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  *Кириленко Тимур Сергійович*  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*

Кириленко Тимура Сергійовича

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 з ключових аспектів програмування. Ці операції дозволяють користувачам взаємодіяти з програмою і отримувати результати її роботи через консольний інтерфейс.

***1. Введення даних***: У C++ для цієї мети використовується об'єкт std::cin, який дозволяє отримувати 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 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)

JSON – це легкий формат для обміну даними, який використовується для зберігання та передачі структурованих даних між різними системами. Він заснований на синтаксисі JavaScript, але підтримується багатьма іншими мовами програмування. JSON зручний для представлення об'єктів та масивів даних у формі, зрозумілій як для людей, так і для програм.

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

***Переваги роботи з JSON:***

1. ***Легкість читання та редагування:*** Простота структури JSON робить його легким для читання та редагування як людьми, так і машинами.

2. ***Підтримка багатьох мов програмування:*** Сучасні мови програмування мають бібліотеки, які дозволяють легко працювати з JSON, що робить його універсальним.

3. ***Гнучкість:*** JSON може представляти різні типи даних, включаючи тексти, числа, масиви, об'єкти та вкладені структури.

4. ***Популярність:*** JSON є одним з найпоширеніших форматів обміну даними, що забезпечує наявність великої кількості інструментів та бібліотек для роботи з ним.

***Недоліки роботи з JSON***:

1. ***Великий обсяг даних:*** JSON може бути неефективним для великих обсягів даних через значну кількість символів для форматування.

2. ***Відсутність механізмів роботи з великими обсягами даних:*** JSON не має вбудованих засобів для індексації або пошуку, що ускладнює роботу з великими файлами.

XML (Extensible Markup Language)

XML – це розширювана мова розмітки, що використовується для представлення та обміну структурованими даними між комп'ютерними системами. Вона має синтаксис тегів, схожий на HTML, але не містить заздалегідь визначених тегів, що дозволяє розширювати її для будь-якого контексту. XML використовується для створення структурованих документів, таких як конфігураційні файли, налаштування програм, обміну даними між веб-серверами та клієнтами, а також для зберігання даних у базах даних.

***Переваги роботи з XML:***

1. ***Структурованість:*** XML має строго визначену структуру, що підходить для представлення складних структур даних та документів з вкладеними елементами.

2***. Можливість валідації:*** XML можна легко перевірити за допомогою схем XSD, що дозволяє перевіряти правильність структури та типів даних.

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

***Недоліки роботи з XML:***

1. ***Великий обсяг даних:*** XML-файли можуть бути досить великими через велику кількість тегів та атрибутів, що збільшує розмір файлу та ускладнює його обробку.

2. ***Складність читання та редагування***: Велика кількість тегів та вкладеності може зробити XML-файли складними для читання та редагування людиною, особливо при великих обсягах даних.

3. Обмежена підтримка: Хоча більшість мов програмування підтримують XML, цей формат може бути менш зручним у порівнянні з JSON, особливо у веб-розробці та обміні даними між веб-серверами та клієнтами.

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

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

**#### 1. Текстові файли**

Ці файли зберігають дані у звичайному текстовому форматі, який може бути легко прочитаний та змінений за допомогою звичайного текстового редактора. У C++ для роботи з текстовими файлами використовуються потоки введення/виведення (ifstream та ofstream).

1. ***Переваги:*** Легкі для розуміння та зручні для редагування людиною, можуть бути оброблені будь-яким текстовим редактором. Ідеально підходять для зберігання конфігураційних файлів, логів, текстових даних тощо.
2. ***Недоліки:*** Менш ефективні для зберігання великих обсягів даних через потребу в додатковому просторі для текстового представлення. Обробка текстових файлів може бути повільнішою порівняно з бінарними файлами через необхідність конвертації даних з/у текстовий формат.

**#### 2. Бінарні файли**

Ці файли містять дані у вигляді двійкового коду, що дозволяє зберігати дані компактніше та ефективніше. Для роботи з бінарними файлами у C++ також використовуються потоки введення/виведення (ifstream та ofstream), але операції читання та запису виконуються у бінарному форматі.

1. ***Переваги:*** Бінарні файли ефективні для зберігання великих обсягів даних, оскільки займають менше місця. Обробка бінарних даних може бути швидшою, оскільки вони не потребують конвертації в текстовий формат.
2. ***Недоліки:*** Бінарні файли непридатні для читання та редагування людиною, оскільки їхній зміст представлений у вигляді двійкового коду. Це ускладнює їх відладку та виправлення помилок.

#### **3. 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. У C++ для роботи з цими форматами можна використовувати сторонні бібліотеки.

***Переваги:*** JSON та XML дозволяють представляти структуровані дані у зручному для читання та обробки форматі. Вони популярні для обміну даними між програмами та веб-серверами завдяки своїй розширюваності та простоті використання.

***Недоліки:*** JSON та XML можуть бути менш ефективними у порівнянні з бінарними форматами при зберіганні великих обсягів даних. Обробка JSON та XML може вимагати додаткових бібліотек або інструментів для парсингу та створення даних.

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

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

***Системи контролю версій (VCS)***

Системи контролю версій (VCS) – це програмні інструменти, які використовуються для відстеження змін у файлах та коді під час розробки програмного забезпечення. Вони зберігають історію змін, дозволяють працювати над проектом декільком розробникам одночасно та відновлювати попередні версії коду при необхідності.

