装

订

线

综合题　　实验报告

合成十 消灭星星

俄罗斯方块

配置文件工具集

计2 1752528 朱世轩

完成时间 2018.5.10

1. 合成十与消灭星星游戏基于代码公用层面的整体设计思路

1.1.两个程序逻辑实现上的共同点和不同点

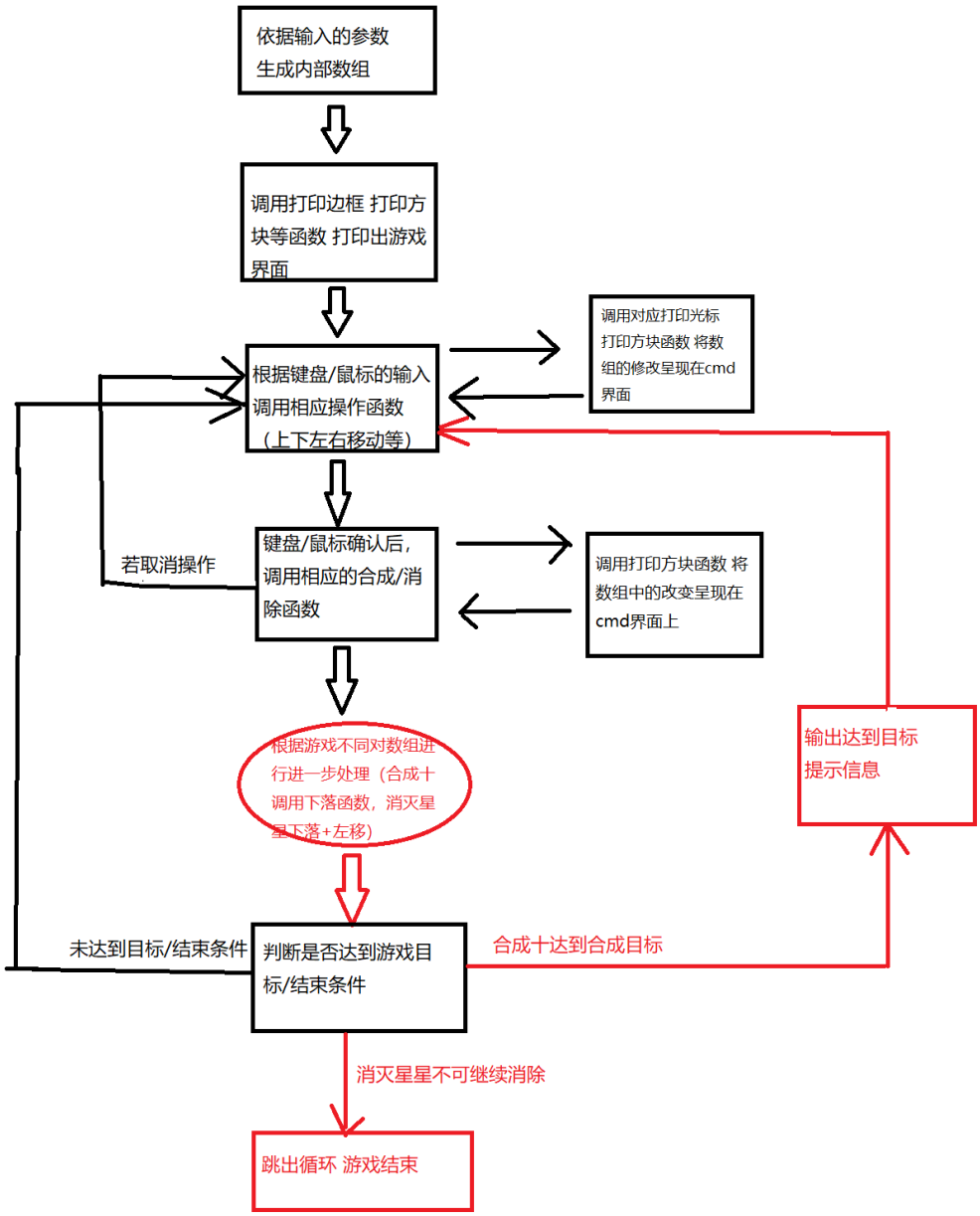
合成十和消灭星星在实现过程中十分相似，有许多共同点。
共同点：
两个游戏实现原理都是采用二维数组来存放每个数字/星星方块的信息，用不同数字代表每个方块的不同状态，通过维护数组来实现游戏中的各种操作；
两个游戏在规则上较相似，都是合并/消除相同状态（数组对应位置值相同）的方块；
在具体实现中，两个游戏都需要实现在数组的值改变后，在CMD界面内实现对应位置消除后下落等效果；
不同点：
合成十的内部数组中存放的是对应方块的数字，而消灭星星是代表对应方块颜色；
合成十消除后需要按概率随机生成新值，而消灭星星消除后直接在对应位置置空值，需要考虑非法位置的处理；
具体实现中，合成十只需要实现下落效果，而消灭星星需要实现下落和整列移动。

1.2.两个程序提取公共代码的原则和基本规则

合成十和消灭星星在实现思路的相似性使得它们可以提取大量公共代码（公共函数形式）
提取公共代码的原则：
代码重复度较高（50%以上），两个程序实现时该部分函数功能几乎一致或相似度较高；
通用性较强，不仅能在合成十和消灭星星游戏中使用，可能能在其他程序中发挥作用；
提取的基本规则：
在合成十的代码中寻找可能能在消灭星星游戏及其他游戏中使用的函数或尚未提取成函数的被部分，进行改进或提取；
修改时要同时考虑两个游戏需要使用的参数，以及将来可能需要的扩充；

