装 订 线

实验报告

报告名称：cmd界面游戏大杂烩..

班级：计算机1班

学号：1651574

姓名：贾昊霖

完成日期：2018年5月15日

1. 题目及基本要求

1.1. 题目

cmd界面游戏大杂烩..

1.2. 基本要求

所有小题放在一个程序当中是真的爽。首先完成内部数组的显示，显示一系列消除-下落-填充-寻找提示等操作；然后伪图形界面中显示，用鼠标操作进行连续的游戏。除此之外，行数、列数要求可变。参数决定细节差异时，要用循环解决。并且，尽量做到高效，减少冗余代码。表达同一个对象的不同性质时，尽量用结构体struct整合，不要用多个对应变量。

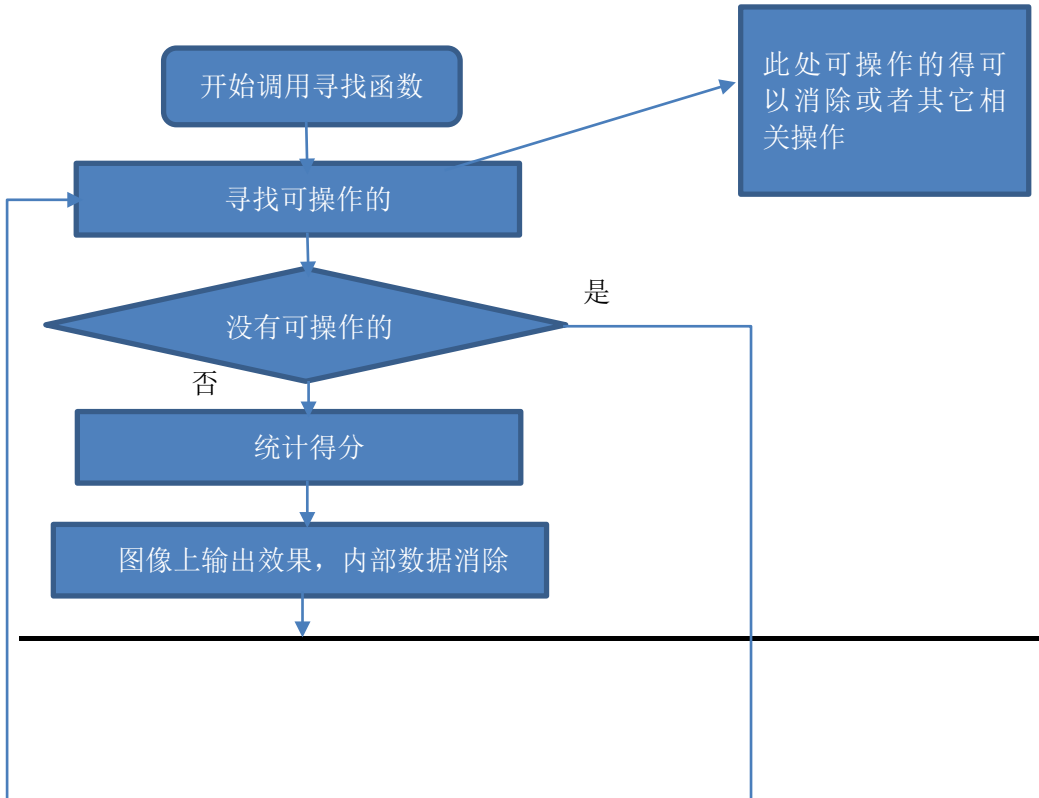
2. 报告内容

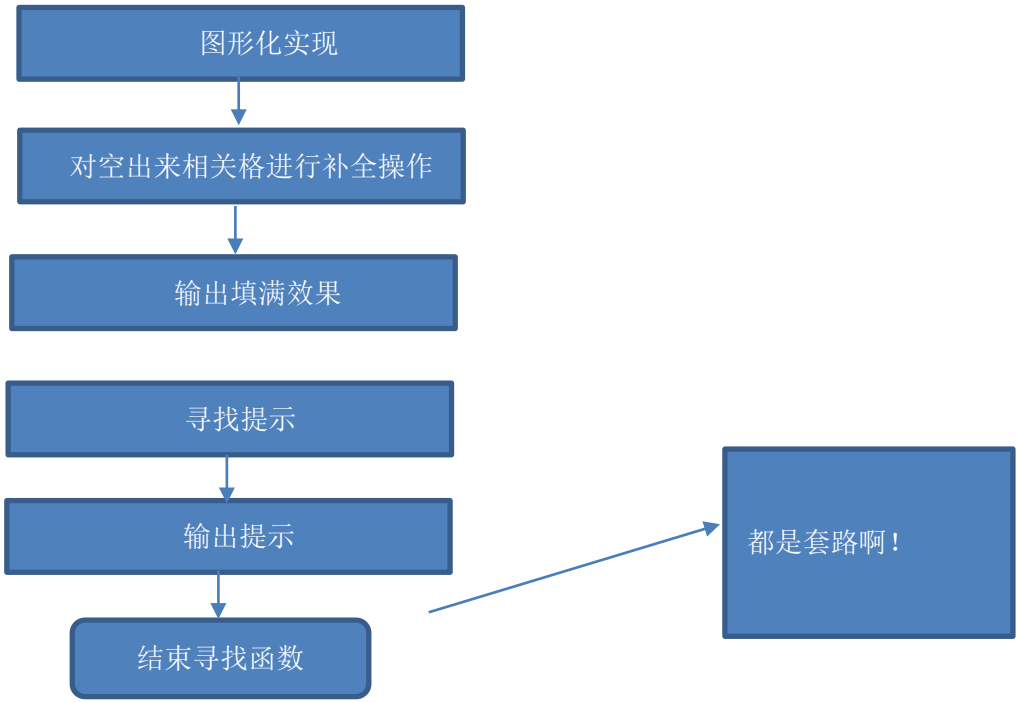
2.1. “合成十”及“消灭星星”游戏基于代码共用层面的整体设计思路

2.1.1. 共同点：都是dfs深搜（当然也可以用栈来实现..）不同点是游戏消除移动规则不相同，导致的移动函数不同

2.1.2. 可以利用老师给的公告函数小技巧来解决公共函数问题，比如print一个符号可以是数字当然也可以是一个小星星，这样使公用代码更多，程序鲁棒性更强

2.1.3. 流程图如下：





装 订 线

2.1.4. 下面这个公共函数是坠棒的！：

```
void giveTetris(TetrisData *dataSet, TetrisCondition *controlPanel) {
    uint64_t tetris;
    dataSet->type[0] = dataSet->type[1];
    dataSet->ori[0] = dataSet->ori[1];
