

UNIVERSITATEA “STEFAN CEL MARE”, SUCEAVA

*FACULTATEA DE INGINERIE ELECTRICA SI STIINTA SPECIALIZAREA
CALCULATOARELOR*

PROIECT
PROGRAMARE ORIENTATĂ PE OBIECTE
X și 0 cu Calculatorul

STUDENTĂ:

LOBIUC ANDRA-EVELINA

CUPRINS

1. ELEMENTE TEORETICE	4
1.1 1.1. Descrierea problemei	4
1.2 Abordarea teoretică a problemei	5
1.3 Elemente specifice POO	5
2. Implementare	7
2.1 Tehnologii folosite	8
2.2 Diagrame de clase	8
3. Analiza soluției implementate	10
3.1 Formatul datelor de intrare/ieșire	10
4. MANUAL DE UTILIZARE-Aplicația X și 0 cu calculatorul	11
5. CONCLUZII.....	13
6. Bibliografie	15

Proiect POO

X și 0 cu calculatorul

Tema și motivarea alegerii

Tema aleasă de mine este X și 0 cu calculatorul.

Am ales această temă deoarece este jocul copilăriei pe care îl jucam atât acasă cu părinții cât și la școală cu colegii. Pe lângă acestea, mai târziu mi-am dat seama că nu este un simplu joc ci ne ajută să ne dezvoltăm. Este un joc de strategie, presupune o gândire logică și nu în ultimul rând constă în descoperirea unor elemente de combinatorică și a unor aranjamente și permutări, dar realizarea unui joc într-un limbaj de programare pentru mine ca și studentă mă ajută să aprofundez limbajul C++ și mă pune în situația de a căuta cele mai eficiente soluții pentru depășirea dificultăților.

Jocul X și 0 este considerat cel mai vechi joc din istorie. În prezent oricine accesează un site cu jocuri online poate juca X și 0 concurând de această dată cu un adversar virtual.

Blocarea adversarului este o strategie esențială atunci când acesta are deja două casuțe completate într-un rând; în caz contrar, următoarea mișcare îi va aduce victoria. Încrucișarea reprezintă o strategie care aduce de cele mai multe ori câștigul; aceasta presupune completarea căsuțelor pe două rânduri diferite, în așa fel încât în momentul în care adversarul va bloca una dintre linii, cealaltă va rămâne liberă.

1. ELEMENTE TEORETICE

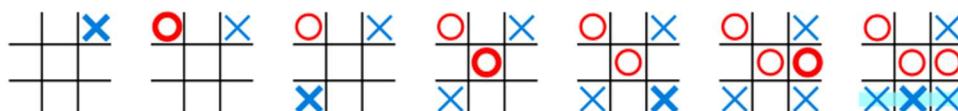
1.1. DESCRIEREA PROBLEMEI

Jocul X și 0 este un joc de strategie de doi jucători, în care fiecare jucător încearcă să formeze o linie continuă de trei simboluri ale sale (X sau 0) pe o tablă de 3x3 pătrată. Jocul începe cu o tablă goală și jucătorii se succed în plasarea simbolului lor pe tablă, alternativ, până când unul din jucători reușește să formeze o linie continuă de trei simboluri ale sale pe orizontală, verticală sau diagonală. Dacă tabla se umple fără ca niciunul dintre jucători să obțină o linie continuă de trei simboluri ale sale, jocul se termină remiză.

În general, regulile jocului sunt următoarele:

1. Primul jucător alege simbolul pe care dorește să îl folosească fie X, fie 0.
2. Jucătorii se succed în plasarea simbolurilor lor pe tabla de joc, alternativ.
3. Niciun jucător nu poate plasa mai mult de un simbol pe o căsuță din tablă.
4. Scopul jocului este de a forma o linie continuă de trei simboluri ale jucătorului pe orizontală, verticală sau diagonală.
5. Jocul se termină atunci când unul din jucători reușește să formeze o linie continuă de trei simboluri sau atunci când se umple tabla și niciunul dintre jucători nu a reușit să formeze o linie continuă de trei.

