ВІДОКРЕМЛЕННИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ

«ОПТИКО-МЕХАНІЧНИЙ ФАХОВИЙ КОЛЕДЖ КИЇВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ІМЕНІ ТАРАСА ШЕВЧЕНКА»

Циклова комісія спеціальності «Інженерія програмного забезпечення»

**З В І Т**

**З НАВЧАЛЬНОЇ ПРАКТИКИ**

спеціальність 121 Інженерія програмного забезпечення

освітньо-кваліфікаційний рівень «фаховий молодший бакалавр»

|  |  |  |
| --- | --- | --- |
| ПІДСУМКОВА ОЦІНКА  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оцінка цифрою та прописом)  Керівники практики від коледжу:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. С. Нікітенко  (оцінка) (підпис)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т. І. Лумпова  (оцінка) (підпис)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ М. С. Ходжаєв  (оцінка) (підпис) |  | Виконав: |
| студента 2-го курсу групи ІПЗ-23 |
| Свідерка Олександра Олександровича  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) | |  | |
| *Свідерка Олександра Олександровича* | | | | |
| (П.І.Б студента.) | | | | |
| Спеціальність | *121 Інженерія програмного забезпечення* | Група | | *ІПЗ-23* |

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

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

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

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

4

Розроб.

Перевір.

Реценз.

В.Я

Н. Контр.

В.Я

Затверд.

Звіт з навчальної практики

Літ.

Аркушів

ІПЗ-22

ОМФК.0121 НП 20

Зміст

[ВСТУП 6](#_Toc169186024)

[1 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ 7](#_Toc169186025)

[2 РОЗДІЛ "ІНФОРМАТИКА" 9](#_Toc169186026)

[**2.1** **Розроблення засобами MS Word документа складної структури** 9](#_Toc169186027)

[**2.2 Розв‘язок задач за допомогою засобів табличного редактора MS Excel** 10](#_Toc169186028)

[**2.3 Виконання** **практичних завдань за допомогою засобів табличного редактора MS Access** 11](#_Toc169186029)

[3 РОЗДІЛ "ПРОГРАМУВАННЯ С++" 13](#_Toc169186030)

[**3.1 Теоретичні відомості** 13](#_Toc169186031)

[3.1.1 Загальні поняття. Елементи мови С++ - константи, змінні, операції, перетворення типів 13](#_Toc169186032)

[3.1.2 Арифметичні операції 15](#_Toc169186033)

[3.1.3 Поняття вхідного та вихідного потоку, найпростіші математичні функції 16](#_Toc169186034)

[3.1.4 Програмування лінійних та розгалужених обчислювальних процесів 17](#_Toc169186035)

[3.1.5 Програмування циклічних обчислювальних процесів 19](#_Toc169186036)

[**3.2 Практичне завдання** 20](#_Toc169186037)

[3.2.1 Завдання 20](#_Toc169186038)

[3.2.2 Постановка задачі 20](#_Toc169186039)

[3.2.3 Код програми 21](#_Toc169186040)

[3.2.4 Скріншот виконання програми 21](#_Toc169186041)

[**3.3 Виконання тестів** 21](#_Toc169186042)

[4 РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++" 22](#_Toc169186043)

[**4.1 Тема 1. Введення та виведення у базових консольних програмах** 22](#_Toc169186044)

[**4.3 Тема 3. Робота з файлами: запис та читання** 25](#_Toc169186045)

[**4.4 Тема 4. Формати файлів та їх обробка** 25](#_Toc169186046)

[**4..5 Тема 5. Системи контролю версій (VCS)** 26](#_Toc169186047)

[**4.6 Тема 6. Мова Markdown для опису проекті** 26](#_Toc169186048)

[**4.7 Розроблення ігрової програми** 28](#_Toc169186049)

[4.7.1 Код програми 28](#_Toc169186050)

[4.7.2 Скріншоти виконання програми 31](#_Toc169186051)

[5 РОЗДІЛ ”ВЕБ-ДИЗАЙН ТА HTML” 32](#_Toc169186052)

[**5.1 Мова розмітки HTML та структура Веб-сторінки** 32](#_Toc169186053)

[**5.2 Розмітка тексту за допомогою HTML. Зображення та посилання** 34](#_Toc169186054)

[**5.3 Каскадні таблиці стилів CSS та їх структура** 37](#_Toc169186055)

[**5.4 Спадковості і каскадування за допомогою CSS** 38](#_Toc169186056)

[5.5.1 Постановки задачі 39](#_Toc169186057)

[5.5.2 HTML документ 39](#_Toc169186058)

[ВИСНОВКИ 44](#_Toc169186059)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ 45](#_Toc169186060)

[ДОДАТКИ 47](#_Toc169186061)

ОМФК.0121 НП 20

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

# ВСТУП

Автор: Свідерок Олександр Олександрович

Дата:

Мета:

Перелік вивчених тем:

# 1 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ

Кабінет інформатики та інформаційно-комунікаційних технологій на­сичений різноманітною електронною технікою, яка живиться від елект­ричної мережі з напругою змінного струму 220 В. Це потребує дотриман­ня додаткових правил безпеки під час проведення навчальних занять і позаурочних заходів.

