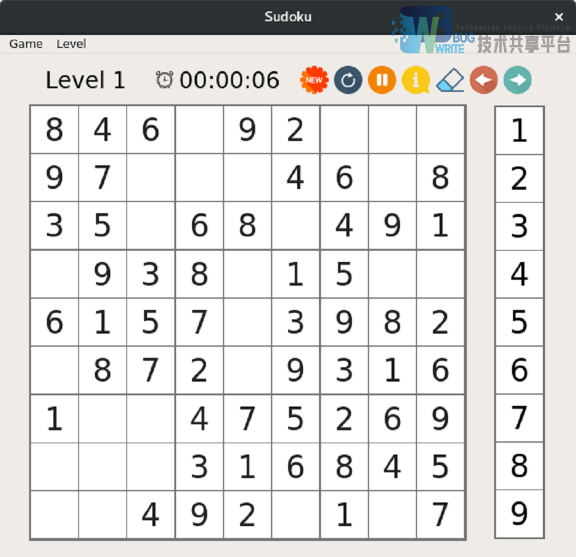
**实习二 基于舞蹈链（Dancing Links）的**

**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、采用舞蹈链（Dancing Link）算法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**
2. **需求分析：**

这次实习需要完成一个具有新游戏，重玩，暂停，提示，清除，撤销操作的数独游戏。完成这些操作主要是根据qt的ui操作完成，都是界面化的操作。非界面化的操作只要需要完成的就是对于给定数独题目的解题操作。因为这次要求需要使用“跳舞链”原理来进行数独的解题的操作。“跳舞链”的原理是完成精确覆盖，对于数独来说，数独的解只有满足以下四种情况才算正确题解：

1.所有格子必须填满数字

2.每一行都是1~9的一个排列

3.每一列都是1~9的一个排列

4.每一区域都是1~9的一个排列

对于数独中已有的元素，可将其进行转化，如（1，3）中填入4，则代表满足条件

（1）（1，3）有数字4，则第（1-1）\*9+3=3列

（2）第1行有数字4，则第（1-1）\*9+4+81=85列

（3）第3列有数字4，则第（3-1）\*9+4+162=184列

（4）第1区域有数字4，则（1-1）\*9+4+243=247列都为1，其他为0

将这些数据插入矩阵中，然后进行跳舞链的精确覆盖。其中0，81，162，247都是四种条件在矩形中的偏移量，为了四种条件存值不冲突分为四个部分。

**（2）类设计:**

class **MainWindow** : public QMainWindow

{

Q\_OBJECT

QWidget \* sudo\_Map;

QGridLayout\*Box\_input;

public:

explicit **MainWindow**(QWidget \*parent = nullptr);

~***MainWindow***();

void **show\_result**(int a,int b,int flag);//x和y轴坐标,flag表示是否是提示或者删除

void **get\_number**();//取得当前界面上存在的数字

void **build\_data**();//构造满足四种条件的数组

void **creatMatrix**();//将满足四种条件的数组进行双向循环连接

void **loadmap**();//对于题目给出的数先进行一次精准覆盖，存入结果，并且选定的节点暂时存到保存结果的数组中，对于选中的节点消去与它同行同列的节点。

void **search**(int t);//主要的“跳舞过程”

private slots:

void **on\_pushButton\_ts\_clicked**();

void **on\_level\_box\_currentIndexChanged**(int index);

void **on\_pushButton\_replay\_clicked**();

void **updateTime200**();

void **on\_pushButton\_begin\_clicked**();

void **on\_pushButton\_delete\_clicked**();

void **on\_pushButton\_submit\_clicked**();

void **on\_pushButton\_stop\_clicked**();

private:

Ui::MainWindow \*ui;

QTimer \* timer; //定时器 每秒更新时间

QTime \* TimeRecord; //记录时间

bool isStart; //记录是否已经开始计时

};

以及辅助函数：

void **cover**(struct Node \* col)//删除与该节点相关的行和列上的节点

void **uncover**(struct Node\*col)//恢复该节点

void **getresult**(void)//将得到的结果存储到数组中

int **dataLeft**(int i)//得到节点的左相邻节点下标

int **dataRight**(int i) //得到节点的右相邻节点下标

int **dataUp**(int i) //得到节点的上相邻节点下标

int **dataDown**(int i) //得到节点的下相邻节点下标

返回索引函数

int **RetNb**(int n){return n/81;}//每个格子

int **RetRw**(int n){return (n/9)%9;}//行

int **RetCl**(int n){return n%9;}//列

int **RetBx**(int n){return ((RetRw(n)/3)\*3)+(RetCl(n)/3);}//宫

int **retSq**(int n){return RetRw(n)\*9+RetCl(n);}//第一条件对应的下标

int **retRn**(int N) { return RetNb(N)\*9 + RetRw(N); }//第二条件对应的下标

int **retCn**(int N) { return RetNb(N)\*9 + RetCl(N); }//第三条件对应的下标

int **retBn**(int N) { return RetNb(N)\*9 + RetBx(N); }//第四条件对应的下标

int **current\_index**(int Nb, int Rw, int Cl) { return Nb\*81 + Rw\*9 + Cl; }//当前下标

