《数据结构》课程实践报告

院、系	计算机学院	年级专业	21 计算机科学与技术	姓名	赵鹏	学号	2127405037
实验布置日期	2022.9.5		提交日期	2022.9.6		成绩	

课程实践实验 1：生命游戏的模拟

一、问题描述+及要求

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。

（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。

二、概要设计

(1) 生命游戏这个实验的本质是模拟网格中的细胞在游戏的规则下一代代更新的过程。

(2) 本项目主要实现了以下功能：

初始化：

1．随机生成每个方格

2．由用户手动输入活细胞数量和坐标

3．从文件读入初始配置

播放：

1. 通过 Y/N 手动播放
2. 设定时间间隔并自动播放

(3) 首先提示由用户输入网格大小(不超过 50*50)

然后设定是或自动播放，若选择自动播放则提示输入播放间隔

随后选择网格生成方式 0-随机生成 1-手动输入坐标 2-从文件读入

```
Please input the number of rows and the number of columns:(no more than 50*50)
10 10
Whether use auto play(Y/N)
Y
Please set the playback interval (ms)
1000
Please choose the way to generate the grid  0-random 1-input coordinates 2-from file
1
Please input the number of living cells
3
Please input 3 living cell coordinates
2 2
2 3
2 4
```

(4) 代码实现中采用线性表(数组)存储网格。

设计了 Life 类用于实现游戏的各种功能并设计了以下方法

initializeGrid	初始化网格以及设置
updateGrid	更新网格
showGird	展示当前网格
setGrid	设置网格行列并初始化网格及哨兵
calcNeighbour	计算八个相邻位置的活细胞数
mainLoop	游戏进程主循环

设计了 DoubleBuffer 类用于解决自动播放时闪屏的问题并具有以下方法：

DoubleBuffer	类的构造函数，初始化
show	展示后台缓冲区
close	关闭缓冲区
clear	清空缓冲区
bufferSwitch	选择缓冲区

本项目中类与类之间的关系较为简单，共设计了 Life 和 DoubleBuffer 两个类，Life 类用于进行游戏，而 DoubleBuffer 类用于解决 Life 类在选择自动播放模式时闪屏的问题。

(5)本项目在设计实现过程中设计了以下程序文件：

utility.h	包含各个 cpp 文件所需的头文件
Life.h	声明 Life 类
DoubleBuffer.h	声明 DoubleBuffer 类
Life.cpp	实现 Life 类中的各种函数
DoubleBuffer.cpp	实现 DoubleBuffer 类中的各种函数
main.cpp	实现程序主进程

三、详细设计

本项目主函数较为简洁，可分为三步:

1.生成 Life 类实例 configuration。

2.调用 configuration 的 initializeGrid 函数进行初始化。

3.调用 mainLoop 函数进行游戏主进程

关键算法设计:

Life 类

1. **initializeGrid 初始化函数**

   首先将网格设置为-1 作为哨兵，并由用户输入行数列数，随后在用户输入的网格范围内进行初始化。

   随后由用户进行模式选择，依次选择是否自动播放若选择自动播放还需要设置播放间隔。

   再选择网格生成方式，支持随机生成、手动输入坐标、从文件读入网格三种方式。若选择随机生成将以 50%的概率随机生成细胞状态，选择手动输入坐标则需要输入活细胞数量和一定数量的细胞。如选择从文件读入则需要输入文件路径，空格表示死细胞，其他表示活细胞。

2. **updateGrid 更新网格函数**

   首先定义一个临时数组用于存储每个细胞周围的活细胞数，循环行和列并调用 calcNeighbour 计算周围活细胞数量，并将结果存储在临时数组中。最后根据原来细胞的状态和临时数组对应位置的值确定新一代的网格中细胞的状态。

DoubleBuffer 类

1. **DoubleBuffer 构造函数**

   申请两个缓冲区并分配缓冲区，随后关闭光标获得更好的显示效果。分配了缓冲区后将在输出时把结果输出到后台缓冲区，在显示新的一代时把缓冲区设置为可见，即可解决闪屏问题。

