**程序设计II大作业实验报告**

姓名：

学号：

## 实验题目与要求

本次实验主要内容是实现一个简单的数独软件，具体要求如下

### 基本功能部分

基本功能部分不需要通过图形化界面实现

下文提到的数独状态指9\*9的数独内容

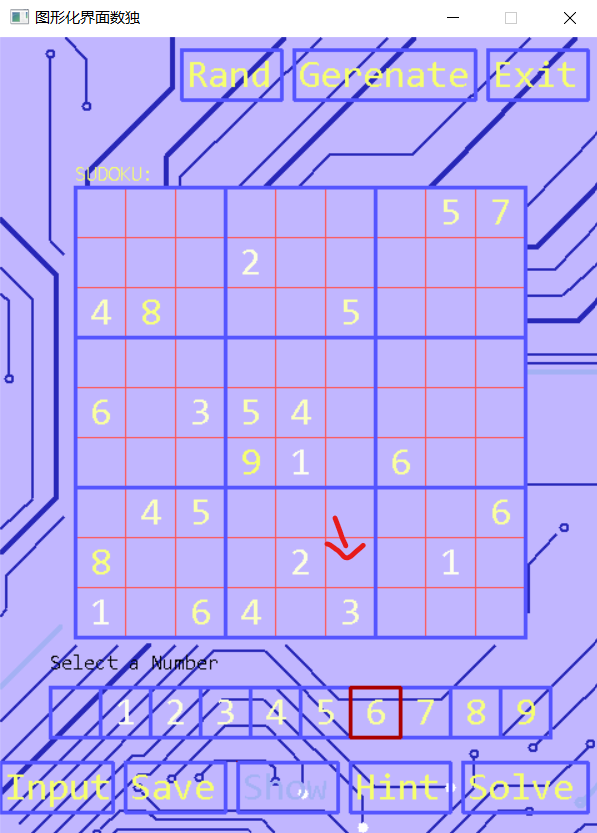
下文使用的数独仅作为格式示例，并不作为内容实例

#### 程序说明信息以及交互

程序运行后，需要输出提示信息，提示信息需要包括下面全部功能的输入说明

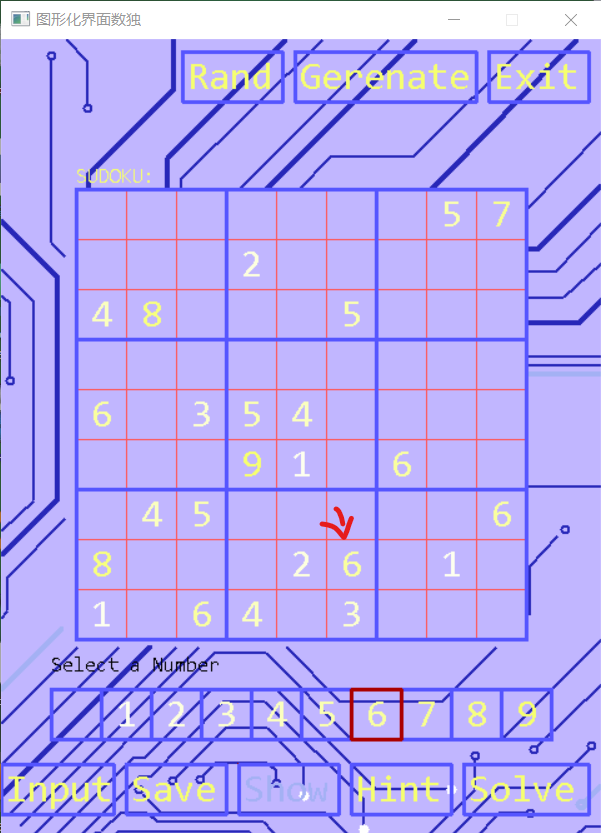
输入大部分指令后，程序响应指令后都应回到准备读取输入指令状态

输入

****

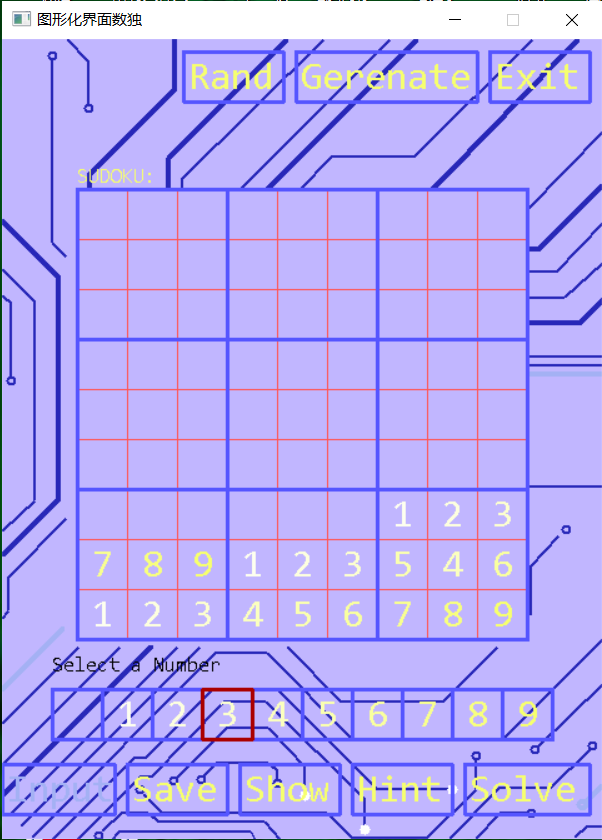
输出

