高级语言程序设计大作业（二）

大作业报告

贰零肆捌

学　号：　2353900

姓　名：　汪嘉晨

学　院：　国豪书院

班　级：　工科二班

二零二四年五月二十五日

一、设计思路与功能概述

(一)设计思路

2048 小游戏整体的设计思路如下：

0. 菜单，规则界面，用于提示用户进行操作；

1. 退出，用于游戏退出；

2. 游戏函数：

0)界面初始化，创建 4*4 的矩阵，并在任意的两个位置上生成 2 或 4；

1)用户操作，输入↑ ↓ ←→，实现不同指令，并计分数与步数；

2)退出，如果输入 esc，那么会询问是否退出，确认后即可恢复菜单；

3)胜利，如果合成了 2048，那么游戏胜利，玩家可以选择是否继续游戏；

4)失败，如果填满了整个表格，并且不能移动每个格子，那么失败。

3. 显示函数，对游戏函数输出的矩阵、分数、步数，以及表格，在窗口进行上色显示。

4. 其他各种函数……（如图 1）

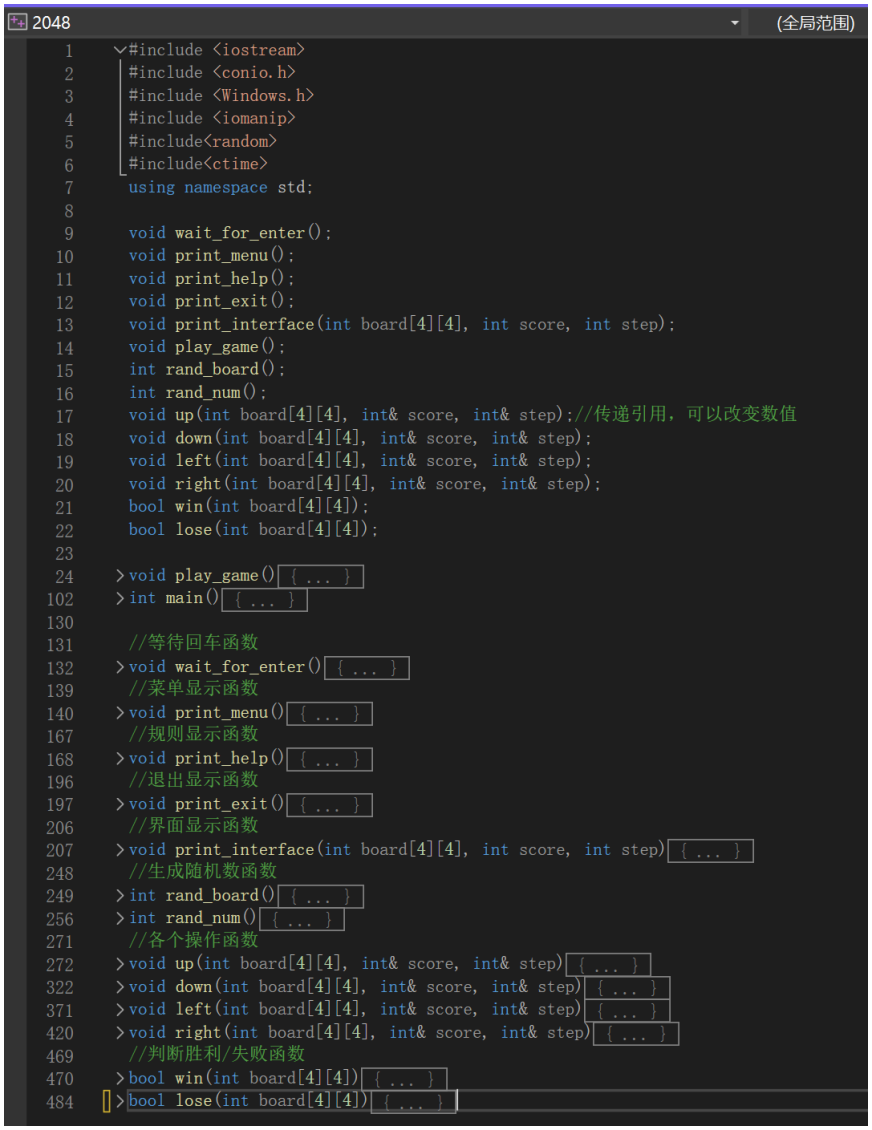

图 1 主要设计思路

(二)功能概述

