ВІДОКРЕМЛЕННИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ

«ОПТИКО-МЕХАНІЧНИЙ ФАХОВИЙ КОЛЕДЖ КИЇВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ІМЕНІ ТАРАСА ШЕВЧЕНКА»

Циклова комісія спеціальності «Інженерія програмного забезпечення»

**З В І Т**

**З НАВЧАЛЬНОЇ ПРАКТИКИ**

спеціальність 121 Інженерія програмного забезпечення

освітньо-кваліфікаційний рівень «фаховий молодший бакалавр»

|  |  |  |
| --- | --- | --- |
| ПІДСУМКОВА ОЦІНКА  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оцінка цифрою та прописом)  Керівники практики від коледжу:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. С. Нікітенко  (оцінка) (підпис)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т. І. Лумпова  (оцінка) (підпис)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ М. С. Ходжаєв  (оцінка) (підпис) |  | Виконав: |
| студента 2-го курсу групи ІПЗ-23 |
| Cеменової Валерії Сергіївни  залікова книжка № \_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (підпис виконавця)  «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2024 року |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. В. Степанюк  (оцінка) (підпис) |  |  |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2024 року |  |  |

м. Київ — 2024

ЗАТВЕРДЖУЮ

Директор ОМФК КНУ

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

«17» травня 2024 р.

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| на період | *навчальної* | | практики | |
|  | (навчальної, технологічної, виробничої) | |  | |
| *Семенової Валерії Сергіївни* | | | | |
| (П.І.Б студента.) | | | | |
| Спеціальність | *121 Інженерія програмного забезпечення* | Група | | *ІПЗ-23* |

|  | **Питання для вивчення** | **Відмітки про виконання** |
| --- | --- | --- |
| **1** | **Вступне заняття.** Вивчення питань охорони праці. Ознайомлення з електронною бібліотекою коледжу. Вивчення правил оформлення звітної документації. |  |
| **2** | **Розділ "Інформатика**" |  |
| 2.1 | Засобами MS Word розробити документ складної структури та зберегти у різних форматах |  |
| 2.2 | Розв’язування рівнянь |  |
| 2.3 | Розв’язування оптимізаційних задач |  |
| 2.4 | Основи статистичного аналізу даних. Ряди даних. Кореляційний аналіз даних |  |
| 2.5 | Обчислення основних статистичних характеристик вибірки |  |
| 2.6 | Візуалізація рядів і трендів даних |  |
| 2.7 | Розв‘язок задач за допомогою засобів табличного редактора MS Excel |  |
| 2.8 | Проектування бази даних та створення бази даних *MS ACCESS* |  |
| 2.9 | Основні поняття реляційних баз даних |  |
| 2.10 | Введення, коригування, вилучення інформації з бази даних *MS ACCESS* |  |
| 2.11 | Вибірка, пошук, фільтрація даних в *MS ACCESS* |  |
| 2.12 | Виконання практичних завдань за допомогою засобів табличного редактора MS Access |  |

|  | **Питання для вивчення** | **Відмітки про виконання** |
| --- | --- | --- |
| **3** | **Розділ "Програмування С++"** |  |
| 3.1 | Алгоритмізація обчислювальних процесів. Розробка постановки задачі, визначення вхідних та вихідних даних, оформлення блок-схем. |  |
| 3.2 | Типи даних в мові С/С++, внутрішнє представлення даних, операції перетворення типів даних. |  |
| 3.3 | Оператор присвоювання. Операції введення- виведення даних |  |
| 3.4 | Програмування лінійних та розгалужених обчислювальних процесів |  |
| 3.5 | Програмування циклічних обчислювальних процесів |  |
| 3.6 | Складання програм |  |
| 3.7 | Виконання тестів |  |
| 4 | **Розділ "** **Практична робота з С++"** |  |
| 4.1 | Тема 1. Введення та виведення у базових консольних програмах |  |
| 4.2 | Тема 2. Структури даних для управління станом програми |  |
| 4.3 | Тема 3. Робота з файлами: запис та читання |  |
| 4.4 | Тема 4. Формати файлів та їх обробка |  |
| 4.5 | Тема 5. Системи контролю версій (VCS) |  |
| 4.6 | Тема 6. Мова Markdown для опису проекті |  |
| 4.7 | Розроблення ігрової програми |  |
| 5 | Розділ ”Веб-дизайн та HTML” |  |
| 5.1 | Мова розмітки HTML та структура Веб-сторінки |  |
| 5.2 | Розмітка тексту за допомогою HTML. Зображення та посилання |  |
| 5.3 | Каскадні таблиці стилів CSS та їх структура |  |
| 5.4 | Спадковості і каскадування за допомогою CSS |  |
| 5.5 | Розробка та оформлення Постановки задачі |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | **Питання для вивчення** | | | | **Відмітки про виконання** | | |
| 5.9 | | Розробка HTML документу | | | |  | | |
| **6** | | Систематизація матеріалів, оформлення звіту з навчальної практики | | | |  | | |
| Голова циклової комісії | |  |  | Савєльєва І.В. | |  |  |
|  | | (підпис) |  | (ініціали, прізвище) | |  | (дата) |
| Керівники практики від коледжу | |  |  |  | |  |  |
|  | |  |  | Нікітенко А.С. | |  | 17.05.2024р. |
|  | | (підпис) |  | (ініціали, прізвище) | |  | (дата) |
|  | |  |  | Лумпова Т.І. | |  | 17.05.2024р. |
|  | | (підпис) |  | (ініціали, прізвище) | |  | (дата) |
|  | |  |  | Нікітенко А.С. | |  | 17.05.2024р. |
|  | | (підпис) |  | (ініціали, прізвище) | |  | (дата) |
|  | |  |  | Лумпова Т.І. | |  | 17.05.2024р. |
|  | | (підпис) |  | (ініціали, прізвище) | |  | (дата) |
|  | |  |  |  | |  |  |
|  | |  |  |  | |  |  |

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

4

Розроб.

Перевір.

Реценз.

В.Я

Н. Контр.

В.Я

Затверд.

Звіт з навчальної практики

Літ.

