《数据结构》课程实践报告

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 院、系 | 计算机学院 | | 年级专业 | 21计科 | 姓名 | 彭光 | 学号 | 2127405052 |
| 实验布置日期 | | 2022.8.31 | | 提交  日期 | 2022.9.12 | | 成绩 |  |

课程实践实验1：生命游戏的模拟

## 问题描述及要求

问题：

生命游戏在一个无边界的矩形网格上进行，这个矩形网格中的每个单元可被占据，或者不被占据。被占据的单元称为活的，不被占据的单元称为死的。哪一个单元是活的是根据其周围活的邻居单元的数目而一代一代地发生变化的。一代一代转换的具体规则如下：

给定单元的邻居单元指的是与它在垂直、水平或对角方向上相接的8个单元。

如果一个单元是活的，则如果它具有2个或3个活的邻居单元，则此单元在下一代还是活的。

如果一个单元是活的，则如果它具有0个或1个、4个或4个以上的活的邻居单元，则此单元在下一代会因为孤独或拥塞而死亡。

如果一个单元是死的，则如果它具有恰好有3个活的邻居，则此单元在下一代会复活，否则该单元在下一代仍然是死的。

要求：

（1）要求编写程序，模拟任意一个初始输入配置以及代代更替的不同状态并进行显示。

（2）修改以上程序,要求生成一个网格时，用“空格”和“x”分别表示网格中每一个死的单元和活的单元，并且可根据用户选择从键盘或者从文件读入初始配置。

## 概要设计

对实验的理解：

本次实验考察的主要内容其是就是对周边活邻居的数目进行判断自己是死是活来进行不断更新，就是考察一些if语句或者是switch语句的使用情况，而第二个要求是去让用户选择键盘或者文件读入，这个可能会有点繁琐。这就是康德的生命游戏。

系统的功能列表：

1. 对数据进行初始化：

主要分为三种：一是从键盘输入，以（-1，-1）作为终止坐标，二是以文件读入但是是以坐标的形式来进行读入且最后一个坐标必须是-1 -1，三是以文件读入并且是网格形式。

1. 对数据更新：

运用围墙方法进行数据的更新。

1. 对邻居活细胞数目进行计数：

因为本身数组里活细胞记为1，死细胞记为0，所以周围活细胞计数=周围所有细胞计数。

程序运行的界面设计：

首先出现屏幕显示，请用户选择输入配置的方式，1从键盘输入活单元坐标2从文件读入。

总体设计思路：

首先是规定了网格的行数为20，列数为60，其次是创建了一个life类，里面分为5个模块，为初始化，打印，更新和判断和计数。其中计数是采用了围墙技术，避免特殊情况的发生。

更新是利用计数来完成单元的生与死。

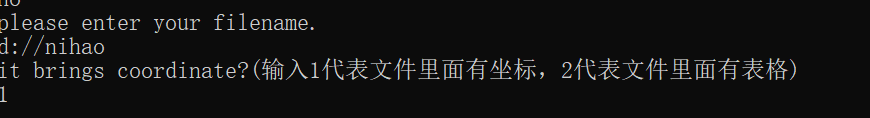
## 三、详细设计

首先要在头文件内设计一个life类，在这里面采用int grid[maxrow+2][maxcol+2]，是采用了围墙技术。其次因为是要求用户选择输入模式，所以又多加了一个bool函数来判断输入方式，在文件中又有两种表示方法，所以我用it brings coordinate?(输入1代表文件里面有坐标，2代表文件里面有表格)来决定是那种形式。在update函数里我用的是根据neighbour——count函数来进行switch语句判断单元的生与死。在print函数里，我用了从1开始递增，因为之前采用了围墙技术。