**До початку роботи:**

* з дозволу вчителя займіть своє робоче місце за комп’ютером, налаш­туйте висоту стола, стільця, підставки для ніг, кут нахилу монітора так, щоб (рис. 2.57):
* середина екрана монітора знаходилася трохи нижче горизонтальної лінії зору;
* відстань від очей до поверхні екрана становила 40-80 см, залежно від розмірів об’єктів на екрані;
* лінія від очей до центра екрана монітора була перпендикулярна до площини екрана;
* пальці рук вільно лежали на клавіатурі;
* руки утворювали в ліктьовому суглобі кут, близький до 90;
* клавіатура лежала на поверхні стола або на спеціальній полиці на відстані 10-30 см від краю і була нахилена під кутом 5-15;
* спина опиралася на спинку стільця;
* ноги опиралися на підлогу або на спеціальну підставку;
* наведіть лад на робочому столі, приберіть з нього предмети, які не потрібні для роботи;
* перевірте чистоту своїх рук, за потреби вимийте їх і витріть насухо;
* за потреби, з дозволу вчителя і тільки спеціальною серветкою протріть екран монітора, клавіатуру, килимок і мишу;
* перевірте відсутність зовнішніх пошкоджень комп’ютерного облад­нання;
* з дозволу вчителя ввімкніть ком­п’ютер.

**Під час роботи:**

* тримайте робоче місце охайним, не розміщуйте на ньому сторонніх ре­чей;
* витримуйте правильну поставу - не нахиляйтеся близько до поверхні ек­рана, не згинайтеся, тримайте руки без напруження;
* після 15-20 хв роботи або при від­чутті втоми виконайте комплекс вправ для очей і для зняття м’язово­го напруження;
* не намагайтеся самостійно усунути перебої в роботі комп’ютера, при їх виникненні негайно покличте учителя;
* не торкайтеся задніх стінок монітора та системного блока, не чіпай­те дротів живлення;
* не торкайтеся екрана монітора руками (це забруднює його і порушує антиблікове покриття);
* категорично забороняється знімати кришки корпусів пристроїв комп’ютера, самостійно без дозволу вчителя приєднувати і від’єдну­вати пристрої комп’ютера;
* акуратно вставляйте і виймайте змінні носії.

**Після закінчення роботи:**

* приберіть своє робоче місце;
* з дозволу вчителя вимкніть комп’ютер або закінчіть сеанс роботи.

# 2 РОЗДІЛ "ІНФОРМАТИКА"

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

Для створення документа складної структури в MS Word необхідно слідувати певним крокам, які забезпечать правильне оформлення та зручність використання документа.

Спочатку варто визначити структуру документа. Основні частини можуть включати титульну сторінку, зміст, основний текст із розділами та підрозділами, виноски, примітки, додатки та бібліографію.

Першим кроком у створенні документа є оформлення титульної сторінки. Для цього у вкладці "Вставка" вибирається "Титульна сторінка", де можна скористатися одним із запропонованих шаблонів або створити власний.

Далі необхідно додати зміст. Це робиться у вкладці "Посилання", де обирається пункт "Зміст". Обраний стиль змісту автоматично оновлюватиметься при додаванні нових заголовків у тексті.

Основний текст документа слід структурувати за допомогою стилів заголовків. Використовуйте стиль "Заголовок 1" для розділів і "Заголовок 2" для підрозділів. Це можна зробити на вкладці "Основне", вибираючи відповідний стиль заголовка. Текст у кожному розділі має бути відформатований із використанням відповідних стилів для абзаців, списків, цитат тощо.

Для додавання виносок та приміток скористайтеся вкладкою "Посилання". Виберіть "Вставити виноску" для додавання виносок або "Вставити примітку" для додавання приміток. Це допоможе організувати додаткову інформацію та посилання в документі.

