**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ДВНЗ «УНІВЕРСИТЕТ БАНКІВСЬКОЇ СПРАВИ»**

**ІНСТИТУТ БАНКІВСЬКИХ ТЕХНОЛОГІЙ ТА БІЗНЕСУ**

**КАФЕДРА КІБЕРБЕЗПЕКИ ТА СОЦІАЛЬНИХ НАУК**

**КУРСОВА РОБОТА**

з дисципліни

**«Об’єктно-орієнтоване програмування»**

(назва дисципліни)

на тему:  **.**

Студента(ки) курсу групи

спеціальності «Кібербезпека»

(прізвище та ініціали)

Керівник

(посада, вчене звання, науковий ступінь,

прізвище та ініціали)

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_

Оцінка: ECTS \_\_\_\_

Члени комісії

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

(підпис) (прізвище та ініціали)

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

(підпис) (прізвище та ініціали)

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

(підпис) (прізвище та ініціали)

Київ - 2019 рік

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ДВНЗ «УНІВЕРСИТЕТ БАНКІВСЬКОЇ СПРАВИ»**

**ІНСТИТУТ БАНКІВСЬКИХ ТЕХНОЛОГІЙ ТА БІЗНЕСУ**

**КАФЕДРА КІБЕРБЕЗПЕКИ ТА СОЦІАЛЬНИХ НАУК**

Спеціальність: «Кібербезпека»

Курс 2 Група 203-КБ Семестр 3

***Завдання***

**На курсову роботу студента**

**Тарасенка Ярослава Юрійовича**

1. Тема курсової роботи:

Розробка гри

«САПЕР»

1. Термін здачі студеннтом закінченої роботи 19.12.2019
2. Вихідні дані до (роботи): інфориаційно-анлітичні інтернет джерела, зразки рішень практичних завдань та задач, нормативно-правова база, щодо оформлення технічної документації програмних продуктів, початкові значення для обробки.
3. Зміст пояснювальної записки (перелік питань, які належить розробити). Обґрунтування та постановка завдання. Теоретично-математичний опис поставленої задачі. Побудова алгоритмів розв’язку поставленої задачі. Висновки.
4. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) Таблиці та схеми представлено в роботі.
5. Дата видачі завдання « 10 » жовтня 2019

**Зміст**

[Вступ 6](#_Toc28034096)

[Розділ 1. Теоретично-маетматичний опис поставленої задачі 7](#_Toc28034097)

[**1.1** **Опис постановки задачі, результати аналізу задачі, вхідні та вихідні дані, обґрунтування вибору структур даних.** 7](#_Toc28034098)

[**Математичний опис поставленої задачі** 8](#_Toc28034108)

[**1.2** **Опис розв’язку задачі словесним способом, використовуючи математичний апарат** 9](#_Toc28034109)

[Розділ 2. Побудова алгоритмів розв’язку поставленої задачі 11](#_Toc28034116)

[**2.1 Графіча формалізація задачі, схема алгоритму розв’язку** 11](#_Toc28034117)

[**2.2 Розробка інтерфейсу користувача** 16](#_Toc28034118)

[Розділ 3. Практична реалізація розв’язку поставленої задачі 18](#_Toc28034119)

[**3.1 Інструкція користувача** 18](#_Toc28034120)

[Висновки 19](#_Toc28034121)

[Список використаних джерел 20](#_Toc28034122)

[Додаток А – Лістинг програми. 21](#_Toc28034123)

# Вступ

Можливості С++ включають оголошення у вигляді виразів, перетворення типів у вигляді функцій, оператори new і delete, тип bool, посилання, розширене поняття константності та змінності, функції, що підставляються, аргументи за замовчуанням, перевизначення, простори імен, класи (включаючи і всі пов'язані з класами можливості, такі як успадкування, функції-члени (методи), віртуальні функції, абстрактні класи і конструктори), перевизначення операторів, шаблони, оператори, обробку винятків, динамічну ідентифікацію і багато що інше. С++ є також мовою строгого типування і накладає більше вимагань щодо дотримання типів, порівняно з С. Компілятор VisualC++ містить багато інструментальних засобів і поліпшених можливостей, надає величезні можливості в плані оптимізації додатків, внаслідок чого можна отримати виграш як відносно розміру програми, так і відносно швидкості її виконання, незалежно від того, що являє собою ваш додаток. Система Microsoft VisualC++ дозволяє створювати як маленькі програми і утиліти для персонального використання, так і корпоративні системи, що працюють з базами даних на різних платформах.

# **Розділ 1. Теоретично-маетматичний опис поставленої задачі**

* 1. **Опис постановки задачі, результати аналізу задачі, вхідні та вихідні дані, обґрунтування вибору структур даних.**

Відеогра́ — це електронна гра, в ігровому процесі якої гравець використовує інтерфейс користувача, щоб отримати зворотну інформацію з відеопристрою.

