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

1.1 需求分析

1. **输入方式**：点击响应的格子，从键盘输入要设定的值即可（只有在输入 1-9 的数字时才响应，其他输入不响应）。已经给定的格子设置为不可点击，并且统一用一种颜色标记出来，正点击的按钮显示一种颜色，输入后的按钮又显示另一种颜色。
2. **提示功能**：点击格子后，选择提示，如果目前填入的数字没有错误，则提示该格子该填的数字；如果当前填入的数字已经导致无解，则提示相关信息（用回溯或舞蹈链获取答案，这里我使用的是回溯算法）。必须先选择格子才能进行提示，若没有选择格子，提示按钮为不可点击的状态。
3. **其他功能**：**重新开始**：重新开始一局游戏，点击后提示用户是否立即重新开始；**切换难度**：从下拉框选择难度切换到不同难度的新游戏，同样，选择后提示用户是否立即更换难度；**撤销**：回到上一步，最多连续撤销 50 步，提示得到的答案不支持撤销，若已经撤销到底，撤销按钮为不可点击的状态；**暂停（未实现）**：停止计时，点击暂停后按钮提示文字变为“继续”，再次点击又变为“继续”。

提交：提交答案，提示正确或错误。

1.2 类设计

由于使用的是回溯算法，我觉得没有必要设计单独的类，所以把内容都写在了对话框的类下面。以下是我设计的内容：

名称	类型/返回值	备注
mapBtn	CMFCButton**	动态按钮，表示数独 9*9 的九宫格，私有成员变量
m_combobox	CComboBox	难度选择的下拉框，私有成员变量
pre_choice	int	上一次选择的难度，私有成员变量
pre_id	int	上一次选择的格子的 ID，私有成员变量
ans	int**	用户输入的答案，私有成员变量
Ans	int**	程序根据当前填写内容计算得到的答案，私有成员变量
m_Button_back	CButton	撤销按钮，私有成员变量
m_Button_tips	CButton	提示按钮，私有成员变量
allaction	vector<action>	存储用户执行所有的步骤，用于撤销。（action 是自定义的一个结构体，存有对应格子的 ID 以及该格子上一次的内容），私有成员变量
easy	int**	简单难度的初始数据，私有成员变量
middle	int**	中等难度的初始数据，私有成员变量
hard	int**	困难难度的初始数据，私有成员变量
hardest	int**	终极难度的初始数据，私有成员变量

preTranslateMessage	BOOL	响应键盘输入，并显示到对应的格子上，公有成员函数
OnCbnSelchangeCombo	void	改变难度，下拉框选择难度时响应，公有成员函数
OnBnClickedButtonSubmit	void	提交按钮，公有成员函数
OnBnClickedButtonBack	void	撤销，回到上一步，公有成员函数
OnBnClikedButtonPauser	void	暂停/继续，公有成员函数
OnBnClickedButtonTips	void	提示，若输入无误，提示一个当前格子该填入的值，公有成员函数
OnBnClickedButtonRestart	void	重新开始游戏，公有成员函数
backtrace	void	进行回溯，计算答案，公有成员函数
isplace	bool	回溯过程的辅助函数，计算当前格子填入是否合理，公有成员函数

1.初始化设置：在对话框的初始化函数中动态创建按钮，设定按钮位置，并默认使用简单难度的初始数据，设置相关按钮的属性以及初始化ans数组，在BOOL CSudokuDlg::OnInitDialog()函数内容中添加以下内容：

```
m_combobox.SetCurSel(0);//默认选择“简单”难度
prechoice = 0;//上一选择为“简单”

for (int i = 0; i < 9; i++) {
    for (int j = 0; j < 9; j++)
    {
        mapBtn[i][j].Create(_T(""), WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON, CRect(
            40 + j * 60,
            20 + i * 60,
            100 + j * 60,
            80 + i * 60), this, IDC_D_BTN + i * 9 + j);//创建按钮
        mapBtn[i][j].m_bDontUseWinXPTheme = TRUE;
        mapBtn[i][j].SetFaceColor(RGB(255, 255, 255));
        if (easy[i][j] != 0) {
            mapBtn[i][j].EnableWindow(false);
            CString Str;
            Str.Format(_T("%d"), easy[i][j]);//int转CString
            mapBtn[i][j].SetWindowTextW(Str);
            mapBtn[i][j].SetFaceColor(RGB(255, 255, 0));
        }
        ans[i][j] = easy[i][j];//初始化用户输入的答案
    }
}
m_Button_back.EnableWindow(false);
m_Button_tips.EnableWindow(false);
```