2. **show 显示函数**

   将后台缓冲区设置为可见，将另一个缓冲区设置为不可见并清空。

3. **clear 清空函数**

   把光标恢复到初始位置并清空缓冲区

四、实验结果

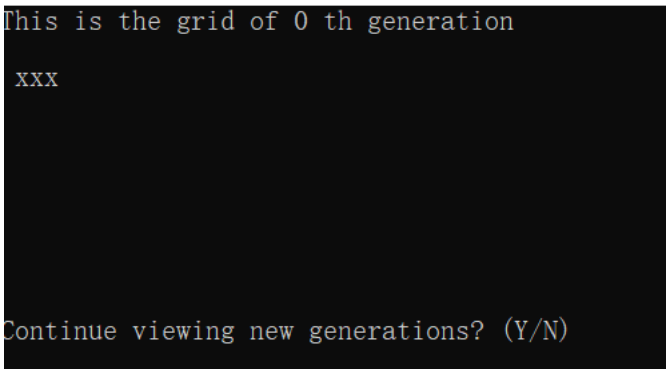
测试输入：

```
10 10
N
1
3
2 2 2 3 2 4
```

测试目的：测试手动输入目标的正确性

正确输出： xxx

实际输出：



测试结论：通过

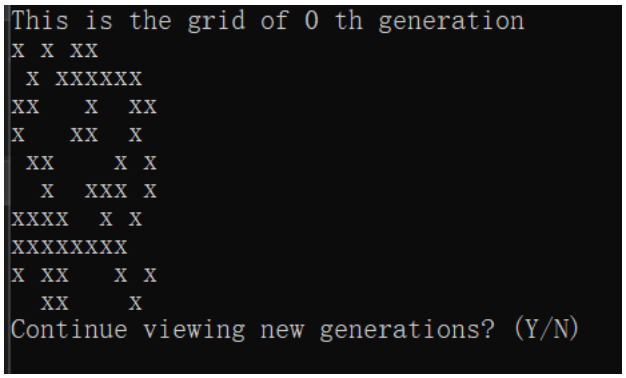
测试输入：

10 10
N
0

测试目的： 测试随机生成网格

正确输出： 一个随机的网格

实际输出：



测试结论：通过

测试输入：

10 10
N
1
2
2 2 2 -5

测试目的：测试输入非法坐标时的处理

正确输出：invalid coordinates,input again

实际输出：

```
lease input the number of rows and the number of columns:(no more than 50*50)
0 10
hether use auto play(Y/N)

lease choose the way to generate the grid  0-random 1-input coordinates 2-from file

lease input the number of living cells

lease input 2 living cell coordinates
 2
 -5
nvalid coordinates,input again
```

