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: |
| студент 2-го курсу групи ІПЗ-23 |
| Капічніков Єгор Сергійович  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) | |  | |
| *Капічнікова Єгора Сергійовича* | | | | |
| (П.І.Б студента.) | | | | |
| Спеціальність | *121 Інженерія програмного забезпечення* | Група | | *ІПЗ-23* |

|  | **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) |
|  | |  |  |  | |  |  |
|  | |  |  |  | |  |  |

Зміст

[ВСТУП 7](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918692)

[1 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ 9](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918693)

[2 РОЗДІЛ "ІНФОРМАТИКА" 11](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918694)

[**2.1** **Розроблення засобами MS Word документа складної структури** 11](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918695)

[**2.2 Розв‘язок задач за допомогою засобів табличного редактора MS Excel** 13](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918696)

[**2.3 Виконання** **практичних завдань за допомогою засобів табличного редактора MS Access** 16](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918697)

[3 РОЗДІЛ "ПРОГРАМУВАННЯ С++" 18](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918698)

[**3.1 Теоретичні відомості** 18](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918699)

[3.1.1 Загальні поняття. Елементи мови С++ - константи, змінні, операції, перетворення типів 18](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918700)

[3.1.2 Арифметичні операції 20](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918701)

[3.1.3 Поняття вхідного та вихідного потоку, найпростіші математичні функції 22](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918702)

[3.1.4 Програмування лінійних та розгалужених обчислювальних процесів 24](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918703)

[3.1.5 Програмування циклічних обчислювальних процесів 27](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918704)

[**3.2 Практичне завдання** 30](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918705)

[3.2.1 Завдання 30](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918706)

[3.2.2 Постановка задачі 30](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918707)

[3.2.3 Код програми 30](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918708)

[3.2.4 Скріншот виконання програми 32](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918709)

[**3.3 Виконання тестів** 32](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918710)

[4 РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++" 33](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918711)

[**4.1 Тема 1. Введення та виведення у базових консольних програмах** 33](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918712)

[**4.2 Тема 2. Структури даних для управління станом програми** 36](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918713)

[**4.3 Тема 3. Робота з файлами: запис та читання** 37](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918714)

[**4.4 Тема 4. Формати файлів та їх обробка** 40](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918715)

[**4..5 Тема 5. Системи контролю версій (VCS)** 44](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918716)

[**4.6 Тема 6. Мова Markdown для опису проекті** 48](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918717)

[**4.7 Розроблення ігрової програми** 51](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918718)

[4.7.1 Код програми 51](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918719)

[4.7.2 Скріншоти виконання програми 54](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918720)

[5 РОЗДІЛ ”ВЕБ-ДИЗАЙН ТА HTML” 56](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918721)

[**5.1 Мова розмітки HTML та структура Веб-сторінки** 56](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918722)

[**5.2 Розмітка тексту за допомогою HTML. Зображення та посилання** 59](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918723)

[**5.3 Каскадні таблиці стилів CSS та їх структура** 63](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918724)

[**5.4 Спадковості і каскадування за допомогою CSS** 64](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918725)

[**5.5 Практична частина** 67](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918726)

[5.5.1 Постановки задачі 67](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918727)

[5.5.2 HTML документ 68](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918728)

[ВИСНОВКИ 69](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918729)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ 72](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918730)

[ДОДАТКИ 74](file:///C:\Users\kapic\Downloads\ZV23_Kasatkina.docx#_Toc167918731)

ОМФК.0121 НП 20

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

# ВСТУП

Автор: Капічніков Єгор Сергійович

Дата: 17.05.2024 – 13.06.2024

Мета:

Перелік вивчених тем:

1. Засобами MS Word розробити документ складної структури та зберегти у різних форматах
2. Розв’язування рівнянь
3. Розв’язування оптимізаційних задач
4. Основи статистичного аналізу даних. Ряди даних. Кореляційний аналіз даних
5. Обчислення основних статистичних характеристик вибірки
6. Візуалізація рядів і трендів даних
7. Розв‘язок задач за допомогою засобів табличного редактора MS Excel
8. Проектування бази даних та створення бази даних MS ACCESS
9. Основні поняття реляційних баз даних
10. Введення, коригування, вилучення інформації з бази даних MS ACCESS
11. Вибірка, пошук, фільтрація даних в MS ACCESS
12. Виконання практичних завдань за допомогою засобів табличного редактора MS Access
13. Алгоритмізація обчислювальних процесів. Розробка постановки задачі, визначення вхідних та вихідних даних, оформлення блок-схем.
14. Типи даних в мові С/С++, внутрішнє представлення даних, операції перетворення типів даних.
15. Оператор присвоювання. Операції введення- виведення даних
16. Програмування лінійних та розгалужених обчислювальних процесів
17. Програмування циклічних обчислювальних процесів
18. Складання програм
19. Виконання тестів
20. Введення та виведення у базових консольних програмах
21. Структури даних для управління станом програми
22. Робота з файлами: запис та читання
23. Формати файлів та їх обробка
24. Системи контролю версій (VCS)
25. Мова Markdown для опису проекті
26. Розроблення ігрової програми
27. Мова розмітки HTML та структура Веб-сторінки
28. Розмітка тексту за допомогою HTML. Зображення та посилання
29. Каскадні таблиці стилів CSS та їх структура
30. Спадковості і каскадування за допомогою CSS
31. Розробка та оформлення Постановки задачі
32. Розробка HTML документу
33. Систематизація матеріалів, оформлення звіту з навчальної практики

# 1 РОЗДІЛ «БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ»

## ***Важливість безпеки при роботі з комп’ютером***

Безпека при роботі з комп’ютером має велике значення з кількох причин. По-перше, недбале ставлення до безпеки може призвести до фізичних травм. Наприклад, погане розташування комп’ютерного столу чи некомфортне крісло можуть спричинити болі у спині або шиї. Крім того, тривала робота за комп’ютером може спричинити очні напруження та інші проблеми зі здоров’ям.

По-друге, безпека даних є іншим важливим аспектом. Крадіжка або втрата цінної інформації може призвести до серйозних проблем, які можуть вплинути на бізнес або особисте життя користувача. Заходи безпеки, такі як використання паролів і шифрування даних, можуть допомогти уникнути таких неприємностей.

## **Основні аспекти безпеки при роботі з комп’ютером**

### ***Віртуальна безпека***

Використання комп’ютера пов’язане з ризиком вірусів, шкідливих програм та кібератак. Щоб забезпечити віртуальну безпеку, потрібно:

#### **Установити антивірусне програмне забезпечення**

Встановлення надійного антивірусного програмного забезпечення допоможе захистити комп’ютер від шкідливих програм та вірусів. Регулярно оновлюйте антивірусну базу даних для забезпечення найвищого рівня захисту.

#### **Застосовувати оновлення системи**

Регулярно оновлюйте операційну систему та програмне забезпечення, оскільки це дозволяє усунути вразливості, які можуть бути використані хакерами для вторгнення в систему.

#### **Бережно поводьтесь з електронною поштою та посиланнями**

Не відкривайте незнайомі або сумнівні електронні листи та посилання. Такі повідомлення можуть містити шкідливі файли або посилання на фішингові сайти, призначені для викрадення особистих даних.

### ***Фізична безпека***

Фізична безпека також є важливим аспектом при роботі з комп’ютером. Ось декілька порад щодо фізичної безпеки:

#### **Вірне розташування робочого місця**

Переконайтеся, що ваше робоче місце обладнане комфортним столом та стільцем. Забезпечення належної підтримки для спини та шиї може допомогти уникнути болю та напруження.

#### **Забезпечення достатнього освітлення**

Маючи достатнє освітлення на робочому місці, ви зменшуєте напруження очей і запобігаєте втомі під час роботи за комп’ютером.

#### **Регулярні перерви**

Не забувайте робити короткі перерви під час тривалої роботи за комп’ютером. Піднімайтесь, розтягуйтеся та робіть прості фізичні вправи для підтримки кровообігу та запобігання м’язовим напругам.

# 2 РОЗДІЛ "ІНФОРМАТИКА"

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

Щоб розробляти документ складної структури, достатньо знати можливості функціоналу MS Word і користуватися корисними операціями.

**1. Створення та налаштування документа**

Першим кроком є створення нового документа. У MS Word це можна зробити через меню "Файл" -> "Створити". Після цього слід налаштувати основні параметри документа:

* *Поля***:** Встановлення розмірів полів через "Розмітка сторінки" -> "Поля". Це може бути стандартний розмір або індивідуальні налаштування.
* *Розмір сторінки***:** Вибір формату сторінки, наприклад, A4 або інший стандарт, через "Розмітка сторінки" -> "Розмір".

**2. Стилі та заголовки**

Для забезпечення чіткої структури документа важливо використовувати стилі та заголовки:

* *Стилі***:** MS Word пропонує набір стандартних стилів для форматування тексту, які можна застосовувати для заголовків, підзаголовків, основного тексту та інших елементів. Це дозволяє легко змінювати форматування всього документа.
* *Заголовки***:** Для створення логічної структури документа використовуються заголовки різних рівнів (Heading 1, Heading 2, і т.д.). Їх можна знайти на вкладці "Головна".

**3. Зміст**

Для зручності навігації та організації документа слід створити зміст:

* Вставка змісту**:** Використовується команда "Посилання" -> "Зміст" -> "Автоматичний зміст".
* *Оновлення змісту***:** Після внесення змін у документ можна оновити зміст за допомогою кнопки "Оновити таблицю".

**4. Робота з таблицями**

Для організації та подання даних можна використовувати таблиці:

* Вставка таблиці**:** Використовується команда "Вставка" -> "Таблиця" -> вибір потрібної кількості рядків і стовпців.
* *Форматування таблиці***:** Зміна стилю таблиці, додавання кольору фону, меж та інших елементів через вкладку "Конструктор таблиць".

**5. Графіки та діаграми**

Для візуалізації даних використовуються графіки та діаграми:

* *Вставка діаграми***:** Команда "Вставка" -> "Діаграма".
* *Налаштування діаграми***:** Вибір типу діаграми (графік, гістограма, кругова діаграма тощо), додавання підписів, легенд та інших елементів.

**6. Додавання ілюстрацій та зображень**

Для підвищення візуальної привабливості документа можна додавати зображення:

* *Вставка зображень***:** "Вставка" -> "Зображення" -> вибір файлу з комп'ютера або онлайн-джерел.
* *Редагування зображень***:** Зміна розміру, обрізка, додавання рамок та інших ефектів через вкладку "Формат".

**7. Посилання та гіперпосилання**

Для полегшення навігації по документу можна використовувати гіперпосилання та перехресні посилання:

* *Гіперпосилання***:** "Вставка" -> "Гіперпосилання" -> вибір місця в документі або зовнішнього ресурсу.
* *Перехресні посилання***:** "Посилання" -> "Перехресне посилання" для створення посилань на інші частини документа, такі як таблиці, графіки або заголовки.

**8. Колонтитули та номери сторінок**

Колонтитули та номери сторінок допомагають у структуризації документа:

* *Вставка колонтитулів***:** "Вставка" -> "Колонтитул" -> вибір стандартного або настроюваного колонтитула.
* *Нумерація сторінок***:** "Вставка" -> "Номер сторінки" -> вибір місця розташування номера (верхній або нижній колонтитул).

**9. Рецензування**

У MS Word також присутні функції коментування, відстеження змін та порівняння документів для спільної та більш зручної роботи над текстом.

**2.2 Розв'язок задач за допомогою засобів табличного редактора MS EXCEL**

**1.1 Розв’язання рівнянь**

 У багатьох задачах певний результат є відомим, а от значення параметрів, за яких цей результат досягається, — ні. Як приклад можна навести задачу, у якій потрібно визначити, через скільки годин скисне молоко або за якого обсягу випуску продукції фірма отримає прибуток у 1 000 000 грн. У математиці клас таких задач є найширшим. Це, зокрема, задачі на розв'язання алгебраїчних рівнянь та нерівностей або на пошук екстремумів.

    У всіх подібних задачах використовується поняття цільової функції — вона має досягти певного значення або оптимізуватися (мінімізуватися чи максимізуватися). В електронній книзі формулу обчислення цільової функції записують у певну клітинку, яку також називають цільовою. Цільова функція залежить від параметрів (часто — від одного параметра), значення яких зберігаються в інших клітинках електронної таблиці.

    Власне кажучи, задача полягає у підборі таких значень параметрів, за яких у цільовій клітинці буде отримано бажаний результат. У табличному процесорі Excel є спеціальні засоби, які автоматично підбирають потрібні значення у клітинках параметрів. Вони називаютьсяПідбір параметра та Пошук розв'язків. Перший із них дозволяє отримати в цільовій клітинці певне значення, а другий — оптимізувати значення цільової функції.

    Засіб **Підбір параметра** застосовують наступним чином:

    1. В одну з клітинок електронної таблиці слід увести формулу цільової функції. Це буде цільова клітинка.

    2. Далі необхідно виконати команду Сервіс - Підбір параметра та заповнити поля у вікні, що відкриється:

* поле Установити у клітинці має містити адресу цільової клітинки;
* у поле Значення слід ввести значення, якого має набути цільова функція
* у поле Змінюючи значення клітинки слід увести адресу клітинки-параметра.

    3. На завершення потрібно клацнути кнопку ОК. У цільовій клітинці буде відображено значення, якого має набувати цільова функція, а в клітинці параметра — шукане значення параметра.

**1.2 Розв’язання оптимізаційних задач**

 При розв’язуванні широкого кола задач (зокрема, в галузі еконо­міки) потрібно знайти оптимальний (найкращий) розв’язок конкретно­го завдання при виконанні деяких заданих умов. Як визначити опти­мальні витрати на рекламу продукції? Як доставити продукцію до споживачів з мінімальними витратами на перевезення? Задачі пошу­ку оптимального розв’язку називаються задачами оптимізації. Кри­терієм оптимальності в задачах є різні параметри: максимальна кіль­кість продукції, максимальний прибуток підприємства, мінімальні ви­трати виробництва тощо.  
    **Цільова функція** — це аналітична залежність між критерієм опти­мальності і параметрами, що підлягають оптимізації, з вказівкою напряму екстремуму. Для цільової функції завжди і обов’язково вказується вид екстремуму: f(x) ® max(f(x) ® min).  
    Фактично, цільова функція — це числове значення, яке показує, наскільки оптимальним є знайдене рішення.

    Пошук оптимального рішення завжди проводиться з урахуванням певних обмежень: підприємство має обмежений набір ресурсів, тран­спортні засоби мають обмежену вантажопідйомність і швидкість, інвестор планує вкласти в розвиток підприємства кошти в межах певної суми. Обмеження — це умови, що накладаються па параметри, від яких зале­жить значення цільової функції.

**Оптимізаційне моделювання** — це пошук таких значень параметрів, при яких цільова функція досягає максимального або мінімального зна­чення при заданих обмеженнях. Для пошуку оптимального рішення зруч­но використовувати надбудову Пошук розв'язування.  
    Першим кроком при розв’язуванні задачі оптимізації є побудова ма­тематичної моделі задачі, яка включає:  
    • перелік невідомих величин, значення яких потрібно знайти;  
    • завдання цільової функції;  
    • визначення критерію оптимізації цільової функції;  
    • завдання системи обмежень у формі лінійних рівнянь і нерів­ностей.

**1.3 Основи статистичного аналізу даних. Ряди даних. Кореляційний аналіз даних**

В багатьох дослідженнях для аналізу даних, установлення певних закономірностей, формулювання висновку, надання рекомендацій, прогнозування тощо потрібно використати багато даних. Методи отримання, опрацювання й аналізу даних, які характеризують масові явища, вивчає наука статистика (лат status —стан).

**Вибірка**

Вибірка – це частина групи, яку ви досліджуєте. Наприклад, якщо ви досліджуєте дохід населення, то ваша група може бути складена з усього населення країни. Вибірка ж – це лише частина цього населення, яку ви обрали для дослідження.

**Параметри та статистики**

У статистиці розрізняють два типи характеристик: параметри та статистики. Параметр – це характеристика, яка описує групу в цілому. Наприклад, середнє значення доходу населення країни є параметром. Статистика – це характеристика, яка описує вибірку. Наприклад, середнє значення доходу вибірки є статистикою.

**Центральна тенденція**

Центральна тенденція – це показник, який описує, де знаходиться “центр” даних вибірки. Це може бути середнє значення, медіана або мода.

**Розкид**

Розкид – це показник, який описує, наскільки далеко розташовані дані від центральної тенденції. Розкид може бути виміряний дисперсією або стандартним відхиленням.

**Обчислення основних статистичних характеристик вибірки**

**Середнє значення**

Середнє значення – це сума всіх значень вибірки, поділена на її розмірність. Це дає загальний “центр” даних у вибірці.

