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 |
| Ромашко Мирослави Олександрівни  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) | |  | |
| *Ромашко Мирослави Олександрівни* | | | | |
| (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-21

ОМФК.0121 НП 20

Зміст

[ВСТУП 6](#_Toc167969935)

[1 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ 7](#_Toc167969936)

[2 РОЗДІЛ "ІНФОРМАТИКА" 9](#_Toc167969937)

[**2.1** **Розроблення засобами MS Word документа складної структури** 9](#_Toc167969938)

[**2.2 Розв‘язок задач за допомогою засобів табличного редактора MS Excel** 9](#_Toc167969939)

[**2.3 Виконання** **практичних завдань за допомогою засобів табличного редактора MS Access** 9](#_Toc167969940)

[3 РОЗДІЛ "ПРОГРАМУВАННЯ С++" 10](#_Toc167969941)

[**3.1 Теоретичні відомості** 10](#_Toc167969942)

[3.1.1 Загальні поняття. Елементи мови С++ - константи, змінні, операції, перетворення типів 10](#_Toc167969943)

[3.1.2 Арифметичні операції 10](#_Toc167969944)

[3.1.3 Поняття вхідного та вихідного потоку, найпростіші математичні функції 10](#_Toc167969945)

[3.1.4 Програмування лінійних та розгалужених обчислювальних процесів 10](#_Toc167969946)

[3.1.5 Програмування циклічних обчислювальних процесів 10](#_Toc167969947)

[**3.2 Практичне завдання** 11](#_Toc167969948)

[3.2.1 Завдання 11](#_Toc167969949)

[3.2.2 Постановка задачі 11](#_Toc167969950)

[3.2.3 Код програми 11](#_Toc167969951)

[3.2.4 Скріншот виконання програми 11](#_Toc167969952)

[**3.3 Виконання тестів** 11](#_Toc167969953)

[4 РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++" 12](#_Toc167969954)

[**4.1 Тема 1. Введення та виведення у базових консольних програмах** 12](#_Toc167969955)

[**4.2 Тема 2. Структури даних для управління станом програми** 16](#_Toc167969956)

[**4.3 Тема 3. Робота з файлами: запис та читання** 20](#_Toc167969957)

[**4.4 Тема 4. Формати файлів та їх обробка** 22](#_Toc167969958)

[**4..5 Тема 5. Системи контролю версій (VCS)** 22](#_Toc167969959)

[**4.6 Тема 6. Мова Markdown для опису проекті** 26](#_Toc167969960)

[**4.7 Розроблення ігрової програми** 29](#_Toc167969961)

[4.7.1 Код програми 29](#_Toc167969962)

[4.7.2 Скріншоти виконання програми 30](#_Toc167969963)

[5 РОЗДІЛ ”ВЕБ-ДИЗАЙН ТА HTML” 32](#_Toc167969964)

[**5.1 Мова розмітки HTML та структура Веб-сторінки** 32](#_Toc167969965)

[**5.2 Розмітка тексту за допомогою HTML. Зображення та посилання** 34](#_Toc167969966)

[Абзаци 34](#_Toc167969967)

[Цитати 35](#_Toc167969968)

[**5.3 Каскадні таблиці стилів CSS та їх структура** 36](#_Toc167969969)

[**5.4 Спадковості і каскадування за допомогою CSS** 38](#_Toc167969970)

[5.5.1 Постановки задачі 40](#_Toc167969971)

[5.5.2 HTML документ 40](#_Toc167969972)

[ВИСНОВКИ 46](#_Toc167969973)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ 47](#_Toc167969974)

[ДОДАТКИ 49](#_Toc167969975)

ОМФК.0121 НП 20

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

# ВСТУП

Автор: Ромашко Мирослава Олександрівна

Дата: 31.05.2024

Мета: Оволодіти практичними навичками програмування гри на мові C++

Перелік вивчених тем:

Тема 1. Введення та виведення у базових консольних програмах

Тема 2. Структури даних для управління станом програми

Тема 3. Робота з файлами: запис та читання

Тема 4. Формати файлів та їх обробка

Тема 5. Системи контролю версій (VCS)

Тема 6. Мова Markdown для опису проектів

# 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**

Об’єкт cout виводить дані на заданий пристрій.