Електронні пристрої, які використовуються для того щоб грати, називаються ігровими платформами. Наприклад, до таких платформ належать: персональний комп'ютер та гральна консоль. Пристрій введення, який використовується для керування грою, називається ігровим контролером. Це може бути, наприклад, джойстик, клавіатура та мишка, геймпад або сенсорний екран.

У 2011 році відеоігри були офіційно визнані видом мистецтва урядом США та Національним фондом мистецтв США. Однак, загальносвітове визнання їх мистецтвом лишається дискутивним питанням. Появі відеоігор передували розвиток програмованих комп'ютерів та технологій формування зображення на екранах електронних пристроїв. Різноманітні електронні та механічні ігрові пристрої існували ще в першій половині XX століття, але не мали досить значного поширення. Попередниками відеоігор є пристрій «Cathode ray tube Amusement Device» (укр. Розважальний пристрій з електронно-променевою трубкою), патент на яку Томас Ґолдсміт Молодший (англ. Thomas T. Goldsmith Jr) та Істл рей Менн (англ. Estle Ray Mann) отримали 14 грудня 1948 року, і шахова комп'ютерна програма, розроблена у 1947 Аланом Тьюрінгом.

Перша успішна спроба створити розважальний пристрій, який використовує для зворотнього зв'язку із гравцем відео, належить Вільяму Гіґінботаму.

У 1958 він розробив Tennis For Two, однак не розглядав гру як щось важливе і зрештою розібрав обладнання для інших, наукових, проектів.

Написання коду

Джерельний код є основою будь-якої відеогри і відповідає за її технічні можливості, від яких залежить контент та ігровий процес. Сучасні ігри здебільшого засновані на готових програмних модулях — ігрових рушіях, де вже реалізовані базові функції, здатні зв'язувати воєдино графіку, звук, об'єкти і їх рухи. Однак програмісти все-одно мусять писати код, щоб налаштувати рушій і сповна реалізувати задуману гру. Деякі розробники створюють власні рушії для конкретної гри. Існують як вільні ігрові рушії, доступні будь-кому, так і ті, що вимагають отримання ліцензії на їх використання. Останні як правило володіють ширшим функціоналом і використовують передові технології.

Тестування

Після завершення праці над кодом, контентом і механікою, за яких гра може функіонувати, відбувається її доопрацювання. Гра, не зібрана до кінця, але в яку можливо грати, називається альфа-версією. Вона може містити значні помилки і недоопрацювання, як відсутність певних можливостей, музики або об'єктів. Виявленням проблем займаються тестери, котрі грають в цю гру, намагаючись сповна скористатися всіма доступними можливостями в ній. Зазвичай на цьому етапі розробники записують рекламний трейлер, показуючи ігровий процес на відео, даючи потенційним гравцям уявлення про свою роботу. На пізнішому етапі виходить бета-версія, до тестування якої можуть залучатися і потенційні покупці гри. В бета-версії відбувається подальший пошук помилок, перевірка коректності взаємодії об'єктів ігрового світу, управління. Можливі внесення змін в оформлення, зміна ігрового балансу, можливостей персонажів.

**Математичний опис поставленої задачі**

Розробити алгоритм програми, яка виконує розстановку мін на полі із площиною 10 (десять) на 10 клітин, міни розставляються хаотично, в рамках поля.

Сапер (гравець) має задачу: розставити на кожній міні в полі по прапорцю.

Сапер має відкрити одну з клітин на полі, на якій буде вказана кількість мін в радіусі клітини, до якої вони торкаються. В іншому випадку гравець може натрапити на міну та програти. Такий варіант може бути реалізований навіть із першого ходу.

В такому випадку гравець отримає повідомлення про те, що гра закінчена.

Правила:

* Ви можете відкрити лише одну клітину за хід.
* Ви можете поставити лише один прапорець на місце, де розташована міна.
* Для того щоб зробити хід, використовйте o(відкрити клітинку) та f(виставити або прибрати прапорець).
* Час необмежений.
* Перемога буде зарахована після встановлення прапорців на всіх мінах в рамках поля

Сенс гри «Сапер» заключається в самій назві гри, тобто для того щоб перемогти нобхідно знайти усі міни на полі. Якщо ви знаходита міну, на неї треба встановити прапорець. Після встановлення 10 (десяти) прапорців на 10-ти мінах, гра завершується із повідомленням про її кінець.

* 1. **Опис розв’язку задачі словесним способом, використовуючи математичний апарат**

Плоске або об'ємне ігрове поле поділене на суміжні комірки (квадрати, шестикутники, куби і т. П.), Деякі з яких «заміновані»; кількість «замінованих» осередків відомо. Метою гри є відкриття всіх осередків, які не містять міни.

Гравець відкриває осередки, намагаючись не відкрити осередок з міною. Відкривши клітинку з міною, він програє. Якщо під відкритою гравцем клітинкою міни немає, то в ній з'являється число, що показує, скільки клітин, що є сусідами з щойно відкритої, «заміновано» (в будь-якому вигляді гри сусідство визначається по-своєму); використовуючи ці числа, гравець намагається розрахувати розташування хв, однак іноді навіть в середині і в кінці гри деякі осередки все ж доводиться відкривати навмання. Якщо під сусідніми осередками теж немає мін, то відкривається деяка «Не замінована» область до клітин, в яких є цифри. «Заміновані» клітини гравець може помітити, щоб випадково не відкрити їх. Відкривши всі «Не заміновані» клітини, гравець виграє.

