ВІДОКРЕМЛЕННИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ

«ОПТИКО-МЕХАНІЧНИЙ ФАХОВИЙ КОЛЕДЖ КИЇВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ІМЕНІ ТАРАСА ШЕВЧЕНКА»

Циклова комісія спеціальності «Інженерія програмного забезпечення»

**З В І Т**

**З НАВЧАЛЬНОЇ ПРАКТИКИ**

спеціальність 121 Інженерія програмного забезпечення

освітньо-кваліфікаційний рівень «фаховий молодший бакалавр»

|  |  |  |
| --- | --- | --- |
| ПІДСУМКОВА ОЦІНКА  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оцінка цифрою та прописом)  Керівники практики від коледжу:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. С. Нікітенко  (оцінка) (підпис)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т. І. Лумпова  (оцінка) (підпис)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ М. С. Ходжаєв  (оцінка) (підпис) |  | Виконав: |
| студента 2-го курсу групи ІПЗ-22 |
| Якубовський Максим Віталійович  залікова книжка № \_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис виконавця)  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2024 року |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. В. Степанюк  (оцінка) (підпис) |  |  |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2024 року |  |  |

м. Київ — 2024

ЗАТВЕРДЖУЮ

Директор ОМФК КНУ

імені Тараса Шевченка  
 \_\_\_\_\_\_\_\_\_\_ Борис ГАПРІНДАШВІЛІ

«17» травня 2024 р.

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| на період | *навчальної* | | практики | |
|  | (навчальної, технологічної, виробничої) | |  | |
| *Якубовський Максим Віталійович* | | | | |
| (П.І.Б студента.) | | | | |
| Спеціальність | *121 Інженерія програмного забезпечення* | Група | | *ІПЗ-22* |

|  | **Питання для вивчення** | **Відмітки про виконання** |
| --- | --- | --- |
| **1** | **Вступне заняття.** Вивчення питань охорони праці. Ознайомлення з електронною бібліотекою коледжу. Вивчення правил оформлення звітної документації. |  |
| **2** | **Розділ "Інформатика**" |  |
| 2.1 | Засобами MS Word розробити документ складної структури та зберегти у різних форматах |  |
| 2.2 | Розв’язування рівнянь |  |
| 2.3 | Розв’язування оптимізаційних задач |  |
| 2.4 | Основи статистичного аналізу даних. Ряди даних. Кореляційний аналіз даних |  |
| 2.5 | Обчислення основних статистичних характеристик вибірки |  |
| 2.6 | Візуалізація рядів і трендів даних |  |
| 2.7 | Розв‘язок задач за допомогою засобів табличного редактора MS Excel |  |
| 2.8 | Проектування бази даних та створення бази даних *MS ACCESS* |  |
| 2.9 | Основні поняття реляційних баз даних |  |
| 2.10 | Введення, коригування, вилучення інформації з бази даних *MS ACCESS* |  |
| 2.11 | Вибірка, пошук, фільтрація даних в *MS ACCESS* |  |
| 2.12 | Виконання практичних завдань за допомогою засобів табличного редактора MS Access |  |

|  | **Питання для вивчення** | **Відмітки про виконання** |
| --- | --- | --- |
| **3** | **Розділ "Програмування С++"** |  |
| 3.1 | Алгоритмізація обчислювальних процесів. Розробка постановки задачі, визначення вхідних та вихідних даних, оформлення блок-схем. |  |
| 3.2 | Типи даних в мові С/С++, внутрішнє представлення даних, операції перетворення типів даних. |  |
| 3.3 | Оператор присвоювання. Операції введення- виведення даних |  |
| 3.4 | Програмування лінійних та розгалужених обчислювальних процесів |  |
| 3.5 | Програмування циклічних обчислювальних процесів |  |
| 3.6 | Складання програм |  |
| 3.7 | Виконання тестів |  |
| 4 | **Розділ "** **Практична робота з С++"** |  |
| 4.1 | Тема 1. Введення та виведення у базових консольних програмах |  |
| 4.2 | Тема 2. Структури даних для управління станом програми |  |
| 4.3 | Тема 3. Робота з файлами: запис та читання |  |
| 4.4 | Тема 4. Формати файлів та їх обробка |  |
| 4.5 | Тема 5. Системи контролю версій (VCS) |  |
| 4.6 | Тема 6. Мова Markdown для опису проекті |  |
| 4.7 | Розроблення ігрової програми |  |
| 5 | Розділ ”Веб-дизайн та HTML” |  |
| 5.1 | Мова розмітки HTML та структура Веб-сторінки |  |
| 5.2 | Розмітка тексту за допомогою HTML. Зображення та посилання |  |
| 5.3 | Каскадні таблиці стилів CSS та їх структура |  |
| 5.4 | Спадковості і каскадування за допомогою CSS |  |
| 5.5 | Розробка та оформлення Постановки задачі |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **Питання для вивчення** | | | | **Відмітки про виконання** | | |
| 5.9 | | Розробка HTML документу | | | |  | | |
| **6** | | Систематизація матеріалів, оформлення звіту з навчальної практики | | | |  | | |
| Голова циклової комісії | |  |  | Савєльєва І.В. | |  |  |
|  | | (підпис) |  | (ініціали, прізвище) | |  | (дата) |
| Керівники практики від коледжу | |  |  |  | |  |  |
|  | |  |  | Нікітенко А.С. | |  | 17.05.2024р. |
|  | | (підпис) |  | (ініціали, прізвище) | |  | (дата) |
|  | |  |  | Лумпова Т.І. | |  | 17.05.2024р. |
|  | | (підпис) |  | (ініціали, прізвище) | |  | (дата) |
|  | |  |  | Нікітенко А.С. | |  | 17.05.2024р. |
|  | | (підпис) |  | (ініціали, прізвище) | |  | (дата) |
|  | |  |  | Лумпова Т.І. | |  | 17.05.2024р. |
|  | | (підпис) |  | (ініціали, прізвище) | |  | (дата) |
|  | |  |  |  | |  |  |
|  | |  |  |  | |  |  |

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

4

Розроб.

Перевір.

Реценз.

В.Я

Н. Контр.

В.Я

Затверд.

Звіт з навчальної практики

Літ.

Аркушів

ІПЗ-21

ОМФК.0121 НП 20

**ЗМІСТ**

[ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 5](#_Toc169276143)

[ВСТУП 6](#_Toc169276144)

[1 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ 7](#_Toc169276145)

[2 РОЗДІЛ "ІНФОРМАТИКА" 8](#_Toc169276146)

[**2.1** **Розроблення засобами MS Word документа складної структури** 8](#_Toc169276147)

[**2.2 Розв‘язок задач за допомогою засобів табличного редактора MS Excel** 13](#_Toc169276148)

[**2.3 Виконання** **практичних завдань за допомогою засобів табличного редактора MS Access** 15](#_Toc169276149)

[3 РОЗДІЛ "ПРОГРАМУВАННЯ С++" 25](#_Toc169276150)

[**3.1 Теоретичні відомості** 25](#_Toc169276151)

[3.1.1 Загальні поняття. Елементи мови С++ - константи, змінні, операції, перетворення типів 25](#_Toc169276152)

[3.1.2 Арифметичні операції 25](#_Toc169276153)

[3.1.3 Поняття вхідного та вихідного потоку, найпростіші математичні функції 25](#_Toc169276154)

[3.1.4 Програмування лінійних та розгалужених обчислювальних процесів 25](#_Toc169276155)

[3.1.5 Програмування циклічних обчислювальних процесів 25](#_Toc169276156)

[**3.2 Практичне завдання** 25](#_Toc169276157)

[3.2.1 Завдання 25](#_Toc169276158)

[3.2.2 Постановка задачі 26](#_Toc169276159)

[3.2.3 Код програми 27](#_Toc169276160)

[3.2.4 Скріншот виконання програми 28](#_Toc169276161)

[**3.3 Виконання тестів** 29](#_Toc169276162)

[4 РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++" 30](#_Toc169276163)

[**4.1 Тема 1. Введення та виведення у базових консольних програмах** 30](#_Toc169276164)

[**4.2 Тема 2. Структури даних для управління станом програми** 31](#_Toc169276165)

[**4.3 Тема 3. Робота з файлами: запис та читання** 31](#_Toc169276166)

[**4.4 Тема 4. Формати файлів та їх обробка** 31](#_Toc169276167)

[**4..5 Тема 5. Системи контролю версій (VCS)** 31](#_Toc169276168)

[**4.6 Тема 6. Мова Markdown для опису проекті** 31](#_Toc169276169)

[**4.7 Розроблення ігрової програми** 31](#_Toc169276170)

[4.7.1 Код програми 31](#_Toc169276171)

[4.7.2 Скріншоти виконання програми 31](#_Toc169276172)

[5 РОЗДІЛ ”ВЕБ-ДИЗАЙН ТА HTML” 31](#_Toc169276173)

[**5.1 Мова розмітки HTML та структура Веб-сторінки** 31](#_Toc169276174)

[**5.2 Розмітка тексту за допомогою HTML. Зображення та посилання** 32](#_Toc169276175)

[**5.3 Каскадні таблиці стилів CSS та їх структура** 32](#_Toc169276176)

[**5.4 Спадковості і каскадування за допомогою CSS** 34](#_Toc169276177)

[**5.5 Практична частина** 34](#_Toc169276178)

[5.5.1 Постановки задачі 34](#_Toc169276179)

[5.5.2 HTML документ 35](#_Toc169276180)

[ВИСНОВКИ 38](#_Toc169276181)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ 39](#_Toc169276182)

[ДОДАТКИ 40](#_Toc169276183)

ОМФК.0121 НП 20

# ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

# ВСТУП

Автор:

Дата:

Мета:

Перелік вивчених тем:

# 1 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ

Безпека життєдіяльності при роботі з комп'ютером є дуже важливою, оскільки включає в себе заходи, які забезпечують безпеку та здоров'я користувача під час використання технологій інформаційної та комунікаційної технології (ІКТ). Основні аспекти безпеки життєдіяльності при роботі з комп'ютером включають:

Фізична безпека

Комфортне робоче місце: Забезпечення належного освітлення, правильної посадки за робочим столом, зручного крісла з підтримкою спини, щоб уникнути напруги в м’язах і захворювань, таких як біль у спині і зниження зірки.

