МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ВІДОКРЕМЛЕНИЙ СТУКТУРНИЙ ПІДРОЗДІЛ

«ПОЛТАВСЬКИЙ ПОЛІТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ   
НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ   
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Циклова комісія дисциплін програмної інженерії

**КУРСОВА РОБОТА**

молодшого спеціаліста

Комп’ютерна гра «Морський бій»

на тему \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Виконав: здобувач освіти 2 курсу,

24

групи \_\_\_\_\_\_\_  
спеціалізації «Розробка програмного забезпечення»

Шпачинський І.М.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (прізвище та ініціали)

Олійник В.В.

Керівник\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 (підпис) (прізвище та ініціали)

Полтава – 2021

**ЗМІСТ**

[ВСТУП 3](#_Toc74395853)

[1. ПОСТАНОВКА ЗАДАЧІ 4](#_Toc74395854)

[1.1. Постановка задачі 4](#_Toc74395855)

[1.2. Основні вимоги до програми 4](#_Toc74395856)

[2. ОПИС ПРОЕКТНИХ РІШЕНЬ 5](#_Toc74395857)

[2.1. Опис програмних засобів та програмного забезпечення 5](#_Toc74395858)

[2.2. Алгоритм програми 7](#_Toc74395859)

[2.3. Опис специфікації функцій 8](#_Toc74395860)

[2.4. Тестування 16](#_Toc74395861)

[3. ІНСТРУКЦІЯ КОРИСТУВАЧА 17](#_Toc74395862)

[ВИСНОВКИ 20](#_Toc74395863)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 21](#_Toc74395864)

[ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ 22](#_Toc74395865)

# ВСТУП

Всі, хто має справу з комп'ютером, так чи інакше стикалися з комп'ютерними іграми. Захоплюючим є не лише процес гри, але і проектування та розробка ігор.

У даній курсовій роботі мова піде про створення ігрової консольної програми «Морської бій», яка і буде об'єктом дослідження.

Гра «Морський бій» відома мабуть всім ще з дитинства. Суть гри полягає в тому, щоб потопити всі кораблі на невідомій карті противника, по черзі називаючи координати. Ігрове поле кожного гравця – квадрат розміром 10 на 10 комірок, на якому розміщується флот кораблів. Перемагає той, хто першим потопить всі кораблі противника.

Завданням даної роботи є розробка ігрової програми мовою програмування С++ використанням середовища Microsoft Visual Studio. Програма повинна надавати можливість ручного та автоматичного розміщення кораблів користувача. Кораблі комп`ютера розміщуються автоматично. Далі гра відбувається за загальними правилами. Першим хід здійснює користувач.

При розробці програми використовується парадигма функціонального програмування - створення програм, в яких основною дією є виклик функції. Для цього необхідні функції (для формування полів з корабляими та для ведення бою) згруповані в окремі користувацькі бібліотеки.

# ПОСТАНОВКА ЗАДАЧІ

## Постановка задачі

Розробити ігрову програму «Морської бій», яка включає ігрову консольну панель, що складається з двох полів - для людини і комп'ютера. Програма повинна дозволяти розставляти кораблі на полі 10\*10, контролювати правильність їх розстановки. На кожному полі розставляються 10 кораблів: 4 кораблі розміром в 1 клітинку(«підводні човни», або «однопалубні»), 3 кораблі розміром в 2 клітинки(«есмінці», або «двопалубні»), 2 кораблі розміром в 3 клітинки(«крейсери», або «трипалубні»), і 1 корабель розміром в 4 клітинки(«лінкор», або «чотирипалубний»).

Після формування ігрових полів програма повинна давати противникам можливість по черзі робити ходи і видавати відповідні інформаційні повідомлення. При попаданні надається можливість додаткового пострілу, в протилежному випадку хід переходить до противника. Перемогу отримує гравець, що потопив всі кораблі супротивника.

## 1.2. Основні вимоги до програми

Ігрова програма «Морський бій» повинна реалізовуватись за класичними правилами між гравцем та комп’ютером, який робить свої кроки випадковим чином. Програма повинна мати зрозуміле меню, що забезпечує інтерактивну взаємодію користувача з комп'ютером та надає можливість:

* Продовжити розпочату раніше гру;
* Почати нову гру;
* Обрати автоматичну розстановку кораблів на полі;
* Задати розташування кораблів в ручному режимі;
* Вводити координати пострілів та отримувати повідомлення про влучність пострілу.

У програмі потрібно реалізувати збереження стану полів гравця та комп'ютера в окремі файли, для того, щоб можна було продовжити незакінчену раніше гру.

# 2. ОПИС ПРОЕКТНИХ РІШЕНЬ

## 2.1. Опис програмних засобів та програмного забезпечення

Мова програмування С++ розроблена в 1979 році. Метою створення С++ було доповнення мови програмування С можливостями, зручними для більш глобальної розробки ПО, зі збереженням гнучкості, швидкості і портативності мови C. Разом з тим творці C++ прагнули зберегти сумісність цієї мови програмування з мовою C.

Зараз мова C++ широко використовується для розробки програмного забезпечення, будучи однією з найпопулярніших мов програмування. З її допомогою створюють операційні системи, різноманітні прикладні програми, драйвера пристроїв, ігри тощо.

Мова програмування C++ є універсальною, оскільки вона підтримує кілька парадигм програмування: об'єктно-орієнтовану, узагальнену та процедурну.

Для розробки ігрової програми «Морський бій» було викоистано середовище розробки Microsoft Visual Studio Community 2019.

Microsoft Visual Studio — серія продуктів фірми Майкрософт, які містять інтегроване середовище розробки програмного забезпечення та низку інших інструментальних засобів. Ці продукти дають змогу розробляти як консольні програми, так і програми з графічним інтерфейсом, включно з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight.

За допомогою Visual Studio можна розробляти:

* Класичні додатки для комп'ютера під керуванням операційної системи Windows;
* Мобільні додатки (Windows, iOS, Android);
* Web-додатки;
* Хмарні додатки;
* Різні розширення для Office, SharePoint, а також створення власних розширень для Visual Studio;
* ігри;
* Бази даних SQL Server і SQL Azure.

У Visual Studio можна використовувати наступні технології і мови програмування: .NET, Node.js, C, C #, C ++, Python, Visual Basic, F #, JavaScript.

Актуальною на поточний момент є версія Visual Studio 2019, випускається вона в наступних редакціях:

* *Community* - безкоштовна версія середовища розробки Visual Studio. Щоб її використовувати, необхідно створити обліковий запис Visual Studio, в іншому випадку вона буде діяти 30 днів. Редакція має менший функціонал в порівнянні з платними редакціями, однак вона включає все необхідне для створення повноцінних додатків. Підходить для індивідуальних розробників і навчання. Саме цю редакцію використано при розробці ігрової програми «Мрський бій»;
* *Professional* - редакція містить професійні інструменти для розробки додатків. Функціонал в цій редакції все одно не повний, наприклад, це помітно в частині інструментів діагностики, налагодження і тестування. Підходить для невеликих груп розробників;
* *Enterprise* - повнофункціональна версія Visual Studio. Комплексне рішення для розробки додатків. Підходить для груп будь-якого розміру з високими вимогами до якості і масштабу.

**Системні вимоги Visual Studio 2019**

Середовище Visual Studio 2019 можна встановити і працювати в ньому на наступних операційних системах (перераховані офіційно підтримувані версії):

* Windows 7 з Service Pack 1;
* Windows 8.1 (з оновленням 2919355);
* Windows 10 (1703 і вище);
* Windows Server 2012 R2 (з оновленням 2919355);
* Windows Server 2016 (Standard і Datacenter);
* Windows Server 2019 (Standard і Datacenter).

**Мінімальні вимоги до обладнання:**

* Процесор з тактовою частотою не нижче 1,8 ГГц. Рекомендується використовувати як мінімум двоядерний процесор;
* 2 ГБ оперативної пам'яті, рекомендується 8 ГБ (якщо встановлювати на віртуальну машину, то мінімум 2.5 ГБ);
* Вільного місця на жорсткому диску від 800 мегабайт до 210 гігабайт, залежно від встановлених компонентів. У більшості випадків виділяйте як мінімум 30 гігабайт, наприклад, я встановлював всього кілька компонентів, і у мене зайняло більше 20 ГБ місця на жорсткому диску. Також Microsoft рекомендує встановлювати Visual Studio на SSD диск, я як раз встановлював на SSD (на своєму ноутбуці), швидкість роботи Visual Studio мене, звичайно, порадувала;
* Відеоадаптер з мінімальним дозволом 1280 на 720 пікселів (для оптимальної роботи Visual Studio рекомендується дозвіл 1366 на 768 пікселів і вище).

## 2.2. Алгоритм програми

На початку гри на екран виводиться запитання про вибір автоматичної чи самостійної розстановки кораблів. При виборі автоматичної розстановки автоматично вибирається напрям корабля і його координати. Якщо ж вибрати самостійну розстановку, то з’явиться повідомлення про вибір напряму і координати першої «палуби» корабля, при цьому здійснюється перевірка суміжних клітин. Корабель не розміститься, якщо на сусідніх клітинах буде виявлено інший корабель. Кораблі комп’ютера розставляються автоматично.

«Палуби» кораблів на полі зображені символом «». Клітини, в які ще не здійснювали постріл, позначені символом «■».

Після розміщення кораблів по полю, починається сама гра. Першим «стріляє» гравець. Надається вибір координати на карті комп’ютера, при цьому вносяться зміни в масиви кораблів комп’ютера, та пострілів гравця. При попаданні клітинка збитої «палуби» корабля замінюється літерою «X», і надається можливість додаткового пострілу, в протилежному випадку пуста клітинка змінюється символом «\*», і хід переходить до противника.

