2 设计分析与算法流程

学生通信录myContracts中的学生信息以xml格式保存在文件中，对学生信息的添加、编辑、删除、查询操作都是通过操作xml文件实现的。

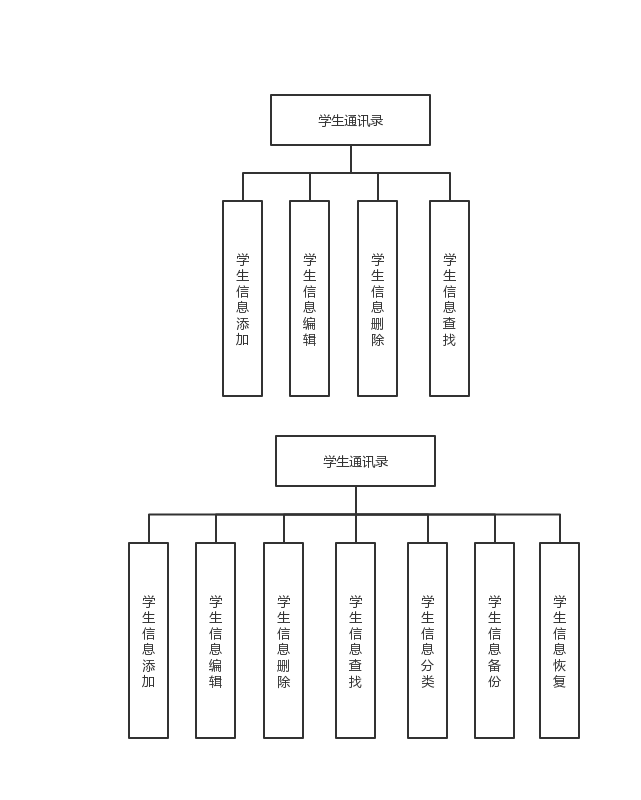
流程图如图16所示

图 16

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：删除学生通信录的学生信息，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信录中的学生信息，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。

