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: |
| студентка 2-го курсу групи ІПЗ-22 |
| Демченко Катерина Сергіївна  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) | |  | |
| *Демченко Катерина Сергіївна* | | | | |
| (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.

Демченко К.С.

Перевір.

Реценз.

В.Я

Н. Контр.

В.Я

Затверд.

Звіт з навчальної практики

Літ.

Аркушів

85

ІПЗ-22

ОМФК.0122 НП 01

**ЗМІСТ**

[ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 5](#_Toc169347633)

[ВСТУП 7](#_Toc169347634)

[1 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ 9](#_Toc169347635)

[2 РОЗДІЛ "ІНФОРМАТИКА" 10](#_Toc169347636)

[**2.1** **Розроблення засобами MS Word документа складної структури** 10](#_Toc169347637)

[**2.2 Розв‘язок задач за допомогою засобів табличного редактора MS Excel** 18](#_Toc169347638)

[**2.3 Виконання** **практичних завдань за допомогою засобів табличного редактора MS Access** 21](#_Toc169347639)

[3 РОЗДІЛ "ПРОГРАМУВАННЯ С++" 27](#_Toc169347640)

[**3.1 Теоретичні відомості** 27](#_Toc169347641)

[3.1.1 Загальні поняття. Елементи мови С++ - константи, змінні, операції, перетворення типів 27](#_Toc169347642)

[3.1.2 Арифметичні операції 30](#_Toc169347643)

[3.1.3 Поняття вхідного та вихідного потоку, найпростіші математичні функції 35](#_Toc169347644)

[3.1.4 Програмування лінійних та розгалужених обчислювальних процесів 37](#_Toc169347645)

[3.1.5 Програмування циклічних обчислювальних процесів 38](#_Toc169347646)

[**3.2 Практичне завдання** 39](#_Toc169347647)

[3.2.1 Завдання 39](#_Toc169347648)

[3.2.2 Постановка задачі 39](#_Toc169347649)

[3.2.3 Код програми 41](#_Toc169347650)

[3.2.4 Скріншот виконання програми 44](#_Toc169347651)

[**3.3 Виконання тестів** 45](#_Toc169347652)

[4 РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++" 46](#_Toc169347653)

[**4.1 Тема 1. Введення та виведення у базових консольних програмах** 46](#_Toc169347654)

[**4.2 Тема 2. Структури даних для управління станом програми** 50](#_Toc169347655)

[**4.3Тема 3. Робота з файлами: запис та читання** 54](#_Toc169347656)

[**4.4 Тема 4. Формати файлів та їх обробка** 58](#_Toc169347657)

[**4.5 Системи контролю версій (VCS)** 62](#_Toc169347658)

[**4.6 Тема 6. Мова Markdown для опису проекті** 65](#_Toc169347659)

[**4.7 Розроблення ігрової програми** 66](#_Toc169347660)

[4.7.1 Код програми 66](#_Toc169347661)

[4.7.2 Скріншоти виконання програми 70](#_Toc169347662)

[5 РОЗДІЛ ”ВЕБ-ДИЗАЙН ТА HTML” 71](#_Toc169347663)

[**5.1 Мова розмітки HTML та структура Веб-сторінки** 71](#_Toc169347664)

[**5.2 Розмітка тексту за допомогою HTML. Зображення та посилання** 73](#_Toc169347665)

[**5.3 Каскадні таблиці стилів CSS та їх структура** 74](#_Toc169347666)

[**5.4 Спадковості і каскадування за допомогою CSS** 75](#_Toc169347667)

[**5.5 Практична частина** 77](#_Toc169347668)

[5.5.1 Постановки задачі 77](#_Toc169347669)

[5.5.2 HTML документ 77](#_Toc169347670)

[ВИСНОВКИ 81](#_Toc169347671)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ 84](#_Toc169347672)

ОМФК.0121 НП 20

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

**Масиви:** Статичні структури даних фіксованого розміру.

**Вектори (STL):** Динамічні масиви, які можуть змінювати свій розмір під час виконання програми.

**Списки (STL):** Двонаправлені зв'язані списки, що дозволяють ефективно додавати або видаляти елементи в середині.

**Множини (STL):** Колекції унікальних елементів.

**Мапи (STL):** Колекції пар "ключ-значення".

**Структури:** Користувацькі типи даних, що об'єднують різні типи даних в одну одиницю.

**STL(Стандартна бібліотека шаблонів) -** бібліотека для C++, що містить набір узгоджених узагальнених алгоритмів, контейнерів, засобів доступу до їхнього вмісту і різних допоміжних функцій.

**JSON (JavaScript Object Notation)** - це текстовий формат обміну даними між комп'ютерами. JSON базується на тексті, може бути прочитаним людиною. Формат дає змогу описувати об'єкти та інші структури даних. Цей формат використовується переважно для передавання структурованої інформації через мережу (завдяки процесу, що називають серіалізацією).

**CSV (Comma-Separated Values) -** файловий формат, котрий є відмежовувальним форматом для представлення табличних даних, у якому поля відокремлюються символом коми та переходу на новий рядок. Поля, що містять коми, декілька рядків, або лапки (позначаються подвійними лапками), мають обмежуватися з обох боків лапками.

**HTML (HyperText Markup Language)** – основа веб-сторінок. Це мова розмітки,

яка використовується для структурування контенту та опису його зовнішнього вигляду в браузері.

**CSS (Cascading Style Sheets)** – мова опису зовнішнього вигляду веб-сторінок, яка контролює шрифти, кольори, розміри, розташування елементів та багато інших аспектів візуального.

# ВСТУП

Автор: Демченко Катерина Сергіївна

Дата: 31.05.2024

Мета: Узагальнення вивчених тем

Перелік вивчених тем:

1. Вивчення питань охорони праці. Ознайомлення з електронною бібліотекою коледжу. Вивчення правил оформлення звітної документації.
2. Засобами MS Word розробити документ складної структури та зберегти у різних форматах.
3. Розв‘язок задач за допомогою засобів табличного редактора MS Excel
4. Виконання практичних завдань за допомогою засобів табличного редактора MS Access.
5. Загальні поняття. Елементи мови С++ - константи, змінні, операції, перетворення типів.
6. Арифметичні операції.
7. Поняття вхідного та вихідного потоку, найпростіші математичні функції.
8. Програмування лінійних та розгалужених обчислювальних процесів.
9. Програмування циклічних обчислювальних процесів.
10. Мова розмітки HTML та структура Веб-сторінки
11. Розмітка тексту за допомогою HTML. Зображення та посилання
12. Каскадні таблиці стилів CSS та їх структура
13. Спадковості і каскадування за допомогою CSS
14. Введення та виведення у базових консольних програмах
15. Структури даних для управління станом програми
16. Робота з файлами: запис та читання
17. Системи контролю версій (VCS)
18. Мова Markdown для опису проекті

# 1 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ

Заходьте до комп’ютерного кабінету тільки з дозволу вчителя, не поспішаючи, не торкаючись обладнання. Не вмикайте та не вимикайте комп’ютери та іншу апаратуру без дозволу викладача. Працюйте на клавіатурі чистими сухими руками, не натискаючи на клавіші без потреби чи навмання. Робоче місце має бути чистим. Не торкайтеся з’єднувальних проводів та проводів живлення. Ніколи не намагайтесь самостійно усунути несправності комп’ютера, негайно повідомте про це викладачеві. Якщо не дотримуватись правил техніки безпеки та поведінки, робота за комп’ютером може завдати шкоди вашому здоров’ю.

# 2 РОЗДІЛ "ІНФОРМАТИКА"

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

***Завдання***

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).

2. Додати в документ текст із файлу «Дива Світу\_текст»;

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.

17. Зберегти документ у форматах .docx і .pdf

18. Оформити звіт, файли виконаної роботи разом із звітом додати в

гугл-клас.

Результат виконання практичних завдань наданий нижче.

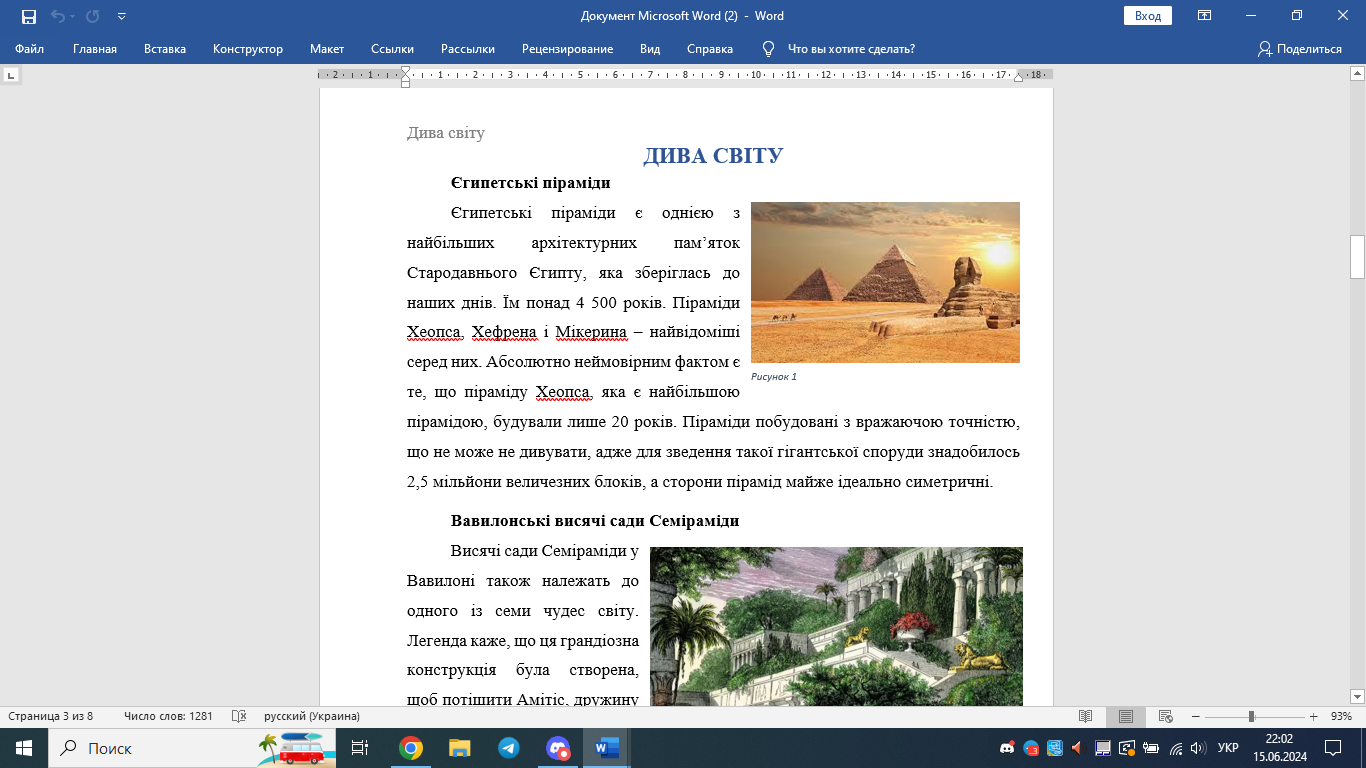
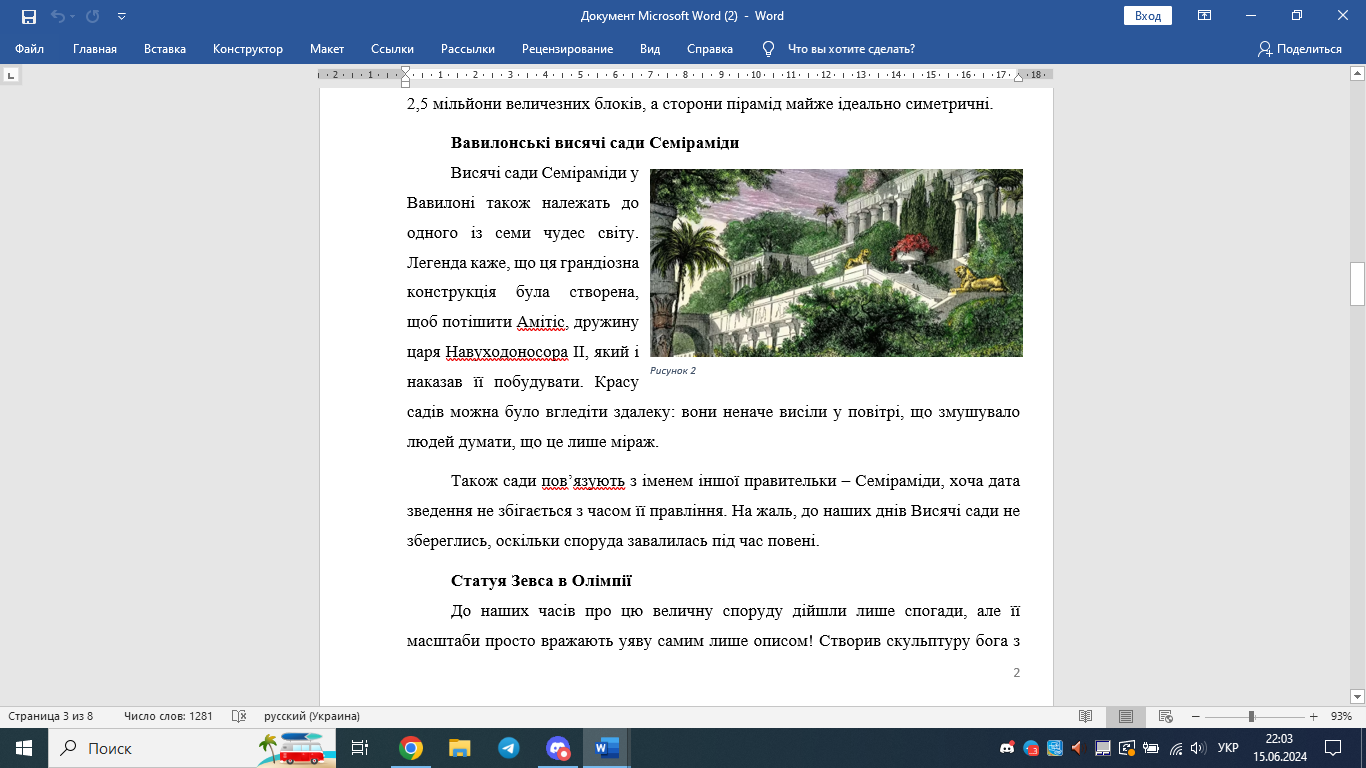


