第二组、数独软件设计

目录

1. 数独游戏简介
2. 设计思想
3. 数独软件生命思想设计
4. 问题定义
5. 可行性研究
6. 需求分析
7. 总体设计
8. 详细分析
9. 编码和单元测试
10. 综合测试
11. 软件维护
12. 数独游戏简介

数独盘面是个九宫，每一宫又分为九个小格。在这八十一格中给出一定的已知数字和解题条件，利用逻辑和推理，在其他的空格上填入1-9的数字。使1-9每个数字在每一行、每一列和每一宫中都只出现一次，所以又称“九宫格”。

1. 设计思想

我们只需要遍历每个个格子，排除不可填的数字之后，随机给它填入一个数字，如果发现当前格子没有可填的值，则退回去上一个格子，将此格子原有的值排除后再重新选一个新的值填入，如果发现也没有合适的值，再退一个格子....在不停的重复这个过程之后，就可以得到一个正确的数独矩阵，这里用到了深度优先算法和回溯算法。

三、数独软件生命周期设计

1. 问题定义：要解决什么问题

设计出数独游戏软件。

1. 可行性研究：探索是否值得去解决

人物力可行，值得解决。

1. 需求分析：确定目标系统必须具备哪些功能

实现可视化、生成残缺的数独、可填入数字、判断输入是否符合数独矩阵、重新填入功能

1. 总体设计：概要设计，如何实现目标系统

实现流程图：



方法设计：生成数独方法、判断矩阵是否符合数独矩阵的方法



1. 详细设计：把总体设计的抽象化，类似工程蓝图



1. 编码和单元测试：

6.1生成数独矩阵方法createFun.py编码设计，实现编码：

import random

import math

matrix = []

# 生成一个随机的数组

def get\_random\_unit():

\_num\_list = [1, 2, 3, 4, 5, 6, 7, 8, 9]

random.shuffle(\_num\_list)#随机打乱序列顺序

return \_num\_list

#打印出矩阵arr中的每一列

def print\_grid(arr):

for i in range(9):

print(arr[i])

#得到这行里已经用过的数

def get\_row(row):

row\_arr = []

for v in matrix[row]:

if v == 0:

continue

row\_arr.append(v)

return row\_arr

#得到这列里已经用过的数

def get\_col(col):

col\_arr = []

for i in range(9):

val = matrix[i][col]

if val == 0:

continue

col\_arr.append(matrix[i][col])

return col\_arr

#得到这个宫已经用过的数

def get\_block(num):

col\_arr = []

seq = num % 3

col\_end = 9 if seq == 0 else seq \* 3

row\_end = int(math.ceil(num / 3) \* 3)

for i in range(row\_end - 3, row\_end):

for j in range(col\_end - 3, col\_end):

val = matrix[i][j]

if val != 0:

col\_arr.append(matrix[i][j])

return col\_arr

#得到所在的第几宫（1到9）

def get\_block\_seq(row, col):

col\_seq = int(math.ceil((col + 0.1) / 3))#进1取整

row\_seq = int(math.ceil((row + 0.1) / 3))

return 3 \* (row\_seq - 1) + col\_seq

#获取可能的填入的数

def get\_enable\_arr(row, col):

avail\_arr = get\_random\_unit()#得到乱序的1到9

seq = get\_block\_seq(row, col)#得到所起的宫号（1到9）

block = get\_block(seq)

row = get\_row(row)

col = get\_col(col)

unable\_arr = list(set(block + row + col))

for v in unable\_arr:

if v in avail\_arr:

avail\_arr.remove(v)#把不可能填入的数排出去

return avail\_arr

def createSD():

can\_num = {}#

# 初始化一个9行9列的数组，初值都为0

for i in range(9):

matrix.append([0] \* 9)

#保存还未填入数字的空格的位置

box\_list = []

for row in range(9):

for col in range(9):

if matrix[row][col] == 0:

box\_list.append({'row': row, 'col': col})

i = 0

while i < len(box\_list): #未填满时循环

position = box\_list[i]#用i来读取未填入数字的位置表示的字典

row = position['row']

col = position['col']

key = '%dx%d' % (row, col)