2048 小游戏整体的功能展示如下：

0.菜单显示：在 2048 小游戏中，首先进入的界面即为菜单界面（如图 2），菜单界面可以输入 a-c，进入经典模式/查看规则或退出游戏。

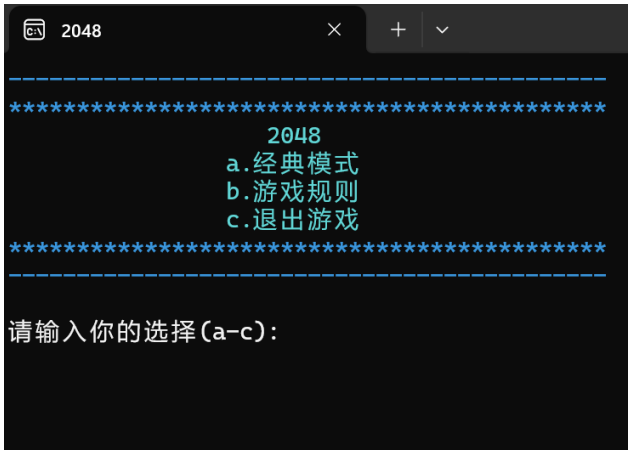

图 2 菜单显示

1.游戏规则：

操作说明：

↓：下 ←：左 ↑：上 →：右 ESC 键：退出

游戏介绍：

每次选择一个方向移动，移动时数字向该方向靠拢，相同数字可合并，移动后空格处会生成随机数字 2/4，如果得到数字 2048，则游戏胜利!如 果 棋盘被数字填满，无法进行移动，则游戏失败!

2.退出游戏：菜单下如果输入 c，那么程序会退出（如图 3）。

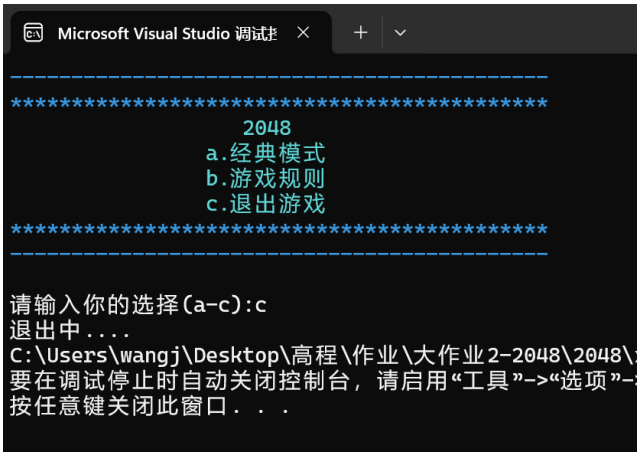

图 3 退出游戏

3.经典模式

1)游戏界面初始化：定义一个 4 行 4 列的矩阵，游戏开始时在任意的两个位置上会出现 2 或 4（在本程序中出现 2 的概率为 90%，出现 4 的概率为 10%，可以根据不同情况修改其他版本）。