    dataSet->type[1] = dataSet->type[2];
    dataSet->ori[1] = dataSet->ori[2];
    dataSet->type[2] = rand() % 7;
    dataSet->ori[2] = rand() & 3;
    tetris = numPatternTable[dataSet->type[0]][dataSet->ori[0]];
    dataSet->y = 0;
    dataSet->x = 6;
    if (checkCollision(dataSet)) {
        dataSet->dead = true;
        printCurrentTetris(dataSet, controlPanel);
    }
    else
        insertTetris(dataSet);
    dataSet->tetrisTotal++;
    dataSet->tetrisCount[dataSet->type[0]]++;
    printNextTetris(dataSet, controlPanel);
    printScore(dataSet);
}
```

2.1.5. 心得体会：

总之就是尽自己可能写程序的过程中尽量合并函数，经验什么的没时间写了！在赶ddl！

2.2. 设计与实现

2.2.1. 描述：俄罗斯方块，老少皆宜，童叟无欺的游戏，比贪玩蓝月还好玩！

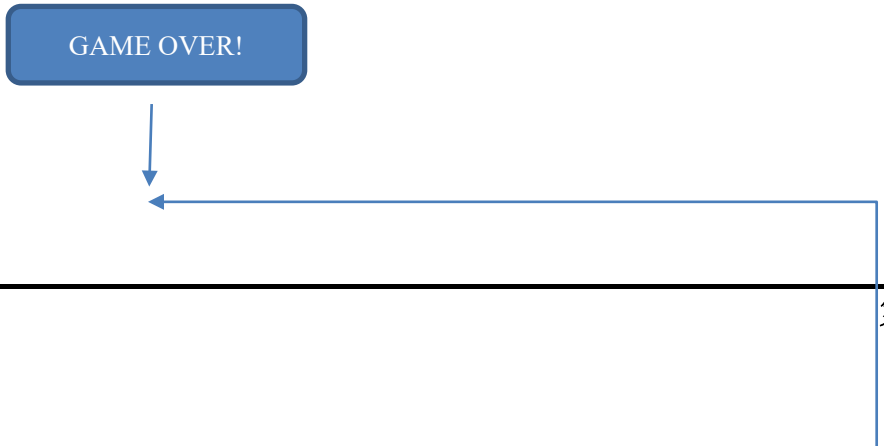
2.2.2. 整体设计思路：都是套路啦！和前两个一样一样的，基本上都是搜索，只不过俄罗斯方块涉及到一个很恶心的就是那个数字的存储问题，我用的二进制位存储，如下图所示：

