ЗМІСТ

[ВСТУП](#_30j0zll) 5

[1 ПОСТАНОВКА ЗАДАЧІ](#_1fob9te) 6

[2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ](#_3znysh7) 7

[2.1 Теоретичні відомості](#_2et92p0) 7

[2.2 Діаграма прецедентів](#_tyjcwt) 10

[2.3 Сценарій роботи програми](#_3dy6vkm) 11

2.3.1 Алгоритм розташування кораблів на полі 11

[2.3.2 Алгоритм](#_88wzdiavzqj5) потоплення корабля 12

2.3.3 Алгоритм ігрового меню 13

[3 ОПИС АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ](#_1ksv4uv) 14

[4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ](#_ctvbp151advs) 16

[5 РЕЗУЛЬТАТИ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ](#_4i7ojhp) 22

[5.1 План тестування](#_2xcytpi) 22

[5.2 Приклади тестування](#_1ci93xb) 24

[6 ІНСТРУКЦІЯ КОРИСТУВАЧА](#_nojkzgm9s52l) 36

[6.1 Робота з програмою](#_2bn6wsx) 36

[6.2 Формат вхідних та вихідних даних](#_mqwuti35frk5) 43

[6.3 Системні вимоги](#_3as4poj) 44

[ВИСНОВОК](#_1pxezwc) 45

[ПЕРЕЛІК ПОСИЛАНЬ](#_qiihl1nb9fj9) 46

[ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ](#_2p2csry) 47

[ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ](#_rhpuvg29w61x) 50

# 

# ВСТУП

Microsoft .NET (читається дот-нет) - програмна технологія, запропонована фірмою Microsoft як платформа для створення як звичайних програм, так і веб-застосунків. Багато в чому є продовженням ідей та принципів, покладених в технологію Java. Однією з ідей .NET є сумісність служб, написаних різними мовами. Хоча ця можливість рекламується Microsoft як перевага .NET, платформа Java має таку саму можливість.

C# (вимовляється Сі-шарп) — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до С++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов С++, Object Pascal, Модула і Smalltalk — С#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

Завдяки своїй здатності швидко пристосовуватися до постійно мінливих потребам в області програмування C#, як і раніше залишається живим і новаторським мовою. А отже, він являє собою один з найбільш ефективних і багатих своїми можливостями мов в сучасному програмуванні.

Для вивчення можливостей мови С# була реалізована задача з проектування та створення гри «Морський бій».

# 1 ПОСТАНОВКА ЗАДАЧІ

Спроектувати та створити консольне застосування гри «Морський бій» на мові C#.

Даними для роботи із застосунком є ввід користувача з клавіатури.

Програма повинна фільтрувати та перевіряти якість отриманих даних. Результатом виконання функціоналу програми має бути консольний вивід.

План реалізації поставленої задачі:

1. Створення функцій для вводу даних користувача з подальшою валідацією.
2. Створити прототипи ігрових полів і функцій курування цими полями.
3. Створити форматований консольний вивід для операцій які його потребують.
4. Спроектувати функціонал гравця.
5. Реалізувати ініціалізацію об’єктів і налаштувати ігровий цикл.
6. Додати можливість збереження результату гри в таблицю рекордів і можливість серіалізації цих даних в XML-файл.
7. Реалізувати меню для користувацького керування процесами.

# 2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 2.1 Теоретичні відомості

«Морський бій» — гра для двох учасників, у якій гравці по черзі називають координати на невідомій їм карті суперника. Якщо у суперника в тому місці є корабель (координати зайняті), то корабель або його частина «тоне», а той, хто влучив, здобуває право зробити ще один хід. Мета гравця — першим потопити всі кораблі супротивника.

Гру вперше випустила у вигляді настільної гри компанія Milton Bradley Company 1931 року.

Ігрове поле — квадрат 10 × 10 кожного гравця, на якому розміщений флот кораблів.

Горизонталі зазвичай нумерують зверху вниз, а вертикалі позначають буквами зліва направо. При цьому використовують літери української (латинської) абетки. Іноді використовується слово «республіка», тому що це 10 буквене слово, де жодна буква не повторюється. Оскільки існують різні варіанти нанесення системи координат, то про це краще заздалегідь домовитися.

Загалом є десять кораблів:

Інший варіант назви — трубні (напр. двотрубний).

* 1 корабель — ряд із 4 клітин («лінкор», або «чотирипалубний»)
* 2 кораблі — ряд із 3 клітин («крейсери», або «трипалубні»)
* 3 кораблі — ряд із 2 клітин («есмінці», або «двопалубні»)
* 4 кораблі — 1 клітина («підводні човни», або «однопалубні»)

При розміщенні кораблі не можуть торкатися один одного кутами (трапляються, однак, варіанти, коли це не заборонено).

Рідко (за домовленістю), але трапляються варіанти гри, коли кораблі можуть бути у вигляді квадрата («чотирипалубні») або літерою «Г» («три-» та «чотирипалубні»).

Поруч зі «своїм» квадратом креслять «чужий» такого самого розміру, лише порожній. Це ділянка моря, де ходять кораблі супротивника.

При влученні в корабель супротивника — на чужому полі ставлять хрестик. Той, хто влучив, стріляє ще раз.

На перший погляд, може здатись, що гра «Морський бій» носить чисто імовірнісний характер, оскільки обстріл гравці ведуть, не знаючи розташування кораблів противника. Але можна помітити, що існують стратегії розстановки кораблів, які зменшують імовірність потрапляння в останній одноклітинний корабель. Наприклад, можна розташувати всі кораблі таким чином, щоб він займав найменшу площу на ігровому полі, а один або два кораблі виставляють на площі, що залишилась на значній відстані один від одного. Пошук кораблів також можна проводити, дотримуючись певних правил, які дозволяють найбільш швидко виявити на початку гри багатоклітинні кораблі, а потім на останньому просторі шукати одноклітинні кораблі.

