/\*

GAME2048

version 4

2048.cpp

\*/

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include <conio.h>

#include <direct.h>

#include <math.h>

#include <windows.h>

#include "MainPart.h"

#include "ScoSavLoad.h"

#include "GamePart.h"

#include "Setting.h"

int main()

{

int mainChoice;

int menuChoice;

int settingChoice;

int ifTestModel = 0;

int \* score;

int scoreNumber = 0;

int stillPlay;

score = &scoreNumber;

srand((unsigned int)time(NULL));

HANDLE hOut;

hOut = GetStdHandle(STD\_OUTPUT\_HANDLE);

system("mode con cols=45 lines=25");

SetConsoleTitleA("2048");

SetConsoleTextAttribute(hOut,

BACKGROUND\_BLUE | BACKGROUND\_GREEN | BACKGROUND\_RED | BACKGROUND\_INTENSITY);

system("cls");

FILE \* file;

errno\_t fileErr;

errno\_t err;

fileErr = fopen\_s(&file, "TEST\\HaveDone.txt", "r");

if (fileErr != 0)

{

\_mkdir("TEST");

err = fopen\_s(&file, "TEST\\HaveDone.txt", "w");

fprintf(file, "HavePrepared\n");

fclose(file);

addFileAddress();

} //2048.cpp 1

do

{

printGreeting(ifTestModel);

mainChoice = getNumberChoice(0, 2);

switch (mainChoice)

{

case(1): //游戏

do

{

game2048(score, ifTestModel);

printEnd();

stillPlay = getNumberChoice(0, 1);

}

while (stillPlay == 1);

break;

case(2): //主页

do

{

menuGreeting();

menuChoice = getNumberChoice(0, 2);

if (menuChoice == 0)

{

stillPlay = 0;

}

else if (menuChoice == 1)

{

loadScore();

stillPlay = 1;

}

else if (menuChoice == 2)

{

do

{

settingGreeting();

settingChoice = getNumberChoice(0, 1);

if (settingChoice == 1)

ifTestModel = toMakeSureColorTest();

}

while (settingChoice != 0);

stillPlay = 1;

}

}

while (stillPlay == 1);

}

}

while (mainChoice != 0);

return 0;

}

//2048.cpp 2

/\*

GAME2048

version 4

MainPart.h

\*/

//主页相关函数声明

void printGreeting(int ifColorTest); //打印欢迎界面

void menuGreeting(); //打印菜单界面

int getNumberChoice(int start, int end); //获取数字选择

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//打印欢迎界面

void printGreeting(int ifColorTest)

{

system("cls");

printf(" \n");

printf("|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n");

printf("| |\n");

printf("| |\n");

printf("| 游戏：2048 |\n");

printf("| |\n");

printf(ifColorTest == 1 ? "| 1：开始(颜色测试模式) |\n" : "| 1：开始 |\n");

printf("| |\n");

printf("| 2：菜单 |\n");

printf("| |\n");

printf("| 0：退出 |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n\n");

}

// MainPart.h 1

获取数字选择

int getNumberChoice(int start, int end)

{

int ans;

int wrong;

do

{

ans = \_getch();

if (((ans - 48) >= start) && ((ans - 48) <= end))

{

wrong = 0;

return ans - 48;

}

else

{

wrong = 1;

}

}

while (wrong == 1);

return 0;

}

//打印菜单界面

void menuGreeting()

{

system("cls");

printf(" \n");

printf("|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n");

printf("| |\n");

printf("| |\n");

printf("| 游戏：2048 |\n");

printf("| |\n");

printf("| 1：分数记录 |\n");

printf("| |\n");

printf("| 2：设置 |\n");

printf("| |\n");

printf("| 0：退出 |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n\n");

}

// MainPart.h 2

/\*

GAME2048

version 4

ScoSavLoad.h

\*/

//保存相关函数声明

void addFileAddress(); //添加资源保存文件夹

void saveScore(int score); //保存分数

void loadScore(); //读取分数

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//保存分数

void saveScore(int score)

