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-21 |
| Богатиирьов Олексій Русланович  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) | |  | |
| *Богатирьов Олексій Русланович* | | | | |
| (П.І.Б студента.) | | | | |
| Спеціальність | *121 Інженерія програмного забезпечення* | Група | | *ІПЗ-21* |

|  | **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**

[ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 6](#_Toc167919156)

[ВСТУП 7](#_Toc167919157)

[1 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ 8](#_Toc167919158)

[2 РОЗДІЛ "ІНФОРМАТИКА" 9](#_Toc167919159)

[**2.1** **Розроблення засобами MS Word документа складної структури** 9](#_Toc167919160)

[**2.2 Розв‘язок задач за допомогою засобів табличного редактора MS Excel** 9](#_Toc167919161)

[**2.3 Виконання** **практичних завдань за допомогою засобів табличного редактора MS Access** 9](#_Toc167919162)

[3 РОЗДІЛ "ПРОГРАМУВАННЯ С++" 10](#_Toc167919163)

[**3.1 Теоретичні відомості** 10](#_Toc167919164)

[3.1.1 Загальні поняття. Елементи мови С++ - константи, змінні, операції, перетворення типів 10](#_Toc167919165)

[3.1.2 Арифметичні операції 10](#_Toc167919166)

[3.1.3 Поняття вхідного та вихідного потоку, найпростіші математичні функції 10](#_Toc167919167)

[3.1.4 Програмування лінійних та розгалужених обчислювальних процесів 10](#_Toc167919168)

[3.1.5 Програмування циклічних обчислювальних процесів 10](#_Toc167919169)

[**3.2 Практичне завдання** 11](#_Toc167919170)

[3.2.1 Завдання 11](#_Toc167919171)

[3.2.2 Постановка задачі 11](#_Toc167919172)

[3.2.3 Код програми 11](#_Toc167919173)

[3.2.4 Скріншот виконання програми 11](#_Toc167919174)

[**3.3 Виконання тестів** 11](#_Toc167919175)

[4 РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++" 12](#_Toc167919176)

[**4.1 Тема 1. Введення та виведення у базових консольних програмах** 12](#_Toc167919177)

[**4.2 Тема 2. Структури даних для управління станом програми** 12](#_Toc167919178)

[**4.3 Тема 3. Робота з файлами: запис та читання** 12](#_Toc167919179)

[**4.4 Тема 4. Формати файлів та їх обробка** 12](#_Toc167919180)

[**4..5 Тема 5. Системи контролю версій (VCS)** 13](#_Toc167919181)

[**4.6 Тема 6. Мова Markdown для опису проекті** 13](#_Toc167919182)

[**4.7 Розроблення ігрової програми** 13](#_Toc167919183)

[4.7.1 Код програми 13](#_Toc167919184)

[4.7.2 Скріншоти виконання програми 13](#_Toc167919185)

[5 РОЗДІЛ ”ВЕБ-ДИЗАЙН ТА HTML” 14](#_Toc167919186)

[**5.1 Мова розмітки HTML та структура Веб-сторінки** 14](#_Toc167919187)

[**5.2 Розмітка тексту за допомогою HTML. Зображення та посилання** 14](#_Toc167919188)

[**5.3 Каскадні таблиці стилів CSS та їх структура** 14](#_Toc167919189)

[**5.4 Спадковості і каскадування за допомогою CSS** 14](#_Toc167919190)

[**5.5 Практична частина** 15](#_Toc167919191)

[5.5.1 Постановки задачі 15](#_Toc167919192)

[5.5.2 HTML документ 15](#_Toc167919193)

[ВИСНОВКИ 16](#_Toc167919194)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ 17](#_Toc167919195)

[ДОДАТКИ 18](#_Toc167919196)

ОМФК.0121 НП 20

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

# ВСТУП

Автор: Богатирьов Олексій Русланович

Дата: 21.05.2024 - 13.06.2024

Мета:

Перелік вивчених тем:

# 1 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ

#### Віртуальна безпека

Робота з комп’ютером пов’язана з ризиком вірусів, шкідливих програм та кібератак. Для забезпечення віртуальної безпеки необхідно:

1. **Установити антивірусне програмне забезпечення**: Надійний антивірус допоможе захистити комп’ютер від шкідливих програм та вірусів. Регулярно оновлюйте антивірусну базу даних для найвищого рівня захисту.
2. **Застосовувати оновлення системи**: Регулярне оновлення операційної системи та програмного забезпечення усуває вразливості, які можуть бути використані хакерами для вторгнення в систему.
3. **Бережно поводитися з електронною поштою та посиланнями**: Уникайте відкривання незнайомих або підозрілих електронних листів та посилань, які можуть містити шкідливі файли або вести на фішингові сайти для викрадення особистих даних.

#### Фізична безпека

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

1. **Правильне розташування робочого місця**: Переконайтеся, що ваше робоче місце обладнане зручним столом і стільцем. Належна підтримка спини та шиї допоможе уникнути болю та напруження.

# 2 РОЗДІЛ "ІНФОРМАТИКА"

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

Оформлення документа охоплює його зовнішню і внутрішню структуру. Зовнішня структура включає параметри сторінок, поля і подібні аспекти, тоді як внутрішня структура описує заголовки, підписи та інші елементи вмісту.

При роботі з великими та складними документами, їх розбивають на частини, розділи, підрозділи, параграфи та інші структурні елементи, які форматуються згідно з певними правилами. Частіше всього така структура використовується для документів з багатьма сторінками, де вони поділяються на розділи, параграфи та пункти.

