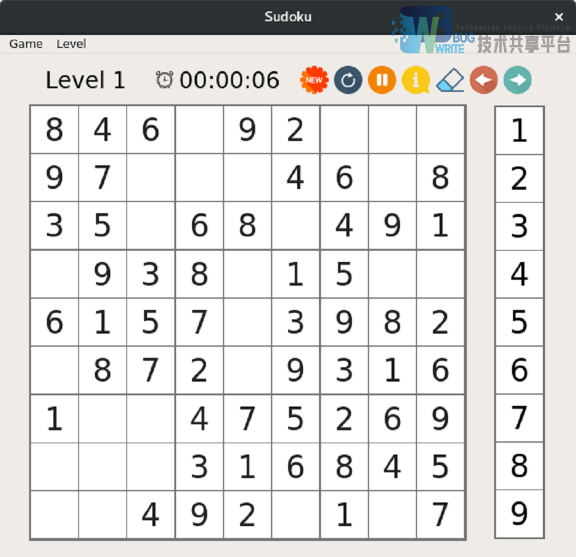
**实习二 基于回溯算法的数独游戏设计与实现**

**一、实习目的与要求**

**【问题描述】**

数独，是源自18世纪瑞士的一种数学游戏。是一种运用纸、笔进行演算的逻辑游戏。玩家需要根据9×9盘面上的已知数字，推理出所有剩余空格的数字，并满足每一行、每一列、每一个粗线宫（3\*3）内的数字均含1-9，不重复。本次实习要求设计并实现一款数独游戏软件（下图为9x9数独游戏示例）。



**【基本要求】**

1、采用回溯算法或舞蹈链（Dancing Link）实现数独游戏的求解；

2、采用控制台或GUI，实现语言不限；

3、设计并实现数独的基本功能，可根据个人情况对以下功能进行裁剪：

（1）新游戏：玩家可以开始一局新的游戏

（2）重玩：玩家可以重新开始本局游戏

（3）暂停：玩家可以暂停该局游戏（即暂停计时）

（4）提示：如果当前已经确定的数都是正确的，玩家将会得到一个未填空格的正确数字；如果当前已经确定的数和答案矛盾，导致整个数独无解，那么所有与答案矛盾的数字将会被粗体标出

（5）清除：清除当前选中格子的所有数字

（6）撤销：撤销前一步的操作，以及取消撤销（最多可支持 50 步撤销）

（7）同时可以通过菜单来实现多达 10 种难度的游戏选择，可以求解任意用户输入的数独问题。

**二、分析与设计**

1. **需求分析与类设计**

**需求分析：**

**我对于该GUI的实现，使用了QT对该项目进行完成**

* + - 1. **需要我们对用户界面开始一个新游戏，需要生成数独游戏。**
      2. **需要我们在结束游戏的时候为玩家提供正确的解**
      3. **需要我们给游戏设置难度。**

