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:Барбара Владислав Андрійович |
| студента 2-го курсу групи ІПЗ-21 |
| прізвище ім‘я по-батькові  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) | |  | |
|  | | | | |
| *Барбари Владислава Андрійовича* | | | | |
| 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: Барбара Владислав Андрійович

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. Систематизація матеріалів, оформлення звіту з навчальної практики

# 1 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ

### Фізична безпека

#### Постава та ергономіка

* **Правильне сидіння**: Сидіть прямо, ноги повинні бути на підлозі або на підставці, коліна повинні бути на рівні стегон або трохи нижче. Спина повинна бути підтримана спинкою крісла.
* **Розташування монітора**: Верхня частина екрана повинна бути на рівні очей або трохи нижче. Відстань до екрана повинна бути приблизно 50-70 см.
* **Клавіатура та миша**: Клавіатура повинна бути розташована так, щоб руки були на рівні ліктів або трохи нижче. Миша повинна бути близько до клавіатури.

#### Здоров’я очей

* **Регулярні перерви**: Робіть перерви кожні 20-30 хвилин, щоб дати очам відпочити. Можна використовувати правило 20-20-20: кожні 20 хвилин дивіться на об’єкт, розташований на відстані 20 футів (приблизно 6 метрів) протягом 20 секунд.
* **Освітлення**: Освітлення має бути рівномірним і не створювати відблисків на екрані.

### Психічна безпека

#### Профілактика стресу

* **Організація робочого місця**: Робоче місце повинно бути чистим і організованим, щоб уникнути стресу від безладу.
* **Режим праці та відпочинку**: Дотримуйтеся регулярного розкладу роботи та відпочинку. Відпочинок допомагає знижувати рівень стресу та підвищує продуктивність.

#### Психоемоційна стабільність

* **Соціальна взаємодія**: Підтримуйте зв’язок з колегами та друзями. Соціальна підтримка є важливим чинником психічного здоров’я.
* **Фізична активність**: Регулярні фізичні вправи допомагають знижувати рівень стресу і покращують загальний стан здоров’я.

### Інформаційна безпека

#### Захист даних

* **Паролі**: Використовуйте складні паролі і змінюйте їх регулярно. Використовуйте двофакторну автентифікацію, де це можливо.
* **Антивірусне програмне забезпечення**: Встановіть та регулярно оновлюйте антивірусне програмне забезпечення.

#### Безпечний доступ до мережі

* **Файрвол**: Використовуйте файрвол для захисту від несанкціонованого доступу до вашої мережі.
* **Оновлення програмного забезпечення**: Регулярно оновлюйте операційну систему та програмне забезпечення, щоб забезпечити захист від нових загроз.

# 2 РОЗДІЛ "ІНФОРМАТИКА"

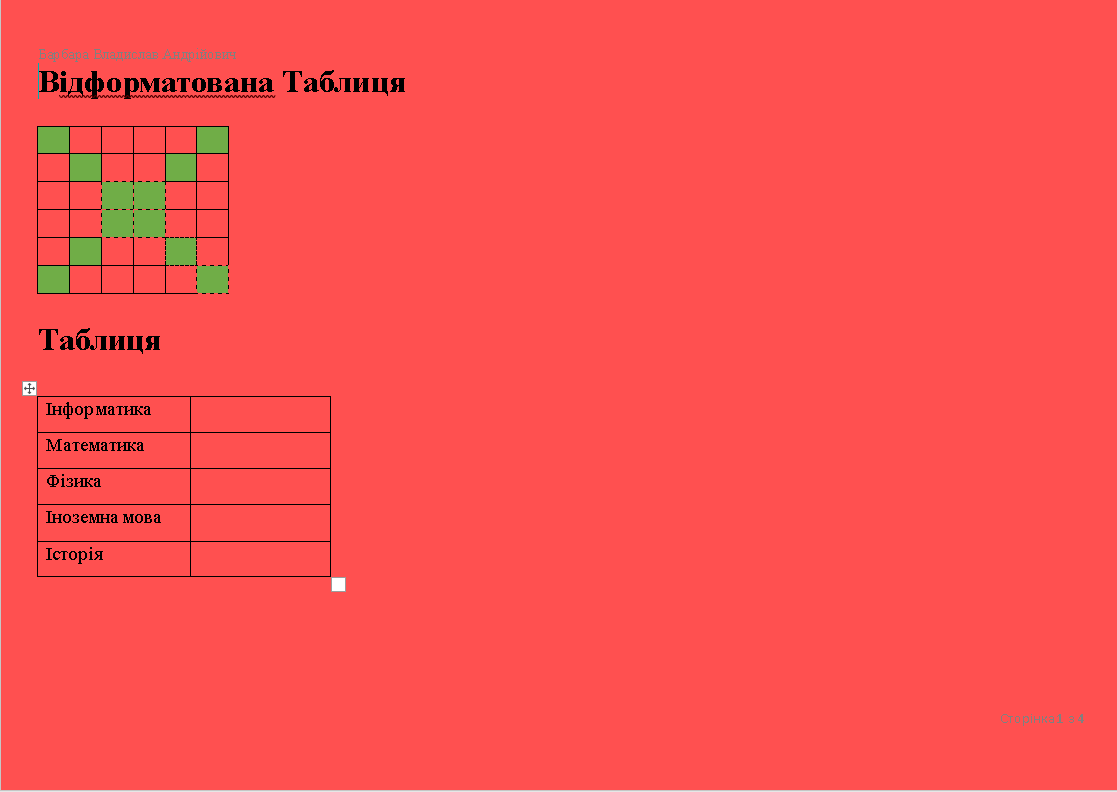
## **2.1 Розроблення засобами MS Word документа складної структури**

