装 订 线

# 数独游戏实现报告

班级：计算机一班
姓名：王哲源
学号：1652228

完成日期：2017.4.3

装

订

线

## 1.题目及其描述

题目主要目的在于通过 CMD 实现数独的伪图形化，其中有以下分问题

### 1.1 建议的内部数组实现

这一小题为游戏实现的最基本步骤，通过直接展现内部数组存储情况以实现游戏最基础的功能

### 1.2 在 1.1 基础上实现伪图形化

本小题通过 cmd 伪图形界面将内部数组展示变为伪图形化界面实现游戏的初步图形化

### 1.3 在 1.2 基础上实现游戏自行寻解

本小题是在伪图形界面展示的前提下实现程序的自动寻解

### *1.4 在 1.2 的基础上改为通过上下左右键选择格子

### *1.5 在 1.2 的基础上改为通过鼠标选择格子

## 2.整体设计思路

以下对于程序的一些设计思路进行概述，主要以 1.2 进行叙述

### 2.1 关于界面的存储

根据数独的格子特性，此处采用 11*11 的二维数组对数独进行存储，其中第 0 行、第 0 列；第 10 行及第 10 列使用-1 填满，以防某些情况下误操作导致的数组下标越界而带来一些不便利的地方。同时，使用另一个 bool 数组存储数独初始项坐标，以便后续操作中判断当前格子是否允许修改

### 2.2 关于数独文件读取与存储

题目要求将同名文件夹下的数独数据文件名读入后供游戏者选择，因此此处可以考虑采用链表存储读入的文件名，并根据文件名长动态分配文件空间。考虑到 1.2 中要求实现文件选择的伪图形界面，其中涉及上下滚动的情况，因此此处考虑采用双向链表实现当前文件的前驱及后继查询，并采用环状链表以实现文件选择界面的滚动

### 2.3 关于步骤的存储

由于游戏同时要支持回退操作，因此同样采用一个链表将操作存储起来，一方面存储本次操

作修改哪个位置的值，另一方面考虑到回退操作为撤销本次操作的影响，因此本次操作所填的数无需被存储，只需要存储该位置前一个数字为多少即可，在进行回退操作时直接覆盖为上一次的数字

## 2.4 关于冲突情况的判断

根据数独游戏规则，数独的冲突情况存在横、纵、子矩阵三种情况，且本游戏支持短暂的不合法填入，因此采用三个二维数组用于存储每行/列/子矩阵中 1-9 的个数，在填入的时候分别判断是否存在即可

### 2.4.1 关于坐标转子矩阵编号

将子矩阵自左向右，自上而下分别编号 1-9，则对点[x][y]其所属子矩阵编号为 i = (x - 1) / 3 * 3 + (y - 1) / 3 + 1，其中除法均为取下整

## 2.5 关于文件的选取

内部数组界面下，只需要根据输入的文件名直接判断是否文件存在即可；在伪图形界面下可以使用一个变量存储当前指向列表的第几个文件，当超出列表范围时对列表进行重新打印即可，以上步骤均可以利用 2.2 中所提及的双向环状链表实现

## 2.6 关于自动寻解

自动寻解采用的是最为简单的深度优先搜索方法，对每个点的所有合法解进行搜索，并在搜索中途不断更新与复原三个判断数组即可，直到最后一个格子搜索完毕。中途可以使用一个静态局部变量用于计数

*搜索加剪枝对于复杂数独寻解可能较为困难，有空时可以再尝试效率更高的 Dancing Link 算法

# 3. 主要功能实现

以下对整个程序的具体实现进行较为详细的介绍

## 3.1 预处理部分

首先利用 _findfirst 与 _findnext 函数及通配符对于当前目录下所有文件名为”sudoku*.txt”文件进行搜索，并将文件名储存于链表中，接下来输出给游戏者选择。在游戏者选择文件之后利用 ifstream 下函数打开文件将数独初始数据读入数组中，一方面为数组添加-1 作为边界，另一方面统计各行/列/子矩阵中所出现的各个数字个数，并以此为依据判断数独文件是否合法，若合法则继续，不合法直接终止接下来的操作并退出当前选项提示游戏者检查游戏数据