Існують варіанти гри з полем непрямокутної форми, в тривимірному просторі, розраховані на багато користувачів варіанти. У деяких різновидах цифри, що позначають кількість замінованих осередків, розрізняються за кольором. У стандартних варіантах «Сапера» і «Mines» цифра 1 - синя, 2 - зелена, 3 - червона, 4 - темно-синя.

При деяких розстановках мін обчислити їх розташування неможливо, і гравець опиняється перед необхідністю відкривати клітини навмання. Сапер для Windows має інтерактивне поле, якщо всі можливі комбінації на полі вже відкриті, то вибуху не буде навіть при натисканні навмання на будь-якій невідкритий ділянці поля. Часто складність поля оцінюють за допомогою величини 3BV (Bechtel's Board Benchmark Value). Ця величина чисельно дорівнює мінімальній кількості безпосередніх відкриттів клітин (в стандартному варіанті «Сапера» відкриття осередків тільки лівою кнопкою миші, без використання прапорів і подвійних кліків), необхідного для відкриття всього поля. Слід зазначити, що ця величина відображає лише кількість певних дій в ідеальному випадку при певній манері гри, а зовсім не складність розстановки для конкретного гравця.

Результат сильно залежить від розташування мін. Теоретично при будь-яких ігрових параметрах є ймовірність проходження одним клацанням. Але практична реалізація генератора випадкових комбінацій не дозволяє отримати дуже просту розстановку на великих дошках. Тому результати на рівнях складності Intermediate і Expert добре відображають рівень гравця.

В офіційних програмах встановлені обмежувачі для простих дощок по 3bv. В даний час вони складають 2 для рівня складності Beginner, 30 - Intermediate і 100 - Expert.

Програма виводить на екран поле розмірністю 10 на 10 клітин. Вхідні дані, які ми вводимо мають керувати встановленням, прибиранням прапорців та відкриттям клітинок. Відкриття клітинки з міною приведе до поразки.

**Розділ 2. Побудова алгоритмів розв’язку поставленої задачі**

**2.1 Графіча формалізація задачі, схема алгоритму розв’язку**

До цих пір ми писали програми єдиним, функціонально неподільним, кодом. Алгоритм програми знаходився в головній функції, причому інших функцій в програмі не було. Ми писали маленькі програми, тому не було потреби в оголошенні своїх функцій. Для написання великих програм, досвід показує, що краще користуватися функціями. Програма складатиметься з окремих фрагментів коду, під окремим фрагментом коду розуміється функція. Окремим, тому, що робота окремої функції не залежить від роботи якої-небудь іншої. Тобто алгоритм в кожної функції функціонально достатній і не залежить від інших алгоритмів програми. Одного разу написавши функцію, її можна буде з легкістю переносити в інші програми. Функція (в програмуванні) – це фрагмент коду або алгоритм, реалізований на якійсь мові програмування, з метою виконання певної послідовності операцій. Отже, функції дозволяють зробити програму модульною, тобто розділити програму на кілька маленьких підпрограм (функцій), які в сукупності виконують поставлене завдання. Ще один величезний плюс функцій в тому, що їх можна багаторазово використовувати. Дана можливість дозволяє багаторазово використовувати один раз написаний код, що в свою чергу, набагато скорочує обсяг коду програми!

Крім того, що в С передбачено оголошення своїх функцій, також можна скористатися функціями визначеними в стандартних заголовних файлах мови програмування С++. Щоб скористатися функцією, визначеної в заголовному файлі, потрібно його підключити.

Реалізація основних алгоритмів представлена у даному підрозділі, користувач може впливати на порядок виконання алгоритму використовуючи задані в програмі команди. Тобто при використанні у діючій програмі команди «о» та «f» відповідно, дозволяють змінити хід виконання алгоритмічних функцій. Тим самим хід роботи програми міняєься в залежності від команд користувача.

Перш за все (пропускаючи оголошення бібліотек) в програмі оголошуються усі функції, які використовуються у програмі. Робимо ігрове поле, для цього використовуємо символьний тип даних «char». Оголошуємо тип даних «time\_t» для представлення арифметичного часу в програмі.

Переходимо до самих функцій.

При написанні програм середнього і високого рівня складності виникає потреба в їх розбитті на частини. Розбиття великої програми на менші частини дозволяє зменшити ризик виникнення помилок, підвищує читабельність програмного коду завдяки його структурування. Крім того, якщо деякий програмний код повторюється кілька разів (або є близьким за змістом), то доцільно організувати його у вигляді функції, яку потім можна викликати багаторазово використовуючи її ім'я. Таким чином, відбувається економія пам'яті, зменшення вихідного коду програми, і т.п ..