Координати за якими стріляє комп’ютер вибираються випадково, при цьому перевіряється, чи вибрав вже комп’ютер ці координати.

Перемогу отримує гравець, що потопив всі кораблі супротивника. Якщо в комп’ютера не залишилося неушкоджених кораблів, на екран виводиться повідомлення «Ой ... Я, здається, програв. Вiтаю, ти - переможець!», якщо ж кораблів не залишилось в гравця то з’являється «Ви програли. Менi дуже шкода.».

## 2.3. Опис специфікації функцій

Проект гри "Морський бій" включає п'ять файлів з вихідними кодами. Три з них мають розширення *.cpp* і два з них заголовочні.

Файли *PlaceShip.cpp* і *PlaceShip.h* здійснюють розстановку кораблів на поле комп'ютера і користувача. Другий файл вміщує прототипи функцій, а в першому файлі знаходиться їх визначення. Розстановка кораблів проводиться випадковим чином з використанням функції для отримання псевдовипадкових чисел. У грі передбачені 1 чотирипалубний корабель(«лінкор»), 2 трипалубника(«крейсери»), 3 двопалубні(«есмінці») і 3 корабля з одного палубою(«підводні човни»). Вся розстановка записується у файли, а вже в ході бою в ці ж файли вносяться необхідні зміни. Кораблі гравця видно користувачеві. Кораблі комп'ютера не відображаються, а замінені полем, на якому відображаються ходи користувача.

Файли *Battle.cpp* і *Battle.h* призначені для ведення бою. У першому файлі знаходиться визначення функцій, оголошених прототипами в другому файлі. У програмі реалізована перевірка попадання в одну і ту ж комірку, перевірка правильності введення координат і повтор ходу після попадання в корабель противника. Попадання в корабель противника позначається хрестиком. Попадання повз позначається зірочкою.

Основна частина програми розміщена в файлі *seabattle.cpp*.

Вихідний код доповнений необхідними коментарями.

Опис специфікації функцій, що розроблені в ході створення програми приведений у таблиці 2.1.

Таблиця 2.1 – Опис специфікації розроблених функцій

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № з/п | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 1 | upload | виведення повідомлення про завантаження | - | - | PlaceShip.h |
| 2 | arpole | функція виведення умовного поля комп'ютера | char\*\* ar –масив, в якому зберігається стан поля комп’ютера | - | PlaceShip.h |
| 3 | aruserpole | функція виведення умовного поля користувача | char\*\* ar\_user - масив, в якому зберігається стан поля комп’ютера | - | PlaceShip.h |
| 4 | chooseship | функція вибору типу корабля | - | int m – кількість палуб корабля | PlaceShip.h |
| 5 | Chooseorient | функція вибору орієнтації корабля | int& k  k=1 автоматичне розміщення корабля  k=0 ручне розміщення корабля | int shipus.d  1 - корабель розміщено вертикально  0 - корабель розміщено горизонтально | PlaceShip.h |
| 6 | HorCoord | функція введення горизонтальної координати | int& k  k=1 автоматичне розміщення корабля  k=0 ручне розміщення корабля | int shipus.x  горизонтальна координата | PlaceShip.h |

Продовження таблиці 2.1 – Опис специфікації розроблених функцій

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № з/п | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 7 | VerCoord | функція введення вертикальної координати | int& k  k=1 автоматичне розміщення корабля  k=0 ручне розміщення корабля | int shipus.y  вертикальна координата | PlaceShip.h |
| 8 | test | функція перевірки існування палуб | масив поля, координати першої палуби, орієнтація карабля та кількість його палуб | 1 - корабель може бути розміщено  0 - не можна розмістити такий корабель | PlaceShip.h |
| 9 | shipplace | функція розміщення корабля | масив поля, координати першої палуби, орієнтація карабля та кількість його палуб | - | PlaceShip.h |
| 10 | amountshipp | функція підрахунку палуб | char\*\* ar\_user масив поля | int sum кількість вже розміщених палуб кораблів | PlaceShip.h |
| 11 | writeuser | функція запису в файл поля користувача | char\*\* ar\_user масив поля | - | PlaceShip.h |
| 12 | readuser | функція зчитування з файлу поля користувача | char\*\* ar\_user масив поля | - | PlaceShip.h |
| 13 | testone | функція перевірки однопалубника | масив поля, координати карабля | 1 - корабель може бути розміщено  0 - не можна розмістити такий корабель | PlaceShip.h |

Продовження таблиці 2.1 – Опис специфікації розроблених функцій

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № з/п | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 14 | placeone | Функція розміщення однопалубника | масив поля, координати карабля |  | PlaceShip.h |
| 15 | onehorCoord | функція введення горизонтальної координати однопалубника |  | горизонтальна координата | PlaceShip.h |
| 16 | oneverCoord | функція введення вертикальної координати однопалубника |  | вертикальна координата | PlaceShip.h |
| 17 | orientKomp | функція вибору орієнтації корабля комп’ютера |  | 1 - корабель вертикальний  0 - корабель розміщено горизонтально | PlaceShip.h |
| 18 | HorCoordK | функція генерації горизонтальної координати корабля комп’ютера |  | горизонтальна координата | PlaceShip.h |
| 19 | VerCoordK | функція генерації вертикальної координати корабля комп’ютера |  | вертикальна координата | PlaceShip.h |
| 20 | placeoneK | Функція розміщення однопалубника | масив поля, координати карабля |  | PlaceShip.h |
| 21 | onehorCoordK | функція генерації горизонтальної координати однопалубника |  | горизонтальна координата | PlaceShip.h |

Продовження таблиці 2.1 – Опис специфікації розроблених функцій

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № з/п | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 22 | oneverCoordK | функція генерації вертикальної координати однопалубника |  | вертикальна координата | PlaceShip.h |
| 23 | testoneK | функція перевірки однопалубника | масив поля, координати карабля | 1 - корабель може бути розміщено  0 - не можна розмістити такий корабель | PlaceShip.h |
| 24 | testK | функція перевірки існування палуб | масив поля, координати першої палуби, орієнтація карабля та кількість його палуб | 1 - корабель може бути розміщено  0 - не можна розмістити такий корабель | PlaceShip.h |
| 25 | shipplaceK | функція розміщення корабля на полі комп’ютера | масив поля, координати першої палуби, орієнтація карабля та кількість його палуб | - | PlaceShip.h |
| 26 | amountshippK | функція підрахунку палуб на полі комп’ютера | char\*\* ar масив поля комп'ютера | int sum кількість вже розміщених палуб кораблів | PlaceShip.h |
| 27 | writeK | функція запису в файл поля комп’ютера | char\*\* ar масив поля | - | PlaceShip.h |
| 28 | readK | функція зчитування з файлу поля комп’ютера | char\*\* ar масив поля | - | PlaceShip.h |

Продовження таблиці 2.1 – Опис специфікації розроблених функцій

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № з/п | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 29 | ppolef | функція генерації початкового видимого поля комп'ютера | char\*\* pole масив поля | - | Battle.h |
| 30 | fppolef | функція запису у файл ходів користувача | char\*\* pole масив поля | - | Battle.h |
| 31 | sppolef | функція зчитування з файлу ходів користувача | char\*\* pole масив поля | - | Battle.h |
| 32 | amountship | функція підрахунку підбитих користувачем кораблів | char\*\* pole масив поля | 1 – пібиті всі кораблі  0 – ще не всі кораблі підбиті | Battle.h |
| 33 | hod\_userhk | введення горизонтальної координати пострілу користувача |  | горизонтальна координата пострілу користувача | Battle.h |
| 34 | hod\_uservk | введення вертикальної координати пострілу користувача |  | вертикальна координата  пострілу користувача | Battle.h |
| 35 | userhit | функція перевірки попадання | масиви полів, координати пострілу | 1- попадання | Battle.h |
| 36 | hod\_komphk | функція генерації горизонтальної координати пострілу комп'ютера |  | горизонтальна координата пострілу комп’ютера | Battle.h |

Продовження таблиці 2.1 – Опис специфікації розроблених функцій

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № з/п | Назва функції | Призначення функції | Опис вхідних параметрів | Опис вихідних параметрів | Заголовний файл |
| 37 | hod\_kompvk | функція генерації вертикальної координати пострілу комп'ютера |  | вертикальна координата пострілу комп’ютера | Battle.h |
| 38 | komphit | перевірка попадання комп'ютера | масиви полів, координати пострілу | 1- попадання | Battle.h |
| 39 | amountSh | функція підрахунку підбитих комп’ютером кораблів | char\*\* ar\_user масив поля | 1 – пібиті всі кораблі  0 – ще не всі кораблі підбиті | Battle.h |

Також приведемо таблицю стандартних функцій, які були використанні у програмі. Таблиця містить в собі назву функції, її призначення, назву стандартної бібліотеки, якій вона належить (Таблиця 2.2).