{

struct tm t; //tm结构指针

time\_t now; //声明time\_t类型变量

time(&now); //获取系统日期和时间

localtime\_s(&t, &now); //获取当地日期和时间

FILE \* scoreFile;

errno\_t fileErr;

fileErr = fopen\_s(&scoreFile, "SAVE\\SCORE\\score.txt", "r+");

fseek(scoreFile, 0 - sizeof(int), SEEK\_END);

fprintf(scoreFile, "%-8d %-4d %-2d %-2d %-2d %-2d %-2d\n0000", score, t.tm\_year + 1900, t.tm\_mon + 1, t.tm\_mday, t.tm\_hour, t.tm\_min, t.tm\_sec);

fclose(scoreFile);

}

//读取分数

void loadScore()

{

FILE \* scoreFile;

errno\_t fileErr;

int counter; //测试数据数量时读取数据的储存变量

int quantityOfData = 0; //数据数量

int printedNumbers[15][7] = { 0 }; //需打印的数据

int page = 0; //总页数

int currentPage = 1; //记录当前页数

int choice; //存储选择

int blanks; //记录不需打印的行数

fileErr = fopen\_s(&scoreFile, "SAVE\\SCORE\\score.txt", "r");

fscanf\_s(scoreFile, "%d", &counter); // ScoSavLoad.h 1

while (counter != 0) //计数据数量

{

for (int i = 0; i < 6; i++)

fscanf\_s(scoreFile, "%d", &counter);

quantityOfData++;

fscanf\_s(scoreFile, "%d", &counter);

}

fclose(scoreFile);

if (quantityOfData != 0) //若有数据

{

page = (int)ceil(quantityOfData / 15.0);

int \* scoreData = (int \*)malloc(7 \* 15 \* page \* sizeof(int));

fileErr = fopen\_s(&scoreFile, "SAVE\\SCORE\\score.txt", "r");

for (int i = 0; i < quantityOfData; i++) //读取数据

{

fscanf\_s(scoreFile, "%d", scoreData + 7 \* i);

if (scoreData + 7 \* i != 0)

for (int j = 1; j < 7; j++)

fscanf\_s(scoreFile, "%d", scoreData + (7 \* i + j));

}

fclose(scoreFile);

do

{

if (currentPage == page)

blanks = 15 \* page - quantityOfData;

else

blanks = 0;

for (int i = 0; i < 15; i++)

for (int j = 0; j < 7; j++)

if (i <= 15 - blanks)

printedNumbers[i][j] =

\*(scoreData + 7 \* (quantityOfData - i - 1) - 15 \* 7 \* (currentPage - 1) + j);

else

printedNumbers[i][j] = 0;

// ScoSavLoad.h 2

system("cls");

printf(" \n"); //打印页面

printf("|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n");

for (int i = 0; i < 15; i++)

if (printedNumbers[i][0] != 0)

{

printf("| %-8d %-4d.%-2d.%-2d %-2d:%-2d:%-2d |\n",

printedNumbers[i][0],

printedNumbers[i][1],

printedNumbers[i][2],

printedNumbers[i][3],

printedNumbers[i][4],

printedNumbers[i][5],

printedNumbers[i][6]);

}

else

{

printf("| |\n");

}

printf("| |\n");

printf("| %3d/%-3d |\n", currentPage, page);

printf("| 1:上一页 2:下一页 0:退出 |\n");

printf("|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n\n");

choice = getNumberChoice(0, 2);

\

switch (choice) //判断选择

{

case(2):

if (currentPage < page) {

currentPage++;

}

break;

case(1):

if (currentPage > 1) {

currentPage--;

}

break;

}

}

while (choice != 0);

free(scoreData);

} // ScoSavLoad.h 1

else

{ //若无数据

system("cls");

printf(" \n");

printf("|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n");

printf("| |\n");

printf("| |\n");

printf("| 游戏：2048 |\n");

printf("| |\n");

printf("| 你还未玩过游戏呢 |\n");

printf("| |\n");

printf("| 快去玩几把吧 |\n");

printf("| |\n");

printf("| 0：我知道了 |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n\n");

choice = getNumberChoice(0, 0);