Створення структури текстового документа у програмі Word відбувається за допомогою стилів форматування текстових об'єктів, таких як "Заголовок 1", "Заголовок 2" і т. д. Для цього необхідно назви структурних частин документа відформатувати відповідно до відповідних стилів. Наприклад, заголовкам першого рівня (розділів) слід застосовувати стиль "Заголовок 1", а заголовкам другого рівня (пунктів) - стиль "Заголовок 2"

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

Microsoft Excel є одним із найбільш популярних програмних засобів для створення і редагування таблиць у всьому світі, розробленим компанією Microsoft. Він містить безліч потужних інструментів, які дозволяють користувачам не лише створювати й форматувати таблиці, а й виконувати широкий спектр обчислень, обробки даних та аналізу.

Основні можливості Excel включають:

1. Створення та форматування таблиць: Ця програма надає можливість легко створювати таблиці, додавати дані та налаштовувати їх вигляд за допомогою різноманітних інструментів для кольорів, шрифтів, рамок тощо.
2. Обчислення та формули: Excel має вбудовані функції і формули для автоматизації різноманітних обчислень. Ви можете виконувати арифметичні операції, обчислювати суми, середні значення, мінімуми та максимуми, а також використовувати складніші функції для аналізу даних.
3. Аналіз даних та графіки: Excel надає можливість виконувати різноманітний аналіз даних, такий як сортування, фільтрація, пошук значень, підсумовування. Крім того, ви можете побудувати різні типи графіків (стовпчасті, лінійні, кругові) для візуалізації даних.
4. Імпорт та експорт даних: Excel підтримує імпорт та експорт даних з різних джерел, таких як бази даних, текстові файли, інші таблиці Excel, що спрощує роботу з інформацією з різних джерел.
5. Автоматизація та макроси: Excel дозволяє створювати макроси і автоматизовані рішення за допомогою мови програмування VBA (Visual Basic for Applications), що полегшує автоматизацію повторюваних завдань та створення додаткових функціональних можливостей.
6. Спільна робота та спільний доступ: Excel підтримує спільну роботу над документами через хмарні сервіси, що дозволяє користувачам працювати в групах над одним файлом і обмінюватися даними з іншими користувачами.

## **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

Мову C++ можна вважати розширенням мови C, оскільки вона зберігає всі можливості мови C та додає до них засоби об’єктно-орієнтованого програмування. C++ використовується для розробки системних та складних програм і є не лише найпопулярнішою мовою програмування, а й важливим засобом спілкування серед програмістів, оскільки на ній пишуться більшість програм та алгоритмів.

Мова C++ є високорівневою і використовується як основа для багатьох систем програмування, таких як Visual Studio і Eclipse. Один з найбільш популярних інструментів для розробки на C++ - Visual Studio. Цей інструмент дозволяє як початківцям, так і професіоналам створювати інтерфейси користувача для різноманітних програм різного рівня складності.

Константи у програмуванні - це фіксовані значення, які не можуть змінюватися під час виконання програми. Вони поділяються на літеральні та типізовані константи в залежності від їхнього типу.

Змінна - це іменована область пам’яті, яка містить дані визначеного типу. Кожна змінна має своє ім'я та інші характеристики, такі як розмір та видимість. Ім'я змінної використовується для доступу до її значення. Перед використанням змінної її потрібно оголосити та видалити для неї пам'ять.

Мова C++ має широкий набір операцій для маніпулювання даними та обчислення виразів. Операції містять дії, які виконуються над операндами (об'єктами), і мають відповідні позначення (наприклад, операція порівняння - "==", операція обчислення залишку від ділення цілих чисел - "%").

### 

### 3.1.2 Арифметичні операції

Операції арифметики в мові програмування C++ включають такі можливості:

* Додавання: + додає значення П до Л.
* Віднімання: - віднімає значення П від Л.
* Зміна знаку: - унарна операція, яка змінює знак величини П.
* Множення: \* множить значення П на Л.
* Ділення: / ділить значення Л на П.
* Залишок від ділення: % повертає залишок від ділення Л на П для цілих чисел.
* Інкремент: ++ збільшує значення змінної на 1 до або після використання.
* Декремент: -- зменшує значення змінної на 1 до або після використання.

Операції присвоювання у мові C++ включають такі можливості:

* Просте присвоювання: = присвоює значення П змінній Л.
* Складне присвоювання: +=, -= додає або віднімає величину П до або від змінної Л.
* \*=, /=, %= множать, ділять або знаходять залишок від ділення змінної Л на величину П.

Операції відношення порівнюють значення Л зі значенням П:

* Менше: < - вказує, що Л менше П.
* Більше: > - вказує, що Л більше П.
* Рівне: == - вказує, що Л дорівнює П.
* Нерівне: != - вказує, що Л не дорівнює П.
* Менше або дорівнює: <= - вказує, що Л менше або дорівнює П.
* Більше або дорівнює: >= - вказує, що Л більше або дорівнює П.

Логічні операції оперують з цілими числами:

* Логічне "AND" (&&) повертає true (1), якщо обидва операнди є true.
* Логічне "OR" (||) повертає true, якщо хоча б один операнд є true.
* Логічне "NOT" (!) змінює true на false і навпаки.

Наприклад, умова 3 < x < 5 записується як x > 3 && x < 5 в мові програмування C++.

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

Пояснення про вхідні та вихідні потоки у програмуванні зазвичай виглядає так:

* Вхідний потік (input stream): Це засіб отримання даних для програми з зовнішніх джерел, таких як клавіатура, файли або інші пристрої введення. Використовується для зчитування даних у програмі.
* Вихідний потік (output stream): Це засіб виведення даних з програми до зовнішніх пристроїв, таких як екран, файли або інші пристрої виведення. Використовується для відображення результатів роботи програми.