Рисунок 2.1 – Скрішот виконаного завдання

Рисунок 2.1 – Продовження

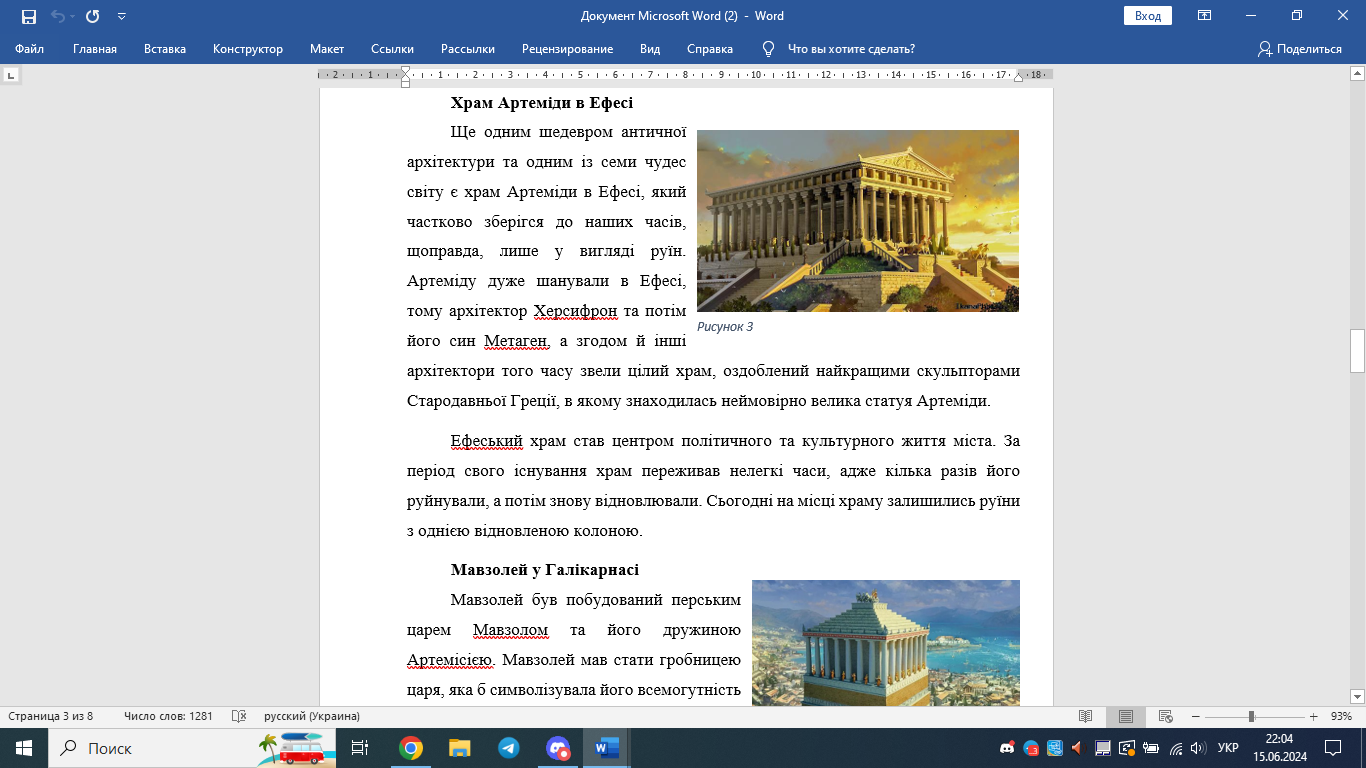


Рисунок 2.2 – Вставка назв об’єктів

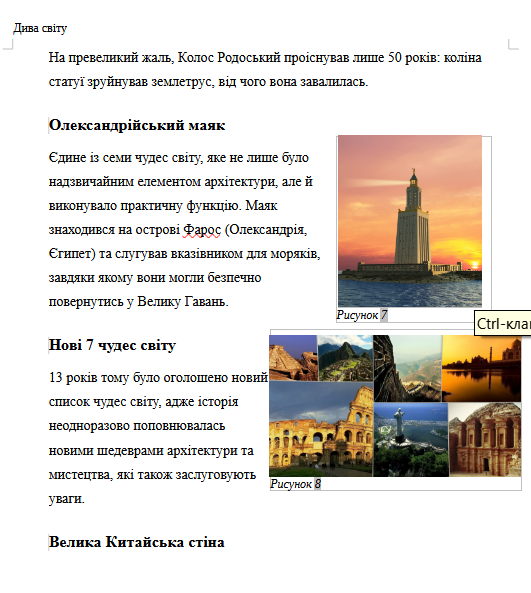


Рисунок 2.3 – Вставка посилань

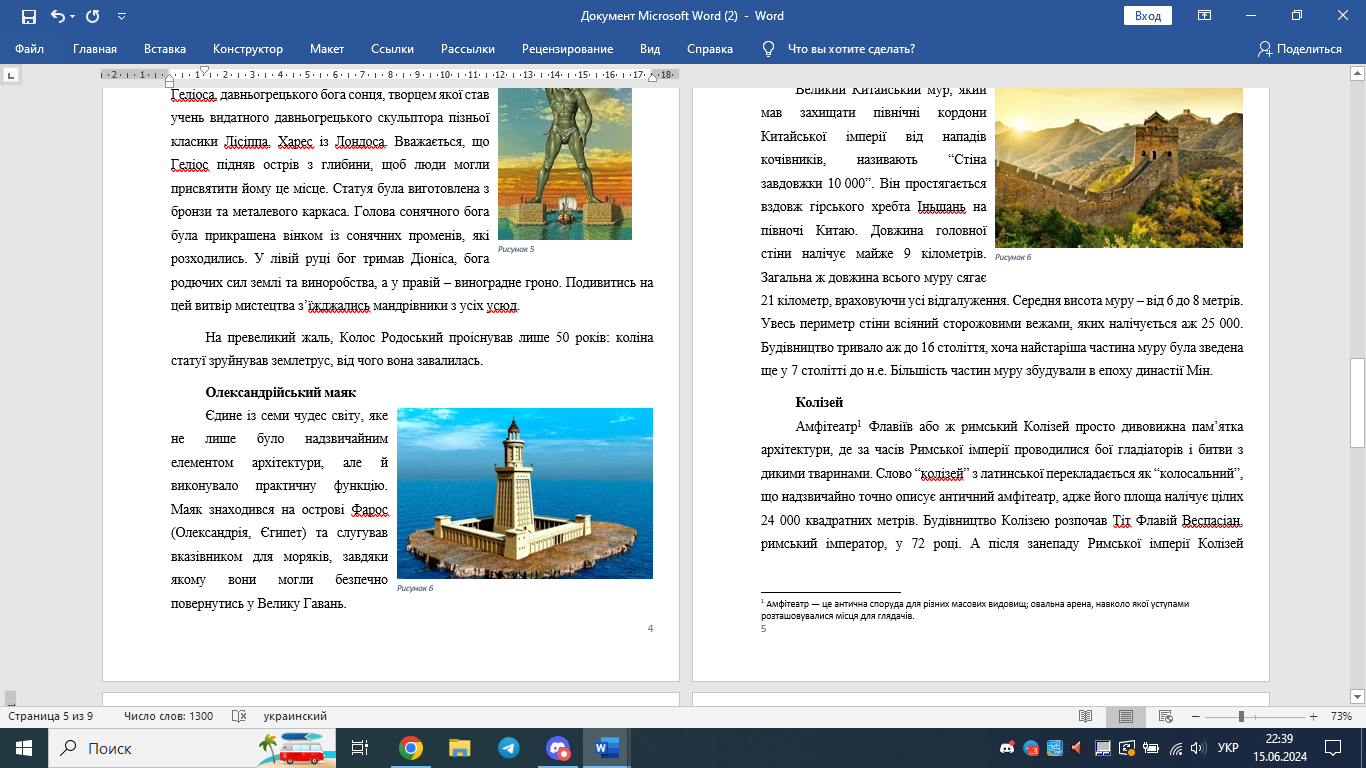


Рисунок 2.4 – Додавання виноски

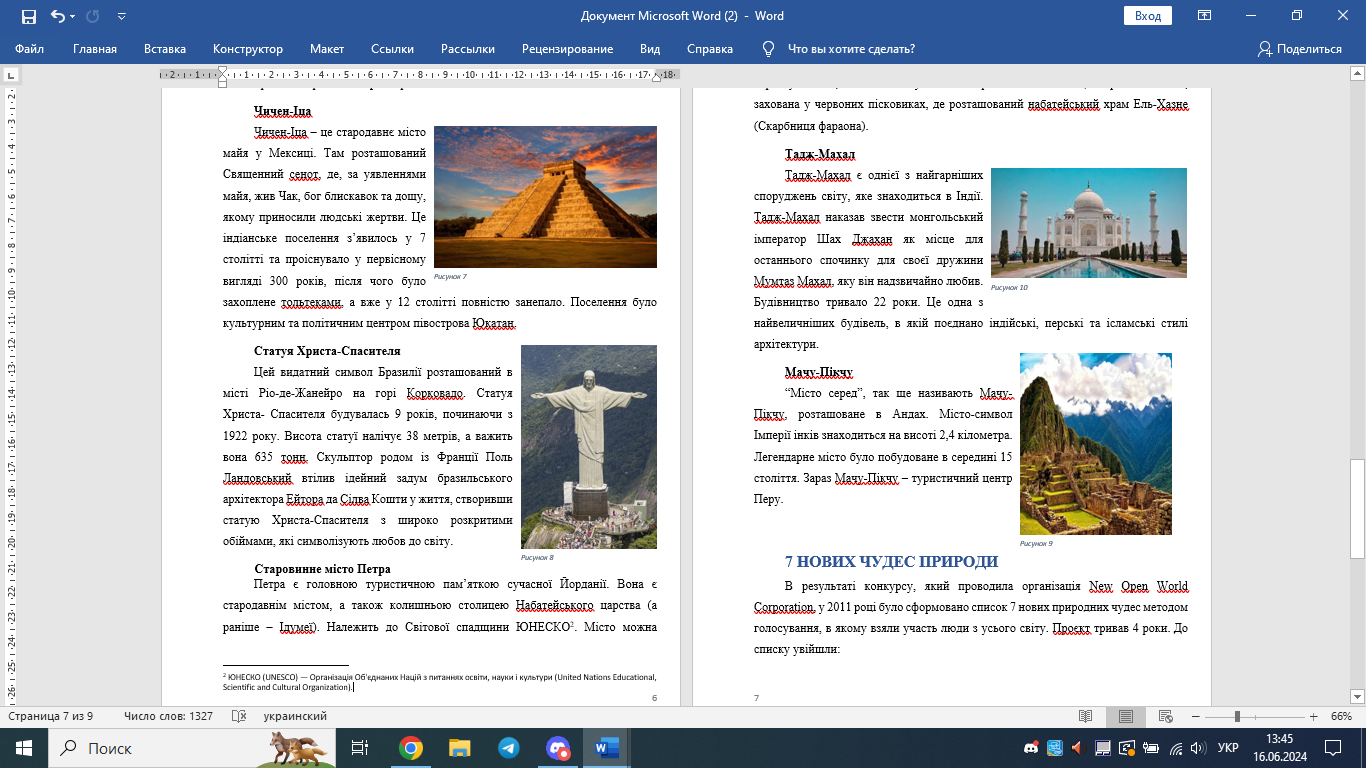


Рисунок 2.6 – Зміна розміру

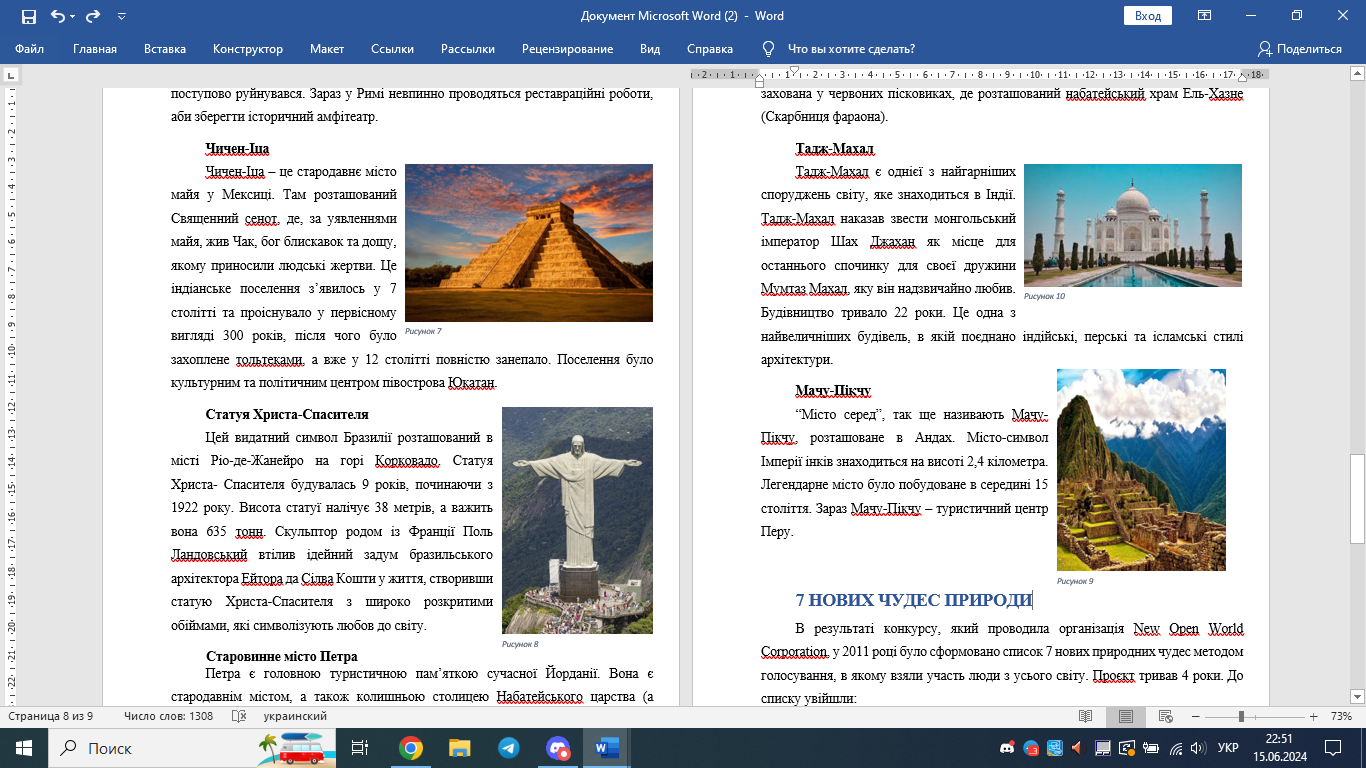


Рисунок 2.7 – Налаштування обтікання рисунків

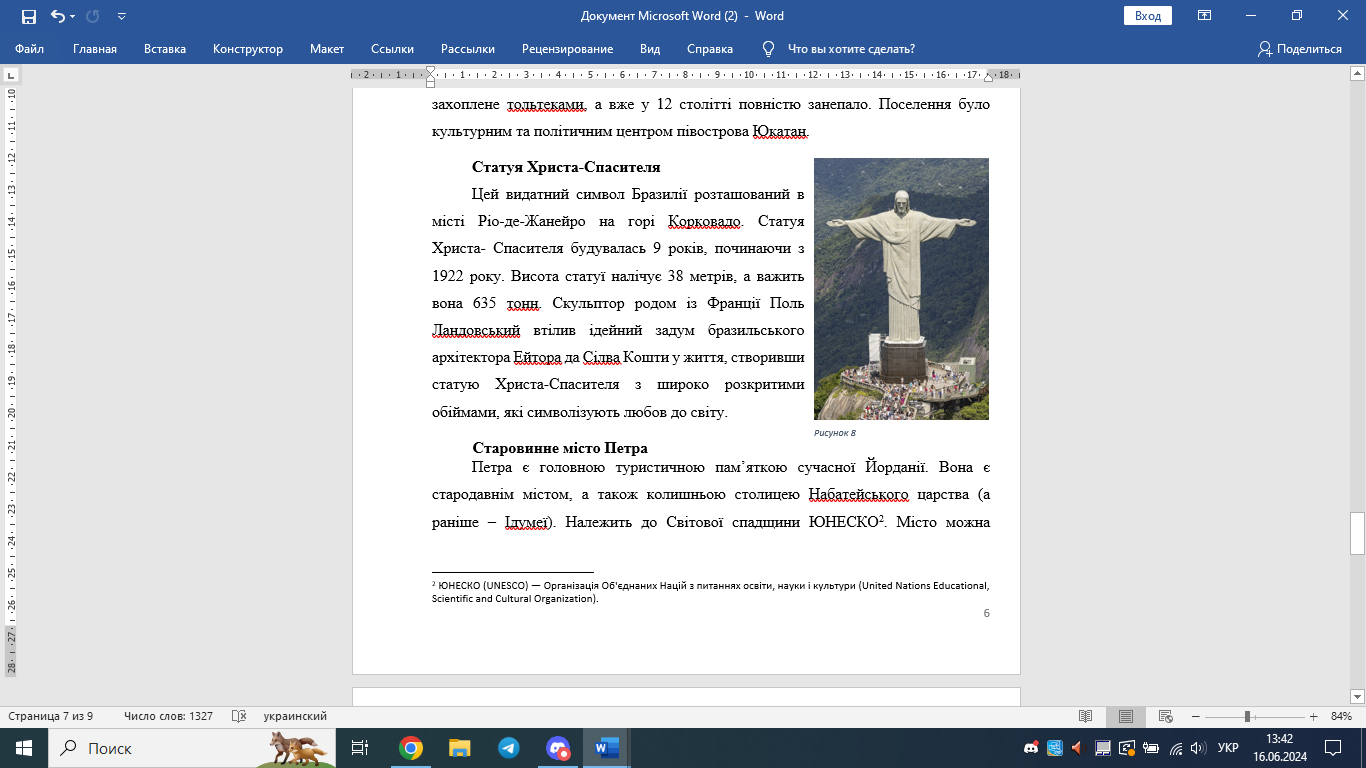


Рисунок 2.8 – Додавання виноски



Рисунок 2.9 – Додавання нового тексту

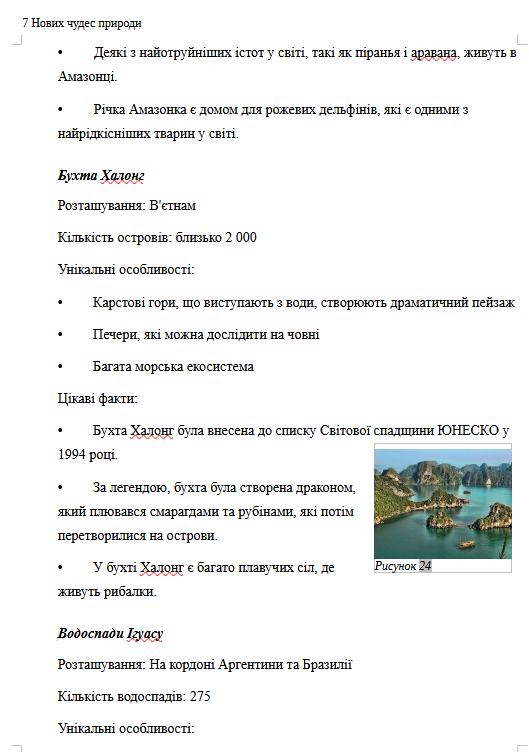


Рисунок 2.10 – Додавання нового тексту

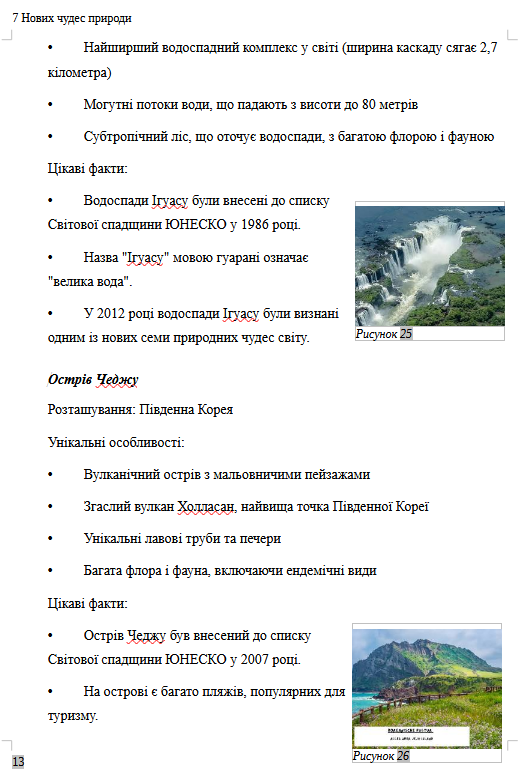


Рисунок 2.11 – Нумерація сторінок

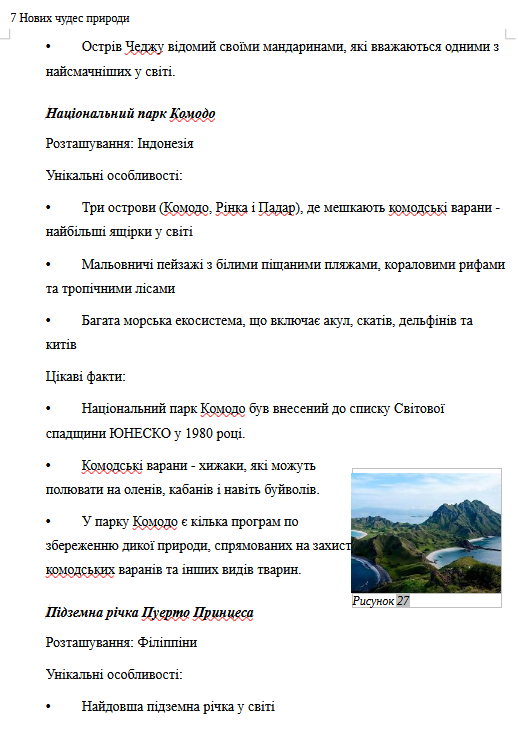


Рисунок 2.12 – Вставка рисунків

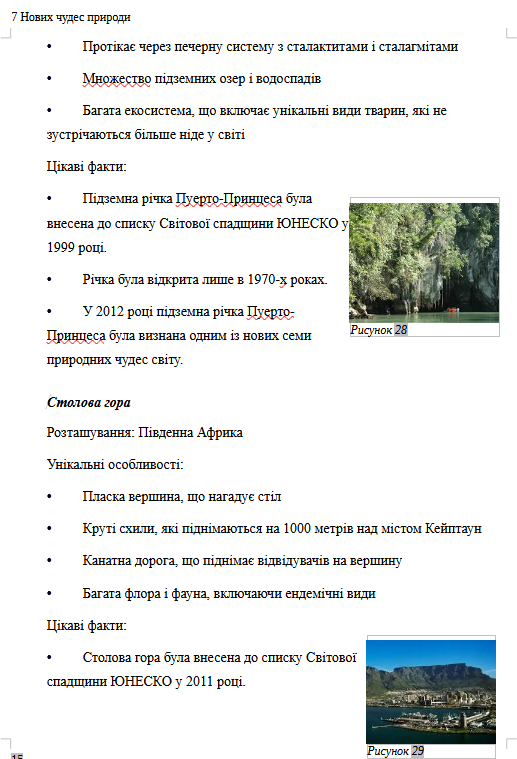


Рисунок 2.13 – Фінальний текст

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

***Завдання:***

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?

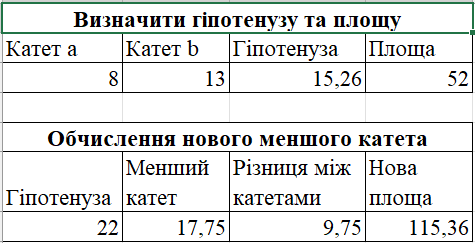


Рисунок 2.14 – Розв’язок до задачі 1

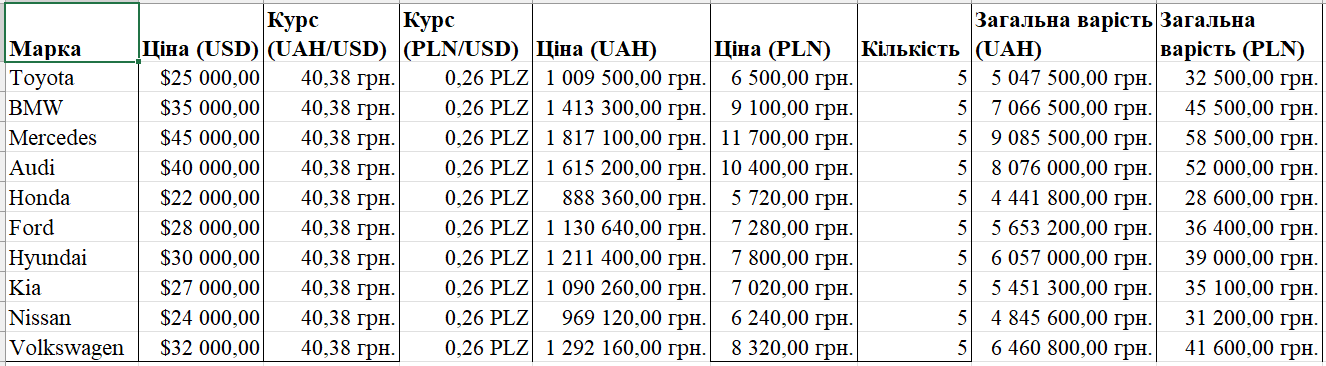
1. Ціну 10 марок автомобілів, задану в $, перерахувати в грошових одиницях України та Польщі, використовуючи дані з інтернету про ціну автомобілів та курси валют. Визначить, яку суму необхідно сплатити автосалону, якщо він закупає по декілька екземплярів кожної моделі. (Марки автомобілів та їх кількість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

Рисунок 2.15 – Розв’язок до задачі 2

1. 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.



Рисунок 2.16 – Розв’язок до задачі 3

1. Засобами табличного процесора здійсніть статистичний аналіз температурного режиму двох весняних місяців (для введення початкових значень температури скористайтесь зразком таблиці на рисунку 2.17).

За веденими даними потрібно визначити:

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



Рисунок 2.17 – Зразок таблиці для виконання завдання

Використовуючи комбіновану діаграму виконайте візуалізацію значень температур за два місяці.

Рисунок 2.18 – Розв'язок до задачі 4

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

***Завдання***

1. Створіть базу даних з назвою ГРУПА
2. Створіть структуру БД ГРУПА із таблицями Студенти, Дисципліни і Успішність. Властивості цих таблиць вказані нижче (див. табл.3.1-3.3).