测试结论：通过

测试输入：

10 10

N

2

test.txt

测试目的：测试从文件读入数据的正确性

正确输出：

```
  xxx


xxx


  xxx
```

实际输出：

```
This is the grid of 0 th generation
   xxx

xxx

  xxx
```

测试结论：通过

测试输入：100 100

N

1

1

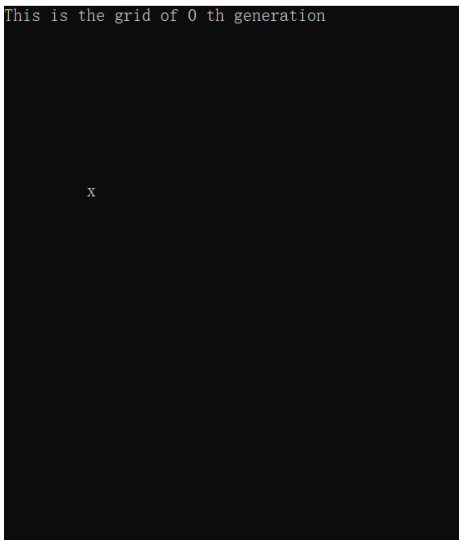
10 10

测试目的：测试尺寸超过限制的情况

正确输出:50*50

x

实际输出：



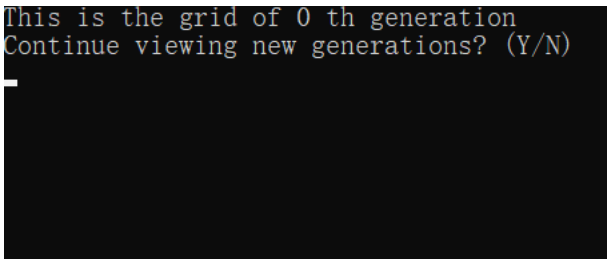
测试结论：通过

测试输入：-1 -2
N
1
1
10 10

测试目的：测试尺寸出现负数限制的情况

正确输出:0*0 的空表格

实际输出：



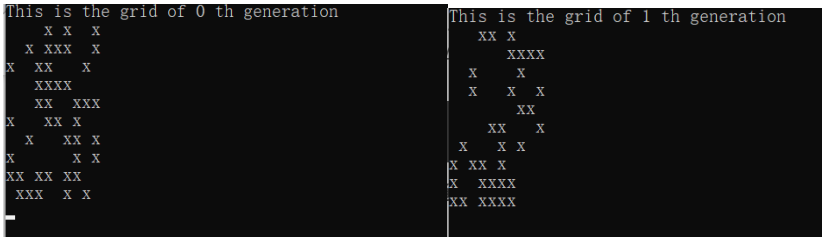
测试结论：通过

测试输入：
10 10
Y
10000
0

测试目的：测试自动播放功能

正确输出:每隔一秒自动播放下一代

实际输出：



测试结论：通过

五、实验分析与探讨

测试结果分析：

测试部分主要测试了 3 种初始化的方式，以及输入数据不合法时的处理情况，和两种播放模式(手动和自动)。程序通过了所有测试，较好的完成了生命游戏的实现。

算法的主要瓶颈在网格的更新部分，时间复杂度和空间复杂度均为 O(Rows*Cols)。在实现网格更新的过程中也可以使用位运算的方法节省掉临时数组的空间。具体方法是把新的数组状态(0/1)保存在原数组的高位上，比如该网格原状态死亡，更新后存活，则可以通过位运算把原网格的值更新为 2(二进制下 10)高位的 1 代表更新后存活，低位的 0 代表原来死亡。在更新完所有网格后将原网格的所有数右移(>>)一位即可。

在本次实验的过程中遇到了较多的问题，如：

(1)

问题：对于文件输入时，若使用 getline 函数读取一行而该行不满列数时，通过文件初始化网格会出现异常错误。

解决办法：在按行读取文件并给网格赋值时判断存储该行的字符串长度，不足 Cols 的用空白补齐即可。

(2)

问题：对于自动播放模式由于不断地清屏和重新输入会出现闪屏的问题，非常影响观感。

解决办法：学习了控制台里双缓冲技术的使用。大致思路是申请两个缓冲区并分配缓冲区，随后关闭光标获得更好的显示效果。分配了缓冲区后将在输出时把结果输出到后台缓冲区，在显示新的一代时把缓冲区设置为可见，即可解决闪屏问题。

(3)

问题：起初将游戏双缓冲类和游戏主流程定义在 main 函数里，导致整个 main 函数较为看起来较为冗长。

解决办法：把游戏的主循环也进行封装，把双缓冲类定义在游戏主循环内。在 main 函数里直接调用主循环函数即可。

六、小结

通过本次的实验，我的 C++代码编写能力有所提升，同时，我仍然需要进一步学习面向对象的思想以及编程方法。

在本次实验种，我实现了手动和自动播放两种模式以及随机初始化、手动输入坐标初始

化和从文件读入三种初始化方式。但程序仍然有所不足，比如会看前几代、将结果保存到文件等功能尚未实现，除此以外，网格的大小也有所限制，测试发现，当网格尺寸过大时会出现异常，故把最大尺寸限制为 50*50.

补充说明：在实验调试时发现，对于文件读入模式，若使用 Visual Studio 调试，直接输入文件名文件相对路径默认为项目的工作文件夹，而直接打开编译生成的 exe 运行，则相对路径为 exe 所在文件夹。为了预防路径问题，文件读入时推荐输入绝对路径。

附录: 源代码

1、实验环境：Windows 11,Visual Studio 2022…

2、

（1） 文件名 utility.h

代码:

```
#pragma once

#include<iostream>
#include<cstring>
#include<ctime>
#include<windows.h>
#include<fstream>
#include<string>
#include "DoubleBuffer.h"

using namespace std;
```

(2) 文件名 Life.h

代码:

```
#pragma once
#include "utility.h"

#define maxCol 60
#define maxRow 60

class Life
{
public:

    void initializeGrid(int &autoPlay,int &Speed);//初始化
    void updateGrid();//更新
    void showGird();//展示
    void mainLoop();
    int autoPlay = 0, interval = 1000;//是否自动播放、播放速度
```

```
private:
	int Generations = 0;//当前代数
	int Rows, Cols;//行列
	int Grid[maxRow + 2][maxCol + 2];
	int calcNeighbour(int row, int col);
	void setGrid(int row, int col);
};

bool UserSayYes(DoubleBuffer &db);
```

