**数据结构课程设计**

**题 目: 迷宫游戏**

**班 级：**

**学 生 姓 名：**

**学 生 学 号：**

**2022 年 3月 6日**

目录

[1. 开发环境 3](#_Toc477208932)

[2. 程序的功能 3](#_Toc477208933)

[3. 代码清单 3](#_Toc477208934)

[4. 运行界面 14](#_Toc477208935)

## 开发环境

VS2022； C# WPF ；.NET Framework 4.8

## 功能描述

1. 输入功能：迷宫最大10行10列，中间文本框输入不可通过点的行号列标，点击添加按钮将其设置到图中，每次修改一个点，同时左侧列表框实时预览已经添加的点，右侧四个文本框输入起点终点坐标，点击开始计算按钮，软件会计算出所有可行的路径，并显示在下方列表框当中，每行按{(x,y)…}的格式，给出一条路径经过的所有点，如果没有合适的路径，则不会显示任何内容。
2. 显示功能：列出所有可能的路径（按经过的点坐标表示）

## 代码清单

本方法是使用递归来完成的，虽然递归函数当中没有显式定义栈，但事实上应用程序在管理递归调用的时候，就是用调用栈的原理完成的。此外为了保存所有的路径，我们在外部定义了一个Stack<Point>栈用来动态添加删除路径点。

下面是具体代码：

实体类

|  |
| --- |
| domain\Point.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace 作业2\_迷宫游戏.domain  {  public class Point  {  public int X { get; set; }  public int Y { get; set; }  public Point() { }  public Point(int x, int y) { X = x; Y = y; }  }  } |

|  |
| --- |
| domain\Graph.cs |
| using System;  using System.Collections.Generic;  using System.Collections;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace 作业2\_迷宫游戏.domain  {  public class Graph  {  public static int MAX\_SIZE = 10;  public static int IS\_PATH = 0;  public static int IS\_WALL = 1;  public int[,] points { get; set; }  public bool[,] isPassed { get; set; }  public Point Start { get; set; }  public Point End { get; set; }  public Graph(List<Point> list,Point start,Point end)  {  points = new int[MAX\_SIZE, MAX\_SIZE];  isPassed = new bool[MAX\_SIZE, MAX\_SIZE];  foreach(Point p in list)  {  points[p.X, p.Y] = IS\_WALL;  }  this.Start = start;  this.End=end;  }  }  } |

dao层

|  |
| --- |
| Dao\GraphDao.cs |
| using System;  using System.Collections;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using 作业2\_迷宫游戏.domain;  namespace 作业2\_迷宫游戏.dao  {  public class GraphDao  {  private Stack<Point> stack;  public List<List<Point>> answers { get; set; }  public Graph graph { get; set; }  private void doSth(int x, int y)  {  if (x == graph.End.X && y == graph.End.Y)  {  stack.Push(new Point(x, y));  List<Point> ans=stack.ToList<Point>();  ans.Reverse();  answers.Add(ans);  stack.Pop();  return;  }  if (x>=0 && x < Graph.MAX\_SIZE && y>=0 && y < Graph.MAX\_SIZE && graph.points[x, y] == Graph.IS\_PATH && graph.isPassed[x, y] == false)  {  graph.isPassed[x, y] = true;  stack.Push(new Point(x, y));  doSth(x + 1, y);  doSth(x, y + 1);  doSth(x - 1, y);  doSth(x, y - 1);  stack.Pop();  graph.isPassed[x, y] = false;  }  }  public List<List<Point>> getAllPaths(List<Point> arrayList, Point start, Point end)  {  graph = new Graph(arrayList, start, end);  stack = new Stack<Point>();  answers = new List<List<Point>>();  doSth(graph.Start.X, graph.Start.Y);  return answers;  }  }  } |

Service层

|  |
| --- |
| service\GraphService.cs |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using 作业2\_迷宫游戏.dao;  using 作业2\_迷宫游戏.domain;  namespace 作业2\_迷宫游戏.service  {  public class GraphService  {  private GraphDao dao=new GraphDao();  public List<List<Point>> getAllPaths(List<Point> arrayList, Point start, Point end)  {  return dao.getAllPaths(arrayList, start, end);  }  }  } |