并等待5秒后关闭命令行窗口

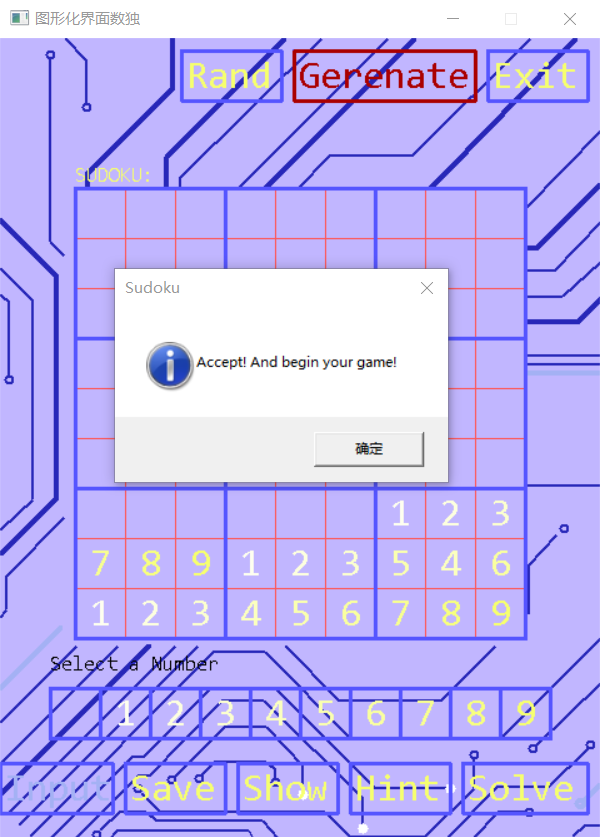
****

#### 输入数独

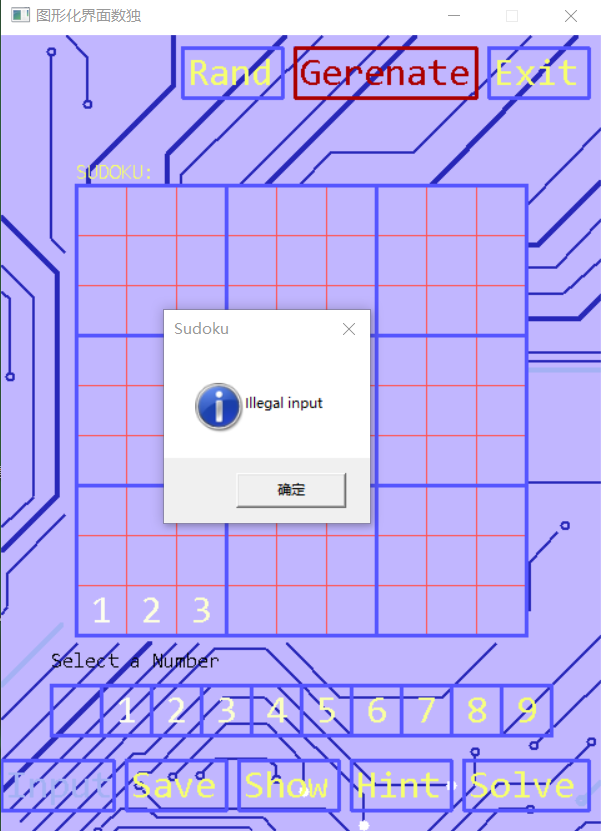
输入

****

如果输入是合法的（输入全是数字字符，且已填入位置的行列对角线九宫格不重复），存储输入到数独状态，输出

****

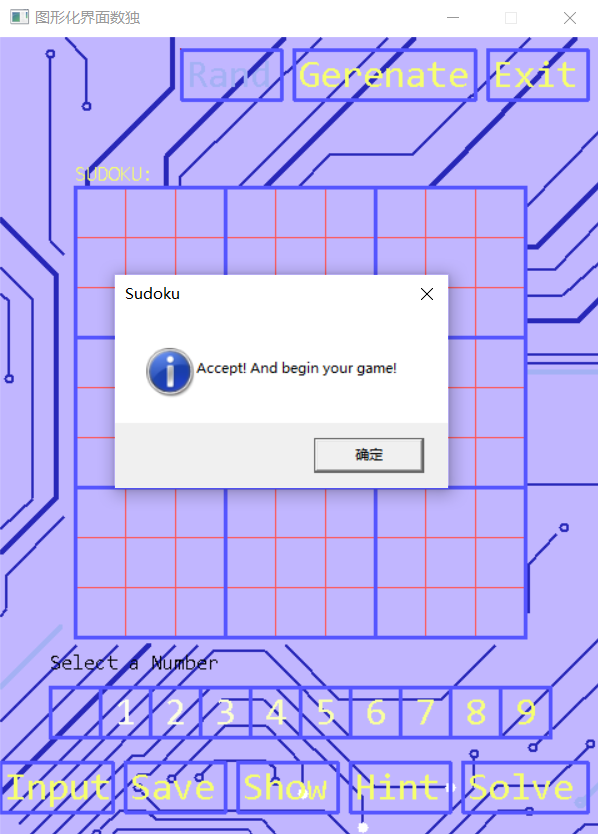
否则数独状态不应发生改变，并且输出

****

输入的数独需要至少17个非零数字，可以有多解

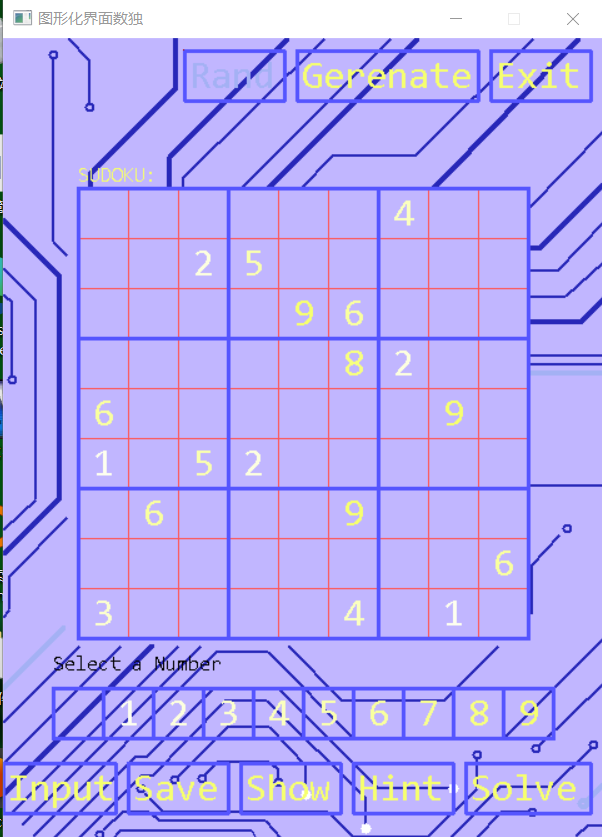
#### 生成数独

输入

****

程序设置数独状态为你生成的数独，并打印数独

输出

