Nombre Alejandro Pérez Díaz Fecha

Profesor Materia

Institución Curso Nota

```
#include <iostream>
#include <fstream>
using namespace std;
//defini constantes de maximas para luego leerla del txt.
Const int Maxfil=100;
Const int Maxcol=100;
//Funcion para leer el archivo.
void Carga(string archivo, int Filas, int Columnas, int Cuadricula[Maxfil][Maxcol]){
        ifstream archivo("archivo.txt");
          if(archivo.is_open(){
            archivo>>Filas>>Columnas;
                if(Filas>Maxfil || Columnas>Maxcol){
                    cout<<"El num se excede del tamanio maximo";
                    }
                    for(int i=0; i<Filas; i++){
                        for(int j=0; j<Columnas; j++){
                        }
                    }
                    archivo.close();
                }
            }
//Funcion para mostrar la cuadricula.
void MostrarCuadricula(int Cuadricula[Maxfil][Maxcol], int Filas, int Columnas){
            for(int i=0; i<Filas; i++){
                for(int j=0; j<Columnas; j++){
                    cout<<(Cuadricula[i][j]?'O':'.')<<" ";
                }
            }
        }
//Funcion para contar la cantidad de celulas vivas.
void ContarCelulasVivas(int Cuadricula[Maxfil][Maxcol], int Filas, int Columnas){
        int contadorviv=0;
            for(int i=0; i<Filas; i++){
                for(int j; j<Columnas; j++){
                    contadorviv=contadorviv+Cuadricula[i][j];
                }
            }
            return contadorviv;
        }
//Funcion para la cantidad de rondas.
void rondas(int Cuadricula[Maxfil][Maxcol], int Columnas, int Filas){
    int nuevaCuadricula[Maxfil][Maxcol]={0};
            for(int i=0; i<Filas; i++){
                for(int j=0; j<Columnas; j++){
                    int vivas=0;
/*Esto es utilizado para que permita inspeccionar todas las celdas
alrededor de la posicion [i][j] */
                    for(int x=-1; x<=1; x++){
                        for(int y=-1; y<=1; y++){
//Esto asegura de no contar la celda actual.
                            if(x!=0 || y!=0){
                                int ni=i+x; /*representa las cordenadas de
                                int nj=j+y; una celda vecina.*/
/*verifica que los indices esta dentro de los limites de la
cuadricula */
```

```
            if(ni>=0 && ni<filas && nj>=0 && nj<columnas){
                vivas = vivas + cuadricula[ni][nj];
            }}
        }
    }
/* Aqui indica si la celda esta viva o esta muerta tambien las reglas
en el cual dice que vive si tiene 2 o 3 y muere si tiene exactamente 3
y si fallas lo anterior o sige en la otra generacion o muere */
nuevaCuadricula[i][j] = (cuadricula[i][j]==1 && (vivas==2 || vivas==3)) || (cuadricula[i][j]
==0 && vivas==3) ? 1 : 0;
// Actualiza la cuadricula con la nueva generacion.
    for(int i=0; i<filas; i++){
        for(int j=0; j<columnas; j++){
            cuadricula[i][j] = nuevaCuadricula[i][j];
        }
    }
}
int main(){
    const char* archivo = "archivo.txt";
    int filas, columnas, generaciones;
// solicita numero de generaciones a simular.
    cout<<"numero de generacion a simular";
    cin>>generaciones;
    Cuadricula[MaxFil][MaxCol]={0};
// carga la configuracion inicial del archivo.
    CargarCarchivo, filas, columnas, cuadricula);
/* Simula el juego por el numero de generaciones especificado, mostrando y
contando las celulas a cada Generacion */
    for(int i=0<generaciones; i++){
        cout<<"Generacion"<<i+1;
        mostrarCuadri(cuadricula, filas, columnas);
        cout<<"Celulas vivas"<<ContarCelulasVivas(cuadricula, fila, Columna);
        rondas(cuadricula, columnas, filas);
    }
    return 0;
}
```