Аркушів

ІПЗ-22

ОМФК.0121 НП 20

Зміст

[ВСТУП 6](#_Toc168176310)

[1 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ 7](#_Toc168176311)

[2 РОЗДІЛ "ІНФОРМАТИКА" 9](#_Toc168176312)

[**2.1** **Розроблення засобами MS Word документа складної структури** 9](#_Toc168176313)

[**2.2 Розв‘язок задач за допомогою засобів табличного редактора MS Excel** 9](#_Toc168176314)

[**2.3 Виконання** **практичних завдань за допомогою засобів табличного редактора MS Access** 9](#_Toc168176315)

[3 РОЗДІЛ "ПРОГРАМУВАННЯ С++" 10](#_Toc168176316)

[**3.1 Теоретичні відомості** 10](#_Toc168176317)

[3.1.1 Загальні поняття. Елементи мови С++ - константи, змінні, операції, перетворення типів 10](#_Toc168176318)

[3.1.2 Арифметичні операції 10](#_Toc168176319)

[3.1.3 Поняття вхідного та вихідного потоку, найпростіші математичні функції 10](#_Toc168176320)

[3.1.4 Програмування лінійних та розгалужених обчислювальних процесів 10](#_Toc168176321)

[3.1.5 Програмування циклічних обчислювальних процесів 10](#_Toc168176322)

[**3.2 Практичне завдання** 11](#_Toc168176323)

[3.2.1 Завдання 11](#_Toc168176324)

[3.2.2 Постановка задачі 11](#_Toc168176325)

[3.2.3 Код програми 11](#_Toc168176326)

[3.2.4 Скріншот виконання програми 11](#_Toc168176327)

[**3.3 Виконання тестів** 11](#_Toc168176328)

[4 РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++" 12](#_Toc168176329)

[**4.1 Тема 1. Введення та виведення у базових консольних програмах** 12](#_Toc168176330)

[**4.2 Тема 2. Структури даних для управління станом програми** 16](#_Toc168176331)

[**4.3 Тема 3. Робота з файлами: запис та читання** 20](#_Toc168176332)

[**4.4 Тема 4. Формати файлів та їх обробка** 22](#_Toc168176333)

[**4..5 Тема 5. Системи контролю версій (VCS)** 22](#_Toc168176334)

[**4.6 Тема 6. Мова Markdown для опису проекті** 26](#_Toc168176335)

[**4.7 Розроблення ігрової програми** 26](#_Toc168176336)

[4.7.1 Код програми 26](#_Toc168176337)

[4.7.2 Скріншоти виконання програми 29](#_Toc168176338)

[5 РОЗДІЛ ”ВЕБ-ДИЗАЙН ТА HTML” 31](#_Toc168176339)

[**5.1 Мова розмітки HTML та структура Веб-сторінки** 31](#_Toc168176340)

[**5.2 Розмітка тексту за допомогою HTML. Зображення та посилання** 33](#_Toc168176341)

[Абзаци 33](#_Toc168176342)

[Цитати 35](#_Toc168176343)

[**5.3 Каскадні таблиці стилів CSS та їх структура** 36](#_Toc168176344)

[**5.4 Спадковості і каскадування за допомогою CSS** 38](#_Toc168176345)

[5.5.1 Постановки задачі 39](#_Toc168176346)

[5.5.2 HTML документ 40](#_Toc168176347)

[ВИСНОВКИ 45](#_Toc168176348)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ 46](#_Toc168176349)

[ДОДАТКИ 48](#_Toc168176350)

ОМФК.0121 НП 20

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

# ВСТУП

Автор: Семенова Валерія Сергіївна

Дата:

Мета:

Перелік вивчених тем:

# 1 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ

Кабінет інформатики та інформаційно-комунікаційних технологій на­сичений різноманітною електронною технікою, яка живиться від елект­ричної мережі з напругою змінного струму 220 В. Це потребує дотриман­ня додаткових правил безпеки під час проведення навчальних занять і позаурочних заходів.

**До початку роботи:**

* з дозволу вчителя займіть своє робоче місце за комп’ютером, налаш­туйте висоту стола, стільця, підставки для ніг, кут нахилу монітора так, щоб (рис. 2.57):
* середина екрана монітора знаходилася трохи нижче горизонтальної лінії зору;
* відстань від очей до поверхні екрана становила 40-80 см, залежно від розмірів об’єктів на екрані;
* лінія від очей до центра екрана монітора була перпендикулярна до площини екрана;
* пальці рук вільно лежали на клавіатурі;
* руки утворювали в ліктьовому суглобі кут, близький до 90;
* клавіатура лежала на поверхні стола або на спеціальній полиці на відстані 10-30 см від краю і була нахилена під кутом 5-15;
* спина опиралася на спинку стільця;
* ноги опиралися на підлогу або на спеціальну підставку;
* наведіть лад на робочому столі, приберіть з нього предмети, які не потрібні для роботи;
* перевірте чистоту своїх рук, за потреби вимийте їх і витріть насухо;
* за потреби, з дозволу вчителя і тільки спеціальною серветкою протріть екран монітора, клавіатуру, килимок і мишу;
* перевірте відсутність зовнішніх пошкоджень комп’ютерного облад­нання;
* з дозволу вчителя ввімкніть ком­п’ютер.

**Під час роботи:**

* тримайте робоче місце охайним, не розміщуйте на ньому сторонніх ре­чей;
* витримуйте правильну поставу - не нахиляйтеся близько до поверхні ек­рана, не згинайтеся, тримайте руки без напруження;
* після 15-20 хв роботи або при від­чутті втоми виконайте комплекс вправ для очей і для зняття м’язово­го напруження;
* не намагайтеся самостійно усунути перебої в роботі комп’ютера, при їх виникненні негайно покличте учителя;
* не торкайтеся задніх стінок монітора та системного блока, не чіпай­те дротів живлення;
* не торкайтеся екрана монітора руками (це забруднює його і порушує антиблікове покриття);
* категорично забороняється знімати кришки корпусів пристроїв комп’ютера, самостійно без дозволу вчителя приєднувати і від’єдну­вати пристрої комп’ютера;
* акуратно вставляйте і виймайте змінні носії.

**Після закінчення роботи:**

* приберіть своє робоче місце;
* з дозволу вчителя вимкніть комп’ютер або закінчіть сеанс роботи.