## 3.2 读入部分

数据的读入分为以下三种

### 3.2.1 键盘坐标+数字读入

这是最为基础的读入方法，通过 getline 将指令全部读入至字符串中，首先判断字符串长度是否符合几种操作，再根据各操作的输入要求分别进行判断，函数有一个 int 型返回值，当 f = 1 时操作合法，当 f = -1 时表示退出操作，当 f = 2 时表示回退操作，当 f 为 0 时表示输入不合法需要重新输入

### 3.2.2 键盘上下左右键控制光标读入

利用_getch 函数读取键位判断当前对光标的操作，并在键位为数字键时判断当前光标所处的位置是否为固定格，直到合法时才退出函数，其返回值同 3.2.2 中函数相同，只是少了 f = 2 的情况

### 3.2.3 鼠标选择格子读入

该读入方式分为两部分，首先时鼠标/VK 键值读入，读入时判断为鼠标右键（退出），Backspace 的 VK 值（退格）及鼠标左键（同时判断是否位于合法操作区间内）；当选择鼠标左键且合法时进行第二步读入，及对数字键盘的读入。同时注意到，由于该方式分为两部分，因此在选择格子之后若没有键盘读入程序将不会继续运行，为了使游戏者能够撤销误选，在第二部分读入时还添加了 ESC 键位的读取，使得游戏者可以不用键入新值就可以重新选择格子

## 3.3 cmd 伪图形界面的输出

伪图形界面的输出有两部分，一部分在每次操作前对新图的打印，还有一部分是在 3.2.2 与 3.2.3 读入操作时需要对选取的格子进行高亮打印

### 3.3.1 整体打印

整体图的打印一共有空格、固定格、不合法区域、冲突数字及前一次操作 5 种情况，前两者可以根据数独数组及预处理的 bool 数组直接进行判断，冲突数字可以根据 2.4 中的数组进行打印，前一次操作也可以通过变量直接定位。因此为了实现剩下一种情况，此处还需要引入一个二维 bool 数组 f[i][j]，其表示 2.4 中提及的第 i 个数组中编号为 j 的行/列/子矩阵存在不合法情况，在对数组中每个数打印时根据情况预先设定背景颜色及前景颜色即可

### 3.3.2 读入部分打印

该部分打印是在实现附加项 1.4 及 1.5 是才需要被调用，及对当前所选格子进行高亮显示。通过分析我们可以发现 1.5 的打印实际上是 1.4 的一种特殊情况（及不要求相邻操作），且两者每次只会涉及一个格子的操作，因此我们可以直接定位到被选格子所在的坐标进行覆盖打印即可。同时注意，对于新选格子，上一次所选的格子需要被复位，因此其所需参数与 3.3.1 中所需参数

没有差别，只需要开一个变量用于存储上一次选择的格子坐标，在新操作判定为合法后直接对旧的坐标抹去并更新为新坐标即可

### 3.4 回退操作

在链表非空时，回退操作可以被看作是一种特殊的键入操作，因此在执行回退操作时，可以将修改的格子坐标及数直接从链表中获取，接下来就可以与键入操作执行相同的操作

## 4. 调试过程中遇到的问题

以下对整个程序的具体实现进行较为以下将对调试过程中遇到的几个主要问题进行概述

### 4.1 读取不到文件

本题的文件名读取为自学部分，网上搜寻到的较为靠谱方法如 2.2 所提及。但在初步实验时发现无论什么情况都都不到文件。一开始认为是通配符存在问题，但通过测试发现通配符并不是问题所在，经过不断测试最终发现相对路径表示法中同名文件夹下文件为.\，而由于文件读取函数中的路径均为字符串表示方法，其中的\在 ASCII 中为转义字符的前缀，实际使用时应当表示为.\\，因此无法正确读取到文件路径并对其进行检索

### 4.2 附加读入方式时无法正确显示前一次操作

在执行附加读入时，在填入数字之后伪图形界面无法正确显示前一次操作的格子，而非附加读入却可以正常显示。通过逐条发现是附加读入中无论本次操作是什么都会先把前一次操作复位，因此在前一次操作完成之后由于没有延时，且附加操作要不停存储前一次操作，使得在极端的时间内前一次操作的格子被重新覆盖打印，因此无法正确显示前一次操作。通过修改后复位的步骤被移动至选择格子后，该情况得到了解决

