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-22 |
| Прокопенко Кирил Юрійович  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) | |  | |
| *Прокопенко Кирил Юрійович* | | | | |
| (П.І.Б студента.) | | | | |
| Спеціальність | *121 Інженерія програмного забезпечення* | Група | | *ІПЗ-22* |

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

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

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

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

4

Розроб.

Перевір.

Реценз.

В.Я

Н. Контр.

В.Я

Затверд.

Звіт з навчальної практики

Літ.

Аркушів

ІПЗ-22

ОМФК.0121 НП 14

**ЗМІСТ**

[ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 5](#_Toc167919156)

[ВСТУП 6](#_Toc167919157)

[1 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ 7](#_Toc167919158)

[2 РОЗДІЛ "ІНФОРМАТИКА" 8](#_Toc167919159)

[**2.1** **Розроблення засобами MS Word документа складної структури** 8](#_Toc167919160)

[**2.2 Розв‘язок задач за допомогою засобів табличного редактора MS Excel** 8](#_Toc167919161)

[**2.3 Виконання** **практичних завдань за допомогою засобів табличного редактора MS Access** 8](#_Toc167919162)

[3 РОЗДІЛ "ПРОГРАМУВАННЯ С++" 9](#_Toc167919163)

[**3.1 Теоретичні відомості** 9](#_Toc167919164)

[3.1.1 Загальні поняття. Елементи мови С++ - константи, змінні, операції, перетворення типів 9](#_Toc167919165)

[3.1.2 Арифметичні операції 9](#_Toc167919166)

[3.1.3 Поняття вхідного та вихідного потоку, найпростіші математичні функції 9](#_Toc167919167)

[3.1.4 Програмування лінійних та розгалужених обчислювальних процесів 9](#_Toc167919168)

[3.1.5 Програмування циклічних обчислювальних процесів 9](#_Toc167919169)

[**3.2 Практичне завдання** 10](#_Toc167919170)

[3.2.1 Завдання 10](#_Toc167919171)

[3.2.2 Постановка задачі 10](#_Toc167919172)

[3.2.3 Код програми 11](#_Toc167919173)

[3.2.4 Скріншот виконання програми 12](#_Toc167919174)

[**3.3 Виконання тестів** 13](#_Toc167919175)

[4 РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++" 13](#_Toc167919176)

[**4.1 Тема 1. Введення та виведення у базових консольних програмах** 13](#_Toc167919177)

[**4.2 Тема 2. Структури даних для управління станом програми** 13](#_Toc167919178)

[**4.3 Тема 3. Робота з файлами: запис та читання** 13](#_Toc167919179)

[**4.4 Тема 4. Формати файлів та їх обробка** 13](#_Toc167919180)

[**4..5 Тема 5. Системи контролю версій (VCS)** 14](#_Toc167919181)

[**4.6 Тема 6. Мова Markdown для опису проекті** 14](#_Toc167919182)

[**4.7 Розроблення ігрової програми** 14](#_Toc167919183)

[4.7.1 Код програми 14](#_Toc167919184)

[4.7.2 Скріншоти виконання програми 14](#_Toc167919185)

[5 РОЗДІЛ ”ВЕБ-ДИЗАЙН ТА HTML” 16](#_Toc167919186)

[**5.1 Мова розмітки HTML та структура Веб-сторінки** 16](#_Toc167919187)

[**5.2 Розмітка тексту за допомогою HTML. Зображення та посилання** 16](#_Toc167919188)

[**5.3 Каскадні таблиці стилів CSS та їх структура** 16](#_Toc167919189)

[**5.4 Спадковості і каскадування за допомогою CSS** 16](#_Toc167919190)

[**5.5 Практична частина** 17](#_Toc167919191)

[5.5.1 Постановки задачі 17](#_Toc167919192)

[5.5.2 HTML документ 17](#_Toc167919193)

[ВИСНОВКИ 18](#_Toc167919194)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ 19](#_Toc167919195)

[ДОДАТКИ 20](#_Toc167919196)

ОМФК.0121 НП 20

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

# ВСТУП

Автор:

Дата:

Мета:

Перелік вивчених тем:

# 1 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ

# 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

Мову C++ можна розглядати як надмножину мови С, бо вона зберігає усі можливості, що надає мова С, і доповнює їх засобами об’єктно-орієнтованого програмування. C++ є універсальною алгоритмічною мовою, яка використовується для розробки системних та складних прикладних програм.

Мова C++ є мовою високого рівня і основою багатьох систем програмування, наприклад **Visual Studio**, **Eclipse**. Найбільш популярною з них вважається **Visual Studio**.

**Алфавіт** мови C++ включає:

* великі (**A-Z**) і малі (**a—z**) літери латинського алфавіту та символ підкреслення ( **\_** );
* арабські цифри від **0** до **9**;
* знаки арифметичних дій **+, -, \*, /, %, ++, —**;
* знаки побітових операцій **<<, >>, &, |, ~, ^**;
* знаки відношень **<, <=, ==, !=, >, >=**;
* знаки логічних операцій **&&, ||, !**;
* розділові знаки **, ; : пропуск**;
* спеціальні знаки **., =, ->, ?, \, $, #, ', "**;
* символи дужок **(, ), [, ], {, }**.
* інших логічно завершених фрагментів програм.