1.3.两个程序完整版实现逻辑的流程框图

合成十及消灭星星实现逻辑流程图



装 订 线

1.4. 公共函数提取示例

公共函数提取我做的并不好，并没有提取得比较满意的函数，所以最后一个大作业的读取配置文件我也没能实现QAQ

第一个是在任意位置打印方块的函数，我觉得提取得很菜，很大程度就是根据不同游戏选项进行不同输出，没有考虑到之后可能需要的修改，后来意识到这一点，但由于时间不充裕和我的懒惰一直没有修改。

```
void free_blocks(int nx, int ny, int z, int game)
{
    int color;
    if (game == 0)
        color = z % 15;
    else if (game == 1)
        color = z % 15 + 8;
    else if (game == 2)
        color = 14;
    if (z == 0 || (game == 2 && (z == 2 || z == 4)))
    {
        showstr(nx, ny, "      ", 15, 15);
        showstr(nx, ny + 1, "      ", 15, 15);
        showstr(nx, ny + 2, "      ", 15, 15);
    }
    else if (z < 10 && z != 0 && game != 2)
    {
        showstr(nx, ny, "┌──┐", color, 0);
        showstr(nx, ny + 1, "│", color, 0);
        if (game == 0)
            cout << " " << z;
        else if (game == 1)
            cout << "★";
        showstr(nx + 4, ny + 1, "│", color, 0);
        showstr(nx, ny + 2, "└──┘", color, 0);
    }
    else if (game = 2)
    {
        showstr(nx, ny, "┌──┐", color, 0);
        showstr(nx, ny + 1, "│", color, 0);
        cout << "★";
        showstr(nx + 4, ny + 1, "│", color, 0);
        showstr(nx, ny + 2, "└──┘", color, 0);
    }
    else if (z >= 10)
    {
        showstr(nx, ny, "┌──┐", color, 0);
        showstr(nx, ny + 1, "│", color, 0);
        cout << z;
        showstr(nx + 4, ny + 1, "│", color, 0);
        showstr(nx, ny + 2, "└──┘", color, 0);
    }
```

装 订 线

```
}
```

第二个是map中的下落函数，因为此函数在两个程序中相似度极高，不需要做过多修改适应，但我没有考虑到有无边框，方块宽度等可能会修改的参数，没有考虑到之后可能修改的情况，直接写死了，应该用变量代替函数中的表示位置坐标的常数（还是自己懒）

```
void mapfall(int map[YL][XL], int & row, int & col, int nr, int game)
{
    setcursor(CURSOR_INVISIBLE);
    int nx;
    int ny;
    int t;
    int flag = 0;
    for (int i = nr; i > 1; i--)
        for (int j = 1; j < XL - 1; j++)
        {
            t = map[i][j];
            map[i][j] = map[i - 1][j];
            map[i - 1][j] = t;
        }
    for (int i = YL - 2; i > 0; i--)
        for (int j = XL - 2; j > 0; j--)
        {
            nx = (j - 1) * 6 + 2;
            ny = (i - 1) * 3 + 1;
            free_blocks(nx, ny, map[i][j], game);
        }

}
```

1.5.提取公共函数的心得体会和经验教训

因为我的公共函数提取不很成功，所以心得体会可能不恰当，就说说经验教训吧。

•我是在写了一部分消灭星星的实现后，才想起提取公共函数，一开始就违背了提取的基本规则，提取的过程也就比较坎坷；

•提取公共函数时，我想着把所有常数参数都转换为宏定义或变量再进行提取实在太麻烦，就采取了简单地将两者相同部分合并，不同部分判断执行这样看似“简单”的方法，实际这样的提取方式不但实际作用不大，当参数出问题时还很难查错误在哪里，要修改时也是十分繁琐，反而浪费了大量时间。。。。。所以想偷懒果然不行，越是想要减少工作量可能要做的就越多，还达不到预期的效果；

•提取公共函数时，不能急着实现，最好先构思完善再开始修改，我在修改时就出现了写到一半突然发现需要哪个参数，然后又回头在函数声明，定义的参数表里修改的情况，费时费力。

