Gliwice, 23.01.2019

Semester: 3
Group: 3

Computer Programming Laboratory Project

Conway's Game of Life

Author: Dawid Gruszka
e-mail: dawigru908@student.polsl.pl

1. Task topic

Game of Life is a cellular automaton, no-player game, its evolution is determined by its initial state. Cells are spread in two dimensional space, each cell can be dead or alive. Depending on amount of alive neighbours (neighbours are cells directly in contact with each other) given cell can change its state in the next iteration.

Main rules:

Any live cell with fewer than two live neighbours dies, as if by underpopulation.
Any live cell with two or three live neighbours lives on to the next generation.
Any live cell with more than three live neighbours dies, as if by overpopulation.
Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

2. Project analysis

class Cell - represents single cell, its current state and number of alive neighbours,

class Board - represents game board which is a two dimensional array of objects of class Cell. Contains following methods:

void printBoard() which prints current game board,
void cleanBoard() which clears current game board (sets all cells to "DEAD" state),
void loadUserData() which takes the initial conditions from user, and creates a game board using them,
void calculateBoard() which calculates number of alive neighbours for each cell and creates game board for the next iteration of the game,

class Choices - used to manage user input choices containing two methods returning user choices (char mainChoice() and char saveChoice()),

class Files - used to manage files, save to them and read from them using methods:

void saveBoard<type>("type") which saves the current game board to a binary file,
Board readBoard() which reads the game board from binary file created by "saveBoard" method,
Board demoBoard() which reads the game board from binary file created by me it contains an exemplary game board with an oscillator,

class Menu - which inherits from Choices and Files and is used to draw the whole user interface.

3. Project requirements

Topics used (4):
- exceptions implemented in class Choices,
- iostream implemented in class Files,
- templates implemented in class Files,
- multiple inheritance- class Menu inherits from classes Files and Choices.

Classes (5):
- Cell,
- Board,
- Choices,
- Files,
- Menu.

4. Manual

Main menu of the program gives user the choice between starting from scratch (n), playing demo board from demo.bin file (d) and reading previously created board from save.bin file (r).

If user chooses to start from scratch program takes user input by method “loadUserData()” in class Board. User chooses how many alive cells he/she wants in the beginning, and then inputs their x and y coordinates.

After that the program runs, to stop the iteration user should input “1” otherwise if user wants to continue she/he should input anything else.

When board iteration stops after inputting “1”, user has a choice between saving current state of the board to save.bin file (y for yes, and n for no).

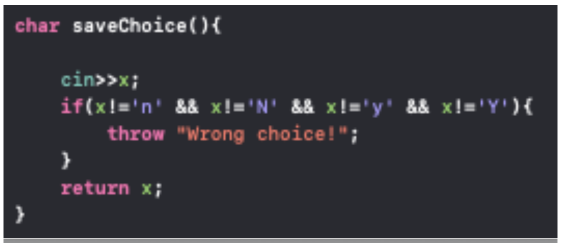
5. Problems with implementation

- Program just writes new lines in the console in every iteration, it doesn’t clear/overwrite it because I couldn’t find suitable library with needed functions for OS I wrote my program on (macos). To make the user interface better I’d have to implement GUI, but I wanted to avoid it.

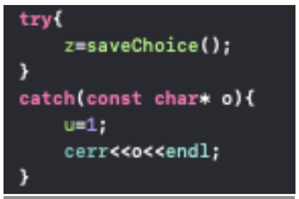
- I had some problems with calculating and drawing the cells on the edges of the board, no matter what I tried I got improper results there, so I just don’t draw them and in every iteration they are set to “dead” state, because it turned out to be the easiest solution I found.

6. Examples of topics implementation

- exceptions:

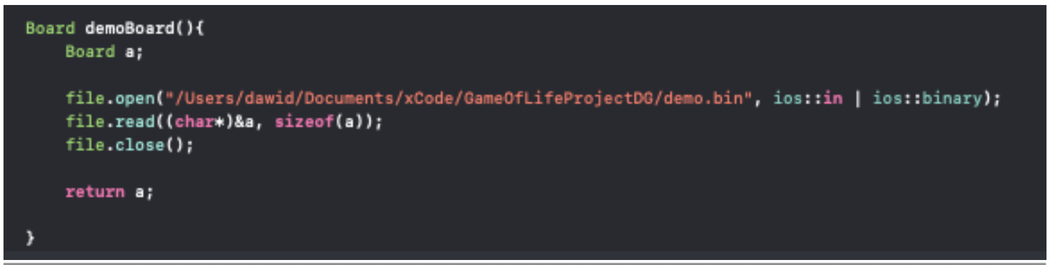


```
char saveChoice(){

    cin>>x;
    if(x!='n' && x!='N' && x!='y' && x!='Y'){
        throw "Wrong choice!";
    }
    return x;
}
```



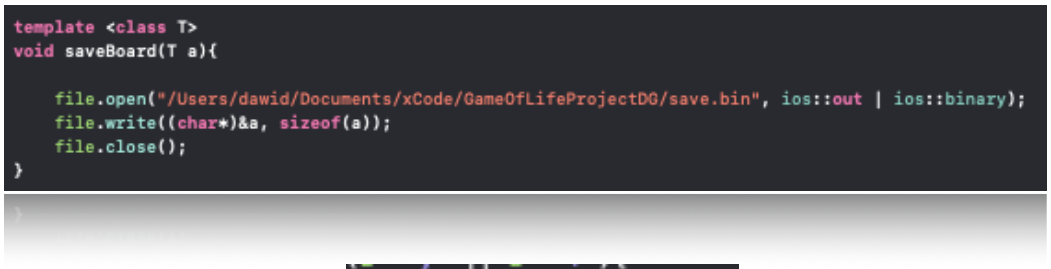
```
try{
    z=saveChoice();
}
catch(const char* o){
    u=1;
    cerr<<o<<endl;
}
```

- iostream:

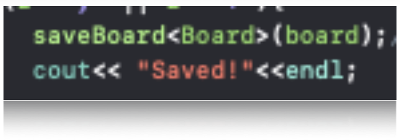


```
Board demoBoard(){
    Board a;

    file.open("/Users/dawid/Documents/xCode/GameOfLifeProjectDG/demo.bin", ios::in | ios::binary);
    file.read((char*)&a, sizeof(a));
    file.close();

    return a;

}
```

- templates:

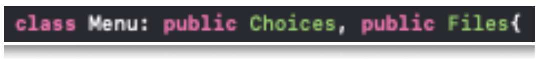


```
template <class T>
void saveBoard(T a){

    file.open("/Users/dawid/Documents/xCode/GameOfLifeProjectDG/save.bin", ios::out | ios::binary);
    file.write((char*)&a, sizeof(a));
    file.close();
}
```



```
saveBoard<Board>(board);
cout<< "Saved!"<<endl;
```

- multiple inheritance:



```
class Menu: public Choices, public Files{
```