Зміст

[Зміст 1](#_Toc26433311)

[Вступ 2](#_Toc26433312)

[Розділ 1. Формулювання вимог до програми 3](#_Toc26433313)

[1.1 Аналіз технічного завдання й структура програми 3](#_Toc26433314)

[1.2 Обґрунтування алгоритму й структури програми 4](#_Toc26433315)

[Розділ 2. Проектування та практична реалізація розв’язку поставлених задач 5](#_Toc26433316)

[2.1 Створення об’єктів і розробка програми. Опис змінних та їх ідентифікаторів, що використовуються. 5](#_Toc26433317)

[Розділ 3. Тестування програм і результати їх виконання 12](#_Toc26433318)

[Висновки 15](#_Toc26433319)

[Списки використаних джерел 15](#_Toc26433320)

Вступ

«C++» - це мова програмування загального призначення, високого рівня, яка була створена Б'ярном Страуструпом, як розширення мови програмування С. Мова значно розширилася з часом, а сучасний «С++» має об’єктно – орієнтовані, функціональні та загальні особливості. У 90 – х роках «С++» стала найуживанішою мовою загального призначення. Її використовували для розробки програмного забезпечення, написання драйверів, системного програмування й для розробок програм та ігор. Ця мова програмування стандартизована комітетом ISO за міжнародним стандартом, і час від часу покращується за допомогою додаванням нових приміток, пунктів та вимог.

Основними перевагами мови «С++» є:

* швидша дія роботи різних програм, ніж на мові програмування «С»;
* гнучкість. Можливість розробляти різні проекти для різноманітних платформ і систем;
* підтримка різних стилів та технологій програмування: ООП, традиційне директивне програмування, метапрограмування (шаблони, макроси), узагальнене програмування;
* має набір засобів (деструктори і конструктори, посилання, стандартні шаблони), що дозволяють майже повністю виключити виділення і звільнення пам'яті вручну і різні операції з вказівниками;
* існує підтримка як об'єктно-орієнтованого, так і процедурного програмування;
* має достатньо констант, шаблонів і вбудованих функцій.

Розділ 1. Формулювання вимог до програми

1.1 Аналіз технічного завдання й структура програми

Основною метою курсової роботи є узагальнення та закріплення знань і навичок програмування за допомогою теоретичних питань і практичних задач.

Для виконання практичних завдань потрібно: середовище розробки, зручна мова програмування й пристрій для виконання.

Інтерфейсом є стандартне вікно, що складається з панелі меню, панелі інструментів і консольного вікна (Рис. 1.1.):