Правильне розміщення обладнання: Клавіатура, монітор і миша повинні бути розміщені так, щоб уникнути зайвого напруження для рук і очей. Екран монітора повинен бути належно регульованим, щоб уникнути надмірної роботи очей.

Електрична безпека: Використання стабілізатора напруги, правильне розміщення кабелів для уникнення перевитягання або вплетення, а також використання правильно заземленого розетки.

Кібербезпека

Антивірусне програмне забезпечення: Встановлення та регулярне оновлення антивірусного програмного забезпечення для захисту від вірусів, шпигунського програмного забезпечення та інших загроз.

Паролі і безпека: Використання складних паролів для доступу до комп’ютера і програм, а також використання двофакторної аутентифікації, де це можливо.

Оновлення програмного забезпечення: Регулярне оновлення операційної системи та інших програм з метою закриття вразливостей, які можуть бути використані для атак.

Захист даних: Регулярне створення резервних копій важливих даних на зовнішніх носіях або хмарних сховищах для захисту від випадкового видалення або атак зловмисників.

Ергономічна безпека

Паузи і розминки: Регулярні перерви під час роботи за комп’ютером для відпочинку очей, розминання м’язів і поправлення посадки.

Організація робочого часу: Організація робочого дня з урахуванням регулярних перерв і розминок, щоб уникнути втоми і напруження.

Соціальна і психологічна безпека

Збалансоване використання часу: Уникання перевантаження і надмірного користування комп’ютером, що може призвести до соціальної ізоляції і стресу.

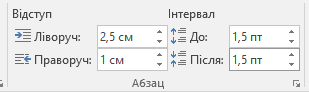
Освітня програма з кібербезпеки: Проведення навчання користувачів щодо базових правил кібербезпеки, виявлення підозрілих дій і використання інструментів для захисту даних.

# 2 РОЗДІЛ "ІНФОРМАТИКА"

## **2.1 Розроблення засобами MS Word документа складної структури**

1. Створити новий документ MS Word (орієнтація книжкова, розмір

А4, поля документа: верхнє –1.5, нижнє – 1.5, ліве – 2.5, праве – 1.0 см).

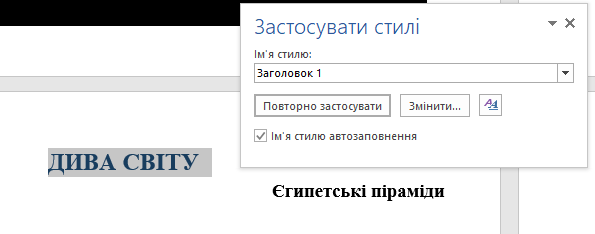
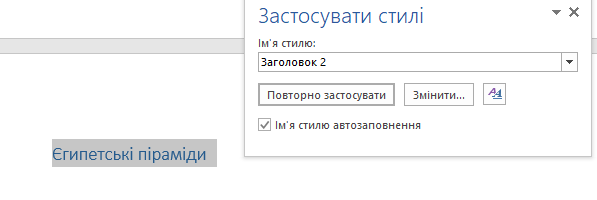


3. Відформатувати доданий текст: для заголовків першого рівня

обрати стиль «Заголовок 1», для заголовків другого рівня – стиль

«Заголовок 2», для основного тексту встановити шрифт Times New

Roman, 14 пт, інтервал між рядками 1.5 пт, відступ абзаца 1.25.

4. Змінити стиль заголовків першого рівня: шрифт Times New

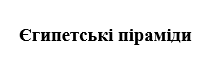
Roman, 18 пт, жирний, всі літери заглавні, колір темно-синій,

розташування посередині сторінки.



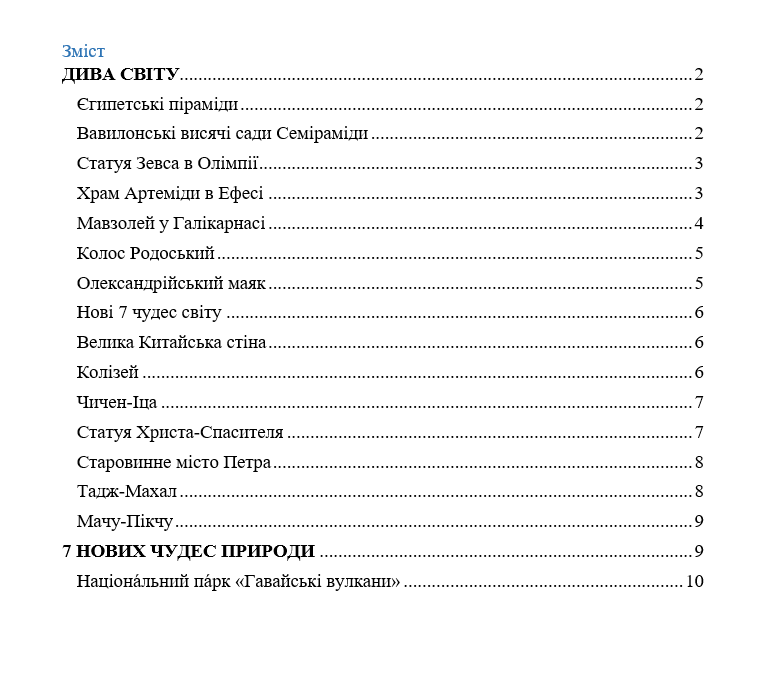
5. Змінити стиль заголовків другого рівня: шрифт Times New

Roman, 16 пт, жирний, колір чорний, розташування ліворуч.

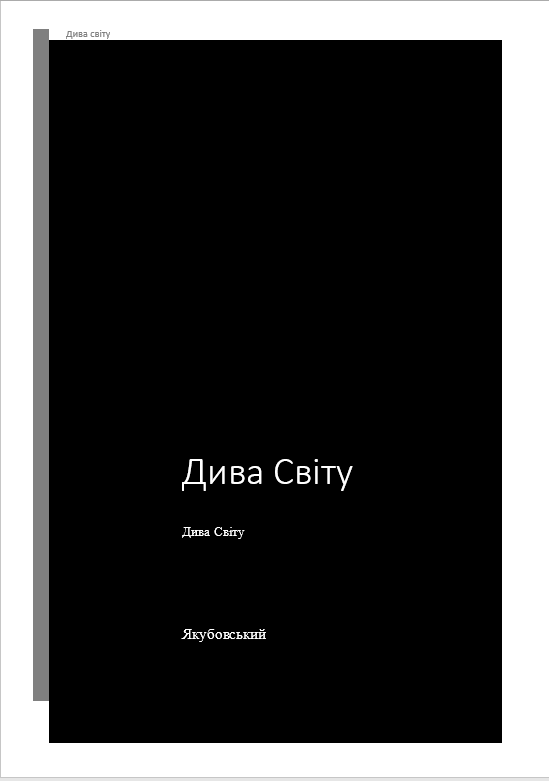


6. Додати порожню сторінку на початок документа і створити на ній

зміст.



7. Додати титульну сторінку документа та оформити її.



8. Вставити нумерацію сторінок документа наступним чином: на

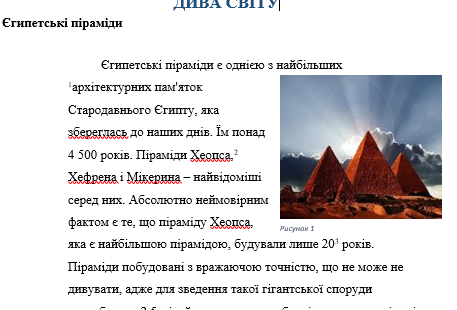
непарних сторінках нумерація розташовується внизу сторінки

праворуч, на парних – внизу сторінки ліворуч.



9. Додати в текст документа картинки (фото) відповідно до його

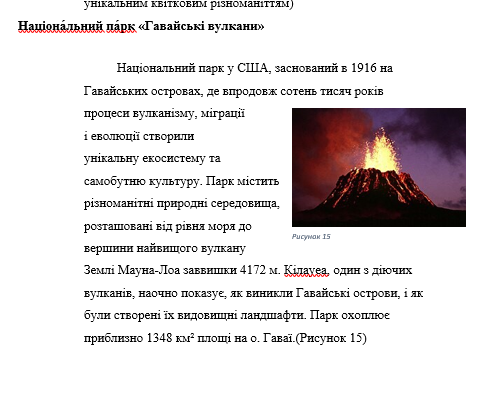
змісту.



10. Розширити розділ «7 НОВИХ ЧУДЕС ПРИРОДИ» додатковою

інформацією з мережі Інтернет. Виділити в ньому заголовки другого

рівня, вставити відповідні фотографії.



11. Користуючись можливостями Word підпису об’єктів документа

вставити підписи до рисунків у форматі: Рисунок 1, Рисунок 2 і т.д:

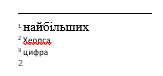
12. В тексті документа створити посилання на рисунки (не менше 5

посилань).

13. Створити виноски не менше ніж до трьох слів тексту (за власним

вибором).



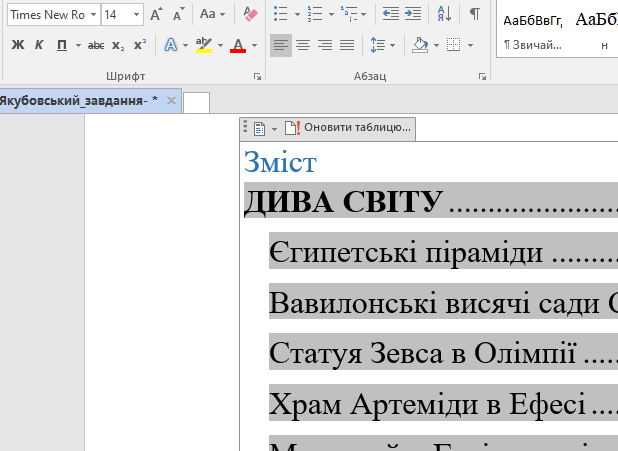
14. Додати верхній колонтитул з назвою розділу тексту.