У більшості мов програмування, наприклад, у C++, для роботи з цими потоками використовуються об'єкти класів std::ifstream (вхідні файли), std::ofstream (вихідні файли), std::cin (стандартний ввід, наприклад, з клавіатури) та std::cout (стандартний вивід, наприклад, на екран).

Щодо математичних операцій:

Додавання (+) - об'єднує два числа, результатом є їхня сума. Наприклад:

int sum = 3 + 5; // sum = 8

Віднімання (-) - віднімає одне число від іншого, результатом є їхня різниця. Наприклад:

int difference = 10 - 4; // difference = 6

Множення (\*) - перемножує два числа, результатом є їхній добуток. Наприклад:

int product = 6 \* 7; // product = 42

Ділення (/) - ділить одне число на інше, результатом є їхня частка.

Наприклад:

double quotient = 20.0 / 5.0; // quotient = 4.0

Оператор залишку (%) - повертає залишок від ділення одного числа на інше (цілочисельне ділення).

Наприклад:

int remainder = 17 % 4; // remainder = 1

### 3.1.4 Програмування лінійних та розгалужених обчислювальних процесів

Програмування лінійних та розгалужених обчислювальних процесів є ключовими аспектами при розробці програмного забезпечення. Давайте детальніше розглянемо кожен з них.

Лінійні обчислювальні процеси Лінійні обчислювальні процеси характеризуються послідовністю виконання команд або інструкцій у програмі, що виконуються одна за одною у тому порядку, в якому вони були написані. Такі процеси працюють без розгалужень або умовних переходів і виконуються прямолінійно.

Приклад лінійного обчислювального процесу в C++

#include <iostream>

int main() {

// Лінійний обчислювальний процес

// Введення даних

int a, b;

std::cout << "Введіть перше число: ";

std::cin >> a;

std::cout << "Введіть друге число: ";

std::cin >> b;

// Обчислення

int sum = a + b;

// Виведення результату

std::cout << "Сума чисел " << a << " і " << b << " дорівнює " << sum << std::endl;

return 0;

}

Розгалужені обчислювальні процеси Розгалужені обчислювальні процеси включають умовні оператори, що дозволяють програмі змінювати свою поведінку в залежності від умови, яка перевіряється під час виконання. Це дозволяє програмі реагувати на різні умови або вхідні дані.

Приклад розгалуженого обчислювального процесу в C++

#include <iostream>

int main() {

// Розгалужений обчислювальний процес

// Введення даних

int number;

std::cout << "Введіть число: ";

std::cin >> number;

// Перевірка умови

if (number > 0) {

std::cout << "Число " << number << " є додатнім." << std::endl;

} else if (number < 0) {

std::cout << "Число " << number << " є від'ємним." << std::endl;

} else {

std::cout << "Число " << number << " є нулем." << std::endl;

}

return 0;

}

### 3.1.5 Програмування циклічних обчислювальних процесів

Програмування циклічних обчислювальних процесів є важливою частиною створення програмного забезпечення. Цикли дозволяють виконувати певну команду або блок коду кілька разів поспіль, автоматизуючи ітераційні завдання або операції, що потребують повторного виконання.

Основні види циклів

Багато мов програмування мають різні типи циклів, які відрізняються за умовами виходу та способом виконання:

Цикл while - це цикл з передумовою:

* + Він виконується, доки задана умова є true.
  + Умова перевіряється перед кожним виконанням тіла циклу

int i = 0;

while (i < 5) {

std::cout << i << " ";

i++;

}

// Результат: 0 1 2 3 4

Цикл do-while - це цикл з післяумовою:

* Він виконується принаймні один раз, а потім перевіряє умову.
* Тіло циклу виконується, поки умова є true.

int i = 0;

do {

std::cout << i << " ";

i++;

} while (i < 5);

// Результат: 0 1 2 3 4

Цикл for - це цикл з контрольованою послідовністю:

* Він виконується певну кількість разів або до досягнення певної умови.
* Цей цикл складається з ініціалізації, умови та кроку ітерації

for (int i = 0; i < 5; i++) {

std::cout << i << " ";

}

// Результат: 0 1 2 3 4

## **3.2 Практичне завдання**

### 3.2.1 Завдання

### 3.2.2 Постановка задачі

Вхідні дані

Вихідні дані

Блок-схема алгоритму роботи програми представлена на рис. ххх

Блок-схема

Рисунок хх - Блок-схема алгоритму роботи програми

### 3.2.3 Код програми

### 3.2.4 Скріншот виконання програми

Скріншот виконання програми представлена на рис. ххх

Скріншот

Рисунок хх - Скріншот виконання програми

## **3.3 Виконання тестів**

# 4 РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++"

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

У програмуванні під оператором зазвичай мають на увазі окрему інструкцію, яка вказує комп’ютеру, що робити. Оператори можуть бути простими (одна інструкція) або складеними (комбіновані інструкції).

Складені оператори утворюються за допомогою ключових слів begin..end в Pascal, фігурних дужок { } у Java, C/C++. Усі оператори, розміщені між цими елементами, розглядаються як одна інструкція.

До основних операторів у простих програмах (консольних) належать оператори введення та виведення. У C++ можна використовувати бібліотеку stdio, але частіше застосовується потокове введення/виведення з бібліотеки iostream.