Acestea sunt regulile generale, însă există și alte variante ale jocului care pot conține unele reguli, modificări sau adăugări.



În reprezentările de mai sus se poate observa modalitatea de a juca acest joc. Primul care a început este "X", iar "0" încearcă să îl blocheze, însă nu reușește ceea ce în final îl face pe "X" câștigător.

1.2 ABORDAREA TEORETICĂ A PROBLEMEI

Aplicația ce urmează să fie implementată este o aplicație în consolă realizată într-un mediu de programare. O aplicație în consolă, cunoscută și sub denumirea de aplicație de linie de comandă este o aplicație software care rulează într-un terminal sau într-o fereastră de comandă și interacționează cu utilizatorul prin introducerea și afișarea de text. O astfel de aplicație poate primi comenzi sau opțiuni de la utilizator și poate furniza rezultate sau afișa informații în mod text.

Operarea cu fișiere într-un program implică manipularea datelor stocate într-un fișier de pe sistemul de fișiere. Aceasta poate include citirea datelor într-un fișier existent, scrierea datelor într-un fișier nou sau modificarea datelor într-un fișier existent.

Operarea cu fișiere este folosită în cadrul aplicației. Salvez într-un fișier starea jocului. Pentru a salva starea unui joc într-un fișier există anumiți pași care ar putea fi urmați: identificarea informațiilor pe care dorești să le stochezi. Acestea pot include poziția jucătorului, scorul, nivelul curent etc.

1.3 ELEMENTE SPECIFICE POO

Pentru modelul de date orientat pe obiecte, se consideră definițiile următoarelor concepte: abstractizarea, obiectul, metoda, clasa, încapsularea, moștenirea, polimorfismul etc.

În cadrul proiectului realizat sunt prezente elementele specifice paradigmei de programare POO.

Primul element specific POO din cadrul proiectului este obiectul. Un obiect este o unitate individualizabilă prin nume, care conține o mulțime de date și funcții. Datele descriu proprietățile și nivelul acestora, iar funcțiile definesc comportamentul. Aceste obiecte pot fi clasificate în mulțimi. O mulțime de obiecte de același fel constituie o clasă de obiecte, descrisă prin modelul comun al obiectelor sale.

Un alt element folosit este clasa care definește un tip abstract de date. Fiecare clasă este caracterizată de metode specifice, fiecare având câte un rol în cadrul programului și a implementării.

Obiectele se generează și se pot inițializa la instanțiere cu ajutorul constructorilor. Constructorii sunt funcții membru care au același nume cu numele clasei și se apelează automat la crearea obiectelor. Funcțiile constructor nu întorc valori, dar nu sunt precedați de cuvântul void. Dacă clasa nu conține constructori, se generează un constructor fără parametri, adică un constructor implicit. El are rolul numai de alocare a obiectelor clasei respective, fără a le inițializa.

Abstractizarea este procesul de definire a unei reprezentări abstracte a unui obiect sau concept într-un program. Scopul abstractizării este de a izola detaliile de implementare și de a se concentra pe caracteristicile esențiale ale obiectului sau conceptului. Abstractizarea este realizată prin intermediul claselor și obiectelor. O clasă

definește o structură abstractă care combină datele (variabilele membru) și comportamentul (funcțiile membru).

Încapsularea se referă la împachetarea datelor și funcțiilor care operează asupra acestora într-o unitate logică numită clasă. Ideea principală a încapsulării este de a ascunde detaliile interne ale clasei și de a oferi un set de interfețe publice prin intermediul cărora utilizatorii pot interacționa cu clasa. Încapsularea este realizată prin intermediul modificatorilor de acces. O clasă poate avea trei niveluri de acces pentru membrii săi: public, privat și protejat. Prin încapsulare, putem proteja datele clasei și putem controla modul în care acestea sunt accesate și modificate din exteriorul clasei.