***Основні переваги VCS:***

1. ***Історія змін*:** Усі зміни до коду або файлів зберігаються, з коментарями про те, хто і коли зробив зміни.
2. ***Робота з версіями:*** Можливість відновлення попередніх версій файлів і порівняння різних версій для виявлення змін.
3. ***Гілки розробки:*** Створення окремих гілок для роботи над різними функціями або аспектами проекту паралельно.
4. ***Співпраця:*** Кілька розробників можуть одночасно працювати над проектом, і VCS автоматично об'єднає їхні зміни.

***Основні недоліки VCS:***

1. ***Крива навчання:*** Вивчення функціоналу та інтерфейсу VCS вимагає додаткових зусиль та часу.
2. ***Розмір бази даних:*** База даних VCS може займати значний обсяг пам'яті, особливо у великих проектах з багатою історією змін.
3. ***Конфлікти злиття:*** При роботі над спільними файлами можуть виникати конфлікти злиття, які потребують ручного вирішення.

***Централізовані та розподілені системи контролю версій:***

* ***Централізовані системи:*** Весь репозиторій зберігається на центральному сервері, а розробники підключаються до нього через мережу. Прикладом є Subversion (SVN).
* ***Розподілені системи:*** Кожен розробник має свою копію повного репозиторію і може працювати незалежно. Найпопулярніший приклад – Git.

***Відгалуження та злиття (Branching and Merging):***

* ***Гілки (Branches):*** Дозволяють розробникам працювати над різними аспектами проекту одночасно без впливу на основну гілку. Кожна гілка має свою історію змін.
* ***Злиття (Merging):*** Використовується для об'єднання змін з однієї гілки в іншу. При конфліктах злиття розробник повинен їх вирішити, переглянувши та об'єднавши відповідні рядки коду.

***Командна робота (Collaboration):***

* ***Зручність співпраці:*** Системи контролю версій дозволяють декільком розробникам працювати над одним проектом одночасно, відстежуючи зміни кожного члена команди.
* ***Збереження інформації:*** Кожна зміна зберігається разом з інформацією про автора, дату та коментарі, що сприяє зручній співпраці та відстеженню історії розробки.

***Робота з галузями (Forking):***

* ***Форки:*** Деякі системи контролю версій, такі як GitHub, дозволяють розробникам створювати власні копії проектів ("форки"). Це дозволяє вносити зміни та експериментувати з кодом без впливу на оригінальний *проект.*

***Відстеження проблем (Issue Tracking):***

* ***Відстеження завдань:*** Деякі платформи для спільної роботи, такі як GitHub та GitLab, дозволяють створювати та відстежувати проблеми (issues) або завдання (tasks) пов'язані з проектом, полегшуючи координацію роботи команди.

## ***Тема 4.5 Мова Markdown для опису проектів***

Markdown – це проста та легка мова розмітки, яка призначена для форматування тексту в інтернеті. Її часто використовують для написання документації, README файлів, блогів і форумів. Ось кілька основних переваг використання Markdown для опису проектів:

1. ***Простота використання:*** Markdown легко вивчити та використовувати завдяки простим синтаксичним правилам, як-от використання символів "" або "\_" для виділення тексту.
2. ***Читабельність:*** Текст у форматі Markdown легко читати як у вихідному коді, так і у відформатованому вигляді на веб-сторінках.
3. ***Широка підтримка:*** Багато веб-сервісів і платформ, включаючи GitHub, GitLab, Bitbucket, Reddit, а також блог-системи, підтримують Markdown.
4. ***Можливості розширення*:** Існують інструменти та розширення для додавання підтримки таблиць, зображень, посилань та інших елементів.
5. ***Інтеграція з іншими мовами:*** Markdown легко інтегрується з HTML, CSS та системами контролю версій, що робить його зручним для різних середовищ розробки.
6. ***Підтримка структурованих документів*:** Markdown дозволяє створювати структуровані документи за допомогою заголовків, списків, блоків коду тощо, що робить опис проектів більш організованим.
7. ***Інтеграція з інструментами:*** Markdown можна інтегрувати з інструментами, такими як Jekyll, Hugo, Pelican для блогів, або Jupyter Notebook для наукових звітів.
8. ***Можливості розмітки:*** Markdown підтримує різні можливості розмітки, включаючи створення посилань, вставку зображень, таблиць, форматування коду та списків, що робить документи інформативнішими.
9. ***Підтримка спеціалізованих розширень*:** Деякі платформи підтримують розширення для вставки відео, аудіо, математичних формул та інших складних елементів.
10. ***Крос-платформеність:*** Markdown-документи можна відкривати та переглядати на будь-якому пристрої з будь-якою операційною системою, що робить їх універсальними для співпраці та обміну інформацією.

Markdown є потужним і зручним інструментом для створення документації проектів. Він забезпечує простий та ефективний спосіб форматування тексту, дозволяючи швидко створювати чіткі та структуровані документи. Основні переваги Markdown включають зручність використання, читабельність, широку підтримку та можливість розширення. Markdown ідеально підходить для спільної роботи та обміну документацією, оскільки його файли можна легко читати на будь-якому пристрої. Однак для деяких складніших форматувань може знадобитися використання інших мов розмітки. У цілому, Markdown є важливим інструментом для ефективного управління та створення документації під час розробки програмного забезпечення.