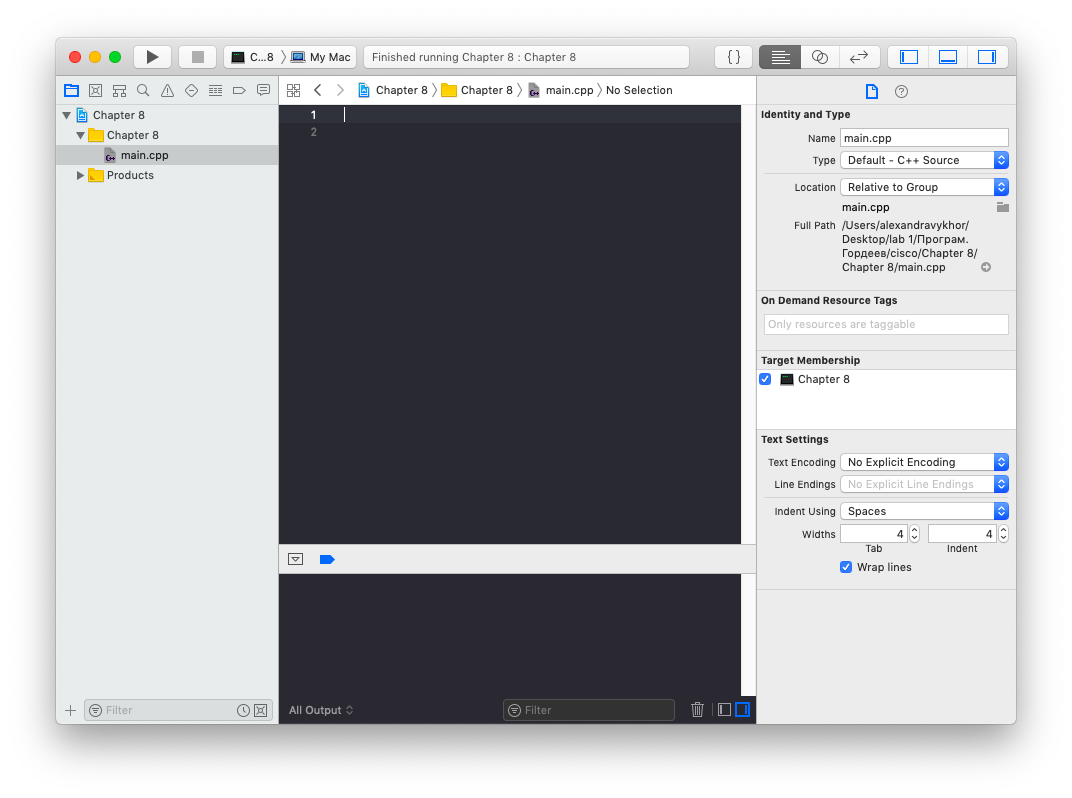


Рис. 1.1. – Панель меню

Видача результатів і «робота» здійснюється в консольному вікні.

Розробка програми, буде написана на мові С++, на інтегрованому середовищі розробки XCode, виробництва Apple, і матиме задачу: створення гри хрестики – нулики.

В засобах реалізації програми будуть використані:

* різні функції;
* цикли;
* масиви;
* сортування;
* порівняння;
* структури;
* вказівники;
* створені об’єкти.

1.2 Обґрунтування алгоритму й структури програми

Правила і опис програми: в гру грають два користувача, які вибирають між собою ніж гратимуть хрестиком або нуликів і по черзі вводять у вікні консолі за допомогою клавіатури числа відповідають за осередки. Виграє той користувач який першим складе виграшно комбінацію з 3-х поспіль хрестиків або нуликів по горизонталі, по вертикалі або по діагоналі.

Розділ 2. Проектування та практична реалізація розв’язку поставлених задач

2.1 Створення об’єктів і розробка програми. Опис змінних та їх ідентифікаторів, що використовуються.

Вказуємо дві бібліотеки: **#include <iostream>** та **#include <string**>.

**Стандартна бібліотека «C++»** - це набір класів і функцій, які написані на базовій мові і є частиною стандарту «C++» ISO.

**#include <iostream> -** бібліотека і відповідний заголовний файл, які використовується для організації введення-виведення в мові «C++».

**#include <string>** - клас з методами і змінними для організації роботи з рядками в мові програмування «C ++».

Також буде прописуємо **using namespace std.**

**Namespace** - означає простір імен. Використовується для логічного угруповання класів, функцій і змінних.

**Функції** - це блоки коду, що виконують певні операції.

**Тип** - визначає, які значення може мати змінна, які операції з нею можна робити і скільки байт в пам'яті вона буде займати.

Створюємо змінну типу **char** для ігрової дошки і роботи з нею.

**Char -** символьний тип - тип даних, призначений для зберігання одного символу (керуючого або друкованого) в певному кодуванні, це числова змінна, яка використовується для представлення символів.

Називаємо цю змінну *board*, вказуємо розмір 9, та заповнюємо цей масив елементами – тире, для подальшої роботи з дошкою. Тире виступають заміною цифрам. **Масив –** скінченна послідовність елементів одного типу даних. Для доступу до елементів масиву потрібно вказати індекс (Рис. 2.1.1.).



Рис. 2.1.1. – Масив char board

Створюємо функцію типу **void** та називаємо її *clearboard().*

Покажчик типу **void** (або ще «загальний покажчик») – це спеціальний тип покажчика, який може вказувати на об’єкти будь-якого типу даних.