15. Оновити сторінку зі змістом документа.



16. Для змісту встановити шрифт Times New Roman, 14 пт.

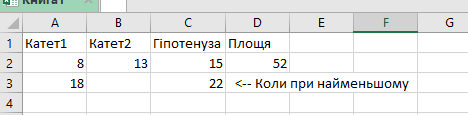


## **2.2 Розв‘язок задач за допомогою засобів табличного редактора MS Excel**

а допомогою засобів табличного редактора MS Excel розв’язав наступні задачі:

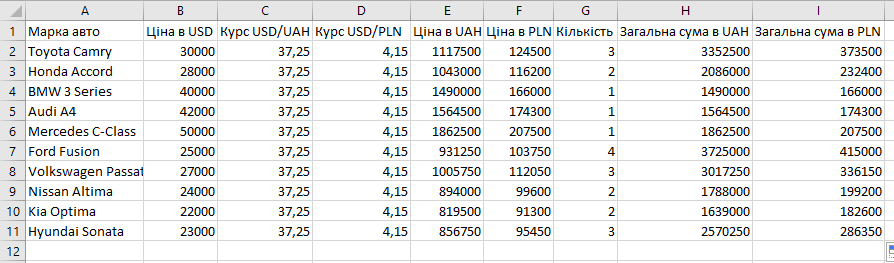
1

Катети прямокутного трикутника дорівнюють 8см і 13см. Визначить гіпотенузу цього трикутника та площу. На скільки має змінитися менший катет заданого трикутника, щоб його гіпотенуза дорівнювала 22 см? Чому дорівнює площа такого трикутника?



2

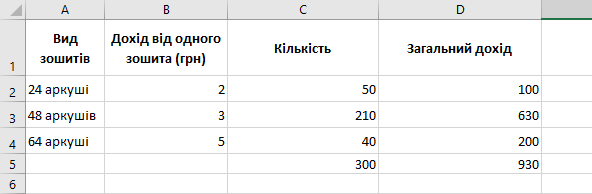
Ціну 10 марок автомобілів, задану в $, перерахувати в грошових одиницях України та Польщі, використовуючи дані з інтернету про ціну автомобілів та курси валют. Визначить, яку суму необхідно сплатити автосалону, якщо він закупає по декілька екземплярів кожної моделі. (Марки автомобілів та їх кількість для автосалону обрати самостійно).



3

Типографія виготовляє зошити трьох видів: по 24 аркуші, по 48 аркушів і по 64 аркуші. Денний дохід від одного зошиту кожного виду 2 грн., 3 грн. і 5 грн. Необхідно визначити, за якої кількості виробів типографія отримає максимальний дохід, з урахуванням того, що:

* загальний обсяг виробництва — 300 зошитів на день;
* підприємству потрібно виготовити 50 зошитів по 24 аркуші для виконання існуючого замовлення;
* 40 зошитів по 48 аркушів — для виконання планового замовлення;
* зошитів по 64 аркуші необхідно виготовляти не більше ніж 40 одиниць на день.



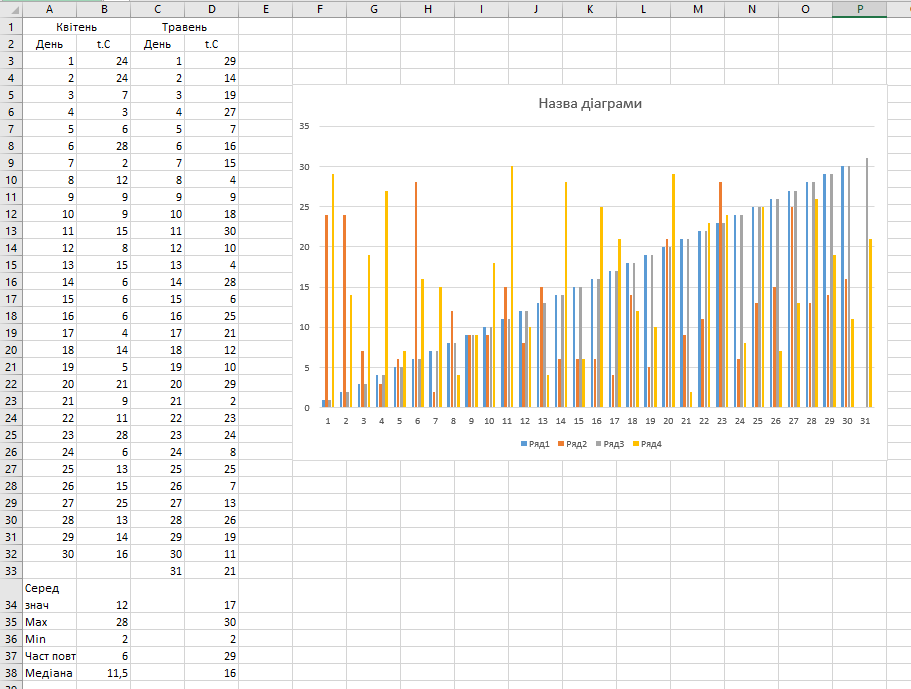
4

Засобами табличного процесора здійсніть статистичний аналіз температурного режиму двох весняних місяців (для введення початкових значень температури скористайтесь зразком таблиці на рисунку 2.18).

За веденими даними потрібно визначити:

* середнє значення температури за місяць;
* максимальну і мінімальну температури;
* значення температури, що найчастіше повторювалось;
* медіану температурних значень;

Використовуючи комбіновану діаграму виконайте візуалізацію значень температур за два місяці.

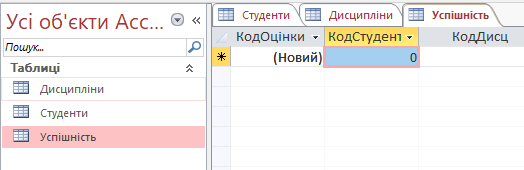


## **2.3 Виконання** **практичних завдань за допомогою засобів табличного редактора MS Access**

За допомогою засобів табличного редактора MS Access виконав наступні завдання:

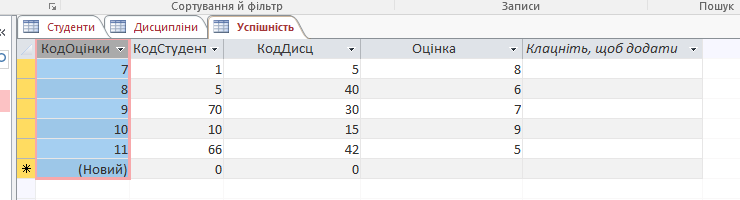
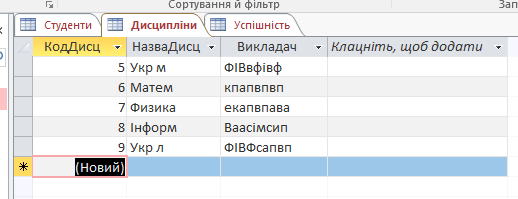
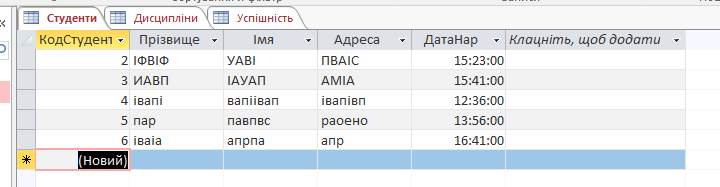
1

Створіть структуру БД ГРУПА із таблицями Студенти, Дисципліни і Успішність. Властивості цих таблиць



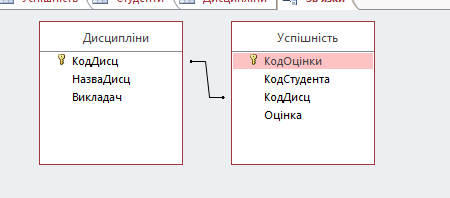
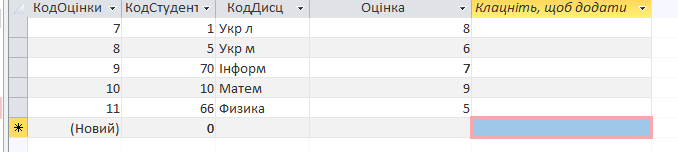
2

В кожну таблицю введіть по п’ять записів, збережіть створену БД.



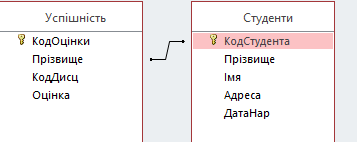
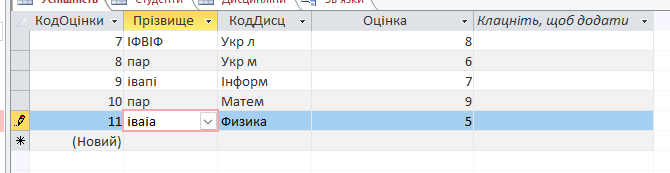
3

Відкрийте таблицю Успішність у режимі конструктора. Для поля КодДисц у списку типів даних виберіть ***Майстер*** ***підстановок***. Підставте замість числових кодів КодДисц таблиці Успішність відповідні назви дисциплін із таблиці Дисципліни. Для поля КодДисц уведіть підпис НазваДисц.