}

}

//添加资源保存文件夹

void addFileAddress()

{

\_mkdir("SAVE");

\_mkdir("SCREEN");

\_mkdir("SAVE\\SCORE");

\_mkdir("SAVE\\GAME\_SAVE");

\_mkdir("SCREEN\\STYLE");

\_mkdir("SCREEN\\CONTROL\_WAY");

errno\_t err;

FILE \* pFile;

err = fopen\_s(&pFile, "SAVE\\SCORE\\score.txt", "w");

fprintf(pFile, "0000");

fclose(pFile);

err = fopen\_s(&pFile, "SAVE\\SCORE\\highest score.txt", "w");

fclose(pFile);

}

// ScoSavLoad.h 2

/\*

GAME2048

version 4

Setting.h

\*/

//设置相关函数

void settingGreeting(); //打印设置界面

int toMakeSureColorTest(); //询问是否打开测试模式

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//打印设置界面

void settingGreeting() {

system("cls");

printf(" \n");

printf("|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n");

printf("| |\n");

printf("| |\n");

printf("| 游戏：2048 |\n");

printf("| |\n");

printf("| 1：颜色测试模式 |\n");

printf("| |\n");

printf("| 0：退出 |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n\n");

}

// Setting.h 1

//询问是否打开测试模式

int toMakeSureColorTest() {

int Choice;

system("cls");

printf(" \n");

printf("|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n");

printf("| |\n");

printf("| |\n");

printf("| 游戏：2048 |\n");

printf("| |\n");

printf("| 确认打开颜色测试？ |\n");

printf("| |\n");

printf("| 1：是 0：否 |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n\n");

Choice = getNumberChoice(0, 1);

return Choice;

}

// Setting.h 2

/\*

GAME2048

version 4

GamePart.h

\*/

//游戏相关函数声明

void game2048(int \* score, int ifTest); //游戏主函数

int getCommand(); //获取游戏指令

void printEnd(); //打印失败界面

void printBlocks(int N[][4], int \* score, int \* highestScore); //打印方块

void addBlocks(int N[][4]); //添加方块

int moveBlocks(int N[][4], int mode); //移动方块

int addUpBlocks(int N[][4], int mode, int \* score); //加和方块

bool testIfEnd(int N[][4]); //测试结束

const int MOVE\_UP = 1, MOVE\_DOWN = 2, MOVE\_LEFT = 3, MOVE\_RIGHT = 4;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//打印方块

void printBlocks(int N[][4], int \* score, int \* highestScore)

{

HANDLE hOut;

hOut = GetStdHandle(STD\_OUTPUT\_HANDLE);

system("cls");

printf("|\*\*\*\*\*\*|\*\*\*\*\*\*|\*\*\*\*\*\*|\*\*\*\*\*\*| 最高分：%d\n", ((\*highestScore > \*score) ? (\*highestScore) : (\*score)));

for (int j = 0; j < 4; j++)

{

for (int i = 0; i < 4; i++)

if (N[i][j] == 0)

{

SetConsoleTextAttribute(hOut,

BACKGROUND\_RED | BACKGROUND\_GREEN | BACKGROUND\_BLUE | BACKGROUND\_INTENSITY);

printf("| ");

}

else

{

SetConsoleTextAttribute(hOut,

BACKGROUND\_RED | BACKGROUND\_GREEN | BACKGROUND\_BLUE | BACKGROUND\_INTENSITY);

printf("|");

switch (N[i][j])

{

case(2):

SetConsoleTextAttribute(hOut,

BACKGROUND\_RED | BACKGROUND\_INTENSITY |

FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_RED | FOREGROUND\_INTENSITY);

printf(" %-5d", N[i][j]);

break; // GamePart.h 1

case(4):