Директива препроцесора **#include<iostream>** забезпечує підключення до програми засобів зв’язку зі стандартними потоками введення-виведення даних. Ці засоби знаходяться у заголовному файлі **iostream.h**, де **і (*input*)** — введення, **о (*output*)** виведення, **stream** — поток, **h (*head*)** — заголовок. Оскільки середовище **Borland C++** не забезпечує затримку результатів на екрані, у текстах прикладів програм тут і далі використано стандартний потік введення **сіn** (бібліотека **iostream**), функцію введення **getch()** (бібліотека **conio.h**) тощо.

**Основні типи даних** часто називають арифметичними, тому що їх можна використовувати в арифметичних операціях. Для опису основних типів мови C++ використовують такі службові слова:

* **int** (цілий);
* **char** (символьний);
* **bool** (логічний);
* **float** (дійсний);
* **double** (дійсний з подвійною точністю);
* **void** (порожній, не має значення).

### 3.1.2 Арифметичні операції

**«Л»** позначено величину, що стоїть ліворуч від знака операції, літерою **«П»** — величину, яка розташована праворуч від знака операції, Л > П напрямок виконання операції. Розглянемо основні операції.

***Арифметичні операції:***

**+** — додає величину **П** до **Л**;

* + — віднімає **П** із **Л**;
  + — унарна операція зміни знаку величини **П**;

**\***     — множення **П** і **Л**;

**/**  — ділення **Л** на **П**;

**%**  — залишок від ділення величини **Л** на величину **П** (для цілих чисел), наприклад, якщо **int g = 12;**, то операція **g = g % 9;** надасть результат: **g = 3;**

**++** — унарна операція інкремент. Якщо змінна розташовується праворуч від знаку операції (префіксна форма), то значення збільшується на 1 до використання. Якщо ж змінна знаходиться ліворуч від знаку операції (постфіксна форма), то її значення збільшується на 1 після використання, наприклад:

**int d;**

**++d;** — префіксний інкремент,

**d++;** — постфіксний інкремент;

-- — унарна операція декремент аналогічно інкременту має двi форми: префіксную (змінна розташована праворуч від знаку операцii) — зменшення значення змінної на 1 відбувається до її використання; постфіксну (змінна знаходиться ліворуч від знака операції) — зменшення значення змінної на 1 після її використання.

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

Вхідний потік (input stream) використовується для приймання даних у програму з різних джерел, таких як клавіатура, файли або мережа. У мові програмування C++, вхідний потік представлений об'єктом типу std::istream або його похідними типами, такими як std::cin. Зазвичай використовується оператор >> для зчитування даних з вхідного потоку.

Вихідний потік (output stream) використовується для виведення даних з програми на екран користувача або в інші медіа, такі як файли або мережа. У C++, вихідний потік представлений об'єктом типу std::ostream або його похідними типами, такими як std::cout. Для виведення даних використовується оператор <<.

Найпростіші математичні функції включають основні арифметичні операції, такі як додавання, віднімання, множення та ділення. Вони використовуються для обчислення числових значень та виконання математичних операцій в програмах. Більш складні математичні функції можуть бути вбудовані у мову програмування або доступні через певні бібліотеки.

### 3.1.4 Програмування лінійних та розгалужених обчислювальних процесів

Лінійні обчислювальні процеси: Лінійний обчислювальний процес - це послідовність інструкцій, які виконуються від початку до кінця без будь-яких умов або переривань. Програмування лінійних обчислювальних процесів досить просте - ви просто вказуєте послідовність дій, які потрібно виконати, кожна з яких виконується послідовно після попередньої.

Наприклад, у програмі для обчислення суми двох чисел спочатку зчитується перше число, потім зчитується друге число, а потім вони додаються разом.

Розгалужені обчислювальні процеси: Розгалужені обчислювальні процеси включають умови або розгалуження, що дозволяють програмі приймати різні шляхи дії в залежності від певних умов. Це робить програми більш гнучкими і здатними реагувати на різні ситуації.

У програмуванні розгалужені обчислювальні процеси реалізуються за допомогою умовних операторів, таких як if, else if та else в C++. Ці оператори дозволяють виконувати різні блоки коду в залежності від умови.

Наприклад, у програмі для визначення парності числа можна використати розгалужений обчислювальний процес: якщо число ділиться на 2 без остачі, то воно парне, інакше - непарне.

### 3.1.5 Програмування циклічних обчислювальних процесів

Циклічні обчислювальні процеси використовуються для повторення певних дій або блоків коду декілька разів. Вони є важливим засобом автоматизації повторюваних завдань в програмуванні.

У програмуванні цикли можуть бути реалізовані за допомогою таких конструкцій як:

Цикл for: Використовується для виконання певного блоку коду певне число разів, відомого заздалегідь.

Цикл while: Виконується, доки певна умова залишається істинною.

Цикл do-while: Схожий на цикл while, за винятком того, що тіло циклу виконується принаймні один раз, незалежно від умови.

## **3.2 Практичне завдання**

### 3.2.1 Завдання

**Завдання**: Обчислити і вивести на екран у табличному вигляді значення функції *f(x)* на заданому інтервалі зміни значень аргументу *х* від *xпоч* до *xкін* з кроком *h.*

Коефіцієнти *a, b, c –* дійсні числа. Значення *a, b, c, xпоч*, *xкін*, *h* вводити з клавіатури. Передбачити перевірку допустимості введених значень.