## 5. 总结部分

### 5.1 对于程序前后衔接的收获

在对本次大作业的三题（不包括附加项）所耗时间进行总结时发现第一题所花时间接近半天，而在第一题检验无误后，第二题所花时间仅为两小时，而第三题更是短至半小时

总结其原因主要是第一题除去打印部分其余均为第二题的共用情况，而第三题则是抛弃读入部分的第二题，从第一题移植至第二题只需要添加伪图形界面输出部分及修改少部分光标移动的位置，而第三题则只需要直接调用第二题中的打印部分即可。

因此对于较大型的程序，不需要急于编写后期部分，而应该在编写好基础部分且确认无误的情况下为后期程序的扩展留有借口或尽量保留共用部分，在后期添加时对于基础部分代码进行直接的调用，或者在原有的基础上稍加修改供新版本使用。这样可以大大的节约重新编写的时间，也能保证程序的稳定性与可修改性

## 5.2 对程序细节考虑的反思

这次程序编写所花时间一大部分在于冲突的判断部分，最初思想是使用 3 个二维 bool 数组用于存储每个数字在当前行/列/子矩阵是否出现过，而在实际实现时发现该方法只能判断填入数字是否合法，而不支持在填入后再复位，否则原有标记也会被覆盖；而在修改为 int 数组进行个数统计之后又发现该方法在输出时不方便整行/整列的标记，而在检查数组中存在多少个冲突时又应用到该部分，最后才改为 3.3.1 中所述的方式。前前后后该部分被修改了至少 4 次并且因此浪费了大量的时间。这一点再次证明程序的初步构思及细节思考是十分重要的，所幸当时只是在编写第一小题部分因此避免了整个程序的大规模修改

## 6. 附件：程序（此处以第二小题为主且不包括附加项）

**游戏主体函数部分：**

```
    void game_img(const int mode, int(*map)[N + 1], bool const(*fix)[N + 1], int(*fx)[9], int(*fy)[9], int(*fz)[9])
    {
        gotoxy(hout, low_X + 2, low_Y);
        For(j, 1, 9)
            cout << "  " << char('a' + j - 1) << "   ";
        For(i, 1, 9)
        {
            gotoxy(hout, low_X, low_Y + 3 * (i - 1) + 2);
            cout << i << ' ';
        }
        Operation *p = NULL;
        bool wrong[3][N + 1] = { 0 };
        int res = check(map, wrong, fx, fy, fz);
        if (res)
        {
            output_img(map, fix, wrong, fx, fy, fz, { 0,0 });
            gotoxy(hout, 0, high_Y + 2);
            if (res == -1)
            {
                cout <<"The origin data has been completed." << endl;
                return;
            }
            cout << "There" << (res == 1 ? " is " : " are ") << res << " collision" << (res == 1 ? " " : "s ")
                << "in the origin data, please check it again !!" << endl;
            return;
        }
        Point P = { 0,0 };
        Point cur = { 1,1 };
        int x;
        while (1)
        {
            output_img(map, fix, wrong, fx, fy, fz, P);

            int f = 0;
            clear_img(high_Y + 2);
            if (mode == 2)
            {
                cout <<"请按行/列/值的顺序输入[例:5c6=第 5 行第 c 列填入 6],输入 bk 表示退回一次,输入 end 退出游戏: ";

                char Enter[256] = { "" };
                f = input(P, x, fix, Enter);
                clear_img(high_Y + 4);
                cout << "Your enter: " << Enter;
            }
            else if (mode == 4)
            {
                cout <<"请通过上下左右键控制光标,数字键进行填值,Backspace 表示回退一次,ESC 退出游戏";
                f = input_keyboard(cur, x, fix, map, wrong, fx, fy, fz);
                P = cur;
            }
            else
            {
                cout <<"请通过鼠标左键选择格子,数字键进行填值ESC表示撤销选择,Backspace 表示回退一次,鼠标右键退出游戏";
                f = input_mouse(P, x, fix, map, wrong, fx, fy, fz);
            }

            if (!f)
                continue;
            if (f == -1)
```

```
            {
                clear_img(high_Y + 3);
                cout << "You End the game" << endl;
                return;
            }
            if (f == 2)
            {
                if (p == NULL)
                {
                    clear_img(high_Y + 3);
                    cout << "This is the beginning" << endl;
                    continue;
                }
                cur = P = p->pos;
                x = p->past;
                Operation *tmp = p;
                p = p->pre;
                delete tmp;
            }
            else
            {
                Operation *tmp = new(nothrow) Operation;
                if (tmp == NULL)
                {
                    clear_img(high_Y + 3);
                    cout << "It's so bad that you've run out of all memory, Game Over" << endl;
                    Free_Operation(p);
                    return;
                }
                if (p == NULL)
                    tmp->pre = NULL;
                else
                    tmp->pre = p;
                p = tmp;
                p->pos = P;
                p->past = map[P.x][P.y];
            }
            if (map[P.x][P.y])
                --fx[P.x][map[P.x][P.y] - 1], --fy[P.y][map[P.x][P.y] - 1], --fz[rotate(P.x, P.y)][map[P.x][P.y] - 1];
            map[P.x][P.y] = x;
            if (x)
                ++fx[P.x][x - 1], ++fy[P.y][x - 1], ++fz[rotate(P.x, P.y)][x - 1];
            res = check(map, wrong, fx, fy, fz);
            clear_img(high_Y + 3);
            if (res > 0)
                cout << "There" << (res == 1 ? " is " : " are ") << res << " collision" << (res == 1 ? " " : "s ") << "inside" << endl;
            else if (!res)
                cout << "It's a legal operation" << endl;
            else
            {
                output_img(map, fix, wrong, fx, fy, fz, P);
                gotoxy(hout, 0, high_Y + 3);
                cout << "Congratulations! You finish the sudoku" << endl;
                Free_Operation(p);
                break;
            }
        }
        clear_img(high_Y + 5);
    }
```

