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 |
| Оганнесян Гарік Паруйрович  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) | |  | |
| *Оганнесян Гарік Паруйрович* | | | | |
| (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-22

ОМФК.0121 НП 20

**ЗМІСТ**

[ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 5](#_Toc167919156)

[ВСТУП 6](#_Toc167919157)

[1 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ 7](#_Toc167919158)

[2 РОЗДІЛ "ІНФОРМАТИКА" 8](#_Toc167919159)

[**2.1** **Розроблення засобами MS Word документа складної структури** 8](#_Toc167919160)

[**2.2 Розв‘язок задач за допомогою засобів табличного редактора MS Excel** 8](#_Toc167919161)

[**2.3 Виконання** **практичних завдань за допомогою засобів табличного редактора MS Access** 8](#_Toc167919162)

[3 РОЗДІЛ "ПРОГРАМУВАННЯ С++" 9](#_Toc167919163)

[**3.1 Теоретичні відомості** 9](#_Toc167919164)

[3.1.1 Загальні поняття. Елементи мови С++ - константи, змінні, операції, перетворення типів 9](#_Toc167919165)

[3.1.2 Арифметичні операції 10](#_Toc167919166)

[3.1.3 Поняття вхідного та вихідного потоку, найпростіші математичні функції 10](#_Toc167919167)

[3.1.4 Програмування лінійних та розгалужених обчислювальних процесів 10](#_Toc167919168)

[3.1.5 Програмування циклічних обчислювальних процесів 11](#_Toc167919169)

[**3.2 Практичне завдання** 11](#_Toc167919170)

[3.2.1 Завдання 11](#_Toc167919171)

[3.2.2 Постановка задачі 11](#_Toc167919172)

[3.2.3 Код програми 12](#_Toc167919173)

[3.2.4 Скріншот виконання програми 14](#_Toc167919174)

[**3.3 Виконання тестів** 14](#_Toc167919175)

[4 РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++" 14](#_Toc167919176)

[**4.1 Тема 1. Введення та виведення у базових консольних програмах** 14](#_Toc167919177)

[**4.2 Тема 2. Структури даних для управління станом програми** 15](#_Toc167919178)

[**4.3 Тема 3. Робота з файлами: запис та читання** 15](#_Toc167919179)

[**4.4 Тема 4. Формати файлів та їх обробка** 15](#_Toc167919180)

[**4..5 Тема 5. Системи контролю версій (VCS)** 15](#_Toc167919181)

[**4.6 Тема 6. Мова Markdown для опису проекті** 16](#_Toc167919182)

[**4.7 Розроблення ігрової програми** 16](#_Toc167919183)

[4.7.1 Код програми 16](#_Toc167919184)

[4.7.2 Скріншоти виконання програми 16](#_Toc167919185)

[5 РОЗДІЛ ”ВЕБ-ДИЗАЙН ТА HTML” 17](#_Toc167919186)

[**5.1 Мова розмітки HTML та структура Веб-сторінки** 17](#_Toc167919187)

[**5.2 Розмітка тексту за допомогою HTML. Зображення та посилання** 17](#_Toc167919188)

[**5.3 Каскадні таблиці стилів CSS та їх структура** 18](#_Toc167919189)

[**5.4 Спадковості і каскадування за допомогою CSS** 19](#_Toc167919190)

[**5.5 Практична частина** 20](#_Toc167919191)

[5.5.1 Постановки задачі 20](#_Toc167919192)

[5.5.2 HTML документ 20](#_Toc167919193)

[ВИСНОВКИ 23](#_Toc167919194)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ 24](#_Toc167919195)

[ДОДАТКИ 25](#_Toc167919196)

ОМФК.0121 НП 20

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

# ВСТУП

Автор:

Дата:

Мета:

Перелік вивчених тем:

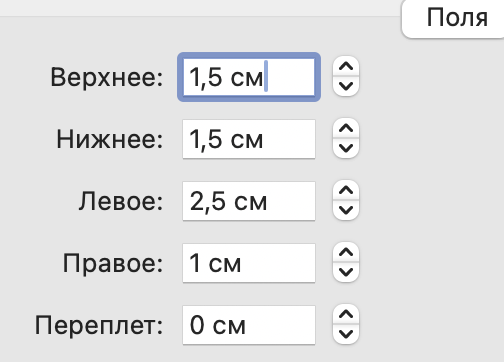
# 1 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ

# 2 РОЗДІЛ "ІНФОРМАТИКА"

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

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

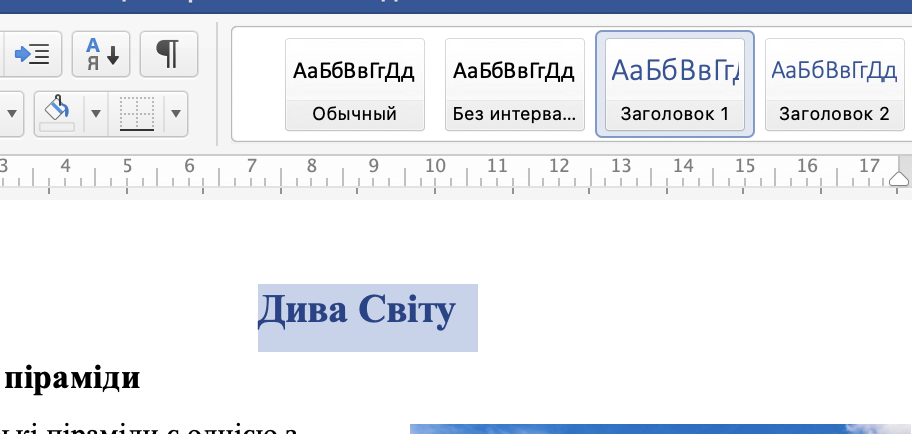
А4, поля документа: верхнє –1.5, нижнє – 1.5, ліве – 2.5, праве – 1.0 см).

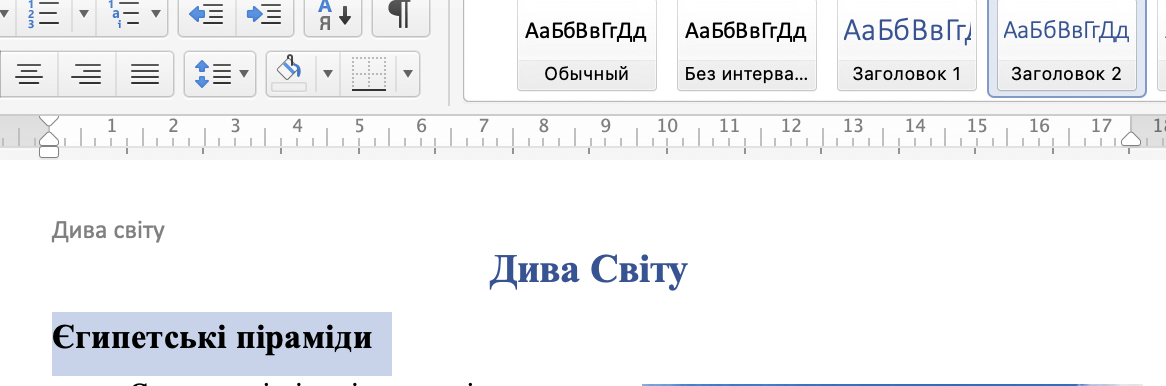


3. Відформатувати доданий текст: для заголовків першого рівня

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

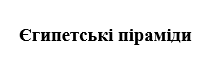
Roman, 18 пт, жирний, всі літери заглавні, колір темно-синій,

розташування посередині сторінки.



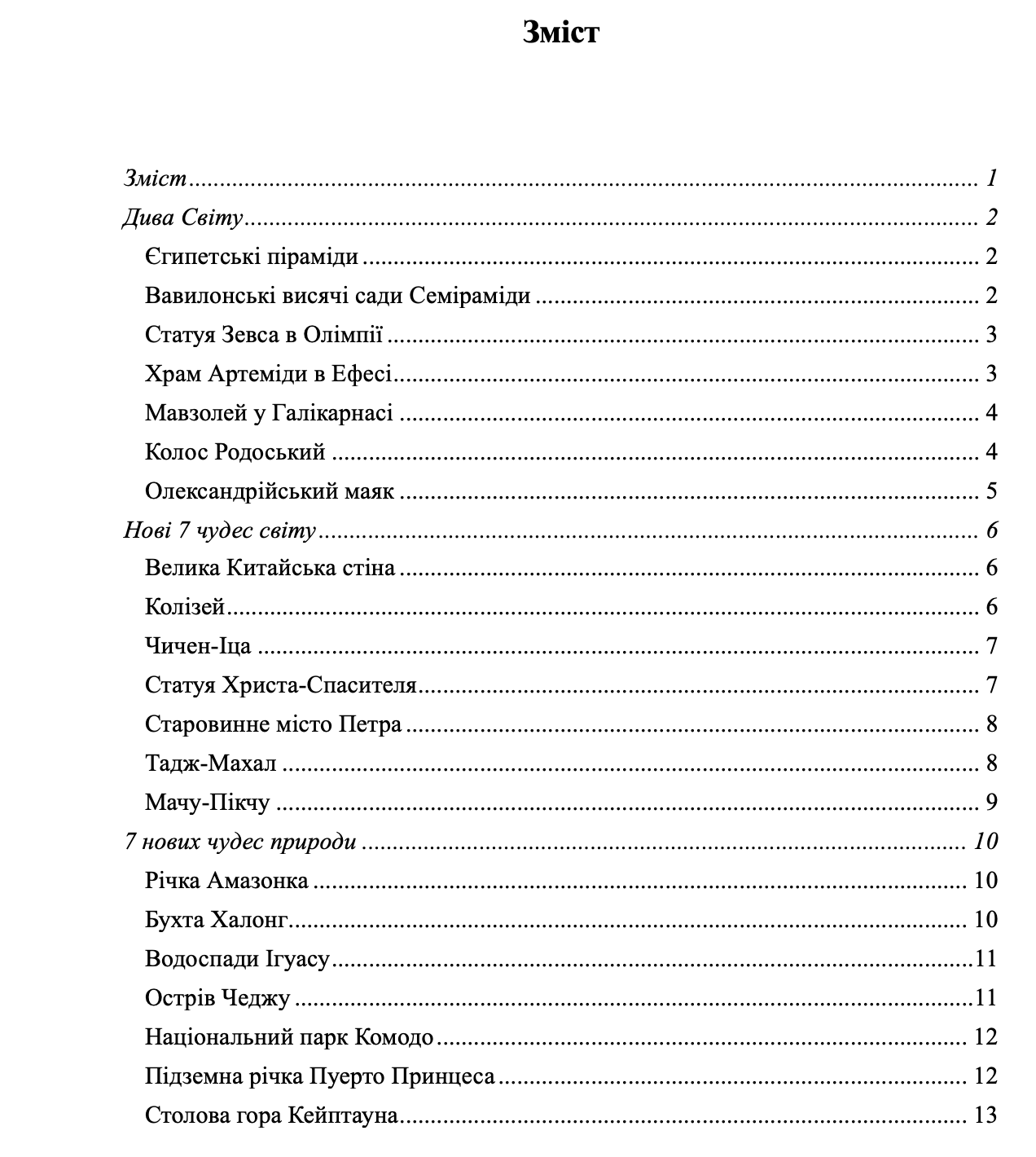
5. Змінити стиль заголовків другого рівня: шрифт Times New