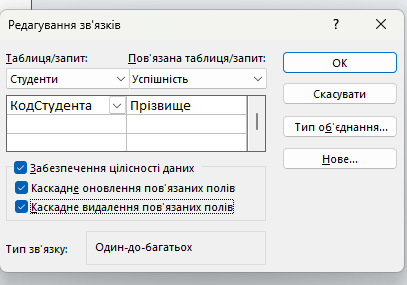
4

У таблиці Успішність для поля КодСтудент налаштуйте підстановку прізвищ з таблиці Студенти і введіть підпис Прізвище.

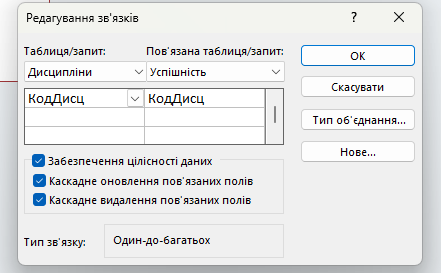


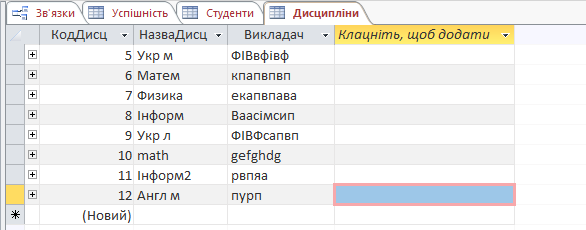
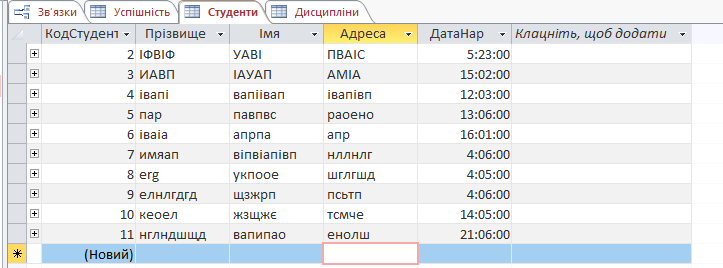
5

Відкрийте вікно ***Зв’язки***, додавши до нього таблиці Дисципліни, Успішність, Студенти. Зверніть увагу, що після налаштування підстановок між таблицями вже з’явилися зв’язки, для яких залишилося зазначити тип і умови цілісності. Викличте контекстне меню на лінії зв’язку, виберіть команду, у вікні ***Редагування зв’язків*** змініть тип зв’язку між таблицями Студенти та Успішність на **один-до-багатьох** і встановіть всі три прапорці. Це забезпечить цілісність даних і каскадне оновлення та видалення записів.



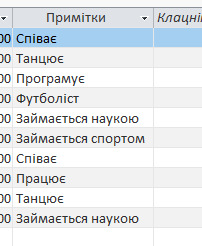
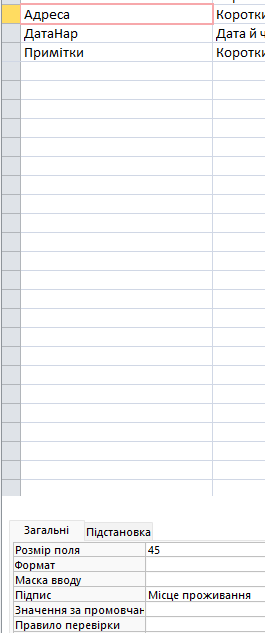
6

Зробіть подібні налаштування для зв’язку між таблицями Дисципліни та Успішність. Збережіть схему даних. Доповніть таблиці Студенти, Успішність до 10 записів, таблицю Дисципліни — до 8 записів.



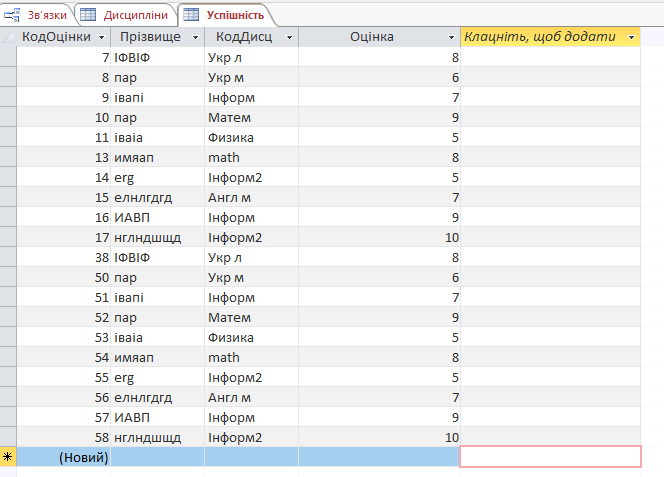
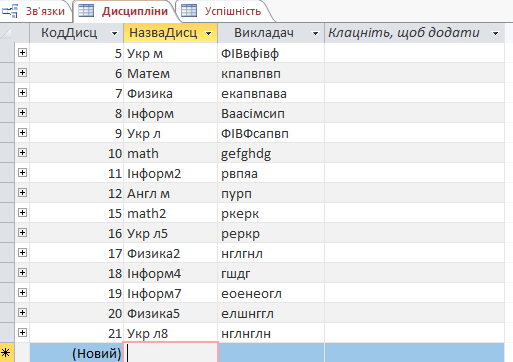
7

У таблиці Студенти для поля Адреса встановіть підпис Місце проживання. У кінці таблиці додайте поле Примітки, у яке введіть дані, наприклад, «Займається спортом», «Співає», «Танцює», «Займається наукою». Збережіть зміни в таблиці.

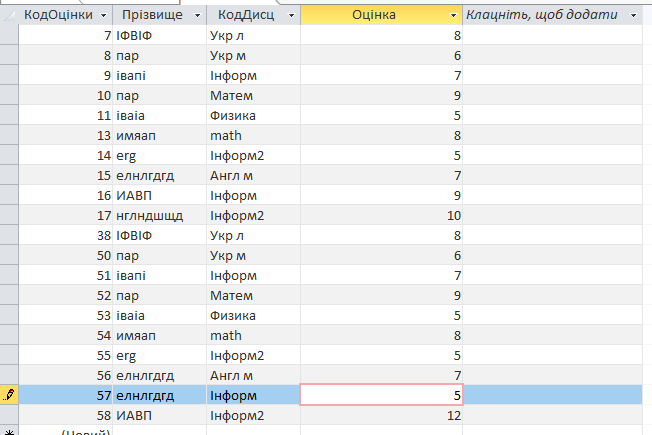


8

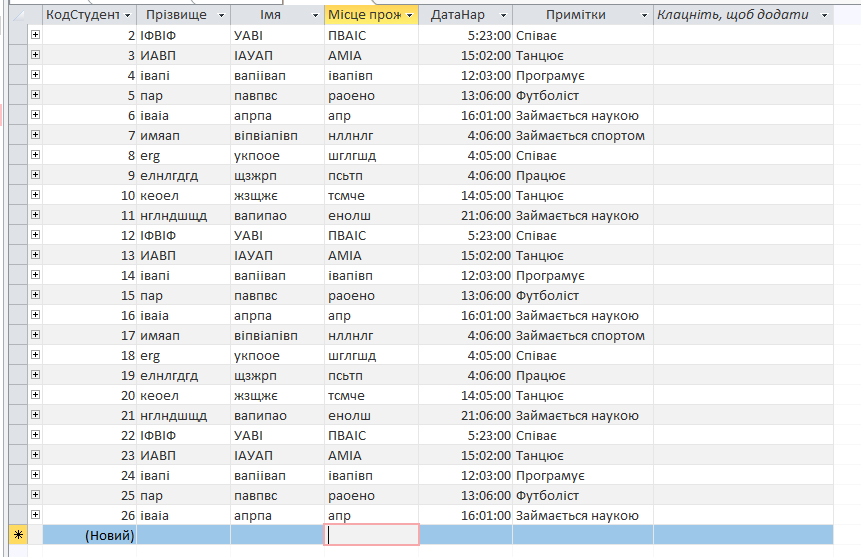
Доповніть таблицю Дисципліни до 15 записів. Таблицю Успішність доповніть до 20 записів шляхом копіювання й вставляння.

 9

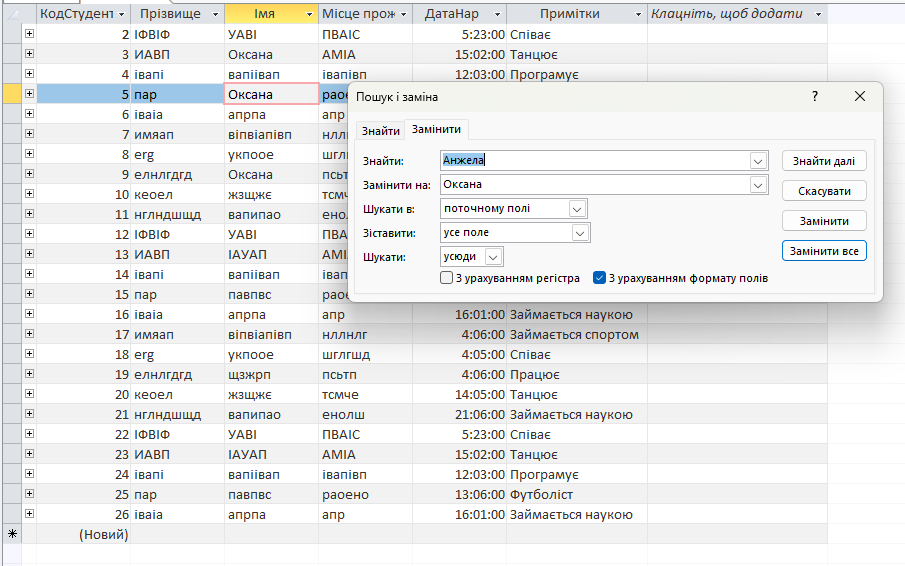
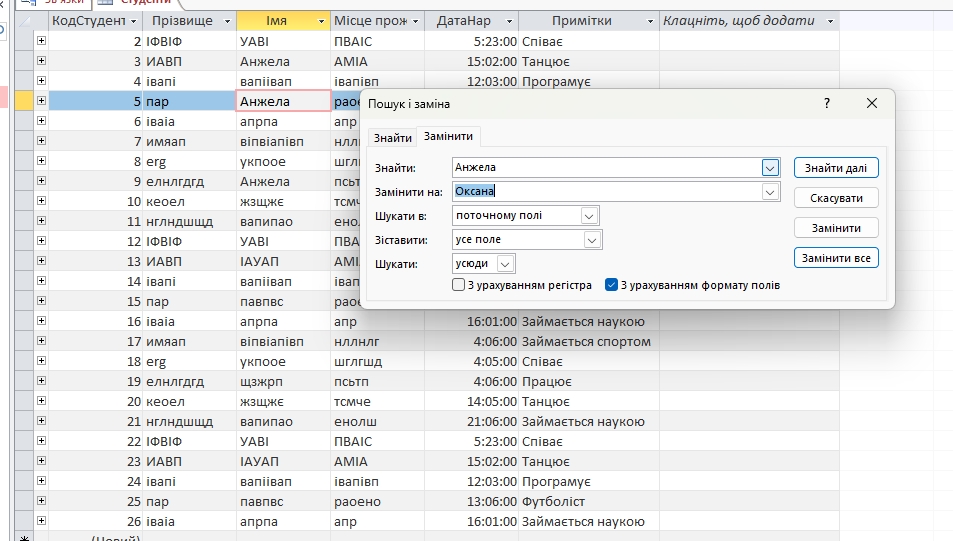
У нових записах вибірково замінить окремі прізвища на нові зі списку, поміняйте оцінки.

