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 |
| Касаткіна Любов Олексіївна  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) | |  | |
| *Касаткіної Любові Олексіївни* | | | | |
| (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 7](#_Toc167918692)

[1 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ 8](#_Toc167918693)

[2 РОЗДІЛ "ІНФОРМАТИКА" 9](#_Toc167918694)

[**2.1** **Розроблення засобами MS Word документа складної структури** 9](#_Toc167918695)

[**2.2 Розв‘язок задач за допомогою засобів табличного редактора MS Excel** 9](#_Toc167918696)

[**2.3 Виконання** **практичних завдань за допомогою засобів табличного редактора MS Access** 9](#_Toc167918697)

[3 РОЗДІЛ "ПРОГРАМУВАННЯ С++" 10](#_Toc167918698)

[**3.1 Теоретичні відомості** 10](#_Toc167918699)

[3.1.1 Загальні поняття. Елементи мови С++ - константи, змінні, операції, перетворення типів 10](#_Toc167918700)

[3.1.2 Арифметичні операції 10](#_Toc167918701)

[3.1.3 Поняття вхідного та вихідного потоку, найпростіші математичні функції 10](#_Toc167918702)

[3.1.4 Програмування лінійних та розгалужених обчислювальних процесів 10](#_Toc167918703)

[3.1.5 Програмування циклічних обчислювальних процесів 10](#_Toc167918704)

[**3.2 Практичне завдання** 11](#_Toc167918705)

[3.2.1 Завдання 11](#_Toc167918706)

[3.2.2 Постановка задачі 11](#_Toc167918707)

[3.2.3 Код програми 11](#_Toc167918708)

[3.2.4 Скріншот виконання програми 11](#_Toc167918709)

[**3.3 Виконання тестів** 11](#_Toc167918710)

[4 РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++" 12](#_Toc167918711)

[**4.1 Тема 1. Введення та виведення у базових консольних програмах** 12](#_Toc167918712)

[**4.2 Тема 2. Структури даних для управління станом програми** 15](#_Toc167918713)

[**4.3 Тема 3. Робота з файлами: запис та читання** 17](#_Toc167918714)

[**4.4 Тема 4. Формати файлів та їх обробка** 20](#_Toc167918715)

[**4..5 Тема 5. Системи контролю версій (VCS)** 20](#_Toc167918716)

[**4.6 Тема 6. Мова Markdown для опису проекті** 22](#_Toc167918717)

[**4.7 Розроблення ігрової програми** 24](#_Toc167918718)

[4.7.1 Код програми 24](#_Toc167918719)

[4.7.2 Скріншоти виконання програми 27](#_Toc167918720)

[5 РОЗДІЛ ”ВЕБ-ДИЗАЙН ТА HTML” 28](#_Toc167918721)

[**5.1 Мова розмітки HTML та структура Веб-сторінки** 28](#_Toc167918722)

[**5.2 Розмітка тексту за допомогою HTML. Зображення та посилання** 31](#_Toc167918723)

[**5.3 Каскадні таблиці стилів CSS та їх структура** 35](#_Toc167918724)

[**5.4 Спадковості і каскадування за допомогою CSS** 36](#_Toc167918725)

[**5.5 Практична частина** 39](#_Toc167918726)

[5.5.1 Постановки задачі 39](#_Toc167918727)

[5.5.2 HTML документ 40](#_Toc167918728)

[ВИСНОВКИ 41](#_Toc167918729)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ 42](#_Toc167918730)

[ДОДАТКИ 43](#_Toc167918731)

ОМФК.0121 НП 20

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

# ВСТУП

Автор: Касаткіна Любов Олексіївна

Дата: 17.05.2024 – 13.06.2024

Мета:

Перелік вивчених тем:

1. Засобами MS Word розробити документ складної структури та зберегти у різних форматах
2. Розв’язування рівнянь
3. Розв’язування оптимізаційних задач
4. Основи статистичного аналізу даних. Ряди даних. Кореляційний аналіз даних
5. Обчислення основних статистичних характеристик вибірки
6. Візуалізація рядів і трендів даних
7. Розв‘язок задач за допомогою засобів табличного редактора MS Excel
8. Проектування бази даних та створення бази даних MS ACCESS
9. Основні поняття реляційних баз даних
10. Введення, коригування, вилучення інформації з бази даних MS ACCESS
11. Вибірка, пошук, фільтрація даних в MS ACCESS
12. Виконання практичних завдань за допомогою засобів табличного редактора MS Access
13. Алгоритмізація обчислювальних процесів. Розробка постановки задачі, визначення вхідних та вихідних даних, оформлення блок-схем.
14. Типи даних в мові С/С++, внутрішнє представлення даних, операції перетворення типів даних.
15. Оператор присвоювання. Операції введення- виведення даних
16. Програмування лінійних та розгалужених обчислювальних процесів
17. Програмування циклічних обчислювальних процесів
18. Складання програм
19. Виконання тестів
20. Введення та виведення у базових консольних програмах
21. Структури даних для управління станом програми
22. Робота з файлами: запис та читання
23. Формати файлів та їх обробка
24. Системи контролю версій (VCS)
25. Мова Markdown для опису проекті
26. Розроблення ігрової програми
27. Мова розмітки HTML та структура Веб-сторінки
28. Розмітка тексту за допомогою HTML. Зображення та посилання
29. Каскадні таблиці стилів CSS та їх структура
30. Спадковості і каскадування за допомогою CSS
31. Розробка та оформлення Постановки задачі
32. Розробка HTML документу
33. Систематизація матеріалів, оформлення звіту з навчальної практики

# 1 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ

**Основні аспекти безпеки при роботі з комп’ютером**

**Віртуальна безпека**

Використання комп’ютера пов’язане з ризиком вірусів, шкідливих програм та кібератак. Щоб забезпечити віртуальну безпеку, потрібно:

**Установити антивірусне програмне забезпечення**

Встановлення надійного антивірусного програмного забезпечення допоможе захистити комп’ютер від шкідливих програм та вірусів. Регулярно оновлюйте антивірусну базу даних для забезпечення найвищого рівня захисту.