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



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

зміст.

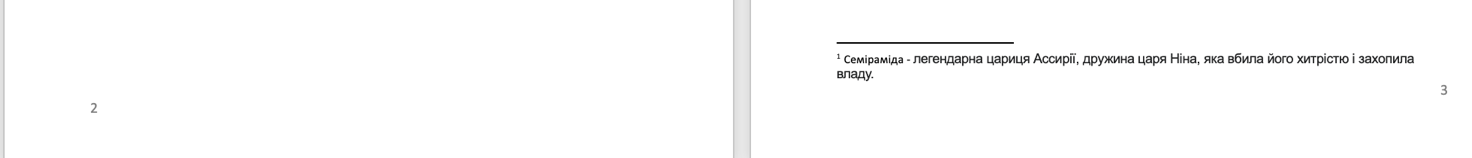


7. Додати титульну сторінку документа та оформити її.

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

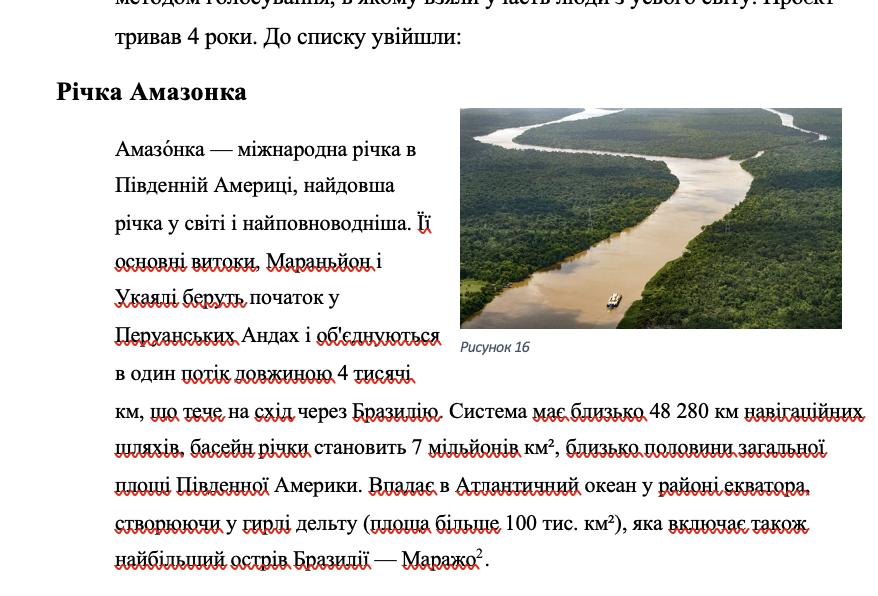
змісту.



10. Розширити розділ «7 НОВИХ ЧУДЕС ПРИРОДИ» додатковою

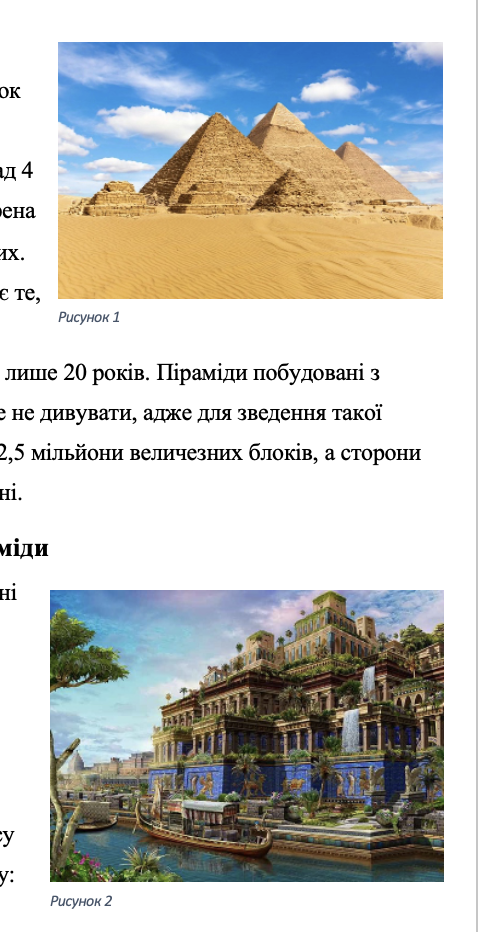
інформацією з мережі Інтернет. Виділити в ньому заголовки другого

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:



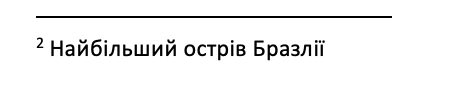
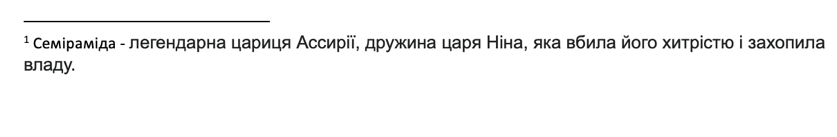
12. В тексті документа створити посилання на рисунки (не менше 5

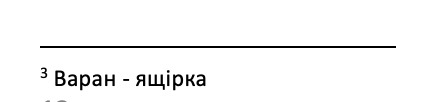
посилань).

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

вибором).