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

Мій код реалізує просту гру "Змійка" на мові програмування C++. У цій грі гравець керує змійкою, яка рухається по полю, намагаючись з'їсти фрукти для збільшення рахунку. Основні компоненти гри включають змійку, фрукти та логіку руху. Гра завершується, якщо змійка зіткнеться зі стіною або самою собою.

Код містить кілька основних функцій:

* Setup(): виконує початкову налаштування гри.
* Draw(): відповідає за відображення стану гри на екрані.
* Input(): обробляє введені дані від користувача.
* Logic(): обчислює логіку гри.

Цей код демонструє базові можливості мови програмування C++ для створення простої консольної гри. Він показує, як можна використовувати цикли, умовні оператори та змінні для створення ігрового середовища. Цей приклад може служити основою для подальшого розширення та вдосконалення гри, додавання нових функцій та покращення користувацького інтерфейсу.

#include <iostream>

#include <conio.h> // For \_getch()

using namespace std;

// Game variables and constants

bool gameOver;

const int width = 20;

const int height = 20;

int x, y, fruitX, fruitY, score;

int tailX[100], tailY[100];

int nTail;

enum eDirection { STOP = 0, LEFT, RIGHT, UP, DOWN };

eDirection dir;

// Function to initialize the game settings

void Setup() {

gameOver = false;

dir = STOP;

x = width / 2; // Starting position of the snake

y = height / 2;

fruitX = rand() % width; // Random position for the fruit

fruitY = rand() % height;

score = 0;

}

// Function to draw the game screen

void Draw() {

system("cls"); // Clear the screen

for (int i = 0; i < width + 2; i++)

cout << "#"; // Top border

cout << endl;

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

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

if (j == 0)

cout << "#"; // Left border

if (i == y && j == x)

cout << "O"; // Snake head

else if (i == fruitY && j == fruitX)

cout << "F"; // Fruit

else {

bool print = false;

for (int k = 0; k < nTail; k++) {

if (tailX[k] == j && tailY[k] == i) {

print = true;

cout << "o"; // Snake body

}

}

if (!print)

cout << " "; // Empty space

}

if (j == width - 1)

cout << "#"; // Right border

}

cout << endl;

}

for (int i = 0; i < width + 2; i++)

cout << "#"; // Bottom border

cout << endl;

cout << "Score: " << score << endl; // Display the score

}

// Function to handle user input

void Input() {

switch (\_getch()) {

case 'a':

if (dir != RIGHT) dir = LEFT;

break;

case 'd':

if (dir != LEFT) dir = RIGHT;

break;

case 'w':

if (dir != DOWN) dir = UP;

break;

case 's':

if (dir != UP) dir = DOWN;

break;

case 'x':

gameOver = true;

break;

}

}

// Function to update the game logic

void Logic() {

int prevX = tailX[0];

int prevY = tailY[0];

int prev2X, prev2Y;

tailX[0] = x;

tailY[0] = y;

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

prev2X = tailX[i];

prev2Y = tailY[i];

tailX[i] = prevX;

tailY[i] = prevY;

prevX = prev2X;

prevY = prev2Y;

}

switch (dir) {

case LEFT:

x--;

break;

case RIGHT:

x++;

break;

case UP:

y--;

break;

case DOWN:

y++;

break;

}

// Check for boundary collision and wrap around

if (x >= width) x = 0; else if (x < 0) x = width - 1;

if (y >= height) y = 0; else if (y < 0) y = height - 1;

// Check for collision with itself

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

if (tailX[i] == x && tailY[i] == y)

gameOver = true;

}

// Check if fruit is eaten

if (x == fruitX && y == fruitY) {

score += 10;

fruitX = rand() % width;

fruitY = rand() % height;

nTail++;

}

}

// Main function to run the game loop

int main() {

Setup();

while (!gameOver) {

Draw();

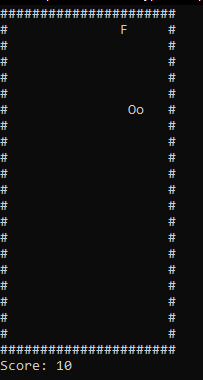
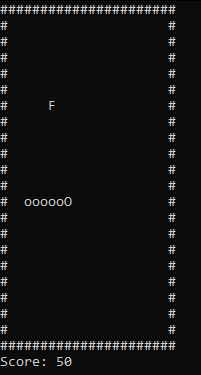
Input();

Logic();

}

return 0;

}

**** ****

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

HTML-документ обробляється браузером і відображається на екрані у зручному для користувача вигляді.

HTML надає можливості для:

- створення структурованого документа за допомогою позначення його складових частин: заголовків, абзаців, списків, таблиць, цитат та інших елементів;

- створення зв’язків між веб-сторінками через гіперпосилання;

- створення інтерактивних форм;

- включення зображень, звуку, відео та іншого мультимедійного контенту.

Весь контент у HTML описується за допомогою спеціальних правил розмітки. Розмітка в HTML складається з чотирьох основних компонентів:

- елементів;

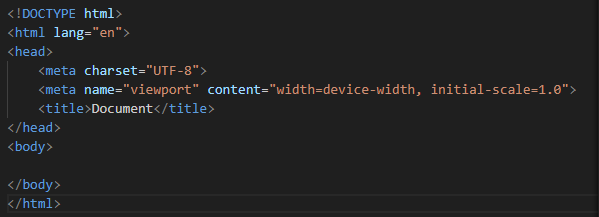
- базових типів даних (визначають тип вмісту атрибутів: символьні дані, числа тощо);

- символьних мнемонік;

- декларації типу документа (оголошує версію HTML).