2. 数字俄罗斯方块的设计与实现

2.1.题目及基本要求

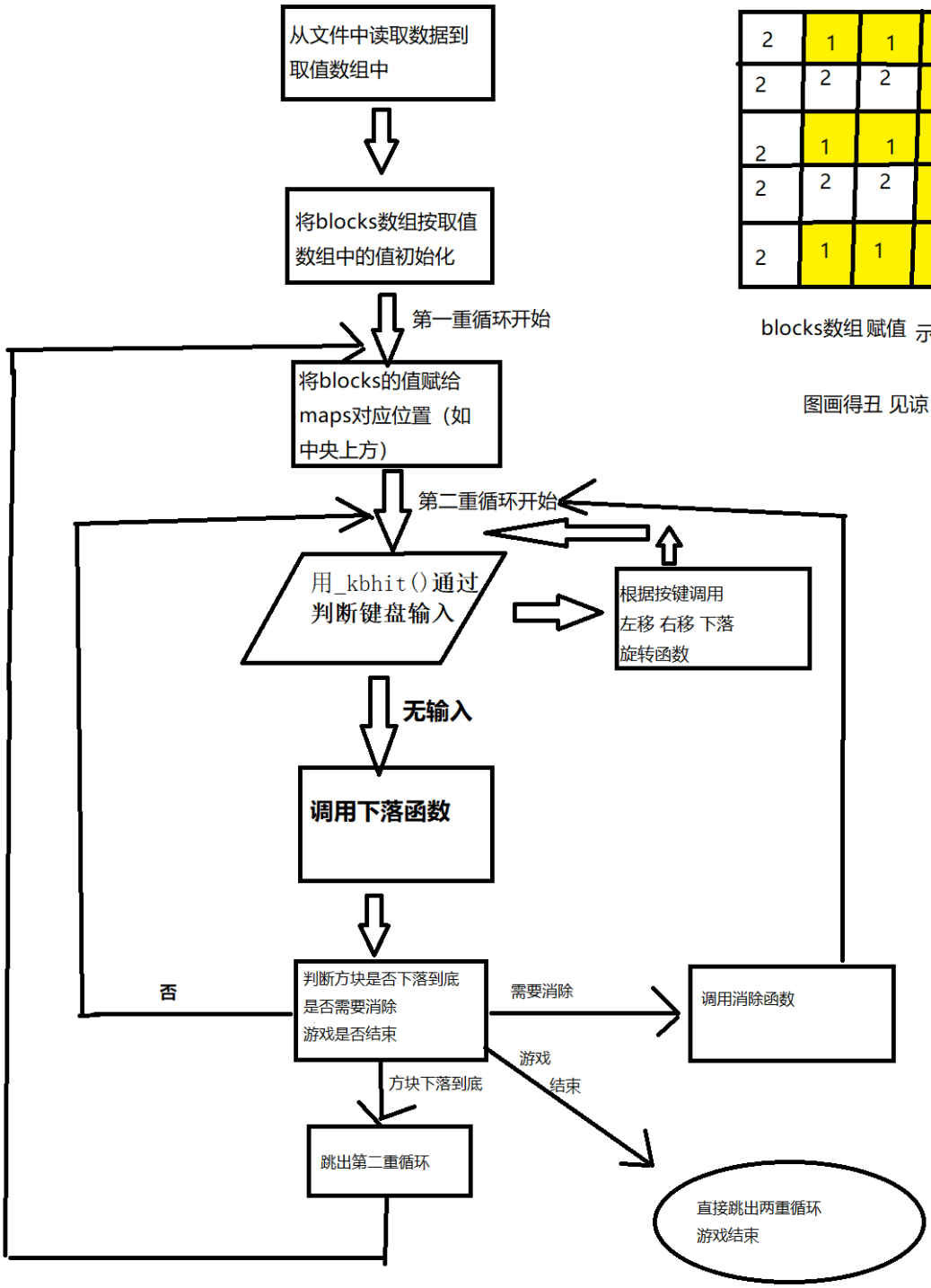
数字俄罗斯方块题目基本要求：
仿照网站上的数字俄罗斯方块游戏，实现
从文件中读取出现的数字顺序，非数字字符忽略，之后使用随机值；
实现3*5的0-9数字显示和自动下落；
键盘方向键控制数字方块的加速下落，左移右移以及旋转；
当某行被方块填满时消除该行；

2.2.整体设计思路

先从文件中读取数据，存入足够大数组中，剩余空间填充随机值；
采用map数组记录游戏区域状态，用blocks数组（5*5）记录当前活跃状态的数字方块（考虑到旋转空间），使用0代表map中未填充空间，1代表blocks中有方块的位置，2表示没有方块的位置，3表示固定在maps中数字方块中有小方块的位置，4代表固定在maps中的数字方块没有小方块的位置，以这四种状态为基础进行处理；
进入循环
用_kbhit()通过判断键盘输入调用不同函数来实现加速下落，左移，右移，旋转操作，没有输入则调用下落函数使数字方块下落；
当数字方块下落到无法继续下落时，判断有无行被填满，按结果执行相应的消除下落操作
到方块触到地图上边界时跳出循环结束游戏