## 2.2 Діаграма прецедентів

Для візуалізації узагальнення опису предметної області програмного забезпечення, представлена діаграма прецедентів системи (рис 2.1), яка також має назву “use case diagram” – діаграма варіантів використання програми. На діаграмі зображено відношення між користувачем та прецедентами (діями, які вони можуть зробити , використовуючи програмне забезпечення).

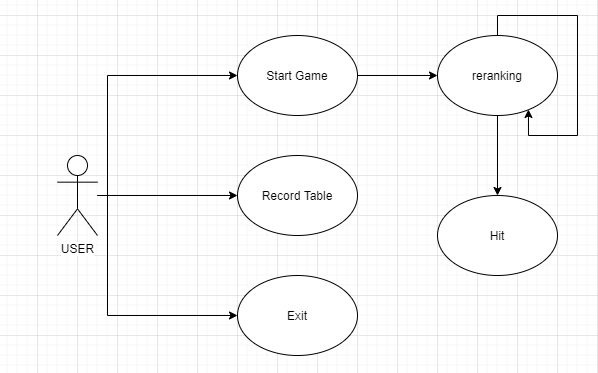


Рисунок 2.1 – Діаграма прецедентів

На діаграмі відображені всі дії, що може зробити гравець в спроектованому застосунку. А саме він може вийти, переглянути таблицю рекордів та розпочати гру. Також до початку гри, гравець може спробувати більш вигідне положення кораблів на карті.

## 2.3 Сценарій роботи програми

Перелік всіх основних змінних та їхнє призначення наведено в таблиці 2.1.

Таблиця 2.1 – Основні змінні та їхні призначення

|  |  |
| --- | --- |
| Змінна | Призначення |
| GArea | Матриця положень на ігровому полі |
| points | Змінна, що є в кожного гравця, відображає кількість підбитих гравцем кораблів |
| UserStep | Відображає чий хід на даний момент |

### 

### 2.3.1 Алгоритм розташування кораблів на полі

1. ПОЧАТОК

2. Ініціалізація змінних координат та орієнтація.

3. Перевірка валідності розташування.

4. Перевірити чи знаходиться в потрібній клітинці шашка

4.1. ЯКЩО за згенерованими координатами або за координатами сусідніх клітинок є корабель або згенеровані координати лежать за межами поля ТО перейти до кроку 2.

5. КІНЕЦЬ

### [2.3.2 Алгоритм](#_88wzdiavzqj5) потоплення корабля

1. ПОЧАТОК

2. ЯКЩО передана точка це підбитий корабель ТО перевірити клітинки навколо.

2.1 Якщо підбитих кораблів навколо немає ТО перейти до кроку 2.1.1

2.1.1 Всі клітинки навколо позначити обстріляними якщо їх координати не виходять за межі ігрового поля.

2.2 ЯКЩО клітинка це підбитий корабель і це не попередній підбитий корабель ТО перейти з цими координатами до кроку 2 маючи ці координати як попередні.

3. КІНЕЦЬ

### 2.3.3 Алгоритм ведення ігрового процесу

1. ПОЧАТОК

2. ЯКЩО користувач ввів в меню «2» ТО

2.1 Десеріалізувати records.xml та виконати форматований вивід даних про рекорди.

3. ЯКЩО користувач ввів в меню «3» ТО перейти до кроку 5.

4. ЯКЩО користувач ввів в меню «1» ТО.

4.1 Ініціалізувати змінні для гри.

4.2 Вивід розташування кораблів.

4.3 ЯКЩО користувач захотів змінити розташування ТО перейти до пункту 4.1.

4.4 Розпочати ігровий процес, а по закінченню внести результат гри до списку рекордів та перейти до пункту 1.

5. КІНЕЦЬ.

### 

# 3 ОПИС АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ

На рисунку 3.1 представлена діаграма класів, яка демонструє основні зв’язки між компонентами програми.