2)根据游戏规则，玩家可以输入上下左右实现操作，上方即为分数（合并的方块的值）以及步数，下方为游戏的主界面（如图 4）。

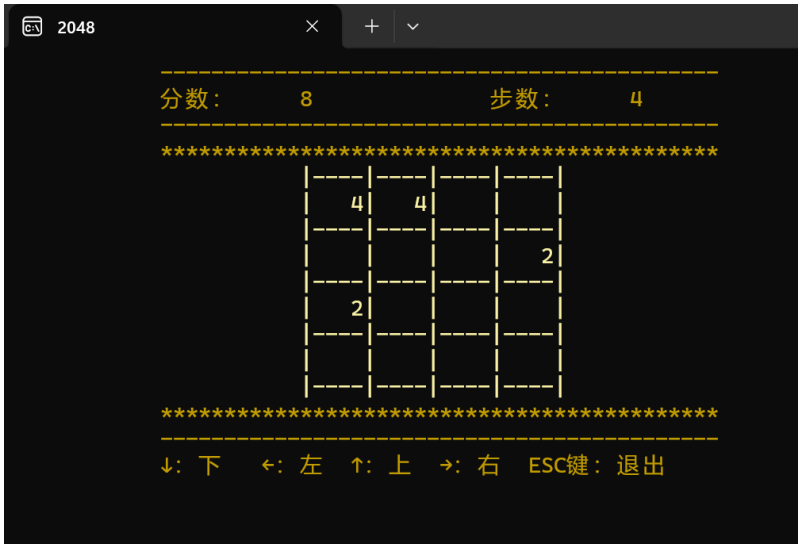

图 4 2048 主界面

3)胜利，如果成功凑到 2048，那么游戏胜利！（如图 5，后台魔改，仅供测试）

4)失败，如果所有格子均填满，并且无法移动，那么游戏失败！（如图 6）

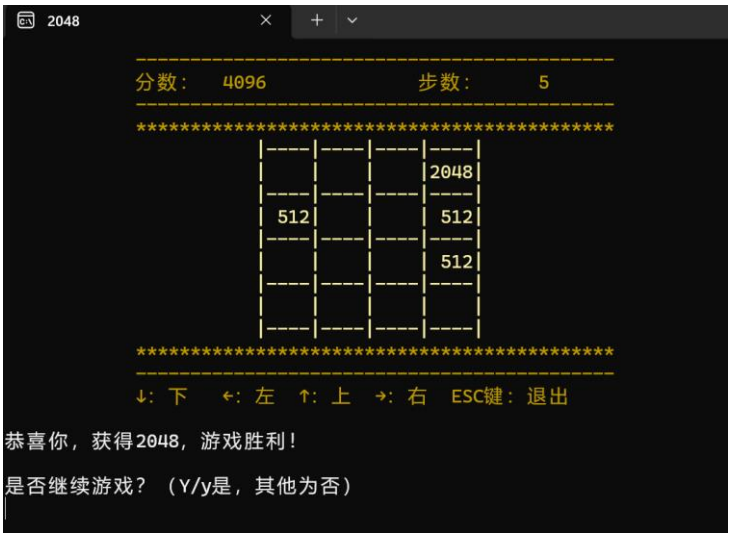

图 5 游戏胜利

图 6 游戏失败

二、问题与解决方法

Q1 使用函数，如何改变/调用变量？

A1 这就与才上过不久的变量存储&内存一节课相关了，在调用的函数的括号中加上个“&”就能解决问题（如图 7）（但是我不理解为什么数组就不需要啊啊啊啊啊啊）。

```
271     //各个操作函数
272   > void up(int board[4][4], int& score, int& step) { ... }
322   > void down(int board[4][4], int& score, int& step) { ... }
371   > void left(int board[4][4], int& score, int& step) { ... }
420   > void right(int board[4][4], int& score, int& step) { ... }
```

图 7 “&”

Q2 如何设置随机数？

A2 毕竟这玩意也没有讲，通过查询 CSDN 和万能的 copilot，我晓得了可以调用 #include<random> #include<ctime>下的 random 函数来设置随机数，通过设置随机数，可以确定我需要把新添加的数放到哪里，新添加的数是多少以及多大概率出现这个数的问题（如图 8）。

```
//生成随机数函数
int rand_board()
{
    random_device rd;  // 用于获取随机数的种子
    mt19937 gen(rd()); // 使用 Mersenne Twister 算法生成随机数
    uniform_int_distribution<> dis(0, 3); // 定义随机数的分布范围[0,3]
    return dis(gen); // 生成随机数
}
int rand_num()
{
    random_device rd;
    mt19937 gen(rd());
    uniform_int_distribution<> dis(0, 9); // 定义随机数的分布范围[0,9]
    int num = dis(gen);
    if (num < 9)
    {
        return 2;
    }
    else
    {
        return 4;
    }
}
```

图 8 设置随机数

Q3 判断胜利或者失败时，由于添加的函数太冗长，内存疑似爆掉了！

A3 之前初始化，随机数，上下左右操作，判断是否需要结束，是胜利还是失败都放到了一个函数下面（游戏函数），导致游戏函数运行不下去了（疑似爆掉），于是考虑多设置几个 bool 或者 void 的函数分担分担。判断胜利，只需要查找是否出现 2048 即可，非常简单；判断失败，需要考虑①4*4 的表格是不是填满了，即 board[i][j]全满，并且要考虑是不是还能继续操作，最后想到可以判断前后