# 2 РОЗДІЛ "ІНФОРМАТИКА"

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

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

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

# 3 РОЗДІЛ "ПРОГРАМУВАННЯ С++"

## **3.1 Теоретичні відомості**

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

### 

### 3.1.2 Арифметичні операції

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

### 3.1.4 Програмування лінійних та розгалужених обчислювальних процесів

### 3.1.5 Програмування циклічних обчислювальних процесів

## **3.2 Практичне завдання**

### 3.2.1 Завдання

### 3.2.2 Постановка задачі

Вхідні дані

Вихідні дані

Блок-схема алгоритму роботи програми представлена на рис. ххх

Блок-схема

Рисунок хх - Блок-схема алгоритму роботи програми

### 3.2.3 Код програми

### 3.2.4 Скріншот виконання програми

Скріншот виконання програми представлена на рис. ххх

Скріншот

Рисунок хх - Скріншот виконання програми

## **3.3 Виконання тестів**

# 

# 4 РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++"

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

У базових консольних програмах C++ об'єкт `cout` використовується для виведення даних на заданий пристрій. Синтаксис `cout << data` дозволяє виводити змінні, константи або вирази. Для форматування виведення даних використовуються різні маніпулятори, такі як `hex` та `oct`, які задають виведення чисел у 16-ковій та 8-ковій системах відповідно. Маніпулятори `setw()`, `setfill()`, і `setprecision()` змінюють стан об’єкта `cout`: `setw()` задає ширину поля для виведення, `setfill()` заповнює незаповнені позиції встановленим символом, а `setprecision(n)` визначає кількість знаків після коми.

Маніпулятори `setiosflags` і `resetiosflags` встановлюють або скидають певні глобальні прапори, які використовує клас `iostream` для визначення поведінки при введенні та виведенні. Наприклад, `setiosflags()` встановлює зазначені прапори, а `resetiosflags()` їх очищає.

У класичному C для виведення даних використовується функція `printf`, яка також доступна у C++. Синтаксис `printf(char \*format, ...)` дозволяє виводити дані з використанням специфікаторів формату. Форматний рядок складається з символів, що виводяться без змін, і спеціальних символів формату, які задають перетворення даних. Специфікатори включають `%d` або `%i` для десяткового цілого зі знаком, `%u` для десяткового цілого без знака, `%x` для шістнадцятирічного цілого без знака, `%f` для чисел з плаваючою точкою, `%e` для експоненційного запису чисел, `%c` для символів, `%s` для рядків, і `%p` для вказівників. Символи `L` або `l` використовуються для довгих типів даних, таких як `long double` або `long int`.

Для введення даних у C++ використовується об’єкт `cin`, синтаксис якого `cin >> змінна` дозволяє вводити змінні. При введенні рядків кожне слово вводиться окремо, а роздільником між ними є пробіл або Enter.

Функція `scanf`, яка також доступна у C, дозволяє вводити дані за допомогою форматних рядків, аналогічних до `printf`. Перед іменами змінних у `scanf` потрібно ставити символ `&`, який означає "взяти адресу" змінної для її зчитування.

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

У темі "Структури даних для управління станом програми" основні поняття включають опис найпоширеніших структур даних у C++ та їхні реалізації.

Масив є інструментом для представлення однорідних даних. Він складається з елементів, кожен з яких має індекс. Властивості масиву включають тип елементів, назву, розмірність і діапазони індексів.

У C++ індекси починаються з 0. Звернення до елемента масиву здійснюється за допомогою індексів у квадратних дужках.

Одновимірний масив, або вектор, має один індекс для кожного елемента. Двовимірний масив, або матриця, має два індекси: номер рядка та номер стовпчика.

**Зв'язний список** — лінійно упорядкована структура (послідовність) даних, елементи якої — вузли — містить два види даних:

* фактичні дані довільного типу;
* вказівник (посилання) на наступний вузол послідовності.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Основні дії зі зв'язним списком:

* додавання елемента у список;
* вилучення елемента зі списку;
* пошук елемента (з певними властивостями) у списку.

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

**Стек** — лінійно упорядкована структура (послідовність) даних, у яку можна вставити елемент або з якої можна вилучити елемент лише з одного кінця.

Стек нагадує стос аркушів паперу у вузькій шухляді: щоб узяти певний аркуш, потрібно спочатку прибрати всі аркущі над ним. Кажуть, що для стеку діє принцип: «Останній зайшов, перший вийшов» (англій­ською LIFO: Last In First Out).

**Изображение выглядит как снимок экрана, линия, дизайн

Автоматически созданное описание**

Основні операції зі стеком:

* вставлення елемента в кінець стеку;
* вилучення елемента з кінця;
* відображення вмісту останнього долученого елемента.

Наявні вбудовані функції для роботи з векторами дозволяють втілити ідею стеку. Контейнер stack безпосередньо втілює принцип: «Останній зайшов, перший вийшов» (англій­ською LIFO: Last In First Out):

**Черга** — лінійно упорядкована структура (послідовність) даних, у яку можна вставити елемент з одного кінця або з якої можна вилучити елемент з іншого кінця

Изображение выглядит как Шрифт, снимок экрана, текст, диаграмма

Автоматически созданное описание

Прообразом цієї структури є чергу людей у магазині: того, що став першим, буде обслужено першим. Якщо розглядати чергу щодо доступу до даних, то вона реалізує принцип: «Першим зайшов, перший вийшов» (англій­ською First In First Out, FIFO). Інакше кажучи, після додавання нового елементу всі елементи, які було додано до цього, потрібно вилучити до того, як новий елемент буде вилучено.

Основні операції з чергою:

* вставлення елемента в кінець черги;
* вилучення елемента з початку черги.

Контейнер queue безпосередньо втілює принцип: «Першим зайшов, перший вийшов» (англій­ською First In First Out, FIFO):

**Асоціативний масив** (словник в інших мовах програмування) — це вбудована структура даних для збереження даних у форматі ключ-значення.

Цю структура даних легко уявити як шафу з підписаними шухлядами. Усі дані зберігають у шухлядах. За назвою можна легко знайти шухляду і взяти з неї те значення, яке там розташовано.

Изображение выглядит как текст, Шрифт, белый, дизайн

Автоматически созданное описание

На відміну від звичайних шаф, в асоціативний масив будь-коли можна додати нові «шухляди» або вилучити вже наявні.

**Set** — контейнер для збереження унікальних значень (кожне не більше одного разу), що втілює математичне поняття множини.