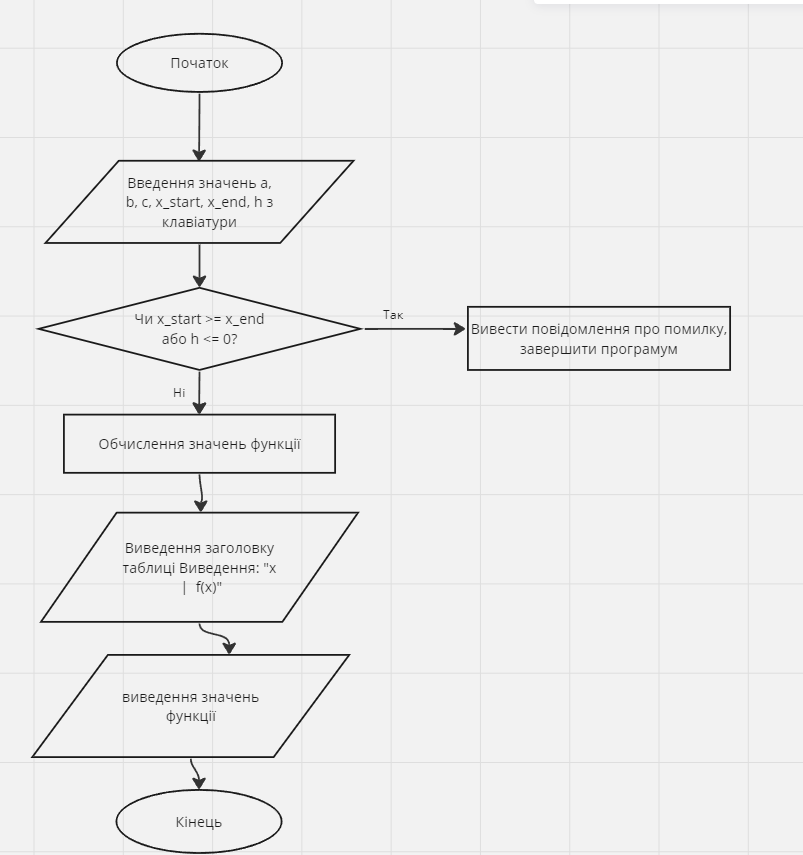
### 3.2.2 Постановка задачі

Вхідні дані

Вихідні дані

Блок-схема алгоритму роботи програми представлена на рис. ххх

Блок-схема



### 3.2.3 Код програми

#include <iostream>

#include <cmath>

using namespace std;

double calculateFunction(double x, double a, double b, double c) {

if (x > 0 && b < 0) {

return 3 \* x + a \* x / (x \* x) + 2 \* c;

} else if (x < 0 && b >= 0) {

return -a \* x \* x + b;

} else {

return 2 \* x / c - a - b \* x \* x;

}

}

int main() {

double a, b, c, x\_start, x\_end, h;

// Введення значень з клавіатури

cout << "Введіть значення a: ";

cin >> a;

cout << "Введіть значення b: ";

cin >> b;

cout << "Введіть значення c: ";

cin >> c;

cout << "Введіть початкове значення x: ";

cin >> x\_start;

cout << "Введіть кінцеве значення x: ";

cin >> x\_end;

cout << "Введіть крок h: ";

cin >> h;

// Перевірка допустимості введених значень

if (x\_start >= x\_end || h <= 0) {

cout << "Неправильно введені значення для інтервалу та кроку." << endl;

return 1;

}

// Виведення заголовку таблиці

cout << "x\t|\tf(x)" << endl;

cout << "-----------------" << endl;

// Обчислення та виведення значень функції на вказаному інтервалі

for (double x = x\_start; x <= x\_end; x += h) {

cout << x << "\t|\t" << calculateFunction(x, a, b, c) << endl;

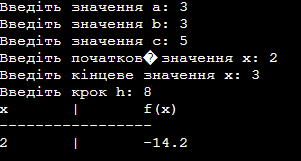
}

return 0;

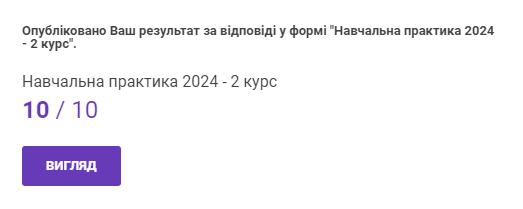
}

### 3.2.4 Скріншот виконання програми

Скріншот



## **3.3 Виконання тестів**

****

# 4 РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++"

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

### C++

#### Введення

Для введення даних використовується оператор cin.

Приклад

#include <iostream>

using namespace std;

int main() {

string name;

int age;

cout << "Enter your name: ";

cin >> name;

cout << "Enter your age: ";

cin >> age;

return 0;

}

#### Виведення

Для виведення даних використовується оператор cout.

Приклад

#include <iostream>

using namespace std;

int main() {

string name;

int age;

cout << "Enter your name: ";

cin >> name;

cout << "Enter your age: ";

cin >> age;

cout << "Hello, " << name << "! You are " << age << " years old." << endl;

return 0;

}

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

У мові програмування C++ для управління станом програми можна використовувати різні структури даних. декілька з них, які є найбільш поширеними і корисними.

### 1. Змінні (Variables)

Змінні використовуються для зберігання окремих значень, які можуть змінюватися під час виконання програми. Це можуть бути прості типи, такі як int, float, char, а також користувацькі типи даних.