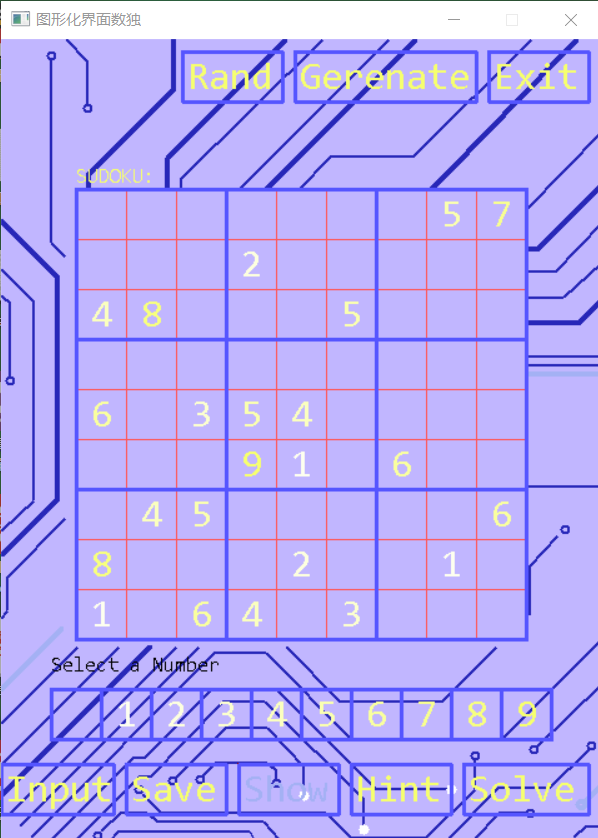
****

注意，生成的数独需要至少17个非零数字，可以有多解或**没有解**，但必须是一个合法的数独 （已填入位置的行列对角线九宫格不重复）

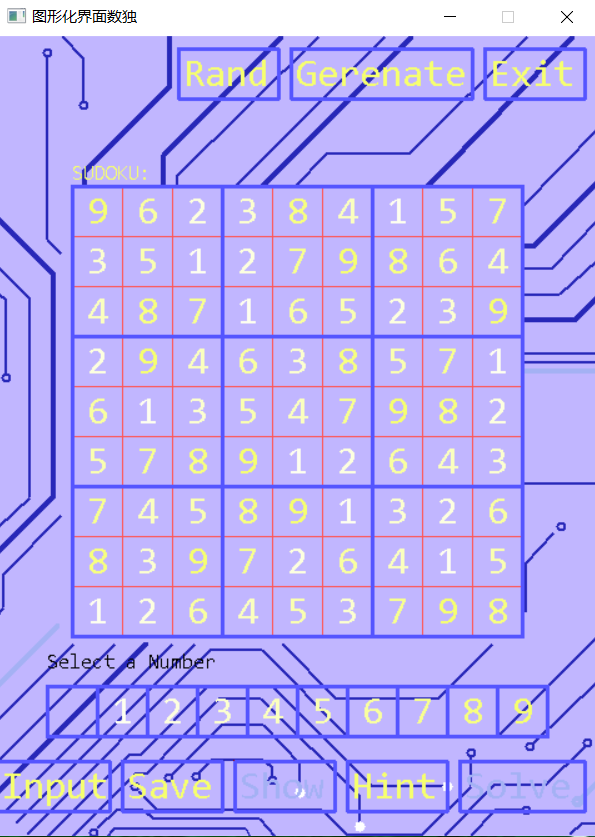
注意，这里合法的数独也可以是没有解的

#### 输出数独的解

输入

****

如果数独有解，输出

****

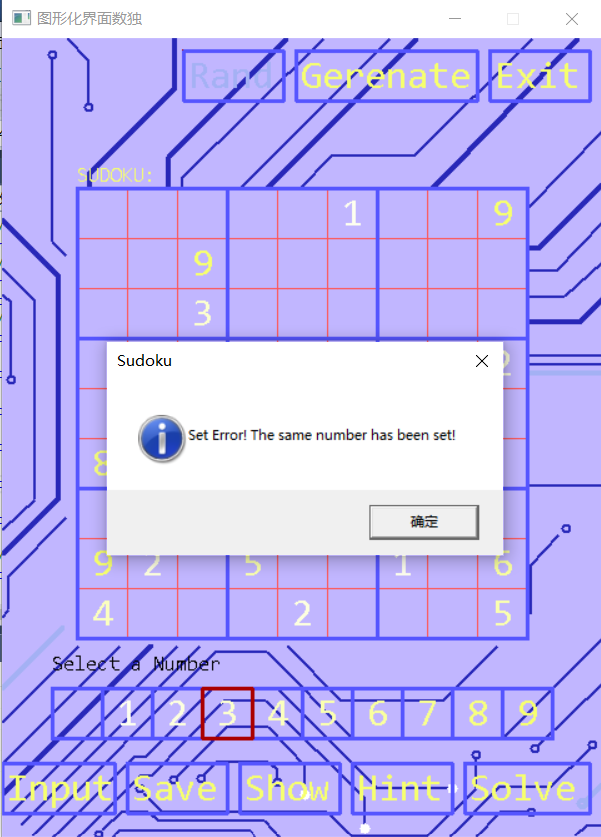
不需要输出所有的解。

注意，在解数独和hint功能的测试中，我们使用的测试样例一定是有解的数独。

#### 填入数字异常情况

你需要处理许多异常输入

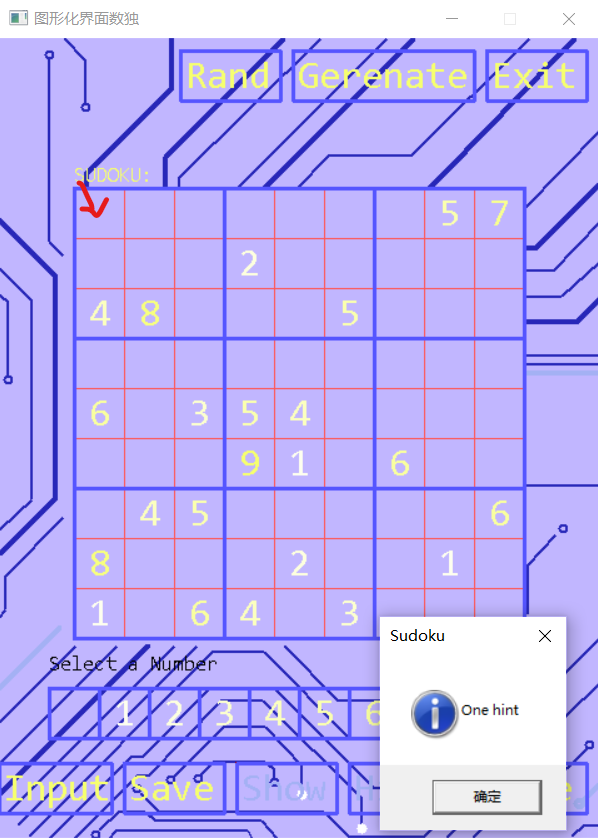
输出