2. IMPLEMENTARE

Jocul X și 0 cu calculatorul. O idee de implementare care la început a părut simplă, însă mai târziu am observat că în spatele acestui joc se ascunde o tehnică complexă de rezolvare.

În primă fază a implementării programului mi-am propus anumite task-uri și etape pe care să le parcurg pentru a putea în final să reușesc implementarea acestui joc.

Prima etapă a implementării a fost să îmi construiesc corpul claselor de care aveam nevoie la început și câteva metode în interiorul acestora. Bineînțeles, mi-am construit și o clasă de bază în care se desfășoară toate operațiile, aceasta se numește clasa "Joc".


În cadrul acestei clase avem implementată o metodă numită "start()", unde se afișează tabla de joc, se decide rândul jucătorului, se verifică dacă poziția este ocupată sau nu, se anunță câștigătorul.

Pentru a înțelege mai bine contextul din spatele acestei implementări am început prin a construi tabla de joc. La început m-am folosit de o matrice de 3x3 pentru a înțelege foarte bine ceea ce am de urmat. Am creat o clasă denumită "Board" cu metode și date specifice acestei clase. La fel ca și clasa "Joc", clasa "Board" este foarte complexă și importantă în cadrul implementării. Clasa "Board" și clasa "Joc" sunt în strânsă legătură: metodele dezvoltate și implementate în clasa "Board" sunt apelate în clasa "Joc", dar nu numai. Clasa "Board" are practic cele mai importante metode dezvoltate care stau la baza funcționării programului nostru. Prin algoritmizare am reușit să construiesc două metode puternice care funcționează pentru orice tip de tablă de joc și orice dimensiuni. Una dintre metode se numește "check_winner()", iar aceasta verifică dacă există o linie, coloană sau diagonală cu doar X sau cu doar 0. În cadrul acestei funcții este un algoritm complex care verifică toate posibilitățile. Cea de a doua metodă se numește "computerMove()" care verifică, de asemenea, toate posibilitățile computerului de a își bloca adversarul și de a deveni câștigător. Un alt element important din această clasă este constructorul acesteia în care construim tabla de joc și inițializăm cu -1 fiecare element al tablei de joc. Această valoare semnifică că poziția respectivă nu a fost completată cu X sau 0.

O altă clasă implementată este clasa "Punct" care are rolul de a exprima locația unui element într-o matrice. Ne vom folosi de aceasta, la scrierea algoritmului care implică contribuția calculatorului.

Aplicația este realizată sub forma unui meniu. Avem implementată și o clasă cu această denumire. În momentul rulării programului, în linia de comandă se afișează

opțiunile meniului și un mesaj în care se cere să alegem opțiunea, ca în fotografia de mai jos.



```
1.Start
2. Salvare în fisier
3.Help
4.Exit
Alegeti o optiune:
```

Fig. 1. Afișarea în linia de comandă

O altă clasă implementată se numește "Input()" și aceasta verifică a cui este rândul: al jucătorului X sau al jucătorului 0.

2.1 TEHNOLOGII FOLOSITE

Am precizat și mai sus ca aplicația implementată este o aplicație în consolă realizată în mediul de programare CODE::Blocks. Acesta este un mediu de dezvoltare integrat (IDE) gratuit și open-source, care oferă un set de instrumente pentru dezvoltarea și testarea aplicațiilor software în diferite limbaje de programare. Cu ajutorul Code::Blocks îți poți organiza și gestiona proiectele, poți crea unele noi, adăuga fișiere sursă, seta opțiuni de compilare etc. Code::Blocks este disponibil pentru diferite platforme, Windows, Linux, macOS.

Limbajul folosit în construirea aplicației este C++. Acesta este un limbaj de programare orientat pe obiecte, ceea ce înseamnă că permite definirea de clase și crearea de obiecte care încapsulează date și comportamente. Facilitează dezvoltarea modulară și reutilizarea codului prin intermediul conceptelor de încapsulare, abstractizare, moștenire. C++ este un limbaj de nivel jos care permite controlul precis asupra resurselor și performanței. Oferă suport pentru lucrul direct cu memoria, manipularea eficientă a pointerilor și optimizarea codului pentru a obține performanțe ridicate.

2.2 DIAGrame DE CLASE

Folosind aplicația web VISUAL PARADIGM ONLINE am realizat diagrama UML de clase în care se poate observa dependența claselor Board și Input de clasa principală Joc.

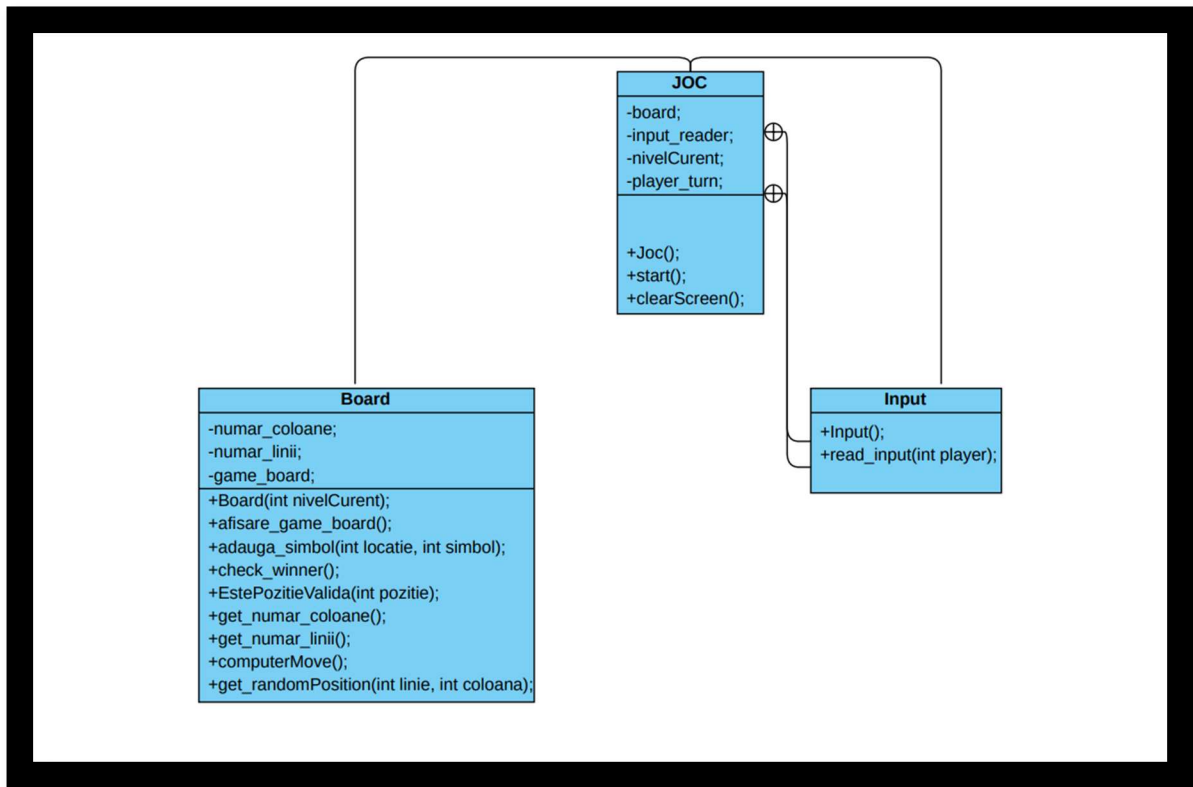


Fig. 2. Diagrama UML de clase a aplicației

3. ANALIZA SOLUȚIEI IMPLEMENTATE

3.1 FORMATUL DATELOR DE INTRARE/IEȘIRE

Operarea cu fișiere o folosim în cadrul acestei aplicații. Salvăm în fișier starea jocului, adică în cadrul aplicației dacă nu s-a finalizat jocul și nu s-a anunțat câștigătorul chiar dacă ieși din aplicație și dorești să te întorci acest lucru este posibil, însă te întorci la nivelul la care ai rămas. Se salvează în fișier numărul nivelului la care ai rămas. Acest lucru constituie un mare avantaj deoarece nu mai este necesar să reiei toate nivelurile și poți trece cu ușurință la nivelul superior.

4. MANUAL DE UTILIZARE-APLICAȚIA X ȘI 0 CU CALCULATORUL

Introducere

Aplicația "X și 0 cu Calculatorul" este un joc clasic de X și 0 (Tic-Tac-Toe) în care utilizatorul se joacă împotriva calculatorului. O particularitate adăugată este faptul că, odată ce utilizatorul câștigă, nivelul jocului crește și tabla de joc devine mai mare (de exemplu, de la o matrice 3x3 la o matrice 4x4, 5x5 etc.).

Interfața Utilizatorului

La deschiderea aplicației, veți fi întâmpinat de o interfață simplă și intuitivă. Veți vedea o matrice de 3x3, reprezentând tabla de joc, afișată pe ecran. Fiecare celulă a tablei de joc poate fi reprezentată de unul dintre următoarele simboluri: X, O sau spațiu gol.

Desfășurarea Jocului