Мнемоніки – це спеціальні позначення (псевдоніми), які використовуються для вставки певних символів.

В загальному, структура HTML-документа є такою:

**

Теги <HTML>...</HTML> визначають початок і кінець HTML-документа. Розділ HEAD документа визначає його заголовок, який розташовується між тегами <TITLE> і </TITLE>, а також включає мета-теги.

Тег <META> містить різну інформацію для браузерів, яка не відображається на екрані. Тег <STYLE> пов'язаний з використанням таблиць стилів, а тег <SCRIPT> використовується для запису сценаріїв.

Елементи є основними компонентами розмітки HTML. Кожен елемент має назву і визначається за допомогою тегів, що записуються у кутових дужках. Вміст елемента записується між відкриваючим і закриваючим тегами. Також елементи можуть мати атрибути, які задаються всередині відкриваючого тега. Значення атрибутів записуються в подвійних лапках. Деякі елементи можуть не мати атрибутів, а інші мають обов'язкові атрибути, без яких вони не можуть функціонувати. Є також елементи, які не мають вмісту і визначаються за допомогою одиночних тегів.

Елементи HTML можна поділити на три основні категорії: структурні, презентаційні та елементи для інтерактивної взаємодії з користувачем.

Структурні елементи використовуються для опису семантики тексту та структури документа. Залежно від змісту документа, структурні елементи поділяються на текст, списки, таблиці, гіперпосилання та додаткові об'єкти (зображення, аплети тощо). Вони не визначають спеціального (візуального) відтворення тексту, але більшість браузерів мають стандартні стилі форматування для кожного елемента. Для подальшого стилізування тексту рекомендується використовувати каскадні таблиці стилів (CSS).

Презентаційні елементи застосовуються для опису зовнішнього вигляду документа і не визначають функцій тексту, до якого вони застосовуються. Більшість цих елементів не рекомендується використовувати в розмітці. Залежно від свого змісту, презентаційні елементи поділяються на елементи додавання каскадних таблиць стилів, візуальні елементи тексту (шрифти, кольори, насиченість і нахил шрифту, лінії тощо) та фрейми.

Елементи інтерактивної взаємодії дозволяють користувачам впливати на вміст або вигляд документа. Вони поділяються на два типи: форми для введення даних користувачами та скрипти для активних документів.

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

Абзаци

Почнемо з тега <p>, який використовується для створення абзаців. За замовчуванням абзаци починаються з нового рядка і мають вертикальні відступи, які можна налаштувати за допомогою стилів.

Заголовки і підзаголовки

Для організації великих текстів використовуються заголовки. У текстових редакторах можна виділити текст як заголовок. В HTML для цього існує сімейство тегів від <h1> до <h6>. Тег <h1> (headline 1) позначає найважливіший заголовок, а тег <h6> — підзаголовок найнижчого рівня.

На практиці підзаголовки нижче третього рівня рідко використовуються. Тому найчастіше застосовують теги <h1>, <h2> і <h3>. Варто зазначити, що пошукові системи надають особливу вагу заголовкам.

Маркований (ненумерований) список

Списки часто використовуються в документах. Користувачі іноді нумерують рядки тексту, але це не створює логічної сутності «список». У HTML є теги для створення списків: невпорядкованих, упорядкованих і списків визначень.

Невпорядковані (або марковані) списки створюються за допомогою тега <ul> (unordered list), який містить теги <li> (list item) для позначення елементів списку.

Впорядкований (нумерований) список

Впорядковані списки створюються за допомогою тега <ol> (ordered list) і містять теги <li>. Елементи впорядкованого списку нумеруються автоматично, що корисно для покрокових інструкцій. Це дозволяє змінювати порядок елементів списку без необхідності ручної перенумерації.

Для зміни маркерів і номерів списків використовується властивість list-style-type в CSS. Наприклад, для маркованих списків можна змінити форму маркера з кола на квадрат, незаштриховане коло, власне зображення або прибрати маркер повністю. Для нумерованих списків можна замінити арабські цифри римськими (I., II., III. або i., ii., iii.), літерами (A., B., C. або a., b., c.) та іншими типами нумерації.

Важливість. Теги <strong> та <b>

Тег <strong> визначає важливість виділеного тексту, тоді як тег <b> (bold) використовується для візуального виділення тексту без надання йому додаткової важливості. Відмінності між цими тегами найбільш помітні для людей, які використовують спеціальні налаштування ОС, зокрема для сліпих і слабозорих. Коли увімкнено функцію читання тексту, програма для читання тексту інтонаційно виділяє слова, позначені тегом <strong>. Те саме стосується і тегів <em> (emphasis) та <i>. Тег <em> також виділяється інтонацією, хоча візуально обидва тега роблять текст жирним. У HTML5 тег <b> отримав новий сенс: раніше він просто робив текст напівжирним і використовувався лише для візуального форматування.

Переноси та розділювачі. Теги <br> та <hr>

Іноді потрібно вставити перенесення рядка без створення нового абзацу, наприклад, при розмітці віршів або пісень. Для цього в HTML використовується одиночний тег <br> (break). Проте, не рекомендується використовувати цей тег для розбивки тексту на "як би абзаци". Для абзаців слід використовувати тег <p>.

