**南 京 师 范 大 学**

**《人机交互》**

**实验报告**



**题 目： 数独游戏**

**学 院： 计算机科学与技术学院**

**专 业： 软件工程**

**姓 名： 钱迪琦**

**学 号： 19160229**

**任课教师： 王琼**

**计算机科学与技术学院 制**

**二○一八 年 十二 月 十三 日**

目录

[1 实验要求 1](#_Toc532674406)

[2 实验环境 1](#_Toc532674407)

[3 程序分析 1](#_Toc532674408)

[4 实现技术 2](#_Toc532674409)

[4.1 生成初始数独 2](#_Toc532674410)

[4.2 数独求解算法 2](#_Toc532674411)

[4.3 基本游戏界面 3](#_Toc532674412)

[4.3.1 显示数独 3](#_Toc532674413)

[4.3.2 背景 4](#_Toc532674414)

[4.3.3 点击事件 5](#_Toc532674415)

[4.4 撤销重做功能 6](#_Toc532674416)

[5 运行示例 7](#_Toc532674417)

[6 后续工作 10](#_Toc532674418)

[7 收获与体会 10](#_Toc532674419)

# 1 实验要求

利用VB.NET实现一个简单完整的数独游戏，要求能够根据现有数独随机生成待填数独，且具有提示、回退和重做功能。

# 2 实验环境

操作系统：Windows 10

实验平台：Visual Studio 2017

语言环境：VB.NET

# 3 程序分析

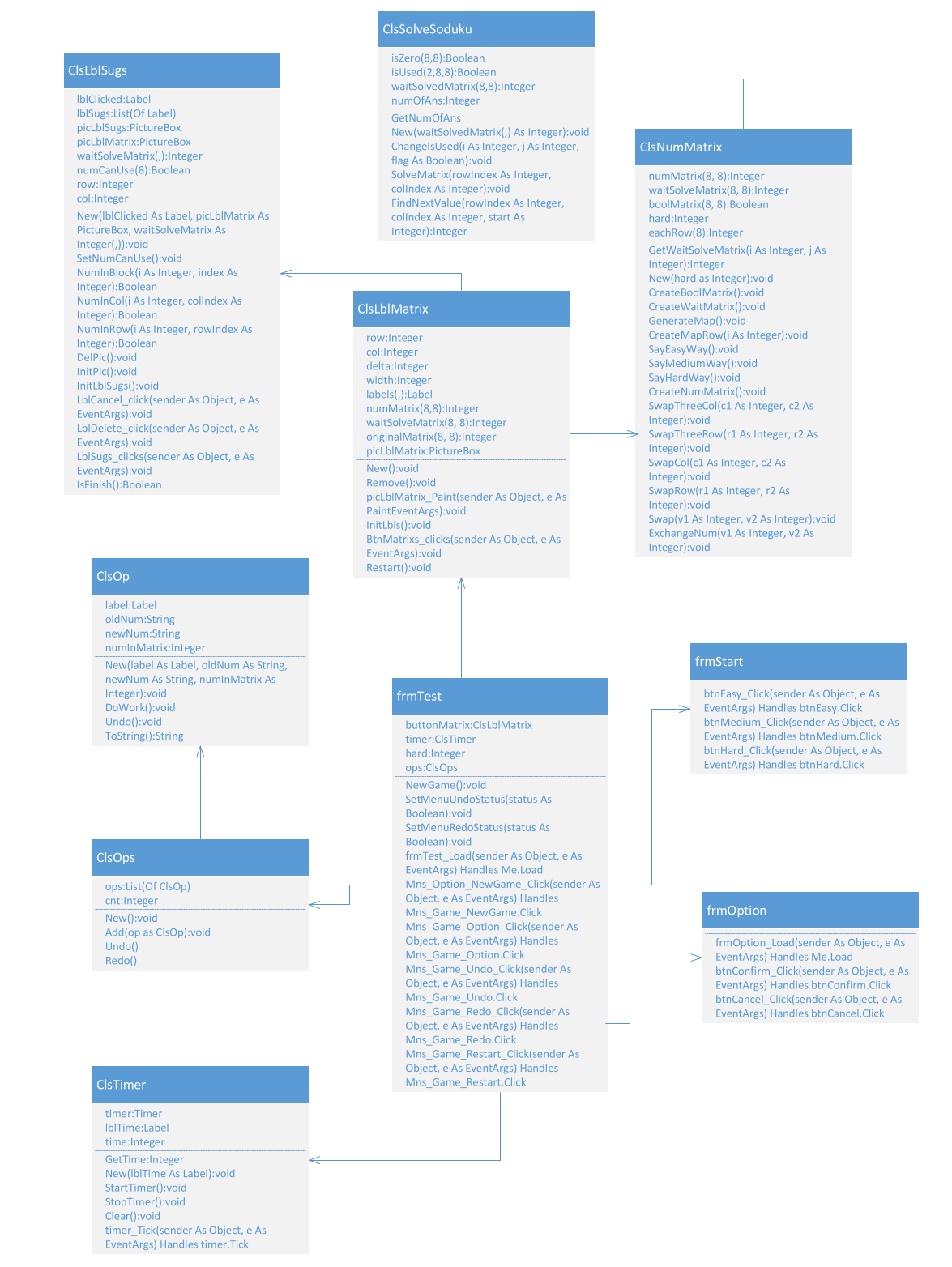


图3.1

本程序的类图如图3.1，其中ClsNumMatrix用于生成一个新的数独矩阵，ClsSolveSoduku用于求解数独，ClsLblMatrix用于绘制数独，ClsLblSugs用于显示数独的提示和填写数独，ClsOps用于管理ClsOp，以实现撤销重做功能，ClsTimer用于游戏计时。frmStart为程序启动时的难度选择窗口，frmTest为程序主窗口，frmOption为选项窗口。

# 4 实现技术

## 4.1 生成初始数独

随机寻找一个现有完整数独作为程序的初始数独。

将这个初始数独进行随机的行、列交换（单行或单列之间交换时必须为一个完整的宫所在的某三行或三列之中的两行或两列）。经过多次随机打乱后，将获得一个新的随机数独矩阵，且此矩阵满足数独的所有特性。

