НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Кафедра автоматизації проектування енергетичних процесів і систем

**КУРСОВА РОБОТА**

з дисципліни «Основи програмування – 3»

(назва дисципліни)

на тему: «Шахи»

Студента групи \_\_\_\_\_\_\_\_\_\_\_\_\_ТІ-01\_\_\_\_\_\_\_\_\_\_\_\_\_\_

напряму підготовки ***6.050103 Програмна інженерія***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Круть К.О.\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Керівник\_\_\_\_\_\_\_\_\_доц., к.т.н. Шпурик В.В.\_\_\_\_\_\_\_\_

Національна оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка: ECTS \_\_\_\_\_\_\_

Члени комісії: \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_доц., к.т.н. Шпурик В.В.\_\_\_\_\_\_

(підпис)

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_асист. Олєнєва К.М.\_\_\_\_\_\_\_\_\_

(підпис)

Київ – 2021 рік

**Національний технічний університет України**

**“Київський політехнічний інститут імені Ігоря Сікорського”**

Факультет ТЕПЛОЕНЕРГЕТИЧНИЙ \_

( повна назва )

Кафедра автоматизації проектування енергетичних процесів та систем \_

( повна назва )

Напрям підготовки 6.050103 Програмна інженерія \_

( шифр і назва )

**З А В Д А Н Н Я**

**НА КУРСОВУ РОБОТУ СТУДЕНТУ**

Круть Катерині Олександрівні

(прізвище, ім’я, по-батькові)

1.Тема роботи: Комп’ютерна гра «Шахи»

керівник курсової роботи – доц.,к.т.н. Шпурик В. В.\_

( прізвище, ім’я, по батькові, науковий ступінь, вчене звання )

2. Строк подання студентом роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_04.06.2021\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до проекту (роботи): середовище розробки – Visual Studio Code \_

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) – розробка та створення комп’ютерної гри «Шахи» мовою програмування C++ в середовищі Visual Studio Code

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів виконання дипломної  роботи | Строк виконання етапів роботи | Примітка |
| 1. | Отримання теми та технічного завдання. | 02.02.21 |  |
| 2. | Уточнення завдання та вхідних матеріалів. | 24.02.21 |  |
| 3. | Створення опису предметної області. | 10.03.21 |  |
| 4. | Формування вимог до системи, що проектується. | 24.03.21 |  |
| 5. | Визначення основних параметрів програмного продукту та засобів розробки. | 14.04.21 |  |
| 6. | Проектування – розробка та реалізація окремих модулів програми. | 5.05.21 |  |
| 7. | Тестування – збірка програмного продукту та його тестування. | 19.05.21 |  |
| 8. | Оформлення пояснювальної записки. | 22.05.21 |  |
| 9. | Захист роботи. | 24.05.21 |  |

**Студент \_\_\_\_\_\_\_\_\_** \_\_\_\_ Круть К.О.\_\_\_\_\_

( підпис ) (прізвище та ініціали)

**Керівник курсової роботи \_\_\_\_\_\_\_\_\_** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

( підпис ) (прізвище та ініціали)

### АНОТАЦІЯ

З моменту винайдення ЕОМ тема розробки комп’ютерних шах завжди викликала інтерес. Проблема розробки програми здатної вирішувати шахові задачі досі залишається актуальною. У ході написання курсової роботи розроблено алгоритм, що дозволяє аналізувати шахову дошку та знаходити всі можливі коректні варіанти ходів. Програма написана на основі цього алгоритму дозволяє парно зіграти в шахи, а зручний інтерфейс робить її використання зрозумілим та простим. Реалізовано програму мовою С++ з використанням об’єктно-орієнтованого програмування.

Обсяг пояснювальної записки 31 аркушів, кількість ілюстрацій – 11, 3 додатка.

**ANNOTATION**

From the moment of invention of the electronic computer the topic of creation computer chess always attracted the interest of people. [The problem of development](https://context.reverso.net/%D0%BF%D0%B5%D1%80%D0%B5%D0%B2%D0%BE%D0%B4/%D0%B0%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9-%D1%80%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9/The+problem+of+development) a program capable to solve chess problems remains relevant. The algorithm, made while writing the course work, allows analyze chessboard and generate all correct possible moves for each figure. The program written on this algorithm provides an opportunity to play a chess game for two players. Comfortable interface makes the game intelligible and simple for using. This program is written in C++ programming language with using principles and methods of object-oriented programming.

The volume of the explanatory note is 31 sheets, the number of illustrations is 11, 3 appendices.

### ЗМІСТ

ВСТУП 5

РОЗДІЛ 1. АНАЛІЗ ЗАВДАННЯ 7

1.1. Теоретична складова 8

1.2. Порівняння різних методів розробки гри в шахи 10

РОЗДІЛ 2. РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ 13

2.1. Середовище розробки VS Code 13

2.2. Опис класів та методів програми 14

2.3. Використані типи даних 17

2.4. Опис роботи коду програми 20

РОЗДІЛ 3. КЕРІВНИЦТВО КОРИСТУВАЧА 24

ВИСНОВКИ 30

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 31

ДОДАТОК 1. Специфікація...…………………………………………………….32

ДОДАТОК 2. Код програми..…………………………………………………….33

ДОДАТОК 3. Опис програми…………………………………………………….45

### ВСТУП

Шахи безумовно можна назвати грою всіх часів та народів, яка не залишала байдужою нікого. Історія гри є дуже давньою та бере свої початки з V століття, за цей час вона стала невід’ємною частиною соціокультурного життя людини, мала вплив на розвиток науки, історію та навіть на політичне життя.

Історія розвитку комп’ютерних технологій нерозривно пов’язаний з розвитком комп’ютерних шахів. В середині XX століття відомі математики та знавці криптоаналізу Клод Шеннон та Алан Тюрінг всерйоз зайнялись дослідженням теми програмування комп’ютерних шахів. І вже в 1950 Тюрінг написав алгоритм для гри в шахи. Це і було початком зародження шахового програмування. Як відомо, одним із перших експериментів проведених комп’ютері MANIAC, спроектованого фон Нейманом, було написання шахової програми. В 1957 році інженер IBM Алекс Бернтейн створив першу повноцінну програму, яка надавала можливість зіграти повну партію в шахи на стандартній дошці з усіма фігурами. Після цього багато інших видатних науковців займались дослідженням цієї теми та вдосконаленням алгоритмів «шахів». Ще Клод Шеннон писав: «Хоча, можливо, це й не має жодного практичного значення, саме питання є, теоретично, цікавим, і сподіватимемося, що вирішення цієї задачі послужить поштовхом для вирішення інших задач аналогічної природи й більшого значення».

Так і є вирішення цієї задачі й досі не втрачає своєї актуальності. Досягнення розвитку шахового програмування мало і має великий вплив на розвиток IT технологій. Розробка комп’ютерної версії цієї гри допомогла вдосконалити багато існуючих та розробити нові алгоритми пов’язані з оцінкою, аналізом та вибором найкращого варіанту. Це привело до створення евристичного програмування. Досвід створення комп’ютерних програм дозволяє сьогодні ефективно використовувати ЕОМ для управління складними економічними та підприємницькими об’єктами. Надзвичайно великий вплив програмування комп’ютерних шахів має на розробку, розвиток та вдосконалення штучного інтелекту, адже воно «вчить машину мислити» [1]. Прикладом високих досягнень розвитку цієї області є перемога в 1996 році Deep Blue в шаховій партії з тринадцятим чемпіоном світу по шахам Гаррі Каспаровим. Іншим прикладом є створення у 2017 році програми AlphaZero, яка проілюструвала небачений раніше рівень генерації можливих варіантів та перемогла багатьох чемпіонів світу в шахи.

Мета цієї курсової роботи полягає в дослідженні теми розробки комп’ютерних шахів, здобуті нових та вдосконалені набутих вмінь та навичок з програмування, визначення основних складових проекту та організації його реалізації.

Завдання: розробити та реалізувати алгоритм для реалізації комп’ютерної гри «Шахи», що надавала б користувачам можливість парної консольної гри із зручним інтерфейсом, аналізувала ходи гравців та контролювала їх коректність.

### РОЗДІЛ 1. АНАЛІЗ ЗАВДАННЯ

Тема реалізації комп’ютерної гри у шахи завжди залишалась актуальною. Вирішення цієї проблеми різними методами мало великий вплив на розвиток багатьох наук: математики, комп’ютерних технологій, програмування, кібернетики та штучного інтелекту. Хоч за довгу історію шахового програмування розроблено не одну версію цієї гри, актуальності ідея вирішення не втрачає, адже кожна реалізація є унікальною та може зробити свій внесок у світ науки. Крім того комп’ютерна версія сьогодні дозволяє легко зіграти в гру, не використовуючи справжньої шахової дошки та фігур, тож робить гру доступною для будь-кого в будь-який час. Така реалізація дозволить зіграти з другом, або ж повправлятися самому, що значно допомагає, наприклад для досвідчених гравців при підготовці до змагань, або ж для розваги.