学生信息备份：对保存学生信息的xml文件进行备份。

学生信息恢复：恢复学生信息文件。

2.3 界面设计

主页面实现（如图17所示）

图 17

查找页面实现（如图18所示）

图 18

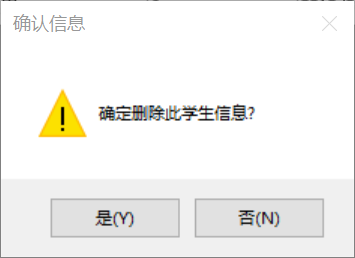
删除页面实现（如图19所示）

图 19

添加页面实现（如图20所示）

图 20

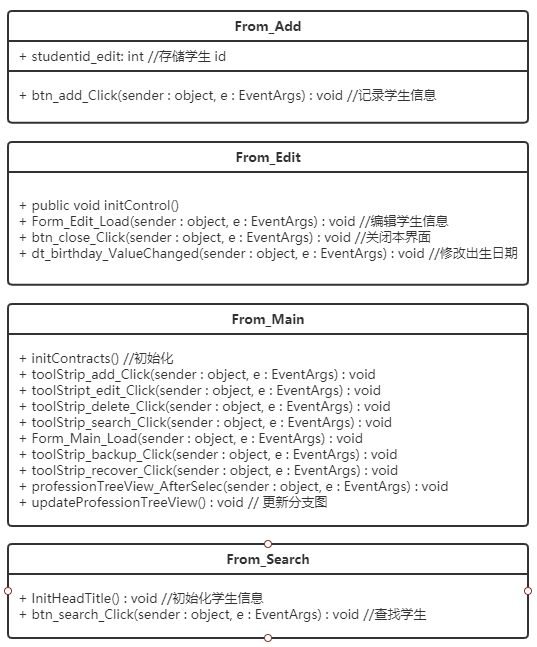
2.4 关键类图（如图21所示）

图 21

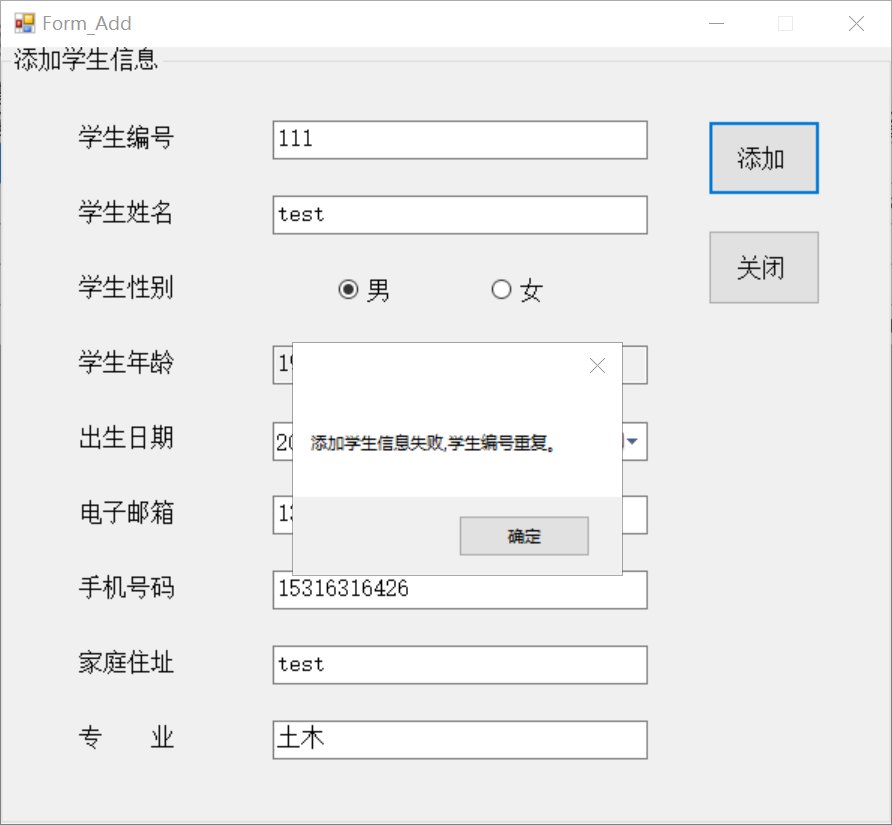
1. 系统实现（运行调试）
2. 添加功能调试（如图22、23所示）

图 22

图 23

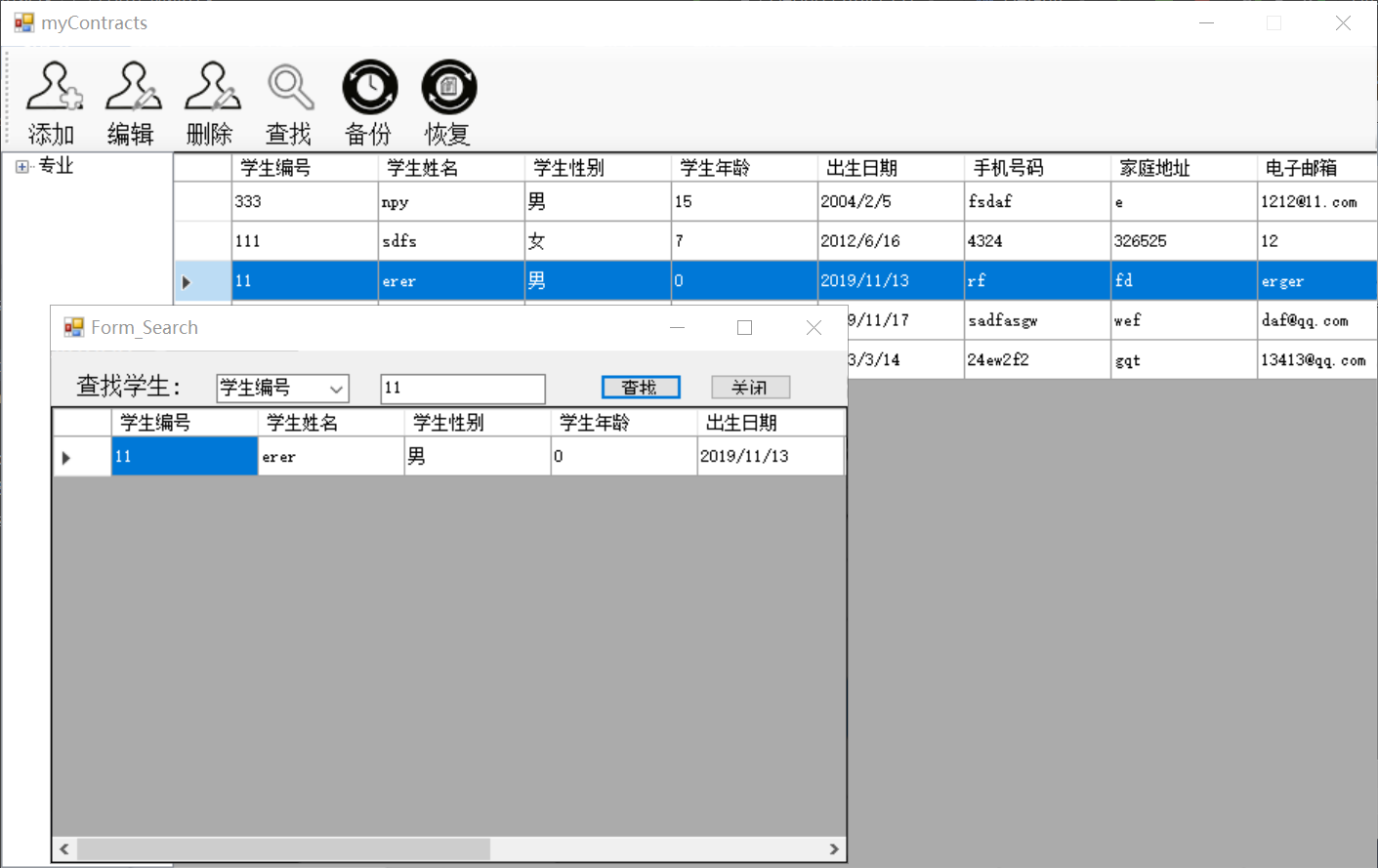
（2）查找功能调试（如图24所示）

图 24

（3）专业树功能调试（如图25所示）

图 25

4 系统扩展

（1）模糊搜索（如图26所示）

在查找时添加模糊搜索功能，可通过输入关键字获取相似信息。

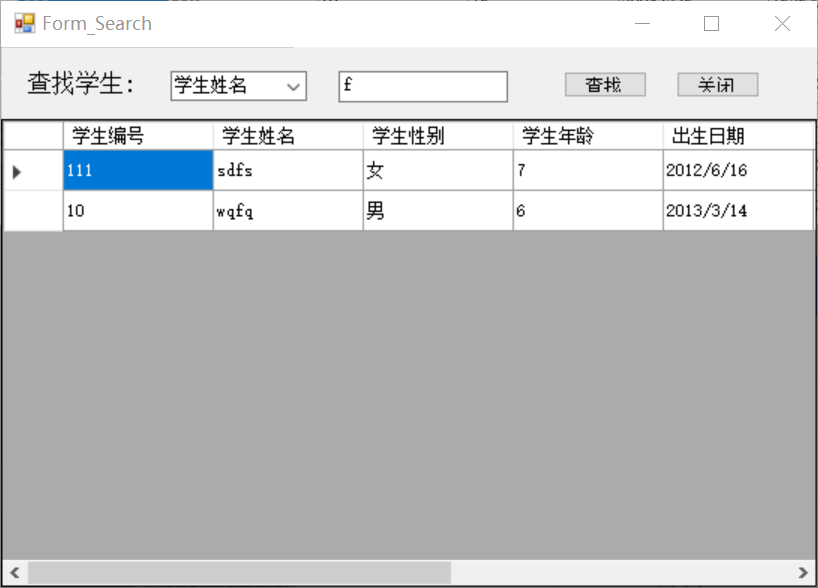
 模糊搜索关键代码：

图 26

public static List<StudentInfo>GetStudentInfoList (StudentInfo param)

{

List<StudentInfo> studentList = new List<StudentInfo> ();

XElement xml = XElement.Load(\_basePath);

Regex r = new Regex(".\*"+ param.Name +".\*");

var studentVar = xml.Descendants("student");

if (param.StudentId != 0)

{