### 2. Масиви (Arrays)

Масиви зберігають колекції елементів одного типу у фіксованому порядку. Розмір масиву визначається під час його створення і не може бути змінений.

### 3. Вектори (Vectors)

Вектори є динамічними масивами, які можуть змінювати свій розмір під час виконання програми. Це дозволяє більш гнучке управління колекціями даних.

### 4. Списки (Lists)

Списки є динамічними структурами даних, які дозволяють додавати і видаляти елементи з будь-якої позиції. В C++ стандартна бібліотека надає std::list для двозв’язних списків.

### 5. Стек (Stack)

Стек використовує принцип "останній прийшов - перший пішов" (LIFO). Це корисно для управління станом в рекурсії або для реалізації обмежених буферів.

### 6. Черга (Queue)

Черга використовує принцип "перший прийшов - перший пішов" (FIFO). Це корисно для управління послідовністю завдань або обробки подій.

### 7. Дек (Deque)

Дек (двостороння черга) дозволяє додавати і видаляти елементи з обох кінців. Це поєднує властивості стеку і черги.

### 8. Мапа (Map)

Мапи зберігають пари ключ-значення, дозволяючи швидкий доступ до значення за його ключем. Це корисно для зберігання і пошуку даних.

### 9. Множина (Set)

Множини зберігають унікальні елементи і використовуються для швидкого пошуку елементів або перевірки наявності.

### 10. Кортежі (Tuples)

Кортежі дозволяють зберігати набір значень різних типів. Це корисно для повернення кількох значень з функцій.

### 11. Пара (Pair)

Пара зберігає два пов’язані між собою значення. Це часто використовується в мапах і інших структурах даних.

Ці структури даних допомагають ефективно керувати станом програми, забезпечуючи зручні механізми для зберігання, доступу і маніпуляції даними.

4o

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

### Запис у файл

Для запису даних у файл використовується клас std::ofstream (output file stream).

#### Кроки для запису у файл:

1. Створити об'єкт класу std::ofstream.
2. Відкрити файл для запису за допомогою методу open() або конструктора.
3. Записати дані у файл за допомогою оператора <<.
4. Закрити файл за допомогою методу close().

### Читання з файлу

Для читання даних з файлу використовується клас std::ifstream (input file stream).

#### Кроки для читання з файлу:

1. Створити об'єкт класу std::ifstream.
2. Відкрити файл для читання за допомогою методу open() або конструктора.
3. Читати дані з файлу за допомогою оператора >> або методів getline() та read().
4. Закрити файл за допомогою методу close().

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

В C++ для обробки різних форматів файлів використовуються різні підходи та бібліотеки. Ось кілька поширених форматів файлів та методи їх обробки:

### Текстові файли (TXT)

Текстові файли містять звичайний текст і можуть бути легко оброблені за допомогою стандартних потокових класів std::ifstream та std::ofstream. Текстові файли читаються і записуються построково або посимвольно.

### CSV (Comma-Separated Values)

CSV файли використовуються для зберігання табличних даних, де значення розділяються комами. Для обробки CSV файлів часто використовуються ті ж самі потокові класи з додатковим парсингом рядків для розділення значень.

### XML (eXtensible Markup Language)

XML файли використовуються для зберігання структурованих даних. Для їх обробки можна використовувати бібліотеки, такі як TinyXML або RapidXML, які дозволяють парсити, створювати та змінювати XML документи.

### JSON (JavaScript Object Notation)

JSON файли також використовуються для зберігання структурованих даних. Для обробки JSON файлів можна використовувати бібліотеки, такі як JSON for Modern C++ (nlohmann/json), яка надає зручні засоби для парсингу та генерації JSON даних.

### Бінарні файли (Binary Files)

Бінарні файли містять дані в двійковому форматі, що дозволяє зберігати складні структури даних у більш компактному вигляді. Для обробки бінарних файлів використовуються потокові класи std::ifstream та std::ofstream у бінарному режимі з прапорцем ios::binary.

### INI файли (Initialization Files)

INI файли використовуються для зберігання конфігураційних налаштувань. Вони мають просту структуру з розділами та ключ-значення парами. Для обробки INI файлів можна використовувати сторонні бібліотеки, такі як INIH.

### YAML (YAML Ain't Markup Language)

YAML файли використовуються для зберігання конфігураційних даних у зручному для читання форматі. Для обробки YAML файлів у C++ можна використовувати бібліотеки, такі як yaml-cpp.

### ZIP файли

ZIP файли використовуються для стиснення та архівації даних. Для роботи з ZIP файлами можна використовувати бібліотеки, такі як zlib або libzip.

### Підходи до обробки файлів:

* **Читання та запис:** Використання потокових класів (std::ifstream, std::ofstream, std::fstream) для базових операцій з файлами.
* **Парсинг:** Використання регулярних виразів або спеціалізованих парсерів для розбору структурованих форматів, таких як CSV, XML, JSON.
* **Бібліотеки:** Застосування сторонніх бібліотек для обробки специфічних форматів файлів, що значно спрощує роботу з ними.