**Медіана**

Медіана – це значення, яке знаходиться в середині вибірки, коли всі значення вибірки впорядковані за величиною. Якщо розмірність вибірки парна, то медіана обчислюється як середнє значення двох центральних значень.

**Мода**

Мода – це значення, яке найчастіше зустрічається в вибірці.

**Дисперсія**

Дисперсія – це показник, який описує, наскільки розкидані дані від середнього значення. Для обчислення дисперсії необхідно взяти кожне значення вибірки, відняти від нього середнє значення, піднести результат до квадрату, підсумувати всі такі значення та розділити їх на розмірність вибірки мінус один.

**Стандартне відхилення**

Стандартне відхилення – це квадратний корінь з дисперсії. Воно показує, наскільки дані розкидані навколо середнього значення.

Формула для обчислення стандартного відхилення:

**Коефіцієнт варіації**

Коефіцієнт варіації – це показник, який вказує на ступінь розкиду даних відносно їх середнього значення. Чим вище коефіцієнт варіації, тим більше розкид даних.

**1.4** **Обчислення основних статистичних характеристик вибірки**

**Обсяг вибірки**— кількість елементів у вибірці.

**Розмах вибірки** — різниця між максимальним і мінімальним значеннями елементів вибірки.

**Середнє арифметичне ряду чисел** — це частка від ділення суми цих чисел на їх кількість (обсяг вибірки).

**Мода ряду чисел** — число, яке найчастіше зустрічається в ряді даних. Якщо дані у вибірці не повторюються, мода не обчислюється. Якщо в ряді даних є числа, які зустрічаються однакову кількість разів, мода буде мати кілька значень.

**Медіана** впорядкованого ряду чисел з непарним числом чле­нів — число, яке виявиться посередині. Медіана впорядкованого ряду чисел із парним числом членів — середнє арифметичне двох чисел, за­писаних посередині. Отже, половина значень вибірки менше за медіану, а половина більше.

**Частота**— число повторень значень вибірки в заданих інтер­валах.

**Відносна частота** — це відношення частоти до загальної кіль­кості даних у вибірці ( частота/100 ).

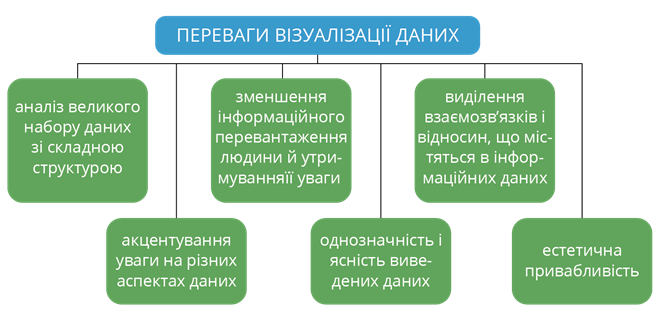
**Стандартне відхилення** (ще його називають середньоквадратичним відхиленням, СКВ) — показник розсіювання статистичних даних відносно середнього значення вибірки.

**1.5 Візуалізація рядів і трендів даних**

**Візуалізація** — це подання відомостей, даних, фактів у візуальній формі.

У наукових експериментах і статистичному аналізі графіки й діаграми не лише наглядно відображають значення, мають змістове навантаження щодо теми дослідження.

Завдяки візуалізації відразу видно загальну картину результатів дослідження, велику кількість даних на обмеженому проміжку; акцентується увага на деяких елементах ряду даних; порівнюються фрагменти даних, демонструється тенденція зміни досліджуваних властивостей тощо.



Візуалізація має передувати інфографіці. Важливою рисою візуалізованих даних є їхня переконливість, тому дуже важливо уникати викривлення інформаційних даних у процесі візуалізації. Зокрема, у хорошій візуалізації наочність загальної картини не заважає сприйняттю деталей.

**Інфографіка** (від латин. informatio — інформування, роз’яснення, і грец. γραφικός — письмовий) — це візуальне відображення інформації, статистичних даних для простої і наочної демонстрації тенденцій, співвідношень, а також зацікавлення в предметі дослідження.

Для наочного подання й аналізу рядів даних використовують діаграми. На діаграмах числові дані подаються геометричними фігурами: точками, відрізками, прямокутниками, секторами круга та ін. Розміри цих фігур пропорційні числовим даним, за якими побудовано діаграму. Це дає можливість візуально оцінити співвідношення між числами в одному або в кількох рядах даних.  
Прості засоби візуалізації включено до сучасних табличних процесорів. Вони не охоплюють усього різноманіття технік, але для простих  задач й оперативного подання даних цілком підходять. Але для побудови якісних візуалізацій краще використовувати спеціалізовані інструменти, тим більше, що серед них є вільні й досить прості у використанні.  
Ряди даних можна використовувати для прогнозування певного явища, процесу.

**Тренд** (англ . trend — тенденція) — це основна тенденція змінення певного процесу .  
**Лінія тренду** — це лінія, уздовж якої розташовуються на діаграмі точки, що зображають дані з певного ряду даних.

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

# Визначення та вирішення задач за допомогою надбудови «Пошук розв’язання»

"Розв’язувач" – це надбудова Microsoft Excel, яка використовується для аналіз "what-if". За її допомогою можна знайти оптимальне (максимальне або мінімальне) значення для формула в одній клітинці (так званій клітинці цільової функції), що обмежується значеннями формул в інших клітинках аркуша. Надбудова "Розв’язувач" працює із групою клітинок (які називаються клітинками змінних рішення або просто клітинками змінних), що використовуються для обчислення формул у цільових функціях і клітинках обмежень. Надбудова регулює значення у клітинках змінних відповідно до меж у клітинках обмежень і виводить потрібний результат у клітинці цільової функції.

Простіше кажучи, ви можете використовувати Розв’язувач, щоб визначати максимальне або мінімальне значення однієї клітинки, змінюючи інші клітинки. Наприклад, можна змінити прогнозований бюджет на рекламу та побачити, яке це вплине на прогнозований прибуток.

У стандартному наборі команд MS Excel команда запуску надбудови **Розв’язувач**

відсутня, її необхідно встановити на стрічку **ДАНІ.** Для цього виконаємо такі

дії.

1. Перейдемо в меню **ФАЙЛ — Параметри**. Виберемо ***Надбудови***.

2. У вікні, що відкрилося, виберемо ***Пакет аналізу*** й активуємо кнопку Перейти — відкриється вікно ***Надбудови***.

3. Поставимо галочку біля **Розв’язувач**. Підтвердимо дії кнопкою ОК.

На стрічці **ДАНІ** з’явилася команда **Розв’язувач**. Скористаємося математичною

моделлю для створення таблиці в середовищі MS Excel (рис. 2.4).

4. У потрібних клітинках введемо значення для виробів а, b, с, можна ввести і по 1.

5. Після запуску команди **Розв’язувач** відкриється вікно ***Параметри розв’язувача***, у якому:

1) у поле Встановіть цільову клітинку введіть потрібну клітинку з підрахунком загального доходу і вкажемо, що функція прямує до максимуму;

2) у поле Змінні клітинки введемо клітинки з невідомими.

6. Для введення в поле Обмеження скористаємося кнопкою Додати,

щоб увести обмеження на значення невідомих — обмеження вводять у вікні

Додати обмеження.

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

Microsoft Access є ефективним інструментом для створення та керування базами даних. Є декілька етапів роботи, щоб виконати завдання у СКБД MS Access.

**1. Створення бази даних**

***Створення нової бази даних:*** Відкрийте MS Access, виберіть "Файл" -> "Створити" -> "Нова база даних". Введіть ім'я файлу та виберіть місце збереження.

**2. Створення та налаштування таблиць**

***Створення таблиці:*** Виберіть "Створити" -> "Таблиця". MS Access автоматично створить нову таблицю в режимі конструктора.

**3. Створення запитів**

Запит полегшує роботу, якщо необхідно переглянути, додати, змінити або видалити дані в базі даних Access. Ось ще кілька підстав для використання запитів:

* швидкий пошук даних за допомогою фільтрації за певними критеріями (умовами);
* обчислення або підсумовування даних;
* автоматизація завдань керування даними, таких як періодичній перегляд набору даних.

**Створення запиту:** Виберіть "Створити" -> "Майстер запитів" або "Конструктор запитів". У режимі конструктора запитів ви можете створити новий запит вручну.

**Додавання таблиць:** Додайте до запиту необхідні таблиці або інші запити.

**4. Створення форм**

**Створення форми:** Виберіть "Створити" -> "Форма" для автоматичного створення форми або "Конструктор форм" для створення форми з нуля.

**Додавання елементів:** Додайте елементи керування, такі як текстові поля, списки, кнопки, за допомогою стрічки інструментів "Конструктор".

**5. Створення звітів**

**Створення звіту:** Виберіть "Створити" -> "Звіт" для автоматичного створення звіту або "Конструктор звітів" для ручного створення.

**6. Макроси**

Макрос – це набір команд та інструкцій, об’єднаних в одну команду для автоматичного виконання завдання.

**Створення макросу:** Виберіть "Створити" -> "Макрос". У режимі конструктора макросів додайте необхідні дії.

**Налаштування дій:** Визначте дії, які повинен виконувати макрос (відкриття форми, запуск запиту, повідомлення тощо).

**Збереження та запуск макросу:** Збережіть макрос і запустіть його для перевірки роботи.

**3 РОЗДІЛ «ПРОГРАМУВАННЯ С++»**

**3.1 Теоретичні відомості**

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

C++ є однією з найпопулярніших мов програмування, яка широко використовується для створення різноманітних програмних додатків. Основними елементами мови C++ є константи, змінні, операції та перетворення типів.

**Константи**

Константи - це значення, які не змінюються під час виконання програми. В C++ є кілька способів визначення констант:

* **Константи з ключовим словом const:**

*const int MAX\_VALUE = 100;*

* **Макроси з директивою препроцесора #define:**

*#define PI 3.14159*

*Директиву* **#define** не можна використовувати для символьних констант.

* **Перечислювальні типи (enum):**

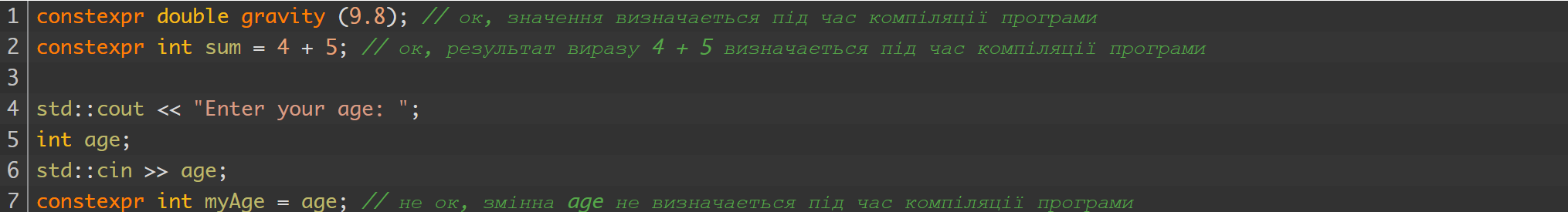
*enum Colors { RED, GREEN, BLUE };*

У мові C++ є два види констант:

   1. Константи часу виконання. Їх значення визначаються тільки під час виконання програми. Змінні типу usersAge і myValue, наведені вище, є константами часу виконання, оскільки компілятор не може визначити їх значення під час компіляції. usersAge залежить від користувацького вводу (який можна отримати тільки під час виконання програми), а myValue залежить від значення, переданого в функцію (яке стане відомим також під час виконання програми).

   2. Константи часу компіляції. Їх значення визначаються під час компіляції програми. Наприклад, змінна зі значенням сили тяжіння на Землі є константою часу компіляції, так як ми її визначаємо під час написання програми (до початку її виконання).

У більшості випадків не має значення який тип константи ви використовуєте: часу виконання або часу компіляції. Однак, все ж є декілька ситуацій, коли мова C++ може вимагати константу часу компіляції замість константи часу виконання (наприклад, при визначенні довжини масиву фіксованого розміру — ми розглянемо це трохи пізніше). Оскільки є 2 типи констант, то компілятору потрібно постійно відслідковувати, до якого з них відноситься певна змінна. Щоб спростити це завдання, в C++11 додали специфікатор constexpr, який повідомляє компілятору, що поточна змінна є константою часу компіляції:



***Змінні***

Змінна є ідентифікатором, що позначає деяку область пам’яті, в якій зберігається значення змінної. Це значення може змінюватися під час виконання програми. У програмуванні змінна, як і в математиці, може мати назву, що складається з однієї латинської літери, але може складатися з декількох символів, цілого слова або декількох слів. Для визначення змінних в C++ використовується наступний синтаксис:

**Приклад оголошення змінних**

int a; // оголошення змінної a цілого типу.  
float b; // оголошення змінної b типу даних із плаваючою комою.  
double c=14.2; // ініціалізація змінної типу double.  
char d = 's'; // ініціалізація змінної типу char.  
bool k = true; // ініціалізація логічної змінної k.

**Операції**

Для здійснення маніпуляцій з даними мова C++ застосовує широкий набір операцій (див. табл. 3.4), що виконують формування і, відповідно, подальше обчислення виразів. Вирази містять одну або декілька операцій, об’єкти яких називають операндами. Операції являють собою деяку дію, що виконується над одним (унарна) або декількома (бінарна, триарна) операндами , і мають позначення (наприклад, операція перевірки на рівність — позначення «==», операція обчислення залишку від ділення цілих чисел — позначення «%» тощо).

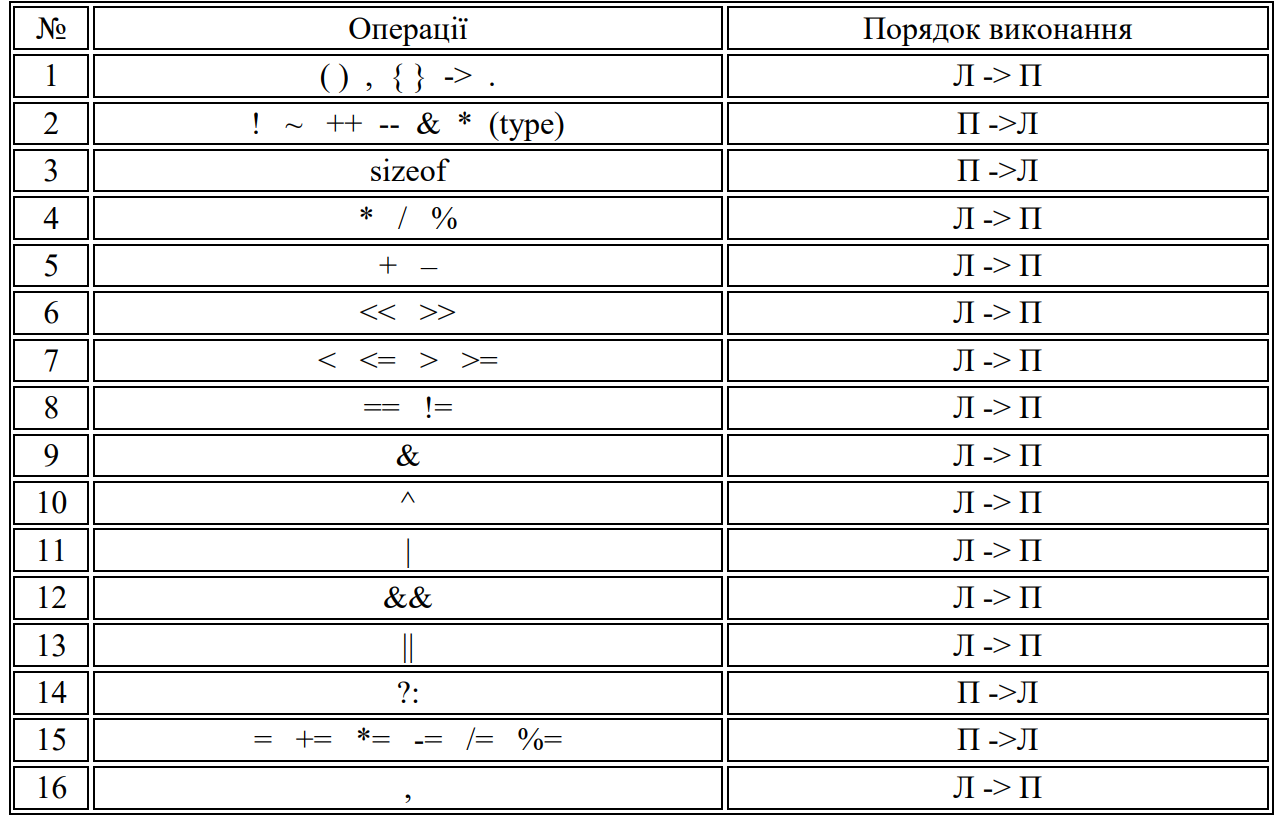
Операції поділяються на:

• унарні або одномісні — &, \*, -, +, ~, !, ++, --, sizeof;

• бінарні або двомісні — +, -, \*, /, %, <>, &, :, ^, , <=, ==, >=, !=, &&, ||, =,\*=, /=, %=, +=, -=, <<=, >>=, &=, |=, ^=, ., ->, ,, ( ), [ ];

• умовну триарну або тримісну операцію — ?: .

