ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ  
«ОПТИКО-МЕХАНІЧНИЙ ФАХОВИЙ КОЛЕДЖ КИЇВСЬКОГО  
НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ІМЕНІ ТАРАСА ШЕВЧЕНКА»  
Циклова комісія програмування та інформаційних технологій

**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:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис) | Виконав:  студент 2-го курсу групи ІПЗ-23  *Костенков Глєб Анатолійович*  залікова книжка № \_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис виконавця)  « \_» 20 року |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис) |  |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис) |  |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис) |  |

« \_»\_ 2024 року

КИЇВ

ЗАТВЕРДЖУЮ

Директор ОМФК КНУ

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

« » 20 р.

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

на період *Навчальної практики*

Костенкова Глєба Анатолійовича

Спеціальність *121 Інженерія програмного забезпечення* Група *ІПЗ-23*

|  |  |  |
| --- | --- | --- |
|  | **Питання до вивчення** | **Відмітки про**  **виконання** |
| 4 | Розділ "Практична робота з С++" |  |
| 4.1 | Введення та виведення у базових консольних програмах |  |
| 4.2 | Робота з файлами: запис та читання |  |
| 4.3 | Формати файлів та їх обробка |  |
| 4.4 | Системи контролю версій (VCS) |  |
| 4.5 | Мова Markdown для опису проектів |  |
| 4.6 | Створення ігрової програми |  |
| 5 | Розділ ”Веб-дизайн та HTML” |  |
| 5.1 | Мова розмітки HTML і 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 |  |
| 6 | Систематизація матеріалів, оформлення звітів і залік з навчальної практики |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Голова циклової комісії |  |  |  |
|  | (підпис) | (ініціали, прізвище) | (дата) |
| Керівники практики від коледжу |  |  |  |
|  | (підпис) | (ініціали, прізвище) | (дата) |
|  |  |  |  |
|  | (підпис) | (ініціали, прізвище) | (дата) |
|  |  |  |  |
|  | (підпис) | (ініціали, прізвище) | (дата) |

**ЗМІСТ**

[ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 5](#_Toc167990294)

[ВСТУП 7](#_Toc167990295)

[ОСНОВНА ЧАСТИНА 9](#_Toc167990296)

[***Тема 4.1 Введення та виведення у базових консольних програмах*** 9](#_Toc167990297)

[***Тема 4.2 Робота з файлами: запис та читання*** 10](#_Toc167990298)

[***Тема 4.3 Формати файлів та їх обробка*** 12](#_Toc167990299)

[***Тема 4.4 Системи контролю версій (VCS)*** 13](#_Toc167990300)

[***Тема 4.5 Мова Markdown для опису проектів*** 15](#_Toc167990301)

[***Тема 4.6 Створення ігрової програми*** 16](#_Toc167990302)

[***Тема 5.1 Мова розмітки HTML та структура Веб-сторінки*** 18](#_Toc167990303)

[***Тема 5.2 Розмітка тексту за допомогою HTML. Зображення та посилання*** 19](#_Toc167990304)

[***Тема 5.3 Каскадні таблиці стилів CSS та їх структура*** 21](#_Toc167990305)

[***Тема 5.4 Спадковості і каскадування за допомогою CSS*** 22](#_Toc167990306)

[Опис постановки задачі 23](#_Toc167990307)

[Висновки 24](#_Toc167990308)

[***До 4 теми:*** 24](#_Toc167990309)

[***До 5 теми:*** 25](#_Toc167990310)

[Список використаних джерел 26](#_Toc167990311)

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

* <html> – позначає початок і кінець HTML-документа.
* <head> – містить метадані веб-сторінки.
* <title> – задає заголовок веб-сторінки, який відображається в заголовку браузера.
* <meta> – використовується для вказання метаданих, таких як кодування символів.
* <style> – містить CSS-стилі для веб-сторінки.
* <body> – визначає основний вміст веб-сторінки.
* <header> – визначає верхній колонтитул документа або розділу.
* <nav> – містить навігаційне меню.
* <ul> – створює ненумерований список.
* <li> – визначає елемент списку.
* <a> – створює посилання на іншу веб-сторінку або документ.
* <main> – визначає головний вміст документа.
* <section> – визначає розділ у документі.
* <h1> до <h6> – визначає заголовки різного рівня на веб-сторінці.
* <p> – створює абзац тексту.
* <br> – вставляє розрив рядка.
* <hr> – вставляє горизонтальну лінію (роздільник).
* <footer> – визначає нижній колонтитул документа або розділу.
* <table> – створює таблицю.
* <thead> – задає заголовок таблиці.
* <tbody> – задає тіло таблиці.
* <tfoot> – задає підвал таблиці.
* <tr> – створює рядок таблиці.
* <th> – створює заголовок стовпця таблиці.
* <td> – створює клітинку таблиці.
* <form> – створює форму для введення даних користувача.
* <input> – створює поле для введення даних.
* <label> – прив'язує мітку до елемента введення.
* <em> – виділяє текст курсивом.
* <b> – виділяє текст жирним.
* <i> – виділяє текст курсивом.
* <u> – підкреслює текст.
* <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.
* <textarea> – створює багаторядкове поле введення тексту.
* <button> – створює кнопку.
* <select> – створює випадаючий список.
* <option> – створює елемент списку в <select>.
* <article> – визначає самостійний фрагмент вмісту.
* <aside> – визначає побічний вміст, наприклад, бокову панель.
* <img> – вбудовує зображення в HTML-документ.
* <audio> – додає аудіо-контент.
* <source> – вказує джерело мультимедіа для <audio> або <video>.
* <video> – додає відео-контент.

# 

# 

# ВСТУП

HTML (HyperText Markup Language - «мова гіпертекстової розмітки») - стандартизована мову розмітки документів у Всесвітній павутині. Більшість веб-сторінок містять опис розмітки на мові HTML (або XHTML). Мова HTML інтерпретується браузерами; отриманий в результаті інтерпретації форматований текст відображається на екрані монітора комп'ютера або мобільного пристрою.

Мова HTML була розроблена британським вченим Тімом Бернсом-Лі приблизно в 1986-1991 роках. HTML створювалась як мова для обміну науковою і технічною документацією, призначеною для використання людьми, які не є фахівцями в області верстки. HTML успішно справлялась з проблемою складності SGML шляхом визначення невеликого набору структурних і семантичних елементів - дескрипторів. Дескриптори також часто називають «тегами». За допомогою HTML можна легко створити відносно простий, але красиво оформлений документ. Крім спрощення структури документа, в HTML внесена підтримка гіпертексту. Мультимедійні можливості були додані пізніше.

Текстові документи, що містять розмітку на мові HTML (такі документи зазвичай мають розширення .html або .htm), обробляються спеціальними додатками, які відображають документ в його форматованому вигляді. Такі додатки, що називаються «браузерами» або «інтернет-оглядачами», зазвичай надають користувачеві зручний інтерфейс для запиту веб-сторінок, їх перегляду (і виведення на інші зовнішні пристрої) і, при необхідності, відправки введених користувачем даних на сервер. Найбільш популярними на сьогоднішній день браузерами є Google Chrome, Mozilla Firefox, Opera, Internet Explorer і Safari.

# ОСНОВНА ЧАСТИНА

**Теоретичні відомості:**

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

Введення та виведення у базових консольних програмах є одним із фундаментальних аспектів програмування. Ці операції дозволяють користувачам спілкуватися з програмою та отримувати результати її роботи через консольний інтерфейс.

***1. Введення даних:*** У мові C++ для цього використовується об'єкт std::cin, який дозволяє отримати дані з клавіатури. Наприклад:

#include <iostream>

int main() {

int number;

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

std::cin >> number;

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

return 0;

}

У цьому прикладі користувач вводить число з клавіатури, а програма виводить його назад у консоль.

***2. Виведення даних:*** Для виведення даних у консоль у мові C++ використовується об'єкт std::cout. Наприклад:

#include <iostream>

int main() {

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

return 0;

}

У цьому прикладі рядок "Hello, world!" виводиться у консоль за допомогою об'єкта std::cout.

Ці операції введення та виведення даних у консоль дозволяють створювати взаємодію між користувачем та програмою, а також надають можливість відображення результатів обчислень та роботи програми.

## ***Тема 4.2 Робота з файлами: запис та читання***

JSON (JavaScript Object Notation) - це простий формат для обміну даними, що використовується для зберігання та передачі структурованих даних між різними системами. Він базується на синтаксисі JavaScript, проте може використовуватися в різних мовах програмування. JSON використовується для представлення об'єктів та масивів даних у формі, зрозумілій як для людей, так і для програм.   
  
Основними складовими JSON є пари "ключ-значення", які можуть бути вкладені одна в одну для представлення складних структур даних. Він широко використовується для обміну даними між клієнтськими та серверними додатками, а також для зберігання налаштувань, конфігурацій та інших структурованих даних.

***Робота з файлами JSON:***

Переваги:

1) Простота читання та редагування: Формат JSON має зрозумілу структуру, яка полегшує як людям, так і комп'ютерам читання та редагування файлів JSON.

2) Підтримка багатьох мов програмування: Більшість сучасних мов мають інструменти для роботи з JSON, що робить його досить універсальним для використання.

3) Гнучкість: JSON дозволяє представляти різноманітні дані, включаючи тексти, числа, масиви та об'єкти.

4) Популярність: JSON є одним з найпоширеніших форматів обміну даними в програмуванні, що сприяє наявності багатьох інструментів та бібліотек для його використання.

Недоліки:

1)Великий обсяг даних: JSON може стати неефективним для обробки великих обсягів даних через багато зайвих символів для форматування.

2)Відсутність вбудованих засобів для роботи з великими обсягами даних: JSON не надає механізмів для індексації або пошуку, що може ускладнити роботу з великими файлами даних.

XML (Extensible Markup Language) - це мова розмітки, яка призначена для створення та обміну структурованими даними між комп'ютерними системами. Вона базується на ідеї про те, що документ повинен мати структуру, яка буде зрозуміла як людям, так і комп'ютерам.

Основою XML є синтаксис тегів, аналогічний HTML, але у відмінність від нього, XML не має попередньо визначених тегів і може бути розширений для використання в різних контекстах. XML застосовується для створення структурованих документів, таких як конфігураційні файли, програмні налаштування, обмін даними між веб-серверами та клієнтами, а також для зберігання даних в базах даних. Він є одним із основних форматів для обміну даними в сучасному програмуванні та веб-розробці.  
   
***Робота з файлами XML:***

Переваги:

1)Структурованість: XML має чітку та строго визначену структуру, що робить його ідеальним для представлення складних структур даних та документів з вкладеними елементами.

2)Можливість валідації: XML легко піддається валідації за допомогою схем XSD (XML Schema Definition), що дозволяє перевіряти коректність структури та типів даних у файлі.

3)Підтримка стандартів: XML підтримується багатьма стандартами та протоколами, що робить його важливим для обміну даними між різними системами.

Недоліки:

1) Великий обсяг даних: XML-файли можуть стати досить великими через значну кількість тегів та атрибутів, що може призвести до збільшення розміру файлу та ускладнення його обробки.

2) Складність читання та редагування: Велика кількість тегів та вкладеність може зробити XML-файли складними для читання та редагування людьми, особливо в разі обробки великих обсягів даних.

## ***Тема 4.3 Формати файлів та їх обробка***

У мові програмування C++ є кілька типових форматів файлів, які використовуються для зберігання та обробки даних. Основні з них:

1. ***Текстові файли:*** Це файли, які зберігають дані у звичайному текстовому форматі. Їхній зміст може бути прочитаний та змінений звичайним текстовим редактором. Для роботи з текстовими файлами у C++ використовуються потоки введення/виведення (ifstream та ofstream).

* Плюси: Легко зрозумілі та зручні для редагування людиною, можуть бути читані та оброблятися звичайними текстовими редакторами. Це зроблює їх ідеальним вибором для зберігання конфігураційних файлів, логів, текстових даних тощо.
* Мінуси: Можуть бути менш ефективними для зберігання великих обсягів даних через потребу у просторі для зберігання текстового представлення. Також, обробка текстових файлів може бути менш ефективною у порівнянні з бінарними файлами через необхідність конвертації даних з/у текстовий формат.

1. ***Бінарні файли:*** Ці файли містять дані у вигляді двійкового коду, що дозволяє зберігати дані більш компактно та ефективно. Для роботи з бінарними файлами у C++ також використовуються потоки введення/виведення (ifstream та ofstream), але оперування даними відбувається за допомогою функцій читання та запису в бінарному форматі.

* Плюси: Бінарні файли можуть бути більш ефективними для зберігання великих обсягів даних, оскільки вони не потребують такого великого обсягу простору для зберігання. Також, обробка бінарних даних може бути швидшою, оскільки вони не потребують конвертації в текстовий формат.
* Мінуси: Бінарні файли не придатні для читання та редагування людиною, оскільки їхній зміст представлений у вигляді двійкового коду. Це робить їх менш зручними для відладки та виправлення помилок.

1. ***Формати JSON та XML:*** JSON (JavaScript Object Notation) та XML (Extensible Markup Language) є форматами обміну даними, які широко використовуються для зберігання та передачі структурованих даних між програмами. Для роботи з цими форматами в C++ можна використовувати

* Плюси: JSON та XML дозволяють представляти структуровані дані у зручному для читання та обробки форматі. Вони є популярними форматами для обміну даними між програмами та веб-серверами через їхню розширюваність та простоту використання.
  + Мінуси: JSON та XML можуть бути менш ефективними у порівнянні з бінарними форматами у випадках, коли потрібно зберігати великі обсяги даних. Також, обробка JSON та XML може вимагати додаткових бібліотек або інструментів для парсингу та створення даних.

Кожен з цих форматів має свої переваги та недоліки і вибір конкретного формату залежить від потреб конкретного проекту та зручності роботи з ним.

## ***Тема 4.4 Системи контролю версій (VCS)***

Системи контролю версій (VCS) - це програмні засоби, які використовуються для відстеження змін у файлів і коду програм під час розробки програмного забезпечення.

Переваги:  
  
1) Історія змін: Усі внесені зміни до коду чи файлів зберігаються та доступні для перегляду. Кожна зміна супроводжується коментарем, який містить інформацію про автора та дату зміни.

2) Робота з різними версіями: Ви маєте можливість відновити попередні версії файлів або порівняти дві версії для виявлення внесених змін.

3) Гілки розробки: Ви можете створювати окремі гілки для розробки різних функціональностей або аспектів проекту паралельно.

Недоліки:

1. Вивчення*:* Використання систем контролю версій вимагає від розробників додаткового часу та зусиль для вивчення їхнього функціоналу та інтерфейсу.
2. Розмір бази даних*:* База даних VCS може займати значний обсяг пам'яті, особливо для великих проектів зі значною кількістю історії змін.
3. Конфлікти злиття: При роботі над спільними файлами можуть виникати конфлікти злиття, коли дві гілки розробки мають зміни в одному й тому ж рядку коду.

Централізовані та розподілені системи контролю версій:

* + ***Централізовані системи****:* У централізованих системах весь репозиторій (сховище з версіями файлів) зберігається на центральному сервері, а розробники отримують доступ до нього через мережу. Прикладом централізованої системи є Subversion (SVN).
  + ***Розподілені системи****:* У розподілених системах кожен розробник має свою копію повного репозиторію, з якою вони можуть працювати незалежно. Git є однією з найпопулярніших розподілених систем контролю версій.

Відгалуження та злиття (Branching and Merging):

* + ***Гілки (Branches)****:* Гілки дозволяють розробникам працювати над різними аспектами проекту одночасно без впливу на головну гілку. Кожна гілка може мати свою власну історію змін.
  + ***Злиття (Merging)****:* Злиття використовується для об'єднання змін з однієї гілки в іншу. При виявленні конфліктів злиття розробник повинен їх вирішити, переглянувши та злиттям відповідні рядки коду.

Командна робота (Collaboration):

* + Системи контролю версій дозволяють декільком розробникам працювати над одним проектом одночасно, вносячи та відстежуючи зміни, які здійснюються кожним членом команди.
  + Кожна зміна зберігається разом з інформацією про автора, дату та коментарі, що сприяє зручній співпраці та відстеженню історії розробки.

Робота з галузями (Forking):

* + Деякі системи контролю версій, такі як GitHub, дозволяють розробникам створювати власні копії проектів під назвою "форків". Це дозволяє вносити зміни та експериментувати з кодом, не впливаючи на оригінальний проект.

Відстеження проблем (Issue Tracking):

* + Деякі платформи для спільної роботи, такі як GitHub та GitLab, дозволяють створювати та відстежувати проблеми (issues) або завдання (tasks) пов'язані з проектом. Це полегшує співпрацю та координацію роботи команди розробників.

Тема 4.5 Мова Markdown для опису проектів  
  
Markdown - це проста у використанні мова розмітки, призначена для форматування тексту у веб-засобах. Часто вона використовується для створення різних документів, таких як описи проектів, README файли, блоги та коментарі на форумах. Markdown має кілька основних переваг, які роблять її популярною для опису проектів.  
  
Простота використання: Markdown - це мова розмітки, яка відрізняється своєю простотою та легкістю використання. Вона базується на зрозумілих синтаксичних правилах, таких як використання символів "\*" або "\_" для виділення тексту.

Читабельність: Текст, написаний у форматі Markdown, легко читати у вихідному коді, оскільки він є простим текстом. Він також має чистий вигляд при перегляді на веб-сторінках або в інших середовищах.

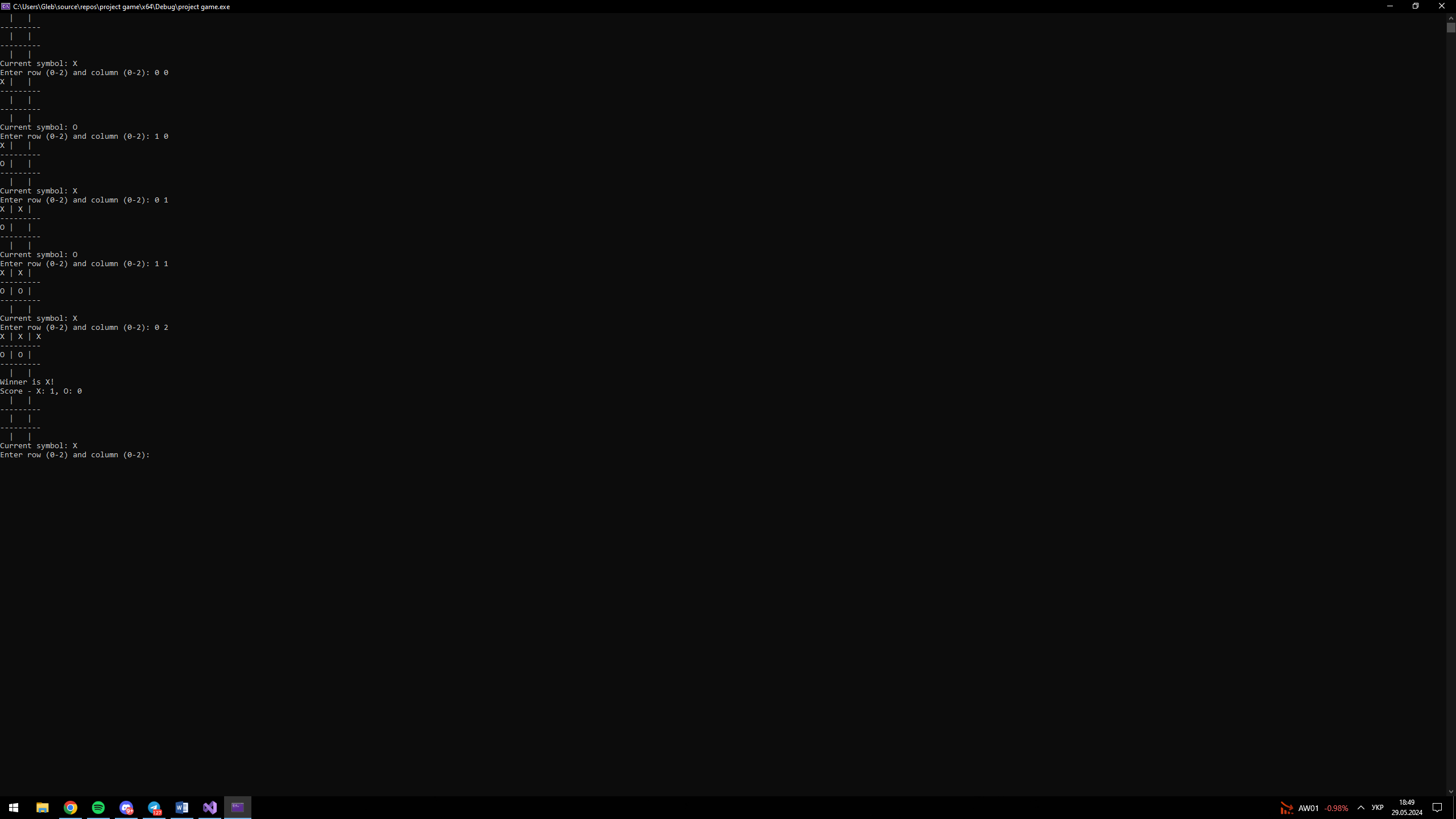
Підтримка великої кількості платформ: Багато веб-сервісів та платформ, таких як GitHub, GitLab, Bitbucket, Reddit, а також багато блог-систем, підтримують формат Markdown для форматування тексту.

Можливості розширення: Існують різні інструменти та розширення, які дозволяють розширити функціональність Markdown, додаючи підтримку для таблиць, зображень, посилань та інших елементів.

## Інтеграція з іншими мовами програмування: Markdown легко інтегрується з іншими мовами програмування, такими як HTML, CSS, а також з системами контролю версій, що дозволяє використовувати його в різних середовищах розробки. ***Тема 4.6 Створення ігрової програми***

Мій код є реалізацією гри "хрестики нолики" на мові програмування C++. В головній функції виконується основний цикл гри. Гравець вводить координати для свого ходу, після кожного ходу перевіряється наявність переможця або нічии, та виводиться відповідне повідомлення. Після закінчення гри поляризація оновлюється для нової гри. drawOneRow(int rowNumber) - виводить один рядок ігрового поля на екран, а drawBoard() - використовує drawOneRow для виведення усього ігрового поля на екран. validateInput(int row, int col) - перевіряє, чи введені координати (рядок і стовпець) знаходяться в межах поля і чи вони не зайняті, setBoardCellValue(int row, int col) - встановлює значення (символ "X" або "O") на відповідному місці на полі, checkWinner() - перевіряє, чи є переможець в поточному стані поля, endGame(const string& winner) - оголошує переможця, оновлює лічильники перемог та очищає поле для наступної гри.

Ця програма створює просте текстове представлення гри "Хрестики-нолики" і дозволяє гравцям змагатися один з одним.

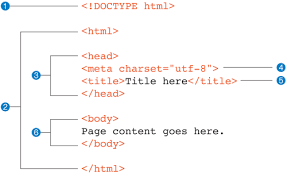


## ***Тема 5.1 Мова розмітки HTML та структура Веб-сторінки***

HTML — це мова розмітки , яка визначає структуру вмісту. HTML складається з ряду елементів , які використовують для включення або обгортання різних частин вмісту, щоб він виглядав певним чином або діяв певним чином. Об’ємні теги можуть створювати гіперпосилання на слово чи зображення в іншому місці, виділяти слова курсивом, збільшувати або зменшувати шрифт тощо.

Кожен HTML-документ, який відповідає специфікації HTML будь-якої версії, повинен починатися з рядка оголошення версії HTML <!DOCTYPE ...>, який зазвичай виглядає приблизно так: <!DOCTYPE html>.

* <html></html>— <html>елемент. Цей елемент огортає весь вміст усієї сторінки та іноді відомий як кореневий елемент. Він також містить langатрибут, що встановлює основну мову документа.
* <head></head>— <head>елемент. Цей елемент діє як контейнер для всього того, що ви хочете включити на сторінку HTML, але не є вмістом, який ви показуєте глядачам вашої сторінки. Це включає такі речі, як ключові слова та опис сторінки, які ви хочете відображати в результатах пошуку, CSS для стилізації нашого вмісту, оголошення набору символів тощо.
* <meta charset="utf-8">— Цей елемент встановлює набір символів у вашому документі на UTF-8, який включає більшість символів переважної більшості письмових мов.
* <title></title>— <title>елемент. Це встановлює назву вашої сторінки, яка відображається на вкладці веб-переглядача, у яку завантажується сторінка.
* <body></body>— <body>елемент. Тут міститься весь вміст, який ви хочете показати веб-користувачам, коли вони відвідують вашу сторінку, будь то текст, зображення, відео, ігри, звукові доріжки для відтворення чи будь-що інше.

 За своїм призначенням та дією елементи HTML можна розділити на три категорії: *структурні, презентаційні* та *елементи для інтерактивної взаємодії з користувачем.*

*Структурні.* Елементи структурної розмітки застосовуються для опису семантики тексту та задання структури документу. В свою чергу, в залежності від змісту документа, структурні елементи поділяються на: текст, списки, таблиці, гіперпосилання та додаткові об'єкти (зображення, аплети та ін). Вони не зазначають ніякого спеціального (візуального) відтворення тексту, проте більшість браузерів мають стандартні стилі форматування для кожного елемента.

*Презентаційні*. Елементи презентаційної розмітки застосовуються задля опису зовнішнього вигляду документа та не зазначають при цьому функцій тексту, до якого вони застосовуються. Більшість з цих елементів є такими, що не рекомендується застосовувати у розмітці. В залежності від свого змісту, елементи презентаційної розмітки поділяють на: елементи додавання каскадних таблиць стилів, візуальні елементи тексту (шрифти, кольори, насиченість та нахил щрифту, лінії та інші).

*Елементи інтерактивної взаємодії.* Елементи інтерактивної взаємодії дозволяють користувачам впливати на вміст або вигляд документу. Вони поділяються на два види: форми для введення данних користувачами та скрипти для активних документів.

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

Щоб реалізувати семантичну розмітку, HTML надає спеціальні теги, які можна використовувати для представлення таких розділів, наприклад:

* заголовок: <header> .
* панель навігації: <nav> .
* основний вміст:з різноманітними підрозділами вмісту, <main>представленими елементами , і .<article><section><div>
* бічна панель: <aside> ; часто розміщують всередині <main>
* колонтитул: <footer>

Абзаци можна утворити завдяки тегу <p>. За замовчуванням абзаци починаються з нового рядка і мають вертикальні відступи, якими можна управляти за допомогою стилів. Але використовувати їх можна тільки всередині тега <body>.

Для заголовків в 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.

Основні елементи для переносу та розділювання рядка: <br>, <hr>,

<br> створює розрив рядка в абзаці; це єдиний спосіб створити жорстку структуру в ситуації, коли вам потрібна серія фіксованих коротких рядків, наприклад, у поштовій адресі чи вірші. Без <br>елементів абзац просто відображатиметься в одному довгому рядку.

<hr>елементи створюють горизонтальну лінію в документі, яка позначає тематичну зміну в тексті (наприклад, зміну теми чи сцени). Візуально це просто виглядає як горизонтальна лінія.

Впорядкований список створюється за допомогою тега <ol> (ordered list), який може містити всередині себе теги <li>. Якщо елементи неупорядкованого списку за замовчуванням позначаються маркерами, то елементи впорядкованого списку - нумеруються.

Невпорядковані (або марковані) списки створюються за допомогою тега <ul>, який може містити всередині себе теги <li>, що позначають «елемент списку». Єдине, що дозволено в межах розмітки маркованого списку (тобто між відкриваючим і закриває тегами ul), - це один або кілька елементів списку. Ви не можете вставити туди інші елементи, і там не може бути нерозміченого тексту.

Посилання створюються за допомогою тегу <a>

Щоб перетворити виділений текст на посилання, помістіть його між відкриваючим і закриваючим тегами <а> та </а> і за допомогою атрибуту href вкажіть URL-адресу сторінки, на яку буде здійснюватися перехід. Вміст елемента якоря стає гіпертекстовим посиланням. Тег <a> без адреси не використовується, оскільки він описує посилання яке нікуди не веде. Також Посилання можна робити не лише за допомогою тексту, але і за допомогою зображень. Для цього потрібно обернути тег <img> в тег <a>.

Зображення додаються за допомогою тегу <img>

Щоб додати на сторінку зображення, потрібно використовувати одиночний тег <img> (image) з атрибутом src, в якому вказана адреса картинки. Щоб управляти шириною або висотою зображення, потрібно використовувати атрибути width і height. Найпоширенішими форматами зображень у мережі є JPEG і PNG.

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

Каскадні таблиці стилів — це набір параметрів форматування, який застосовується до елементів веб-сторінки для управління їх виглядом і положенням.

CSS можна додати до елементів HTML трьома способами:

* Inline (вбудований або рядковий) - використовуючи атрибут style в HTML елементах
* Internal (внутрішній) - використовуючи елемент <style> в разділі <head>
* External (зовнішній) - за допомогою зовнішнього файлу CSS

Властивість CSS backgraund є скороченням ряду фонових довгописних властивостей. А властивість background-color визначає колір фону для будь-якого елемента в CSS. Вона приймає будь-які дійсні <color>. A background-color розширюється під полем вмісту та заповнення елемента. Також можна додавати фонові зображення за допомогою властивість background-image яка дозволяє відображати зображення на фоні елемента.

Властивість background-repeat використовується для керування поведінкою мозаїки зображень. Доступні значення:

* no-repeat— повністю припинити повторення фону.
* repeat-x— повторити по горизонталі.
* repeat-y— повторити вертикально.
* repeat— за замовчуванням; повторити в обох напрямках.
* space— повторіть стільки разів, скільки можливо, додаючи пробіл між зображеннями, якщо є додатковий простір.
* round— схожий на space, але розтягує зображення, щоб заповнити додатковий простір

## ***Тема 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.

<html>

<head> </ head>

<body>

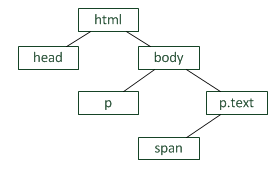
<p> Текст документа </ p>

         <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 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.

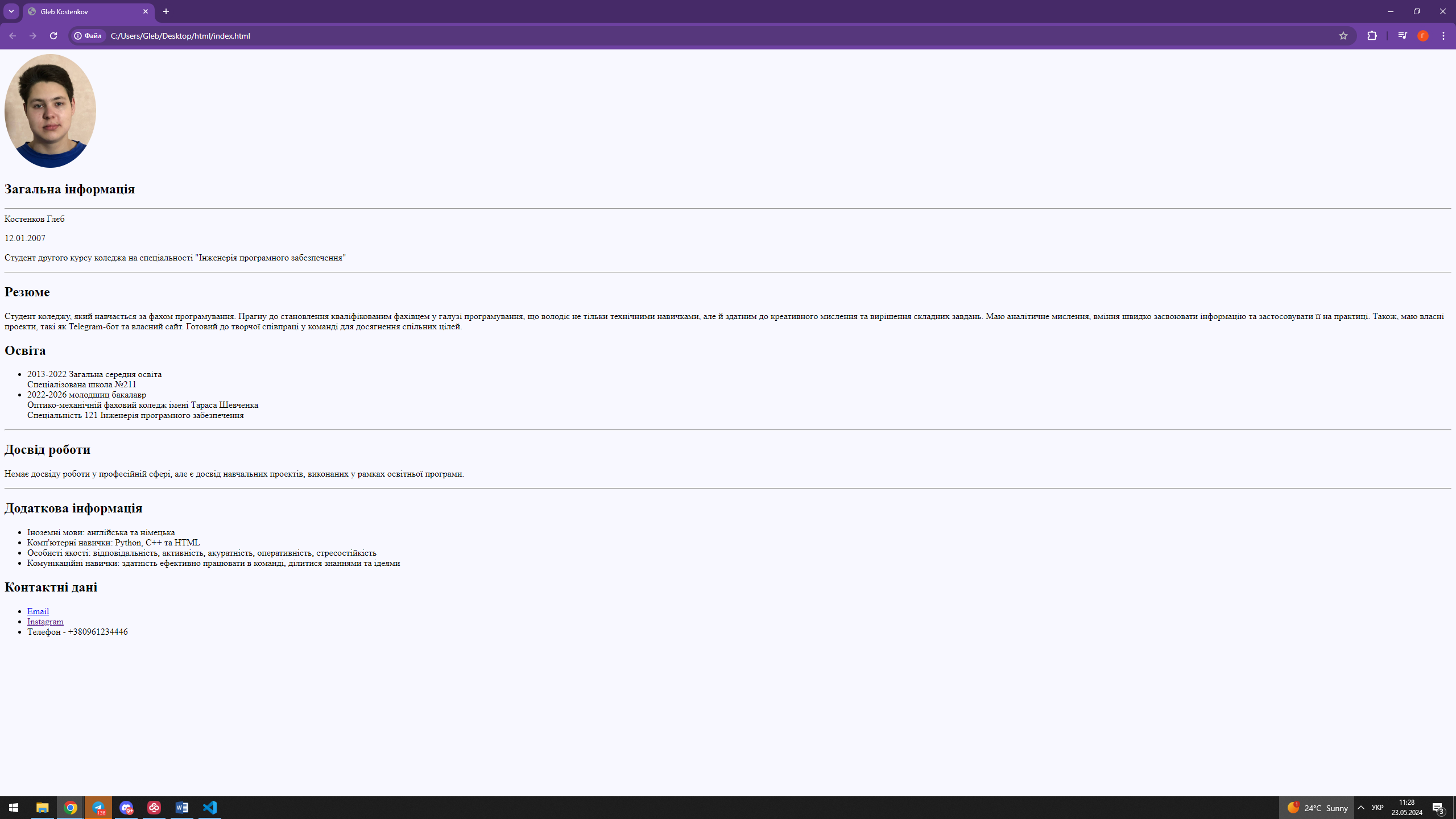
Каскадність якраз і визначає, які саме властивості з цих джерел застосуються до даного абзацу. Є три основні концепції, керуючі порядком, в якому застосовуються CSS-властивості:

1. важливість;
2. специфічність;
3. порядок вихідного коду.

# Опис постановки задачі

Основною задачею було створити веб-сайт у вигляді резюме. Метою цього завдання було вивчення та розбір мови розмітки HTML та структури Веб-сторінки, знайомство з CSS та каскадуванням стилів, а також вивчення та застосування їх основних функцій.

**Перевірка/тестування веб-сторінки**



**Розроблена веб-сторінка**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Gleb Kostenkov</title>

    <style>

        body {

            background-color: ghostwhite;

        }

    </style>

</head>

<body>

    <img src="images/Gleb Kostenkov.jpg" alt="" height="200" style="border-radius: 50%;" />

    <h2>Загальна інформація</h2>

    <hr>

    Костенков Глєб

    <p>12.01.2007</p>

   <p> Студент другого курсу коледжа на спеціальності "Інженерія програмного забезпечення"</p>

   <hr>

   <h2>Резюме</h2>

   <p>Студент коледжу, який навчається за фахом програмування. Прагну до становлення кваліфікованим фахівцем у галузі програмування, що володіє не тільки технічними навичками, але й здатним до креативного мислення та вирішення складних завдань.

    Маю аналітичне мислення, вміння швидко засвоювати інформацію та застосовувати її на практиці. Також, маю власні проекти, такі як Telegram-бот та власний сайт. Готовий до творчої співпраці у команді для досягнення спільних цілей.

   </p>

    <h2>Освіта</h2>

    <ul>

         <li style="">2013-2022 Загальна середня освіта <br> Спеціалізована школа №211 </li>

         <li>2022-2026 молодшиц бакалавр <br> Оптико-механічній фаховий коледж імені Тараса Шевченка <br> Спеціальність 121 Інженерія програмного забезпечення </li>

    </ul>

    <hr>

    <h2>Досвід роботи</h2>

    <p>Немає досвіду роботи у професійній сфері, але є досвід навчальних проектів, виконаних у рамках освітньої програми.</p>

    <hr>

    <h2>Додаткова інформація</h2>

    <ul>

        <li> Іноземні мови: англійська та німецька</li>

        <li> Комп'ютерні навички: Python, C++ та HTML </li>

        <li> Особисті якості: відповідальність, активність, акуратність, оперативність, стресостійкість</li>

        <li> Комунікаційні навички: здатність ефективно працювати в команді, ділитися знаннями та ідеями</li>

    </ul>

    <h2>Контактні дані</h2>

    <ul>

        <li> <a href="mailto:kostenkov3003@gmail.com" target="\_blank">Email</a></li>

        <li><a href="https://www.instagram.com/untiledex/" target="\_blank">Instagram</a></li>

        <li> Телефон - +380961234446 </li>

    </ul>

</body>

</html>

# Висновки

## ***До 4 теми:***

Під час виконання навчальної практики у розділі "Практична робота з С++" я набув важливі практичні навички, які будуть сприяти подальшому розвитку у сфері програмування.  
Основні аспекти які я вивчив під час цієї практики:  
  
1) Введення та виведення у консольних програмах:

Навчився працювати з введенням та виведенням даних через консоль за допомогою стандартних бібліотек C++. Це базова навичка , яка дозволяє створювати зрозумілі інтерфейси в консольних додатках.

2) Робота з файлами: запис та читання:

Оволодів навичками роботи з файлами, зокрема збереження та обробку даних у зовнішніх файлах. Це корисно для зберігання даних між сеансами програми.

3) Робота з форматами файлів:

Ознайомився з форматами JSON та XML, що дозволяє структуровано зберігати та обмінюватися даними між різними системами.

4) Системи контролю версій (VCS):

Вивчив принципи управління версіями коду та співпрацю в команді через систему контролю версій Git.

5) Markdown для документації:

Дізнався про мову Markdown, яка дозволяє створювати зрозумілу та структуровану документацію для проектів, полегшуючи комунікацію в команді та з користувачами.

6) Створення ігрової програми:

Створив гру "хрестики нолики" на мові C++, що дозволило застосувати отримані знання у практичній роботі.

## ***До 5 теми:***

Під час моєї навчальної практики у веб-дизайні та HTML я вивчив основні принципи створення сучасних та адаптивних веб-сторінок, відповідних стандартам індустрії. Я отримав глибокі знання з HTML, CSS та веб-дизайну, які дозволяють мені ефективно створювати інтерфейси, залучаючі та зручні для користувачів. Практичні завдання допомогли закріпити теоретичні знання та забезпечили мене необхідними навичками для успішної реалізації веб-проектів. Цей досвід значно підвищив мої професійні здібності, і я впевнено планую продовжувати вдосконалювати свої навички у сфері веб-дизайну та розробки.