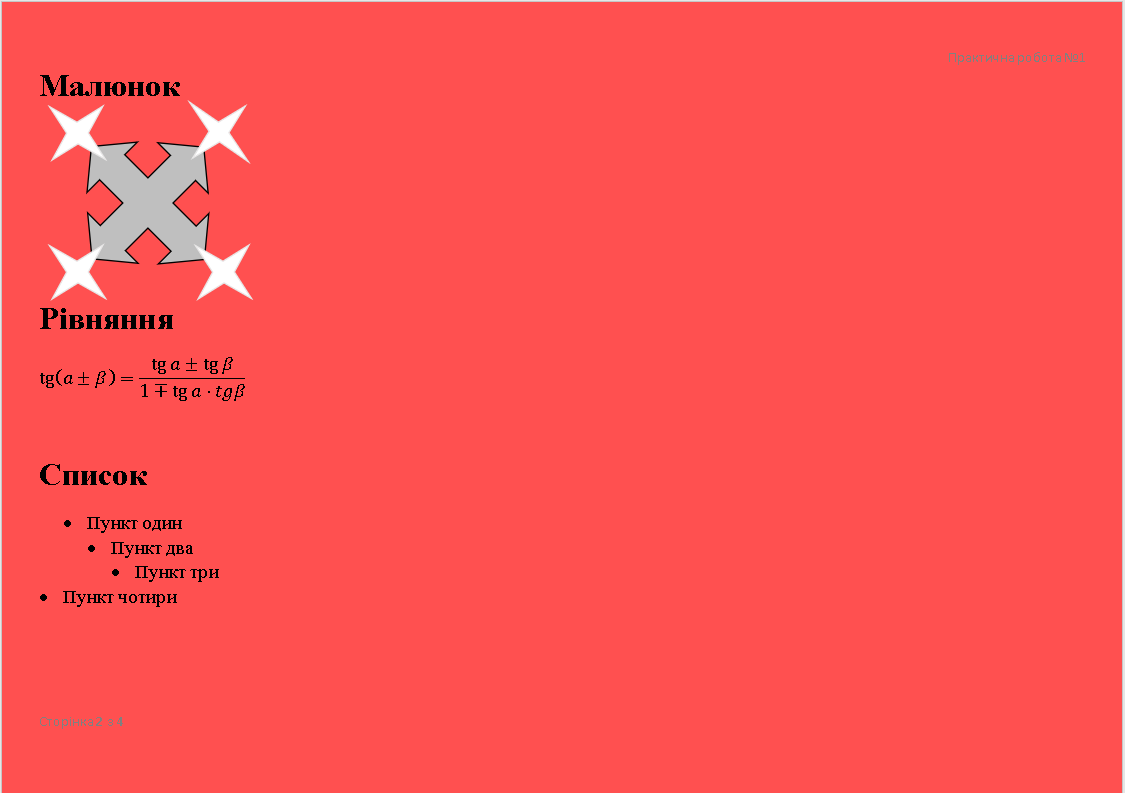
Під оформленням документа розуміють його структуру: зовнішню (параметри сторінок, поля тощо) і внутрішню (заголовки, підписи тощо).

Великі складні документи розбивають на частини, розділи, підрозділи, параграфи та інші структурні елементи , відформатовані з дотриманням певних правил.

Найчастіше структуру створюють для багатосторінкових документів, розділяючи їх на розділи, параграфи, пункти. Створення структури текстового документа в текстовому процесорі Word відбувається із застосуванням стилів оформлення текстових об’єктів Заголовок 1, Заголовок 2, …, Заголовок 9. Для цього потрібно назви всіх структурних частин документа відформатовати з використанням відповідного стилю. А саме, до назв структурних частин першого рівня (розділів) застосувати стиль Заголовок 1, назви частин другого рівня (пунктів) відформатовати із застосуванням стилю Заголовок 2 і т. д.

Виконана робота:







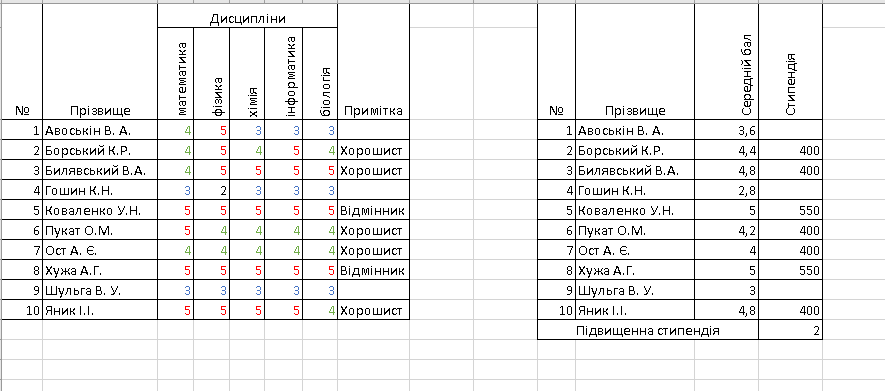


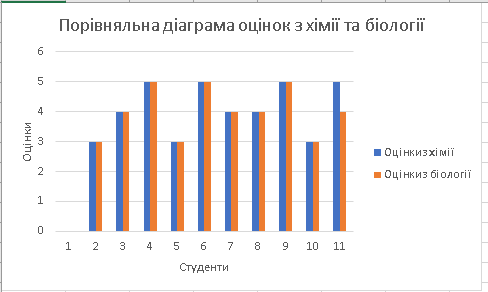
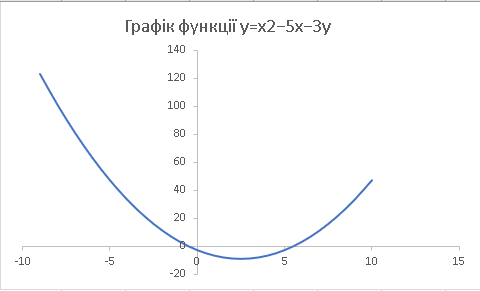
## **2.2 Розв‘язок задач за допомогою засобів табличного редактора MS Excel**

MS Excel є одним з найпопулярніших табличних редакторів у світі, розробленим компанією Microsoft. Він включає в себе ряд потужних інструментів, які дозволяють користувачам не лише створювати і форматувати таблиці, а й виконувати різноманітні обчислення, обробку даних і аналіз.

Основні можливості Excel включають:

1. **Створення таблиць і форматування**: Ви можете легко створювати таблиці, вставляти дані і форматувати їх за допомогою широкого набору інструментів для кольорів, шрифтів, рамок і т.д.
2. **Обчислення і формули**: Excel має вбудовані функції і формули для автоматизації обчислень. Ви можете виконувати арифметичні операції, вираховувати суми, середні значення, мінімуми та максимуми, а також використовувати складніші функції для статистики, фінансів, логіки та ін.
3. **Аналіз даних і графіки**: Excel дозволяє вам виконувати різноманітний аналіз даних, такий як сортування, фільтрація, пошук значень, підсумовування за умовами. Крім того, ви можете будувати різні типи графіків (стовпчасті, лінійні, колові тощо), щоб візуалізувати дані.
4. **Імпорт і експорт даних**: Excel підтримує імпорт і експорт даних з різних джерел, таких як бази даних, текстові файли, інші таблиці Excel, що спрощує обробку інформації з різних джерел.
5. **Автоматизація і макроси**: Excel дозволяє створювати макроси і автоматизовані рішення за допомогою мови програмування VBA (Visual Basic for Applications), що дозволяє автоматизувати повторювані завдання і створювати додаткові функціональні можливості.
6. **Спільна робота і спільний доступ**: Excel підтримує роботу в групах, зокрема спільну роботу над одним файлом через хмарні сервіси, що спрощує спільну роботу і обмін даними між 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 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. Це не тільки найпоширеніша мова програмування, але й мова спілкування програмістів, оскільки більшість програм алгоритмів написані на C++.

