**实验报告**

****

报告名称：合成十游戏

班级：计算机1班

学号：1651574

姓名：贾昊霖

完成日期：2017年12月23日

1. **题目及基本要求**
   1. **题目**

合成十游戏

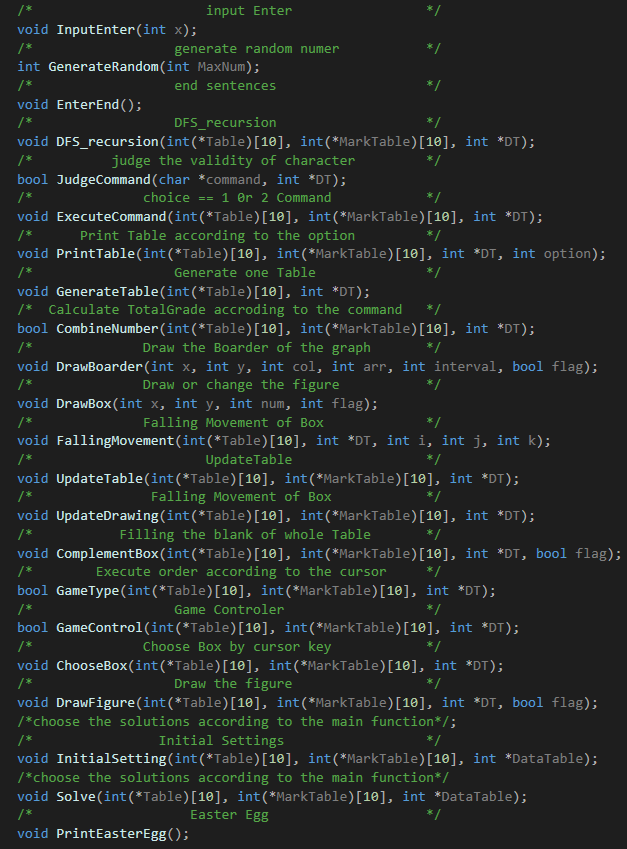
* 1. **基本要求**

类似于2048的那个游戏，就是要用控制台实现比较复杂的动画以并满足各种繁杂的要求..pdf没有太过仔细看，基本照着老师的exe做的

1. **整体设计思路**

输入之后进行菜单选择，先初始化函数，然后设置一系列的参数，然后根据不同的选择，从而在一个选择函数中选择不同的解决方案，这个实验与Hanoi不同的是需要更多的“控制函数”，就像硬件中控制系统一样，通过与控制函数的交互，实现对逻辑函数，即核心操作的函数进行控制，然而反复调用Solve那个函数，尽享二次选择，从而大大减少了代码量..此选择函数相当于控制器，在控制器中控制整个程序流程。

1. **主要功能的实现**

每个函数都写有非常详尽的注释，并且函数以及变量名都起得让读程序者一下就明白，所以我这里不在过多赘述，给出大体框架结构：

有几个非常关键的函数，其1：DFS，深搜递归思想，也是整个题的主干，在棋盘中寻找符合要求的元素，并展示出来，其2：把彩色打印图标那个函数写得好，可以以后做模板，方便以后编程的需要，但是开始的时候，我为了追求效率，而写的很乱，最后改的时候非常困难..吸取了教训，其3：把相同的元素删除，并添加新的随机数这里非常麻烦，并且开始自己想到的算法不仅很复杂，还使得做到最后8，9菜单时不能很好地兼容，使得我花大量的时间重新改UpdateTable那个函数

1. **调试过程碰到的问题**

遇到调试长时间的地方都用//important注释标注出来了，这里或者因为自己马虎，或者因为没有想到而造成的问题，其次遇到最大的问题就是自己写的程序与标准程序之间的差别，比如哪里没有光标显示，哪里需要把颜色调回黑白..其余没有遇到太多的问题，唯一就是很花时间..断点调试还不是很熟练，不知道vs有没有像Linux下条件断点的那种强大功能..另外还有一个问题就是由于不能出现任何的全局变量，因此参数传递需要非常多，故，我把他们全部都放到了DataTable这个数组之中，也用枚举变量清晰地实现了dataTable的改变,在传递过程中方便了许多.

1. **心得体会**

本次作业其中有一个下午是在上课的时候写的，另外两天晚上从1点写到凌晨1：00左右，然后周六早上写到晚上8点…一直完善并写实验报告..总体来说这次作业量很大很足..我写程序有一个习惯，只要能构造函数把代码压缩的，或者遇到相同的、重叠的函数、代码，都会想尽一切办法来用复杂的逻辑，把他们合并在一起，虽然这次作业老师没有要求做，但是我还是尽可能地合并了非常多的代码与函数，配合上显而易懂的命名，我认为这个程序显得很美.....但是做这个花费了确实很多的时间，但是也学到了很多的东西，并增强了写程序的熟练度，如何在短时间内高质高效地完成.

