**目 录**

[1 选题意义 1](#_Toc28352486)

[1.1 意义 1](#_Toc28352487)

[1.2 目标 1](#_Toc28352488)

[2 系统功能模块设计 2](#_Toc28352489)

[2.1 功能模块设计 2](#_Toc28352490)

[2.2 功能模块间的联系 3](#_Toc28352491)

[3 主要功能的实现 4](#_Toc28352492)

[3.1 cut\_arry():二维数组分割模块 4](#_Toc28352493)

[3.2 play():对一维数组移动和运算 5](#_Toc28352494)

[3.3 over\_detection():判断二维数组中的数是否被非零数放满 7](#_Toc28352495)

[3.4 if\_over\_function():对无零的数组进行游戏结束判断 7](#_Toc28352496)

[4 测试及调试 8](#_Toc28352497)

[4.1 输出模块的调试 8](#_Toc28352498)

[4.2 随机数生成模块的测试 8](#_Toc28352499)

[4.3 数组单向移动和相同数字合并的测试 8](#_Toc28352500)

[4.4 结束条件的判断模块的测试 9](#_Toc28352501)

[4.5 游戏最终运行截图 9](#_Toc28352502)

[5 总结与心得 10](#_Toc28352503)

[6 附件 11](#_Toc28352504)

1 选题意义

1.1 意义

最近以来，移动手游越来越成为当下游戏产业中重要的一环，市场也在加大对这一产业的投入，涌现出了愤怒的小鸟，水果忍者，2048等诸多优秀的手游，然而任何一款手游能够风靡，主要还是依靠其简单的操作性，因此本次C语言课程设计我选择其中的典型2048小游戏作为本次课程设计的课题。

现代社会对休闲小游戏的需求是：提高游戏的操作可行性，降低游戏的操作难度，降低游戏的上手难度，降低游戏的竞争需求，对使用设备的要求趋于简单化和多样化，游戏界面更加人性化，功能更加贴合使用。而我所设计的2048小游戏就可以满足操作简单，趣味性高的要求。

我设计的2048游戏规则很简单，玩家每次可以选择上下左右其中一个方向去移动，每移动一次，所有的数字方块都会往移动的方向靠拢外，系统也会在空白的地方随机出现一个数字方块，相同数字的方块在靠拢、相撞时会相加。系统给予的数字方块不是2就是4（出现数字4的概率低），玩家要想办法在这小小的16格范围中凑出“2048”这个数字方块。能够显示游戏的当前得得分，并记录游戏的最高分并显示，当在次打开游戏时，能够显示最高分；游戏结束要有相应的提示：当所有格子被数字占满切不能再发生合并时，游戏结束，玩家失败；当游戏格子中出现2048数字时，游戏结束，玩家获得胜利。

1.2 目标

1．通过本课程设计，培养上机动手操作能力，使我能巩固《C语言程序设计》课程学习的内容，掌握工程软件设计的基本方法，强化上机动手能力，闯过编程关；

2．使我能更深入地理解和掌握该课程中的有关基本概念，程序设计思想和方法。为后续各门计算机课程的学习打下坚实基础。

3．培养个人勇于探索、严谨推理、实事求是、有错必改，用实践来检验理论，全方位考虑问题等科学技术人员应具有的素质。培养个人综合运用所学知识独立完成课题的能力

4．提高我对工作认真负责、一丝不苟，对同学团结友爱，协作攻关的基本素质.培养我从资料文献、科学实验中获得知识的能力，提高我从别人经验中找到解决问题的新途径的悟性，初步培养工程意识和创新能力。

5．对我个人掌握知识的深度、运用理论去处理问题的能力、实验能力、课程设计能力、书面及口头表达能力进行考核。

2 系统功能模块设计

2.1 功能模块设计

一．根据我的程序功能需求：

（1）游戏的主界面上，显示所有数字和方格。

（2）能够显示游戏的当前得得分，并记录游戏的最高分并显示，当在次打开游戏时，能够显示最高分。

（3）游戏的操作是通过上下左右移动来将相邻相同的数字相加，通过不断地累加最终加到2048。

（4）游戏过程中要随机刷新出2或4的数字。

（5）游戏过程中可以退出和重置。

（6）游戏结束要有相应的提示。

二．得出程序所需要设计的功能模块有：

1. 界面输出的功能模块
2. 记录最高分的功能模块
3. 键盘操作读取的功能模块
4. 计算和移动数字的功能模块
5. 随机数生成的功能模块
6. 重置和退出游戏的功能模块
7. 判断游戏是否结束的功能模块
8. 输出不同结束提示的功能模块