#print(key)

if key in can\_num:

enable\_arr = can\_num[key]

else:

enable\_arr = get\_enable\_arr(row, col)

can\_num[key] = enable\_arr #以位置为键可能放入的数列表为值存入字典

if len(enable\_arr) <= 0:#没有可填入数字时，回退一个格子在重新新选入一个新的可能值填入

i -= 1

if key in can\_num:

del (can\_num[key])

matrix[row][col] = 0

continue #结束本次循环

else:

matrix[row][col]= enable\_arr.pop()#从可能数字中探出最后一个填入

i += 1

return matrix

6.2判断是否符合数独矩阵IsSDFun.py编码设计，实现编码：

from createFun import \*

def isSD(mat):

TS="提示消息!" #提示消息

flag=True

for i in range(9):

for j in range(9):#双循环遍历矩阵

if mat[i][j]==0: #有等于0的数说明，还有格子没有填入数字

flag=False

TS="输入有误！还有未填入数字！"

return flag,TS

elif mat[i][j] not in [1,2,3,4,5,6,7,8,9]:

flag=False

TS="输入有误！请填入到1到9的数字！"

return flag,TS

n=0 #记录mat[i][j]出现的次数

g\_num=get\_block\_seq(i,j)#得到所在的宫号

g\_arr=get\_block(g\_num) #得到这个宫已经用过的数字

#print(g\_num,g\_arr)

for k in g\_arr: #遍历宫里的数字，统计同宫mat[i][j]的个数

if mat[i][j]==k:

n+=1

for x in range(9):

if mat[i][j]==mat[i][x]:#统计同行mat[i][j]的个数

n+=1

if mat[i][j]==mat[x][j]:#统计同列mat[i][j]的个数

n+=1

if n>3: #为什么是3？从行、列、宫里都数了一次这个数

flag=False

TS="输入有误！与"+str(x+1)+"行"+str(j+1)+"列同宫、同行、同列的数"+str(mat[x][j])+"总个数有："+str(n-2)+"个"

return flag,TS

6.3主函数main.py编码设计，实现编码：

from createFun import \*

from IsSDFun import \*

import wx

import random,math

createSD() #创建数独矩阵 return matrix

print("完整矩阵")

print\_grid(matrix)

#matrix1用于储存残缺矩阵,赋初值0; matrix2用于作改变的中间矩阵

#matrix2和matrix1设置一样的初值，但是直接matrix2=matrix1，因为这样赋值是指针赋值，更改一个时另一个也变化

matrix1=[]

matrix2=[]

for i in range(9):

matrix1.append([0] \* 9)

matrix2.append([0] \* 9)

n=70 #设置显示个数

t=0

while t<n:

i=random.randint(0,8)

j=random.randint(0,8)

if matrix1[i][j]==0:

matrix1[i][j]=matrix[i][j] #从matrix中给matrix1对应位置赋值

matrix2[i][j]=matrix[i][j]

t+=1

print("残缺矩阵")

print\_grid(matrix1)

#生成一个字典保存文本框对象，以（0到80）为键，81个文本框对象为值，

#用于残缺矩阵和窗口文本框之间传值

dic={}

class Frame1(wx.Frame):

def \_\_init\_\_(self,superior):

wx.Frame.\_\_init\_\_(self,parent=superior,title="数独",size=(800,600))

panel=wx.Panel(self)

for i in range(9): #循环显示出matrix1,并给字典赋值

for j in range(9):

dic[i\*9+j]=wx.TextCtrl(panel,value=str(matrix1[i][j]),pos=(55\*j,55\*i),size=(50,50),style=wx.TE\_CENTER)

#dic[0].SetValue("10")

#print(dic[0].GetValue())

#print(dic)

self.btnCS=wx.Button(parent=panel,label=u"测 试",pos=(600,100),size=(100,30))

self.btnCL=wx.Button(parent=panel,label=u"重 来",pos=(600,150),size=(100,30))

self.TSText=wx.TextCtrl(parent=panel,pos=(550,250),size=(200,200))