里面最难办的是initialize函数的接口问题：首先会问你屏幕截图 2022-09-12 211202

如果想从键盘输入就打yes，然后就慢慢输入坐标就好，注意要以-1 -1终止

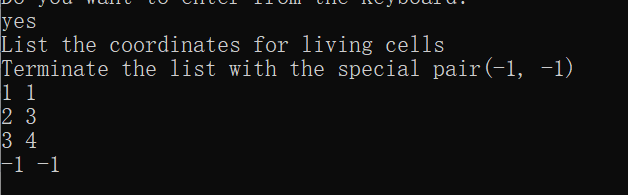
如果想从文件读入就输入no，然后就要求输入文件名，（注意不用加后缀”.txt”），



接下来就会输出结果

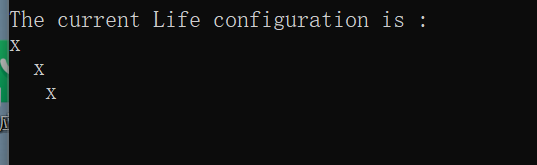
## 四、实验结果

测试输入：

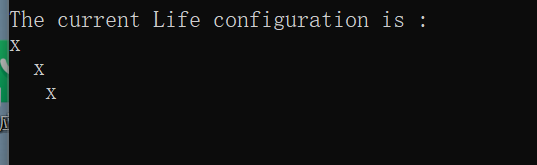


测试目的：看正常从键盘输入是否会出现编译错误

正确输出：

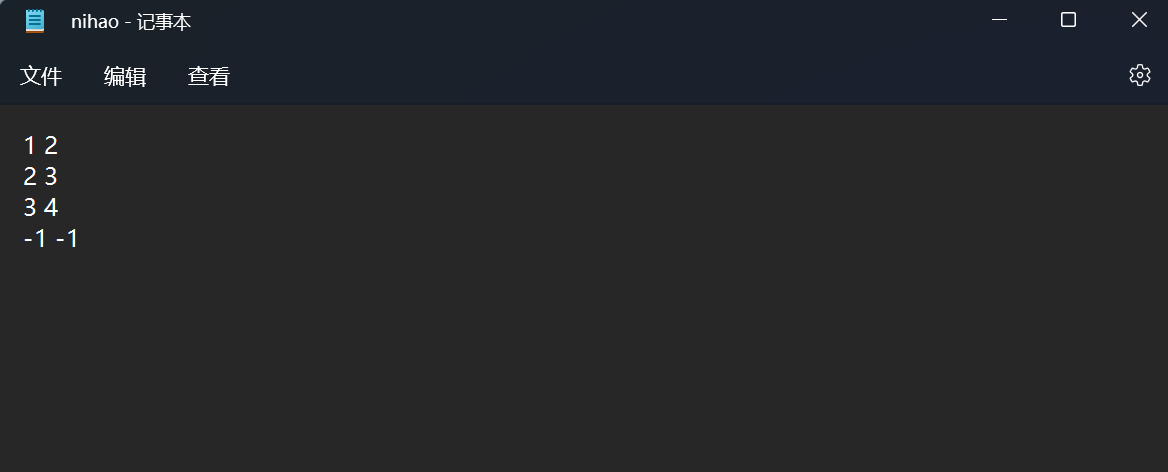


实际输出：



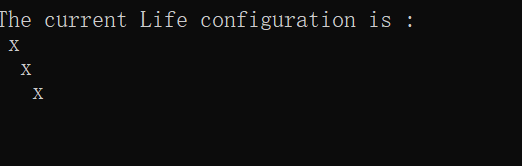
测试结论：通过

测试输入：

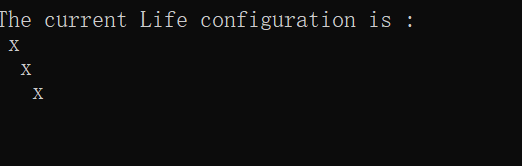


测试目的：测试文件读入是否可以成功

正确输出：



实际输出：



测试结论：通过

## 五、实验分析与探讨

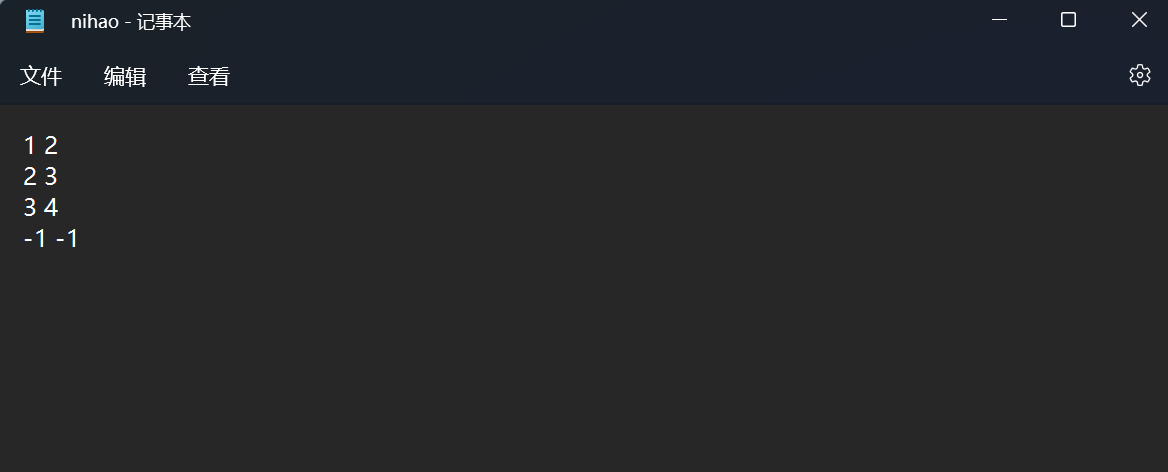
时间复杂度为O(n^2),因为算法最高用到了两次循环而且是一直到底的循环。

空间复杂度为O(1),因为后来都没有涉及到关于n的占用空间。

算法优化：里面有坐标的文件我是利用最后一个坐标（-1，-1）来控制结束循环的，这样就是对用户产生了困扰，所以感觉可以利用程序不利用最后一个坐标是（-1，-1）来解决，而可以把它放在外边来实现，这样就可以让用户使用的更顺心。

## 小结

在本次实验中，我了解了lifegame的具体思路和怎样运行，更加了解了类和头文件的使用，这个lifegame我觉得很有意思，我相信我会继续把它完善下去。

补充说明：输入文件若是坐标，就用如下模式：

若是表格：

## 附录：源代码

1. 实验环境：**Visual Studio 2022**

**2**

1. **life.h**

#pragma once

#include<iostream>

#include<fstream>

#include<string>

#include<iomanip>

using namespace std;

const int maxrow = 20, maxcol = 60; //living cells共有20行，60列

class Life

{ //存放在\*.h文件中的类体声明

public:

void initialize(); //初始化living cells的状态

void print(); //打印输出当前living cells的状态

void update(); //进行deadliving间的转换

bool say\_initialize\_yes();//进行从键盘输入还是直接文件输入判断