У цьому прикладі cout використовується для виведення цілого числа, дійсного числа та рядка на консоль

#include <iostream>

int main() {

int number = 10;

double pi = 3.14159;

std::string message = "Привіт!";

// Виведення цілого числа

std::cout << "Число: " << number << std::endl;

// Виведення дійсного числа

std::cout << "Значення Pi: " << pi << std::endl;

// Виведення рядка

std::cout << "Повідомлення: " << message << std::endl;

return 0;

}

Також для cout використовуються різноманітні маніпулятори. Наприклад:

Маніпулятори setw(), setprecision() і setfill()

setw(): Цей маніпулятор встановлює ширину поля виводу для наступного виводу у консоль. Наприклад, setw(10) встановить ширину поля на 10 символів.

setfill(): Цей маніпулятор встановлює символ, який буде заповнювати незайняті позиції виводу. Наприклад, setfill('\*') встановить символ '\*' для заповнення незайнятих позицій..

setprecision(n): встановлює кількість знаків після коми для виведення дійсного числа. Наприклад, setprecision(4) обмежить кількість знаків після коми до 4.

Маніпулятори setiosflags і resetiosflags використовуються для встановлення та скидання певних прапорів стану потоку виводу, таких як науковий формат виводу чисел, вирівнювання та інші. Вони дозволяють контролювати різноманітні аспекти виводу, такі як формат чисел, вирівнювання тексту, формат виводу дати і часу тощо.

У класичному C функція printf використовується для виведення даних. Вона перейшла у C++ без змін. Синтаксис функції виглядає наступним чином:

printf(char \*fotmat, <додаткові аргументи>)

У рядку форматування символи виводяться без змін, а спеціальні символи формату вказують на перетворення даних. Для здійснення цих перетворень використовуються додаткові аргументи.

Специфікація перетворення формату має наступний вигляд: %[прапори][ширина][.точність][L|l]тип.

Де:

* Прапори включають:
  + Мінус (-) - вирівнювання по лівому краю поля.
  + Плюс (+) - виводиться знак числа "+" або "-".
* Ширина визначає мінімальне число виведених символів, які можуть доповнюватися пробілами або нулями.
* Точність визначає число знаків після десяткової крапки для чисел з рухомою точкою і максимальне число виведених цифр для змінних цілого типу.

тип:  
d або i -десяткове ціле зі знаком (int);  
u – десяткове ціле без знака (unsigned int);  
x – шістнадцятирічне ціле без знака (unsigned int);  
f – виведення числа з плаваючою точкою (float, double);  
e – виведення числа з плаваючою точкою із використанням

експоненціальної форми (float, double);

c – виведення символу (char); s – виведення рядка;  
p – виведення по вказівці.

L | l – застосовується в комбінації із символом типу:  
Lf і Le - довгий із плаваючою точкою (long double); ld або li - довгий цілий (long int);  
lu – беззнаковий довгий цілий (unsigned long int).

Синтаксис cin використовується для зчитування вводу з клавіатури

У даному прикладі введене користувачем ціле число зберігається у змінній number за допомогою cin, а потім виводиться на екран за допомогою cout.

#include <iostream>

int main() {

int number;

std::cout << "Введіть ціле число: ";

std::cin >> number;

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

return 0;

}

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

Розуміння структур даних є ключовою частиною процесу розробки програмного забезпечення. Давайте розглянемо кілька найпоширеніших структур даних та їхній кодовий виконання в мові C++.

**Масиви**: Вони представляють собою найпростіший вид структури даних, що дозволяє зберігати набір елементів одного типу в послідовному порядку в пам'яті. Вони можуть використовуватися для зберігання даних, які визначають стан програми, і доступу до них за допомогою індексів.

У мові C++ неможливо змінити властивості елементів масиву під час виконання програми, хоча в інших мовах це може бути можливо. Групування розмірності та діапазонів у масиві називається його формою.

У коді програми доступ до значення елемента масиву здійснюється, вказавши його назву у квадратних дужках та індекси у вигляді арифметичних виразів, що приймають значення з діапазону порядкового типу. Наприклад, `a[3]` вказує на третій елемент масиву `a`, а `c[j]` - на j-ий елемент масиву `c`.

Елемент масиву може бути простим (неструктурованим) або складеним типом, таким як масив або рядок. Глибина вкладеності не обмежена, але загальна довжина структури масиву обмежена. У пам'яті всі елементи масиву зберігаються послідовно, індекси масиву записуються зліва направо.

Під час вирішення задач часто використовуються одновимірні та двовимірні масиви. Одновимірний масив також відомий як лінійний масив або вектор, де кожному елементу відповідає один індекс, а в математиці він асоціюється з поняттям послідовності та індексами масиву.

**Стеки (stacks)**: Це абстрактна структура даних, що працює за принципом "Last In, First Out" (LIFO). Елементи додаються та видаляються з одного кінця стеку, який називається вершиною. Стеки часто використовуються для управління послідовністю операцій або викликів функцій.