10

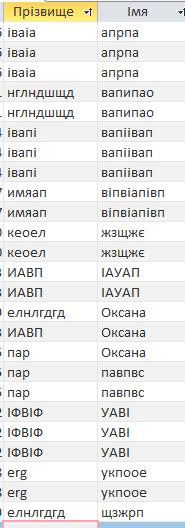
Відкрийте таблицю Студенти, додайте нові записи (до 25) шляхом копіювання та вставляння.

11

У нових записах замініть прізвища на інші, кілька жіночих імен замініть на ім’я *Анжела*, необов’язкові поля Адреса, ДатаНар і Примітка можна не змінювати. За допомогою команди ***Замінити*** у таблиці змініть ім’я Анжела на ім’я Оксана.

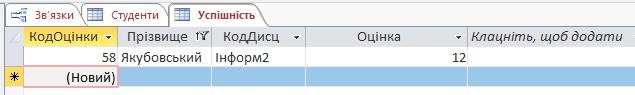
12

Впорядкуйте в таблиці Студенти дані одночасно за полями Прізвище та Імя за алфавітом.

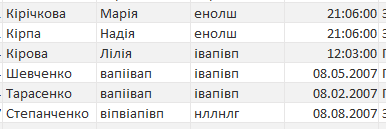


13

У таблиці Успішність виконайте фільтрування за виділеним власним прізвищем.

14

Доповніть таблицю Студенти, щоб у ній були два-три записи з прізвищами, що починаються з літери «К», та іменами, що закінчуються літерами «ія» (наприклад, Марія, Надія, Лілія), а також два-три прізвища із закінченням «енко», рік народження — 2007.



# 3 РОЗДІЛ "ПРОГРАМУВАННЯ С++"

## **3.1 Теоретичні відомості**

C++ — це мова програмування загального призначення, яка підтримує різні парадигми програмування, зокрема процедурне, об'єктно-орієнтоване та узагальнене програмування. Вона була розроблена Б'ярне Страуструпом як розширення мови C. Ось основні концепції C++:

### 3.1.1 Загальні поняття. Елементи мови С++ - константи, змінні, операції, перетворення типів

Змінні:

Іменовані області пам'яті, де зберігаються значення.

Оголошуються з вказанням типу.

Наприклад: int age = 25;.

Операції:

Арифметичні: +, -, \*, /, %.

Логічні: &&, ||, !.

Порівняння: ==, !=, <, >, <=, >=.

Присвоєння: =, +=, -=, \*=, /=, %=.

Перетворення типів:

Неявні: автоматичні перетворення типів компілятором (наприклад, з int в float).

Явні: примусове перетворення типів за допомогою операторів або функцій (наприклад, static\_cast<int>(value)).

Ці основні елементи мови C++ дозволяють створювати ефективні та структуровані програми

### 3.1.2 Арифметичні операції

Додавання: +

Віднімання: -

Множення: \*

Ділення: /

Ділення за модулем (остача): %

### 3.1.3 Поняття вхідного та вихідного потоку, найпростіші математичні функції

Вхідний потік (cin):

Потік для отримання введених даних з клавіатури.

Використовується з оператором >>.

Вихідний потік (cout):

Потік для виведення даних на екран консолі.

Використовується з оператором <<.

Найпростіші математичні функції

1. Додавання: a + b
2. Віднімання: a - b
3. Множення: a \* b
4. Ділення: a / b
5. Модуль числа: abs(x)
6. Піднесення до степеня: pow(x, n)
7. Корінь квадратний: sqrt(x)

### 3.1.4 Програмування лінійних та розгалужених обчислювальних процесів

1. **Лінійний процес**:
   * Виконується послідовно від початку до кінця без переривань.
   * Приклад: обчислення суми двох чисел.

**Програмування розгалужених обчислювальних процесів:**

1. Розгалужений процес:
   * Має різні шляхи виконання залежно від умов.
   * Використовує умовні оператори (if, else, switch).
   * Приклад: обчислення максимуму з двох чисел залежно від умови.

### 3.1.5 Програмування циклічних обчислювальних процесів

**Циклічний процес**:

* Виконується декілька разів до досягнення заданої умови.
* Використовується циклічний оператор (for, while, do-while).
* Приклад: обчислення суми чисел від 1 до N.

## **3.2 Практичне завдання**

Написати код на мові програмування C++ для розрахунку по формулам

### 3.2.1 Завдання

Завдання: Обчислити і вивести на екран у табличному вигляді значення функції *f(x)* на заданому інтервалі зміни значень аргументу *х* від *xпоч* до *xкін* з кроком *h.*

Коефіцієнти *a, b, c –* дійсні числа. Значення *a, b, c, xпоч*, *xкін*, *h* вводити з клавіатури. Передбачити перевірку допустимості введених значень.

### 3.2.2 Постановка задачі

Вхідні дані

A =5

B = 4

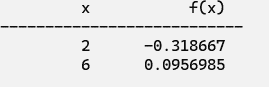
C = 8

Початковий X = 2

Кінцевий X = 8

Крок = 4

Вихідні дані



Вхідні дані

A =0

B = 5

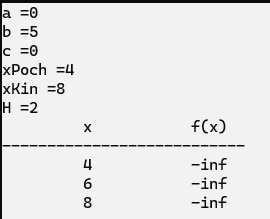
C = 0

Початковий X = 4

Кінцевий X = 8

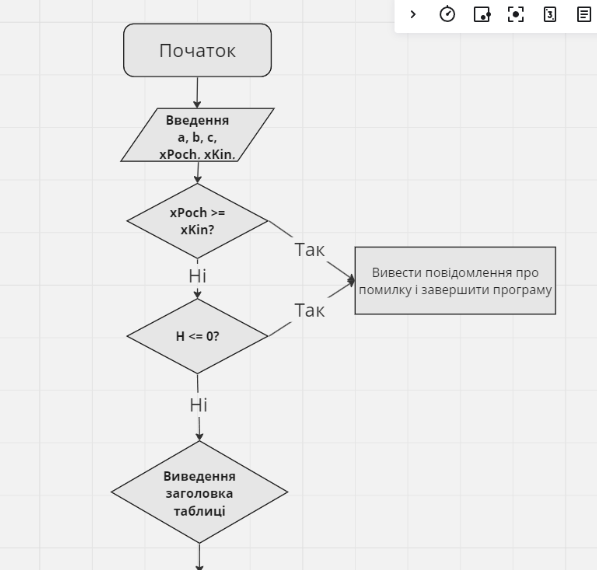
Крок = 2

Вихідні дані



Блок-схема алгоритму роботи програми представлена на Рисунок 1

**Блок-схема**



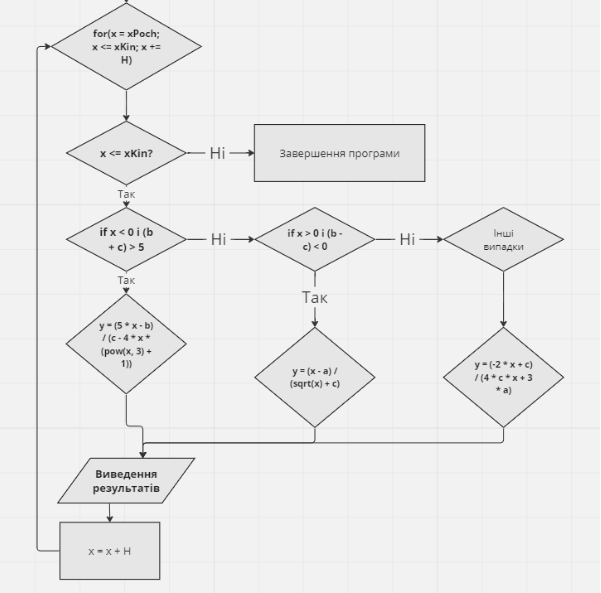


Рисунок 1

### 3.2.3 Код програми

#include <iostream> // Підключаємо бібліотеку для вводу-виводу

#include <cmath> // Підключаємо бібліотеку для математичних обчислень

#include <iomanip> // Підключаємо бібліотеку для форматування виведення

#include <windows.h> // Підключаємо бібліотеку для налаштування консолі (колір)

using namespace std; // Використовуємо стандартний простір імен

// Оголошення функції Fun1 для обчислення значень функції на заданому інтервалі

int f(float a, float b, float c, float xPoch, float xKin, float H);

int main()