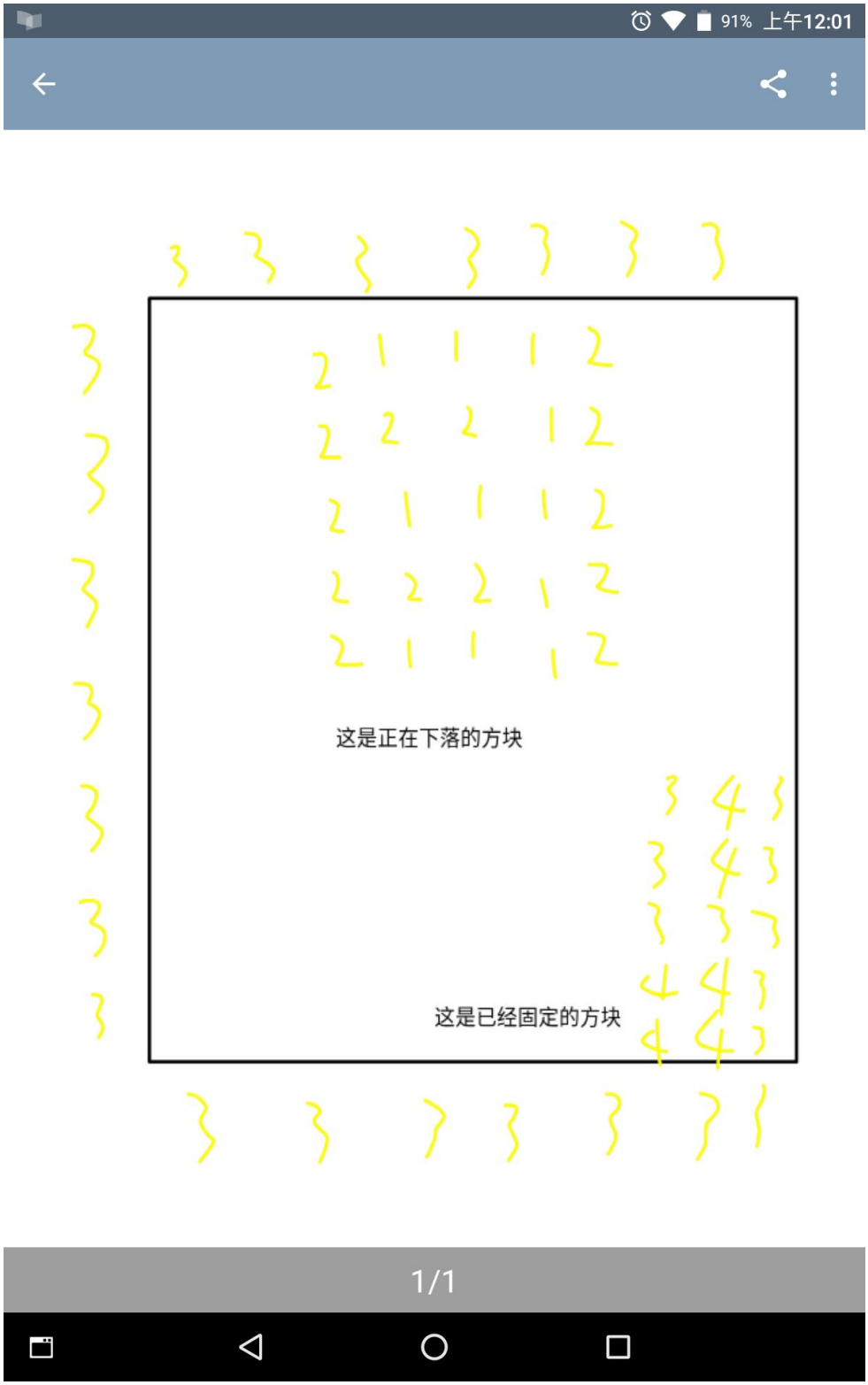
2.3. 主要功能的实现

主要结构流程图：



然后是灵魂画师展示内部数组处理原理

数组边界置3，内部其余地方置0



装 订 线

2.4. 调试过程中遇到的问题及解决

实现过程中我遇到了许多问题；

1. 由于刚开始没有考虑周全，只了用3个数字表示数字方块的状态，由于我的左右移动和下落函数是基于判断方块内的数字和周围数字的关系，导致数字方块在触底后，再次调用下落函数时会出现非空白处继续下落的情况；我最终采用使用另外两个数字单独表示下落完毕固定后的方块状态，解决了这个问题
2. 在实现旋转函数时，由于赋值位置的错误判断出现了旋转错位的错误；我通过画出内部数组各项位置，然后打印出内部数组，与预期结果进行对比，最终找出了错误

3. 数字俄罗斯方块和前两个大作业的代码共用

3.1 基本相同部分

基本相同不需要做太多修改的部分只有数据的输入，打印方块等函数基本不需要修改

```
void free_blocks(int nx, int ny, int z, int game)   //任意位置打印方块函数
{
    int color;
    if (game == 0)
        color = z % 15;
    else if (game == 1)
        color = z % 15 + 8;
    else if (game == 2)
        color = 14;
    if (z == 0 || (game == 2 && (z == 2 || z == 4)))
    {
        showstr(nx, ny, "      ", 15, 15);
        showstr(nx, ny + 1, "      ", 15, 15);
        showstr(nx, ny + 2, "      ", 15, 15);
    }
    else if (z < 10 && z != 0 && game != 2)
    {
        showstr(nx, ny, "╔══╗", color, 0);
        showstr(nx, ny + 1, "║", color, 0);
        if (game == 0)
            cout << " " << z;
        else if (game == 1)
            cout << "★";
        showstr(nx + 4, ny + 1, "║", color, 0);
        showstr(nx, ny + 2, "╚══╝", color, 0);
    }
    else if (game = 2)
    {
        showstr(nx, ny, "┌──┐", color, 0);
        showstr(nx, ny + 1, "│", color, 0);
        cout << "★";
        showstr(nx + 4, ny + 1, "│", color, 0);
        showstr(nx, ny + 2, "└──┘", color, 0);
    }
    else if (z >= 10)
    {
```

```
            showstr(nx, ny, "┌──┐", color, 0);
            showstr(nx, ny + 1, "│", color, 0);
            cout << z;
            showstr(nx + 4, ny + 1, "│", color, 0);
            showstr(nx, ny + 2, "└──┘", color, 0);
        }
    }
    void CIN(int*s, int a, int b)              //输入检查  报错  s为输入数据 ab是正确范围
    {
        while (1)
        {
            int x, y;
            getxy(x, y);
            cin >> *s;
            if (!cin.good())
            {
                cout << "输入非法请重新输入：";
                cin.clear();
                cin.ignore(1024, '\n');
                gotoxy(x, y);
                continue;
            }
            else if (*s > b || *s < a)
            {
                cin.clear();
                cin.ignore(1024, '\n');
                cout << "输入非法请重新输入：";
                gotoxy(x, y);
                continue;
            }
            else
                break;
        }
        cin.ignore(1024, '\n');
    }
```