2.键盘输入响应：从键盘输入1-9间的数字时响应，将键盘输入的内容响应到按钮上，并设置按钮颜色，同时将这一步加入到步骤记录器allaction中，当allaction已满时，还需要弹出最早加入的那一步后再加入。代码如下：

```
//从键盘输入值
BOOL CSudokuDlg::PreTranslateMessage(MSG* pMsg)
{
    UINT nID = GetWindowLong(pMsg->hwnd, GWL_ID);//获取点击按钮的ID
    if (nID>= IDC_D_BTN&&nID<=IDC_D_BTN+80) {//点击的是九宫格区域的按钮
        if (pMsg->message == WM_CHAR) {
            int c = (int)pMsg->wParam - 48;
            if (c > 0 && c <= 9) {//输入1-9间的数字才响应
                CMFCButton* but = (CMFCButton*)GetDlgItem(nID);
                CString before;
                but->GetWindowTextW(before);
                if (_ttoi(before) != c) {//将这一步存到allaction中，若前后内容相同则不加入
                    action Step(nID, _ttoi(before));
                    if (allaction.size() < 50) {//未到50步，直接加入
                        allaction.push_back(Step);
                        m_Button_back.EnableWindow(true);
                    }
                    else {//已经存储了50步，将最先加入的那一步删除再加入
                        allaction.erase(allaction.begin());
                        allaction.push_back(Step);
                        m_Button_back.EnableWindow(true);
                    }
                }
                CString Str;
                Str.Format(_T("%d"), c);
                but->SetWindowTextW(Str);//设置按钮内容
                int row = (nID - IDC_D_BTN) / 9;
                int col = (nID - IDC_D_BTN) % 9;
                ans[row][col] = c;//响应到ans
                but->SetFaceColor(RGB(204, 255, 153));//设置颜色
            }
        }
    }
    return CDialogEx::PreTranslateMessage(pMsg);
}
```

3. 其他内容见源码。

2、算法设计与分析

利用回溯算法实现提示的功能：

首先需要获取点击的格子的 ID，由于点击“提示”按钮后就获取不到原先点击的按钮，所以就需要有一个变量 pre_id 来记录上一次点击的按钮的 ID。

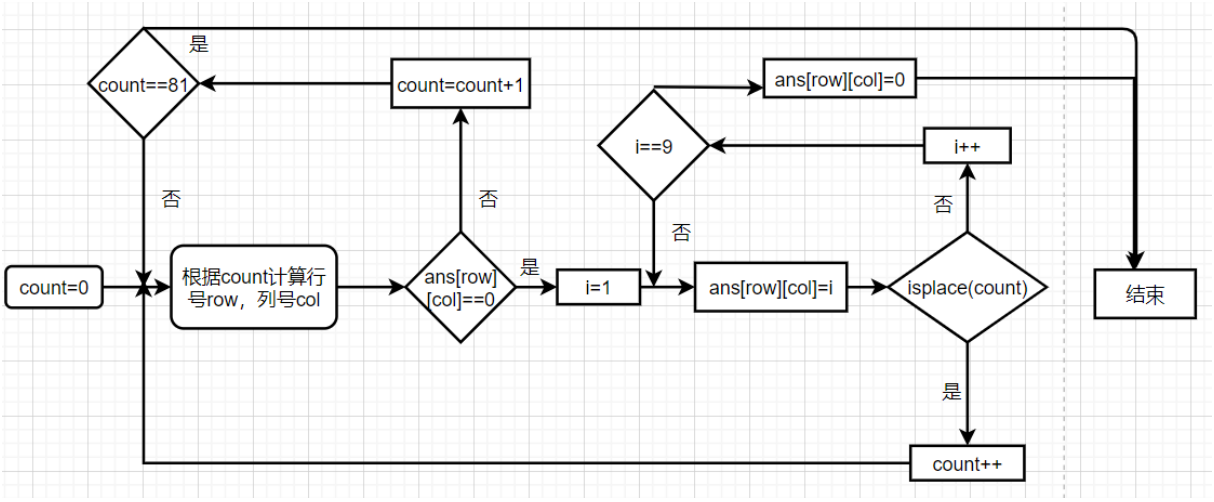
获取到按钮的 ID 后，就可以计算出该按钮在数组中对应的行号和列号。然后进行回溯，获取到答案，如果答案中当前按钮的答案为 0，那就说明用户输入的内容已经造成了无解，这是提示不能正常进行，需给出错误提示；如果答案中当前按钮答案不为 0 则将该按钮设置成答案即可。