Мова C++ є мовою високого рівня і основою багатьох систем програмування, наприклад **Visual Studio**, **Eclipse**. Найбільш популярною з них вважається **Visual Studio**. За допомогою цієї системи візуального об’єктно-орієнтованого програмування як користувач-початківець, так і програміст-професіонал мають можливість створювати інтерфейс користувача до прикладних програм різноманітних класів, що виглядає однаково професійно.

Константи являють собою фіксовані значення, що не можуть змінюватися впродовж виконання всієї програми.

Спосіб визначення кожної константи залежить від її типу. Константи мови С++ слід поділяти на літеральні та типізовані.

**Змінна** — це іменована область пам’яті, у якій зберігаються дані визначеного типу. Змінна має ім’я, розмір та інші атрибути, такі як видимість, час існування тощо. *Ім’я* *змінної* служить для звертання до області пам’яті, у якій зберігається її значення. Кожна змінна повинна мати своє ім’я, причому в одному блоці не може бути двох змінних з однаковим ім’ям. ***Перед використанням будь-яка змінна повинна бути описана***, при цьому для неї резервується деяка область пам’яті, розмір якої залежить від конкретного типу змінної. Під час виконання програми змінна може приймати різні значення.

Для здійснення маніпуляцій з даними мова C++ застосовує широкий набір операцій (див. *табл. 5*), що виконують формування і, відповідно, подальше обчислення виразів. Вирази містять **одну** або **декілька** операцій, об’єкти яких називають операндами. Операції являють собою деяку дію, що виконується над **одним** (***унарна***) або **декількома** (***бінарна, триарна***) операндами, і мають позначення (наприклад, операція перевірки на рівність — позначення «==», операція обчислення залишку від ділення цілих чисел — позначення «%» тощо).

**Перетворення типів**

***Правила приведення типів:***

1. Автоматично здійснюються тільки ті перетворення, які перетворюють операнди з меншим діапазоном значень в операнди з більшим діапазоном значень, наприклад:

**int i\_var = 5;**

**float f\_var = 2.5, summa;**

**…**

**summa = i\_var + f\_var;**

2. Вирази, що не мають змісту (наприклад, число з плаваючою комою в ролі індексу), не пропускаються компілятором ще на етапі трансляції:

***float f;***

***…***

***mas [f]=25; // викликає помилку трансляції (Error)***

3. Вирази, в яких могла б втрачатися інформація (наприклад, при присвоєнні довгих цілих коротшим або дійсних цілим), можуть викликати попередження (Warning), але вони допустимі:

***int i;***

***float f=3.2;***

***i=f; // попередження (Warning) при трансляції***

### 3.1.2 Арифметичні 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.

**Операції присвоювання:**

**=** — присвоювання значення **П** змінній **Л**;

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

**–=** — віднімає величину **П** від змінної **Л**;

**\*=**  — множення змінної **Л** на величину **П**;

**/=** — ділення **Л** на **П**;

**%=** — видає залишок від ділення **Л** на **П**.

Просте присвоювання здійснює операція «=». Допускається одночасне зчіплювання декількох операцій присвоювання за умови, що всі операнди мають однаковий тип, наприклад:

int і, j, с;

i **= j = с = 0;**

Операції **«+=», «-=», «\*=», «/=»** виконують складні присвоювання і дозволяють записувати вирази коротше, наприклад:

**s += 7;**       //s=s + 7;

**і \*= j + 5;**   //i=i\*(j +5);

g%=9;        //g=g%9;.

**Операції відношення** порівнюють значення **Л** зі значенням **П**:

< — менше;

<= — менше або дорівнює (не перевищує);

== — дорівнює;

> — більше;

>= — більше або дорівнює (не менше);

!= — не дорівнює.

У мові C++ «істина» — це ненульова величина, «неправда» — це нуль **(0)**. У більшості випадків одиниця **(1)** викориcтовується як ненульове значення.

Операції відношення повертають ціле значення 1, якщо умова вірна, або 0, якщо умова помилкова.

**Логічні операції** оперують з цілими розмірами або з розмірами, які можна перетворити на цілі. Обчислення зупиняється, які тільки визначиться, чи є вираз правдивим («істина») або помилковим («неправда»). При цьому, як і для операцій відношення, значенням «істина» відповідає 1, а значенням «неправда» — 0.

&& — логічне **«AND»** (кон’юнкція);

|| — логічне **«OR»** (диз’юнкція);

!= — логічне **«NOT»** (заперечення).

Результат операції «&&» є «істина» **(1)**, якщо обидва її oпeранди правдиві (не рівні 0). Результат операції «||» — «істина» **(1)**, якщо хоча б один з її операндів є «істина». Логічне заперечення «!=» перетворює свій операнд на «істину» **(1)**, якщо він дорівнює **0**, і на «неправду» **(0)**, якщо він не дорівнює **0**.

З  використанням логічних операцій та операцій відношення записуються різні умовні вирази, наприклад, умова **3 < х < 5** ма­тиме вигляд: **х > 3 && х < 5**.

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

**Поняття вхідного та вихідного потоку**

У контексті програмування, вхідний потік (input stream) та вихідний потік (output stream) є основними поняттями для роботи з введенням та виведенням даних.

- **Вхідний потік** (input stream): Це потік, через який програма отримує дані з зовнішніх джерел, таких як клавіатура, файли чи інші пристрої введення. Використовується для читання даних у програмі.

- **Вихідний потік** (output stream): Це потік, через який програма виводить дані до зовнішніх пристроїв, таких як екран, файли або інші пристрої виведення. Використовується для відображення результатів роботи програми.

У більшості мов програмування, зокрема у C++ з використанням стандартної бібліотеки (`iostream`), для роботи з вхідними та вихідними потоками використовуються об'єкти класів `std::ifstream` (для вхідних файлів), `std::ofstream` (для вихідних файлів), `std::cin` (стандартний ввід, наприклад, з клавіатури) та `std::cout` (стандартний вивід, наприклад, на екран).

Найпростіші математичні функції

Найпростіші математичні функції, які часто використовуються у програмуванні, включають базові операції:

1. **Додавання** (`+`) - об'єднує два числа в одне, результатом є їхня сума.

Наприклад:

```cpp

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

```

2. **Віднімання** (`-`) - віднімає одне число від іншого, результатом є їхня різниця.

Наприклад:

```cpp

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

```

3. **Множення** (`\*`) - перемножує два числа, результатом є їхній добуток.

Наприклад:

```cpp

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

```

4. **Ділення** (`/`) - ділить одне число на інше, результатом є їхній частка.

Наприклад:

```cpp

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

```

5. **Модуль** (`%`) - повертає залишок від ділення одного числа на інше (цілочисельне ділення).

Наприклад:

```cpp

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

```

### 3.1.4 Програмування лінійних та розгалужених обчислювальних процесів

Програмування лінійних та розгалужених обчислювальних процесів є основоположними аспектами розробки програмного забезпечення. Давайте розглянемо кожен з них детальніше.