**读入部分：**

```
    int input(Point &P, int &x, bool const(*fix)[N + 1], char *res)
    {
        P = { 0,0 };
        char op[256] = { "TaihouDaisuki" };
        cin.getline(op, 256);
        strcpy(res, op);
        if (strlen(op) == 2 && alpha_upper(op[0]) == 'B' && alpha_upper(op[1]) == 'K')
            return 2;
        if (strlen(op) != 3)
        {
            cout << "Input Error" << endl;
            return 0;
        }
        if (alpha_upper(op[0]) == 'E' && alpha_upper(op[1]) == 'N' && alpha_upper(op[2]) == 'D')
            return -1;
        if (op[0]<'1' || op[0]>'9')
        {
            cout << "line" << op[0] << "does not fall in [1,9]" << endl;
            return 0;
        }
        if (alpha_upper(op[1]) < 'A' || alpha_upper(op[1]) > 'I')
        {
            cout << "column" << op[1] << "does not fall in [a,i]" << endl;
            return 0;
        }
        if (op[2]<'0' || op[2]>'9')
        {
            cout << "value" << op[2] << "does
```

```
not fall in [0,9]" << endl;
                return 0;
            }
            P.x = op[0] - '0';
            P.y = alpha_upper(op[1]) - 'A' + 1;
            if (fix[P.x][P.y])
            {
                cout << "You cannot change this
value" << endl;
                return 0;
            }
            x = op[2] - '0';
            return 1;
        }
```

打印函数部分：

```
        void menu_output_img(File *f_head, const
int limit)
        {
            File *p = f_head;
            int i = 1;
            setconsoleborder(hout, 100, 40);
            gotoxy(hout, menu_X - 2, menu_Y - 2);
            cout << "数独样本文件：";
            gotoxy(hout, menu_X - 2, menu_Y - 1);
            cout << "┌──────────────────┐";
            do
            {
                gotoxy(hout, menu_X - 2, menu_Y
+ i - 1);
                cout << "│";
                cout << setiosflags(ios::left)
<< setw(20) << p->name;
                cout << "│" << endl;
                ++i;
                p = p->nxt;
            } while (i <= limit);
            gotoxy(hout, menu_X - 2, menu_Y +
limit);
            cout << "└──────────────────┘";
        }
        void output_img(int(*map)[N + 1], bool
const(*fix)[N + 1], bool const(*f)[N + 1], int
const(*fx)[9],    int    const(*fy)[9],    int
const(*fz)[9], Point P)
        {
            For(i, 1, 9) For(j, 1, 9)
            {
                int x = map[i][j] - 1;
                setcolor(hout, f[0][i] | f[1][j]
| f[2][rotate(i, j)] ? 7 : 0,
                    (i == P.x && j == P.y) ? 10 :
((fx[i][x] > 1) | (fy[j][x] > 1) | (fz[rotate(i,
j)][x] > 1) ? 12 : (fix[i][j] ? 11 : 14)));
                int tX = low_X + 2 + 6 * (j - 1);
                gotoxy(hout, tX, low_Y + 3 * (i
- 1) + 1);
                cout << "┌──┐";
                gotoxy(hout, tX, low_Y + 3 * (i
- 1) + 2);
                cout <<"│" << map[i][j] <<"│
";
                gotoxy(hout, tX, low_Y + 3 * (i
- 1) + 3);
                cout << "└──┘";
            }
            setcolor(hout, 0, 7);
        }
```