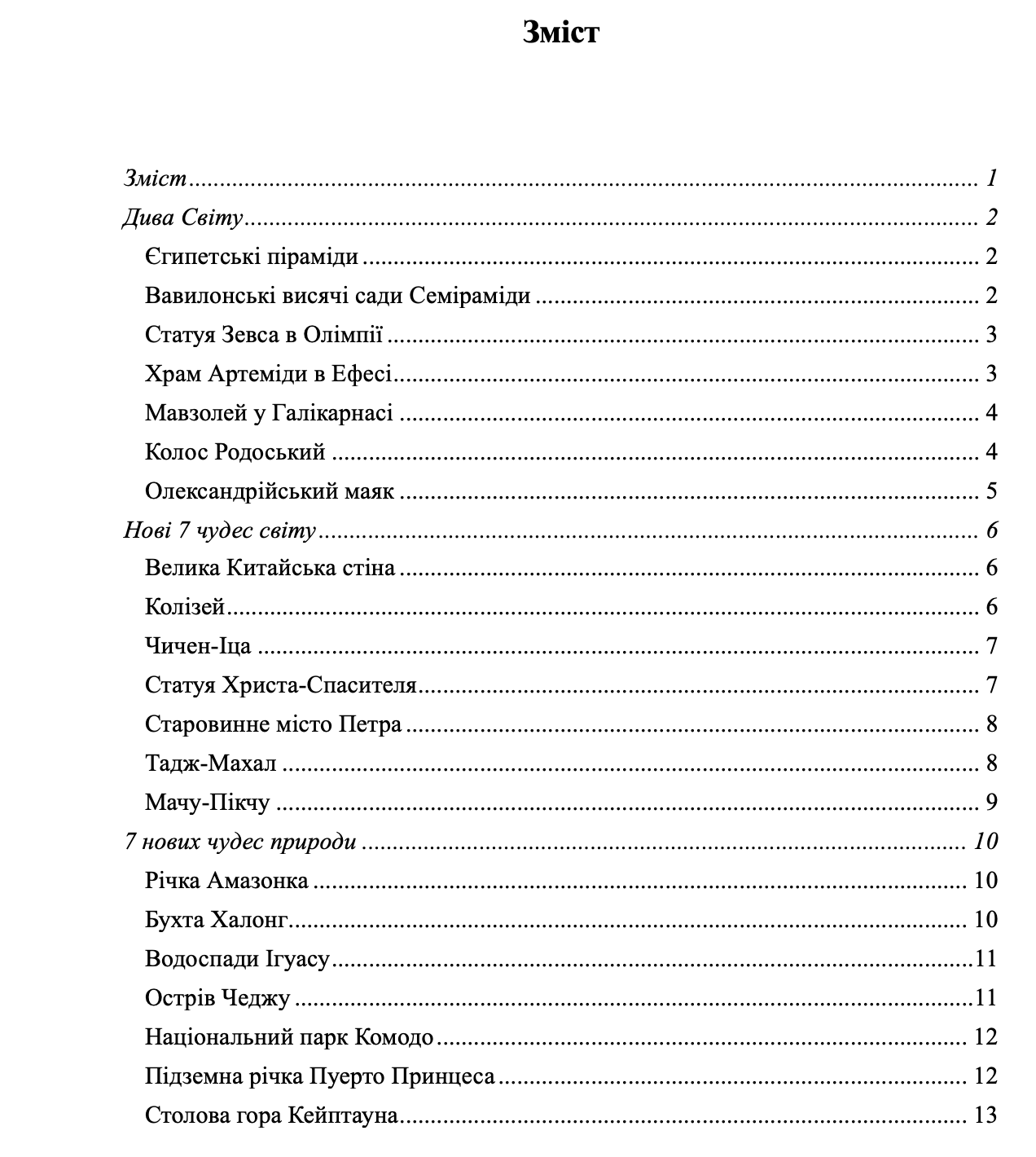




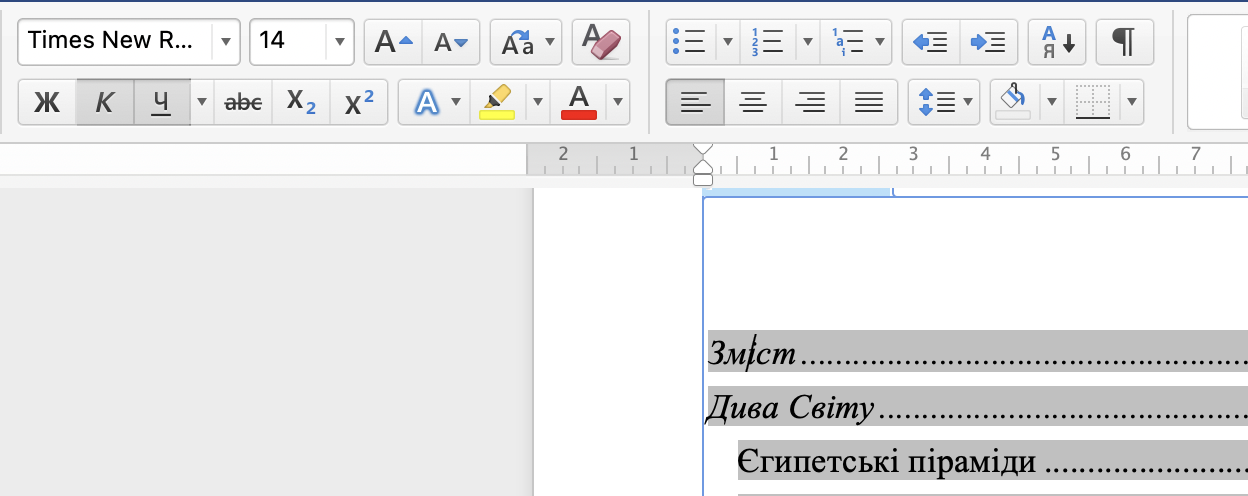
14. Додати верхній колонтитул з назвою розділу тексту.



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:

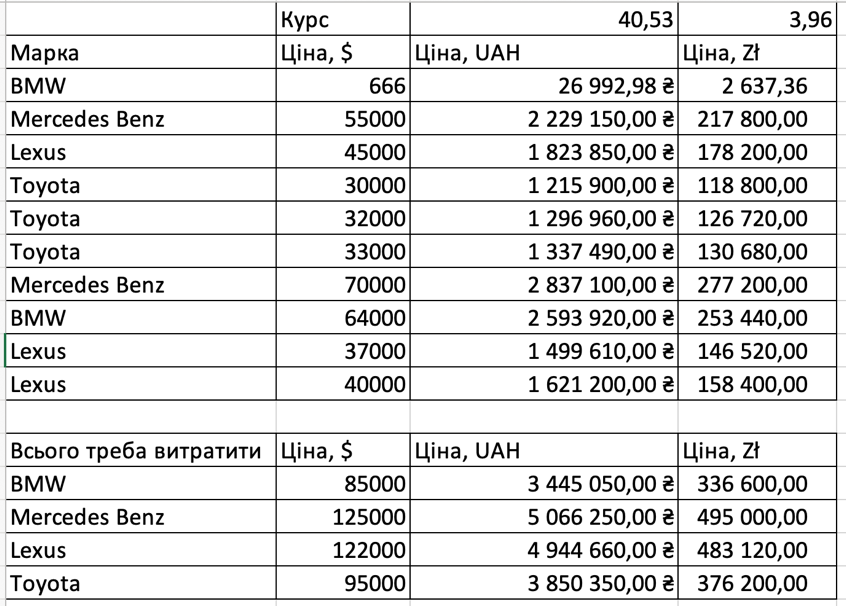
1

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

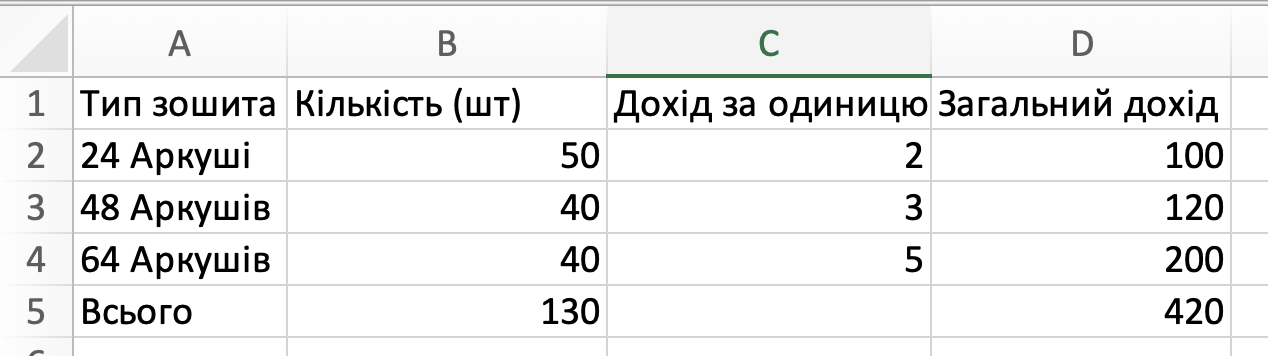
Ціну 10 марок автомобілів, задану в $, перерахувати в грошових одиницях України та Польщі, використовуючи дані з інтернету про ціну автомобілів та курси валют. Визначить, яку суму необхідно сплатити автосалону, якщо він закупає по декілька екземплярів кожної моделі. (Марки автомобілів та їх кількість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



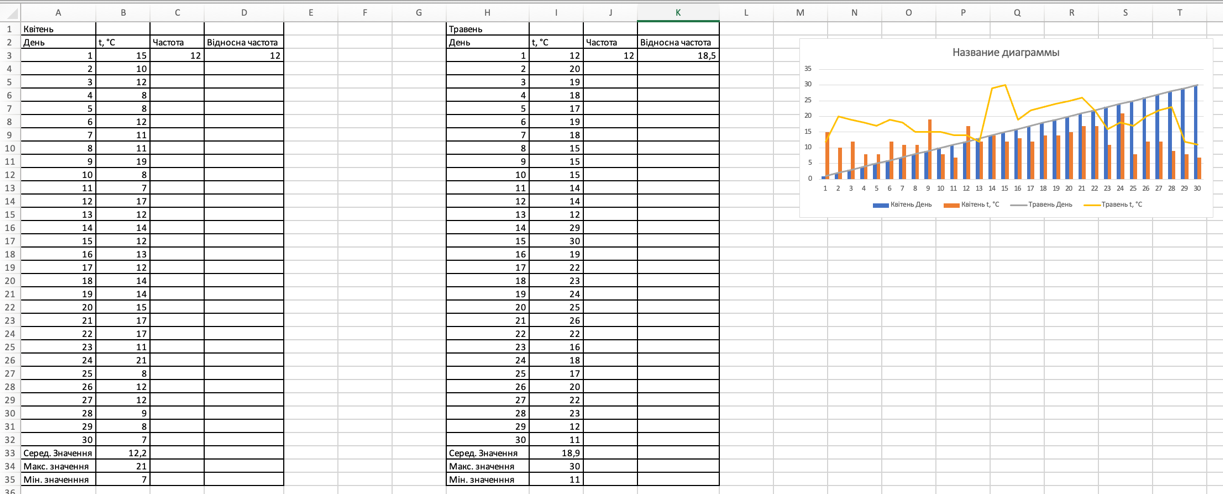
4

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.