**Черги (queues)**: Це інша абстрактна структура даних, яка працює за принципом "First In, First Out" (FIFO). Елементи додаються в кінець черги, а видаляються з початку. Черги часто використовуються для керування послідовністю завдань або подій, які потрібно виконати у порядку їх надходження.

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

Для запису даних у файл у мові C++ використовується об'єкт типу **ofstream** (output file stream). Перед тим, як почати запис, файл потрібно відкрити за допомогою методу open(). Після запису даних файл слід закрити.

Для читання даних з файлу використовується об'єкт типу **ifstream** (input file stream). Також, як і при записі, перед читанням файл потрібно відкрити за допомогою методу open(). Після прочитання файл слід закрити.

Функція **getline** використовується для зчитування рядка з потоку введення до першого символу нового рядка ('\n') або до кінця файлу. Вона може бути використана для зчитування рядків з клавіатури або з файлу.

Функція **fail()** використовується для перевірки стану потоку вводу/виводу . Вона повертає true, якщо сталася помилка під час останньої операції з потоком, і false, якщо операція пройшла успішно.

Зазвичай функція fail() перевіряється після проведення операцій введення або виведення, таких як cin >>, cout <<, getline, тощо. Вона дозволяє виявити помилки вводу/виводу, такі як некоректні дані, неправильний формат, недостатній обсяг пам'яті, помилки читання або запису файлу тощо.

**Приклад запису і читання файлів C++**

**Читання і запис текствого формату:**

#include <iostream>

#include <fstream>

#include <string>

int main() {

std::ofstream outfile("example.txt"); // відкриваємо файл для запису

if (outfile.is\_open()) {

outfile << "Це приклад текстового запису у файл.\n";

outfile.close();

}

std::ifstream infile("example.txt"); // відкриваємо файл для читання

if (infile.is\_open()) {

std::string line;

while (std::getline(infile, line)) {

std::cout << line << '\n';

}

infile.close();

}

return 0;

}

**Робота з бінарними файлами:**

`#include <iostream>

#include <fstream>

#include <vector>

int main() {

std::vector<int> data = {1, 2, 3, 4, 5};

// Записуємо дані у бінарний файл

std::ofstream outfile("example.bin", std::ios::binary);

if (outfile.is\_open()) {

outfile.write(reinterpret\_cast<const char\*>(data.data()), data.size() \* sizeof(int));

outfile.close();

} else {

std::cerr << "Unable to open file for writing!" << std::endl;

return 1;

}

// Зчитуємо дані з бінарного файлу

std::ifstream infile("example.bin", std::ios::binary);

if (infile.is\_open()) {

std::vector<int> loaded\_data(data.size());

infile.read(reinterpret\_cast<char\*>(loaded\_data.data()), loaded\_data.size() \* sizeof(int));

infile.close();

// Виводимо зчитані дані

for (const auto& value : loaded\_data) {

std::cout << value << ' ';

}

std::cout << std::endl;

} else {

std::cerr << "Unable to open file for reading!" << std::endl;

return 1;

}

return 0;

}

Загалом відмінність між текстовими та бінарними форматами полягає у тому, як дані представлені у файлі: у вигляді тексту або у вигляді бінарних значень.  
**Текстові файли** зазвичай містять дані, що може прочитати людина, оскільки вони представлені у вигляді символів. Ці файли зберігають дані у вигляді тексту, який можна інтерпретувати, часто з використанням спеціальних форматів, таких як JSON, XML або CSV. Наприклад, у текстовому файлі числа часто представлені як послідовності символів, що відповідають цифрам.  
**Бінарні файли** містять дані у вигляді бінарних значень, які представлені у вигляді бітів. Вони не призначені для прямого читання людьми, але зазвичай мають специфікацію, що визначає, як дані організовані в файлі. Бінарні файли можуть зберігати будь-який тип даних, включаючи числа, тексти, зображення, звуки та інші структуровані або неструктуровані дані.

**Приклади як читати і записувати файли форматів JSON, CSV, XML. Бібліотеки.**

Почнемо з JSON  
**Для роботи з JSON у C++ можна використовувати бібліотеку nlohmann/json. Ця бібліотека зручна та легка у використанні**.

