# 上机实验9

| | |
|---|---|
| Created | @December 14, 2022 11:53 PM |
| Class | 程序设计基础 |
| Type | homework |
| Materials | |
| Reviewed | ☐ |
| Person | |

姓名：孙弋戈

学号：2022403354

班级：计算机系交换生

```
//源代码
//  main.cpp
//  2048
//
//  Created by Yige Sun on 2022/12/14.
//

#include <iostream>
#include <ctime>
#include <cmath>
#include <cstdlib>
#include <fstream>
#include <stdio.h>
#include <stdlib.h>
using namespace std;
//Global Variables
int arrayList[16];
//Check whether the game can continue
//(there's still one or more than one spaces left, or )
int gameStatus = 0;

//FUNCTIONS
void readInstruct();
void play(char input);
//draw() function group
void draw();
int countLength(int num);
void printLine(int count);
void origin();
int getRand();
void newAppear();
//meltUp() function group
void meltUp();
void upOrderZero();
void upMelt();
```

```
//meltDown() function group
void meltDown();
void downOrderZero();
void downMelt();
//meltLeft() function group
void meltLeft();
void leftOrderZero();
void leftMelt();
//meltRight() function group
void meltRight();
void rightOrderZero();
void rightMelt();

//main function
int main() {
    //read the instructs
    readInstruct();
    return 0;
}


void readInstruct(){
    ifstream fin;
    fin.open("cmd.txt");
    //test whether can read instructs from the file
    int test = 0;
    //if the file can be opened
    if(fin){
        //if the file can be opened and the file is not empty,
        //test should be >0
        while(!fin.eof()){
            string instruct;
            cin >> instruct;
            test++;
        }
    }else{
        //if the file can be opened but is empty, test=0
        test = 0;
    }
    fin.close();
    //if test=0, which means the file is not existed or empty,
    //the user will use keyboard-input
    if(test == 0){
        int input;
        system("stty -icanon");
        system("stty -echo");
        printf("Please input i (up) /k (down) /j (left) /l (right)/q (quit) :\n");
        //initialize arraylist
        origin();
        //show the original game page to the user
        draw();
        while((input = getchar()) != 'q'){
            //input instruct to decide which direction to melt
            play(input);
            //if there's still space, the game will continue
            if(gameStatus == 0){
                continue;
            }else{
                //if there's no space(gameStatus == 1), the game will be over
                cout << "Game Over";
                break;
            }
```

```
        }
        system("stty icanon");
        system("stty echo");
    }
}

void play(char input){
    if(input == 'i'){
        meltUp();
    }
    if(input == 'k'){
        meltDown();
    }
    if(input == 'j'){
        meltLeft();
    }
    if(input == 'l'){
        meltRight();
    }
    //check whether there's any space, if yes, test=1
    int test = 0;
    for(int i=0; i<16; i++){
        if(arrayList[i] == 0){
            test = 1;
        }
    }
    //if test != 1, it means that there's no space.
    //The game will be over
    if(test == 0){
        gameStatus = 1;
        return;
    }else{
        gameStatus = 0;
    }
    //a new random number(2 or 4) will appear on a random empty space
    newAppear();
    //output the matrix
    draw();
}

void meltUp(){
    //move all non-zero number to the top
    upOrderZero();
    //modify the number pairs
    upMelt();
    //move all non-zero number to the top
    upOrderZero();
}

void upOrderZero(){
    for(int i = 0; i<4; i++){
        int zeroNum = 0;
        for(int j=0; j<4; j++){
            //calculate how many 0s in one row
            if(arrayList[i + 4*j] == 0){
                zeroNum ++;
            }
            else{//if is not 0, move it up with the distance of the number of 0s
                arrayList[i + 4*j - 4*zeroNum] = arrayList[i + 4*j];
            }
        }
        //we have known about how many 0s we have, so we change the top numbers to be 0
```

```
        for(int j = 0; j < zeroNum; j++){
            arrayList[12 + i - j*4] = 0;
        }
    }
}
void upMelt(){
    for(int i=1; i<4; i++){
        for(int j=0; j<4; j++){
            //if the number and the number above it is the same, and both of them are not 0,
            //double the above one
            if ((arrayList[j + 4*i] != 0) &&
                (arrayList[j + 4*i] == (arrayList[j + 4*(i-1)]))){
                arrayList[j + 4*i] = 0;
                arrayList[j + 4*(i-1)] = 2 * arrayList[j + 4*(i-1)];
            }
        }
    }
}

void meltDown(){
    downOrderZero();
    //modify the number pairs
    downMelt();
    downOrderZero();
}

void downOrderZero(){
    for(int i = 0; i<4; i++){
        int zeroNum = 0;
        for(int j=3; j>=0; j--){
            //calculate how many 0s in one row
            if(arrayList[i + 4*j] == 0){
                zeroNum ++;