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

Системи контролю версій VCS є важливими інструментами в розробці програмного забезпечення, які дозволяють відстежувати та керувати змінами в коді. Вони надають можливість розробникам працювати разом над одним проектом, зберігати історію змін та повертатися до попередніх версій. Ось огляд основних концепцій і популярних систем контролю версій:

### Основні концепції VCS

1. **Репозиторій (Repository)**
   * Центральне місце для зберігання всього коду та історії змін.
   * Може бути локальним (на комп'ютері розробника) або віддаленим (на сервері).
2. **Коміт (Commit)**
   * Одиниця зміни в VCS, що зберігає знімок файлів в певний момент часу.
   * Кожен коміт містить метадані, такі як автор, дата та повідомлення коміту.
3. **Гілка (Branch)**
   * Відгалуження від основної лінії розробки, дозволяє працювати над окремими функціями або виправленнями незалежно від основного коду.
   * Може бути злитою (merged) назад в основну гілку.
4. **Злиття (Merge)**
   * Процес об'єднання змін з однієї гілки в іншу.
   * Може викликати конфлікти, якщо зміни стосуються одних і тих самих рядків коду.
5. **Відмітка (Tag)**
   * Статична мітка на певний коміт, зазвичай використовується для маркування релізів.

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

Markdown – це легка мова розмітки, яка дозволяє форматувати текст за допомогою простих текстових символів. Вона часто використовується для написання документації, README файлів, блогів та інших текстових документів у програмуванні. Ось основні елементи синтаксису Markdown:

### Заголовки

Заголовки створюються за допомогою символу #. Чим більше #, тим нижчий рівень заголовка.

# Заголовок першого рівня

## Заголовок другого рівня

### Заголовок третього рівня

#### Заголовок четвертого рівня

##### Заголовок п'ятого рівня

###### Заголовок шостого рівня

### Абзаци

Абзаци створюються простим введенням тексту з пропуском між рядками.

Це перший абзац.

Це другий абзац.

### Виділення тексту

Виділення тексту виконується за допомогою зірочок або підкреслень.

\*Курсив\* або \_Курсив\_

\*\*Жирний\*\* або \_\_Жирний\_\_

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

### Списки

Списки можуть бути нумерованими або маркованими.

#### Марковані списки

- Пункт 1

- Пункт 2

- Підпункт 2.1

- Підпункт 2.2

- Пункт 3

#### Нумеровані списки

1. Пункт 1

2. Пункт 2

1. Підпункт 2.1

2. Підпункт 2.2

3. Пункт 3

### Посилання

Посилання створюються за допомогою квадратних та круглих дужок.

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

[Текст посилання з підказкою](https://example.com "Підказка")

### Зображення

Зображення вбудовуються схожим чином до посилань, але з символом ! на початку.

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

![Альтернативний текст з підказкою](https://example.com/image.jpg "Підказка")

### Код

#### Однорядковий код

Код в рядку виділяється символами `.

Це `код`.

## **4.7 Розроблення ігрової програми**

### 4.7.1 Код програми

#include <iostream>

#include <vector>

#include <limits>

using namespace std;

const char PLAYER = 'X';

const char AI = 'O';

const char EMPTY = ' ';

void printBoard(const vector<vector<char>>& board) {

    cout << "  0 1 2\n";

    for (int i = 0; i < 3; i++) {

        cout << i << " ";

        for (int j = 0; j < 3; j++) {

            cout << board[i][j];

            if (j < 2) cout << "|";

        }

        cout << endl;

        if (i < 2) cout << "  -----\n";

    }

}

bool isMovesLeft(const vector<vector<char>>& board) {

    for (int i = 0; i < 3; i++)

        for (int j = 0; j < 3; j++)

            if (board[i][j] == EMPTY)

                return true;

    return false;

}

int evaluate(const vector<vector<char>>& board) {

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

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

            if (board[row][0] == AI)

                return +10;

            else if (board[row][0] == PLAYER)

                return -10;

        }

    }

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

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

            if (board[0][col] == AI)

                return +10;

            else if (board[0][col] == PLAYER)

                return -10;

        }

    }

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

        if (board[0][0] == AI)

            return +10;

        else if (board[0][0] == PLAYER)

            return -10;

    }

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

        if (board[0][2] == AI)

            return +10;

        else if (board[0][2] == PLAYER)

            return -10;

    }

    return 0;

}

int minimax(vector<vector<char>>& board, int depth, bool isMax) {

    int score = evaluate(board);

    if (score == 10)

        return score - depth;

    if (score == -10)

        return score + depth;

    if (!isMovesLeft(board))

        return 0;

    if (isMax) {

        int best = numeric\_limits<int>::min();

        for (int i = 0; i < 3; i++) {

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

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

                    board[i][j] = AI;

                    best = max(best, minimax(board, depth + 1, !isMax));

                    board[i][j] = EMPTY;

                }

            }

        }

        return best;

    } else {

        int best = numeric\_limits<int>::max();

        for (int i = 0; i < 3; i++) {

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

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

                    board[i][j] = PLAYER;

                    best = min(best, minimax(board, depth + 1, !isMax));

                    board[i][j] = EMPTY;

                }

            }

        }

        return best;

    }

}

pair<int, int> findBestMove(vector<vector<char>>& board) {

    int bestVal = numeric\_limits<int>::min();

    pair<int, int> bestMove = {-1, -1};

    for (int i = 0; i < 3; i++) {

        for (int j = 0; j < 3; j++) {

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

                board[i][j] = AI;

                int moveVal = minimax(board, 0, false);

                board[i][j] = EMPTY;

                if (moveVal > bestVal) {

                    bestMove.first = i;

                    bestMove.second = j;

                    bestVal = moveVal;