Таблиця 3.1 – Властивості таблиці Студенти

|  |  |  |
| --- | --- | --- |
| Ім’я поля | Тип даних | Додаткові властивості |
| КодСтудента | Автонумерація | За замовчуванням |
| Прізвище | Короткий текст | Розмір — 20; обов’язкове поле |
| Імя | Короткий текст | Розмір — 15; обов’язкове поле |
| Адреса | Короткий текст | Розмір — 45; необов’язкове поле |
| ДатаНар | Дата й час | Обов’язкове поле |

Таблиця 3.2 – Властивості таблиці Дисципліни

|  |  |  |
| --- | --- | --- |
| Ім’я поля | Тип даних | Додаткові властивості |
| КодДисц | Автонумерація | За замовчуванням |
| НазваДисц | Короткий текст | Розмір — 20; обов’язкове поле |
| Викладач | Короткий текст | Розмір — 20; обов’язкове поле |

Таблиця 3.3 – Властивості таблиці Успішність

|  |  |  |
| --- | --- | --- |
| Ім’я поля | Тип даних | Додаткові властивості |
| КодОцінки | Автонумерація | За замовчуванням |
| КодСтудента | Число | Обов’язкове поле |
| КодДисц | Число | Обов’язкове поле |
| Оцінка | Число | Обов’язкове поле, <13 |

1. В кожну таблицю введіть по п’ять записів, збережіть створену БД.
2. 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.
3. У таблиці Успішність для поля КодСтудент налаштуйте підстановку прізвищ з таблиці Студенти і введіть підпис Прізвище.
4. 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.
5. 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.
6. 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.
7. Доповніть таблицю Дисципліни до 15 записів. Таблицю Успішність доповніть до 20 записів шляхом копіювання й вставляння.
8. У нових записах вибірково замінить окремі прізвища на нові зі списку, поміняйте оцінки.
9. Відкрийте таблицю Студенти, додайте нові записи (до 25) шляхом копіювання та вставляння.
10. 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.
11. Впорядкуйте в таблиці Студенти дані одночасно за полями Прізвище та Імя за алфавітом.
12. У таблиці Успішність виконайте фільтрування за виділеним власним прізвищем.
13. 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.
14. Очистіть усі фільтри. Проведіть фільтрування записів таблиці за допомогою фільтра за формою з умовою: прізвище починається з літери «К», ім’я закінчується літерами «ія».
15. Проведіть фільтрування записів таблиці за допомогою фільтра за формою з умовою: прізвище закінчується на «енко», рік народження — 2007.
16. Налаштуйте фільтр так, щоб вибрати разом усі записи, згадані в пунктах 2.17 і 2.18.
17. Оформіть звіт, файли виконаної роботи разом із звітом додайте в гугл-клас.

Результат виконання практичних завдань наданий нижче.