**Основні операції мови C++**

****

**Основні арифметичні операції**

**Додавання**

Операція додавання виконується для додавання декількох чисел.

int a = 2;

int b = 4;

int sum = a + b; // sum дорівнює 6

**Віднімання**

Операція віднімання використовується для віднімання декількох чисел

int a = 2;

int b = 4;

int difference = b-a; // difference дорівнює 2

**Множення**

Операція множення використовується для множення декількох чисел.

int a = 4;

int b = 2;

int product = a \* b; // product дорівнює 8

**Ділення**

Операція ділення використовується для ділення одного числа на інше. Варто зазначити, що ділення цілих чисел призводить до цілочисельного результату.

int a = 4;

int b = 2;

int quotient = a / b; // quotient дорівнює 2 (цілочисельний результат)

**Залишок від ділення (%)**

Операція залишку від ділення (модуль) використовується для отримання залишку після ділення одного числа на інше.

int a = 4;

int b = 3;

int remainder = a % b; // remainder дорівнює 1

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

У мові програмування C++, поняття вхідного та вихідного потоку використовуються для операцій введення та виведення даних. Давайте розглянемо їхнє використання разом із прикладами найпростіших математичних функцій.

**Вхідний потік (Input Stream)**

Вхідний потік в C++ представлений об'єктом класу std::istream і використовується для зчитування даних з консолі або інших вхідних джерел. Найбільш використовуваний об'єкт вхідного потоку - std::cin.

#include <iostream>

int main() {

int num;

std::cin >> num;

std::cout << "Введене число: " << num << std::endl;

return 0;

}

**Вихідний потік (Output Stream)**

Вихідний потік в C++ представлений об'єктом класу std::ostream і використовується для виведення даних в консоль або інші вихідні місця. Найбільш використовуваний об'єкт вихідного потоку - std::cout.

#include <iostream>

int main() {

int a = 5, b = 3;

double result;

result = a + b;

std::cout << "Результат додавання: " << result << std::endl;

return 0;

}

**Найпростіші математичні функції в C++**

Для виконання математичних операцій у мові C++ доступні стандартні математичні функції, які мають відповідні вбудовані функції або оператори:

1. **Математичні функції**:
   * Натуральний логарифм: std::log(x)
   * Експонента: std::exp(x)
   * Тригонометричні функції (синус, косинус, тангенс): std::sin(x), std::cos(x), std::tan(x)

### **3.1.4 Програмування лінійних та розгалужених обчислювальних процесів**

**Лінійний обчислювальний процес**

Лінійний обчислювальний процес означає послідовність дій, які виконуються одна за одною без розгалужень або повторень. Програма виконується від початку до кінця, кожна команда виконується після попередньої.

Наприклад, проста програма на C++, яка обчислює суму перших чотирьох чисел:

#include <iostream>

int main() {

int sum = 0;

sum += 1;

sum += 2;

sum += 3;

sum += 4;

std::cout << "Сума перших чотирьох чисел: " << sum << std::endl;

return 0;

}

**Розгалужений обчислювальний процес**

Розгалужений обчислювальний процес включає в себе умовні конструкції, які визначають, які дії виконувати в залежності від певних умов. Найпоширенішою формою розгалуження є конструкція if-else.

Наприклад, програма, яка перевіряє, чи є число парним або непарним:

*#include <iostream>*

int main() {

int number;

std::cout << "Введіть число: ";

std::cin >> number;

if (number % 2 == 0) {

std::cout << "Число " << number << " є парним." << std::endl;

} else {

std::cout << "Число " << number << " є непарним." << std::endl;

}

return 0;

}

### **3.1.5 Програмування циклічних обчислювальних процесів**

Програмування циклічних обчислювальних процесів є важливою складовою будь-якої програми, оскільки дозволяє автоматизувати повторювані дії. У мові програмування C++ для цього використовуються різні типи циклів, такі як for, while та do-while.

**Цикл for**

Цикл for використовується для повторення одного блоку коду потрібну кількість разів. Наприклад, виведення чисел від 1 до 5 на консоль за допомогою цикла for.

#include <iostream>

int main() {

for (int i = 1; i <= 5; ++i) {

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

}

std::cout << std::endl;

return 0;

}

**Цикл while**

Цикл while виконується, доки вказана умова є істинною. Наприклад, виведення чисел від 1 до 4 на консоль за допомогою цикла while.

#include <iostream>

int main() {

int i = 1;

while (i <= 4) {

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

++i;

}

std::cout << std::endl;

return 0;

}

**Цикл do-while**

Цикл do-while виконується принаймні один раз, навіть якщо умова відразу невірна, а потім повторюється, доки умова виконується.

#include <iostream>

int main() {

int number;

do {

std::cout << "Введіть число (введіть 1 для завершення): ";

std::cin >> number;

std::cout << "Ви ввели: " << number << std::endl;

} while (number != 1);

return 0;

}

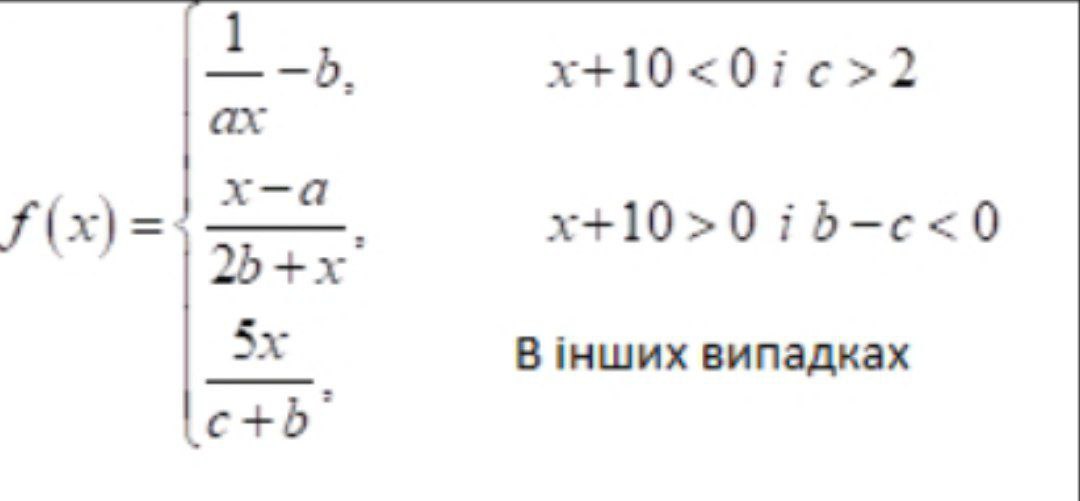
## **3.2 Практичне завдання**

**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.

**3.2.2 Постановка задачі**

****

**3.2.3 Код програми**

//Капічніков Єгор, ІПЗ-23

//підключення бібліотек

#include <iostream>

#include <cmath>

#include <iomanip>

#include <stdexcept>

int main() {

double a, b, c, xs, xend, step; //xs - х початок; хend - х кінцевий

std::cout << "Enter value for a: ";

std::cin >> a;

std::cout << "Enter value for b: ";

std::cin >> b;

std::cout << "Enter value for c: ";

std::cin >> c;

//введення змінних з клавіатури

std::cout << "Enter a start value of x: ";

std::cin >> xs;

std::cout << "Enter an ending value of x: ";

std::cin >> xend;

std::cout << "Enter value for a step: ";

std::cin >> step;

//виведення заголовку таблиці

std::cout << std::setw(15) << "x" << std::setw(30) << "f(x)" << std::endl;

try {

//обчислюється та виводиться значення f(x) для кожного х

for (double x = xs; x <= xend; x += step) {

double result;

//умови

if (x + 10 < 0 && c > 2) {

//перевірка, чи немає ділення на нуль

if (x == 0 && a == 0) {

throw std::invalid\_argument("There's a problem - you can't divide by zero. Be confident that no one from arguments doesn't amount to 0.");

}

result = 1/a\*x - b;

}

else if (x + 10 > 0 && b - c < 0) {

if (x == -2\*b) {

throw std::invalid\_argument("There's a problem - you can't divide by zero. Be confident that -2b != x.");

}

result = x-a/2\*b+x;

}

else {

if (c == -b) {

throw std::invalid\_argument("There's a problem - you can't divide by zero. Be confident that -с != b.");

}

result = 5\*x/c+b;

}

//виведення значень х та f(x) у вигляді таблиці

std::cout << std::setw(15) << x << std::setw(30) << result << std::endl;

}

}

catch (const std::invalid\_argument& e) {

//проводиться оброблення винятків та виведення повідомлень при надходженні помилок

std::cerr << "Error: " << e.what() << std::endl;

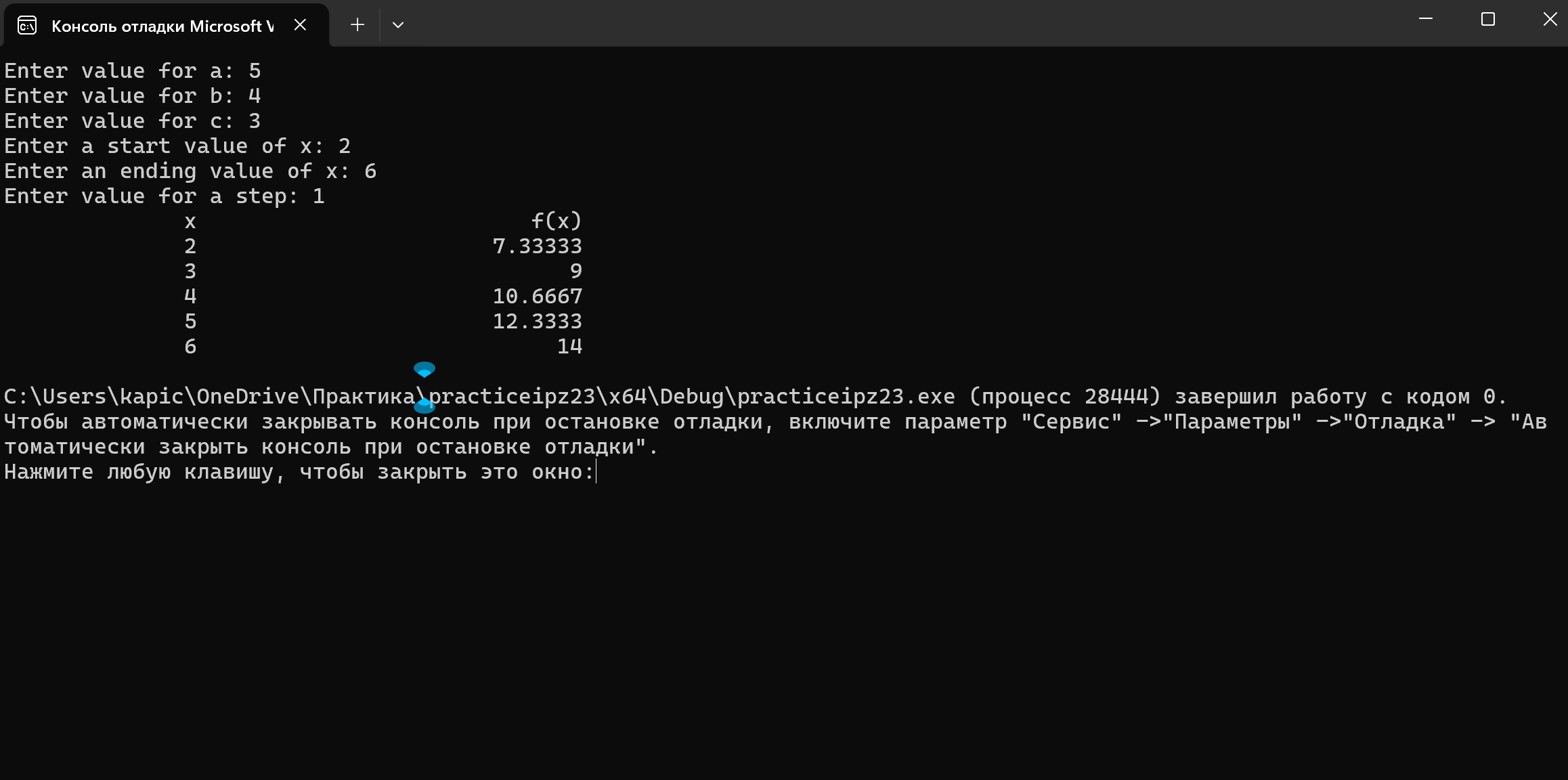
return 1;

}

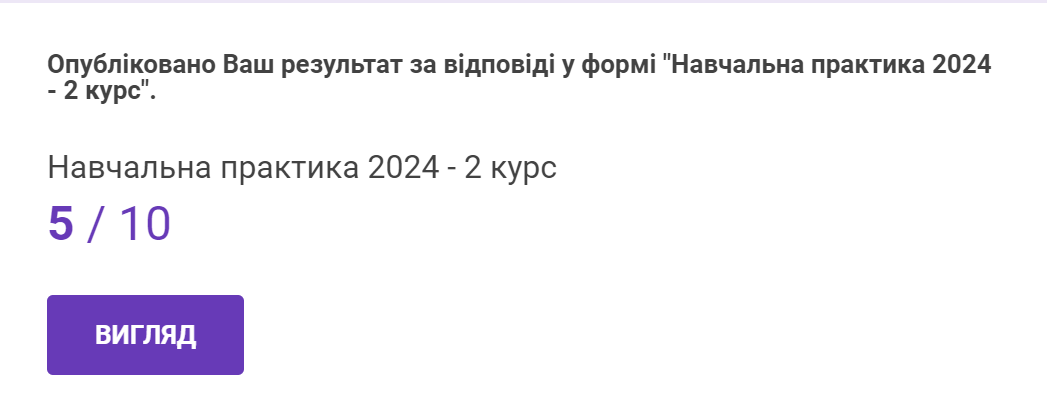
return 0;

}

**3.2.4 Скріншот виконання програми**

****

**3.3 Виконання тестів**

****

**4 РОЗДІЛ «ПРАКТИЧНА РОБОТА С++»**

**4.1 Введення-виведення у базових консольних програмах.**

У мові C++ дії, що пов’язані з операціями введення і виве­дення, виконуються за допомогою функцій бібліотек. Функції ведення і виведення бібліотек мови дозволяють читати дані з файлів та пристроїв і писати дані у файли і на пристрої.

Бібліотека мови C++ підтримує три рівня введення-виведення даних:

* введення-виведення потоку;
* введення-виведення нижнього рівня;
* введення-виведення для консолі і порту.

При введенні-виведенні потоку всі дані розглядаються як потік окремих байтів. Для користувача потік — це файл на дис­ку або фізичний пристрій, наприклад, дисплей чи клавіатура, або пристрій для друку, з якого чи на який направляється потік даних. Операції введення-виведення для потоку дозволяють об­робляти дані різних розмірів і форматів від одиночного символу до великих структур даних. Програміст може використовувати функції бібліотеки, розробляти власні і включати їх у бібліоте­ку. Для доступу до бібліотеки цих класів треба включити в програму відповідні заголовні файли.

За замовчуванням стандартні введення і виведення повідомлень про помилки відносяться до консолі користувача (клавіатури та екрана). Це означає, що завжди, коли програма очікує введення зі стандартного потоку, дані повинні надходити з клавіатури, а якщо програма виводить дані — то на екран.

У мові C++ існує декілька бібліотек, які містять засоби введення-виведення, наприклад:**stdio.h, iostream.h.** Найчастіше застосовують потокове введення-виведення даних, операції яко­го включені до складу класів **istream** або **iostream**. Доступ до бібліотеки цих класів здійснюється за допомогою використання у програмі директиви компілятора **#include <iostream.h>**.

**Для потокового введення даних** вказується операція «**>>**» («читати з»). Це перевантажена операція, визначена для всіх простих типів і покажчика на **char**. Стандартним потоком введення є **cin**.

Формат запису операції введення має вигляд:

**сin [>> values];**

де ***values*** — змінна.