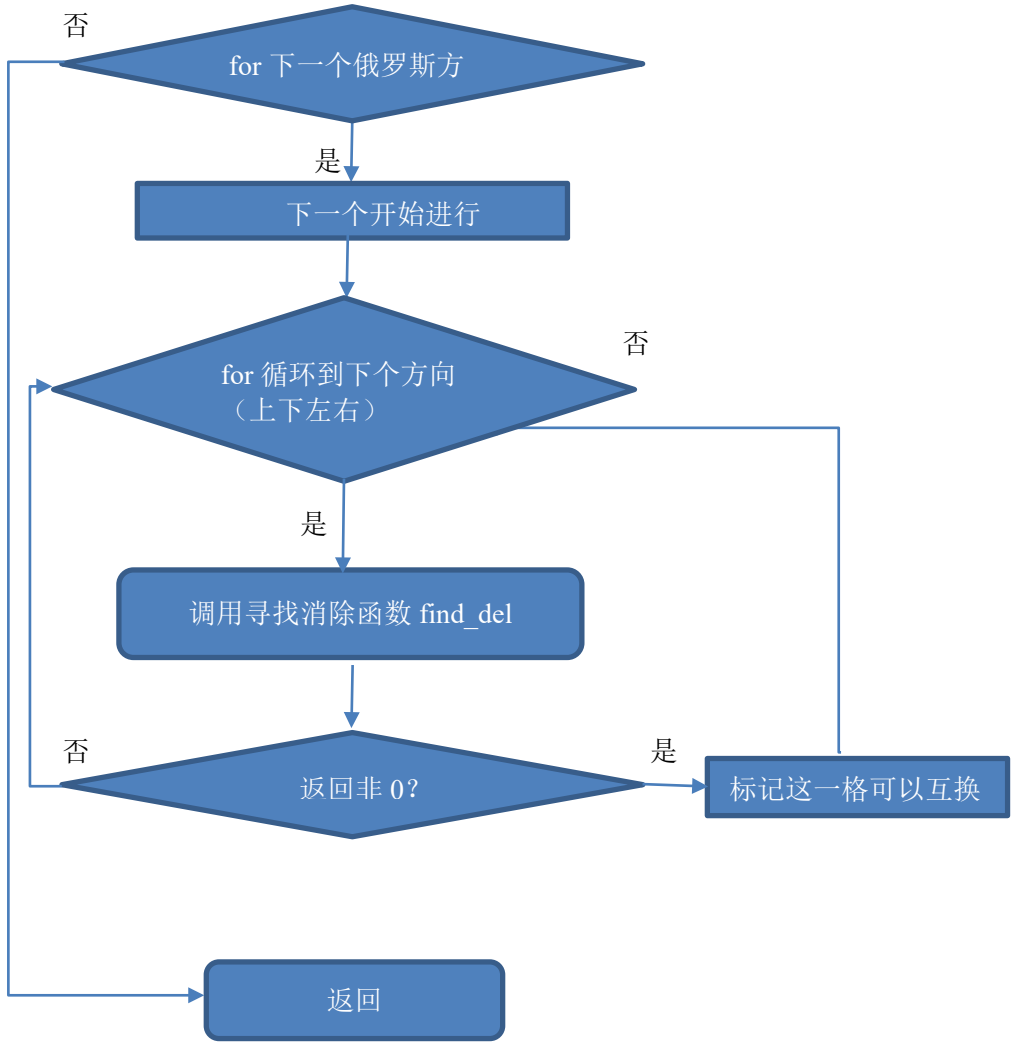
```
static const uint64_t numPatternTable[10][4] = {
    { 0x00000007050505070U, 0x00000000001F111FU, 0x00000007050505070U, 0x00000000001F111FU },         // 0
    { 0x00000002020202020U, 0x00000000000001F00U, 0x00000002020202020U, 0x00000000000001F00U },         // 1
    { 0x00000007010704070U, 0x00000000001D1517U, 0x00000007010704070U, 0x00000000001D1517U },         // 2
    { 0x00000007010701070U, 0x00000000001515FU, 0x00000007040704070U, 0x00000000001F1515U },         // 3
    { 0x00000005050701010U, 0x00000000007041FU, 0x00000004040705050U, 0x00000000001F041CU },         // 4
    { 0x00000007040701070U, 0x00000000017151DU, 0x00000007040701070U, 0x00000000017151DU },         // 5
    { 0x00000007040705070U, 0x00000000001F151DU, 0x00000007050701070U, 0x00000000017151FU },         // 6
    { 0x00000007010101010U, 0x00000000001011FU, 0x00000004040404070U, 0x00000000001F1010U },         // 7
    { 0x00000007050705070U, 0x00000000001F151FU, 0x00000007050705070U, 0x00000000001F151FU },         // 8
    { 0x00000007050701070U, 0x00000000017151FU, 0x00000007040705070U, 0x00000000001F151DU }          // 9
};

static const uint16_t modelPatternTable[ARRAY] = {
    0x8001U, 0x8001U, 0x8001U, 0x8001U, 0x8001U, 0x8001U, 0x8001U, 0x8001U,
    0x8001U, 0x8001U, 0x8001U, 0x8001U, 0x8001U, 0x8001U, 0x8001U, 0x8001U,
    0xFFFFU
};
```

只是程序实现的时候非常复杂非常麻烦，出现很多bug. . .