Одиночний тег <hr> (horizontal rule) використовується для створення горизонтальної лінії-розділювача.

Цитати

HTML пропонує кілька тегів для цитат:

- <blockquote> для довгих цитат, які можуть складатися з кількох абзаців. Цей тег виділяє цитату як окремий блок тексту з відступами.

- <q> (quote) для коротких цитат у тексті. Текст всередині цього тега автоматично обрамляється лапками.

- <cite> використовується для виділення джерела цитати, як-от назви твору, але не автора.

Посилання

Посилання зазвичай виглядає як підкреслена частина тексту, при натисканні на яку користувач переходить на іншу сторінку, відкриває зображення або завантажує файл. У HTML посилання створюються за допомогою тега <a>. Наприклад:

<a href="посилання">Назва</a>

Адреси можуть бути різними: абсолютними, відносними, вказувати на сторінку, файл, зображення або містити якір. Посилання може вказувати на будь-який об'єкт в Інтернеті – сторінку, файл тощо. При натисканні на посилання, яке веде на файл, браузер запропонує його завантажити, але якщо браузер підтримує цей тип файлів, він відкриє його прямо в браузері.

Робота з картинками

Для додавання зображень на сторінку використовується одиночний тег <img> з атрибутом src (source), який вказує на адресу картинки. Наприклад:

<img src="logo.png">

Найпоширенішими форматами зображень у мережі є JPEG і PNG. Для управління шириною або висотою зображення використовуються атрибути width і height. Наприклад:

<img width="100" src="logo.png">

Це задає ширину зображення у 100 пікселів. Якщо зображення не завантажується, браузер відображає альтернативний текст, який задається за допомогою атрибуту alt (alternative). Наприклад:

<img src="logo.png" alt="Логотип">

Посилання можна створювати не лише за допомогою тексту, але й за допомогою зображень. Для цього тег <img> обертають у тег <a>. Такі посилання-зображення часто використовуються в галереях, де на зменшену версію зображення ставиться посилання на повнорозмірну картинку.

## ***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 надає можливість створювати правила, які легко змінювати, редагувати і застосовувати до усіх визначених нами елементів.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Властивість та її значення

Властивість описує елемент, який вводиться, а значення визначають параметри властивостей.

Способи підключення таблиці стилів

Існує два основні способи підключення таблиці стилів:

1. **Вбудовані стилі (Inline CSS**)

У цьому способі CSS додається безпосередньо до HTML-документу за допомогою атрибута `style` всередині будь-якого HTML-тега, який знаходиться в контейнері `<body>`.

2. **Вбудовані стилі в заголовку (Internal CSS)**

У цьому способі CSS додається до HTML-документу за допомогою тега `<style>`, розміщеного в контейнері `<head>`. Тут описуються всі стилі, які будуть використовуватися в документі.

**Робота з кольором та фоном за допомогою CSS**

Одна з основних функцій CSS – це робота з кольором і фоном.

- Властивість `color`

Визначає колір елемента. Наприклад:

h1 { color: #ff0000; }

- Властивість `background-color`

Визначає колір фону елемента. Наприклад:

body { background-color: #FFCC66; }

h1 {

  color: #000000;

  background-color: #FC9804;

  text-align: center;

}

**-** Властивість `background-image`

Використовується для додавання фонового зображення. Наприклад:

body { background-image: url(1.gif); }

- Властивість `background-attachment`

Визначає, чи фіксується фонове зображення, або прокручується разом зі вмістом сторінки.

- Властивість `background-position`

Визначає розташування фонового зображення. Координати можна задавати у відсотках ширини екрану, у фіксованих одиницях (пікселі, сантиметри тощо) або за допомогою слів `top`, `bottom`, `center`, `left` і `right`.

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

**Ієрархічне дерево**

HTML-документ являє собою ієрархічне дерево. У кожного елемента (крім кореневого) є тільки один батьківський елемент, тобто елемент, всередині якого він розташовується. Кореневий елемент не має батьківського елемента. Наприклад, найпростіша сторінка виглядає так:

<html>

<head></head>

<body>

<p>Текст документа</p>

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

</body>

</html>

Для цієї сторінки можна уявити таке ієрархічне дерево:

html

├── head

└── body

├── p

└── p.text

└── span

Ця структура відображає вкладеність елементів. В даному прикладі видно, що елемент `span` має батьківський елемент `p.text`, а у `p.text` батьківським елементом є `body`. Ієрархічна структура документа визначає основи концепції спадковості.

**Спадковість**

Спадковість в CSS – це механізм, за допомогою якого значення властивостей батьківського елемента передаються його дочірнім елементам. Стилі, застосовані до одного елемента, успадковуються всіма його нащадками (вкладеними елементами), якщо вони не перевизначені явно. Наприклад, розмір шрифту та його колір часто застосовуються до `body`, щоб всі елементи всередині мали ті ж властивості. Спадковість дозволяє скоротити розмір таблиці стилів, але якщо стилів багато, то відстеження того, який батьківський елемент встановив певну властивість, стає складнішим.

**Спадкові властивості**

До спадкових властивостей належать, перш за все, 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`

Також до спадкових властивостей належать `list-style`, `cursor`, `visibility`, `border-collapse` та деякі інші. Ці властивості можна і потрібно задавати через батьківські елементи, 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 у стилях самих блоків.