            }
            else{//if is not 0, move it down with the distance of the number of 0s
                arrayList[i + 4*j + 4*zeroNum] = arrayList[i + 4*j];
            }
        }
        //we have known about how many 0s we have, so we change the top numbers to be 0
        for(int j = 0; j < zeroNum; j++){
            arrayList[i + j*4] = 0;
        }
    }
}
void downMelt(){
    for(int i=2; i>=0; i--){
        for(int j=0; j<4; j++){
            //if the number and the number below it is the same, and both of them are not 0,
            //double the below one
            if ((arrayList[j + 4*i] != 0) &&
                (arrayList[j + 4*i] == (arrayList[j + 4*(i+1)]))){
                arrayList[j + 4*i] = 0;
                arrayList[j + 4*(i+1)] = 2 * arrayList[j + 4*(i+1)];
            }
        }
    }
}

void meltRight(){
    rightOrderZero();
    //modify the number pairs
    rightMelt();
```

```
    rightOrderZero();
}

void rightOrderZero(){
    for(int i = 0; i<4; i++){
        int zeroNum = 0;
        for(int j=0; j<4; j++){
            //calculate how many 0s in one column
            if(arrayList[3 - j + i*4] == 0){
                zeroNum ++;
            }
            else{//if is not 0, move it right with the distance of the number of 0s
                arrayList[3-j + zeroNum + i*4] = arrayList[3-j + i*4];
            }
        }
        //we have known about how many 0s we have, so we change the left numbers to be 0
        for(int j = 0; j < zeroNum; j++){
            arrayList[j + i*4] = 0;
        }
    }
}
void rightMelt(){
    for(int i=0; i<4; i++){
        for(int j=2; j>=0; j--){
            //if the number and the number on the right side of it is the same, and both of them are not 0,
            //double the right side one
            if ((arrayList[j + i*4] != 0) &&
                (arrayList[j + i*4] == (arrayList[j + 1 + i*4]))){
                arrayList[j + i*4] = 0;
                arrayList[j+1 + i*4] = 2 * arrayList[j+1 + i*4];
            }
        }
    }
}

void meltLeft(){
    leftOrderZero();
    //modify the number pairs
    leftMelt();
    leftOrderZero();
}

void leftOrderZero(){
    for(int i = 0; i<4; i++){
        int zeroNum = 0;
        for(int j=0; j<4; j++){
            //calculate how many 0s in one column
            if(arrayList[j + i*4] == 0){
                zeroNum ++;
            }
            else{//if is not 0, move it left with the distance of the number of 0s
                arrayList[j - zeroNum + i*4] = arrayList[j + i*4];
            }
        }
        //we have known about how many 0s we have, so we change the left numbers to be 0
        for(int j = 0; j < zeroNum; j++){
            arrayList[3-j + i*4] = 0;
        }
    }
}
void leftMelt(){
    for(int i=0; i<4; i++){
```

```
        for(int j=1; j<4; j++){
            //if the number and the number on the left side of it is the same, and both of them are not 0,
            //double the left side one
            if ((arrayList[j + i*4] != 0) &&
                (arrayList[j + i*4] == (arrayList[j - 1 + i*4]))){
                arrayList[j + i*4] = 0;
                arrayList[j-1 + i*4] = 2 * arrayList[j -1 + i*4];
            }
        }
    }
}

void newAppear(){
    srand((unsigned int)time (NULL));
    //get a random number from 0 to 15 to be the new random position
    int newPos = rand()%16;
    //test whether the position is empty
    while(arrayList[newPos] != 0){
        newPos = rand()%16;
    }
    //get a new number between 2 and 4
    int newNum = getRand();
    arrayList[newPos] = newNum;
}

void draw(){
    cout << endl;
    int largestNum = 0;
    //get the largest number in the arraylist and count the length of it
    for(int i=0; i < 16; i++){
        if(arrayList[i] > largestNum){
            largestNum = arrayList[i];
        }
    }
    int largestCount;
    largestCount = countLength(largestNum);
    //print the line of matrix above each number. This line will change
    //within the length of the largest number
    printLine(largestCount);
    //print "|" between each number the framework
    for(int i = 0; i < 16; i++){
        cout << "|";
        //change line if the number is the last num in a line
        if((i + 1) % 4 == 0){
            int count = countLength(arrayList[i]);
            int blankNeed = largestCount - count;
            for(int j=0; j < blankNeed; j++){
                cout << " ";

            }
            if(arrayList[i] != 0){
                cout << arrayList[i] << "|" << endl;
            }else{
                cout << " " << "|" << endl;
            }
            printLine(largestCount);
        }else{
            int count = countLength(arrayList[i]);
            int blankNeed = largestCount - count;
            for(int j=0; j < blankNeed; j++){
                cout << " ";
            }
```