- Inițial, toate celulele tablei de joc vor fi goale. Utilizatorul și calculatorul se vor alterna pentru a face mișcări.
- Utilizatorul va începe jocul și va selecta o celulă goală în care să plaseze simbolul X. Acest lucru se poate face prin introducerea coordonatelor (rând și coloană) sau prin interacțiunea cu interfața grafică.
- Calculatorul va efectua o mișcare automată, plasând simbolul O într-o celulă goală disponibilă.
- Jocul va continua în acest fel, cu utilizatorul și calculatorul alternându-se, până când unul dintre ei câștigă sau tabla de joc este completă (egalitate).
- Dacă utilizatorul câștigă jocul, nivelul se va înălța, iar tabla de joc va crește în dimensiune. De exemplu, o matrice 3x3 se va transforma într-o matrice 4x4 pentru nivelul următor.
- Procesul se va repeta pentru fiecare nivel câștigat, mărin dimensiunea tablei de joc la fiecare nivel nou.

Sfaturi și Observații

1. Luați în considerare strategii și tactici pentru a câștiga împotriva calculatorului.

Observați că, odată ce tabla de joc devine mai mare, va fi mai dificil să obțineți o linie, coloană sau diagonală completă de simboluri.

2. Mențineți-vă atenția asupra mișcărilor calculatorului și încercați să preveniți câștigul acestuia.

3. Jucați cu strategie și adaptați-vă la dimensiunile tablei de joc în creștere.

Încheiere

Aplicația "X și 0 cu Calculatorul" vă oferă o modalitate distractivă de a juca jocul clasic de X și 0 împotriva calculatorului, adăugând o provocare suplimentară prin creșterea dimensiunii tablei de joc odată cu progresul în joc. Bucurați-vă de joc și deveniți campionul X și 0!

Pentru orice întrebări sau probleme, vă rugăm să consultați documentația.

Spor la joc!

5. CONCLUZII

Am început implementarea cu pași mici, adică am început de la lucrurile ușoare și am evoluat spre cele mai complicate. Pentru început mi-am propus câteva task-uri pe care le-am îndeplinit, însă nu au fost de ajuns. Lucrurile s-au mai complicat, iar noutățile nu au încetat să apară. Într-adevăr, au fost momente când m-am blocat și nu am mai putut continua, însă cautând în mai multe surse am reușit să trec peste acele momente.