三．根据以上分析设计的函数模块有：

（1）print() 屏幕输出函数，输出整个界面和刷新输出界面

（2）play() 主要的计算函数，将分割的以为数组逐个进行运算

（3）gotoxy() 光标控制函数，用于控制输出的位置

（4）cut\_arry() 通过读取按键操作把二维数组分割为不同的一维数组

（5）resetgame() 重置游戏

（6）random\_data() 随机生成数字2或4

（7）over\_detection() 判断游戏方格中是否被占满

（8）if\_over\_fuction() 判断游戏是否结束

（9）gameover() 游戏是否结束提示

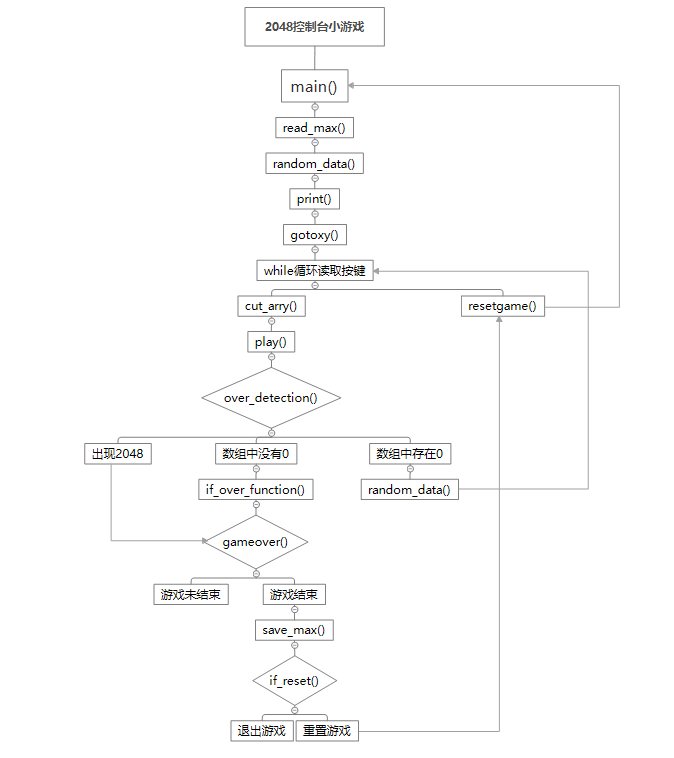
（10）if\_reset() 判断是否重置游戏

（11）save\_max() 保存最高分

（12）read\_max() 读取最高分

四．此程序为我个人单独设计，我担任所有的模块设计。

2.2 功能模块间的联系



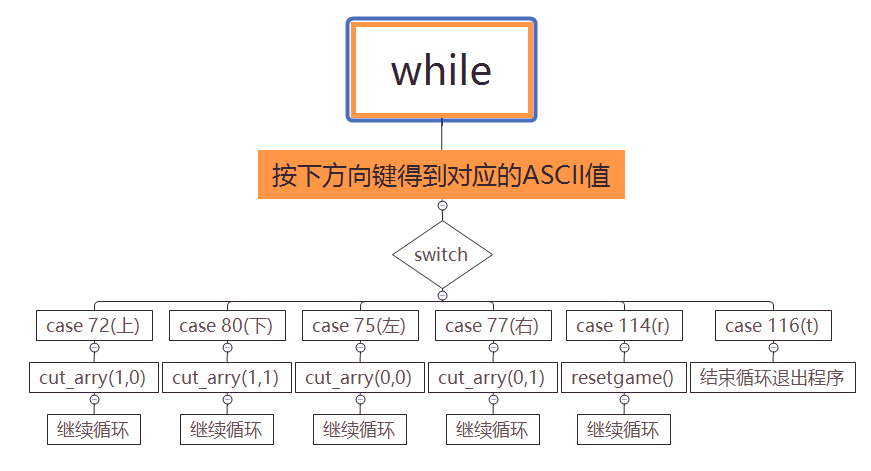
程序从主函数开始，主函数首先调用read\_max()函数，用于读取保存的最高分或首次运行时生成新的保存文件，然后调用两次random\_data()函数，用来在表格中重置两个随机数，之后random\_data()调用print()，print()函数会调用gotoxy()移动光标到初始位置，然后输出界面。界面输出结束后会立即进入while循环进行按键操作的读取，然后通过读取的按键值，按下方向键向cut\_arry()函数进行对应的参数传递；或者当检测到按下‘r’键进行游戏重置模块resetgame()。当向cut\_arry()传递参数后，cut\_arry()函数会通过对应的参数，把二维数组以不同的分割方法分割为一维数组。然后进行play()函数，将分割好的一维数组进行从右至左的移动和合并，移动合并结束后再将一维数组中的数放回原二维数组中，之后当cut\_arry()将二维数组分割并进行

运算后，就调用over\_detection()函数判断二维数组是否全部被非零的数字占满，如果没有被占满，游戏继续，调用random\_data()函数生成随机数、输出界面并继续进行while循环。如果被占满，就对二维数组分割，然后调用函数if\_over\_function()判断是否满足结束条件，如果满足结束条件，就会改变标记变量，判断结束后调用gameover()函数，验证标记是否发生变化，如果发生变化，就进行对应的提示输出，如果gameover()函数判断游戏结束（胜利\死亡），会终止while循环并调用if\_reset()函数询问用户是否重置游戏，如果用户输入‘Y/y’，就调用resetgame()函数清除数据然后运行main()函数重置游戏，否则输入‘N/n’游戏结束。如果gameover()函数判断游戏还可以继续，就输出提示语句，提示玩家还可以继续操作。游戏会一直运行下去，直至检测到游戏结束或玩家按下‘r’键重置游戏。