studentVar = xml.Descendants("student").Where(a => a.Attribute("studentid").Value == param.StudentId.ToString());

}

else if (!String.IsNullOrEmpty(param.Name))

{

/// studentVar = xml.Descendants("student").Where(a => a.Element("name").Value == param.Name);

studentVar = xml.Descendants("student").Where(a => r.IsMatch(a.Element("name").Value));

}

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,

HomeAddress = student.Element("homeaddress").Value,

Email = student.Element("email").Value,

Profession = student.Element("profession").Value

}).ToList();

return studentList;

}

（2）增加“备份”、“恢复”按钮（如图27所示）

单击“备份”按钮时，把保存学生信息的Students.xml（名字可以自定义）文件保存到其他位置（用户可以定义保存的位置）；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。

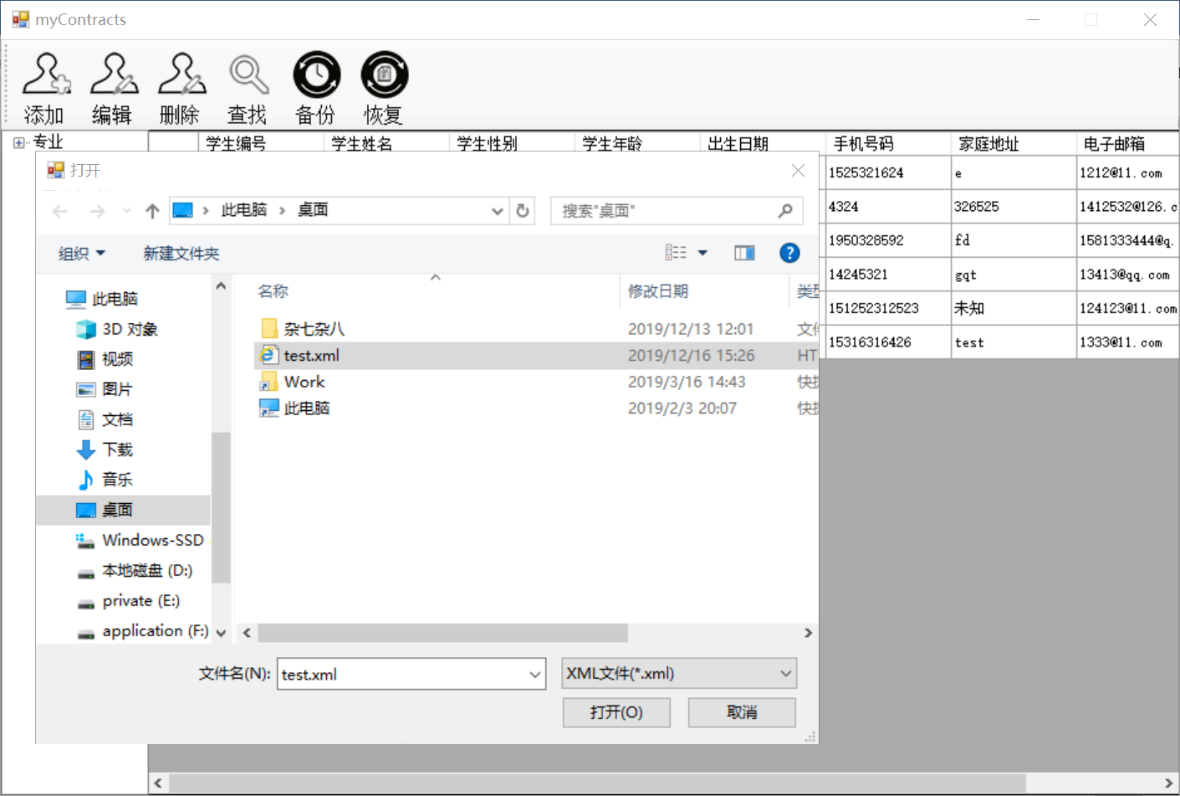
根据数据库的原理，若为脚本插入数据，如果一条条恢复将极为繁忙，若只能保存一次则使用性太低，因而采取了保存本地，这样可以保存多个版本。