这次作业花费最长时间的不在于哪个函数，而是原本我个人习惯把纵轴当成x，横轴当成y，而老师给的，以及编译器系统给的是横轴为x，纵轴为y，因此我一开始用的非常乱，直接的后果就是我做大后面程序除了bug，我断点调试查一句，错一句，最后我生气地把好几个函数大体框架留着，其他细节涉及到坐标的，全部删除。重新写...所以这个宝贵的经验就是，以后一定要把横轴当成x，纵轴当成y...

其中1，2小题最简单，就是构造一个搜索函数

另外3，4小题也可归为一组，就是在原本的基础上继续增加，但这也遇到了写问题，比如，开始调试程序的时候总是不按照我的命令运行，我发现是忘记初始化，即每次运行都要memset我设立地DataTable那个指令集数组

而5，6小题...纯画图..很累，因为这块花了很长的时间，开始是因为自己太草率，写的程序健壮性很差，所以饮恨删除了重新写，写的比较完好，以后也可以用的边框模板。

7，8，9小题可以统一的归为一组，和1，2，3，4小题类似，在原本数字上加入了图画呈现，难点在于Box下落的动态，以及下落过程中遇到边框要重新打印的问题，但是这点不算很难，就是在连续写10小时以上程序，头脑不灵敏时，检查bug的效率极其低下..所以我认为不应该像我一样一头脑的写完，应该适当休息...下次要提升些效率了..

1. **附件：源程序**

/\*1651574 1班 贾昊霖\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <iomanip>

#include <cstring>

#include <conio.h>

#include <windows.h>

#include <stdlib.h>

#include <ctime>

#include <cmath>

#include "cmd\_console\_tools.h"

#include "90-b2.h"

using namespace std;

/\* main \*/

int main()

{

int TotalGrade = 0;

int Table[10][10];

int MarkTable[10][10];

int DataTable[20];

srand(unsigned(time(NULL)));

while (true) {

setconsoleborder(INITIAL\_Y, INITIAL\_X);

setcursor(CURSOR\_VISIBLE\_NORMAL);

system("color 0F");

memset(DataTable, 0, sizeof DataTable);

std::cout << "-----------------------------------" << endl;

std::cout << "1.命令行找出可合成项并标识（非递归）" << endl;

std::cout << "2.命令行找出可合成项并标识（递归）" << endl;

std::cout << "3.命令行完成一次合成（分步骤显示）" << endl;

std::cout << "4.命令行完整版（分步骤显示）" << endl;

std::cout << "5.伪图形界面显示初始数组（无分隔线）" << endl;

std::cout << "6.伪图形界面显示初始数组（有分隔线）" << endl;

std::cout << "7.伪图形界面下用箭头键选择当前色块" << endl;

std::cout << "8.伪图形界面完成一次合成（分步骤）" << endl;

std::cout << "9.伪图形界面完整版" << endl;

std::cout << "0.退出" << endl;

std::cout << "-----------------------------------" << endl;

std::cout << "[请选择0-9]";

do {

DataTable[Choice] = \_getch();

DataTable[Choice] -= '0';

} while (DataTable[Choice] < 0 || DataTable[Choice] > 9);

if (!DataTable[Choice])

break;

std::cout << DataTable[Choice] << endl;

InitialSetting(Table, MarkTable, DataTable);

Solve(Table, MarkTable, DataTable);

system("cls");

}

gotoxy(0, INITIAL\_X - 1);

setconsoleborder(75, 50);

system("color F0");

setcolor(COLOR\_HWHITE, COLOR\_BLACK);

PrintEasterEgg();

return 0;

}

#include "90-b2.h"

/\* Filling the blank of whole Table \*/

void ComplementBox(int(\*Table)[10], int(\*MarkTable)[10], int \*DT, bool flag)

{

int x, y;

for (int i = 0; i < DT[Array]; i++)

for (int j = 0; j < DT[Column]; j++)

if (MarkTable[i][j] == SIGNED) {

x = 2 + j \* 6 + 2 \* j;

y = 2 + i \* 3 + i;

if (flag)

DrawBox(x, y, Table[i][j], 0);

else if (i == DT[Orig\_y] && j == DT[Orig\_x])

DrawBox(x, y, Table[i][j], 2);

else

DrawBox(x, y, Table[i][j], 1);

}

}

/\* Execute order according to the cursor \*/

bool GameType(int(\*Table)[10], int(\*MarkTable)[10], int \*DT)

{

setcolor(COLOR\_BLACK, COLOR\_HWHITE);

gotoxy(0, DT[Bottom] - 1);

cout << "箭头键移动并取消选择，回车键合成\n";

int x, y;

MarkTable[DT[Orig\_y]][DT[Orig\_x]] = SIGNED;