装 订 线

3.2 需要改进或扩充的部分

下落函数不再是所有非0方块下落，而是一部分方块下落

```
void fall(int map[YL][XL], int blocks[5][3], int game, int & row, int & col, int & endflag)
{
    setcursor(CURSOR_INVISIBLE);
    int nx;
    int ny;
    int t;
    int flag = 0;
    for (int i = YL - 1; i > 0; i--)
        for (int j = XL - 1; j > 0; j--)
            if (map[i][j] == 3 && map[i - 1][j] == 1)
            {
                endflag = 1;
                break;
            }
    for (int i = YL - 1; i > 0; i--)
```

```
    {
        for (int j = XL - 1; j > 0; j--)
        {

            if ((map[i][j] == 0 || map[i][j] == 4) && (map[i - 1][j] == 1 || map[i - 1][j] ==
2))
            {
                t = map[i][j];
                map[i][j] = map[i - 1][j];
                map[i - 1][j] = t;
            }
            if (map[i][j] == 3 && map[i - 1][j] == 2)
            {
                map[i - 1][j] = 0;
            }
        }
        if (endflag == 1)
            break;
    }
    for (int i = YL - 2; i > 0; i--)
        for (int j = XL - 2; j > 0; j--)
        {
            nx = (j - 1) * 6 + 2;
            ny = (i - 1) * 3 + 1;
            free_blocks(nx, ny, map[i][j], game);
        }

    for (int i = 1; i < YL; i++)
        for (int j = 1; j < XL; j++)
        {
            if ((map[i][j] == 1 || map[i][j] == 2) && flag == 0)
            {
                row = i;
                col = j;
                flag = 1;
            }
        }

}
```

3.3 不同的部分

以我的程序实现方法，俄罗斯方块和前两个作业还是有着较大不同

比如数字方块的左移右移，和之前光标的移动，或是整行整列的移动还是有着较大不同；

而且还多了消除整行的操作和旋转操作

如下是实现旋转的函数。。。还挺长的 就省略了一些重复部分

```
int spin(int map[YL][XL], int blocks[5][3], int game, int & row, int & col, int & spinflag)
{
    int nx;
    int ny;
    int r = row;
    int c = col;
    int flag = 0;
```

```
gotoxy(56, 62);
if (spinflag == 1 || spinflag == 3)        //横向
{
    r = r - 1;
    for (int i = r; i < r + 5; i++)
    {
        for (int j = c; j < c + 5; j++)
        {
            if (map[i][j] == 3)
                return -1;
        }
    }
}
if (spinflag == 0 || spinflag == 2)        //竖向
{
    c = c - 1;
    for (int i = r; i < r + 5; i++)
    {
        for (int j = c; j < c + 5; j++)
        {
            if (map[i][j] == 3)
                return -1;
        }
    }
}
if (spinflag == 0)
{
    col = col - 1;
    for (int i = row; i < row + 5; i++)
        for (int j = col; j < col + 5; j++)
            map[i][j] = 0;
    row = row + 1;
    for (int j = col; j < col + 5; j++)
        for (int i = row; i < row + 3; i++)
        {
            map[i][j] = blocks[j - col][i - row];
        }
    for (int i = 1; i < YL - 1; i++)
        for (int j = 1; j < XL - 1; j++)
        {
            nx = (j - 1) * 6 + 2;
            ny = (i - 1) * 3 + 1;
            free_blocks(nx, ny, map[i][j], game);
        }
}
if (spinflag == 1)
{
    row = row - 1;
    for (int i = row; i < row + 5; i++)
        for (int j = col; j < col + 5; j++)
            map[i][j] = 0;
    col = col + 1;
    for (int i = row; i < row + 5; i++)
        for (int j = col; j < col + 3; j++)
```

装 订 线

```
                {
                    map[i][j] = blocks[-i + row + 4][-j + col + 2];
                }
            for (int i = 1; i < YL - 1; i++)
                for (int j = 1; j < XL - 1; j++)
                {
                    nx = (j - 1) * 6 + 2;
                    ny = (i - 1) * 3 + 1;
                    free_blocks(nx, ny, map[i][j], game);
                }
        }
        if (spinflag == 2)
        {
            ......
        }
        if (spinflag == 3)
        {
            .......
        }

        if (flag == 0)
            spinflag = spinflag + 1;
        return 0;
}
```

而这是实现单向移动的函数

```
void turn_left(int map[YL][XL], int blocks[5][3], int game, int & row, int & col)
{
    setcursor(CURSOR_INVISIBLE);
    int nx;
    int ny;
    int t;
    int flag = 0;
    int eflag = 0;
    for (int j = 0; j < XL - 1; j++)
        for (int i = 0; i < YL; i++)
        {
            if (map[i][j] == 3 && map[i][j + 1] == 1)
            {
                eflag = 1;
            }
        }
    if (eflag == 0)
    {
        for (int i = 1; i < YL; i++)
            for (int j = 1; j < XL; j++)
            {
                if ((map[i][j] == 0 || map[i][j] == 4) && (map[i][j + 1] == 1 || map[i][j + 1] ==
2))
                {
                    t = map[i][j];
                    map[i][j] = map[i][j + 1];
                    map[i][j + 1] = t;
                }
                if (map[i][j] == 2 && map[i][j - 1] == 3)
                {
```

```
                    map[i][j] = 4;
                }
            }

        for (int i = 1; i < YL - 1; i++)
            for (int j = 1; j < XL - 1; j++)
            {
                nx = (j - 1) * 6 + 2;
                ny = (i - 1) * 3 + 1;
                free_blocks(nx, ny, map[i][j], game);
            }
        for (int i = 0; i < YL; i++)
            for (int j = 0; j < XL; j++)
            {
                if ((map[i][j] == 1 || map[i][j] == 2) && flag == 0)
                {
                    row = i;
                    col = j;
                    flag = 1;
                }
            }
    }
    else
        eflag = 0;
}
```