图 27

（2）正则判断：

为降低使用时的输入错误情况，在添加及编辑过程中，添加判断邮箱和id 唯一且为数字的格式判断，通过正则表达式实现，以及各种判空情况。

在保障功能的情况下提高项目的使用性并提高代码的可读性及简化，使用正则表达式。

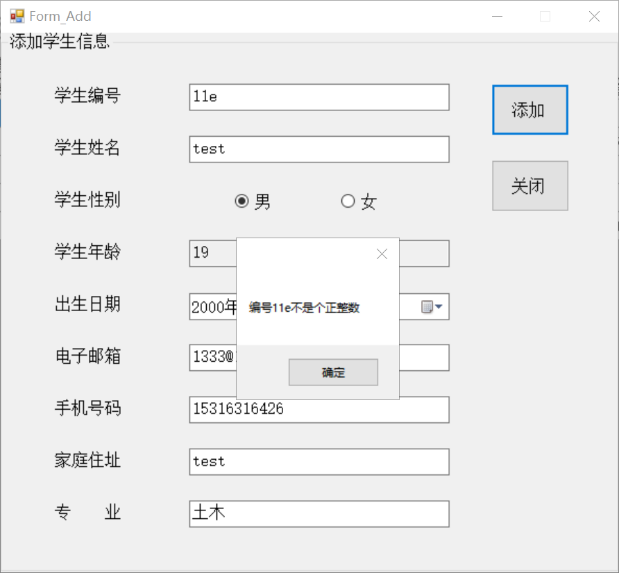
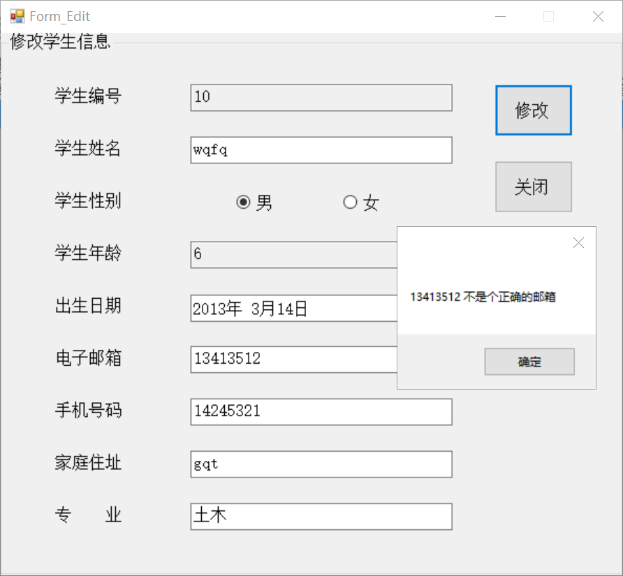
正则判断调试（如图28、29所示）

图 28

图 29

正则判断实现代码：

Regex r = new Regex("^([a-zA-Z]|[0-9])(\\w|\\-)+@[a-zA-Z0-9]+\\.([a-zA-Z]{2,4})$");

if (!r.IsMatch(txt\_email.Text.ToString()))

{

MessageBox.Show(txt\_email.Text.ToString() + " 不是个正确的邮箱");

return;

}

Regex id = new Regex("^\\+?[1-9][0-9]\*$");

if (!id.IsMatch(txt\_studengid.Text))

{

MessageBox.Show("编号"+ txt\_studengid.Text +"不是个正整数");

return;

}

5 总结

通过本项目，让我更进一步练习了C# 的操作，作为本身已经有了项目经验的我来说，这个项目很简单，如果要加入数据库操作的话，不建议是应用程序，最好是web端或者移动端，可以用java或者python的相关框架完成。

**实验四 拼图游戏**

1 系统概述

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矩阵或其他类型矩阵）并随机加载到图片框矩阵中，用户使用鼠标拖动图片框中的图片进行拼图，系统能够自动判别拼图是否成功并进行提示。

2 系统设计

2.1 设计目标

设计开发一个支持鼠标拖动拼图的游戏软件myPuzzle，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。

主要功能：

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中随时查看原图。

随机切换图片：能在软件自身携带的图片中进行随即切换，也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：选择相应图片框中图片后，按下鼠标左键进行拖动，放置到适当位置即可，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（如图30所示）