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

DT[Tmp\_y] = DT[Orig\_y] + MOVE\_XY[i][0];

DT[Tmp\_x] = DT[Orig\_x] + MOVE\_XY[i][1];

DFS\_recursion(Table, MarkTable, DT);

}

if (!DT[Count]) {

setcolor(COLOR\_BLACK, COLOR\_HYELLOW);

std::cout << "周围无相同值!";

setcolor(COLOR\_BLACK, COLOR\_HWHITE);

std::cout << "请重新输入\n";

return false;

}

ComplementBox(Table, MarkTable, DT, false);

/\*sparkle\*/

x = 2 + DT[Orig\_x] \* 6 + 2 \* DT[Orig\_x];

y = 2 + DT[Orig\_y] \* 3 + DT[Orig\_y];

for (int i = 1; i <= TIMES; i++) {

DrawBox(x, y, Table[DT[Orig\_y]][DT[Orig\_x]], 2);

Sleep(5);

DrawBox(x, y, Table[DT[Orig\_y]][DT[Orig\_x]], 3);

Sleep(5);

}

return true;

}

/\* Game Controler \*/

bool GameControl(int(\*Table)[10], int(\*MarkTable)[10], int \*DT)

{

char c;

if (!GameType(Table, MarkTable, DT))

return true;

UpdateDrawing(Table, MarkTable, DT);

GenerateTable(Table, DT);

ComplementBox(Table, MarkTable, DT, true);

if (DT[Choice] == 8)

return false;

setcolor(COLOR\_BLACK, COLOR\_HWHITE);

gotoxy(0, DT[Bottom] - 1);

cout << "本次合成结束,按Q退出游戏\n";

while (true) {

c = \_getch();

if (c == 'q' || c == 'Q')

return false;

return true;

}

}

/\* Choose Box by cursor key \*/

void ChooseBox(int(\*Table)[10], int(\*MarkTable)[10], int \*DT)

{

int End\_col = DT[Column], End\_arr = DT[Array];

int col = 0, arr = 0, x = 2, y = 2;

unsigned short \_char;

setcursor(CURSOR\_INVISIBLE);

while (true) {

DrawBox(x, y, Table[arr][col], 1);

showch(0, DT[Bottom] - 1, ' ', COLOR\_BLACK, COLOR\_HWHITE, CLEAR);

setcolor(COLOR\_BLACK, COLOR\_HWHITE);

gotoxy(0, DT[Bottom] - 1);

cout << "方向键控制移动，回车确定\n";

\_char = \_getch();

if (\_char == 0xd) {

if (DT[Choice] == 7)

break;

else {

DT[Orig\_x] = col;

DT[Orig\_y] = arr;

DT[Count] = 0;

if (!GameControl(Table, MarkTable, DT))

break;

memset(MarkTable, UNSIGNED, sizeof(int) \* 10 \* 10); //important!

}

}

else if (\_char != 0xe0)

continue;

else {

\_char = \_getch();

DrawBox(x, y, Table[arr][col], 0);

if (\_char == 0x4b && col > 0)//left

col--;

else if (\_char == 0x50 && arr < End\_arr - 1)//down

arr++;

else if (\_char == 0x4d && col < End\_col - 1)//right

col++;

else if (\_char == 0x48 && arr > 0)//up

arr--;

x = 2 + col \* 6 + 2 \* col;

y = 2 + arr \* 3 + arr;

DrawBox(x, y, Table[arr][col], 1);

}

}

setcolor(COLOR\_BLACK, COLOR\_HWHITE);

gotoxy(0, DT[Bottom] - 1);

}

/\* Draw the figure \*/

void DrawFigure(int(\*Table)[10], int(\*MarkTable)[10], int \*DT, bool flag)

{

int arr = DT[Array], col = DT[Column];

int Boarder\_x = 4 + col \* 2 \* 3 + flag \* (col - 1) \* 2;

int Boarder\_y = 3 + arr \* 3 + flag \* (arr - 1);

int cur\_x, cur\_y;

setconsoleborder(Boarder\_x + 1, Boarder\_y + 5);

setcursor(CURSOR\_INVISIBLE);

cout << "屏幕当前设置为:" << Boarder\_y + 5 << "行" << Boarder\_x + 2 << "列" << endl;

DrawBoarder(0, 1, col, arr, 4, flag);

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

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

cur\_x = 2 + j \* 3 \* 2 + flag \* j \* 2;

cur\_y = 2 + i \* 3 + flag \* i;

DrawBox(cur\_x, cur\_y, Table[i][j], 0);

}

}

DT[Bottom] = Boarder\_y + 3;

setcolor(COLOR\_BLACK, COLOR\_HWHITE);

gotoxy(0, Boarder\_y + 2);

}

#include "90-b2.h"

/\* judge the validity of character \*/

bool JudgeCommand(char \*command, int \*DT)

{

if (strlen(command) < 2)

return false;

if (command[0] >= 'a' && command[0] <= 'a' + DT[Array] - 1)

command[0] -= 32;

if (command[0] < 'A' || command[0] > 'A' + DT[Array] - 1)

return false;

if (command[1] < '0' || command[1] > '0' + DT[Column] - 1)

return false;

return true;

}

/\* choice == 1 0r 2 Command \*/

void ExecuteCommand(int(\*Table)[10], int(\*MarkTable)[10], int \*DT)

{

int cur\_x, cur\_y;

char command[10];

while (true) {

std::cout << "请以字母+数字形式[例：c1]输入矩阵坐标：";

getxy(cur\_x, cur\_y);

while (true) {

std::cin >> command;

if (JudgeCommand(command, DT))

break;

showstr(cur\_x, cur\_y, " ", COLOR\_BLACK, COLOR\_HWHITE);

std::cout << "\n输入错误，请重新输入.";

gotoxy(cur\_x, cur\_y);

}

showch(0, cur\_y + 1, ' ', COLOR\_BLACK, COLOR\_HWHITE, CLEAR);

gotoxy(0, cur\_y + 1);

std::cout << "输入为" << command[0] << "行" << command[1] << "列\n";

DT[Orig\_y] = command[0] - 'A';

DT[Orig\_x] = command[1] - '0';

DT[Count] = 0;

MarkTable[DT[Orig\_y]][DT[Orig\_x]] = SIGNED;

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

DT[Tmp\_y] = DT[Orig\_y] + MOVE\_XY[i][0];

DT[Tmp\_x] = DT[Orig\_x] + MOVE\_XY[i][1];

DFS\_recursion(Table, MarkTable, DT);

}

if (DT[Count])

break;

std::cout << "输入的矩阵坐标位置处无连续相同值，请重新输入\n";

}

putchar('\n');

}

/\* Print Table according to the option \*/

void PrintTable(int(\*Table)[10], int(\*MarkTable)[10], int \*DT, int option)

{

if (option == 1)

std::cout << "当前数组：\n";

else if (option == 2)

std::cout << "寻找结果数组：\n";

else if (option == 3)

std::cout << "当前数组(不同色标识)：\n";

else if (option == 4)

std::cout << "相同值归并后的数组(不同色标识)：\n";

else if (option == 5)

std::cout << "除0后的数组(不同色标识)：\n";

else if (option == 6)

std::cout << "新值填充后的数组(不同色标识)：\n";

std::cout << " |";

for (int i = 0; i < DT[Column]; i++)

std::cout << " " << i << " ";

std::cout << "\n--+";

for (int i = 0; i < DT[Column]; i++)

std::cout << "---";

putchar('\n');

for (int i = 0; i < DT[Array]; i++) {

std::cout << char('A' + i) << " |";

for (int j = 0; j < DT[Column]; j++) {

if (option == 1)

std::cout << " " << Table[i][j] << " ";

else if (option == 2) {

if (MarkTable[i][j] == SIGNED)

std::cout << " \* ";

else

std::cout << " 0 ";

}

else if (option == 3 || option == 6) {

if (MarkTable[i][j] == SIGNED) {

setcolor(COLOR\_BLACK, COLOR\_HYELLOW);

std::cout << " " << Table[i][j] << " ";

setcolor(COLOR\_BLACK, COLOR\_HWHITE);

}

else

std::cout << " " << Table[i][j] << " ";

}

else if (option == 4) {

if (MarkTable[i][j] == SIGNED) {

setcolor(COLOR\_BLACK, COLOR\_HYELLOW);

if (i == DT[Orig\_y] && j == DT[Orig\_x])

std::cout << " " << Table[i][j] << " ";

else

std::cout << " 0 ";

setcolor(COLOR\_BLACK, COLOR\_HWHITE);

}

else

std::cout << " " << Table[i][j] << " ";

}

else if (option == 5) {

if (MarkTable[i][j] == SIGNED) {

setcolor(COLOR\_BLACK, COLOR\_HYELLOW);

std::cout << " 0 ";

setcolor(COLOR\_BLACK, COLOR\_HWHITE);

}

else

std::cout << " " << Table[i][j] << " ";

}

}

putchar('\n');

}

putchar('\n');

}

/\* Generate one Table \*/

void GenerateTable(int(\*Table)[10], int \*DT)

{

for (int i = 0; i < DT[Array]; i++)

for (int j = 0; j < DT[Column]; j++)

if (!Table[i][j])

Table[i][j] = GenerateRandom(DT[Level]);

}

/\* Calculate TotalGrade accroding to the command \*/

bool CombineNumber(int(\*Table)[10], int(\*MarkTable)[10], int \*DT)

{

int X = DT[Orig\_x], Y = DT[Orig\_y], tmp = 0;

char c;

cout << "请确认是否把相邻的相同值合并到A2中(Y/N/Q)：";

while (true) {

c = \_getch();

if (c == 'Y' || c == 'y') {

cout << c << endl;

for (int i = 0; i < DT[Array]; i++)

for (int j = 0; j < DT[Column]; j++)

if (MarkTable[i][j] == SIGNED) {

tmp += 3 \* Table[i][j];

if (i == Y && j == X)

Table[Y][X]++;

else

Table[i][j] = 0;

}

DT[TotalGrade] += tmp;

PrintTable(Table, MarkTable, DT, 4);

cout << "本次得分：" << tmp << " 总得分：" << DT[TotalGrade] << " 合成目标：" << DT[Goal] << endl;

putchar('\n');

UpdateTable(Table, MarkTable, DT);

PrintTable(Table, MarkTable, DT, 5);

InputEnter(5);

GenerateTable(Table, DT);

PrintTable(Table, MarkTable, DT, 6);

if (DT[Choice] == 3)

return false;

InputEnter(8);

memset(MarkTable, UNSIGNED, sizeof(int) \* 10 \* 10); //important!

return true;

}

else if (c == 'n' || c == 'N') {

cout << c << endl;

if (DT[Choice] == 3)

return false;

PrintTable(Table, MarkTable, DT, 1);

}

else if (c == 'q' || c == 'Q') {

cout << c << endl;

return false;

}

}

}

/\* Draw the Boarder of the graph \*/

void DrawBoarder(int x, int y, int col, int arr, int interval, bool flag)

{

setcolor(COLOR\_HWHITE, COLOR\_BLACK);

gotoxy(x, y);

int end\_i = 2 + 3 \* arr + flag \* (arr - 1);

int end\_j = 2 + 3 \* col + flag \* (col - 1);

int t = interval;

for (int i = 1; i <= end\_i; i++) {

for (int j = 1; j <= end\_j; j++) {

if (i == 1 && j == 1)

cout << "╔";

else if (i == 1 && j == end\_j)

cout << "╗";

else if (i == end\_i && j == 1)

cout << "╚";

else if (i == end\_i && j == end\_j)

cout << "╝";

else if (i == 1) {

if (flag && j % t == 1)

cout << "╤";

else

cout << "═";

}

else if (j == 1) {

if (flag && i % t == 1)

cout << "╟";

else

cout << "║";

}

else if (j == end\_j) {

if (flag && i % t == 1)

cout << "╢";

else

cout << "║";

}

else if (i == end\_i) {

if (flag && j % t == 1)

cout << "╧";

else

cout << "═";

}

else if (flag && j % t == 1 && i % t == 1)

cout << "┼";

else if (flag && i % t == 1)

cout << "─";

else if (flag && j % t == 1)

cout << "│";

else

cout << " ";

}

putchar('\n');

Sleep(30);

}

}

/\* Draw or change the figure \*/

void DrawBox(int x, int y, int num, int flag)

{

if (flag == 0)

setcolor(ColorNumber[num], COLOR\_BLACK);

else if (flag == 1)

setcolor(COLOR\_HYELLOW, COLOR\_HRED);

else if (flag == 2)

setcolor(COLOR\_HYELLOW, COLOR\_HBLUE);

else if (flag == 3)

setcolor(COLOR\_YELLOW, COLOR\_BLACK);

else if (flag = 9)

setcolor(COLOR\_HWHITE, COLOR\_HWHITE);

gotoxy(x, y);

cout << "╔═╗";

gotoxy(x, y + 1);

cout << "║" << setw(2) << num << "║";

gotoxy(x, y + 2);

cout << "╚═╝";

Sleep(30);

}

/\* Falling Movement of Box \*/

void FallingMovement(int(\*Table)[10], int \*DT, int i, int j, int k)

{

int x = 2 + i \* 6 + 2 \* i;

int y = 2 + k \* 3 + k;

int end = 2 + j \* 3 + j;

int counter = 0;

while (y < end) {

counter++;

DrawBox(x, y, Table[k][i], 9);

if (!(counter % 4)) {

showstr(x, y, "───", COLOR\_HWHITE, COLOR\_BLACK);

}

y++;

DrawBox(x, y, Table[k][i], 0);

}

}

/\* UpdateTable \*/

void UpdateTable(int(\*Table)[10], int(\*MarkTable)[10], int \*DT)

{

gotoxy(0, DT[Bottom] - 2);

setcolor(COLOR\_BLACK, COLOR\_HWHITE);

InputEnter(1);

int k;

MarkTable[DT[Orig\_y]][DT[Orig\_x]] = UNSIGNED;

if (Table[DT[Orig\_y]][DT[Orig\_x]] == DT[Level])

DT[Level]++; //upgrade

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 (Table[k][i]) {

if (MarkTable[k][i] == UNSIGNED) {

Table[j][i] = Table[k][i];

MarkTable[j][i] = MarkTable[k][i];

if (DT[Choice] == 8 || DT[Choice] == 9)

FallingMovement(Table, DT, i, j, k);

//Table[k][i] = 0;

MarkTable[k][i] = SIGNED;

break;

}

}

if (k < 0) {

Table[k][i] = 0;

MarkTable[j][i] = SIGNED;

}

}

}

}

/\* Falling Movement of Box \*/

void UpdateDrawing(int(\*Table)[10], int(\*MarkTable)[10], int \*DT)

{

int X = DT[Orig\_x], Y = DT[Orig\_y], tmp = 0;

int x, y;

setcolor(COLOR\_BLACK, COLOR\_HWHITE);

gotoxy(0, DT[Bottom] - 1);

showch(0, DT[Bottom] - 1, ' ', COLOR\_BLACK, COLOR\_HWHITE, CLEAR);//clear

for (int i = 0; i < DT[Array]; i++) {

for (int j = 0; j < DT[Column]; j++) {

if (MarkTable[i][j] == SIGNED) {

tmp += 3 \* Table[i][j];

x = 2 + j \* 6 + 2 \* j;

y = 2 + i \* 3 + i;

if (i == Y && j == X) {

Table[Y][X]++;

DrawBox(x, y, Table[Y][X], 0);

}

else {

Table[i][j] = 0;

DrawBox(x, y, 0, 9);

}

}

}

}

Sleep(100);

UpdateTable(Table, MarkTable, DT);

}

#pragma once

/\*1651574 1班 贾昊霖\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <iomanip>

#include <cstring>

#include <conio.h>

#include <windows.h>

#include <stdlib.h>

#include <ctime>

#include <cmath>

#include "cmd\_console\_tools.h"

using namespace std;

#define INITIAL\_X 25

#define INITIAL\_Y 50

#define UNSIGNED 0 //unchecked

#define SIGNED 1 //checked

#define TIMES 6

#define CLEAR 40

void UpdateTable(int(\*Table)[10], int(\*MarkTable)[10], int \*DT);

const int MOVE\_XY[4][2] = { { -1,0 },{ 0,-1 },{ 1,0 },{ 0,1 } };

const int ColorNumber[] = { 1,9,3,2,11,10,12,13,4,5,8,7,15 };//the order of colors

enum DataSet { Choice, Array, Column, Goal, TotalGrade, Level, Tmp\_x, Tmp\_y, Count, Orig\_x, Orig\_y, Bottom };

/\* input Enter \*/

void InputEnter(int x);

/\* generate random numer \*/

int GenerateRandom(int MaxNum);

/\* end sentences \*/

void EnterEnd();

/\* DFS\_recursion \*/

void DFS\_recursion(int(\*Table)[10], int(\*MarkTable)[10], int \*DT);

/\* judge the validity of character \*/

bool JudgeCommand(char \*command, int \*DT);

/\* 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);

/\* 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);

/\* Falling Movement of Box \*/

void FallingMovement(int(\*Table)[10], int \*DT, int i, int j, int k);

/\* UpdateTable \*/

void UpdateTable(int(\*Table)[10], int(\*MarkTable)[10], int \*DT);

/\* Falling Movement of Box \*/

void UpdateDrawing(int(\*Table)[10], int(\*MarkTable)[10], 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);

/\* Game Controler \*/

bool GameControl(int(\*Table)[10], int(\*MarkTable)[10], int \*DT);

/\* 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);

/\* Easter Egg \*/

void PrintEasterEgg();

#include "90-b2.h"

/\* input Enter \*/

void InputEnter(int x)

{

switch (x) {

case(0):

std::cout << "\n按回车键继续\n";

break;

case(1):

cout << "按回车键进行数组下落除0操作..." << endl;

break;

case(5):

cout << "按回车键进行新值填充..." << endl;

break;

case(8):

cout << "本次合成结束，按回车键继续新一次的合成..." << endl;

break;

}

while (\_getch() != '\r')

;

}

/\* generate random numer \*/

int GenerateRandom(int MaxNum)

{

int tmp = rand() % (20) + 1;

switch (MaxNum) {

case(1):case(2):case(3):

return rand() % 3 + 1;

case(4):

for (int i = 1; i <= 3; i++)

if (tmp <= i \* 6)

return i;

return 4;

case(5):

for (int i = 1; i <= 4; i++)

if (tmp <= i \* 5)

return i;

if (tmp <= 18)

return 4;

return 5;

case(6):

for (int i = 1; i <= 4; i++)

if (tmp <= i \* 4)

return i;

if (tmp <= 19)

return 5;

return 6;

default:

for (int i = 1; i <= MaxNum - 3; i++)

if (tmp <= i \* 16 / (MaxNum - 3))

return i;

if (tmp <= 18)

return MaxNum - 2;

if (tmp <= 19)

return MaxNum - 1;

return MaxNum;