При додаванні нового елемента в даному контейнері буде проведено упорядкування елементів. Повторювані елементи буде вилучено.

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

Робота з файлами у мові програмування C++ дозволяє програмістам зберігати та отримувати дані з файлової системи комп'ютера. Це може бути корисно для збереження конфігураційних даних, роботи зі структурованими даними, збереження результатів обчислень та багато іншого.

Для роботи з файлами у C++ використовуються два основних класи: ifstream і ofstream. Перший використовується для читання з файлу, а другий - для запису у файл. Вони базуються на базовому класі fstream, який також можна використовувати як універсальний інструмент для роботи з файлами.

Щоб використовувати файли у програмі, спочатку треба їх відкрити. Це робиться за допомогою конструктора класу ifstream або ofstream, якому передається шлях до файлу. Якщо файл існує, його відкриють для роботи, а якщо ні, то він буде створений.

Після відкриття файлу, програма може зчитувати дані з нього, записувати дані у нього або виконувати будь-які інші операції, пов'язані з файлом. Після завершення роботи з файлом його слід закрити за допомогою методу close().

Приклади використання:

* Для читання з файлу:

ifstream inputFile("example.txt");

* Для запису в файл:

ofstream outputFile("example.txt");

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

Текстові файли:

Текстові файли містять дані у формі тексту, що може бути зрозумілим для людини. Це може бути рядок символів, числа або інші дані, які можна зчитати та редагувати звичайним текстовим редактором.

Дані у текстових файлах зазвичай розділяються символами пробіла, комою, крапкою з комою або іншими розділовими знаками.

Бінарні файли:

Бінарні файли містять дані у бінарному форматі, який не призначений для прямого читання людиною. Дані зазвичай зберігаються у вигляді послідовності бітів, що представляють числа, структури даних тощо.

Бінарні файли зазвичай потребують спеціальної обробки для читання та запису, оскільки дані можуть бути представлені у складних форматах.

У мові програмування C++ для читання та запису файлів використовуються класи ifstream для читання та ofstream для запису. Крім того, для обробки бінарних файлів можна використовувати функції fread та fwrite.

Іноді для обробки певних форматів файлів (наприклад, JSON, XML) використовують спеціалізовані бібліотеки, які надають зручний інтерфейс для читання та запису даних у цих форматах.

## **4..5 Тема 5. Системи контролю версій (VCS)**

Система контролю версій (VCS) - це програмний інструмент, який дозволяє розробникам зберігати історію змін у вихідному коді свого програмного забезпечення. Основна мета VCS полягає в тому, щоб забезпечити зручну та безпечну роботу з кодом для розробників та команд розробників.

Використання систем контролю версій дозволяє:

* Зберігати історію змін: Кожне збереження (commit) в системі контролю версій записує зміни, внесені у вихідний код. Це дозволяє розробникам переглядати, аналізувати та повертатися до попередніх версій коду у разі потреби.
* Спільно працювати над проектом: VCS дозволяє розробникам працювати над проектом одночасно та координувати свою роботу. Кожен розробник може вносити свої зміни, а система автоматично об'єднує їх.
* Експериментувати та розгалужувати розробку: Розгалуження розробки (branching) дозволяє розробникам експериментувати з новими функціями або розвивати окремі функціональність у відокремлених гілках, не впливаючи на основний код проекту.

Варіанти систем контролю версій включають:

* Локальні системи контролю версій: Розробники працюють локально на своїх комп'ютерах і зберігають історію змін лише на своєму пристрої. Це простий підхід, але він не підходить для спільної роботи в команді.
* Централізовані системи контролю версій: Всі зміни зберігаються на центральному сервері, і розробники працюють з віддаленими копіями коду. Цей підхід підходить для команд, але може стати однією точкою відмови.
* Розподілені системи контролю версій: Кожен розробник має повну копію історії проекту. Це дозволяє працювати навіть без підключення до мережі та має більшу стійкість до відмов серверів.

Важливо також відзначити, що історія розвитку систем контролю версій проходить три покоління, починаючи з простих локальних систем і закінчуючи розподіленими системами з розгалуженням розробки та можливістю злиття гілок.

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

**Markdown** — полегшена мова розмітки даних, яку створено з ухилом на прочитність та зручність у публікації з подальшим перетворенням її на structurally valid XHTML або HTML. Чимало ідей для мови позичено з існуючих домовленостей у розмітці тексту в електронних листах. Першу реалізацію Markdown написано Грубером на Perl, але з часом з'явились багато реалізацій від сторонніх розробників. Реалізація на Perl розповсюджується за ліцензією BSD.

Такі сайти, як GitHub, Reddit та Stack Overflow використовують Markdown для полегшення обговорень між користувачами.

Ось декілька основних функцій мови Markdown, які були використані у оформлені мого репозиторію:

* Заголовки

Для того що б додати заголовок потрібно додати від одного до шести #-символів перед тестом заголовка. Кількість #-символів, яка використовується, визначатиме рівень ієрархії та розмір гарнітури заголовка. Наприклад:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Коли використовуєтсья два або більше заголовків, GitHub автоматично створює зміст, до якого можна отримати доступ, клацнувши в заголовку файлу. Кожна назва заголовка перерахована у змісті, і користувач може клацнути назву, щоб перейти до вибраного розділу.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

* Стилізація тексту

|  |  |  |  |
| --- | --- | --- | --- |
| **Стиль** | **Синтаксис** | **Приклад** | **Вихідний результат** |
| Жирний | \*\* \*\* або \_\_ \_\_ | \*\*This is bold text\*\* | **This is bold text** |
| Курсивний | \* \* або \_ \_ | \_This text is italicized\_ | This text is italicized |
| Закреслення | ~~ ~~ | ~~This was mistaken text~~ | ~~This was mistaken text~~ |
| Жирний і вкладений курсив | \*\* \*\* and \_ \_ | \*\*This text is \_extremely\_ important\*\* | **This text is extremely**  **important** |
| Усі жирні та курсивні | \*\*\* \*\*\* | \*\*\*All this text is important\*\*\* | ***All this text is important*** |

* Посилання

Користувач може створити вбудоване посилання, помістивши текст посилання в квадратні дужки [ ], а потім увімкнувши URL-адресу в дужки ( ), або скористатися комбінацією клавіш Command+K, щоб створити посилання. Вибравши текст, можна вставити URL-адресу з буфера обміну, щоб автоматично створити посилання з виділеного тексту.