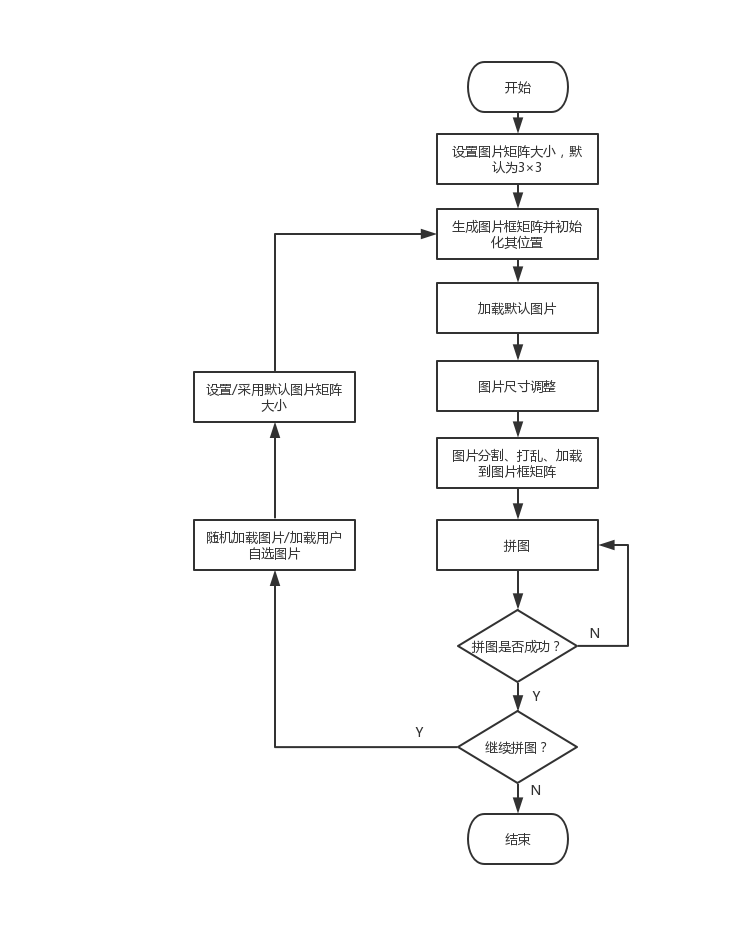


图 30

2.3 界面设计

主页面设计（如图31所示）

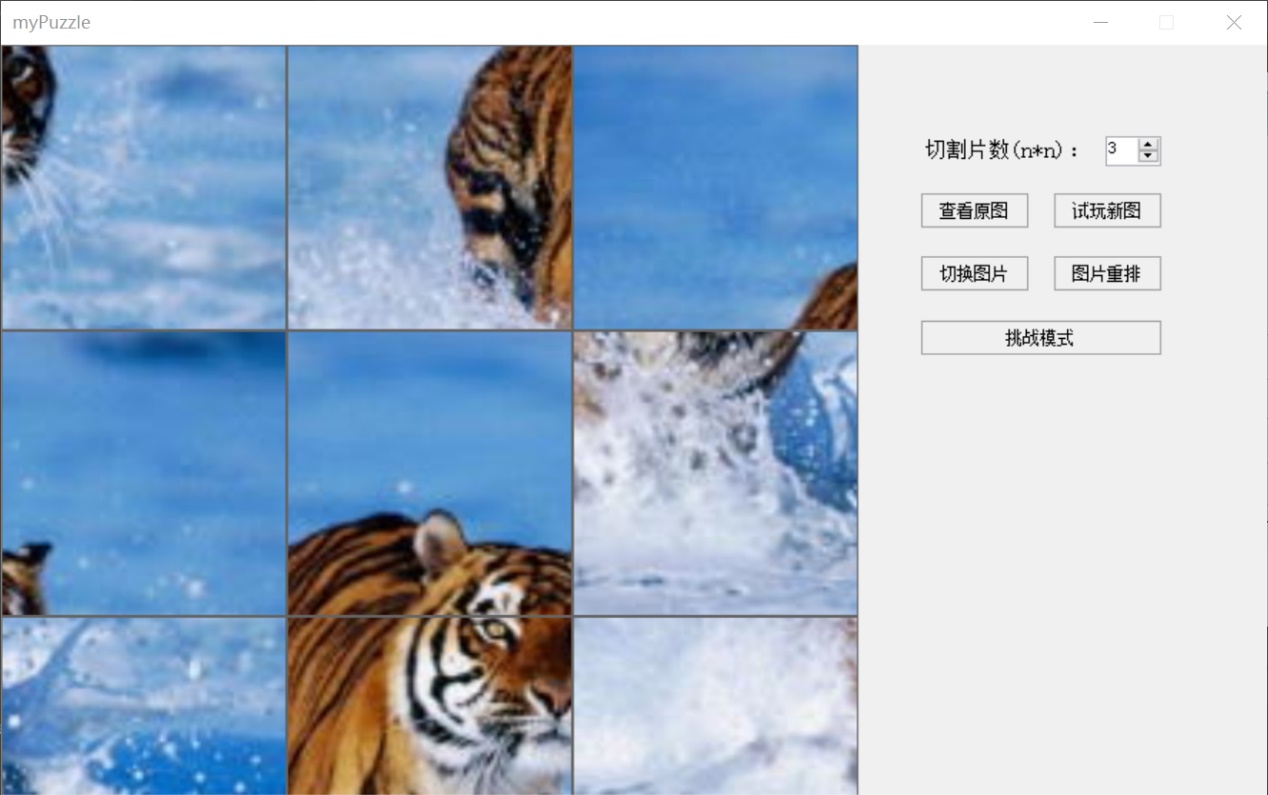


图 31

查看原图设计（如图32所示）

图 32

2.4 关键类图（如图33所示）