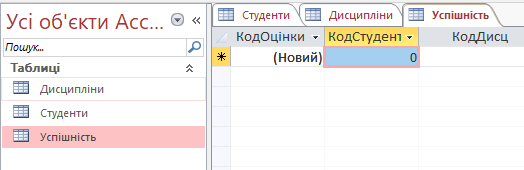


## **2.3 Виконання** **практичних завдань за допомогою засобів табличного редактора MS Access**

За допомогою засобів табличного редактора MS Access виконав наступні завдання:

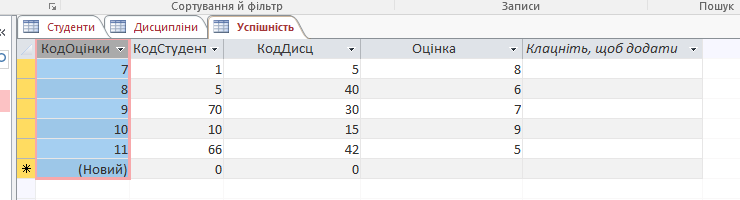
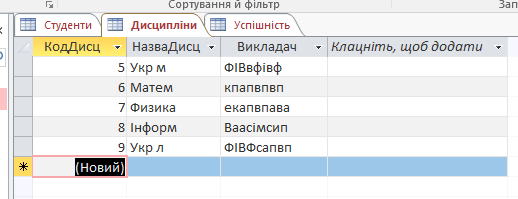
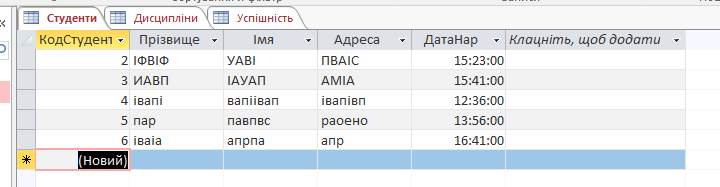
1

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



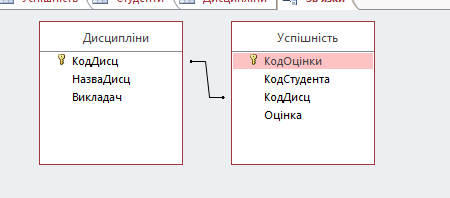
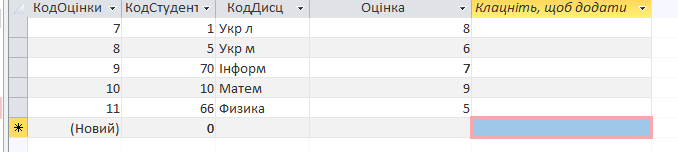
2

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



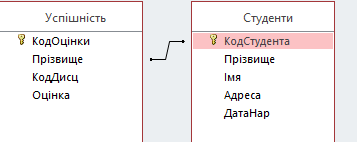
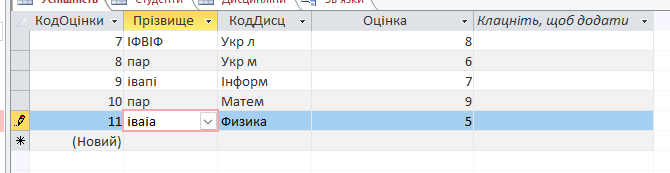
3

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.



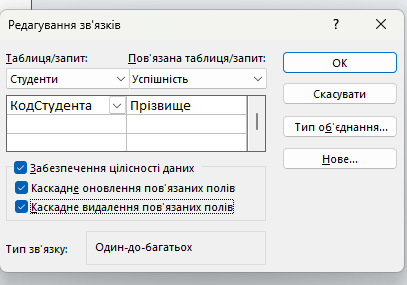
4

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

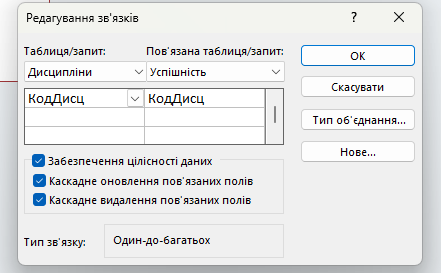


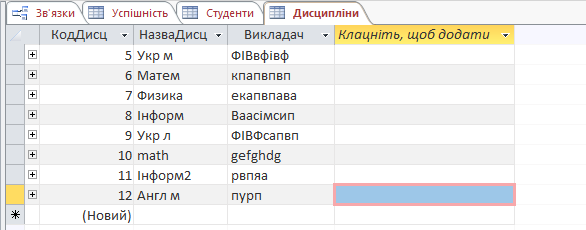
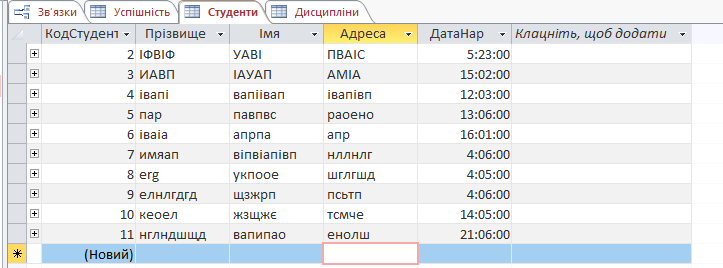
5

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.



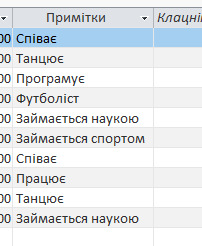
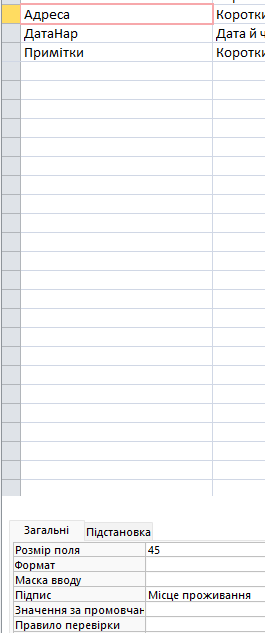
6

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.



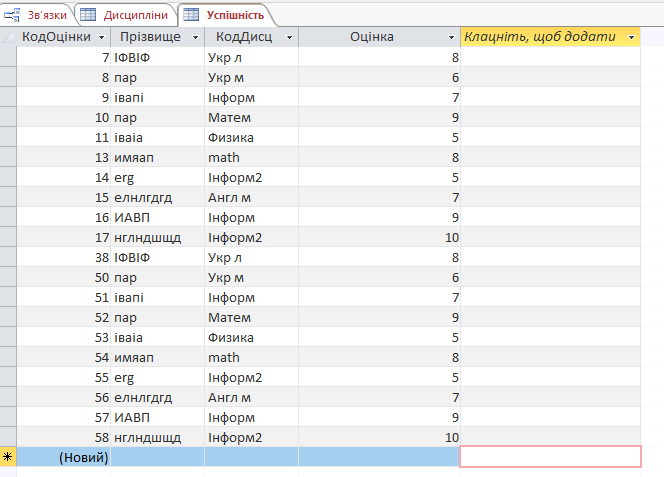
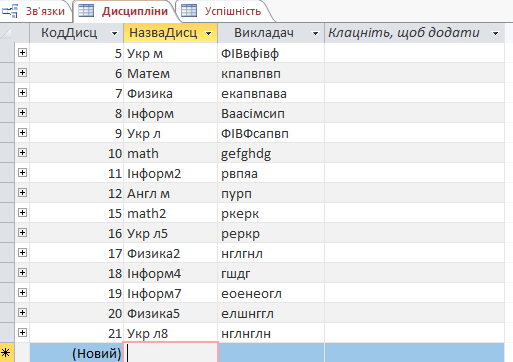
7

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.

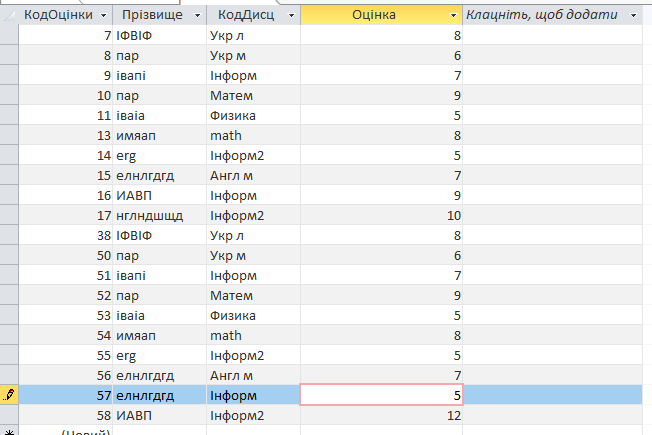


8

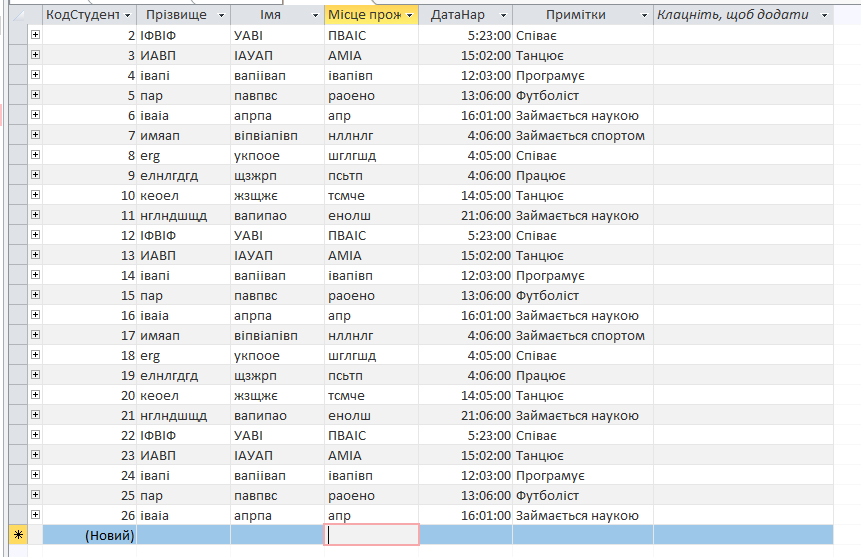
Доповніть таблицю Дисципліни до 15 записів. Таблицю Успішність доповніть до 20 записів шляхом копіювання й вставляння.