根据不同的难度等级随机从新的数独矩阵中抠出不同个数的元素，并通过算法求解此数独，确保此数独只具有唯一解。这个具有唯一解的数独即为此次游戏的待求解数独。

## 4.2 数独求解算法

通过深度遍历的方式递归求解数独的解。

若以数独的左上角第一个宫编号为0，向右向下依次为1，2，3……则数独的第i行第j列个元素与其所在的宫的标号index有如下关系：index=3 \* (i / 3) + (j / 3)。

维护两个多维数组，二维数组isZero(8,8)表示数独的第i行第j个位置是否待求解，true代表待求解，false代表不需要求解，此位置已经有值。三位数组isUsed(2,8,8)，其中isUsed(0,i,j)表示在i行是否已经存在数字j+1，true代表有，false代表没有；isUsed(1,i,j)表示在i列是否已经存在数字j+1，isUsed(2,i,j)表示在第i个九宫格是否已经存在数字j+1。

算法的描述如下：

①根据待求解数独，初始化isZero数组和isUsed数组。

②从待解数独的i行j列开始往后遍历，寻找第一个待求解（即isZero(i,j)=0）的位置i，j。

③根据isUsed矩阵挑选i行j列的可能解k，即同时满足isUsed(0,i,k)= false，isUsed(1,j,k)=false，isUsed(2,index,k)=false。每次从上次已选的可能解的后一个往后挑选，确保不会重复。

④如果找到了一个可能解，则重复②③，直到所有待求解位置均求解完毕，说明找到了该数独的一个解。如果本次没有挑选出合适的可能解，则说明之前的求解失败，返回上一级继续求解。

## 4.3 基本游戏界面

### 4.3.1 显示数独

每一个数独的元素通过一个标签元素显示，将整个数组标签集合通过picturebox.Controls.Add(label)方法添加到一个pictureBox框之中，方便对于其的管理。

ReDim labels(row - 1, col - 1)

For i = 0 To row - 1

For j = 0 To col - 1

Dim label As Label

If numMatrix(i, j) <> 0 Then

label = New Label With {

.Name = "Label " & i & " " & j,

.Size = New System.Drawing.Size(width, width),

.Location = New System.Drawing.Point(delta + j \* (width), delta + i \* (width)),