Також можна створити гіперпосилання Markdown, виділивши текст і скориставшись комбінацією клавіш Command+V. Якщо користувач хоче замінити текст посиланням, слід скористатися комбінацією клавіш Command+Shift+V.

Изображение выглядит как текст, Шрифт, снимок экрана, белый

Автоматически созданное описание

## Зображення

Користувач може відобразити зображення, додавши ! і альтернативний текст в [ ]. Альтернативний текст – це короткий текстовий еквівалент інформації на зображенні. Потім слід помістити посилання на зображення в дужки ().

Изображение выглядит как текст, мультфильм, графическая вставка, Анимация

Автоматически созданное описание

* Список

Невпорядкований список можна створити, поставивши перед одним або декількома рядками тексту -, \* або +.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Щоб упорядкувати список, перед кожним рядком слід додати номер.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

## **4.7 Розроблення ігрової програми**

### 4.7.1 Код програми

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

#include <ctime>

#include <json/json.h>

using namespace std;

// Function to write logs to a CSV file

void write\_logs(const string& filename, const string& message) {

ofstream file(filename, ios::app);

if (file.is\_open()) {

time\_t now = time(0);

tm\* timeinfo = localtime(&now);

char timestamp[20];

strftime(timestamp, sizeof(timestamp), "%Y-%m-%d %H:%M:%S", timeinfo);

file << timestamp << "," << message << endl;

file.close();

} else {

cerr << "Error: Unable to open file " << filename << endl;

}

}

// Function to read logs from a CSV file

vector<pair<string, string>> read\_logs(const string& filename) {

vector<pair<string, string>> logs;

ifstream file(filename);

if (file.is\_open()) {

string line;

while (getline(file, line)) {

size\_t pos = line.find(',');

if (pos != string::npos) {

string timestamp = line.substr(0, pos);

string message = line.substr(pos + 1);

logs.push\_back(make\_pair(timestamp, message));

}

}

file.close();

} else {

cerr << "Error: Unable to open file " << filename << endl;

}

return logs;

}

// Function to read wins from a JSON file

Json::Value read\_wins(const string& filename) {

Json::Value wins;

ifstream file(filename);

if (file.is\_open()) {

file >> wins;

file.close();

} else {

cerr << "Error: Unable to open file " << filename << endl;

}

return wins;

}

// Function to write wins to a JSON file

void write\_wins(const string& filename, const Json::Value& wins) {

ofstream file(filename);

if (file.is\_open()) {

file << wins;

file.close();

} else {

cerr << "Error: Unable to open file " << filename << endl;

}

}

void winner(int player, Json::Value& wins) {

if (player % 2 == 0) {

cout << "Player 2 wins!" << endl;

wins["wins"]["X"] = wins["wins"]["X"].asInt() + 1;

} else {

cout << "Player 1 wins!" << endl;

wins["wins"]["O"] = wins["wins"]["O"].asInt() + 1;

}

write\_wins("wins.json", wins);

}

int main() {

char mass[3][3] = { {' ', ' ', ' '}, {' ', ' ', ' '}, {' ', ' ', ' '} };

int player = 1;

Json::Value wins = read\_wins("wins.json");

while(true){

int a, b;

cout << "Enter the field (row and column): ";

cin >> a >> b;

// Check array bounds

if (a < 0 || a >= 3 || b < 0 || b >= 3) {

cout << "Invalid input. Please enter values between 0 and 2." << endl;

continue;

}

// Checking if the field is empty

if (mass[a][b] != ' ') {

cout << "Field already occupied. Please choose another one." << endl;

continue;

}

// Player tag settings

if (player % 2 == 0) {

mass[a][b] = 'X';

} else {

mass[a][b] = 'O';

}

// Display field

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

cout << '|';

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

cout << mass[i][j] << '|';

}

cout << endl;

}

// Log the move

write\_logs("logs.csv", "User chose the (" + to\_string(a) + "," + to\_string(b) + ") cell");

// Checking if there is no win

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

if ((mass[i][0] == mass[i][1] && mass[i][1] == mass[i][2] && mass[i][0] != ' ') ||

(mass[0][i] == mass[1][i] && mass[1][i] == mass[2][i] && mass[0][i] != ' ')) {

winner(player, wins);

return 0;

}

}

if ((mass[0][0] == mass[1][1] && mass[1][1] == mass[2][2] && mass[0][0] != ' ') ||

(mass[0][2] == mass[1][1] && mass[1][1] == mass[2][0] && mass[0][2] != ' ')) {

winner(player, wins);

return 0;

}

// Checking for a tie

bool draw = true;

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

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

if (mass[i][j] == ' ') {

draw = false;

break;

}

}

}

if (draw) {

cout << "Draw!" << endl;

write\_wins("wins.json", wins);

return 0;

}

// Pass the code to the next player

player++;

}

return 0;

}

### 4.7.2 Скріншоти виконання програми

Початок програми:

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Середина програми:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Кінець програми:

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Якщо користувач вибере вже заповнену клітинку:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

# 5 РОЗДІЛ ”ВЕБ-ДИЗАЙН ТА HTML”

HTML (від англ. HyperText Markup Language – «мова гіпертекстової розмітки») – стандартна мова розмітки документів в мережі Інтернет. Біль- шість веб-сторінок містять опис розмітки на мові HTML (або XHTML). Мова HTML інтерпретується браузерами і відображається у вигляді документа в зручній для людини формі.

Мова HTML є додатком SGML (стандартної узагальненої мови роз- мітки) і відповідає міжнародному стандарту ISO 8879.

У всесвітній павутині HTML-сторінки, як правило, передаються брау- зерам від сервера по протоколах HTTP або HTTPS, у вигляді простого тексту або з використанням шифрування.

Мова HTML була розробленою британським вченим Тімом Бернерс-Лі приблизно в 1986-1991 роках в стінах Європейського Центру ядерних дослі- джень в Женеві (Швейцарія). Елементами є структурні і семантичні елементи – дескриптори. Дескриптори також часто називають «тегами». Також, в HTML внесена підтримка гіпертексту.