 9

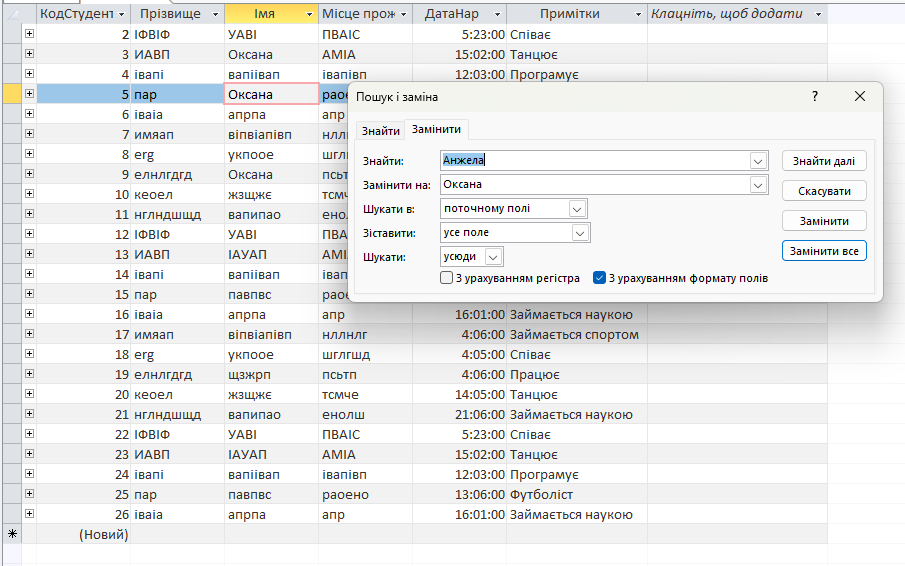
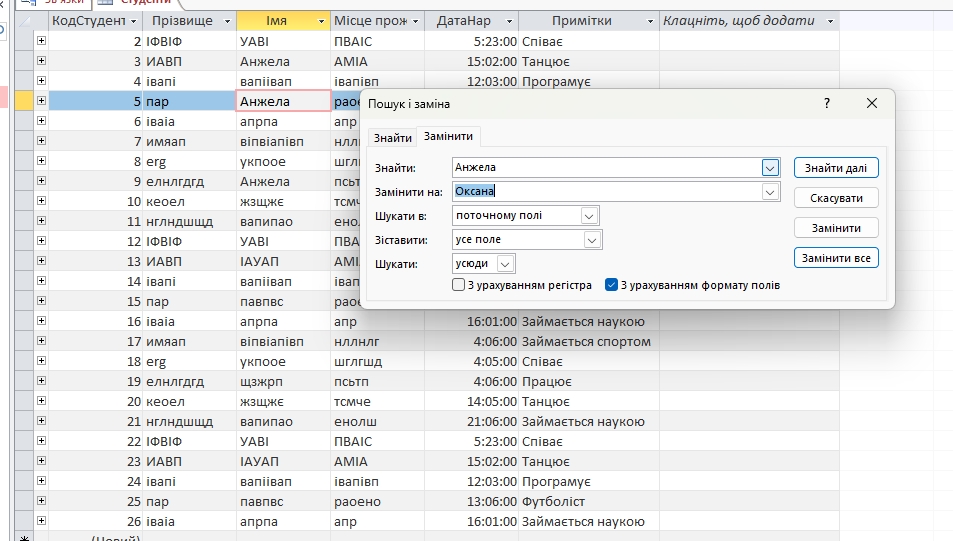
У нових записах вибірково замінить окремі прізвища на нові зі списку, поміняйте оцінки.

10

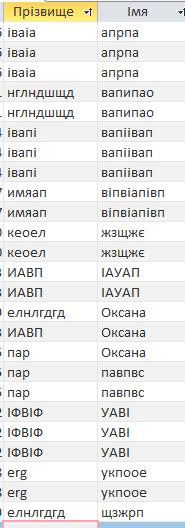
Відкрийте таблицю Студенти, додайте нові записи (до 25) шляхом копіювання та вставляння.

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.

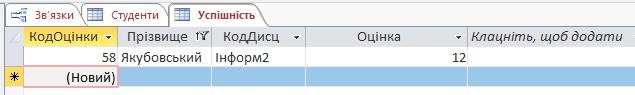
12

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

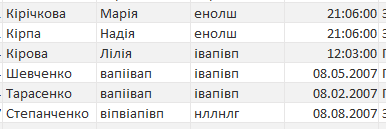


13

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

### Елементи мови:

### Константи:

### Фіксовані значення, які не змінюються під час виконання програми.

### Оголошуються з ключовим словом const.

### Наприклад: const int MAX\_SIZE = 100;.

### 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

1. **Вхідний потік (cin)**:
   * Потік для отримання введених даних з клавіатури.
   * Використовується з оператором >>.
2. **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.*

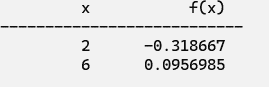
Коефіцієнти *a, b, c –* дійсні числа. Значення *a, b, c, xпоч*, *xкін*, *h* вводити з клавіатури. Передбачити перевірку допустимості введених значень.

### 3.2.2 Постановка задачі

Вхідні дані

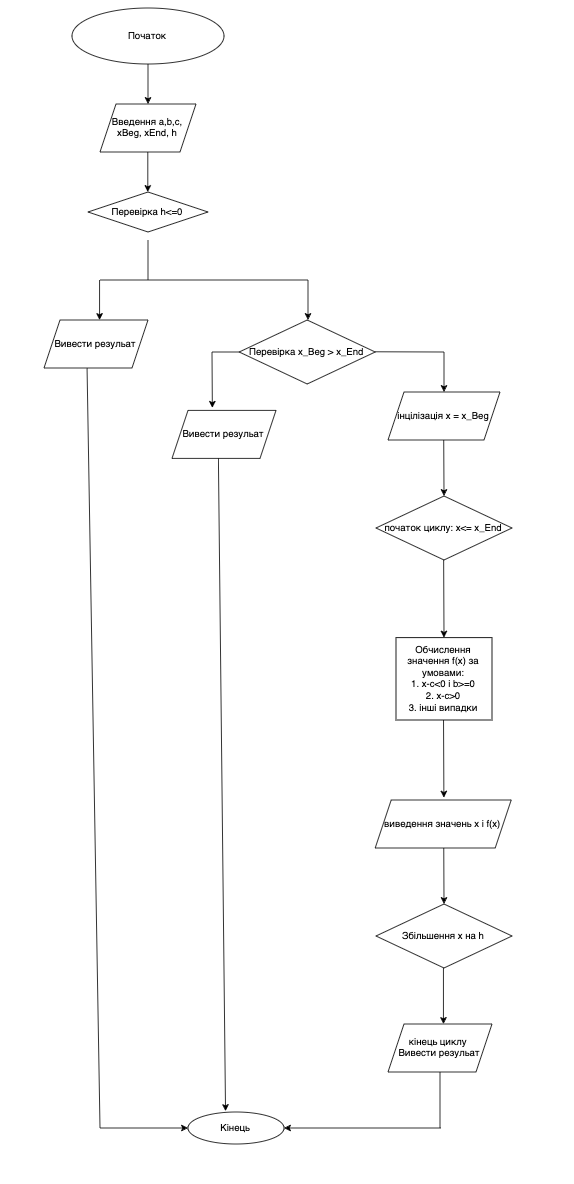
5,4,8,2,8,4

Вихідні дані



Блок-схема алгоритму роботи програми представлена на рис. ххх

Блок-схема



### 3.2.3 Код програми

#include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

// Функція для обчислення значення f(x)

double calculateFunction(double a, double b, double c, double x) {

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

return (a \* x \* x + b \* x) / c;

} else if (x - c > 0) {

return sqrt(x) / (2 \* x - a) + a / (3 \* x + c);

} else {

return (x \* sqrt(x + b \* c)) / (c + b);

}

}

int main() {

double a, b, c, x\_start, x\_end, h;

// Введення коефіцієнтів та меж

cout << "Введіть значення a: ";

cin >> a;

cout << "Введіть значення b: ";

cin >> b;

cout << "Введіть значення c: ";

cin >> c;

cout << "Введіть початкове значення x (x\_поч): ";

cin >> x\_start;

cout << "Введіть кінцеве значення x (x\_кін): ";

cin >> x\_end;

cout << "Введіть крок h: ";

cin >> h;

// Перевірка допустимості введених значень

if (h <= 0) {

cout << "Крок h повинен бути більшим за нуль." << endl;

return 1;

}

if (x\_start > x\_end) {

cout << "Початкове значення x повинно бути меншим або рівним кінцевому значенню." << endl;

return 1;

}

// Виведення заголовку таблиці

cout << setw(10) << "x" << setw(20) << "f(x)" << endl;

cout << "----------------------------------------" << endl;

// Обчислення і виведення значень функції у табличному вигляді

for (double x = x\_start; x <= x\_end; x += h) {

double fx = calculateFunction(a, b, c, x);

cout << setw(10) << x << setw(20) << fx << endl;

}

return 0;

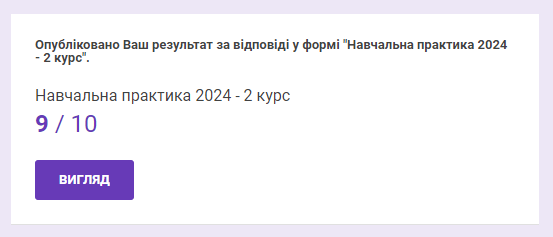
}

### 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**

введення і виведення даних здійснюється за допомогою потоків cin і cout, які належать до бібліотеки iostream.