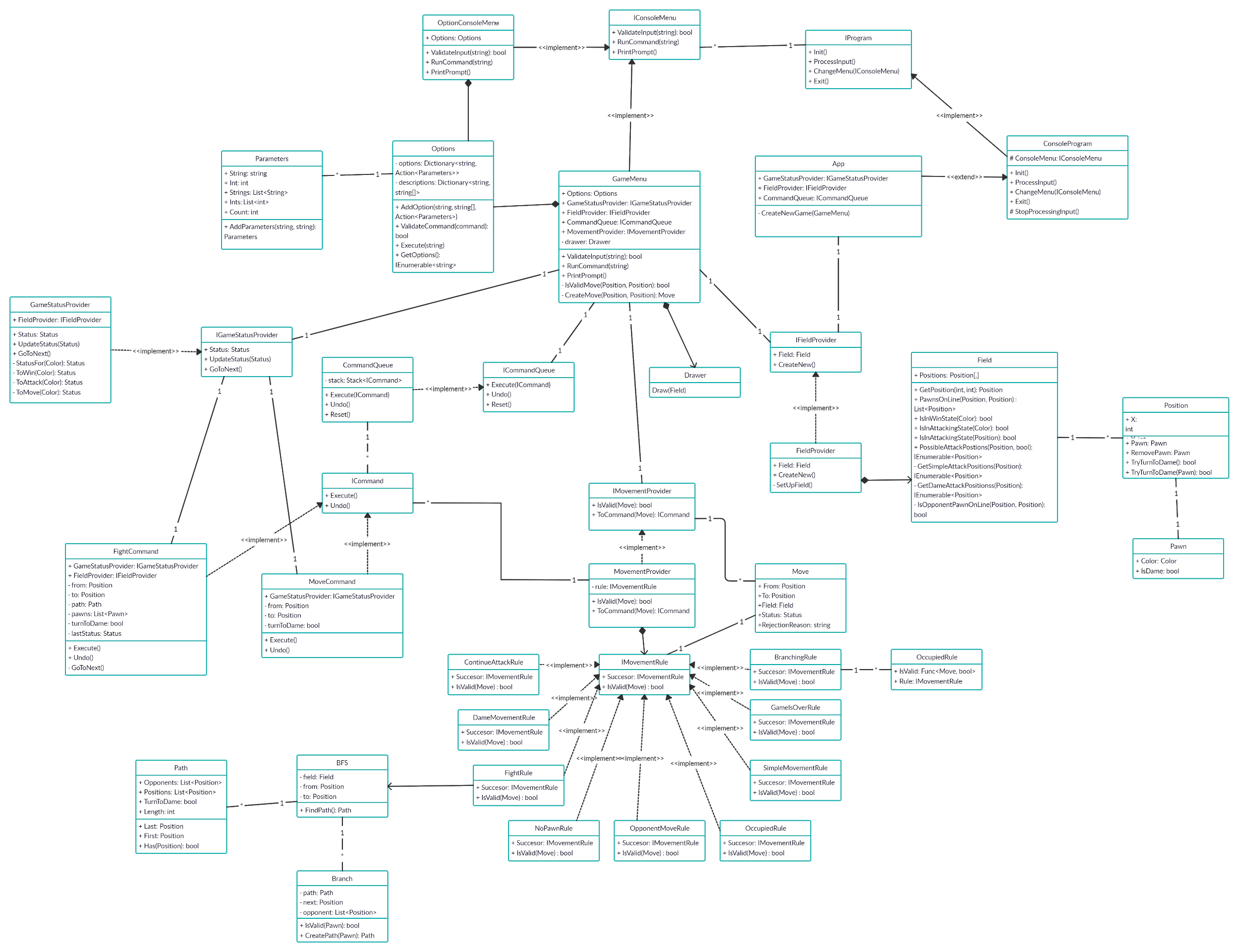


Рисунок 3.1 – Діаграма класів

# 

# 4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Користувацькі функції описані в таблиці 4.1.1

Таблиця 4.1.1 – Перелік користувацьких функцій

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Назва классу | Назва  методу | Призначення  методу | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 1 | Player | Win | Визначає чи переміг гравець | Відсутні | Position позиція за координатами x y | Player.cs |
| 2 | Player | strike | Передає дані про координати для пострілу | Відсутні | Dot | Player.cs |
| 3 | Player | coord | Отримує координату і перевіряє її валідність | Відсутні | Coord | Player.cs |
| 4 | Bot | Win | Визначає чи переміг гравець | Відсутні | Bool  true якщо повинен атакувати, інакше false | Bot.cs |
| 5 | Bot | Hit | Визначає чи можлива атака з даної позиції | Відсутні | Bool  true якщо може атакувати, інакше false | Bot.cs |
| 6 | Area | isAvalLocation | Визначає всі можливі атаки з даної позиції | Dot | IEnumerable<Position> всі можливі позиції атаки з даної | Area.cs |
| 7 | Area | isAval | Розпочинає роботу програми | Dot | Відсутні | Area.cs |
| 8 | Area | isAval | Зчитує введені команди користувачем | Dot, param | Відсутні | Area.cs |
| 9 | Area | ranking | Змінює поточне меню програми | Відсутні | Відсутні | Area.cs |
| 10 | Area | Ranking1 | Завершує роботу програми | Відсутні | Відсутні | Area.cs |
| 11 | Area | Ranking2 | Визначає чи коректна команда | Відсутні | Bool  true якщо команда коректна, інакше false | Area.cs |
| 12 | Area | Ranking3 | Виконує команду | Відсутні | Відсутні | Area.cs |
| 13 | Area | Ranking4 | Виводить підказку користувачу | Відсутні | Відсутні | Area.cs |
| 14 | Area | sunkAval | Визначає чи коректна команда | Dot | Bool  true якщо команда коректна, інакше false | Area.cs |
| 15 | Area | Sunken | Виконує команду | Dot | Відсутні | Area.cs |
| 16 | Area | Around | Виводить підказку користувачу | From and curent | Відсутні | Area.cs |
| 17 | Area | inRange | Друкує ігрове поле до консолі | Dot | Відсутні | Area.cs |
| 18 | Area | Hit | Прибирає шашку з позиції | Dot | Pawn шашка, що знаходилась на позиції | Area.cs |
| 19 | Area | Area | Перетворює шашку в дамку | Відсутні | Відсутні | Area.cs |
| 20 | ConsoleInput | MenuValidInput | Перетворює шашку в дамку | Відсутні | Bool  true якщо вдалося перетворити, інакше false | ConsoleInput.cs |
| 21 | ConsoleInput | Input | Переміщає шашку в вказану позицію | Відсутні | Відсутні | ConsoleInput.cs |
| 22 | ConsoleOutput | PrintUserArea | Відміняє переміщення шашки | Відсутні | Відсутні | ConsoleOutput.cs |
| 23 | ConsoleOutput | Menu | Атакую вказаною шашкою шашки супротивника | Відсутні | Відсутні | ConsoleOutput.cs |
| 24 | FileManager | Pack | Відміняє атаку шашкою | Відсутні | Відсутні | FileManager.cs |
| 25 | FileManager | Unpack | Виконує команду | Відсутні | Відсутні | FileManager.cs |
| 26 | FileManager | GetInstance | Відміняє останню команду | Відсутні | Відсутні | FileManager.cs |
| 27 | Game | Reranking | Очищає історію команд | Відсутні | Відсутні | Game.cs |
| 28 | Game | Initial | Перетворює рядок в позицію | Відсутні | Відсутні | Game.cs |
| 29 | Game | Start | Перевіряє чи даний хід відповідає всім правилам | Відсутні | Відсутні | Game.cs |
| 30 | Menu | Open | Перетворює хід в команду | Відсутні | Відсутні | Menu.cs |

# 5 РЕЗУЛЬТАТИ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 5.1 План тестування

Головною метою тестування є визначення чи коректно програма опрацьовує потрібні дані та повідомляє про помилки під час роботи. Основна частина тестування була проведена з метою підтвердження очікуваної поведінки програми на дії користувача. Важливо також перевірити чи графічний інтерфейс правильно відображає дії користувача. Для перевірки коректності програми були проведені наступні тести:

а) Тестування коректного опрацювання переміщення шашок.