**Лінійні обчислювальні процеси**

Лінійні обчислювальні процеси характеризуються послідовністю виконання команд або інструкцій у програмі. Вони виконуються одна за одною в порядку, в якому вони були написані. Такі процеси не мають розгалужень або умовних переходів, що дозволяє їм виконуватися прямолінійно.

Приклад лінійного обчислювального процесу в C++

```cpp

#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, які 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++

```cpp

#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 складовою розробки програмного забезпечення. Цикли дозволяють виконувати одну чи кілька команд або блоків коду декілька разів підряд, що дозволяє автоматизувати ітеративні завдання або операції, які потребують повторення.

Основні типи циклів

У більшості мов програмування існують кілька типів циклів, що відрізняються за умовами виходу і способом виконання:

1. **Цикл while** - цикл з передумовою:

- Виконується, поки задана умова є true.

- Умова перевіряється перед кожним виконанням тіла циклу.

```cpp

int i = 0;

while (i < 5) {

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

i++;

}

// Виведе: 0 1 2 3 4

```

2. **Цикл do-while** - цикл з післяумовою:

- Виконується принаймні один раз, потім перевіряється умова.

- Тіло циклу виконується, поки умова є true.

```cpp

int i = 0;

do {

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

i++;

} while (i < 5);

// Виведе: 0 1 2 3 4

```

3. **Цикл for** - цикл з контрольованою послідовністю:

- Виконується задану кількість разів або до досягнення певної умови.

- Складається з ініціалізації, умови та кроку ітерації.

```cpp

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

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

}

// Виведе: 0 1 2 3 4

```

## **3.2 Практичне завдання**

### 3.2.1 Завдання

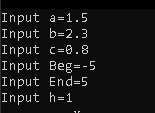
**Завдання**: Обчислити і вивести на екран у табличному вигляді значення функції *f(x)* на заданому інтервалі зміни значень аргументу *х* від *xпоч* до *xкін* з кроком *h.*

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

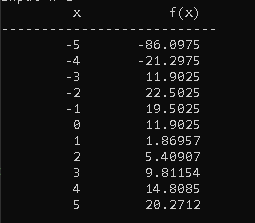


### 3.2.2 Постановка задачі

Вхідні дані



Вихідні дані



Блок-схема алгоритму роботи програми представлена на рис. ххх

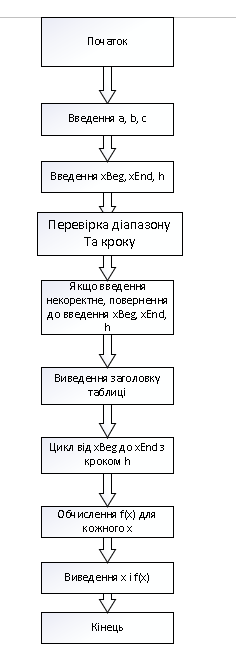


Рисунок хх - Блок-схема алгоритму роботи програми

### 3.2.3 Код програми

#include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

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

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

return 2 \* c \* x + b \* pow(x, 3);

}

else if (x > 0 && c >= 0) {

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

}

else {

return a \* pow(x, 2) \* (x + c) + pow(x - a \* b, 2);

}

}

void printTable(double a, double b, double c, double xBeg, double xEnd, double h) {

cout << setw(10) << "x" << setw(15) << "f(x)" << endl;

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

if (xBeg <= xEnd && h > 0) {

for (double x = xBeg; x <= xEnd; x += h) {

double fx = calculateFunction(a, b, c, x);

cout << setw(10) << x << setw(15) << fx << endl;

}

}

else if (xBeg >= xEnd && h < 0) {

for (double x = xBeg; x >= xEnd; x += h) {

double fx = calculateFunction(a, b, c, x);

cout << setw(10) << x << setw(15) << fx << endl;

}

}

else {

cout << "Error: Invalid range or step size" << endl;

}

}

int main() {

double a, b, c, xBeg, xEnd, h;

cout << "Input a="; cin >> a;

cout << "Input b="; cin >> b;

cout << "Input c="; cin >> c;

int attempts = 0;

do {

if (attempts++ > 0) cout << "Error: Invalid range or step size" << endl;

cout << "Input Beg="; cin >> xBeg;

cout << "Input End="; cin >> xEnd;

cout << "Input h="; cin >> h;

} while ((xBeg > xEnd && h > 0) || (xBeg < xEnd && h < 0) || h == 0);

printTable(a, b, c, xBeg, xEnd, h);

return 0;

}

### 3.2.4 Скріншот виконання програми

Скріншот виконання програми представлена на рис. ххх

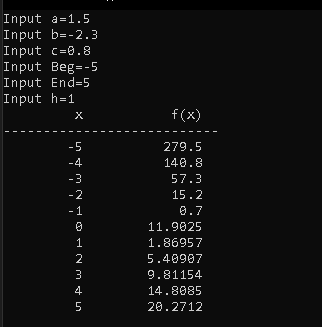


Рисунок хх - Скріншот виконання програми

## **3.3 Виконання тестів**

# 4 РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++"

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

Оскільки основна задача будь-якої програми опрацьовувати дані – користувачеві хочеться бачити результат цієї роботи. Тому деякі дані треба виводити. Для цього існуються оператори виведення

При використанні оператора виведення як параметри можна задавати не лише конкретні значення, імена змінних, але і навіть цілі вирази, при цьому буде виведено результат виконання(обчислення) цього виразу.

В C++ залишається можливість використання бібліотеки stdio, однак частіше використовують потокове введення/виведення з бібліотеки **[iostream](http://www.cplusplus.com/reference/iostream/" \t "_blank)**

#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;

## 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

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, щомають 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.  
  
Cукупність розмірності й діапазонів називають формою масиву.  
  
**Звернення до значення елемента масиву** у коді програми здійснюють, вказавши після його назви у квадратних дужках індекси у вигляді (арифметичних) виразів, що набувають значень з діапазопу порядкового типу.  
  
Елемент масиву може мати і тип простої (неструктурованої) величини і складений тип (масиву, рядка тощо), тобто може існувати масив масивів, масив рядків тощо. Глибину вкладеності (а значить, і кількість індексів) необмежено, але загальну довжину структури обмежено. У пам'яті ПК всі елементи масиву зберігають послідовно: при переході від молодших до старших адрес першим буде змінено індекс, записаний останнім (праворуч).  
  
Під час розв'язування задач переважно використовують одновимірні та двовимірні масиви.  
  
**Одновимірний масив** інакше називають лінійним масивом або вектором. Кожному його елементу ставлять у відповідність один індекс. У математиці лінійному масиву відповідає поняття послідовності, а номеру члена послідовності — індекс масиву.  
  
**Двовимірний масив** інакше називають таблицею або матрицею. Кожному його елементу ставлять у відповідність два індекси: номер рядка та номер стовпчика. Прикладом унаочнення двовимірного масиву є шахівниця.  
  