**Каскадність**

CSS розшифровується як "Cascading Style Sheets" або "каскадні таблиці стилів". Каскадність означає, що до одного і того ж елемента може застосовуватися кілька CSS-правил (наборів CSS-властивостей). Серед цих властивостей можуть бути й конфліктуючі. Існують інструкції, які визначають, який саме набір властивостей буде застосований до елемента. Наприклад, для елемента:

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

CSS-правила можуть існувати як мінімум у трьох різних місцях:

1. У підключеному файлі `style.css` для селекторів `p` або `.text`.

2. В атрибуті `style`.

3. У стандартних стилях відображення, вбудованих у браузер.

Каскадність визначає, які властивості з цих джерел застосовуються до даного абзацу. Існують три основні концепції, які керують порядком застосування CSS-властивостей:

1. Важливість (importance)

2. Специфічність (specificity)

3. Порядок вихідного коду (source order)

# Опис постановки задачі

Основною задачею було створити веб-сайт у вигляді резюме. Основною 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. Мій проєкт наповнений ненумерованими списками, відформатованим текстом, а саме текстом зі зміненим кольором, розміром шрифту, стилю і т.д. Також у моїй роботі є такий контент як фото, активні посилання та елементи оформлення об’єктів для кращого сприйняття їх персоною, яка буде переглядати мій веб-сайт.

**Розроблена веб-сторінка**

**HTML файл:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Programmer's Resume</title>

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

    <script src="https://kit.fontawesome.com/b6d9c9201e.js" crossorigin="anonymous"></script>

</head>

<body>

    <header>

        <div class="photo">

            <img src="image.jpg" class="image" width="150"  height="150" />

        </div>

        <h1>Tymur Kyrylenko</h1>

        <p>Software Engineer</p>

    </header>

    <section class="about">

        <h2>About Me</h2>

        <p>Dedicated and passionate Software Engineering student currently pursuing a degree in Computer Science. Eager to leverage academic knowledge and practical experience to contribute to innovative software projects. Possesses strong problem-solving skills, proficiency in multiple programming languages, and a keen interest in emerging technologies. Committed to continuous learning and enthusiastic about starting a professional career in software development to drive impactful and efficient solutions.</p>

    </section>

    <section class="skills">

        <h2>Skills</h2>

        <ul>

            <li>Python</li>

            <li>С++</li>

            <li>HTML, CSS</li>

        </ul>

    </section>

    <section class="education">

        <h2>Education</h2>

        <ul>

            <li>

                Lyceum "Holosiivskyi"  №241

            </li>

            <li>

                Optical and Mechanical College of the Taras Shevchenko National University of Kyiv

            </li>

        </ul>

    </section>

    <section class="contact">

        <h2>Contact Information</h2>

        <div class="social-buttons">

            <a href="mailto:timur.kirilenko2017@gmail.com" target="\_blank" class="button email"><b><i class="fa-solid fa-envelope"></i>&nbsp;Email</b></a>

            <a href="https://www.instagram.com/timdenofff/" target="\_blank"class="button instagram"><b><i class="fa-brands fa-instagram"></i>&nbsp;Instagram</b></a>

            <a href="https://www.facebook.com/share/1Ub62V6Zs4XgWZDK/?mibextid=LQQJ4d" target="\_blank" class="button facebook"><b><i class="fa-brands fa-facebook"></i>&nbsp;Facebook</b></a>

            <a href="https://github.com/Timdenoff" target="\_blank" class="button github"><b><i class="fa-brands fa-github"></i>&nbsp;GitHub</b></a>

        </div>

    </section>

    <span class="icon icon\_vk"></span>

    <footer>

        <p>&copy; 2024 Tymur Kyrylenko</p>

    </footer>

</body>

</html>

**CSS файл:**

body {

    height: 100%;

    font-family: Arial, sans-serif;

    line-height: 1.6;

    margin: 0;

    padding: 0;

    background-color: #f4f4f4;

}

header {

    background-color: #333;

    color: #fff;

    padding: 20px 0;

    text-align: center;

}

header h1 {

    margin: 0;

    font-size: 2.5em;

}

header p {

    margin: 0;

    font-size: 1.2em;

}

section {

    padding: 20px;

    margin: 20px;

    background-color: #fff;

    border-radius: 8px;

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

}

h2 {

    border-bottom: 2px solid #333;

    padding-bottom: 10px;

}

ul {

    list-style-type: none;

    padding: 0;

}

ul li {

    background: #f4f4f4;

    margin: 5px 0;

    padding: 10px;

    border-left: 5px solid #333;

}

footer {

    text-align: center;

    padding: 20px 0;

    background-color: #333;

    color: #fff;

    position: bottom;

    bottom: 0;

    width: 100%;

}

.social-buttons {

    margin-top: 20px;

}

.button{

    display: inline-block;

    width: 98%;

    padding: 10px 20px;

    margin: 5px;

    font-size: 1em;

    color: #000000;

    text-align: center;

    outline: none;

    text-decoration: none;

    background: none;

    border-radius: 5px;

    border: 2px solid black;

}

.photo{

    text-align: center;

}

.image{

    border-radius: 100px;