Першою функцією позначаємо діапазон розташування мін на полі, через цикл «if». Трапляються ситуації, коли програмі потрібно вибрати, яку операцію їй виконати, в залежності від певної умови. Оператор if служить для того, щоб виконати будь-яку операцію в тому випадку, коли умова є вірним. Умовна конструкція в С ++ завжди записується в круглих дужках після оператора if. Усередині фігурних дужок вказується тіло умови. Якщо умова виконається, то почнеться виконання всіх команд, які знаходяться між фігурними дужками. Кожному оператору if відповідає тільки один оператор else. Сукупність цих операторів - else if означає, що якщо не виповнилося попереднє умова, то перевірити дане. Якщо жодна з умов не вірно, то виконується тіло оператора else. Якщо після оператора if, else або їх зв'язки else if повинна виконуватися тільки одна команда, то фігурні дужки можна не ставити. Позначаємо їх літерою «Х», вони мають розміщуватися на полі хаотично на лініях «i» та «j» в межах поля, тобто із значенням від 0 до 10. Щоб реалізувати таку розстановку мін на полі використовуємо фунцію, в якій створюємо генератор довільних чисел та обмежуємо його значенням самого поля.

Після цього створюємо масив який буде виконувати функцію закритого від гравця поля. На практиці часто виникає необхідність в обробці даних у вигляді довільного набору значень, тобто масивів. **Масив** являє собою кінцеву іменовану послідовність величин од­ного типу, які розрізняються за порядковим номером. Опис ма­сивiв у програмі відрізняється від опису простої змінної наявніс­тю після імені квадратних дужок **«[ ]»,** в яких задається кількість елементів масиву (розмірність). Слід нагадати, що у мові C++ нумерація елементів масиву починається з 0. При оголошенні одновимірного масиву, коли масив відразу iніціюється, можна не вказувати його розмір. Якщо ж ініціюван­ня не здійснюється під час оголошення масиву, то кількість ін­дексів слід задати обов’язково константним виразом. За замовчуванням, якщо в оголошеному масиві ініціюється тільки декілька перших елементів, то його інші елементи ініціюються нулями. Так, у випадку, коли **float mas[10]= {2.2,34.56};,** останні вісім елементів масиву одержать значення**0**. Крім одновимірних масивів, тобто таких, де позиція еле­мента визначається за допомогою одного індексу, у практиці розв’язання задач часто застосовуються багатовимірні масиви. У них позиція елемента визначається записом декількох індек­сів. Найбільш розповсюджені **двовимірні масиви** або матриці. Матриці являють собою порядковий запис декількох однови­мірних масивів. Місце розташування кожного елемента визна­чається за допомогою двох індексів — номера рядка і номера стовпця, тобто порядкового номера в рядку. Індекси двовимір­них масивів записуються в квадратних дужках і нумерація ін­дексів починається з нуля **(0).**

тобто масив задається або списком елементів у тому порядку, и якому вони розташовані у пам’яті, або подається як масив ма­сивів, кожний з яких поміщається в свої фігурні дужки**«{}»**. При оголошенні і одночасному ініціюванні багатовимірних ма­сивів можна опускати кількість індексів тільки першого виміру. Якщо ініціювання не здійснюється під час оголошення масиву, то кількість індексів треба вказувати явно.

Далі створюємо наступну функцію із новим масивом. Він відповідає за створення границь таблиці.

Наступними функціями будуть «void open\_cell()» та «void\_place\_or\_remove\_flag()». У C тип функції означає, якого типу буде значення, яке функцією. Але якщо замість звичайного типу перед ім'ям функції варто специфікатор void, то з ним загальне правило не працює. Це специфікатор означає, що у функції немає значення, що повертається. Як наслідок, оператор return, що відповідає за значення, що повертається, прибирається.

Специфікатор void виконує і інші функції. Наприклад, в разі деклараторів функції, якщо void стоїть на місці параметрів функції, це означає, що функція не має параметрів, при цьому маючи прототип. Вони використовуються для визначення клітинки які будуть відкриватися гравцем щоб перевірити на наявність міни а також будуть розміщати прапорці на місце, де теоретично розташована міна. Перша функція позначає усі відкриті міни після поразки, тобто коли Ви натрапляєте на міну, усі клітинки у яких розташовані міни відкриваються для того щоб гравець міг побачити свій прогрес. Друга використовується для встановлення на кожну міну по прапорцю. Додатково встановлюємо лічильник прапорців. Після цього вводимо наші команди «o» або «f» для виклику команд керування діями сапера. Використовується конструкція із оператором «switch», Конструкція switch-case - це зручна заміна довгою if-else конструкції, яка порівнює змінну з декількома константними значеннями, наприклад int або char.Переменная в дужках порівнюється зі значеннями, описаними після ключового слова case. Після двокрапки знаходиться код, який буде виконаний в разі якщо змінна виявилася рівною поточного значення. break необхідний для того, щоб перервати виконання switch.

Наступна функція відкриває для гравця міни на полі у випадку його програшу.

Для цього використовується алгоритм довільної розстановки, який використовувався для розстановки мін на полі.