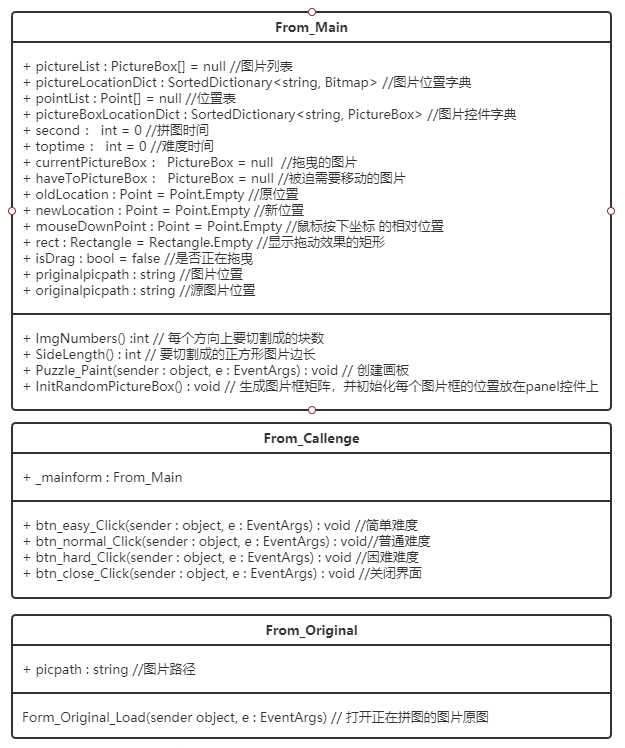


图 33

3 系统实现（运行调试）

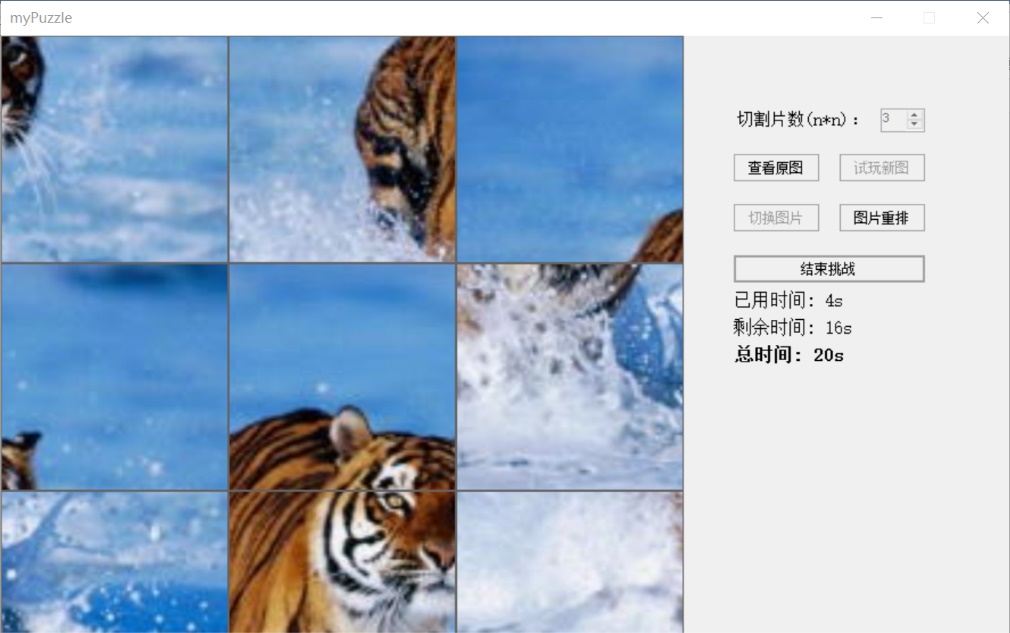
游戏中调试（如图34所示）

图 34

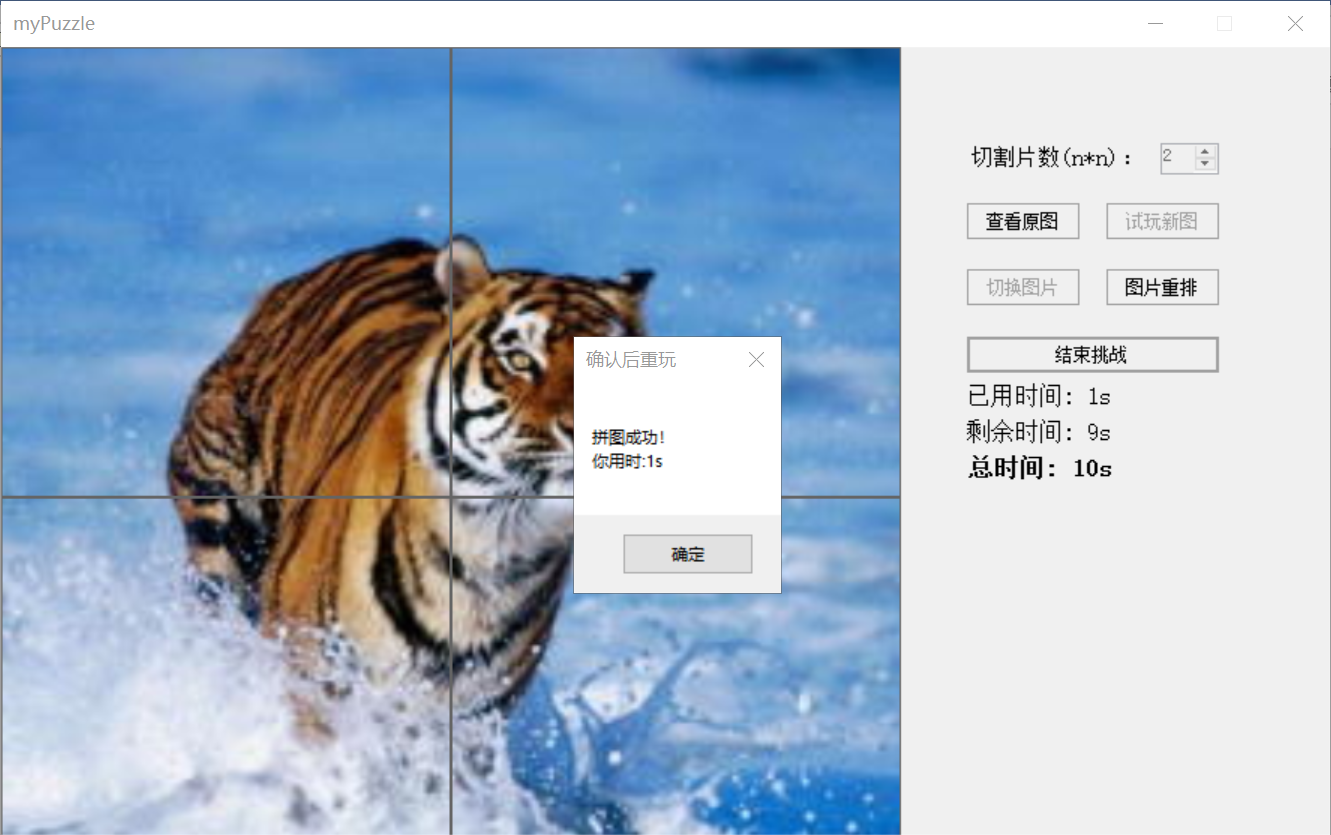
游戏中调试（如图35所示）