**类设计：对于本次的实习设计了以下：**

**对于数独游戏程序提供了两个头文件：generate.h 和 solve.h。**

**·initialiseSudoku函数实现了数独的数组的初始化  
void initialiseSudoku(int grid[9][9]){**

**for(int i = 0; i < 9; i++){**

**for(int j = 0; j < 9; j++){**

**grid[i][j] = 0;**

**}**

**}**

**}**

**·newgameinit函数实现了数组的内容的初始化。  
void newgameinit(int grid[9][9], int arr[], int level){**

**srand(time(0));**

**initialiseSudoku(grid);**

**grid[0][0] = ((int)rand() % 9) + 1;**

**grid[4][4] = ((int)rand() % 9) + 1;**

**grid[8][8] = ((int)rand() % 9) + 1;**

**if(solve(grid, arr)){**

**for(int row = 0; row < 9; row++){**

**for(int col = 0; col < 9; col++){**

**if(level == 1){**

**if((int)rand() % 4 == 0){**

**grid[row][col] = 0;**

**}**

**}else if(level == 2){**

**if((int)rand() % 3 == 0){**

**grid[row][col] = 0;**

**}**

**}else{**

**if((int)rand() % 2 == 0){**

**grid[row][col] = 0;**

**}**

**}**

**}**

**}**

**}**

**}**

**·findEmpty函数的作用是在数独棋盘中查找空白位置。**

**void findEmpty(int grid[9][9], int arr[]){**

**for(int i = 0; i < 9; i++){**

**for(int j = 0; j < 9; j++){**

**if(grid[i][j] == 0){**

**arr[0] = i;**

**arr[1] = j;**

**}**

**}**

**}**

**}**

**·isValid 函数的作用是判断数独棋盘中的某个位置是否可以填入数字。**

**bool isValid(int grid[9][9], int row, int col, int num){**

**for(int i = 0; i < 9; i++){**

**if(grid[row][i] == num && col != i){**

**return false;**

**}**

**}**

**for(int i = 0; i < 9; i++){**

**if(grid[i][col] == num && row != i){**

**return false;**

**}**

**}**

**int xCord = (row / 3) \* 3;**

**int yCord = (col / 3) \* 3;**

**for (int i = xCord; i < xCord + 3; i++){**

**for(int j = yCord; j < yCord + 3; j++){**

**if((grid[i][j] == num) && (i != row && j != col)){**

**return false;**

**}**

**}**

**}**

**return true;**

**}**

**·solve函数的作用是实现了数独的求解功能**

**bool solve(int grid[9][9], int arr[]){**

**arr[0] = -1;**

**arr[1] = -1;**

**findEmpty(grid, arr);**

**if(arr[0] == -1){**

**return true;**

**}**

**int row = arr[0];**

**int col = arr[1];**

**for(int i = 1; i < 10; i++){**

**if (isValid(grid, row, col, i)){**

**grid[row][col] = i;**

**if (solve(grid, arr)){**

**return true;**

**}**

**grid[row][col] = 0;**

**}**

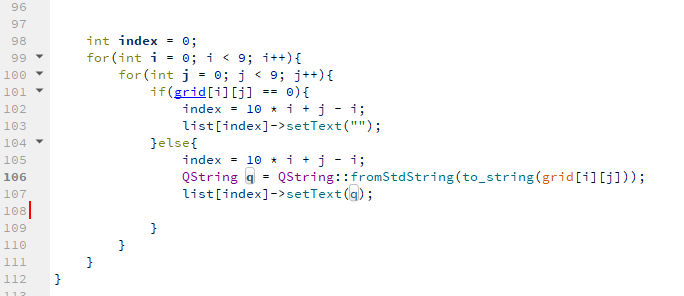
**}**

**return false;**

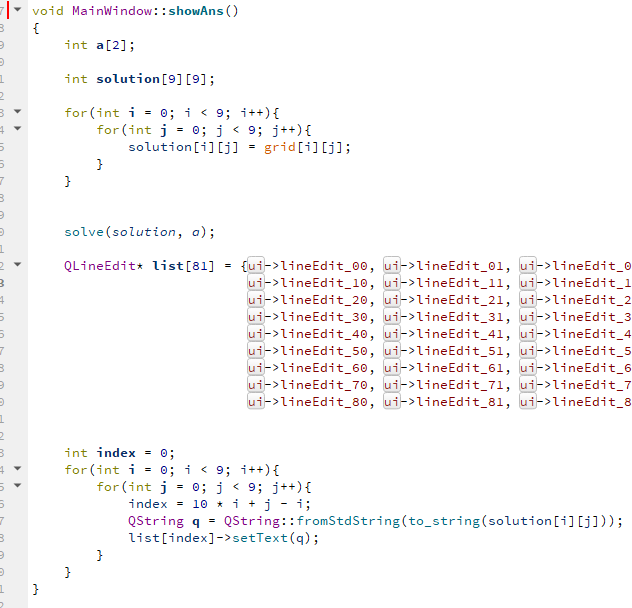
**}**

**·initbtn函数实现了代码的选择难度的功能。**

**·Givenums函数中将一个二维数组 grid 中的数独数据渲染到 UI 界面上对应的 QLineEdit 控件中。其中数独中的空格对应的 QLineEdit 控件的文本设置为空字符串，已有数字对应的 QLineEdit 控件的文本则设置为相应数字的字符串。这样，用户就可以通过 UI 界面来查看数独的当前状态。**



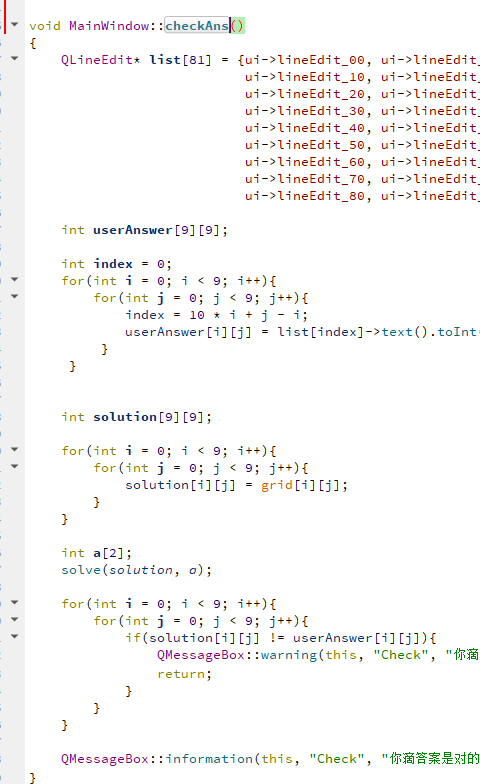
**·showAns函数实现了数独游戏中的显示解法功能，代码中定义了一个二维数组solution来保存已解出的数独矩阵，然后通过调用solve函数来求解数独，并将结果存入solution中。接着，通过遍历数独界面上的81个QLineEdit控件（每个控件对应数独矩阵中的一个格子），将solution中对应位置的数字设置为相应QLineEdit控件的文本。这样，当用户点击该按钮时，数独界面就会显示已解出的数独矩阵。**