左右是不是一样，这样就能解决问题（如图 9）。

```
        print_interface(board, score, step);// 打印游戏界面
        // 判断游戏是否结束，如果结束则跳出循环
        if (win(board))
        {
            cout << "恭喜你，获得2048，游戏胜利！" << endl;
            wait_for_enter();
            break;
        }
        if (lose(board))
        {
            cout << "很遗憾，游戏失败！" << endl;
            wait_for_enter();
            break;
        }
```

图 9 判断胜利/失败

三、心得体会

0. 本次大作业感觉较为简单，一是因为 2048 游戏太好玩太爱玩，二是因为联想到之前的小作业“数字板”，感觉 2048 游戏可以拿来参考，三时游戏的主体已经搭建完毕，只需要做一个 play 函数。
1. 经历过 Lena 的折磨，（还有两周时间看起来非常紧急）（还有朋友圈有人发说半天就能搞完）,这回终于敢多多尝试了，勇气可嘉（雾）。
2. 制作 2048 游戏，用了一个晚上思考各个函数有什么作用，用了一个高程上机课去编写，又用了一个晚上去思考哪里会出现 bug（判断胜利，判断失败，移动时会不会超范围，分数添加，步数添加），经过多方面测试，可以拿来实战了！
3. 至于不足，可能我的 2、4、8、16、32、64、128、256、512、1024、2048 没有上不同的颜色！
4. 还有一点吐槽的，和群里面某位同学吐槽的那样，一次又一次地清屏真的会看眼花(◎＿◎)。
5. 其他也没什么可写的啦，那就祝老师助教少脱发，多开心吧！（长期熬夜脱发人士是这样的。。。）
~~6. 五月二十九日，与他人交流这道 2048 如何编写时看到了 2048 以后应该怎么办我还没有写，遂做补充！（还要仔细读题啊）~~