#按钮点击事件绑定函数

self.Bind(wx.EVT\_BUTTON,self.CS,self.btnCS)

self.Bind(wx.EVT\_BUTTON,self.CL,self.btnCL)

def CS(self,event):

for i in range(9): #循环用把文本框的内容赋值给matrix2

for j in range(9):

matrix2[i][j]=eval(dic[i\*9+j].GetValue())

flag,ts=isSD(matrix2)

self.TSText.SetValue(ts)

def CL(self,event):

for i in range(9): #循环显示出matrix1，并给字典赋值

for j in range(9):

num=matrix1[i][j]

dic[i\*9+j].SetValue(str(num))

if \_\_name\_\_=="\_\_main\_\_":

app=wx.App()

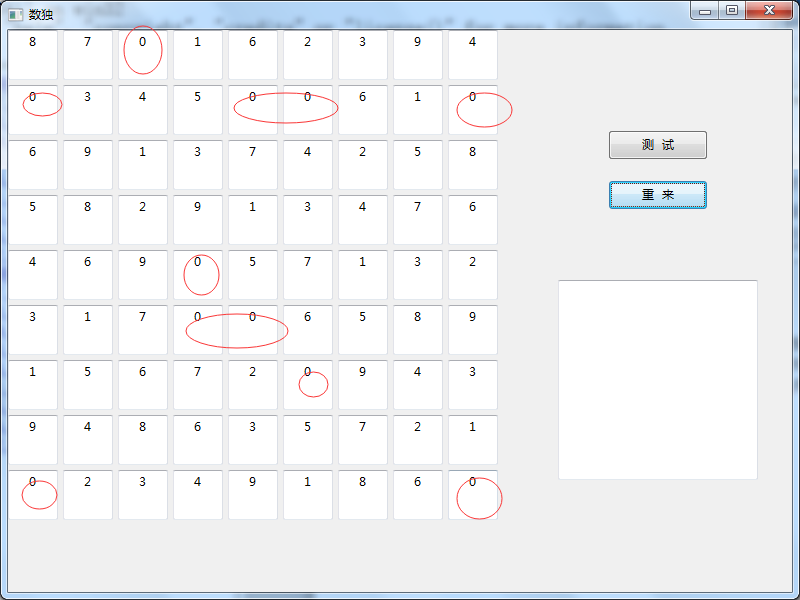
frame=Frame1(None)

frame.Show()

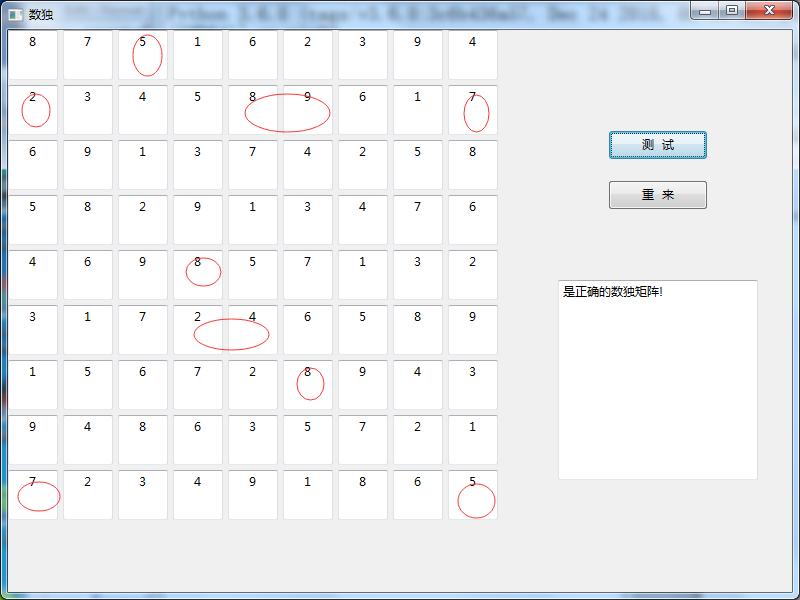
app.MainLoop()

1. 综合测试

运行程序：



填充数字后，点击测试：



1. 软件维护

可视化窗口不够美观，代码结构需要优化等……