Додатки розміщуються в кінці документа. Кожен додаток слід позначити відповідним заголовком, наприклад, "Додаток А", "Додаток B". Це дозволяє чітко відокремити додаткову інформацію від основного тексту.

Для створення бібліографії використовуйте вкладку "Посилання" і виберіть "Бібліографія". Оберіть стиль цитування та додайте необхідні джерела.

Нумерація сторінок додається через вкладку "Вставка", вибравши "Номери сторінок". Розмістіть номери внизу сторінки та налаштуйте їх так, щоб на непарних сторінках вони були праворуч, а на парних — ліворуч. Це можна зробити, обравши параметр "Різні непарні та парні сторінки".

Колонтитули додаються також через вкладку "Вставка". Оберіть стиль колонтитула, що підходить для вашого документа. Щоб вставити заголовок розділу у верхній колонтитул, використовуйте "Швидкі частини" -> "Поле" та виберіть "StyleRef" із відповідним стилем заголовка.

Для створення розділів використовуйте розриви розділів через вкладку "Макет", вибравши "Розриви" -> "Наступна сторінка". Це дозволить налаштувати макети для кожного розділу окремо, змінюючи, наприклад, орієнтацію сторінки або поля.

У підсумку, структура вашого документа може виглядати так: титульна сторінка, зміст, розділи з підрозділами, додатки та бібліографія. Дотримуючись цих кроків, ви зможете створити документ із правильно структурованими заголовками, колонтитулами, нумерацією сторінок та іншими елементами, що забезпечить зручність у користуванні та професійний вигляд.

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

MS Excel є потужним інструментом для розв'язання різноманітних задач завдяки широкому набору функцій та можливостей для обробки, аналізу і візуалізації даних. Використання Excel дозволяє ефективно виконувати математичні обчислення, створювати таблиці, будувати графіки та діаграми, а також проводити складні аналізи даних.

У процесі роботи з Excel спочатку створюється таблиця, в якій дані організовуються у вигляді рядків і стовпців. Наприклад, для фінансового аналізу можна створити таблицю з заголовками "Дата", "Опис", "Сума", "Категорія", де кожен рядок буде представляти окрему транзакцію. Це дозволяє легко впорядковувати та фільтрувати дані для подальшого аналізу.

Для автоматизації обчислень Excel надає широкий спектр функцій. Наприклад, функція SUM використовується для підсумовування чисел у діапазоні клітинок, функція AVERAGE - для обчислення середнього значення, а функція IF - для логічних перевірок. Завдяки цим функціям можна легко розв'язувати різноманітні задачі, від простих обчислень до складних аналітичних задач.

Графіки та діаграми в Excel допомагають візуалізувати дані, що сприяє кращому розумінню інформації та прийняттю обґрунтованих рішень. Наприклад, лінійні графіки можуть показувати тренди у часі, стовпчасті діаграми - порівняння між різними категоріями, а кругові діаграми - структуру розподілу. Для створення графіка достатньо виділити необхідні дані і вибрати відповідний тип графіка у вкладці "Вставка".

Для більш глибокого аналізу даних в Excel використовуються зведені таблиці. Вони дозволяють групувати та підсумовувати дані, створювати багаторівневі звіти і швидко отримувати відповіді на конкретні питання. Наприклад, зведена таблиця може показувати сумарні продажі по регіонах або середній дохід по категоріях товарів.

Інструмент "Пошук рішення" (Solver) в Excel використовується для розв'язання оптимізаційних задач. Він дозволяє знаходити оптимальні значення змінних, які задовольняють задані обмеження. Це особливо корисно для задач, де необхідно мінімізувати витрати або максимізувати прибуток за наявності певних ресурсів і обмежень.

Завдяки всім цим можливостям MS Excel є незамінним інструментом для роботи з даними у багатьох сферах діяльності. Його застосування дозволяє ефективно розв'язувати задачі, аналізувати великі обсяги інформації та приймати обґрунтовані рішення на основі отриманих результатів.

## 

## **2.3 Виконання** **практичних завдань за допомогою засобів табличного редактора MS Access**

MS Access є потужним інструментом для створення та управління базами даних, що дозволяє виконувати різноманітні практичні завдання, пов'язані з обробкою, збереженням і аналізом даних. Його широкі можливості та інтуїтивний інтерфейс роблять Access ідеальним вибором для створення реляційних баз даних, автоматизації процесів та генерації звітів.