表示层（窗口XAML及对应C#代码）

|  |
| --- |
| MainWindow.xaml |
| <Window x:Class="作业2\_迷宫游戏.MainWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:local="clr-namespace:作业2\_迷宫游戏"  mc:Ignorable="d"  Title="迷宫游戏求解 - C#初体验" Height="457" Width="825">  <Grid>  <GroupBox HorizontalAlignment="Left" Height="177" Header="编辑迷宫点（设置不可经过位置）" Margin="246,10,0,0" VerticalAlignment="Top" Width="235">  <StackPanel HorizontalAlignment="Center" Height="142" Margin="27,0,0,0" VerticalAlignment="Center" Width="194">  <WrapPanel Height="52" Width="186">  <TextBlock TextWrapping="Wrap" Text="X坐标：" Height="52" Width="84" HorizontalAlignment="Center" VerticalAlignment="Center"/>  <TextBox x:Name="txbNewX" TextWrapping="Wrap" Text="0" Width="99" Height="32"/>  </WrapPanel>  <WrapPanel Height="48" Width="187">  <TextBlock TextWrapping="Wrap" Text="Y坐标：" Height="48" Width="85"/>  <TextBox x:Name="txbNewY" TextWrapping="Wrap" Text="0" Width="99" Height="32"/>  </WrapPanel>  <Button x:Name="btnInsert" Content="插入" Height="27" Width="118" Click="btnInsert\_Click"/>  </StackPanel>  </GroupBox>  <GroupBox HorizontalAlignment="Left" Height="238" Header="计算结果" Margin="10,187,0,0" VerticalAlignment="Top" Width="804">  <ListBox x:Name="lbAnswers" HorizontalAlignment="Left" Height="205" Margin="10,0,0,0" VerticalAlignment="Top" Width="776" />  </GroupBox>  <GroupBox HorizontalAlignment="Left" Height="177" Header="计算前预处理" Margin="500,10,0,0" VerticalAlignment="Top" Width="314">  <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center" Margin="10,0,0,5" Width="294">  <WrapPanel Height="46" Margin="5">  <TextBlock TextWrapping="Wrap" Text="起点X坐标：" Height="46"/>  <TextBox x:Name="txbStartX" TextWrapping="Wrap" Text="0" Width="67"/>  <TextBlock TextWrapping="Wrap" Text="起点Y坐标：" Width="71"/>  <TextBox x:Name="txbStartY" TextWrapping="Wrap" Text="0" Width="78"/>  </WrapPanel>  <WrapPanel Height="46" Margin="5">  <TextBlock TextWrapping="Wrap" Text="终点X坐标：" Height="47" Width="68"/>  <TextBox x:Name="txbEndX" TextWrapping="Wrap" Text="0" Width="66"/>  <TextBlock TextWrapping="Wrap" Text="终点Y坐标：" Width="72"/>  <TextBox x:Name="txbEndY" TextWrapping="Wrap" Text="0" Width="78"/>  </WrapPanel>  <Button x:Name="btnOk" Content="设置好了，立即计算" Height="27" Width="194" Margin="5" Click="btnOk\_Click"/>  </StackPanel>  </GroupBox>  <ListBox x:Name="lbPoints" HorizontalAlignment="Left" Height="155" Margin="10,27,0,0" VerticalAlignment="Top" Width="215" />  <TextBlock HorizontalAlignment="Left" Height="24" Margin="10,10,0,0" TextWrapping="Wrap" Text="当前已添加点列表：" VerticalAlignment="Top" Width="215"/>  </Grid>  </Window> |

|  |
| --- |
| MainWindow.xaml.cs |
| using System;  using System.Collections;  using System.Collections.Generic;  using System.ComponentModel;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Data;  using System.Windows.Documents;  using System.Windows.Input;  using System.Windows.Media;  using System.Windows.Media.Imaging;  using System.Windows.Navigation;  using System.Windows.Shapes;  using 作业2\_迷宫游戏.domain;  using 作业2\_迷宫游戏.service;  namespace 作业2\_迷宫游戏  {  /// <summary>  /// MainWindow.xaml 的交互逻辑  /// </summary>  public partial class MainWindow : Window  {  List<domain.Point> list = new List<domain.Point>();  //考虑欠佳，自己起的类名和SDK类名重复。。。  public MainWindow()  {  InitializeComponent();  }  private void btnInsert\_Click(object sender, RoutedEventArgs e)  {  int nx = Int32.Parse(txbNewX.Text);  int ny = Int32.Parse(txbNewY.Text);  lbPoints.Items.Add("X: " + nx + "\tY: " + ny);  domain.Point np =new domain.Point(nx,ny);  list.Add(np);  }  private void btnOk\_Click(object sender, RoutedEventArgs e)  {  GraphService service = new GraphService();  int startX = Int32.Parse(txbStartX.Text);  int startY = Int32.Parse(txbStartY.Text);  domain.Point start =new domain.Point(startX,startY);  int endX = Int32.Parse(txbEndX.Text);  int endY = Int32.Parse(txbEndY.Text);  domain.Point end = new domain.Point(endX, endY);  //清除上一次输入  lbPoints.Items.Clear();  //清除上一次结果  lbAnswers.Items.Clear();  var answers = service.getAllPaths(list, start, end);  foreach(var answer in answers)  {  StringBuilder sb = new StringBuilder();  sb.Append("{ ");  foreach(var pt in answer)  {  int ax = pt.X;  int ay = pt.Y;  sb.Append("(" + ax + "," + ay + ")");  }  sb.Append(" }");  lbAnswers.Items.Add(sb.ToString());  }  }  }  } |