#include <iostream>

Ця бібліотека містить простір імен std, який включає оператор cout (console out). Синтаксис цього оператора виглядає так:

std::cout << operand1 << operand2 << operand3 << ...;

Кожен елемент, який потрібно вивести на екран, записується через << (дві кутні дужки вліво/два знака менше). Наприклад:

cout << “Slava “ << 31;

cout << x << y << z;

За замовчуванням пробіли між елементами не ставляться, тому їх потрібно додавати вручну:

cout << x << “ “ << y << “ “ << z;

Особливості виведення: Мови Java, C/C++ підтримують спеціальні керуючі послідовності символів у рядках, які використовуються при виведенні. Усі такі послідовності починаються із символа \ (backslash):

* \n – новий рядок
* \t – табуляція

Оскільки деякі символи в мові програмування використовуються для позначення певних елементів і розпізнаються як частина мови, для їх виведення використовують додаткові керуючі послідовності:

* \\ – символ \
* \" – символ "
* \' – символ '
* \% – символ %
* \? – символ ?

Введення: Якщо дані, які потрібно опрацювати, заздалегідь не відомі або можуть мати різні значення, для подальшої роботи з ними необхідно отримати ці значення. Програма може отримувати дані з різних джерел, одне з них – клавіатура. Для введення даних з клавіатури у консольному режимі використовуються стандартні оператори введення.

У мові C++ також є можливість використання форматного введення scanf, але частіше використовується потокове введення з бібліотеки iostream. Функція cin має свій особливий синтаксис. Кожен наступний елемент введення записується через оператор >> (дві кутні дужки вправо/два знака більше). Імена змінних вводяться без операції розіменування (тобто без знака &).

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() {

std::string str;

int age;

std::cout << "Enter your family name: ";

std::cin >> str;

std::cout << "Enter your age: ";

std::cin >> age;

std::cout << "Mr. " << str << ", " << age << " years old." << std::endl;

std::cout << "Enter x, y, z: ";

float x, y, z;

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

return 0;

}

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

Масив – це група значень, які розташовані в пам'яті під однією назвою (ідентифікатором) та мають різні порядкові номери (індекси). Це поняття в програмуванні відповідає математичним поняттям послідовності та таблиці (матриці). Елемент масиву – це одна з величин, що складають масив. Це відповідає математичному поняттю елемента послідовності або матриці. Індекс масиву – це значення цілого типу, яке вказує на конкретний елемент масиву. Це відповідає математичному поняттю номера елемента послідовності або номера рядка/стовпчика таблиці (матриці). У мові C++ найменше значення індексу – 0.

Масив має такі властивості:

* тип(и) елементів масиву;
* назва (ідентифікатор) – однакова для всіх елементів масиву;
* розмірність – кількість індексів, необхідних для визначення розташування елемента в масиві;
* діапазони зміни індексів.

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

Контейнер list у мові C++ задає двонаправлений список. У такі списки можна швидко вставляти та видаляти елементи. Доступ до елементів списку здійснюється за допомогою вказівників (ітераторів).

#include <list> підключення 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() перевірка, чи 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) вставити елемент e в позицію i списку l

l.erase(j, k) видалити зі списку l послідовність елементів від позиції j до позиції k

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

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

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

Формати файлів і методи їх обробки є важливими аспектами роботи з інформацією в різних галузях. Нижче наведено деякі з найбільш поширених форматів файлів і способи їх обробки:

### Текстові файли

1. **TXT**: простий текстовий файл без форматування, використовується для зберігання необробленого тексту.
   * **Обробка**: редагування в 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 (бібліотека pandas), R тощо.

**Приклад:**

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

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

**Приклад:**

{

"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

}

]

}

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

**Приклад:**

<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>

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

### Використання форматів файлів на практиці

#### JSON

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

#### XML

Для роботи з XML файлами можна використовувати бібліотеку **pugixml**. Це легка і потужна бібліотека, яка підтримує повний набір функцій для роботи з XML документами, включаючи парсинг, модифікацію та серіалізацію.

### Робота з файлами в C++

Для роботи з файлами формату XML, CSV та JSON у C++ можна використовувати різні бібліотеки. Для читання та запису JSON файлів часто використовують бібліотеку nlohmann/json.

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

Системи контролю версій (VCS) — це програмні інструменти, що 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 та 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.

### Основні поняття Git

* **Гілка (Branch):** Це окрема версія проекту, яка існує незалежно від інших версій проекту. Гілки дозволяють розробникам працювати над різними завданнями або функціями одночасно, не заважаючи один одному. 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):** 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.

#### Основні гілки

* **Main (або Master):** Ця гілка завжди містить стабільну версію коду, готову до випуску. 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.

#### Допоміжні гілки

* **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.

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

### Використання Markdown для опису проекту

### Заголовки

## 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:**
  1. Пункт 1
  2. Пункт 2
  3. Пункт 3

#### Посилання та зображення

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

#### Код

* **Інлайн-код:** інлайн-код

#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

## 

## **4.7 Розроблення ігрової програми**

### 4.7.1 Код програми

#include <iostream>

using namespace std;

void drawBoard(char board[3][3])

{

cout << "-------------\n";

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

cout << "| ";

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

cout << board[i][j] << " | ";

}

cout << "\n-------------\n";

}

}

bool checkWin(char board[3][3], char player)

{

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

if (board[i][0] == player && board[i][1] == player

&& board[i][2] == player)

return true;

if (board[0][i] == player && board[1][i] == player

&& board[2][i] == player)

return true;

}

if (board[0][0] == player && board[1][1] == player

&& board[2][2] == player)

return true;

if (board[0][2] == player && board[1][1] == player

&& board[2][0] == player)

return true;

return false;

}

int main()

{

char board[3][3] = { { ' ', ' ', ' ' },

{ ' ', ' ', ' ' },

{ ' ', ' ', ' ' } };

char player = 'X';

int row, col;

int turn;

cout << "Welcome to Tic-Tac-Toe!\n";

for (turn = 0; turn < 9; turn++) {

drawBoard(board);

while (true) {

cout << "Player " << player

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

cin >> row >> col;

if (board[row][col] != ' ' || row < 0 || row > 2

|| col < 0 || col > 2) {

cout << "Invalid move. Try again.\n";

