Työn valitseminen

Valitessani työtä kuulin vanhaparroilta että tulevilla kursseilla tehtäisiin laivanupotuspeli, ja laiskana ihmisenä ajattelin välttyä ylimääräiseltä työltä. Suunnittelin siis tekeväni sen jo nyt valmiiksi. Laiskuus tosin pääsi yllättämään, tästä johtuen päädyin helpottamaan hieman tekemisiäni. Päädyin lopulta tekemään ristinollapelin. Jotta ei ihan liian helpolla pääsisi, ajattelin harjoitella “grafiikkaa” terminaalissa. Lähistölläni oli tosin vain linuxia ajava läppäri, ja laiskuudesta johtuen päädyin siis opettelemaan ncurses-kirjastoa conion sijaan.

Tästä laiskuudesta johtuen siis taisin aiheuttaa mahdollisia lisätöitä Lehtoreille, käsittääkseni ncurses:ia ei windowsilla voi ajaa muuta kuin windows subsystem for linuxin avulla. En kyllä myönnä tietäväni että onko näihin valmiiksi myöskään kyseistä pakettia asennettu, mutta etteköhän te siitä selviä.

Kääntäjänä käytin tuoreinta GCC:tä.

Työn sisältö

Hieman jäin miettimään josko tässä olisi voinut selvitä hieman selkeämmin vähemmällä modularisoinnilla. Tosin tämä mahdollisti main-funktion kutistamisen vain 8 riviin ja mielestäni suhteellisen selkeäksi. En sitten tiedä kuinka moni muu asia tässä on sitten selkeää.

Päätin jakaa koodini kahteen “pääfunktioon” eli board() ja game():n. Näistä game() keskittyy varsinaiseen pelin logiikkaan, ja board() lähinnä vain tulostaa pelilaudan.

Näistä toki kutsutaan erinäisiä muita funktioita, kuten initNCurses() johon pitkälti laitoin nCursesin toiminnan vaatimat aloitusfunktiot. Tärkeimmät muista funktioista ovat winCheck ja updateBoard. Uskoisin näiden nimien kertovan kaiken tarvittavan.

Valitettavasti tämä koodin formatointi tässä pdf:ssä on hieman puutteellista, mutta onneksi voitte lueskella alkuperäisestä tiedostosta hieman selkeämmin koodia.

HEADER.H

```
#pragma once

#define TRUE 1
#define FALSE 0

#define P1 1
#define P2 2

#define OKAY 5 //rewrite this shit

void printX(int y, int x);
void printO(int y, int x);

void initNCurses();
void board();
void boardInit(int *boardData);
int game();
int winCheck(int *boardData);
int updateBoard(int *boardData, int x, int y, int xOrO);
```

MAIN.CPP

```
#include "header.h"
#include <ncurses.h>
using namespace std;


int main(){
      initNCurses();
      do{
            board();

      }while (game());//Either q or ctrl+C kills program
```

```
endwin();//deallocates memory, ends ncurses
}
```

FUNC.CPP

```
#include "header.h"
#include <ncurses.h>

using namespace std;

void initNCurses() {

	initscr();//allocates memory, initialize screen, clear screen
	cbreak();//Make sure ^C kills program
	noecho();//No more keys echoed to screen when pressed
	keypad(stdscr, TRUE);//read arrow keys etc
}//initNCurses


void board() {
	for(int i = 0; i <= 6 ; i++) {
		mvaddch(i, 0, '|');
		mvaddch(i, 4, '|');
		mvaddch(i, 8, '|');
		mvaddch(i, 4, '|');
		mvaddch(i, 8, '|');
		mvaddch(i, 12,'|');

		if(i%2 == 0){
			for(int j = 0; j <= 12; j++) {
				mvaddch(i, j, '-');
			}//for loop j
		}//for loop i
		move(1,2);
	}//for loop vertical lines
	refresh();
}//board


void boardInit(int *boardData){
```

```
	for(int i = 0; i <= 8; i++){
		boardData[i] = i+2;
	}//for loop i
	}//boardInit




int winCheck(int *boardData) {

//Row check
if(boardData[0] == boardData[1] && boardData[1] == boardData[2]){
	return TRUE;
}
else if(boardData[3] == boardData[4] && boardData[4] == boardData[5]){
	return TRUE;
}
else if(boardData[6] == boardData[7] && boardData[7] == boardData[8]){
	return TRUE;
}
//column check
else if(boardData[0] == boardData[3] && boardData[3] == boardData[6]){
	return TRUE;
}
else if(boardData[1] == boardData[4] && boardData[4] == boardData[7]){
	return TRUE;
}
else if(boardData[2] == boardData[5] && boardData[5] == boardData[8]){
	return TRUE;
}
//Diagonal check
else if(boardData[0] == boardData[4] && boardData[4] == boardData[8]){
	return TRUE;
}
else if(boardData[2] == boardData[4] && boardData[4] == boardData[6]){
	return TRUE;
}
else{
	return FALSE;
}
}//int winCheck

int updateBoard(int *boardData, int x, int y, int XorO) {
```

```
//Check first row
if(y == 1){
	if(x == 2){
		if(boardData[0] > 1){
			boardData[0] = XorO;
			return OKAY;
		}//if(boardData[0] > 1)
	}//if(x == 2)
	else if(x == 6){
		if(boardData[1] > 1){
			boardData[1] = XorO;
			return OKAY;
		}//if(boardData[1] > 1)
	}//else if(x == 6}
	else if(x == 10){
		if(boardData[2] > 1){
			boardData[2] = XorO;
			return OKAY;
		}//if(boardData[2] > 1)
	}//else if (x == 10)
}//if(y == 1)

//Check second row
else if(y == 3){
	if(x == 2){
		if(boardData[3] > 1){
			boardData[3] = XorO;
			return OKAY;
		}//if(boardData[3] > 1)
	}//if(x == 2)
else if(x == 6){
	if(boardData[4] > 1){
		boardData[4] = XorO;
		return OKAY;
	}//if(boardData[4] > 1)
}//else if(x == 6)
else if(x == 10){
	if(boardData[5] > 1){
		boardData[5] = XorO;
			return OKAY;
			}//if(boardData[5] > 1)
		}//else if(x == 10)
	}//else if(y == 3)
```

```
//Check third row
else if(y == 5){
	if(x == 2){
		if(boardData[6] > 1){
			boardData[6] = XorO;
			return OKAY;
		}//if(boardData[6] > 1)
}//if(x == 2)
else if(x == 6){
	if(boardData[7] > 1){
		boardData[7] = XorO;
		return OKAY;
	}//if(boardData[7] > 1)
}//else if (x == 6)
else if(x == 10){
	if(boardData[8] > 1){
		boardData[8] = XorO;
		return OKAY;
	}//if(boardData[8] > 1)
}//else if(x == 10)
}//else if(y == 5)
return FALSE;
}//int updateBoard
```