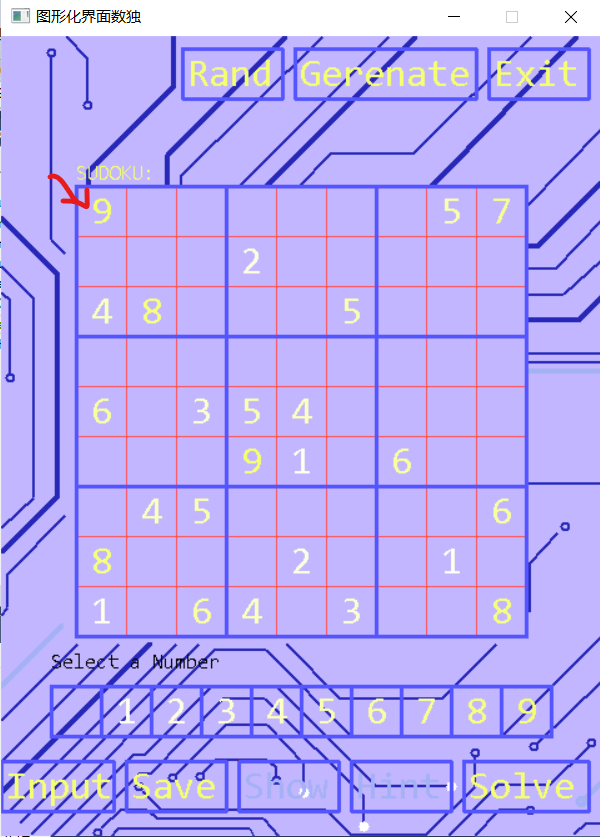
****

#### 数独提示信息

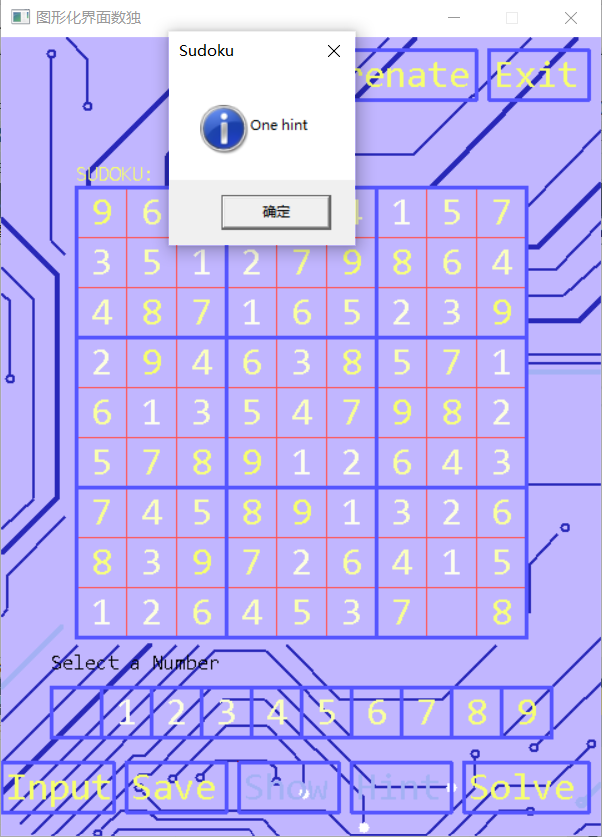
输入

程序选择一个未填入数字的空格，并填入一个非零数字作为提示信息，保存到数独状态后打印输出

****

****

注意，对于所有hint之前有解的数独，hint之后的数独也需要有解。（也就是说不能乱提示）。可以通过不停的hint获得最终的解。

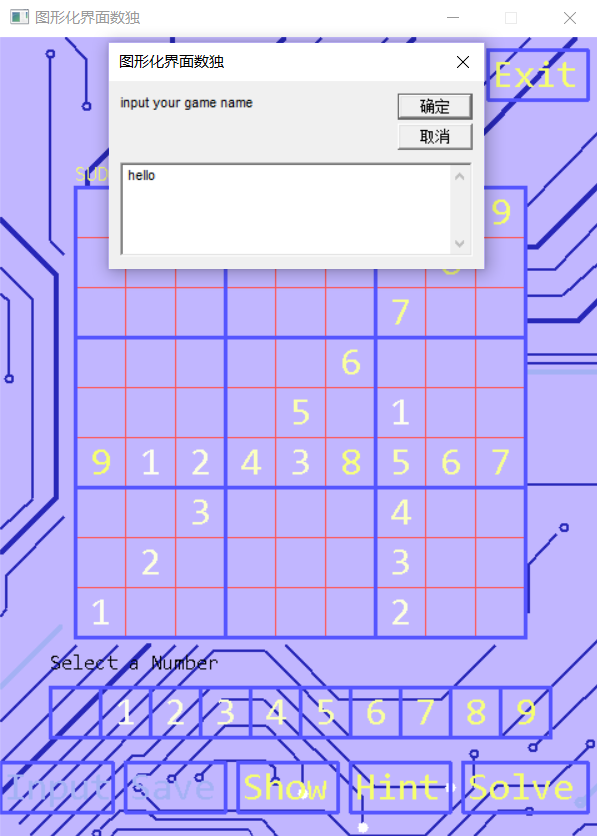
****

#### 保存游戏状态

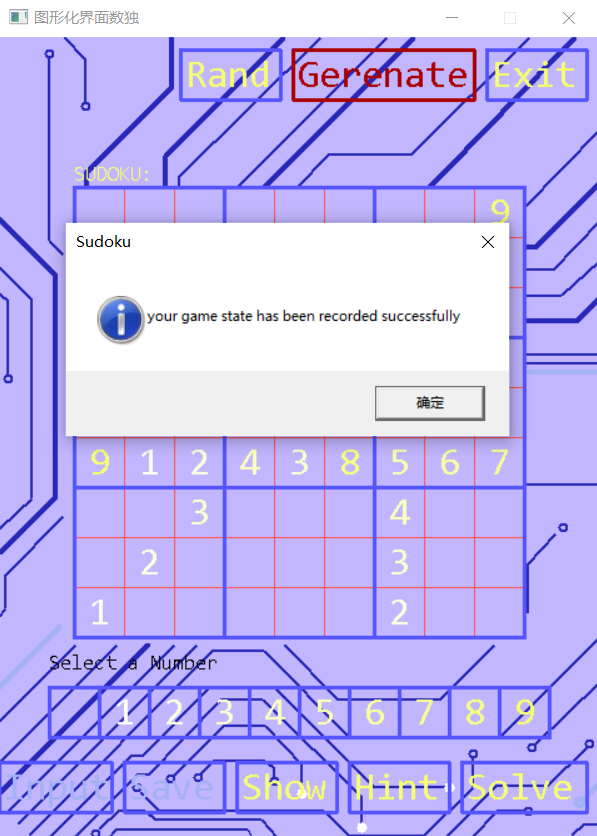
由于有些数独很难，可能一时半会儿解不出来，可以使用save 命令保存当前的数独状态，输入