四、源代码

~~感觉自我良好的我就黄底啦！~~

```
#include <iostream>
#include <conio.h>
#include <Windows.h>
#include <iomanip>
```

```
#include<random>
#include<ctime>
using namespace std;

void wait_for_enter();
void print_menu();
void print_help();
void print_exit();
void print_interface(int board[4][4], int score, int step);
void play_game();
int rand_board();
int rand_num();
void up(int board[4][4], int& score, int& step);//传递引用，可以改变数值
void down(int board[4][4], int& score, int& step);
void left(int board[4][4], int& score, int& step);
void right(int board[4][4], int& score, int& step);
bool win(int board[4][4]);
bool lose(int board[4][4]);

void play_game()
{
    int board[4][4] = {}; // 4*4的棋盘
    int score = 0;          // 分数
    int step = 0;           // 步数
    char choice = '\0';     // 用户选择
    int start = 0;//看是不是第一次进入游戏
    // 游戏初始化，包括生成随机数等操作
    // 生成两个随机数,代表棋盘上的两个格子
    // 游戏开始时生成两个随机数放在这两个格子上面，有可能是2，也有可能是4，2的
可能性更大，不妨定为90%
    int count = 0;
    while (count < 2)
    {
        int x = rand_board();
        int y = rand_board();
        if (board[x][y] == 0)//如果第二个生成的和第一个重合，那么也会重新生成
        {
            board[x][y] = rand_num();
            count++;
        }
    }
    // 打印游戏界面
    print_interface(board, score, step);
    while (1)
```

```
{
        choice = _getch();// 获取用户输入
        switch (choice)// 根据用户输入进行相应操作
        {
        case 72://上
        {
                up(board, score, step);
                break;
        }
        case 80://下
        {
                down(board, score, step);
                break;
        }
        case 75://左
        {
                left(board, score, step);
                break;
        }
        case 77://右
        {
                right(board, score, step);
                break;
        }
        case 27://ESC键
        {
                cout << "是否退出游戏？（Y/y是，其他为否）" << endl;
                choice = _getch();
                if (choice == 'Y' || choice == 'y')
                {
                        print_menu();
                        return;
                }
                else
                {
                        continue;
                }
        }
        }
        print_interface(board, score, step);// 打印游戏界面
        // 判断游戏是否结束，如果结束则跳出循环
        if (win(board)&&start==0)
        {
                cout << "恭喜你，获得2048，游戏胜利！" << endl;
```

```
                cout << endl;
                cout<<"是否继续游戏？（Y/y是，其他为否）" << endl;
                choice = _getch();
                if (choice == 'Y' || choice == 'y')
                {
                    start = 1;
                    continue;
                }
                else
                {
                    wait_for_enter();
                    break;
                }
            }
            if (lose(board))
            {
                cout << "很遗憾，游戏失败！" << endl;
                wait_for_enter();
                break;
            }
        }
}
int main()
{
    char choice = '\0';
    SetConsoleTitle(TEXT("2048"));   // 设置控制台标题为2048
    while (1)
    {
        print_menu();
        choice = _getche();

        // 根据用户选择进行相应操作
        switch (choice)
        {
        case 'a':
            play_game();
            break;
        case 'b':
            print_help();
            break;
        case'c':
            print_exit();
            return 0;
        default:
```

```
                cout << "\n输入错误，请从新输入" << endl;
                wait_for_enter();
            }
        }
        return 0;
}
//等待回车函数
void wait_for_enter()
{
        cout << endl << "按回车键继续";
        while (_getch() != '\r')
            ;
        cout << endl << endl;
}
//菜单显示函数
void print_menu()
{
        // 清屏
        system("CLS");
        // 获取标准输出设备句柄
        HANDLE handle_out = GetStdHandle(STD_OUTPUT_HANDLE);
        // 设置控制台文字颜色
        SetConsoleTextAttribute(handle_out, FOREGROUND_BLUE | FOREGROUND_GREEN);
        // 打印菜单
        cout << "-----------------------------------------\n";
        cout << "*****************************************\n";
        // 设置控制台文字颜色
        SetConsoleTextAttribute(handle_out, FOREGROUND_BLUE | FOREGROUND_GREEN |
FOREGROUND_INTENSITY);
        // 打印标题
        cout << "                    2048\n";
        cout << "                a.经典模式\n";
        cout << "                b.游戏规则\n";
        cout << "                c.退出游戏\n";
        // 设置控制台文字颜色
        SetConsoleTextAttribute(handle_out, FOREGROUND_BLUE | FOREGROUND_GREEN);
        // 打印菜单
        cout << "*****************************************\n";
        cout << "-----------------------------------------\n";
        // 恢复控制台文字颜色为默认颜色
        SetConsoleTextAttribute(handle_out, FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_BLUE | FOREGROUND_INTENSITY);
        cout << "\n请输入你的选择(a-c):";
}
```

```
//规则显示函数
void print_help()
{