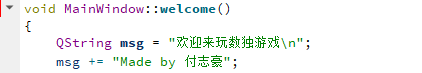
**·checkAns函数是当用户填完数独后，点击此按钮，用于检查用户是否填写了正确的答案。**

**该函数的实现分为以下几个步骤：**

1. **从QLineEdit类型的数组中获取用户填写的数独答案，存储到userAnswer数组中。**
2. **将数独谜题的解存储到solution数组中，通过调用solve函数获得。**
3. **将用户填写的答案与正确的答案进行比较，如果不一致，显示错误信息；否则，显示成功信息。**



**·Welcome函数则展示了一个欢迎界面。**



1. **算法设计与分析**

**1.我们的solve函数实现了数独的求解功能，使用了递归的思想，其作用是在给定的数独棋盘上尝试填入数字，如果能够成功填入并且完成整个数独，则返回 true，否则返回 false。**

**函数的参数包括一个 9x9 的二维数组 grid，表示数独棋盘，还有一个长度为 2 的一维数组 arr，其中 arr[0] 和 arr[1] 分别表示数独棋盘中下一个要填数字的位置的行和列。**

**首先在棋盘中找到一个没有填数字的空格，如果所有的空格都已经填满了，说明数独已经被解决，返回 true。**

**然后尝试在该空格中填入数字，如果填入的数字有效（即不违反数独规则），则继续递归调用 solve 函数，尝试在下一个空格中填入数字。**

**如果能够成功填满整个数独，函数返回 true，否则回溯到之前的状态，重新尝试其他的数字填入。**

**2.我们的isValid 函数的作用是判断数独棋盘中的某个位置是否可以填入数字。它的参数是数独棋盘 grid，要填入的数字 num，以及要填入数字的行列坐标 row 和 col。函数中分别判断在该位置所在的行、列、3x3 小九宫格是否已经存在数字 num，如果已经存在，则返回 false，否则返回 true**

**3.我们的findEmpty 函数的作用是在数独棋盘中查找空白位置。它的参数是一个 9x9 的二维数组 grid，表示数独棋盘中的所有位置；一个长度为 2 的一维数组 arr，表示空白位置的坐标。在函数中，双重循环遍历整个数独棋盘，当遇到值为 0 的位置时，将其坐标存入数组 arr 中，返回这个数组**

1. **测试与改进**

**对于本次实习，我们可以在程序中加入更多算法来提高程序求解的效率和准确性，例如有候选数法、独立数法等。此外我认为还可以优化UI界面，增加更多的交互功能，例如可以提示玩家下一步可选数字等内容。**

**三、实习小结**

本次实习的内容对我来说难度不算小，第一次接触QT的开发为我的界面实现还是创造了不小的难度。很多功能只有很简单的实现，不过通过本次实习的内容，我学习到了不少东西。比如Qt框架的基本使用：本次实习主要使用了Qt Creator进行开发，并学习了如何使用Qt Creator创建GUI应用程序、设计UI界面、添加控件、处理事件等。

数独算法的实现：数独算法是实现数独游戏的核心，我们使用C++语言实现了数独的生成和求解算法。；在求解算法中，我们采用了递归回溯的方式，能够较快地求解出数独。

界面交互和逻辑处理：数独游戏界面的交互和逻辑处理也是本次实习的重点之一。我们学习了如何使用信号和槽连接控件，如何响应用户的操作，如何处理游戏逻辑等。

**四、课程学习总结（个人收获与教学建议）**

一学期的算法的课程结束的很快，对我来说可以说是一瞬而过。在刚开始学习的时候我认为难度还是很大的，各种独特的想法，代码的实现，都是阻挡在算法路上的困难。但是随着老师带领我们不断学习，我认为算法课其实是我这个学期收获最多的一门课。内容很有意思，让我更加感受到了代码之间的数学关系的神奇。这是以前所不能了解到的。老师的教学方式十分棒，可以说是上大学以来最喜欢的课程。老师很温柔很喜欢，而且教学内容条理清晰，让人易懂。

-------------------------------

成绩评定：

教师签名：

批改日期：