#include <iostream>

int main() {

std::string name; // оголошуємо змінну для збереження імені

int age; // оголошуємо змінну для збереження віку

// Виводимо повідомлення і чекаємо на введення імені

std::cout << "Введіть ваше ім'я: ";

std::cin >> name; // зчитуємо ім'я з консолі

// Виводимо повідомлення і чекаємо на введення віку

std::cout << "Введіть ваш вік: ";

std::cin >> age; // зчитуємо вік з консолі

// Виводимо отримані дані

std::cout << "Ваше ім'я: " << name << "\n";

std::cout << "Ваш вік: " << age << " років\n";

return 0;

}

1. **Підключення бібліотеки:**

#include <iostream>

Ми підключаємо бібліотеку iostream, яка містить визначення для потоків введення (cin) і виведення (cout).

1. **Функція main:**

int main() {

Це основна функція, з якої починається виконання програми.

1. **Оголошення змінних:**

std::string name;

int age;

Ми оголошуємо дві змінні: name для збереження імені користувача та age для збереження віку.

1. **Виведення запиту на введення імені:**

std::cout << "Введіть ваше ім'я: ";

Використовуючи cout, ми виводимо на екран повідомлення, щоб користувач ввів своє ім'я.

1. **Зчитування імені:**

std::cin >> name;

Використовуючи cin, ми зчитуємо введене користувачем ім'я і зберігаємо його у змінній name.

1. **Виведення запиту на введення віку:**

std::cout << "Введіть ваш вік: ";

Ми виводимо на екран повідомлення, щоб користувач ввів свій вік.

1. **Зчитування віку:**

std::cin >> age;

Використовуючи cin, ми зчитуємо введений користувачем вік і зберігаємо його у змінній age.

1. **Виведення отриманих даних:**

std::cout << "Ваше ім'я: " << name << "\n";

std::cout << "Ваш вік: " << age << " років\n";

Ми виводимо на екран ім'я та вік користувача.

1. **Завершення програми:**

return 0;

Повертаємо 0, щоб вказати, що програма завершилась успішно.

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

Управління станом програми в C++ може бути здійснене за допомогою різних структур даних. Ось приклади структур даних, які часто використовуються для управління станом програми: масиви, структури (struct), класи (class), вектори (std::vector), списки (std::list), мапи (std::map) та множини (std::set).

### ****Масиви****(Arrays) дозволяють зберігати набір елементів одного типу в безперервній області пам'яті. Вони використовуються, коли потрібно зберігати фіксовану кількість елементів. Масиви мають перевагу швидкого доступу до елементів за індексом, але їх розмір не можна змінювати під час виконання програми.

**Структури (struct)**дозволяють згрупувати різні типи даних під одним іменем. Це зручно для зберігання пов'язаних даних, таких як інформація про особу (ім'я, вік). Структури підтримують агрегацію різних типів даних і надають простий спосіб організувати дані.

**Класи(class)** схожі на структури, але вони також підтримують інкапсуляцію і можуть містити методи для роботи з даними. Класи дозволяють створювати об'єкти з приватними і публічними членами, забезпечуючи кращу організацію та захист даних. Вони можуть мати конструктори, деструктори і різні методи для взаємодії з внутрішнім станом об'єкта.

### ****Вектори**** (std::vector) - це динамічні масиви, які можуть змінювати свій розмір під час виконання програми. Вектори забезпечують гнучкість у роботі з наборами даних змінного розміру і мають зручні методи для додавання, видалення і доступу до елементів. Вони є частиною стандартної бібліотеки C++ і забезпечують автоматичне управління пам'яттю.

### ****Списки**** (std::list) - це двозв'язні списки, які дозволяють ефективно вставляти і видаляти елементи в будь-якій точці. Вони особливо корисні, коли потрібні часті операції вставки і видалення, але доступ до елементів за індексом є повільнішим порівняно з масивами і векторами.

**Мапи** **(std::map)** - це асоціативні контейнери, які зберігають пари "ключ-значення". Вони забезпечують швидкий пошук, додавання і видалення елементів за ключем. Мапи корисні для зберігання даних, які потребують доступу за унікальним ключем.

**Множини (std::set)**- це контейнери, які зберігають унікальні елементи у впорядкованому вигляді. Вони забезпечують швидкий пошук і перевірку наявності елемента. Множини корисні, коли важливо зберігати тільки унікальні значення і підтримувати впорядкованість.

Кожна з цих структур даних має свої переваги і недоліки, і вибір конкретної структури залежить від вимог до управління станом програми і характеристик операцій, які потрібно виконувати над даними.

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

### Підключення бібліотеки

Для роботи з файлами в C++ використовується бібліотека <fstream>, яка містить класи для введення та виведення файлів. Основні класи включають:

* std::ifstream для читання з файлів.
* std::ofstream для запису у файли.

### Запис у файл

1. **Створення об'єкта для запису:** Для запису у файл створюється об'єкт класу ofstream, до якого передається ім'я файлу. Якщо файл не існує, він буде створений.
2. **Перевірка відкриття файлу:** Після створення об'єкта перевіряється, чи файл відкрився успішно за допомогою методу is\_open(). Це важливо, оскільки файл може не відкритися з різних причин (наприклад, відсутність прав доступу).
3. **Запис даних:** Якщо файл успішно відкритий, можна записувати дані, використовуючи оператор виведення (<<). Це можуть бути рядки тексту, числа або інші дані.
4. **Закриття файлу:** Після завершення запису файл закривається за допомогою методу close(). Це звільняє ресурси і гарантує, що всі дані записані правильно.
5. **Обробка помилок:** Якщо файл не вдалося відкрити, необхідно обробити цю ситуацію, наприклад, вивести повідомлення про помилку.

### Читання з файлу

1. **Створення об'єкта для читання:** Для читання з файлу створюється об'єкт класу ifstream, до якого передається ім'я файлу, який потрібно прочитати.
2. **Перевірка відкриття файлу:** Перевіряється, чи файл відкрився успішно за допомогою методу is\_open().
3. **Читання даних:** Якщо файл успішно відкритий, дані зчитуються з файлу рядок за рядком або іншими способами (наприклад, символ за символом). Для зчитування рядків часто використовується функція getline().
4. **Виведення прочитаних даних:** Прочитані дані можна виводити на екран або обробляти іншим чином.
5. **Закриття файлу:** Після завершення читання файл закривається за допомогою методу close().
6. **Обробка помилок:** Якщо файл не вдалося відкрити, необхідно обробити цю ситуацію, наприклад, вивести повідомлення про помилку.

**Ось приклад виконання коду на С++**

#include <iostream>

#include <fstream> // Підключаємо бібліотеку для роботи з файлами

int main() {

// Запис у файл

std::ofstream outFile("example.txt"); // Відкриваємо файл для запису

if (outFile.is\_open()) { // Перевіряємо, чи файл відкритий

outFile << "Hello, World!\n"; // Записуємо дані у файл

outFile << "This is a file I/O example.\n";

outFile.close(); // Закриваємо файл

} else {

std::cerr << "Unable to open file for writing\n";

}

// Читання з файлу

std::ifstream inFile("example.txt"); // Відкриваємо файл для читання

std::string line;

if (inFile.is\_open()) { // Перевіряємо, чи файл відкритий

while (getline(inFile, line)) { // Читаємо рядки з файлу

std::cout << line << "\n"; // Виводимо прочитані рядки на екран

}

inFile.close(); // Закриваємо файл

} else {

std::cerr << "Unable to open file for reading\n";

}

return 0;

}

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

Формати файлів в C++ визначають спосіб, у який дані зберігаються у файлі. Основні формати включають текстові файли, бінарні файли та файли у форматі CSV (Comma-Separated Values). Кожен з цих форматів має свої особливості обробки у C++.

### Текстові файли

1. **Особливості:** У текстових файлах дані зберігаються у вигляді звичайного тексту, де символи кодують різні значення.
2. **Обробка:** Для обробки текстових файлів використовуються потоки ifstream для читання та ofstream для запису. Дані зчитуються рядок за рядком або посимвольно за допомогою методів, таких як getline() або get(). Запис даних в файл також здійснюється через потоки, використовуючи оператор виведення (<<).

#include <iostream>

#include <fstream>

#include <string>

int main() {

// Запис у текстовий файл

std::ofstream outFile("example.txt");

if (outFile.is\_open()) {

outFile << "Hello, World!\n";

outFile << "This is a text file.\n";

outFile.close();

} else {

std::cerr << "Unable to open file for writing\n";

}

// Читання з текстового файлу

std::ifstream inFile("example.txt");

std::string line;

if (inFile.is\_open()) {

while (getline(inFile, line)) {

std::cout << line << "\n";

}

inFile.close();

} else {

std::cerr << "Unable to open file for reading\n";

}

return 0;

}

### Бінарні файли

1. **Особливості:** Бінарні файли зберігають дані у вигляді бінарного коду, що дозволяє ефективніше зберігати складні структури даних.
2. **Обробка:** Для роботи з бінарними файлами використовуються такі ж потоки, але зазвичай використовуються функції read() та write() для читання і запису бінарних даних. Крім того, при роботі з бінарними файлами необхідно пам'ятати про правильну серіалізацію та десеріалізацію даних.

#include <iostream>

#include <fstream>

#include "json.hpp"

using json = nlohmann::json;

int main() {

// Читання з JSON файлу

std::ifstream jsonFile("data.json");

if (jsonFile.is\_open()) {

json jsonData;

jsonFile >> jsonData;

jsonFile.close();

// Обробка JSON даних

std::string name = jsonData["name"];

int age = jsonData["age"];

std::cout << "Name: " << name << "\n";

std::cout << "Age: " << age << "\n";

} else {

std::cerr << "Unable to open JSON file\n";

}

return 0;

}

### CSV файли (Comma-Separated Values)

1. **Особливості:** Файли у форматі CSV представляють дані у вигляді таблиці, де значення розділені комами або іншими символами розділювачами.
2. **Обробка:** Для обробки CSV файлів використовуються функції для роботи з рядками та розділювачами. Найчастіше для цього використовуються бібліотеки для обробки CSV даних або власний код для розбиття рядків на частини та обробки окремих значень.