                }

            }

        }

    }

    return bestMove;

}

void playGame() {

    vector<vector<char>> board(3, vector<char>(3, EMPTY));

    while (true) {

        printBoard(board);

        int row, col;

        cout << "Enter your move (row and column): ";

        cin >> row >> col;

        if (board[row][col] != EMPTY) {

            cout << "Cell already occupied! Try again." << endl;

            continue;

        }

        board[row][col] = PLAYER;

        if (evaluate(board) == -10) {

            printBoard(board);

            cout << "You win!" << endl;

            break;

        }

        if (!isMovesLeft(board)) {

            printBoard(board);

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

            break;

        }

        auto aiMove = findBestMove(board);

        board[aiMove.first][aiMove.second] = AI;

        if (evaluate(board) == 10) {

            printBoard(board);

            cout << "AI wins!" << endl;

            break;

        }

        if (!isMovesLeft(board)) {

            printBoard(board);

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

            break;

        }

    }

}

int main() {

    char playAgain;

    do {

        playGame();

        cout << "Do you want to play again? (y/n): ";

        cin >> playAgain;

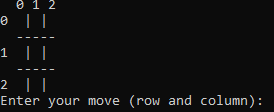
    } while (playAgain == 'y' || playAgain == 'Y');

    return 0;

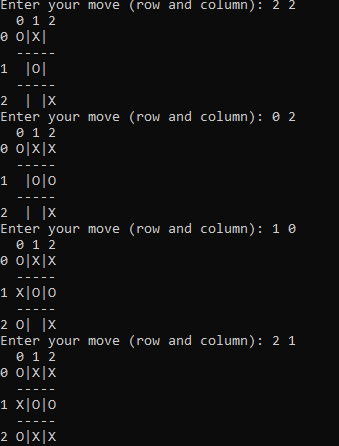
}

### 4.7.2 Скріншоти виконання програми

Початок гри



Середина гра



Кінець гри



# 5 РОЗДІЛ ”ВЕБ-ДИЗАЙН ТА HTML”

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

Мова розмітки HTML (Hypertext Markup Language) використовується для створення веб-сторінок. HTML визначає структуру документа та включає різноманітні елементи, які визначають різні частини сторінки, такі як заголовки, параграфи, зображення, посилання тощо.

Структура веб-сторінки в HTML зазвичай складається з таких основних елементів:

1. **DOCTYPE**: Це спеціальний елемент, який вказує на версію HTML, що використовується в документі.
2. **<html>**: Елемент <html> є кореневим елементом кожного HTML-документа і визначає початок та кінець HTML-коду.
3. **<head>**: В цьому елементі зазвичай розміщуються метатеги, які містять інформацію про документ, таку як заголовок сторінки, метатеги для SEO, посилання на зовнішні ресурси тощо.
4. **<title>**: Елемент <title> визначає заголовок документа, який відображається в рядку заголовка браузера.
5. **<body>**: Всі видимі елементи сторінки, такі як текст, зображення, посилання, розміщуються в елементі <body>.
6. **Елементи контенту**: Внутрішнім елементам <body> відповідає різноманітна контентна інформація, така як заголовки (<h1>, <h2>, тощо), параграфи (<p>), зображення (<img>), посилання (<a>), таблиці (<table>), списки (<ul>, <ol>) тощо.
7. **Коментарі**: Ви також можете вставляти коментарі у код HTML за допомогою <!-- коментар -->, щоб пояснити або залишити певні помітки.

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

 **Параграфи**: Використовують відокремлені абзаци для розділення параграфів тексту.

 **Заголовки**: Використовують знаки # перед текстом заголовка для визначення його рівня. Чим більше знаків #, тим нижчий рівень заголовка.

 **Напівжирний та курсивний текст**: Використовують символи \*\* для напівжирного тексту та \_ для курсивного тексту.

 **Зображення**: Використовують синтаксис ![альтернативний текст](шлях\_до\_зображення) для вставки зображення.

 **Посилання**: Використовують синтаксис [Текст посилання](адреса\_посилання) для створення посилань.

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

Каскадні таблиці стилів (CSS) - це мова стилів, що використовується для опису презентації документа, написаного мовою розмітки, такою як HTML. CSS дозволяє вам контролювати вигляд і форматування веб-сторінок, включаючи кольори, шрифти, макет і багато інших аспектів презентації.

### Основна структура CSS

1. **Правила CSS**: CSS складається з набору правил (правил стилів). Кожне правило складається з одного або декількох селекторів і блоку декларацій.
2. **Селектори**: Селектори визначають, до яких елементів HTML буде застосовано стилі. Існує кілька типів селекторів:
   * **Типові селектори**: вибирають елементи за їх тегом (наприклад, p, h1, div).
   * **Класові селектори**: вибирають елементи за класом (починаються з крапки, наприклад, .className).
   * **Ідентифікаційні селектори**: вибирають елементи за ідентифікатором (починаються з решітки, наприклад, #idName).
   * **Атрибутивні селектори**: вибирають елементи за атрибутом (наприклад, [type="text"]).
   * **Псевдокласи**: вибирають елементи в певному стані (наприклад, :hover, :focus).
   * **Псевдоелементи**: вибирають частини елементів (наприклад, ::before, ::after).
3. **Блок декларацій**: Блок декларацій містить одну або кілька декларацій, що описують стилі. Він починається і закінчується фігурними дужками { }.
4. **Декларації**: Декларація складається з властивості та значення, розділених двокрапкою. Кожна декларація закінчується крапкою з комою.