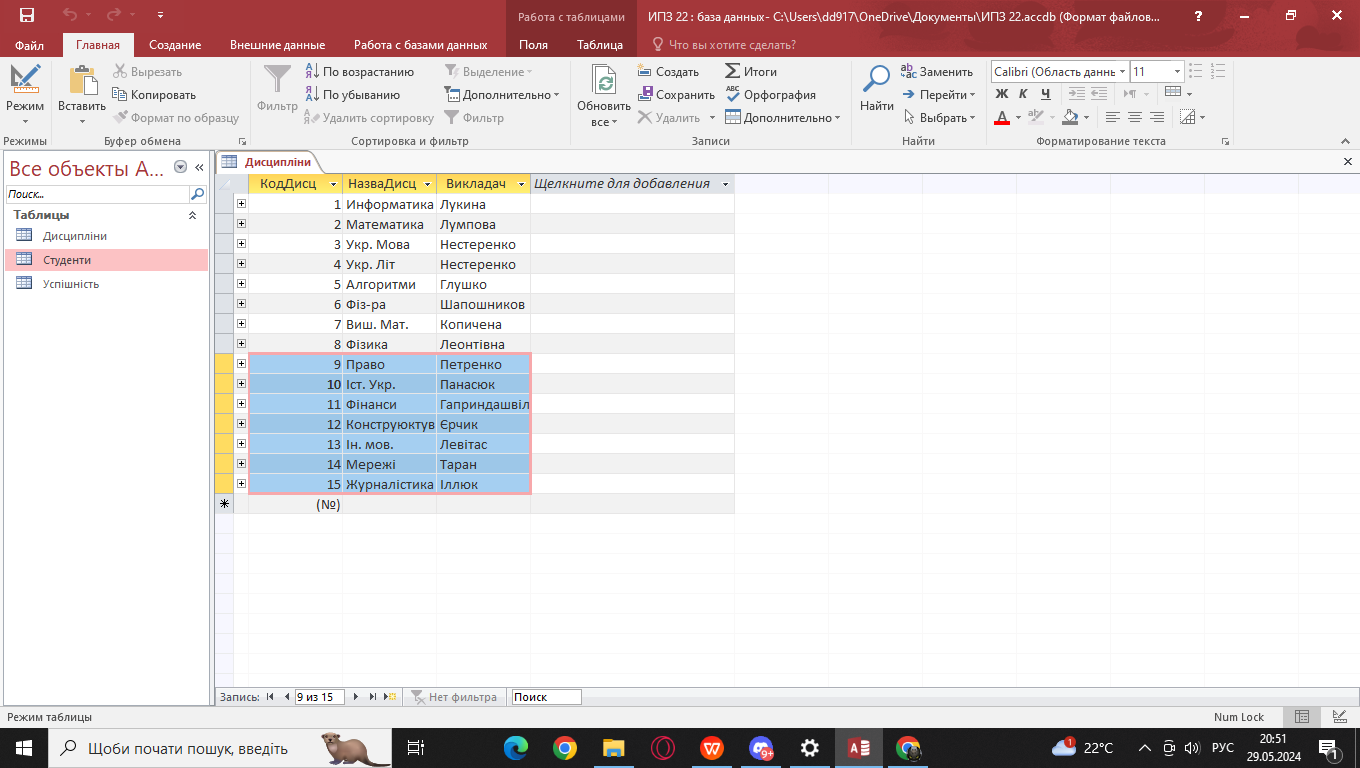


Рисунок 2.19 – Додавання тексту

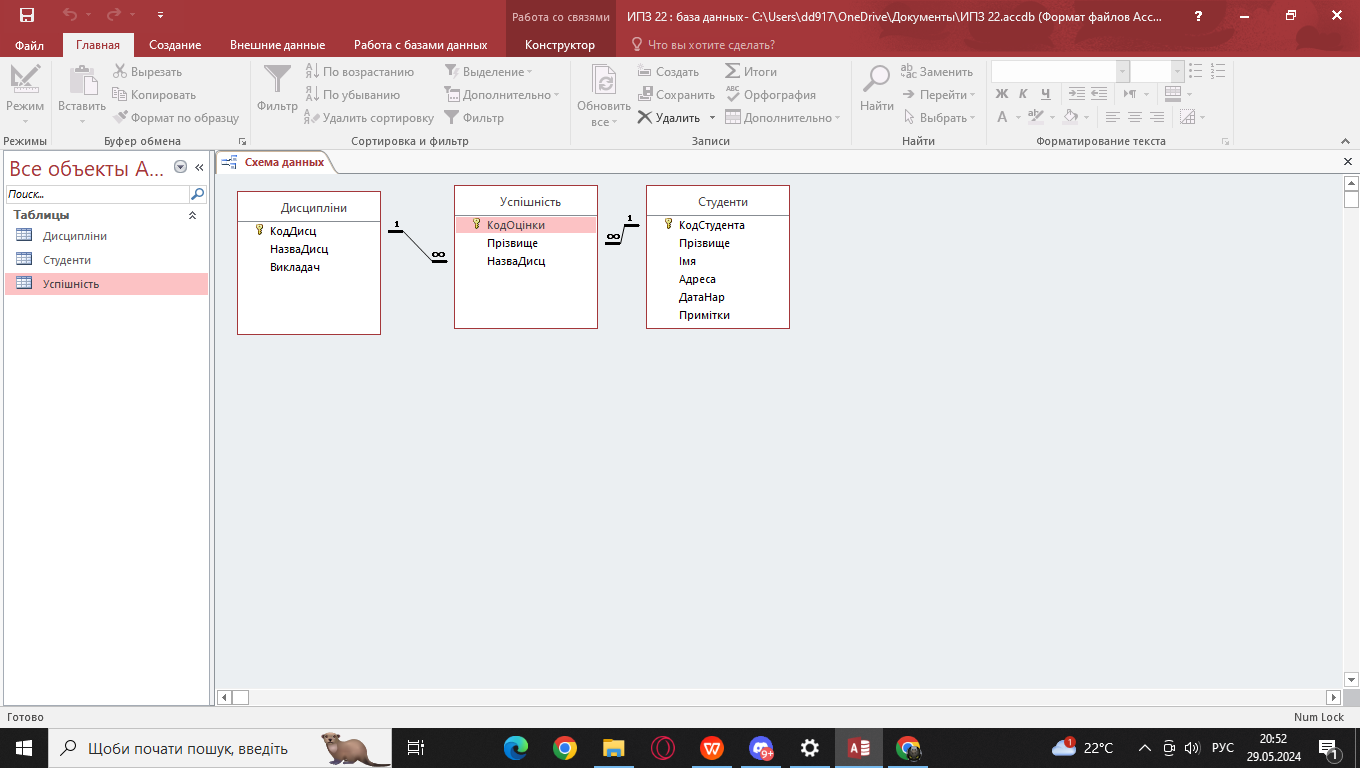


Рисунок 2.20 – Схема

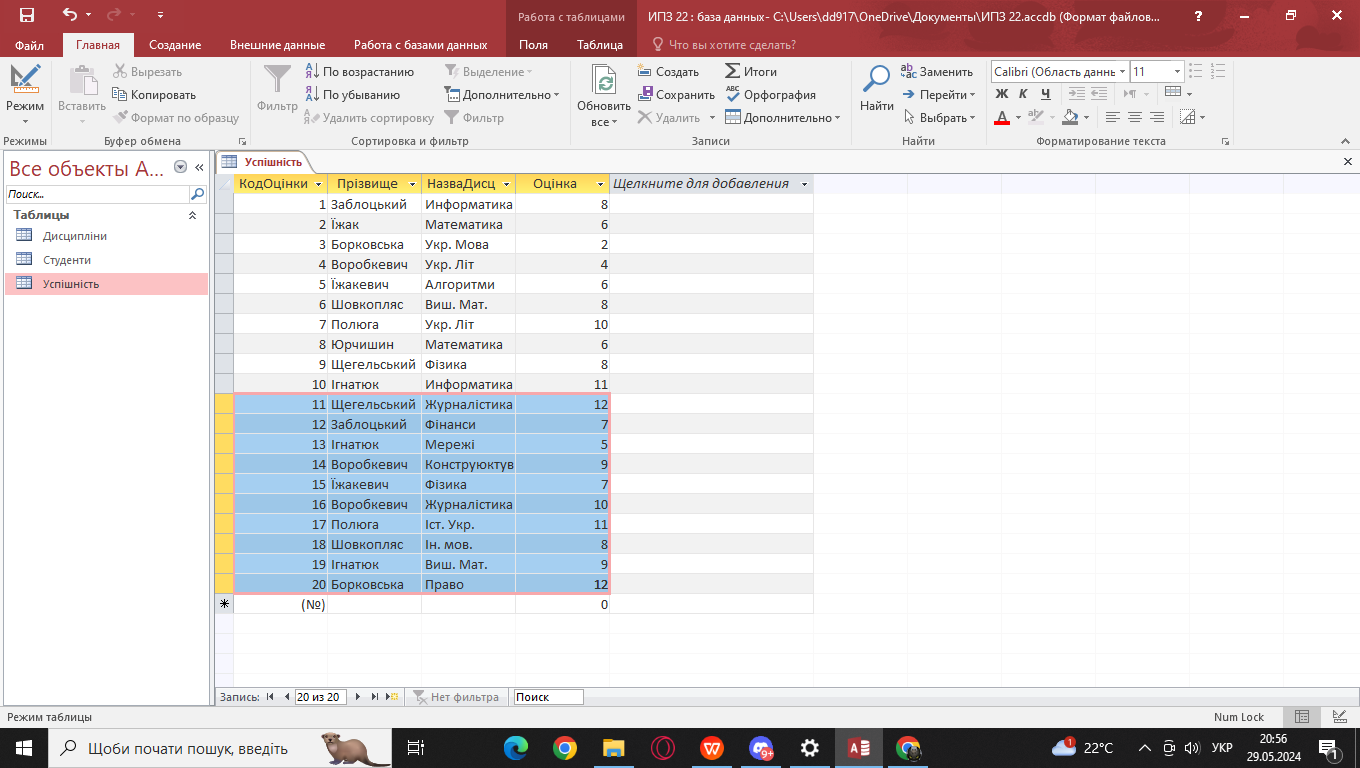


Рисунок 2.21 – Додавання тексту

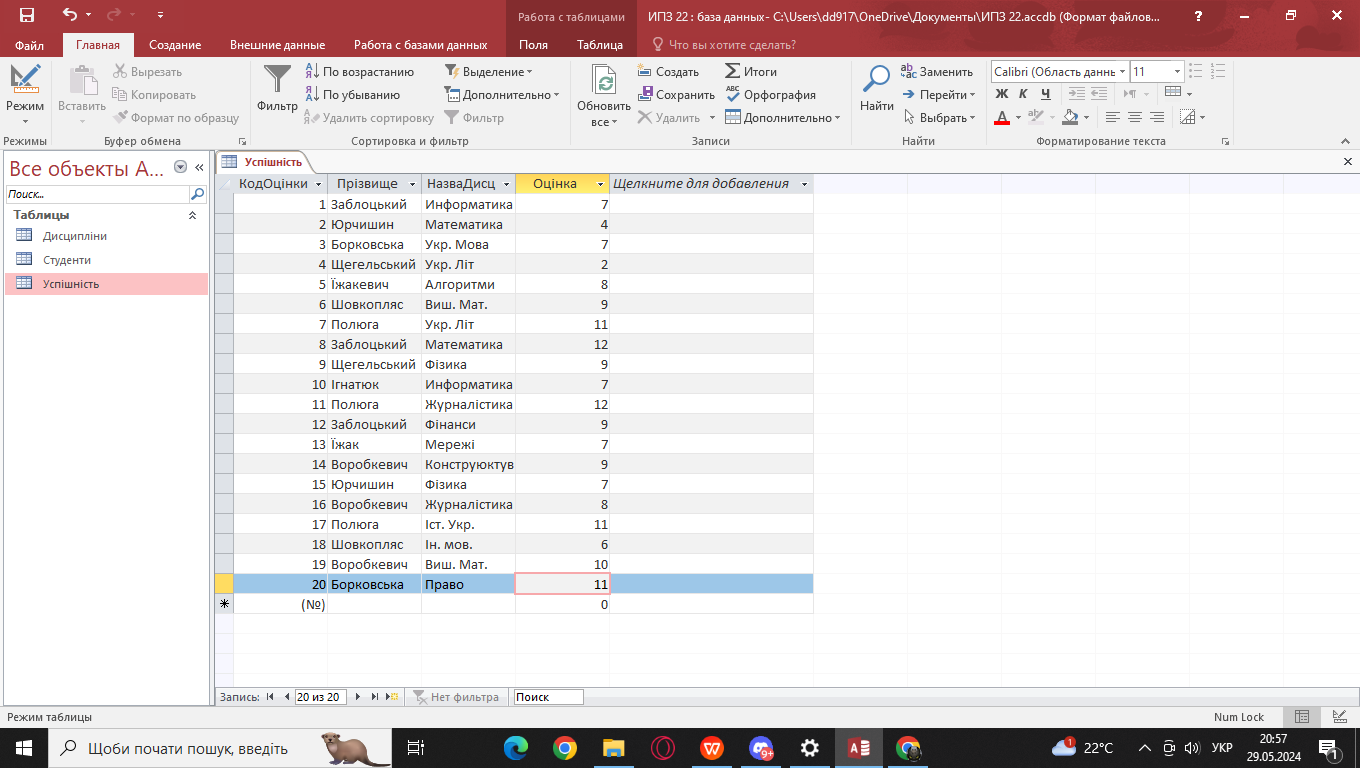


Рисунок 2.22 – Показ працюючої схеми

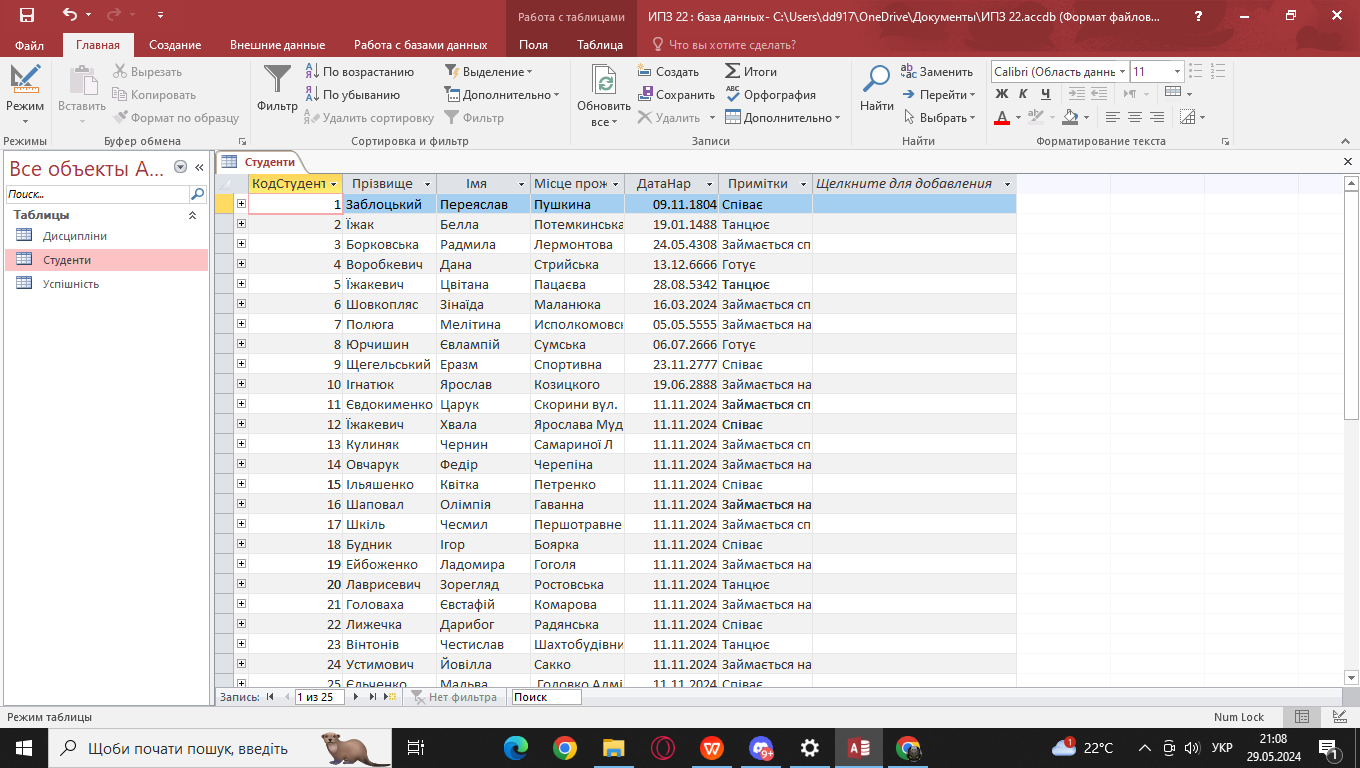


Рисунок 2.23 – Додавання дати народження

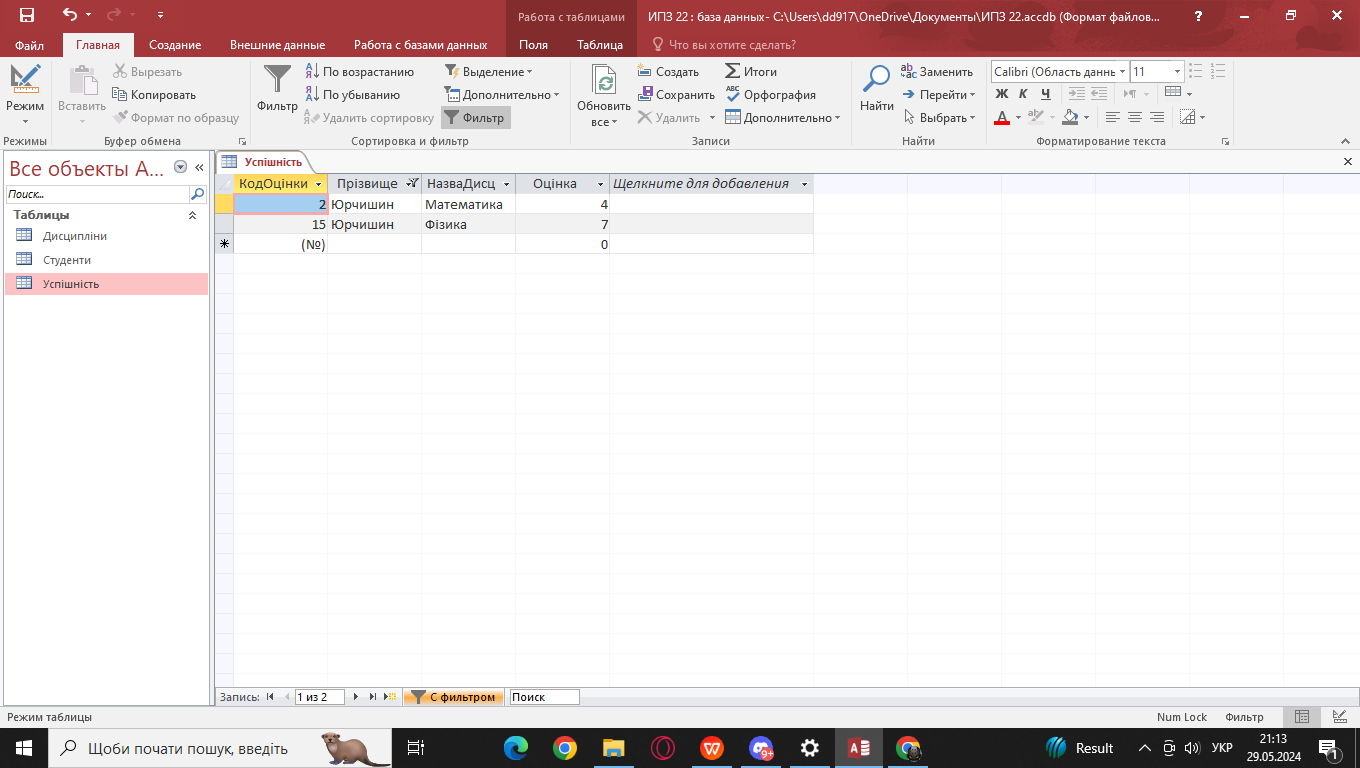


Рисунок 2.24 – Робота схеми

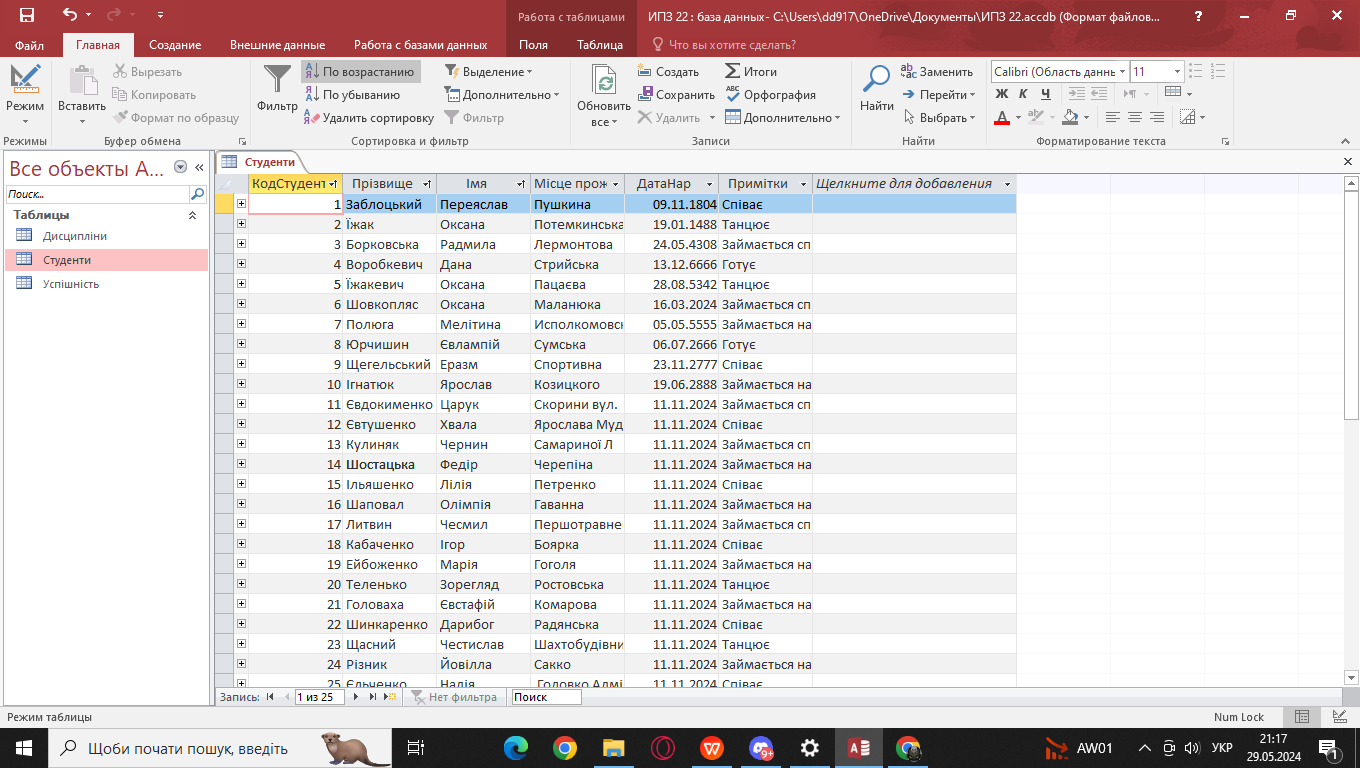


Рисунок 2.25 – Показ всього тексту

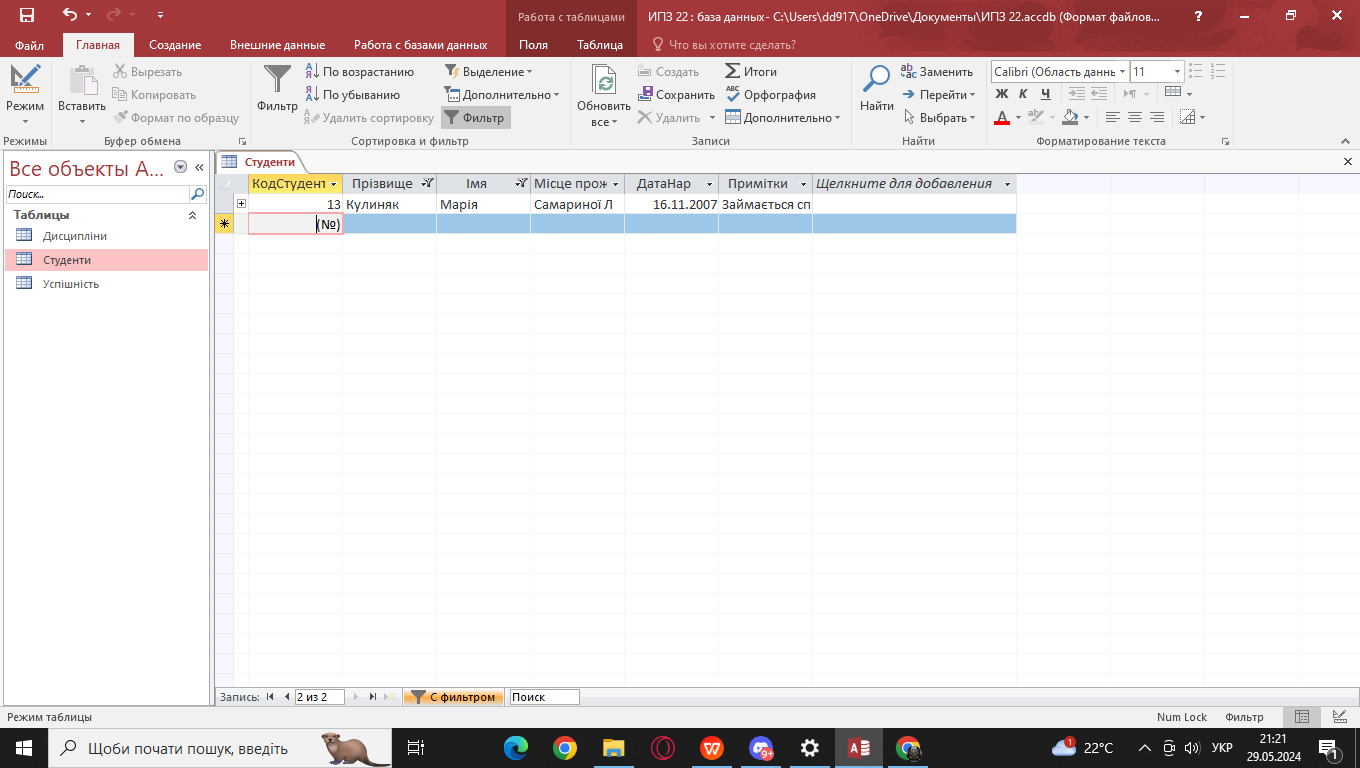


Рисунок 2.26 – Фільтрування

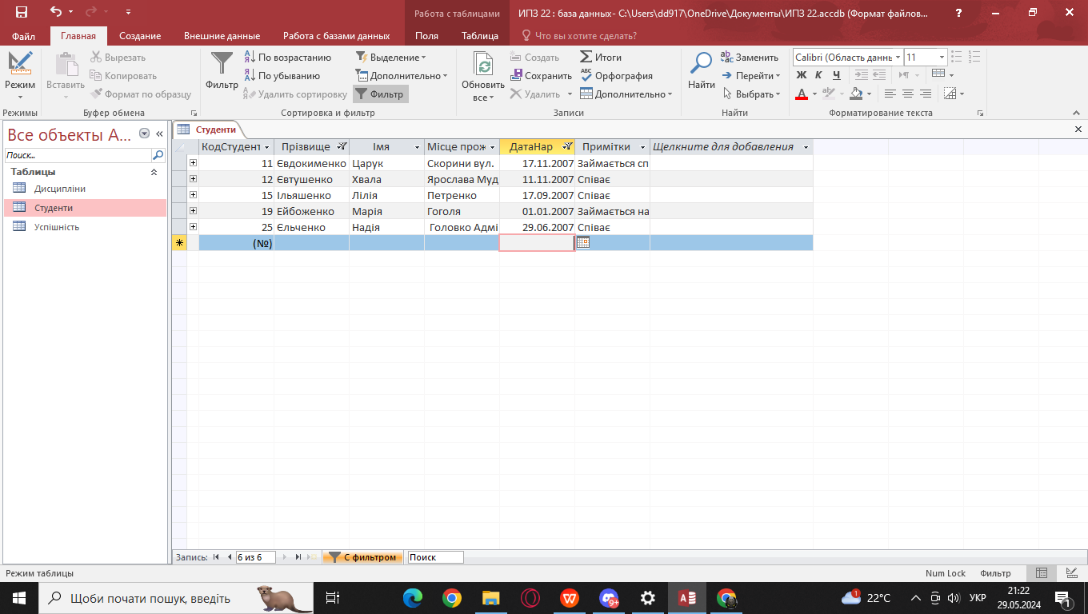


Рисунок 2.27 – Фільтрування

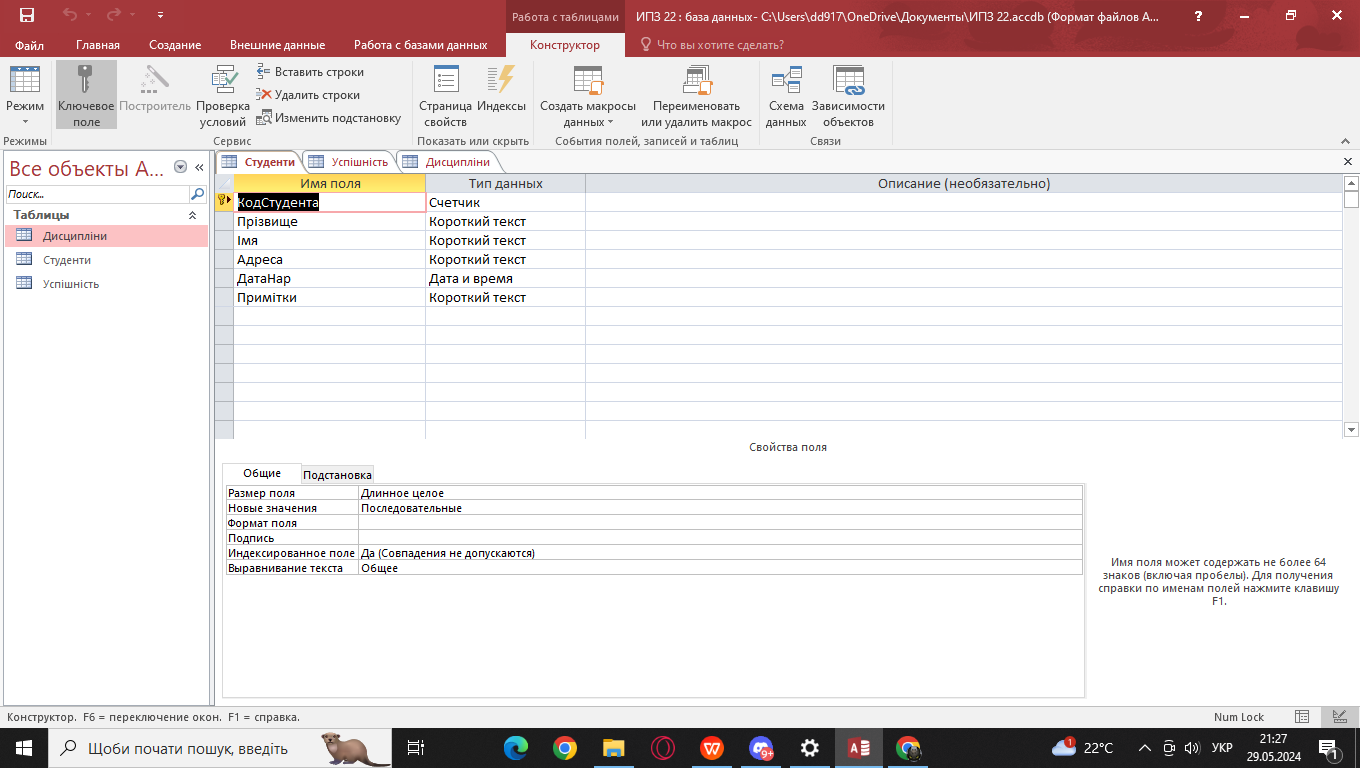


Рисунок 2.28 – Ключ

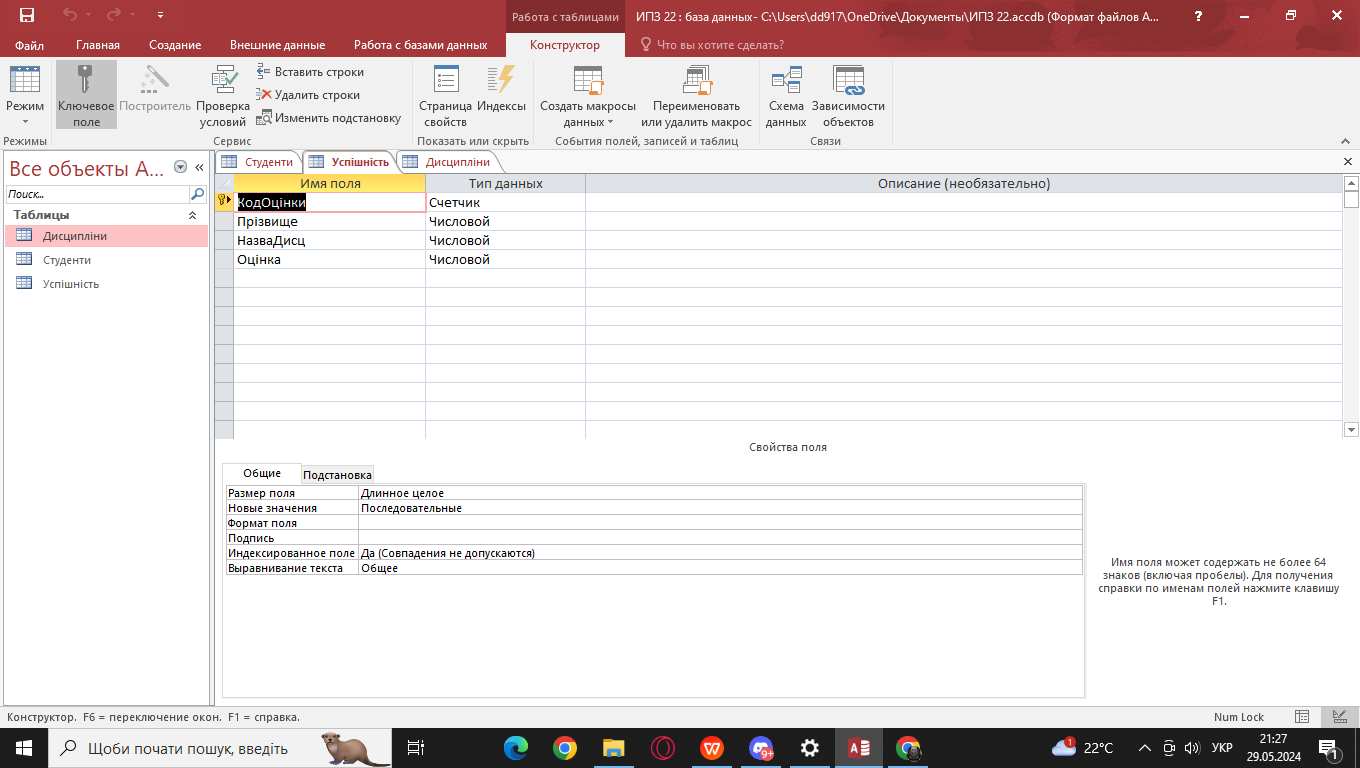


Рисунок 2.29 – Ключ "Дисциаліна"

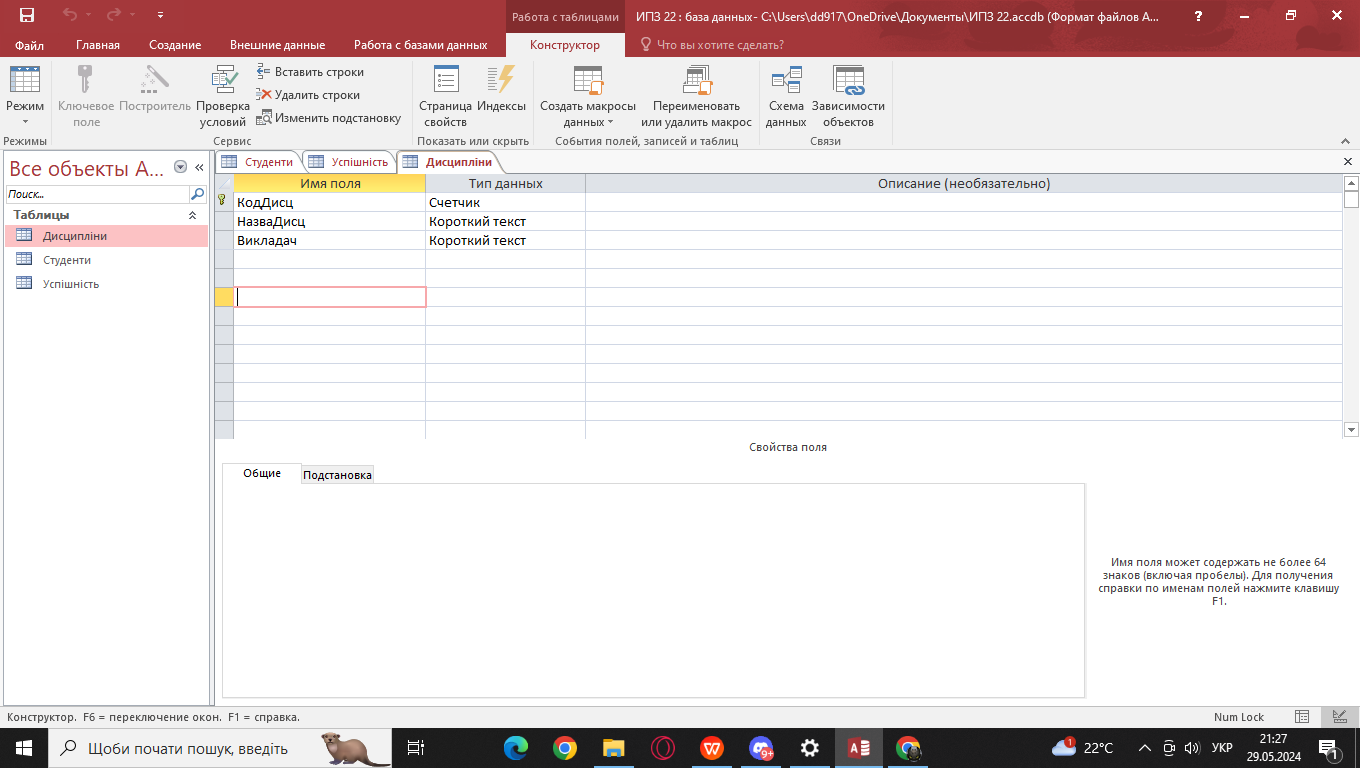


Рисунок 2.30 – Ключі

Рисунок 29 ключи

# 3 РОЗДІЛ "ПРОГРАМУВАННЯ С++"

## **3.1 Теоретичні відомості**

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

Алгоритм - це чітко визначена послідовність дій, що описує процес перетворення вхідних даних в результат вирішення задачі.