Таблиця 2.2 – Опис стандартних функцій

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва функції | Призначення функції | Заголовний файл |
| 1 | setlocale | Задає локаль, яка використовуватиметься поточною програмою. Можна змінити всі параметри локалі, або її конкретні частини. | windows.h |
| 2 | system | Функція виконує задану, через параметр системну команду. Насправді, функція не сама виконує команду, вона викликає командний процесор для виконання команд. Після виконання команди, командний процесор повертає керування програмі, повертаючи цілочисельне значення, інтерпретація якого залежить від системи. | stdlib.h |

Продовження таблиці 2.2 – Опис стандартних функцій

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Назва функції | Призначення функції | Заголовний файл |
| 3 | srand | Функція srand виконує ініціалізацію генератора випадкових чисел rand. Генератор псевдо-випадкових чисел инициализируется за допомогою аргументу seed, який грає роль зерна. | cstdlib |
| 4 | time | Функція повертає поточне календарний час системи. | ctime |
| 5 | rand | Функція rand генерує випадкові числа, повертає псевдовипадкове ціле число в діапазоні від 0 до RAND\_MAX.  Це число генерується алгоритмом, який повертає послідовність псевдо-випадкових чисел. Цей алгоритм використовує свого роду “насіння” – число, для створення серій випадкових чисел. Тобто, якщо насіння завжди буде одне і те ж, то згенерована послідовність чисел не буде змінюватися, зникне фактор стохастичности. | cstdlib |
| 6 | fopen | Функція fopen відкриває файл, ім'я якого зазначено в параметрі fname і пов'язує його з потоком, який може бути ідентифікований для виконання різних операцій з файлом. | cstdio |
| 7 | fputc | Функція fputc записує символ в потік і переміщує позицію індикатора положення. Символ записується в поточну позицію в потоці filestream, після чого, поточна позиція індикатора положення зсувається на один символ. | cstdio |
| 8 | fgetc | Функція fgets прочитує символи з потоку і зберігає їх у вигляді рядка в параметр string до тих пір поки не настане кінець рядка або поки не буде досягнутий кінець файлу. | cstdio |
| 9 | fclose | Функція fclose закриває і роз'єднує файл filestream, пов'язаний з потоком.  Всі внутрішні буфери, пов'язані з потоком скидаються: зміст будь-якого незаписаного буфера записується і зміст будь-якого непрочитаного буфера скидається. | cstdio |

## Тестування

Тестування програми здійснювалося на комп’ютерах в яких була встановлена операційна система Windows 7 або Windows 10.

В процесі тестування функцій автоматичного формування ігрових полів програми виявлено наступні недоліки:

1. Автоматичне формування поля користувача та комп’ютера в деяких випадках потребує багато часу;
2. При провірці можливості розміщення корабля в деяких випадках не враховуються той факт, шо кораблі не можуть розміщуватись на сусідніх клітинах.

В подальшому планується вдосконалення алгоритму автоматичної розстановки кораблів на ігрових полях користувача та комп’ютера.

Що стосується безпосередньо ведення самого бую, то доцільним було б запрограмувати алгоритм логічного ведення бою комп’ютером. На разі кооординати кожного наступного ходу комп’ютера вибираються випадковим чином, що в деяких випадках не є раціональним. Наприклад, у випадку попадання доцільно проводити обстріл сусідніх клітин.

Всі описані вище недоліки можуть бути виправлені в процесі подальшого вдосконалення створеної консольної ігрової програми «Морський бій».

# 3. ІНСТРУКЦІЯ КОРИСТУВАЧА

Запуск програми здійснюється таким чином: необхідно відкрити папку, що містить дану гру, і знайти виконуваний файл SeaBattle.exe. На цьому файлі слід виконати подвійне клацання лівою клавішею маніпулятора мишки.

При запуску програми в першу чергу з’являється повідомення про можливість вибору: почати нову гру чи продовжити попередню (Рисунок 3.1). Така опція можлива саме за рахунок того, що після кожного ходу стан полів користувача та комп’ютера записується у текстовий файл. Таким чином, завжди можна продовжити попередньо почату гру.

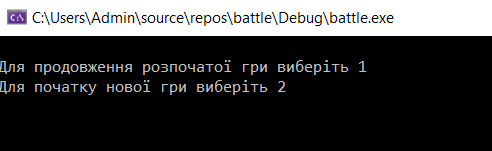


Рисунок 3.1 – Меню вибору попередньої чи нової гри

При виборі продовження раніше розпочатої гри ми повертаємось до того стану гри, що був збережений у файлі (Рисунок 3.2).

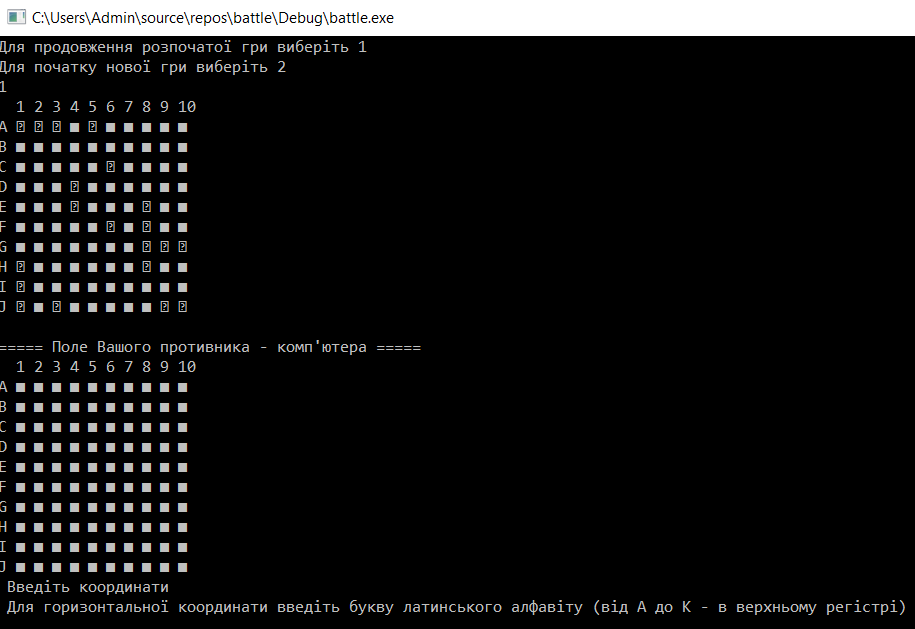


Рисунок 3.2 – Продовження раніше розпочатої гри

У випадку вибору нової гри з`являється повідомлення про вибір автоматичного та самостійного розташування кораблів (Рисунок 3.3).

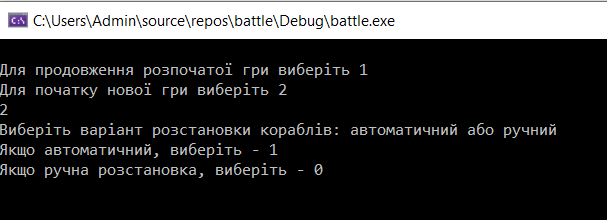


Рисунок 3.3 – Меню виборо режиму розташування кораблів

При виборі автоматичного режиму спочатку генеруються Ваше поле, та поле комп'ютера. Після цього починається сама гра (Рисунок 3.4).

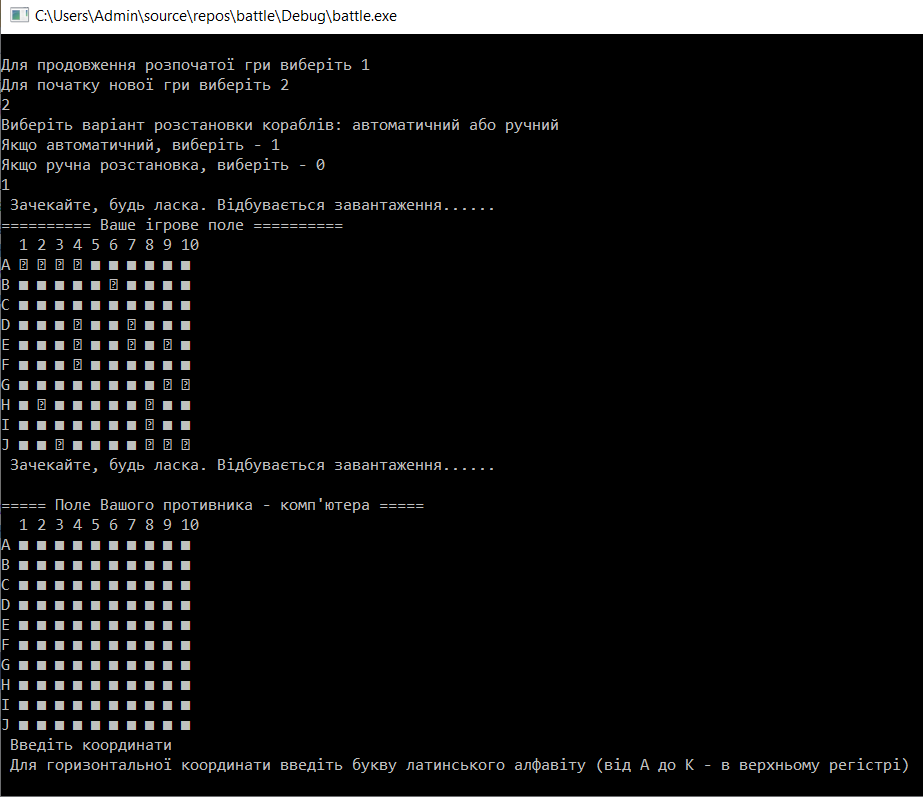
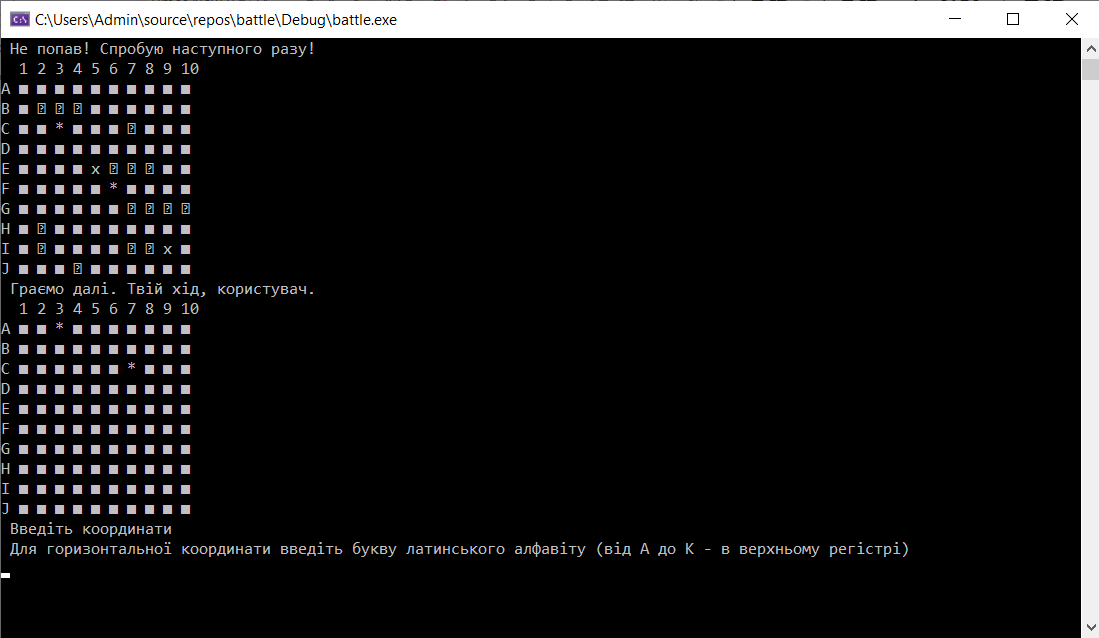