Так, для введення значень змінних**х і у** можна записати:

**сin >> х >> у;**

Кожна операція «>>» передбачає введення одного значення. При такому введенні даних необхідно дотримуватись конкретних вимог:

* для послідовного введення декількох чисел їх слід розділяти символом пропуску (« ») або **Enter** (дані типу **char** роздiляти пропуском необов’язково);
* якщо послідовно вводиться символ і число (або навпаки), пропуск треба записувати тільки в тому випадку, коли символ (типу **char**) є цифрою;
* потік введення ігнорує пропуски;
* для введення великої кількості даних одним оператором їх можна розташовувати в декількох рядках (використовуючи Enter);
* операція введення з потоку припиняє свою роботу тоді, коли всі включені до нього змінні одержують значення. Наприклад, для операції введення**х і у**, що вказана вище, можна ввести значення **х та у** таким чином:

**2.345   789**  
або   **2.345**  
**789.**

Оскільки в цьому прикладі пропуск є роздільником між значеннями, що вводяться, то при введенні рядків, котрі мiстять пропуски у своєму складі, цей оператор не використовується. У такому випадку треба застосовувати функції **getline(), get()** тощо (див.*розділ 7*). У мові C++ бажано здійснювати потокові введення-виведення даних.

**Для потокового виведення даних** необхідна операція **«<<»** («записати в»), що використовується разом з ім’ям вихідного потоку **cout**. Наприклад, вираз

**cout << х;**

означає виведення значення змінної **х**(або запис у потік). Ця операція вибирає необхідну функцію перетворення даних у потік байтів.

Формат запису операції виведення представляється як:

**cout << data [<< data1];**,

де **data, data1** — це змінні, константи, вирази тощо.

Потокова операція виведення може мати вигляд:  
**cout << “у =” << x + а – sin(x) << “\n“;.**

Застосовуючи логічні операції, вирази треба брати в дужки:  
**cout << “р =” << (а && b || с) << “\n“;.**

Символ переведення на наступний рядок записується як рядкова константа, тобто **“\n”**, інакше він розглядається не як символ керуючої послідовності, а як число 10 (код символу). Таких помилок можна уникнути шляхом присвоювання значення ке­руючих символів змінним, тобто:

**#define << sp " "**

**#define << ht "\t"**

**#define << hl "\n".**

Тепер операцію виведення можна здійснити так:  
**cout <<** **“у =” << x** **+ а – sin(x)<<** **hl;** .

Слід пам’ятати, що ***при виведенні даних з використанням «cout <<» не виконується автоматичний перехід на наступ­ний рядок, для реалізації такого переходу застосовується так переведення рядка*** ***“\n” або операція*** ***endl***

**4.2 Структури даних для управління станом програми**

Масив — СД, згруповані за місцем розташування у пам'яті величини, що мають одну назву (ідентифікатор) і різні порядкові номери (індекси). Це поняття програмування відповідає математичним поняттям послідовності й таблиці (матриці).

Елемент масиву — одна з величин, що утворюють масив. Це поняття програмування відповідає математичному поняттю елемента послідовності чи матриці.

Індекс масиву — величина перелічуваного (зазвичай цілого) типу, яка (сукупність яких) вказує на конкретний елемент масиву. Це поняття програмування відповідає математичному поняттю номера елемента послідовності чи номеру рядка/стовпчика таблиці (матриці). У мові C++ найменше значення індексу — 0.

Масив має такі властивості:

* Всі елементи мають однаковий тип.
* Масив містить фіксовану (незмінну протягом роботи програми) кількість елементів.
* Елементи масиву розташовуються в пам’яті один за одним.
* Всі елементи зберігаються під одним іменем.

У мові C++ властивості елементів масиву не можна змінити протягом виконання програми. В деяких інших мовах це можливо.

Контейнер list (список) у мові С++ задає двонаправлений список. У ці списки можна швидко вставляти, а також видаляти елементи з них. Доступ до елементів списку (як і всіх наступних структур) здійснюють за допомогою вказівників (ітераторів).

#include <list> — замовлення необхідної бібліотеки;

list <тип\_даних> назва\_списку; — опис списку;

list <тип\_даних> :: iterator назва\_ітератора; — опис ітератора (вказівника);

l.begin() — вказівник на початок списку l;

l.end() — вказівник на кінець списку l;

l.rbegin() — реверсивний вказівник на кінець списку l;

l.rend() — реверсивний вказівник на початок списку l;

l.empty() — істинність висловлювання: «Список l порожній»;

l.push\_back(e) — додати елемент e в кінець списку l;

l.pop\_back() — видалити останній елемент списку l;

l.front() — перший елемент списку l;

l.back() — останній елемент списку l;

l.insert(i,e) — вставлення елемента e в позицію i списку l;

l.erase(і, у) — видалення зі списку l послідовності елементів від позиції і до позиції у.

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

**Класи файлового вводу/виводу**

Є три основні класи файлового вводу/виводу в мові C++:

**ifstream** (є дочірнім класу istream);

**ofstream** (є дочірнім класу ostream);

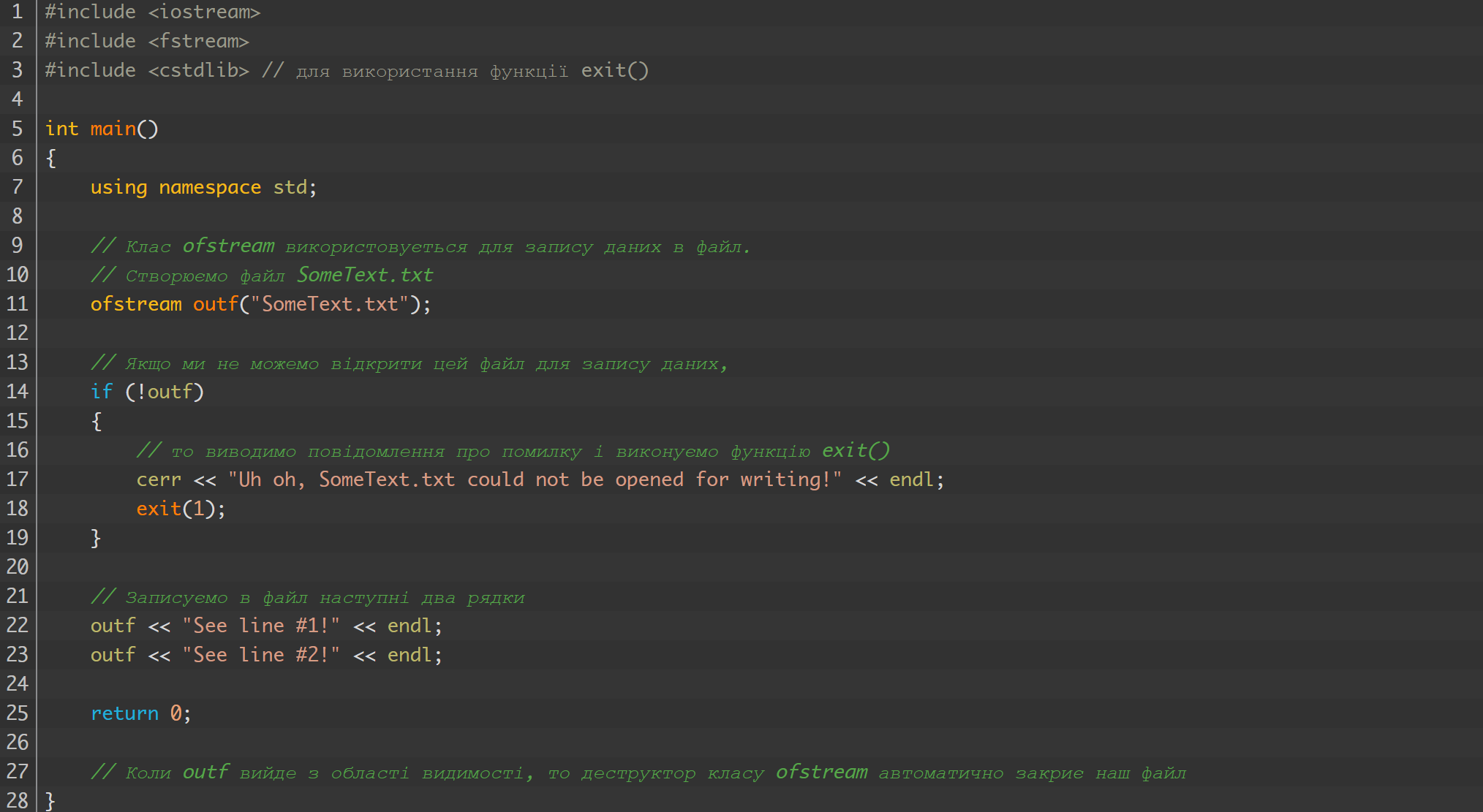
**fstream** (є дочірнім класу iostream).

За допомогою цих класів можна виконувати однонаправлений файловий ввід, однонаправлений файловий вивід і двонаправлений файловий ввід/вивід. Для їх використання потрібно всього лише підключити заголовок fstream.

На відміну від потоків cout, cin, cerr і clog, які відразу ж можна використовувати, файлові потоки повинні бути явно встановлені програмістом. Тобто, щоб відкрити файл для читання і/або запису, потрібно створити об’єкт відповідного класу файлового вводу/виводу, вказавши ім’я файлу в якості параметра. Потім, за допомогою оператора вставки (<<) або оператора вилучення (>>), можна записувати дані в файл або зчитувати вміст файлу. Після виконання даних дій потрібно закрити файл — явно викликати метод close() або просто дозволити файловій змінній вводу/виводу вийти з області видимості (деструктор файлового класу вводу/виводу закриє цей файл автоматично замість нас).

## **Файловий вивід**

Для запису в файл використовується клас ofstream. Наприклад:



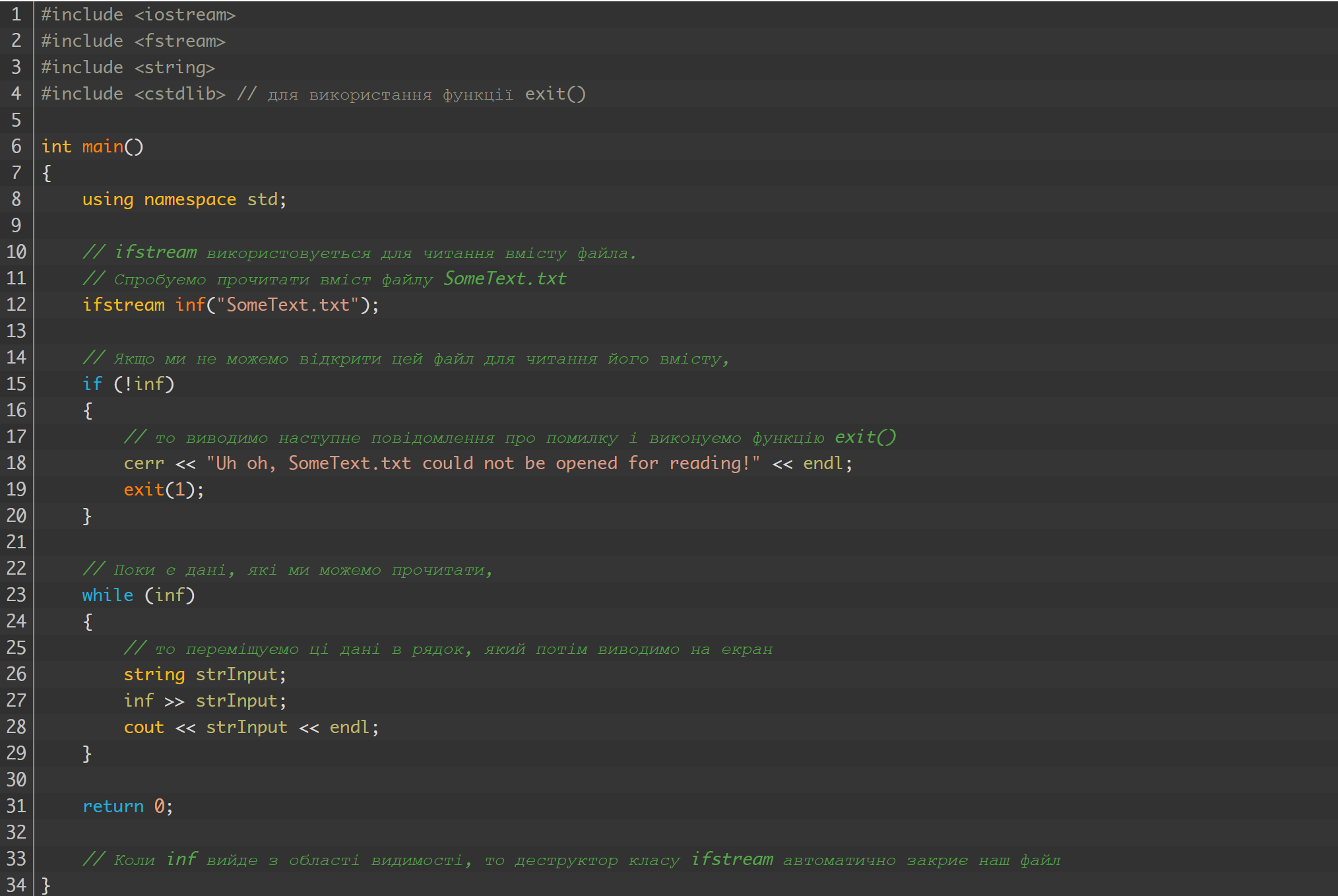
Якщо зайти в каталог проекту (ПКМ по вкладці з назвою .cpp-файлу в Visual Studio > “Открыть содержащую папку”), то можна побачити файл з ім’ям SomeText.txt, в якому знаходяться наступні рядки:

**See line #1!  
See line #2!**

Також можемо використати **метод put()** для запису одного символу в файл.

## **Файловий ввід**

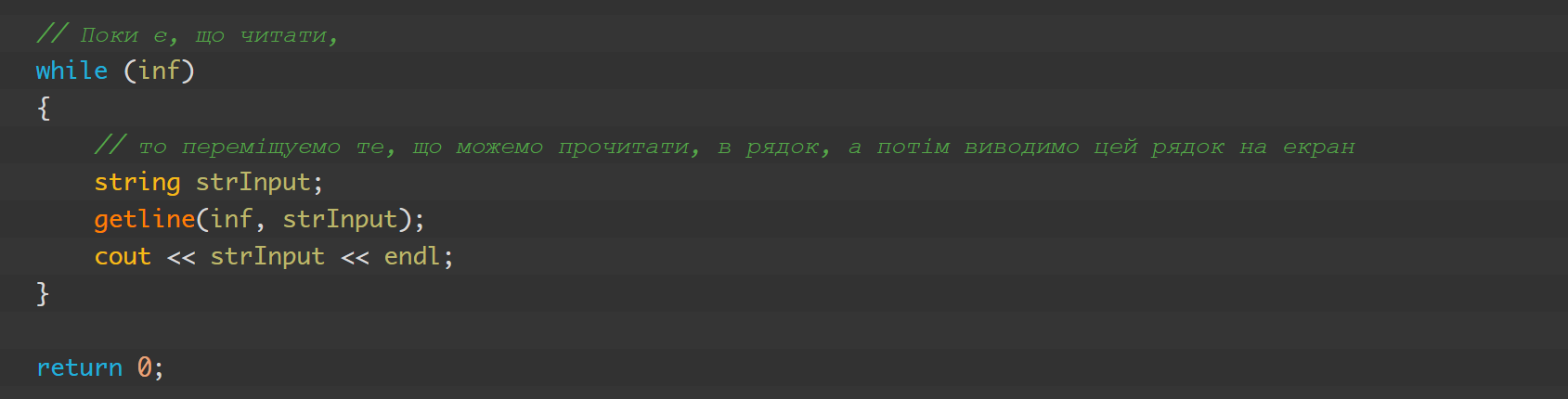
Тепер ми спробуємо прочитати вміст файлу, який створили в попередньому прикладі. Зверніть увагу, ifstream поверне 0, якщо ми досягли кінця файлу (це зручно для визначення «довжини» вмісту файлу). Наприклад:



Результат виконання програми:

**See  
line  
#1!  
See  
line  
#2!**

Оператор вилучення працює з «відформатованими даними», тобто він ігнорує всі пробіли, символи табуляції і символ нового рядка. Щоб прочитати весь вміст як є, без його ділення на частини (як у вищенаведеному прикладі), нам потрібно використати метод **getline():**



Результат виконання програми:

**See line #1!  
See line #2!**

## **Буферизований вивід**

Вивід в мові C++ може бути буферизований. Це означає, що все, що виводиться в файловий потік, не може відразу ж бути записано на диск (в конкретний файл). Це зроблено, в першу чергу, з міркувань продуктивності. Коли дані буфера записуються на диск, то це називається очищенням буфера. Одним із способів очищення буфера є закриття файлу. У такому випадку весь вміст буфера буде переміщено на диск, а потім файл буде закрито.