1) Тестування коректного переміщення шашки.

2) Тестування переміщення шашки з порожнього поля.

3) Тестування переміщення шашки на зайняте поле.

4) Тестування переміщення шашки в неправильному напрямку.

5) Тестування переміщення шашки на неправильну кількість клітинок.

6) Тестування переміщення шашки під час ходу супротивника.

7) Тестування коректного переміщення дамки.

б) Тестування коректного опрацювання атаки шашок.

1) Тестування коректної атаки шашкою.

2) Тестування атаки шашкою з порожнього поля.

3) Тестування атаки шашкою на зайняте поле.

4) Тестування атаки шашкою в неправильному напрямку.

5) Тестування атаки шашкою без шашки опонента на шляху.

6) Тестування атаки шашкою під час ходу супротивника.

7) Тестування коректної атаки дамкою.

8) Тестування коректної переміщення шашки під час атаки.

г) Тестування коректного опрацювання вводу користувача.

1) Тестування коректної опрацювання команди game.

2) Тестування коректної опрацювання команди exit.

3) Тестування коректної опрацювання команди back.

4) Тестування коректної опрацювання команди restart.

5) Тестування коректної опрацювання команди undo.

6) Тестування коректної опрацювання команди move.

7) Тестування команди move при введенні позиції в неправильному форматі.

8) Тестування при введенні невідомої команди.

## 5.2 Приклади тестування

Результати тестів описані в таблицях

Таблиця 5.1 ‑ Приклад роботи програми при переміщенні шашки.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність переміщення шашки |
| Початковий стан програми | Відкрите ігрове меню |
| Вхідні дані | Дані з пристрою вводу. Значення клітинки, з якої перемістити шашку, значення клітинки, на яку перемістити |
| Схема проведення тесту | Обробка вводу користувача та виконання переміщення шашки |
| Очікуваний результат | Шашка переміститься на вказану позицію |
| Стан програми після проведення випробувань | Шашка знаходиться в заданій позиції |

Таблиця 5.2 ‑ Приклад роботи програми при переміщенні шашки з порожнього поля.

|  |  |
| --- | --- |
| Мета тесту | Перевірити переміщення шашки з порожнього поля |
| Початковий стан програми | Відкрите ігрове меню |
| Вхідні дані | Дані з пристрою вводу. Значення клітинки, з якої перемістити шашку, значення клітинки, на яку перемістити |
| Схема проведення тесту | Обробка вводу користувача та виконання переміщення шашки |
| Очікуваний результат | Програма виведе повідомлення, що початкова клітинка порожня |
| Стан програми після проведення випробувань | На екран виведено повідомлення про порожню клітинку |

Таблиця 5.3 ‑ Приклад роботи програми при переміщенні шашки на зайняте поле.

|  |  |
| --- | --- |
| Мета тесту | Перевірити переміщення шашки на зайняте поле |
| Початковий стан програми | Відкрите ігрове меню |
| Вхідні дані | Дані з пристрою вводу. Значення клітинки, з якої перемістити шашку, значення клітинки, на яку перемістити |
| Схема проведення тесту | Обробка вводу користувача та виконання переміщення шашки |
| Очікуваний результат | Програма виведе повідомлення, що кінцева клітинка зайнята |
| Стан програми після проведення випробувань | На екран виведено повідомлення про зайняту клітинку |

Таблиця 5.4 ‑ Приклад роботи програми при переміщенні шашки в неправильному напрямку.

|  |  |
| --- | --- |
| Мета тесту | Перевірити переміщення шашки в неправильному напрямку |
| Початковий стан програми | Відкрите ігрове меню |
| Вхідні дані | Дані з пристрою вводу. Значення клітинки, з якої перемістити шашку, значення клітинки, на яку перемістити |
| Схема проведення тесту | Обробка вводу користувача та виконання переміщення шашки |
| Очікуваний результат | Програма виведе повідомлення про спробу перемістити шашку в неправильному напрямку |
| Стан програми після проведення випробувань | На екран виведено повідомлення про неправильний напрямок ходу |

# 

Таблиця 5.5 ‑ Приклад роботи програми при переміщенні шашки на неправильну кількість клітинок.

|  |  |
| --- | --- |
| Мета тесту | Перевірити переміщення шашки на неправильну кількість клітинок |
| Початковий стан програми | Відкрите ігрове меню |
| Вхідні дані | Дані з пристрою вводу. Значення клітинки, з якої перемістити шашку, значення клітинки, на яку перемістити |
| Схема проведення тесту | Обробка вводу користувача та виконання переміщення шашки |
| Очікуваний результат | Програма виведе повідомлення про спробу перемістити шашку на неправильну кількість клітинок |
| Стан програми після проведення випробувань | На екран виведено повідомлення про неправильний хід шашкою |

Таблиця 5.6 ‑ Приклад роботи програми при переміщенні шашки під час ходу супротивника.

|  |  |
| --- | --- |
| Мета тесту | Перевірити переміщення шашки під час ходу супротивника |
| Початковий стан програми | Відкрите ігрове меню |
| Вхідні дані | Дані з пристрою вводу. Значення клітинки, з якої перемістити шашку, значення клітинки, на яку перемістити |
| Схема проведення тесту | Обробка вводу користувача та виконання переміщення шашки |
| Очікуваний результат | Програма виведе повідомлення про спробу перемістити шашку під час ходу супротивника |
| Стан програми після проведення випробувань | На екран виведено повідомлення, що неможливо перемістити шашку протягом ходу противника |