Для наступної функції робота заключається у тому щоб перевірити таблицю на наявність десяти прапорців, які встановлюються на мінах. В тому випадку, якщо їх буде достатня кількість, а саме 10 (десять), якщо 0, то програма повинна закінчити свою роботу, якщо 1 – продовжити.

Цикл while є найпростішим з чотирьох циклів, які є в C ++, і дуже схожий на розгалуження if / else

Цикл while оголошується з використанням ключового слова while. На початку циклу обробляється умова. Якщо його значенням є true (будь-нульове значення), то тоді виконується тіло циклу.

Однак, на відміну від оператора if, після завершення виконання тіла циклу, управління повертається назад до while і процес перевірки умови повторюється. Якщо умова знову є true, то тоді тіло циклу виконується ще раз

З іншого боку, якщо умова циклу завжди приймає значення true, то і сам цикл буде виконуватися нескінченно. Це називається нескінченним циклом.

Єдиний спосіб вийти з нескінченного циклу - використовувати оператори return, break, exit, goto або викинути виняток.

Програми, які працюють до тих пір, поки користувач не вирішить зупинити їх, іноді навмисно використовують нескінченні цикли разом з операторами return, break або exit для завершення циклу. Поширена така практика в серверних веб-додатках, які працюють безперервно і постійно обслуговують веб-запити.

Часто нам потрібно буде, щоб цикл виконувався певну кількість разів. Для цього зазвичай використовується змінна у вигляді лічильника циклу. Лічильник циклу - це целочисленная змінна, яка оголошується з єдиною метою: рахувати, скільки разів виконався цикл. У прикладах вище змінна count є лічильником циклу.

Лічильникам циклу часто дають прості імена, такі як i, j або k. Однак, в цих іменах є одна серйозна проблема. Якщо ви захочете дізнатися, де у вашій програмі використовується лічильник циклу і скористаєтеся функцією пошуку i, j або k, то в результаті отримаєте половину своєї програми, так як i, j або k використовуються в багатьох іменах. Отже, краще використовувати iii, jjj або kkk як імен для лічильників. Вони більш унікальні, їх значно простіше знайти, і вони виділяються в коді. А ще краще використовувати «реальні» імена для змінних, наприклад, count або будь-яке інше ім'я, яке надає контекст використання цієї змінної.

Також для лічильників циклу краще використовувати тип signed int. Використання unsigned int може призвести до несподіваних результатів.

Кожне виконання циклу називається итерацией (або ще «повтором»).

Оскільки тіло циклу зазвичай є блоком, і оскільки цей блок виконується за новою з кожним повтором, то будь-які змінні, оголошені всередині тіла циклу, створюються, а потім і знищуються за новою.

Останній цикл виводить на екран результат гри. Перемога чи поразка, в залежності від того чи всі міни ви знайшли, чи натрапили хоча б на одну з них.

Кінець програми.

**2.2 Розробка інтерфейсу користувача**

Графічний інтерфейс користувача, або Graphical User Interface (скор. GUI) являє собою всю сукупність елементів, з якими взаємодіє користувач для роботи з програмою. Програми можуть працювати і без GUI, наприклад, у фоновому режимі, виконуючи якісь системні завдання, а можуть взаємодіяти через текстовий термінал - консоль. Але у таких програм, як правило, обмежена область застосування, і їх досить небагато. Більшість програм відображають GUI, у вигляді вікон або зображень (скінів). Дана міні-стаття як раз про те, якими засобами можна реалізувати GUI в додатку на C ++

Microsoft Visual C ++

Підтримка GUI в цій IDE залежить від обраної технології та типу проекту. Почнемо з простого, Win32 Project. З цим типом проекту програмісту доступні наступні технології: Windows Template Library (WTL) і Dialog Templates.

WTL являє собою набір шаблонних класів, обертаючих стандартні елементи управління Windows. Досить незначно спрощує роботу, в порівнянні з "голим" Win32 API, але дозволяє писати об'єктний код, і спрощує деякі маніпуляції з GUI.

плюси:

Стандартні елементи управління представлені об'єктами

мінуси:

* Має досить обмежений набір класів
* Незручна робота з подіями
* Немає власного конструктора форм

Інтерфейс програми виконано у стандартній консолі Windows CLI (Command Line Interface), у форматі файлу «.EXE».

Взаємодія користувача з програмою реалізується за допомогою введення прописаних програмою команд у буфер пам’яті програми.