SetConsoleTextAttribute(hOut, BACKGROUND\_RED |

FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_RED | FOREGROUND\_INTENSITY);

printf(" %-5d", N[i][j]);

break;

case(8):

SetConsoleTextAttribute(hOut, BACKGROUND\_RED | BACKGROUND\_GREEN | BACKGROUND\_INTENSITY |

FOREGROUND\_BLUE);

printf(" %-5d", N[i][j]);

break;

case(16):

SetConsoleTextAttribute(hOut, BACKGROUND\_RED | BACKGROUND\_GREEN |

FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_RED | FOREGROUND\_INTENSITY);

printf(" %-5d", N[i][j]);

break;

case(32):

SetConsoleTextAttribute(hOut, BACKGROUND\_GREEN | BACKGROUND\_INTENSITY |

FOREGROUND\_BLUE);

printf(" %-5d", N[i][j]);

break;

case(64):

SetConsoleTextAttribute(hOut, BACKGROUND\_GREEN |

FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_RED | FOREGROUND\_INTENSITY);

printf(" %-5d", N[i][j]);

break;

case(128):

SetConsoleTextAttribute(hOut, BACKGROUND\_BLUE | BACKGROUND\_INTENSITY |

FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_RED | FOREGROUND\_INTENSITY);

printf(" %-5d", N[i][j]);

break;

case(256):

SetConsoleTextAttribute(hOut, BACKGROUND\_BLUE |

FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_RED | FOREGROUND\_INTENSITY);

printf(" %-5d", N[i][j]);

break;

case(512):

SetConsoleTextAttribute(hOut, BACKGROUND\_RED | BACKGROUND\_BLUE | BACKGROUND\_INTENSITY |

FOREGROUND\_BLUE);

printf(" %-5d", N[i][j]);

break;

case(1024):

SetConsoleTextAttribute(hOut, BACKGROUND\_RED | BACKGROUND\_BLUE |

FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_RED | FOREGROUND\_INTENSITY);

printf(" %-5d", N[i][j]);

break;

case(2048):

SetConsoleTextAttribute(hOut,

BACKGROUND\_BLUE | BACKGROUND\_GREEN | BACKGROUND\_INTENSITY |

FOREGROUND\_BLUE); // GamePart.h 2

printf(" %-5d", N[i][j]);

break;

case(4096):

SetConsoleTextAttribute(hOut, BACKGROUND\_BLUE | BACKGROUND\_GREEN |

FOREGROUND\_BLUE | FOREGROUND\_GREEN | FOREGROUND\_RED | FOREGROUND\_INTENSITY);

default:

printf(" %-5d", N[i][j]);

break;

}

}

SetConsoleTextAttribute(hOut,

BACKGROUND\_RED | BACKGROUND\_GREEN | BACKGROUND\_BLUE | BACKGROUND\_INTENSITY);

if (j == 0)

printf("| 分数 ：%d\n", \*score);

else

printf("|\n");

printf("|\*\*\*\*\*\*|\*\*\*\*\*\*|\*\*\*\*\*\*|\*\*\*\*\*\*|\n");

}

printf(" \* 用WASD控制 \* Q退出\n");

printf(" \* 关闭输入法或切换成英文以获得最佳游戏体验\n");

}

//添加方块

void addBlocks(int N[][4])

{

int i, j;

bool dup;

int addBlockNumber;

int randBlockNumber;

do

{

dup = false;

randBlockNumber = rand() % 20;

if (randBlockNumber >= 0 && randBlockNumber <= 15)

addBlockNumber = 2;

if (randBlockNumber >= 16 && randBlockNumber <= 20)

addBlockNumber = 4;

i = (rand() % 4);

j = (rand() % 4);

if (N[i][j] == 0)

N[i][j] = addBlockNumber;

else

dup = true;

} while (dup);

} // GamePart.h 3

//移动方块

int moveBlocks(int N[][4], int mode) {

int lastOne, thisOne, haveMove = 0;

int up = 0, down = 0, left = 0, right = 0;

switch (mode)