Основою роботи в MS Access є створення таблиць, які використовуються для зберігання даних. Кожна таблиця складається з полів, де кожне поле представляє певний тип даних, наприклад, текст, числові значення, дати або логічні значення. Таблиці можуть бути пов'язані між собою за допомогою ключів, що дозволяє створювати реляційні бази даних. Це забезпечує ефективне зберігання інформації та зменшує дублювання даних.

Для введення та перегляду даних в таблицях Access надає форми. Форми дозволяють створювати зручні інтерфейси для користувачів, спрощуючи процес введення даних та забезпечуючи їх коректність. Наприклад, за допомогою форми можна ввести інформацію про нових клієнтів, додаючи їх дані до відповідної таблиці бази даних.

Запити є одним із ключових інструментів MS Access, що дозволяє здійснювати вибірку, фільтрацію та сортування даних з таблиць. За допомогою запитів можна отримати необхідні дані, використовуючи різноманітні критерії. Наприклад, запит може знайти всіх клієнтів з певного міста або відобразити всі замовлення за конкретний період. Запити можуть бути простими або складними, включаючи об'єднання даних з кількох таблиць.

MS Access також надає можливості для створення звітів, які дозволяють форматувати і друкувати дані в зручному вигляді. Звіти використовуються для презентації даних користувачам і можуть включати текст, графіку та діаграми. Наприклад, звіт може містити інформацію про місячні продажі, відображаючи їх у вигляді таблиці та графіка.

Крім того, Access підтримує макроси та модулі, які дозволяють автоматизувати рутинні завдання і розширити функціональність бази даних. Макроси є набором команд, що виконуються послідовно, наприклад, для автоматичного оновлення даних або запуску звітів. Модулі використовують мову програмування VBA (Visual Basic for Applications) для створення більш складних програмних рішень.

Таким чином, MS Access є потужним інструментом для розв'язання практичних завдань, пов'язаних зі створенням та управлінням базами даних. Його можливості дозволяють ефективно організовувати дані, забезпечувати їх цілісність, автоматизувати процеси і створювати інформативні звіти. Завдяки MS Access можна легко створити базу даних, яка задовольняє потреби організації, забезпечуючи зручний доступ до інформації та підтримку прийняття обґрунтованих рішень.

# 3 РОЗДІЛ "ПРОГРАМУВАННЯ С++"

## **3.1 Теоретичні відомості**

### 3.1.1 Загальні поняття. Елементи мови С++ - константи, змінні, операції, перетворення типів

С++ є мовою програмування загального призначення, яка широко застосовується для розробки програмного забезпечення завдяки своїй потужності та гнучкості. До основних елементів мови С++ входять константи, змінні, операції та перетворення типів.

Константи-значення які не змінюються під час виконання програми. Константи оголошуються за допомогою слова *const*. Воно забезпечує їх незмінність. Константи бувають різних типів:

* числові (цілі та дробові числа);

const int MAX\_USERS = 10;

const double PI = 3.14159;

* символьні (окремі символи, які беруться в лапки);

const char NEWLINE = '\n';

* рядкові (послідовність символів, які беруться в лапки).

const char\* GREETING = "Hello, World!";

Змінні-ділянки пам’яті, які іменуються та зберігають дані. Їх значення можуть змінюватись під час виконання програми. Змінні бувають локальними (ті, які доступні в межах однієї функції) та глобальними (ті, які доступні в усьому коді).

Змінні поділяються на прості типи та складні. Прості типи включають в себе *int, float, double, char, bool*:

int age = 18;

double salary = 12345.67;

char grade = 'A';

bool is\_active = true;

Складні ж включають в себе *масиви, структури* та *класи*:

int scores[5] = {90, 85, 88, 92, 79};

struct Person {

std::string name;

int age;

};

Person john = {"John Doe", 30};

Операції в С++ призначені для того, щоб виконувати дії над змінними та константами. Вони включають в себе:

* арифметичні операції (додавання, віднімання, множення, ділення, модуль)

int sum = 10 + 5; // додавання

int difference = 10 - 5; // віднімання

int product = 10 \* 5; // множення

int quotient = 10 / 5; // ділення

int remainder = 10 % 3; // модуль (остання від ділення)

* операції для порівняння (перевірка рівності/нерівності, більше чи менше, тощо)

bool isEqual = (a == b); // перевірка рівності

bool isNotEqual = (a != b); // перевірка нерівності

bool isGreater = (a > b); // перевірка, що a більше b

логічні операції (логічне І/АБО/НЕ)

bool andOperation = (a > b) && (c < d);

bool orOperation = (a > b) || (c < d);

bool notOperation = !(a > b);