Рисунок 3.4 – Згенеровані поля та початок гри

Сам хід гри відбувається наступним чином: комп’ютер запитує в користувача координати пострілу, перевіряє чи є попадання. Якщо є, то користувач робить ще один хід. Якщо ж ні, то хід робить комп’ютер.



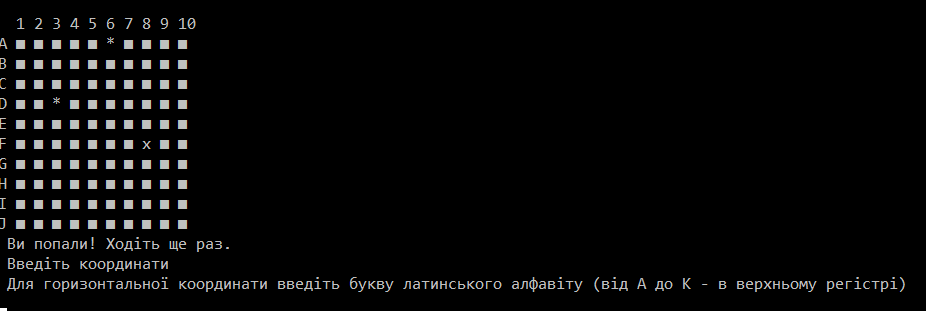


Рисунок 3.5 – Хід гри

# ВИСНОВКИ

В ході виконання даної курсової роботи була розроблена консольна ігрова програма «Морський бій», яка надає наступний функціонал:

1. Створення ігрових полів користувача в одному із 2-х режимів (автоматичний та ручний);
2. Генерація ігрового поля комп`ютера;
3. Ведення бою відповідно до відомої настільної гри «Морський бій»;
4. Запис поточного стану гри в текстові файли;
5. Можливість продовження раніше розпочатої гри.

Для розробки програми вткористовувалась мова програмування С++ та середовище розробки Microsoft Visual Studio Community 2019. Програма розроблялась з врахуванням парадигм функціонального програмування – було створено дві бібліотеки з набором функцій для генерації полів та веденням гри.

В ході тестування створеної програми було виявлено кілька недоліків, які будуть усунені в процесі вдосконалення гри. Загалом поставлене завдання можна вважати виконаним.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Белецкий Я. Энциклопедия языка СИ / Я. Белецкий. – Москва: Мир, 1992. – 687 с.
2. Дьюхарст С. Программирование на C++ / С. Дьюхарст, К. Старк. – Київ: ДіаСофт, 1993. – 272 с.
3. Уэйт М. Язык Си. Руководство для начинающих / М. Уэйт, С. Прата, Д. Мартин. – Москва: Мир, 1988. – 512 с.
4. Страуструп Б. Язык программирования C++ / Б. Страуструп. – Київ: ДіаСофт, 1993. – 1213 с.
5. Шпак З.Я. Програмування мовою С – Львів: Оріяна-Нова, 2006. – 432с.
6. Культин Н. Б. C/C++ в задачах и примерах / Н. Б. Культин. – Санкт-Петербург: БХВ-Петербург, 2005. – 288 с.
7. Шилдт Г. Полный справочник по С++ / Герберт Шилдт. – Москва: Издательский дом "Вильямс", 2006. – 800 с. – (4).
8. Вайнер Р. Пінсон Л. С++ изнутри / Вайнер Р. Пінсон Л. – Київ: “ДіаСофт”, 1993. – 304 с.
9. Джамаса К.. Учимся програмировать на языке С++ / Джамаса К. – Москва: Мир, 1997. – 320 с.

# ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ

#include <iostream>

#include <Windows.h>

#include <stdio.h>

#include <cstdlib>

#include "PlaceShip.h"

#include "Battle.h"

using namespace std;

int main()

{

int m = 0;

int ship, horCoordShip, verCoordShip, Orient;

char\*\* pole = new char\* [10];

for (int i = 0; i < 10; i++)

pole[i] = new char[10];

ppolef(pole);

//--------- умовне поле для комп'ютера в невидимому варiантi ----------------

char\*\* ar = new char\* [13];

for (int i = 0; i < 13; i++)

ar[i] = new char[13];

for (int i = 0; i < 13; i++)

{

for (int j = 0; j < 13; j++)

{

ar[i][j] = -2;

}

}

//-------------------масив користувацького видимого поля----------------------

char\*\* ar\_user = new char\* [13];

for (int i = 0; i < 13; i++)

ar\_user[i] = new char[13];

for (int i = 0; i < 13; i++)

for (int j = 0; j < 13; j++)

{

ar\_user[i][j] = -2;

}

cout << endl;

//------------------вибираємо продовження гри або ж почати нову гру--------------

int k;

int t;

char\* str = new char[101];

char\* st = new char[101];

setlocale(LC\_ALL, "Ukrainian");

cout << "Для продовження розпочатої гри виберiть 1\n";

cout << "Для початку нової гри виберiть 2\n";

setlocale(LC\_ALL, ".OCP");

cin >> t;

switch (t)

{

case 1:

readuser(ar\_user);

aruserpole(ar\_user);

break;

case 2:

setlocale(LC\_ALL, "Ukrainian");

cout << "Виберiть варiант розстановки кораблiв: автоматичний або ручний\n";

cout << "Якщо автоматичний, виберiть - 1 \n";

cout << "Якщо ручна розстановка, виберiть - 0 \n";

setlocale(LC\_ALL, ".OCP");

cin >> k;

//------------------------------------------------розстановка кораблiв користувачем---------------------------------

//------------------------------------------------автоматична---------------------------------------

setlocale(LC\_ALL, "Ukrainian");

upload();

setlocale(LC\_ALL, ".OCP");

switch (k)

{

case 1:

// розмiщення 4-палубного корабля

do

{

int testp = 0;

int count = 0;

int ship = 4;

Orient = chooseorient(k);

horCoordShip = HorCoord(k);

verCoordShip = VerCoord(k);

do

{

testp += test(ar\_user, horCoordShip, verCoordShip, Orient, count, ship);

count++;

} while (count < 4);

if (testp == 0)

{

shipplace(ar\_user, horCoordShip, verCoordShip, Orient, ship);

}

else if (testp > 0) continue;

int sum = amountshipp(ar\_user);

if (sum == 4) break;

} while (1);

// розмiщення 3-палубного корабля

do

{

int testp = 0;

int count = 0;

int ship = 3;

Orient = chooseorient(k);

horCoordShip = HorCoord(k);

verCoordShip = VerCoord(k);

do

{

testp += test(ar\_user, horCoordShip, verCoordShip, Orient, count, ship);

count++;

} while (count < 3);

if (testp == 0)

{

shipplace(ar\_user, horCoordShip, verCoordShip, Orient, ship);

}

else if (testp > 0) continue;

int sum = amountshipp(ar\_user);

if (sum == 10) break;

} while (1);

// розмiщення 2-палубного корабля

do

{

int testp = 0;

int count = 0;

int ship = 2;

Orient = chooseorient(k);

horCoordShip = HorCoord(k);

verCoordShip = VerCoord(k);

do

{

testp += test(ar\_user, horCoordShip, verCoordShip, Orient, count, ship);

count++;

} while (count < 2);

if (testp == 0)

{

shipplace(ar\_user, horCoordShip, verCoordShip, Orient, ship);

}

else if (testp > 0) continue;

int sum = amountshipp(ar\_user);

if (sum == 16) break;

} while (1);

// розмiщення 1-палубного корабля

do

{

int testp = 0;

horCoordShip = onehorCoord();

verCoordShip = oneverCoord();

testp += testone(horCoordShip, verCoordShip, ar\_user);

if (testp == 0)

{

placeone(horCoordShip, verCoordShip, ar\_user);

}

else if (testp > 0) continue;

int sum = amountshipp(ar\_user);

if (sum == 18) break;