}

.button:hover{

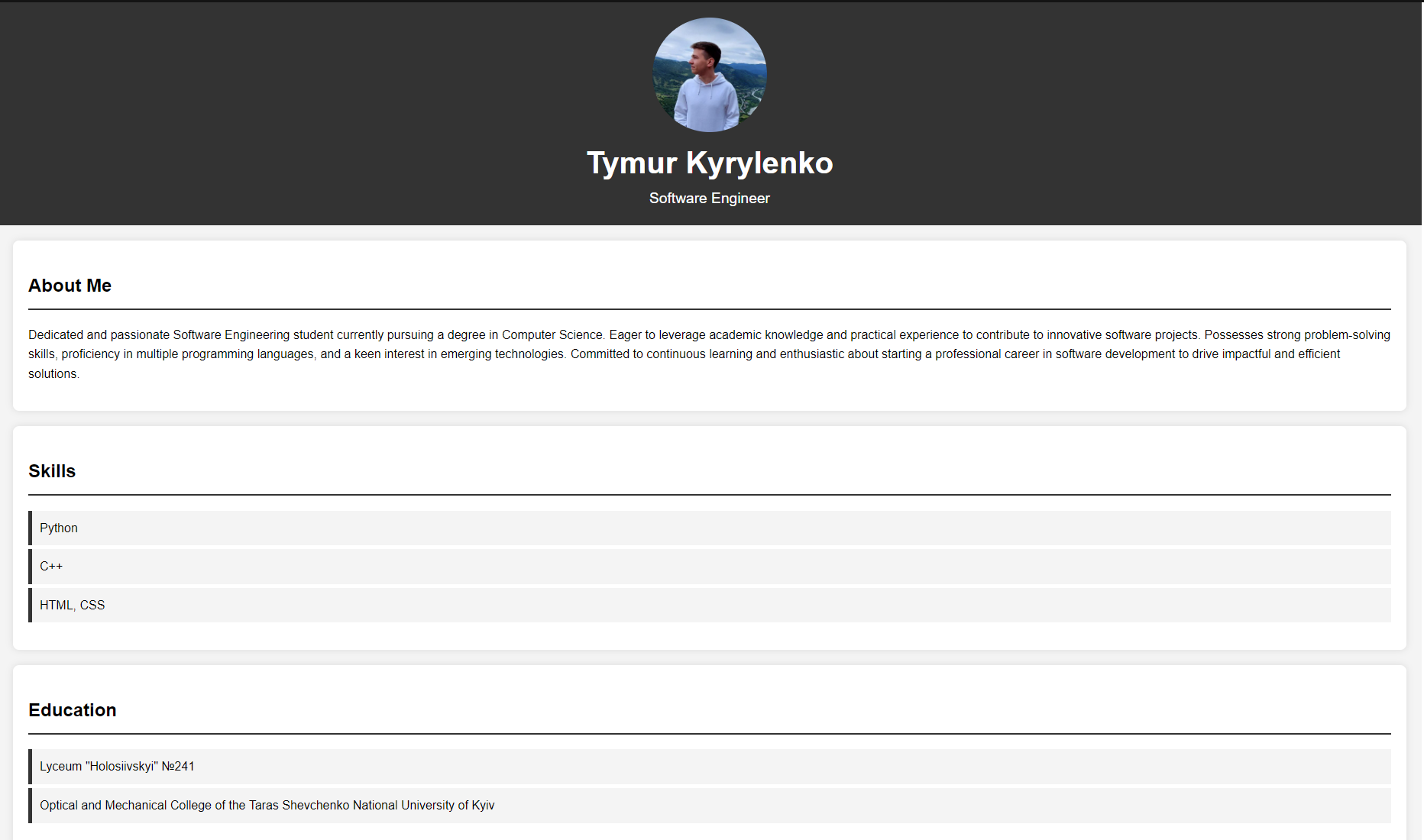
    background-color: #363535;