3 主要功能的实现

3.1 cut\_arry():二维数组分割模块

cut\_arry()函数有两个参数。通过main函数中的while循环读取按键和switch选择结构调用的。当按下方向键时，会返回对应的ascii值，然后通过case选择语句，选择对应的向cur\_arry()函数传递参数的方法，从而调用cut\_arry()函数对二维数组数组进行不同的分割和排序。

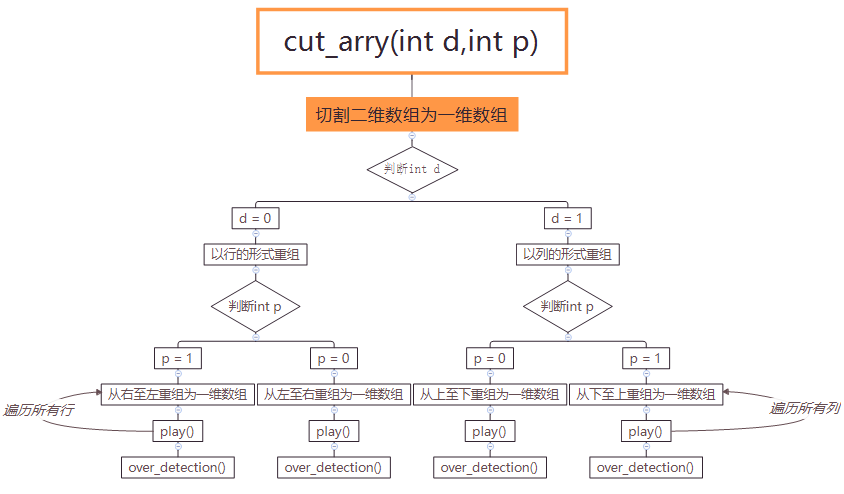


While

按下方向键生成对应的ASCII值

当cut\_arry()接收到传递的参数后，会将二维数组根据传递参数的不同切割为不同的一维数组。

然后将切割重组好的一维数组传递给play()函数进行运算。当所有行或列都已经运算后，就进行over\_detection()函数，用来判断二维数组是否非零的数占满。



分割二维数组为一维组

3.2 play():对一维数组移动和运算

play()函数有3个参数，分别为int arrs[]、int j、int z\_o\_f。arrs[]用于传递切割好的一位数组；j用于表示传递进来的一维数组是从二维数组的第j行/列切割出来的；z\_o\_f表示切割的一维数组的排序方式(从左到右、从上至下…),主要是为了标记将运算好的一维数组放回二维数组的顺序方式。

主要流程：

首先定义一个nextI变量用于存储下标并且定义初始值为-1。然后从下标为n=0处开始循环遍历整个数组，把下标n对应的数作为判断的变量，再从n+1处开始遍历整个数组中的数，寻找非零的数。

如果没有找到就结束内层循环,n++，如果找到，就让标记变量的值改变为对应的非零数的下标，然后结束循环。

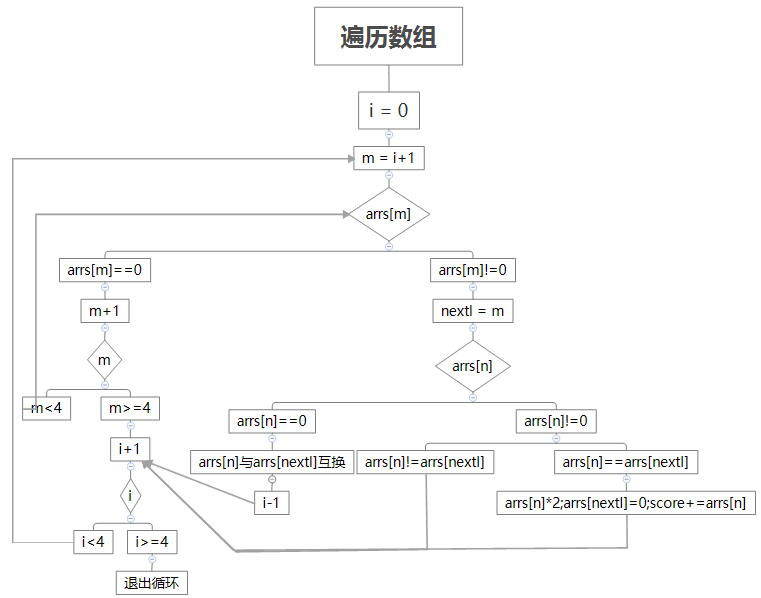
循环结束后，判断nextI的值是否发生改变，如果发生改变，就说明在n之后找到了一个非零数，然后判断n的位置对应的数是否为零。