2.2.3. 流程：





装 订 线

2.2.4. 遇到的问题：

```
void initGame(TetrisData *, TetrisCondition *);
void restartGame(TetrisData *, TetrisCondition *, bool flag = false);
void giveTetris(TetrisData *, TetrisCondition *);
bool checkCollision(const TetrisData *);
```

这部分函数的实现极为困难，尤其在位运算的情况下..

2.3. 和前两个大作业代码的公用：

2.3.1. 基本相同部分的函数我截图如下：

```
/*      Print Table according to the option         */
void PrintTable(int(*Table)[10], int(*MarkTable)[10], int *DT, int option);
/*                  Generate one Table              */
void GenerateTable(int(*Table)[10], int *DT);
/*  Calculate TotalGrade accroding to the command   */
bool CombineNumber(int(*Table)[10], int(*MarkTable)[10], int *DT);
/*              Draw the Boarder of the graph       */
void DrawBoarder(int x, int y, int col, int arr, int interval, bool flag);
/*              Draw or change the figure           */
void DrawBox(int x, int y, int num, int flag);
/*      Check whether there is box to eliminate     */
bool CheckContinue(int(*)[10], int *);
/*      previous operation before choose box        */
void PreChooseBox(int(*)[10], int *, bool, bool, int, int, int, int);
/*                 Restore the Table                */
void RestoreTable(int(*Table)[10], int(*MarkTable)[10], int *DT);
/*              Falling Movement of Box             */
void UpdateTable(int(*Table)[10], int(*MarkTable)[10], int *DT, bool);
/*               Falling Movement of Box            */
void UpdateDrawing(int(*Table)[10], int(*MarkTable)[10], int *DT, bool);
/*      Move the column to the black place          */
void MoveColumn(int(*Table)[10], int j, int *DT);
/*           Filling the blank of whole Table       */
void ComplementBox(int(*Table)[10], int(*MarkTable)[10], int *DT, bool flag);
/*        Execute order according to the cursor     */
bool GameType(int(*Table)[10], int(*MarkTable)[10], int *DT);
```

装 订 线

2.3.2. 同样没时间文字表述了，分别截图如下：

```
/*calculate the coordination according to col & arr */
void CalcCoord(int &, int &, const int, const int);
/*              Calculate the Score                 */
void CalcScore(int *DT, int N);
/*                  end sentences                   */
void EnterEnd();
/*                  DFS_recursion                   */
void DFS_recursion(int(*Table)[10], int(*MarkTable)[10], int *DT);
/*          judge the validity of character         */
bool JudgeCommand(int(*Table)[10], char *command, int *DT);
/*              Falling Movement of Box             */
void FallingMovement(int(*Table)[10], int *DT, int i, int j, int k);
/*              choice == 1 0r 2 Command            */
void ExecuteCommand(int(*Table)[10], int(*MarkTable)[10], int *DT);
/*      Print Table according to the option         */
void PrintTable(int(*Table)[10], int(*MarkTable)[10], int *DT, int option);
```

```
/*        Execute order according to the cursor     */
bool GameType(int(*Table)[10], int(*MarkTable)[10], int *DT);
/*                  Game Controler                  */
bool GameControl(int(*Table)[10], int(*MarkTable)[10], int *DT, bool);
/*              Choose Box by cursor key            */
void ChooseBox(int(*Table)[10], int(*MarkTable)[10], int *DT);
/*                  Draw the figure                 */
void DrawFigure(int(*Table)[10], int(*MarkTable)[10], int *DT, bool flag);
/*choose the solutions according to the main function*/;
/*                Initial Settings                  */
void InitialSetting(int(*Table)[10], int(*MarkTable)[10], int *DataTable);
/*choose the solutions according to the main function*/
void Solve(int(*Table)[10], int(*MarkTable)[10], int *DataTable);
```

2.3.3. 剩下的不同部分就是整体抛去我所截图的那些函数以及调用的..还有不同的数据类型等

2.4.配置文件：

2.4.1. 这个作业可是真的爽啊，因为我时间不足的原因，我只能挑了一个简化版来实现了，总体来说就是老师让我们把前3次大作业全改了，改到能控制一切输出为止..比如边框等等等...

2.4.2. 设计思路就是文件读取以后寻找目标框，例如在Linux下常见的这样的配置文件：

```
[配置组1]

配置组1-设置项A = 123
配置组1-设置项B = 123.450000
配置组1-设置项C = Hello
配置组1-设置项D = helloworld
配置组1-设置项E =
配置组1-设置项F = 99999
配置组1-设置项Y = helloworld
```

2.4.3. 实现差异：我没有完成C版本的...

2.4.4. 调试遇到问题：各种文件写不进去，发现了是那个

experimental头文件下的函数没有用——即没有修改文件写指针以及文件大小，修改之后就可以正常修改文件了。此前的另一种解决办法是用链表一行一行读取进文字然后清空文件重新输出，但这种极大地浪费内存。被我否定掉了...

2.5.综合体中进行地改造：

2.5.1. 没时间改之前的题了..所以没啥体会..