  
  
**Зв'язний список** — лінійно упорядкована структура (послідовність) даних, елементи якої — вузли — містить два види даних:

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

Використовують також двічі зв'язні списки, в яких кожен вузол має вказіваник і на наступний, і на попередній елемент списку.  
  
**Основні дії зі зв'язним списком**:

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

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

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

## Класи файлового вводу/виводу

**Є три основні класи файлового вводу/виводу в мові C++:**

**ifstream** (є [**дочірнім**](https://acode.com.ua/urok-162-bazove-spadkuvannya/) класу **[istream](https://acode.com.ua/urok-216-funktsional-klasu-istream/" \t "_blank)**);

**ofstream** (є дочірнім класу **[ostream](https://acode.com.ua/urok-217-funktsional-klasiv-ostream-i-ios-formatuvannya-vyvodu/" \t "_blank)**);

**fstream** (є дочірнім класу iostream).

За допомогою цих класів можна виконувати однонаправлений файловий ввід, однонаправлений файловий вивід і двонаправлений файловий ввід/вивід. Для їх використання потрібно всього лише підключити [**заголовок**](https://acode.com.ua/urok-24-zagolovkovi-fajly/) fstream.

На відміну від потоків cout, cin, cerr і clog, які відразу ж можна використовувати, файлові потоки повинні бути явно встановлені програмістом. Тобто, щоб відкрити файл для читання і/або запису, потрібно створити об’єкт відповідного класу файлового вводу/виводу, вказавши ім’я файлу в якості параметра. Потім, за допомогою оператора вставки (<<) або оператора вилучення (>>), можна записувати дані в файл або зчитувати вміст файлу. Після виконання даних дій потрібно закрити файл — явно викликати **метод close()** або просто дозволити файловій змінній вводу/виводу вийти з області видимості ([**деструктор**](https://acode.com.ua/urok-128-destruktory/) файлового класу вводу/виводу закриє цей файл автоматично замість нас).

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

1. Текстові файли

Текстові файли є одними з найбільш простих та загальних форматів для зберігання даних. Вони містять символьні дані у формі, зрозумілій для людини. Кожен запис або значення може бути розділене пробілами, комами або іншими символами, щоб дозволити легше зчитування та редагування.

# Обробка текстових файлів

Текстові файли у мові C++ зазвичай обробляються за допомогою об'єктів потоків (std::ifstream для зчитування та std::ofstream для запису), які забезпечують читання та запис послідовностей символів.

2. Бінарні файли

Бінарні файли зберігають дані у бінарному форматі, який призначений для ефективного зберігання й обробки даних комп'ютером. Вони використовуються для зберігання складних структур даних, чисел, зображень та інших типів даних, які необхідно зберігати у вигляді бітів і байтів.

# Обробка бінарних файлів

Бінарні файли зазвичай використовуються для зберігання структурованих даних, таких як числа, масиви або складні структури даних. Оскільки вони зберігають дані у бінарному форматі, їх потрібно читати та записувати у відповідному форматі байтів.

3. CSV (Comma-Separated Values)

CSV є форматом текстових файлів, де кожне поле відокремлене від іншого комою або іншим роздільником. Використовується для зберігання табличних даних, таких як бази даних або електронні таблиці.

# Обробка CSV файлів CSV файли зазвичай містять табличні дані, де кожен рядок представляє запис, а кожне значення у рядку розділене комою або іншим роздільником.

# Зчитування з CSV файлів: Зчитування з CSV файлу в C++ вимагає розбиття кожного рядка на поля за допомогою роздільника (наприклад, коми).

# Запис у CSV файли: Запис у CSV файли виконується шляхом формування рядків з даними та вставки роздільників між значеннями полів.

4. XML (Extensible Markup Language)

XML є розширюваним мовним форматом для зберігання структурованих даних у вигляді тексту. Використовується для обміну даними між різними програмами та системами.

5. JSON (JavaScript Object Notation)

JSON є легким форматом обміну даними, який широко використовується в сучасному програмуванні для зберігання та передачі структурованих даних у текстовому форматі.

# Обробка XML та JSON файлів XML та JSON файли зазвичай використовуються для обміну структурованими даними між різними системами. Для їх обробки в C++ можуть використовуватися бібліотеки, такі як RapidXML або бібліотека JSON для C++, які забезпечують зручний інтерфейс для роботи з цими форматами.

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

**Системи керування версіями**(Version Control System, VCS) – програмні інструменти, які допомагають розробникам керувати змінами в програмному коді з плином часу. Система відстежує всі зміни, які вносяться в код та зберігає їх в спеціальній базі даних. Кожна зміна зберігається як версія. При виявленні помилки розробник може повернутися назад і виконати порівняння з попередніми версіями коду та, за необхідністю, відмінити зміни. Також система дозволяє без побоювань експериментувати з кодом та зберігати найбільш вдалий результат.

Розробляти програмне забезпечення можна і без системи керування версіями, але такий підхід наражає проект на величезний ризик, тому програмне забезпечення для керування версіями є невід’ємною частиною повсякденної професійної практики сучасної команди розробників програмного забезпечення.

**Типи систем керування версіями**

* локальні
* централізовані
* розподілені

**Локальна система керування версіями** – це локальна база даних, розташована на  особистому комп’ютері, в якій кожна зміна файлу зберігається як версія. Кожна версія містить лише зміни, внесені до файлу з моменту його останньої версії.

**Централізовані системи керування версіями** мають один сервер, який містить усі версії файлів, і кілька клієнтів можуть одночасно отримувати доступ до файлів на сервері, копіювати їх на свій локальний комп’ютер або надсилати зміни на сервер. Таким чином, клієнти можуть знати, чим займаються всі інші, а адміністратори можуть контролювати, хто що може робити. Це  дозволяє легко співпрацювати один з одним в команді. Протягом багатьох років це був стандарт для керування версіями.

**Розподілені системи керування версіями**використовуються для подолання недоліків централізованої системи контролю версій. Клієнти повністю клонують сховище файлів проекту (репозиторій), включаючи його повну історію. Всі копії є рівноправними і можуть синхронізуватися між собою. Якщо сервер вийде з ладу, будь-яке із клієнтських сховищ можна скопіювати на сервер, що допоможе його відновити. Кожен клон є повною резервною копією всіх даних.

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

Markdown - це легкий і простий у використанні форматувальний синтаксис для написання тексту на веб-сторінках. Він дозволяє створювати структуровані документи, не використовуючи складних HTML або CSS.

Ось основні елементи Markdown:

Заголовки

Для створення заголовків використовуються символи решітки (`#`). Кількість символів визначає рівень заголовка:

```

# Заголовок рівня 1

## Заголовок рівня 2

### Заголовок рівня 3

#### Заголовок рівня 4

##### Заголовок рівня 5

###### Заголовок рівня 6

```

Абзаци

Абзаци розділяються пустими рядками:

```

Це перший абзац.

Це другий абзац.