	// 清屏
	system("CLS");
	// 获取标准输出设备句柄
	HANDLE handle_out = GetStdHandle(STD_OUTPUT_HANDLE);
	// 设置控制台文字颜色
	SetConsoleTextAttribute(handle_out, FOREGROUND_BLUE | FOREGROUND_GREEN);
	cout << "------------------------------------------\n";
	cout << "******************************************\n\n";
	// 设置控制台文字颜色
	SetConsoleTextAttribute(handle_out, FOREGROUND_BLUE | FOREGROUND_GREEN |
FOREGROUND_INTENSITY);
	// 打印游戏规则
	cout << "操作说明：\n\n";
	cout << "↓：下　←：左　↑：上　→：右　ESC键：退出\n\n";
	cout << "游戏介绍：\n\n";
	cout << "每次选择一个方向移动，移动时数字向该方向靠拢\n";
	cout << "相同数字可合并，移动后空格处会生成随机数字2/4\n";
	cout << "如果得到数字2048，则游戏胜利!\n";
	cout << "如果棋盘被数字填满，无法进行移动，则游戏失败!\n\n";
	// 设置控制台文字颜色
	SetConsoleTextAttribute(handle_out, FOREGROUND_BLUE | FOREGROUND_GREEN);
	cout << "******************************************\n";
	cout << "------------------------------------------\n";
	// 恢复控制台文字颜色为默认颜色
	SetConsoleTextAttribute(handle_out, FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_BLUE | FOREGROUND_INTENSITY);
	wait_for_enter();
}
//退出显示函数
void print_exit()
{
	cout << "\n退出中";
	for (int i = 4; i > 0; --i)
	{
		Sleep(200);
		cout << ".";
	}
}
//界面显示函数
void print_interface(int board[4][4], int score, int step)
{
```

```
    // 清屏
    system("CLS");
    //获取标准输入设备句柄
    HANDLE handle_out = GetStdHandle(STD_OUTPUT_HANDLE);
    // 设置控制台文字颜色
    SetConsoleTextAttribute(handle_out, FOREGROUND_RED | FOREGROUND_GREEN);
    // 打印游戏界面
    cout << "                    ------------------------------------------\n";
    cout << "                    分数:  " << setw(6) << score << "                    步数:  " <<
setw(6) << step << endl;
    cout << "                    ------------------------------------------\n";
    cout << "                    ******************************************\n";
    // 设置控制台文字颜色
    SetConsoleTextAttribute(handle_out, FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_INTENSITY);
    cout << "                                |----|----|----|----|\n";

    for (int i = 0; i < 4; i++)
    {
        cout << "                                |";
        for (int j = 0; j < 4; j++)
        {
            if (board[i][j] != 0)
            {
                cout << setw(4) << board[i][j] << "|";
            }
            else
            {
                cout << "    |";
            }
        }
        cout << "\n                                |----|----|----|----|\n";
    }
    // 设置控制台文字颜色
    SetConsoleTextAttribute(handle_out, FOREGROUND_RED | FOREGROUND_GREEN);
    cout << "                    ******************************************\n";
    cout << "                    ------------------------------------------\n";
    cout << "                    ↓: 下  ←: 左  ↑: 上  →: 右  ESC键: 退出\n\n";
    // 恢复控制台文字颜色为默认颜色
    SetConsoleTextAttribute(handle_out, FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_BLUE | FOREGROUND_INTENSITY);
}
//生成随机数函数
int rand_board()
```

```
{
	random_device rd;   // 用于获取随机数的种子
	mt19937 gen(rd()); // 使用 Mersenne Twister 算法生成随机数
	uniform_int_distribution<> dis(0, 3); // 定义随机数的分布范围[0,3]
	return dis(gen); // 生成随机数
}
int rand_num()
{
	random_device rd;
	mt19937 gen(rd());
	uniform_int_distribution<> dis(0, 9); // 定义随机数的分布范围[0,9]
	int num = dis(gen);
	if (num < 9)
	{
		return 2;
	}
	else
	{
		return 4;
	}
}
//各个操作函数
void up(int board[4][4], int& score, int& step)
{
	bool flag = false;
	for (int j = 0; j <= 3; j++)
	{
		for (int i = 0; i <= 3; i++)
		{
			if (board[i][j] == 0)
			{
				for (int k = i + 1; k <= 3; k++)
				{
					if (board[k][j] != 0)
					{
						board[i][j] = board[k][j];
						board[k][j] = 0;
						flag = true;//如果可以去掉空格，可以移动
						break;
					}
				}
			}
		}//去掉空格子之后再去判断
		for (int i = 0; i <= 2; i++)
```

```
            {
                if (board[i][j] != 0 && board[i][j] == board[i + 1][j])
                {
                    board[i][j] *= 2;
                    score += board[i][j];
                    for (int k = i + 1; k <= 2; k++)
                    {
                        board[k][j] = board[k + 1][j];
                    }
                    board[3][j] = 0;
                    flag = true;
                }
            }
        }
        //更新数据
        if (flag == true)//如果有移动，那么步数可以增加，并且可以生成新的数字
        {
            step++;
            int x = rand_board();
            int y = rand_board();
            while (board[x][y] != 0)//如果有数字，那么重新生成
            {
                x = rand_board();
                y = rand_board();
            }
            board[x][y] = rand_num();
        }
}
void down(int board[4][4], int& score, int& step)
{
        bool flag = false;
        for (int j = 0; j <= 3; j++)
        {
            for (int i = 3; i >= 0; i--)
            {
                if (board[i][j] == 0)
                {
                    for (int k = i - 1; k >= 0; k--)