**Розділ 3. Практична реалізація розв’язку поставленої задачі**

**3.1 Інструкція користувача**

Таблиця 1

|  |  |
| --- | --- |
| Позначення клавіш | Виконувана функція клавіш |
| o + вказування номерів клітинок по осі x та y; | Відкриття однієї клітинки за хід; |
| o + вказування номерів клітинок по осі x та y; | Встановлення прапорця на міну в конкретній клітниці; |

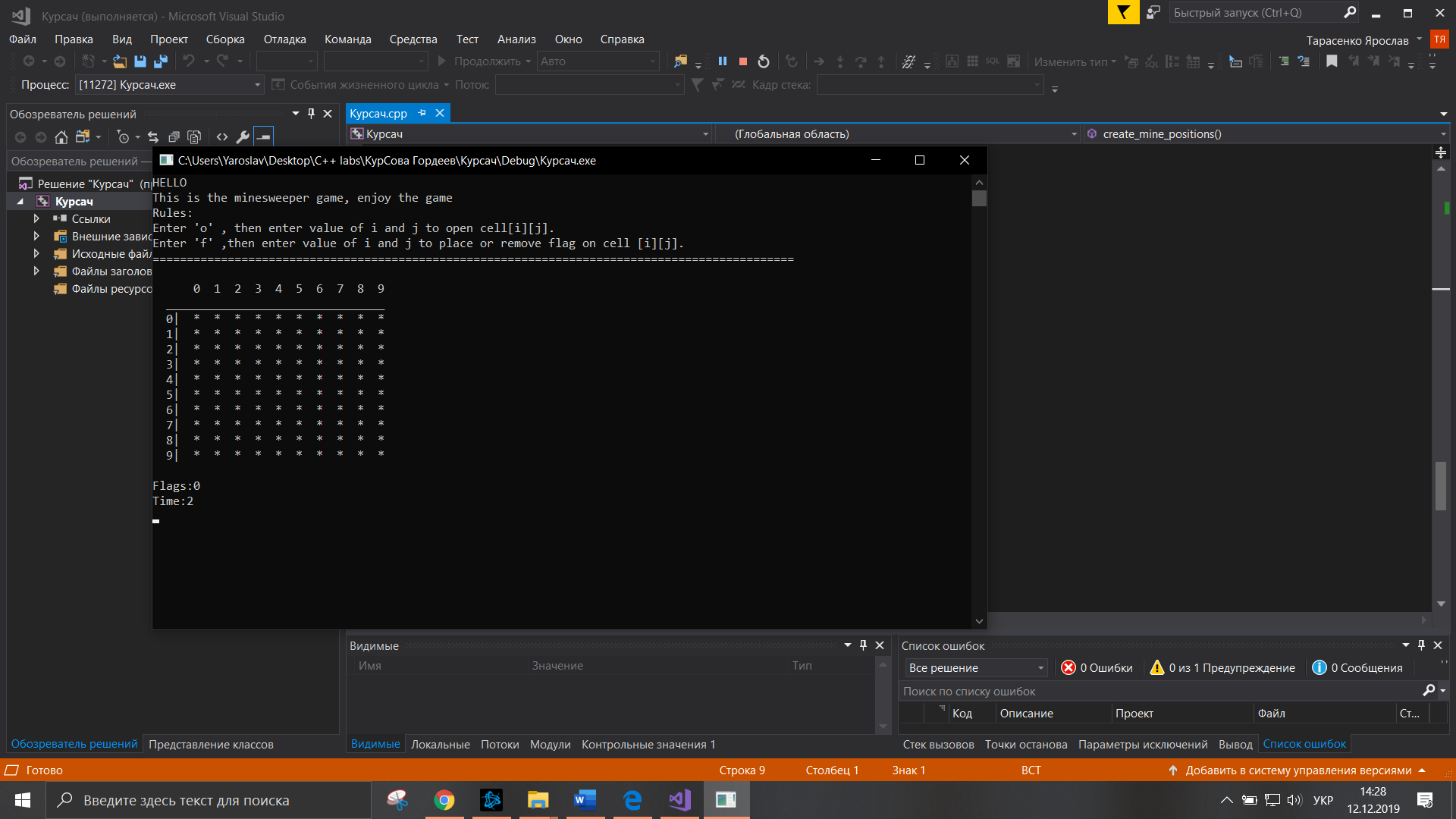


Рисунок 3.1 – Основні правила гри

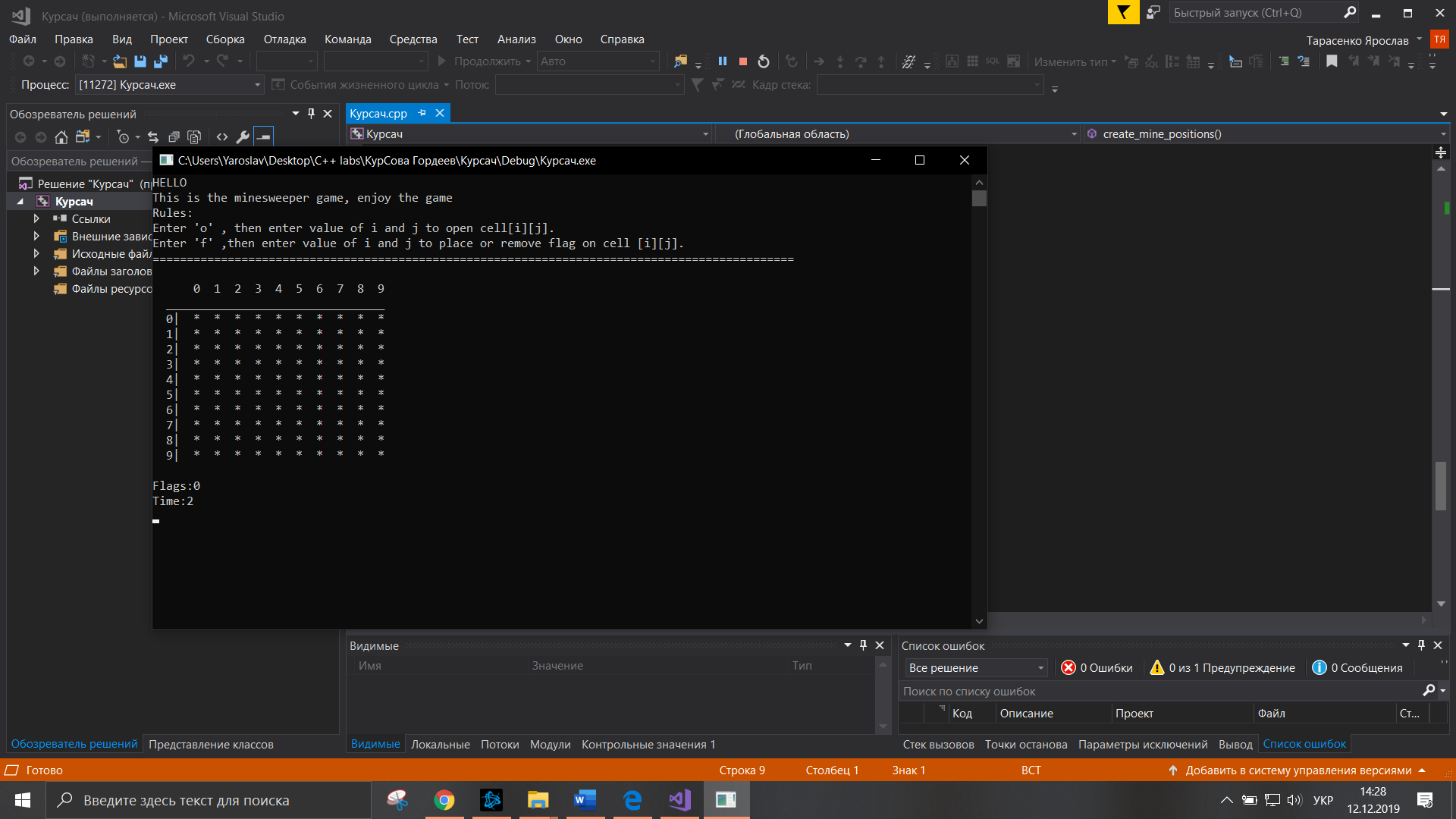


Рисунок 3.2 – Поле гри

**Висновки**

В даній курсовій роботі було описано роботу над розробкою програмного забезпечення гри «Сапер». Результат розробки представлено у додатку.

Для розробки програми використовувалося середовище програмування яке забезпечує коректну роботу програми. Назва середовища: Microsoft Visual C++ 2017. Для виконання даної роботи були використані та отримані певні навички в області розробки програмного забезпечення.

Перевагами програмування на мові С++ можна назвати: швидкодію, масштабність, можливість створення узагальнених алгоритмів для різних типів даних, їх спеціалізація, і обчислення на етапі компіляції.

Програми на платформі С++ відрізняється від інших платформ своєю швидкодією. Швидкодія цієї платформи практично не поступається іншим в швидкості обробки даних, хоча на даний момент є багато інших можливостей. Також однією з переваг програмування на мові С++ є масштаб, тобто цю мову використовують для різних платформ, систем та призначень.

З недоліків можна визначити: погана підтримка модульності. Підключення інтерфейсу зовнішнього модуля через препроцесорну вставку заголовного файлу (#include) серйозно уповільнює компіляцію, при підключенні великої кількості модулів.

**Список використаних джерел**

1. Цикли в с++. цикл while [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: <http://cherto4ka.xyz/2018/10/21/%D1%86%D0%B8%D0%BA%D0%BB%D0%B8-%D0%B2-%D1%81/>.
2. What are some simple games that can be created in C/C++ for beginners? [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: <https://www.quora.com/What-are-some-simple-games-that-can-be-created-in-C-C++-for-beginners>.
3. Implementation of Minesweeper Game [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/cpp-implementation-minesweeper-game/>.
4. Игра сапер из C++ в C# [Електронний ресурс]. – 2012. – Режим доступу до ресурсу: <http://www.cyberforum.ru/csharp-beginners/thread602232.html>.
5. Відеогра [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%93%D1%80%D0%B0.

### Додаток А – Лістинг програми.

#include "pch.h"

#include<iostream>

#include<time.h>

#include<stdlib.h>

#include<stdio.h>

#include<iomanip>

#include <time.h>

/\*Rules:

The player enters 'o' , then enters value of i and j to open cell[i][j].