输出

再输入自定义的记录名称hello

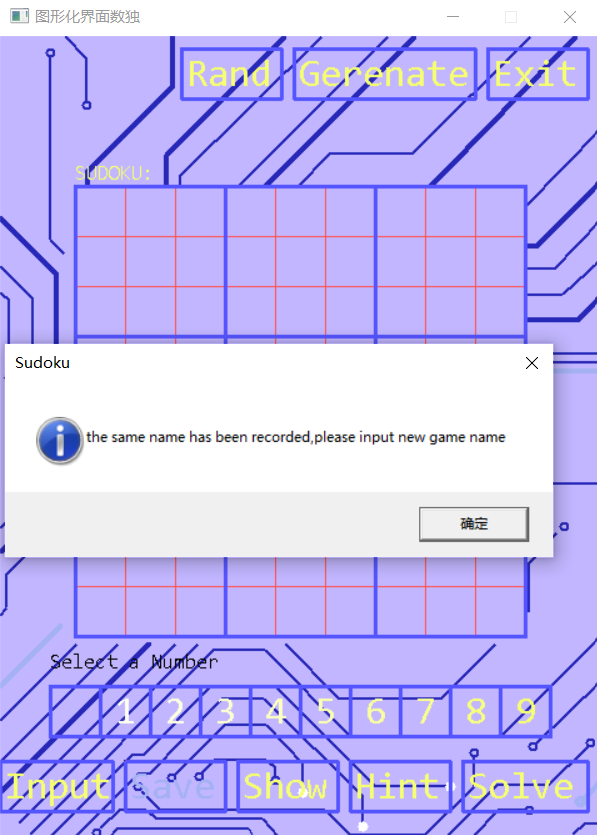
****

输出

****

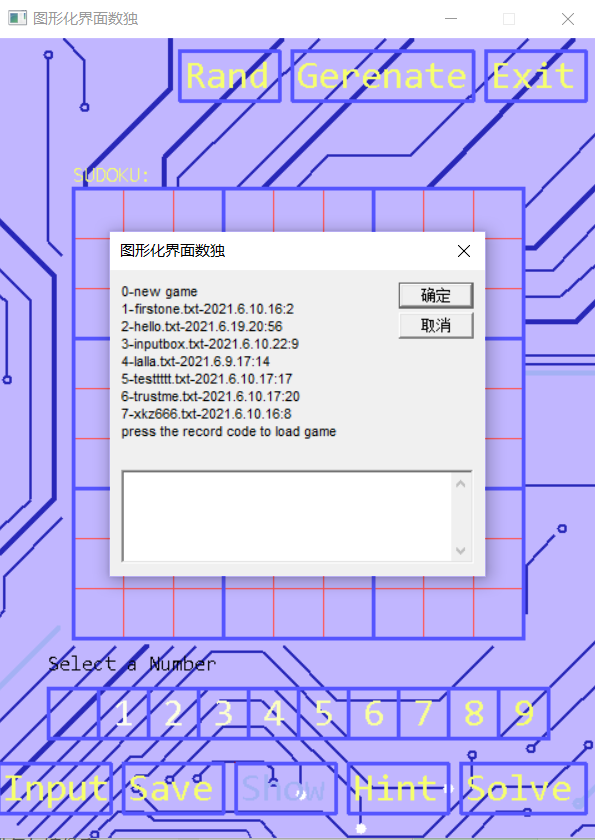
将当前的数独状态，记录名称，当前时间保存到同一目录的文件下

如果输入的记录名称重复，应该输出提示信息，程序回到接收记录名称输入状态

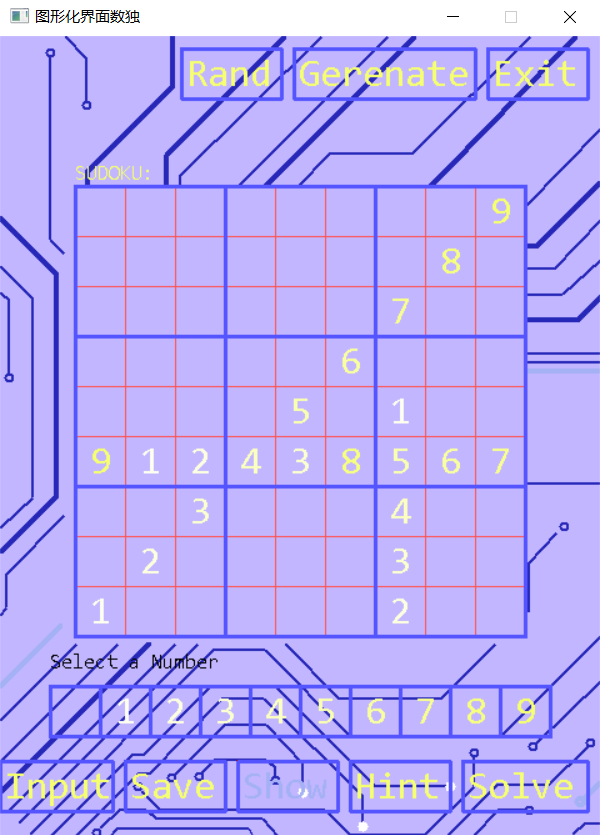
****

#### 加载游戏状态

可以查看已保存的游戏状态，输入

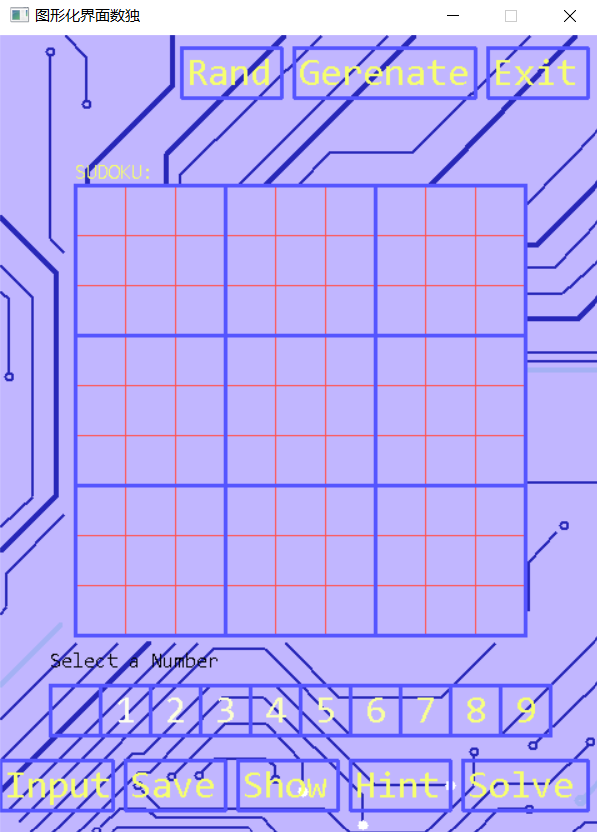
****

输出

****

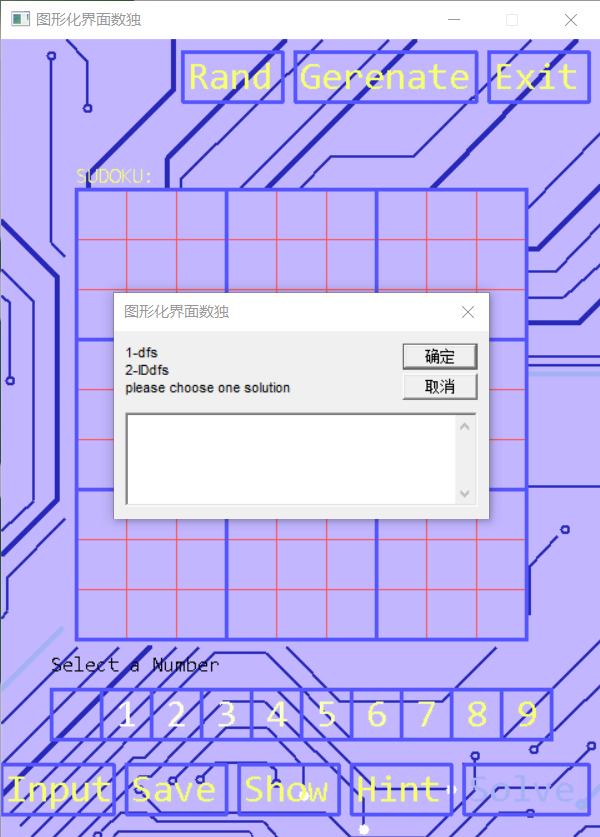
### 扩展部分

#### 图形化界面

****

#### 高级搜索技巧

学习并实现下列算法中的两种：迭代加深DFS，A搜索算法，迭代加深A

****

## 具体设计

该部分**简单介绍**实验中的函数拆分与图形化实现的细节，例如，包括哪些函数，函数参数是什么，作用是什么等。扩展功能的介绍等。

1. //下面的是设置按钮的函数
2. **void** iRAND();   //这个是随机生成的数独按钮
3. **void** iEXIT();   //这个是退出按钮
4. **void** iINPUT();  //这个是用户自行输入的按钮
5. **void** iSAVE();    //这个是用户进行保存游戏状态的按钮
6. **void** iSHOW();    //这个是展示已保存游戏状态的按钮
7. **void** iHINT();    //这个是数独提示的按钮
8. **void** iSOLVE();   //这个是进行解数独的按钮
9. **void** iGENERATE();  //这个是确定完成设定数独初始状态的按钮
11. //获取鼠标信息
12. **int** MouseMessage(**int** \*myx, **int** \*myy);   //获取非input状态下的鼠标信息
13. **int** Num\_MouseMessage(**int** \*myx, **int** \*myy);  //获取input状态下的鼠标信息
14. //这些是一些关于页面布局的函数
15. **void** Set(**int** n, **int** x, **int** y); //这个是将点击相应区域的数字赋值到相应的九宫格并显示
16. **void** Drawimage();    //对于这个页面的布局设计 包括所有的按钮生成，数独九宫格的设计
17. **void** Grain();          //这个是对于页面的纹路设计，起装饰作用