如果为零就将找到的数移动到n的位置，然后再从n的位置再次向后寻找下一个非零的数。

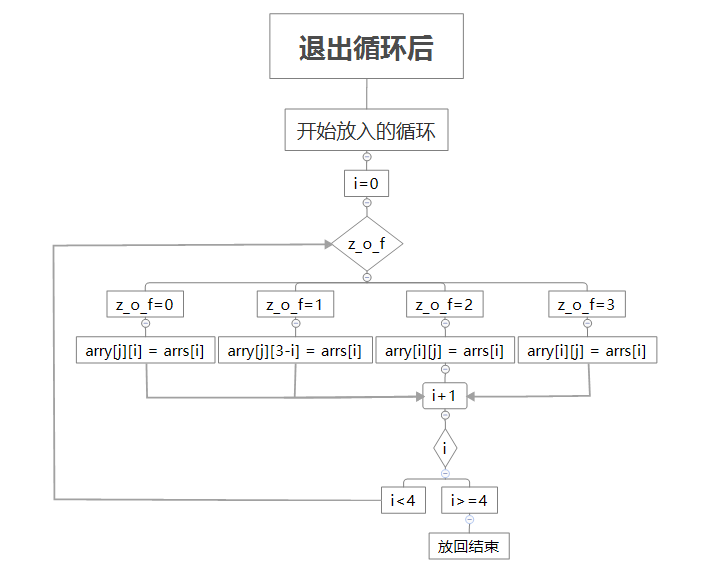
如果n的位置不为零，就要判断n对应的数和nextI对应的数是否相等。

如果相等就将n处的数乘以二（相当于两者相加），同时加入分数中，然后将nextI处的数变为0，n继续循环。

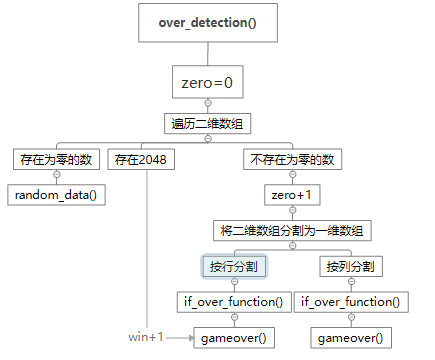
如果两者不相等就什么也不干，n继续循环，从而实现通过循环完成对一维数组的移动和数字相加合并的操作（合并完成的数是全部在一维数组左边的）。



当数组遍历结束后，一维数组中的数就会全部移动到数组的左端，并且已经完成了可以进行的运算操作。这时候再把一维数组中的数按照传入的参数z\_o\_f（放回原二维数组的方式）和j（放回第几行）放回原数组中：



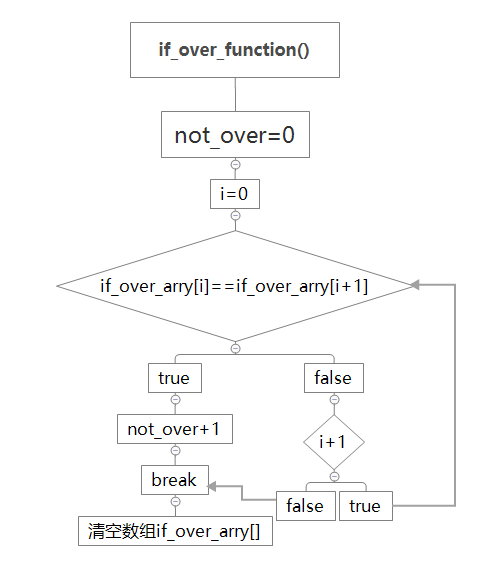
3.3 over\_detection():判断二维数组中的数是否被非零数放满



over\_detection()函数模块没有参数，其功能是判断play()函数结束后二维数组中的数是否被非零的数占满，如果没被占满就可以生成随机数，如果被占满就将数组分别按行和列分割存入全局的if\_over\_arry[行]和if\_over\_arry2[列]的一维数组然后调用函数if\_over\_fuction()进行游戏结束的判断。

3.4 if\_over\_function():对无零的数组进行游戏结束判断

if\_over\_function()没有参数，其功能是over\_detection()检测到数组中的数没有零的时候，这时候就要判断游戏是否结束，如果数组中的数出现相邻的两个数字相同，证明游戏还未结束，操作还可以继续，否则就是游戏结束，操作结束，玩家失败。



4 测试及调试

4.1 输出模块的调试

在设计程序时，主要也考虑了输出界面的美观性，在输出界面选择上也挑选了多种方案，刚开始是直接输出整个界面，然后在其中只改变数字，想以此形式完成界面的输出和数字的替换，后来多次测试后发现这样的情况下，当输出的数比价大的时候就会使图案变形，之后又想到了用循环输出，先输出不会变动的总体框架然后在其中通过循环输出数字和部分的分隔框架，并且针对于不同大小的数进行不同的输出样式，超过四位的数字用2的幂形式表示，这样就可以保证输出的界面不会发生改变。同时在每次输出前，调用gotoxy()函数将光标移动到屏幕的最开始的位置，使其覆盖上一次的输出，以显示出只改变数字的效果。