} while (1);

writeuser(ar\_user);

readuser(ar\_user);

aruserpole(ar\_user);

break;

//---------------------------------------------ручна розстановка-------------------------------------------

case 0:

setlocale(LC\_ALL, "Ukrainian");

cout << " Перед вами поле. Необхiдно здiйснити розстановку кораблiв. " << endl;

setlocale(LC\_ALL, ".OCP");

aruserpole(ar\_user);

do

{

do

{

ship = chooseship(); // вибiр типу корабля

if (ship == 0) cout << " Error. Try again." << endl;

else break;

} while (1);

switch (ship)

{

case 4:

do

{

int count = 0;

int testp = 0;

Orient = chooseorient(k); // вибiр орiєнтацiї корабля

verCoordShip = VerCoord(k); // вертикальна координата

horCoordShip = HorCoord(k); // горизонтальна координата

do

{

testp += test(ar\_user, horCoordShip, verCoordShip, Orient, count, ship); //перевiрка можливостi розмiщення всiх пабул обраного корабля

count++;

} while (count < 4);

if (testp > 0) continue;

else if (testp == 0) break;

} while (1);

shipplace(ar\_user, horCoordShip, verCoordShip, Orient, ship);

writeuser(ar\_user);

readuser(ar\_user);

system("cls");

aruserpole(ar\_user);

break;

case 3:

do

{

int count = 0;

int testp = 0;

Orient = chooseorient(k); // вибiр орiєнтацiї корабля

verCoordShip = VerCoord(k); // вертикальна координата

horCoordShip = HorCoord(k); //горизонтальна координата

do

{

testp += test(ar\_user, horCoordShip, verCoordShip, Orient, count, ship); //перевiрка можливостi розмiщення всiх пабул обраного корабля

count++;

} while (count < 3);

if (testp > 0) continue;

else if (testp == 0) break;

} while (1);

shipplace(ar\_user, horCoordShip, verCoordShip, Orient, ship);

writeuser(ar\_user);

readuser(ar\_user);

system("cls");

aruserpole(ar\_user);

break;

case 2:

do

{

int count = 0;

int testp = 0;

Orient = chooseorient(k); // вибiр орiєнтацiї корабля

verCoordShip = VerCoord(k); // вертикальна координата

horCoordShip = HorCoord(k); //горизонтальна координата

do

{

testp += test(ar\_user, horCoordShip, verCoordShip, Orient, count, ship); //перевiрка можливостi розмiщення всiх пабул обраного корабля

count++;

} while (count < 2);

if (testp > 0) continue;

else if (testp == 0) break;

} while (1);

shipplace(ar\_user, horCoordShip, verCoordShip, Orient, ship);

writeuser(ar\_user);

readuser(ar\_user);

system("cls");

aruserpole(ar\_user);

break;

case 1:

do

{

int count = 0;

int testp = 0;

verCoordShip = VerCoord(k); // вертикальна координата

horCoordShip = HorCoord(k); //горизонтальна координата

testp += test(ar\_user, horCoordShip, verCoordShip, Orient, count, ship); //перевiрка можливостi розмiщення всiх пабул обраного корабля

if (testp > 0) continue;

else if (testp == 0) break;

} while (1);

shipplace(ar\_user, horCoordShip, verCoordShip, Orient, ship);

writeuser(ar\_user);

readuser(ar\_user);

system("cls");

aruserpole(ar\_user);

break;

}

int summa = amountshipp(ar\_user);

if (summa == 18) break;

m++;

} while (1);

cout << " You've done it! " << endl;

break;

}

//-----------------автоматична разстановка кораблiв комп'ютера-----------

setlocale(LC\_ALL, "Ukrainian");

upload();

setlocale(LC\_ALL, ".OCP");

do

{

int testp = 0;

int counter = 0;

int Ship = 4;

Orient = orientKomp();

horCoordShip = HorCoordK();

verCoordShip = VerCoordK();

do

{

testp += testK(ar, horCoordShip, verCoordShip, Orient, counter, Ship);

counter++;

} while (counter < 4);

if (testp == 0)

{

shipplaceK(ar, horCoordShip, verCoordShip, Orient, Ship);

}

else if (testp > 0) continue;

int sum = amountshippK(ar);

if (sum == 4) break;

} while (1);

do

{

int testp = 0;

int counter = 0;

int Ship = 3;

Orient = orientKomp();

horCoordShip = HorCoordK();

verCoordShip = VerCoordK();

do

{

testp += testK(ar, horCoordShip, verCoordShip, Orient, counter, Ship);

counter++;

} while (counter < 3);

if (testp == 0)

{

shipplaceK(ar, horCoordShip, verCoordShip, Orient, Ship);

}

else if (testp > 0) continue;

int sum = amountshippK(ar);

if (sum == 10) break;

} while (1);

do

{

int testp = 0;

int counter = 0;

int Ship = 2;

Orient = orientKomp();

horCoordShip = HorCoordK();

verCoordShip = VerCoordK();

do

{

testp += testK(ar, horCoordShip, verCoordShip, Orient, counter, Ship);

counter++;

} while (counter < 2);

if (testp == 0)

{

shipplaceK(ar, horCoordShip, verCoordShip, Orient, Ship);

}

else if (testp > 0) continue;

int sum = amountshippK(ar);

if (sum == 16) break;

} while (1);

do

{

int testp = 0;

horCoordShip = onehorCoordK();

verCoordShip = oneverCoordK();

testp += testoneK(horCoordShip, verCoordShip, ar);

if (testp == 0)

{

placeoneK(horCoordShip, verCoordShip, ar);

}

else if (testp > 0) continue;

int sum = amountshippK(ar);

if (sum == 18) break;

} while (1);

writeK(ar);

readK(ar);

fppolef(pole);

/\*arpole (ar);\*/

break;

}

cout << endl;

//---------------------------------------органiзацiя бою------------------------------

//----------------- сформуємо видиме для користувача поле -------------------------

char horCoord = 0;

int horCoor = 0;

int verCoord = 0;

int userHit = 0;

//записуємо поле в окремий файл

//---------------------------ходи користувача-------------------------------------------

do

{

readuser(ar\_user);

sppolef(pole);

while (1) //хiд користувача

{

setlocale(LC\_ALL, "Ukrainian");

horCoord = hod\_userhk();

verCoord = hod\_uservk();

cout << endl;

userHit = userhit(horCoord, verCoord, ar, pole); //перевiрка ходу на попадання

setlocale(LC\_ALL, ".OCP");

if (userHit == 0)

{

sppolef(pole);

cout << endl;

pole[(int)horCoord - 65][verCoord - 1] = '\*';

fppolef(pole);

system("cls");

setlocale(LC\_ALL, "Ukrainian");

cout << " Не попали! Спробуйте наступного разу! " << endl;

setlocale(LC\_ALL, ".OCP");

break;

}

else if (userHit == 1)

{

sppolef(pole);

cout << endl;

pole[(int)horCoord - 65][verCoord - 1] = 'x';

fppolef(pole);

system("cls");

setlocale(LC\_ALL, "Ukrainian");

cout << " Ви попали! Ходiть ще раз. " << endl;

setlocale(LC\_ALL, ".OCP");

continue;

}

else if (userHit == -1)

{

setlocale(LC\_ALL, "Ukrainian");

cout << " Сюди вже стрiляли. Спробуйте ще раз! " << endl;

setlocale(LC\_ALL, ".OCP");

continue;

}

}

sppolef(pole);

int amountShip = amountship(pole); //перевiрка кiлькостi пiдбитих палуб

setlocale(LC\_ALL, "Ukrainian");

switch (amountShip)

{

case 1:

cout << " Ой ... Я, здається, програв. Вiтаю, ти - переможець!" << endl;

break;

case 0:

cout << " Граємо далi. Хiд за комп'ютером. " << endl;

break;

}

setlocale(LC\_ALL, ".OCP");

// ---------------------------------ходи комп'ютера ---------------------------------------

do

{

horCoor = hod\_komphk();

verCoord = hod\_kompvk();

cout << endl;

userHit = komphit(horCoor, verCoord, ar\_user); //перевiрка ходу на попадання

if (userHit == 0)

{

readuser(ar\_user);

cout << endl;

ar\_user[horCoor - 1][verCoord - 1] = '\*';

writeuser(ar\_user);

system("cls");

setlocale(LC\_ALL, "Ukrainian");

cout << " Не попав! Спробую наступного разу! " << endl;

setlocale(LC\_ALL, ".OCP");

break;

}

else if (userHit == 1)

{

readuser(ar\_user);

cout << endl;

ar\_user[horCoor - 1][verCoord - 1] = 'x';

writeuser(ar\_user);

system("cls");

setlocale(LC\_ALL, "Ukrainian");

cout << " Ура !!! Попав! Знову мiй хiд. " << endl;

setlocale(LC\_ALL, ".OCP");

continue;

}

else if (userHit == -1)

{

setlocale(LC\_ALL, "Ukrainian");

cout << "Ой ... Сюди вже стрiляв! Якщо не заперечуєш, спробую ще раз! " << endl;

system("cls");

cout << " Почекай. Я задумався. " << endl;

setlocale(LC\_ALL, ".OCP");

continue;

}

} while (1);

readuser(ar\_user);

aruserpole(ar\_user);

int amountShipus = amountSh(ar\_user);

setlocale(LC\_ALL, "Ukrainian");

switch (amountShipus)

{

case 1:

cout << " Ви програли. Менi дуже шкода." << endl;

break;

case 0:

cout << " Граємо далi. Твiй хiд, користувач. " << endl;

break;

}

setlocale(LC\_ALL, ".OCP");

if (amountShipus == 1 || amountShip == 1)

{

cout << " Game over! " << endl;

break;

}

} while (1);

delete[]str;

delete[]st;

for (int i = 0; i < 13; i++)

delete[]ar\_user[i];

delete[]ar\_user;

for (int i = 0; i < 13; i++)

delete[]ar[i];

delete[]ar;

return 0;

}

Лістинг файлу PlaceShip.h

#ifndef Place\_Ship

#define Place\_Ship

//-----------------структура для ручної розстановки кораблів-----------------

struct ship\_user

{

bool d; //орієнтації корабля - вертикально (0) або горизонтально (1)

char x; //горизонтальна координата х(рядок)

int y; //вертикальна координата у (стовпчик)

};

//-----------прототипи функцій---------------------------------

//---функції користувацької ручної і автоматичної розстановки кораблів------

void aruserpole(char\*\* ar\_user); // поле користувача

int chooseship(); // функція вибору типу корабля

int chooseorient(int& k); // функція вибору орієнтації корабля

int HorCoord(int& k); // функція введення горизонтальної координати

int VerCoord(int& k); // функція введення вертикальної координати

int test(char\*\* ar\_user, int& horCoordShip, int& verCoordShip, int& Orient, int& count, int& sh); // функція перевірки існування палуб

void shipplace(char\*\* ar\_user, int& horCoordShip, int& verCoordShip, int& Orient, int& sh); // функція розміщення корабля

int amountshipp(char\*\* ar\_user); // функція підрахунку палуб

void writeuser(char\*\* ar\_user); // функція запису в файл поля користувача

void readuser(char\*\* ar\_user); // функція зчитування з файлу поля користувача

int testone(int& horCoordShip, int& verCoordShip, char\*\* ar\_user); // функція перевірки однопалубника

void placeone(int& horCoordShip, int& verCoordShip, char\*\* ar\_user); //Функція розміщення однопалубника

int onehorCoord(); //функція введення горизонтальної координати однопалубника

int oneverCoord(); //функція введення вертикальної координати однопалубника

void upload(); //функція вивеження повідомлення про завантаження

//-------функції автоматичної розстановки для комп'ютера---------------------

void arpole(char\*\* ar); // поле комп'ютера

int orientKomp(); //функція вибору орієнтації корабля комп’ютера

int HorCoordK(); //функція генерації горизонтальної координати корабля комп’ютера

int VerCoordK(); //функція генерації вертикальної координати корабля комп’ютера

int testK(char\*\* ar, int& horCoordShip, int& verCoordShip, int& Orient, int& counter, int& Ship); //функція перевірки існування палуб

void shipplaceK(char\*\* ar, int& horCoordShip, int& verCoordShip, int& Orient, int& Ship); //функція розміщення корабля на полі комп’ютера

int amountshippK(char\*\* ar); //функція підрахунку палуб на полі комп’ютера

void writeK(char\*\* ar); //функція запису в файл поля комп’ютера

void readK(char\*\* ar); //функція зчитування з файлу поля комп’ютера

int testoneK(int& horCoordShip, int& verCoordShip, char\*\* ar); // функція перевірки однопалубника

void placeoneK(int& horCoordShip, int& verCoordShip, char\*\* ar); //Функція розміщення однопалубника

int onehorCoordK(); //функція генерації горизонтальної координати однопалубника

int oneverCoordK(); //функція генерації вертикальної координати однопалубника

#endif

Лістинг файлу PlaceShip.cpp

#include <iostream>

#include <cstdio>

#include <ctime>

#include "PlaceShip.h"

using namespace std;

ship\_user shipus;

//-------------------------------функція вивеження повідомлення про завантаження--------------------------

void upload()

{

cout << " Зачекайте, будь ласка. Відбувається завантаження......" << endl;

}

// ------------------------------ функція виведення умовного поля комп'ютера -------------------------

void arpole(char\*\* ar)

{

for (int i = 1; i < 11; i++)

{

for (int j = 1; j < 11; j++)

{

cout << ar[i][j]<<" ";

}

cout << endl;

}

}

//--------------------------------функція виведення умовного поля користувача---------------------------

void aruserpole(char\*\* ar\_user)

{

for (int i = 0; i < 2; i++)

cout << ' ';

for (int i = 0; i < 10; i++)

cout << i + 1;

cout << endl;

for (int i = 1; i < 11; i++)

{

cout << char(i + 64) << ' ';

for (int j = 1; j < 11; j++)

{

cout << ar\_user[i][j]<<" ";

}

cout << endl;

}

}

//---------------------------------функція вибору типу корабля---------------------------------

int chooseship()

{

setlocale(LC\_ALL, "Ukrainian");

int m;

cout << "Необхідно розташувати 10 кораблів. 1 - чотирипалубний, 2-трьохпалубні, 3-двохпалубні і 4 - однопалубні\n";

cout << "Необхідно ввести координати для окремої палуби корабля.\n";

cout << "Виберіть 4, щоб розташувати корабель з 4-а палубами\n" << endl;

cout << "Виберіть 3, щоб розташувати корабель з 3-а палубами\n" << endl;

cout << "Виберіть 2, щоб розташувати корабель з 1-а палубами\n" << endl;

cout << "Виберіть 1, щоб розташувати однопалубний корабель\n" << endl;

cin >> m;

setlocale(LC\_ALL, ".OCP");

if (m == 4 || m == 3 || m == 2 || m == 1)

return m;

else return 0;

}

//---------------------------------функція вибору орієнтації корабля-----------------------------

int chooseorient(int& k)

{

if (k == 1)

{

srand(time(NULL));

return rand() % 2;

}

else if (k == 0)

{

setlocale(LC\_ALL, "Ukrainian");

cout << "Виберіть 1, якщо ваш корабель вертикальний\n" << endl;

cout << "Виберіть 0, якщо ваш корабель горизонтальний\n" << endl;

cin >> shipus.d;

setlocale(LC\_ALL, ".OCP");

}

return shipus.d;

}

//---------------------------------функція введення горизонтальної координати------------------------

int HorCoord(int& k)

{

srand(time(NULL));

if (k == 1)

{

return rand() % 10 + 1;

}

else if (k == 0)

{

do

{

setlocale(LC\_ALL, "Ukrainian");

cout << "Введіть горизонтальну координату першої палуби латинські сівмоли (A - J) ->" << endl;

setlocale(LC\_ALL, ".OCP");

cin >> shipus.x;

if (shipus.x >= 65 && shipus.x <= 74) break;

else cout << " Error. Try again." << endl;

} while (1);

return (int)shipus.x - 64;

}

}

//---------------------------------функція введення вертикальної координати-----------------------

int VerCoord(int& k)

{

srand(time(NULL));

if (k == 1)

{

return rand() % 10 + 1;

}

else if (k == 0)

{

do

{

setlocale(LC\_ALL, "Ukrainian");

cout << "Введіть вертикальну координату першої палуби число від 1 до 10 ->" << endl;

cin >> shipus.y;

setlocale(LC\_ALL, ".OCP");

if (shipus.y > 10 || shipus.y < 1) cout << " Error. Try again." << endl;

else break;

} while (1);

return shipus.y;

}

}

//---------------------------------функція перевірки існування палуб----------------------------

int test(char\*\* ar\_user, int& horCoordShip, int& verCoordShip, int& Orient, int& count, int& sh)

{

int sum = 0;

switch (Orient)

{

case 1:

switch (sh)

{

case 4:

if (horCoordShip == 10 || horCoordShip == 9 || horCoordShip == 8)

{

for (int i = horCoordShip + 1 - count; i > horCoordShip - 2 - count; i--)

for (int j = verCoordShip - 1; j < verCoordShip + 2; j++)

{

int d = (ar\_user[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

else

{

for (int i = horCoordShip - 1 + count; i < horCoordShip + 2 + count; i++)

for (int j = verCoordShip - 1; j < verCoordShip + 2; j++)

{

int d = (ar\_user[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

break;

case 3:

if (horCoordShip == 10 || horCoordShip == 9)

{

for (int i = horCoordShip + 1 - count; i > horCoordShip - 2 - count; i--)

for (int j = verCoordShip - 1; j < verCoordShip + 2; j++)

{

int d = (ar\_user[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

else

{

for (int i = horCoordShip - 1 + count; i < horCoordShip + 2 + count; i++)

for (int j = verCoordShip - 1; j < verCoordShip + 2; j++)

{

int d = (ar\_user[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

break;

case 2:

if (horCoordShip == 10)

{

for (int i = horCoordShip + 1 - count; i > horCoordShip - 2 - count; i--)

for (int j = verCoordShip - 1; j < verCoordShip + 2; j++)

{

int d = (ar\_user[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

else

{

for (int i = horCoordShip - 1 + count; i < horCoordShip + 2 + count; i++)

for (int j = verCoordShip - 1; j < verCoordShip + 2; j++)

{

int d = (ar\_user[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

break;

}

break;

case 0:

switch (sh)

{

case 4:

if (horCoordShip == 10 || horCoordShip == 9 || horCoordShip == 8)

{

for (int i = horCoordShip - 1; i < horCoordShip + 2; i++)

for (int j = verCoordShip + 1 - count; j < verCoordShip - 2 - count; j--)

{

int d = (ar\_user[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

else

{

for (int i = horCoordShip - 1; i < horCoordShip + 2; i++)

for (int j = verCoordShip - 1 + count; j < verCoordShip + 2 + count; j++)

{

int d = (ar\_user[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

break;

case 3:

if (horCoordShip == 10 || horCoordShip == 9)

{

for (int i = horCoordShip - 1; i < horCoordShip + 2; i++)

for (int j = verCoordShip + 1 - count; j < verCoordShip - 2 - count; j--)

{

int d = (ar\_user[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

else

{

for (int i = horCoordShip - 1; i < horCoordShip + 2; i++)

for (int j = verCoordShip - 1 + count; j < verCoordShip + 2 + count; j++)

{

int d = (ar\_user[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

break;

case 2:

if (horCoordShip == 10)

{

for (int i = horCoordShip - 1; i < horCoordShip + 2; i++)

for (int j = verCoordShip + 1 - count; j < verCoordShip - 2 - count; j--)

{

int d = (ar\_user[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

else

{

for (int i = horCoordShip - 1; i < horCoordShip + 2; i++)

for (int j = verCoordShip - 1 + count; j < verCoordShip + 2 + count; j++)

{

int d = (ar\_user[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

break;

}

break;

}

}

//---------------------------------функція розміщення корабля--------------------------

void shipplace(char\*\* ar\_user, int& horCoordShip, int& verCoordShip, int& Orient, int& sh)

{

switch (Orient)

{

case 1:

switch (sh)

{

case 4:

if (horCoordShip == 10 || horCoordShip == 9 || horCoordShip == 8)

{

for (int i = horCoordShip; i > horCoordShip - 4; i--)

ar\_user[i][verCoordShip] = 127;

}

else

{

for (int i = horCoordShip; i < horCoordShip + 4; i++)

ar\_user[i][verCoordShip] = 127;

}

break;

case 3:

if (horCoordShip == 10 || horCoordShip == 9)

{

for (int i = horCoordShip; i > horCoordShip - 3; i--)

ar\_user[i][verCoordShip] = 127;

}

else

{

for (int i = horCoordShip; i < horCoordShip + 3; i++)

ar\_user[i][verCoordShip] = 127;

}

break;

case 2:

if (horCoordShip == 10)

{

for (int i = horCoordShip; i > horCoordShip - 2; i--)

ar\_user[i][verCoordShip] = 127;

}

else

{

for (int i = horCoordShip; i < horCoordShip + 2; i++)

ar\_user[i][verCoordShip] = 127;

}

break;

}

break;

case 0:

switch (sh)

{

case 4:

if (verCoordShip == 10 || verCoordShip == 9 || verCoordShip == 8)

{

for (int i = verCoordShip; i > verCoordShip - 4; i--)

ar\_user[horCoordShip][i] = 127;

}

else

{

for (int i = verCoordShip; i < verCoordShip + 4; i++)

ar\_user[horCoordShip][i] = 127;

}

break;

case 3:

if (verCoordShip == 10 || verCoordShip == 9)

{

for (int i = verCoordShip; i > verCoordShip - 3; i--)

ar\_user[horCoordShip][i] = 127;

}

else

{

for (int i = verCoordShip; i < verCoordShip + 3; i++)

ar\_user[horCoordShip][i] = 127;

}

break;

case 2:

if (verCoordShip == 10)

{

for (int i = verCoordShip; i > verCoordShip - 2; i--)

ar\_user[horCoordShip][i] = 127;

}

else

{

for (int i = verCoordShip; i < verCoordShip + 2; i++)

ar\_user[horCoordShip][i] = 127;

}

break;

}

break;

}

}

//---------------------------------функція підрахунку палуб ------------------------------

int amountshipp(char\*\* ar\_user)

{

int sum = 0;

for (int i = 1; i < 11; i++)

for (int j = 1; j < 11; j++)

{

int k = (ar\_user[i][j] == 127) ? 1 : 0;

sum += k;

}

return sum;

}

//-------------------------------функція запису в файл поля користувача-------------------------

void writeuser(char\*\* ar\_user)

{

FILE\* f;

f = fopen("seabattleuser.txt", "w");

if (f != NULL)

{

for (int i = 0; i < 13; i++)

for (int j = 0; j < 13; j++)

fputc(ar\_user[i][j], f);

}

else cout << " Error. I can't open the file." << endl;

fclose(f);

}

//---------------------------------функція зчитування з файлу поля користувача----------------------

void readuser(char\*\* ar\_user)

{

FILE\* f;

f = fopen("seabattleuser.txt", "r");

for (int i = 0; i < 13; i++)

{

for (int j = 0; j < 13; j++)

ar\_user[i][j] = fgetc(f);

}

fclose(f);

}

int testone(int& horCoordShip, int& verCoordShip, char\*\* ar\_user)

{

int sum = 0;

for (int i = horCoordShip - 1; i < horCoordShip + 2; i++)

for (int j = verCoordShip - 1; j < verCoordShip + 2; j++)

{

int d = (ar\_user[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

void placeone(int& horCoordShip, int& verCoordShip, char\*\* ar\_user)

{

ar\_user[horCoordShip][verCoordShip] = 127;

}

int onehorCoord()

{

srand(time(NULL));

return rand() % 10 + 1;

}

int oneverCoord()

{

srand(time(NULL));

return rand() % 10 + 1;

}

//--------------------------------------автоматична розстановка кораблів для комп'ютера-------------------------

//--------------------------------------орієнтація корабля----------------------------------------

int orientKomp()

{

srand(time(NULL));

return rand() % 2;

}

//-------------------------------------горизонтальна координата---------------------------------------

int HorCoordK()

{

srand(time(NULL));

return rand() % 10 + 1;

}

//------------------------------------вертикальна координата---------------------------------------------

int VerCoordK()

{

srand(time(NULL));

return rand() % 10 + 1;

}

//------------------------------------функції для однопалубного корабля---------------------------------------

void placeoneK(int& horCoordShip, int& verCoordShip, char\*\* ar)

{

ar[horCoordShip][verCoordShip] = 127;

}

int onehorCoordK()

{

srand(time(NULL));

return rand() % 10 + 1;

}

int oneverCoordK()

{

srand(time(NULL));

return rand() % 10 + 1;

}

//---------------------------------тестування можливості розміщення однопалубного корабля---------------------------------------

int testoneK(int& horCoordShip, int& verCoordShip, char\*\* ar)

{

int sum = 0;

for (int i = horCoordShip - 1; i < horCoordShip + 2; i++)

for (int j = verCoordShip - 1; j < verCoordShip + 2; j++)

{

int d = (ar[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

//-------------------------------------перевірка існування палуб------------------------------------------

int testK(char\*\* ar, int& horCoordShip, int& verCoordShip, int& Orient, int& counter, int& Ship)

{

int sum = 0;

switch (Orient)

{

case 1:

switch (Ship)

{

case 4:

if (horCoordShip == 10 || horCoordShip == 9 || horCoordShip == 8)

{

for (int i = horCoordShip + 1 - counter; i > horCoordShip - 2 - counter; i--)

for (int j = verCoordShip - 1; j < verCoordShip + 2; j++)

{

int d = (ar[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

else

{

for (int i = horCoordShip - 1 + counter; i < horCoordShip + 2 + counter; i++)

for (int j = verCoordShip - 1; j < verCoordShip + 2; j++)

{

int d = (ar[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

break;

case 3:

if (horCoordShip == 10 || horCoordShip == 9)

{

for (int i = horCoordShip + 1 - counter; i > horCoordShip - 2 - counter; i--)

for (int j = verCoordShip - 1; j < verCoordShip + 2; j++)

{

int d = (ar[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

else

{

for (int i = horCoordShip - 1 + counter; i < horCoordShip + 2 + counter; i++)

for (int j = verCoordShip - 1; j < verCoordShip + 2; j++)

{

int d = (ar[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

break;

case 2:

if (horCoordShip == 10)

{

for (int i = horCoordShip + 1 - counter; i > horCoordShip - 2 - counter; i--)

for (int j = verCoordShip - 1; j < verCoordShip + 2; j++)

{

int d = (ar[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

else

{

for (int i = horCoordShip - 1 + counter; i < horCoordShip + 2 + counter; i++)

for (int j = verCoordShip - 1; j < verCoordShip + 2; j++)

{

int d = (ar[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

break;

}

break;

case 0:

switch (Ship)

{

case 4:

if (horCoordShip == 10 || horCoordShip == 9 || horCoordShip == 8)

{

for (int i = horCoordShip - 1; i < horCoordShip + 2; i++)

for (int j = verCoordShip + 1 - counter; j < verCoordShip - 2 - counter; j--)

{

int d = (ar[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

else

{

for (int i = horCoordShip - 1; i < horCoordShip + 2; i++)

for (int j = verCoordShip - 1 + counter; j < verCoordShip + 2 + counter; j++)

{

int d = (ar[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

break;

case 3:

if (horCoordShip == 10 || horCoordShip == 9)

{

for (int i = horCoordShip - 1; i < horCoordShip + 2; i++)

for (int j = verCoordShip + 1 - counter; j < verCoordShip - 2 - counter; j--)

{

int d = (ar[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

else

{

for (int i = horCoordShip - 1; i < horCoordShip + 2; i++)

for (int j = verCoordShip - 1 + counter; j < verCoordShip + 2 + counter; j++)

{

int d = (ar[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

break;

case 2:

if (horCoordShip == 10)

{

for (int i = horCoordShip - 1; i < horCoordShip + 2; i++)

for (int j = verCoordShip + 1 - counter; j < verCoordShip - 2 - counter; j--)

{

int d = (ar[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

else

{

for (int i = horCoordShip - 1; i < horCoordShip + 2; i++)

for (int j = verCoordShip - 1 + counter; j < verCoordShip + 2 + counter; j++)

{

int d = (ar[i][j] == 127) ? 1 : 0;

sum += d;

}

if (sum > 0) return 1;

else if (sum == 0) return 0;

}

break;