La început m-am gândit doar la clasele de bază, însă pe parcurs au mai apărut alte clase, alte metode, dar și alte funcții pe care nu le-am mai folosit până atunci.

O clasă nouă implementată pe parcurs a fost clasa "Punct" pe care am folosit-o în momentul scrierii algoritmului care implică contribuția calculatorului și care are rolul de a exprima locația unui element într-o matrice.

Am folosit și biblioteci, funcții pe care nu le-am mai folosit până acum. De exemplu, `#include <chrono>`, `#include <algorithm>` care mi-au simplificat uneori munca. Funcția `std::shuffle` este o funcție inclusă în biblioteca `algorithm`, iar cu ajutorul acesteia am amestecat elementele unui vector, iar jucătorul 0, adică calculatorul la început alege o poziție random din matrice.

Aplicația propusă de la început s-a bazat pe o tablă de joc normală, adică o matrice de 3x3, însă pe parcurs am ales să personalizez implementarea și să împart aplicația pe mai multe niveluri: la primul nivel avem o matrice de 3x3, la al doilea nivel creștem matricea, adică va fi de 4x4, la nivelul trei vom avea o matrice de 5x5 etc. Acest lucru va face ca jocul să devină mai interactiv și mai distractiv.

Aplicația finală respectă cerințele și etapele propuse de la început, dar cu mai multe adăugări. Pot spune ca aplicația finală este mult mai complexă decât cea pe care mi-am creat-o la început, este bine organizată, împărțită pe fișiere și destul de sugestivă.

AVANTAJE:

1. Mai multă complexitate: Prin creșterea tablei de joc la fiecare nivel câștigat, jocul devine mai complex și mai provocator. Jucătorii trebuie să se adapteze la dimensiuni și configurații de joc în continuă schimbare, ceea ce poate duce la strategii mai elaborate și decizii mai interesante.

2. Păstrarea interesului jucătorilor avansați: Odată ce jucătorii câștigă nivelurile inițiale, creșterea tablei de joc oferă o continuare captivantă și o provocare suplimentară pentru jucătorii avansați. Aceasta îi ține implicați și le oferă oportunitatea de a se dezvolta în joc.

3. Varietate și diversitate: Creșterea tablei de joc adaugă varietate și diversitate la joc. Fiecare nivel poate avea o configurație unică a tablei de joc, ceea ce înseamnă că fiecare partidă poate fi diferită și imprevizibilă. Aceasta menține interesul jucătorilor și îi determină să încerce noi strategii și abordări.

Dezavantaje:

1. Complexitate crescută pentru începători: Creșterea tablei de joc poate face jocul mai dificil pentru jucătorii începători sau mai puțin experimentați. Acest lucru poate duce la o curbă de învățare mai abruptă și la o mai mare dificultate în înțelegerea și aplicarea strategiilor de joc.

2. Timp de joc mai mare: Odată ce tabla de joc crește, timpul de joc poate fi extins. Partidele pot deveni mai lungi și mai complexe, deoarece există mai multe celule și mai multe combinații posibile. Acest aspect poate fi un dezavantaj pentru jucătorii care doresc partide scurte și rapide.

3. Posibile erori de implementare: Implementarea corectă și eficientă a creșterii tablei de joc poate fi complexă și poate fi predispusă la erori de programare. În timp ce adăugarea funcționalității de creștere a tablei poate aduce avantaje, trebuie acordată o atenție sporită asigurării corectitudinii și performanței în implementare.

6. BIBLIOGRAFIE

Surse bibliografice:

- <https://online.visual-paradigm.com/diagrams/features/uml-tool/>
- <https://ramonnastase.ro/blog/programare-orientata-pe-obiecte-poo/>
- <https://en.cppreference.com/w/cpp/algorithm>