Enter 'f' ,then enter value of i and j to place a flag on cell[i][j] \*/

using namespace std;

void reveal(int, int); // reveals a cell with given coordinates

void create\_mine\_positions();

void cell\_number(); //increases the number of a cell with 1

void create\_table(); //creates the game table

void open\_cell(); // opens a cell

void game();

void print\_table(char); // prints the game table

char table[10][10]; //the game table visible to the player

char table\_mine\_positions[10][10]; //table with the positions of the mines and the number of each cell

char symbol; //the input symbol, 'o' or f'

int flag\_counter = 0;

int mines\_flagged\_counter = 0;

bool end\_game\_lose = false;

time\_t time\_since\_epoch = time(0); //stopwatch counting seconds since game started

time\_t game\_time;

void cell\_number(int i, int j)// creating a table 10x10

{

if (i >= 0 && i < 10 && j >= 0 && j < 10 && table\_mine\_positions[i][j] != 'X')

table\_mine\_positions[i][j]++;

}

void create\_mine\_positions() // setting mines randomly on the table

{

int counter = 0;

srand(time(NULL));

//making new setting any time the program started

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

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

table\_mine\_positions[i][j] = '0';

int i = 0;

int j = 0;

while (counter < 10)

{

int i = rand() % 10; // limit for setting mines into the table from 0 t 10

int j = rand() % 10;

if (table\_mine\_positions[i][j] == '0') {

table\_mine\_positions[i][j] = 'X';

cell\_number(i - 1, j);

cell\_number(i + 1, j);

cell\_number(i, j - 1);

cell\_number(i, j + 1);

cell\_number(i - 1, j - 1);

cell\_number(i - 1, j + 1);

cell\_number(i + 1, j - 1);

cell\_number(i + 1, j + 1);

counter++;

}

}

}

void create\_table() // making massive for setting mines

{

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

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

table[i][j] = '\*';

}

void print\_table(char arr[10][10]) // making table border printed in every program launch

{

cout << " ";

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

cout << setw(3) << i;

cout << endl << " ";

for (int i = 0; i < 32; i++)

cout << "\_";

cout << endl;

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

cout << setw(3) << i << "|";

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

cout << setw(3) << arr[i][j];

cout << endl;

}

}

// opening cell

void open\_cell()

{

int i, j;

do

cin >> i >> j;

while (i < 0 || i>9 || j < 0 || j>9);

if (table\_mine\_positions[i][j] == 'X')

{

table[i][j] = 'X';

end\_game\_lose = true;

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

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

if (table\_mine\_positions[i][j] == 'X')

table[i][j] = 'X';

}

else

reveal(i, j);

}

void place\_or\_remove\_flag()

{

int i, j;

do

cin >> i >> j;

while (i < 0 || i>9 || j < 0 || j>9);

if (table[i][j] == '\*')

{

table[i][j] = 'F';

flag\_counter++;

if (table\_mine\_positions[i][j] == 'X')

mines\_flagged\_counter++;

}

else if (table[i][j] == 'F')

{

table[i][j] = '\*';

flag\_counter--;

if (table\_mine\_positions[i][j] == 'X')

mines\_flagged\_counter--;

}

}

void input\_symbol()

{

cin >> symbol;

switch (symbol) {

case 'o': open\_cell(); break;

case 'f': place\_or\_remove\_flag(); break;

default: input\_symbol();

}

}

void reveal(int i, int j)

{

if (table[i][j] == '\*'&&table\_mine\_positions[i][j] != 'X'&&i >= 0 && i < 10 && j >= 0 && j < 10)

{

table[i][j] = table\_mine\_positions[i][j];

if (table\_mine\_positions[i][j] == '0')

{

reveal(i, j - 1);

reveal(i, j + 1);

reveal(i - 1, j - 1);

reveal(i + 1, j - 1);

reveal(i + 1, j + 1);

reveal(i - 1, j + 1);

reveal(i - 1, j);

reveal(i + 1, j);

}

}

}

bool end\_game\_win\_check()

{

if (flag\_counter == 10 && mines\_flagged\_counter == 10)

return 1;

else

return 0;

}

void game()

{

create\_table();

create\_mine\_positions();

while (!end\_game\_lose && !end\_game\_win\_check())

{

game\_time = time(0);

print\_table(table);

cout << endl << "Flags:" << flag\_counter << endl;

cout << "Time:" << game\_time - time\_since\_epoch << endl;

input\_symbol();

}

if (end\_game\_lose) {

print\_table(table);

cout << endl << "GAME OVER" << endl;

}

else

if (end\_game\_win\_check())

{

cout << "Time to complete:" << game\_time - time\_since\_epoch << endl;

cout << endl << "YOU WIN!" << endl;

}

}

int main()

{

cout

<< "HELLO"

<<endl << "This is the minesweeper game, enjoy the game"

<<endl << "Rules:"

<< endl << "Enter 'o' , then enter value of i and j to open cell[i][j]."

<< endl << "Enter 'f' ,then enter value of i and j to place "

<< "or remove flag on cell [i][j]."

<< endl << "=============================================================================================="

<< endl << endl;

game();

return 0;

}