#include <iostream>

#include <fstream>

#include <nlohmann/json.hpp>

using json = nlohmann::json;

int main() {

// Запис JSON у файл

json data = {

{"name", "Myra"},

{"age", 17},

{"city", "Chernihiv"}

};

std::ofstream outfile("example.json");

outfile << std::setw(4) << data << std::endl;

outfile.close();

// Читання JSON з файлу

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

json loaded\_data;

infile >> loaded\_data;

std::cout << "Name: " << loaded\_data["name"] << ", Age: " << loaded\_data["age"] << ", City: " << loaded\_data["city"] << std::endl;

infile.close();

return 0;

}

**Для роботи з форматом CSV в C++ ми можемо використати бібліотеку CSVParser.**

#include <iostream>

#include "csv.h" // Підключаємо бібліотеку CSVParser

int main() {

// Створюємо об'єкт для запису даних у CSV-файл

csv::ofstream outfile("example.csv");

// Записуємо заголовок у файл

outfile << "Name,Age,City" << csv::endl;

// Записуємо дані у файл

outfile << "Myra,17,Chernihiv" << csv::endl;

outfile << "Ulyana,17,Prague" << csv::endl;

// Закриваємо файл

outfile.close();

// Читаємо дані з CSV-файлу

csv::ifstream infile("example.csv");

// Зчитуємо дані з файлу та виводимо їх на екран

csv::CSVReader reader(infile);

for (csv::CSVRow& row : reader) {

for (csv::CSVField& field : row) {

std::cout << field.get() << '|';

}

std::cout << std::endl;

}

// Закриваємо файл

infile.close();

return 0;

}

**Для роботи з XML в C++ можна використовувати бібліотеку pugixml.**

#include <iostream>

#include <fstream>

#include <pugixml.hpp>

int main() {

// Запис XML у файл

pugi::xml\_document doc;

pugi::xml\_node node = doc.append\_child("person");

node.append\_child("name").append\_child(pugi::node\_pcdata).set\_value("Myra");

node.append\_child("age").append\_child(pugi::node\_pcdata).set\_value("17");

node.append\_child("city").append\_child(pugi::node\_pcdata).set\_value("Chernihiv");

doc.save\_file("example.xml");

// Читання XML з файлу

pugi::xml\_document loaded\_doc;

if (loaded\_doc.load\_file("example.xml")) {

pugi::xml\_node loaded\_node = loaded\_doc.child("person");

std::cout << "Name: " << loaded\_node.child\_value("name") << ", Age: " << loaded\_node.child\_value("age") << ", City: " << loaded\_node.child\_value("city") << std::endl;

}

return 0;

}

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

Формати файлів

Текстові формати

1.**TXT** - це текстовий файл, який містить збережений текст у вигляді послідовності символів. У форматі TXT відсутні будь-які специфічні форматування або розмітки, які визначають структуру документа, такі як шрифти, кольори або таблиці.

2.**CSV (Comma-Separated Values)**: Використовує коми або інші роздільники для відокремлення полів даних. Часто використовується для збереження табличних даних.

**Переваги**: Простота у використанні інтерфейсу, підтримка багатьох програм, швидкий та ефективний для обробки великих обсягів даних.  
**Використання**: Зазвичай використовується для зберігання табличних даних, таких як дані з баз даних, реєстри, тощо. Також може бути корисним для обміну даними між різними програмами.

**Приклад:**

Name,Age,City

Myra,17,Chernihiv

Ulyana,17,Prague

3. **JSON (JavaScript Object Notation)**: Легко зрозумілий формат обміну даними, що використовується для зберігання структурованих даних у вигляді пар "ключ-значення". Широко використовується в веб-розробці та API.

**Переваги**: Гнучкість у представленні структурованих даних, легкість для читання та розуміння людиною, підтримка вкладених об'єктів та масивів.  
**Використання**: Широко використовується для зберігання конфігурацій, обміну даними між сервером та клієнтом у веб-додатках.

**Приклад:**

{

"players": [

{

"name": "Myra",

"age": 17,

"city": "Chernihiv"

},

{

"name": "Ulyana",

"age": 17,

"city": "Prague"

}

]

}

4. **XML (eXtensible Markup Language)**: Інша форма для зберігання та обміну структурованими даними, яка використовує теги для визначення структури даних.

**Переваги**: Розширюваність та можливість створення власних тегів, підтримка вкладених структур даних, підтримка різних кодувань символів.  
**Використання**: Застосовується в багатьох сферах, таких як розмітка документів у веб-розробці, обмін даними між різними системами, конфігураційні файли.

**Приклад:**

<players>

<player>

<name>Myra</name>

<age>17</age>

<city>Chernihiv</city>

</player>

<player>

<name>Ulyana</name>

<age>17</age>

<city>Prague</city>

</player>

</players>

Бінарні файли:

* **BMP (Bitmap)**: Формат зображення, який зберігає графічні дані у вигляді бінарних даних. Часто використовується для зберігання зображень.
* **MP3 (MPEG Audio Layer III)**: Формат аудіофайлу, який зберігає звук у стисненому бінарному форматі.
* **PDF (Portable Document Format)**: Формат документа, який зберігає тексти, зображення та інші типи даних у бінарному форматі. Часто використовується для створення документів, які зберігають форматування та макет сторінки.