Буферизація виводу зазвичай не є проблемою, але при певних обставинах вона може викликати проблеми у необережних новачків. Наприклад, коли в буфері зберігаються дані, а програма передчасно завершує своє виконання (або в результаті збою, або шляхом виклику функції exit()). У таких випадках деструктори класів файлового вводу/виводу не виконуються, файли ніколи не закриваються, буфери не очищаються і наші дані губляться назавжди. Ось чому гарною ідеєю є явне закриття всіх відкритих файлів перед викликом функції exit().

Також буфер можна очистити вручну, використовуючи метод ostream::flush() або відправивши std::flush в вихідний потік. Будь-який з цих способів може бути корисний для забезпечення негайного запису вмісту буфера на диск в разі збою програми.

**Режими відкриття файлів**

Що відбудеться, якщо ми спробуємо записати дані в уже існуючий файл? Повторний запуск вищенаведеної програми (найперша) показує, що вихідний файл повністю перезаписується при повторному запуску програми. А що, якщо нам потрібно додати дані в кінець файлу? Виявляється, [конструктори](https://acode.com.ua/urok-124-konstruktory/) файлового потоку приймають необов’язковий другий параметр, який дозволяє вказати програмісту спосіб відкриття файлу. В якості цього параметра можна передавати наступні флаги (які знаходяться в класі ios):

**app**— відкриває файл в режимі додавання;

**ate** — переходить в кінець файлу перед читанням/записом;

**binary** — відкриває файл в бінарному режимі (замість текстового режиму);

**in** — відкриває файл в режимі читання (за замовчуванням для ifstream);

**out**— відкриває файл в режимі запису (за замовчуванням для ofstream);

**trunc** — видаляє файл, якщо він вже існує.

Можна вказати відразу кілька флагів шляхом використання побітового АБО ().

**ifstream** за замовчуванням працює в режимі ios::in;

**ofstream** за замовчуванням працює в режимі ios::out;

**fstream** за замовчуванням працює в режимі ios::in АБО ios::out, що означає, що ви можете виконувати як читання вмісту файлу, так і запис даних в файл.

## **Явне відкриття файлів за допомогою функції open()**

Точно так же, як ми явно закриваємо файл за допомогою методу close(), ми можемо явно відкривати файл за допомогою **функції open()**. Функція open() працює аналогічно конструкторам класу файлового вводу/виводу: приймає ім’я файлу і режим (необов’язково), в якому потрібно відкрити файл, в якості параметрів.

**4.4 Формати файлів та їх обробка**

Формати файлів та їх обробка є ключовими аспектами роботи з інформацією у різних сферах діяльності. Ось деякі з найпоширеніших форматів файлів та способи їх обробки:

Текстові файли

1. TXT: простий текстовий файл без форматування. Використовується для зберігання необробленого тексту.

- Обробка: редагування у будь-якому текстовому редакторі (Notepad, Sublime Text), програмна обробка мовами Python, Java, C++ тощо.

2. JSON (JavaScript Object Notation): легкий формат обміну даними, який легко читається і записується людиною.

- Обробка: програмна обробка мовами JavaScript, Python (json бібліотека), Java тощо.

3. CSV (Comma-Separated Values): формат для зберігання табличних даних, де значення розділені комами.

- Обробка: відкриття в електронних таблицях (Excel, Google Sheets), програмна обробка мовами Python (pandas), R тощо.

4. XML (eXtensible Markup Language): формат для зберігання та передачі структурованих даних.

- Обробка: використання парсерів (lxml в Python, DOM у Java), редагування спеціалізованими редакторами (XMLSpy).

**JSON**

JSON - текстовий формат обміну даними, що базується на JavaScript. Як і багато інших текстових форматів, JSON легко читається людьми.

Незважаючи на походження від JavaScript (точніше, від підмножини мови стандарту ECMA-262 1999), формат вважається незалежним від мови і може використовуватися практично з будь-якою мовою програмування. Для багатьох мов існує готовий код для створення та обробки даних у форматі JSON.

За рахунок своєї лаконічності, порівняно з XML, формат JSON може бути більш підходящим для серіалізації складних структур. Застосовується у веб-додатках як для обміну даними між браузером та сервером (AJAX), так і між серверами (програмні зв'язки HTTP).

Оскільки формат JSON є підмножиною синтаксису мови JavaScript, він може бути швидко десеріалізований вбудованою функцією JSON.parse().

**CSV**

CSV - текстовий формат, призначений для представлення табличних даних. Рядок таблиці відповідає рядку тексту, який містить одне або кілька полів, розділених комами.

Кожен рядок файлу – це один рядок таблиці.

Розділювачем значень колонок є символ комою (,). Проте практично часто використовуються інші роздільники, тобто формат плутають з DSV і TSV.

Значення, що містять зарезервовані символи (подвійна лапка, кома, крапка з комою, новий рядок) обрамляються подвійними лапками ("). Якщо у значенні зустрічаються лапки - вони подаються у файлі у вигляді двох лапок поспіль.

**Читання/запис JSON файлів**

**Читання**

Читання даних з формату JSON і перетворення їх в об’єкти JavaScript здійснюється за допомогою методу JSON.parse(). Цей метод приймає як аргумент рядок JSON і повертає відповідний JavaScript об’єкт.

Приклад:

#include <iostream>

#include <fstream>

#include <sstream>

#include <rapidjson/document.h>

#include <rapidjson/istreamwrapper.h>

int main() {

std::ifstream file("example.json");

if (!file.is\_open()) {

std::cerr << "Не вдалося відкрити файл" << std::endl;

return 1;

}

rapidjson::IStreamWrapper isw(file);

rapidjson::Document doc;

doc.ParseStream(isw);

if (doc.HasParseError()) {

std::cerr << "Помилка парсингу JSON" << std::endl;

return 1;

}

// Приклад доступу до даних

std::string name = doc["name"].GetString();

int age = doc["age"].GetInt();

const rapidjson::Value& hobbies = doc["hobbies"];

std::cout << "Name: " << name << std::endl;

std::cout << "Age: " << age << std::endl;

std::cout << "Hobbies: ";

for (rapidjson::SizeType i = 0; i < hobbies.Size(); ++i) {

std::cout << hobbies[i].GetString() << " ";

}

std::cout << std::endl;

file.close();

return 0;

}

**Запис**

Запис даних у формат JSON виконується за допомогою методу JSON.stringify(). Цей метод приймає об’єкт JavaScript і перетворює його на рядок JSON.

Приклад:

*#include <iostream>*

*#include <fstream>*

*#include <nlohmann/json.hpp> // Підключення заголовного файлу nlohmann/json*

*using json = nlohmann::json;*

*int main() {*

*json j;*

*// Створення JSON-об'єкта*

*j["name"] = "Alice";*

*j["age"] = 30;*

*j["hobbies"] = {"reading", "hiking", "coding"};*

*// Запис JSON-об'єкта у файл*

*std::ofstream file("example.json");*

*if (file.is\_open()) {*

*file << j.dump(4); // 4 означає відступ у 4 пробіли для форматування*

*file.close();*

*std::cout << "JSON записано у файл example.json" << std::endl;*

*} else {*

*std::cerr << "Не вдалося відкрити файл" << std::endl;*

*}*

*return 0;*

*}*

**Читання/запис CSV файлів**

Для роботи з CSV файлами можна використовувати бібліотеку rapidcsv.

*#include "rapidcsv.h"*

*#include <iostream>*

*#include <vector>*

*#include <string>*

*int main() {*

*try {*

*// Читання CSV-файлу*

*rapidcsv::Document doc("example.csv");*

*// Отримання даних з колонки "Name"*

*std::vector<std::string> names = doc.GetColumn<std::string>("Name");*

*std::cout << "Імена: ";*

*for (const auto& name : names) {*

*std::cout << name << " ";*

*}*

*std::cout << std::endl;*

*// Отримання даних з рядка 1*

*std::vector<std::string> row = doc.GetRow<std::string>(1);*

*std::cout << "Рядок 1: ";*

*for (const auto& cell : row) {*

*std::cout << cell << " ";*

*}*

*std::cout << std::endl;*

*} catch (const std::exception &ex) {*

*std::cerr << "Помилка: " << ex.what() << std::endl;*

*}*

*return 0;*

*}*

**Читання/запис XML файлів**

Для роботи з CSV файлами можна використовувати бібліотеку pugixml.

#include <iostream>

#include "pugixml.hpp"

int main() {

pugi::xml\_document doc;

// Завантаження XML-файлу

pugi::xml\_parse\_result result = doc.load\_file("example.xml");

if (!result) {

std::cerr << "Не вдалося завантажити файл: " << result.description() << std::endl;

return 1;

}

// Отримання кореневого елемента

pugi::xml\_node root = doc.child("root");

// Читання даних

for (pugi::xml\_node node = root.child("item"); node; node = node.next\_sibling("item")) {

std::cout << "ID: " << node.attribute("id").as\_int()

<< ", Name: " << node.child\_value("name")

<< ", Value: " << node.child\_value("value") << std::endl;

}

return 0;

}

**Порівняння: JSON та CSV**

## **1. Структура та Формат**

### **JSON**

* **Формат**: Текстовий, базується на синтаксисі об’єктів JavaScript.
* **Структура**: Підтримує вкладені структури (масиви та об'єкти), тому може зберігати складні ієрархічні дані.

### **CSV**

* **Формат**: Текстовий, де кожен рядок представляє запис, а кожне значення в рядку розділено комами або іншими роздільниками.
* **Структура**: Плоска таблиця, не підтримує вкладеність, тому підходить для табличних даних.

**2**. **Читабельність і легкість використання**

### **JSON**

* **Читабельність**: Легко читається людиною і добре підходить для складних структур даних.
* **Легкість Використання**: Легко парситься у багатьох мовах програмування. Зазвичай використовується для структурованих даних.

### **CSV**

* **Читабельність**: Легко читається людиною, особливо для табличних даних, але складніше для вкладених структур.
* **Легкість Використання**: Легко обробляється у багатьох мовах програмування та офісних додатках (Excel, Google Sheets).

## **3. Зберігання та Об'єм Даних**

### **JSON**

* **Зберігання**: Може займати більше місця через додаткову синтаксичну інформацію.
* **Об'єм Даних**: Підходить для невеликих до середніх обсягів даних. Іноді може бути великим через дублювання ключів.

### **CSV**

* **Зберігання**: Зазвичай займає менше місця, оскільки має просту структуру без додаткової синтаксичної інформації.
* **Об'єм Даних**: Підходить для великих обсягів табличних даних. Компактний, але не підходить для складних структур.

## **4. Гнучкість та Складність Даних**

### **JSON**

* **Гнучкість**: Підтримує складні та вкладені структури даних.
* **Складність**: Легко представляє об'єкти і масиви. Використовується для складних, багаторівневих даних.

### **CSV**

* **Гнучкість**: Обмежений до плоских структур даних. Складні структури важко представляти.
* **Складність**: Простий для табличних даних, але не підходить для складних структур.

## **5**. **Підтримка та Сумісність**

### **JSON**

* **Підтримка**: Широко підтримується багатьма мовами програмування і веб-API.
* **Сумісність**: Легко інтегрується з сучасними веб-технологіями та REST API.

### **CSV**

* **Підтримка**: Широко підтримується більшістю мов програмування, баз даних, електронних таблиць.
* **Сумісність**: Легко імпортується/експортується з більшості інструментів для роботи з даними.

## **6. Використання**

### **JSON**

* **Переваги**:
  + Підходить для складних структур даних.
  + Хороша підтримка у веб-розробці.
  + Легко читається і пишеться.
* **Недоліки**:
  + Може займати більше місця.
  + Менш ефективний для простих табличних даних.

### **CSV**

* **Переваги**:
  + Простий формат, легко читається і пишеться.
  + Компактний для табличних даних.
  + Широка підтримка в офісних додатках.
* **Недоліки**:
  + Не підтримує вкладеність.
  + Важко працювати зі складними структурами.

**Отже,**

* **Вибір формату залежить від завдання:**
  + Якщо потрібна робота з вкладеними і складними структурами даних – обирайте **JSON**.
  + Якщо ви працюєте з простими табличними даними і вам потрібна компактність – **CSV** стане кращим вибором.

**Що займає менше місця – JSON чи CSV?**

Зазвичай **CSV** займає менше місця порівняно з **JSON**, особливо для простих табличних даних.

**Приклад займаного місця**

**CSV**:

***id,name,age***

***1,Alice,30***

***2,Bob,25***

Невеликий розмір, мінімальний синтаксичний вклад.

**JSON:**

***[***

***{***

***"id": 1,***

***"name": "Alice",***

***"age": 30***

***},***

***{***

***"id": 2,***

***"name": "Bob",***

***"age": 25***

***}***

***]***

***Розмір більший,*** оскільки для кожного елементу присутні ключі "id", "name", "age", а також синтаксис масивів і об'єктів.

## **Співвідношення Розміру**

У випадку простих табличних даних CSV буде меншим через відсутність додаткової синтаксичної інформації:

### **Проста Таблиця**

* **CSV**: Всі дані записуються в одну лінію без ключів, що зменшує об'єм.
* **JSON**: Містить додаткові символи для ключів, ком, квадратних і фігурних дужок.

### **Порівняння Розміру:**

**Приклад простого набору даних** (3 рядки, 3 стовпці):

* **CSV**: ~30 байтів
* **JSON**: ~70 байтів (включаючи ключі і синтаксичні символи)

Якщо структура даних має вкладені об'єкти або масиви, JSON забезпечить збереження таких структур, тоді як CSV потребує значних зусиль для представлення аналогічних даних.

Наприклад, список покупок для кожного користувача у JSON:

[

{

"name": "Alice",

"purchases": ["milk", "bread"]

},

{

"name": "Bob",

"purchases": ["eggs", "bacon"]

}

]

У CSV потрібно було б створювати додаткові стовпці або використовувати складніші методи обробки даних.

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

**VCS** – місце зберігання коду. Вона уточнена саме на розробку продуктів. Тобто зберігання коду, синхронізацію роботи кількох людина, створення релізів (бранчів).

Як працює **VCS:**

**1. Створити репозиторій**

Вихідно потрібно створити місце, де лежатиме код. Воно називається репозиторій. Створюється один раз адміністратором.

**2. Встановити проект з репозиторія**

Тепер потрібно отримати проект із репозиторію. Забрати код таким чином потрібно рівно один раз

**Що з допомогою треба робити:**

**1. Оновити проект, забрати останню версію з репозиторію**

**2. Вносити зміну у репозиторій**

Основні типи VCS

1. Локальні VCS:

- Зберігають версії файлів на локальному комп'ютері.

- Простий, але має обмеження у можливостях командної роботи.

2. Централізовані VCS (CVCS):

- Використовують центральний сервер для зберігання всіх версій файлів.

- Забезпечують спільну роботу над проектами, але є залежність від центрального сервера.

3. Розподілені VCS (DVCS):

- Кожен розробник має повну копію історії змін проекту на своєму локальному комп'ютері.

- Більш гнучкі і стійкі до збоїв центрального сервера.

**Гілка (Branch)** у Git - це окрема версія проекту, яка існує незалежно від інших версій проекту. Гілки дозволяють розробникам працювати над різними завданнями або функціями одночасно, не заважаючи роботі один одного. Основна гілка в більшості проектів називається main або master. Коли створюється нова гілка, вона є копією основної гілки або будь-якої іншої гілки, від якої вона була створена.

**Коміт (Commit)** у Git - це зафіксований знімок змін у проекті. Коміт створюється, коли розробник виконує команду git commit, що зберігає поточний стан змін у репозиторії. Кожен коміт містить унікальний ідентифікатор (хеш), метадані (автора, дату) та повідомлення, яке описує зміни.

**Git Flow**

Git Flow - це методологія управління розробкою програмного забезпечення з використанням системи контролю версій Git. Вона допомагає командам організувати роботу над проектом, розділяючи його на різні гілки, які мають свої завдання та життєві цикли. Цей підхід включає в себе кілька основних і допоміжних гілок, які дозволяють легко керувати новими функціями, релізами і виправленнями багів.

Існує дві основні гілки, які завжди присутні у проекті:

- Main (або Master): ця гілка завжди містить стабільну версію коду, готову до випуску. Будь-який час, коли користувачі або клієнти отримують код, вони отримують його з цієї гілки. Всі релізи проекту йдуть з main гілки.

- Develop: це гілка, де ведеться активна розробка. Всі нові функції, вдосконалення і виправлення помилок спочатку додаються до цієї гілки. Вона є основною робочою гілкою для розробників. Після завершення роботи і тестування зміни з цієї гілки потрапляють до main через процес підготовки релізу.

Окрім основних гілок, Git Flow передбачає використання допоміжних гілок для різних типів завдань:

- Feature гілки: використовуються для розробки нових функцій. Кожна нова функція створюється в окремій гілці, яка береться з develop. Після завершення роботи над функцією, ця гілка зливається назад в develop. Це дозволяє працювати над новими функціями незалежно від основного коду.

- Release гілки: використовуються для підготовки нового релізу. Коли гілка develop готова до випуску, створюється release гілка. В цій гілці відбувається останнє тестування і виправлення незначних помилок перед релізом. Після завершення, release гілка зливається в main (що створює новий реліз) і в develop (для синхронізації всіх виправлень).

- Hotfix гілки: використовуються для термінового виправлення критичних багів в основній гілці. Вони створюються від main, і після виправлення, зміни зливаються назад у main і develop. Це дозволяє швидко реагувати на критичні проблеми, не заважаючи поточній розробці.

**Основні команди Git**

Git є найпопулярнішою DVCS і має потужний набір команд для управління версіями.

Ініціалізація репозиторію:

*git init*

Клонування існуючого репозиторію:

*git clone https://github.com/user/repository.git*

Перегляд статусу файлів:

*git status*

Додавання файлів до індексу:

*git add filename*

*git add .*

Перегляд історії комітів*:*

*git log*

Створення нової гілки:

*git branch new-branch*

Перемикання між гілками:

*git checkout new-branch*

Злиття гілок:

*git merge branch-name*

Оновлення локального репозиторію з віддаленого:

*git pull*

Відправка локальних змін до віддаленого репозиторію:

*git push*

**4.6 Мова Markdown для опису проекті**

Markdown – мова розмітки, яку використовують для створення нотаток, документів, вебсайтів, технічної документації тощо. Цікаво те, що файли Markdown можна перетворювати на презентації, які зручно переглядати в браузері.

За допомогою **Markdown** можна робити таку розмітку:

1. Заголовки 6 рівнів

# Заголовок 1

## Заголовок 2

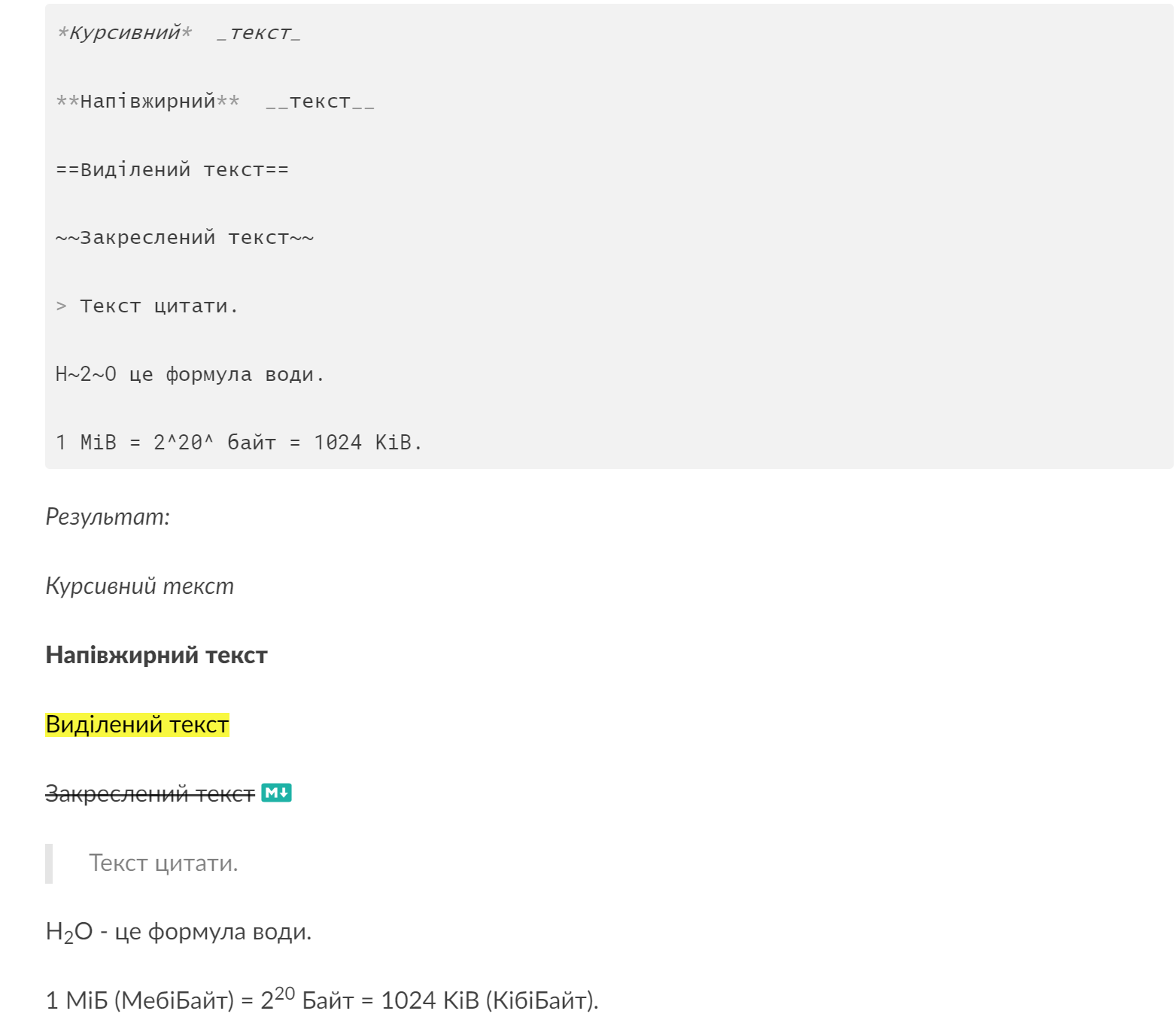
### Заголовок 3

#### Заголовок 4

##### Заголовок 5

###### Заголовок 6

2. Форматування тексту



3. Створювати списки

- елемент 1

- елемент 2

- елемент 3

Також у списках можуть бути вкладені елементи

- елемент 1

- елемент 2

\* вкладений елемент 2.1

\* вкладений елемент 2.2

- елемент 3

4. Посилання

Це [посилання](https://example.com).

Це [приклад][1] [кількох][2] [посилань][id] з розміткою як у виносок в тексті. Можна використовувати [короткий запис][] без `id`.

[1]: https://example.com "Назва посилання 1"

[2]: https://example.com/some

[id]: https://example.com/links (Назва посилання id)

[короткий запис]: https://example.com/short

5. Ідентифікатори заголовків

### Горизонтальна лінія

[Горизонтальна лінія](#горизонтальна-лінія)

Результат:

Горизонтальна лінія

6. Зображення

![alt](https://bit.ly/33B4VxM "shih-tzu")

Результат:



Також можна змінювати розмір зображення

![alt](https://bit.ly/33B4VxM "shih-tzu" = 80х120)

Результат:



## **4.7 Розроблення ігрової програми**

**4.7.1 Код програми**

// Kapychnykov Yegor, ІПЗ-23

#include <iostream>

using namespace std;

// Ініціалізація поля

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

char current\_marker;

int current\_player;

void drawBoard() {

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

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

cout << "|";

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

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

}

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

}

}

bool placeMarker(int slot) {

int row = (slot - 1) / 3;

int col = (slot - 1) % 3;

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

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

return true;

}

return false;

}

int winner() {

// Перевірка строк

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

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

return current\_player;

}

// Перевірка стовбців

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

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

return current\_player;

}

// Перевірка діагоналей

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() {

cout << "Player 1, choose your option (X or O): ";

char marker\_p1;

cin >> marker\_p1;

current\_player = 1;

current\_marker = marker\_p1;

drawBoard();

int player\_won;

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

cout << "Player " << current\_player << ", choose number of cell (1-9): ";

int slot;

cin >> slot;

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

cout << "Incorrect number of cell. You can choose numbers only from 1 to 9.\n";

i--;

continue;

}

if (!placeMarker(slot)) {

cout << "Cell;s already chosen. Try another one.\n";

i--;

continue;

}

drawBoard();

player\_won = winner();

if (player\_won == 1) {

cout << "Player 1 won!\n";

break;

}

if (player\_won == 2) {

cout << "Player 2 won!\n";

break;

}

swapPlayerAndMarker();

}

if (player\_won == 0)

cout << "It's a draw\n";

}

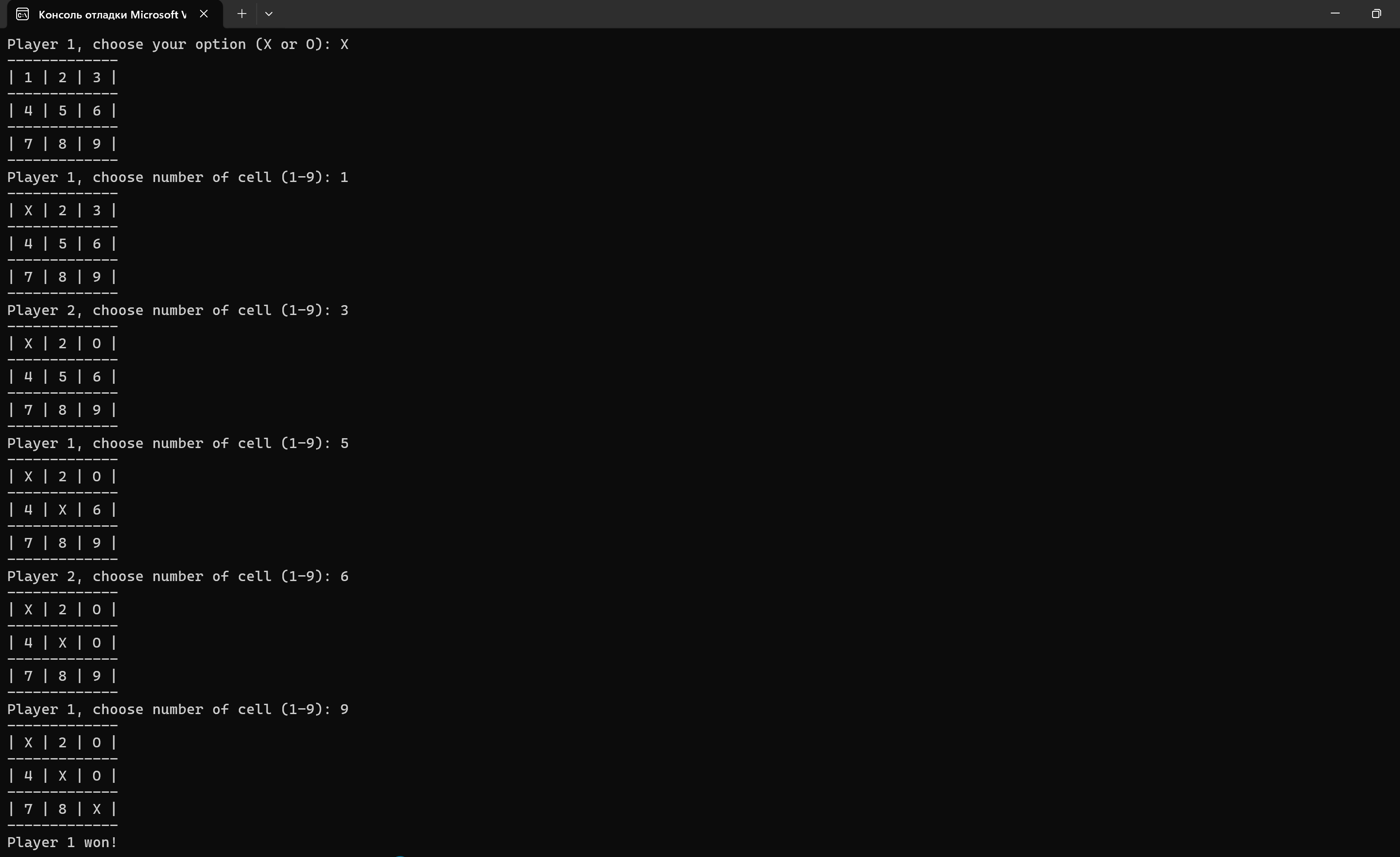
int main() {

game();

return 0;

}

**4.7.2 Скріншоти виконання программи**



**5 РОЗДІЛ «ВЕБ-ДИЗАЙН ТА HTML»**

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

Вебсторінка — це текстовий документ, розмічений тегами HTML, відображений у вікні браузера.

HTML-документ (HTML-файл) є текстовим документом, тому створити його можна в будь-якому текстовому редакторі.

Тег — це записана в кутових дужках < > позначка мови HTML, яку опрацьовує браузер.

HTML-документ складається з елементів, які визначаються за допомогою тегів. Теги зазвичай йдуть парами: відкриваючий тег <tagname> і закриваючий тег </tagname>. Деякі теги можуть бути самозакриваючимися, наприклад, <img/>.

Структура веб-сторінки визначається її HTML-кодом, який організований у певному порядку, щоб браузери могли правильно відображати її вміст. Основні частини HTML-документа включають оголошення типу документа, кореневий елемент, голову та тіло документа:

• <!DOCTYPE html>: Оголошення типу документа, яке вказує на версію HTML.

• <html>: Кореневий елемент, що містить всю сторінку.

• <head>: Містить метаінформацію про документ, наприклад, заголовок (<title>), посилання на стилі (<link>) та скрипти (<script>).

• <body>: Містить вміст веб-сторінки, який відображається в браузері.

***ПЕРЕЛІК ТЕГІВ, УМОВНИХ ПОЗНАЧЕНЬ, СПЕЦІАЛЬНИХ СИМВОЛІВ, ТЕРМІНІВ І ОДИНИЦЬ***

-'<html>` – визначає початок та кінець HTML документа.

- `<head>` – містить метадані веб-сторінки.

- `<title>` – визначає заголовок веб-сторінки, який відображається в браузері.

- `<meta>` – використовується для вказання метаданих, таких як кодування символів.

- `<style>` – містить CSS стилі для веб-сторінки.

- `<body>` – визначає основний контент веб-сторінки.

- `<header>` – визначає верхній колонтитул документа або розділу.

- `<nav>` – містить навігаційне меню.

- `<ul>` – визначає ненумерований список.

- `<li>` – визначає елемент списку.

- `<a>` – визначає посилання на іншу веб-сторінку або документ.

- `<main>` – визначає головний вміст документа.

- `<section>` – визначає розділ в документі.

- `<h1>` до `<h6>` – визначає заголовки різного рівня на веб-сторінці.

- `<p>` – визначає абзац тексту.

- `<em>` – виділяє текст курсивом.

- `<b>` – виділяє текст жирним.

- `<i>` – виділяє текст курсивом.

- `<u>` – підкреслює текст.

- `<table>` – визначає таблицю.

- `<hr>` – вставляє горизонтальну лінію (роздільник).

- `<footer>` – визначає нижній колонтитул документа або розділу.

- `<article>` – визначає самостійний фрагмент вмісту.

- `<aside>` – визначає побічний вміст, наприклад, бокову панель.

- `<figure>` – визначає самостійний контейнер для графічного вмісту.

- `<figcaption>` – визначає підпис до елементу `<figure>`.

- `<img>` – вбудовує зображення в HTML документ.

- `<audio>` – визначає аудіо-контент.

- `<source>` – вказує джерело мультимедіа для `<audio>` або `<video>`.

- `<video>` – визначає відео-контент.

- `<br>` – вставляє розрив рядка.

- `<hr>` – вставляє горизонтальну лінію (роздільник).

- `<footer>` – визначає нижній колонтитул документа або розділу.

- `<article>` – визначає самостійний фрагмент вмісту.

- `<aside>` – визначає побічний вміст, наприклад, бокову панель.

- `<figure>` – визначає самостійний контейнер для графічного вмісту.

- `<figcaption>` – визначає підпис до елементу `<figure>`.

- `<img>` – вбудовує зображення в HTML документ.

- `<audio>` – визначає аудіо-контент.

- `<source>` – вказує джерело мультимедіа для `<audio>` або `<video>`.

- `<video>` – визначає відео-контент.

- `<table>` – визначає таблицю.

- `<thead>` – визначає заголовок таблиці.

- `<tbody>` – визначає тіло таблиці.

- `<tfoot>` – визначає підвал таблиці.

- `<tr>` – визначає рядок таблиці.

- `<th>` – визначає заголовок стовпця таблиці.

- `<td>` – визначає клітинку таблиці.

- `<form>` – визначає форму для введення даних користувача.

- `<input>` – визначає поле введення даних.

- `<label>` – визначає мітку для елемента введення.

- `<textarea>` – визначає багаторядкове поле введення тексту.

- `<strong>` – виділяє текст жирним шрифтом, підкреслюючи його важливість.

- `<small>` – відображає текст меншим шрифтом.

- `<blockquote>` – визначає цитату з відступом.

- `<code>` – відображає текст у вигляді коду.

- `<pre>` – відображає попередньо відформатований текст, зберігаючи пробіли та розриви рядків.

- `<button>` – визначає кнопку.

- `<select>` – визначає випадаючий список.

- `<option>` – визначає елемент списку в `<select>`.

- `<div>` – визначає блоковий контейнер.

- `<span>` – визначає стрічковий контейнер.

Теги з помилками або такі, що не підтримуються браузером, ігноруються. Тег може містити атрибути, значення яких впливають на його дію. Атрибути записуються у відкривальному тегу.



Для створення маркованого списку використовують тег <ul>…</ul> з атрибутом type, який задає вигляд маркера:  
"disk" — 5.PNG; "circle" — 6.PNG; "square" — 7.PNG.

Для створення нумерованих списків використовують тег <ol>…</ol> з атрибутами type (тип нумерації: "1" — арабські цифри, "А" — латинські літери, "І" — римська нумерація тощо) і start (задання початкового номера арабськими цифрами).

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

Для управління текстом і зміни його виду існує досить велика кількість різних елементів. Що не дивно, адже текст - це найпопулярніший вид інформації.

### **Теги рядка та абзацу**

Текст в HTML вводиться в будь-якому місці між тегами **<BODY></BODY>.** Текст автоматично розміщується за шириною вікна браузера. Якщо ж необхідно на Веб-сторінці помістити порожній рядок або забезпечити в потрібному місці перехід до наступного рядка, слід використовувати спеціальні теги: тег розриву абзацу **<P></P**>, та розриву рядка **<BR>.**

Тег розриву абзацу дає команду браузеру закінчити поточний абзац і помістити порожній рядок перед наступним абзацом. Тег розриву рядка вказує про закінчення поточного рядка і переходу до наступного. Для тегу розриву абзацу використання закриваючого тегу **</P>** не є необхідним.

Мова HTML надає можливість виділяти фрагменти тексту напівжирним шрифтом, курсивом, підкресленням тощо.

**Теги виділення тексту**

|  |  |  |
| --- | --- | --- |
| **Тег** | **Стиль** | **Приклад** |
| **<B>** | Напівжирний шрифт | **Напівжирний** |
| **<strong>** | Напівжирний шрифт | **Напівжирний** |
| **<I>** | Курсив | *Курсив* |
| **<em>** | Курсив | *Курсив* |
| **<U>** | Підкреслення | Підкреслений текст |
| **<SUB>** | Підрядковий текст | Текст підрядковий текст |
| **<SUP>** | Надрядковий текст | Текст надрядковий текст |
| **<STRIKE>** | Перекреслення | ~~Перекреслений~~ |

### **Теги заголовків**

На сайті заголовки дуже важливі. **Заголовки в HTML** допомагають пошуковим системам виділити на сайті найважливішу інформацію, а пошукові системи винагороджують ваш сайт високими позиціями у рейтингу.

У мові HTML для виділення заголовків передбачено ціле сімейство тегів: від **<h1>** до **<h6>.**Тег **<h1>** позначає найважливіший заголовок (заголовок верхнього рівня), а тег **<h6>** позначає підзаголовок самого нижнього рівня.

На практиці рідко зустрічаються тексти, в яких зустрічаються підзаголовки нижче третього рівня. Тому найбільш часто використовуваними тегами заголовків є: **<h1>, <h2> і <h3>.**

Стандарт мови HTML нараховує 11 атрибутів тега заголовка, але не всі вони реалізовані у більшості програм-браузерів.

Найбільш використовуваний атрибут для тегу **<H>** -  **ALIGN**, який дозволяє вирівнювати текст за правою, лівою межею або по центру. За замовчуванням текст вирівнюється за лівою межею. Даний атрибут застосовується також до графіки і таблиць.  Значення атрибута **ALIGN** такі ж, як і для тегу розбиття тексту на абзаци **<р>.**

**Списки**[**​**](https://goit.global/textbooks/marathons/html-css/uk/docs/text/#%D1%81%D0%BF%D0%B8%D1%81%D0%BA%D0%B8)

Списки дозволяють упорядкувати колекції і представити їх в наочному і зручному для користувача вигляді. Список - це контейнер, дітьми якого можуть бути тільки елементи списку - теги <li>.

[Тег <ol>](https://html.spec.whatwg.org/multipage/grouping-content.html#the-ol-element) - створює нумерований (упорядкований) список, тобто кожен елемент списку пронумерований. Браузер автоматично нумерує елементи по порядку і, якщо видалити один або кілька елементів такого списку, інші номери будуть автоматично перераховані. Нумерацією елементів можна управляти за допомогою спеціальних атрибутів списку.

Використовується для перерахування дій в певному порядку, наприклад рецепт.

[Тег <ul>](https://html.spec.whatwg.org/multipage/grouping-content.html#the-ul-element) - створює маркований (невпорядкований) список, кожен елемент якого починається з невеликого символу (маркера). За допомогою CSS маркер можна прибрати або замінити.

Використовується для перелічення набору у довільному порядку, наприклад список курортів.

### **Вкладені списки**[**​**](https://goit.global/textbooks/marathons/html-css/uk/docs/text/#%D0%B2%D0%BA%D0%BB%D0%B0%D0%B4%D0%B5%D0%BD%D1%96-%D1%81%D0%BF%D0%B8%D1%81%D0%BA%D0%B8)

### Зробити багаторівневий список просто - вкладаємо в елемент списку ще один список. Це часто застосовується для створення багаторівневих меню.



**Посилання**[**​**](https://goit.global/textbooks/marathons/html-css/uk/docs/text/#%D0%BF%D0%BE%D1%81%D0%B8%D0%BB%D0%B0%D0%BD%D0%BD%D1%8F)

[Тег <a>](https://html.spec.whatwg.org/multipage/text-level-semantics.html#the-a-element) - призначений для створення посилань, тексту, клікаючи на який, переходимо на іншу сторінку, завантажуємо файл тощо. Текст посилання відображається в браузері з підкресленням, колір шрифту синій, при наведенні на посилання, курсор миші змінює вигляд.

Адреса посилання задається в обов'язковому атрибуті href="адреса". Адреса - це [URL](https://semantica.in/blog/chto-takoe-url.html),яка може вказувати на сторінку, файл, будь-який ресурс.

**<a href="https://google.com">Посилання на головну сторінку Google</a>**

### Атрибути target і rel[​](https://goit.global/textbooks/marathons/html-css/uk/docs/text/#%D0%B0%D1%82%D1%80%D0%B8%D0%B1%D1%83%D1%82%D0%B8-target-%D1%96-rel)

За замовчуванням посилання відкривається в поточній вкладці браузера. Атрибут target="значення" визначає те, в який вкладці повинен відкриватися документ, на який веде посилання. Якщо значення - \_blank, сторінка відкривається у новій вкладці браузера.

Атрибут rel доповнює атрибут href інформацією про відношення між поточним і пов'язаним документом, і використовується в парі з target="\_blank".

### Атрибут download[​](https://goit.global/textbooks/marathons/html-css/uk/docs/text/#%D0%B0%D1%82%D1%80%D0%B8%D0%B1%D1%83%D1%82-download)

Якщо в href посиланні зазначений шлях до файлу, браузер спробує його відкрити у поточному вікні, якщо вміє обробляти файли цього типу. Зазвичай так відбувається з зображеннями і PDF-файлами.

Атрибут download="ім'я файлу" повідомляє браузеру, що зазначений ресурс не повинен відкриватися, а завантажуватися в момент, коли користувач клікне на посилання. Значення атрибута задає ім'я файлу, що завантажується, тобто можна змінити ім'я під час завантаження. Атрибут можна використовувати без вказівки значення, тоді буде використано ім'я оригінального файлу.

### Спеціальні значення href[​](https://goit.global/textbooks/marathons/html-css/uk/docs/text/#%D1%81%D0%BF%D0%B5%D1%86%D1%96%D0%B0%D0%BB%D1%8C%D0%BD%D1%96-%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%BD%D1%8F-href)

Посилання мають можливість не тільки переходити на інші сторінки і завантажувати файли, але й здійснювати дзвінки на телефони, відправляти повідомлення або телефонувати по скайпу.

***<!-- Посилання на телефонний номер -->***

**<a href="tel:+14251234563">+1 (425) 123-45-63</a>**

***<!-- Посилання на адресу електронної пошти -->***

**<a href="mailto:example@mail.ua">example@mail.ua</a>**

**Вставляння малюнків**

Щоб вставити малюнок на вебсторінку, необхідно скористатися тегом **<img src = "my.jpg">**, де **img**— ім’я тегу, призначеного для вставляння зображення,**src = "my.jpg"** — атрибут, в якому вказується ім’я файлу з малюнком.

В атрибуті можна використовувати імена графічних файлів із розширеннями *.jpg, .jpеg, .gif, .png*або *.bmp*. Якщо малюнок і HTML-файл містяться в різних папках, то, крім імені файлу, потрібно зазначати шлях до нього.

**Додавання гіперпосилань**

Сайт може складатись із багатьох пов’язаних гіперпосиланнями вебсторінок, одна з яких є головною. Головна вебсторінка відкривається першою.

Для створення гіперпосилань на вебсторінки сайту та зовнішні ресурси мережі інтернет використовують тег **<a>...</a>**.

Атрибут **href**визначає URL-адресу сторінки на яку вказує посилання. Оскільки в якості адреси посилання може використовуватися документ будь-якого типу, то результат переходу за посиланням залежить від кінцевого файлу. За замовчуванням новий документ завантажується в поточне вікно браузера, однак це властивість можна змінити за допомогою атрибута **target**. Якщо атрибут **href**відсутній, то теґ **<a>** не є гіперпосиланням.

Нехай в одній папці з файлом *index.html* створено файл **prf.html,** який містить вашу фотографію. У файлі*index.html* до фрази *Подивитися фото* можна додати гіперпосилання, яке матиме такий вигляд:  
**<a href = "prf.html"> Подивитися фото </a>**

Гіперпосилання можна додати до малюнка. Для цього треба вказати тег вставляння малюнка:  
**<a href = "prf.html"><img src = "foto1.jpg"></a>.**

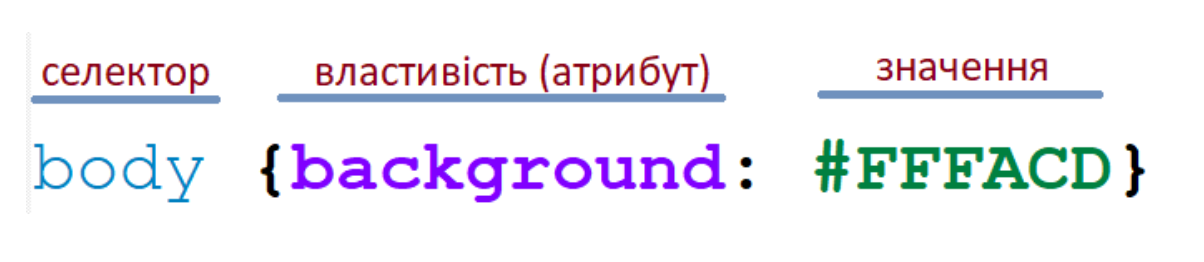
**5.3 Каскадні таблиці стилів CSS та їх структура**

Стилем або CSS (Cascading Style Sheets, каскадні таблиці стилів) називається набір параметрів форматування, який застосовується до елементів документа, щоб змінити їх зовнішній вигляд.

Можливість роботи зі стилями здавна включають в розвинені видавничі системи і текстові редактори, тим самим дозволяючи одним натисканням кнопки надати тексту заданий, заздалегідь встановлений вид. Тепер це доступно і творцям сайту, коли колір, розміри тексту і інші параметри зберігаються в певному місці і легко «прикручуються» до будь-якого тегу. Ще однією перевагою стилів є те, що вони пропонують набагато більше можливостей для форматування, ніж звичайний HTML.

## **Базовий синтаксис CSS**

Стильові правила записуються в своєму форматі, відмінному від **HTML**. Основним поняттям виступає **селектор** - це деяке ім'я стилю, для якого додаються параметри форматування. Як селектори виступають **теги**, **класи** і **ідентифікатори**. Загальний спосіб запису має наступний вигляд.

****

Спочатку пишеться ім'я селектора, наприклад, **BODY**, це означає, що всі стильові параметри будуть застосовуватися до тегу **<body>**, потім йдуть фігурні дужки, в яких записується стильова властивість, а її значення вказується після двокрапки. Стильові властивості поділяються між собою крапкою з комою, в кінці цей символ можна опустити.

CSS не чутливий до регістру, перенесення рядків, пробілів і символів табуляції, тому форма запису залежить від бажання розробника.

## **Глобальні стилі**

При використанні глобальних стилів властивості **CSS** описуються в самому документі **html** і розташовуються в заголовку веб-сторінки. Цей спосіб додавання стилю дозволяє зберігати стилі в одному місці, в даному випадку прямо на тій же сторінці за допомогою контейнера **<style></style>** .

## **Пов'язані (зовнішні) стилі**

При використанні пов'язаних стилів опис селекторів та їх значень розташовується в окремому файлі, як правило, з розширенням **.css**, а для зв'язування документа з цим файлом застосовується тег **<link>**. Даний тег поміщається в контейнер **<head></head>**.

**<link rel="stylesheet" type="text/css" href="style.css">**

Значення атрибута тега **<link>** - **rel** залишається незмінним незалежно від коду. Значення **href** задає шлях до CSS-файлу, він може бути заданий як відносно, так і абсолютно. Зауважте, що таким чином можна підключати таблицю стилів, яка знаходиться на іншому сайті.

При застосуванні стилів **css** форматування елементів перенесено у окремий файл **style.css**, а в html-документі залишилася лише розмітка за допомогою тегів. При чому і два останні абзаци також відформатовані, тому що правила, записані для тега **<p>** в файлі css, впливають на кожен цей тег в файлі html.

**5.4 Спадковості і каскадування за допомогою CSS**

**Ієрархічне дерево**

HTML-документ являє собою ієрархічне дерево. Це означає, що у кожного елемента (крім кореневого) є тільки один батько, тобто елемент, усередині якого він розташовується. У кореневого розділу батько відсутній. Розглянемо найпростішу сторінку:

<html>

<head> </ head>

<body>

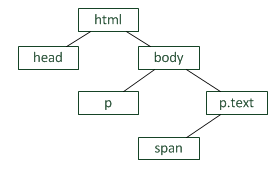
<p> Текст документа </ p>

          <p class = "text"> Виділена <span> рядок </ span> </ p>

</ body>

</ html>

Для цієї сторінки можна намалювати таке ієрархічне дерево:



Воно схематично відображає структуру вкладеності елементів. В даному прикладі видно, що в елемента span батьком є p.text, а у p.text батько - Body. Ієрархічна структура документа визначає основи концепції наслідування.

**Наслідування**

Наслідування в CSS - механізм, за допомогою якого значення властивостей елемента-батька передаються його елементам-нащадкам. Стилі, присвоєні деякого елементу, успадковуються всіма нащадками (вкладеними елементами), якщо вони не перевизначені (перекриті) явно. Наприклад, розмір шрифту і його колір досить часто застосувати до body, щоб всі елементи всередині мали ті ж властивості. Наслідування дозволяє скоротити розмір таблиці стилів, але якщо стилів багато, то відстежити який батьківський елемент встановив певну властивість стає складніше.

**Наслідування властивості**

До спадкоємною властивостей відносяться в першу чергу властивості, що визначають параметри відображення тексту:

font-size, font-family, font-style, font-weight, color, text-align, text-transform, text-indent, line-height, letter-spacing, word-spacing, white-space, direction і т. д.

Також до спадкоємною властивостей відносяться list-style, cursor, visibility, border-collapse і деякі інші. Але вони використовуються значно рідше. Весь список успадкованих властивостей дивіться в [стандарті CSS.](https://translate.google.com/translate?hl=ru&prev=_t&sl=ru&tl=uk&u=http://www.w3.org/TR/CSS21/propidx.html) Значення yes в колонці Inherited?.

Ці властивості можна і потрібно задавати через предків, слідуючи семантиці документа. Наприклад, параметри тексту часто не змінюються в межах окремих блоків сторінки: меню, основного змісту, інформаційних панелей. Тому загальні параметри тексту (колір, розмір, гарнітура) зазвичай вказують в стилях самих блоків.

**Ненаслідувані властивості**

Вище ми перерахували основні наслідувані властивості. Всі інші відносяться до ненаслідуваних. Це параметри позиціонування, розмірів, відступів, фону, рамок і т. д. А саме: background, border, padding, margin, width, height, position і ін. Весь список ненаслідуваних властивостей дивіться в [стандарті CSS.](https://translate.google.com/translate?hl=ru&prev=_t&sl=ru&tl=uk&u=http://www.w3.org/TR/CSS21/propidx.html) Значення no в колонці Inherited?. Ці властивості не унаслідуються з міркувань здорового глузду. Наприклад: якщо для будь-якого блоку встановлений внутрішній відступ, автоматично виставляти такий же відступ кожному вкладеному елементу немає ніякої потреби. Ці параметри найчастіше унікальні для кожного окремого блоку.

**Примусове  наслідування**

Для кожної властивості може бути задано значення inherit. Воно означає, що дана властивість приймає таке ж значення, що і у батьківського елементу. Значення inherit може бути використано для посилення наслідуваних значень, а також у властивостях, які зазвичай не унаслідуються.

**Каскадність**

CSS розшифровується як «Cascading Style Sheets» або «каскадні таблиці стилів». Каскадність означає, що до одного і того ж елементу може застосовуватися кілька CSS-правил (наборів CSS-властивостей). Серед цих властивостей можуть бути і конфліктуючі між собою. Тому існують інструкції, які визначають, яким буде фінальний набір властивостей елемента. Наприклад, для елемента:

**<p class = "text" style = "color: red;"> </ p>**

CSS-правила існують як мінімум в трьох різних місцях:

1. в файлі, що підключається style.css для селекторів p або .text;

2. в атрибуті style;

3. в стандартних стилях відображення, вбудованих в браузер.

Каскадність якраз і визначає, які саме властивості з цих джерел застосуються до даного абзацу. Є три основні концепції, керуючі порядком, в якому застосовуються CSS-властивості:

1. важливість;

2. специфічність;

3. порядок вихідного коду.

**Спадковість**

Спадковість у CSS означає, що деякі властивості стилю, задані для батьківського елемента, автоматично передаються його дочірнім елементам. Це допомагає скоротити код і забезпечити узгодженість стилів на веб-сторінці.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Спадковість у CSS</title>

<style>

body {

font-family: Arial, sans-serif;

color: navy;

}

p {

margin: 20px;

}

</style>

</head>

<body>

<p>Цей текст наслідує колір та шрифт від body.</p>

<div>

<p>Цей текст теж наслідує колір та шрифт від body.</p>

</div>

</body>

</html>

У цьому прикладі властивості font-family та color, задані для <body>, наслідуються всіма <p> елементами на сторінці.

**Специфічність**

Специфічність визначає пріоритет стилів, якщо для одного елемента задано кілька різних правил CSS. Специфічність обчислюється на основі типів селекторів, що використовуються:

* Вбудовані стилі (inline) мають найвищий пріоритет.
* Ідентифікатори (#id) мають вищий пріоритет, ніж класи (.class), псевдокласи і атрибути.
* Класи, псевдокласи та атрибути мають вищий пріоритет, ніж селектори елементів (теги).

**Опис постановки задачі**

Основною задачею було зрозуміти, як влаштований HTML та таблиця стилів CSS, навчитися розміщати та форматувати текст, посилання та зображення в HTML, навчитися створювати свою Веб-сторінку та працювати над покращенням її зовнішнього вигляду.

**Розроблена веб-сторінка**

HTML-код:

<!DOCTYPE html> #Kapychnikov Yegor

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewpirt" content="width=device-width,minitial-scale=1.0">

<title>Kapychnikov Yegor</title> #Заголовок файлу

<link rel="stylesheet" href="practicproj.css" /> #Посилання на зовнішній файл CSS

</head>

<body> #«Тіло»(основна частина) коду

<div class="container"> #Застосування стилю

<header class="imagine"> #Застосування стилю та виділення верхньої частини сторінки

<div>

<img src="my\_photo.jpg" alt="" id="centered"> #Додавання зображення

</div>

<div>

<h1>Kapichnikov Yegor</h1>

<h1>Student of second course</h1>

</div>

</header>

<h1>Summary</h1>

<hr> #Додавання горизонтальної лінії

<p>

I'm beginner specialist in the field of computer networks, databases, servers. I'll be student on that faculty next three years, and I expect to be professional specialist in this sphere.

</p> #Додавання параграфу з текстом

<p>

Also have experience in programming on python and C++ languages. I'm trying hard to manage to do the work on time, to be responsible and well organised, to have a serious immersion in the task.

</p>

<h1>Education</h1>

<hr />

<ul> #Ненумерований список

<li>

<h2>2013-2022 basic secondary education</h2> <br /> specialized school with in-depth study of foreign languages №152

</li>

<li>

<h2>2022-until now Junior Bachelor</h2> <br /> Taras Shevchenko Technical and Mechanical Vocational College <br /> Specialty: 121 Software engineering

</li>

</ul>

<h1>Experience</h1>

<hr />

<p>

<h2>Python-developer(January2024-April2024)</h2> <br /> Telegram bot development. Written in Python, it integries the mp4 file from Youtube to mp3.

</p>

<footer> #Нижня частина веб-сторінки

<ul>

<li><div>

<a href="mailto:kapichnikov.egor@gmail.com" target="\_blank">Email <img src="mail.gif" class="im" /></a> #Додавання посилання

</div></li>

</ul>

<br /> #Перенос на наступний рядок

<ul>

<li>

<a href="https://www.instagram.com/ekapyk/" target="\_blank">Instagram <img src="instagram.gif" class="im" /></a>

</li>

</ul>

</footer>

</div>

</body>

</html>

CSS-файл:

ul {

list-style-type: none;

}

a { #Стиль елемента коду з посиланнями

border: 3px solid lawngreen;

border-radius: 20px;

margin: 0 auto;

padding: 11px;

text-decoration: none;

text-align: center;

width: 500px;

height: 30px;

display: block;

}

a:hover, a:visited { #Зміна кольору при наведені курсора на посилання

background-color: lawngreen;

}

.container { #Стиль усієї ділянки Веб-сторінки

width: 1570px;

background-color: mistyrose;

margin: 0 auto;

}

.im { #Клас зображення

height: 20px;

}

h1 { #Разом з h2 – стилі заголовків

color: black;

}

h2 {

color: black;

}

.imagine { #Стиль зображення

text-align: center;

}

#centered { #id для зображення

border-radius: 100%;

height: 180px;

width: 180px;

}

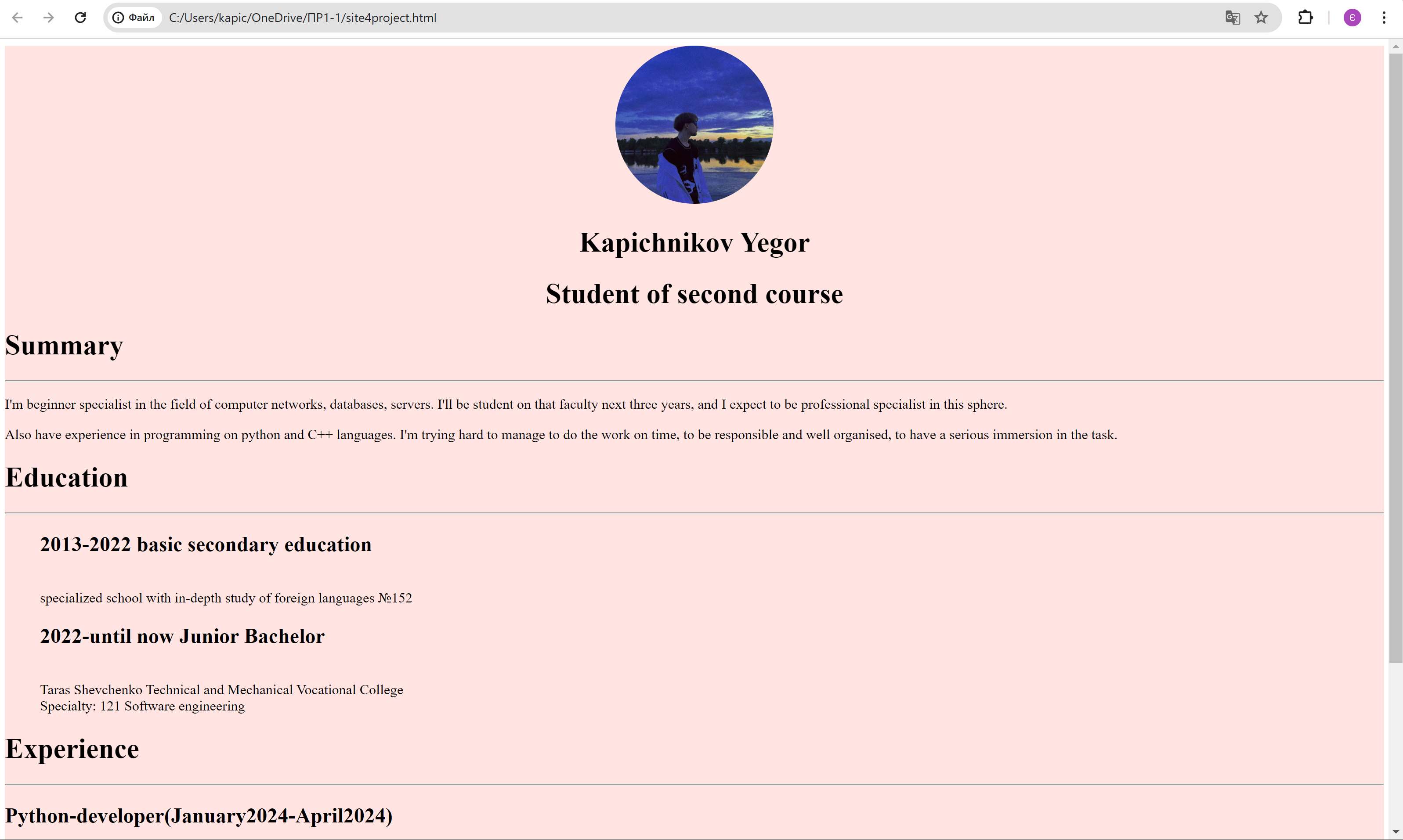
.center { #Вирівнювання по центру

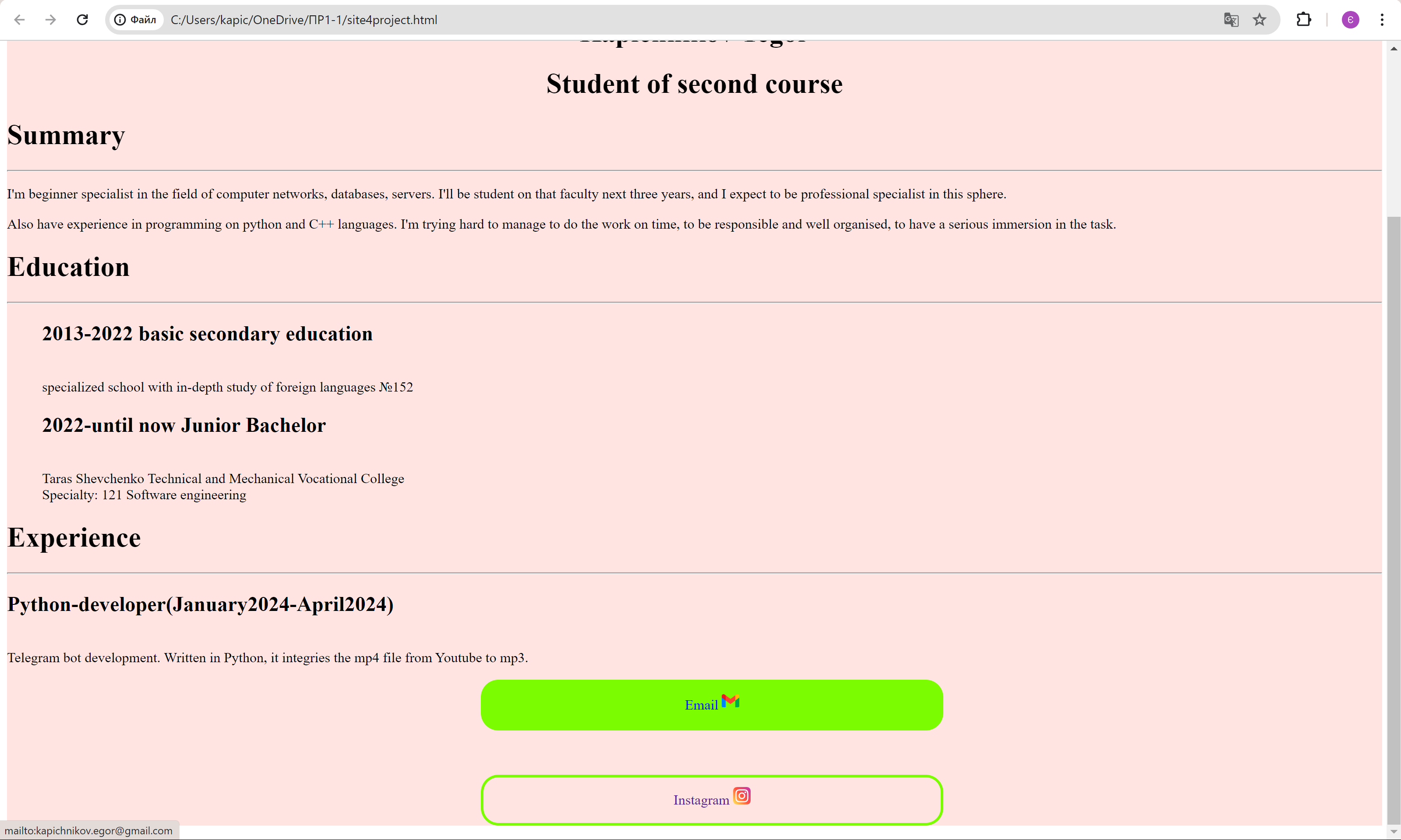
text-align: center;

color: black

}

**Перевірка/тестування веб-сторінки**





**Висновок**

Під час цієї навчальної практики я навчився створювати свою Веб-сторінку за допомогою HTML, форматувати та розміщувати текст, додавати посилання, зображення та стилі, і за допомогою стилі впливати на зовнішній вигляд Веб-сторінки.

Завдяки цій практиці в мене тепер є навички роботи з HTML-файлами, бо тепер я вмію правильно читати код, розуміти, яка строчка коду на що впливає. Відтепер мені набагато легше розуміти код, завдяки чому я покращую свої навички у програмуванні.

Цей досвід мені точно знадобиться у майбутньому тому, що Веб-сайти завжди матимуть великий попит в IT-сфері, і я вже на власному досвіді переконався, що ця практика була вивчена не дарма.

**Список використаних джерел**

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

2. «Мій клас», Лекція 1. Поняття про HTML. [https://www.miyklas.com.ua/p/informatica/8-klas/stvorennia-ta-publikatciia-veb-resursiv-444304/poniattia-pro-movu-gipertekstovoyi-rozmitki-444805/re-1a3f7169-2c1e-4c20-a0c8-a728b4727791#:~:text=HTML%20(%D0%B0%D0%BD%D0%B3%D0%BB.,%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20%D0%B4%D0%BB%D1%8F%20%D1%81%D1%82%D0%B2%D0%BE%D1%80%D0%B5%D0%BD%D0%BD%D1%8F%20%D0%B3%D1%96%D0%BF%D0%B5%D1%80%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D0%B8%D1%85%20%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D1%96%D0%B2](https://www.miyklas.com.ua/p/informatica/8-klas/stvorennia-ta-publikatciia-veb-resursiv-444304/poniattia-pro-movu-gipertekstovoyi-rozmitki-444805/re-1a3f7169-2c1e-4c20-a0c8-a728b4727791#:~:text=HTML%20(%D0%B0%D0%BD%D0%B3%D0%BB.,%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20%D0%B4%D0%BB%D1%8F%20%D1%81%D1%82%D0%B2%D0%BE%D1%80%D0%B5%D0%BD%D0%BD%D1%8F%20%D0%B3%D1%96%D0%BF%D0%B5%D1%80%D1%82%D0%B5%D0%BA%D1%81%D1%82).

3. «Studfiles» – файловий архів студентів. <https://studfile.net/preview/8904866/page:33/#94>

4. [Інформатика+ - Сайт учителя Сімакової Олени](https://sites.google.com/view/computer-science-teach/%D0%B2%D0%B5%D0%B1-%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D1%96%D1%97?authuser=0)

5. Е-helper.com.ua «Лекція 5. Наслідування та каскадування. Оформлення тексту за допомогою CSS»/ <http://e-helper.com.ua/node/952>