* операції присвоєння значення змінній:

int x = 10;

x += 5; // еквівалентно x = x + 5;

x \*= 2; // еквівалентно x = x \* 2;

* Перетворення типів - це механізм в C++, який дозволяє змінювати тип даних змінної. Це може бути корисно в різних ситуаціях, наприклад, коли потрібно присвоїти значення одного типу змінної іншій змінній з іншим типом, або коли необхідно виконати операцію над значеннями різних типів.неявне (те, яке виконується компілятором)

**1. Неявне перетворення:**

* Виконується компілятором автоматично.
* Зазвичай використовується для зручності програміста.
* Приклади:
* Присвоєння значення змінної іншого типу:

int i = 42;

double d = i; // i неявно перетворюється в double

**2.Явне перетворення:**

Програміст явно вказує на необхідність перетворення.

Використовуються оператори cast:

* + static\_cast: для явного перетворення типів.

double d = 9.99;

int i = static\_cast<int>(d); // явне перетворення double в int

dynamic\_cast: для безпечного перетворення до динамічних типів.

const\_cast: для зняття константного кваліфікатора.

reinterpret\_cast: для низькорівневого доступу до пам'яті, ризиковане перетворення, яке може призвести до помилок.

### 3.1.2 Арифметичні операції

Арифметичні операції - це базові операції в C++, які використовуються для роботи з числовими даними.

**Доступні операції:**

* Додавання (+)
* Віднімання (-)
* Множення (\*)
* Ділення (/)
* Отримання залишку від ділення (%)

**Важливі моменти:**

* Операції ділення цілих чисел завжди повертають ціле число.
* Операцію % можна використовувати лише з цілими числами.
* Пріоритет операцій визначає порядок їх виконання.
* Використовуйте дужки для зміни пріоритету.

**Приклади:**

int a = 5;

int b = 3;

int sum = a + b; // sum = 8

int difference = a - b; // difference = 2

int product = a \* b; // product = 15

int quotient = a / b; // quotient = 5

int remainder = a % b; // remainder = 1

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

Вхідний та вихідний потік - це фундаментальні концепції в C++, що описують процес введення та виведення даних з програми. Робота з потоками здійснюється за допомогою бібліотек iostream та iomanip.

**Вхідний потік**

Вхідний потік використовується для **введення даних** з зовнішніх джерел, таких як клавіатура або файли, до програми. Основним класом для роботи з вхідними потоками є std::istream.

**Приклад:**

#include <iostream>

int main() {

int number;

std::cout << "Введіть число: ";

std::cin >> number; // Введення даних з клавіатури

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

return 0;

}

**Вихідний потік**

Вихідний потік використовується для **виведення даних** з програми на зовнішні пристрої або файли. Основним класом для роботи з вихідними потоками є std::ostream.

**Приклад:**

#include <iostream>

int main() {

int number = 10;

std::cout << "Число: " << number << std::endl; // Виведення даних на екран

return 0;

}

**Функції для роботи з потоками**

Бібліотека iostream пропонує безліч функцій для роботи з потоками, наприклад:

* std::cout: використовується для виведення даних на стандартний вихідний потік (зазвичай екран).
* std::cin: використовується для введення даних зі стандартного вхідного потоку (зазвичай клавіатура).
* std::getline: використовується для введення рядка тексту з потоку.
* std::ofstream: використовується для створення та роботи з файлами виводу.
* std::ifstream: використовується для відкриття та роботи з файлами вводу.

**Форматування виводу**

Бібліотека iomanip надає інструменти для форматування виводу даних, наприклад:

* std::setw: встановлює ширину поля для наступного виведеного значення.
* std::setprecision: встановлює точність для виведення чисел з комою.
* std::fixed: використовується для фіксованого точкового запису чисел.
* std::scientific: використовується для наукового запису чисел.

**Висновок**

Вхідний та вихідний потік є важливими інструментами для взаємодії з користувачем та роботи з даними в програмах на C++. Розуміння цих концепцій та використання бібліотек iostream та iomanip є ключовими для написання ефективних та зручних програм.

### 

### 3.1.4 Програмування лінійних та розгалужених обчислювальних процесів

Лінійні та розгалужені обчислювальні процеси - це фундаментальні концепції програмування, які описують логіку роботи програм. Розуміння цих принципів є ключовим для написання чітких, ефективних та гнучких програм.

**Лінійні обчислювальні процеси**

Лінійні обчислювальні процеси виконуються **послідовно**, згідно з чітко визначеною послідовністю інструкцій. В таких програмах відсутні умовні оператори або цикли, а код виконується крок за кроком, без "розгалужень".

