МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

КУРСОВА РОБОТА

з дисципліни «Основи програмування. частина 2»

на тему

Розробка програмного продукту «Сапер»

Виконала студентка

І курсу групи КП-42

Дітріх Валерія

Керівник роботи

_____, _____

Оцінка

_____

(дата, підпис)

КИЇВ 2025

Зміст

1. Короткий опис теми та призначення програми

Моя програма є консольною реалізацією гри «Сапер». Основне призначення цієї програми полягає в відтворенні повноцінного ігрового досвіду «Сапера» в консольному вікні. У програмі реалізовані всі ключові механіки оригінальної гри: можливість відкривати клітинки та встановлювати чи знімати прапорці на потенційно замінованих клітинках. Перемога в грі досягається, коли гравець успішно відкриває всі клітинки, що не містять бомб. Розташування бомб на ігровому полі завжди випадкове, через що кожна нова партія гри є унікальна. Крім базових функцій, були додані різні рівні складності: легкий, середній, складний та користувацький. В користувацькому режимі гравцеві буде надана можливість встановити розміри поля та кількість бомб власноруч. Керування в грі реалізовано за допомогою стрілочок для переміщення курсора, клавіші “F” для встановлення/зняття прапорця та клавіші “Enter” для вибору клітинки. Моїм завданням було створити повністю функціональну консольну версію гри «Сапер», яка б демонструвала моє розуміння принципів ігрової логіки, обробки користувацького введення та ефективного виведення інформації в консоль.

2. Архітектура програми

2.1. Компоненти програми

Class Sapper - Центральний клас, що зберігає основні дані та поточний стан ігрового поля, а також базові параметри гри.

Class Gameplay - Відповідає за основну ігрову логіку, обробку дій гравця та керування перебігом гри.

Class ConsoleOutput - Керує візуальним представленням ігрового поля у консолі.

Class InitializationMethods - Містить методи для ініціалізації гри.

Class Levels - Обробляє логіку різних рівнів складності та налаштувань користувацького поля.

Class Menu - Відповідає за відображення головного меню та обробку вибору користувача в ньому.

2.2. Функції компонентів

Class Sapper: Зберігає ключові дані гри, такі як: розмір поля, поточні координати курсора, кількість бомб та доступних прапорців. Також містить логічні масиви для відстеження стану кожної клітинки та двовимірний масив для представлення вмісту клітинок. Його конструктор використовується для ініціалізації та створення ігрового поля відповідних розмірів.

Class Gameplay: Метод InGameMovement - як вхідні дані він приймає натиснуту гравцем клавішу та двовимірний масив який в собі містить усі основні характеристики поля. Під час ігрового процесу обробляє натискання клавіш гравцем: рух курсора, відкриття клітинок,

встановлення/зняття прапорців з клітинки. Метод RevealedEmpty - реалізує алгоритм для автоматичного відкриття всіх суміжних порожніх клітинок та клітинок з цифрами до обраної гравцем порожньої клітинки. Метод CheckWin - перевіряє умову перемоги, за допомогою підрахунку правильно відкритих клітинок.

Class ConsoleOutput: Метод PrintGameField – візуалізує поточний стан гри та кількість наданих прапорців в консольному вікні. Метод PrintAnswer – у випадку перемоги чи поразки цей метод виводить у консоль відображення повного ігрового поля з усіма відкритими клітинками.

Class InitializationMethods: Метод FillBombs – розміщує визначену кількість бомб випадковим чином у не зайняті комірки поля. Метод CheckNeighboringCell – приватний допоміжний метод для методу FillBombs, який перевіряє усі клітинки навколо клітинки з бомбою та чи знаходяться вони в межах поля, після чого збільшує значення цієї клітинки на 1, якщо вона в межах поля та не є бомбою, цим формуючи підказку.

Class Levels: Метод CreateCustomLevel – використовує допоміжні приватні методи CheckRows, CheckColumns, CheckBombs для валідації даних, які ввів користувач та передає ці дані у метод Level. Метод Level – ініціалізує та запускає гру з обраними параметрами. Для цього він використовує *Class InitializationMethods, Class ConsoleOutput* та *Class Gameplay*.