Головним завданням цієї курсової роботи є розробка консольної гри в шахи. Реалізацію цієї гри не можна назвати простою, адже перш за все необхідно розробити алгоритм, який би забезпечив збереження усіх стандартів гри: наявність абстракцій шахової дошки та всіх фігур, дотримання всіх сформованих правил, та визначення закінчення партії й сторони, яка виграла. Такий алгоритм повинен вміти аналізувати стан шахової дошки та кожної фігури окремо, оцінювати її положення та на основі цих даних генерувати всі можливі найкращі ходи, тобто ті, що відповідають правилам гри та є коректними в певній ситуації, вміти визначати шах (check) та (checkmate), тобто в останньому випадку кінець гри.

Програма на основі цього алгоритму повинна бути реалізована у вигляді консольного застосунку та бути зрозумілою й зручною для користувача у використанні. Повинне бути передбачене меню програми, де йому надавалося б декілька варіантів можливих дій: зіграти партію, прочитати правила шахів та вийти з додатку. Програма повинна бути написана так, щоб виводилась шахова дошка з всіма тридцятьма двома фігурами на ній та варіанти всіх ходів, що можна здійснити в певний момент, з можливістю вибору одного. Після кожного ходу стан дошки та варіанти ходів повинні змінюватись. Після завершення партії необхідно виводити дані про витрачений час, кількість ходів та сторону, котра виграла у партії.

### 1.1 Теоретична складова

Оскільки програма повинна бути реалізована на мові С++, то варто описати деякі особливості цієї мови та деякі її парадигми.

Мова програмування С++ є високорівневою компільованою мовою програмування, яка підходить для розробки найрізноманітніших додатків. Сьогодні С++ є однією з найуживаніших мов програмування.

Свої початки вона бере з розробленої Деннісом Рітчі в 1969-1973 мови С. У 1980-х данський програміст Бйорн Страуструп спочатку розробив мову С++ просто як розширення до С. Тобто вона лише мала доповнювати Сі деякими можливостями об'єктно-орієнтованого програмування. І тому сам Страуструп спочатку називав його як «C with classes». Згодом нова мова стала набирати популярності.

С++ успадкувала від С багато можливостей, особливо роботи з пам’яттю. Тому вона часто застосовується в системному програмуванні, зокрема, при створенні операційних систем, драйверів, різних утиліт та антивірусів навіть ОС Windows здебільшого написана на С ++. Проте через свою швидкість та продуктивність він використовується і для розробки графічних додатків та прикладних програм, а також для створення ігор. Крім того, сьогодні С++ широко використовується для розробки мобільних додатків, веб-додатків та веб-розробки. На відміну від С мову C++ дозволяє писати програми в об'єктно-орієнтованому стилі.

Об'єктно-орієнтоване програмування (object-oriented programming) – це метод програмування, який базується на поданні програми як сукупності взаємодіючих між собою об'єктів, кожен з яких є екземпляром певного класу, котрі в свою чергу є членами певної ієрархії наслідування.

Основним завданням такого підходу програмування ООП є зробити складний та великий за об’ємом код простіше. Саме для цього були розроблені принципи об'єктно-орієнтованого програмування. Об'єктно-орієнтоване програмування є поєднанням кращих ідей структурного програмування та потужних концепцій, які дозволяють оптимізувати та вдосконалити програми.

Об’єктно-орієнтоване програмування дозволяє уявити програму як об’єкт, який можна умовно розбити на інші об’єктами, які містять свої окремі частини коду з унікальними властивостями. Це що дозволяє спростити написання коду та дає можливість створювати більші за обсягом програми.

Основу об'єктно-орієнтованого програмування складають чотири основні концепції: [інкапсуляція](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D1%96%D1%8F), [успадкування](https://uk.wikipedia.org/wiki/%D0%A3%D1%81%D0%BF%D0%B0%D0%B4%D0%BA%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), [поліморфізм](https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%96%D0%BC%D0%BE%D1%80%D1%84%D1%96%D0%B7%D0%BC_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) та абстракція.

Абстракція – в об'єктно-орієнтованому програмуванні це додання об'єкту характеристик, які відрізняють його від всіх об'єктів, чітко визначаючи його концептуальні межі. Основна ідея абстракції полягає в тому, щоб відокремити спосіб використання складових об'єктів даних від деталей їх реалізації у вигляді більш простих об'єктів. Такий підхід дозволяє працювати з об'єктами, не вдаючись у особливості їх реалізації. У кожному конкретному випадку застосовується той чи інший підхід: інкапсуляція, поліморфізм або успадкування.

Абстракція даних – популярна і зазвичай неправильно визначена техніка програмування. Фундаментальна ідея полягає в поділі несуттєвих деталей реалізації підпрограми і характеристик істотних для коректного її використання. Такий поділ може бути виражено через спеціальний «інтерфейс». З точки зору теорії множин, процес являє собою організацію для групи підмножин свого безлічі. Див. Також Закон зворотного відносини між змістом і обсягом поняття.

Інкапсуляція (incapsulation) – це механізм, що зв'язує воєдино код і дані, якими він маніпулює, а також забезпечує їх захист від зовнішнього втручання та неправильного використання. В мовах, що підтримують ООП код і дані можна поміщати в об’єкти так звані «чорні ящики». Інакше кажучи, об'єкти – це засоби інкапсуляції. В С++ інкапсуляція реалізується через специфікатор доступу. Усередині об'єкту код і дані можуть бути закритими (private) або відкритими (public). Закритий код або дані об'єкта доступні тільки з іншої частини цього ж об'єкта. Інакше кажучи, до закритої частини коду або даних неможливо звернутися ззовні. Якщо код або дані є відкритими, вони доступні з будь-якої частини програми. Як правило, відкрита частина коду забезпечує взаємодію із закритими елементами об'єкта. Як з синтаксичної, так і з семантичної точки зору об'єкт являє собою змінну, тип якої визначено користувачем.

Об’єктно-орієнтоване програмування підтримує поліморфізм (polymorphism), який можна охарактеризувати фразою «один інтерфейс, кілька методів». Простіше кажучи, поліморфізм – це атрибут, що дозволяє за допомогою одного інтерфейсу управляти доступом до цілого класу методів. Вибір здійснюється в залежності від ситуації, котра виникла Виконання кожного конкретної дії буде визначатися типом даних. Зазвичай поліморфізм виражається в перезавантаженні методів класу. Тобто функції з одним і тим же ім’ям є визначеними для різних типів даних та можуть виконувати одні дії або певним чином відрізнятись. Це означає, що можна створити загальний інтерфейс для групи близьких за змістом дій. Таким чином поліморфізм допомагає спрощувати та узагальнювати програми, адже завдання вибору правильного методу покладається на компілятор. Ще одним застосуванням поліморфізму є перезавантаження до операторів, що за своїм принципом подібне до перезавантаження функцій.

Наслідування (inheritance) – це процес, в ході якого один об'єкт може набувати властивості іншого. Він має велике значення для об’єктно-орієнтованого програмування, оскільки підтримує концепцію класифікації (classification). Якби класифікації не існувало було б неможливо точно описати властивості об'єктів. Однак при цьому необхідно вказувати тільки унікальні властивості, що дозволяють виділити об’єкт серед інших в даному класу. Саме цей принцип лежить в основі механізму наслідування. Це дає можливість вважати конкретний об'єкт специфічним екземпляром більш загального різновиду [2].

### 1.2. Порівняння різних методів розробки гри в шахи

Розробка комп’ютерних програм для гри в шахи нерозривно пов’язана зі штучним інтелектом, адже в будь-якого ходу є результат – змінені стани фігур на дошці. Таким чином, доречно говорити про необхідність алгоритму, що здійснював би перевірку ходів та можливих гральних ситуацій на дошці. В основі кожного такого алгоритму лежить поняття пошуку, що є одним із фундаментальних методів класичного AI (artificial intelligent) і означає процес проб і помилок при обході можливих шляхів вирішення задачі, у нашому випадку – вибору коректних ходів. Зазвичай перебір (пошук) варіантів вибору можна уявити у вигляді дерева, що має свою глибину.