2.5.2. 老师写的工具函数是坠棒的！极其好用，以后用到的时候肯定很多，函数接口等提供的非常清晰，简介，兼容性极好！

2.6.心得体会：

2.6.1. 例如这种例子：

装 订 线

```
/*                    UpdateTable                    */
void UpdateTable(int(*Table)[10], int(*MarkTable)[10], int *DT, bool flag = true)
{
    gotoxy(0, DT[Bottom] - 2);
    setcolor(COLOR_BLACK, COLOR_HWHITE);
    if (flag)
        InputEnter(1);
    int k;
    for (int i = 0; i < DT[Column]; i++)
        for (int j = DT[Array] - 1; j >= 1; j--) {
            if (MarkTable[j][i] == SIGNED) {
                for (k = j - 1; k >= 0; k--) {
                    if (MarkTable[k][i] == UNSIGNED && Table[k][i]) {
                        Table[j][i] = Table[k][i];
                        MarkTable[j][i] = UNSIGNED;
                        if (DT[Choice] == 8 || DT[Choice] == 9)
                            FallingMovement(Table, DT, i, j, k);
                        MarkTable[k][i] = SIGNED;
                        Table[k][i] = 0;
                        break;
                    }
                }
                if (k < 0) {
                    if (j == DT[Array])
                        MoveColumn(Table, j, DT);
                    Table[k][i] = 0;
                    MarkTable[j][i] = SIGNED;
                }
            }
        }
}
```

我总是想把它改得尽量简介，所以花了很多时间与精力在这方面，我觉得可能还是因为自己写程序前思考的不多，下次应该想好了再写码

装 订 线

```
void printHint() {
    setcolor(COLOR_BLACK, COLOR_HYELLOW);
    gotoxy(90, 36);
    cout << "Control Panel:";
    gotoxy(90, 37);
    cout << "Move left:              ← A 4";
    gotoxy(90, 38);
    cout << "Move right:                → D 6";
    gotoxy(90, 39);
    cout << "Move downfoward:           ↓ S 2";
    gotoxy(90, 40);
    cout << "Clockwise rotation:            ↑ W 8";
    gotoxy(90, 41);
    cout << "Anti-clockwise rotation:       0";
    gotoxy(90, 42);
    cout << "Drop to the ground:            space";
    gotoxy(90, 43);
    cout << "Pause:              enter";
    gotoxy(90, 44);
}
```

我能提供良好界面地菜单提示，玩这款游戏地人一定很快乐..

```
void insertTetris(TetrisData *dataSet) {
    uint64_t tetris = numPatternTable[dataSet->type[0]][dataSet->ori[0]];


    if (dataSet->ori[0] == 1 || dataSet->ori[0] == 3) {
        for (int i = 0; i < 3; i++)
            dataSet->pool[dataSet->y + 2 - i] |= (((tetris >> (0x00 + 8 * i)) & 0x1F) << (11 - dataSet->x));
    }
    else {
        for (int i = 0; i < 5; i++)
            dataSet->pool[dataSet->y + 4 - i] |= (((tetris >> (0x00 + 8 * i)) & 0x07) << (13 - dataSet->x));
    }
}

void removeTetris(TetrisData *dataSet) {
    uint64_t tetris = numPatternTable[dataSet->type[0]][dataSet->ori[0]];

    if (dataSet->ori[0] == 1 || dataSet->ori[0] == 3) {
        for (int i = 0; i < 3; i++)
            dataSet->pool[dataSet->y + 2 - i] &= ~(((tetris >> (0x00 + i * 8)) & 0x001F) << (11 - dataSet->x));
    }
    else {
        for (int i = 0; i < 5; i++)
            dataSet->pool[dataSet->y + 4 - i] &= ~(((tetris >> (0x00 + i * 8)) & 0x007) << (13 - dataSet->x));
    }

}
```

这种位运算的例子真的很锻炼我的逻辑思维能力，尤其是把程序精简成一行，这样的自虐型编写代码方式...

收获颇多欸！

2.6.2. **我认为老师提供的底层函数，例如鼠标的控制以及键盘的输入，还有那个文件的写操作，读取文件地址操作，都是我值得思考与学习的，这些的应用很广，尤其是文件读写。**

3. 调试过程碰到的问题

在调试过程中，基本遇到问题都能够立即发现问题出在哪里，并且很快解决，暂时没有遇到在算法上的问题。

在完成作业当中遇到的最大问题还是不够专注。遇到问题的时候，会花很多时间来思考，但是之后开始动手写代码的时候，基本都不会遇到太大的问题。

4. 心得体会

5.1.经验教训

在设计函数的时候，有时候调用一些函数的功能可能需要输出多个字符串中的一个，以前的方法是用if-else if语句，而这次作业当中，在完成作业的过程中想到应该用数组来解决这个问题。这样的代码显示更加清晰，调整也更方便。同理，还有上下左右四个方向，以前的代码是直接列举四个方向，现在改为了用数组来表示方向的调整值，减少了相似作用代码的重复，也更加容易修改。