图 35

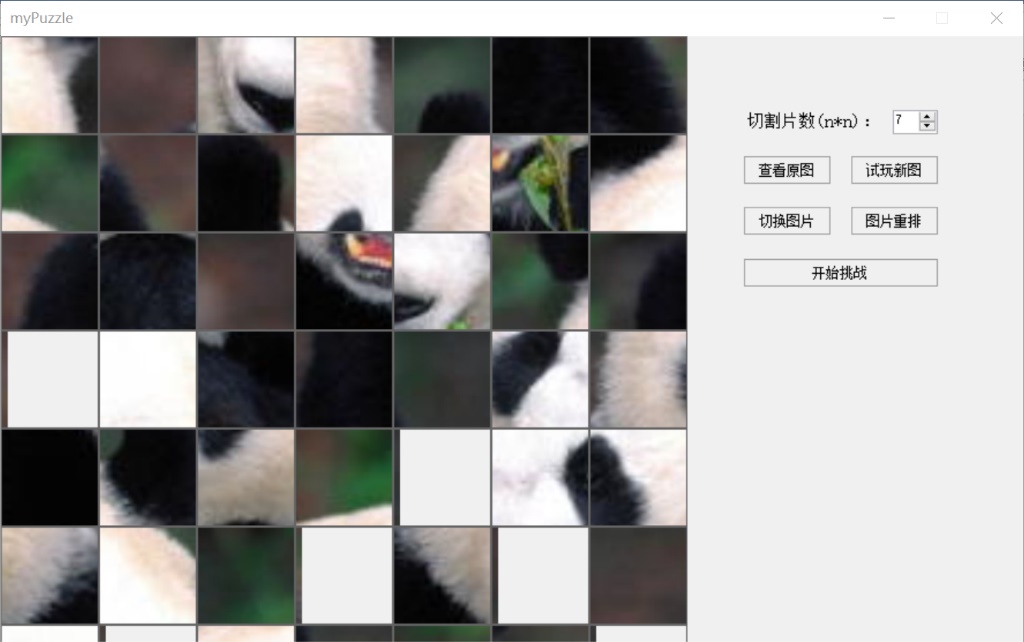
游戏中调试（如图36所示）

图 36

1. 系统扩展

（1）扩展拼图游戏myPuzzle的功能，增加把图片动态切割成除3\*3矩阵外的其他矩阵的功能，据此把拼图游戏的难易程度分成容易，中等，困难三个级别。（如图37所示）