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

Що таке система контролю версій

Система контролю версій (VCS) - це програмне забезпечення, яке дозволяє відстежувати зміни у файловій системі з часом. Вона зберігає копії файлів на різних етапах розвитку, дозволяючи користувачам відновлювати попередні версії файлів, порівнювати зміни між версіями, а також спільно працювати над проектами.

Основні переваги систем контролю версій включають:

1. **Відстеження змін**: Система контролю версій зберігає історію змін файлів, що дозволяє користувачам переглядати, хто і коли змінив конкретний файл, і які саме зміни були внесені.
2. **Відновлення попередніх версій**: Користувачі можуть відновити попередні версії файлів або проектів, які були втрачені або пошкоджені.
3. **Гілки розробки**: Системи контролю версій дозволяють створювати гілки розробки, що дозволяє розробникам відокремлювати різні функціональності або експериментувати з різними версіями коду, не впливаючи на основний проект.
4. **Спільна робота**: Кілька користувачів можуть працювати над одним і тим же проектом, завантажуючи, змінюючи і оновлюючи файли з централізованого репозиторію.
5. **Аудит і безпека**: Системи контролю версій ведуть журнали змін, що дозволяє виконувати аудит, визначати, хто і коли змінював файли, і в разі потреби відновлювати старі версії файлів.

Деякі з найпопулярніших систем контролю версій включають Git, Subversion (SVN), Mercurial, CVS тощо. Git є однією з найпопулярніших і широко використовуваних систем контролю версій, особливо в сфері розробки програмного забезпечення.

**Приклади команд в Git:**

1.git init: Ініціалізує порожній репозиторій Git у поточному каталозі.

2.git clone : Клонує віддалений репозиторій на локальний комп'ютер.

3.git add : Додає зміни файлу до індексу для подальшого коміту.

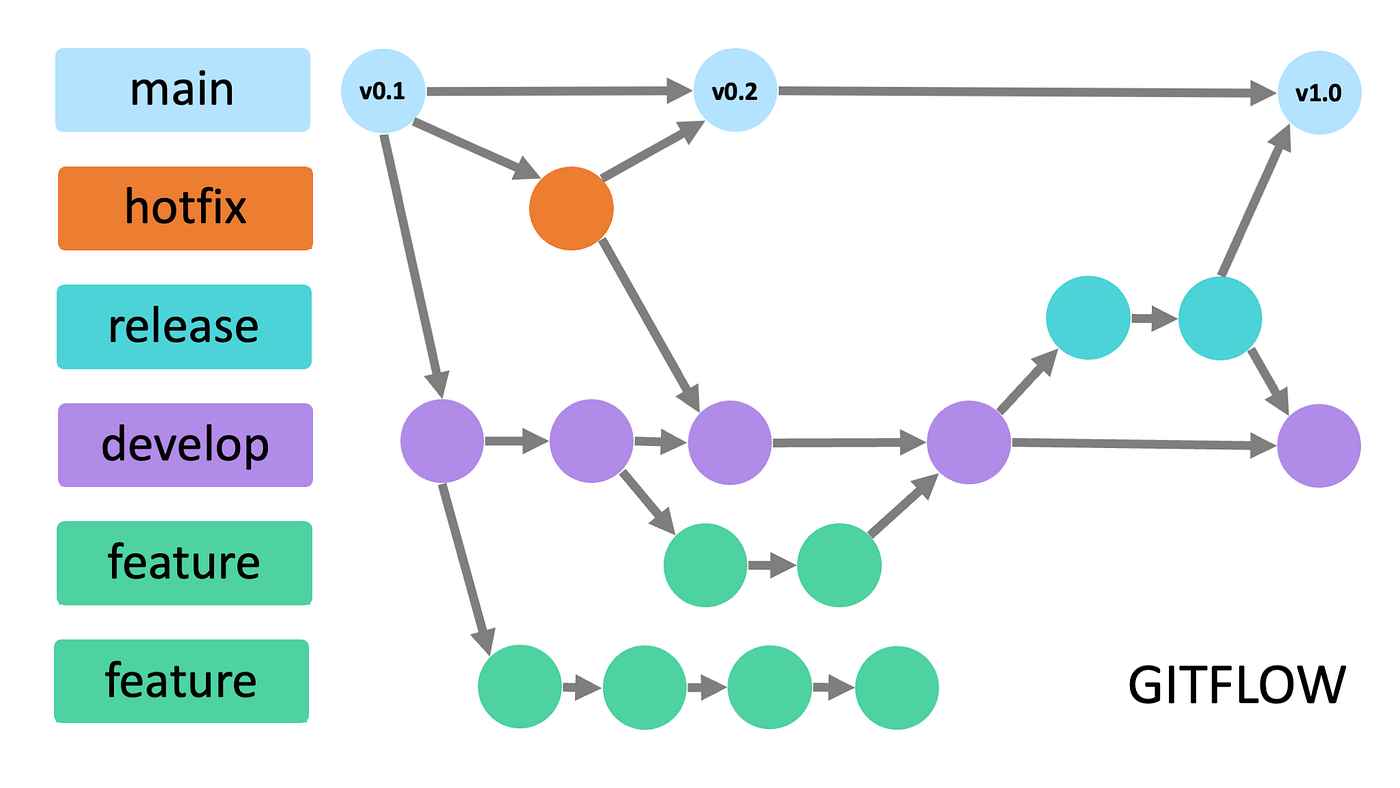
4.git pull: Витягує та об'єднує зміни з віддаленого репозиторію у локальний.