                    {
                        if (board[k][j] != 0)
                        {
                            board[i][j] = board[k][j];
                            board[k][j] = 0;
                            flag = true;
```

```
                    break;
                }
            }
        }
    }
    for (int i = 3; i >= 1; i--)
    {
        if (board[i][j] != 0 && board[i][j] == board[i - 1][j])
        {
            board[i][j] *= 2;
            score += board[i][j];
            for (int k = i - 1; k >= 1; k--)
            {
                board[k][j] = board[k - 1][j];
            }
            board[0][j] = 0;
            flag = true;
        }
    }
}
if (flag == true)
{
    step++;
    int x = rand_board();
    int y = rand_board();
    while (board[x][y] != 0)
    {
        x = rand_board();
        y = rand_board();
    }
    board[x][y] = rand_num();
}
}
void left(int board[4][4], int& score, int& step)
{
    bool flag = false;
    for (int i = 0; i <= 3; i++)
    {
        for (int j = 0; j <= 3; j++)
        {
            if (board[i][j] == 0)
            {
                for (int k = j + 1; k <= 3; k++)
                {
```

```
                        if (board[i][k] != 0)
                        {
                            board[i][j] = board[i][k];
                            board[i][k] = 0;
                            flag = true;
                            break;
                        }
                    }
                }
            }
            for (int j = 0; j <= 2; j++)
            {
                if (board[i][j] != 0 && board[i][j] == board[i][j + 1])
                {
                    board[i][j] *= 2;
                    score += board[i][j];
                    for (int k = j + 1; k <= 2; k++)
                    {
                        board[i][k] = board[i][k + 1];
                    }
                    board[i][3] = 0;
                    flag = true;
                }
            }
        }
        if (flag == true)
        {
            step++;
            int x = rand_board();
            int y = rand_board();
            while (board[x][y] != 0)
            {
                x = rand_board();
                y = rand_board();
            }
            board[x][y] = rand_num();
        }
}
void right(int board[4][4], int& score, int& step)
{
        bool flag = false;
        for (int i = 0; i <= 3; i++)
        {
            for (int j = 3; j >= 0; j--)
```

```
            {
                if (board[i][j] == 0)
                {
                    for (int k = j - 1; k >= 0; k--)
                    {
                        if (board[i][k] != 0)
                        {
                            board[i][j] = board[i][k];
                            board[i][k] = 0;
                            flag = true;
                            break;
                        }
                    }
                }
            }
            for (int j = 3; j >= 1; j--)
            {
                if (board[i][j] != 0 && board[i][j] == board[i][j - 1])
                {
                    board[i][j] *= 2;
                    score += board[i][j];
                    for (int k = j - 1; k >= 1; k--)
                    {
                        board[i][k] = board[i][k - 1];
                    }
                    board[i][0] = 0;
                    flag = true;
                }
            }
        }
        if (flag == true)
        {
            step++;
            int x = rand_board();
            int y = rand_board();
            while (board[x][y] != 0)
            {
                x = rand_board();
                y = rand_board();
            }
            board[x][y] = rand_num();
        }
    }
    //判断胜利/失败函数
```

```
bool win(int board[4][4])
{
	for (int i = 0; i <= 3; i++)
	{
		for (int j = 0; j <= 3; j++)
		{
			if (board[i][j] == 2048)
			{
				return true;
			}
		}
	}
	return false;
}
bool lose(int board[4][4])
{
	bool full = true;
	for (int i = 0; i <= 3; i++)
	{
		for (int j = 0; j <= 3; j++)
		{
			if (board[i][j] == 0)
			{
				full = false;
			}
		}
	}
	if (full == true)
	{
		for (int i = 0; i <= 3; i++)
		{
			for (int j = 0; j <= 3; j++)
			{
				if (i != 3 && board[i][j] == board[i + 1][j])
				{
					continue;
				}
				else if (i != 0 && board[i][j] == board[i - 1][j])
				{
					continue;
				}
				else if (j != 3 && board[i][j] == board[i][j + 1])
				{
					continue;
```

```
                    }
                    else if (j != 0 && board[i][j] == board[i][j - 1])
                    {
                        continue;
                    }
                    else
                    {
                        return true;
                    }
                }
            }
        }
        return false;
    }
```