**Застосовувати оновлення системи**

Регулярно оновлюйте операційну систему та програмне забезпечення, оскільки це дозволяє усунути вразливості, які можуть бути використані хакерами для вторгнення в систему.

**Бережно поводьтесь з електронною поштою та посиланнями**

Не відкривайте незнайомі або сумнівні електронні листи та посилання. Такі повідомлення можуть містити шкідливі файли або посилання на фішингові сайти, призначені для викрадення особистих даних.

**Фізична безпека**

Фізична безпека також є важливим аспектом при роботі з комп’ютером. Ось декілька порад щодо фізичної безпеки:

**Вірно розташуйте робоче місце**

Переконайтеся, що ваше робоче місце обладнане комфортним столом та стільцем. Забезпечення належної підтримки для спини та шиї може допомогти уникнути болю та напруження.

**Забезпечте достатнє освітлення**

Маючи достатнє освітлення на робочому місці, ви зменшуєте напруження очей і запобігаєте втомі під час роботи за комп’ютером.

**Регулярно робіть перерви**

Не забувайте робити короткі перерви під час тривалої роботи за комп’ютером. Піднімайтесь, розтягуйтеся та робіть прості фізичні вправи для підтримки кровообігу та запобігання м’язовим напругам.

# 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**

Виведення

В програмуванні під оператором найчастіше розуміють деяку окрему інструкцію, вказівку комп’ютеру “що робити”.

Оператори можуть бути, як простими (одна інструкції) так і складеними (рос. “составными”).

Складені оператори формуються за допомогою ключових слів begin..end на мові Pascal, фігурних дужок { } на мовах Java, C/C++. Усі поміщені оператори між цих елементів інтерпретуються як одна(!) інструкція.

Одними з основних операторів простих програм (консольних) є оператори введення та виведення.

В C++ залишається можливість використання бібліотеки stdio, однак частіше використовують потокове введення/виведення з бібліотеки iostream

*#include <iostream>*

Дана бібліотека містить простір імен std який містить оператор cout (console out). Даний оператор містить наступний синтаксис:

*std::cout<<operand1<<operand2<<operand3<<…;*

Перед кожний елементом який необхідно вивести на екран записуються через << (дві кутні дужки вліво/два знака менше)

Наприклад:

*cout<<“Slava “<<31;*

*cout<<x<<y<<z;*

За замовчуванням пробіли між елементами при виведенні НЕ ставляться тому за необхідності їх треба поставити самому:

*cout<<x<<” “<<y<<” “<<z;*

Особливості виведення:

Такі мови як Java, C/C++ підтримують спеціальні керуючі послідовності символів у рядках. Які можна використовувати при виведенні. Усі такі послідовності починаються із символа “\” (backslash)

\n – новий рядок

\t – табуляція

Оскільки деякі деякі символи в мові програмування використовуються для позначення тих чи інших елементів, і розпізнаються як частина мови програмування, для їх виведення використовують додаткові керуючу послідовності

\\ – символ \

\” – символ “

\’ – символ ‘

\% – символ %

\? – символ ?

Введення

Якщо дані, які треба опрацювати завчасно не відомі, або можуть мати різне значення, для подальшої роботи з ними необхідно це значення дізнатись. Програма може отримувати дані з будь-яких джерел, одне з них – клавіатура

Для введення даних з клавіатури у консольному режимі роботи використовують стандартні оператори введення з клавіатури.

В мові C++ все так же є можливість використання форматного введення scanf, так найчастіше використовують потокове введення бібліотеки

Функція cin має свій особливий синтаксис. Кожний наступний елемент введення записується через оператор “>>” (дві кутні дужки вправо/ два знака більше). Імена змінних вводяться без операції розіменування (тобто без знака &).

Наприклад:

*std::cin>>x>>y>>z;*

Щоб не писати постійно std:: можна включити простір імен std на початку програми.

Приклад використання:

*#include <iostream.h>*

*int main (){*

*string str;*

*int age;*

*cout<<“Enter your family name: “;*

*cin>>str;*

*cout<<“Enter your age: “;*

*cin>>age;*

*cout<<“Mr. “<<str<<“, “<<age<<” years old.”<<endl;*

*cout<<“Enter x, y, z: “;*

*float x, y, z;*

*cin>>x>>y>>z;*

*return 0;*

*}*

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

Масив — згруповані за місцем розташування у пам'яті величини, що мають одну назву (ідентифікатор) і різні порядкові номери (індекси). Це поняття програмування відповідає математичним поняттям послідовності й таблиці (матриці).

Елемент масиву — одна з величин, що утворюють масив. Це поняття програмування відповідає математичному поняттю елемента послідовності чи матриці.

Індекс масиву — величина перелічуваного (зазвичай цілого) типу, яка (сукупність яких) вказує на конкретний елемент масиву. Це поняття програмування відповідає математичному поняттю номера елемента послідовності чи номеру рядка/стовпчика таблиці (матриці). У мові C++ найменше значення індексу — 0.

Масив має такі властивості:

* тип (типи) елементів (масиву);
* назва (ідентифікатор) — одна й та сама для всіх елементів масиву;
* розмірність — кількість номерів (індексів), необхідних для визначення розташування елемента у масиві;
* діапазони зміни номерів (індексів).

У мові C++ властивості елементів масиву не можна змінити протягом виконання програми. В деяких інших мовах це можливо.

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).