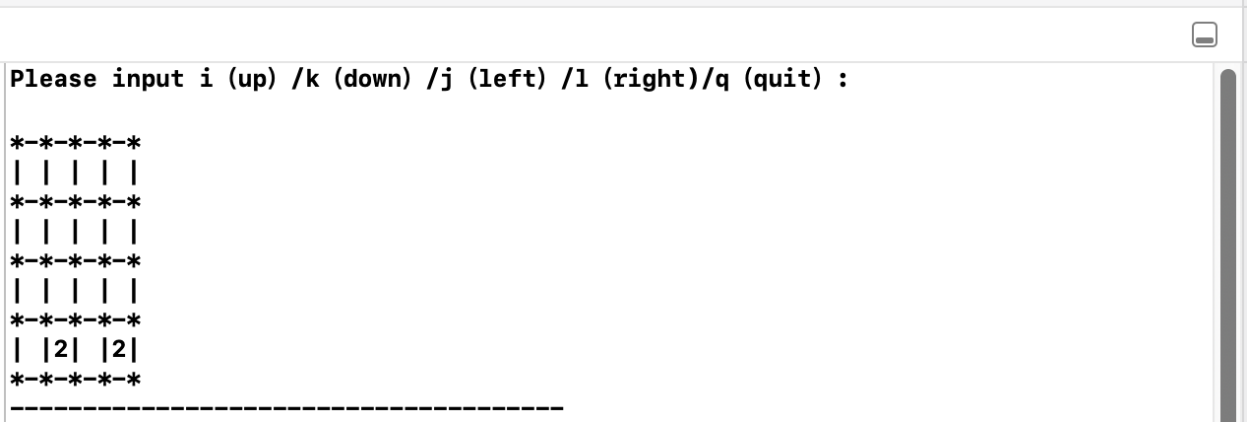
```
            if(arrayList[i] != 0){
                cout << arrayList[i];
            }else{
                cout << " ";
            }
        }
    }
    cout << "-------------------------------------" << endl;
}


int countLength(int num){
    int count =0;
    if(num == 0){
        count++;
    }else{
        while(num!=0){
            num=num/10;
            count ++;
        }
    }
    return count;
}


void printLine(int count){
    for(int i=0; i < 4; i++){
        cout << "*";
        for(int i=0; i<count; i++){
            cout << "-";
        }
    }
    cout << "*" << endl;
}

//initialize arraylist
void origin(){
    srand((unsigned int)time (NULL));
    //get the 1st random num
    int randPos1 = rand()%16;
    int randNum1 = getRand();
    arrayList[randPos1] = randNum1;
    //get the 2nd random num
    int randPos2 = rand()%16;
    while(randPos2 == randPos1){
        randPos2 = rand()%16;
    }
    int randNum2 = getRand();
    arrayList[randPos2] = randNum2;
    for(int i=0; i<16; i++){
        if((arrayList[i] != 2) && (arrayList[i] != 4)){
            arrayList[i] = 0;
        }
    }
}

//choose a number randomly from 2 and 4
int getRand(){
    srand((unsigned int)time (NULL));
    int randNum = rand()%2;
    int twoOrFour =4;
    if(randNum == 1){
```

```
            twoOrFour = 2;
        }
        return twoOrFour;
    }
```