{

system("color F0"); // Встановлюємо колір фону консолі

float a, b, c, xPoch, xKin, H; // Оголошуємо змінні для зберігання введених даних

cout << "a ="; cin >> a; // Запитуємо і зчитуємо значення a

cout << "b ="; cin >> b; // Запитуємо і зчитуємо значення b

cout << "c ="; cin >> c; // Запитуємо і зчитуємо значення c

cout << "xPoch ="; cin >> xPoch; // Запитуємо і зчитуємо початкове значення x

cout << "xKin ="; cin >> xKin; // Запитуємо і зчитуємо кінцеве значення x

cout << "H ="; cin >> H; // Запитуємо і зчитуємо крок зміни x

// Викликаємо функцію Fun1 з введеними параметрами

int ret = f(a, b, c, xPoch, xKin, H);

return 0; // Повертаємо 0 для позначення успішного завершення програми

}

// Реалізація функції Fun1 для обчислення значень функції на заданому інтервалі

int f(float a ,float b ,float c ,float xPoch ,float xKin ,float H)

{

float x, y; // Оголошуємо змінні для зберігання поточних значень x та y

// Перевірка допустимості введених значень

if (xPoch >= xKin) {

cerr << "Помилка: xPoch повинно бути менше xKin." << endl; // Виводимо повідомлення про помилку

return 1; // Повертаємо 1 для позначення помилки

}

if (H <= 0) {

cerr << "Помилка: крок H повинен бути більшим за 0." << endl; // Виводимо повідомлення про помилку

return 1; // Повертаємо 1 для позначення помилки

}

// Виведення заголовків таблиці

cout << setw(10) << "x" << setw(15) << "f(x)" << endl; // Виводимо заголовки стовпців

cout << "---------------------------" << endl; // Виводимо роздільник

// Цикл для обчислення значень f(x) на заданому інтервалі

for (x = xPoch; x <= xKin; x += H) {

// Обчислення значення функції залежно від умови

if (x < 0 && (b + c) > 5) {

y = (5 \* x - b) / (c - 4 \* x \* (pow(x, 3) + 1)); // Перша умова

}

else if (x > 0 && (b - c) < 0) {

y = (x - a) / (sqrt(x) + c); // Друга умова

}

else {

y = (-2 \* x + c) / (4 \* c \* x + 3 \* a); // Всі інші випадки

}

// Виведення значень x та f(x)

cout << setw(10) << x << setw(15) << y << endl; // Виводимо поточні значення x та f(x)

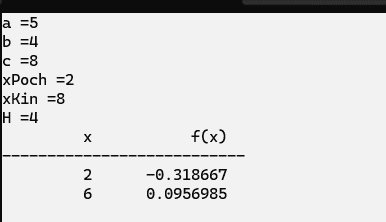
}

return 0; // Повертаємо 0 для позначення успішного завершення функції

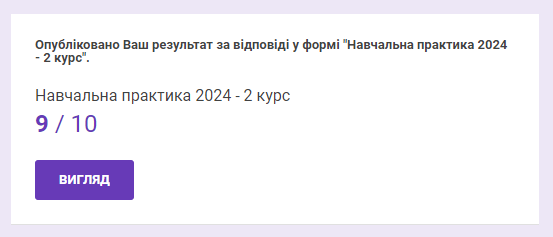
}

### 3.2.4 Скріншот виконання програми

Скріншот



## **3.3 Виконання тестів**



# 4 РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++"

## **4.1 Тема 1. Введення та виведення у базових консольних програмах**

Програма для виведення тексту

#include <iostream>

int main() {

std::cout << "Привіт, світ!" << std::endl;

return 0;

}

Введення та виведення чисел

#include <iostream>

int main() {

int a, b;

std::cout << "Введіть два числа: ";

std::cin >> a >> b;

std::cout << "Сума: " << a + b << std::endl;

return 0;

}

Введення рядка

#include <iostream>

#include <string>

int main() {

std::string name;

std::cout << "Введіть ваше ім'я: ";

std::cin >> name;

std::cout << "Привіт, " << name << "!" << std::endl;

return 0;

}

Введення і виведення чисел з плаваючою комою

#include <iostream>

int main() {

double num1, num2;

std::cout << "Введіть два числа з плаваючою комою: ";

std::cin >> num1 >> num2;

std::cout << "Сума: " << num1 + num2 << std::endl;

return 0;

}

## **4.2 Тема 2. Структури даних для управління станом програми**

Приклад використання структур даних:

Масив board: відображення та оновлення стану дошки.

void drawBoard() {

cout << "-------------\n";

for (int i = 0; i < 3; i++) {

cout << "| ";

for (int j = 0; j < 3; j++) {

cout << board[i][j] << " | ";

}

cout << "\n-------------\n";

}

}

Змінні current\_marker і current\_player: збереження та зміна поточного гравця і його маркера.

void swapPlayerAndMarker() {

current\_marker = (current\_marker == 'X') ? '0' : 'X';

current\_player = (current\_player == 1) ? 2 : 1;

}

Лічильники перемог і нічиїх: підрахунок результатів гри.

if (player\_won == 1) {

player1\_wins++;

} else if (player\_won == 2) {

player2\_wins++;

} else {

ties++;

}

Вектор game\_logs: збереження ходів гравців для подальшого аналізу або виведення.

game\_logs.push\_back("Player " + to\_string(current\_player) + " placed " + current\_marker + " in slot " + to\_string(slot));

JSON об'єкт game\_state: збереження стану гри у файл.

void saveGameState() {

json game\_state;

game\_state["current\_player"] = current\_player;

game\_state["current\_marker"] = current\_marker;

game\_state["player1\_wins"] = player1\_wins;

game\_state["player2\_wins"] = player2\_wins;

game\_state["ties"] = ties;

ofstream file("game\_state.json");

if (file.is\_open()) {

file << game\_state.dump(4);

file.close();

} else {

cout << "Unable to open file for writing game state.\n";

}

}

Ці структури даних забезпечують ефективне управління станом гри, збереження прогресу, обробку введення/виведення та підрахунок результатів.

## **4.3 Тема 3. Робота з файлами: запис та читання**

Запис у текстовий файл

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ofstream outFile("example.txt"); // Відкриття файлу для запису

if (outFile.is\_open()) {

outFile << "Це перший рядок у файлі.\n";

outFile << "Це другий рядок у файлі.\n";

outFile.close(); // Закриття файлу

cout << "Дані успішно записано у файл.\n";

} else {

cout << "Не вдалося відкрити файл для запису.\n";

}

return 0;

}

Читання з текстового файлу

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

ifstream inFile("example.txt"); // Відкриття файлу для читання

if (inFile.is\_open()) {

string line;

while (getline(inFile, line)) { // Читання рядків з файлу

cout << line << endl; // Виведення кожного рядка на екран

}

inFile.close(); // Закриття файлу

} else {

cout << "Не вдалося відкрити файл для читання.\n";

}

return 0;

}

Запис і читання даних у форматі JSON

#include <iostream>

#include <fstream>

#include <nlohmann/json.hpp>

using namespace std;

using json = nlohmann::json;

int main() {

// Запис у JSON файл

json j;

j["name"] = "John";

j["age"] = 30;

j["is\_student"] = false;

ofstream outFile("data.json");

if (outFile.is\_open()) {

outFile << j.dump(4); // Запис JSON з відступом у 4 пробіли

outFile.close();

cout << "Дані успішно записано у JSON файл.\n";

} else {

cout << "Не вдалося відкрити JSON файл для запису.\n";

}

// Читання з JSON файлу

ifstream inFile("data.json");

if (inFile.is\_open()) {

json j\_in;

inFile >> j\_in; // Читання JSON даних з файлу

cout << "Ім'я: " << j\_in["name"] << endl;

cout << "Вік: " << j\_in["age"] << endl;

cout << "Студент: " << (j\_in["is\_student"] ? "Так" : "Ні") << endl;

inFile.close();

} else {

cout << "Не вдалося відкрити JSON файл для читання.\n";

}

return 0;

}

Ці приклади демонструють, як можна працювати з файлами у C++ для запису та читання текстових, двійкових та JSON даних.

## **4.4 Тема 4. Формати файлів та їх обробка**

**JSON (JavaScript Object Notation)**

Переваги:

Легкість читання: JSON файли мають зрозумілий людині формат, що робить їх легкими для читання та налагодження.

Сумісність: JSON є універсальним форматом, який підтримується багатьма мовами програмування.

Гнучкість: JSON дозволяє вкладати структури даних, що зручно для складних ієрархій.

Недоліки:

Обмеження типів даних: JSON підтримує лише базові типи даних (рядки, числа, масиви, об'єкти, булеві значення, null).

Продуктивність: Обробка великих JSON файлів може бути повільнішою порівняно з іншими форматами, такими як CSV.

Приклад обробки JSON у C++:

#include <iostream>

#include <fstream>

#include <nlohmann/json.hpp>

using namespace std;

using json = nlohmann::json;

int main() {

// Запис у JSON файл

json j;

j["name"] = "John";

j["age"] = 30;

j["is\_student"] = false;

ofstream outFile("data.json");

if (outFile.is\_open()) {

outFile << j.dump(4);

outFile.close();

cout << "Дані успішно записано у JSON файл.\n";

} else {

cout << "Не вдалося відкрити JSON файл для запису.\n";

}

// Читання з JSON файлу

ifstream inFile("data.json");

if (inFile.is\_open()) {

json j\_in;

inFile >> j\_in;

cout << "Ім'я: " << j\_in["name"] << endl;

cout << "Вік: " << j\_in["age"] << endl;

cout << "Студент: " << (j\_in["is\_student"] ? "Так" : "Ні") << endl;

inFile.close();

} else {

cout << "Не вдалося відкрити JSON файл для читання.\n";

}

return 0;

}

**CSV (Comma-Separated Values)**

Переваги:

Простота: CSV файли мають просту структуру, що робить їх легкими для читання і написання.