}

break;

}

}

//-----------------------------------------------функція растановкой кораблів комп'ютера---------------------------------

void shipplaceK(char\*\* ar, int& horCoordShip, int& verCoordShip, int& Orient, int& Ship)

{

switch (Orient)

{

case 1:

switch (Ship)

{

case 4:

if (horCoordShip == 10 || horCoordShip == 9 || horCoordShip == 8)

{

for (int i = horCoordShip; i > horCoordShip - 4; i--)

ar[i][verCoordShip] = 127;

}

else

{

for (int i = horCoordShip; i < horCoordShip + 4; i++)

ar[i][verCoordShip] = 127;

}

break;

case 3:

if (horCoordShip == 10 || horCoordShip == 9)

{

for (int i = horCoordShip; i > horCoordShip - 3; i--)

ar[i][verCoordShip] = 127;

}

else

{

for (int i = horCoordShip; i < horCoordShip + 3; i++)

ar[i][verCoordShip] = 127;

}

break;

case 2:

if (horCoordShip == 10)

{

for (int i = horCoordShip; i > horCoordShip - 2; i--)

ar[i][verCoordShip] = 127;

}

else

{

for (int i = horCoordShip; i < horCoordShip + 2; i++)

ar[i][verCoordShip] = 127;

}

break;

}

break;

case 0:

switch (Ship)

{

case 4:

if (verCoordShip == 10 || verCoordShip == 9 || verCoordShip == 8)

{

for (int i = verCoordShip; i > verCoordShip - 4; i--)

ar[horCoordShip][i] = 127;

}

else

{

for (int i = verCoordShip; i < verCoordShip + 4; i++)

ar[horCoordShip][i] = 127;

}

break;

case 3:

if (verCoordShip == 10 || verCoordShip == 9)

{

for (int i = verCoordShip; i > verCoordShip - 3; i--)

ar[horCoordShip][i] = 127;

}

else

{

for (int i = verCoordShip; i < verCoordShip + 3; i++)

ar[horCoordShip][i] = 127;

}

break;

case 2:

if (verCoordShip == 10)

{

for (int i = verCoordShip; i > verCoordShip - 2; i--)

ar[horCoordShip][i] = 127;

}

else

{

for (int i = verCoordShip; i < verCoordShip + 2; i++)

ar[horCoordShip][i] = 127;

}

break;

}

break;

}

}

int amountshippK(char\*\* ar)

{

int sum = 0;

for (int i = 1; i < 11; i++)

for (int j = 1; j < 11; j++)

{

int k = (ar[i][j] == 127) ? 1 : 0;

sum += k;

}

return sum;

}

//---------------------------------------------функції запису і зчитування з файлу поля комп'ютера---------------------------------------

void writeK(char\*\* ar)

{

FILE\* f;

f = fopen("seabattle.txt", "w");

if (f != NULL)

{

for (int i = 0; i < 13; i++)

for (int j = 0; j < 13; j++)

fputc(ar[i][j], f);

}

else cout << " Error. I can't open the file." << endl;

fclose(f);

}

void readK(char\*\* ar)

{

FILE\* f;

f = fopen("seabattle.txt", "r");

for (int i = 0; i < 13; i++)

{

for (int j = 0; j < 13; j++)

ar[i][j] = fgetc(f);

}

fclose(f);

}

Лістинг файлу Battle.h

#ifndef Battle\_s

#define Battle\_s

void ppolef(char\*\* pole); //функція генерації початкового видимого поля комп'ютера

void fppolef(char\*\* pole); //функція запису у файл ходів користувача

void sppolef(char\*\* pole); //функція зчитування з файлу ходів користувача

int amountship(char\*\* pole); //функція підрахунку підбитих користувачем кораблів

char hod\_userhk(); //функція введення горизонтальної координати пострілу користувача

int hod\_uservk(); //функція введення вертикальної координати пострілу користувача

//функція перевірки попадання користувачем

int userhit(char& horCoord, int& verCoord, char\*\* arr, char\*\* pole);

int hod\_komphk(); //функція генерації горизонтальної координати пострілу комп'ютера

int hod\_kompvk(); //функція генерації вертикальної координати пострілу комп'ютера

int komphit(int& horCoor, int& verCoord, char\*\* ar\_user); //перевірка попадання комп'ютера

int amountSh(char\*\* ar\_user); //функція підрахунку підбитих комп’ютером кораблів

int choosehod(); // функція вибора ходу

#endif

Лістинг файлу Battle.cpp

#include <iostream>

#include <stdlib.h>

#include <stdio.h>

#include <ctime>

#include "Battle.h"

using namespace std;

//----------------------------------------функції користувача--------------------------

//----------------------------------------функція генерації початкового видимого поля комп'ютера------------------------

void ppolef ( char \*\*pole )

{

//заповнимо поле

for ( int i = 0; i < 10; i ++)

for ( int j = 0; j < 10; j ++ )

pole [i][j] = -2;

}

//----------------------------------функція запису у файл ходів користувача------------------------------

void fppolef (char \*\*pole) // функция записи поля

{

FILE \*f;

f = fopen ("seabattle1.txt", "w");

if (f != NULL)

{

for ( int i = 0; i < 10; i ++)

for( int j = 0; j < 10; j ++)

fputc (pole [i][j], f);

}

else

{

cout << " Помилка. Не можу відкрити файл. " << endl;

}

fclose (f);

}

//----------------------------------функція зчитування з файлу ходів користувача------------------------------

void sppolef (char \*\*pole)

{

FILE \*f;

f = fopen ("seabattle1.txt", "r");

if (f!=NULL)

{

for ( int i = 0; i < 10; i ++ )

{

for( int j = 0; j < 10; j ++)

{

pole[i][j] = fgetc(f);

}

}

}

else

{

cout << " Помилка. Не можу відкрити файл. " << endl;

}

fclose (f);

for ( int i = 0; i < 2; i ++)

cout << ' ';

for ( int i = 0; i < 10; i ++)

cout << i + 1;

cout << endl;

for ( int i = 0; i < 10; i ++)

{

cout << char(i + 65) << ' ';

for ( int j = 0; j < 10; j ++)

{

cout <<pole [i][j];

}

cout <<endl;

}

}

//--------------------------------------функція підрахунку підбитих користувачем кораблів-----------------------------

int amountship (char \*\*pole)

{

int count = 0;

int sum = 0;

for ( int i = 0; i < 10; i ++)

for ( int j = 0; j < 10; j ++)

{

count = ( pole [i][j] == 'x')? 1:0;

sum += count;

}

if (sum == 18) return 1;

if (sum < 18) return 0;

}

//-----------------------------------------функція введення горизонтальної координати пострілу користувача------------------------------

char hod\_userhk ( )

{

char a;

cout << " Введіть координати " << endl;

cout << " Для горизонтальної координати введіть букву латинського алфавіту (від A до K - в верхньому регістрі) " << endl;

while (1)

{

cin >> a;

if (((int) a < 65) || ((int) a > 74) )

{

cout << " Помилка. Ви ввели невірну координату! Спробуйте ще раз. " << endl;

continue;

}

else

{

cout << " Горизонтальна координата - " << a;

break;

}

}

return a;

}

//----------------------------------------функція введення горизонтальної координати пострілу користувача--------------------------

int hod\_uservk ()

{

int k;

cout << " Для вертикальної координати введіть цифру (від 1 до 10) " << endl;

while (1)

{

cin >> k;

if ( k < 1 || k > 10)

{

cout << " Помилка. Ви ввели невірну координату! Спробуйте ще раз. " << endl;

continue;

}

else

{

cout << " Вертикальна координата - " << k << endl;

break;

}

}

return k;

}

//--------------------------------------------функція перевірки попадання-------------------------------------------------------

int userhit ( char &horCoord, int &verCoord, char \*\* ar,char \*\*pole) //проверка хода пользователя насчет попадания

{

if ( ar [(int)horCoord - 65][verCoord - 1] == 127)

{

return 1;

}

else if ( pole [(int)horCoord - 65][verCoord - 1] == '\*' || pole [(int)horCoord - 65][verCoord - 1] == 'x')

{

return -1;

}

else

{

return 0;

}

}

//--------------------------------функції комп'ютера---------------------------------------

//--------------------------------функція генерації горизонтальної координати пострілу комп'ютера--------------------

int hod\_komphk ()

{

int ar\_time [100];

srand (time (NULL));

for (int i = 0; i < 99; i++)

ar\_time [i] = rand ()%10 + 1;

return ar\_time [rand ()%100];

}

//----------------------------------функція генерації вертикальної координати пострілу комп'ютера-----------------------

int hod\_kompvk ()

{

int ar\_time [100];

srand (time (NULL));

for (int i = 0; i < 99; i ++)

ar\_time [i] = rand ()%10 + 1;

return ar\_time [rand ()%100];

}

//------------------------------------ перевірка попадання комп'ютера-----------------------------------

int komphit (int &horCoor, int &verCoord, char \*\*ar\_user)

{

if ( ar\_user [horCoor - 1][verCoord - 1] == 127)

{

return 1;

}

else if ( ar\_user [horCoor - 1][verCoord - 1] == '\*' || ar\_user [horCoor - 1][verCoord - 1] == 'x')

{

return -1;

}

else

{

return 0;

}

}

int amountSh (char \*\*ar\_user)

{

int count = 0;

int sum = 0;

for ( int i = 0; i < 10; i ++)

for ( int j = 0; j < 10; j ++)

{

count = ( ar\_user [i][j] == 'x')? 1:0;

sum += count;

}

if (sum == 18) return 1;

if (sum < 18) return 0;

}