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

В CSS (Cascading Style Sheets) спадковість (inheritance) і каскадування (cascading) є двома основними принципами, які впливають на те, як стилі застосовуються до елементів HTML. Давайте розглянемо кожен з цих принципів детальніше.

### Спадковість (Inheritance)

Спадковість означає, що деякі властивості CSS передаються від батьківського елемента до дочірнього. Це означає, що якщо на батьківському елементі встановлено певну властивість, вона може бути автоматично застосована до всіх дочірніх елементів.

* color
* font-family
* font-size
* line-height
* text-align

### Каскадування (Cascading)

Каскадування означає, що коли для одного і того ж елемента застосовуються кілька стилів, браузер повинен визначити, який стиль має пріоритет. Пріоритет визначається за допомогою правил каскадування, які включають:

1. **Важливість (Importance)**: Стилі, позначені як !important, мають найвищий пріоритет.
2. **Специфічність (Specificity)**: Стилі з більш високою специфічністю мають пріоритет над менш специфічними.
3. **Порядок визначення (Source Order)**: Якщо стилі мають однакову специфічність, останній визначений стиль має пріоритет.

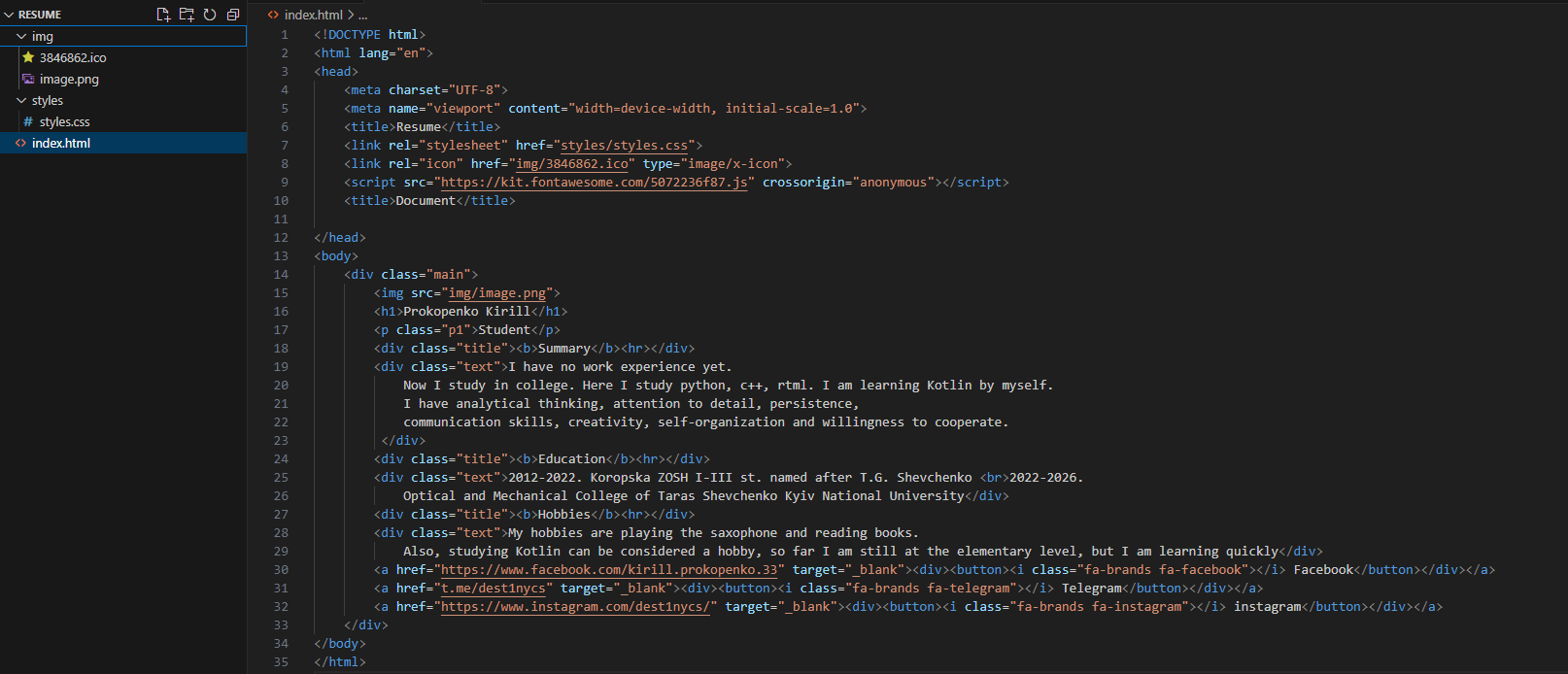
## **5.5 Практична частина**

### 5.5.1 Постановки задачі

Необхідно розробити веб-сторінку з власним резюме (CV) вказавши власні дані, фото, контакти. Веб-сторінка має відповідати прикладу за посиланням:

<https://t.ly/4ns0q>

### 5.5.2 HTML документ



<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Resume</title>

    <link rel="stylesheet" href="styles/styles.css">

    <link rel="icon" href="img/3846862.ico" type="image/x-icon">

    <script src="https://kit.fontawesome.com/5072236f87.js" crossorigin="anonymous"></script>

    <title>Document</title>

</head>

<body>

    <div class="main">

        <img src="img/image.png">

        <h1>Prokopenko Kirill</h1>

        <p class="p1">Student</p>

        <div class="title"><b>Summary</b><hr></div>

        <div class="text">I have no work experience yet.