4.2 随机数生成模块的测试

在游戏开始以及游戏过程中会随机刷新出一个数字2或者4，其中2的生成概率要比4大，所以要同时保证找到的生成位置和生成的数都是十分随机的，刚开始我是随机在0-3之内随机取两个数然后通过判断是否为零生成再随机数，但是在测试过程中发现生成的数并不是很随机，有时候还会多次在同一区域或方格中重复生成，后来为了解决此难处，我通过百度知道了可以借助时间参数来生成一个随机种子，然后在通过随机种子生成两个随机数，这样产生的随机数是在时刻变换的，然后将生成的随机数取余与4，这样会产生一个0-4之间的数字，也减少了判断的时间量。然后借助生成的数字，判断二维数组中所在位置是否为0，不是就生成随机数，且生成随机数时，为了保证生出数字4和2的概率，也是通过随机种子进行一个计算，计算结果为2或3的时候生成4，否则生成2。

4.3 数组单向移动和相同数字合并的测试

游戏最核心的操作就是对数组的移动和移动过程中对值相同的数字进行相加合并。通过对方向按键的读取，会进行不同方向的移动，分为上、下、左、右四个方向，按下方向左键后，二维数组中的数都会移动到当前行的左边，并且会把其中相邻的相同的数相加，按下方向右键就向相反方向移动；当按下方向上键后二维数组中的数就会向当前列的顶部移动，且将相同数字合并，方向下键相反。

为了满足这个操作，刚开始我设想的是创建4个函数，每个按键对应一个函数，然后进行相应的操作，然后执行过程中发现这样写的代码量过于庞大，不适合后期的修改和错误的排查。

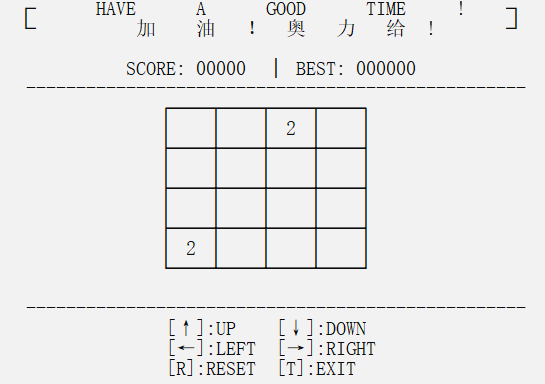
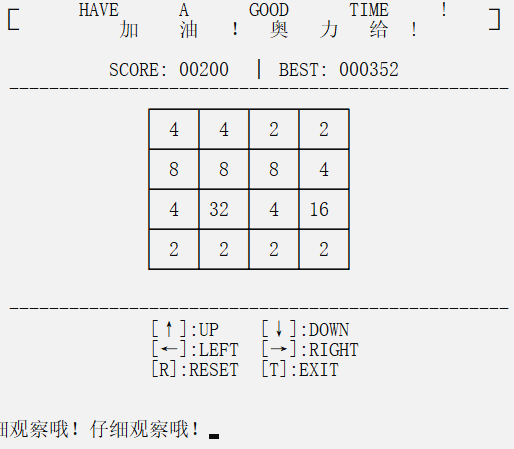
又通过一段时间的测试后，我发现这些功能都有相同的地方和重复的代码，就萌生了将操作分为两步的想法，将不同按键的操作都规整到两个函数中，一个函数负责将数组分割为一维数组，另一个函数负责将分割好的一维数组进行移动和相加并放回原二维数组中。于是我创建了一个cut\_arry()函数和一个play()函数，cut\_arry()函数负责将二维数组分割为一维数组，分割的方法依据传入的参数，参数d代表按行或列进行分割（按行分割代表左右移动，按列分割代表上下移动），参数p代表取出的顺序（从左至右或从右至左）:d和p一起控制着分割重组为一维数组的方法。当cut\_arry()分割完数组后就将数组传入到play()函数中，play()函数会对每一个每次传入的一维数组进行向左移动和数字合并的操作，操作结束后再根据参数将数组放回原来的二维数组中，从而逐一完成所有数组的操作。这种方式使代码量减少，同时很适合后期修改代码，方便管理。

4.4 结束条件的判断模块的测试

游戏有结束的条件，当数组中出现2048游戏结束，玩家胜利，当数组中的数字被非零的数字占满时且不可以再进行移动操作时，游戏结束，玩家失败。所以我需要监控数组中数字的变化，和判断数组中的数字是否被非零的数占满。刚开始不知道应该将这个模块放到哪个地方，后来多次测试后，发现在每次移动结束后进行一次数组中的数值判断是最合适的，因为移动后数组要生成随机数，但是如果没有空位，是生成不了随机数的，所以就要先判断是否还有空位，如果没有就不生成随机数，有就继续生成随机数还可以避免重复判断。

这个判断模块的测试也是花费了我大量的时间，也出现过bug，比如中间有一个全局的变量not\_over用于记录是否有相邻的相同的数，全局初始化为0，如果判断有相邻的相同数字就会加一，然后调用gameover()函数后，gameover函数会通过这个标记输出相应的提示，但是经过我的不断测试，发现当提示一次”请继续观察”(也就是数组中的数组虽满但是有相邻的相同数字)后，再次出现占满的情况后并且是真的不能再移动的时候游戏却在一直提示继续，原因是因为变量是全局的，修改一次后就会被改变，所以我又在每次判断前都将其重新赋值为零，顺利解决了这个漏洞。