#include <iostream>

#include <fstream>

#include <sstream>

#include <vector>

#include <string>

int main() {

// Читання з CSV файлу

std::ifstream csvFile("data.csv");

if (csvFile.is\_open()) {

std::string line;

while (getline(csvFile, line)) {

std::stringstream ss(line);

std::vector<std::string> row;

std::string cell;

while (getline(ss, cell, ',')) { // Розділення рядка на коми

row.push\_back(cell);

}

// Обробка рядка CSV даних

for (const auto& data : row) {

std::cout << data << "\t";

}

std::cout << "\n";

}

csvFile.close();

} else {

std::cerr << "Unable to open CSV file\n";

}

return 0;

}

### Пояснення

* **Текстові файли:** Зручні для зчитування та редагування людиною, але можуть займати більше місця на диску.
* **Бінарні файли:** Менш займають місця на диску і швидше обробляються комп'ютером, але менш зрозумілі для людини.
* **CSV файли:** Використовуються для обміну даними між програмами, зручні для обробки таблиць даних, але можуть потребувати додаткової обробки для коректного розбору даних.

При роботі з будь-якими форматами файлів важливо враховувати правильність обробки даних, обробку помилок та забезпечення безпеки даних.

Приклади файлів (CSV, JSON, XML), порівняння їх між собою, які в кого переваги і де краще їх використовувати

### CSV (Comma-Separated Values)

**Приклад:**

#include <iostream>

#include <fstream>

#include <sstream>

#include <vector>

int main() {

// Запис у CSV файл

std::ofstream outFile("data.csv");

if (outFile.is\_open()) {

// Запис заголовків

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

// Запис рядків даних

outFile << "Alice,30,Engineer\n";

outFile << "Bob,25,Designer\n";

outFile << "Charlie,35,Manager\n";

outFile.close();

} else {

std::cerr << "Unable to open file for writing\n";

}

// Читання з CSV файлу

std::ifstream inFile("data.csv");

if (inFile.is\_open()) {

std::string line;

// Читання заголовків

if (getline(inFile, line)) {

std::cout << "Header: " << line << "\n";

}

// Читання даних

while (getline(inFile, line)) {

std::stringstream ss(line);

std::string name, age, occupation;

getline(ss, name, ',');

getline(ss, age, ',');

getline(ss, occupation, ',');

std::cout << "Name: " << name << ", Age: " << age << ", Occupation: " << occupation << "\n";

}

inFile.close();

} else {

std::cerr << "Unable to open file for reading\n";

}

return 0;

}

**Переваги:**

* **Простота:** CSV файли прості для розуміння та створення.
* **Швидкість:** Зчитування та записування CSV файлів відбувається швидко.
* **Сумісність:** Підтримуються багатьма програмами, включаючи таблиці, такі як Excel.

**Недоліки:**

* **Обмежена структура:** Підтримує лише плоску структуру даних.
* **Відсутність типізації:** Всі дані зберігаються у вигляді рядків, без вбудованої підтримки типів даних.

**Коли використовувати:**

* **Обмін даними між програмами:** Коли потрібно обмінюватися табличними даними між різними програмами.
* **Прості набори даних:** Коли дані мають просту, табличну структуру.

### JSON (JavaScript Object Notation)

**Приклад:**

{

"employees": [

{

"name": "Alice",

"age": 30,

"occupation": "Engineer"

},

{

"name": "Bob",

"age": 25,

"occupation": "Designer"

},

{

"name": "Charlie",

"age": 35,

"occupation": "Manager"

}

]

}

**Переваги:**

* **Читабельність:** JSON файли легко читаються людиною.
* **Структура:** Підтримують вкладені структури (масиви, об'єкти).
* **Сумісність:** Підтримуються більшістю мов програмування, особливо JavaScript.

**Недоліки:**

* **Розмір:** JSON файли можуть бути більшими за розміром порівняно з CSV.
* **Продуктивність:** Зчитування і запис JSON файлів може бути повільнішим через їх структурованість.

**Коли використовувати:**

* **Веб-розробка:** Коли потрібно передавати дані між сервером і веб-додатком.
* **API:** Для обміну даними між сервісами через API.

### XML (eXtensible Markup Language)

**Приклад:**

<employees>

<employee>

<name>Alice</name>

<age>30</age>

<occupation>Engineer</occupation>

</employee>

<employee>

<name>Bob</name>

<age>25</age>

<occupation>Designer</occupation>

</employee>

<employee>

<name>Charlie</name>

<age>35</age>

<occupation>Manager</occupation>

</employee>

</employees>

**Переваги:**

* **Гнучкість:** XML може зберігати дані з дуже складною структурою.
* **Валідація:** Підтримує валідацію структури через DTD або XML Schema.
* **Універсальність:** Підтримується багатьма системами та стандартами.

**Недоліки:**

* **Розмір:** XML файли зазвичай більші порівняно з CSV та JSON.
* **Складність:** Більш складні для читання і написання людиною.

**Коли використовувати:**

* **Складні дані:** Коли потрібно зберігати або передавати складно структуровані дані.
* **Інтеграція з іншими системами:** Для інтеграції з системами, які використовують XML як стандарт обміну даними.

### Порівняння

1. **Простота та продуктивність:**
   * **CSV**: Найпростіший і швидкий для читання та запису, але обмежений у структурі.
   * **JSON**: Більш структурований і зручний для вкладених даних, але може бути повільнішим і більшим за розміром.
   * **XML**: Найбільш гнучкий і структурований, але найскладніший і повільніший у порівнянні з іншими форматами.
2. **Читабельність:**
   * **CSV**: Легко читається, але тільки для простих табличних даних.
   * **JSON**: Легко читається, зручно для вкладених структур.
   * **XML**: Більш складний для читання, але добре підходить для складних структур даних.
3. **Використання:**
   * **CSV**: Коли потрібно зберігати або обмінюватися простими табличними даними.
   * **JSON**: Для веб-розробки та API, коли потрібна підтримка вкладених структур.
   * **XML**: Для складних структур даних та інтеграції з системами, що використовують XML.

Кожен з цих форматів має свої переваги і недоліки, і вибір конкретного формату залежить від специфічних потреб вашої програми або проекту.

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

Системи контролю версій (VCS) - це програмне забезпечення, яке дозволяє зберігати та відстежувати зміни в файловій системі, особливо корисне при роботі над проектами програмного забезпечення. У C++ існують різні VCS, які можна використовувати для керування версіями програм та збереження їх історії. Ось кілька популярних систем контролю версій на C++:

1. **Git**: Git є однією з найпоширеніших та потужних систем контролю версій. Він дозволяє зберігати версії програмного коду, відстежувати зміни, створювати гілки для різних функціональних розробок і об'єднувати їх у головний код. Git також підтримує роботу у команді, де різні розробники можуть працювати одночасно над проектом.
2. **Subversion (SVN)**: SVN - це інша популярна система контролю версій, яка пропонує подібні можливості до Git. Вона зберігає історію змін, дозволяє працювати з гілками і об'єднувати зміни. SVN використовує централізовану модель зберігання даних, що відрізняється від розподіленої моделі Git.
3. **Mercurial (Hg)**: Mercurial - ще один варіант VCS, схожий на Git. Він дозволяє зберігати версії програми, керувати гілками і виконувати об'єднання змін. Mercurial також має свою власну командну модель роботи.
4. **Perforce (P4)**: Perforce - це VCS, яка зазвичай використовується у великих комерційних проектах. Вона підтримує об'єднання змін, роботу у команді і керування дозволами доступу до коду.

Ці системи контролю версій дозволяють зберігати і відстежувати зміни в коді, забезпечуючи можливість спільної роботи над проектами, відновлення попередніх версій коду та вирішення конфліктів при об'єднанні змін в різних гілках розробки. Кожна з цих систем має свої особливості і переваги, тому вибір системи контролю версій залежить від конкретних потреб та вимог проекту.

1. **git init**: Ця команда ініціалізує новий репозиторій Git в поточному каталозі.
2. **git clone**: Клонує віддалений репозиторій на ваш комп'ютер.
3. **git add**: Додає змінені або нові файли до індексу для подальшого коміту.
4. **git commit**: Комітує зміни, що були додані до індексу, і створює новий коміт у локальному репозиторії.
5. **git push**: Відправляє локальні коміти на серверний репозиторій.
6. **git pull**: Оновлює локальний репозиторій до останньої версії з віддаленого репозиторію.
7. **git branch**: Показує список гілок у репозиторії або створює нову гілку.
8. **git checkout**: Перемикається між гілками або відновлює файли до попереднього стану.
9. **git merge**: Об'єднує зміни з іншої гілки у поточну гілку.
10. **git log**: Показує історію комітів у репозиторії.

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

**Markdown** - це простий та легко зрозумілий формат для написання тексту з форматуванням, який часто використовується для опису проектів у веб-розробці та інших галузях. Він дозволяє швидко створювати документацію, README файли для GitHub та інших сервісів, блоги тощо. Основні елементи мови Markdown для опису проекту:

1. **Заголовки:**

# Назва проекту

## Підзаголовок

1. **Текст:**

Це звичайний текст.