4. 配置文件工具函数集的设计与实现

4.1 题目及基本要求

写一组工具函数集，要求能够实现对具有一定格式的配置文件的读取，修改操作

具体要求的操作如下：

打开，关闭配置文件；

添加，删除配置组；

添加，删除，更新配置项；

读取指定配置项的值

4.2 整体设计思路

打开，关闭配置文件：

先用追加写方式打开，再关闭，然后用二进制读写方式打开，实现没有文件就创建，有也不会改变原有内容；

添加，删除配置组；

找到需要删除的配置组名以及下一个配置组名所在位置，将下一个配置组名后数据全部存储，再覆盖到要删除组名的位置，然后缩小文件大小；

添加，删除，更新配置项；

类似添加删除配置组

读取指定配置项的值：

找到配置项，去掉注释及等号前部分，按类型读取值

4.3 C和C++方式的差别

1.文件操作时时有差异，c++采用fstream流对象操作，有读写文件指针；而c通过函数操作，只有一个文件指针；c++文件指针越界或读到文件尾后需要清除错误状态才能继续使用，而c可以直接移动

2.读取文件中一行时，c++可以使用流成员函数getline，而c需要使用gets函数；

3.输出信息时有差别；

4.4 调试过程中遇到的问题

1.文件指针定位错误导致函数工作不正常 解决方法：逐个分析，打印位置，比对

2.没有考虑到各种特殊情况导致的问题（如要删除的组/项在最后一行） 解决方法：加入特殊处理

装

5. 配置文件工具函数集用于前次综合题

这一部分由于时间原因和前两次公共函数提取不成功 没有做出来QAQ 所以我也编不下去了

订

6. 完成三次作业的心得体会

通过这三次作业，虽然我做的不好，但在放弃配置文件工具集用于前次大作业时，我认识到了将工具函数提取出来，以及考虑函数通用性，程序可读性的重要性（找bug时看不懂前面写的）；如果现在让我完成某个具体项目，我应该会有意识地提取工具函数，但考虑可能还是会不周全

注意排版格式，对齐方式，可以采用两列排版，正文采用宋体，小五，行距为单倍行距

线

附件：俄罗斯方块源程序

Common_base 中：

```
void fall(int map[YL][XL], int blocks[5][3], int game, int & row, int & col, int & endflag)
{
    setcursor(CURSOR_INVISIBLE);
    int nx;
    int ny;
    int t;
    int flag = 0;
    for (int i = YL - 1; i > 0; i--)
        for (int j = XL - 1; j > 0; j--)
            if (map[i][j] == 3 && map[i - 1][j] == 1)
            {
                endflag = 1;
                break;
            }
    for (int i = YL - 1; i > 0; i--)
    {
        for (int j = XL - 1; j > 0; j--)
        {

            if ((map[i][j] == 0 || map[i][j] == 4) && (map[i - 1][j] == 1 || map[i - 1][j] ==
2))
            {
                t = map[i][j];
                map[i][j] = map[i - 1][j];
                map[i - 1][j] = t;
            }
            if (map[i][j] == 3 && map[i - 1][j] == 2)
            {
                map[i - 1][j] = 0;
            }
        }
        if (endflag == 1)
            break;
    }
    for (int i = YL - 2; i > 0; i--)
        for (int j = XL - 2; j > 0; j--)
        {
            nx = (j - 1) * 6 + 2;
            ny = (i - 1) * 3 + 1;
            free_blocks(nx, ny, map[i][j], game);
        }

    for (int i = 1; i < YL; i++)
        for (int j = 1; j < XL; j++)
        {
            if ((map[i][j] == 1 || map[i][j] == 2) && flag == 0)
            {
                row = i;
                col = j;
                flag = 1;
            }
```