private:

int grid[maxrow + 2][maxcol + 2]; //借助两维数组存放living cells

//注意：为处理一致，添加了虚拟的2行2列，放围墙

int neighbor\_count(int row, int col); //统计邻居cell的状态

};

bool Life::say\_initialize\_yes()

{

string answer;

cout << "Do you want to enter from the keyboard?" << endl;

cin >> answer;

if (answer == "yes")

return true;

else return false;

}

void Life::initialize()

{

int row, col;

for (row = 0; row <= maxrow + 1; row++)

for (col = 0; col <= maxcol + 1; col++)

grid[row][col] = 0;//对二维数组进行初始化为0；

if (say\_initialize\_yes())//决定从键盘输入

{

int row, col;

cout << "List the coordinates for living cells" << endl;

cout << "Terminate the list with the special pair(-1, -1)" << endl;

cin >> row >> col;

while (row != -1 || col != -1)

{

if (row >= 1 && row <= maxrow)

if (col >= 1 && col <= maxcol) grid[row][col] = 1;

else cout << "Column" << col << " is out of range" << endl;

else cout << "Row" << row << " is out of range" << endl;

cin >> row >> col;

}//要求输入的序号是大于等于1，小于等于maxrow和maxcol。用循环来控制输入，

}

else

{

string file;

cout << "please enter your filename." << endl;

cin >> file;

cout << "it brings coordinate?(输入1代表文件里面有坐标，2代表文件里面有表格)" << endl;

int n;

cin >> n;

ifstream fin(file + ".txt");

if (fin.good() == true)

{

if (n == 1)//这里的文档里必须要求最后一行是-1 -1.