Таблиця 5.7 ‑ Приклад роботи програми при переміщенні дамки.

|  |  |
| --- | --- |
| Мета тесту | Перевірити переміщення дамки |
| Початковий стан програми | Відкрите ігрове меню |
| Вхідні дані | Дані з пристрою вводу. Значення клітинки, з якої перемістити дамку, значення клітинки, на яку перемістити |
| Схема проведення тесту | Обробка вводу користувача та виконання переміщення дамки |
| Очікуваний результат | Дамка переміститься на вказану позицію |
| Стан програми після проведення випробувань | Дамка знаходиться в заданій позиції |

Таблиця 5.8 ‑ Приклад роботи програми при атаці шашкою.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність атаки шашкою |
| Початковий стан програми | Відкрите ігрове меню |
| Вхідні дані | Дані з пристрою вводу. Значення клітинки, з атакуючою шашкою, значення кінцевої клітинки |
| Схема проведення тесту | Обробка вводу користувача та виконання атаки шашкою |
| Очікуваний результат | Шашка побила шашку супротивника та перемістилася на кінцеву позицію |
| Стан програми після проведення випробувань | Шашка знаходиться в заданій позиції та шашка противника вибита з гри |

Таблиця 5.9 ‑ Приклад роботи програми при атаці шашкою з порожньої клітинки.

|  |  |
| --- | --- |
| Мета тесту | Перевірити атаки шашки з порожньої клітинки |
| Початковий стан програми | Відкрите ігрове меню |
| Вхідні дані | Дані з пристрою вводу. Значення клітинки, з атакуючою шашкою, значення кінцевої клітинки |
| Схема проведення тесту | Обробка вводу користувача та виконання атаки шашкою |
| Очікуваний результат | Програма виведе повідомлення, що початкова клітинка порожня |
| Стан програми після проведення випробувань | На екран виведено повідомлення про порожню клітинку |

Таблиця 5.10 ‑ Приклад роботи програми при атаці шашкою на зайняте поле.

|  |  |
| --- | --- |
| Мета тесту | Перевірити атаки шашки з переміщенням на зайняте поле |
| Початковий стан програми | Відкрите ігрове меню |
| Вхідні дані | Дані з пристрою вводу. Значення клітинки, з атакуючою шашкою, значення кінцевої клітинки |
| Схема проведення тесту | Обробка вводу користувача та виконання атаки шашкою |
| Очікуваний результат | Програма виведе повідомлення, що кінцева клітинка зайнята |
| Стан програми після проведення випробувань | На екран виведено повідомлення про зайняту клітинку |

Таблиця 5.11 ‑ Приклад роботи програми при атаці шашкою в неправильному напрямку.

|  |  |
| --- | --- |
| Мета тесту | Перевірити атаки шашки в неправильному напрямку |
| Початковий стан програми | Відкрите ігрове меню |
| Вхідні дані | Дані з пристрою вводу. Значення клітинки, з атакуючою шашкою, значення кінцевої клітинки |
| Схема проведення тесту | Обробка вводу користувача та виконання атаки шашкою |
| Очікуваний результат | Програма виведе повідомлення, напрямок атаки неправильний |
| Стан програми після проведення випробувань | На екран виведено повідомлення невірний напрямок атаки |

Таблиця 5.12 ‑ Приклад роботи програми при атаці шашкою в напрямку без шашки супротивника.

|  |  |
| --- | --- |
| Мета тесту | Перевірити атаки шашки в напрямку без шашки супротивника |
| Початковий стан програми | Відкрите ігрове меню |
| Вхідні дані | Дані з пристрою вводу. Значення клітинки, з атакуючою шашкою, значення кінцевої клітинки |
| Схема проведення тесту | Обробка вводу користувача та виконання атаки шашкою |
| Очікуваний результат | Програма виведе повідомлення, що на шляху відсутня шашка противника |
| Стан програми після проведення випробувань | На екран виведено повідомлення, що відсутній шашка противника на шляху |

Таблиця 5.13 ‑ Приклад роботи програми при атаці шашкою під час ходу супротивника.

|  |  |
| --- | --- |
| Мета тесту | Перевірити атаки шашки під час ходу супротивника |
| Початковий стан програми | Відкрите ігрове меню |
| Вхідні дані | Дані з пристрою вводу. Значення клітинки, з атакуючою шашкою, значення кінцевої клітинки |
| Схема проведення тесту | Обробка вводу користувача та виконання атаки шашкою |
| Очікуваний результат | Програма виведе повідомлення про спробу атаки шашкою під час ходу супротивника |
| Стан програми після проведення випробувань | На екран виведено повідомлення, що неможливо атакувати шашкою протягом ходу противника |

Таблиця 5.14 ‑ Приклад роботи програми при атаці дамкою.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність атаки дамкою |
| Початковий стан програми | Відкрите ігрове меню |
| Вхідні дані | Дані з пристрою вводу. Значення клітинки, з атакуючою дамкою, значення кінцевої клітинки |
| Схема проведення тесту | Обробка вводу користувача та виконання атаки дамкою |
| Очікуваний результат | Дамка побила шашку супротивника та перемістилася на кінцеву позицію |
| Стан програми після проведення випробувань | Дамка знаходиться в заданій позиції та шашка противника вибита з гри |

Таблиця 5.15 ‑ Приклад роботи програми при переміщення шашки під час атаки.

|  |  |
| --- | --- |
| Мета тесту | Перевірити переміщення шашки під час атаки |
| Початковий стан програми | Відкрите ігрове меню |
| Вхідні дані | Дані з пристрою вводу. Значення клітинки, з атакуючою шашкою, значення кінцевої клітинки |
| Схема проведення тесту | Обробка вводу користувача та виконання атаки шашкою |
| Очікуваний результат | Програма виведе повідомлення про спробу переміщення шашки під час атаки |
| Стан програми після проведення випробувань | На екран виведено повідомлення, що неможливо переміщати шашки під час атаки |