#include <list> — замовлення необхідної бібліотеки;

list <тип\_даних> назва\_списку; — опис списку;

list <тип\_даних> :: iterator назва\_ітератора; — опис ітератора (вказівника);

l.begin() — вказівник на початок списку l;

l.end() — вказівник на кінець списку l;

l.rbegin() — реверсивний вказівник на кінець списку l;

l.rend() — реверсивний вказівник на початок списку l;

l.empty() — істинність висловлювання: «Список l порожній»;

l.push\_back(e) — додати елемент e в кінець списку l;

l.pop\_back() — видалити останній елемент списку l;

l.front() — перший елемент списку l;

l.back() — останній елемент списку l;

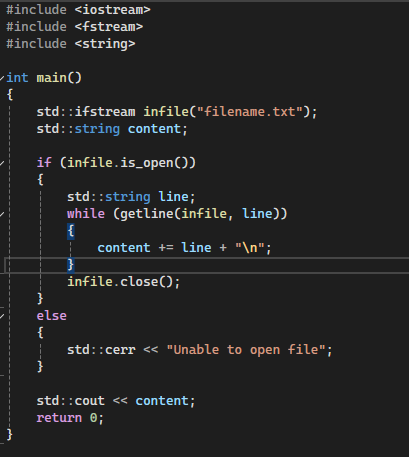
l.insert(i,e) — вставлення елемента e в позицію i списку l;

l.erase(j,k) — видалення зі списку l послідовності елементів від позиції j до позиції k

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

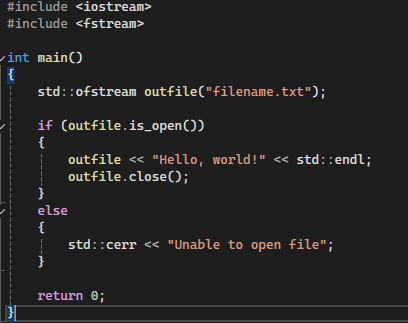
Робота з файлами в C++ також є досить простою і здійснюється за допомогою стандартної бібліотеки C++ <fstream>. Ця бібліотека надає класи ifstream для читання файлів, ofstream для запису файлів та fstream для читання і запису файлів.

Читання файлів:

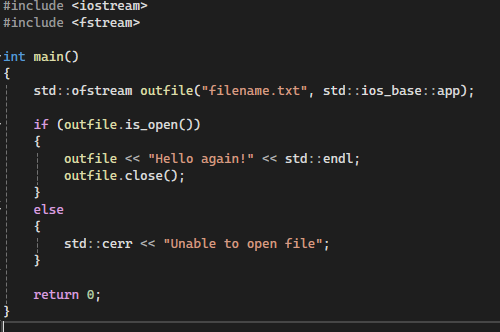
:

Запис у файли

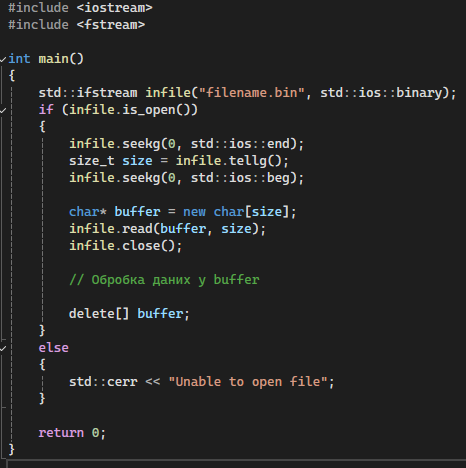
Запис тексту у файл:



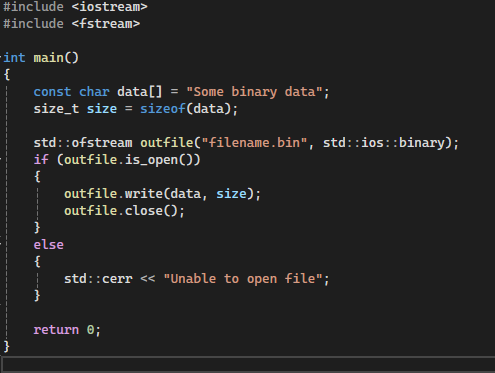
Додавання тексту у файл:



Читання бінарних файлів:



Запис бінарних файлів:



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

Формати файлів та їх обробка є ключовими аспектами роботи з інформацією у різних сферах діяльності. Ось деякі з найпоширеніших форматів файлів та способи їх обробки:

Текстові файли

1. TXT: простий текстовий файл без форматування. Використовується для зберігання необробленого тексту.

- Обробка: редагування у будь-якому текстовому редакторі (Notepad, Sublime Text), програмна обробка мовами Python, Java, C++ тощо.

2. CSV (Comma-Separated Values): формат для зберігання табличних даних, де значення розділені комами.

- Обробка: відкриття в електронних таблицях (Excel, Google Sheets), програмна обробка мовами Python (pandas), R тощо.

Date,Level,Message

2023-06-07,INFO,Application started

2023-06-07,ERROR,Failed to connect to database

2023-06-08,INFO,User logged in

3. JSON (JavaScript Object Notation): легкий формат обміну даними, який легко читається і записується людиною.

{

"team": "Warriors",

"wins": 50,

"losses": 20,

"players": [

{

"name": "Stephen Curry",

"position": "Point Guard",

"points\_per\_game": 29.1

},

{

"name": "Klay Thompson",

"position": "Shooting Guard",

"points\_per\_game": 22.3

}

]

}

- Обробка: програмна обробка мовами JavaScript, Python (json бібліотека), Java тощо.

4. XML (eXtensible Markup Language): формат для зберігання та передачі структурованих даних.

- Обробка: використання парсерів (lxml в Python, DOM у Java), редагування спеціалізованими редакторами (XMLSpy).

<team>

<name>Warriors</name>

<wins>50</wins>

<losses>20</losses>

<players>

<player>

<name>Stephen Curry</name>

<position>Point Guard</position>

<points\_per\_game>29.1</points\_per\_game>

</player>

<player>

<name>Klay Thompson</name>

<position>Shooting Guard</position>

<points\_per\_game>22.3</points\_per\_game>

</player>

</players>

</team>

JSON

Уявімо, що я розробляю веб-додаток для відстеження статистики спортивних команд, наприклад, команд NBA. У моему додатку потрібно зберігати інформацію про команди, їхні перемоги, поразки та гравців з їхніми індивідуальними статистиками. JSON буде ідеальним форматом для цього завдання.

Кожна команда має кількість перемог і поразок, а також список гравців з їхніми показниками, такими як ім'я, позиція на полі та середня кількість очок за гру. JSON дозволяє зберігати такі вкладені структури даних, що робить його зручним для представлення складних об'єктів. Крім того, JSON легко обробляється мовами програмування, зокрема JavaScript, який широко використовується на клієнтській стороні веб-додатків.