{

int row, col;

while (fin.eof()==false)

{

fin >> row >> col;

while (row != -1 || col != -1)

{

if (row >= 1 && row <= maxrow)

if (col >= 1 && col <= maxcol) grid[row][col] = 1;

else cout << "Column" << col << " is out of range" << endl;

else cout << "Row" << row << " is out of range" << endl;

fin >> row >> col;

}

}

}

else if (n == 2)//这里是文件里有表格

{

string temp;

int r=-1;

char a[maxrow+2][maxcol+2];

while (fin.eof() == false)

{

getline(fin, temp);

r++;

for (int i = 0; i < temp.size(); i++)

{

a[r+1][i+1] = temp[i];

}

}

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

{

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

{

if (a[i][j] == '\*')

{

grid[i][j] = 1;

}

}

}

}

}

}

}

void Life::print()

{

int row, col;

cout << "\nThe current Life configuration is : " << endl;

for (row = 1; row <= maxrow; row++)

{

for (col = 1; col <= maxcol; col++)

if (grid[row][col] == 1) cout << "x";

else cout << " ";

cout << endl;

}

cout << endl;

}

void Life::update()

{

int row, col, new\_grid[maxrow + 2][maxcol + 2];

for (row = 1; row <= maxrow; row++)

for (col = 1; col <= maxcol; col++)

switch (neighbor\_count(row, col))

{

//调用统计函数,按结果分情况

case 2: new\_grid[row][col] = grid[row][col]; break;//不变

case 3: new\_grid[row][col] = 1; break; //激活

default: new\_grid[row][col] = 0; //dead

}

for (row = 1; row <= maxrow; row++)

for (col = 1; col <= maxcol; col++)

grid[row][col] = new\_grid[row][col];//将临时数组中的数据拷贝回原grid数组

}

int Life::neighbor\_count(int row, int col)

{

int i, j, count = 0;

for (i = row - 1; i <= row + 1; i++)

for (j = col - 1; j <= col + 1; j++)

count += grid[i][j];//如果存活，则累加；否则为0

count -= grid[row][col]; //去除自己

return count;

}

1. **life.cpp**

#include "life.h"

using namespace std;

void instructions()

/\* Pre: None.

Post: Instructions for using the Life program have been printed. \*/

{

cout << "Welcome to Conway’s game of Life." << endl;

cout << "This game uses a grid of size "

<< maxrow << " by " << maxcol << " in which each" << endl;

cout << "cell can either be occupied by an organism or not." << endl;

cout << "The occupied cells change from generation to generation" << endl;

cout << "according to how many neighboring cells are alive." << endl;

}

bool user\_says\_yes()

{

int c;

bool initial\_response = true;

do { // Loop until an appropriate input is received.

if (initial\_response)

cout << " (y,n)? " << flush;

else

cout << "Respond with either y or n: " << flush;

do { // Ignore white space.

c = cin.get();

} while (c == '\n' || c == ' ' || c == '\t');

initial\_response = false;

} while (c != 'y' && c != 'Y' && c != 'n' && c != 'N');

return (c == 'y' || c == 'Y');

}

int main()

{

Life configuration;

instructions();

configuration.initialize();

configuration.print();

cout << "Continue viewing new generations? " << endl;

while (user\_says\_yes())

{

configuration.update();

configuration.print();

cout << "Continue viewing new generations? " << endl;

}

return 0;

}