Таблиця 5.16 ‑ Приклад роботи програми при обробці команди game.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність роботи команди game |
| Початковий стан програми | Відкрите початкове меню |
| Вхідні дані | Дані з пристрою вводу. Значення введеної команди |
| Схема проведення тесту | Обробка вводу користувача та виконання команди |
| Очікуваний результат | Відкрилось ігрове меню |
| Стан програми після проведення випробувань | На екрані знаходиться ігрове меню |

Таблиця 5.17 ‑ Приклад роботи програми при обробці команди exit.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність роботи команди exit |
| Початковий стан програми | Відкрите початкове меню |
| Вхідні дані | Дані з пристрою вводу. Значення введеної команди |
| Схема проведення тесту | Обробка вводу користувача та виконання команди |
| Очікуваний результат | Робота програми закынчилась |
| Стан програми після проведення випробувань | Програма завершилась |

Таблиця 5.18 ‑ Приклад роботи програми при обробці команди back.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність роботи команди back |
| Початковий стан програми | Відкрите ігрове меню |
| Вхідні дані | Дані з пристрою вводу. Значення введеної команди |
| Схема проведення тесту | Обробка вводу користувача та виконання команди |
| Очікуваний результат | Відкрилось стартове меню |
| Стан програми після проведення випробувань | На екрані знаходиться стартове меню |

Таблиця 5.19 ‑ Приклад роботи програми при обробці команди restart.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність роботи команди restart |
| Початковий стан програми | Відкрите ігрове меню |
| Вхідні дані | Дані з пристрою вводу. Значення введеної команди |
| Схема проведення тесту | Обробка вводу користувача та виконання команди |
| Очікуваний результат | Гра почалась спочатку |
| Стан програми після проведення випробувань | Розпочалась нова гра |

Таблиця 5.20 ‑ Приклад роботи програми при обробці команди undo.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність роботи команди undo |
| Початковий стан програми | Відкрите ігрове меню |
| Вхідні дані | Дані з пристрою вводу. Значення введеної команди |
| Схема проведення тесту | Обробка вводу користувача та виконання команди |
| Очікуваний результат | Остання команда відмінилась і гра повернулась у попередній стан |
| Стан програми після проведення випробувань | Гра знаходится в попередньому стані |

Таблиця 5.21 ‑ Приклад роботи програми при обробці команди move.

|  |  |
| --- | --- |
| Мета тесту | Перевірити коректність роботи команди move |
| Початковий стан програми | Відкрите ігрове меню |
| Вхідні дані | Дані з пристрою вводу. Значення введеної команди |
| Схема проведення тесту | Обробка вводу користувача та виконання команди |
| Очікуваний результат | Шашка перемістилася в залежності від введених координат |
| Стан програми після проведення випробувань | Шашка перемістилась згідно правил гри |

Таблиця 5.22 ‑ Приклад роботи програми при обробці команди move з неправильним форматом позиції.

|  |  |
| --- | --- |
| Мета тесту | Перевірити роботу команди move при введенні позиції в неправильному форматі |
| Початковий стан програми | Відкрите ігрове меню |
| Вхідні дані | Дані з пристрою вводу. Значення введеної команди |
| Схема проведення тесту | Обробка вводу користувача та виконання команди |
| Очікуваний результат | Програма виведе повідомлення про невірний формат позиції |
| Стан програми після проведення випробувань | На екран виведено повідомлення, про неправильний формат позиції |

Таблиця 5.23 ‑ Приклад роботи програми при обробці невідомої команди.

|  |  |
| --- | --- |
| Мета тесту | Перевірити роботу команди move при введенні позиції в неправильному форматі |
| Початковий стан програми | Відкрите ігрове меню |
| Вхідні дані | Дані з пристрою вводу. Значення введеної команди |
| Схема проведення тесту | Обробка вводу користувача та виконання команди |
| Очікуваний результат | Програма виведе повідомлення про невідому команду |
| Стан програми після проведення випробувань | На екран виведено повідомлення, що такої команди не існує |

# 

# 6 ІНСТРУКЦІЯ КОРИСТУВАЧА

## 6.1 Робота з програмою

Після запуску виконавчого файлу з розширенням \*.exe (для Windows) відкривається головне меню програми (Рисунок 6.1)



Рисунок 6.1 – Початкове меню програми

Далі за допомогою клавіатури необхідно вести необхідну команду. Для початку гри потрібно ввести команду “game”, після якої відкриється ігрове меню (Рисунок 6.2):

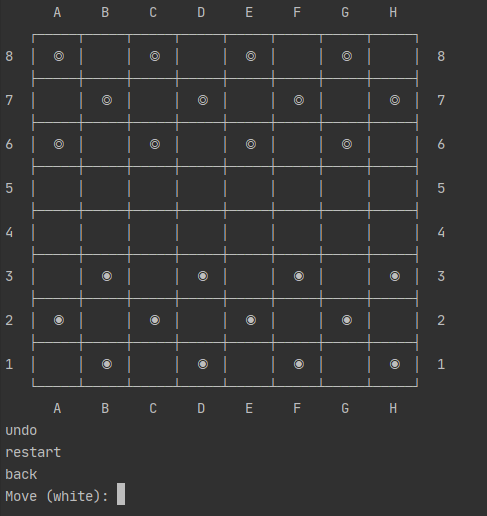


Рисунок 6.2 – Ігрове меню програми

Для виходу з програми потрібно ввести команду “exit” (Рисунок 6.3):

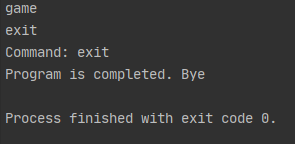


Рисунок 6.3 – Завершення роботи програми

Якщо ввести команду, якої немає в списку, то програма виведе повідомлення про неправильну когманду та запропоную ввести іншу команду (Рисунок 6.4):