**CSV**

Уявімо інший сценарій: я адміністратор системи і мені потрібно зберігати логи роботи вашого додатка. Ці логи включають дату і час події, рівень важливості (інформація, помилка) та повідомлення. CSV буде найкращим вибором для цього завдання.

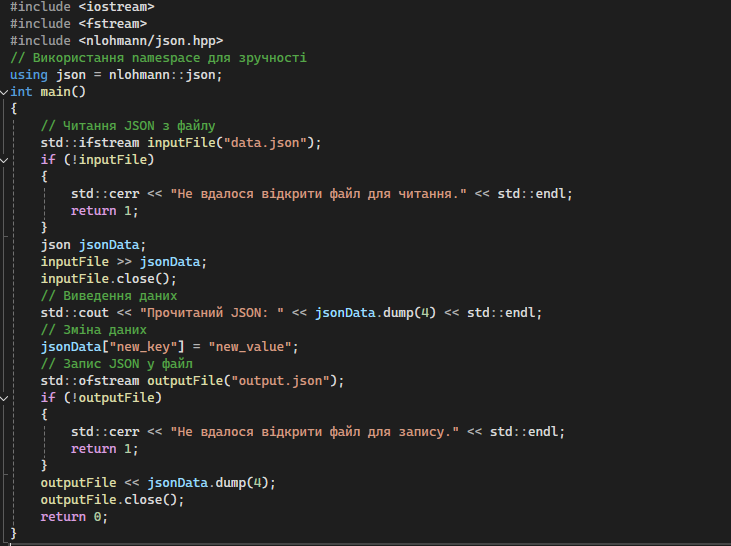
Лог-файли зазвичай складаються з багатьох рядків, де кожен рядок містить однаковий набір полів.

Файли у форматі CSV легко читати і записувати, вони займають мінімум місця, оскільки не містять додаткових символів чи тегів, як у випадку з JSON або XML. Крім того, CSV файли можна легко імпортувати в електронні таблиці (наприклад, Excel або Google Sheets) для подальшого аналізу.

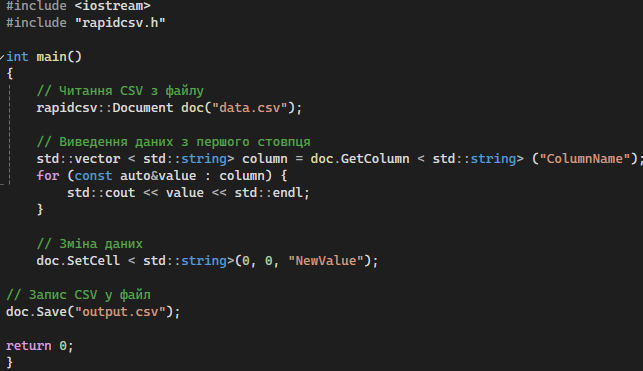
В C++ для роботи з файлами формату XML, CSV та JSON можна використовувати різні бібліотеки.

**Читання/запис JSON файлів**

Для роботи з JSON файлами можна використовувати бібліотеку nlohmann/json.

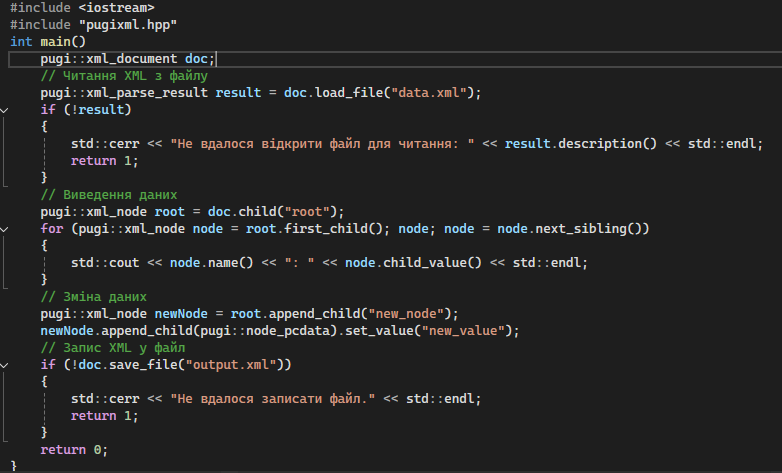


**Читання/запис CSV файлів**

Для роботи з CSV файлами можна використовувати бібліотеку rapidcsv. 

Читання/запис XML файлів

Для роботи з XML файлами можна використовувати бібліотеку pugixml.



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

Системи контролю версій (VCS) - це програмні інструменти, які допомагають керувати змінами у вихідному коді або інших типах файлів під час розробки програмного забезпечення. Основні завдання VCS включають відстеження змін, створення резервних копій, злиття змін, а також забезпечення можливості спільної роботи над проектами.

Основні типи VCS

1. Локальні VCS:

Зберігають версії файлів на локальному комп'ютері.

Простий, але має обмеження у можливостях командної роботи.

Приклад: RCS (Revision Control System).

2. Централізовані VCS (CVCS):

Використовують центральний сервер для зберігання всіх версій файлів.

Забезпечують спільну роботу над проектами, але є залежність від центрального сервера.

Приклад: SVN (Subversion), CVS (Concurrent Versions System).

3. Розподілені VCS (DVCS):

Кожен розробник має повну копію історії змін проекту на своєму локальному комп'ютері.

Більш гнучкі і стійкі до збоїв центрального сервера.

Приклад: Git, Mercurial.

**Переваги використання VCS**

Історія змін: відстеження кожної зміни у файлах з можливістю перегляду попередніх версій.

Відновлення: можливість повернення до будь-якої попередньої версії файлу.

Спільна робота: легке управління змінами від різних розробників.

Відгалуження та злиття: створення окремих гілок для нових функцій або виправлення помилок та їх подальше злиття з основною гілкою.

**Гілка (Branch)** у Git - це окрема версія проекту, яка існує незалежно від інших версій проекту. Гілки дозволяють розробникам працювати над різними завданнями або функціями одночасно, не заважаючи роботі один одного. Основна гілка в більшості проектів називається main або master. Коли створюється нова гілка, вона є копією основної гілки або будь-якої іншої гілки, від якої вона була створена.