5.git push: Відправляє локальні коміти на віддалений репозиторій.

6.git status: Показує статус змін у вашому репозиторії.

7.git log: Показує історію комітів у репозиторії.

**Git Flow:**



**main**: Гілка main відображає продуктовий код. Кожен коміт у main повинен відповідати випущеній версії програми.

**develop**: Гілка develop є основною робочою гілкою, де розробляється новий код. Всі функціональність, що входить до наступного випуску, повинні бути злиті у цю гілку.

**feature branches**: Кожна нова функціональність або фіча розробляється у власній гілці, яка виходить з develop. Після завершення роботи над функціоналом гілка зливається назад у develop.

**release branches**: Коли розробники готові до випуску нової версії програми, вони створюють гілку release з develop. У цій гілці вони можуть виконати остаточні тести та виправлення перед випуском нової версії.

**hotfix branches**: Гілки hotfix використовуються для виправлення критичних помилок у продукції, які виявляються в main. Ці гілки виходять з main, виправляють помилки та зливаються як у main, так і в develop.

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

**Markdown** - це простий інструмент для створення форматованого тексту за допомогою легко зрозумілих та простих у використанні тегів. Він широко використовується для написання документації, статей у блогах, README файлів у репозиторіях на GitHub та іншому середовищі розробки, а також для створення простих веб-сторінок.

Markdown використовує простий синтаксис, що дозволяє легко форматувати текст. Наприклад, для створення заголовків використовується символ #, для надання тексту жирного виду - \*\*текст\*\*, для посилань - [текст посилання](URL), тощо.

**Приклад:**

## ### 1.Заголовки:

## # Заголовок першого рівня

## ## Заголовок другого рівня

## ### Заголовок третього рівня

## ### 2.Текст:

## \*Курсив\*

## \*\*Напівжирний\*\*

## \*\*\*Курсив та напівжирний\*\*\*

## `Код`

## ### 3.Списки:

## - Елемент списку 1

## - Елемент списку 2

## - Вкладений елемент списку

## 1. Елемент списку з номером 1

## 2. Елемент списку з номером 2

## ### 4.Посилання:

## [Текст посилання](https://www.example.com)

## ### 5.Зображення:

## ![Опис зображення](https://www.example.com/image.jpg)

## ### 6.Таблиці:

## | Заголовок 1 | Заголовок 2 |

## |-------------|-------------|

## | Вміст 1 | Вміст 2 |

## ### 7.Цитати:

## > Це цитата.

## ### 8.Горизонтальні лінії:

## ---

## ### 9.Видалення та підкреслення:

## ~~Видалено~~

## <u>Підкреслено</u>

## **4.7 Розроблення ігрової програми**

### 4.7.1 Код програми

#include <iostream>

using namespace std;

// Function to draw the Tic-Tac-Toe board

void drawBoard(string board[3][3]) {

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

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

cout << board[i][j];

if (j < 2) cout << "|";

}

cout << endl;

if (i < 2) cout << "-----" << endl;

}

}

// Function to check for a win

string checkWinner(string board[3][3]) {

// Check rows, columns, and diagonals

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

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

return board[i][0];

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

return board[0][i];

}

if (board[0][0] != " " && board[0][0] == board[1][1] && board[1][1] == board[2][2])

return board[0][0];

if (board[0][2] != " " && board[0][2] == board[1][1] && board[1][1] == board[2][0])

return board[0][2];

return "";

}

int winsX = 0;

int winsO = 0;

int main() {

string board[3][3] = {

{" ", " ", " "},

{" ", " ", " "},

{" ", " ", " "}

};

string currentPlayer = "X";

int moves = 0;

string winner;

cout << "Welcome to Tic-Tac-Toe game!" << endl;

while (true) {

drawBoard(board);

int row, col;