本次作业比起上学期的大作业，在难度上有所提高，但是为了适应新学的结构体，还是有一些需要调整的部分。本次作业直接在原来的文件上改动，尽管在一些细节上还保留了很多没有必要的部分，还是发现了很多以前程序上的问题，并对这些问题做出了修改。

其次，在写代码的时候也感受到了，**如果以前写的函数能够直接挪用过来或者做微调就可以直接使用，会大幅降低新程序的工作量**。

另外，在函数的参数方面，也做出了一些修改。由于使用了结构体作为参数，在传递参数的时候，可以选择是整个结构体作为形参传入还是作为指针或者以引用的方式传入，这使得参数的调用更加自由。同时，由于数组的大小只在结构体当中出现一次，在修改的时候比写死数值方便很多。

5.2.作业总结

在做复杂程序时，需要分成若干部分，并且考虑到前后的关系。在本次作业当中，在写一些函数的时候，会先用注释标明这个函数的逻辑，然后分块考虑是否有已经写好的函数可以调用，或者写一些新的函数。对于图形化界面和基础数组的输出的区别还是很难整合，在写一些函数的时候没办法把这两块区整合起来，但是写代码时能考虑到这每一块当中内部的联系。

重用代码其中一个重要的点是使用头文件。使用头文件可以看得更加清晰，并且便于修改。本次作业当中，还体会到了在一些相似的重复代码当中存在的一些联系，有办法能用数组等方式把这些代码联系在一起。

装 订 线

5. 附件：源程序

```
/* 1651574 贾昊霖 1班*/
#include "90-b2.h"

void printPoolBorder() {
    int8_t y;

    int endx = 2 + COLUMN * 6, endy = ROW_BEGIN + (ARRAY - 1) * 3;
    setcolor(COLOR_HWHITE, COLOR_HWHITE);
    for (y = ROW_BEGIN; y < endy; ++y)
    {
        gotoxy(COL_BEGIN, y);
        cout << "  ";
        gotoxy(endx, y);
        cout << "  ";
    }
    gotoxy(0, y);
    cout << setw(88) << " ";
}

void printTetrisPool(const TetrisData *dataSet, const TetrisCondition *controlPanel)
{
    int x, y;

    for (y = 0; y < ARRAY - 1; ++y) {
        for (x = 1; x <= COLUMN; ++x) {
            int posx = x * 6 - 4, posy = 1 + y * 3;
            if ((dataSet->pool[y] >> (COLUMN + 1 - x)) & 1) {
                setcolor(COLOR_BLACK, controlPanel->color[y][x]);
                DrawBox(posx, posy);
            }
            else {
                setcolor(COLOR_BLACK, COLOR_BLACK);
                DrawBox(posx, posy);
            }
        }
    }
}

void printCurrentTetris(const TetrisData *dataSet, const TetrisCondition *controlPanel) {
    int x, y, posx, posy;
    y = (dataSet->y > 0) ? (dataSet->y - 1) : 0;
    for (; y < ARRAY - 1 && y < dataSet->y + 6; y++) {
        x = (dataSet->x > 1) ? (dataSet->x - 1) : 1;
        for (; x < COLUMN + 1 && x < dataSet->x + 6; x++) {
            posx = x * 6 - 4;
            posy = 1 + y * 3;
```

```
                if ((dataSet->pool[y] >> (COLUMN + 1 - x)) & 1) {
                    setcolor(COLOR_BLACK, controlPanel->color[y][x]);
                    DrawBox(posx, posy);
                }
                else  if (y < ARRAY) {
                    setcolor(COLOR_BLACK, COLOR_BLACK);
                    DrawBox(posx, posy);
                }
            }
        }

}

void printNextTetris(const TetrisData *dataSet, const TetrisCondition *controlPanel) {
    setcolor(COLOR_BLACK, COLOR_HCYAN);
    gotoxy(90, 13);
    cout << "Next Number is :        Next Next Number is :";
    int type = dataSet->type[1], ori = dataSet->ori[1];
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 5; j++) {
            int tmp = (((numPatternTable[type][ori] >> (0x00 + 8 * i)) >> j) & 1);
            int posx = 6 * (4 - j), posy = 3 * (4 - i);
            if (tmp) {
                setcolor(COLOR_BLACK, dataSet->type[1] | 8);
                DrawBox(90 + posx, 15 + posy);
            }
            else {
                setcolor(COLOR_BLACK, COLOR_BLACK);
                DrawBox(90 + posx, 15 + posy);
            }
        }
    }
    type = dataSet->type[2];
    ori = dataSet->ori[2];
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 5; j++) {
            int tmp = (((numPatternTable[type][ori] >> (0x00 + 8 * i)) >> j) & 1);
            int posx = 6 * (4 - j), posy = 3 * (4 - i);
            if (tmp) {
                setcolor(COLOR_BLACK, dataSet->type[2] | 8);
                DrawBox(125 + posx, 15 + posy);
            }
            else {
                setcolor(COLOR_BLACK, COLOR_BLACK);
                DrawBox(125 + posx, 15 + posy);
            }
        }
    }
}

void printScore(const TetrisData *dataSet) {
    static const char *tetrisName = "ITLJZSO";

    setcolor(COLOR_BLACK, COLOR_HGREEN);

    gotoxy(90, 1);
    cout << "Score：" << dataSet->score;
```