Першопроходцем в дослідженні комп’ютерних шахів був Клод Шеннон, який ще в 60-х роках XX століття придумав два основних типи пошуку: «Тип А» та «Тип В» – відомі також як методи Шеннона

В основі Методу Шенона(А) лежить підхід «грубої сили» (brute force), тобто відбувається перебір усіх можливих ходів з певною глибиною пошуку. Тож, як відомо, на кожному ході є максимум 38 варіантів ходу, і для вибору з них “кращих” в методі Шеннона передбачено використання евристичної функції, тобто методу оцінки одного ходу відносно іншого. Але враховуючи, що середня кількість ходів в партії для одного гравця – 42, а для двох 84, маємо, то за гру необхідно перебрати варіантів. Отже, алгоритм побудований на цьому методі є неефективним [3].

Пізніше Шеннон запропонував використання алгоритмів «Типу В» для уникнення аналізу непідходящих ходів. Цей метод є вдосконаленням методу «Тип А». Алгоритм вже не перебиратиме всі можливі варіанти, а тільки ті є підходящими для кожної позиції в даній ситуації. Швидкодія цього методу набагато вища ніж у інших [3].

Метод Ньюела, Саймона та Шоу або alpha-beta: алгоритм alpha-beta був створений з метою зменшення кількості оцінюваних варіантів та збільшення швидкодії програм. Зазвичай його використовують в антагоністичних комп’ютерних іграх, де необхідний аналіз можливих варіантів кроків. Головна ідея алгоритму у відкиданні некоректних варіантів. Тепер користувач триматиме інтервал відсікань (нижня і верхня межі – альфа і бета відповідно) і оцінки всіх вузлів, які не потрапляють в інтервал знизу не розглядатиметься (так як вони не впливають на результат – це просто гірші ходи, ніж вже знайдений), а сам інтервал звужується по мірі знаходження кращих ходів. Якщо прийняти, що в середині партії в одного боку є приблизно 40 різних ходів, то час алгоритму можна оцінити як , де – глибина дерева ходів. У кращому випадку за допомогою альфа-бета відсікання можна уникнути перевірки кореня з числа всіх ходів в мінімакс. Для того, щоб уникнути занадто довгого виконання, перебір в дереві роблять на якусь фіксовану величину і там проводять оцінку ходу [4].

Усі наведені вище алгоритми сьогодні не використовуються у своєму чистому вигляді. Найбільш продуманим але найменш швидким алгоритмом є алгоритм повного перебору (метод Шенона(А)). Найбільш швидким методом побудови алгоритмів є метод Шенона(В), проте за продуманістю та практичністю цей метод все одно програє першому. Третій метод (alpha-beta) є гібридом перших двох методів, тому за швидкістю він не відстає від методу “Типу В”, а за результатами, які він надає не програє методу повного перебору.

### РОЗДІЛ 2. ПРОГРАМНА РЕАЛІЗАЦІЯ

Першим етапом перед безпосередньою розробкою алгоритму та написанням коду програми є визначення середовища, яке буде використане для розробки продукту. Серед широкого кола доступних ми зупинились на Visual Studio Code.

### 2.1. Середовище розробки VS Code

Visual Studio Code — один з найпопулярніших засобів, що дозволяє легко та зручно редагувати та налагоджувати програми, сучасні веб-застосунків та навіть хмарні системи. VS Code (Visual Studio Code) є відносять до «легких» кроспалтфорних редокторів, отже дозволяє обійтися без повного інтегрованого середовища розробки. Він є безкоштовним та доступним на найпопулярніших платформах: Windows, Linux та macOS.

VS Code включає в себе вбудований відлагоджувач містить інструменти для роботи з Git, підтримує рефакторинг, надає можливість навігації по коду, автодоповнення типових конструкцій і контекстної підказки, крім того, дозволяє налаштовувати редактор під користувача.

Як було сказано, VS Code – це перш за все редактор вихідного коду. Він підтримує багато популярних мов програмування: [Batch,](https://ru.wikipedia.org/wiki/Batch_file) [C](https://ru.wikipedia.org/wiki/C_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), [C#](https://ru.wikipedia.org/wiki/C_Sharp), [C++](https://ru.wikipedia.org/wiki/C%2B%2B), [CSS](https://ru.wikipedia.org/wiki/CSS), [Go](https://ru.wikipedia.org/wiki/Go_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), [HTML](https://ru.wikipedia.org/wiki/HTML), [Java](https://ru.wikipedia.org/wiki/Java_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), [JavaScript](https://ru.wikipedia.org/wiki/JavaScript) та багато інших. Варто загадати про доступну в редакторі технологію IntelliSense розроблену Microsoft Visual Studio. Крім перечислених можливостей та переваг, VS Code підтримує підсвітку коду для багатьох мов, серед яких і C++, налаштування сполучення клавіш для виконання дій, у редактора зручний та інтуїтивно зрозумілий інтерфейс. Visual Studio Code дає надає можливість легко та швидко через [Visual Studio Marketplace](https://marketplace.visualstudio.com/vscode) підключати плагіни, що включають в себе доповнення до редактора, підключення підтримки додаткових мов програмування та різні аналізатори коду. Це все дозволяє розробникам легко та швидко налаштувати середовище для роботи та зробити його максимально зручним за себе. Саме тому VS Code ідеально підходить як для початківців, так і для досвідчених програмістів, професіоналів.

Наш вибір припав саме на середовище Visual Studio Code через ряд вагомих причин:

* підтримка необхідної мови написання, а саме C++;
* підключена функція дебаггінгу в поєднанні з легкістю застосунку;
* доступна технологія IntelliSense;
* можливість легко редагувати, структурувати та виправляти код;
* зручний інтерфейс та можливість його налаштування.

### 2.2. Опис класів та методів програми

Оскільки гра написана мовою С++ з використанням об’єктно-орієнтованого програмування, то головна роль в коді, звичайно, належить класам, саме в них описані всі функції, необхідні для роботи цієї програми.

Клас можна назвати рушієм С++, адже він призначений для трансляції абстракцій в користувацькі типи. Він комбінує представлення «властивостей» (змінних, також поля) та «методів» (функцій, необхідних для роботи з цими полями), він призначений для роботи з ними (властивостями) в межах одного акуратного так званого «пакету».

Змінна-об'єкт, що відноситься до заданого класом типу, називається екземпляром цього класу. Хоча і клас можна в деякому контексті розглядати як об’єкт, це явно виражається під час виконання програм за допомогою динамічної ідентифікації типу даних. Правильне програмування класів полягає в логічному об’єднанні його властивостей, його цілісності з необхідними методами, використанні механізмів ООП та забезпеченні простоти інтерфейсу [5].

В програмі визначено два класи: Chess, що є головним, адже він містить всі її головні функції, та ChessGame – допоміжний клас, що призначений для забезпечення зациклення виконання програми. Саме ці класи з їх властивостями та методами і становлять основу програми.

Клас Chess складається з декількох перерахувань, що є дуже важливими: Figures – це перечислення фігур гри, з врахуванням пустої клітинки та границі; Angels – відповідає за перечислення векторів напрямку. Також невід’ємною частиною програми є структури, їх у коді три. Структура Cell можна назвати головною, вона відповідає за збереження стану клітинки: значення фігури, можливої траекторії руху фігури, індикатор атакованого поля та індикатор для позначення фігури під зв’язкою. Дві додаткові структури – це Move та Extra. Перша призначена для збереження координат зробленого ходу: координати звідки, куди та фігури, яка здійснила хід. Друга відповідає за збереження координат пішака, який бере учать у взятті на проході та індикатору для подвійної зв’язки.

В реалізації використовується два масиви, а саме: moveptr\_list – масив для збереження кількості ходів та castling\_active необхідний для перевірки фігур, що беруть участь у рокіруванні. Також важливу роль відіграє vector moves, що призначений для зберігання усіх можливих в даний момент варіантів ходів з використанням кортежів, що дають змогу одночасно виконувати дії над координатами клітинок пересування.

Для задання початкового стану дошки, було використано стрінг \*starting, на якому визначенні фігури, перший хід та стан рокіровок. В реалізації гри передбачене використання чотирьох індикаторів, а саме: white\_moves та black\_moves, що є лічильниками ходів для білих та чорних фігур відповідно, side – для визначення сторони гравця та, звичайно, дуже важливий – attacks\_on\_king, необхідний для підрахунку атак на короля (шах).

Однією з головних складових програмної реалізації шахів є абстракція на дошку. В цій програмі вона представлена як об’єкт структури Cell, що зберігає стани кожної клітинки, тож для них необхідна, в першу чергу ініціалізація за що відповідає функція init\_desk та функції очистки дошки clear\_desk та clear\_desk\_state, призначена для очистки особливих станів клітинок дошки. Обробка другої, допоміжної, дошки, що відповідає за збереження доступних ходів належить таким функціям:

* void clear\_move\_table – очистка масиву ходів;
* void init\_move\_table – ініціалізація масиву для ходів;
* void destroy\_move\_table – очистка дошки ходів.