```

Списки

Markdown підтримує нумеровані та марковані списки:

Нумерований список:

```

1. Пункт перший

2. Пункт другий

3. Пункт третій

```

Маркований список:

```

- Пункт перший

- Пункт другий

- Пункт третій

```

Жирний та курсивний текст

Жирний текст відзначається подвійними зірочками (`\*\*`) або підкресленнями (`\_\_`):

```

\*\*Це жирний текст\*\*

\_\_Це теж жирний текст\_\_

```

Курсивний текст відзначається однією зірочкою (`\*`) або одним підкресленням (`\_`):

```

\*Це курсивний текст\*

\_Це теж курсивний текст\_

```

Посилання

Посилання вказуються у квадратних дужках, а URL в круглих:

```

[Текст посилання](http://example.com)

```

Зображення

Зображення вставляються аналогічно до посилань, але з префіксом `!`:

```

![Опис зображення](http://example.com/image.jpg)

```

Цитати

Цитати вказуються з символом `>` в початку кожного рядка:

```

> Це цитата

> З нового рядка

```

Таблиці

Таблиці створюються за допомогою вертикальних та горизонтальних рисок:

| Заголовок 1 | Заголовок 2 |

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

| Комірка 1.1 | Комірка 1.2 |

| Комірка 2.1 | Комірка 2.2 |

Це основні елементи Markdown, які дозволяють створювати чистий і чіткий текстовий формат для документів, блогів та інших веб-сторінок.

## **4.7 Розроблення ігрової програми**

### 4.7.1 Код програми

#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

#include <fstream>

#include <chrono>

#include <ctime>

using namespace std;

void logEvent(const string& event) {

ofstream logFile("hangman\_log.txt", ios\_base::app);

if (logFile.is\_open()) {

auto now = chrono::system\_clock::now();

time\_t now\_time = chrono::system\_clock::to\_time\_t(now);

char timeStr[26];

ctime\_s(timeStr, sizeof(timeStr), &now\_time);

timeStr[strlen(timeStr) - 1] = '\0';

logFile << "[" << timeStr << "] " << event << endl;

}

}

void displayHangman(int incorrectGuesses) {

vector<string> hangmanStages = {

" +---+\n |\n |\n |\n ===",

" +---+\n O |\n |\n |\n ===",

" +---+\n O |\n | |\n |\n ===",

" +---+\n O |\n /| |\n |\n ===",

" +---+\n O |\n /|\\ |\n |\n ===",

" +---+\n O |\n /|\\ |\n / |\n ===",

" +---+\n O |\n /|\\ |\n / \\ |\n ==="

};

cout << hangmanStages[incorrectGuesses] << endl;

}

void displayWord(const string& word, const vector<bool>& guessed) {

for (int i = 0; i < word.size(); i++) {

if (guessed[i]) {

cout << word[i] << ' ';

}

else {

cout << "\_ ";

}

}

cout << endl;

}

bool isWordGuessed(const vector<bool>& guessed) {

for (bool letterGuessed : guessed) {

if (!letterGuessed) {

return false;

}

}

return true;

}

void updateWinsFile(int wins) {

ofstream winsFile("hangman\_wins.txt");

if (winsFile.is\_open()) {

winsFile << "Победа: " << wins << endl;

}

}

int readWinsFile() {

ifstream winsFile("hangman\_wins.txt");

int wins = 0;

if (winsFile.is\_open()) {

string label;

winsFile >> label >> wins;

}

return wins;

}

void playHangman(const string& word) {

int incorrectGuesses = 0;

vector<bool> guessed(word.size(), false);

vector<char> incorrectLetters;

while (incorrectGuesses < 6) {

system("cls");

displayHangman(incorrectGuesses);

displayWord(word, guessed);

cout << "Incorrect guesses: ";

for (char letter : incorrectLetters) {

cout << letter << ' ';

}

cout << endl;

cout << "Enter your guess: ";

char guess;

cin >> guess;

logEvent("Player guessed: " + string(1, guess));

if (find(incorrectLetters.begin(), incorrectLetters.end(), guess) != incorrectLetters.end()) {

cout << "You already guessed that letter. Try again." << endl;

logEvent("Repeated guess: " + string(1, guess));

continue;

}

bool correctGuess = false;

for (int i = 0; i < word.size(); i++) {

if (word[i] == guess) {

guessed[i] = true;

correctGuess = true;

}

}

if (!correctGuess) {

incorrectLetters.push\_back(guess);

incorrectGuesses++;

}

if (isWordGuessed(guessed)) {

system("cls");

displayHangman(incorrectGuesses);

displayWord(word, guessed);

cout << "Congratulations! You guessed the word: " << word << endl;

logEvent("Victory! Player won the game.");

int wins = readWinsFile();

wins++;

updateWinsFile(wins);

return;

}

}

system("cls");

displayHangman(6);

cout << "Game over! The word was: " << word << endl;

logEvent("Player lost the game. The word was: " + word);

}

int main() {

cout << "Welcome to Hangman!" << endl;

int wins = readWinsFile();

cout << "Total wins: " << wins << endl;

cout << "Player 1, enter the word to guess: ";

string word;

cin >> word;

system("cls");

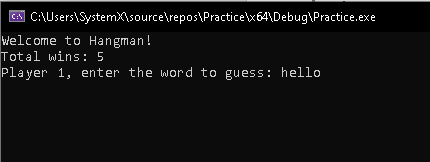
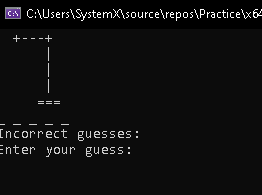
logEvent("New game started. Word to guess: " + word);

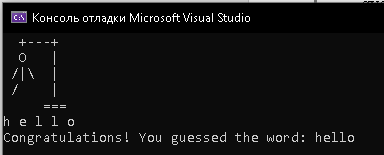
playHangman(word);

return 0;

}

### 4.7.2 Скріншоти виконання програми



# 5 РОЗДІЛ ”ВЕБ-ДИЗАЙН ТА HTML”

HTML (від англ. HyperText Markup Language — «мова гіпертекстової розмітки») — стандартизована мова гіпертекстової розмітки документів для перегляду веб-сторінок у браузері. Веб-браузери отримують HTML-документ від сервера за протоколами HTTP/HTTPS або відкривають з локального диска, далі інтерпретують код в інтерфейс, який буде відображатися на екрані монітора.

Елементи HTML є будівельними блоками HTML-сторінок. За допомогою HTML різні конструкції, зображення та інші об'єкти, такі як інтерактивна веб-форма, можуть бути вбудовані в відображувану сторінку. HTML надає засоби для створення заголовків, абзаців, списків, посилань, цитат та інших елементів. Елементи HTML виділяються тегами, записаними з використанням кутових дужок. Такі теги, як <img> та <input>, безпосередньо вводять контент на сторінку. Інші теги, такі як <p>, оточують і оформлюють текст всередині себе і можуть включати інші теги як піделементи. Браузери не відображають HTML-теги, але використовують їх для інтерпретації вмісту сторінки.