装 订 线

```
    gotoxy(90, 2);
    cout << "Total times:  " << dataSet->erasedTotal;
    for (int i = 0; i < 9; i++) {
        gotoxy(90, 3 + i);
        cout << i << " times" << ":      " << dataSet->erasedCount[i];
    }
}

void printHint() {
    setcolor(COLOR_BLACK, COLOR_HYELLOW);
    gotoxy(90, 36);
    cout << "Control Panel:";
    gotoxy(90, 37);
    cout << "Move left:                  ← A 4";
    gotoxy(90, 38);
    cout << "Move right:                 → D 6";
    gotoxy(90, 39);
    cout << "Move downfoward:            ↓ S 2";
    gotoxy(90, 40);
    cout << "Clockwise rotation:             ↑ W 8";
    gotoxy(90, 41);
    cout << "Anti-clockwise rotation:        0";
    gotoxy(90, 42);
    cout << "Drop to the ground:             space";
    gotoxy(90, 43);
    cout << "Pause:                  enter";
    gotoxy(90, 44);
}

void runGame(TetrisData *dataSet, TetrisCondition *controlPanel) {
    clock_t clockLast, clockNow;

    clockLast = clock();
    printTetrisPool(dataSet, controlPanel);

    while (!dataSet->dead)
    {
        while (_kbhit())
        {
            keydownControl(dataSet, controlPanel, _getch());
        }

        if (!controlPanel->pause)
        {
            clockNow = clock();

            if (clockNow - clockLast > 0.85F * CLOCKS_PER_SEC) {
                clockLast = clockNow;
                keydownControl(dataSet, controlPanel, 80);
            }
        }
    }
}

bool ifPlayAgain() {
    int ch;
```

```
    setcolor(COLOR_HWHITE, COLOR_HRED);
    gotoxy(40, 25);
    cout << "GAME OVER";
    gotoxy(22, 26);
    cout << "PRESS 'Y' TO TRY AGAIN!, THEN PRESS 'N' TO END";

    do {
        ch = _getch();
        if (ch == 'Y' || ch == 'y') {
            return true;
        }
        else if (ch == 'N' || ch == 'n') {
            setcolor(COLOR_BLACK, COLOR_BLACK);
            gotoxy(0, 0);
            return false;
        }
    } while (1);
}
#include "90-b2.h"

void insertTetris(TetrisData *dataSet) {
    uint64_t tetris = numPatternTable[dataSet->type[0]][dataSet->ori[0]];


    if (dataSet->ori[0] == 1 || dataSet->ori[0] == 3) {
        for (int i = 0; i < 3; i++)
            dataSet->pool[dataSet->y + 2 - i] |= (((tetris >> (0x00 + 8 * i)) & 0x1F) << (11 -
dataSet->x));
    }
    else {
        for (int i = 0; i < 5; i++)
            dataSet->pool[dataSet->y + 4 - i] |= (((tetris >> (0x00 + 8 * i)) & 0x07) << (13 -
dataSet->x));
    }
}

void removeTetris(TetrisData *dataSet) {
    uint64_t tetris = numPatternTable[dataSet->type[0]][dataSet->ori[0]];

    if (dataSet->ori[0] == 1 || dataSet->ori[0] == 3) {
        for (int i = 0; i < 3; i++)
            dataSet->pool[dataSet->y + 2 - i] &= ~(((tetris >> (0x00 + i * 8)) & 0x001F) << (11 -
dataSet->x));
    }
    else {
        for (int i = 0; i < 5; i++)
            dataSet->pool[dataSet->y + 4 - i] &= ~(((tetris >> (0x00 + i * 8)) & 0x007) << (13 -
dataSet->x));
    }

}

void setPoolColor(const TetrisData *dataSet, TetrisCondition *controlPanel) {


    int num, posx, posy;
```

```
    bool flag = false;

    uint64_t tetris = numPatternTable[dataSet->type[0]][dataSet->ori[0]];
    if (dataSet->ori[0] == 1 || dataSet->ori[0] == 3) {
        for (int i = 0; i < 3; i++) {
            num = (tetris >> (0x00 + 8 * i) & 0x001F);
            posy = dataSet->y + 2 - i;
            for (int j = 0; j < 5; j++) {
                posx = dataSet->x + j;
                if (((num >> (4 - j)) & 1))
                    controlPanel->color[posy][posx] = (dataSet->type[0] | 8);
            }
        }
    }
    else {
        for (int i = 0; i < 5; i++) {
            num = (tetris >> (0x00 + 8 * i) & 0x0007);
            posy = dataSet->y + 4 - i;
            for (int j = 0; j < 3; j++) {
                posx = dataSet->x + j;
                if (((num >> (2 - j)) & 1))
                    controlPanel->color[posy][posx] = (dataSet->type[0] | 8);
            }
        }
    }
}

void rotateTetris(TetrisData *dataSet, TetrisCondition *controlPanel) {
    int8_t ori = dataSet->ori[0];

    removeTetris(dataSet);

    int orix = dataSet->x;