**Коміт (Commit)** у Git - це зафіксований знімок змін у проекті. Коміт створюється, коли розробник виконує команду git commit, що зберігає поточний стан змін у репозиторії. Кожен коміт містить унікальний ідентифікатор (хеш), метадані (автора, дату) та повідомлення, яке описує зміни.

**Git Flow**

Git Flow - це методологія управління розробкою програмного забезпечення з використанням системи контролю версій Git. Вона допомагає командам організувати роботу над проектом, розділяючи його на різні гілки, які мають свої завдання та життєві цикли. Цей підхід включає в себе кілька основних і допоміжних гілок, які дозволяють легко керувати новими функціями, релізами і виправленнями багів.

Існує дві основні гілки, які завжди присутні у проекті:

- Main (або Master): ця гілка завжди містить стабільну версію коду, готову до випуску. Будь-який час, коли користувачі або клієнти отримують код, вони отримують його з цієї гілки. Всі релізи проекту йдуть з main гілки.

- Develop: це гілка, де ведеться активна розробка. Всі нові функції, вдосконалення і виправлення помилок спочатку додаються до цієї гілки. Вона є основною робочою гілкою для розробників. Після завершення роботи і тестування зміни з цієї гілки потрапляють до main через процес підготовки релізу.

Окрім основних гілок, Git Flow передбачає використання допоміжних гілок для різних типів завдань:

- Feature гілки: використовуються для розробки нових функцій. Кожна нова функція створюється в окремій гілці, яка береться з develop. Після завершення роботи над функцією, ця гілка зливається назад в develop. Це дозволяє працювати над новими функціями незалежно від основного коду.

- Release гілки: використовуються для підготовки нового релізу. Коли гілка develop готова до випуску, створюється release гілка. В цій гілці відбувається останнє тестування і виправлення незначних помилок перед релізом. Після завершення, release гілка зливається в main (що створює новий реліз) і в develop (для синхронізації всіх виправлень).

- Hotfix гілки: використовуються для термінового виправлення критичних багів в основній гілці. Вони створюються від main, і після виправлення, зміни зливаються назад у main і develop. Це дозволяє швидко реагувати на критичні проблеми, не заважаючи поточній розробці.

Git Flow надає структурований підхід до розробки, забезпечуючи чітке розділення завдань і гнучкість у керуванні проектом. Це дозволяє командам ефективно співпрацювати, впроваджувати нові функції, готувати релізи та швидко реагувати на проблеми, зберігаючи при цьому стабільність і якість коду.

Основні команди Git

Git є найпопулярнішою DVCS і має потужний набір команд для управління версіями. Ось деякі з основних команд Git:

Ініціалізація репозиторію:

*git init*

Клонування існуючого репозиторію:

*git clone https://github.com/user/repository.git*

Перегляд статусу файлів:

*git status*

Додавання файлів до індексу:

*git add filename*

*git add .*

Фіксація змін:

*git commit -m "Your commit message"*

Перегляд історії комітів*:*

*git log*

Створення нової гілки:

*git branch new-branch*

Перемикання між гілками:

*git checkout new-branch*

Злиття гілок:

*git merge branch-name*

Оновлення локального репозиторію з віддаленого:

*git pull*

Відправка локальних змін до віддаленого репозиторію:

*git push*

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

Markdown для опису проекту:

### Заголовки

# Заголовок 1 рівня

## Заголовок 2 рівня

### Заголовок 3 рівня

### Виділення тексту

\*Курсив\* або \_Курсив\_

\*\*Жирний\*\* або \_\_Жирний\_\_

\*\*\*Жирний курсив\*\*\* або \_\_\_Жирний курсив\_\_\_

### Списки

\*\*Нумеровані:\*\*

1. Пункт 1

2. Пункт 2

3. Пункт 3

\*\*Марковані:\*\*

- Пункт 1

- Пункт 2

- Пункт 3

### Посилання та зображення

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

![Альтернативний текст](https://example.com/image.jpg)

### Код

\*\*Інлайн-код:\*\*

`інлайн-код`

С++

#include <iostream>

int main() {

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

return 0;

}

### Таблиці

| Заголовок 1 | Заголовок 2 |

| ----------- | ----------- |

| Рядок 1 | Дані 1 |

| Рядок 2 | Дані 2 |

### Цитати та горизонтальні лінії

\*\*Цитати:\*\*

> Це цитата

\*\*Горизонтальна лінія:\*\*

---

## **4.7 Розроблення ігрової програми**

### 4.7.1 Код програми

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

// Розмір дошки

const int SIZE = 3;

// Структура для зберігання історії перемог

struct Score {

int xWins;

int oWins;

int draws;

};

// Функція для зчитування історії перемог з файлу

Score readScoreFromFile(const string& filename) {

ifstream file(filename);

Score score = { 0, 0, 0 };

if (file.is\_open()) {

file >> score.xWins >> score.oWins >> score.draws;

file.close();

}

return score;

}

// Функція для запису історії перемог у файл

void writeScoreToFile(const string& filename, const Score& score) {

ofstream file(filename);

if (file.is\_open()) {

file << score.xWins << " " << score.oWins << " " << score.draws;

file.close();

}

}

// Функція для відображення дошки

void displayBoard(char board[SIZE][SIZE]) {

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

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

cout << board[i][j];

if (j < SIZE - 1) cout << " | ";

}

cout << endl;

if (i < SIZE - 1) cout << "--+---+--" << endl;

}

}

// Функція для перевірки переможця

char checkWinner(char board[SIZE][SIZE]) {

// Перевірка рядків і стовпців

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

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

return board[i][0];

}

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

return board[0][i];

}

}

// Перевірка діагоналей

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

return board[0][0];

}

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

return board[0][2];

}

// Нічия

return ' ';

}

// Функція для перевірки, чи залишилися пусті клітинки

bool isBoardFull(char board[SIZE][SIZE]) {

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

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

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

return false;

}

}

}

return true;

}

int main() {

const string filename = "score.txt";