**Властивості алгоритмів:**

Детермінірованість: однозначність розуміння та виконання для будь-якого виконавця, що гарантує отримання однакового результату.

Результативність: гарантія досягнення кінцевого результату за скінченне число кроків.

Масовість: можливість застосування для розв'язання задач з певного класу.

Способи представлення алгоритмів:

Текстуальний: опис алгоритму за допомогою природної мови.

Операторний: опис алгоритму за допомогою операторів мови програмування.

Графічний: опис алгоритму за допомогою блок-схем, які візуально представляють послідовність дій.

**Структури алгоритмів:**

Лінійна: структура, де команди виконуються послідовно одна за одною.

З розгалуженням: структура, де в залежності від умови виконується одна з декількох гілок алгоритму.

Циклічна: структура, де певний фрагмент алгоритму повторюється декілька разів.

**Мова програмування C++**

C++ - це універсальна мова програмування, розроблена Денісом Рітчі та Б'ярном Страуструпом. Вона використовується для створення програм різного призначення, від системного програмування до розробки веб-додатків.

**Основні характеристики C++:**

Потужність: C++ надає широкий спектр інструментів для вирішення складних задач.

Гнучкість: C++ дозволяє програмісту контролювати роботу з пам'яттю, що робить її гнучкою та ефективною.

Швидкість: C++ код компілюється в машинний код, що забезпечує високу швидкість виконання програм.

Об'єктно-орієнтоване програмування: C++ підтримує об'єктно-орієнтований підхід до програмування, що робить код більш модульним та структурованим.

**Структура програми C++:**

Директиви препроцесора: інструкції для компілятора, наприклад, підключення бібліотек. Оголошення змінних: визначення типу та імені змінних, що використовуються в програмі.

Функції: блоки коду, які виконують певну задачу.

Головна функція: точка входу в програму, з якої починається її виконання.

**Типи даних:**

C++ підтримує широкий спектр типів даних, включаючи:

Цілі: для зберігання цілих чисел.

Символьні: для зберігання символів.

Логічні: для зберігання логічних значень (true/false).

Дійсні: для зберігання дійсних чисел.

Покажчики: для зберігання адрес інших даних.

Масиви: для зберігання колекції однотипних даних.

Класи: для створення власних типів даних.

**Рекомендації щодо написання програм на C++**

Використання коментарів: коментарі роблять код більш зрозумілим, пояснюючи його логіку та призначення.

Оголошення змінних: перед використанням змінних їх слід оголосити, щоб вказати їх тип та розмір. Використання фігурних дужок: фігурні дужки використовуються для групування операторів, що робить код більш чітким та структурованим.

Відступи: використання відступів покращує читабельність коду, роблячи його візуально структурованим. Оголошення типів даних: чітке оголошення типів даних робить код більш зрозумілим та гарантує його правильну роботу.

**Приклад лінійного алгоритму:**

void printNumbers(int from, int to) {

for (int i = from; i <= to; ++i) {

std::cout << i << " ";

}

std::cout << std::endl;

}

**Приклад алгоритму з розгалуженням:**

bool isEven(int number) {

if (number % 2 == 0) {

return true;

} else {

return false;

}

}

**Рекомендації щодо написання програм на C++:**

// Ця функція обчислює суму двох чисел

int sum(int a, int b) {

return a + b;

}

**Приклад використання фігурних дужок:**

if (x > 0) {

std::cout << "x is positive" << std::endl;

} else {

std::cout << "x is not positive" << std::endl;

}

**Приклад використання відступів:**

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

std::cout << i << " ";

}

std::cout << std::endl;

### 3.1.2 Арифметичні операції

Операції - це дії, які виконуються над даними. Вони поділяються на дві основні групи:

Унарні операції: застосовуються до одного операнда. Прикладами унарних операцій є "+" (додатне число), "-" (від'ємне число), "!" (негація) тощо.

Бінарні операції: застосовуються до двох операндів. Прикладами бінарних операцій є "+", "-", "\*", "/", "%", "==", "!=", "<", ">", "<=", ">=" тощо.

Операнди - це дані, над якими виконуються операції. Вони можуть бути константами, змінними або виразами.

Вирази - це комбінації операндів та операцій, що описують обчислення значення. Значення виразу може бути числом, логічним значенням, адресою пам'яті або іншим типом даних.

**Унарні операції:**

int x = 5;

int y = -x; // y = -5

**Бінарні операції:**

int x = 10;

int y = 5;

int z = x + y; // z = 15

**Вирази:**

int x = 10;

int y = 5;

int z = x + y \* 2; // z = 20

Старшинство операторів визначає порядок виконання операцій у виразі. Оператори з вищим старшинством виконуються раніше, ніж оператори з нижчим старшинством. У C++ існують такі правила старшинства операторів:

Унарні оператори: "+" (додатне число), "-" (від'ємне число), "!" (негація) тощо.

Множення та ділення: "\*", "/".

Додавання та віднімання: "+", "-".

Порівняння: "<", ">", "<=", ">=".

Рівність та нерівність: "==", "!=".

Логічне "і": "&&".

Логічне "або": "||".

Якщо в виразі є оператори з однаковим старшинством, то вони виконуються зліва направо.

Асоціативність операторів визначає напрямок виконання операцій з однаковим старшинством. Оператори з лівобічною асоціативністю виконуються зліва направо, а оператори з правобічною асоціативністю - справа наліво. У C++ більшість операторів мають лівобічну асоціативність.

**Приклади старшинства та асоціативності:**int x = 5 + 3 \* 2; // x = 17

int y = (5 + 3) \* 2; // y = 16

**Операції**

Операції - це дії, які виконуються над даними. Вони поділяються на дві основні групи:

* **Унарні операції:** застосовуються до одного операнда. Прикладами унарних операцій є "+" (додатне число), "-" (від'ємне число), "!" (негація) тощо.
* **Бінарні операції:** застосовуються до двох операндів. Прикладами бінарних операцій є "+", "-", "\*", "/", "%", "==", "!=", "<", ">", "<=", ">=" тощо.

**Операнди**

Операнди - це дані, над якими виконуються операції. Вони можуть бути константами, змінними або виразами.

**Вирази**

Вирази - це комбінації операндів та операцій, що описують обчислення значення. Значення виразу може бути числом, логічним значенням, адресою пам'яті або іншим типом даних.

**Приклади**

* **Унарні операції:**

int x = 5;

int y = -x; // y = -5

* **Бінарні операції:**

int x = 10;

int y = 5;

int z = x + y; // z = 15

* **Вирази:**

int x = 10;

int y = 5;

int z = x + y \* 2; // z = 20

**Старшинство операторів**

Старшинство операторів визначає порядок виконання операцій у виразі. Оператори з вищим старшинством виконуються раніше, ніж оператори з нижчим старшинством. У C++ існують такі правила старшинства операторів:

* **Унарні оператори:** "+" (додатне число), "-" (від'ємне число), "!" (негація) тощо.
* **Множення та ділення:** "\*", "/".
* **Додавання та віднімання:** "+", "-".
* **Порівняння:** "<", ">", "<=", ">=".
* **Рівність та нерівність:** "==", "!=".
* **Логічне "і":** "&&".
* **Логічне "або":** "||".

Якщо в виразі є оператори з однаковим старшинством, то вони виконуються зліва направо.

**Асоціативність операторів**

Асоціативність операторів визначає напрямок виконання операцій з однаковим старшинством.

Оператори з лівобічною асоціативністю виконуються зліва направо, а оператори з правобічною асоціативністю - справа наліво.

У C++ більшість операторів мають лівобічну асоціативність.

**Приклади старшинства та асоціативності**

int x = 5 + 3 \* 2; // x = 17

int y = (5 + 3) \* 2; // y = 16

У першому виразі операція множення буде виконана раніше, ніж операція додавання, тому x = 5 + (3 \* 2) = 5 + 6 = 11.

У другому виразі дужки змушують виконати операцію додавання раніше, ніж операцію множення, тому y = (5 + 3) \* 2 = 8 \* 2 = 16.

**Типи даних**

Операції та операнди можуть мати різні типи даних. Тип даних визначає те, як дані інтерпретуються та обробляються. У C++ існують такі основні типи даних:

* **Цілі числа:** int, short, long, long long.
* **З дійсною комою:** float, double, long double.
* **Символьні:** char.
* **Логічні:** bool.

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

С++ містить широкий спектр математичних функцій, які можна використовувати для виконання різних обчислень. До деяких поширених функцій належать:

sqrt: обчислює квадратний корінь числа.

pow: обчислює значення числа, піднесене до степеня.

exp: обчислює експоненту (e^x).

log: обчислює натуральний логарифм числа.

log10: обчислює десятковий логарифм числа.

abs: обчислює абсолютне значення числа.

sin: обчислює синус кута.

cos: обчислює косинус кута.

tan: обчислює тангенс кута.

acos: обчислює арккосинус числа.

asin: обчислює арксинус числа.

atan: обчислює арктангенс числа.

ceil: округлює число до найближчого більшого цілого.

floor: округлює число до найближчого меншого цілого.

Для використання цих функцій необхідно підключити заголовний файл <cmath>.

Математичні константи

C++ також надає ряд математичних констант, таких як π (pi) та e. Ці константи можна знайти в заголовному файлі <cmath>.

Операції введення-виведення C++ пропонує два основних способи введення та виведення даних: потокове введення-виведення та форматоване введення-виведення.

Потокове введення-виведення використовується для введення та виведення даних у послідовності. Для цього використовуються об'єкти istream та ostream, які зазвичай представлені стандартними потоками cin (введення) та cout (виведення). Форматоване введення-виведення використовується для більш точного контролю над форматом введених та виведених даних. Для цього використовуються функції з заголовного файлу <iostream>.

**Обчислення квадратного кореня:**

#include <iostream>

#include <cmath>

using namespace std;

int main() {

double x;

cout << "Введіть число: ";

cin >> x;

double squareRoot = sqrt(x);

cout << "Квадратний корінь з " << x << " дорівнює "

<<squareRoot << endl;

return 0;

}

**Виведення значення константи π:**

#include <iostream>

#include <cmath>

using namespace std;

int main() {

cout << "Значення π (pi) дорівнює " << M\_PI << endl;

return 0;

}

**Форматований вивід даних:**

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

double price = 123.4567;

cout << fixed << setprecision(2) << "Ціна: $" << price << endl;

return 0;

C++ надає широкий спектр можливостей для роботи з математичними функціями, константами та операціями введення-виведення. Ці можливості дозволяють розробникам програмного забезпечення виконувати різні обчислення та обробляти дані ефективно.

### 3.1.4 Програмування лінійних та розгалужених обчислювальних процесів

Програмування лінійних процесів передбачає створення алгоритмів, де всі дії виконуються послідовно одна за одною. Кожен крок обчислень має чітко визначеного попередника та наступника, що забезпечує простоту реалізації та передбачуваність результатів.

Розгалужені обчислювальні процеси, навпаки, включають умови та рішення, які можуть змінювати хід виконання програми.

Вони використовують конструкції умовних операторів (if, else, switch), дозволяючи програмі виконувати різні дії залежно від вхідних даних або проміжних результатів. Це дає можливість створювати більш гнучкі та складні програми, здатні адаптуватися до різних ситуацій та умов.

В цілому, поєднання лінійних і розгалужених процесів дозволяє створювати ефективні та багатофункціональні обчислювальні програми, здатні вирішувати широкий спектр завдань.

### 3.1.5 Програмування циклічних обчислювальних процесів

Програмування циклічних обчислювальних процесів стосується використання циклів у програмуванні для повторного виконання набору інструкцій. Цикли є основним інструментом для автоматизації та повторення завдань. Основні типи циклів включають:

1. **Цикл "for"**: Використовується для повторення дій певну кількість разів, коли заздалегідь відома кількість ітерацій.
2. **Цикл "while"**: Виконує дії, поки умова залишається істинною, ідеально підходить для ситуацій, коли кількість ітерацій невідома.
3. **Цикл "do-while"**: Подібний до циклу "while", але умова перевіряється після виконання тіла циклу, тому він виконується щонайменше один раз.

Цикли дозволяють ефективно вирішувати задачі, які вимагають багаторазового виконання однакових операцій, такі як обробка масивів даних, генерування послідовностей чисел, реалізація алгоритмів обчислення тощо.

## **3.2 Практичне завдання**

### 3.2.1 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.

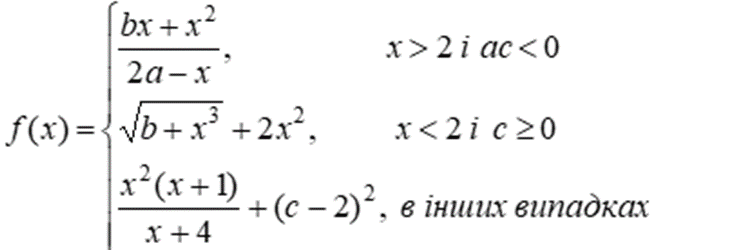


Рисунок 3.1 – Завдання

### 3.2.2 Постановка задачі

Вхідні дані:

* **a** - коефіцієнт рівняння, введене користувачем.
* **b** - коефіцієнт рівняння, введене користувачем.
* **c** - коефіцієнт рівняння, введене користувачем.
* **x\_start** - початкове значення x, введене користувачем.
* **x\_end** - кінцеве значення x, введене користувачем.
* **h** - крок для x, введений користувачем.

Вихідні дані:

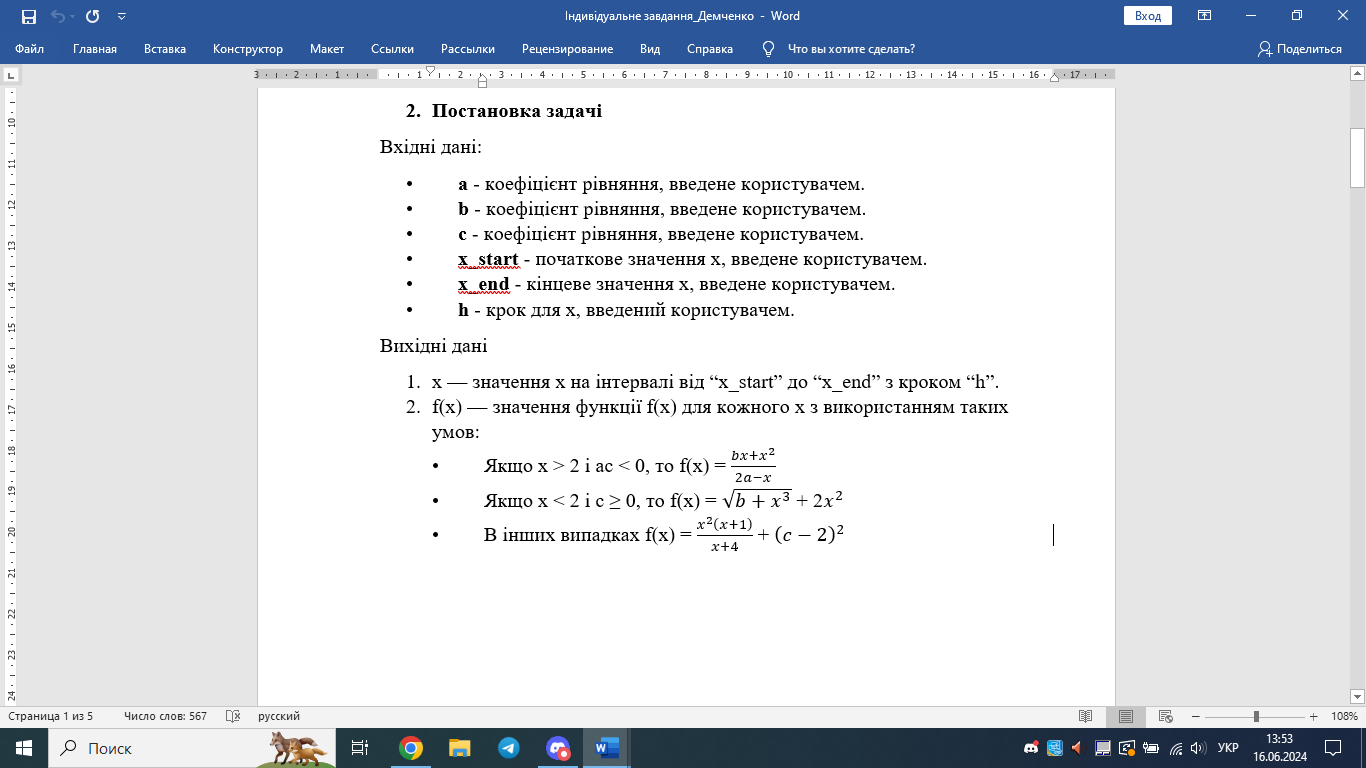


Рисунок 3.2 – Дані

Блок-схема алгоритму роботи програми представлена на рис. 1

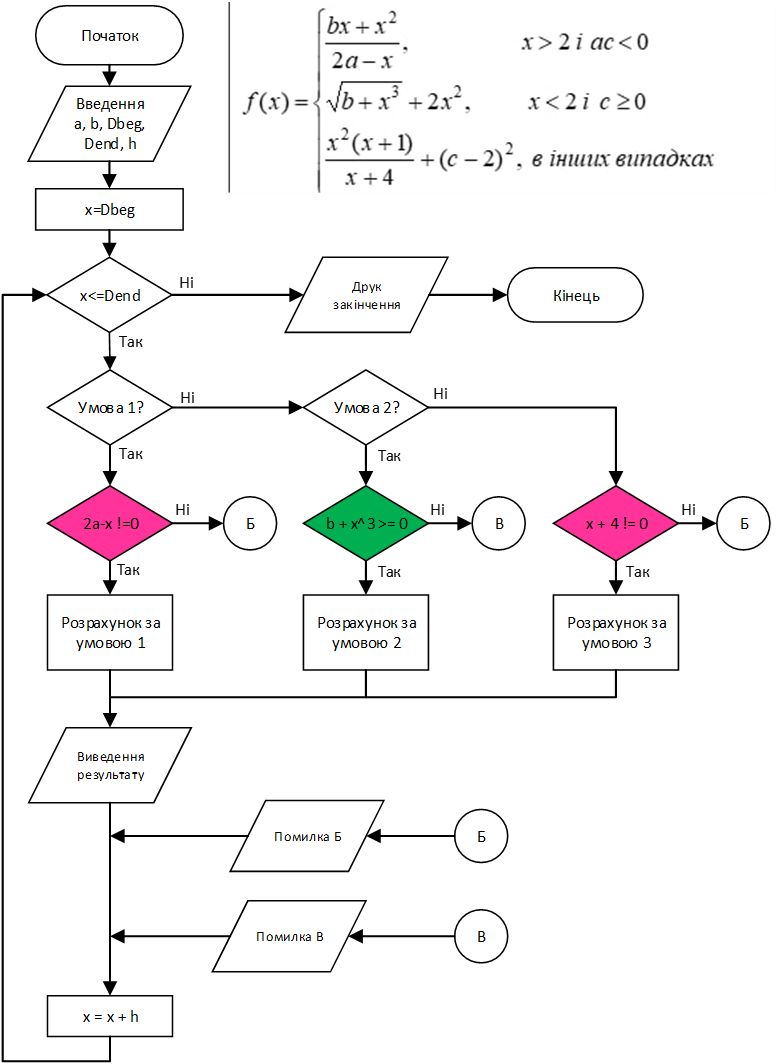


Рисунок 3.3 - Блок-схема алгоритму

### 3.2.3 Код програми

#include <iostream>

#include <cmath>

#include <iomanip>

double calculate\_fx(double a, double b, double c, double x, bool &error) {

error = false;

if (x > 2 && a \* c < 0) {

if (2 \* a - x == 0) { // Check for division by zero

error = true;

return 0;

}

return (b \* x + x \* x) / (2 \* a - x);

} else if (x < 2 && a \* c >= 0) {

if (b + x \* x \* x < 0) { // Check for negative value under the square root

error = true;

return 0;

}

return std::sqrt(b + x \* x \* x) + 2 \* x \* x;

} else {

if (x + 4 == 0) { // Check for division by zero

error = true;

return 0;

}

return (x \* x \* (x + 1)) / (x + 4) + (c - 2) \* (c - 2);

}

}

int main() {

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

// User input

std::cout << "Enter coefficient a: ";

std::cin >> a;

std::cout << "Enter coefficient b: ";

std::cin >> b;

std::cout << "Enter coefficient c: ";

std::cin >> c;

std::cout << "Enter initial value of x (x\_start): ";

std::cin >> x\_start;

std::cout << "Enter final value of x (x\_end): ";

std::cin >> x\_end;

std::cout << "Enter step for x (h): ";

std::cin >> h;

// Error check

if (x\_start >= x\_end) {

std::cerr << "Error: x\_start must be less than x\_end." << std::endl;

return 1;

}

// Setting precision for output

std::cout << std::fixed << std::setprecision(4);

// Calculation and output

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

bool error = false;

double fx = calculate\_fx(a, b, c, x, error);

if (error) {

if (2 \* a - x == 0) {

std::cerr << "Error: Division by zero at x = " << x << std::endl;

} else if (b + x \* x \* x < 0) {

std::cerr << "Error: Negative value under square root at x = " << x << std::endl;

} else if (x + 4 == 0) {

std::cerr << "Error: Division by zero at x = " << x << std::endl;

}

} else {

std::cout << "x = " << x << ", f(x) = " << fx << std::endl;

}

}

return 0;

}

### 3.2.4 Скріншот виконання програми

Скріншот виконання програми представлена на рис.