Class Menu: Метод ShowMenu – відповідає за виведення в консольне вікно коректної сторінки меню та її параметрів. Метод PrintMenu – виводить обрану сторінку меню в консольне вікно. Метод

MovementMenu – обробляє взаємодію гравця через клавіші з вікном меню.

3. Обґрунтування технічних рішень

При створенні консольної гри Сапер було прийнято рішення, які мали на меті забезпечити зрозумілу структуру програми, можливість її подальшого розширення та ефективну роботу.

3.1. ООП

Попри те, що Сапер досить проста гра, був обраний об'єктно-орієнтований підхід. Це значить що уся програма була розділена на логічні, самостійні блоки - класи. Кожен клас відповідає за свою частину, усі класи та їх ролі у програмі були описані у пункті №2 Архітектура програми. Таким розділенням підвищується читабельність коду. Також легше виправляти помилки та баги. Якщо виникає певний баг, можливо точно визначити в якому класі шукати помилку, яка привела до багу. Завдяки такій структурі, додавати нові функції чи змінювати існуючі механіки гри простіше, оскільки зміни зазвичай обмежуються одним або кількома пов'язаними класами, мінімізуючи ризик впливу на інші частини програми.

3.2. Організація ігрового поля

Для представлення ігрового поля був обраний тип даних двовимірний масив. Таке рішення дозволило ефективно зберігати інформацію про кожну клітинку та дуже швидко отримувати доступ до будь-якої з них через її координати.

4. Опис ключових алгоритмів та логіки

4.1. Class Sapper

4.1.1. Поля та конструктор Sapper

Клас Sapper представляє з себе набір даних для побудови поля та гри. Він в собі містить параметри основного поля GameField: висота та ширина. Також він містить змінні для курсора, який рухається по цьому полю. На основі головного поля створюються два логічних поля Open та Flag, перше відповідає за те чи відкрита клітинка чи ні, друге відповідає за те чи стоїть на клітинці прапорець. Також клас містить поле для кількості бомб та кількості прапорців на рівень. Для кращого розуміння коду, в класі представленні константи Bomb та Empty, які використовуються в перевірках в інших класах та методах. Конструктор класу Sapper приймає в себе лише 3 параметри: ширину поля, висоту поля, та кількість бомб на рівень. З цих головних параметрів вираховуються усі інші, з ширини та висоти поля створюються: GameField, Open та Flag. З кількості бомб - лічильник прапорців Flags та сама кількість бомб Bombs

4.2. Class Levels.

4.2.1. Метод Level.

Цей метод створює рівень з заданими параметрами. Для початку він викликає метод FillBombs для ініціалізації поля, на якому буде грати гравець. Далі створює змінну key в яку буде далі приймати усі клавіші, які натиснув користувач, після цього починається цикл з якого можна вийти за умови, що буде натиснута клавіша Escape. На початку циклу перша команда чистить консоль, далі перевіряється умова перемоги. Якщо метод CheckWin повернув число, яке збігається з площею поля, то гравець переміг. Йому висвічується повністю відкрите поле з

позначками усіх бомб і підказок, та знизу пишеться повідомлення про те що він переміг. Після цього його повертають на голову сторінку. Якщо ж умова перемоги не справдилась, то програма виводить поле, яке є на нинішній момент. Далі вона очікує на ввід клавіші від гравця у змінну key, яку потім передаєм в метод InGameMovement, який коректно її обробляє. Після - цикл знов повторюється до моменту настання перемоги, або виходу гравця з циклу за допомогою клавіші Escape.

4.3. Class InitializationMethods.

4.3.1. Метод FillBombs.