Широка підтримка: Багато програм (наприклад, Excel) та бібліотек підтримують CSV формат.

Висока продуктивність: Обробка CSV файлів зазвичай швидша, ніж JSON або XML.

Недоліки:

Обмеження структури: CSV не підтримує вкладені структури, що робить його непридатним для складних даних.

Неявні типи даних: Всі дані у CSV є рядками, тому необхідно додатково обробляти і перетворювати їх.

Приклад обробки CSV у C++:

#include <iostream>

#include <fstream>

#include <vector>

#include <sstream>

using namespace std;

int main() {

// Запис у CSV файл

ofstream outFile("data.csv");

if (outFile.is\_open()) {

outFile << "Name,Age,IsStudent\n";

outFile << "John,30,false\n";

outFile << "Jane,25,true\n";

outFile.close();

cout << "Дані успішно записано у CSV файл.\n";

} else {

cout << "Не вдалося відкрити CSV файл для запису.\n";

}

// Читання з CSV файлу

ifstream inFile("data.csv");

if (inFile.is\_open()) {

string line;

while (getline(inFile, line)) {

stringstream ss(line);

string name, age, is\_student;

getline(ss, name, ',');

getline(ss, age, ',');

getline(ss, is\_student, ',');

cout << "Ім'я: " << name << ", Вік: " << age << ", Студент: " << is\_student << endl;

}

inFile.close();

} else {

cout << "Не вдалося відкрити CSV файл для читання.\n";

}

return 0;

}

**XML (eXtensible Markup Language)**

Переваги:

Гнучкість: XML може зберігати складні вкладені структури даних.

Сумісність: Широко використовується у веб-сервісах і багатьох програмах.

Схемна валідація: Можливість використання схем (XSD) для валідації структури даних.

Недоліки:

Складність: Більш складний формат порівняно з JSON та CSV.

Розмір: XML файли зазвичай більші через використання тегів.

Продуктивність: Обробка XML може бути повільнішою через складність парсингу.

Приклад обробки XML у C++ з використанням бібліотеки pugixml:

#include <iostream>

#include <pugixml.hpp>

using namespace std;

using namespace pugi;

int main() {

// Створення XML документа

xml\_document doc;

xml\_node root = doc.append\_child("People");

xml\_node person1 = root.append\_child("Person");

person1.append\_child("Name").text().set("John");

person1.append\_child("Age").text().set(30);

person1.append\_child("IsStudent").text().set(false);

xml\_node person2 = root.append\_child("Person");

person2.append\_child("Name").text().set("Jane");

person2.append\_child("Age").text().set(25);

person2.append\_child("IsStudent").text().set(true);

doc.save\_file("data.xml");

cout << "Дані успішно записано у XML файл.\n";

// Читання XML документа

xml\_document doc\_in;

if (doc\_in.load\_file("data.xml")) {

for (xml\_node person = doc\_in.child("People").child("Person"); person; person = person.next\_sibling("Person")) {

cout << "Ім'я: " << person.child("Name").text().as\_string() << ", Вік: " << person.child("Age").text().as\_int() << ", Студент: " << (person.child("IsStudent").text().as\_bool() ? "Так" : "Ні") << endl;

}

} else {

cout << "Не вдалося відкрити XML файл для читання.\n";

}

return 0;

}

Висновок

Кожен формат файлів має свої переваги та недоліки:

JSON підходить для зберігання складних вкладених структур і має зрозумілий формат, але може бути менш продуктивним для великих обсягів даних.

CSV добре підходить для простих табличних даних, забезпечує високу продуктивність, але не підтримує складні структури.

XML є гнучким і підходить для складних ієрархій, але має більший розмір файлів і складніший парсинг.

Вибір формату залежить від конкретних вимог до структури даних, продуктивності та зручності використання.

## **4..5 Тема 5. Системи контролю версій (VCS)**

**GitHub**

GitHub — це веб-платформа для хостингу репозиторіїв, заснована на системі контролю версій Git. GitHub пропонує інструменти для управління проєктами та спільної роботи над кодом, включаючи трекінг помилок, запити на злиття, CI/CD інтеграцію та інші функції.

Основні поняття GitHub

Репозиторій (Repository):

Репозиторій — це місце, де зберігаються всі файли проекту, включаючи історію їх змін.

Гілка (Branch):

Гілка — це ізольована лінія розробки. Основна гілка зазвичай називається main або master. Інші гілки створюються для розробки нових функцій або виправлення помилок без впливу на основний код.

Коміт (Commit):

Коміт — це знімок проекту в конкретний момент часу. Коміти містять зміни, які були внесені в проект, і повідомлення, що описує ці зміни.

Запит на злиття (Pull Request):

Запит на злиття дозволяє об'єднати зміни з однієї гілки до іншої. Під час створення запиту на злиття інші учасники команди можуть переглянути зміни, обговорити їх і затвердити або відхилити.

Форк (Fork):

Форк — це копія репозиторію, яка дозволяє вносити зміни без впливу на оригінальний репозиторій. Після внесення змін у форкований репозиторій можна створити запит на злиття для об'єднання змін з оригінальним репозиторієм.

## **4.6 Тема 6. Мова Markdown для опису проекті**

Markdown — це легка мова розмітки, яка дозволяє швидко і просто форматувати текст для веб-документів. Markdown широко використовується на платформах для спільної розробки програмного забезпечення, таких як GitHub, для написання README файлів, документації та коментарів.

**Основи Markdown**

Заголовки

Заголовки створюються за допомогою символів #. Чим більше символів #, тим нижчий рівень заголовка.

# Заголовок рівня 1

## Заголовок рівня 2

### Заголовок рівня 3

Форматування тексту

Жирний текст: обгортається подвійними зірочками “\*\*” або подвійними підкресленнями” \_\_”.

\*\*Це жирний текст\*\*

\_\_Це також жирний текст\_\_

Курсив: обгортається одинарними зірочками “\*” або одинарними підкресленнями “\_”

\*Це курсив\*

\_Це також курсив\_

Закреслений текст: обгортається подвійними тильдами “~~”.

~~Це закреслений текст~~

Списки

Маркірований список: використовується “-“, “\*” або +.

- Елемент списку 1

- Елемент списку 2

- Піделемент

Нумерований список: використовуються числа з крапками.

1. Перший елемент

2. Другий елемент

1. Піделемент

Посилання

Звичайне посилання:

[Назва посилання](URL)

Посилання з підказкою:

[Назва посилання](URL "Підказка")

Вставка зображення:

![Альтернативний текст](URL-зображення)

Приклад README файлу на Markdown для GitHub

# Назва проекту

Короткий опис вашого проекту.

## Початок роботи

Інструкції з налаштування проекту на локальному комп'ютері.