Мова XHTML є більш суворим варіантом HTML, вона дотримується синтаксису XML і є застосуванням мови XML у сфері розмітки гіпертексту.

В HTML можна вбудувати мову програмування JavaScript за допомогою тега. Також включення CSS в HTML дозволяє задавати зовнішній вигляд і макет сторінки.

## **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 ім'я тега. В імені літери великий і малі не розрізняються.

Теги бувають двох видів — одинарні й парні. Одинарний тег ставиться перед фрагментом тексту, який він керує. Парні теги складаються з двох частин — початкової та завершальної. Початкова частина вказує точку документа, з якої починається команда, а завершальна — точку її закінчення. Завершальна частина відрізняється від початкової тільки тим, що перед ім'ям тега ставиться знак "/" (прямий слеш). Наприклад, запис <b> Синій </b> вказує браузеру, що текст "Синій" слід відобразити напівжирним шрифтом.

Розроблення Web-сторінки складається з двох етапів:

1. Створення макета.

2. Вставка тегів мовою HTML.

При створенні макета визначається зміст Web-сторінки та її зовнішній вигляд. Вставка тегів забезпечує потрібний зовнішній вигляд при перегляді Web-сторінки за допомогою браузера.

Ознакою того, що текстовий файл є HTML-документом, служить пара тегів <HTML> </HTML>. Він починається й завершується цими документами, тобто HTML-документ має вигляд:

```html

<HTML>

текст документа

</HTML>

```

HTML-документ складається з двох частин — заголовка й тіла. Заголовок позначається парними тегами <HEAD> ... </HEAD>, а тіло — парними тегами <BODY> ... </BODY>. Текст, що відображається у заголовку вікна браузера, розміщується в парному тегі <TITLE> ... </TITLE>. Структура HTML-документа має вигляд:

```html

<HTML>

<HEAD>

<TITLE> Заголовок документа </TITLE>

</HEAD>

<BODY>

текст документа

</BODY>

</HTML>

```

У будь-якому місці HTML-документа можна помістити коментар. Це текст, який всередині документа не відображається, а служить для того, щоб розробник документа міг робити позначки. У браузері текст коментаря поміщається в рамку <!--...-->.

**Форматування абзаців і символів**

Щоб текст сторінки не відображався суцільним текстом, використовують такі одинарні теги:

- `<p>` - початок абзацу (текст після тега починається з нового абзацу);

- `<br>` - початок нового рядка (текст після тега починається з нового рядка);

- `<hr>` - горизонтальна лінія (у рядку, де розміщено тег, відображається горизонтальна лінія).

При використанні тега `<p>` спочатку вставляється порожній рядок. Далі текст виводиться з початку нового рядка. Якщо вжито тег `<br>`, то порожній рядок не вставляється.

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

```html

<Hn> ... </Hn>

```

де n — номер рівня.