|  |
| --- |
| App.xaml App.xaml.cs App.config |
| 未修改，均为vs2022生成的默认方案 |

## 运行界面

软件启动后显示主窗口

图形用户界面

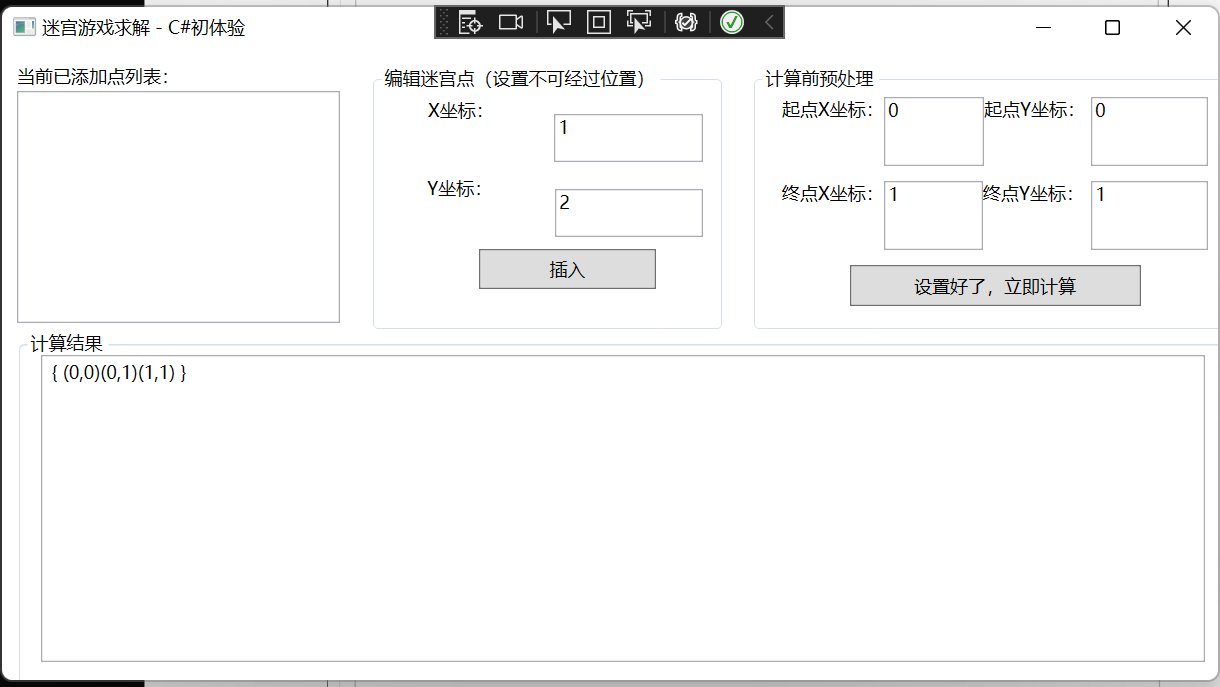
描述已自动生成

中间输入一些点：（最大支持10\*10迷宫）

图形用户界面, 应用程序

描述已自动生成

点击计算按钮，给出结果



就是。。。输入有点累手。。。

输入样例1（以图的形式表示）

0 1 1

0 0 1

1 1 1

0 0

1 1

输出样例1：

{(0,0)(0,1)(1,1)}

输入样例2：

0 0 0 1

0 1 0 1

0 0 0 1

1 1 1 1

0 0

2 2

输出样例2

{(0,0)(0,1)(0,2)(1,2)(2,2)}

{(0,0)(1,0)(2,0)(2,1)(2,2)}