            Now I study in college. Here I study python, c++, rtml. I am learning Kotlin by myself.

            I have analytical thinking, attention to detail, persistence,

            communication skills, creativity, self-organization and willingness to cooperate.

         </div>

        <div class="title"><b>Education</b><hr></div>

        <div class="text">2012-2022. Koropska ZOSH I-III st. named after T.G. Shevchenko <br>2022-2026.

            Optical and Mechanical College of Taras Shevchenko Kyiv National University</div>

        <div class="title"><b>Hobbies</b><hr></div>

        <div class="text">My hobbies are playing the saxophone and reading books.

            Also, studying Kotlin can be considered a hobby, so far I am still at the elementary level, but I am learning quickly</div>

        <a href="https://www.facebook.com/kirill.prokopenko.33" target="\_blank"><div><button><i class="fa-brands fa-facebook"></i> Facebook</button></div></a>

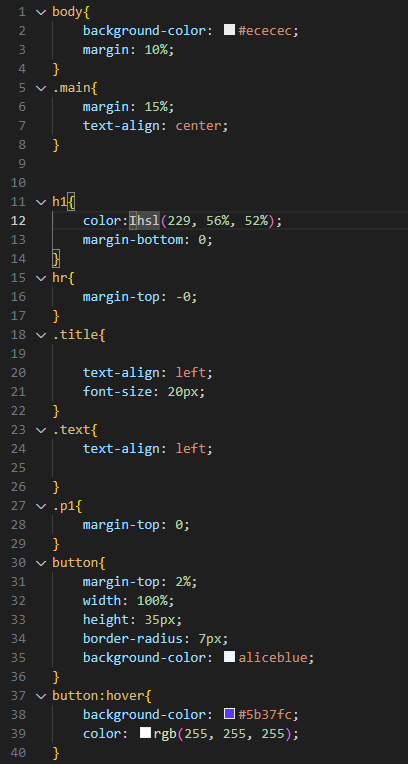
        <a href="t.me/dest1nycs" target="\_blank"><div><button><i class="fa-brands fa-telegram"></i> Telegram</button></div></a>

        <a href="https://www.instagram.com/dest1nycs/" target="\_blank"><div><button><i class="fa-brands fa-instagram"></i> instagram</button></div></a>

    </div>

</body>

</html>

****

****

body{

    background-color: #ececec;

    margin: 10%;

}

.main{

    margin: 15%;

    text-align: center;

}

h1{

    color:Іhsl(229, 56%, 52%);

    margin-bottom: 0;

}

hr{

    margin-top: -0;

}

.title{

    text-align: left;

    font-size: 20px;

}

.text{

    text-align: left;

}

.p1{

    margin-top: 0;

}

button{

    margin-top: 2%;

    width: 100%;

    height: 35px;

    border-radius: 7px;

    background-color: aliceblue;

}

button:hover{

    background-color: #5b37fc;

    color: rgb(255, 255, 255);

}

img

{

    width: 150px;

    height: 150px;

    border-radius: 50%;

    object-fit: cover;

    margin-bottom: 10px;

}

.text{

    font-size: 18px;

    padding-bottom: 15px;

}

# ВИСНОВКИ

# 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. Мова програмування С++: конспект лекцій / О. В. Галкін, М. М. Верес. — К.: ДП “Вид. дім “Персонал”, 2017. — 260 с. — Біблі- огр.: с. 249.: <https://maup.com.ua/assets/files/lib/book/c_plisplus.pdf>
4. «ОСНОВИ ПРОГРАМУВАННЯ на C++» Зеленський О.С., Лисенко В.С. – Кривий Ріг: Державний університет економіки і технологій, 2023.-269 с.: <https://dspace.duet.edu.ua/jspui/bitstream/123456789/831/1/НП%20Osnovy_C%2B%2B.pdf>
5. Матеріал з Вікіпедії — вільної енциклопедії. «Markdown»: <https://uk.wikipedia.org/wiki/Markdown>
6. Е-helper.com.ua «Лекція 2. Розмітка тексту. Зображення. Посилання.»: <http://e-helper.com.ua/node/949>
7. «CASCADING STYLE SHEETS (CSS) - КАСКАДНІ ТАБЛИЦІ СТИЛІВ»: <https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=http://vv-steshyn.edu.kpi.ua/katalog/files/Lekciya-3.-CSS-kaskadni-tablici-stiliv-68.pdf&ved=2ahUKEwi5utTekqKGAxXD4AIHHaW3DtcQFnoECBIQAQ&usg=AOvVaw3_uxBU96i5-43dUKH8RfG4>
8. Е-helper.com.ua «Лекція 5. Наслідування та каскадування. Оформлення тексту за допомогою CSS»: <http://e-helper.com.ua/node/952>
9. А. Каплун, Ю. В. Баришев, А. В. Остапенко «ТЕХНОЛОГІЯ ПРОГРАМУВАННЯ ЛАБОРАТОРНИЙ ПРАКТИКУМ»: <https://web.posibnyky.vntu.edu.ua/fitki/14baryshev_tehnologiya_programuvannya/08.html>
10. Зображення біля посилань <https://fontawesome.com/icons/>

# ДОДАТКИ