回溯算法求解答案的具体内容如下：

从 0 号开始回溯，但检测到有错误填写或者到 81 号时结束。

如果遇到 ans[i][j]=0，则尝试填入 1-9 的数字，再调用 isplace 判断此处填写这个数字是否合适，

合适则直接进行下一个格子的处理，否则该格子就只能为 0 并结束回溯。流程图表示如下：



代码如下：

```
void CSudokuDlg::backtrace(int count) {
    if (count == 81)
    {
        for (int i = 0; i < 9; i++) {
            for (int j = 0; j < 9; j++) {
                Ans[i][j] = ans[i][j];
            }
        }
        return;
    }
    int row = count / 9;
    int col = count % 9;
    if (ans[row][col] == 0)//如果该位置为0，
    {
        for (int i = 1; i <= 9; ++i)
        {
            ans[row][col] = i;//将1-9填入该位置
            if (isplace(count)) //判断填入的数时候能够放入
            {
                backtrace(count + 1);//如果可以放入，则对下一个位置进行操作
            }
        }
        ans[row][col] = 0;
    }
    else//如果该位置不为0，则直接对下个位置的进行操作
    {
        if (isplace(count)) {
            backtrace(count + 1);
        }
        else {  return;}
    }
}
```

3、测试与改进

3.1 测试

刚开始时，由于没选择格子，提示按钮为灰色；还未进行操作，所以撤销按钮也为灰色：

现在随便输入一个，撤销和提示按钮都可以点击了：

此时点击提示，会提示已经无解：

再撤销这一步，又变成原来的样子：

全部使用提示的效果：


数独游戏
9 8 5 1 4 3 2 7 6
1 2 7 5 9 6 3 8 4
4 3 6 2 8 7 1 5 9
8 1 2 7 5 9 6 4 3
6 9 4 3 1 8 5 2 7
5 7 3 6 2 4 9 1 8
2 6 8 4 3 5 7 9 1
3 4 1 9 7 2 8 6 5
7 5 9 8 6 1 4 3 2
简单
重新开始
暂停
提示
撤销
提交
退出
提示：点击格子，从键盘键入要填的值

提交：


数独游戏
9 8 5 1 4 3 2 7 6
1 2 7 5 9 6 3 8 4
4 3 6 9
8 1 2 3
6 9 4 7
5 7 3 8
2 6 8 1
3 4 1 9 7 2 8 6 5
7 5 9 8 6 1 4 3 2
提示
恭喜你！答对了！
确定
简单
重新开始
暂停
提示
撤销
提交
退出
提示：点击格子，从键盘键入要填的值

其他难度：


数独游戏
中等
重新开始
暂停
提示
撤销
提交
退出
提示：点击格子，从键盘键入要填的值


数独游戏
困难
重新开始
暂停
提示
撤销
提交
退出
提示：点击格子，从键盘键入要填的值


数独游戏
8 1 2 7 5 3 6 4 9
9 4 3 6 8 2 1 7 5
6 7 5 4 9 1 2 8 3
1 5 4 2 3 7 8 9 6
3 6 9 8 4 5 7 2 1
2 8 7 1 6 9 5 3 4
5 2 1 9 7 4 3 6 8
4 3 8 5 2 6 9 1 7
7 9 6 3 1 8 4 5 2
终极
重新开始
暂停
提示
撤销
提交
退出
提示：点击格子，从键盘键入要填的值

3.2 改进

1.计时/暂停/继续的功能未实现。主要是对计时相关内容不熟悉。
2.采用舞蹈链算法求解。
3.可选择的难度还不够。

三、实习小结

本次实习练习的是回溯的算法，实际上刚开始是想要用舞蹈链来实现，但始终不太能理解舞蹈链的结构，最后不得不放弃舞蹈链采用回溯算法。在求解一些不太复杂的数独时，回溯算法求解的速度也不慢，与舞蹈链的差别应该也不是很大。但是在求解一些复杂的数独时，比如这里的“终极”难度，回溯算法刚开始提示时会感觉明显慢了一点，可能这就是舞蹈链的优势所在，这也是这次实习还需要优化的地方。

四、课程学习总结（个人收获与教学建议）

个人收获：这学期后半个学期开始上算法设计与分析这门课，相比上学期数据结构课中的算法内容，这学期讲的内容更加深入，也更加详细，同时也明显的感觉到比数据结构中讲到的算法难度更大了。但是通过每周的 leetcode 练习，通过自己去运用学到的算法思想解决问题， 也能加深对这些算法的理解。有时候做几道题下来基本就能大概知道在使用这种算法时的常规流程与要特别主义的地方。总之通过这学期的学习，我对那些算法有了更深的理解，理解了一些算法的基本框架与流程，有些时候甚至能真切地感受到算法带来的好处。但也还存在一些待提高的地方：比如还是不能看到问题就里面想出该用什么算法，于是就卡在算法的选择上。

教学建议：感觉这学期两次实习都没太能锻炼算法的设计与使用，反而大部分时间和经理都花费在界面以及其他内容上。希望以后能在实习中侧重算法的实现。

成绩评定：

教师签名：

批改日期：