}

else {

break;

}

}

board[row][col] = player;

if (checkWin(board, player)) {

drawBoard(board);

cout << "Player " << player << " wins!\n";

break;

}

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

}

drawBoard(board);

if (turn == 9 && !checkWin(board, 'X')

&& !checkWin(board, 'O')) {

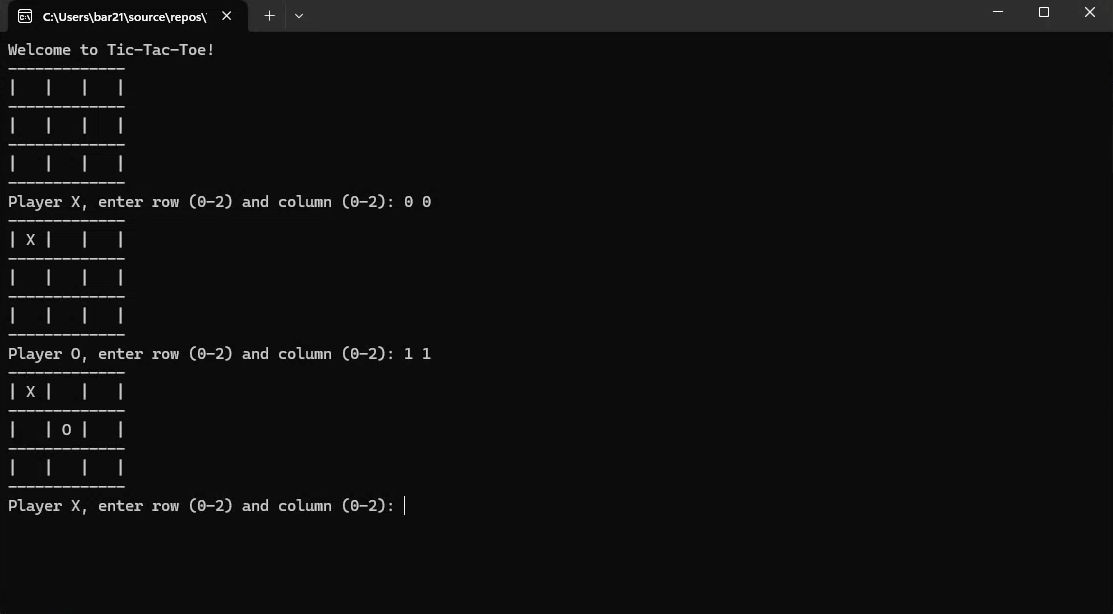
cout << "It's a draw!\n";