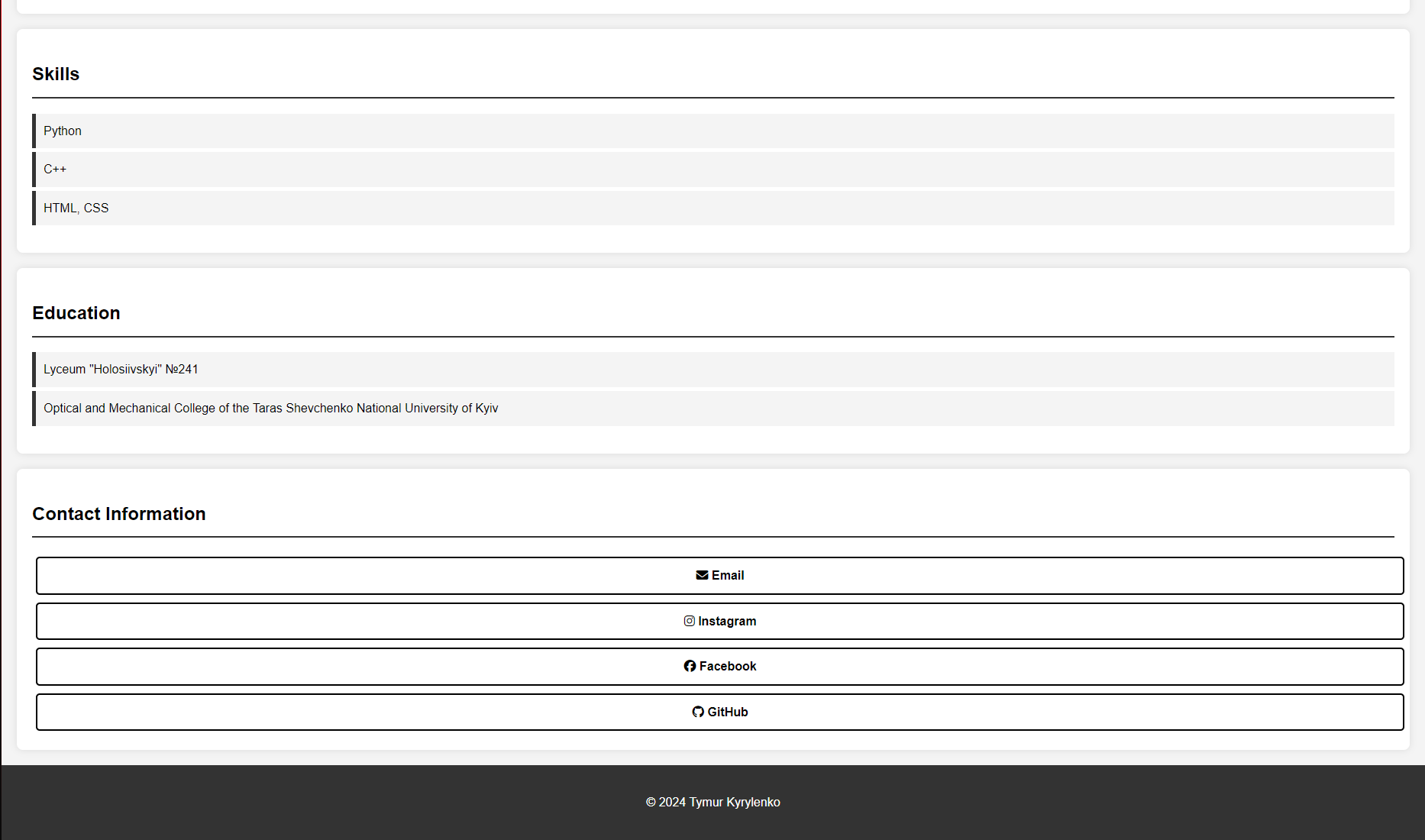
    color:white;

    transition: 0.2s;

}

**Перевірка/тестування веб-сторінки**

****

****

# Висновки

## ***До 4 теми:***

Під час навчальної практики з розділу "Практична робота з C++" я суттєво розширив свої знання та здобув цінні практичні навички, які є важливими для мого подальшого розвитку в програмуванні. Основні аспекти, які я освоїв під час практики, включають:

1. Введення та виведення в консольних програмах:

Я навчився використовувати стандартні бібліотеки C++ для взаємодії програм з користувачем через консоль. Ці знання є фундаментальними для створення складніших програм і розробки ефективних та зрозумілих користувацьких інтерфейсів у консольних додатках.

2. Робота з файлами: запис та читання:

Практичні завдання з роботи з файлами допомогли мені зрозуміти, як зберігати та обробляти дані у зовнішніх файлах. Я навчився працювати з текстовими та двійковими файлами, що є критично важливим для створення додатків, які оперують великими обсягами даних або зберігають дані між сесіями.

3. Формати файлів та їх обробка:

Ознайомлення з форматами JSON та XML дало мені розуміння структурованого зберігання даних і їх обміну між різними системами. Це дозволяє інтегрувати мої програми з іншими сервісами та додатками, забезпечуючи ефективну обробку даних.

4. Системи контролю версій (VCS):

Використання системи контролю версій Git допомогло мені зрозуміти принципи управління версіями коду, відстежування змін та командної роботи. Ці навички є необхідними для сучасних команд розробників, де контроль за змінами коду та спільна робота є критично важливими.

5. Мова Markdown для опису проектів:

Markdown виявився простою та ефективною мовою розмітки для створення документації. Я навчився створювати зрозумілі, структуровані та привабливі технічні описи, що полегшує комунікацію в команді та з користувачами.

6. Створення ігрової програми:

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

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

## ***До 5 теми:***

Під час навчальної практики з розділу "Веб-дизайн та HTML" я отримав ґрунтовні знання та практичні навички, які є ключовими для мого подальшого розвитку в галузі веб-розробки. Вивчення та застосування HTML, CSS і основних принципів веб-дизайну дозволило мені створювати сучасні та адаптивні веб-сторінки, які відповідають актуальним вимогам та стандартам індустрії.

Під час практики я опанував базові концепції структури веб-документів, методи їх стилізації та взаємодії з користувачем. Я навчився використовувати різні інструменти для оптимізації роботи з HTML-кодом та створення зручного й естетично привабливого інтерфейсу. Особливу увагу було приділено принципам адаптивного дизайну, що забезпечує коректне відображення сайтів на різних пристроях.

Практичні завдання та проекти допомогли закріпити теоретичні знання та наблизили мене до реальних умов роботи веб-розробника. Робота над власними проектами дала змогу відчути процес створення веб-сайтів від початкового етапу до їх повної реалізації, включаючи планування, розробку та тестування.

Завдяки отриманому досвіду я значно підвищив свої професійні компетенції та можу впевнено продовжувати вдосконалюватися в напрямку веб-дизайну та фронтенд-розробки. Навички, здобуті під час практики, стануть міцною основою для подальшого навчання та професійного зростання у сфері інформаційних технологій.

# 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