Цей метод ініціалізує поле, задає де будуть стояти бомби та підказки відносно них. Спочатку створює змінну bombcount, яка в циклі буде збільшуватися в момент коли бомбу поставили на поле. Тепер запускаємо цикл, який буде працювати до моменту поки bombcount буде меншим за кількість бомб, яку нам передав клас Sapper. Використовуючи бібліотеку Random задаємо координати де буде розташовуватися бомба, якщо на цих координатах бомби нема, ставимо її туди та за допомогою метода CheckNeighboringCell виставляємо підказки навколо неї. Збільшуємо bombcount на 1, та переходим на наступну ітерацію циклу.

4.4. Class Gameplay.

4.4.1. Метод RevealedEmpty.

Цей метод відкриває порожні клітинки на полі. Для початку створюємо логічний масив visited, для записування вже відвіданих клітинок та створюємо чергу, яка буде містити в собі об’єкт координат (x,y). Далі в чергу записуємо нинішні координати курсора та запускаємо цикл який завершиться коли в черзі не буде елементів координат клітинок. В змінну, записуємо наш об’єкт з координатами з черги, та далі

перевіряємо чи коректні ці координати. Якщо так то йдем далі, якщо ні - переходимо на наступну ітерацію циклу. Далі йде перевірка на те чи є ця клітинка відкрита або відвідана, якщо ні - йдем далі, якщо так - переходимо на наступну ітерацію циклу. Тепер, задаємо в масив visited, що ця клітинка вже відвідана, та також в логічний масив Open кажемо, що вона відкрита. Далі перевіряємо чи стоїть на цій клітинці прапорець, якщо так то прибираємо його, та до лічильника прапорців додаєм 1, якщо ні - йдемо далі. Наступне перевіряємо чи порожня ця клітинка, якщо так то йдемо далі, якщо ні - переходимо на наступну ітерацію циклу. Якщо клітинка пройшла усі перевірки, то в чергу додаємо усі клітинки навколо неї, окрім неї самої та переходимо на наступну ітерацію циклу.

4.4.2. Метод InGameMovement.

Метод, який обробляє натискання клавіш під час гри. Він працює через switch() case. При натисканні стрілочки вгору, якщо cursorY більше за 0, то зменшуємо cursorY на 1. При натисканні стрілочки вниз, якщо cursorY менший за висоту поля – 1, то cursorY збільшуємо на 1. При натисканні стрілочки вліво, якщо cursorX більше за 0, то зменшуємо cursorX на 1. При натисканні стрілочки вправо, якщо cursorX менший за ширину поля – 1, то cursorX збільшуємо на 1. Це базове переміщення на полі. При натисканні клавіши Escape гравець повертається в головне меню. При натисканні на клавішу F якщо клітинка відкрита, та якщо лічильник прапорців більше за 0, при умові, що на клітинці стоїть прапорець, він зникає та до лічильника додається 1. Якщо ж на клітинці нема прапорця, він встановлюється та від лічильника віднімається 1. Якщо ж лічильник прапорців дорівнює 0, то якщо на клітинці є прапорець до лічильника додається 1, та прапорець зникає з цієї клітинки. При натисканні клавіш Enter в залежності від клітинку на яку

ви попали є 3 шляхи розвитку подій. Якщо ви потрапили на бомбу, то ви програєте, виводиться у консоль поле з усіма відкритими клітинками, та повідомлення що ви програли, та через 2,5 секунди вас повертає до головного меню. Якщо ви потрапили на не пусту клітинку, то вона просто відкривається, а якщо на цій клітинці прапорець, то вона відкривається, а прапорець зникає. Якщо ви потрапили на пусту клітинку, викликається метод RevealedEmpty, який відкриває усі клітинки навколо зупиняючись на клітинках із підказками.

4.4.3. Метод CheckWin.

Цей метод повертає число правильно відкритих клітинок. Спочатку ініціалізуємо змінну win, яка на початку дорівнює 0. Далі починається цикл, який ходить по масиву, та робить перевірки. Якщо клітинка не є бомбою та вона відкрита або клітинка є бомбою та не відкрита, до змінної win додається 1. В іншому випадку цикл переходить на наступну ітерацію. В кінці метод повертає число правильно відкритих клітинок.