(3) 文件名:DoubleBuffer.h
代码

```
#pragma once
#include <iostream>
#include <windows.h>
using namespace std;

class DoubleBuffer {
private:
	HANDLE l_buffer1;
	HANDLE l_buffer2;
	HANDLE l_stdBuffer;
	TCHAR* _data = NULL;
	int nowActive;

	void bufferSwitch();

public:
	DoubleBuffer();
	void show(); //展示
	void close(); //关闭
	void clear(); //清空缓冲区

};:
```

(4) 文件名：DoubleBuffer.cpp
代码:

```
#include "DoubleBuffer.h"

//private
void DoubleBuffer::bufferSwitch(){
	nowActive = nowActive == 1 ? 2 : 1;
```

```
}

//public
DoubleBuffer::DoubleBuffer(){
	CONSOLE_CURSOR_INFO info;
	info.bVisible = false;
	info.dwSize = 1;
	CONSOLE_SCREEN_BUFFER_INFO consoleInfo;
	l_stdBuffer = GetStdHandle(STD_OUTPUT_HANDLE);
	l_buffer1 = CreateConsoleScreenBuffer(GENERIC_WRITE | GENERIC_READ,
FILE_SHARE_READ | FILE_SHARE_WRITE, NULL,
		CONSOLE_TEXTMODE_BUFFER, NULL);
	l_buffer2 = CreateConsoleScreenBuffer(GENERIC_WRITE | GENERIC_READ,
FILE_SHARE_READ | FILE_SHARE_WRITE, NULL,
		CONSOLE_TEXTMODE_BUFFER, NULL);
	SetConsoleActiveScreenBuffer(l_buffer1);
	_data = new TCHAR[5000 * 5000]; // 分配_data 缓冲区
	nowActive = 1;

	// 关闭光标
	SetConsoleCursorInfo(l_stdBuffer, &info);
	SetConsoleCursorInfo(l_buffer1, &info);
	SetConsoleCursorInfo(l_buffer2, &info);
}

void DoubleBuffer::show(){
	static COORD coord = {0, 0};
	static DWORD b;
	//cout << nowActive << endl;
	ReadConsoleOutputCharacter(l_stdBuffer, _data, 8000, coord, &b);
	if(nowActive == 1){
		WriteConsoleOutputCharacter(l_buffer2, _data, wcslen(_data), coord, &b);
		SetConsoleActiveScreenBuffer(l_buffer2);
	}else{
		WriteConsoleOutputCharacter(l_buffer1, _data, wcslen(_data), coord, &b);
		SetConsoleActiveScreenBuffer(l_buffer1);
	}
	clear();
	bufferSwitch();
}

void DoubleBuffer::clear()
{
	static COORD coord = { 0, 0 };
```

```
	static DWORD b;
	FillConsoleOutputCharacter(l_stdBuffer, ' ', 8000, coord, &b);
	SetConsoleCursorPosition(l_stdBuffer, coord);
}


void DoubleBuffer::close()
{
	CloseHandle(l_buffer1);
	CloseHandle(l_buffer2);
	delete[] _data;
}
```