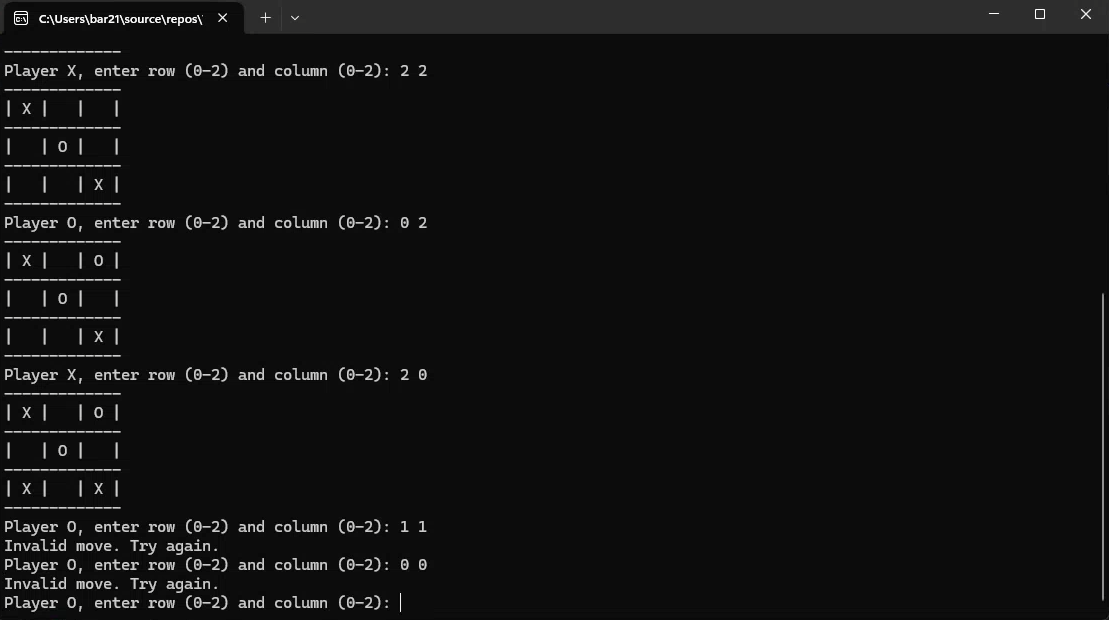
}

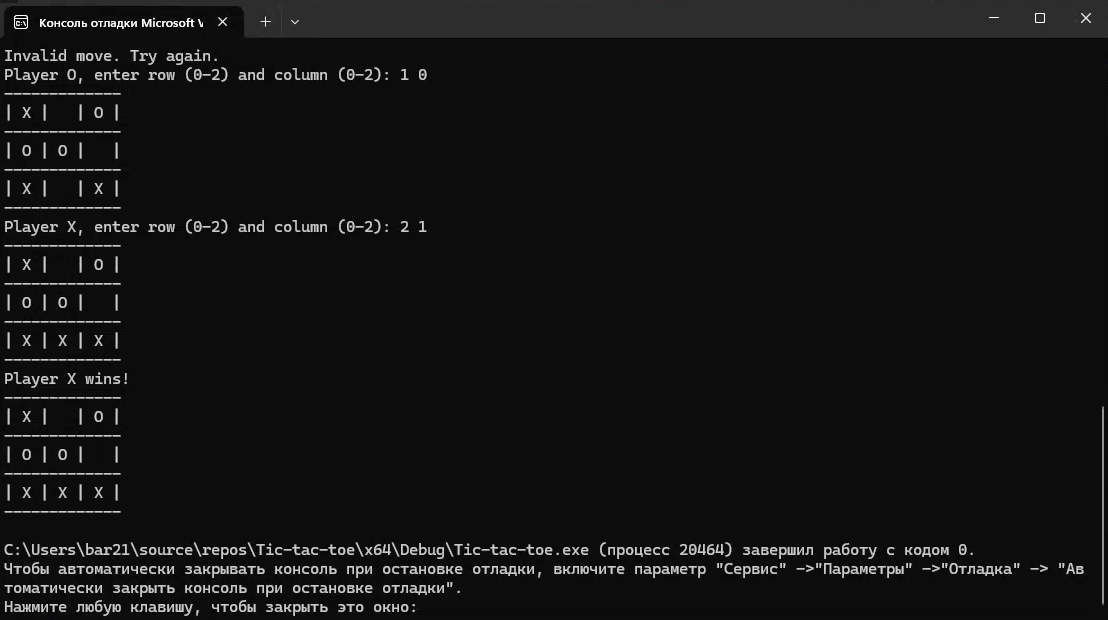
return 0;

}

### 4.7.2 Скріншоти виконання програми







# 5 РОЗДІЛ ”ВЕБ-ДИЗАЙН ТА HTML”

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

### Теги та їх використання в HTML

Тег - це інструкція для браузера про те, як слід відображати певну частину документа. Тег починається з символу "<" і завершується символом ">". Між цими символами розташовується назва тега, причому малі та великі літери не розрізняються.

Існують два види тегів: одинарні та парні.

* **Одинарні теги** використовуються самостійно. Наприклад, тег <br> означає перенесення тексту на новий рядок, як у записі:

<br> Зошит

Тут слово "Зошит" відображатиметься з нового рядка.

 **Парні теги** мають початкову та завершальну частини. Початкова частина вказує, де починається дія, а завершальна – де вона закінчується. Завершальний тег відрізняється наявністю символу «/» перед назвою тега. Наприклад,

<b> Синій олівець </b>

означає, що текст "Синій олівець" буде відображатися напівжирним шрифтом.

### Етапи розроблення веб-сторінки

Розробка веб-сторінки складається з двох основних етапів:

1. **Створення макета**: визначення змісту та зовнішнього вигляду веб-сторінки.
2. **Вставка тегів HTML**: забезпечення потрібного вигляду сторінки при перегляді у браузері.

### Основи HTML-документа

HTML-документ визначається парним тегом <HTML>...</HTML>, який обрамлює весь документ:

<HTML>

текст документа

</HTML>

HTML-документ поділяється на дві частини:

* **Заголовок**: позначається тегами <HEAD>...</HEAD>.
* **Тіло**: позначається тегами <BODY>...</BODY>.

Текст, що відображається у заголовку вікна браузера, розміщується у тегах <TITLE>...</TITLE>:

<HTML>

<HEAD>

<TITLE> Заголовок документа </TITLE>

</HEAD>

<BODY>

Тіло документа

</BODY>

</HTML>

### Коментарі

Коментарі в HTML-документі використовуються для пояснень і не відображаються у браузері. Вони розміщуються між тегами <!-- ... -->.

### Форматування тексту

* Тег <H1> Текст... </H1> створює заголовок першого рівня.
* Абзаци та заголовки можуть вирівнюватися за центром, лівим або правим краєм за допомогою параметра ALIGN:
  + ALIGN="CENTER": по центру
  + ALIGN="LEFT": по лівому краю
  + ALIGN="RIGHT": по правому краю

### Збереження пробілів

Якщо в HTML-документі існують додаткові пробіли, браузер може їх ігнорувати. Щоб зберегти пробіли, використовуйте тег <pre>...</pre>, який означає попередньо відформатований текст і зберігає всі пробіли та символи нового рядка. Цей тег часто використовується для відображення тексту у вигляді таблиці.

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

### Теги для логічної розмітки тексту

Теги в HTML визначають, як браузер повинен відображати частини документа. Теги розміщуються всередині тегу <body>.

#### Тег <p>

Тег <p> використовується для створення абзаців. Абзаци починаються з нового рядка і мають вертикальні відступи, які можна змінювати за допомогою стилів.

#### Заголовки та підзаголовки

Для структуризації великих текстів використовуються заголовки різних рівнів. HTML надає шість рівнів заголовків, від <h1> до <h6>, де <h1> є найбільш важливим заголовком, а <h6> – найменш важливим. Наприклад:

<h1> Заголовок першого рівня </h1>

<h2> Заголовок другого рівня </h2>

Зазвичай використовуються заголовки перших трьох рівнів.

#### Ненумеровані списки

Ненумеровані списки створюються за допомогою тегу <ul>, а кожен елемент списку позначається тегом <li>:

<ul>

<li>Перший пункт списку</li>

<li>Другий пункт списку</li>

<li>Третій пункт списку</li>

</ul>

#### Додавання зображень

Для додавання зображень використовується одиночний тег <img>. Адреса зображення вказується в атрибуті src:

<img src="logo.png">

Найпоширенішими форматами зображень в інтернеті є JPEG і PNG.