Функція буде відповідати за очистку дошки. Усередині створюємо цикл **for,** в якому будемо проходитися по дошці і змінювати все в початкову (пусту) позицію, надаючи значення – ‘-‘, за допомогою масиву *board.*

**Цикл** – багаторазове проходження по одному і тому ж коду програми.

**Int –** цілочисельний тип даних (Рис. 2.1.2.).

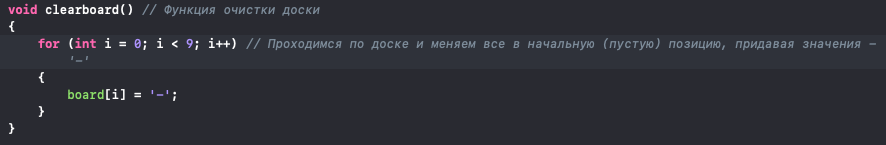


Рис. 2.1.2. – Функція для очистки дошки

Створюємо функцію типу **void** *print\_board()* для виводу робочої дошки на консоль.

Усередині функції за допомогою **cout (cout** - стандартний вихідний потік**)** створюємо робочу дошку й замінюємо її внутрішні елементи елементами *board* (тобто, замінюємо «робочими» лініями)для можливості вибору.

**Endl –** перехід на нову строку (Рис. 2.1.3.).

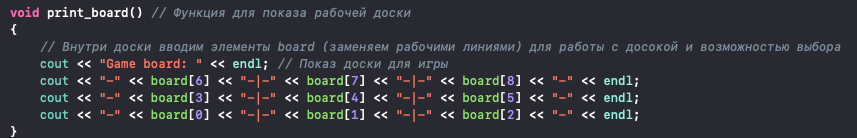


Рис. 2.1.3. – Функція для виводу робочої дошки на консоль

Створюємо функцію типу **int** *get\_move()***,** яка буде повертати значення нашого ходу.

**Return –** оператор для повернення значень. (Використовується у всіх функціях, крім void).

Усередині вводимо «фальшиву» дошку за допомогою **cout**, яка буде допомагати нам у розумінні, що потрібно натиснути для вибору ходу (Рис. 2.1.4.).

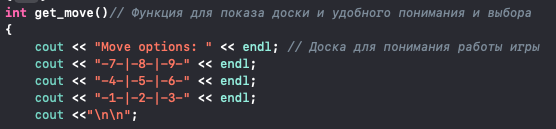


Рис. 2.1.4. – Показ «фальшивої» дошки

Після використовуємо функцію *print\_board()* для виводу робочої дошки. Просимо користувача зробити свій хід. Створюємо змінну типу **int** *move*для введення ходу. Вводимо хід за допомогою **cin**. (**Cin** – стандартний вхідний потік) (Рис. 2.1.5.).

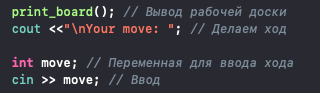


Рис. 2.1.5. – Частина коду виводу «робочої» дошки та показ ходу

За допомогою циклу **while (**цикл буде виконуватися до тих пір, поки буде виконуватися умова всередині умови циклу**)** перевіряємо на правильність введення й на зайнятість поля. Якщо одна з умов перевірки виконалася, то виводимо попередження та повторне прохання на введення коректного числа.

Умови: якщо введений хід > 9, якщо введений хід < 1, якщо поле дошки вже зайняте.

Повертаємо значення нашого ходу (Рис. 2.1.6.).

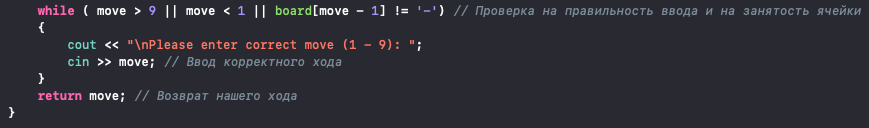


Рис. 2.1.6. – Цикл перевірки на правильність введення ходу

Створюємо булеву функцію **bool** *win\_b (char player),* яка приймає значення (знак) того гравця, який робив хід (Х або О). **Bool –** логічний тип даних, який повертає 1 або 0 (true or false).

Створюємо двовимірний масив типу **int** *wins[8][3].* Усередині нього перераховуємо всі виграшні комбінації клітинок. (Для доступу до елементів двовимірного масиву потрібно вказати два індекси) (Рис. 2.1.7.).