4.5 游戏最终运行截图

5 总结与心得

在这次的课程设计中不仅检验了我所学习的知识，也培养了我的实践能力，让我知道遇到一个问题，如何去寻找思路，如何去解决问题，最终完成整个事情。课程设计是我们专业课程知识综合应用的实践训练，是我们迈向社会，从事职业工作前一个必不少的过程。这次课程设计不仅是对这学期所学知识的一种综合检验，而且也是对自己动手能力的一种提高，增强了自己实践能力。通过这次课程设计使我明白了自己知识还比较欠缺，只是学习书本知识还是远远不够的，自己不会的东西还有太多，学习需要自己长期的积累，在以后的学习、工作中都应该不断的学习，将课本的理论知识与生活中的实践知识相结合，不断提高自己文化知识和实践能力。由于本次试验的很多知识都是现学现用，以致很多地方的代码显得累赘繁琐。但总体上主要功能是实现了的，并且美化了一下主界面，使得界面更加简洁。

过而能改，善莫大焉。在课程设计过程中，我不断发现错误，不断改正，不断领悟，不断获取。最终的检测调试环节，本身就是在践行“过而能改，善莫大焉”的知行观。这次课程设计终于顺利完成了，在设计中遇到了很多问题，最后在老师和同学以及谷歌百度的指导下，终于游逆而解。在今后社会的发展和学习实践过程中，一定要不懈努力，不能遇到问题就想到要退缩，一定要不厌其烦的发现问题所在，然后一一进行解决，只有这样，才能成功的做成想做的事，才能在今后的道路上劈荆斩棘，而不是知难而退，那样永远不可能收获成功，收获喜悦，也永远不可能得到社会及他人对你的认可！

最后也再次感谢老师和同学给予的耐心指导与帮助！

6 附件

#include <stdio.h>

#include <stdlib.h>

#include <windows.h>

#include <conio.h> //按键操作

#include <time.h> //时间函数，用于获得时间从而获得时间种子

#define MAXRANDOMVALUE 3 //设置一个最大随机值，(每次得到的最大随机数)，因为4\*4二维数组的最大下标值为3，所以设置最大随机值为3

#define TXT\_FILE "max.txt" //定义一个文件用于保存历史最高分

//全局声明文件中所包括的功能模块，便于调用，避免多次声明

void print(); //屏幕输出函数，输出整个界面和刷新输出界面

int play(int arr[],int j,int z\_o\_f); //主要的计算函数，将分割的以为数组逐个进行运算，并得出结果

void gotoxy(int x,int y); //光标控制函数，用于控制输出的位置

void cut\_arry(int d,int p); //通过读取按键操作，把二维数组分割为不同的一维数组进行操作

void resetgame(); //重置游戏

void random\_data(); //随机生成数字2或4;

void over\_detection(); //判断游戏方格中是否被占满，如果方格满就通过if\_over\_fuction()判断游戏是否结束

void if\_over\_fuction(); //判断游戏是否结束

void gameover(); //游戏是否结束提示

void if\_reset(); //判断是否重置游戏

void save\_max(); //保存最高分

void read\_max(); //读取最高分

//声明全局变量做用于全局

int arry[4][4] = {0}; //声明一个4\*4的二维数组，用于抽象化的操作和存储数值

int if\_over\_arry[4] = {0}; //用于游戏结束判断操作

int if\_over\_arry2[4] = {0}; //用于游戏结束判断操作

int not\_over = 0; //用于标记游戏是否结束

int win = 0; //用于标记游戏是否获得胜利

int score = 0; //用于计算所得分数

int best = 0; //用于存储游戏最历史高分

int cycle = 1; //用于标记按键是否需要继续操作

int main()

{

read\_max(); //读取文件中保存的最高分

random\_data(); //开局随机生成两个数在表格中

random\_data();

int ch=0;

while (cycle) //cycle 是全局的变量，开始为 1 (死循环)；游戏结束或退出时变为 0 (结束循环)

{

ch=getch(); //将读取的值赋值给ch

switch (ch)

{

case 72: cut\_arry(1,0);break; //按下方向上键，数字整体向上移动并计算

case 80: cut\_arry(1,1);break; //按下方向下键，数字整体向下移动并计算

case 75: cut\_arry(0,0);break; //按下方向左键，数字整体向左移动并计算

case 77: cut\_arry(0,1);break; //按下方向右键，数字整体向右移动并计算

case 114: resetgame();break; //r -> 重置程序

case 116: cycle = 0;break; //t -> 退出程序

break;

}

}

}

void save\_max() //保存最高分

{

FILE \*fp;

fp=fopen(TXT\_FILE,"w");

fprintf(fp,"%d",best);

fclose(fp);

}

void read\_max() //读取最高分

{

FILE \*fp;

fp=fopen(TXT\_FILE,"r");

fscanf(fp,"%d",&best);

fclose(fp);

}

void over\_detection() //游戏结束判断

{

int i,j;

int zero = 0; //初始定义二维数组中没有 0 也就是没有空位

for (i=0;i<4;i++) //循环判断数组中是否有 0（空格）

{

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

{

if (arry[i][j] == 2048) //如果数组中的数字有 2048 游戏结束，玩家胜利

{

win += 1; //标记游戏胜利

gameover();

break;

}

else if (arry[i][j] == 0)

{

zero += 1; //如果有 0（空格）就将标记改变

break;

}

}

}

if (zero == 0) //如果没有找到空格，就要判断游戏是否结束

{

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

{

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

{

if\_over\_arry[j] = arry[i][j]; //将二维数组依次分割为行

if\_over\_arry2[j] = arry[j][i]; //将二维数组依次分割为列

}

if\_over\_fuction(); //调用函数对分割后的一维数组进行结束判断

}

gameover(); //通过判断后的结果来输出对应的提示信息

}

else //否则继续生成随机数

{

random\_data();

}

}

void if\_over\_fuction() //判断游戏是否结束

{

not\_over = 0;

int i;

//初始定义标记not\_over为结束(0)

for (i=0;i<3;i++) //循环判断一维数组的状态

{

if ((if\_over\_arry[i] == if\_over\_arry[i+1]) || (if\_over\_arry2[i] == if\_over\_arry2[i+1])) //如果某一个数字相邻的位置与其相等，就代表游戏未结束

{

not\_over += 1; //标记未结束

break;

}

}

memset(if\_over\_arry,0,sizeof(if\_over\_arry)); //清空判断时用于存储的数组，方便进行下一次判断

memset(if\_over\_arry2,0,sizeof(if\_over\_arry2));

}

void random\_data()

{

int no;

srand(time(NULL)); //利用系统时间返回一个随机数

int i,j;

do

{

i=(rand())%(MAXRANDOMVALUE+1); //通过取余数来得到一个整数，切随机整数的最大值不会超过MAXRANDOMVALUE

j=(rand())%(MAXRANDOMVALUE+1);

}while(arry[i][j]!=0); //如果随机生成的坐标处没有数字(0),就在那个位置随机生成数字

no=2\*((rand ()%10)+1); //通过判断随机数的余数来生成2或4，生成4的几率比2小

if(no==3 || no==2)

{

arry[i][j]=4;

}

else

{

arry[i][j]=2;

}

print();

}

void resetgame() //重置游戏

{

int i,j;

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

{

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

{

arry[i][j]=0; //将数组中的所有数字重置为 0

}

}

system("cls"); //清屏

score = 0; //将分数清零

cycle = 1; //将循环标记重置

not\_over = 0; //重置未结束标记

win = 0; //重置获胜标记

main();

}

void gotoxy(int x,int y) //光标控制函数，控制光标到控制台的某一(初始)位置

{

//调用win32 API 去设置控制台的光标位置

//1.找到控制台的窗口

HANDLE handle=GetStdHandle(STD\_OUTPUT\_HANDLE);

//2.光标结构体

COORD coord;

//3.设置坐标

coord.X=x;

coord.Y=y;

//4.同步到控制台/设置控制台光标的位置

SetConsoleCursorPosition(handle,coord);

}

void print() //界面打印输出函数

{

if (score >= best) //如果当前分数比最高分高，就重置最高分

{

best = score;

}

gotoxy(0,0); //控制光标到初始位置

printf("\t ┌ HAVE A GOOD TIME ! ┐\n");

printf("\t └ 加 油 ！ 奥 力 给 ! ┘\n\n");

printf("\t\t SCORE: %05d │ BEST: %06d\n", score, best);

printf("\t\t--------------------------------------------------");

/\* 绘制方格和数字 \*/

printf("\n\t\t\t ┌────┬────┬────┬────┐\n");

int i;

for (i = 0; i < 4; i++) //用循环输出界面

{

printf(" │");

int j;

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

{

if (arry[i][j] != 0)

{

if (arry[i][j] < 10)

{

printf(" %d │", arry[i][j]);

}

else if (arry[i][j] < 100)

{

printf(" %d │", arry[i][j]);

}

else if (arry[i][j] < 1000)

{

printf(" %d│", arry[i][j]);

}

else if (arry[i][j] < 10000)

{

printf("%4d│", arry[i][j]);

}

else

{

int n = arry[i][j];

int k;

for (k = 1; k < 20; ++k)

{

n = n >> 1;

if (n == 1)

{

printf("2^%02d│", k); /\* 超过四位的数字用2的幂形式表示，如2^13形式 \*/

break;

}

}

}

}

else printf(" │");

}

if (i < 3)

{

printf("\n\t\t\t ├────┼────┼────┼────┤\n");

}

else

{

printf("\n\t\t\t └────┴────┴────┴────┘\n");

}

}

printf("\n");

printf("\t\t--------------------------------------------------\n");

printf("\t\t [↑]:UP [↓]:DOWN\n");

printf("\t\t [←]:LEFT [→]:RIGHT\n");

printf("\t\t [R]:RESET [T]:EXIT\n\n\n");

}

void cut\_arry(int d,int p) //将二维数组切割为一维数组在进行计算，通过参数控制切割方式(d,p的值为0,1)

{

int i,j;

if (d == 0 && p == 0) //d=0表示切割的是行,p=0表示将数字按从左到右取出并从左到右放置在新的一维数组中(按下方向左键)

{

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

{

int arrys[4] = {};

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

{

arrys[j] = arry[i][j];

}

play(arrys,i,0); //将切割的数组进行计算移动

}

over\_detection(); //判断移动后的数组是否结束

}

else if (d == 0 && p == 1) //d=0表示切割的是行,p=1表示将数字按从左到右取出并从右到左放置在新的一维数组中(按下方向右键)

{

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

{

int arrys[4] = {};

for (j=3;j>=0;j--)

{

arrys[3-j] = arry[i][j];

}

play(arrys,i,1);

}

over\_detection();

}

else if (d == 1 && p == 0) //d=1表示切割的是行,p=1表示将数字按从上到下取出并从左到右放置在新的一维数组中(按下方向上键)

{

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

{

int arrys[4] = {};

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

{

arrys[j] = arry[j][i];

}

play(arrys,i,2);

}

over\_detection();

}

else

{

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

{

int arrys[4] = {};

for (j=3;j>=0;j--)

{

arrys[3-j] = arry[j][i]; //否则将数字按从上到下取出并从右到左放置在新的一维数组中(按下方向下键)

}

play(arrys,i,3);

}

over\_detection();

}

}

int play(int arrs[],int j,int z\_o\_f) //j代表二维数组的行或列，z\_o\_f代表将计算后的数字放回二维数组方式

{

int i,nextI,m; //定义i和m作为循环控制变量;定义indexI作为存储下标的变量

for (i = 0; i < 4; i++) //遍历数组，从下标为0的值开始

{

nextI = -1; //先为indexI定义一个不为[0,4]的值，用于保存目标值的下标和条件判断，每次循环indexI都重置为-1

for (m = i+1; m < 4; m++) //从arry[i]自身向数组后面遍历，直到找到除自身外第一个不为0的数字，如果没有，就什么也不做

{

if(arrs[m] != 0)

{

nextI = m;

break;

}

}

if (nextI !=-1)

{

//存在下个不为0的位置

if (arrs[i] == 0) //如果arry[i]的数是0，就把这个数和找到的第一个数交换，然后再从arry[i]的位置开始往后寻找下一个数字

{

arrs[i] = arrs[nextI];

arrs[nextI] = 0;

i--; //目的是让循环再次从i开始，找到下一个不为零的数字，(i--与循环结束后的i++抵消)

}

else if (arrs[i] == arrs[nextI]) //如果arry[i]和找到的那个数字相等，arry[i]就自身乘以2，并将找到的数字的值变为0，相当于两者相加并将值放在arry[i]中

{

arrs[i] \*= 2;

arrs[nextI] = 0;

score += arrs[i];

}

//如果上面两个条件都不符合，也就是指找到的两个数不相等，就什么也不做

}

}

for (i=0;i<4;i++) //将移动计算后的数组中的数字放回二维数组中

{

//z\_o\_f 用于标记放回的方式

if (z\_o\_f == 0) //z\_o\_f=0表示将一维数组从左到右放置到二维数组的第J行从左到右的位置

{

arry[j][i] = arrs[i];

}

else if (z\_o\_f == 1) //z\_o\_f=1表示将一维数组从左到右放置到二维数组的第J行从右到左的位置

{

arry[j][3-i] = arrs[i];

}

else if (z\_o\_f == 2) //z\_o\_f=2表示将一维数组从左到右放置到二维数组的第J列从上到下的位置

{

arry[i][j] = arrs[i];

}

else //z\_o\_f=表示将一维数组从左到右放置到二维数组的第J列从下到上的位置

{

arry[3-i][j] = arrs[i];

}

}

}

void gameover() //游戏结束提示

{

if (not\_over == 0) //如果未结束的标记为 0，代表游戏结束，玩家失败

{

save\_max(); //保存最高分

cycle = 0; //结束按键读取操作

system("cls"); //清屏

printf(" \t\t GAME OVER !\n\n\

\t\t YOU SCORE IS <%5d> \n\n\n",score);

if\_reset(); //询问玩家是否重置游戏

}

else if (not\_over != 0) //如果未结束的标记不为 0，代表游戏未结束，玩家还可以移动，输出提示信息

{

printf("仔细观察哦！");

}

else if (win != 0) //如果获胜的标记不为 0，代表游戏结束，玩家胜利

{

save\_max(); //保存最高分

cycle = 0; //结束按键读取循环

system("cls"); //清屏

printf("YOU WIN !!! \n");

if\_reset(); //询问玩家是否重置游戏

}

}

void if\_reset() //询问玩家是否重置游戏

{

char ifreset;

printf("DO YOU WANT RESET THE GAME(y/n)?\n");

scanf("%s",&ifreset);

if (ifreset == 'y' || ifreset == 'Y') //玩家输入 y/Y 重置游戏

{

resetgame();

}

else //否则就结束游戏

{

printf("\nOK ! GOOD BYE !");

}

}