**Приклад:**

#include <iostream>

int main() {

int a = 5;

int b = 3;

int sum = a + b; // Просте лінійне обчислення

std::cout << "Сума: " << sum << std::endl;

return 0;

}

**Розгалужені обчислювальні процеси**

На відміну від лінійних, розгалужені обчислювальні процеси забезпечують **виконання різних дій** залежно від виконання певних **умов**. Це досягається за допомогою операторів умовного вибору, таких як if, else, else if, а також операторів switch.

#include <iostream>

int main() {

int number;

std::cout << "Введіть число: ";

std::cin >> number;

if (number % 2 == 0) {

std::cout << "Число парне." << std::endl;

} else {

std::cout << "Число непарне." << std::endl;

}

return 0;

}

**Переваги використання лінійних та розгалужених процесів**

* **Лінійні процеси:**
  + Прості та зрозумілі.
  + Ефективні для виконання послідовних задач.
* **Розгалужені процеси:**
  + Забезпечують гнучкість та динаміку програм.
  + Дозволяють реалізовувати складні алгоритми.
  + Роботу з умовними ситуаціями.

**Висновок**

Лінійні та розгалужені обчислювальні процеси є двома базовими типами логіки, які використовуються для створення програм різної складності. Розуміння цих концепцій необхідне для будь-якого програміста, який прагне писати чіткий, ефективний та надійний код.

### 3.1.5 Програмування циклічних обчислювальних процесів

Такі процеси дозволяють програмі повторювати певні дії до досягнення заданої умови. У мові програмування C++ цикли поділяються на три основні типи: *for, while* та *do-while*.

Перши тип це цикл for. Зазвичай він використовується в тих випадках, коли заздалегідь відомо скільки разів потрібно виконати набір інструкцій. Цикл For складається з трьох частин: ініціалізації, умови та модифікації.

for (ініціалізація; умова; модифікація) {

// Тіло циклу

}

Наступний тип-цикл while. Цей цикл використовують коли заздалегідь невідома і залежить від умови, яка перевіряється перед кожною ітерацією. Якщо умова є істинною, тіло циклу виконується, інакше цикл завершується.

while (умова) {

// Тіло циклу

}

І ще один дуже важливий тип циклів-це do-while. Цикл do-while схожий на цикл while, але з однією важливою відмінністю: умова перевіряється після виконання тіла циклу. Таким чином, тіло циклу гарантовано виконується хоча б один раз.

do {

// Тіло циклу

} while (умова);

Прикладом для демонстрації роботи усіх трьох типів циклів може слугувати елементарна програма виводу чисел від 1 до 5. Ось коди цих програм, у яких задіяні різни види циклів:

for (int i = 1; i <= 5; ++i) std::cout << i << " ";

int i = 1;

while (i <= 5) {

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

++i;

}

int i = 1;

do {

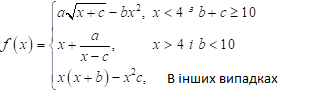
std::cout << i << " ";

++i;

} while (i <= 5);

## **3.2 Практичне завдання**

### 3.2.1 Завдання



### 3.2.2 Постановка задачі

Метою розробки задачі є вивчення нових аспектів у роботі з мовою програмування С++, а саме ті, що використовуються при написанні програми для розв’язання прикладів, перевірки вхідних значень і повернення результату.

Вхідні дані

Вихідні дані

Блок-схема алгоритму роботи програми представлена на рис. ххх

Блок-схема

Рисунок хх - Блок-схема алгоритму роботи програми

### 3.2.3 Код програми

### 3.2.4 Скріншот виконання програми

Скріншот виконання програми представлена на рис. ххх

Скріншот

Рисунок хх - Скріншот виконання програми

## **3.3 Виконання тестів**

# 

# 4 РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++"

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

У базових консольних програмах C++ об'єкт `cout` використовується для виведення даних на заданий пристрій. Синтаксис `cout << data` дозволяє виводити змінні, константи або вирази. Для форматування виведення даних використовуються різні маніпулятори, такі як `hex` та `oct`, які задають виведення чисел у 16-ковій та 8-ковій системах відповідно. Маніпулятори `setw()`, `setfill()`, і `setprecision()` змінюють стан об’єкта `cout`: `setw()` задає ширину поля для виведення, `setfill()` заповнює незаповнені позиції встановленим символом, а `setprecision(n)` визначає кількість знаків після коми.