Оскільки на початку роботи програми відбувається встановлення початкових станів, то необхідні функції, які б дозволили це зробити. Головною з них є unpack\_position(), яка містить наступні: unpack\_desk відповідає за “розпакування” стартової дошки, unpack\_castling – встановлення початкових станів фігур рокіровки, unpack\_move – установка, першого ходу, set\_current\_move – функція запису першого ходу в структуру.

Функції, необхідні для здійснення та перевірки ходів є невід’ємною частиною будь-якої головоломки, комп’ютерні шахи – не виняток:

* void add\_move –додавання ходу в масив з можливими ходами;
* void release\_current\_move –звершення вибраного ходу;
* void extra\_check – перевірка на подвійну зв’язку;
* int extra\_figure – перевірка фігур для подвійної зв’язки.

За обробку одного із спеціальних ходів у шахах «castling», тобто рокірування, відповідають update\_castling та castle\_handler.

До допоміжними, базових, функцій перевірок коректності ходів належать такі функції як get\_mobility для визначення траекторії напрямку руху, is\_correct, призначена для перевірки на вихід за границі, X\_handler обробляє напрямки можливого пересування фігур по діагоналі та горзонталі / вертикалі, is\_direction\_correct, що перевіряє правильність напрямку та is\_moveable – правильність ходу.

За перевірку фігури відповідає is\_figure, is\_ally та is\_enemy, останні дві відповідають за визначення її кольору (союзна чи ворожа відповідно).

Одну з найважливіших ролей відіграють так звані функції-обробники атак, кожної клітинки дошки та фігури окремо. Функція position\_handler – збірна функція, що містить всі функції для обробки всіх станів та клітинок дошки, а desk\_handler необхідна для передачі координат кожної клітинки до інших оброблювальників. Так, наприклад, atack\_handler – визначає клітинки, на яких може бути загроза, pierce\_detector перевіряє клітинки для блокування ходів, а pierce\_block в свою чергу блокує ходи фігур під зв’язкою.

До функції для обробки союзних та ворожих фігур, тобто генерації ходів для них належать такі: ally\_line\_handler, ally\_figure\_handler та enemy\_line\_handler, enemy\_figure\_handler.

Для обробників фігур належать наступні:

* ally\_knight\_handler, enemy\_knight\_handler – функція обробки коня;
* ally\_king\_handler, enemy\_king\_handler – функція обробки короля;
* ally\_rook\_handler, enemy\_rook\_handler – функція обробки тури;
* ally\_bishop\_handler, enemy\_bishop\_handler – функція обробки слона;
* ally\_queen\_handler, enemy\_queen\_handler – функція обробки королеви;
* ally\_pawn\_handler, enemy\_pawn\_handler – функція обробки пішака;
* pawn\_kill\_move – функція перевірки ходів для побиття пішаком.

За перевірку на шах відповідають is\_check та is\_multicheck. Конвертація фігур з типу char в int та навпаки відбувається за допомогою get\_figure\_symbol та get\_figure\_number.

Функції, що відповідають за взаємодію з користувачем – це void update\_moves(), що відповідає за запис ходів, void Play() призначена для запуску гри, bool Turn (int number) відповідає за запис ходу та виклик оброблювальників та void StartGame(), що відповідає за зациклення гри, а void Setup() виводить меню гри.

За виведення інформації додаткової інформації, void Rules() для правил гри, а void game\_info() – результатів гри.

### 2.3. Використані типи даних

Як уже було сказано, клас є одним із найважливіших понять об’єктно-орієнтованого підходу в програмуванні. Під поняттям клас мають на увазі деяку «сущність», яка задає певний вид поведінки для об’єктів, тобто визначає певні правила (методи класу), за якими об'єкт може взаємодіяти з іншими об'єктами. Крім того, самі класи можуть бути в певних відносинах один між одним, наприклад наслідування.

Клас – це один із варіантів опису «сущності» і в теорії програмування його називають абстрактним типом даних або також користувацьким типом даних. Як уже було сказано, клас – це набір з внутрішньої структури – його полів (множина значень абстрактного типу або внутрішній стан об’єкту) та зовнішньої – методів (інтерфейсу). А екземпляр, відповідно до цієї ідеї, є значенням абстрактного типу даних, заданих класом. Класи надають багато можливостей програмістам, і ті можуть змінювати структуру класів, як дозволить фантазія та можливості мови програмування, поєднувати їх різними зв’язками (наслідуванням, асоціаціями, агрегацією, композицією), отже, використовувати їх різні типи: базові, похідні (або дочірні) абстрактні й віртуальні.

Отже можна сказати, що клас – це користувацький тип даних, що надає програмісту багато можливостей, дозволяє спрощувати написання програм та є основним поняттям об’єктно-орієнтованого програмуванням, до якого відноситься мова С++.

Одну з найважливіших ролей в програмі відіграють структури, без яких неможлива реалізація цієї гри. Вони дозволяють зберігати значення полів, їх станів і властивостей.

Структура – це сукупність змінних, об'єднаних одним ім'ям, що надає можливість спільного зберігання інформації. Оголошення структури призводить до утворення шаблону, використовуваного для створення її об'єктів. Змінні, що утворюють структуру, називаються її членами або полями. Зазвичай всі члени структури логічно пов'язані один з одним. Клас можна ж назвати похідною структури, тільки вдосконаленою та з більшими можливостями.

В коді програми використовується декілька структур, одна з них Cell – структура що зберігає стани клітинок, здійснений хід, координати для взяття на проході, масиви з ходами та для здійснення рокірування.

Головна структура гри Cell складається з таких змінних:

* char figure – поле що зберігає значення фігури;
* char mobility – змінна, що зберігає можливу траекторію руху;
* char color – колір фігури у клітинці;
* char attacked – флаг для виявлення клітинки атакованої ворожою фігурою;
* char special – флаг для позначення фігури під зв’язкою.

Move зберігає координати фігур складається з таких змінних:

* int fromX – початкова координата ходу по осі абсицс;
* int fromY – початкова координата по осі ординат;
* int toX – кінцева координата по по осі абсицс;
* int toY – кінцева координата по осі ординат;
* int figure – змінна для збереження номеру фігури.

Extra – структура для реалізації подвійної зв’язки та взяття на проході складається з таких полів:

* int X – координати по Х;
* int Y – координати по Y;
* int exist – існування фігури.

Окрім, структур важливу роль в цій реалізації гри відіграють enumerations. Enumeration – це список іменованих цілочисельних констант. В нашій програмі використовуються два перечислення: Figures та Angels.

Перечислення Figures містить у собі константи, якими позначаються фігури гри, пуста клітинка та границя: EMPTY, PAWN, QUEEN, KING, KNIGHT, BISHOP, ROOK, BARRIER.

Enum Angels – перечислення для визначення траекторії руху фігур. Вона містить в собі чотири змінні: ANGLE0, ANGLE45, ANGLE90, ANGLE135.

Кортеж (tuple) – це особливий тип даних, його можна назвати колекцією, пакунком елементів з фіксованим розміром. Принцип його роботи дуже подібний до pair, але tuple дозволяє працювати з великою кількістю аргументів, які він з'єднує під час компіляції. В якості елементів кортежа можуть бути змінні як одного, так і різних типів. Кортежі можна створювати, порівнювати, розпаковувати, деякі мови дозволяють їх змінювати.

Використання кортежів зручне тоді, коли необхідно прийняти і повернути декілька змінних або функцій. Багато мов програмування, такі як Python, Haskell, Erlang, мають вбудовані кортежі, проте не С++. У ній збереглися основні властивості, притаманні кортежам – їх можна створювати, порівнювати і розпаковувати, крім того значення можна змінювати. Кортежі описуються шаблонним класом std::tuple<> в заголовки, параметри спеціалізації якого – перерахування всіх типів змінних кортежу. Звернення до елементів кортежу відбувається через точку по константному індексу. Кортежі можна передавати в функції, де вони розгортаються в списки аргументів. Для конструювання об’єктів з існуючих змінних використовується стандартна функція make\_tuple(), а для доступу до елементів, розпакування кортежу використовується стандартний метод tie(). В цій реалізації шахів також використовується кортеж для роботи з масивом ходів [6].

В програмі також використано шаблон мови С++ – вектор, що є найбільш універсальним контейнерним класом. Вектор представляє з себе динамічний масив, розміри якого можуть змінюватися під час виконання програми відповідно до необхідності, що надає гнучкості програмі та запобігає витіку пам'яті [7]. Крім того, для роботи з векторами передбачено дуже багато методів, що полегшують реалізацію багатьох програм. Багато програмістів називають вектор найбільш ефективним способом визначення масивів

У програмі вектори використовуються в поєднані з іншим типом даних – кортежами: vector<tuple<int, int, int, int>> moves.