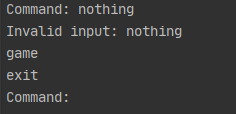


Рисунок 6.4 – Повідомлення при введенні неправильної команди

Під час гри для переміщення шашки на полі потрібно вести початкову позицію, на якій знаходиться шашка та кінцеву позицію (Рисунок 6.5):

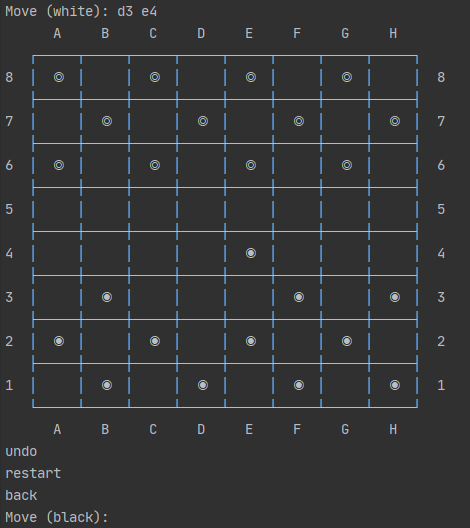


Рисунок 6.5 – Стан гри після ходу D3 E4

Під час гри для атаки шашки противника потрібно вести початкову позицію, на якій знаходиться шашка та кінцеву позицію, на якій повинна бути шашка після атаки (Рисунок 6.6):

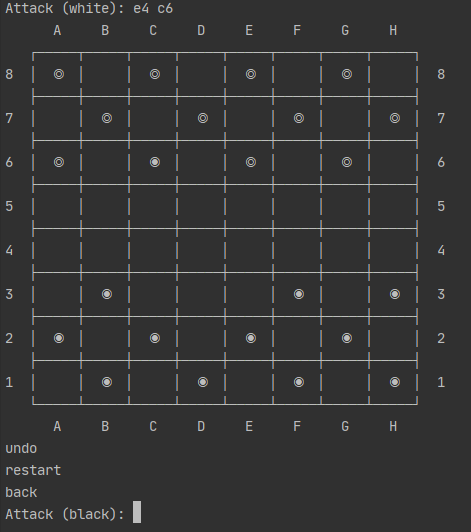


Рисунок 6.6 – Стан гри після ходу E3 C6

Якщо ввести неправильну початкову позицію або кінцеву позицію програма повідомить про помилку та запропонує ввести позицію ще раз (Рисунок 6.7):

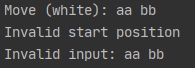


Рисунок 6.7 – Повідомлення при введенні неправильної позиції

Якщо під час виконання команди переміщення або атаки, буде виявлено, що даний хід протирічить правилам (наприклад хід на зайняту клітинку), програма повідомить про помилку та запропонує походити ще раз (Рисунок 6.8):

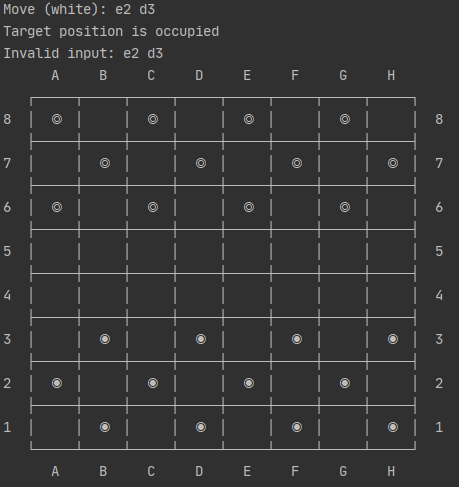


Рисунок 6.8 – Повідомлення про помилку під час ходу E2 D3

Для відміни останнього ходу необхідно ввести команду “undo” (Рисунок 6.6):

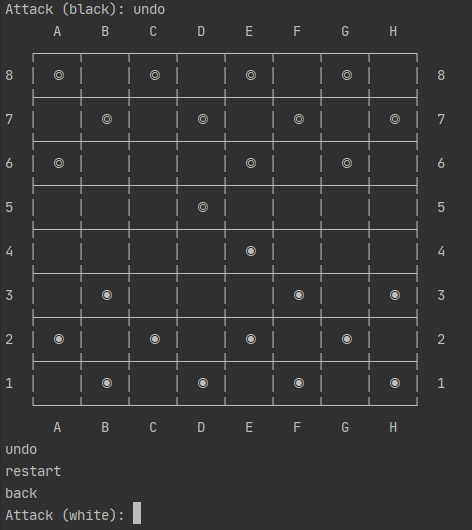


Рисунок 6.6 – Стан гри після відміни ходу E3 C6

Для початку нової гри необхідно ввести команду “restart” (Рисунок 6.7):

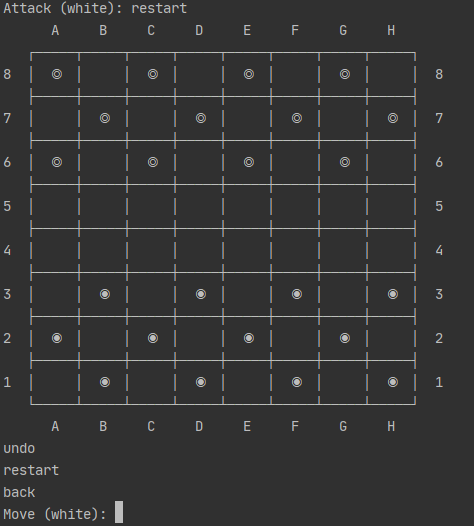


Рисунок 6.7 – Нова гра після команди “restart”

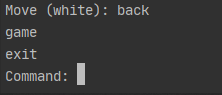
Для повернення до початкового меню необхідно ввести команду “back” (Рисунок 6.8):

Рисунок 6.6 – Початкове меню після команди “back”

## 6.2 Формат вхідних та вихідних даних

Користувач вводить координати і якщо введені дані валідні, відображає на екрані результат пострілу.