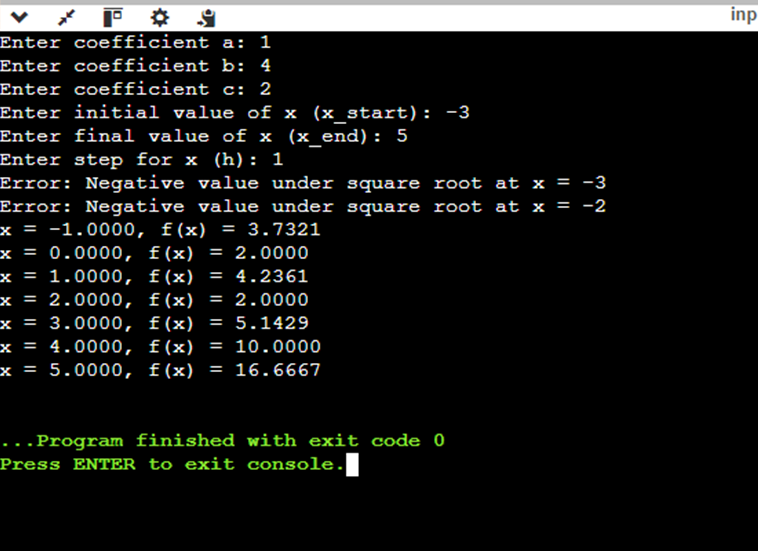


Рисунок 3.4 – Скріншот виконання програми

## **3.3 Виконання тестів**

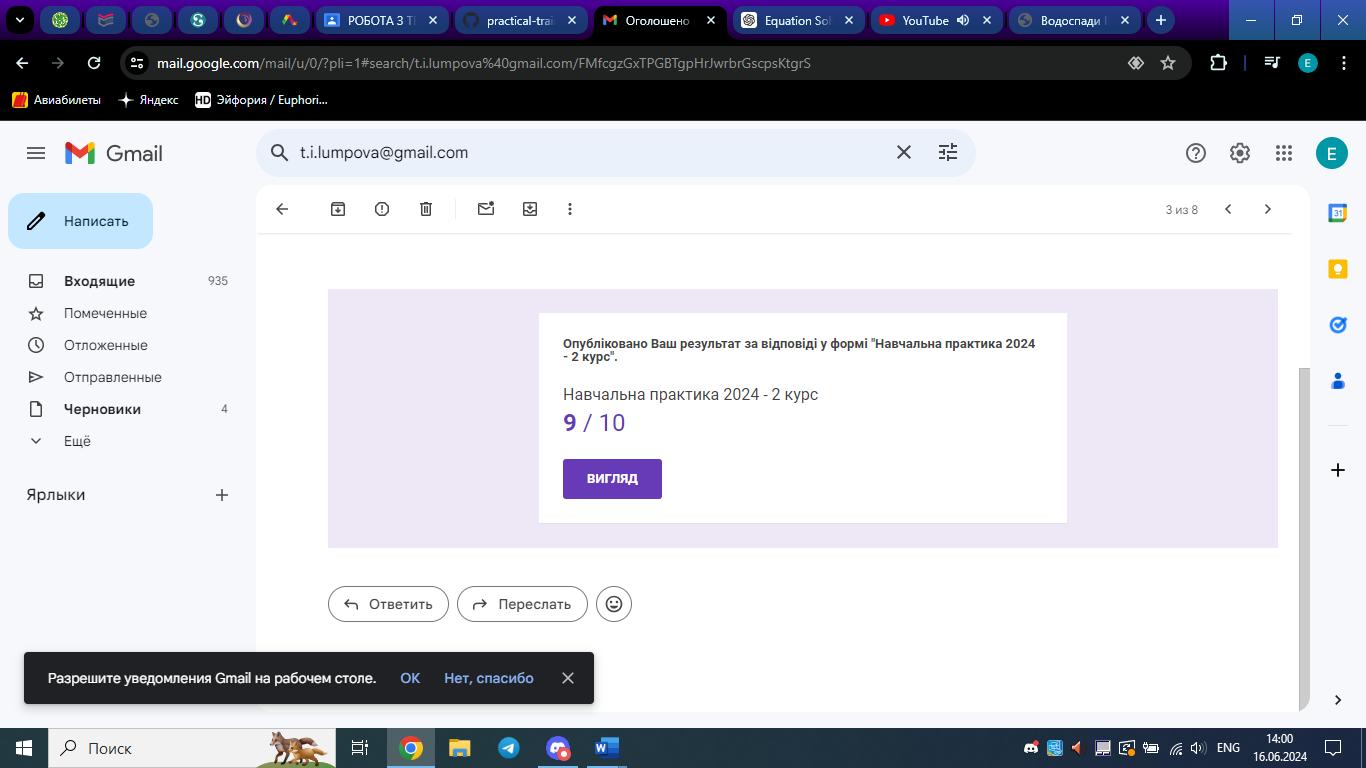


Рисунок 3.5 – Тест

# 4 РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++"

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

Введення та виведення в базових консольних програмах на C++ здійснюється за допомогою стандартної бібліотеки `iostream`, яка містить основні потоки для роботи з консоллю: `cin` (введення) та `cout` (виведення).

**Основи введення та виведення**

*Виведення даних (`cout`)*

Для виведення даних на консоль використовується об'єкт `cout` з бібліотеки `iostream`. Оператор `<<` використовується для передачі даних у потік виведення.

#include <iostream>

int main() {

std::cout << "Hello, World!" << std::endl;

return 0;

}

***Введення даних (`cin`)***

Для введення даних з консолі використовується об'єкт `cin`. Оператор `>>` використовується для зчитування даних з потоку введення.

#include <iostream>

int main() {

int number;

std::cout << "Enter a number: ";

std::cin >> number;

std::cout << "You entered: " << number << std::endl;

return 0;

}

***Введення та виведення чисел***

#include <iostream>

int main() {

int a, b;

std::cout << "Enter two integers: ";

std::cin >> a >> b;

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

return 0;

}

***Введення та виведення рядків***

*Для зчитування цілих рядків використовується функція `std::getline`.*

#include <iostream>

#include <string>

int main() {

std::string name;

std::cout << "Enter your name: ";

std::getline(std::cin, name);

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

return 0;

}

***Введення та виведення дробових чисел***

#include <iostream>

int main() {

double num;

std::cout << "Enter a floating point number: ";

std::cin >> num;

std::cout << "You entered: " << num << std::endl;

return 0;

}

***Робота з кількома змінними***

include <iostream>

int main() {

int age;

double height;

std::string name;

std::cout << "Enter your name: ";

std::getline(std::cin, name);

std::cout << "Enter your age: ";

std::cin >> age;

std::cout << "Enter your height in meters: ";

std::cin >> height;

std::cout << "Name: " << name << ", Age: " << age << ", Height: " << height << " meters" << std::endl;

return 0;

}

***Простий калькулятор***

#include <iostream>

int main() {

double num1, num2;

char operation;

std::cout << "Enter first number: ";

std::cin >> num1;

std::cout << "Enter operator (+, -, \*, /): ";

std::cin >> operation;

std::cout << "Enter second number: ";

std::cin >> num2;

switch (operation) {

case '+':

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

break;

case '-':

std::cout << "Result: " << num1 - num2 << std::endl;

break;

case '\*':

std::cout << "Result: " << num1 \* num2 << std::endl;

break;

case '/':

if (num2 != 0)

std::cout << "Result: " << num1 / num2 << std::endl;

else

std::cout << "Error: Division by zero!" << std::endl;

break;

default:

std::cout << "Invalid operator!" << std::endl;

}

return 0;

}

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

Управління станом програми в C++ може бути реалізовано за допомогою різних структур даних. Основні з них включають масиви, вектори, списки, множини, мапи та структури. Кожна з цих структур має свої переваги та недоліки, і вибір відповідної залежить від конкретних вимог програми.

***Масиви***

Масиви дозволяють зберігати елементи одного типу і мають фіксований розмір.

#include <iostream>

int main() {

int numbers[5] = {1, 2, 3, 4, 5};

for (int i = 0; i < 5; ++i) {

std::cout << numbers[i] << " ";

}

return 0;

}

***Вектори***

Вектори (стандартна бібліотека STL) є динамічними масивами, які можуть змінювати свій розмір під час виконання програми.

#include <iostream>

#include <vector>

int main() {

std::vector<int> numbers = {1, 2, 3, 4, 5};

numbers.push\_back(6); // додавання елемента

for (int number : numbers) {

std::cout << number << " ";

}

return 0;

}

***Списки***

Списки (STL) є двонаправленими зв'язаними списками, що дозволяє ефективно додавати або видаляти елементи в середині.

#include <iostream>

#include <list>

int main() {

std::list<int> numbers = {1, 2, 3, 4, 5};

numbers.push\_back(6); // додавання елемента в кінець

for (int number : numbers) {

std::cout << number << " ";

}

return 0;

}

***Множини***

Множини (STL) використовуються для зберігання унікальних елементів у відсортованому порядку.

#include <iostream>

#include <set>

int main() {

std::set<int> numbers = {3, 1, 4, 1, 5, 9};

numbers.insert(2); // додавання елемента

for (int number : numbers) {

std::cout << number << " ";

}

return 0;

}

***Мапи***

Мапи (STL) зберігають пари "ключ-значення" і дозволяють швидко знаходити значення за ключем.

#include <iostream>

#include <map>

int main() {

std::map<std::string, int> age;

age["Alice"] = 30;

age["Bob"] = 25;

for (const auto& pair : age) {

std::cout << pair.first << ": " << pair.second << std::endl;

}

return 0;

}

***Структури***

Структури (struct) дозволяють об'єднувати дані різних типів у єдину логічну одиницю.

#include <iostream>

struct Person {

std::string name;

int age;

};

int main() {

Person alice;

alice.name = "Alice";

alice.age = 30;

std::cout << alice.name << " is " << alice.age << " years old." << std::endl;

return 0;

}

***Приклад використання декількох структур***

Розглянемо приклад, де використовується структура `Person`, а список осіб зберігається у векторі.

#include <iostream>

#include <vector>

struct Person {

std::string name;

int age;

};

int main() {

std::vector<Person> people;

people.push\_back({"Alice", 30});

people.push\_back({"Bob", 25});

for (const auto& person : people) {

std::cout << person.name << " is " << person.age << " years old." << std::endl;

}

return 0;

}

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

Робота з файлами здійснюється за допомогою стандартної бібліотеки `fstream`, яка містить класи `ifstream` для читання з файлів, `ofstream` для запису у файли та `fstream` для читання та запису.

***Основи роботи з файлами***

Для роботи з файлами необхідно включити заголовковий файл `fstream`.

#include <fstream>

***Запис у файл***

Для запису у файл використовується клас `ofstream`.

#include <iostream>

#include <fstream>

int main() {

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

if (outFile.is\_open()) {

outFile << "Hello, File!" << std::endl; // записуємо рядок у файл

outFile.close(); // закриваємо файл

} else {

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

}

return 0;

}

***Читання з файлу***

Для читання з файлу використовується клас `ifstream`.

#include <iostream>

#include <fstream>

#include <string>

int main() {

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

if (inFile.is\_open())

{

std::string line;

while (std::getline(inFile, line)) { // зчитуємо файл пострічково

std::cout << line << std::endl;

}

inFile.close(); // закриваємо файл

} else {

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

}

return 0;

}

***Читання та запис у той самий файл***

Для читання та запису використовується клас `fstream`.

#include <iostream>

#include <fstream>

#include <string>

int main() {

std::fstream file("example.txt", std::ios::in | std::ios::out | std::ios::app); // відкриваємо файл для читання та запису

if (file.is\_open()) {

// Запис у файл

file << "Additional line" << std::endl;

// Переміщуємо вказівник файлу на початок для читання

file.seekg(0);

// Читання з файлу

std::string line;

while (std::getline(file, line)) {

std::cout << line << std::endl;

}

file.close(); // закриваємо файл

} else {

std::cerr << "Unable to open file for reading and writing" << std::endl;

}

return 0;

}

***Обробка помилок***

Перевірка на успішне відкриття файлу та перевірка стану потоку під час роботи з файлами є важливими для обробки можливих помилок.

#include <iostream>

#include <fstream>

int main() {

std::ifstream inFile("nonexistent.txt"); // спробуємо відкрити неіснуючий файл

if (!inFile) {

std::cerr << "File could not be opened" << std::endl;

return 1;

}

// Читання з файлу

std::string line;

while (std::getline(inFile, line)) {

if (inFile.bad()) { // перевіряємо стан потоку

std::cerr << "Error reading file" << std::endl;

break;

}

std::cout << line << std::endl;

}

inFile.close();

return 0;

}

***Робота з бінарними файлами***

Для запису та читання бінарних файлів використовується той самий `fstream`, але з використанням флагу `std::ios::binary`.

#include <iostream>

#include <fstream>

int main() {

// Запис у бінарний файл

std::ofstream outFile("binary.dat", std::ios::binary);

if (outFile.is\_open()) {

int number = 12345;

outFile.write(reinterpret\_cast<char\*>(&number), sizeof(number));

outFile.close();

} else {

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

}

// Читання з бінарного файлу

std::ifstream inFile("binary.dat", std::ios::binary);

if (inFile.is\_open()) {

int number;

inFile.read(reinterpret\_cast<char\*>(&number), sizeof(number));

std::cout << "Read number: " << number << std::endl;

inFile.close();

} else {

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

}

return 0;

}

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

***JSON***

**Переваги:**

1. Легкість читання і запису: Формат легко читається і записується як людиною, так і машиною.

2. Простота синтаксису: Використовує простий синтаксис ключ-значення.

3. Популярність в веб-розробці: Широко використовується в веб-розробці, особливо для обміну даними між клієнтом та сервером.

4. Підтримка складних структур даних: Може зберігати вкладені об'єкти та масиви, що дозволяє зберігати складні дані

**Недоліки:**

1. Відсутність схем: JSON не має вбудованої схеми для валідації даних.

2. Великий об'єм даних: Може бути менш компактним у порівнянні з CSV для табличних даних.

**Застосування:**

* Обмін даними в веб-розробці (API).
* Зберігання конфігураційних файлів.
* Робота з вкладеними структурами даних.

**Приклад:**

{

"name": "John",

"age": 30,

"isStudent": false,

"courses": ["Math", "Science"],

"address": {

"city": "New York",

"zipcode": "10001"

}

}

***CSV***

**Переваги:**

1. Простота і компактність: Формат є дуже простим і компактним, підходить для табличних даних.

2. Легкість імпорту/експорту: Підтримується багатьма програмами, включаючи Excel та інші таблиці.

3. Висока швидкість обробки: Легкість формату дозволяє швидко читати та писати файли.

**Недоліки**:

1. Відсутність підтримки вкладених структур: Не підтримує вкладені структури даних.

2. Обмеження типів даних: Всі дані зберігаються як рядки, що може призводити до помилок при інтерпретації.

3. Проблеми з роздільниками: Наявність ком в даних може вимагати додаткової обробки (кавунація).

**Приклад**

name,age,isStudent,courses,address\_city,address\_zipcode

John,30,false,"Math;Science",New York,10001

**Порівняння JSON та** **CSV:**

***JSON:***

1.Формат: Текстовий, ключ-значення

2.Читання людиною: Легко читається, особливо при наявності вкладених структур

3.Читання машиною: Легко парситься за допомогою багатьох бібліотек

4.Використання: Широко використовується для API, зберігання конфігурацій, складних структур даних

5.Компактність: Менш компактний для табличних даних

6.Типи даних: Підтримує різні типи даних (рядки, числа, булеві значення, масиви, об'єкти)

***CSV:***

1.Формат: Текстовий, значення розділені комами

2.Читання людиною: Легко читається для простих табличних даних

3.Читання машиною: Легко парситься, але потребує обробки для спеціальних випадків  
4.Використання: Використовується для табличних даних, імпорт/експорт даних в таблицях

5.Компактність: Дуже компактний для табличних даних  
6.Типи даних: Всі дані зберігаються як рядки

**Коли JSON кращий:**

*Складні структури даних:* JSON добре підходить для зберігання та передачі даних з вкладеними об'єктами та масивами.

*Веб-розробка:* JSON є стандартом для обміну даними між клієнтом і сервером в веб-розробці.

*Гнучкість:* JSON підтримує різні типи даних, такі як рядки, числа, булеві значення, об'єкти та масиви.

*Легкість читання і запису:* JSON легко читається і записується як людиною, так і машиною, що робить його зручним для налаштування і конфігурації.

**Коли CSV кращий:**

*Табличні дані:* CSV є ідеальним для зберігання простих табличних даних, таких як списки чи таблиці.

*Імпорт/експорт даних:* CSV файли легко імпортуються та експортуються в програми для роботи з таблицями, такі як Microsoft Excel або Google Sheets.

*Простота:* CSV є простим і компактним форматом, який легко читається та записується.

*Висока швидкість обробки:* Простота формату дозволяє швидко обробляти дані.

## **4.5 Системи контролю версій (VCS)**

Система контролю версій (Version Control System, VCS) — це інструмент, який допомагає розробникам відслідковувати зміни у вихідному коді, співпрацювати над проектами та зберігати історію змін. VCS забезпечує:

* Відстеження історії змін.
* Спільну роботу над проектами.
* Можливість відкату до попередніх версій.
* Розгалуження та злиття гілок.

***Основні типи VCS***

1. Локальні VCS: Зберігають всі версії файлів на одному комп'ютері.

2. Централізовані VCS (CVCS): Використовують центральний сервер для зберігання всіх версій файлів (наприклад, SVN, CVS).

3. Розподілені VCS (DVCS): Кожен розробник має повну копію всього репозиторію (наприклад, Git, Mercurial).

***Популярні VCS***

* Git: Найпоширеніша система контролю версій, розроблена Лінусом Торвальдсом.
* Subversion (SVN): Популярна централізована система контролю версій.
* Mercurial: Розподілена система контролю версій, схожа на Git.

***Ініціалізація репозиторію***

git init

***Клонування репозиторію***

git clone https://github.com/username/repository.git

***Додавання файлів у індекс***

git add filename.cpp

git add .

***Створення коміту***

git commit -m "Коментар до коміту"

***Перегляд статусу репозиторію***

git status

***Перегляд історії комітів***

git log

***Створення нової гілки***

git branch new-branch

***Перемикання між гілками***

git checkout new-branch

***Злиття гілок***

git checkout main

git merge new-branch

***Відправка змін на віддалений репозиторій***

git push origin main

***Отримання змін з віддаленого репозиторію***

git pull origin main

**Приклад використання Git**

***Створення проекту***

1. Створіть новий директорій для проекту:

mkdir my\_cpp\_project

cd my\_cpp\_project

2. Ініціалізуйте Git-репозиторій:

git init

3. Створіть файл `main.cpp`:

#include <iostream>

int main() {

std::cout << "Hello, World!" << std::endl;

return 0;

}

4. Додайте файл до індексу та створіть коміт:

git add main.cpp

git commit -m "Додано файл main.cpp з початковим кодом"

5. Створіть файл `.gitignore` для виключення непотрібних файлів з контролю версій:

echo "\*.o" >> .gitignore

echo "\*.exe" >> .gitignore

git add .gitignore

git commit -m "Додано .gitignore"

6. Підключіть віддалений репозиторій (наприклад, на GitHub):

git remote add origin https://github.com/username/my\_cpp\_project.git

git push -u origin main

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

Markdown — це легка мова розмітки, яка дозволяє форматувати текст у простий та зрозумілий спосіб. Markdown широко використовується для написання документації, особливо на таких платформах, як GitHub. Ось приклад оформлення документації для проекту у стилі Markdown з поясненням кожної функції:

**Пояснення функцій Markdown:**

1. Заголовки: Використовуються для структурування документа (`#`, `##`, `###` і т.д.).

2. Списки: Для створення нумерованих (`1.`, `2.`) або ненумерованих списків (`-`, `\*`).

3. Кодові блоки: Для вставки коду використовуються потрійні зворотні лапки (\```).

4. Посилання: Створюються за допомогою квадратних дужок та круглих дужок (`[текст](URL)`).