运行：

本代码在完成上机实验9要求的基础上，参考2048游戏加入了一些细节。比如将为0的位置改为空白输出，使用户可以更清楚地看到哪个方框里有数字；在融合部分，增加了如果同行（排）前两个数为成对数，第三个数不参与融合的情况下，前两个数融合后第三个数会占据第二个数空出的空位效果。

前十次运行截图：




初始化输出图形。此处为了增加游戏可玩性，初始化时设置为有两个数值为2或4的随机数同时出现（而非一个）。

```
j
*-*-*-*-*
| | | | |
*-*-*-*-*
| | | | |
*-*-*-*-*
| | | | |
*-*-*-*-*
|4|2| | |
*-*-*-*-*
-----------------------------------
i
*-*-*-*-*
|4|2| | |
*-*-*-*-*
| | | | |
*-*-*-*-*
| |2| | |
*-*-*-*-*
| | | | |
*-*-*-*-*
-----------------------------------
k
*-*-*-*-*
| | | | |
*-*-*-*-*
| | | |2|
*-*-*-*-*
| | | | |
*-*-*-*-*
|4|4| | |
*-*-*-*-*
-----------------------------------
l
*-*-*-*-*
| | | | |
*-*-*-*-*
| | | |2|
*-*-*-*-*
| | | | |
*-*-*-*-*
| | |4|8|
*-*-*-*-*
-----------------------------------
```

前四次输出。

```
------------------------------------
j
*-*-*-*-*
| | | | |
*-*-*-*-*
|2| | | |
*-*-*-*-*
| | | |4|
*-*-*-*-*
|4|8| | |
*-*-*-*-*
------------------------------------
k
*-*-*-*-*
| |2| | |
*-*-*-*-*
| | | | |
*-*-*-*-*
|2| | | |
*-*-*-*-*
|4|8| |4|
*-*-*-*-*
------------------------------------
l
*-*-*-*-*
| | | |2|
*-*-*-*-*
| | |4| |
*-*-*-*-*
| | | |2|
*-*-*-*-*
| |4|8|4|
*-*-*-*-*
------------------------------------
i
*-*-*-*-*
| |4|4|4|
*-*-*-*-*
| | |8|4|
*-*-*-*-*
| | | | |
*-*-*-*-*
| |2| | |
*-*-*-*-*
```

5-8次输出。

```
--------------------------------------
j
*-*-*-*-*
|8|4| | |
*-*-*-*-*
|8|4| | |
*-*-*-*-*
| | | | |
*-*-*-*-*
|2|2| | |
*-*-*-*-*
--------------------------------------
k
*--*--*--*--*
|  |  |  |  |
*--*--*--*--*
| 4|  |  |  |
*--*--*--*--*
|16| 8|  |  |
*--*--*--*--*
| 2| 2|  |  |
*--*--*--*--*
--------------------------------------
```

9-10次输出。

```
*--*--*--*--*
--------------------------------------
qProgram ended with exit code: 0
```

输入q，程序退出。

```
--------------------------------------
k
*--*--*--*--*
| 4| 2| 4| 2|
*--*--*--*--*
| 8| 4| 8| 4|
*--*--*--*--*
|32|16|64|16|
*--*--*--*--*
|64|32| 8| 2|
*--*--*--*--*
--------------------------------------
jGame OverProgram ended with exit code: 0
```

此时无法移动，输出：Game Over