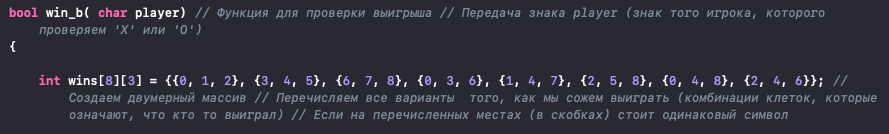


Рис. 2.1.7. – Двомірний масив варіантів виграшу

Створюємо перший цикл **for** для перевірки виграшних комбінацій того гравця, якого ми прийняли (перевіряємо 8 разів, так як 8 виграшних комбінацій). Створюємо змінну для підрахунку кількості однакових знаків в конкретній комбінації чисел. Усередині циклу створюємо другий цикл **for** (так як масив двовимірний), та перевіряємо його 3 рази (так як однакових символів повинно бути 3). За допомогою оператора вибору **if** перевіряємо умову**:** якщо на дошці квадратик, який заданий в масиві *wins,* по елементу *[i][j],* буде дорівнювати елементу, який передається (Х або О)*,* то збільшуємо *count* на один.

Виходимо з другого циклу. За допомогою оператора вибору **if** перевіряємо умову: якщо *count* буде дорівнювати 3, то повернути значення **true** (тобто, виграшу).

Перевіряємо виграшні комбінації 8 разів. Закінчуємо перевірку. Якщо не знайшли переможця, то повертаємо **false** (тобто, програшу) (Рис. 2.1.8.).

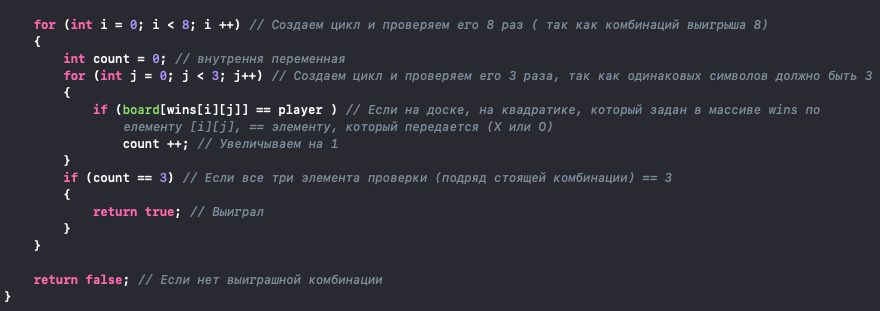


Рис. 2.1.8. – Цикли для перевірки гравця на виграш

**Оператори вибору** дозволяють прийняти програму рішення, ґрунтуючись на істинність або хибність умови. Якщо умова істинно (**true**) означає, оператор в тілі **if** виконується, після чого виконується наступний по порядку оператор. Якщо умова помилкова (**false**) означає, оператор в тілі **if** не виконується (ігнорується або пропускається) і відразу ж виконується наступний по порядку оператор.

Створюємо функцію типу **char** *play()* для виводу переможця.

Задаємо змінну типу **int** *turn = 0;* для першого ходу.

Використовуємо цикл **while (**виконується до тих пір, поки не виграв перший гравець та другий). Усередині створюємо змінну типу **int** *move,* якій присвоюємо значення, яке передає функція *get\_move* (тобто, значення нашого ходу). Показуємо наш хід (Рис. 2.1.9.).

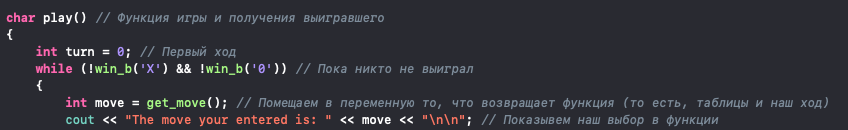


Рис. 2.1.9. – Початок циклу виводу переможця

За допомогою оператора вибору **if,** який буде визначати який знак помістити всередину вибраного квадратика. (Так як гравці ходять по черзі то умова буде виконуватися наступним чином):

якщо *turn* при діленні з остатком (%) буде дорівнювати 0, то змінюємо вказану цифру (*move*) на Х. Перевіряємо умову виграшу за допомогою оператора **if**, використовуючи функцію *win\_b(‘X’),* в яку передаємо значення Х. Якщо умова виконалася, то гравець Х виграв. Повертаємо значення ‘X’. Кінець гри (Рис. 2.1.10.).