5. Виділення тексту: Можна використовувати одинарні або подвійні зірочки для виділення тексту (\*italic\* або \*\*bold\*\*).

## **4.7 Розроблення ігрової програми**

### 4.7.1 Код програми

#include <iostream>

using namespace std;

char board[3][3];

char currentPlayer;

void initializeBoard() {

char start = '1';

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

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

board[i][j] = start++;

}

}

}

void printBoard() {

cout << "\n";

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

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

cout << board[i][j];

if (j < 2) cout << " | ";

}

cout << "\n";

if (i < 2) cout << "--+---+--\n";

}

cout << "\n";

}

bool isValidMove(int move) {

int row = (move - 1) / 3;

int col = (move - 1) % 3;

return (board[row][col] != 'X' && board[row][col] != 'O');

}

void makeMove(int move) {

int row = (move - 1) / 3;

int col = (move - 1) % 3;

board[row][col] = currentPlayer;

}

bool isWinner() {

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

if (board[i][0] == board[i][1] && board[i][1] == board[i][2]) return true;

if (board[0][i] == board[1][i] && board[1][i] == board[2][i]) return true;

}

if (board[0][0] == board[1][1] && board[1][1] == board[2][2]) return true;

if (board[0][2] == board[1][1] && board[1][1] == board[2][0]) return true;

return false;

}

bool isDraw() {

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

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

if (board[i][j] != 'X' && board[i][j] != 'O') return false;

}

}

return true;

}

void switchPlayer() {

currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';

}

int main() {

initializeBoard();

currentPlayer = 'X';

int move;

while (true) {

printBoard();

cout << "Player " << currentPlayer << ", enter your move (1-9): ";

cin >> move;

if (move < 1 || move > 9 || !isValidMove(move)) {

cout << "Invalid move. Try again.\n";

continue;

}

makeMove(move);

if (isWinner()) {

printBoard();

cout << "Player " << currentPlayer << " wins!\n";

break;

}

if (isDraw()) {

printBoard();

cout << "The game is a draw!\n";

break;

}

switchPlayer();

}

return 0;

}

### 4.7.2 Скріншоти виконання програми

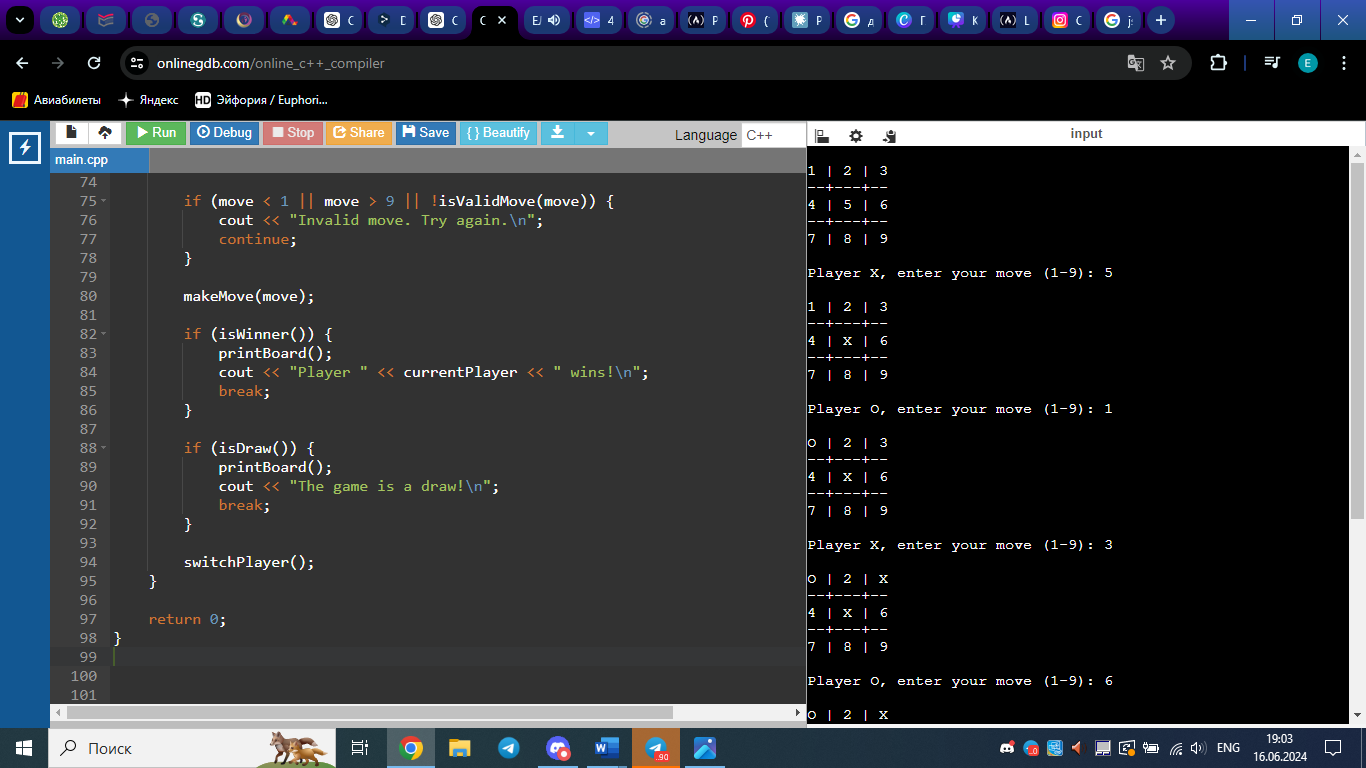


Рисунок 4.1 – Запуск гри

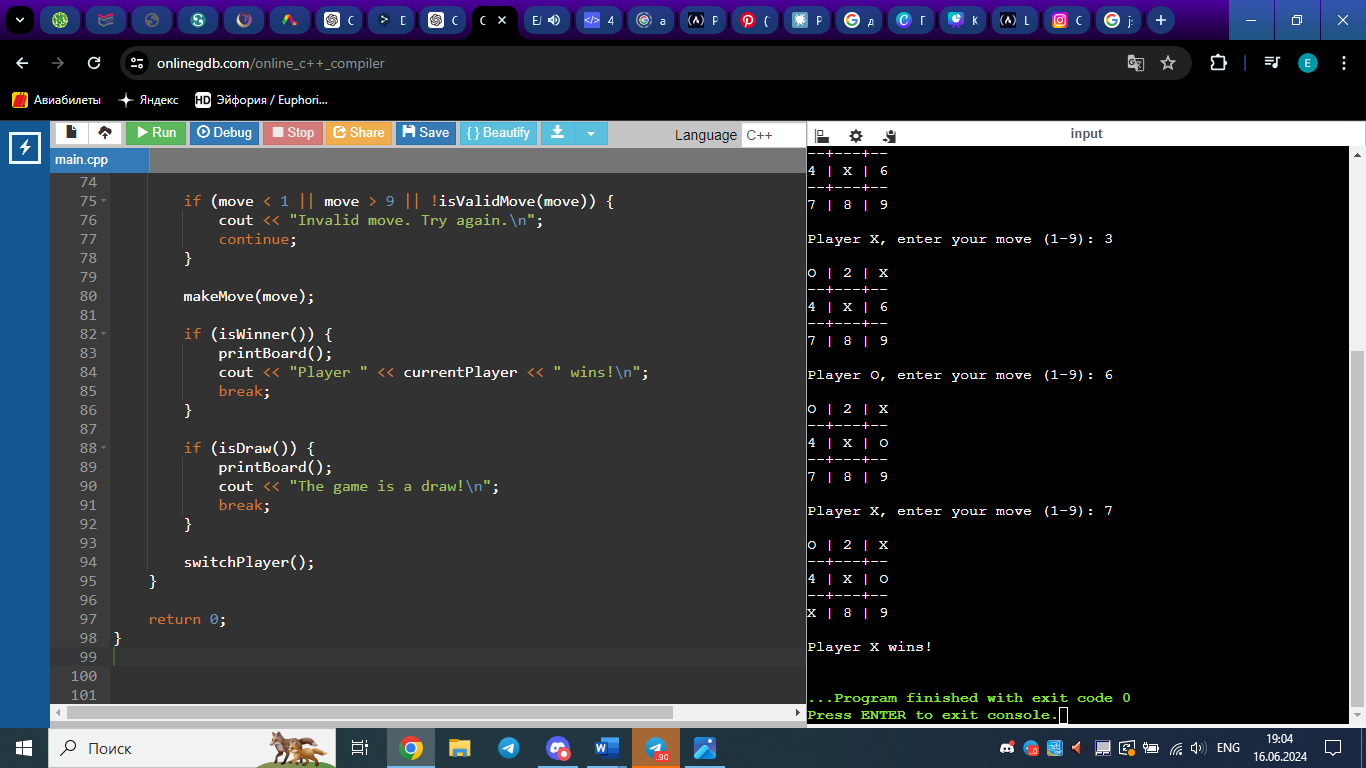


Рисунок 4.2 – Перемога гравця

# 5 РОЗДІЛ ”ВЕБ-ДИЗАЙН ТА HTML”

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

**HTML (HyperText Markup Language)** – це основа веб-сторінок. Це мова розмітки, яка використовується для структурування контенту та опису його зовнішнього вигляду в браузері.

**Структура HTML-документа:**

* **<!DOCTYPE html>**: Ця декларація вказує тип документа браузеру.
* **<html>**: Цей тег є кореневим елементом HTML-документа і охоплює весь його контент.
* **<head>**: Цей тег містить метадані про веб-сторінку, такі як заголовок, кодування символів та посилання на зовнішні ресурси (CSS, JavaScript).
* **<body>**: Цей тег містить видимий контент веб-сторінки, який буде відображатися в браузері користувача.
* **Теги**: HTML використовує теги для визначення різних елементів на веб-сторінці, таких як заголовки, абзаци, списки, зображення, посилання тощо.
* **Атрибути**: Теги можуть мати атрибути, які надають додаткову інформацію про елемент.

**Важливі моменти:**

* HTML не є мовою програмування. Це мова розмітки, яка використовується для опису структури та вмісту веб-сторінки.
* Існує багато типів тегів HTML, кожен з яких має свою функцію.
* Правильна структура HTML-документа є важливою для забезпечення його правильної роботи в різних браузерах.
* Ви можете використовувати інструменти розробника в своєму браузері, щоб переглянути HTML-код веб-сторінки.
* Існує багато ресурсів для вивчення HTML, включаючи онлайн-підручники, книги та курси.

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

**Зображення:**

* Використовуйте тег <img> для вставки зображення в HTML-документ.
* Атрибути <img>:
  + - src: Шлях до файлу зображення.
    - alt: Альтернативний текст для зображення (відображається, якщо зображення не завантажується).
    - width: Ширина зображення.
    - height: Висота зображення.

**Посилання:**

* Використовуйте тег <a> для створення посилання в HTML-документі.
* Атрибути <a>:
* href: URL-адреса, на яку веде посилання.
* target: Цільове вікно для посилання (\_blank, \_self тощо).
* title: Підказка для посилання (з'являється при наведенні курсору)

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

**Що таке CSS?**

CSS (Cascading Style Sheets) - це мова опису зовнішнього вигляду веб-сторінок. Вона дає змогу контролювати шрифти, кольори, розміри, розташування елементів та багато інших аспектів візуального оформлення. CSS використовується разом з HTML, мовою розмітки веб-сторінок, для створення привабливих та зручних сайтів.

**Переваги використання CSS:**

* **Розділення змісту та оформлення:** CSS відділяє опис структури та змісту веб-сторінки (HTML) від її візуального оформлення (CSS). Це робить код більш чітким, гнучким та легким для обслуговування.
* **Повторне використання:** CSS-стилі можна повторно використовувати для різних елементів на сторінці, що економить час та код.
* **Зовнішні таблиці стилів:** CSS можна зберігати в окремих файлах (.css), які потім підключаються до HTML-сторінок. Це дозволяє легко змінювати стилі для багатьох сторінок одночасно.
* **Доступність:** CSS робить веб-сторінки більш доступними для людей з обмеженими можливостями, таких як користувачі скріншот-ридерів.

**Структура CSS:**

* **Селектори:** Селектори визначають, до яких елементів HTML застосовуються стилі.
* **Властивості:** Властивості описують аспекти візуального оформлення елементів, наприклад, колір фону, розмір шрифту, відступи тощо.
* **Значення:** Значення властивостей визначають конкретні характеристики візуального оформлення.

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

**Спадковість**

Спадковість в CSS - це механізм, який дозволяє властивостям CSS автоматично застосовуватися до елементів, що не мають явно заданих значень для цих властивостей. Це спрощує код CSS, роблячи його більш лаконічним і керованим.

Існує два типи спадковості:

* **Пряма спадковість:** Властивості CSS успадковуються від батьківського елемента до його дочірніх елементів.
* **Непряма спадковість:** Властивості CSS успадковуються від батьківського елемента до його дочірніх елементів, а також до їхніх дочірніх елементів і так далі.

**Каскадування**

Каскадування в CSS - це процес, який визначає, яке значення властивості CSS буде використовуватися, коли для одного і того ж елемента задано декілька значень.

При каскадуванні діють наступні правила:

* **Специфічність:** Більш специфічні селектори мають пріоритет над менш специфічними.
* **Порядок:** Значення властивостей, задані пізніше, мають пріоритет над значеннями, заданими раніше.
* **Важливість:** Значення властивостей, позначені як !important, мають пріоритет над усіма іншими значеннями.

**Важливі моменти**

* Спадковість можна використовувати для створення базових стилів, які автоматично застосовуються до всіх елементів на сторінці.
* Каскадування дозволяє вам перевизначити значення властивостей CSS для конкретних елементів або груп елементів.
* Знання принципів спадковості та каскадування є ключовими для написання ефективного та лаконічного CSS-коду.

## **5.5 Практична частина**

### 5.5.1 Постановки задачі

Необхідно розробити веб-сторінку з власним резюме (CV) вказавши власні дані, фото, контакти.

### 5.5.2 HTML документ

Код:

**Index.html:**

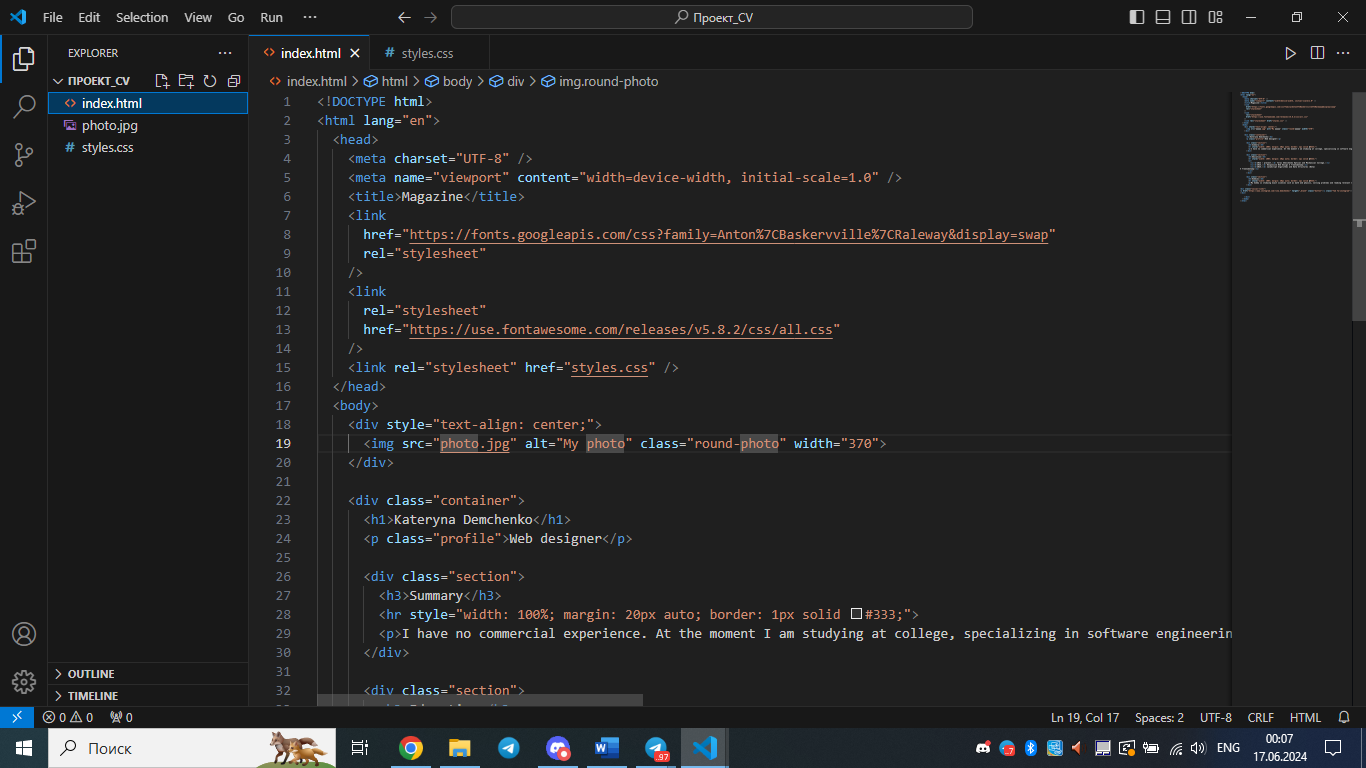


Рисунок 5.1 – Код

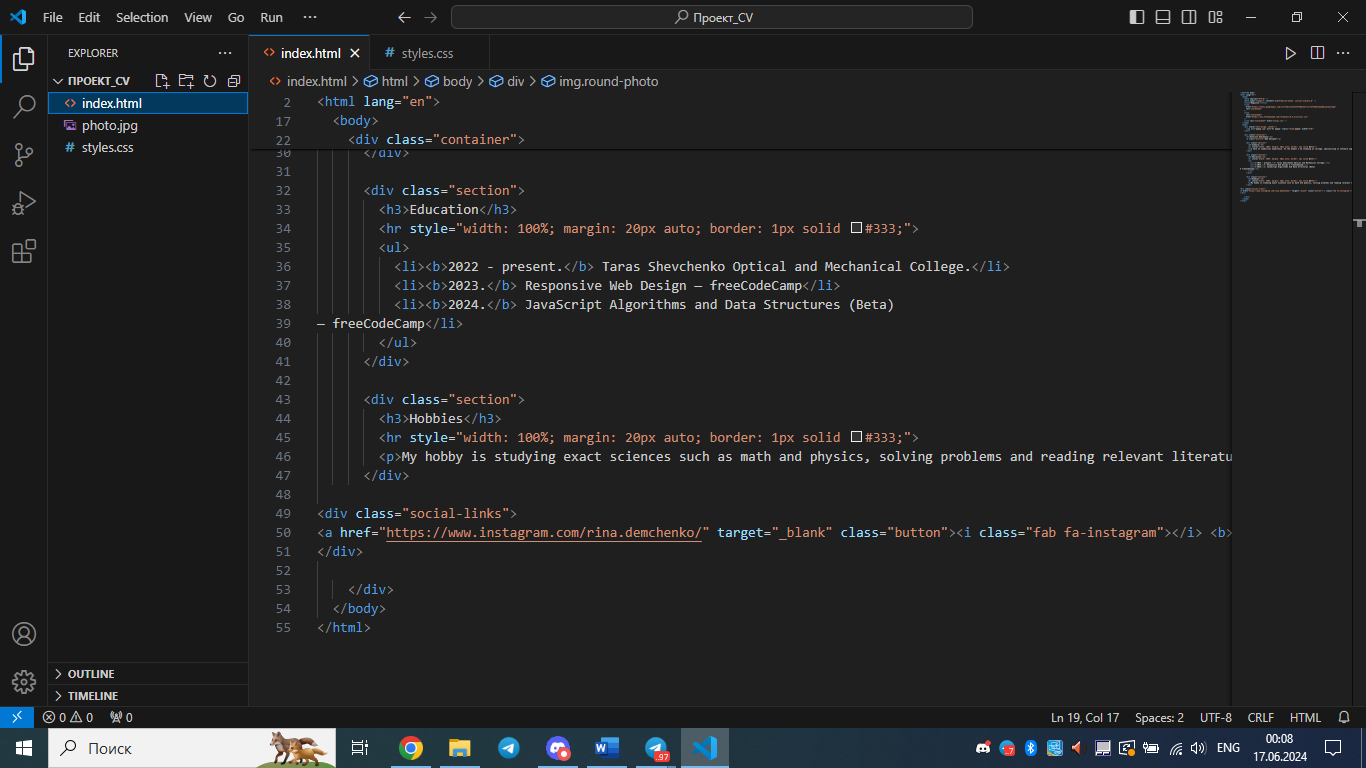


Рисунок 5.2 – Код

**Style.css:**

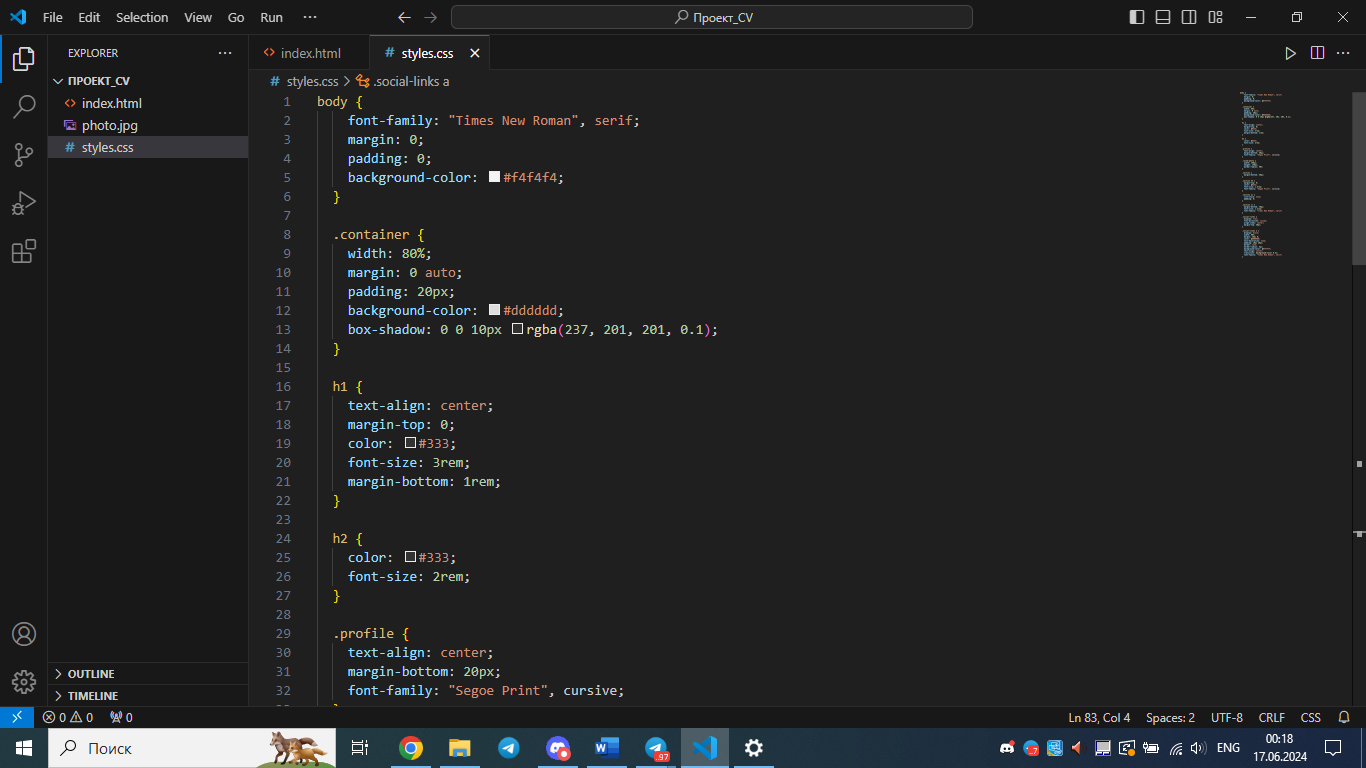


Рисунок 5.3 – Код

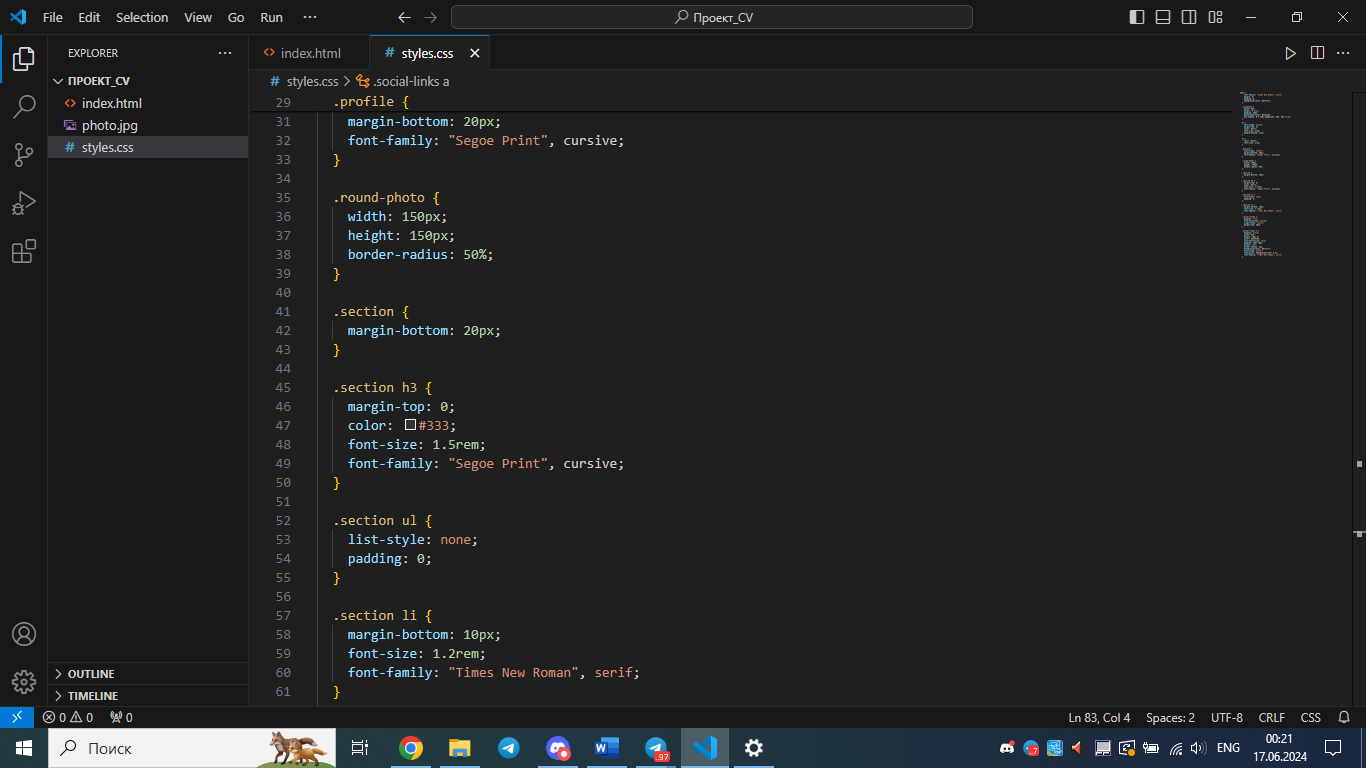


Рисунок 5.4 – Код

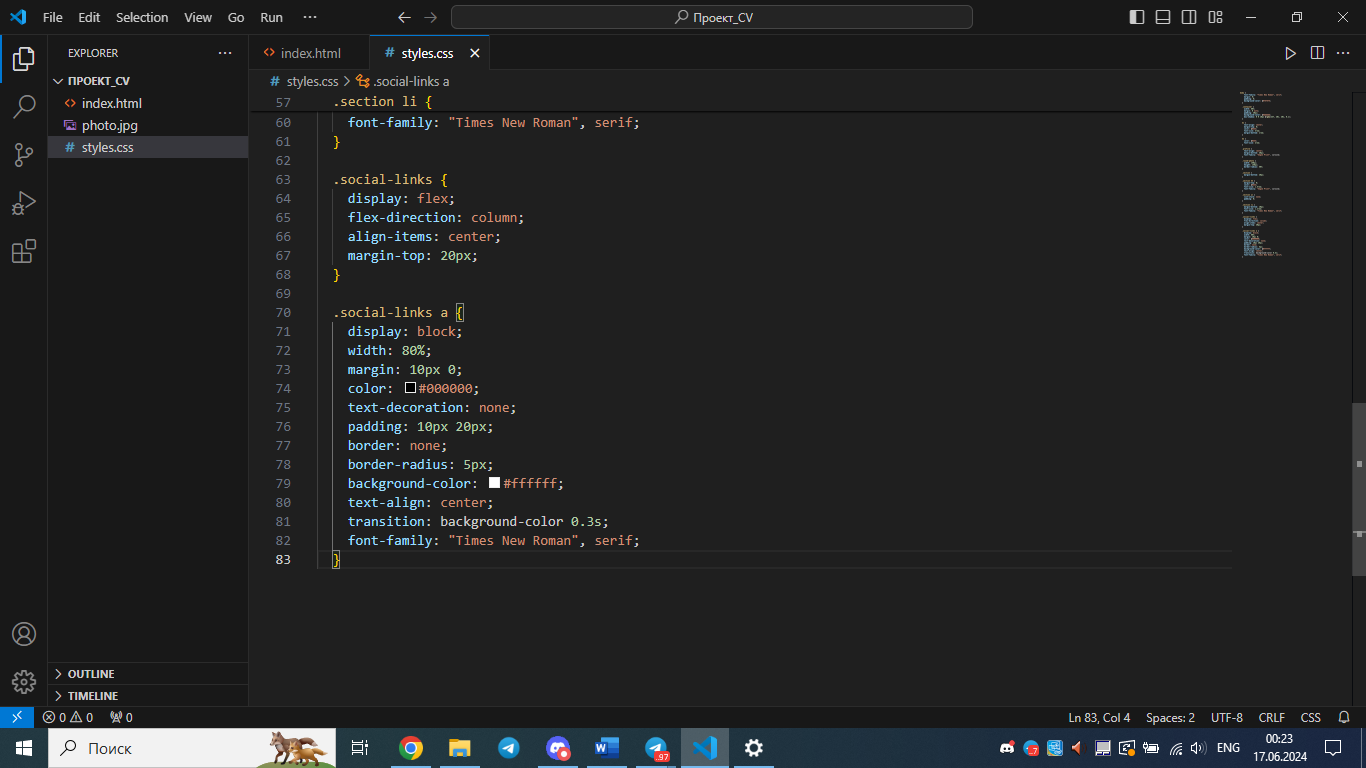


Рисунок 5.5 – Код

Рисунок 41 код

**Скріншот сайту:**

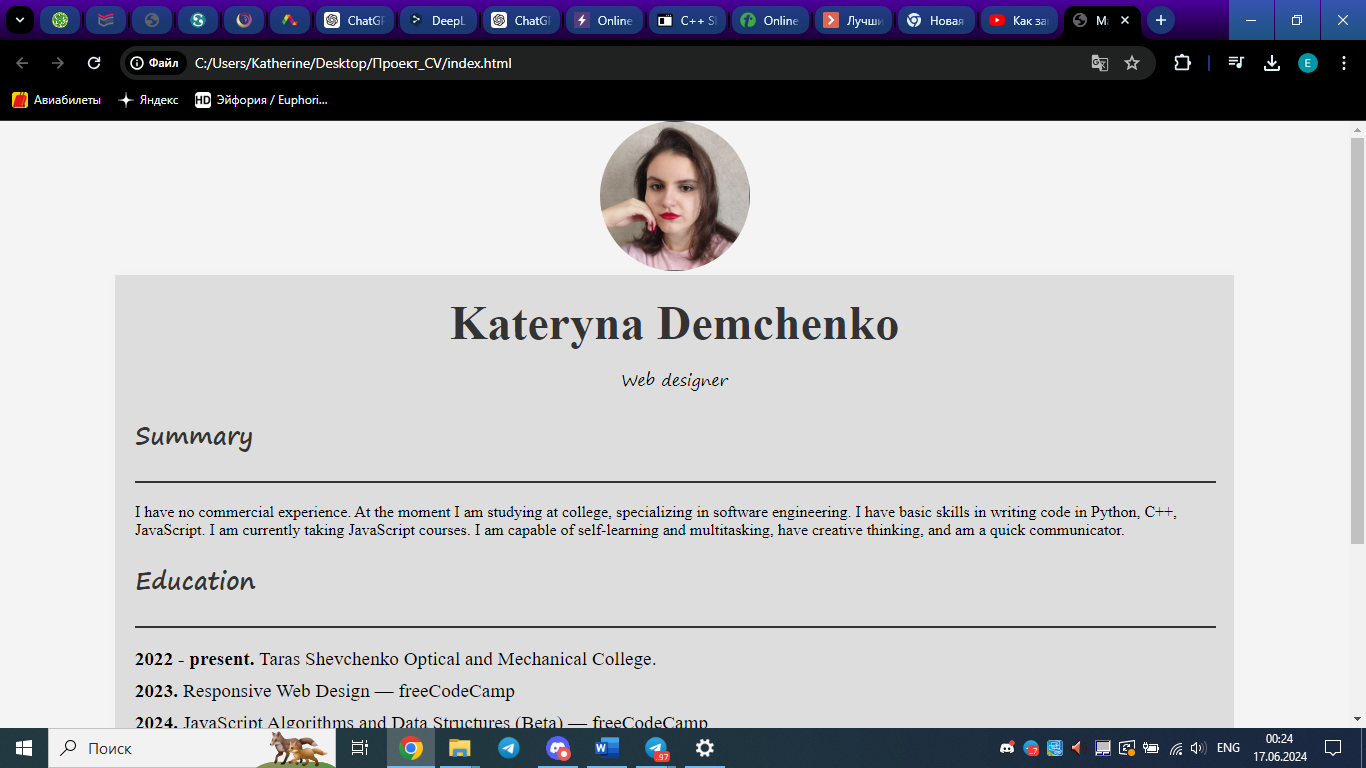


Рисунок 5.6 – Сайт

# ВИСНОВКИ

Я налаштувала параметри сторінок у документі за допомогою команд групи "Параметри сторінок" на стрічці "Макет". Встановила поля для верхнього, нижнього, лівого та правого країв сторінки. Обрала орієнтацію сторінки: книжкову для тексту та альбомну для сторінок з таблицями. Розділила документ на секції для різних форматувань і використала розриви для створення нових розділів.

Додала колонтитули для зазначення назви видання, розділів та номерів сторінок. Для вставки таких полів, як дата або номер сторінки, скористалася комбінацією клавіш Ctrl+F9. Створила структуру документа для полегшення форматування та нумерації розділів. Додала підписи до рисунків і таблиць, а також створила перехресні посилання для зручної навігації по документу.

Вставила виноски для пояснення термінів та забезпечила автоматичне оновлення всіх цих елементів, щоб документ завжди залишався актуальним та зрозумілим.

Комп'ютерне моделювання є важливим процесом створення інформаційних моделей за допомогою комп'ютерів, що має багато переваг, таких як повторюваність дій, висока наочність, безпечність, швидкість та зручність аналізу результатів. У MS Excel є потужні інструменти для розв’язання рівнянь і оптимізаційних задач, таких як метод "Підбір параметра" для моделей з одним невідомим і надбудова "Розв’язувач" для задач з декількома параметрами та обмеженнями. Також MS Excel надає можливості для статистичного аналізу даних, включаючи обчислення середнього, моди, медіани, частоти та стандартного відхилення, що допомагає проводити кореляційний аналіз та інші статистичні дослідження.

Я навчилася створювати порожню базу даних в Access, проектувати структуру таблиць і задавати типи даних для полів.

Я створила таблицю "Студенти" та налаштувала первинний ключ для поля "КодСтудента". Також я дізналася, як встановлювати обмеження для полів, наприклад, щоб оцінки не перевищували 12. Потім я навчилася вводити дані в таблицю та встановлювати зв’язки між таблицями для отримання реляційної бази даних. Закінчивши роботу, навчившись сортувати, шукати та фільтрувати дані для ефективного управління великою кількістю записів.

Алгоритм — це чітка послідовність дій для вирішення задачі. Він має властивості детермінованості, результативності та масовості. Алгоритми можуть бути описані текстуально, операторно або графічно. Основні структури алгоритмів: лінійні, з розгалуженням і циклічні.

C++ — універсальна мова програмування, розроблена Денісом Рітчі та Б'ярном Страуструпом. Вона потужна, гнучка та швидка, підтримує об'єктно-орієнтоване програмування. Програма на C++ складається з директив препроцесора, оголошень змінних, функцій та головної функції. C++ підтримує різні типи даних, такі як цілі числа, символи, логічні значення, дійсні числа, покажчики та масиви.

Для написання зрозумілих програм на C++ слід використовувати коментарі, відступи, оголошення типів даних та фігурні дужки. Арифметичні операції поділяються на унарні та бінарні, з певним порядком виконання та асоціативністю. C++ надає широкий спектр математичних функцій та констант.

Програмування обчислювальних процесів може бути лінійним, розгалуженим або циклічним. Лінійні процеси виконують дії послідовно, розгалужені використовують умовні оператори,

а циклічні повторюють дії за допомогою циклів "for", "while" та "do-while".

Я навчилася основам введення та виведення даних у C++ за допомогою бібліотеки iostream.

Для виведення використовується об'єкт cout, а для введення — cin.

Наприклад, можна вивести рядок на екран або запитати у користувача число і потім вивести його. Я також освоїла роботу з різними типами даних, такими як цілі числа, дробові числа та рядки, з використанням std::getline.

Ще ми розглядали структури даних для управління станом програми, зокрема масиви, вектори, списки, множини та мапи. Вони допомагають організовувати дані зручно та ефективно.

Я також вивчила, як працювати з файлами за допомогою бібліотеки fstream — запис і читання даних із текстових та бінарних файлів. Це важливо для збереження даних програми.

Крім того, ми порівнювали формати файлів JSON та CSV. JSON підходить для складних структур даних, тому я вибрала його для збереження стану гри. Він легко інтегрується з різними мовами програмування і зручний для читання.

Нарешті, я ознайомилася з мовою розмітки Markdown, яка спрощує створення документації для проектів.

HTML-документ починається з декларації <!DOCTYPE html> і містить теги <html>, <head>, та <body>. Основні елементи включають заголовки, абзаци, зображення та посилання. Зображення вставляються за допомогою тега <img>, а посилання створюються за допомогою тега <a>.

CSS допомагає розділити візуальне оформлення від структури сторінки, роблячи код більш гнучким та легким у підтримці. Важливі аспекти CSS включають селектори, властивості і значення, а також принципи спадковості і каскадування, які дозволяють ефективно організувати і застосовувати стилі до елементів веб-сторінки.

Ці знання допомагають створювати красиві та зручні веб-сторінки, забезпечуючи їх правильне відображення в різних браузерах.

# 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.

3. Лумпова Т.І. (2024). Безпека життєдіяльності при роботі з комп'ютером. Презентація. Отримано: <https://github.com/TetyanaLumpova/practical-training/tree/main/%D0%9D%D0%B0%D0%B2%D1%87%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%20%D0%BF%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0%20%D0%86%D0%9F%D0%97%202%20%D0%BA%D1%83%D1%80%D1%81/2024>  
4. Лумпова Т.І. (2024). Загальні поняття. Елементи мови С++ - константи, змінні, операції, перетворення типів. Word. Отримано: <https://github.com/TetyanaLumpova/practical-training/tree/main/%D0%9D%D0%B0%D0%B2%D1%87%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%20%D0%BF%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0%20%D0%86%D0%9F%D0%97%202%20%D0%BA%D1%83%D1%80%D1%81/2024>  
5. Лумпова Т.І.(2024). Арифметичні операції. Поняття вхідного та вихідного потоку, найпростіші математичні функції. Word. Отримано: <https://github.com/TetyanaLumpova/practical-training/tree/main/%D0%9D%D0%B0%D0%B2%D1%87%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%20%D0%BF%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0%20%D0%86%D0%9F%D0%97%202%20%D0%BA%D1%83%D1%80%D1%81/2024>  
6. Б'ярн Страуструп.(2021). A Tour of C++. Книга. Отримано: <https://elhacker.info/manuales/Lenguajes%20de%20Programacion/C++/A%20Tour%20of%20C++%20-%20Bjarne%20Stroustrup%20(Addison-Wesley,%202014)(193p).pdf>

7. Д.Н.Роббинс “Веб-дизайн для початківців: HTML, CSS, Javascript та веб-графіка”, 5 видання.

8. B.Frein “Responsive Web Design with HTML5 and CSS 3” 2nd edition.

9. R Nixon “Learning PHP, MySQL and Javascript” sixth edition.

10. MDN Web Docs - an open-source, collaborative project documenting Web platform technologies : <https://developer.mozilla.org/en-US/>

11. **Офіційна документація бібліотеки STL:** cppreference.com - STL

12.Документація бібліотеки nlohmann/json: [nlohmann/json on GitHub](https://github.com/nlohmann/json)

13.Книга по С++: <https://learn.saylor.org/course/view.php?id=65&gad_source=1&gclid=CjwKCAjw1K-zBhBIEiwAWeCOF1Fyn_6EDkKzxb3U7pqqptm3Lv4ebfJLKatBCCwIiOylpdFe8sVgdBoCarUQAvD_BwE>

14. **Документація Markdown**: https://docs.github.com/ru/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax