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:  студентка 2-го курсу групи ІПЗ-23  *Стешенко Анастасії Сергіївни*  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*

Стешенко Анастасії Сергіївни

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 XHTML є більш суворим варіантом HTML, вона являється додатком синтаксису XML і є додатком мови XML в області розмітки гіпертексту. У всесвітній павутині HTML-сторінки, як правило, передаються браузерам від сервера по протоколах HTTP або HTTPS, у вигляді простого тексту або з використанням шифрування.

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 прості 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 легкий формат обміну даними, який використовується для зберігання та передачі структурованих даних між різними системами. Він базується на синтаксисі JavaScript, але може бути використаний в багатьох інших мовах програмування. JSON використовується для представлення об'єктів та масивів даних у формі, зрозумілій як людям, так і програмам.

Основні складові JSON - це пари "ключ-значення", які можуть бути вкладені одна в одну для представлення складних структур даних. JSON широко використовується для обміну даними між клієнтськими та серверними додатками, а також для зберігання налаштувань, конфігурацій та інших структурованих даних.

***Робота з файлами JSON:***

***Плюси****:*

1. Легкість читання та редагування: Формат JSON має просту та легку для сприйняття структуру, що робить файли JSON легкими для читання та редагування як людиною, так і машинами.
2. Підтримка мов програмування: Більшість сучасних мов програмування мають бібліотеки або пакети, які дозволяють легко читати та записувати дані у форматі JSON, що робить його досить універсальним для використання.
3. Гнучкість: JSON дозволяє представляти різноманітні дані, включаючи тексти, числа, масиви, об'єкти та вкладені структури даних.
4. Популярність: JSON є одним з найбільш поширених форматів обміну даними в сучасному програмуванні, що означає, що для нього існують багато засобів підтримки та бібліотек для різних мов програмування.

***Мінуси:***

1. Великий обсяг даних: JSON може виявитися неефективним для великих обсягів даних, оскільки він зазвичай використовує багато зайвих символів для форматування.
2. Брак сховищ даних: JSON не надає вбудованих механізмів для роботи з великими обсягами даних, такими як індексація або пошук, що може ускладнити обробку великих файлів.

XML (Extensible Markup Language) - це розширювана мова розмітки, яка використовується для представлення та обміну структурованими даними між комп'ютерними системами. XML базується на тезі, що документ має бути структурований та зрозумілий як комп'ютерам, так і людям.

В основі XML лежить синтаксис тегів, подібний до HTML, але він не має заздалегідь визначених тегів та може бути розширений для використання в будь-якому контексті. XML використовується для створення структурованих документів, таких як конфігураційні файли, налаштування програм, обміну даними між веб-серверами та клієнтами, а також для зберігання даних в базах даних. Він є одним з основних форматів для обміну даними в сучасному програмуванні та веб-розробці.

***Робота з файлами XML:***

*Плюси:*

1. Структурованість: XML має строго визначену структуру, що робить його ідеальним для представлення складних структур даних та документів з вкладеними елементами.
2. Можливість валідації: XML може бути легко валідований за допомогою схем XSD (XML Schema Definition), що дозволяє перевіряти правильність структури та типів даних у файлі.
3. Підтримка стандартів: XML підтримується багатьма стандартами та протоколами, що робить його корисним для обміну даними між різними системами.

*Мінуси:*

1. Великий обсяг даних: XML-файли можуть бути досить об'ємними через велику кількість тегів та атрибутів, що може призвести до збільшення розміру файлу та ускладнення його обробки.
2. Складність читання та редагування: Велика кількість тегів та вкладеності може зробити XML-файли складними для читання та редагування людиною, особливо у випадку великих обсягів даних.
3. Обмежена підтримка: Хоча більшість мов програмування підтримують роботу з XML, цей формат може виявитися менш зручним у порівнянні з іншими форматами, такими як JSON, зокрема у веб-розробці та обміні даними між веб-серверами та клієнтами.

## ***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.

## ***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.

Основна мета VCS полягає в тому, щоб зберігати історію змін, дозволяти співробітникам працювати одночасно над одним проектом та відновлювати попередні версії коду у випадку помилок або потреби у відновленні попередньої функціональності.

Основні плюси систем контролю версій включають:

1. ***Історія змін****:* Всі зміни до коду або файлів зберігаються і можуть бути переглянуті, зроблені коментарі до змін, включаючи інформацію про того, хто та коли зробив зміну.
2. ***Робота з різними версіями****:* Ви можете відновити попередню версію файлу або навіть порівняти дві версії, щоб побачити, що було змінено.
3. ***Гілки розробки****:* Ви можете створювати окремі гілки розробки для роботи над різними функціональностями або різними аспектами проекту паралельно.
4. ***Співпраця****:* Декілька розробників може одночасно працювати над одним проектом, а VCS автоматично об'єднає їхні зміни.

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 легка у використанні мова розмітки, призначена для форматування тексту у веб-засобах. Вона часто використовується для написання різноманітних документів, включаючи опис проектів, README файли, блоги та форуми.

***Основні можливості Markdown***

1. ***Заголовки****:* Використовуються знаки # для створення заголовків різних рівнів. Кількість # визначає рівень заголовка (від 1 до 6).

markdown

# Заголовок 1

## Заголовок 2

### Заголовок 3

1. ***Форматування тексту****:*
   * ***Жирний текст***: Обгорніть текст подвійними зірочками або підкресленнями (\*\*текст\*\* або \_\_текст\_\_).
   * ***Курсив****:* Обгорніть текст однією зірочкою або підкресленням (\*текст\* або \_текст\_).
   * ***Закреслений текст***: Використовуйте подвійні тильди (~~текст~~).

markdown

Це \*\*жирний текст\*\*, це \*курсив\*, і це ~~закреслений текст~~.

1. ***Списки****:*
   * ***Нумеровані списки****:* Використовуйте числа з крапкою.
   * ***Марковані списки****:* Використовуйте зірочки, плюси або дефіси.

markdown

1. Пункт 1

2. Пункт 2

- Підпункт 1

- Підпункт 2

- Пункт 1

- Пункт 2

1. ***Код****:*
   * ***Інлайн код****:* Обгорніть текст зворотніми лапками (``).
   * ***Блок коду***: Використовуйте три зворотні лапки (```).

markdown

Це `інлайн код`.

```cpp

#include <iostream>

using namespace std;

int main() {

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

return 0;

}

1. ***Посилання та зображення****:*
   * ***Посилання****:* Обгорніть текст в квадратні дужки, а URL - в круглі дужки.
   * ***Зображення*:** Схоже на посилання, але з префіксом !.

markdown

[Google](https://www.google.com)

![Alt text](https://example.com/image.jpg)

1. ***Цитати****:* Використовуйте знак більше (>) для створення цитат.

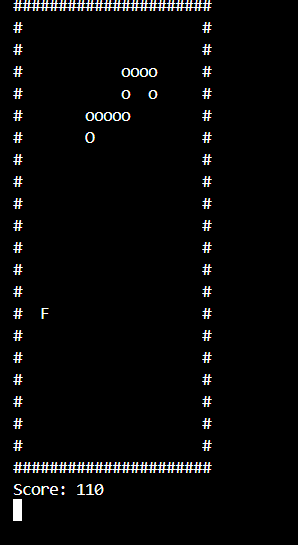
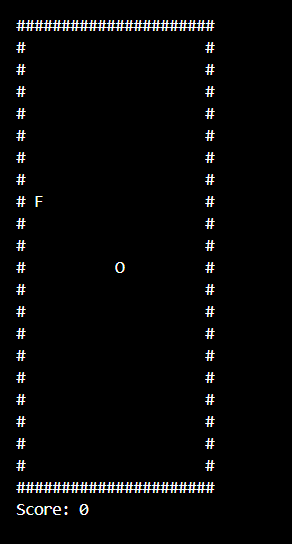
markdown

> Це цитата.

## ***Тема 4.6 Створення ігрової програми***

Мій код є реалізацією простої гри "Змійка" на мові програмування C++. Гравець керує змійкою, яка рухається по полі, намагаючись з'їсти фрукти, щоб збільшити свій рахунок. Основні елементи гри включають змійку, фрукти та логіку руху. Гра завершується, якщо змійка зіткнеться зі стіною або самою собою. У коді реалізовано функції Setup() для початкової настройки гри, Draw() для відображення стану гри на екрані, Input() для обробки введених даних від користувача та Logic() для обчислення логіки гри.

Код демонструє базовий приклад використання мови програмування C++ для створення простої консольної гри. Він показує, як можна використовувати цикли, умовні оператори та змінні для створення ігрового середовища. Цей код може бути використаний як основа для розширення та вдосконалення гри, додавання нових функцій та покращення інтерфейсу користувача.

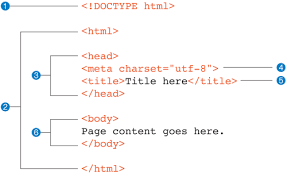
****

## ***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>

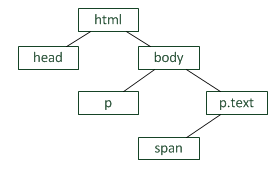
<p> Текст документа </ p>

         <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.

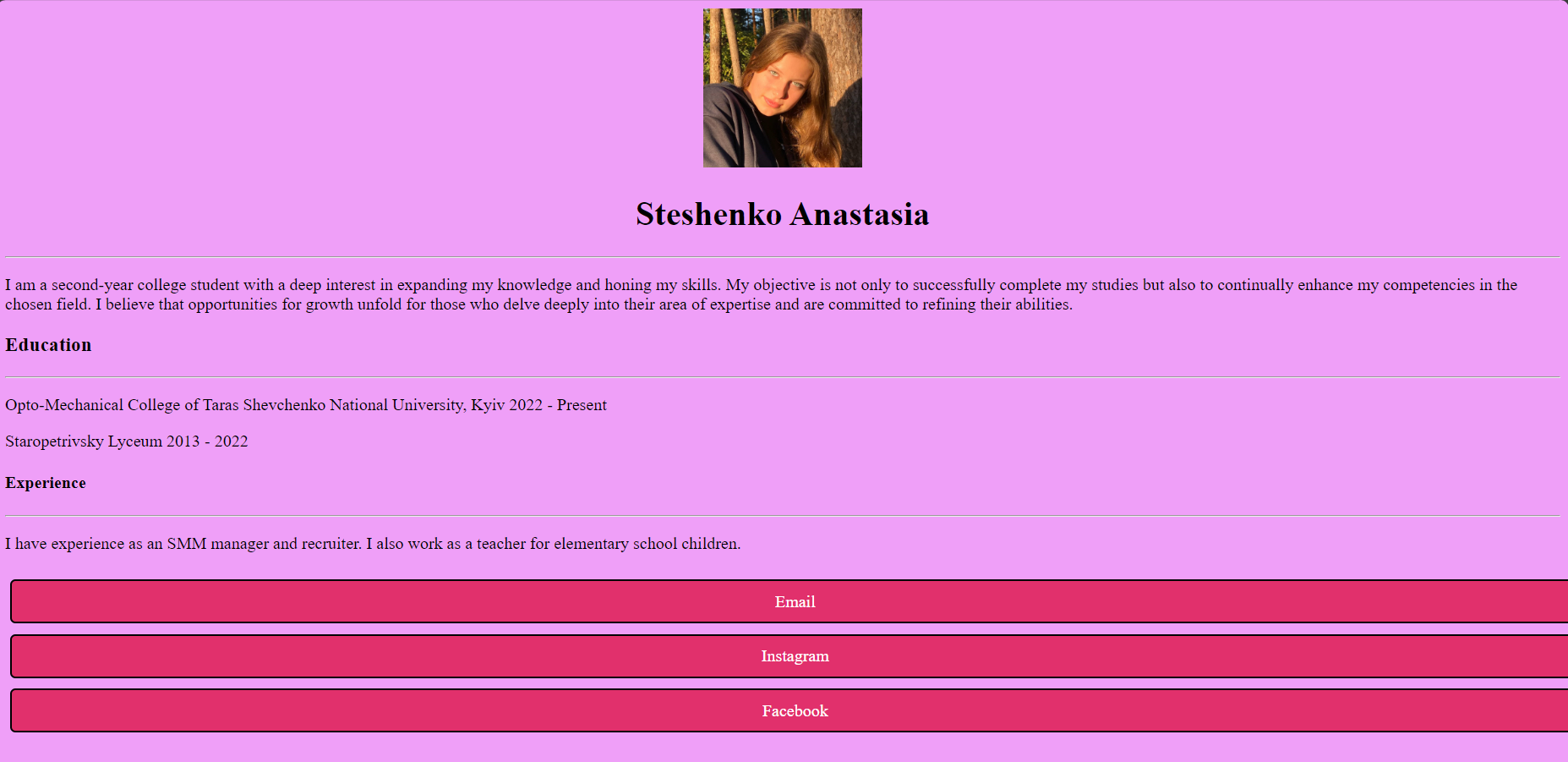
Каскадність якраз і визначає, які саме властивості з цих джерел застосуються до даного абзацу. Є три основні концепції, керуючі порядком, в якому застосовуються CSS-властивості:

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

# Опис постановки задачі

Основним завданням було створення веб-сайту у вигляді резюме. Головною метою цієї роботи було ознайомлення із мовою розмітки HTML та структурою веб-сторінки, засвоєння CSS та принципів каскадного оформлення стилів, а також оволодіння основними їх функціями. Мій проєкт включає в себе текст, що відформатований зміною кольору, розміру шрифту, стилю та інших параметрів для кращої читабельності. Крім того, в ньому містяться фотографії, активні посилання для зручного переходу між розділами та елементи оформлення об'єктів, щоб забезпечити краще сприйняття відвідувачами веб-сайту.

***Розроблена веб-сторінка***



# Висновки

## ***До 4 теми:***

Під час навчальної практики за розділом "Практична робота з С++" я значно розширила свої знання та здобула цінні практичні навички, які є фундаментальними для подальшого розвитку в галузі програмування. Ось ключові аспекти, які були охоплені під час практики:

1. Введення та виведення у базових консольних програмах

Цей розділ дозволив мені опанувати основи взаємодії програм з користувачем через консоль. Я навчилась використовувати стандартні бібліотеки C++ для введення та виведення даних, що є основою для створення більш складних програм. Ці навички є необхідними для розробки ефективних і зрозумілих користувацьких інтерфейсів у консольних додатках.

1. Робота з файлами: запис та читання

Практичні завдання з роботи з файлами допомогли мені зрозуміти, як зберігати і обробляти дані у зовнішніх файлах. Це включає навички роботи з текстовими та двійковими файлами, що є критично важливим для створення додатків, які працюють з великими обсягами даних або зберігають дані між сесіями.

1. Формати файлів та їх обробка

Ознайомлення з форматами JSON та XML надало мені розуміння структурованого зберігання даних і їх обміну між різними системами. Я навчилася працювати з цими форматами, що дозволяє інтегрувати мої програми з іншими сервісами та додатками, забезпечуючи ефективну обробку даних.

1. Системи контролю версій (VCS)

Використання системи контролю версій Git дало змогу зрозуміти принципи управління версіями коду, відстежування змін та співпраці в команді. Ці навички є необхідними для роботи в сучасних командах розробників, де контроль за змінами коду та спільна робота є критично важливими.

1. Мова Markdown для опису проектів

Markdown є простою та ефективною мовою розмітки для створення документації. Вивчення та використання Markdown допомогли мені створювати зрозумілі, структуровані та привабливі технічні описи, що полегшує комунікацію в команді та з користувачами.

1. *Створення ігрової програми*

Реалізація гри "Змійка" на мові C++ стала практичним застосуванням всіх вивчених концепцій. Проектування ігрової логіки, обробка введення користувача та використання базових графічних елементів у консолі дозволили мені зрозуміти, як застосовувати теоретичні знання на практиці. Цей проект надав мені впевненості у своїх навичках програмування та вміння вирішувати складні завдання.

Ця практика стала важливим етапом у моєму професійному розвитку. Отримані знання та навички створили міцну базу для подальшого навчання та роботи у сфері програмування. Я впевнена, що зможу успішно застосовувати їх у своїй майбутній кар'єрі, вирішуючи складні завдання та створюючи ефективні та якісні програмні рішення.

## ***До 5 теми:***

Розробка веб-сайту за допомогою HTML стала важливим етапом у моєму навчанні та пізнанні основ веб-розробки. Під час цієї практики я опанувала не лише основи мови розмітки HTML, а й навчилася створювати структуровані та зрозумілі веб-сторінки. Відправний пункт веб-дизайну, який я здобула, дозволив мені розуміти, як кожен елемент впливає на загальний вигляд і функціональність сторінки.

Одне з ключових відкриттів для мене полягало в тому, як зв'язувати структуру та вміст веб-сторінки за допомогою тегів та атрибутів. Це дозволило мені краще організувати інформацію та створювати візуально зручний інтерфейс для користувачів.

У процесі створення веб-сайту я також вивчила базові засоби форматування тексту, вставки зображень та створення посилань. Це відкрило для мене широкі можливості для реалізації різноманітних ідей та концепцій.

Окрім технічних аспектів, ця практика дала мені можливість поглибити розуміння процесу створення веб-сайтів від початку до кінця. Від планування до розробки і тестування, я зрозуміла важливість кожного етапу та його вплив на кінцевий продукт.

Завдяки цій практиці я відчула себе більш впевнено у власних знаннях та здібностях у галузі веб-розробки. Це стало моєю важливою базою для подальшого розвитку і поглиблення в цій сфері.

# Список використаних джерел

1. <https://cpto.dp.ua/public_html/posibnyky/basic_html/urok1.html>
2. <https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web>
3. <https://e-tk.lntu.edu.ua/mod/page/view.php?id=2894>
4. <https://e-helper.com.ua/node/949>
5. <https://acode.com.ua/uroki-po-cpp/>
6. <https://cherto4ka.xyz/2021/11/03/%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B8-%D0%BA%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F-%D0%B2%D0%B5%D1%80%D1%81%D1%96%D1%8F%D0%BC%D0%B8/>
7. <https://highload.today/uk/mova-rozmitki-markdown-povnij-gajd-iz-sintaksisu-ta-formatuvannya-tekstu/>
8. https://web.posibnyky.vntu.edu.ua/fitki/14baryshev\_tehnologiya\_programuvannya/08.html