**C++游戏“合成十”大作业**

**完成情况报告**

**/\* 2152189 信04 汪林辉 \*/**

**完成日期：2021.12.31**

**1.题目：综合题2 - 游戏“合成十”的实现**

**1.1 项目命名以及作业提交要求（以下文件均已提供模板）**

单机版：

cmd\_console\_tools.cpp ：图形演示界面下基本功能函数的具体实现(不提交)

cmd\_console\_tools.h ：图形演示界面下基本功能函数的声明(不提交)

90-b2-tools.cpp ：放内部数组或图形需要的公共函数

90-b2-console.cpp ：放cmd窗口各项图形演示的实现

90-b2-main.cpp ：放主函数及菜单部分函数

90-b2-base.cpp ：放与内部数组实现有关的各函数

90-b2.h ：放上述源程序的公共声明部分及其他所需内容

网络版：

lib\_md5\_sha\_tools.lib ：用于加密用户密码的静态库(不提交)

lib\_mto10\_net\_tools.lib ：用于进行网络传输的静态库(不提交)

lib\_mto10\_net\_tools.h ：用于网络传输的头文件(不提交)

1. b2-net.cpp ：放网络版对应的各函数(不提交)

（因本人能力有限未能完成网络版，故之后的报告将更侧重于普通版）

**1.2 随机值产生的规则**

初始/合并后最大值为3 ：等概率产生1~3

合并后最大值为4 ：1~3各30%，4为10%的概率

合并后最大值为5 ：1~3各25%，4为10%，5为10%概率

合并后最大值为6 ：1~4各20%，5为15%，6为5%概率

合并后最大值为x(x>6) ：1~x-3各20%，x-2为10%，x-1为5%，x为5%

**1.3 子题目划分**

（为降低难度，本体分为多个小题，每小题完成后可获得相应的分数）

**菜单1： 命令行方式找出可合并项并标识**

主要过程：键盘输入行列数，随机产生数组，随后输入要合并的行列坐标，找出所有可合成

的位置并用不同颜色显示出来。

注意：本小题中在一个矩阵中找相邻位置相同数的方法要求用一个函数实现，该函数实现为非递归形式。

提示：(如何标记相邻数据)可以用另一个相同大小的二维数组来记录标记，也可以将两个相同大小的二维数组放在同一结构体中集中管理。Demo输出了查找结果数组，可供参考。

**菜单2： 命令行方式找出可合并项并标识**

注意：本小题中在一个矩阵中找相邻位置相同数的方法要求用一个函数实现且该函数的实现方

式为递归形式 其余要求同菜单1

**菜单3： 命令行方式下完成一次合成**

合成选项：首先找出所有可合并的位置并标识，之后再确定是否需要合并

“N”：放弃本次操作 重新输入坐标并继续

“Y”：完成本次合并

“Q”：放弃游戏

如果某次合并后无法找到可合并位置 则自动提示游戏结束

具体过程：每次合成操作包括查找相邻项、合并相邻项、计算得分下落、消除零、在零位置产生新数据，要求在在程序实现时逐步打印出来。

**菜单4：命令行方式完整版**

每次合成操作的要求同子题目3，需要分步骤打印出来

合成到预期目标之后，给出提示信息，但不结束，可继续进行游戏 直至退出或无可合并项

**菜单5：cmd伪图形窗口显示内部数组的内容**

显示要求：键盘输入行列数随机产生数组，将数组的内容在cmd窗口中用伪图形显示出来，要求cmd窗口的大小随着输入的行列数动态变化（高度、宽度可适当放大2到3行/列）

注意：不同数字的前景色、背景色都不相同，具体设置可自行定义。

画图过程中适当加延时，具体延时值可自行定义，看清楚过程即可。

本菜单中数字色块中无分割线！

**菜单6：cmd伪图形窗口显示内部数组的内容**

本菜单中数字色块中有分割线，其余要求同菜单5

**菜单7：cmd伪图形窗口下的当前选择色块的选择**

在子题目6的基础上，用箭头键实现“当前选择”色块的选择，按回车确认

移动要求：初始位置定位左上角，用不同颜色刚出当前四原则色块越过边界后，采用环绕的方式（例如右箭头向右移动时到达最右侧时绕回左侧）

其他：为方便调试，延时可以取消。

注意：同时支持鼠标选择，左键表示确认（鼠标移动时，下方提示行的信息要及时变化）

**菜单8：cmd伪图形窗口下完成一次合成**

过程：箭头选择色块并回车键或左键选定合成位置，将所有可合成的色块标注出来选定后再次按箭头键则取消此本次选定重新选择，按回车则进行一次合成。一次完整的合成操作，包括合并相邻项、下落消除0、0位置产生新数据、计算得分，且要在程序实现时逐步展现出来。

注意：合成位置的色块颜色与其他可合成项要有区别

其中下落必须有动画效果下落

进行数填充、合成后下落是行优先还是列优先，可以与demo不同

**菜单9：cmd伪图形窗口完整版**

过程：每次合成操作的若干步骤间不再需要按键执行，但仍需要有动画效果。达到合成目标后，游戏不结束，合成目标+1后才结束。

注意：如果某次合并后无法找到可合并位置，则提示游戏结束；

cmd窗口的上下各有一个状态栏，显示得分、目标、操作提示等，具体内容可以和demo

不同，表达清楚即可。

**（最后：如果完成前9项，则满分为十分）**

由于我自己也没能完成网络版，因此这里也没有对网络版进行叙述

**1.4 编译器的要求**

仅要求VS2019编译通过，做到“0 errors，0 warnings”即可

**1.6 分辨率的要求**

在1920\*1080的屏幕下（FHD）显示正常

**1.7 无强制要求的内容**

1.字体、字号等无强制要求

2.标记相同值所用的内部数组无强制要求

3.画边框的顺序无强制要求

4.延时快慢无强制要求（建议比demo小，节约检查时间)，但必须达到动画效果

5.各种提示信息、状态栏的内容等无强制要求

6.被标识项、边框、每种数字、选中色块等的颜色无强制要求

7.出错时的各种提示无强制要求，清晰明了即可

8.本题是人工判题，不是自动判题（即:不必太在意细节处理)

**1.8 特别说明**

**1.8.1 命名**

要求项目名称为90-b2，项目由11个文件组成，要求编译生成的exe文件名必须是90-b2.exe(正常情况下不做特殊设置就是此文件名)。

**1.8.2 控制台**

已给出demo供参考。注意在本程序实现时cmd设置为旧版控制台 去除快速编辑模式和插入模式。

**1.8.3 其他**

1.本次作业不允许使用尚未讲授过的任何后续课程的知识点，包括但不限于指针、引用、结构体、类等概念!

2.整个程序不允许使用任何形式的全局变量、全局数组、全局指针，允许使用全

局的宏定义或常变量

3.不允许使用scanf/printf进行输入/输出

**2.整体设计思路**

主函数中先套上大的while循环，调用菜单函数后，若为0，直接结束return,否则进行行列目标的赋值，然后据菜单分别跳入普通版和图形豪华版两个准备函数。由于1~3、5~8分别为4、9的一部分，整体采用以前面的菜单为基础，通过if条件一层层判断叠加后续操作来实现，其中4和9需要外套while循环，几乎每个菜单结尾、甚至一些菜单内部都会有的“按回车键继续”零写函数传入暂停时想输出的句子来实现。

**3.主要功能的实现**

**3.1 主函数**

由于程序需要在一次游戏完成后能够连续运行，故设置：菜单项是0为跳出循环的唯一条件，后进入循环先需要进行cmd窗口的设置，之后判断菜单项是否小于5来判定时应当进入哪一种准备函数。

**3.2 菜单函数**

返回值为int，因为菜单的输入“有回显”，同时尝试着让屏幕上只显现出符合要求的输入,使用\_getch()函数无回显读入，若输入正确再cout输出这个数字（读入时的数字是char型，为方便才采用int 型，return时在之后减去’0’即可）

**3.3 两组准备函数**

为菜单为48准备好初始界面、进行延时的输入，完成后再进入hanoi递归函数

另外，直接实现菜单679而不进入循环

**3.4 变色函数**

分管三个功能，有参数解决。首先是选择时刷出变色（前景色为银白）；取消选择时恢复（默认前景色为黑）；最后是在进行消除后对相应项进行刷白，可通过判断附属函数内部相应位置是否被赋了\*来判断。

**3.5 产生随机数的函数**

可以先根据给定的概率来进行赋值，假设先产生20个数 然后20个数中有三个数被服成那个词所以这样就有了15%的概率，是一个非常好的想法。

**3.6 图形化构造函数**

分为屏幕准备打底画框和着色三部分。而且其中“画数字”则再分出一个函数picture，只需传入数组中的横纵坐标就能在屏幕上画出相应的颜色，然后在在画图函数中for循环将每个数组中的元素都画出来即可，同时再增加一个参数：零为无分割线，对应5；其他均打印分割线，对应剩余菜单。

**3.7读取鼠标和键盘的函数**

按照console tools里给的框架分成按下了键盘还是鼠标，然后再分别进行读取。由于此过程篇幅比较长且有很多变量、有循环，故简化过程把它抽离成一个函数。再通过指针（或者引用）赋值传出最后选了那个。而且为了在数组中对应行列更为方便，最后一行再执行一个函数返回究竟选择的是哪个坐标。

**3.8 判断鼠标或键盘在哪个框的函数**

分成三部分。第一、输入一个坐标，判断是否在数组的坐标里；第二、通过for循环套第一个函数，达到输入鼠标所处的一个坐标可判断是否在某个框里；第三、知道在某个框里后，进一步知道在哪个框里，即返回矩阵数组中的行列位置。

**4.调试过程碰到的问题**

**4.1 全局变量的重新定义**

最初的程序是副函数最初定义三个数组全局变量、计算总移动次数全局变量并都在定义时初始化。但后来调试时发现如果使用一次不退出直接回车进行下一次菜单时会导致盘子数量和上一次累加，此时才意识到这种情况下的全局变量不能定义时初始化，而应该把初始化放在输入层数、起始柱的函数里，保证每次重新进行菜单选取时全局变量们会焕然一新。

**4.2 控制生成随机数概率的问题**

因为目前所掌握的只有rand函数思维局限于在网上搜索是否有其他可以控制概率的函数，但最后是询问同学想法才得知可以使用赋值的概率来控制生成数字的概率，属于自己的想法不够灵活。

**4.3 函数内容的混乱性**

在输入行列和第一次错误后如果连续输入错误两次会陷入死循环而且如何调试也没有用处，最终在大群里询问老师、助教和同学，得出结果：当cin.get()函数读入数组时最后的回车没有被读走，因此下一次再读时读到回车就立即停止，陷入死循环并且由于第二次读取时 已经出现状态错误，返回EOF即使用忽略函数也没有办法。学到的是：经常cin,good判断状态,如cin.ignore()在cin.good()为0是会停止工作然后针对性的加cin.clear()等。

int main()

{

srand((unsigned int)(time(0)));

while (1) {

int choice = menu(); //为了保证End()在清屏后回到菜单界面，menu应该放在里面

if (choice == 0) //0是退出循环的唯一方式

break;

int ROW, COLUMN, goal; //大写为总行列数

int a[8][10] = { 0 }; //相较于while是全局变量为什么要初始化！！！！

char sub\_a[8][10] = { '\0' }; //相较于while是全局变量为什么要初始化！！！！

input\_RCG(&ROW, &COLUMN, &goal, choice);

initialize\_array(a, ROW, COLUMN);

if (choice<5)

prepare(choice, a, sub\_a, ROW, COLUMN, goal);

else

deluxe\_pre(choice, a, sub\_a, ROW, COLUMN, goal);

}

return 0;

}

void deluxe\_pre(int menu, int a[8][10], char sub\_a[8][10], int ROW, int COLUMN, int goal)

{

int Record\_X = 0, Record\_Y = 0; //用以记录落在数组哪个元素框内

cct\_enable\_mouse();

cct\_setcursor(CURSOR\_INVISIBLE);

deluxe\_draw(a, ROW, COLUMN, menu - 5); //菜单5参数为0,其余均非0

if (menu < 7) {

int position\_X = (menu == 5 ? (3 \* ROW + 2 + 3) : (3 \* ROW + ROW - 1 + 2 + 3));

cover(0, position\_X);

End();

}

else {

int choose\_X = 0, choose\_Y = 0;

int Q\_or\_not = 0;

int times = 1;

bian\_se(a[Record\_X][Record\_Y], Record\_X, Record\_Y, 0);

cover(0, 3 \* ROW + (ROW - 1) + 2 + 2);

cout << "箭头键/鼠标移动，回车键/单击鼠标左键选择并结束";

if (menu == 7) {

choose\_what\_coord(a, ROW, COLUMN, Record\_X, Record\_Y, 0, &choose\_X, &choose\_Y, &Q\_or\_not);

cct\_gotoxy(0, 3 \* ROW + (ROW - 1) + 2 + 3);

End();

}

else {

int fenshu = 0;

int zongfen = 0;

while (!Q\_or\_not) {

int k, i;

for (k = 0; k < ROW; k++)

for (i = 0; i < COLUMN; i++)

sub\_a[k][i] = '\0';

/\*判断游戏结束\*/

if (over(a, ROW, COLUMN)) {

cover(0, 3 \* ROW + (ROW - 1) + 2 + 2);

cct\_setcolor(BG\_COLOR, COLOR\_RED);

cout << "真可惜，没有可合并的项了，游戏结束！" << endl;

cct\_setcolor();

End();

break;

}

while (1) {

choose\_what\_coord(a, ROW, COLUMN, Record\_X, Record\_Y, times, &choose\_X, &choose\_Y, &Q\_or\_not);

times++;

if (Q\_or\_not) {

cover(0, 3 \* ROW + (ROW - 1) + 2 + 2);

cover(0, 3 \* ROW + (ROW - 1) + 2 + 3);

End();

return;

}

else {

if (check(a, choose\_X, choose\_Y)) {

seek\_same\_2(a, sub\_a, choose\_X, choose\_Y);

int k, i;

for (k = 0; k < ROW; k++) {

for (i = 0; i < COLUMN; i++) {

if (sub\_a[k][i] == '\*') {

bian\_se(a[k][i], k, i, 0);

}

}

Sleep(1);

}

cover(0, 3 \* ROW + (ROW - 1) + 2 + 2);

cout << "箭头键/鼠标移动取消当前选择，回车键/单击左键合成";

int out = 0;

int loop = 1;

int press\_keyboard = 0;

/\*loop的使用为内循环跳出外循环提供了一个解决的好办法\*/

while (loop) {

int X1, Y1;

int ret, maction;

int keycode1, keycode2;

ret = cct\_read\_keyboard\_and\_mouse(X1, Y1, maction, keycode1, keycode2);

if (ret == CCT\_MOUSE\_EVENT) {

if (in\_this\_kuang\_ma(choose\_X, choose\_Y, X1, Y1)) {

if (maction == MOUSE\_LEFT\_BUTTON\_CLICK) {

loop = 0;

out = 1; //又在框里又点了左键，跳出两个循环

break;

}

}

else {

loop = 0; //移出框框，跳出loop循环continue

break;

}

}

else

/\*我估摸着Record\_X即使针对整个main但完成choose函数后依然清零了,因此不用Record\_来计步\*/

if (ret == CCT\_KEYBOARD\_EVENT) {

switch (keycode1) {

case 224:

switch (keycode2) {

case KB\_ARROW\_UP: //上箭头

if (choose\_X > 0)

choose\_X--;

else

choose\_X = ROW - 1;

break;

case KB\_ARROW\_DOWN: //下箭头

if (choose\_X < ROW - 1)

choose\_X++;

else

choose\_X = 0;

break;

case KB\_ARROW\_LEFT: //左箭头

if (choose\_Y > 0)

choose\_Y--;

else

choose\_Y = COLUMN - 1;

break;

case KB\_ARROW\_RIGHT: //右箭头

if (choose\_Y < COLUMN - 1)

choose\_Y++;

else

choose\_Y = 0;

break;

}

/\*表示按下后以正常方式对待，但立即跳出loop循环contiue\*/

cover(0, 3 \* ROW + (ROW - 1) + 2 + 2);

cout << "[当前键盘] " << (char)('A' + choose\_X) << "行" << choose\_Y << "列";

bian\_se(a[choose\_X][choose\_Y], choose\_X, choose\_Y, 0);

/\*为保证循环性，此处赋这次移动的值给Record，以无缝进行下一次循环\*/

Record\_X = choose\_X;

Record\_Y = choose\_Y;

press\_keyboard = 1;

loop = 0;

break;

case 13: //回车键

loop = 0;

out = 1; //没有其他动作的按下回车，认为是选定了，跳出两个循环

break;

case 81:

case 113:

cover(0, 3 \* ROW + (ROW - 1) + 2 + 2);

cover(0, 3 \* ROW + (ROW - 1) + 2 + 3);

End();

return;

}

}

}