    if (dataSet->ori[0] == 1 || dataSet->ori[0] == 3)
        dataSet->x++;
    else
        dataSet->x--;
    dataSet->ori[0] = (controlPanel->clockwise) ? ((ori + 1) & 3) : ((ori + 3) & 3);

    if (checkCollision(dataSet))
    {
        dataSet->x = orix;
        dataSet->ori[0] = ori;
        insertTetris(dataSet);
    }
    else {
        insertTetris(dataSet);
        setPoolColor(dataSet, controlPanel);
        printCurrentTetris(dataSet, controlPanel);
    }
}


void horzMoveTetris(TetrisData *dataSet, TetrisCondition *controlPanel) {
    int x = dataSet->x;

    removeTetris(dataSet);
```

```
    controlPanel->direction == 0 ? (--dataSet->x) : (++dataSet->x);

    if (checkCollision(dataSet)) {
        dataSet->x = x;
        insertTetris(dataSet);
    }
    else {
        insertTetris(dataSet);
        setPoolColor(dataSet, controlPanel);
        printCurrentTetris(dataSet, controlPanel);
    }
}

void moveDownTetris(TetrisData *dataSet, TetrisCondition *controlPanel) {
    int8_t y = dataSet->y;
    removeTetris(dataSet);
    ++dataSet->y;

    if (checkCollision(dataSet)) {
        dataSet->y = y;
        insertTetris(dataSet);
        if (checkErasing(dataSet, controlPanel))
            printTetrisPool(dataSet, controlPanel);
    }
    else {
        insertTetris(dataSet);
        setPoolColor(dataSet, controlPanel);
        printCurrentTetris(dataSet, controlPanel);
    }
}

void dropDownTetris(TetrisData *dataSet, TetrisCondition *controlPanel) {
    removeTetris(dataSet);
    for (; dataSet->y < ROW_END; ++dataSet->y)
        if (checkCollision(dataSet))
            break;
    --dataSet->y;

    insertTetris(dataSet);
    setPoolColor(dataSet, controlPanel);

    checkErasing(dataSet, controlPanel);
    printTetrisPool(dataSet, controlPanel);
}

bool checkErasing(TetrisData *dataSet, TetrisCondition *controlPanel) {
    static const unsigned scores[5] = { 0, 10, 30, 90, 150 };
    int8_t count = 0;
    int8_t k = 0, y;
    if (dataSet->ori[0] == 1 || dataSet->ori[0] == 3)
        y = dataSet->y + 2;
    else
        y = dataSet->y + 4;

    do {
        if (y < ARRAY && dataSet->pool[y] == 0xFFFFU) {
            ++count;
```

```
                memmove(dataSet->pool + 1, dataSet->pool, sizeof(uint16_t) * y);
                memmove(controlPanel->color[1], controlPanel->color[0],
                    sizeof(int8_t[16]) * y);
            }
            else {
                --y;
                ++k;
            }
        } while (y >= dataSet->y && k < 4);
        //dataSet->erasedTotal += count;
        dataSet->score += scores[count];
        if (count > 0)
            dataSet->erasedCount[count - 1]++;
        giveTetris(dataSet, controlPanel);
        setPoolColor(dataSet, controlPanel);
        return (count > 0);
    }

    void keydownControl(TetrisData *dataSet, TetrisCondition *controlPanel, int key) {
        if (key == 13)  // pause
            controlPanel->pause = !controlPanel->pause;

        if (controlPanel->pause)  // pause state
            return;

        switch (key) {
        case 'w':
        case 'W':
        case '8':
        case 72:  //up
            controlPanel->clockwise = true;  // rotate
            rotateTetris(dataSet, controlPanel);
            break;
        case 'a':
        case 'A':
        case '4':
        case 75:  // left
            controlPanel->direction = 0;
            horzMoveTetris(dataSet, controlPanel);
            break;
        case 'd':
        case 'D':
        case '6':
        case 77:  // right
            controlPanel->direction = 1;  // right
            horzMoveTetris(dataSet, controlPanel);
            break;
        case 's':
        case 'S':
        case '2':
        case 80:  // down
            moveDownTetris(dataSet, controlPanel);
            break;
        case ' ':  // speed up
            dropDownTetris(dataSet, controlPanel);
            break;
        case '0':  // reverse
```

```
            controlPanel->clockwise = false;
            rotateTetris(dataSet, controlPanel);
            break;
        default:
            break;
        }
}
```

装 订 线