Він призначений для «вивантаження» згенерованих ходів, та установки вибраного користувачем варіанту. Його використання робить набагато простішим процес взаємодії користувача та функцій обробки ходів.

### 2.4. Опис роботи коду програми

В програмі передбачено родільне компілювання: головний модуль гри – main.cpp містить всі функції “логіки” програми та функцій для взаємодії з користувачем.

Крім того, створено хедерний файл «chess.h», в який занесено всі стандартні необхідні для роботи програми, та прототип функції void Rules(), в модуль chess.cpp винесено функцію, що забезпечує виведення основних правил гри в шахи.

В тілі головної функції коду int main() – відбувається створення об’єкту допоміжного класу ChessGame, в якому визначена лише одна функція StartGame. У ній відбувається створення об’єкту головного класу та виклик меню Setup. Головним завданням цієї функції та класу ChessGame є для зациклення гри.

Отже, при запуску функції Setup відбувається виведення стартової панелі гри, де надається можливість вибрати одну з доступних опцій:

* play;
* rules;
* exit.

Залежно від обраного варіанту відбувається виведення основних правил шах, початок гри чи вихід з програми, якщо ж введено неправильний варіант, то відбувається виведення повідомлення про це та прохання спробувати ввести якусь опцію ще раз.

Після введення користувачем «rules» відбувається виклик функції Rules(), яка виводить користувачеві головні правила шахів, стандартний вигляд початкової дошки з порядком фігур на ній, правила ходів для кожної фігури та короткі відомості про головні поняття: «шах», «мат», «Promotion», «En passant» та «Castling». Відбувається повторний запит.

При введені «exit» – відбувається вихід з програми за допомогою стандартної бібліотечної функції exit.

Коли користувач ввів «play» відбувається початок гри за допомогою виклику функції Play(). Ця функція забезпечує взаємодію користувача й програми: тут відбувається виведення дошки функцією output\_desk та можливих варіантів ходів, запит користувача обрати один з них.

Виклик функції Turn забезпечує запам’ятовування зробленого ходу та оновлення варіантів, зміну гравця. Всі дії Play відбуваються до тих пір, поки є можливі ходи, в іншому випадку – гра завершується, game\_info виводить час гри та кількість ходів, загальну й зроблену кожною стороною, потім гра оновлюється і відбувається новий запит користувача.

Проте логіка гри реалізована через інші функції, що безпосередньо не взаємодіють з користувачем. В основі алгоритму програми – генерації можливих ходів на основі тільки зробленого.

Виходячи з цього в конструкторі головного класу Chess за допомогою функцій init\_move\_table, unpack\_position(starting), position\_handler та update\_moves відбувається ініціалізація динамічного масиву з варіантами ходів, встановлення початкового стану дошки та виклик функції генерації можливих коректних варіантів на основі ходу за замовчуванням, який для не порушення правил гри вибраний як «e8 -> e8». Така комбінація вибрана навмисне, вона означає, що хід не відбувається, адже переміщення на поточну позицію – неможливе, проте це дає нам можливість запустити механізм обробки. Кожен зроблений користувачем хід впливає на позицію всіх фігур та стан дошки в цілому, тому необхідно проводити аналіз відштовхуючись від нього.

На цьому і побудована логіка програми: вибраний хід запам’ятовується як current\_move, після чого йде його реалізація та обробка дошки. Головною, збірною функцією можна назвати position\_handler(), адже вона забезпечує виклик усіх найголовніших функцій. Отже, спочатку через функцію clear\_move\_table() відбувається очистка масививу, які зберігає дані про кількість можливих ходів, та через clear\_desk\_state() – оновлення станів клітинок дошки.

Функція release\_current\_move безумовно належить до найважливіших функцій коду. В ній реалізована зміну кольору, а отже гравця, викликаються функції перевірки рокіровки, відбуваються перевірки для окремих “спеціальних” випадків у шахах: взяття на проході та рокіровка, трансформація пішака, якщо хід задовольняє одну з умов для конкретного випадку, то відбувається реалізація цього ходу, якщо ж ні, то відбувається просте переміщення самої фігури.

Наступним етапом є виклик функцій обробки (handlers) – desk\_handler приймає або функцію enemy\_figure\_handler, або ally\_figure\_handler та передає в них координати кожної клітинки шахової дошки. Для прикладу, ally\_figure\_handler (int x, int y) отримує як аргументи координати клітинки та перевіряє їх на наявність союзної фігури та загрози королю. Якщо точка пройшла перевірку, то відбувається підбір відповідної функції оброки для кожної союзної фігури (ally\_pawn\_handler, ally\_queen\_handler, ally\_king\_handler, ally\_knight\_handler, ally\_bishop\_handler, ally\_rook\_handler) та вже обробка клітинки на можливість ходу фігурою. Робота enemy\_figure\_handler – аналогічна. Викликаючи castle\_handler ми перевіряємо на можливість виконання рокірування.

Як не дивно, але найважчою фігурою є пішак, саме тому в грі передбачено багато функцій пов’язаних з ним, а саме: extra\_check, pawn\_kill\_move, enemy\_pawn\_handler, ally\_pawn\_handler.

Важливими безумовно є функції pierce\_detector, enemy\_line\_handler, pierce\_block та atack\_handler – вони всі пов’язані з визначенням та позначенням атакованих клітинок. Функція pierce\_detector працює з індикаторами для визначенням різних випадків: шаху, подвійної зв’язки, enemy\_line\_handler – з даними, отриманими в попередньо описаній функції та визначає точки “під зв’язкою”, pierce\_block – забезпечує блокування ходів, через обмеження траекторії руху. atack\_handler – фунція для визначення клітинок як тих, на яких можлива атака, та шаху.

До базисних функцій, які відповідно забезпечують основні перевірки та без яких неможлива коректна робота, належать get\_mobility, is\_check, is\_multicheck, is\_correct, is\_figure, is\_ally, is\_enemy, is\_direction\_correct, is\_moveable та X\_handler.

### РОЗДІЛ 3. КЕРІВНИЦТВО КОРИСТУВАЧА

Запуск програми починається з виклику меню (рисунок 3.1). Користувачу виводиться привітання «Welcome to Chess game!» та запит обрати одну з доступних в грі опцій: «play», «rules» та «exit», назви цих опцій обрано лаконічними, короткими та зрозумілими для кожного.

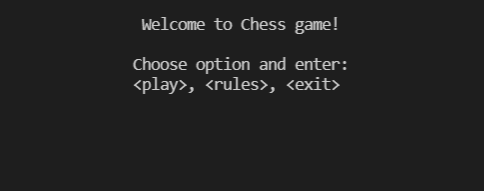


Рисунок 3.1. Головний інтерфейс програми

Меню програми передбачене таким чином, що для успішного вибору опції користувачеві необхідно повністю вписати назву дії, яку він хоче, щоб програма виконала, наприклад, правильним варіантом буде така форма введення: «rules». Якщо ж користувач ввів неправильний варіант опції, то йому виведеться повідомлення про це, та після цього програма виведе запит на повторний ввід (рисунок 3.2). Він виглядає ідентично з початковим меню, окрім повідомлення про некоректність: «Your answer is incorrect.»

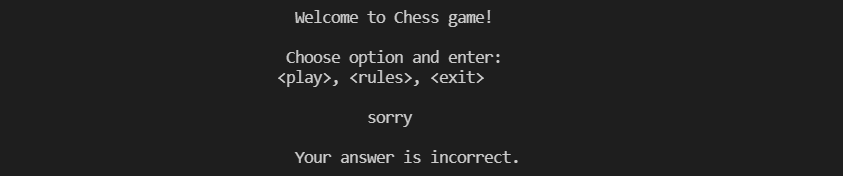


Рисунок 3.2. Повідомлення про помилкове ввдення

Коли користувач ввів варіант «rules» програма очистить вікно та виведе всі основні правила, що необхідні для гри в шахи. Тут передбачені основні відомості про гру: кількість гравців, їх назви, кількість фігур для кожної сторони та їх позначення в даній програмі (рисунок 3.3).