辅助函数部分：

```
        void clear_img(int line)
        {
            gotoxy(hout, 0, line);
            For(i, 1, 100)
                cout << " ";
            gotoxy(hout, 0, line);
        }
        int rotate(const int x, const int y)
        {
            return (x - 1) / 3 * 3 + (y - 1) / 3
+ 1;
        }
        int check(int(*map)[N + 1], bool(*f)[N +
1], int const(*fx)[9], int const(*fy)[9], int
const(*fz)[9])
        {
            For(i, 1, N)
                f[0][i] = f[1][i] = f[2][i] = 0;

            For(i, 1, 9) For(k, 0, 8)
            {
                if (fx[i][k] > 1)
                    f[0][i] = 1;
                if (fy[i][k] > 1)
                    f[1][i] = 1;
                if (fz[i][k] > 1)
                    f[2][i] = 1;
            }
            int cnt_error = 0;
            For(i, 1, N)
                cnt_error += f[0][i] + f[1][i] +
f[2][i];
            if (cnt_error > 0)
                return cnt_error;
            For(i, 1, N) For(j, 1, N)
                if (!map[i][j])
                    return 0;
            return -1;
        }
```

文件操作部分：

```
        int get_file(File *&head, int &cnt)
        {
            File *tail = NULL;
            long fst_file;
```

```
        struct _finddata_t file_info;
        if          ((fst_file          =
_findfirst(".\\sudoku*.txt", &file_info)) ==
-1L)
        {
            cout << "No Files" << endl;
            return 0;
        }
        do
        {
            ++cnt;
            File *p = new(nothrow) File;
            if (p == NULL)
            {
                cout << "No Free Memory" <<
endl;
                return -1;
            }
            if (head == NULL)
                head = tail = p;
            else
                tail->nxt = p, p->pre = tail,
tail = p;
            p->name     =     new(nothrow)
char[strlen(file_info.name) + 1];
            if (p->name == NULL)
            {
                cout << "No Free Memory" <<
endl;
                return -1;
            }
            strcpy(p->name,
file_info.name);
            p->nxt = NULL;
        }    while    (!_findnext(fst_file,
&file_info));
        _findclose(fst_file);

        if (head != tail)
            head->pre = tail, tail->nxt =
head;
        else
            head->pre = tail->nxt = head;
        return 1;
    }
    void Free_file(File *&head)
    {
        File *p = head;
        do
        {
            File *tmp = p;
            p = p->nxt;
            delete[] tmp->name;
            delete tmp;
        } while (p != head);
    }
    void select_img(int(*map)[N + 1], File
*head, const int lim)
    {
        gotoxy(hout, menu_X, menu_Y);
        setcolor(hout, 7, 0);
        File *cur = head;
        int k = 1;
        cout << setiosflags(ios::left) <<
setw(20) << cur->name;

        int op;
        while ((op = _getch()) != 13)
        {
            if (op != 224)
                continue;
            op = _getch();
            if (op != 72 && op != 80)
                continue;
            if (op == 72)//up
                if (k == 1)
                {
                    cur = cur->pre;
                    File *p = cur;
                    setcolor(hout, 0, 7);
                    For(i, 1, lim)
                    {
                        gotoxy(hout,
menu_X, menu_Y + i - 1);
                        cout     <<     "
";
                        gotoxy(hout,
menu_X, menu_Y + i - 1);
                        cout          <<
setiosflags(ios::left) << setw(20) << p->name;
                        p = p->nxt;
                    }
                }
                else
                {
                    gotoxy(hout,   menu_X,
menu_Y + k - 1);
                    setcolor(hout, 0, 7);
                    cout               <<
setiosflags(ios::left)   <<   setw(20)   <<
cur->name;
                    --k;
                    cur = cur->pre;
                }
            if (op == 80)//down
                if (k == lim)
                {
                    cur = cur->nxt;
                    File *p = cur;
                    setcolor(hout, 0, 7);
                    opFor(i, lim, 1)
                    {
```

```
                            gotoxy(hout,
menu_X, menu_Y + i - 1);
                            cout     <<     "
";
                            gotoxy(hout,
menu_X, menu_Y + i - 1);
                            cout          <<
setiosflags(ios::left) << setw(20) << p->name;
                            p = p->pre;
                        }
                    }
                    else
                    {
                        gotoxy(hout,   menu_X,
menu_Y + k - 1);
                        setcolor(hout, 0, 7);
                        cout               <<
setiosflags(ios::left)   <<   setw(20)   <<
cur->name;
                        ++k;
                        cur = cur->nxt;
                    }
                gotoxy(hout, menu_X, menu_Y + k
- 1);
                setcolor(hout, 7, 0);
                cout << setiosflags(ios::left)
<< setw(20) << cur->name;
            }
            setcolor(hout, 0, 7);

            ifstream infile;
            infile.open(cur->name, ios::in);
            For(i, 1, N) For(j, 1, N)
                infile >> map[i][j];
            infile.close();
        }
```