装 订 线

```
        }

}
void turn_right(int map[YL][XL], int blocks[5][3], int game, int & row, int & col)
{
    setcursor(CURSOR_INVISIBLE);
    int nx;
    int ny;
    int t;
    int flag = 0;
    int eflag = 0;
    for (int j = XL - 1; j >= 0; j--)
        for (int i = YL - 1; i >= 0; i--)
        {
            if (map[i][j] == 1 && map[i][j + 1] == 3)
            {
                eflag = 1;
            }
        }
    if (eflag == 0)
    {
        for (int j = XL; j > 0; j--)
            for (int i = YL; i > 0; i--)
            {
                if ((map[i][j] == 0 || map[i][j] == 4) && (map[i][j - 1] == 1 || map[i][j -
1] == 2))
                {
                    t = map[i][j];
                    map[i][j] = map[i][j - 1];
                    map[i][j - 1] = t;
                }
                if (map[i][j] == 3 && map[i][j - 1] == 2)
                {
                    map[i][j - 1] = 4;
                }

            }

        for (int j = XL - 2; j > 0; j--)
            for (int i = YL - 2; i > 0; i--)
            {
                nx = (j - 1) * 6 + 2;
                ny = (i - 1) * 3 + 1;
                free_blocks(nx, ny, map[i][j], game);
            }
        for (int i = 0; i < YL; i++)
            for (int j = 0; j < XL; j++)
            {
                if ((map[i][j] == 1 || map[i][j] == 2) && flag == 0)
                {
                    row = i;
                    col = j;
                    flag = 1;
                }
```

```
            }
        }
        eflag = 0;
    }

    void turn_left(int map[YL][XL], int blocks[5][3], int game, int & row, int & col)
    {
        setcursor(CURSOR_INVISIBLE);
        int nx;
        int ny;
        int t;
        int flag = 0;
        int eflag = 0;
        for (int j = 0; j < XL - 1; j++)
            for (int i = 0; i < YL; i++)
            {
                if (map[i][j] == 3 && map[i][j + 1] == 1)
                {
                    eflag = 1;
                }
            }
        if (eflag == 0)
        {
            for (int i = 1; i < YL; i++)
                for (int j = 1; j < XL; j++)
                {
                    if ((map[i][j] == 0 || map[i][j] == 4) && (map[i][j + 1] == 1 || map[i][j +
1] == 2))
                    {
                        t = map[i][j];
                        map[i][j] = map[i][j + 1];
                        map[i][j + 1] = t;
                    }
                    if (map[i][j] == 2 && map[i][j - 1] == 3)
                    {
                        map[i][j] = 4;
                    }
                }

            for (int i = 1; i < YL - 1; i++)
                for (int j = 1; j < XL - 1; j++)
                {
                    nx = (j - 1) * 6 + 2;
                    ny = (i - 1) * 3 + 1;
                    free_blocks(nx, ny, map[i][j], game);
                }
            for (int i = 0; i < YL; i++)
                for (int j = 0; j < XL; j++)
                {
                    if ((map[i][j] == 1 || map[i][j] == 2) && flag == 0)
                    {
                        row = i;
                        col = j;
                        flag = 1;
```

装 订 线

```
                }
            }
        }
        else
            eflag = 0;
    }

    int spin(int map[YL][XL], int blocks[5][3], int game, int & row, int & col, int & spinflag)
    {
        int nx;
        int ny;
        int r = row;
        int c = col;
        int flag = 0;
        gotoxy(56, 62);
        if (spinflag == 1 || spinflag == 3)        //横向
        {
            r = r - 1;
            for (int i = r; i < r + 5; i++)
            {
                for (int j = c; j < c + 5; j++)
                {
                    if (map[i][j] == 3)
                        return -1;
                }
            }
        }
        if (spinflag == 0 || spinflag == 2)        //竖向
        {
            c = c - 1;
            for (int i = r; i < r + 5; i++)
            {
                for (int j = c; j < c + 5; j++)
                {
                    if (map[i][j] == 3)
                        return -1;
                }
            }
        }
        if (spinflag == 0)
        {
            col = col - 1;
            for (int i = row; i < row + 5; i++)
                for (int j = col; j < col + 5; j++)
                    map[i][j] = 0;
            row = row + 1;
            for (int j = col; j < col + 5; j++)
                for (int i = row; i < row + 3; i++)
                {
                    map[i][j] = blocks[j - col][i - row];
                }
            for (int i = 1; i < YL - 1; i++)
                for (int j = 1; j < XL - 1; j++)
                {
```

装 订 线

```
                nx = (j - 1) * 6 + 2;
                ny = (i - 1) * 3 + 1;
                free_blocks(nx, ny, map[i][j], game);
            }
        }
        if (spinflag == 1)
        {
            row = row - 1;
            for (int i = row; i < row + 5; i++)
                for (int j = col; j < col + 5; j++)
                    map[i][j] = 0;
            col = col + 1;
            for (int i = row; i < row + 5; i++)
                for (int j = col; j < col + 3; j++)
                {
                    map[i][j] = blocks[-i + row + 4][-j + col + 2];
                }
            for (int i = 1; i < YL - 1; i++)
                for (int j = 1; j < XL - 1; j++)
                {
                    nx = (j - 1) * 6 + 2;
                    ny = (i - 1) * 3 + 1;
                    free_blocks(nx, ny, map[i][j], game);
                }
        }
        if (spinflag == 2)
        {
            col = col - 1;
            for (int i = row; i < row + 5; i++)
                for (int j = col; j < col + 5; j++)
                    map[i][j] = 0;
            row = row + 1;
            for (int j = col; j < col + 5; j++)
                for (int i = row; i < row + 3; i++)
                {
                    map[i][j] = blocks[j - col][2 - i + row];
                }
            for (int i = 1; i < YL - 1; i++)
                for (int j = 1; j < XL - 1; j++)
                {
                    nx = (j - 1) * 6 + 2;
                    ny = (i - 1) * 3 + 1;
                    free_blocks(nx, ny, map[i][j], game);
                }
        }
        if (spinflag == 3)
        {
            row = row - 1;
            for (int i = row; i < row + 5; i++)
                for (int j = col; j < col + 5; j++)
                    map[i][j] = 0;
            col = col + 1;
            for (int j = col; j < col + 3; j++)
                for (int i = row; i < row + 5; i++)
```

装 订 线

```
            {
                map[i][j] = blocks[i - row][j - col];
            }
        for (int i = 1; i < YL - 1; i++)
            for (int j = 1; j < XL - 1; j++)
            {
                nx = (j - 1) * 6 + 2;
                ny = (i - 1) * 3 + 1;
                free_blocks(nx, ny, map[i][j], game);
            }
        spinflag = 0;
        flag = 1;
    }

    if (flag == 0)
        spinflag = spinflag + 1;
    return 0;
}
```