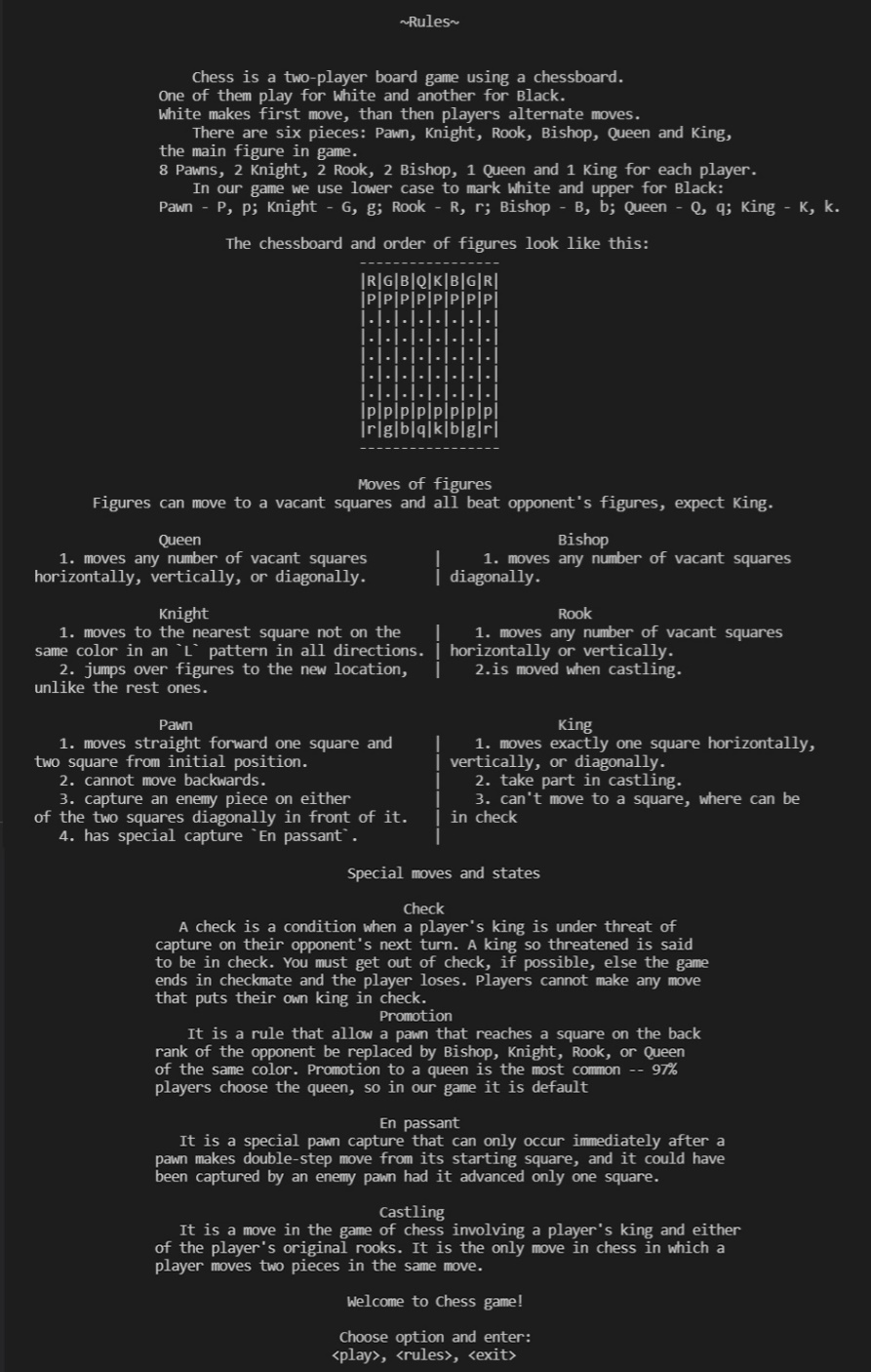


Рисунок 3.3. Основні відомості про гру

Для кращого розуміння правил та наочного представлення наведено приклад дошки з фігурами на початкових позиціях (рисунок 3.4) відповідно до особливостей позначення в програмі.

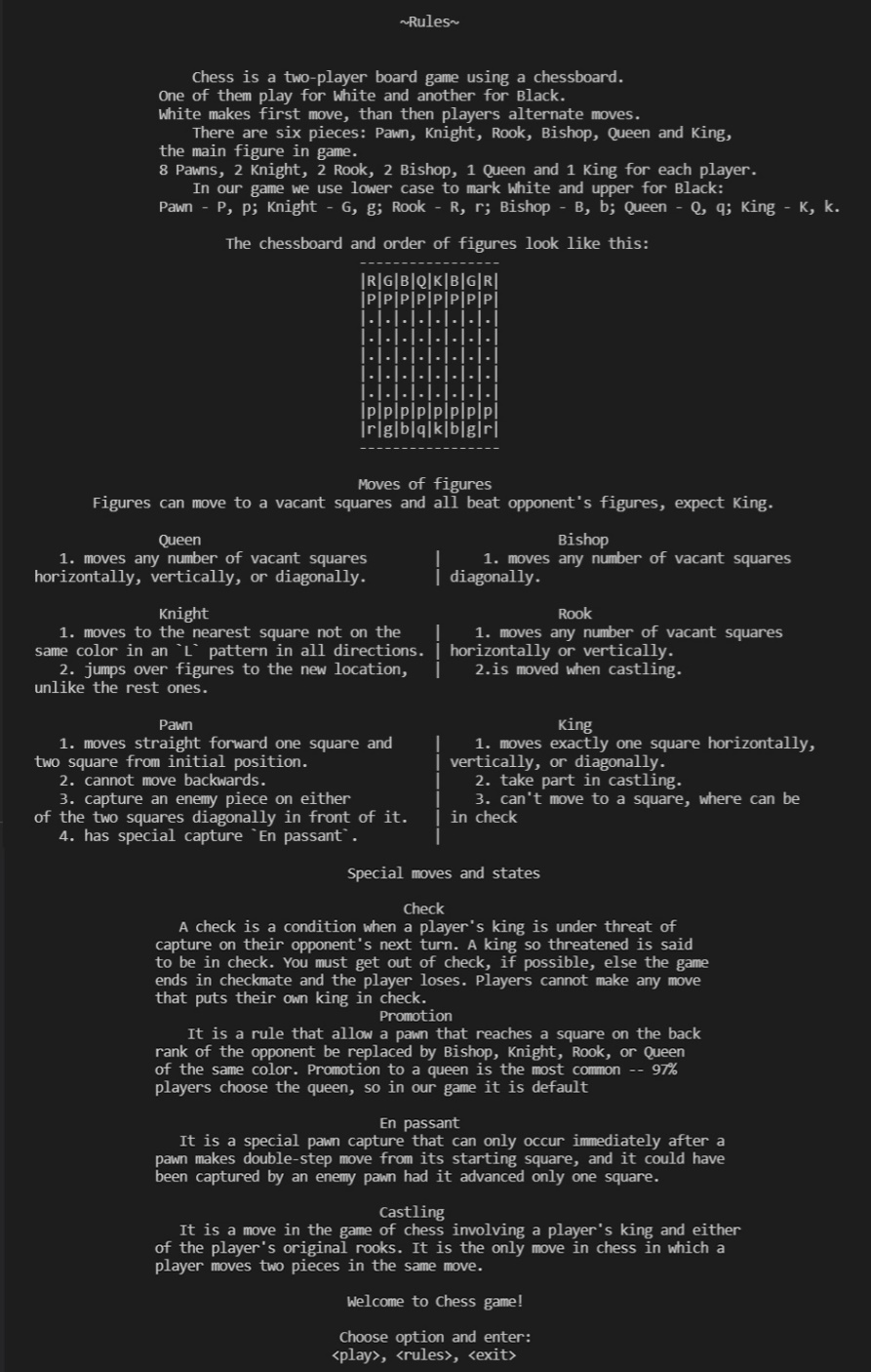


Рисунок 3.4. Приклад початкової дошки

Оскільки правила пересування фігур є одним із найважливіших моментів у грі в шахи, то для них виділяється окреме місце в цьому вікні. Тут виписані всі головні правила ходів для кожної фігури (рисунок 3.5).