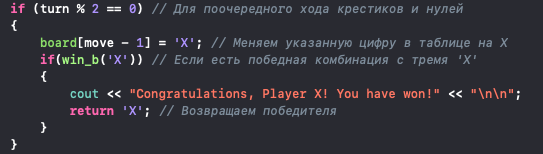


Рис. 2.1.10. – Код оператора вибору для почергового введення символів та перевірка гравця «X»

Якщо *turn* при діленні з остатком (%) буде дорівнювати 1, то змінюємо вказану цифру (*move*) на О. Перевіряємо умову виграшу за допомогою оператора **if**, використовуючи функцію *win\_b(‘Y’),* в яку передаємо значення Y. Якщо умова виконалася, то гравець O виграв. Повертаємо значення ‘O’. Кінець гри (Рис. 2.1.11.).

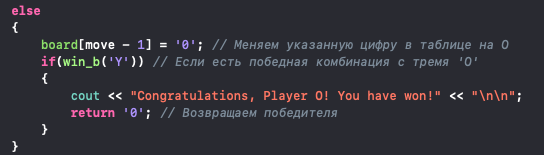


Рис. 2.1.11. – Код оператора else та перевірка гравця «Y»

Якщо гра продовжується, то збільшуємо *turn* один і хід переходить до наступного гравця (Рис. 2.1.12.).



Рис. 2.1.12. – Збільшення змінної для наступного ходу

Якщо всі ходи зроблені (9 ходів), то виводимо повідомлення про те, що виконується нічия. Повертаємо нічию: ‘D’ (Рис. 2.1.13.).

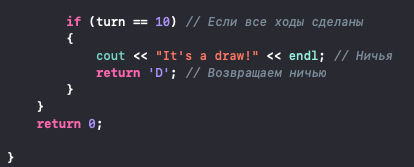


Рис. 2.1.13. – Оператор вибору для нічиєї

В основній функції **int main ()** за допомогою **cout** виводимо повідомлення про початок гри. Створюємо змінну типу **string** *replay = “yes”;* (текстовий рядок - відповідь) для подальшої відповіді й гри. Використовуємо цикл **while** (виконується до тих пір, поки рядок *replay* дорівнюватиме *yes)* для гри в хрестики – нулики.Усередині циклу використовуємо наступні функції: *clearboard ( ) –* очищуємо дошку*; play ( ) –* основна функція гри*; print\_board ( ) –* для виведення робочої дошки*.*

Виводимо повідомлення в якому питаємо, чи гравець бажає продовжити гру. Вводимо відповідь, якщо *replay*: “*yes*”, то гру починаємо спочатку. Якщо відповідь інша: завершуємо гру (Рис. 2.1.14.).

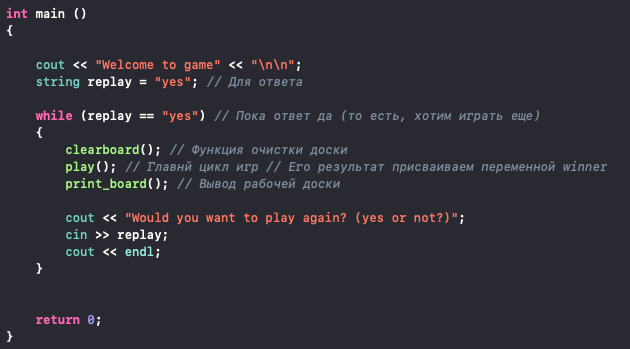


Рис. 2.1.14. – Основна функція

Розділ 3. Тестування програм і результати їх виконання

Початок гри: показ неробочої дошки для розуміння того, що потрібно нажимати. Показ робочої дошки для гри. Вибір ходу першого гравця.

Кожний наш хід перевіряється на виграшну комбінацію (Рис. 3.1.).