}

}

/\* end sentences \*/

void EnterEnd()

{

int cur\_x, cur\_y;

char tmp[20];

std::cout << "本小题结束，请输入End继续...";

getxy(cur\_x, cur\_y);

while (true) {

cin >> tmp;

if (strlen(tmp) == 3 && (tmp[0] == 'e' || tmp[0] == 'E')\

&& (tmp[1] == 'n' || tmp[1] == 'N')\

&& (tmp[2] == 'd' || tmp[2] == 'D'))

break;

showstr(cur\_x, cur\_y, " ", COLOR\_BLACK, COLOR\_HWHITE);

std::cout << "\n输入错误，请重新输入.";

gotoxy(cur\_x, cur\_y);

}

showch(0, cur\_y + 1, ' ', COLOR\_BLACK, COLOR\_HWHITE, CLEAR);

}

/\* DFS\_recursion \*/

void DFS\_recursion(int(\*Table)[10], int(\*MarkTable)[10], int \*DT)

{

int x = DT[Tmp\_x], y = DT[Tmp\_y];

if ((x < 0 || x >= DT[Column] || y < 0 || y >= DT[Array]))

return;

//if (Table[x][y] == 0) // means to be (x<1 || x>Array || y<1 || y>Column)

// return;

if (Table[DT[Orig\_y]][DT[Orig\_x]] != Table[y][x] || MarkTable[y][x] != UNSIGNED)

return;

MarkTable[y][x] = SIGNED;

DT[Count]++; // counter

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

DT[Tmp\_y] += MOVE\_XY[i][0];

DT[Tmp\_x] += MOVE\_XY[i][1];

DFS\_recursion(Table, MarkTable, DT);

DT[Tmp\_y] -= MOVE\_XY[i][0];

DT[Tmp\_x] -= MOVE\_XY[i][1]; // important!

}

return;

}

/\*choose the solutions according to the main function\*/

void Solve(int(\*Table)[10], int(\*MarkTable)[10], int \*DataTable)

{

int \*&DT = DataTable;

setcolor(COLOR\_BLACK, COLOR\_HWHITE);

switch (DT[Choice]) {

case(1):case(2):

setconsoleborder(INITIAL\_Y, INITIAL\_X);

GenerateTable(Table, DT);

PrintTable(Table, MarkTable, DT, 1);

ExecuteCommand(Table, MarkTable, DT);

PrintTable(Table, MarkTable, DT, 2);

PrintTable(Table, MarkTable, DT, 3);

break;

case(3):case(4):

setconsoleborder(INITIAL\_Y, INITIAL\_X);

GenerateTable(Table, DT);

do {

PrintTable(Table, MarkTable, DT, 1);

ExecuteCommand(Table, MarkTable, DT);

PrintTable(Table, MarkTable, DT, 2);

PrintTable(Table, MarkTable, DT, 3);

} while (CombineNumber(Table, MarkTable, DT));

break;

case(5):

GenerateTable(Table, DT);

DrawFigure(Table, MarkTable, DT, false);

break;

case(6):

GenerateTable(Table, DT);

DrawFigure(Table, MarkTable, DT, true);

break;

case(7):

GenerateTable(Table, DT);

DrawFigure(Table, MarkTable, DT, true);

ChooseBox(Table, MarkTable, DT);

break;

case(8):case(9):

GenerateTable(Table, DT);

DrawFigure(Table, MarkTable, DT, true);

ChooseBox(Table, MarkTable, DT);

}

EnterEnd();

}

/\* Initial Settings \*/

void InitialSetting(int(\*Table)[10], int(\*MarkTable)[10], int \*DataTable)

{

int cur\_x, cur\_y;

std::cout << "请输入行数(5-8)：";

while (true) {

getxy(cur\_x, cur\_y);

std::cin >> DataTable[Array];

if (std::cin.good() && DataTable[Array] >= 5 && DataTable[Array] <= 8)

break;

showstr(cur\_x, cur\_y, " ", COLOR\_BLACK, COLOR\_HWHITE);

std::cin.clear();

std::cin.ignore(1024, '\n');

std::cout << "\n输入不合法，请重新输入\n";

gotoxy(cur\_x, cur\_y);

}

getxy(cur\_x, cur\_y);

showch(cur\_x, cur\_y, ' ', COLOR\_BLACK, COLOR\_HWHITE, CLEAR);

gotoxy(cur\_x, cur\_y);

std::cout << "请输入列数(5-10)：";

while (true) {

getxy(cur\_x, cur\_y);

std::cin >> DataTable[Column];

if (std::cin.good() && DataTable[Column] >= 5 && DataTable[Column] <= 10)

break;

showstr(cur\_x, cur\_y, " ", COLOR\_BLACK, COLOR\_HWHITE);

std::cin.clear();

std::cin.ignore(1024, '\n');

std::cout << "\n输入不合法，请重新输入\n";

gotoxy(cur\_x, cur\_y);