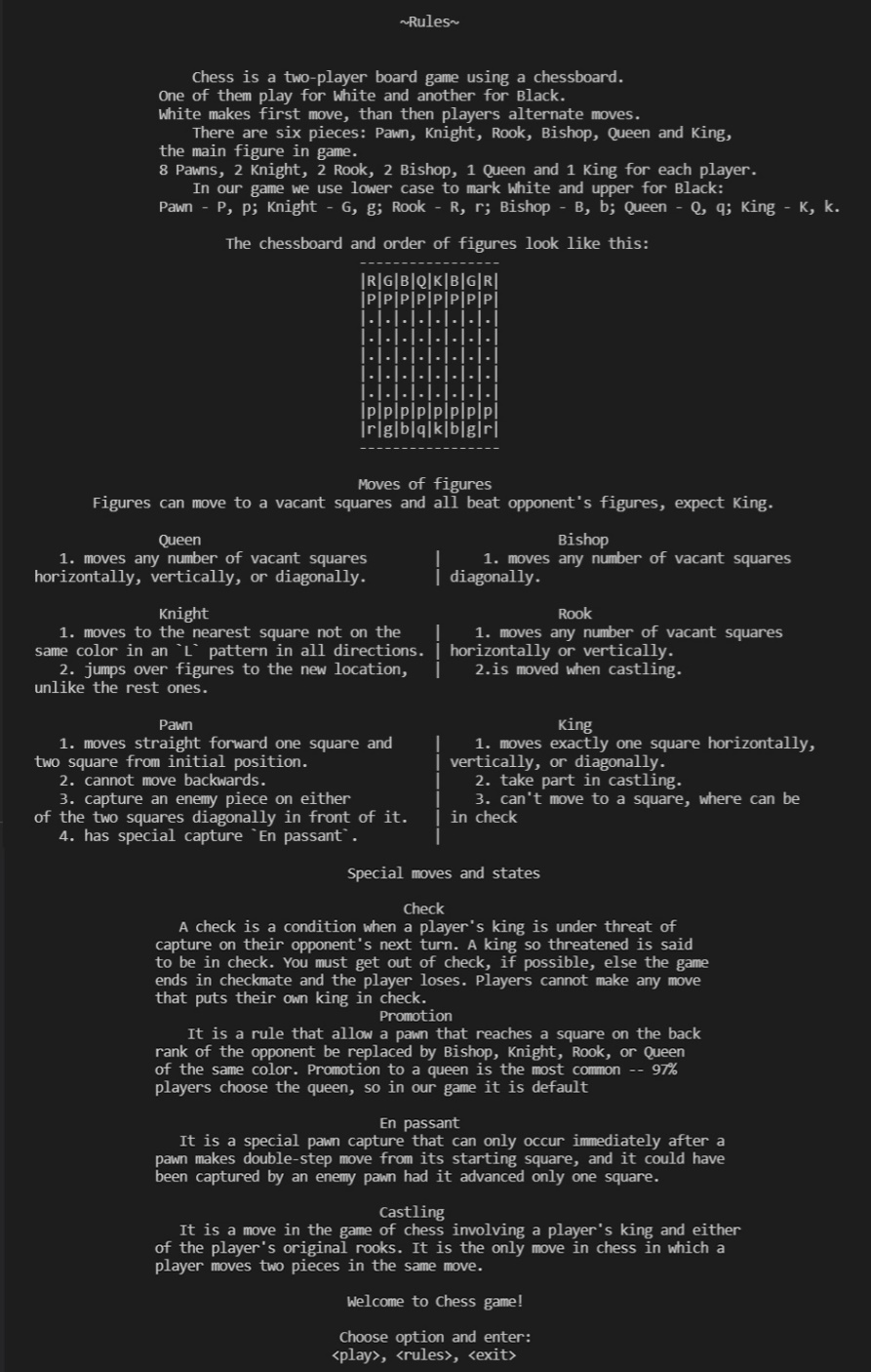


Рисунок 3.5. Правила ходів для кожної фігури

Також виписані найважливіші спеціальні стани і ходи в шахах (рисунок 3.6): «check», «chechmate», «Promotion» (трансформація пішака), «En passant» (взяття на проході) та «Castling» (рокіровка).

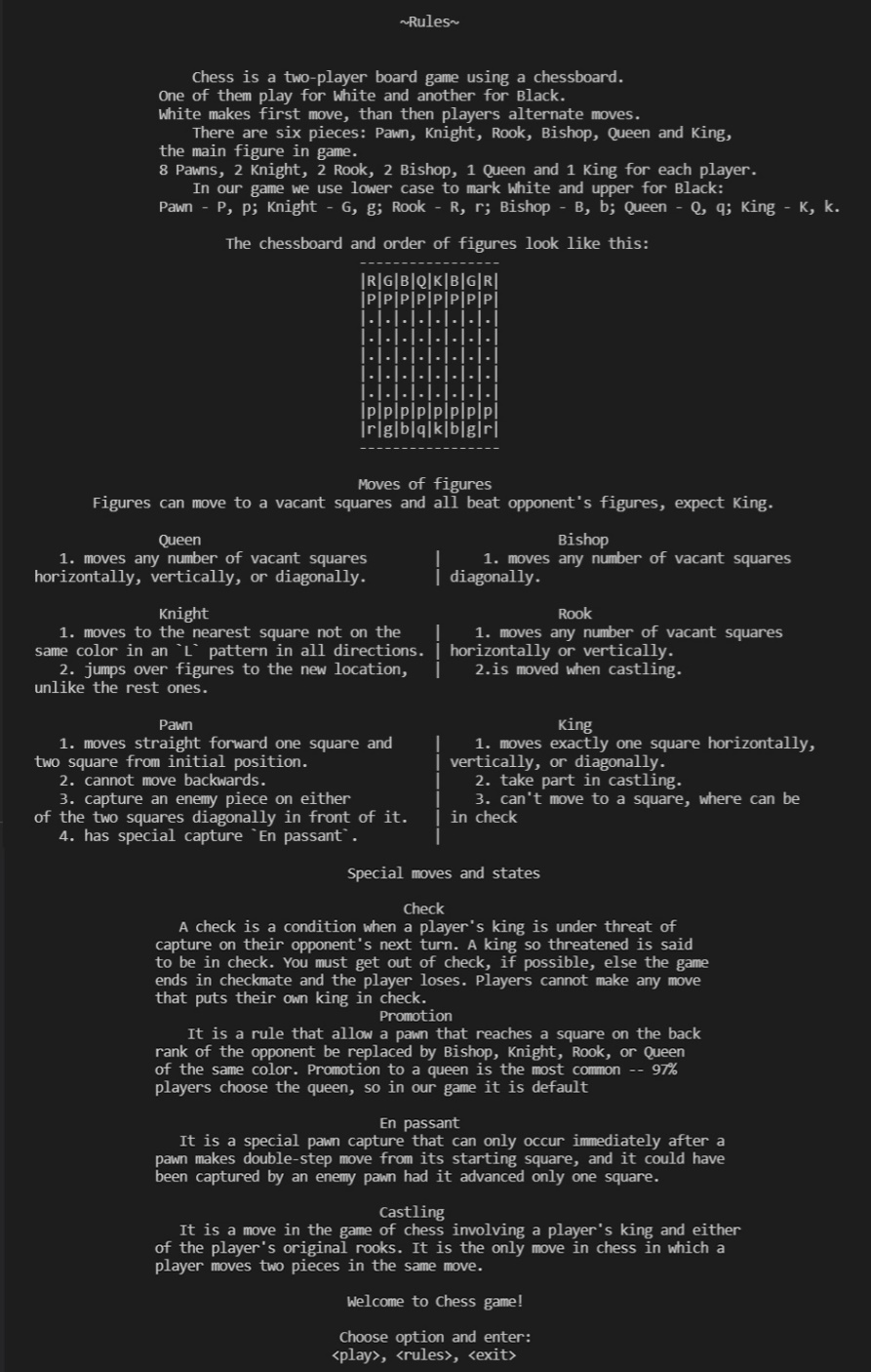


Рисунок 3.6. Опис спеціальних станів і ходів в шахах

Вибір опції «play» забезпечує початок гри. Тож користувач зможе побачити перед собою дошку з всіма фігурами, що стоять на стартових позиціях (рисунок 3.7). Дошка виводиться таким чином, що знизу розташовані білі фігури, а зверху – чорні.

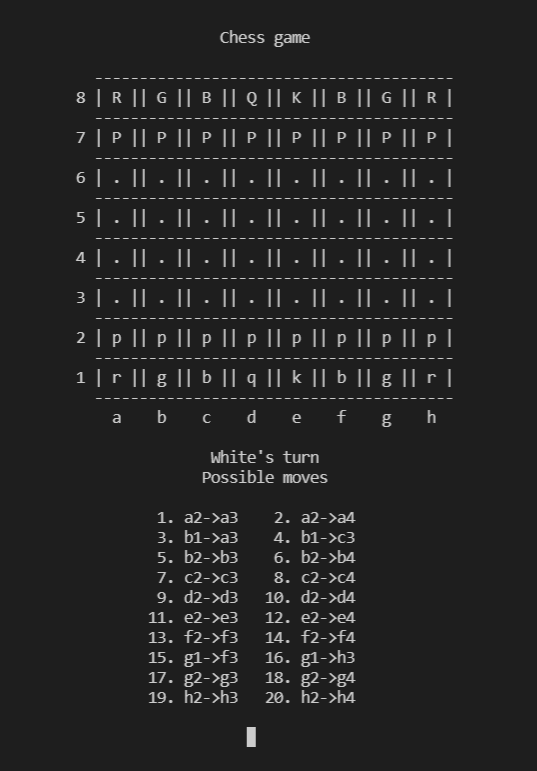


Рисунок 3.7. Початковий стан дошки

Також з початком партії користувачеві стає доступною панель з усіма можливими на цей момент ходами (рисунок 3.8). Вони подаються у вигляді «звідки - куди», тобто позначення клітинки, з якої ходять і позначення клітинки, на яку ходять у стандартному вигляді: «b1 -> c3». Можливі варіанти є пронумерованими, для зручнішого вибору. Поле для введення є доступним одразу після списку можливих ходів та розташоване в центрі.