Таким чином, веб-сторінки створюються за допомогою мови розмітки гіпертексту HTML. Опис веб-сторінок міститься в HTML-програмі, який збе- рігається у звичайному текстовому файлі з розширенням htm чи html. Про- грами мовою HTML містять інструкції (коди), що називаються тегами. Всі теги мови HTML виділяються символами-обмежувачами < і >, між якими за- писується ідентифікатор (ім’я) тега і, можливо, його параметри. Більшість те- гів HTML використовується попарно, тобто для окремого тега (відкриваючо- го) в документі є відповідний закриваючий тег, який записується так само, як і відкриваючий, але з символом / перед іменем тега (закриваючі теги не вико- ристовують параметри).

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

Документ HTML оброблюється браузером та відтворюється на екрані у звичному для людини вигляді.

HTML надає засоби для:  
• створення структурованого документу шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше; •  створення зв’язків між веб-сторінками через гіперпосилання; •  створення інтерактивних форм; •  включення зображень, звуку, відео та ін.

Весь контент описується з використанням спеціальних правил – правил розмітки. Розмітка в HTML складається з чотирьох основних компонентів:

•  елементів;

•  базових типів даних (визначають тип вмісту атрибутів: символьні дані,

числа,..);

•  символьних мнемонік;

•  декларації типу документа (оголошує версію HTML).

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

В загальному, структура HTML-документа є такою:

*Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание*

Парою тегів <HTML>...</HTML> визначається початок і кінець документа. Розділ документа **HEAD** визначає його заголовок, розміщений між парою тегів <TITLE> і </TITLE>, а також мета-теги. Тег <META> містить різнорідну інформацію для броузерів, яка не відображається у вікні. Тег STYLE> пов’язаний з використанням таблиць стилів. Тег <SCRIPT> використовується для запису сценаріїв.

Елементи являють собою базові компоненти розмітки HTML. Кожен елемент має назву і задається за допомогою тегів, що записуються у кутових дужках. Вміст (контент) елемента записують між відкриваючим і закриваючим тегами. Також елементи можуть мати атрибути (властивості), які задаються всередині відкриваючого тегу.

Значення атрибута записують у подвійних лапках. Деякі елементи, можуть не мати атрибутів, інші ж мають обов'язкові атрибути, без яких вони не зможуть виконувати свої функції. Існують також елементи, які не мають вмісту (контенту) і задаються за допомогою одиночних тегів.

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

Структурні. Елементи структурної розмітки застосовуються задля 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. Для подальшого стилізування тексту рекомендується використовувати Каскадні таблиці стилів (CSS).

Презентаційні. Елементи презентаційної (візуальної) 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) та фрейми.

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.

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

# Абзаци

Почнемо з найпростішого тега <p>, за допомогою якого створюються абзаци. За замовчуванням абзаци починаються з нового рядка і мають вертикальні відступи, якими можна управляти за допомогою стилів.

Заголовки і підзаголовки

Для створення структури великих текстів зазвичай використовуються заголовки. У текстових редакторах є можливість виділити частину тексту  як заголовок. В мові HTML для виділення заголовків передбачено ціле сімейство тегів: від <h1> до <h6>. Тег <h1> (headline 1) позначає найважливіший заголовок (заголовок верхнього рівня), а тег <h6> позначає підзаголовок самого нижнього рівня.

На практиці нечасто зустрічаються тексти, в яких зустрічаються підзаголовки нижче третього рівня. Тому самими часто використовуваними тегами заголовків є: <h1>, <h2> і <h3>. Варто відзначити, що пошукові системи надають особливе значення заголовкам.

Маркований (ненумерований) список

Списки часто використовуються в різних документах. Іноді, щоб зробити список, користувач просто нумерує рядки тексту. Такий підхід не є хорошим, оскільки в документі відсутня логічна сутність «список». У HTML існує сімейство тегів для створення списків: невпорядкованих, упорядкованих і списків визначень.

Невпорядковані (або марковані) списки створюються за допомогою тега <ul> (unordered list), який може містити всередині себе теги <li> (list item),

що позначають «елемент списку».

Впорядкований (нумерований) список

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. Нумеровані або впорядковані списки призначені для елементів, які слідують в певному порядку, наприклад покрокові інструкції. Вони функціонують так само, як і описані раніше марковані списки, за винятком того, що визначені за допомогою тега <ol>, та замість маркерів браузер автоматично вказує номера перед елементами списку, тому вам не потрібно проставляти їх у вихідному документі. Це дозволяє легко міняти місцями елементи списку без повторної нумерації.

Для зміни маркерів і номерів списків варто використовувати властивість list-style-type таблиці стилів CSS. Наприклад для маркованих списків Ви можете змінити форму маркера з кола на квадрат або незаштриховане коло, на власне зображення, або видалити маркер повністю. Для нумерованих списків - замінити традиційні арабські цифри римськими (I., II., IІІ. або i., ii., iii.), літерами (А., В., С., або а., b., с.) та іншими типами нумерації.

Важливість. Теги <strong> и <b>

Тег <strong> визначає важливість зазначеного тексту. Тег <b> (bold) призначений для виділення тексту без додання йому особливої ​​важливості.

Найкраще відмінності цих тегів будуть помітні людям, які використовують спеціальні налаштування ОС, зокрема, сліпим і слабозорим. Коли вони включать функцію читання тексту, то «читалка» буде інтонацією виділяти слова з тегом <strong>. Те ж саме стосується і тегів <em> (emphasis) та <i>. Тег <em> «читалка» виділятиме інтонацією. Візуально обидва тега однакові – вони виділяють текст жирним шрифтом. Відзначимо, що новий сенс тегу <b> надали в HTML5. Раніше це був тег, який просто робить текст напівжирним. Тобто він був призначений лише для візуального форматування.

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

Іноді виникає необхідність вставити в текст перенесення рядка, не створюючи при цьому абзац. Наприклад, при розмітці віршів або текстів пісень. Для цього в HTML передбачений одиночний тег <br>(break). Іноді цей тег використовується для розбиття тексту на «як би абзаци», що є поганим підходом. Використовуйте для розмітки абзаців тег <p>.

Одиночний тег <hr> (horisontal rule) використовується для того, щоб створити горизонтальну лінію-розділювач.