{

case(MOVE\_UP):

up = 1;

break;

case(MOVE\_DOWN):

down = 1;

break;

case(MOVE\_LEFT):

left = 1;

break;

case(MOVE\_RIGHT):

right = 1;

break;

default:

break;

}

for (int j = 0; j < 4; j++)

for (int i = 0; i < 3; i++)

{

lastOne = N[j\*(up + down) + i \* left + (3 - i)\*right]

[j\*(right + left) + i \* up + (3 - i)\*down];

thisOne = N[j\*(up + down) + (i + 1)\*left + (2 - i)\*right]

[j\*(right + left) + (i + 1)\*up + (2 - i)\*down];

if (lastOne == 0 && thisOne != 0)

{

lastOne = thisOne;

thisOne = 0;

N[j\*(up + down) + i \* left + (3 - i)\*right]

[j\*(right + left) + i \* up + (3 - i)\*down] = lastOne;

N[j\*(up + down) + (i + 1)\*left + (2 - i)\*right]

[j\*(right + left) + (i + 1)\*up + (2 - i)\*down] = thisOne;

haveMove++;

}

}

return haveMove;

}

// GamePart.h 4

//加和方块

int addUpBlocks(int N[][4], int mode, int \* score) {

int lastOne, thisOne, haveAdd = 0;

int up = 0, down = 0, left = 0, right = 0;

switch (mode)

{

case(MOVE\_UP):

up = 1;

break;

case(MOVE\_DOWN):

down = 1;

break;

case(MOVE\_LEFT):

left = 1;

break;

case(MOVE\_RIGHT):

right = 1;

break;

default:

break;

}

for (int j = 0; j < 4; j++) {

for (int i = 0; i < 3; i++) {

lastOne = N[j\*(up + down) + i \* left + (3 - i)\*right][j\*(right + left) + i \* up + (3 - i)\*down];

thisOne = N[j\*(up + down) + (i + 1)\*left + (2 - i)\*right][j\*(right + left) + (i + 1)\*up + (2 - i)\*down];

if (lastOne == thisOne && lastOne != 0 && thisOne != 0) {

haveAdd++;

lastOne = 2 \* thisOne;

\*score += (2 \* thisOne);

thisOne = 0;

N[j\*(up + down) + i \* left + (3 - i)\*right][j\*(right + left) + i \* up + (3 - i)\*down] = lastOne;

N[j\*(up + down) + (i + 1)\*left + (2 - i)\*right][j\*(right + left) + (i + 1)\*up + (2 - i)\*down] = thisOne;

}

}

}

return haveAdd;

}

// GamePart.h 5

//获取游戏指令

int getCommand()

{

int doChoice;

do

{

doChoice = \_getch();

} while ((doChoice != 'w') && (doChoice != 's') && (doChoice != 'a') && (doChoice != 'd') && (doChoice != 'q'));

switch (doChoice)

{

case('w'):

return 1;

break;

case('s'):

return 2;

break;

case('a'):

return 3;

break;

case('d'):

return 4;

break;

case('q'):

return 0;

break;

default:

return 0;

break;

}

}

// GamePart.h 6

//测试结束

bool testIfEnd(int N[][4])

{

int up, down, left, right;

int lastOne, thisOne;

bool end = true;

bool noBlank = true;

for (int j = 0; j < 4; j++)

for (int i = 0; i < 4; i++)

if (N[i][j] == 0)

{

noBlank = false;

end = false;

}

for (int k = 0; k < 4; k++)

{

up = 0, down = 0, left = 0, right = 0;

switch (k)

{

case(0):

up = 1;

break;

case(1):

down = 1;

break;

case(2):

left = 1;

break;

case(3):

right = 1;

break;

}

if (noBlank)

for (int j = 0; j < 4; j++)

for (int i = 0; i < 3; i++)

{

lastOne = N[j\*(up + down) + i \* left + (3 - i)\*right]

[j\*(right + left) + i \* up + (3 - i)\*down];

thisOne = N[j\*(up + down) + (i + 1) \* left + (2 - i)\*right]