18. **void** Outtext(**int** i, **int** x, **int** y);      // i 的范围是0-9 当i==0时 打印的是一个空格
19. //当i在1-9范围内，打印相应的数字，x，y代表在页面布局下相应的打印位置
20. **void** Reshape(**int** number, **COLORREF** color); //这个是重新设定按钮颜色的函数，为了使文本框//看起来像是按钮，我们设定这个函数，来确定按钮是否被点击，color是重新设定的函数，number参数代表需要重新设定颜色的按钮
21. //这些是一些数独基本功能的实现函数
22. **void** Print();   //将数独现在的状态通过 Outtext()打印
23. **void** Exit();    //5秒后关闭程序
24. **void** Input();   //用户自定义的初始数独输入函数
25. **void** Rand();    //系统随机生成的合法的数独
26. **bool** Generate(**int** count);   //判断是否可以生成的函数 count代表已经输入或生成的数字，当//count>=17时，则输入是合法的
27. **void** Solve();  //求解当前状态数独
28. **void** Hint();   //从当前状态数独获取提示
29. **void** Save();   //保存当前状态到本地内
30. **void** Show();      //显示已保存的数独游戏
31. **void** dfs(**int** x, **int** y);  //使用dfs算法来求解数独和提示
32. **bool** IDdfs(**int** x, **int** y, **int** dep, **int** maxdep); //使用迭代加深算法来求解数独和提示
34. **bool** Win();   //判断用户是否胜利，当用户成功完成游戏后，判断用户胜利，并在5秒后结束游戏
35. **void** GetFileName(string path, vector<string>& files);   //从系统本地内获取已保存文件名//的名称，其中path代表保存路径，files代表获取的txt文件名称
36. **void** Wchar\_tToString(string& szDst, **wchar\_t**\*wchar);
37. //通过这个函数将Wchar\_t字符转化为string字符 作用是将whcar转化为szDst
38. **LPCWSTR** StringToLPCWSTR(std::string orig);
39. //通过这个函数将string转化为LPCWSTR字符类型  作用是将orig转化为LPCWSTR 并返回//LPCSWSTR参数