.TextAlign = ContentAlignment.MiddleCenter,

.BackColor = Color.Transparent,

.Parent = picLblMatrix,'让标签的父标签为picLblMatrix,设置标签backColor为Transparent实现（假）透明效果

.BorderStyle = BorderStyle.FixedSingle

}

label.Text = numMatrix(i, j).ToString

Else

label = New Label With {

.Name = "Label " & i & " " & j,

.Size = New System.Drawing.Size(width, width),

.Location = New System.Drawing.Point(delta + j \* (width), delta + i \* (width)),

.Cursor = Cursors.IBeam,

.TextAlign = ContentAlignment.MiddleCenter,

.BackColor = Color.Transparent,

.Parent = picLblMatrix,'让标签的父标签为picLblMatrix,设置标签backColor为Transparent实现（假）透明效果

.BorderStyle = BorderStyle.FixedSingle

}

AddHandler label.Click, AddressOf BtnMatrixs\_clicks

End If

labels(i, j) = label

Next

Next

For i = 0 To row - 1

For j = 0 To row - 1

picLblMatrix.Controls.Add(labels(i, j))

Next

Next

### 4.3.2 背景

在4.3.1介绍到所有的标签元素均放在了picturebox框中，所以我们可以通过直接在picturebox中绘制图形的方式来给游戏添加背景，只需要将picturebox框中的label背景色设为Transparent即可正确的显示背景。

Private Sub picLblMatrix\_Paint(sender As Object, e As PaintEventArgs)

Dim g As Graphics = e.Graphics

Dim pen As Pen = New Pen(Color.Black, 3)

g.Clear(Color.White)

'画初始有数字的格子的背景色

For i = 0 To 8

For j = 0 To 8

If numMatrix(i, j) <> 0 Then

Dim p As Point = New Point(delta + j \* (width), delta + i \* (width))

Dim grayBrush As SolidBrush = New SolidBrush(Color.FromArgb(200, 200, 200))

g.FillRectangle(grayBrush, p.X, p.Y, width, width)

End If

Next

Next

'画所有的横线

For i = 0 To 3

Dim sp As Point = New Point(delta, delta + i \* (width \* 3))

Dim ep As Point = New Point(delta + 9 \* width, delta + i \* (width \* 3))

g.DrawLine(pen, sp, ep)

Next

'画所有的竖线

For i = 0 To 3

Dim sp As Point = New Point(delta + i \* (3 \* width), delta)

Dim ep As Point = New Point(delta + i \* (3 \* width), delta + 9 \* width)

g.DrawLine(pen, sp, ep)

Next

End Sub

### 4.3.3 点击事件

通过AddHandler方法给动态生成的标签矩阵添加点击事件来给数独的空白位置填写数字，要求游戏具有提示的功能，所以在点击后创建一个picturebox框用来显示可选择填写的数字。

这里我们使用了BringToFront方法，这个方法可以将父控件中的某一子控件提升至父控件的最上方，从而达到一种悬浮的效果。

Private Sub InitPic()

picLblSugs.Name = "picLblSugs"

picLblSugs.Size = New System.Drawing.Size(3 \* width, 4 \* height)

picLblSugs.BorderStyle = BorderStyle.FixedSingle

'picLblSugs.Parent = lblClicked

'picLblSugs.BackColor = Color.FromArgb(0, 255, 255, 255)

Dim basePoint As Point = New Point(lblClicked.Location.X + Fix(lblClicked.Width / 2), lblClicked.Location.Y + Fix(lblClicked.Height / 2))

'picLblSugs.Location = New System.Drawing.Point(0, 0)

If row < 4 Then

If col < 4 Then

picLblSugs.Location = New System.Drawing.Point(basePoint)

Else

picLblSugs.Location = New System.Drawing.Point(basePoint.X - picLblSugs.Width, basePoint.Y)

End If

Else

If col < 4 Then

picLblSugs.Location = New System.Drawing.Point(basePoint.X, basePoint.Y - picLblSugs.Height)

Else

picLblSugs.Location = New System.Drawing.Point(basePoint.X - picLblSugs.Width, basePoint.Y - picLblSugs.Height)

End If

End If

picLblMatrix.Controls.Add(picLblSugs)

picLblSugs.BringToFront()