Наприклад, фрагмент HTML-документа

```html

<H1> 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 вставляти парний тег <pre> ... </pre>. Він позначає попередньо відформатований фрагмент тексту, що відображається шрифтом фіксованої ширини й зберігає всі пробіли й символи кінця рядка. Часто цей тег використовується для вставки програмних кодів.

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

**Вставляння малюнків**

Щоб вставити малюнок на вебсторінку, необхідно скористатися тегом **<img src = "my.jpg">**, де **img**— ім’я тегу, призначеного для вставляння зображення,**src = "my.jpg"** — атрибут, в якому вказується ім’я файлу з малюнком.

В атрибуті можна використовувати імена графічних файлів із розширеннями *.jpg, .jpеg, .gif, .png*або *.bmp*. Якщо малюнок і HTML-файл містяться в різних папках, то, крім імені файлу, потрібно зазначати шлях до нього.

*Приклад:*

*Якщо малюнок міститься у вкладеній папці images:****<img src = "images/my.jpg">*** *Якщо малюнок міститься у зовнішній папці images:****<img src = "../images/my.jpg">***

Для визначення способу розташування малюнка на вебсторінці використовують окремі атрибути тегу.

Додавання гіперпосилань

Сайт може складатись із багатьох пов’язаних гіперпосиланнями вебсторінок, одна з яких є головною. Головна вебсторінка відкривається першою.

Для створення гіперпосилань на вебсторінки сайту та зовнішні ресурси мережі інтернет використовують тег **<a>...</a>**.

Атрибут **href**визначає URL-адресу сторінки на яку вказує посилання. Оскільки в якості адреси посилання може використовуватися документ будь-якого типу, то результат переходу за посиланням залежить від кінцевого файлу. За замовчуванням новий документ завантажується в поточне вікно браузера, однак це властивість можна змінити за допомогою атрибута **target**. Якщо атрибут **href**відсутній, то теґ **<a>** не є гіперпосиланням.

Нехай в одній папці з файлом *index.html* створено файл **prf.html,** який містить вашу фотографію. У файлі*index.html* до фрази *Подивитися фото* можна додати гіперпосилання, яке матиме такий вигляд:  
**<a href = "prf.html"> Подивитися фото </a>**

Гіперпосилання можна додати до малюнка. Для цього треба вказати тег вставляння малюнка:  
**<a href = "prf.html"><img src = "foto1.jpg"></a>**

*Приклад:*

*HTML-код головної вебсторінки може мати такий вигляд:  
<html>  
<body>  
     <a href = "first.html">Перша сторінка</a><br>  
     <a href = "second.html">Друга сторінка</a><br>  
     <a href = "third.html">Третя сторінка</a><br>  
</body>  
</html>  
Після клацання напису Перша сторінка (Друга сторінка, Третя сторінка) відбудеться перехід за гіперпосиланням.*

Для повернення на головну сторінку на інших вебсторінках мають бути відповідні гіперпосилання. Наприклад:  
**<a href = "index.html">Головна</a>**

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

**CSS** розшифровується як **C**ascading **S**tyle **S**heets - Каскадна таблиця стилів.

CSS описує, **як HTML елементи мають відображатись на екрані, папері або інших носіях.**.

CSS **економить багато часу**. CSS може контролювати макет кількох веб-сторінок одночасно.

CSS можна додати до елементів HTML трьома способами:

* **Inline** (вбудований або рядковий) - використовуючи атрибут style в HTML елементах
* **Internal** (внутрішній) - використовуючи елемент <style> в разділі <head>
* **External** (зовнішній) - за допомогою зовнішнього файлу CSS

Найбільш розповсюджений спосіб додати CSS - це зберігати стилі в окремих файлах CSS. Тим не менше, тут ми будемо використовувати вбудований і внутрішній стиль, тому що для навчання це легше продемонструвати, і вам простіше буде спробувати написання CSS-коду самостійно.

## **Inline CSS - Вбудований (рядковий) стиль**

Вбудований CSS використовується для застосування унікального стиля до одного HTML элемента.

Вбудований CSS використовує атрибут style HTML елемента.

Цей приклад встановлює колір тексту елемента <h1> в синій:[Спробуйте самі »](https://w3schoolsua.github.io/htmltryit/tryhtml_css_inline.html)

**Internal CSS - Внутрішній стиль**

Внутрішній CSS використовується для визначення стилю окремої HTML сторінки.

Внутрішній CSS вказується в розділі <head> HTML сторінки за допомогою елемента <style>:[Спробуйте самі »](https://w3schoolsua.github.io/htmltryit/tryhtml_css_internal.html)

## **External CSS - Зовнішній стиль**

Зовнішня таблиця стилів використовується для визначення стилю для багатьох HTML-сторінок.

**За допомогою зовнішньої таблиці стилів ви можете змінити зовнішній вигляд всього веб-сайту, змінивши лише один файл!**

Щоб використовувати зовнішню таблицю стилів, додайте посилання на неї в розділі <head> HTML сторінки:[Спробуйте самі »](https://w3schoolsua.github.io/htmltryit/tryhtml_css_external.html)

Зовнішню таблицю стилів також, як і HTML-код, можна написати в будь-якому текстовому редакторі. Файл не повинен містити HTML-код і повинен бути збережений з розширенням .css.

## **CSS Шрифти**

CSS властивість color визначає колір тексту, який буде використовуватися.

CSS властивість font-family визначає сімейство шрифту, яке буде використовуватися.

CSS властивість font-size визначає розмір тексту, який буде використовуватися.

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

**Спадковість**

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

<p class = "text"> Рядок з виділеним <span> словом </ span> </ p>

Уявімо, що нам потрібно встановити червоний колір шрифту для всього тексту. Задаємо CSS-властивості наступним чином:

.text {

color: red;

}

Завдяки наслідуванню колір шрифту в тезі span автоматично стане червоним:

Рядок з виділеним словом

А так би виглядав результат, якби наслідування не працювало:

Рядок з виділеним словом

**Каскадність**

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

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

CSS-правила існують як мінімум в трьох різних місцях:

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

2. в атрибуті style;

3. в стандартних стилях відображення, вбудованих в браузер.

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

1. важливість;

2. специфічність;

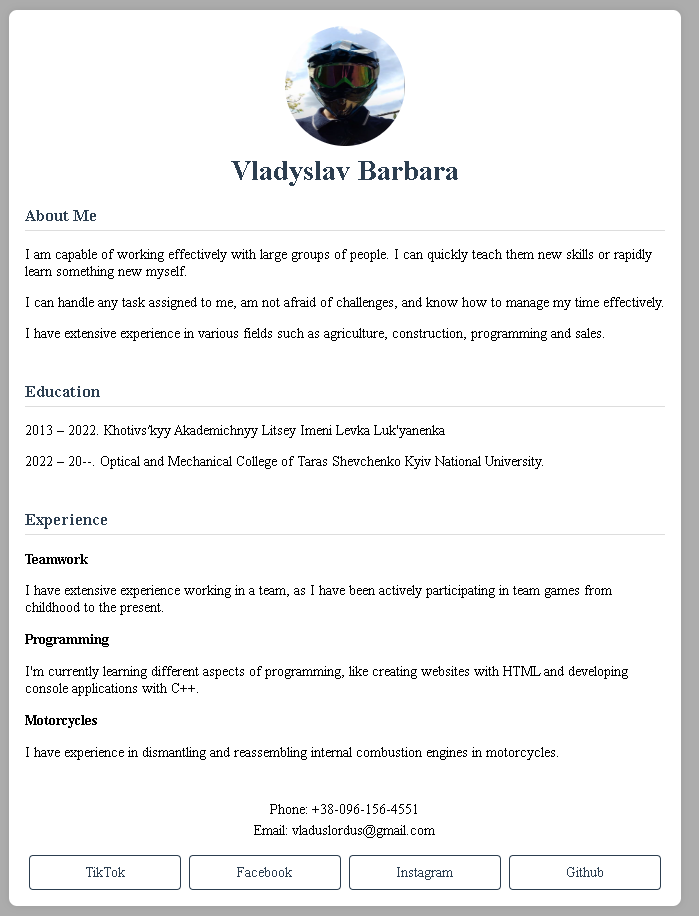
3. порядок вихідного коду.

## **5.5 Практична частина**

### 5.5.1 Постановки задачі

Основна мета проекта була створення веб-сайту у формі резюме для вивчення HTML і CSS. Проект включав в себе текст зі зміненими кольорами, розмірами шрифтів та стилями для поліпшення читабельності, а також фотографії, активні посилання для зручної навігації та інші оформлені елементи для кращого сприйняття відвідувачами.

### 5.5.2 HTML документ



# ВИСНОВКИ

**РОЗДІЛ "ІНФОРМАТИКА**

В процесі роботи з MS Word для створення документа складної структури важливо правильно організувати інформацію, використовувати засоби форматування, таблиці, списки та інші елементи для зручності читання і візуальної привабливості документа. Цей інструмент є незамінним для створення звітів, презентацій, наукових робіт та інших видів документації.

Співпраця MS Word з MS Excel стає вирішальною у вирішенні завдань, які вимагають аналізу і 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 багатьох інших сфер.

**РОЗДІЛ "ПРОГРАМУВАННЯ С++**

В процесі практичних завдань я отримаув досвід розв'язання реальних задач, використовуючи можливості мови C++ для ефективної обробки даних, взаємодії з користувачем і створення модульних програмних компонентів. Використання різноманітних бібліотек і стандартних функцій дозволяє розширювати можливості програм і підвищувати їх функціональність.

**РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++**

Процес розробки консольної програми на мові C++ дозволив глибше ознайомитися з основними концепціями програмування, які включають у себе структури даних, умовні конструкції, цикли, функції та роботу з масивами. В ході роботи ми навчилися створювати модульні програмні компоненти, які взаємодіють між собою для досягнення певних функціональних цілей.

Основною частиною практичної роботи було вирішення конкретної задачі, що вимагала використання засобів мови C++ для обробки даних, керування потоком виконання програми та взаємодії з користувачем через консольний інтерфейс. Ми також вивчили методи введення і виведення даних, що є необхідним для створення користувацьких інтерфейсів консольних програм.

**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.

Використання різноманітних тегів для тексту, таблиць, списків і форм дало можливість ефективно структурувати інформацію на сторінці. Також вивчення основ CSS дозволило застосовувати каскадні стилі для створення привабливого візуального оформлення сторінки, змінюючи колір, розмір шрифту, розташування елементів та інші параметри.

Підсумовуючи практику, вона надала мені не лише технічних навичок у роботі з HTML і CSS, але й зрозуміння важливості правильного організації контенту для досягнення максимальної ефективності та користувацької задоволеності. Цей досвід є цінним кроком у моєму професійному розвитку у 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\>

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