**自动寻解部分：**

```
        bool dfs(int x, int y, int(*map)[N + 1],
bool   const(*fix)[N   +   1],   int(*fx)[9],
int(*fy)[9], int(*fz)[9])
        {
            static int step = 0;
            if (y > 9)
                y -= 9, ++x;
            if (x == 10)
                return 1;
            if (fix[x][y])
                return dfs(x, y + 1, map, fix, fx,
fy, fz);
            int z = rotate(x, y);
            For(i, 1, 9)
            {
                int num = i - 1;
                if (fx[x][num] | fy[y][num] |
fz[z][num])
                    continue;
                ++fx[x][num];
                ++fy[y][num];
                ++fz[z][num];
                map[x][y] = i;
                ++step;
                gotoxy(hout, 14, high_Y + 1);
                cout << step;

                output_img(map, fix, zeros, fx,
fy, fz, { x,y });
                if (dfs(x, y + 1, map, fix, fx,
fy, fz))
                    return 1;
                map[x][y] = 0;
                --fx[x][num];
                --fy[y][num];
                --fz[z][num];
            }
            output_img(map, fix, zeros, fx, fy,
fz, { 0,0 });
            return 0;
        }
        void work_auto(int(*map)[N + 1], bool
const(*fix)[N + 1], int(*fx)[9], int(*fy)[9],
int(*fz)[9])
        {
            bool wrong[3][N + 1] = { 0 };
            int res = check(map, wrong, fx, fy,
fz);

            gotoxy(hout, low_X + 2, low_Y);
            For(j, 1, 9)
                cout << "  " << char('a' + j - 1)
<< "   ";
            For(i, 1, 9)
            {
                gotoxy(hout, low_X, low_Y + 3 *
(i - 1) + 2);
                cout << i << ' ';
            }
            output_img(map, fix, wrong, fx, fy,
fz, { 0,0 });

            if (res)
            {
                gotoxy(hout, 0, high_Y + 2);
                if (res == -1)
                {
                    cout <<"The origin data has
been completed." << endl;
                    return;
                }
                cout << "There" << (res == 1 ? "
is " : " are ") << res << " collision" << (res
== 1 ? " " : "s ")
                    << "in the origin data,
```

```
please check it again !!" << endl;
                return;
          }

          clear_img(high_Y + 1);
          cout << "当前搜索次数: ";
          int ans = dfs(1, 1, map, fix, fx, fy,
fz);
          clear_img(high_Y + 2);
          if (ans)
               cout << "The sudoku has been
completed";
          else
               cout   <<   "The   sudoku   is
unsolvable";
          clear_img(high_Y + 4);
     }
```

装

订

线