End Sub

在建好的picturebox框上再利用类似于4.3.1中的方式显示提示框，可填写的数字以黑色显示，可通过点击它填写对应的数字；不可填写的数字以灰色显示，且不可点击。具体代码这里不再给出。

## 4.4 撤销重做功能

定义ClsOp类用于记录一次操作，其拥有label属性用于记录发生操作的标签，oldNum和newNum分别用来记录操作前的数字和操作后的数字。DoWork()和Undo()方法分别用来记录本次操作和撤销本次操作。

另有ClsOps类维护一个操作列表，和一个指针。每发生一次操作则将本次操作添加入操作列表，并将指针指向当前的位置。

撤销操作即调用指针所指位置操作的Undo函数，并将指针指向前一个操作；重做操作即调用指针所指位置操作的DoWork()函数，并将指针指向后一个操作。

Public Class ClsOp

Private label As Label

Private oldNum As String

Private newNum As String

Private numInMatrix As Integer

Public Sub New(label As Label, oldNum As String, newNum As String, numInMatrix As Integer)

Me.label = label

Me.oldNum = oldNum

Me.newNum = newNum

Me.numInMatrix = numInMatrix

End Sub

Public Sub DoWork()

If newNum = "" Then

label.Text = ""

numInMatrix = 0

Else

label.Text = newNum

numInMatrix = newNum

End If

End Sub

Public Sub Undo()

If oldNum = "" Then

label.Text = ""

numInMatrix = 0

Else

label.Text = oldNum

numInMatrix = oldNum

End If

End Sub

Public Overrides Function ToString() As String

Dim str As String = label.Name & " oldNum: " & oldNum & " newNum:" & newNum

Return str

End Function

End Class

Public Class ClsOps

Private ops As List(Of ClsOp)

Private cnt As Integer

Public Sub New()

ops = New List(Of ClsOp)

cnt = 0

End Sub

Public Sub Add(op As ClsOp)

ops.Add(op)

cnt = ops.Count

frmTest.SetMenuUndoStatus(True)

frmTest.SetMenuRedoStatus(False)

End Sub

Public Sub Undo()

'If cnt = 0 Then Return

ops(cnt - 1).Undo()

cnt -= 1

frmTest.SetMenuRedoStatus(True)

If cnt = 0 Then frmTest.SetMenuUndoStatus(False)

End Sub

Public Sub Redo()

'If cnt = ops.Count Then Return

ops(cnt).DoWork()

cnt += 1

frmTest.SetMenuUndoStatus(True)

If cnt = ops.Count Then frmTest.SetMenuRedoStatus(False)