GAME.CPP

```
#include "header.h"
#include <ncurses.h>

using namespace std;

int game() {

	int boardData[9];
	int playerNum = 1;
	int inputChar;
	int x = 2, y = 1;
	int didWin = 0;
```

```
	int canMove = 0;
	int moveNum = 0;

	boardInit(boardData);

	mvprintw(8, 0, "Place X with space bar");
	move(y, x);
	refresh();

	while (inputChar != 'q') {

		if(moveNum == 9) {
		mvprintw(10, 0, "Tie. Get good.");
		inputChar = getch();
		erase();
		return 3;
	}//moveNum
	inputChar = getch();

	if(inputChar != ' '){

		switch (inputChar){

		case KEY_UP:{
			if(y == 3 || y == 5){
			move(y-=2, x);
			}//if
			break;
		}//KEY_UP

		case KEY_DOWN:{
			if(y == 1 || y == 3){
				move(y+=2, x);
			}//if
			break;
		}//KEY_DOWN
		case KEY_LEFT:{
			if(x == 10 || x == 6){
				move(y, x-=4);
			}//if
			break;
		}//KEY_LEFT
		case KEY_RIGHT:{
			if(x == 2 || x == 6){
```

```
                        move(y, x+=4);
                  }//if
                  break;
            }//KEY_RIGHT
      }
}//Whilr inputChar != 'q'

      else if(playerNum == P1 && inputChar == ' '){

      getyx(stdscr, y, x);
      canMove = updateBoard(boardData, x, y,1);

      if(canMove == OKAY) {

      mvaddch(y, x, 'X');

      // Returns 1 if last move won
      didWin = winCheck(boardData);

      if(didWin){

            mvprintw(10,0, "X gets the epic victory royale");
            mvprintw(11,0,"To claim prize, send credit card number, month/year
and the 3 special numbers on the back to mikko.wrightson@tuni.fi");
            inputChar = getch();
            erase();
            return TRUE;
      }//if didWin
      moveNum += 1;

      playerNum = 2;
            mvprintw(8, 0, "Place O with space bar");
            move(y, x);
      }//if canMove == OKAY
}//else if playerNum P1
      else if(playerNum == P2 && inputChar == ' ') {

      getyx(stdscr, y, x);
      canMove = updateBoard(boardData, x, y,0);

if(canMove == OKAY) {

      mvaddch(y, x, 'O');
```

```
      //Returns 1 if last move won
      didWin = winCheck(boardData);

      if(didWin){

            mvprintw(10,0, "O wins epic victory royale");
            mvprintw(11,0,"To claim prize, send credit card number, month/year
and the 3 special numbers on the back to mikko.wrightson@tuni.fi");
            inputChar = getch();
            erase();
            return TRUE;
      }//if(didWin)
      moveNum += 1;

      playerNum = 1;
      mvprintw(8, 0, "Place X with space bar");
      move(y, x);
      }//if(canMove == OKAY)
      }//else if playerNum == P2
      refresh();
      }//while (inputChar != 'q')
      return FALSE;
      endwin();

}//game()
```

Mikko Wrightson
TAMK

MAKEFILE

```
output: main.o func.o game.o
	g++ -lncurses main.o func.o game.o -o output

main.o: main.cpp header.h
	g++ -c main.cpp

func.o: func.cpp header.h
	g++ -c func.cpp

game.o: game.cpp header.h
	g++ -c game.cpp

clean:
	rm *.o output
```