**Увод**

Идеята на проекта: Да се риализира играта ,,Морски шах”.

Целта: Да се популяризира по-големия вариянт на играта, който е с размер 9 на 9, която се оказва непозната за мнозинството. Тя е доста по-сложна, изисква по-прецизно обмисляне на предстоящите ходове и простата статегия, известна на мнозиството, с която може да се победи, в класическата игра 3 на 3, е доста по-трудно приложима. Също така цел е да се симулира изкуствен интелект, който не прави случайни ходове всеки път. Той се опитва да предвиди кога може да попречи на противвника си да го победи и кога той самият може да направи това. Цел е и просто играчите да се насладят на добрата стара игра без да се налага да хабят листа. А като задача, е поставено историята на играта да бъде запазена във файл, за да може играчите да проследят всеки ход, наравен по време на играта.

Структура на документацията: Първо ще бъде прегладана предметната област, като това включва, пробламите възникнали по време на работа и кратко описание на решението им. След това самото проектиране – архитектурата на проекта. И накрая реялизацията с помощтта на части от кода.

**1.Преглед на предметната област:**

В проекта са използвани полиморфизъм и наследяване. Също така има и работа с файлове.

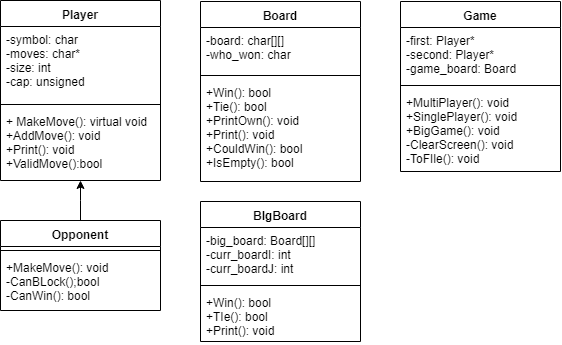
* 1. Полиморфизъм - ,,в [обектно ориентираното програмиране](https://bg.wikipedia.org/wiki/%D0%9E%D0%B1%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%B0%D0%BD%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5) представлява свойството на [обектите](https://bg.wikipedia.org/wiki/%D0%9E%D0%B1%D0%B5%D0%BA%D1%82_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) от един и същи тип да имат един и същи [интерфейс](https://bg.wikipedia.org/w/index.php?title=%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)&action=edit&redlink=1), но с различна реализация на този интерфейс.”
  2. Наследяване - ,,Организира и подпомага полиморфизма и капсулирането, като позволява да бъдат дефинирани и създавани обекти, които са специализирани варианти на вече съществуващи обекти. Новите обекти могат да използват (и разширяват) вече дефинираното поведение, без да е необходимо да реализират това поведение отново.”

Проблеми възникнаха при принтирането на дъската за голямата игра ,тъй като за да може да се преизползва функцията от малката игра бе необходимо да се премества курсорът на конзолата, така че отделните малки дъски да не се принтират всичките една под друга. Също така проблем възникна при приоритизирането от изкуственият интелект на победен ход срещу този за блокиране на опонента.

Първият проблем бе решен с помощта на библиотеката *,,Windows.h”* и вградената в нея функция за преместване на курсора по подадени координати, която позволява принтиране на конзолата на точно определена позиция. Вторият проблем бе решен като търсенето на ход се прекретява веднага след като е намерен победен ход, а при наличие на ход за блокиране той се запазва в променлива, а търсенето продължава докато не се обходи цялата дъска или не бъде намерен победен ход.

**2. Проектиране:**

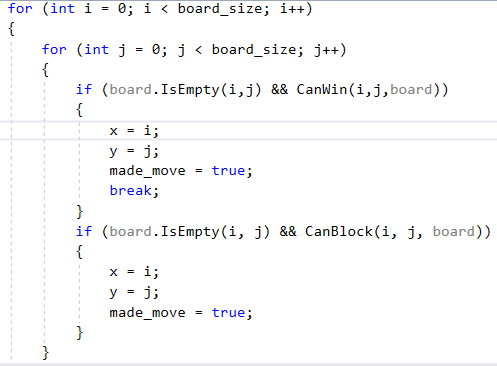
Общата архитектура е доста проста. Има клас Player. Има клас Opponent, който наследява класа Player, в който се реялизира играта на компютърът. Има клас Board, който съдържа представянето на класическата игра. Клас BigBoard, в който има двумерен масив с тип Board, който съдържа представянето на по-голямата версия на играта. Също така има и клас Game, в който има два пойнтъра от тип Player и член данна от тип Board. В него се реализират трите вида игра –класическата, играта срещу компютърът и голямата версия на играта. Всичко това е показано на фигура 1.1.