End Sub

End Class

# 5 运行示例

程序启动后选择难度如图5.1：

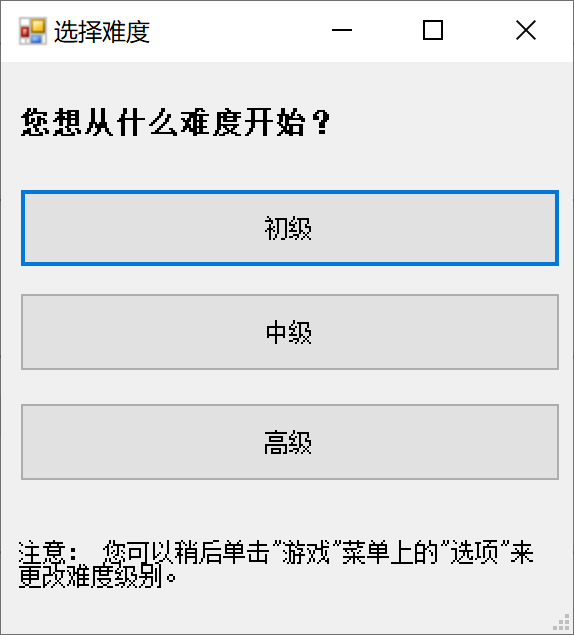


图5.1

选择难度后主界面如图5.2：

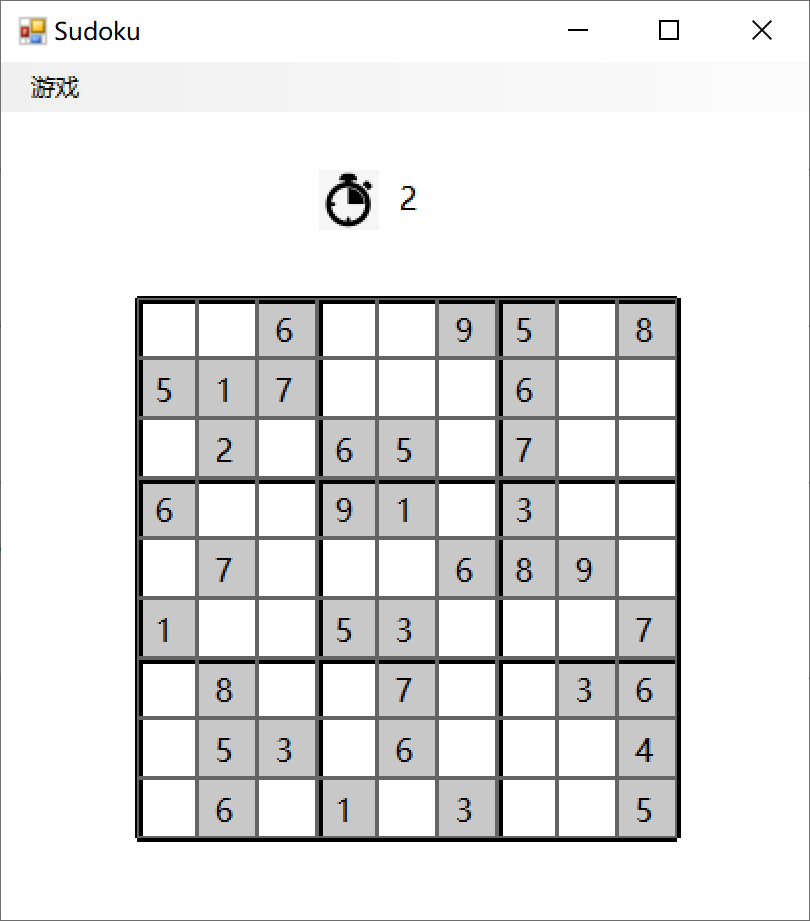


图5.2

点击空白位置填写数字，如图5.3：

菜单选项如图5.4：

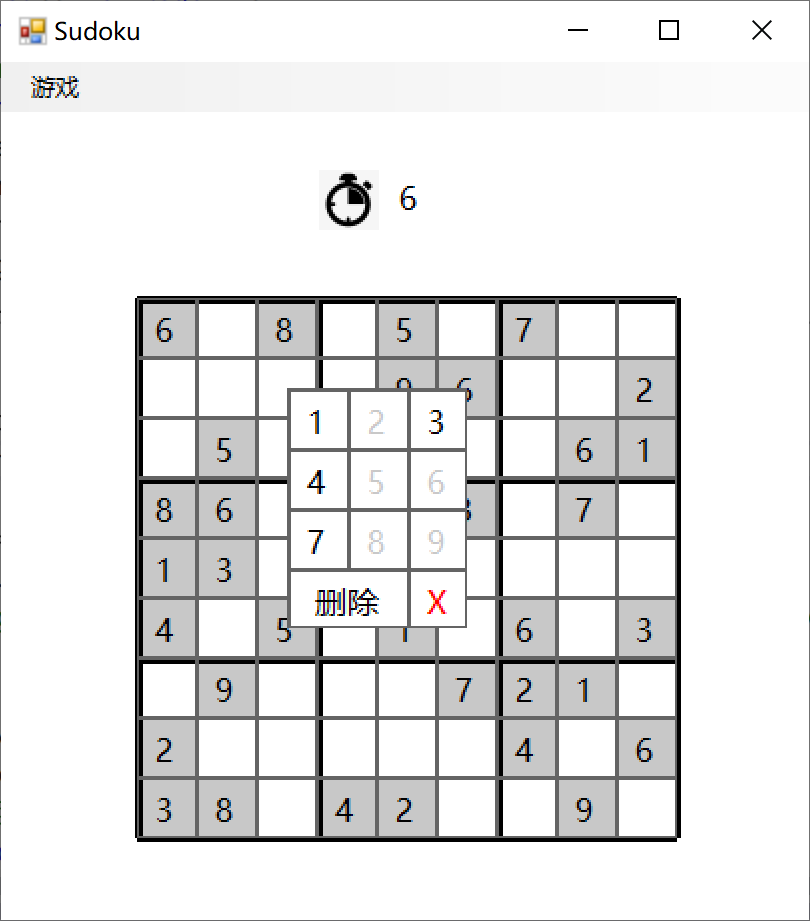


图5.3

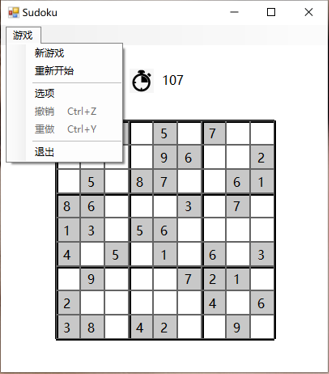


图5.4

选项窗口如图5.5：

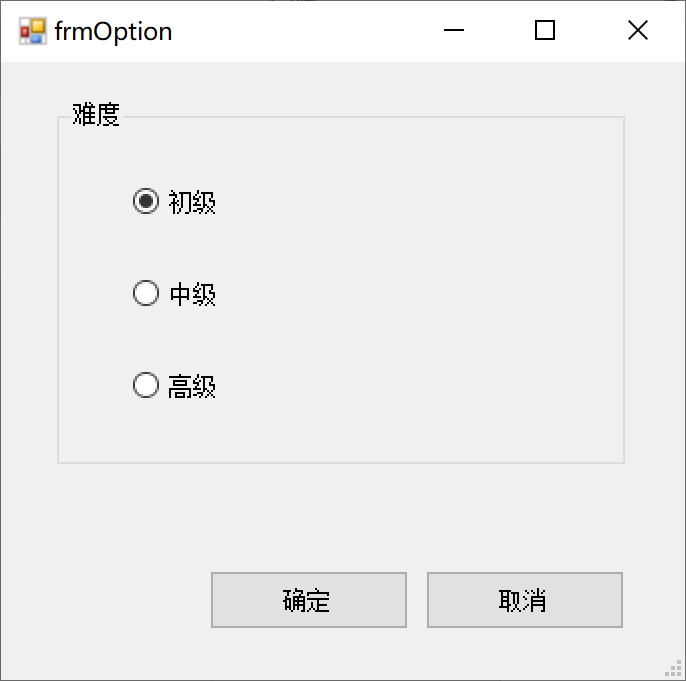


图5.5

成功完成后弹出提示对话框如图5.6：



图5.6

# 6 后续工作

此程序已经基本完成了既定的所有要求，但是仍然有一些可能的方向继续扩展。

①建立数独矩阵库：本程序利用一个现有数独进行变换而得到了一个随机的初始数独，这样得到的数独随机性太大，不利于准确的进行等级划分。且为了要求数独只有唯一解，程序先进行了数次的解数独操作，不利于生成难度过大的数独。所以后续可以建立一个数独库，将不同的数独按更有效的方式分难度放入库中，这样程序也可以设置成闯关的模式。

②提示窗口的半透明：本程序利用了BringToFront的方法来显示提示框，透明效果并不好，在图5.3中可看出提示框会遮挡住下面的数字，不利于用户的观察。通过和老师的沟通我觉得可以使用一个新的窗口的方式来显示提示窗口，这样可以设置半透明化，从而不遮挡下面的数字。

③增加记忆功能，程序现在并不具有保存，下次继续修改的功能，这也是后续可以添加的一个功能，即将现在的数独状态和用户的填写状态全部保存下来，下次用户再次打开可以继续上次的进度

# 7 收获与体会

本次的项目是我第一次使用VB.NET平台进行开发，这次实践给了我一个绝好的学习VB.NET语言的机会，这个项目结束，使我能够加深对于VB.NET的理解。

这次项目开发也是我第一次利用图形化编程去完成一个完整的项目，这帮助我将之前零散的图形化界面开发知识能够进行一个整合，让我熟悉并逐渐习惯图形化编程的方式。

项目实现过程中也使用了很多数据结构的知识，之前对于数据结构的学习多是局限于题目和算法实现之上，而这次的实验给了我一次将数据结构与真正的项目相结合的经历，帮助我更好理解数据结构的知识。