Результатом виконання програми є графічне відображення в консолі дій гравців в залежності від введених команд. Під час некоректного ходу з’являється повідомлення про невірні дані і для продовження користувач має ввести дані знову. Після будь-якої правильної команди гравці повинні мати змогу побачити всі зміни на полі.

## 6.3 Системні вимоги

|  |  |  |
| --- | --- | --- |
|  | Мінімальні | Рекомендовані |
| Операційна система | Windows XP+, Linux (Ubuntu 14.04+, Fedora 25+) | Windows 10, Linux (Fedora 27+, Ubuntu 16.04 +) |
| Процесор | 2.0 GHz Intel Celeron с підтримкою SSE2 інструкцій | 2.5 GHz Intel Core i3 с підтримкою SSE2 інструкцій |
| Оперативна пам'ять | 512 MB RAM | 1 GB RAM |
| Відеоадаптер | Не потребує | |
| Дисплей | 800х600 | 1024х768 або краще |

|  |  |  |
| --- | --- | --- |
|  | Мінімальні | Рекомендовані |
| Прилади введення | Клавіатура | |
| Додаткове програмне забезпечення | Відсутнє | |

# ВИСНОВОК

В рамках даної курсової роботи було розглянуто основні підходи до проектування консольних застосувань та поглибленні знання в забезпеченнях з модульно-монолітною архітектурою. Були засвоєні знання про шаблони проектування або патерни в розробці програмного забезпечення.

Також в рамках даної роботи було розроблено власне застосування: гра «Морський бій» на платформі .NET, використовуючи сценарну мову C#. Розроблена програма може бути запущена на більшості актуальних операційних системах, таких як Windows 10, Linux (Ubuntu, Linux), та потребує мінімальних ресурсів.

# 

# ПЕРЕЛІК ПОСИЛАНЬ

1. Статья про морський бій [Електронний ресурс] — Режим доступу : <https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D1%80%D1%81%D1%8C%D0%BA%D0%B8%D0%B9_%D0%B1%D1%96%D0%B9>
2. Статья про математичні моделі в грі морський бій [Електронний ресурс] — Режим доступу :

http://www.irbis-nbuv.gov.ua/cgi-bin/irbis\_nbuv/cgiirbis\_64.exe?C21COM=2&I21DBN=UJRN&P21DBN=UJRN&IMAGE\_FILE\_DOWNLOAD=1&Image\_file\_name=PDF/dpitsca\_2018\_1\_7.pdf

1. Статья про ігри з повною інформацією [Електронний ресурс] — Режим доступу : <https://uk.wikipedia.org/wiki/%D0%93%D1%80%D0%B0_%D0%B7_%D0%BF%D0%BE%D0%B2%D0%BD%D0%BE%D1%8E_%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D1%94%D1%8E>
2. Основний код програми [Електронний ресурс] — Режим доступу :

<https://github.com/Andrey2102/sea-fight>

# ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ

КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського

Кафедра

автоматизованих систем обробки інформації та управління

Затвердив

Керівник Крамар Юлія Михайлівна

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 р.

Виконавець:

Студент Тютюнник Андрій Олександрович

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання курсової роботи

на тему: Проектування та розробка застосування на платформі .NET. Створення гри “Морський бій”

з дисципліни:

«Об'єктно-орієнтоване програмування - 1»

Київ 2020

1. *Мета*: Метою курсової роботи є розробка консольного програмного забезпечення «Морський бій»
2. *Дата початку роботи*: «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2020 р.
3. *Дата закінчення роботи*: «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2020 р.
4. *Вимоги до програмного забезпечення*.
5. Функціональні вимоги:

* Можливість розпочати нову гру
* Можливість атакувати позицію суперника
* Можливість переглянути результати попередніх ігор
* Врахувати правильність розташування кораблів на полі
* Врахувати відображення підбитої області навколо знищеного корабля
* Наявність почергового гри двох гравців
* Перевірка коректності ходів гравців та повідомлення їх при неправильних діях
* Нефункціональні вимоги:
* Програмне застосування повинно бути розроблено на платформі .NET, використовуючи мову C#
* Бути доступною для гравців будь-якого рівня та досвіду
* Можливість працювати під час критичних навантажень
* Бути стійкою до збоїв
* Робота на операційній системі Windows
* Все програмне забезпечення та супроводжуюча технічна документація повинні задовольняти наступним ДЕСТам:

ГОСТ 29.401 - 78 - Текст програми. Вимоги до змісту та оформлення.

ГОСТ 19.106 - 78 - Вимоги до програмної документації.

ГОСТ 7.1 - 84 та ДСТУ 3008 - 95 - Розробка технічної документації.

1. *Стадії та етапи розробки*:
2. Об'єктно-орієнтований аналіз предметної області задачі (до\_\_.\_\_.2020 р.)
3. Об'єктно-орієнтоване проектування архітектури програмної системи (до \_\_.\_\_.2020 р.)
4. Розробка програмного забезпечення (до \_\_.\_\_.2020 р.)
5. Тестування розробленої програми (до \_\_.\_\_.2020 р.)
6. Розробка пояснювальної записки (до \_\_.\_\_.2020 р.).
7. Захист курсової роботи (до \_\_.\_\_.2020 р.).
8. *Порядок контролю та приймання*. Поточні результати роботи над КР регулярно демонструються викладачу. Своєчасність виконання основних етапів графіку підготовки роботи впливає на оцінку за КР відповідно до критеріїв оцінювання.

# 

# ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ

(Вид носія даних)

*CD-RW*

(Найменування програми (документа))

*Тексти програмного коду програмного забезпечення гри «Морський бій»*

(Обсяг програми (документа), арк., Кб)

*арк, 29593 Кб*

*студента групи ІП-82 II курсу*

*Тютюнника А.О.*