图 37

（2）要根据矩阵的大小，动态生成图片框矩阵，并初始化图片框矩阵中每个图片框的位置。并根据图片框的大小把图片切割成大小相同的图片，实际上调用CutPicture类中的Cut方法即可。

（3）在游戏期间禁止切换图片、试玩新图、改变切割片数，提高游戏合理性，通过设置相关组件的Enabled 为 false实现。

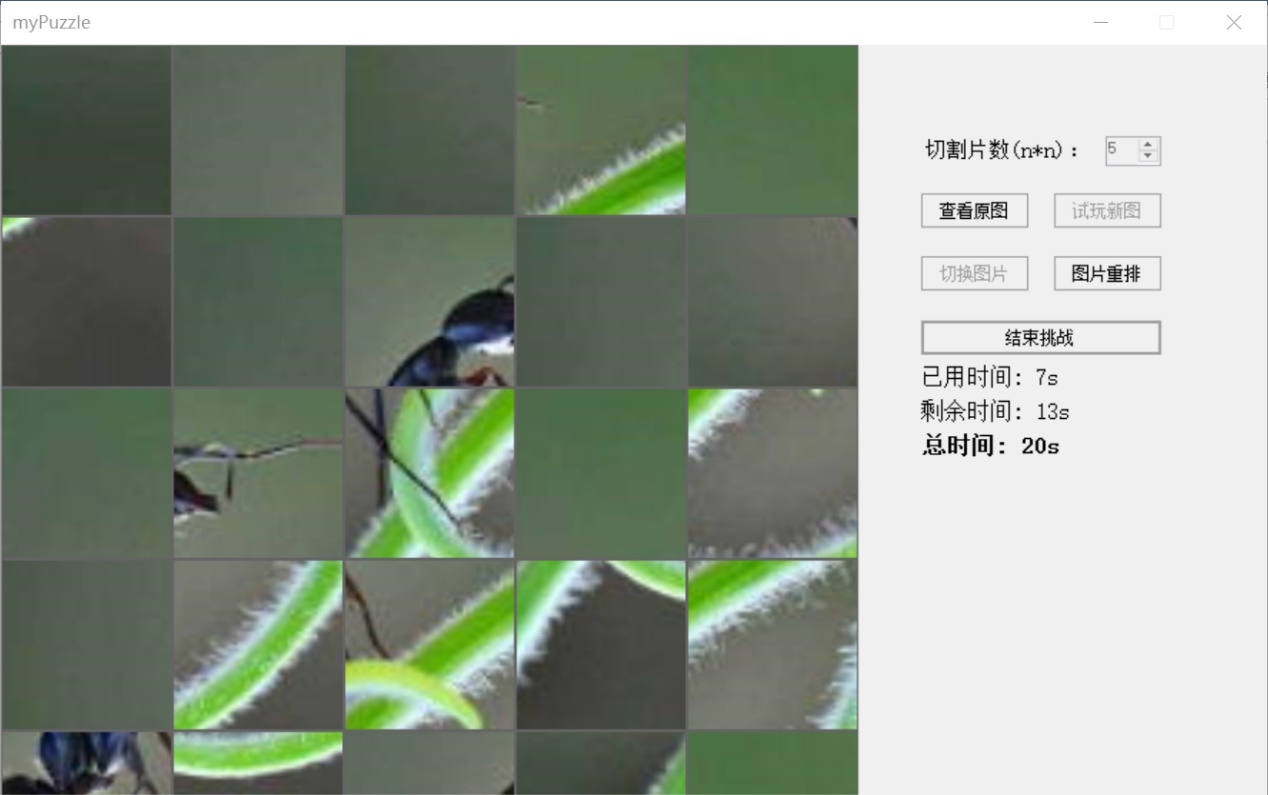
（2）（3）修改调试：（如图38所示）

图 38

禁用按钮的相关代码：

public void startGame()

{

timer\_usetime.Interval = 1000;

lab\_Usetime.Text = "已用时间: 0s";

lab\_AllTime.Text = "总时间: " + toptime + "s";

lab\_RestTime.Text = "剩余时间: " + toptime + "s";

second = 0;

timer\_usetime.Start();

btn\_Challenge.Click -= new EventHandler(btn\_Challenge\_Click);

btn\_Challenge.Click += new EventHandler(btn\_ChallengeStop\_Click);

btn\_import.Enabled = false;

btn\_Changepic.Enabled = false;

numericUpDown1.Enabled = false;

btn\_Challenge.Text = "结束挑战";

}

（4）解决了原代码中初始图片无法更换的bug

可以更新图片的相关代码：

if (!Directory.Exists(Application.StartupPath.ToString() + "\\Picture"))

{

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

}

Properties.Resources.\_0.Save(Application.StartupPath.ToString() + "\\Picture\\0.jpg");

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

Properties.Resources.\_1.Save(Application.StartupPath.ToString() + "\\Picture\\2.jpg");

5 总结

本项目中的难度对我来说主要出在了编译器上，因为Rider的Resource界面没有可视化界面，因而出现了图片的导入问题，后又发现书上代码存在逻辑问题，在保存图片的过程中判断了文件夹是否存在，如若存在，新加入的图片则无法保存，将if语句中保存图片的内容移至if以外，方可修改图片。