(5) 文件名：Life.cpp
代码：

```
#include "Life.h"

bool UserSayYes(DoubleBuffer& db)//是否看下一代
{
	string Command;
	cin >> Command;
	if (Command == "Y" || Command == "y" || Command == "YES" || Command == "yes" ||
Command == "1")
		return true;
	else
		return false;
}

void Life::mainLoop()
{
	system("cls");
	(*this).showGird();
	Sleep((*this).interval);
	system("cls");
	DoubleBuffer db;


	if ((*this).autoPlay)//自动播放
	{
		while (true)
		{
			(*this).updateGrid();
			(*this).showGird();
			db.show();
```

```
                Sleep((*this).interval);

            }
        }
        else//手动播放
        {
            db.close();
            (*this).showGird();
            cout << "Continue viewing new generations? (Y/N)" << endl;

            while (UserSayYes(db))
            {

                system("cls");
                (*this).updateGrid();
                (*this).showGird();
                cout << "Continue viewing new generations? (Y/N)" << endl;

            }
        }
}
void Life::setGrid(int row, int col)//设定表格大小并初始化表格
{


    Rows = max(0,row);
    Cols = max(0,col);

    for (int i = 1; i <= Rows; i++)
        for (int j = 1; j <= Cols; j++)
            Grid[i][j] = 0;
}

void Life::initializeGrid(int &autoPlay,int &Speed)//初始化表格，选择播放设置
{
    memset(Grid, -1, sizeof Grid);

    int rows, cols, liveNums, speed;
    string choice;
    cout << "Please input the number of rows and the number of columns:(no more than
50*50)" << endl;
    cin >> rows >> cols;
    setGrid(min(rows,50), min(cols,50));
```

```
    cout << "Whether use auto play(Y/N)" << endl;
    cin >> choice;
    if (choice == "Y" || choice == "y")
    {
        autoPlay = 1;
        cout << "Please set the playback interval (ms)"<<endl;
        cin >> speed;
        Speed = speed;
    }
    else
        autoPlay = 0;

    int choice0 = 0;
    cout << "Please choose the way to generate the grid   0-random 1-input coordinates
2-from file" << endl;
    cin >> choice0;
    if (choice0 == 0)
    {
        srand((unsigned)time(NULL));
        for (int i = 1; i <= rows; i++)
            for (int j = 1; j <= cols; j++)
                Grid[i][j] = rand() % 2;
        return;
    }
    else if (choice0==1)
    {

        cout << "Please input the number of living cells" << endl;
        cin >> liveNums;

        cout << "Please input " << liveNums << " living cell coordinates" << endl;

        int nums = 0;
        while(nums<liveNums)
        {
            int x, y;
            cin >> x >> y;
            if (x >= 1 && x <= Rows && y >= 1 && y <= Cols)
                Grid[x][y] = 1, nums++;
            else
            {
                cout << "invalid coordinates,input again" << endl;
            }
```

```
            }

        }
        else
        {
            cout << "Please input the path of the file"<<endl;
            string path, line;
            cin >> path;
            ifstream iFile;
            iFile.open(path);
            while (!iFile.is_open())
            {
                cout << "Path error ,input the path again" << endl;
                cin >> path;
                iFile.open(path);
            }
            for (int i = 1; i <= Rows; i++)
            {
                getline(iFile, line);
                while (line.size() < Cols)line += " ";

                for (int j = 0; j < Cols; j++)
                {
                    if (line[j] != ' ')Grid[i][j + 1] = 1;
                    else Grid[i][j+1] = 0;
                }
            }

        }
}


int Life::calcNeighbour(int row, int col)//计算周围活细胞数量
{
        int Num = 0;
        for (int i = row - 1; i <= row + 1; i++)
            for (int j = col - 1; j <= col + 1; j++)
                Num += (Grid[i][j] == 1);
        Num -= Grid[row][col];
        return Num;
}

void Life::updateGrid()//更新表格
{
```

```
	Generations++;

	int newGrid[maxRow + 2][maxCol + 2];

	memset(newGrid, 0, sizeof newGrid);

	for (int i = 1; i <= Rows; i++)
	{
		for (int j = 1; j <= Cols; j++)
		{
			int Num = calcNeighbour(i, j);

			if ((Num == 2 || Num == 3) && Grid[i][j])
				newGrid[i][j] = 1;
			if (Grid[i][j] && !(Num == 2 || Num == 3))
				newGrid[i][j] = 0;
			if (!Grid[i][j] && Num == 3)
				newGrid[i][j] = 1;

		}
	}
	for (int i = 1; i <= Rows; i++)
		for (int j = 1; j <= Cols; j++)
			Grid[i][j] = newGrid[i][j];

}



void Life::showGird()//展示
{
	cout << "This is the grid of " << Generations << " th generation" << endl;
	for (int i = 1; i <= Rows; i++)
	{
		for (int j = 1; j <= Cols; j++)
			cout << (Grid[i][j] ? "x" : " ");
		cout << endl;
	}



}
```

（6） 文件名：main.cpp
代码：

```
#include "Life.h"
#include "utility.h"

using namespace std;


int main()
{

    Life configuration; //生存游戏类
    configuration.initializeGrid(configuration.autoPlay, configuration.interval);//初始化
    configuration.mainLoop();

    return 0;
}
```