\*Курсив\* або \_курсив\_

\*\*Напівжирний\*\* або \_\_напівжирний\_\_

\*\*\*Напівжирний курсив\*\*\* або \_\_\_напівжирний курсив\_\_\_

`Кодовий шрифт`

1. **Списки:**

### Неупорядкований список:

- Елемент 1

- Елемент 2

- Вкладений елемент

### Упорядкований список:

1. Перший елемент

2. Другий елемент

1. **Посилання та зображення:**

[Текст посилання](https://www.example.com)

![Текст замінки для зображення](шлях\_до\_зображення.png)

1. **Код:**

```javascript

console.log('Привіт, світ!');

Копировать код

1. **Таблиці:**

| Колонка 1 | Колонка 2 |

|-----------|-----------|

| Рядок 1,1 | Рядок 1,2 |

| Рядок 2,1 | Рядок 2,2 |

1. **Цитати:**

> Це цитата.

1. **Горизонтальні лінії:**

---

\*\*\*

1. **Виділення коду:**

`Кодовий фрагмент`

## **4.7 Розроблення ігрової програми**

### 4.7.1 Код програми

#include <iostream>

**using** **namespace** std;

**char** board[3][3] = {{'1', '2', '3'}, {'4', '5', '6'}, {'7', '8', '9'}};

**char** current\_marker;

**int** current\_player;

**void** drawBoard() {

cout << " " << board[0][0] << " | " << board[0][1] << " | " << board[0][2] << endl;

cout << "---|---|---" << endl;

cout << " " << board[1][0] << " | " << board[1][1] << " | " << board[1][2] << endl;

cout << "---|---|---" << endl;

cout << " " << board[2][0] << " | " << board[2][1] << " | " << board[2][2] << endl;

}

**bool** placeMarker(**int** slot) {

**int** row = slot / 3;

**int** col = slot % 3;

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

board[row][col] = current\_marker;

**return** **true**;

} **else** {

**return** **false**;

}

}

**int** winner() {

// Check rows

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

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

**return** current\_player;

}

}

// Check columns

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

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

**return** current\_player;

}

}

// Check diagonals

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

**return** current\_player;

}

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

**return** current\_player;

}

**return** 0;

}

**void** swapPlayerAndMarker() {

**if** (current\_marker == 'X') {

current\_marker = 'O';

} **else** {

current\_marker = 'X';

}

**if** (current\_player == 1) {

current\_player = 2;

} **else** {

current\_player = 1;

}

}

**void** game() {

current\_player = 1;

current\_marker = 'X';

drawBoard();

**int** player\_won;

**for** (**int** i = 0; i < 9; i++) {

cout << "It's player " << current\_player << "'s turn. Enter your slot: ";

**int** slot;

cin >> slot;

**if** (slot < 1 || slot > 9) {

cout << "Invalid slot! Try again." << endl;

i--;

**continue**;

}

**if** (!placeMarker(slot - 1)) {

cout << "Slot occupied! Try again." << endl;

i--;

**continue**;

}

drawBoard();

player\_won = winner();

**if** (player\_won == 1) {

cout << "Player Х wins!" << endl;

**break**;

}

**if** (player\_won == 2) {

cout << "Player О wins!" << endl;

**break**;

}

swapPlayerAndMarker();

}

**if** (player\_won == 0) {

cout << "It's a tie!" << endl;

}

}

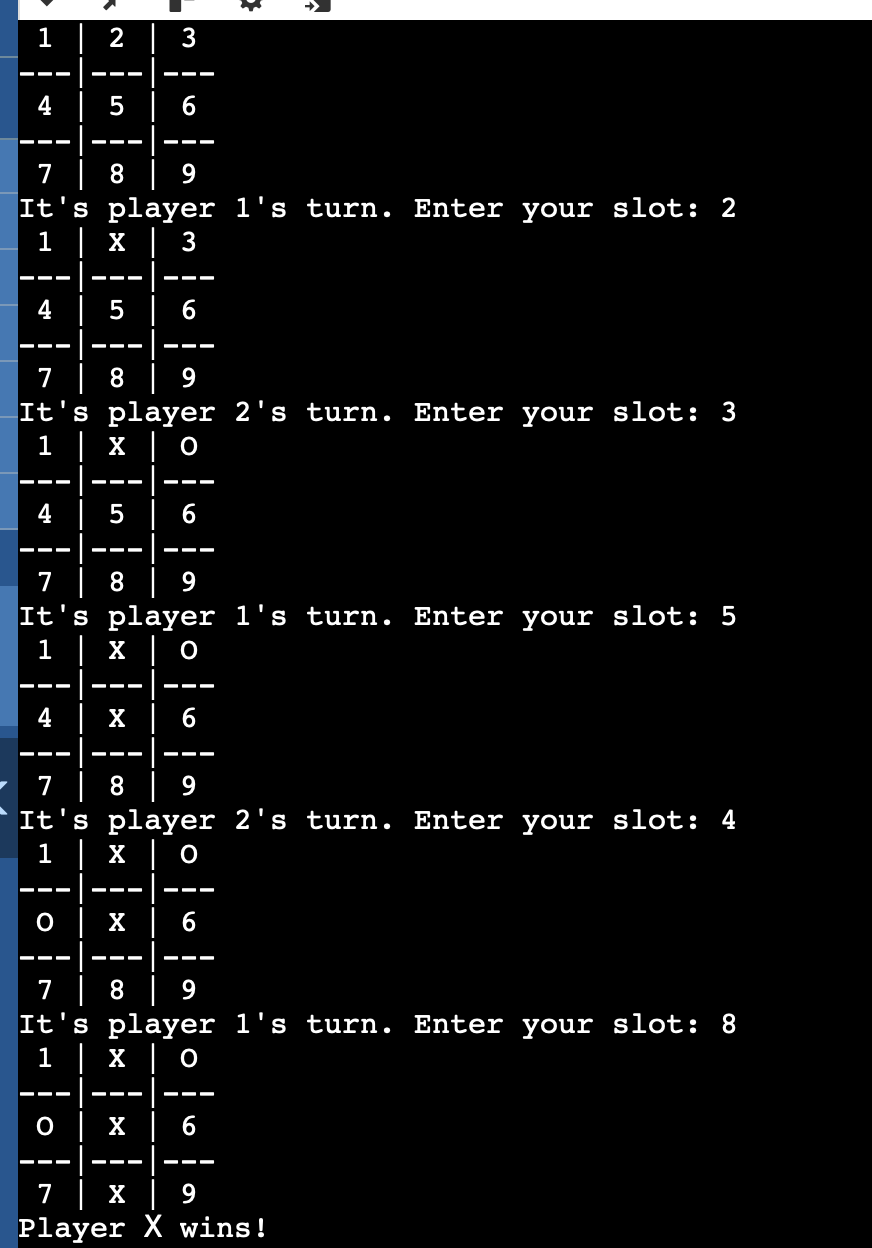
**int** main() {

game();

**return** 0;

}

### 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

Код HTML

      <!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Оганнесян Гарік - CV</title>

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

<script src="https://kit.fontawesome.com/5366f83049.js" crossorigin="anonymous"></script>

</head>

<style>

\* {

margin: 0;

padding: 0;

}

</style>

</head>

<body>

<div class="container">

<div class="header">

<img src="IMG\_6799.JPG" alt="Profile Image" class="profile-img">

<h1>Оганнесян Гарік</h1>

<h2>Website Developer </h2>

</div>

<div class="summary">

<h3>Summary</h3>

<hr style="color: rgb(0, 0, 0)">

<p>At the moment I have some experience in website development, I am learning C++, HTML,CSS and Java script. I have strong creative thinking, attention to detail, small details that many will not notice, I also have good communication skills and understanding of people. I’m ready to move forward and achieve my goals no matter how hard it is. If there is a task, I will solve it</p>

</div>

<br/>

<div class="education">

<h3>Education</h3>

<hr style="color: rgb(0, 0, 0)">

<p>2013-2022. Secondary general education school Liceum №1</p>

<p>2022-2026. Optical and Mechanical College of Taras Shevchenko Kyiv National University</p>

</div>

<div class="experience">

<h3>Experience</h3>

<hr style="color: rgb(0, 0, 0)">

<div class="job">

<h4>Store Website Developer (March 2024 - Now):</h4>

<p>Developing, updating and supporting website</p>

</div>

</div>

<div class="contact">

<a href="mailto:garikogannesan299@gmail.com" class="btn email"> <i class="fa-solid fa-envelope"> </i> Email</a>

<br/>

<a href="https://www.instagram.com/garik\_0707\_/" class="btn instagram" target="\_blank"><i class="fab fa-instagram"></i> Instagram</a>

</div>

</body>

</html>

**CSS**

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background-color: #F2F3FB;

}

.container {

width: 80%;

margin: 0 auto;

padding: 20px;

background-color: #F2F3FB;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

.header {

text-align: center;

margin-bottom: 20px;

}

.profile-img {

width: 150px;

height: 150px;

border-radius: 50%;

}

h1, h2 {

margin: 10px 0;

}

h3 {

border-bottom: 2px solid #F2F3FB ;

padding-bottom: 5px;

margin-bottom: 10px;

}

p {

line-height: 1.6;

margin-bottom: 10px;

}

.experience .job {

margin-bottom: 15px;

}

.contact {

text-align: center;

margin-top: 20px;

}

.btn {

display: inline-block;

margin: 5px;

padding: 20px 450px;

border: 1px solid black;

border-radius: 5px;

color: black;

background-color: none;

text-decoration: none;

position: relative;

padding-left: 40px;

}

.icon1{

position: relative;

cursor: pointer;

}

.icon1 {

content: "\f0e0";

font-weight: 900;

position: absolute;

left: 10px;

top: 50%;

transform: translateY(-50%);

}

.btn.email::before {

content: "";

width: 200px;

}

.btn.instagram::before {

content: "";

width: 200px;

}

.btn:hover {

background-color: black;

color: white;

}

    color: white;

}

# 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