Score score = readScoreFromFile(filename);

char board[SIZE][SIZE] = {

{' ', ' ', ' '},

{' ', ' ', ' '},

{' ', ' ', ' '}

};

char currentPlayer = 'X';

bool gameOver = false;

while (!gameOver) {

displayBoard(board);

int row, col;

cout << "Player " << currentPlayer << ", enter your move (row and column): ";

cin >> row >> col;

if (row >= 0 && row < SIZE && col >= 0 && col < SIZE && board[row][col] == ' ') {

board[row][col] = currentPlayer;

char winner = checkWinner(board);

if (winner != ' ') {

displayBoard(board);

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

if (winner == 'X') {

score.xWins++;

}

else {

score.oWins++;

}

gameOver = true;

}

else if (isBoardFull(board)) {

displayBoard(board);

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

score.draws++;

gameOver = true;

}

else {

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

}

}

else {

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

}

}

writeScoreToFile(filename, score);

cout << "X wins: " << score.xWins << endl;

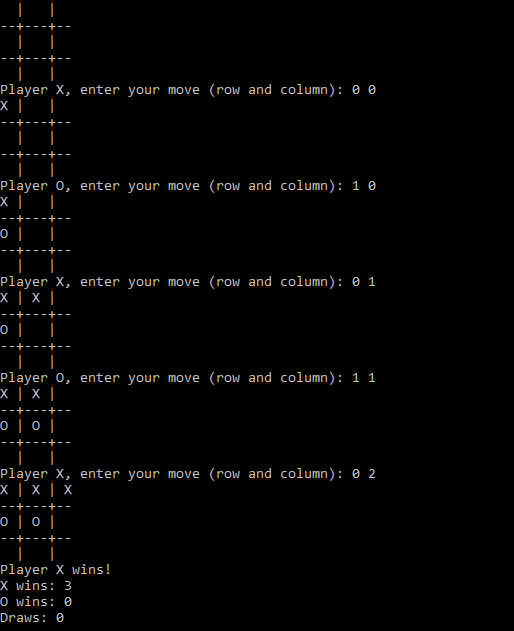
cout << "O wins: " << score.oWins << endl;

cout << "Draws: " << score.draws << endl;

return 0;

}

### 4.7.2 Скріншоти виконання програми



# 5 РОЗДІЛ ”ВЕБ-ДИЗАЙН ТА HTML”

HTML (HyperText Markup Language - «мова гіпертекстової розмітки») є стандартною мовою для створення розмітки документів у Всесвітній павутині. Ця мова використовується для опису структури більшості веб-сторінок, які потім інтерпретуються браузерами і відображаються на екранах комп'ютерів або мобільних пристроїв у форматованому вигляді.

XHTML, більш сувора версія HTML, базується на синтаксисі XML і також застосовується для розмітки гіпертексту. Веб-сторінки, написані на HTML, зазвичай передаються від серверів до браузерів через протоколи HTTP або HTTPS, або у вигляді простого тексту, або з використанням шифрування.

HTML була створена британським вченим Тімом Бернерсом-Лі між 1986 і 1991 роками. Ця мова призначалася для обміну науковою та технічною документацією і була розроблена для людей, які не є фахівцями у верстці. HTML вирішила проблему складності SGML, ввівши невеликий набір структурних та семантичних елементів, відомих як дескриптори або «теги». Завдяки HTML можна легко створювати відносно прості, але гарно оформлені документи. Окрім спрощення структури документа, HTML також підтримує гіпертекст, а згодом до нього були додані мультимедійні можливості.

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

Тег - це вказівка браузера на те, як відображати фрагмент документа. Тег завжди починається символом "<" і закінчується символом ">". Між цими символами вказується імʼя тега. В імені малі й великі літери не розрізняються.  
Теги бувають двох видів - одинарні й парні.  
Одинарний тег ставиться перед фрагментом тексту, яким він керує. Наприклад,

запис  
<br> *Зошит* означає, що слово „Зошит" відображається з нового рядка.  
Парні теги складаються з двох частин - початкової та завершальної. Початкова частина вказує точку документа, з якої починається дія команди, а завершальна - точку її закінчення. Завершальна частина відрізняється від початкової наявністю символу « / » (похила риска) перед імʼям команди. Наприклад, запис <b> *Синій олівець* </b> означає, що текст „Синій олівець" відображається напівжирним шрифтом.  
Розроблення Web-сторінки складається з двох етапів:

* Створення макета
* Вставка тегів мови HTML.

При створенні макета визначається зміст Web-сторінки та іІ зовнішній вигляд.  
Вставка тегів забезпечує потрібний зовнішній вигляд при перегляді Web-сторінки за допомогою браузера.  
Ознакою того, що текстовий файл є HTML-документом, служить парний тег  
<HTML> </HTML>. Він починає й завершує весь документ, тобто HTML-документ має вигляд:

<HTML>  
текст документа  
</HTML>  
HTML-документ складається з двох частин - заголовка й тіла. Заголовок позначається парним тегом <HEAD> ... </HEAD>, а тіло - парним тегом <BODY> ...

</BODY>. Текст, що вводиться в заголовку вікна браузера, розміщається в парному тезі <TITLE> </TITLE>. Структура HTML-документа має вигляд  
<HTML>  
<HEAD>  
<TITLE> Заголовок документа </TITLE>  
</HEAD>  
<BODY>  
Тіло документа  
</BODY>  
</HTML>  
У будь-якому місці HTML-документа можна помістити коментар. Це текст, який служить для пояснень розробника Web-сторінки та не відображається на екрані під час перегляду Web-сторінки у браузері. Текст коментаря поміщається в парний тег <!- … ->

**Форматування абзаців і символів.**

Фрагмент HTML-документа  
<Н1> Текст....... </H1> відображається як заголовок першого рівня.  
Абзаци й заголовки можна вирівнювати по центру, лівому чи правому краю.