Маніпулятори `setiosflags` і `resetiosflags` встановлюють або скидають певні глобальні прапори, які використовує клас `iostream` для визначення поведінки при введенні та виведенні. Наприклад, `setiosflags()` встановлює зазначені прапори, а `resetiosflags()` їх очищає.

У класичному C для виведення даних використовується функція `printf`, яка також доступна у C++. Синтаксис `printf(char \*format, ...)` дозволяє виводити дані з використанням специфікаторів формату. Форматний рядок складається з символів, що виводяться без змін, і спеціальних символів формату, які задають перетворення даних. Специфікатори включають `%d` або `%i` для десяткового цілого зі знаком, `%u` для десяткового цілого без знака, `%x` для шістнадцятирічного цілого без знака, `%f` для чисел з плаваючою точкою, `%e` для експоненційного запису чисел, `%c` для символів, `%s` для рядків, і `%p` для вказівників. Символи `L` або `l` використовуються для довгих типів даних, таких як `long double` або `long int`.

Для введення даних у C++ використовується об’єкт `cin`, синтаксис якого `cin >> змінна` дозволяє вводити змінні. При введенні рядків кожне слово вводиться окремо, а роздільником між ними є пробіл або Enter.

Функція `scanf`, яка також доступна у C, дозволяє вводити дані за допомогою форматних рядків, аналогічних до `printf`. Перед іменами змінних у `scanf` потрібно ставити символ `&`, який означає "взяти адресу" змінної для її зчитування.

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

**Масив** - це колекція однотипних даних, які зберігаються в суцільній пам'яті. Властивості масивів:

* **Тип елементів:** Визначає тип даних, які містяться в масиві (int, double, char тощо).
* **Назва:** Ідентифікатор масиву в програмі.
* **Розмірність:** Визначає кількість елементів в масиві.
* **Індекси:** Цілі числа, які використовуються для доступу до конкретних елементів масиву.

**В C++ індекси масивів починаються з 0.**

**Доступ до елементів масиву:**

int myArray[5]; // Оголошення масиву з 5 елементів типу int

myArray[2] = 10; // Запис значення 10 в третій елемент (індекс 2)

int value = myArray[3]; // Зчитування значення з четвертого елемента

**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.

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).

**Стек** - це лінійна структура даних, яка використовує принцип **LIFO** (Last In, First Out - "Останній зайшов, перший вийшов"). Основні операції зі стеком:

* **push()**: Вставка елемента в кінець стеку.
* **pop()**: Вилучення елемента з кінця стеку.
* **top()**: Отримання доступу до верхнього елемента стеку (без видалення).

#include <stack>

int main() {

std::stack<int> myStack;

myStack.push(1); // Вставка елемента 1

myStack.push(2); // Вставка елемента 2

myStack.push(3); // Вставка елемента 3

std::cout << "Top element: " << myStack.top() << std::endl; // Виведення верхнього елемента

myStack.pop(); // Видалення верхнього елемента

while (!myStack.empty()) {

std::cout << myStack.top() << " ";

myStack.pop();

}

return 0;

}

Прообразом цієї структури є чергу людей у магазині: того, що став першим, буде обслужено першим. Якщо розглядати чергу щодо доступу до даних, то вона реалізує принцип: «Першим зайшов, перший вийшов» (англій­ською First In First Out, FIFO). Інакше кажучи, після додавання нового елементу всі елементи, які було додано до цього, потрібно вилучити до того, як новий елемент буде вилучено.

Основні операції з чергою:

* вставлення елемента в кінець черги;
* вилучення елемента з початку черги.

Контейнер queue безпосередньо втілює принцип: «Першим зайшов, перший вийшов» (англій­ською First In First Out, FIFO):

**Асоціативний масив** (словник в інших мовах програмування) — це вбудована структура даних для збереження даних у форматі ключ-значення.

Цю структура даних легко уявити як шафу з підписаними шухлядами. Усі дані зберігають у шухлядах. За назвою можна легко знайти шухляду і взяти з неї те значення, яке там розташовано.

На відміну від звичайних шаф, в асоціативний масив будь-коли можна додати нові «шухляди» або вилучити вже наявні.

**Множина (Set)** - це контейнер в C++, який використовується для зберігання **унікальних значень**. Він ґрунтується на математичному понятті множини, де кожен елемент може бути присутнім лише один раз.

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

Робота з файлами у мові програмування C++ дозволяє програмістам зберігати та отримувати дані з файлової системи комп'ютера. Це може бути корисно для збереження конфігураційних даних, роботи зі структурованими даними, збереження результатів обчислень та багато іншого.