// Read input position

cout << "Player " << currentPlayer << ", enter row (0-2) and column (0-2): ";

cin >> row >> col;

// Validate input

if (row < 0 || row > 2 || col < 0 || col > 2 || board[row][col] != " ") {

cout << "Invalid move. Try again." << endl;

continue;

}

// Set board cell value with current symbol

board[row][col] = currentPlayer;

moves++;

// Check for a winner

winner = checkWinner(board);

if (winner != "") {

drawBoard(board);

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

if (winner == "X")

winsX++;

else

winsO++;

break;

}

// If all cells are filled, end the game

if (moves == 9) {

drawBoard(board);

cout << "It's a draw!" << endl;

break;

}

// Update current symbol

currentPlayer = (currentPlayer == "X") ? "O" : "X";

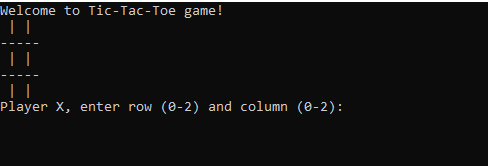
}

return 0;

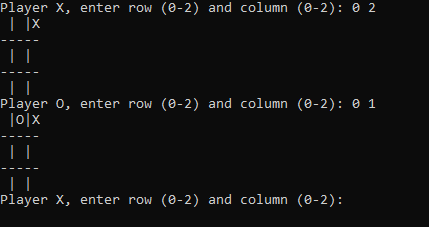
### }

### 4.7.2 Скріншоти виконання програми

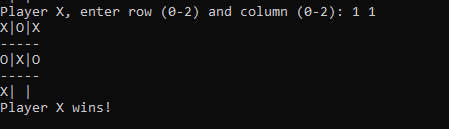
Початок програми:



Середина програми:



Кінець програми:



# 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.

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:

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:

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**

<html>

    <head><script src="https://kit.fontawesome.com/b946b3bc8f.js" crossorigin="anonymous"></script></head>

    <body>

        <div class="img">

            <img src="dumpty.JPG" id="img1" width="150" height="150">

        </div>

        <h2>Romashko Myroslava</h2>

        <h4>Student</h4>

        <h3> Summary </h3>

        <hr>

        <p class="p1"> Overall, I don't have any experience yet, but I'm a very hardworking individual and always open to various opportunities. I have a good command of the English language. I'm interested in various programming languages, particularly C# and C++.

            I also have experience working in Unity. I successfully completed a <i>Unity Developer</i> course.</p>

            <br>

            <h3> Education </h3>

            <hr>

        <p class="p1"> 2022-now. Opto-Mechanical College of Taras Shevchenko National University in Kyiv. </p>

        <br>

        <h3> Experience </h3>

        <hr>

        <p class="p1"> <dt class="p1"><span class="bold1">College project</span> in Unity <span class="bold1">(February 2024 - now):</span></dt>

            <dd class="p1">3d first-person shooter game</dd> </p>

            <br>

            <div id="button">

                <i class="fa-solid fa-envelope" id="im3"></i>

                <a href="mailto:mirosya2701@gmail.com" class="link" target="\_blank">Email</a>

            </div>

                <br>

                <div id="button1">

                    <i class="fa-brands fa-facebook" id="im2"></i>

                <a href="https://www.facebook.com" class="link" target="\_blank">Facebook</a>

                </div>

                <br>

                <div id="button2">

                    <i class="fa-brands fa-instagram" id="im1"></i>

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

                </div>

                <br>

                <div id="button3">

                    <i class="fa-brands fa-github" id="im"></i>

                <a href="https://github.com" class="link" target="\_blank">Github</a>

                </div>

            </div>

    </body>

</html>

**CSS**

        <style>

            h2 {

                text-align: center;

                color: darkslategrey;

                margin: 5px;

                padding: 5px;

            }

            h4 {

                text-align: center;

                padding: 3px;

                margin: 10px;

                color: rgb(50, 51, 51);

                font-weight: lighter;

            }

            h3 {

                line-height: 0.2;

                margin: 3px;

            }

            .p1 {

                color: rgb(50, 51, 51);

                margin: 0,5px;

            }

            .bold1 {

                font-weight: bold;

            }

            body {

                font-family: Arial, Helvetica, sans-serif;

                font-size: 15;

                padding-left: 60px;

                padding-right: 60px;

                margin: 0 auto;

                background-color: aliceblue;

            }

            #button {

                text-align: center;

                text-align: center;

                border: 2px solid darkslategray;

                margin: auto auto;

                width: 90vw;

                width: 100%;

                border-radius: 5px;

                cursor: pointer;

            }

            #button:hover {

                background-color: rgb(50, 51, 51);

            }

            #button:hover a {

                color: rgb(185, 185, 185);