Для цього в імені команди в початковій частині тега вказується параметр ALIGN (вирівняти) й після знака «=» (дорівнює) одне з наступних значень:

CENTER - по центру;  
LEFT - по лівому краю;  
RIGHT - по правому краю.  
Якщо в тексті HTML-документа існують додаткові пробіли між словами, то при виводі в браузері вони можуть вилучатися. Щоб зберегти пробіли при відображенні, треба вказати парний тег <pre>... </pre>. Він позначає попередньо відформатований текст, тобто текст, що відображається шрифтом фіксованої ширини із збереженням усіх пробілів. Часто цей тег використовується для виведення тексту у вигляді таблиці.

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

**Абзаци**

В минулій лекції ми познайомилися з тегами, необхідними для створення найпростішої HTML-сторінки, і з деякими службовими тегами, які не відображаються в браузері. На цій лекції ми будемо вивчати теги для логічної розмітки тексту. Використовувати їх можна тільки всередині тега <body>.

Почнемо з найпростішого тега <p> (paragraph), 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 Щоб текст сторінки не відображався суцільним текстом, використовують такі одинарні теги:  
<p> - початок абзацу (текст після тега починається з нового абзацу);  
<bг> - початок нового рядка (текст після тега починається з нового рядка);  
«hr> - горизонтальна лінія (у рядку, де розміщено тег, відображається горизонтальна лінія).  
При використанні тега <р> спочатку вставляється порожній рядок. Далі текст виводиться з початку нового рядка. Якщо вжито тег <br>, то порожній рядок не  
вставляеться.  
Перед деякими фрагментами тексту можуть розміщатися заголовки. Вони виділяються більшим шрифтом. HTML має теги для завдання величини заголовків (шість рівнів). Найбільшим шрифтом виділяються заголовки першого рівня, а найдрібнішим - шостого.  
Текст заголовка визначається парним тегом

<Нn> … </Н>, де n - номер рівня.  
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, тому необхідно вчитися правильно їх використовувати.

**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».

Єдине, що дозволено в межах розмітки маркованого списку (тобто між відкриваючим і закриває тегами ul), - це один або кілька елементів списку. Ви не можете вставити туди інші елементи, і там не може бути нерозміченого тексту. Наприклад:

<ul>                                                 - відкриваємо маркований список

         <li>Перший пункт списку</li>       -додаємо пункти списку

<li>Другий пункт списку</li>        -додаємо пункти списку

<li>Наступний пункт списку</li>  -додаємо пункти списку

</ul>                                                - закриваємо маркований список

**Робота з картинками.**

**Додання зображень**

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.

**Розміри зображень**

Щоб управляти шириною або висотою зображення, потрібно використовувати атрибути width і height. Приклад:

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

В прикладі зображенню задана ширина 100px (пікселів). Зверніть увагу, що в атрибуті width після цифри немає px. Якщо ви задаєте розмір картинки в пікселях, то використовуйте просто цифри. Додавати px не потрібно, такий стандарт.

У наступному прикладі, зображенню задана відносна ширина 50 відсотків:

<img width = "50%" src = "..."

Висоту у відсотках зазвичай не задають. Якщо задати лише один з розмірів, ширину або висоту, то другу розмірність браузер обчислить самостійно виходячи з пропорцій зображення. Якщо ж задати і ширину, і висоту для картинки:

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

то браузер може порушити пропорції вихідного зображення.

**Альтернативний текст**

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). Приклад:

<img src = "cat.png" alt = "Кіт у повному розквіті сил">

**Зображення з посиланням**

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

<a href="[http://keksby.ru](http://keksby.ru/)">

   <img src = "cat.png" alt = "Кіт">

</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 (CSS) представляють собою набір параметрів форматування, які застосовуються до елементів веб-сторінки для контролю їхнього вигляду та розташування. CSS можна застосовувати до елементів HTML трьома способами:

- Inline (вбудований або рядковий) - за допомогою атрибуту style в HTML елементах.

- Internal (внутрішній) - за допомогою елемента <style> у розділі <head>.

- External (зовнішній) - через зовнішній файл CSS.

Властивість CSS background є скороченням для набору фонових властивостей. Властивість background-color визначає колір фону для будь-якого елемента в CSS і приймає будь-яке дійсне значення <color>. Background-color поширюється на поле вмісту та заповнення елемента. Фонові зображення можна додавати за допомогою властивості background-image, яка дозволяє відображати зображення на фоні елемента.

Властивість background-repeat використовується для контролю поведінки повторення фонових зображень. Доступні такі значення:

- no-repeat — повністю припиняє повторення фону.

- repeat-x — повторює по горизонталі.

- repeat-y — повторює вертикально.

- repeat — за замовчуванням повторює в обох напрямках.

- space — повторює стільки разів, скільки можливо, додаючи пробіл між зображеннями, якщо є додатковий простір.

- round — подібний до space, але розтягує зображення, щоб заповнити додатковий простір.

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

**Абзаци**

**Спадковість**

Спадковість в CSS визначає, як властивості стилів передаються від батьківських елементів до дочірніх. Деякі властивості автоматично наслідуються дочірніми елементами, наприклад, властивості шрифту і кольору тексту.

Приклад: Якщо на батьківському елементі встановлено колір тексту, всі його дочірні елементи також отримають цей колір, якщо не вказано інше.

body {

color: blue;

}

p {

/\* Абзаци наслідують синій колір від body \*/

}

**Каскадування**

Каскадування визначає, як CSS обробляє конфлікти між різними правилами стилів. При виникненні конфлікту між кількома правилами для одного й того ж елемента, застосовується правило з найбільшою специфічністю. Специфічність визначається за такими критеріями:

1. Ідентифікатори (ID) мають найвищу специфічність.
2. Класи, псевдокласи та атрибути мають середню специфічність.
3. Теги та псевдоелементи мають найнижчу специфічність.

Приклад: Правила з вищою специфічністю перекривають правила з нижчою специфічністю.

#header {

color: red; /\* Найвища специфічність \*/

}

.header {

color: green; /\* Середня специфічність \*/

}

header {

color: blue; /\* Найнижча специфічність \*/

}

**Використання важливості (!important)**

Щоб перекрити всі інші правила, можна використовувати декларацію !important. Це змушує браузер застосувати це правило незалежно від специфічності інших правил.