}

getxy(cur\_x, cur\_y);

showch(cur\_x, cur\_y, ' ', COLOR\_BLACK, COLOR\_HWHITE, CLEAR);

gotoxy(cur\_x, cur\_y);

std::cout << "请输入合成目标(5-20)：";

if (DataTable[Choice] != 1 && DataTable[Choice] != 5 && DataTable[Choice] != 6)

while (true) {

getxy(cur\_x, cur\_y);

std::cin >> DataTable[Goal];

if (std::cin.good() && DataTable[Goal] >= 5 && DataTable[Goal] <= 20)

break;

showstr(cur\_x, cur\_y, " ", COLOR\_BLACK, COLOR\_HWHITE);

std::cin.clear();

std::cin.ignore(1024, '\n');

std::cout << "\n输入不合法，请重新输入\n";

gotoxy(cur\_x, cur\_y);

}

memset(Table, 0, sizeof(int) \* 10 \* 10);

memset(MarkTable, UNSIGNED, sizeof(int) \* 10 \* 10);

DataTable[Level] = 3;

}

/\* Easter Egg \*/

void PrintEasterEgg()

{

printf("::\n :;J7, :, \

::;7:\n ,ivYi, , ;\

LLLFS:\n :iv7Yi :7ri;j5\

PL\n ,:ivYLvr ,ivrrirrY2X,\n\

:;r@Wwz.7r: :ivu@kexianli.\n \

:iL7::,:::iiirii:ii;::::,,irvF7rvvLujL7ur\n \

ri::,:,::i:iiiiiii:i:irrv177JX7rYXqZEkvv17\n \

;i:, , ::::iirrririi:i:::iiir2XXvii;L8OGJr71i\n :\

,, ,,: ,::ir@mingyi.irii:i:::j1jri7ZBOS7ivv,\n ,:\

:, ::rv77iiiriii:iii:i::,rvLq@huhao.Li\n ,, ,, \

,:ir7ir::,:::i;ir:::i:i::rSGGYri712:\n ::: ,v7r:: ::rrv7\

7:, ,, ,:i7rrii:::::, ir7ri7Lri\n , 2OBBOi,iiir;r:: \

,irriiii::,, ,iv7Luur:\n ,, i78MBBi,:,:::,:, :7FSL\

: ,iriii:::i::,,:rLqXv::\n : iuMMP: :,:::,:ii;2GY7OBB0v\

iiii:i:iii:i:::iJqL;::\n , ::::i ,,,,, ::LuBBu BBBBBEri\

i:i:i:i:i:i:i:r77ii\n , : , ,,:::rruBZ1MBBqi, :,,,\

:::,::::::iiriri:\n , ,,,,::::i: @arqiao. \

,:,, ,:::ii;i7:\n :, rjujLYLi ,,:::::,:::::::::,, ,:i,\

:,,,,,::i:iii\n :: BBBBBBBBB0, ,,::: , ,:::::: , ,,\

,, ,,:::::::\n i, , ,8BMMBBBBBBi ,,:,, ,,, , , , , ,\

:,::ii::i::\n : iZMOMOMBBM2::::::::::,,,, ,,,,,,:,,,::\

::i:irr:i:::,\n i ,,:;u0MBMOG1L:::i:::::: ,,,::, ,,, ::::::\

i:i:iirii:i:i:\n : ,iuUuuXUkFu7i:iii:i:::, :,:,: ::::::::i:i:\

::::iirr7iiri::\n : :rk@Yizero.i:::::, ,:ii:::::::i:::::i::,\

::::iirrriiiri::,\n : 5BMBBBBBBSr:,::rv2kuii:::iii::,:i:,,\

, ,,:,:i@petermu.,\n , :r50EZ8MBBBBGOBBBZP7::::i::,:::::,\

: :,:,::i;rrririiii::\n :jujYY7LS0ujJL7r::,::i::,:::::\

:::::::::iirirrrrrrr:ii:\n ,: :@kevensun.:,:,,,::::i:i::\

:::,,::::::iir;ii;7v77;ii;i,\n ,,, ,,:,::::::i:iiiii:\

i::::,, ::::iiiir@xingjief.r;7:i,\n , , ,,,:,,::::::::iiiiii\

iiii:,:,:::::::::iiir;ri7vL77rrirri::\n :,, , ::::::::i:::i\

:::i:i::,,,,,:,::i:i:::iir;@Secbone.ii:::\n");

std::cout << "再次把纯洁、爽朗而又不失礼节的笑容送给您\n";

}