Free_blocks 函数上有 省略

Base 中

```
char tetris(char choose, int game, char & last)
{
    int X = 0;
    int Y = 0;
    int endflag = 0;
    int score = 0;
    int numnow;
    int spinflag = 0;
    int count = 0;
    int row, col;                          //记录数字方块第一个小方块的位置
    char c = '0';
    int map[YL][XL];
    int blocks[5][3] = { 0 };

    if (choose == 'Y')
    {
        for (int i = 0; i < YL; i++)                    //给范围内数组赋初值
            for (int j = 0; j < XL; j++)
            {
                map[i][j] = 3;
            }
        for (int i = 1; i < YL - 1; i++)                    //给范围内数组赋初值
            for (int j = 1; j < XL - 1; j++)
            {
                map[i][j] = 0;
            }

        setconsoleborder((XL - 1) * 6 + 30, (YL - 1) * 3 + 50);
        free_border(YL - 2, XL - 2, X, Y, 0, 0);
        tips(YL - 2, XL - 2, 0, 0, (YL - 2) * 3, 13, 0);
        for (int i = 1; i < YL - 1; i++)
            for (int j = 1; j < XL - 1; j++)
```

```
        {
            X = (j - 1) * 6 + 2;
            Y = (i - 1) * 3 + 1;
            free_blocks(X, Y, map[i][j], game);
        }


    char i, hc;
    int a[30];
    ifstream fin; //fin 为变量名
    fin.open("tetris.dat", ios::in);//打开文件
    srand((unsigned)time(NULL));
    for (i = 0; i < 30; i++)
    {
        a[i] = rand() % 10;
    }
    if (fin.is_open())
    {

        for (i = 0; i < 30; )
        {
            fin >> hc;
            if (hc >= '0'&&hc <= '9')
            {
                a[i] = hc - '0';
                i++;
            }
        }
        fin.close(); //关闭文件
    }
    for (i = 0; i < 30; i++)
    {
        /*setcolor(6, 5);
        cout << a[i];*/
        //test
        numnow = a[i];
        numberblocks(blocks, a[i]);
        changearray(blocks, map, 0);
        tips(YL - 2, XL - 2, a[i], score, (YL - 2) * 3, 13, 1);
        while (1)
        {
            int keycode;
            if (_kbhit())
            {
                keycode = _getch();
                if (keycode == '\120') //下键
                {
                    fall(map, blocks, game, row, col, endflag);
                    count = count + 1;
                }
                if (keycode == '\113') //左键
                {
                    turn_left(map, blocks, game, row, col);
                }
```

装 订 线

```
        if (keycode == '\115') //右键
        {
            turn_right(map, blocks, game, row, col);
        }
        if (keycode == '\110') //下键
        {
            spin(map, blocks, game, row, col, spinflag);
        }

        /*gotoxy(0,62);
        setcolor(6, 5);
        for (int i = 0; i < YL; i++)           //test
        {
            for (int j = 0; j < XL; j++)
                cout << map[i][j] << " ";
            cout << endl;
        }
        Sleep(200);
        */
        continue;
    }
    Sleep(240);
    fall(map, blocks, game, row, col, endflag);
    count = count + 1;
    /*gotoxy(0, 68);
    setcolor(6, 5);
    for (int i = 0; i < YL; i++)           //test   测试内部数组
    {
        for (int j = 0; j < XL; j++)
            cout << map[i][j] << " ";
        cout << endl;
    }
    */


    if (count >= YL - 3 || endflag == 1)
    {
        for (int i = 1; i < YL - 1; i++)
            for (int j = 1; j < XL - 1; j++)
            {
                if (map[i][j] == 1)
                    map[i][j] = 3;
                if (map[i][j] == 2)
                    map[i][j] = 4;
            }
        count = 0;
        endflag = 0;
        spinflag = 0;
        for (int i = 1; i < YL - 1; i++)                        //消除一行
        {
            int xc = 0;
            for (int j = 1; j < XL - 1; j++)
            {
```

```
                    if (map[i][j] == 3)
                        xc = xc + 1;
                }
                if (xc == XL - 2)
                {
                    for (int j = 1; j < XL - 1; j++)
                        map[i][j] = 0;
                    mapfall(map, row, col, i, game);
                    score = score + 10;
                    tips(YL - 2, XL - 2, numnow, score, (YL - 2) * 3, 13, 1);
                }

            }
            for (int i = 1; i < XL - 1; i++)                    //判断是否结束
            {
                if (map[1][i] == 3)
                {
                    endflag = 2;
                    tips(YL - 2, XL - 2, numnow, score, (YL - 2) * 3, 13, 2);
                }
            }
            break;
        }
    }
    if (endflag == 2)
    {
        last = _getch();
        break;
    }
}
}
return c;
}
```

根据数字初始化方块的函数省略

Main 中

```
int main()
{
    char choose;
    char last = '0';
    char menu[12][60] = {
        {"★数字俄罗斯方块★"},
        { "Y.开始游戏" },
        {"Q.退出"},
        { "[请选择A-Q] " },
    };
    int game = 2;
    choose = cd(menu, game);
    while (1)
    {
        setcolor(0, 7);
        tetris(choose, game, last);
        if (last == 'c')
```

```
            continue;
        else
            break;
    }
    return 0;
}
```

其余函数不重要 省略

装

订

线