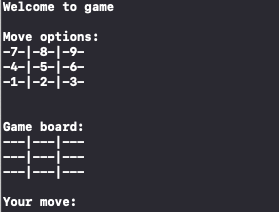


Рис. 3.1. – Початок гри

Хід першого гравця: 7. Поле заповнюється «Х». Гра продовжується (Рис. 3.2.).

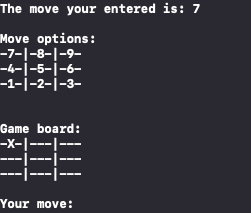


Рис. 3.2. – Перший хід першого гравця

Хід другого гравця: 8. Поле заповнюється «О» Гра продовжується (Рис. 3.3.).

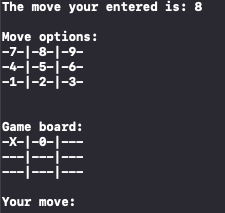


Рис. 3.3. – Другий хід другого гравця

Хід першого гравця: 4. Поле заповнюється «Х» Гра продовжується (Рис. 3.4.).

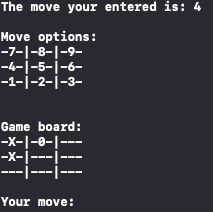


Рис. 3.4. – Третій хід першого гравця

Хід другого гравця: 5. Поле заповнюється «О» Гра продовжується (Рис. 3.5.).

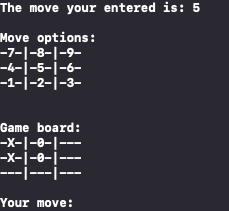


Рис. 3.5. – Четвертий хід другого гравця

Хід першого гравця: 1. Поле заповнюється «Х» Гравець виграв, зібравши виграшну комбінацію. Показ питання: «Чи бажаємо грати знову?» (Рис. 3.6.)

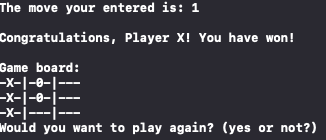


Рис. 3.6. - П’ятий хід першого гравця

Відповідь: «Так» - “Yes”. Гра починається знову (Рис. 3.7.).

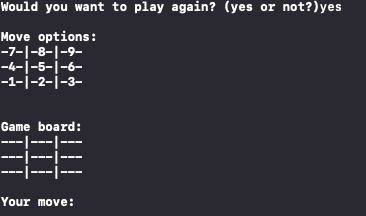


Рис. 3.7. – Шостий хід другого гравця

Робимо нічию (Рис. 3.8.) :

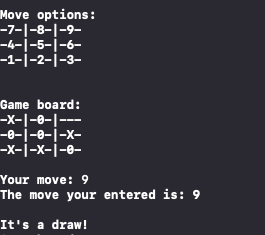


Рис. 3.8. - Нічия

Якщо вводимо зайняте поле (Рис. 3.9.):

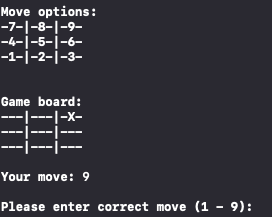


Рис. 3.9. – Ввід зайнятого поля

Якщо вводимо некоректне число (Рис. 3.10.):

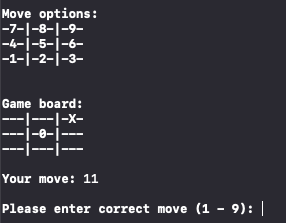


Рис. 3.10. – Ввід некоректного числа

Завершення гри (Рис. 3.11.):



Рис. 3.11. – Завершення гри

Висновки

Написання даної курсової закріпило мої набуті знання й покращило роботу виконання програмних задач. Ця робота краще розвинула мої творчі та розумові здібності в написані гри. Я більше навчилася працювати з різними масивами даних. Можу створювати різні типи функцій та їх реалізацію.

Мета курсової роботи: розробити гру хрестики – нулики, була виконана.

Списки використаних джерел

1. cppreference.com [Електронний ресурс] // Справка по С++ // URL: https://ru.cppreference.com/w/
2. Программирование [Електронний ресурс] // URL: https://prog-cpp.ru/
3. Ravesli [Електронний ресурс] // URL: https://ravesli.com/
4. CppStudio [Електронний ресурс] // URL: http://cppstudio.com/