# Цитати

У HTML існує декілька тегів для позначення цитат:

• <blockquote> призначений для виділення довгих цитат, які можуть складатися з кількох абзаців. Тег виділяє цитату як окремий блок тексту з відступами.

• <q> (quote) призначений для виділення коротких цитат в тексті пропозиції. Текст всередині цього тега автоматично обрамляється лапками.

• <cite> використовується для того, щоб виділити джерело цитати – назву твору, але не автора цитати.

Посилання

Посилання, зазвичай, виглядає як підкреслена частина тексту, клацаючи на якій ви переходите на іншу сторінку, відкриваєте зображення або починаєте завантажувати файл. Якщо уявити, що Інтернет це величезна мережа з безліччю вузлів, то посилання будуть нитками, що з'єднують всі вузли цієї мережі. Посилання створюються за допомогою дуже простого і короткого тегу <a> Наприклад:

<a href="Саме посилання"> Назва</a>

Адреси бувають різні: абсолютні, відносні, ті що вказують на сторінку, на файл, зображення, ті що містять якір і так далі. А значить і посилання може вказувати на будь-який об'єкт в Інтернеті – на будь-яку веб-сторінку, на будь-який файл. Якщо клацнути по посиланню, що веде на файл, то браузер запропонує його завантажити. Однак, якщо браузер вміє обробляти файли цього типу, то вміст файлу відкриється прямо в браузері. Найчастіше так відбувається із зображеннями. Останнім часом

браузери навчилися відкривати .pdf файли і багато інших.

Робота з картинками.

Щоб додати на сторінку зображення, потрібно використовувати одиночний тег <img> (image) з атрибутом src (source), в якому вказана адреса картинки. наприклад:

<img src = "logo.png">

До речі, найпоширенішими форматами зображень у мережі є JPEG і PNG.

Щоб управляти шириною або висотою зображення, потрібно

використовувати атрибути width і height. Як ось, наприклад:

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

В прикладі зображенню задана ширина 100px (пікселів).

Якщо у користувача відключені зображення чи їх неможливо завантажити, то в браузері відображається альтернативний текст. Наприклад, якщо меню зроблено за допомогою зображень, то альтернативний текст допоможе зрозуміти, куди веде кожен пункт. Загалом, задавати альтернативний текст хороший стиль програмування, хоча і не є обов`язковим. Альтернативний текст зображення задається за допомогою атрибуту alt (alternative)

Посилання можна робити не лише за допомогою тексту, але і за допомогою зображень. Для цього потрібно обернути тег <img> в тег <a>.

Часто посилання-зображення використовуються в галереях, коли на зменшену версію зображення ставиться посилання на повнорозмірну картинку.

## **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 описує елемент, що вводиться.

Значення визначають (природу) параметри властивостей.

Існує два основні способи підключення таблиці стилів:

1. Додавання CSS в HTML-тег.

У цьому способі CSS додається в HTML- документ за допомогою HTML-атрибуту style у середині будь-якого HTML-тегу, що знаходиться у контейнері <body>.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

1. Спосіб другий: Установка стилю для тегів в HTML-документі.

CSS додається в HTML-документ за допомогою HTML-тегу <style> в середині контейнеру <head>. В ньому описуються всі стилі, що будуть використані.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Однією з основних дій, яку можна використовувати за допомогою CSS, є робота з кольором та фоном. Властивість color описує колір елемента. Приклад:

H1 { color: #ff0000; } Відображення у браузері:

Изображение выглядит как текст, снимок экрана, веб-страница, программное обеспечение

Автоматически созданное описание

Властивість background-color описує колір фону елемента. Приклад:  
body { background-color: #FFCC66; } h1{color:#000000;background-color:

#FC9804;text-align:center;}

Відображення у браузері:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

CSS-властивість background-image використовується для додавання фонового зображення. Приклад: body { background-image: url(1.gif); }

Блокування фонового зображення [background- attachment]. Властивість background-attachment визначає, чи фіксується фоновий малюнок, або прокручується разом з вмістом сторінки.

Розташування фонового малюнка [background-position]. Є багато способів встановити значення background-position. Тим не менше, всі вони представляють собою набір координат. Координати можна вказувати у відсотках ширини екрана, у фіксованих одиницях (пікселі, сантиметри, і т.п.), або можна використовувати слова top, bottom, center, left і right.

## **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. Як приклад, найпростіша сторінка:

<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:

Изображение выглядит как диаграмма, текст, линия, Шрифт

Автоматически созданное описание

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

Наслідування

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

Наслідування властивості

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

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

Також до спадкоємною властивостей відносяться list-style, cursor, visibility, border-collapse і деякі інші. Але вони використовуються значно рідше.

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

Каскадність

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

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

CSS-правила існують як мінімум в трьох різних місцях:

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

2.                в атрибуті style;

3.                в стандартних стилях відображення, вбудованих в браузер.

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

1.                важливість;

2.                специфічність;

3.                порядок вихідного коду.

**5.5 Практична частина**

### 5.5.1 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.

### 5.5.2 HTML документ

**НTML**

<!DOCTYPE HTML> #Semenova Valeriia

<html lang="en">.

<head>

<meta charset="UTF-8">

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

<title>Valeriia Semenova</title> #задаємо назву веб-сайту

<link rel="stylesheet" href="практика\_сайт.css"> #посилання на файл css

</head>

<body> #основна частина «тіло» коду

<div class="container"> #застосовуємо стиль

<header class="imagine"> #відділяємо верхню частину сторінки + стиль

<div>

<img src="photo2.jpg" alt="" id="centered"> #додаємо фото

</div>

<div class="center"> #додаємо текст і застосовуємо стиль

<h1>Семенова Валерія</h1>

<h4>студентка 2-го курсу</h3>

</div>

</header>

<h2 class="mood">Загальна інформація</h2> #додаємо текст + стиль

<hr> #добавляємо горизонтальну лінію

<p>Стресостійка фахівчиня, з вмінням гарно комунікувати з людьми, критичним мисленням, творчими здібностями та

клієнтоорієнтованістю. Загалом маю 2 роки досвіду роботи адміністратором в закладі та декілька розроблених

самостійно проєктів, таких як розробка Telegram-боту, наукова робота та проєкт у сфері Data Analytics.</p> #додаємо параграф з текстом