[j\*(right + left) + (i + 1)\*up + (2 - i)\*down];

if (lastOne == thisOne)

end = false;

}

}

return end;

} // GamePart.h 7

//打印失败界面

void printEnd()

{

printf(" \n");

printf("|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n");

printf("| |\n");

printf("| 你输了！ |\n");

printf("| |\n");

printf("| 1：再来一次 |\n");

printf("| |\n");

printf("| 0：返回主页 |\n");

printf("| |\n");

printf("|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n\n");

}

//游戏主函数

void game2048(int \* score, int ifTest)

{

bool end;

int doChoice;

int blockNumber[4][4];

int addNumber = 0, moveNumber = 0;

int quit = 0;

int surequit = 0;

int highestScore = 0;

int \* pHighestScore;

pHighestScore = &highestScore;

FILE \* scoreFile;

errno\_t fileErr;

int counter; //测试数据数量时读取数据的储存变量

int quantityOfData = 0; //数据数量

fileErr = fopen\_s(&scoreFile, "SAVE\\SCORE\\score.txt", "r");

fscanf\_s(scoreFile, "%d", &counter);

while (counter != 0) //计数据数量

{

for (int i = 0; i < 6; i++)

fscanf\_s(scoreFile, "%d", &counter);

quantityOfData++;

fscanf\_s(scoreFile, "%d", &counter);

}

fclose(scoreFile);

// GamePart.h 8

if (quantityOfData != 0) //若有数据

{

int \* scoreData = (int \*)malloc(7 \* quantityOfData \* sizeof(int));

fileErr = fopen\_s(&scoreFile, "SAVE\\SCORE\\score.txt", "r");

for (int i = 0; i < quantityOfData; i++) //读取数据

{

fscanf\_s(scoreFile, "%d", scoreData + 7 \* i);

if (scoreData + 7 \* i != 0)

for (int j = 1; j < 7; j++)

fscanf\_s(scoreFile, "%d", scoreData + (7 \* i + j));

}

fclose(scoreFile);

for (int i = 0; i < quantityOfData; i++)

if (highestScore < \*(scoreData + 7 \* i))

highestScore = \*(scoreData + 7 \* i);

}

else //若无数据

{

highestScore = 0;

}

\*score = 0;

for (int j = 0; j < 4; j++)

for (int i = 0; i < 4; i++)

blockNumber[i][j] = 0;

if (ifTest)

{

for (int j = 0; j < 4; j++)

for (int i = 0; i < 4; i++)

blockNumber[i][j] = (int)pow(2.00, (double)i + (double)j \* 4 + 1);

blockNumber[3][3] = 0;

}

else

{

addBlocks(blockNumber);

}

// GamePart.h 9

do

{

printBlocks(blockNumber, score, pHighestScore);

if (ifTest)

doChoice = 4;

else

doChoice = getCommand();

if (doChoice == 0)

{

system("cls");

printf(" \n");

printf("|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n");

printf("| |\n");

printf("| |\n");

printf("| 确定退出？ |\n");

printf("| |\n");

printf("| 1：退出 |\n");

printf("| |\n");

printf("| 0：返回 |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("| |\n");

printf("|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n\n");

surequit = getNumberChoice(0,1);

if (surequit == 1)

{

end = true;

quit = true;

}

else

{

end = false;

quit = false;

}

}

if (!surequit)

{ // GamePart.h 10

for (int i = 0; i < 4; i++)

{

if (i == 2)

addNumber += addUpBlocks(blockNumber, doChoice, score);

else

moveNumber += moveBlocks(blockNumber, doChoice);

}

while ((addNumber + moveNumber) != 0)

{

addBlocks(blockNumber);

addNumber = 0, moveNumber = 0;

}

if (doChoice != 0)

end = testIfEnd(blockNumber);

}

} while (!end);

printBlocks(blockNumber, score, pHighestScore);

if (!ifTest && !quit)

saveScore(\*score);

}

// GamePart.h 11