### Передумови

Що потрібно для встановлення програмного забезпечення та як це встановити.

```bash

sudo apt-get install example

## **4.7 Розроблення ігрової програми**

Розроблена гра Хрестики Нулики на мові програмування C++

### 4.7.1 Код програми

#include <iostream> // Including the library for input and output streams

#include <fstream> // Including the library for file streams

#include <string> // Including the string library for using to\_string

#include <nlohmann/json.hpp> // Including the JSON library

#include <vector> // Including the vector library for storing moves

using namespace std;

using json = nlohmann::json; // Using the nlohmann JSON namespace

// Initializing the game board

char board[3][3] = { {'1', '2', '3'},

{'4', '5', '6'},

{'7', '8', '9'} };

char current\_marker; // Variable to store the marker of the current player

int current\_player; // Variable to store the number of the current player

int player1\_wins = 0; // Counter for Player 1 wins

int player2\_wins = 0; // Counter for Player 2 wins

int ties = 0; // Counter for ties

vector<string> game\_logs; // Vector to store the game logs

void drawBoard() {

cout << "-------------\n"; // Top border of the board

for (int i = 0; i < 3; i++) { // Iterating through the rows of the board

cout << "| "; // Left border of the row

for (int j = 0; j < 3; j++) { // Iterating through the columns of the board

cout << board[i][j] << " | "; // Printing the cell value and the right border of the cell

}

cout << "\n-------------\n"; // Bottom border of the row

}

}

// Function to place the marker on the board

bool placeMarker(int slot) {

int row = (slot - 1) / 3; // Calculating the row based on the entered slot

int col = (slot - 1) % 3; // Calculating the column based on the entered slot

// Checking if the cell is not occupied

if (board[row][col] != 'X' && board[row][col] != '0') {

board[row][col] = current\_marker; // Placing the marker on the board

return true; // Successfully placed the marker

}

else {

return false; // Cell is occupied

}

}

// Function to determine the winner

int winner() {

// Checking rows for a winning combination

for (int i = 0; i < 3; i++) {

if (board[i][0] == board[i][1] && board[i][1] == board[i][2]) {

return current\_player; // Current player wins

}

}

// Checking columns for a winning combination

for (int i = 0; i < 3; i++) {

if (board[0][i] == board[1][i] && board[1][i] == board[2][i]) {

return current\_player; // Current player wins

}

}

// Checking diagonals for a winning combination

if (board[0][0] == board[1][1] && board[1][1] == board[2][2]) {

return current\_player; // Current player wins

}

if (board[0][2] == board[1][1] && board[1][1] == board[2][0]) {

return current\_player; // Current player wins

}

return 0; // No winner

}

// Function to swap the current player and their marker

void swapPlayerAndMarker() {

current\_marker = (current\_marker == 'X') ? '0' : 'X'; // Switching the marker

current\_player = (current\_player == 1) ? 2 : 1; // Switching the player number

}

// Function to save the game state to a JSON file

void saveGameState() {

json game\_state; // Creating a JSON object

game\_state["current\_player"] = current\_player;

game\_state["current\_marker"] = current\_marker;

game\_state["player1\_wins"] = player1\_wins;

game\_state["player2\_wins"] = player2\_wins;

game\_state["ties"] = ties;

ofstream file("game\_state.json"); // Opening the JSON file for writing

if (file.is\_open()) {

file << game\_state.dump(4); // Dumping the JSON object to the file with 4-space indentation

file.close(); // Closing the file

}

else {

cout << "Unable to open file for writing game state.\n";

}

}

// Function to load the game state from a JSON file

void loadGameState() {

ifstream file("game\_state.json"); // Opening the JSON file for reading

if (file.is\_open()) {

json game\_state;

file >> game\_state; // Reading the JSON object from the file

current\_player = game\_state.value("current\_player", 1); // Loading the current player, default is 1

current\_marker = game\_state.value("current\_marker", 'X'); // Loading the current marker, default is 'X'

player1\_wins = game\_state.value("player1\_wins", 0);

player2\_wins = game\_state.value("player2\_wins", 0);

ties = game\_state.value("ties", 0);

file.close(); // Closing the file

}

else {

cout << "No previous game state found. Starting a new game.\n";

}

}

// Function to save the game logs to a CSV file

void saveGameLogs() {

ofstream file("game\_logs.csv", ios::app); // Opening the CSV file in append mode

if (file.is\_open()) {

for (const string& log : game\_logs) {

file << log << endl; // Writing each log entry to the file

}

file.close(); // Closing the file

}

else {

cout << "Unable to open file for writing game logs.\n";

}

}

// Function to run the game

void game() {

loadGameState(); // Load the game state from the JSON file

char marker\_p1; // Variable to store the marker of the first player

do {

cout << "Player 1, choose your marker (X or 0): "; // Prompting the first player to choose their marker

cin >> marker\_p1; // Reading the marker

} while (marker\_p1 != 'X' && marker\_p1 != '0'); // Ensuring the marker is either 'X' or '0'

current\_player = 1; // Setting the first player

current\_marker = marker\_p1; // Setting the marker of the first player

drawBoard(); // Displaying the initial board

int player\_won = 0; // Variable to store the game result (0 means no winner yet)

for (int i = 0; i < 9 && player\_won == 0; i++) { // Loop for making moves, max 9 moves

int slot; // Variable to store the chosen slot

cout << "Player " << current\_player << "'s turn. Enter your slot: "; // Prompting the current player to enter a slot

cin >> slot; // Reading the chosen slot

// Checking if the slot is valid and not occupied

if (slot < 1 || slot > 9 || !placeMarker(slot)) {

cout << "Invalid slot! Try again.\n"; // Invalid slot message

i--; // Decrease i to repeat the iteration for the same player

continue; // Skip the rest of the loop and continue with the next iteration

}

drawBoard(); // Displaying the updated board

saveGameState(); // Saving the game state after each move

game\_logs.push\_back("Player " + to\_string(current\_player) + " placed " + current\_marker + " in slot " + to\_string(slot)); // Logging the move

player\_won = winner(); // Checking for a winner

if (player\_won == 0) swapPlayerAndMarker(); // If no winner, switch player and marker

}

// Displaying the result of the game

if (player\_won) {

cout << "Player " + to\_string(player\_won) + " wins! Congratulations!\n";

game\_logs.push\_back("Player " + to\_string(player\_won) + " wins!");

if (player\_won == 1) {

player1\_wins++; // Incrementing Player 1's win counter

}

else {

player2\_wins++; // Incrementing Player 2's win counter

}

}

else {

cout << "It's a tie!\n";

game\_logs.push\_back("It's a tie!");

ties++; // Incrementing the tie counter

}

//saveGameResult(player\_won); // Saving the game result to a CSV file

saveGameLogs(); // Saving the game logs to a CSV file

saveGameState(); // Saving the final game state including counters to JSON

}

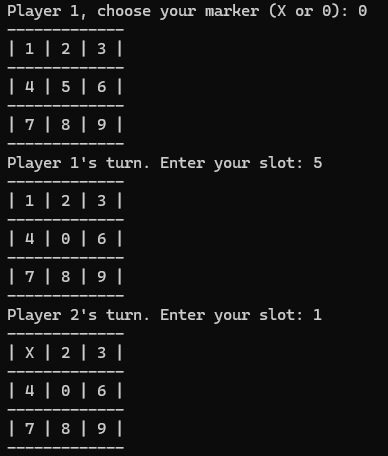
int main() {

game(); // Calling the game function

return 0; // Ending the program

}

### 4.7.2 Скріншоти виконання програми



# 5 РОЗДІЛ ”ВЕБ-ДИЗАЙН ТА HTML”

## **5.1 Мова розмітки HTML та структура Веб-сторінки**

Основи HTML

HTML (HyperText Markup Language) - це основна мова для створення веб-сторінок та веб-додатків. Вона використовується для структуризації вмісту та визначення елементів на сторінці. HTML складається з набору елементів, які представлені у вигляді тегів.

Структура HTML-документа

Типовий HTML-документ складається з наступних частин:

DOCTYPE декларація: Вказує тип документа і версію HTML.

Елемент <html>: Кореневий елемент, що містить весь вміст HTML-документа.

Елемент <head>: Містить метадані про документ (наприклад, заголовок, стилі, метаінформацію).

Елемент <body>: Містить весь вміст, який відображається на веб-сторінці (тексти, зображення, посилання тощо).

## **5.2 Розмітка тексту за допомогою HTML. Зображення та посилання**

HTML надає різноманітні теги для форматування тексту:

* **Заголовки**: <h1> до <h6>

<h1>Головний заголовок</h1>

<h2>Підзаголовок</h2>

* **Параграфи**: <p>

html

Копіювати код

<p>Це параграф тексту.</p>

* **Жирний текст**: <strong>

html

Копіювати код

<p>Це <strong>жирний текст</strong>.</p>

## **5.3 Каскадні таблиці стилів CSS та їх структура**

CSS (Cascading Style Sheets) - мова стилів для оформлення HTML-документів.

**Структура CSS**

CSS правило складається з селектора та набору декларацій:

селектор {

властивість: значення;

}

**Селектори**

* **Теги**: p { color: blue; }
* **Класи**: .class-name { font-size: 20px; }
* **ID**: #id-name { background-color: yellow; }
* **Комбіновані**: div p { margin: 10px; }

**Типи CSS**

1. **Вбудовані**: <p style="color: red;">Текст</p>
2. **Внутрішні**: В межах тегу <style> в HTML:

<style>

p { color: blue; }

</style>

1. **Зовнішні**: У файлі .css, підключається через <link>:

<link rel="stylesheet" href="styles.css">

**Каскадність та успадкування**

* **Каскадність**: Стилі переписуються залежно від специфічності та порядку.
* **Успадкування**: Деякі властивості успадковуються від батьківських елементів.

**Приклад:**

<link rel="stylesheet" href="styles.css">

/\* styles.css \*/

p {

color: green;

}

CSS дозволяє створювати красиві та структуровані веб-сторінки за допомогою стилів.

## **5.4 Спадковості і каскадування за допомогою CSS**

**Спадковість**

Деякі CSS властивості передаються від батьківських елементів до дочірніх.

body {

font-family: Arial, sans-serif;

}

p {

color: blue;

}

**Каскадування**

CSS правила застосовуються за пріоритетом селекторів і порядком визначення.

p {

color: blue; /\* менш специфічний \*/

}

p.special {

color: red; /\* більш специфічний \*/

}

**Пріоритет селекторів**

1. Вбудовані стилі (inline): style="color: green;"
2. ID селектори: #id
3. Класові селектори: .class
4. Селектори тегів: p

## **5.5 Практична частина**

Зробити сайт

### 5.5.1 Постановки задачі

Зробити сайт про себе з посиланнями на свої соц мережі

### 5.5.2 HTML документ

**HTML**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel = "stylesheet" href = "statik/style.css">

    <title>Document</title>

    <script src="https://kit.fontawesome.com/4d69be9202.js" crossorigin="anonymous"></script>

</head>

<body>

    <div class="main">

        <img src="statik/я2.png">

        <h1>Максим Якубовський</h1>

        <p class="p1">Студент</p>

        <div class="title"><b>Резюме</b><hr></div>

        <div class="text">3 роки вивчав мову програмування python

            маю декілька меленьких проектів за цей час.</div>

        <div class="title"><b>Освіта</b><hr></div>

        <div class="text">9 класів школи <br>Наразі навчаюся в Оптико

            Механічному Коледжі</div>

        <div class="title"><b>Хоббі</b><hr></div>

        <div class="text">Грати у комп'ютерні ігри</div>

        <a href="mailto: maximyakubovskiy12042007@gmail.com" target="\_blank"><div><button><i class="fa-regular fa-envelope"></i> Gmail</button></div></a>

        <a href="https://github.com/fd-fg" target="\_blank"><div><button><i class="fa-brands fa-github"></i> Git hub</button></div></a>

        <a href="https://www.hoyolab.com/accountCenter/postList?id=48007247" target="\_blank"><div><button><i class="fa-solid fa-gamepad"></i> Hoyolab</button></div></a>

    </div>

</body>

</html>

**CSS**

body{

    background-color: aliceblue;

    margin: 10%;

}

.main{

    margin: 15%;

    text-align: center;

}

h1{

    color:rgb(96, 96, 241);

    margin-bottom: 0;

}

hr{

    margin-top: -0;

}

.title{

    text-align: left;

    font-size: 20px;

}

.text{

    text-align: left;

}

.p1{

    margin-top: 0;

}

button{

    margin-top: 2%;

    width: 100%;

    height: 35px;

    border-radius: 7px;

    background-color: aliceblue;

}

button:hover{

    background-color: rgb(0, 0, 0);

    color: rgb(255, 255, 255);

}

img{

    border-radius: 100px;

}

.text{

    font-size: 18px;

    padding-bottom: 15px;

}

# ВИСНОВКИ

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ

1. ГОСТ 2.105-95 ЕСКД Общие требования к текстовым документам.

2. ДСТУ 1.5-93 Загальні вимоги до побудови, викладу, оформлення та змісту.

# ДОДАТКИ