<p>Маю досвід роботи у стресовий ситуаціях, завжди вкладаюсь у час поставлений для виконання завдання, відкрита до нових

складних завдань, які вимагають швидкого навчання та пошуку оптимального рішення задачі. </p>

<h2 class="mood">Освіта</h2>

<hr>

<ul> #ненумерований список

<li> <b>2012-2022 Базова середня освіта з відзнакою</b>

<br> Боярський академічний ліцей "Гармонія"

</li>

<li><b>2022-дотепер Молодший бакалавр</b>

<br> Опико-механічний фаховий коледж імені Тараса Шевченка <br> Спеціальність:

121 Інженерія програмного забезпечення</li>

</ul>

<h2 class="mood">Досвід роботи</h2>

<hr>

<ul>

<li><b>Адміністратор (2021-2023)</b>

<br> Місце роботи:вейк-парк "Піратська бухта" м.Київ

</li>

<li><b>Data Analytics (березень-квітень 2023)</b>

<br> Розробка аналітичного дашборду, а саме проведення аналітики

кінофільмів за жанрами, країнами, в яких ці фільми були випущені, та мови, якими вони були озвучені.

</li>

<li><b>Python-розробник (січень-квітень 2024)</b>

<br> Розробка Telegram-боту. Написаний мовою Python, цей бот має всі

навички Штучного Інтелекту та гарний інтерфейс, що дозволяє зручно ним користуватись.

</li>

</ul> #ненумерований список

<footer> #відділяємо нижню частину веб-сторінки

<ul>

<li><div>

<a href="mailto:lerasem47@gmail.com" target="\_blank">Email <img src="post.png" class="im"></a>

</div></li>

<div class=""><li>

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

</li></div>

</ul>

</footer>

</div>

</body>

**CSS**

ul {

list-style-type: none;

}

a {

margin: 10px;

padding: 10px;

text-decoration: none;

text-align: center;

width: 90%;

height: 25px;

display: block;

border: 4px solid rgb(255, 215, 165);

text-decoration: none;

border-radius: 20px;

}

a:hover,

a:visited {

background-color: rgba(217, 184, 143, 0.881);

color: black;

}

.container {

background-color: rgba(218, 203, 186, 0.387);

width: 95vw;

margin: 0 auto;

}

.mood {

color: rgb(93, 82, 68);

}

.imagine {

text-align: center;

}

#centered {

border-radius: 100%;

height: 150px;

width: 150px;

}

.center {

color: rgb(93, 82, 68);

text-align: center;

}

.im {

height: 20px;

}

**Изображение выглядит как текст, снимок экрана, Веб-сайт, веб-страница

Автоматически созданное описание**

**Изображение выглядит как текст, веб-страница, Веб-сайт, программное обеспечение

Автоматически созданное описание**

# ВИСНОВКИ

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ

1. ДСТУ 1.5-93 Загальні вимоги до побудови, викладу, оформлення та змісту.
2. Мова програмування С++: конспект лекцій / О. В. Галкін, М. М. Верес. — К.: ДП “Вид. дім “Персонал”, 2017. — 260 с. — Біблі- огр.: с. 249.: <https://maup.com.ua/assets/files/lib/book/c_plisplus.pdf>
3. «ОСНОВИ ПРОГРАМУВАННЯ на C++» Зеленський О.С., Лисенко В.С. – Кривий Ріг: Державний університет економіки і технологій, 2023.-269 с.: <https://dspace.duet.edu.ua/jspui/bitstream/123456789/831/1/НП%20Osnovy_C%2B%2B.pdf>
4. Формальні та фактичні параметри, хеш-таблиця (словник), масив, список, стек, черга мовою C++: <https://www.google.com/url?Sa=i&url=https%3A%2F%2Fwww.kievoit.ippo.kubg.edu.ua%2Fkievoit%2F2016%2F65_C%2B%2B%2Findex.html&psig=aovvaw0e_izvl6jk9vdsho-jqp2y&ust=1717139073705000&source=images&cd=vfe&opi=89978449&ved=0cacqrpomahckewiah9ui7lsgaxuaaaaahqaaaaaqba>
5. Міністерство освіти і науки україни Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» «Управління версіями програмних засобів проекту»: <https://ipze.kpi.ua/wp-content/uploads/2024/02/04-Upravlinnia_versiiamy_prohramnykh_zasobiv_proektu.pdf>
6. Матеріал з Вікіпедії — вільної енциклопедії. «Markdown»: <https://uk.wikipedia.org/wiki/Markdown>
7. MDN Web Docs - an open-source, collaborative project documenting Web platform technologies : <https://developer.mozilla.org/en-US/>
8. Міністерство освіти і науки україни державний вищий навчальний заклад «Ужгородський національний університет» математичний факультет, кафедра кібернетики і прикладної математики «Основи HTML та CSS»: <https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/40974/1/Основи%20HTML.pdf>
9. Міністерство освіти і науки України державний вищий навчальний заклад “Ужгородський національний університет” математичний факультет, кафедра системного аналізу і теорії оптимізації «Вступ до web-програмування. Основи html»: <https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/32871/1/Вступ%20до%20WEB-програмування.%20Основи%20HTML.pdf>
10. Е-helper.com.ua «Лекція 2. Розмітка тексту. Зображення. Посилання.»: <http://e-helper.com.ua/node/949>
11. «CASCADING STYLE SHEETS (CSS) - КАСКАДНІ ТАБЛИЦІ СТИЛІВ»: <https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=http://vv-steshyn.edu.kpi.ua/katalog/files/Lekciya-3.-CSS-kaskadni-tablici-stiliv-68.pdf&ved=2ahUKEwi5utTekqKGAxXD4AIHHaW3DtcQFnoECBIQAQ&usg=AOvVaw3_uxBU96i5-43dUKH8RfG4>
12. Е-helper.com.ua «Лекція 5. Наслідування та каскадування. Оформлення тексту за допомогою CSS»: <http://e-helper.com.ua/node/952>
13. А. Каплун, Ю. В. Баришев, А. В. Остапенко «ТЕХНОЛОГІЯ ПРОГРАМУВАННЯ ЛАБОРАТОРНИЙ ПРАКТИКУМ»: <https://web.posibnyky.vntu.edu.ua/fitki/14baryshev_tehnologiya_programuvannya/08.html>

# ДОДАТКИ