#### Розміри зображень

Щоб задати розміри зображень, використовуйте атрибути width і height:

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

У прикладі зображенню задана ширина 100 пікселів. Якщо потрібно задати відносну ширину, можна використовувати відсотки:

<img width="50%" src="logo.png">

#### Альтернативний текст

Альтернативний текст задається за допомогою атрибуту alt. Він відображається, якщо зображення не може бути завантажене:

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

#### Зображення з посиланням

Зображення можуть бути посиланнями, якщо їх обгорнути в тег <a>:

<a href="http://keksby.ru">

<img src="cat.png" alt="Кіт">

</a>

## **5.3 Каскадні таблиці стилів CSS та їх структура**

CSS каскадні таблиці стилів визначають набір правил форматування, які застосовуються до елементів веб-сторінки для управління їх виглядом і розташуванням. Є три способи застосування CSS до HTML елементів:

* Вбудований (Inline) - за допомогою атрибуту style в елементах HTML.
* Внутрішній (Internal) - за допомогою елемента <style> у розділі <head>.
* Зовнішній (External) - через зовнішній файл CSS.

Властивість background в CSS є скороченням для набору фонових властивостей. Наприклад, властивість background-color встановлює колір фону для будь-якого елемента в CSS і приймає будь-яке дійсне значення <color>. background-color застосовується до області вмісту та заповнення елемента. Фонові зображення можна додавати через властивість background-image, яка дозволяє розміщувати зображення на фоні елемента.

Для контролю поведінки повторення фонових зображень використовується властивість background-repeat. Ця властивість має такі значення:

* no-repeat — зупиняє повторення фону.
* repeat-x — повторює горизонтально.
* repeat-y — повторює вертикально.
* repeat — за замовчуванням повторює в обох напрямках.
* space — повторює зображення з пробілом між ними.
* round — розтягує зображення для заповнення простору.

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

Наслідування У CSS наслідування визначає, як властивості стилів передаються від батьківських елементів до дочірніх. Деякі властивості автоматично спадковуються дочірніми елементами, такі як стиль шрифту та колір тексту.

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

body {

color: blue;

}

p {

/\* Абзаци у спадковості отримують синій колір від body \*/

}

Каскадування Каскадування визначає, як CSS вирішує конфлікти між різними правилами стилів. При конфлікті між кількома правилами для одного елемента, використовується правило з найбільшою специфічністю.

Специфічність визначається так:

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

Приклад: Правила з більшою специфічністю перекривають правила з меншою специфічністю.

#header {

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

}

.header {

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

}

header {

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

}

Використання важливості (!important) Для перекриття всіх інших правил можна використовувати декларацію !important. Це змушує браузер застосовувати це правило незалежно від специфічності інших правил.

Приклад:

p {

color: green !important;

}

Порядок підключення стилів Порядок підключення стилів також впливає на використання правил. Правила, що з'являються пізніше в CSS файлі або в документах HTML, мають вищий пріоритет.

Приклад:

<head>

<style>

p {

color: red;

}

</style>

<style>

p {

color: blue;

}

</style>

</head>

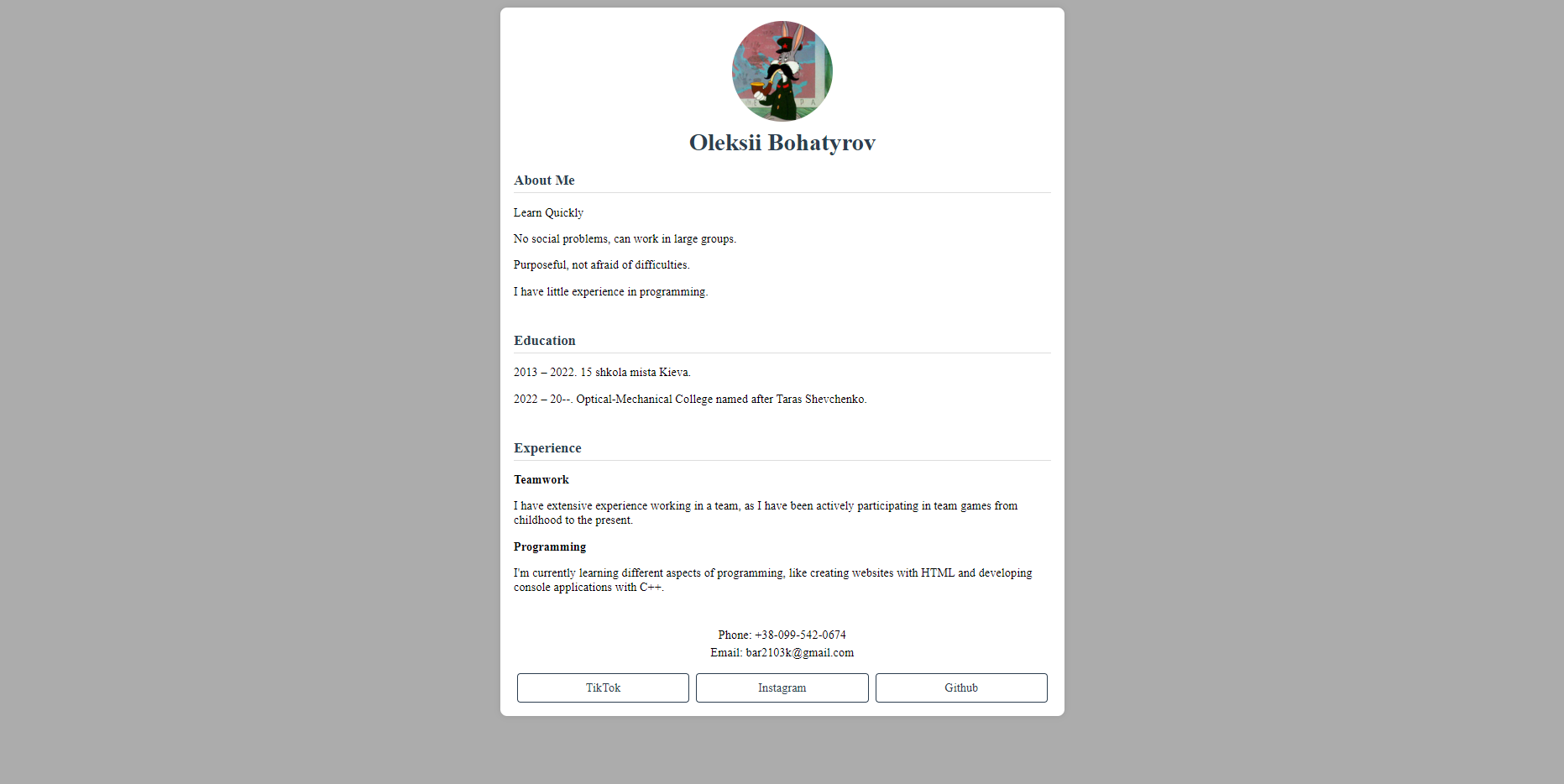
У цьому прикладі абзаци будуть синіми, оскільки друге правило з'являється пізніше.