结点的结构体：

struct **Node**{

struct Node \* Header;

struct Node \* Left;

struct Node \* Right;

struct Node \* Up;

struct Node \* Down; //头 上下左右的指针

char Name; //列头名

int Number; //列头数

};

1. **算法设计与分析**

在用跳舞链进行数组解题的过程中主要流程是:

1.先创建矩阵分为四个部分，每个部分分别满足上述四个要求。

Data[S\_Offset + retSq(index)][index] = 1; //规则1-单元格

Data[R\_Offset + retRn(index)][index] = 1; //规则2-行

Data[C\_Offset + retCn(index)][index] = 1; //规则3-列

Data[B\_Offset + retBn(index)][index] = 1; //规则4-宫

每个区域用偏移量隔开

2.然后将结点上下左右连接起来，构成一个双向循环链表。

//左邻居

do {

i = dataLeft(i);

} while (Data[i][j]==0);

Matrix[a][b].Left = &Matrix[i][j];

//右邻居

do {

i = dataRight(i);

} while (Data[i][j]==0);

Matrix[a][b].Right = &Matrix[i][j];

//上邻居

do {

j = dataUp(j);

} while (Data[i][j]==0);

Matrix[a][b].Up = &Matrix[i][j];

//下邻居

do {

j = dataDown(j);

} while (Data[i][j]==0);

Matrix[a][b].Down = &Matrix[i][j];

上述操作都是对于一个结点上下左右结点值不为0的连接。

3.然后读数据，对于题目进行一次“跳舞”，并且将结果存入对应的数组。

4.然后选择一个结点，进行带有回溯思想的“跳舞”。

对于跳舞的具体操作：

1.选定并删除相同行列的结点

col ->Right->Left = col->Left;

col->Left->Right = col->Right;

for(struct Node \* node = col->Down;node!=col;node = node->Down){

for(struct Node \* temp = node->Right;temp!=node;temp = temp->Right){

temp->Up->Down = temp->Down;

temp->Down->Up = temp->Up;

}

}

2.恢复该结点在矩阵中的位置

for(struct Node\*node = col->Up;col!=node;node = node->Up){

for(struct Node\*temp = node->Left;temp!=node;temp= temp->Left){

temp->Down->Up = temp;

temp->Up->Down = temp;

}

}

col->Left->Right = col;

col->Right->Left = col;

3.判断是否符合精确覆盖的条件，符合就输出，不符合就回溯，并且将之前存入的数据删除。当根节点的左节点和右节点相等，说明跳完舞，接下来就要判断当前存入的数值的数量。

if((RootNode->Left == RootNode && RootNode->Right==RootNode) || k == (81-added)){

getresult();

finish = 1;

return;

}

struct Node \*Column = ChooseColumn(); //向右取列

cover(Column);

struct Node \*RowNode;

struct Node \*RightNode;

for(RowNode = Column->Down; RowNode!=Column && !finish; RowNode = RowNode->Down) {

Result[n++] = RowNode->Number;//在内部插入列

for(RightNode = RowNode->Right; RightNode!=RowNode; RightNode = RightNode->Right) {

cover(RightNode->Header);

}

search(k+1);

//如果不满足要求恢复

for(RightNode = RowNode->Right; RightNode!=RowNode; RightNode = RightNode->Right) {

uncover(RightNode->Header);

}

Result[--n] = 0;//移除列

}

uncover(Column);

1. **测试与改进**

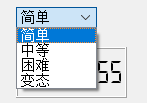
主界面：



当按下开始之后新的游戏就开始了



可以更改难度：



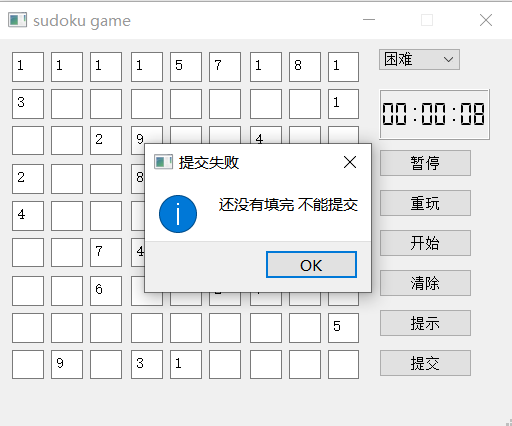
游戏解答都正确时：



当游戏解答有错误时：



如果没有填写完成：



**三、实习小结**

这次实习主要根据跳舞链这一数据结构进行数独问题的解答，利用的就是跳舞链的精确覆盖原理。通过这次实习，我从互联网等渠道了解到了跳舞链的结构和原理，可以说跳舞链这种结构对于我来说非常复杂，理解起来也很抽象，特别是和数独问题连结起来之后，我需要用很长的时间进行理解。跳舞链与数组对问题进行操作的比较，跳舞链能很简便的完成对结点的恢复和删除，而数组则显得更复杂，这是跳舞链的方便之处，但是在构建时非常复杂。最后完成了算法的设计之后，运用了qt的ui进行了界面化的设计，第一次运用Qtime定义了一个计时器，因为缺乏qt的使用，界面操作也很生疏，不能很好的运用模板，每个组件都是自拖拉定位置，每个textedit都单独进行操作，一共操作81个textedit显得非常冗余，不会使用更简便的方法。

**四、课程学习总计**

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

成绩评定：

教师签名：

批改日期：