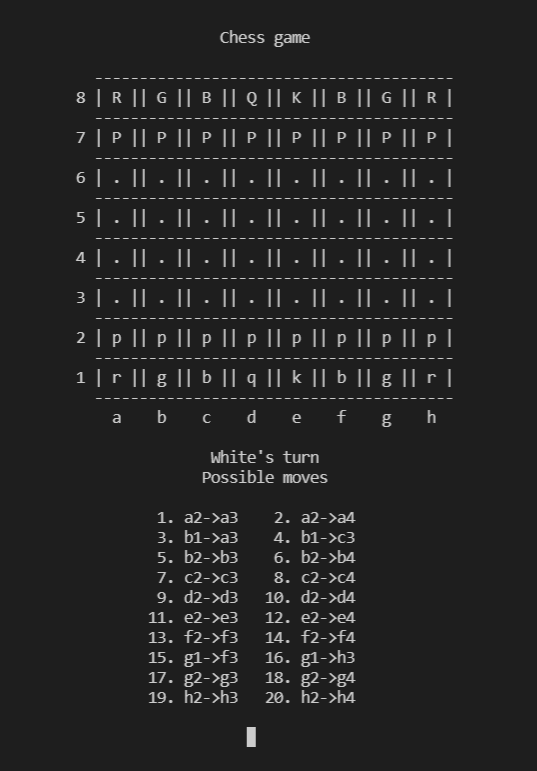


Рисунок 3.8. Панель доступних ходів

Для того, щоб здійснити хід, необхідно ввести номер одного з доступних, програма зчитає та здійснить обраний хід. Після цього вікно програми очиститься та користувач зможе побачити вже оновлену дошку та згенеровані відповідно до стану фігур нові варіанти ходів.

Як видно з рисунку 3.9, програма оцінює хід, аналізує стан дошки та позиції фігур і на основі цих даних генерує нові можливі варіанти переміщень. Користувачу це стає зрозуміло з оновленої панелі доступних ходів, яку він може побачити під оновленою дошкою.

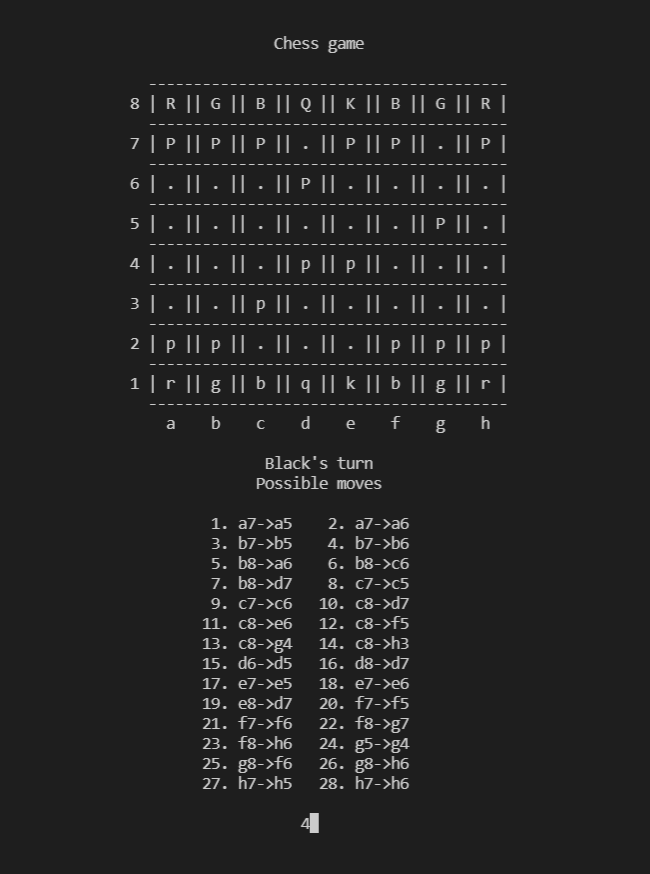


Рисунок 3.9. Вигляд дошки після звершення ходу

Якщо користувач хоче вийти з програми, йому необхідно ввести «exit». Виведеться повідомлення з подякую (рисунок 3.10), після чого програма закінчує свою роботу.

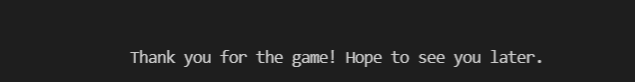


Рисунок 3.10. Повідомлення при завершенні програми.

Якщо один з користувачів захоче вийти під час гри, що означає «задатися» в шахах, то йому необхідно ввести слово замість номеру, наприклад, як і в меню, «exit». Після цього йому буде виведена інформація про його програш та дані партії: витрачений час та кількість ходів (рисунок 3.11).



Рисунок 3.11. Вихід з програми посеред гри

Проаналізувавши інтерфейс програми, можемо сказати, що він досить простий і зручний. Є все необхідне для гри, що краще, є згенеровані варіанти можливих ходів, що полегшує гру для користувача. Гравцям не потрібно вводити повну назву ходу «e2 -> e4», а лише номер цього ходу.

### ВИСНОВКИ

Під час виконання курсової роботи здобуто знання з теми «комп’ютерні шахи», закріплено та вдосконалення вміння працювати з різними типами даних: особлива увага приділялась роботі з класами та обє’ктами, як основі об’єктно-орієнтованого програмування, структурам та масивам. Вдалось покращити вміння розробляти власні алгоритми аналізу та створювати програми, що базуються на них та були б здатні генерувати можливі коректні варіанти.

В результаті роботи було виконано поставлене на початку завдання та розроблено алгоритм інтелектуального аналізу даних та із врахуванням парадигм об’єктно-орієнтованого програмування створено гру «Шахи» зі зручним та зрозумілим інтерфейсом, що за допомогою передбаченого меню дозволяє користувачеві зробити певний вибір щодо розвитку дій та, відповідно до завдання, парно зіграти бажану кількість партій.

Для виконання поставленого завдання було обрано інтегроване середовище розробки Visual Studio Code як найбільш зручне та підходяще для цієї цілі. За допомогою інструментів VS Code створено консольну гру, просту у використанні та інтуїтивно зрозумілу для користувача.

Крім того, досліджено та вивчено нову тему «User-defined header files» та створено власний хедерний файл chess.h та таким чином оформлено розроблену програму у декількох файлах.

Отже, під час написання курсової роботи на тему «Шахи» вдосканалено знання розробки мовою пограмування С++, повторено та використано під час написання коду вивчений матеріал з курсу «Основ програмування – 2», вдосконалено навички роботи із середовищем розробки Visual Studio Code.

### СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гик Е. Я. Легенда про мудреця і електронні обчислюванні машини / Математика на шаховій дошці [Електронний ресурс] / Е. Я. Гик – Режим доступу до ресурсу: https://forany.xyz/a-16?pg=4.
2. Герберт Шилдт “Повний довідник по C++”[Електронний ресурс] – Режим доступу https://codernet.ru/books/c\_plus/polnyj\_spravochnik\_po\_c\_4-e\_izdanie/
3. Симанцев М. О. Комп’ютерне моделювання ігрових ситуацій у шахах / Михайло Олександрович Симанцев // ХНУМГ. – 2012. – Режим доступу до ресурсу: http://eprints.kname.edu.ua/25406/1/2009
4. Корнілов “Програмування шахів та інших логічних ігор” [Електронний ресурс] – Режим доступу <https://whychess.ru/560rpogrammirovanie-shahmat-i-logicheskih_igr.html>
5. Клас (ООП) [Електронний ресурс] – Режим доступу до ресурсу: https://dic.academic.ru/dic.nsf/ruwiki/969035.
6. Кортежі в мовах програмування. Частина 1. [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: <https://habr.com/ru/post/276871/>.
7. Введення в std::vector [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://ravesli.com/urok-95-vvedenie-v-std-vector-vektory/.