Приклад:

p {

color: green !important;

}

**Порядок підключення стилів**

Порядок підключення стилів також впливає на те, яке правило буде застосоване. Правила, що з'являються пізніше в CSS файлі або в документах HTML, мають вищий пріоритет.

Приклад:

Html

<head>

<style>

p {

color: red;

}

</style>

<style>

p {

color: blue;

}

</style>

</head>

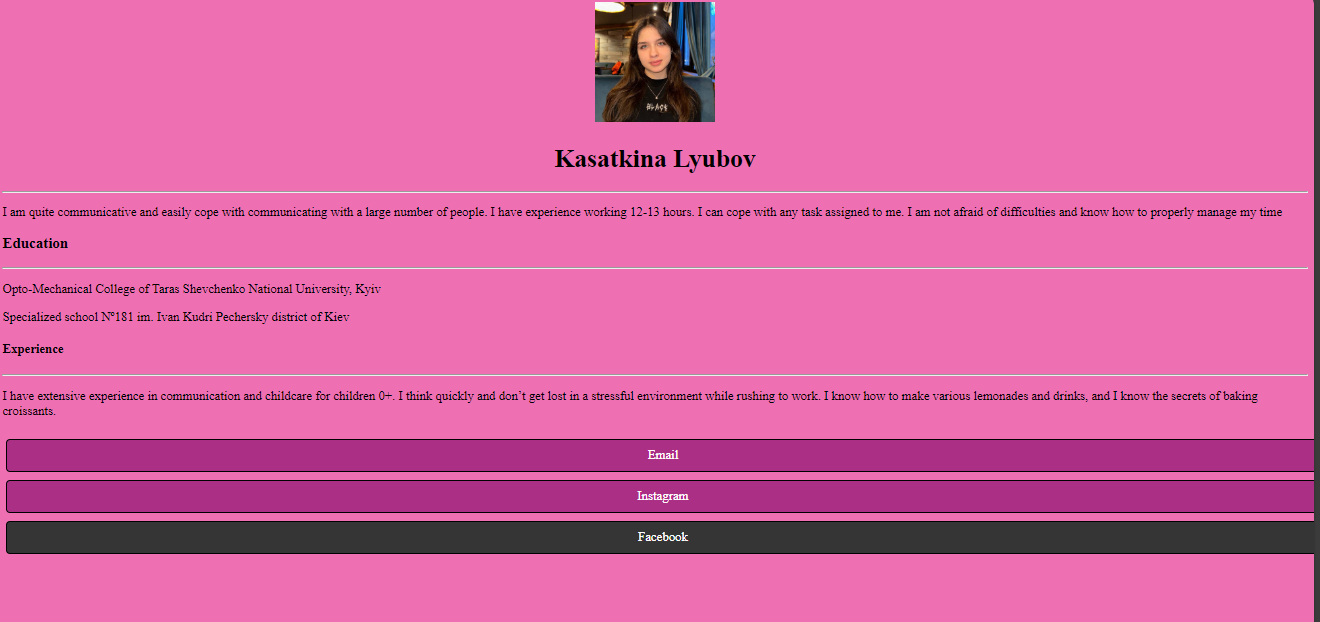
У цьому прикладі абзаци будуть синіми, оскільки друге правило з'являється пізніше.

## **5.5 Практична частина**

### 5.5.1 Постановки задачі

Основним завданням було створення веб-сайту у формі резюме. Метою цієї роботи було вивчення мови розмітки HTML та структури веб-сторінки, освоєння CSS та принципів каскадного оформлення стилів, а також оволодіння їх основними функціями. Мій проєкт містить текст, відформатований за допомогою зміни кольору, розміру шрифту, стилю та інших параметрів для покращення читабельності. Також у ньому є фотографії, активні посилання для зручного переходу між розділами, та елементи оформлення, що забезпечують краще сприйняття відвідувачами веб-сайту.

### 5.5.2 HTML документ



# ВИСНОВКИ

Протягом цієї практики я отримала цінний досвід роботи з HTML, що є фундаментом для побудови будь-якого веб-сайту. Я навчилася створювати різні елементи веб-сторінки, такі як заголовки, параграфи, списки, зображення та посилання. Це дало мені розуміння, як організувати контент у логічній та структурованій формі.

Оволодіння CSS дозволило мені значно покращити зовнішній вигляд та функціональність веб-сайту. Я вивчила, як застосовувати стилі до різних елементів, змінюючи колір, розмір шрифту, стиль та інші параметри, що суттєво покращило читабельність та естетичне сприйняття контенту. Також я навчився використовувати CSS для розташування елементів на сторінці, що допомогло створити зручну та інтуїтивно зрозумілу навігацію.

У процесі створення проєкту я інтегрувала фотографії та активні посилання, що забезпечили зручний перехід між розділами веб-сайту. Це додало проєкту динамічності та зробило його більш інтерактивним і корисним для користувачів. Використання елементів оформлення допомогло виділити ключові аспекти резюме та зробило загальний дизайн привабливішим.

Загалом, навчальна практика дала мені глибше розуміння основ веб-розробки та дозволила застосувати отримані знання на практиці. В результаті я створила функціональний і привабливий веб-сайт-резюме, що відображає не лише мої навички та досягнення, але й мою здатність ефективно використовувати HTML та CSS для досягнення поставлених цілей. Цей досвід став важливим кроком у моєму професійному розвитку та підготував мене до подальшого вивчення більш складних аспектів веб-розробки

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ

1. ГОСТ 2.105-95 ЕСКД Общие требования к текстовым документам.

2. ДСТУ 1.5-93 Загальні вимоги до побудови, викладу, оформлення та змісту.

3. 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>

4. Урок: Структура веб-сторінок. Поняття про мову HTML: <https://vseosvita.ua/lesson/struktura-veb-storinok-poniattia-pro-movu-html-71025.html>

5. 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>

6. Е-helper.com.ua «Лекція 2. Розмітка тексту. Зображення. Посилання.»: <http://e-helper.com.ua/node/949>

7. Е-helper.com.ua «Лекція 5. Наслідування та каскадування. Оформлення тексту за допомогою CSS»:

<http://e-helper.com.ua/node/952>

# ДОДАТКИ