在这里主要介绍一下图形化数独布局的函数

1. **void** Sudoku::Drawimage()
2. {
3. setbkcolor(0xFFB6C1);                 //这里是设置背景颜色
4. cleardevice();                                 //清除所有状态
5. Grain();                                         //装饰的网格
6. setlinestyle(PS\_SOLID | PS\_ENDCAP\_SQUARE, 0);          //设置线条的格式
7. setlinecolor(0x5555FF);            //设置线条的颜色
8. // 九宫格的起始点
9. **int** x = 60;
10. **int** y = 120;
12. // 绘制九宫格，左上角坐标为（x ，y）每隔40画一条横线，并画一条竖线
13. **for** (**int** i = 0; i <= 9; i++)
14. {
15. line(x, y + i \* 40, 420, y + i \* 40);
16. line(x + i \* 40, y, x + i \* 40, 480);
17. }
19. setlinestyle(PS\_SOLID | PS\_ENDCAP\_SQUARE, 3);
20. setlinecolor(0xFF5555);
22. // 分割为九份，左上角坐标为（x , y）每隔120画一条横线，并画一条竖线
23. **for** (**int** i = 0; i < 4; i++)
24. {
25. line(x, y + i \* 120, 420, y + i \* 120);
26. line(x + i \* 120, y, x + i \* 120, 480);
27. }
29. settextcolor(BLACK);                  //设置背景颜色
30. setbkmode(TRANSPARENT);        //设置背景模式
31. settextstyle(20, 0, \_T("consolas"));         //设置输出字体
32. outtextxy(40, 490, \_T("Select a Number "));  //在(40,490)位置打印该字符串
33. //接下来这里是设置1~9以及清除状态的格子
34. line(40, 520, 440, 520);
35. line(40, 560, 440, 560);
36. **for** (**int** i = 0; i <= 10; i++)
37. {
38. line(40 + 40 \* i, 520, 40 + 40 \* i, 560);
39. }
41. // 给框格内填入数字
42. **for** (**int** i = 0; i <= 9; i++)
43. {
44. setbkmode(TRANSPARENT);
45. Outtext(i, 40 + 40 \* i + 12, 522);
46. }
47. //这里是设置按钮
48. iEXIT();
49. iINPUT();
50. iSAVE();
51. iSHOW();
52. iHINT();
53. iSOLVE();
54. iGENERATE();
55. iRAND();
57. settextstyle(20, 0, \_T("consolas"));
58. outtextxy(60, 99, \_T("SUDOKU: "));
59. }

下面是迭代加深的介绍

1. **bool** IDdfs(**int** x, **int** y, **int** dep, **int** maxdep);
2. //这是函数原型，x代表行，y代表列，dep代表当前层数，maxdep是最大允许层数
3. //这里是调用部分
4. **int** maxdep = 1;
5. **while** (!IDdfs(1, 1, 1, maxdep))
6. {
7. maxdep++;
8. **if** (maxdep > 81)
9. **break**;
10. }