****

1.1 Архитектира

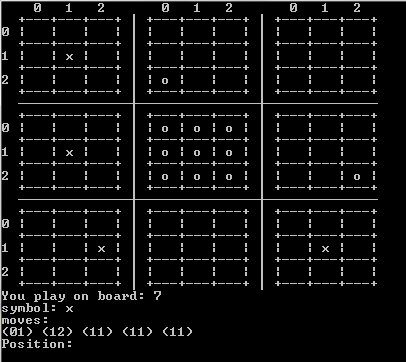
**3. Реализация**

Класът Player пази в себе си символът, с който играе и има масив, в която се запазва историята на неговите ходове. В главната функция MakeMove, която е виртуална и получава като аргумет обект от тип Board, се проверява дали хода на играча е коректен с помощта на друга функция и хода се записва в историята. Във функцията Print на конзолата се изпринтват символът и историята на обекта от тип Player, върху когото е извикана функцията. Класът Opponent наследява класът Player. Той няма собствени член данни. Той има три главни член функции. Пренаписаната MakeMove, която в себе си извиква булевите функции CanBlock и CanWin(1.2) и ако те върнат отрицателен отговор, функцията избира случаен ход.



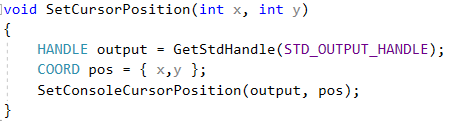
1.2 Win or Block

В класът Board има двумерен масив, който на всяка позиция пази дали там има символ х, символ о или все още никой не е играл на тази позиция. Също така класът пази символът на играча, който е победил, което помога за определянето на победителя в голямата игра. Board притежава функция Win, на която се подава символът на играча и последният му ход и се проверява дали реда или колоната на последният ход са запълнени от символа, а ако последният е някъде в двата главни диагонала се проверяват и те. Има функция Tie, която проверява дали всички позиции в масива са непразни. Двете функции Print и PrintOwn реализират възможността функцията за принтирането на малката дъска да се преизползва за принтирането на голямата като едната функция принтира цифрите на колоните и редовете, а другата не. Класът BigBoard има двумерен масив от тип Board и два индекса, пазещи на коя от малките дъски се играе в момента.



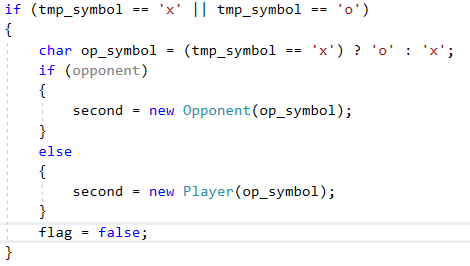
1.3 Принтирани на голямата дъска

Във функцията Print се извиква фунцията за принтиране на всяка една от малките дъски, като за да бъдат изпринтирани на правилните си места, така че да създадат една дъска с размери 9 на 9(1.3), се извиква функцията SetCursorPosition(1.4) реализирана във файла Utility.



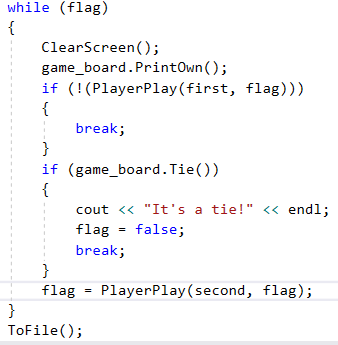
1.4 SetCursorPsotion

Тук отново има функции Win и Tie, като тук те не проверяват отделните позиции, а проверяват победителят от отделните малки дъски. В класът Game ката член данни има два пойнтъра от тип Player и един обект от тип Board. Член функцията на класа ClearScreen, която се извиква след всяка промяна върху дъската, изчиства екрана. Има три главни функции реализиращи трите вида игри. Функции SinglePlayer и MultiPlayer са с почти еднакво тяло затова то е изнесено в отделна функция StartTheGame, на която се подава булев аргумент, от който зависи дали на вторият играч ще се присвои обект от тип Player, или такъв от тип Opponent.(1.5)



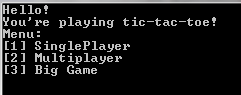
1.5 Opponent or Player

Следователно в SinglePlayer, StartTheGame се извиква с аргумент true и в нея се заделя обект от тип Opponent, а в MultiPlayer обратното. Самата функция StartTheGame има един основен цикъл(1.6), в който играчите се редуват, като след всеки ход екрана си изчиства, а дъската се изпринтира наново. След всики ход също се проверява дали играчат е победил или не се е стигнало до равенство.В края на играта се извиква функция ToFile, която записва данните от играта във файл.



1.6 Основен цикъл

Във функцията BigGame се създава обект от тип BigBoard и отново има цикъл, в който играчите се редуват, изпринтира се дъската и т.н.. Но тялото на тази се различава от телата на другите две, затова тук не се извиква функцията StartTheGame.



1.7 Меню

При стартиране на програмата на конзолата се появяват три опции за трите различни вида игри(1.7). Потребителят избира една от тях и играта започва, питайки го с какъв символ желае да играе. След края на всяка игра се дава възможност да се изиграе още една игра или програмата да приключи.

**Заключение**

Така предоставена играта, тя лесно може да бъде разпространена и да бъде популяризирана. Също така опитът да се симулира изкуствен интелект бе успешен. За в бъдеще проектът може да бъде доразвит като се направи така че изкуственият интелект да може да играе и голямата версия на играта, а също така и историята от голямата игра да се запазва във файл, тъй като в момента тази опция не се поддържа. Също може да се добави и опция да се пази резултатът от поредните игри.

Източноци:

* 1. <https://bg.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D1%8A%D0%BC_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)>
  2. <https://bg.wikipedia.org/wiki/%D0%9E%D0%B1%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%B0%D0%BD%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5>