## **5.5 Практична частина**

### 5.5.1 Постановки задачі

Основна мета полягала в створенні веб-сайту у формі резюме з метою вивчення HTML, CSS та принципів каскадного оформлення стилів. У моєму проєкті я використав різноманітні стилізаційні елементи, такі як зміна кольору, розміру шрифту, стилю тексту та інші параметри для покращення зручності читання. Крім того, на сайті є фотографії, активні посилання для легкого переходу між розділами, а також елементи дизайну, що підвищують його зрозумілість для відвідувачів.

### 5.5.2 HTML документ



# ВИСНОВКИ

РОЗДІЛ "ІНФОРМАТИКА При роботі з MS Word для створення складних документів важливо грамотно структурувати інформацію та використовувати засоби форматування, таблиці, списки та інші елементи для зручності читання та естетичного оформлення документа. Цей інструмент є незамінним для створення різноманітних документів, таких як звіти, презентації, наукові роботи та інша документація.

Співпраця MS Word з MS Excel є важливою для розв'язання завдань, що вимагають аналізу та обробки даних. MS Excel дозволяє створювати складні таблиці, виконувати розрахунки, будувати графіки та діаграми, що допомагає в ілюстрації результатів досліджень, фінансових моделей, статистичних аналізів та інших областей.

РОЗДІЛ "ПРОГРАМУВАННЯ С++ У процесі виконання практичних завдань я отримав досвід розв'язання реальних проблем, використовуючи можливості мови C++ для ефективної обробки даних, взаємодії з користувачем та створення модульних програмних компонентів. Використання різноманітних бібліотек і стандартних функцій дозволяє розширювати можливості програм та підвищувати їх функціональність.

РОЗДІЛ "ПРАКТИЧНА РОБОТА З С++ Під час розробки консольної програми на мові C++ я отримав поглиблене розуміння основних концепцій програмування, таких як структури даних, умовні конструкції, цикли, функції та робота з масивами. Ми також навчились створювати модульні програмні компоненти, які взаємодіють між собою для досягнення певних функціональних цілей.

РОЗДІЛ "ВЕБ-ДИЗАЙН ТА HTML Під час практики з розробки HTML документу я отримав важливий досвід у створенні структурованих веб-сторінок. Освоївши основні елементи HTML і їх властивості, я здобув знання, як правильно організувати та відформатувати контент для досягнення оптимальної читабельності та користувацької зручності. Також вивчення основ CSS дозволило застосовувати каскадні стилі для створення привабливого візуального оформлення сторінки, змінюючи колір, розмір шрифту, розташування елементів та інші параметри. Цей досвід є важливим для мого професійного зростання в сфері 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. Введення та виведення:

<https://vvasilchenko.wordpress.com/2016/12/13/prog-lesson-4-io/>

4. Структури даних для управління станом програми: <https://www.kievoit.ippo.kubg.edu.ua/kievoit/2016/65_C++/index.html>

5. Базовий файловий ввід і вивід:

<https://acode.com.ua/urok-220-bazovyj-fajlovyj-vvid-i-vyvid/>

Системи керування версіями:

<https://cherto4ka.xyz/2021/11/03/%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B8-%D0%BA%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F-%D0%B2%D0%B5%D1%80%D1%81%D1%96%D1%8F%D0%BC%D0%B8/>

3. HTML:

<https://ru.wikipedia.org/wiki/HTML>

4.Урок:Структура веб-сторінок. Поняття про мову HTML.  
[https://vseosvita.ua/lesson/struktura-veb-storinok-poniattia-pro-movu-html-71025.html\](https://vseosvita.ua/lesson/struktura-veb-storinok-poniattia-pro-movu-html-71025.html/)

5. Вставляння малюнків і гіперпосилань : <https://www.miyklas.com.ua/p/informatica/8-klas/stvorennia-ta-publikatciia-veb-resursiv-444304/poniattia-pro-movu-gipertekstovoyi-rozmitki-444805/re-9a1035c8-5e25-4eab-ae75-c249d3d7c592>

## 6.Стилізація HTML за допомогою CSS. Каскадні таблиці стилів: <https://w3schoolsua.github.io/html/html_css.html#gsc.tab=0>

## 7. Наслідування та каскадування. Оформлення тексту за допомогою CSS: <http://e-helper.com.ua/node/952>

# ДОДАТКИ