具体实现功能如下：

1. **bool** Sudoku::IDdfs(**int** x, **int** y, **int** dep, **int** maxdep)
2. {
3. **if** (dep > maxdep)                //当层数超出时直接返回flase
4. **return** **false**;
6. **if** (sudoku[x][y] != 0)   //不等于0的直接跳过
7. {
8. **if** (x == 9 && y == 9)
9. {
10. Print();                    //找到答案并输出
11. **return** **true**;          //返回正确
12. }
13. **if** (y == 9) //行到顶端后搜索列
14. {
15. **if** (IDdfs(x + 1, 1, dep + 1, maxdep))
16. **return** **true**;              //可以进行下一次搜索 返回正确
17. }
18. **else** //搜索行
19. **if** (IDdfs(x, y + 1, dep + 1, maxdep))
20. **return** **true**;         //可以进行下一次搜索 返回正确
21. }
22. **if** (sudoku[x][y] == 0) //首先这里是没有填写的
23. {
24. **for** (**int** sudoku\_temp = 1; sudoku\_temp <= 9; sudoku\_temp++)
25. {
26. **int** num\_occupy, num\_row, num\_col, num\_house;
27. **int** house;
28. num\_occupy = (x - 1) \* 9 + y;
29. num\_row = (x - 1) \* 9 + sudoku\_temp + 81;
30. num\_col = (y - 1) \* 9 + sudoku\_temp + 162;
31. house = **int**((x - 1) / 3) \* 3 + **int**((y - 1) / 3) + 1;
32. num\_house = (house - 1) \* 9 + sudoku\_temp + 243;
33. //首先行列宫都没有重复的，再判断主次对角线有无重复，若有则跳过，没有则填
34. **if** (flag[num\_row] == 0 && flag[num\_col] == 0 && flag[num\_house] == 0)
35. {
36. //当主次对角线没有相同数字时 我可以让他填入
37. **if** ((x == y && diaflag[sudoku\_temp] == 0 && (x != 5 && y != 5))
38. ||(x + y == 10 && diaflag[sudoku\_temp + 9] == 0)
39. && (x != 5 && y != 5) ||(x == 5 && y == 5
40. && diaflag[sudoku\_temp] == 0
41. && diaflag[sudoku\_temp + 9] == 0))
42. {
43. //设置当前输入的状态，考虑行列宫对角线
44. **if** (x == y && diaflag[sudoku\_temp] != 3
45. && (x != 5 && y != 5))
46. diaflag[sudoku\_temp] = 2;
47. **if** (x + y == 10 && diaflag[sudoku\_temp + 9] != 3
48. && (x != 5 && y != 5))
49. diaflag[sudoku\_temp + 9] = 2;
50. **if** (x == 5 && y == 5 && diaflag[sudoku\_temp] != 3
51. && diaflag[sudoku\_temp + 9] != 3)
52. {
53. diaflag[sudoku\_temp] = 2;
54. diaflag[sudoku\_temp + 9] = 2;
55. }
56. sudoku[x][y] = sudoku\_temp;
57. flag[num\_occupy] = flag[num\_row] =
58. flag[num\_col] = flag[num\_house] = 2;
59. **if** (x == 9 && y == 9)
60. {
61. Print();
62. **return** **true**;
63. }
64. **if** (y == 9)      //列满后搜索下一行
65. **if** (IDdfs(x + 1, 1, dep + 1, maxdep))
66. **return** **true**;
67. **else**
68. {
69. sudoku[x][y] = 0;//当前格退出 并且将状态清空
70. flag[num\_occupy] = flag[num\_row] = flag[num\_col]
71. = flag[num\_house] = 0;
72. **if** (x == y && diaflag[sudoku\_temp] == 2
73. && (x != 5 && y != 5))
74. diaflag[sudoku\_temp] = 0;
75. **if** (x + y == 10 && diaflag[sudoku\_temp + 9] == 2
76. && (x != 5 && y != 5))
77. diaflag[sudoku\_temp + 9] = 0;
78. **if** (x == 5 && y == 5 && diaflag[sudoku\_temp] == 2
79. && diaflag[sudoku\_temp + 9] == 2)
80. {
81. diaflag[sudoku\_temp] = 0;
82. diaflag[sudoku\_temp + 9] = 0;
83. }
84. }
85. **else**           //搜索下一列
86. **if** (IDdfs(x, y + 1, dep + 1, maxdep))
87. **return** **true**;
88. **else**
89. {
90. sudoku[x][y] = 0;//当前格退出
91. flag[num\_occupy] = flag[num\_row] = flag[num\_col]
92. = flag[num\_house] = 0;
93. **if** (x == y && diaflag[sudoku\_temp] == 2
94. && (x != 5 && y != 5))
95. diaflag[sudoku\_temp] = 0;
96. **if** (x + y == 10 && diaflag[sudoku\_temp + 9] == 2
97. && (x != 5 && y != 5))
98. diaflag[sudoku\_temp + 9] = 0;
99. **if** (x == 5 && y == 5 && diaflag[sudoku\_temp] == 2
100. && diaflag[sudoku\_temp + 9] == 2)
101. {
102. diaflag[sudoku\_temp] = 0;
103. diaflag[sudoku\_temp + 9] = 0;
104. }
105. }
107. }
108. //当主次对角线有相同数字 或者 不是在对角线上
109. //那么对角线的就不能填了，不是对角线的就可以继续填入
110. **else** **if** (x != y && x + y != 10)
111. {
112. sudoku[x][y] = sudoku\_temp;
113. flag[num\_occupy] = flag[num\_row] = flag[num\_col]
114. = flag[num\_house] = 2;
116. **if** (x == 9 && y == 9)
117. {
118. Print();
119. **return** **true**;
120. }
121. **if** (y == 9)
122. **if** (IDdfs(x + 1, 1, dep + 1, maxdep))
123. **return** **true**;
124. **else**
125. {
126. sudoku[x][y] = 0;//当前格退出
127. flag[num\_occupy] = flag[num\_row] = flag[num\_col]
128. = flag[num\_house] = 0;
129. }
130. **else**
131. **if** (IDdfs(x, y + 1, dep + 1, maxdep))
132. **return** **true**;
133. **else**
134. {
135. sudoku[x][y] = 0;//当前格退出
136. flag[num\_occupy] = flag[num\_row] = flag[num\_col]
137. = flag[num\_house] = 0;
138. }
139. }
140. }
141. }
142. }
143. **return** **false**;
144. }

## 实验过程

该部分介绍实验中遇到的bug与报错

**报错内容**

1. 关于数独解的bug

2. 关于Hint()函数不能一直进行直到解出数独

3. Set()函数的打印显示bug

4. Input()结束后和用户进行输入时Set()函数的衔接部分出现数组越界

5. 对于进行Input()函数的死循环的bug

6. 关于Show()函数下的文件名读取的bug

7. 关于文件保存的时间位置在文件名上导致与用户输入保存名是否重复的冲突的bug

8. 清除数字时的bug

9. 使用Show()后的输入有误

10. 输入0-new game 后无法进行新的input

**报错原因**

1.原因是在（5，5）这个位置没有考虑清楚，（5，5）满足两条对角线，其flag值需要特殊判断

2.原因是每使用一次Hint()后得到的hint数字，没有进行flag的赋值

3.使用Set()函数来直接进行判断数字是否输入有误（即有同行同列同宫同对角线有相同数字）的情况下，在允许相同位置覆盖的情况下，下一次输入已被覆盖数字时会报错

4.原因在于当鼠标移动时，也会获取鼠标的信息，这样的情况下，得到的x，y坐标返回给Set后会转化为越界的数组维数

5.当我进行Input()时，在Input()函数内判断终止对于鼠标信息并没有帮助，即在Input()函数内声明的终止信息是无效的，导致了死循环

6.文件名读取会把当前文件夹(path)下的所有文件名给读入数组状态

7.原因是一开始我将时间衔接到文件名后，导致无法判断是否重复的bug

8.用空格时会清除初始的数独状态

9.原有的数独判断状态未被清空（flag、diaflag、sudoku）

10.原有的flag\_input 没有清除状态

**解决方案**

1.特殊判断（5，5）位置，分为非（5，5）对角线，非对角线，（5，5）位置三种情况考虑

2.对其进行flag的赋值

3.不允许相同位置进行赋值，可以使用空格来清除已填入的数字(非初始值)

4.进行特殊判断，即sudoku[i][j]的i，j状态只允许1-9范围内

5.声明一个flag\_input，来判断是否Generate()来确定输入完毕

6.声明一个临时string，将文件名最后三位赋值给string，来特判txt文本文档

7.将时间保存到文档内第一个位置，然后再从文本读取数组时，现用一个string将时间给读取即可进行接下来的数组读取

8.设置flag多一个3，为初始状态，保证初始状态是不能被改变的

9.调用Show()函数时，清空sudoku和flag和diaflag

10.清除flag\_input的状态

## 实验总结

首先最大的时延收获是学会了EaxyX这种图形库，了解其使用以及内置的函数：绘图窗口initgraph、closegraph、cleardevice等函数、line画线、rectangle画矩形等等一系列基本但重要的函数。接着是进一步的掌握了c++的程序编写，对面向对象编程有了个更深一步的了解与体会，也是第一次编写一个功能较完整的、工程量较大的程序。同时学会了许多知识点，例如c++文件读写输入，时间，文件信息结构体，一些字符的转化，深度有限搜索算法(DFS)，迭代加深DFS，同时提高了从中发现并修改bug的能力。