            }

            #button:hover i {

                color: rgb(185, 185, 185);

            }

            #button1 {

                text-align: center;

                border: 2px solid darkslategray;

                margin: auto auto;

                width: 90vw;

                width: 100%;

                border-radius: 5px;

                cursor: pointer;

            }

            #button1:hover {

                background-color: rgb(50, 51, 51);

            }

            #button1:hover a {

                color: rgb(185, 185, 185);

            }

            #button1:hover i {

                color: rgb(185, 185, 185);

            }

            #button2 {

                text-align: center;

                text-align: center;

                border: 2px solid darkslategray;

                margin: auto auto;

                width: 90vw;

                width: 100%;

                border-radius: 5px;

                cursor: pointer;

            }

            #button2:hover {

                background-color: rgb(50, 51, 51);

            }

            #button2:hover a {

                color: rgb(185, 185, 185);

            }

            #button2:hover i {

                color: rgb(185, 185, 185);

            }

            #button3 {

                text-align: center;

                text-align: center;

                border: 2px solid darkslategray;

                margin: auto auto;

                width: 90vw;

                width: 100%;

                border-radius: 5px;

                cursor: pointer;

            }

            #button3:hover {

                background-color: rgb(50, 51, 51);

            }

            #button3:hover a {

                color: rgb(185, 185, 185);

            }

            #button3:hover i {

                color: rgb(185, 185, 185);

            }

            .link {

                text-align: center;

                line-height: 45px;

                text-decoration: none;

                color: rgb(50, 51, 51);

            }

            .img {

                text-align: center;

                padding: 10px;

            }

            #img1 {

                border-radius: 150%;

            }

            #im {

                color: rgb(65, 70, 75);

            }

            #im1 {

                color: rgb(65, 70, 75);

            }

            #im2 {

                color: rgb(65, 70, 75);

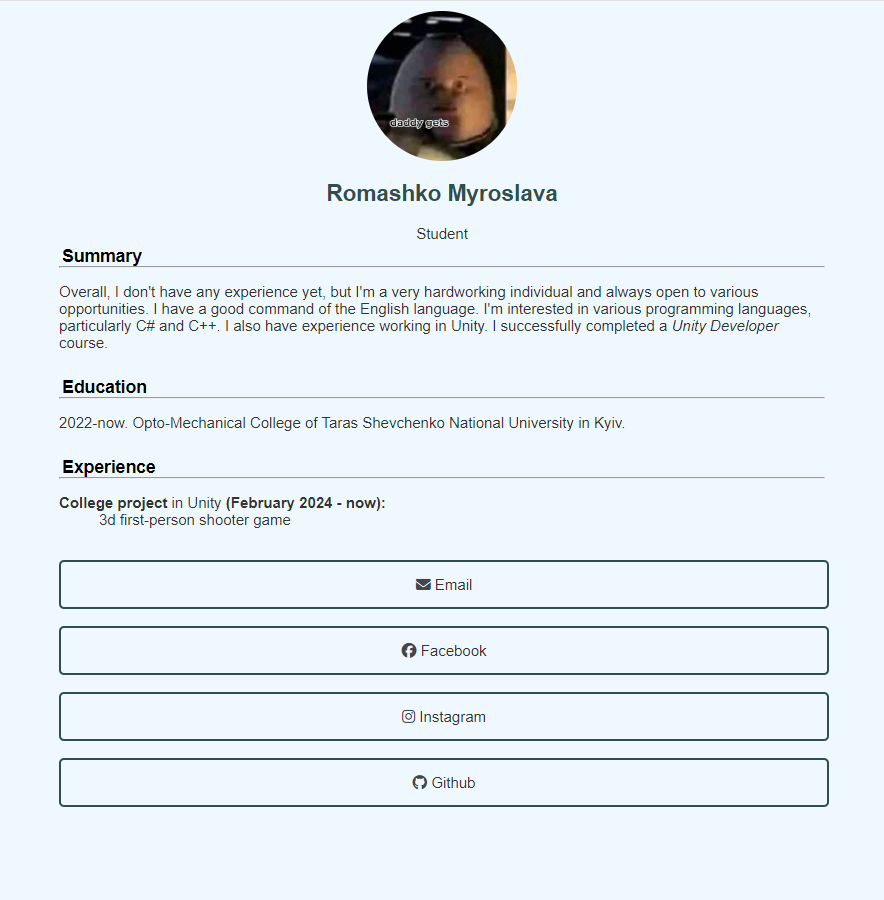
            }

            #im3 {

                color: rgb(65, 70, 75);

            }

        </style>



# 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>
14. Зображення біля посилань <https://fontawesome.com/icons/>

# ДОДАТКИ