Для роботи з файлами у C++ використовуються два основних класи: ifstream і ofstream. Перший використовується для читання з файлу, а другий - для запису у файл. Вони базуються на базовому класі fstream, який також можна використовувати як універсальний інструмент для роботи з файлами.

Щоб використовувати файли у програмі, спочатку треба їх відкрити. Це робиться за допомогою конструктора класу ifstream або ofstream, якому передається шлях до файлу. Якщо файл існує, його відкриють для роботи, а якщо ні, то він буде створений.

Після відкриття файлу, програма може зчитувати дані з нього, записувати дані у нього або виконувати будь-які інші операції, пов'язані з файлом. Після завершення роботи з файлом його слід закрити за допомогою методу close().

Приклади використання:

* Для читання з файлу:

ifstream inputFile("example.txt");

* Для запису в файл:

ofstream outputFile("example.txt");

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

Текстові файли:

Текстові файли містять дані у формі тексту, що може бути зрозумілим для людини. Це може бути рядок символів, числа або інші дані, які можна зчитати та редагувати звичайним текстовим редактором.

Дані у текстових файлах зазвичай розділяються символами пробіла, комою, крапкою з комою або іншими розділовими знаками.

Бінарні файли:

Бінарні файли містять дані у бінарному форматі, який не призначений для прямого читання людиною. Дані зазвичай зберігаються у вигляді послідовності бітів, що представляють числа, структури даних тощо.

Бінарні файли зазвичай потребують спеціальної обробки для читання та запису, оскільки дані можуть бути представлені у складних форматах.

У мові програмування C++ для читання та запису файлів використовуються класи ifstream для читання та ofstream для запису. Крім того, для обробки бінарних файлів можна використовувати функції fread та fwrite.

Іноді для обробки певних форматів файлів (наприклад, JSON, XML) використовують спеціалізовані бібліотеки, які надають зручний інтерфейс для читання та запису даних у цих форматах.

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

Система контролю версій (VCS) - це програмний інструмент, який дозволяє розробникам зберігати історію змін у вихідному коді свого програмного забезпечення. Основна мета VCS полягає в тому, щоб забезпечити зручну та безпечну роботу з кодом для розробників та команд розробників.

Використання систем контролю версій дозволяє:

* Зберігати історію змін: Кожне збереження (commit) в системі контролю версій записує зміни, внесені у вихідний код. Це дозволяє розробникам переглядати, аналізувати та повертатися до попередніх версій коду у разі потреби.
* Спільно працювати над проектом: VCS дозволяє розробникам працювати над проектом одночасно та координувати свою роботу. Кожен розробник може вносити свої зміни, а система автоматично об'єднує їх.
* Експериментувати та розгалужувати розробку: Розгалуження розробки (branching) дозволяє розробникам експериментувати з новими функціями або розвивати окремі функціональність у відокремлених гілках, не впливаючи на основний код проекту.

Варіанти систем контролю версій включають:

* Локальні системи контролю версій: Розробники працюють локально на своїх комп'ютерах і зберігають історію змін лише на своєму пристрої. Це простий підхід, але він не підходить для спільної роботи в команді.
* Централізовані системи контролю версій: Всі зміни зберігаються на центральному сервері, і розробники працюють з віддаленими копіями коду. Цей підхід підходить для команд, але може стати однією точкою відмови.
* Розподілені системи контролю версій: Кожен розробник має повну копію історії проекту. Це дозволяє працювати навіть без підключення до мережі та має більшу стійкість до відмов серверів.

Важливо також відзначити, що історія розвитку систем контролю версій проходить три покоління, починаючи з простих локальних систем і закінчуючи розподіленими системами з розгалуженням розробки та можливістю злиття гілок.

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

**Markdown** — полегшена мова розмітки даних, яку створено з ухилом на прочитність та зручність у публікації з подальшим перетворенням її на structurally valid XHTML або HTML. Чимало ідей для мови позичено з існуючих домовленостей у розмітці тексту в електронних листах. Першу реалізацію Markdown написано Грубером на Perl, але з часом з'явились багато реалізацій від сторонніх розробників. Реалізація на Perl розповсюджується за ліцензією BSD.

Markdown - це легка мова розмітки для форматування тексту.

**Переваги:**

* Простота
* Універсальність
* Чіткість
* Перетворення в HTML

**Функції:**

* Заголовки: #, ##, ###...
* Жирний шрифт: \*\*текст\*\*
* Курсив: \*текст\*
* Списки: \*, -, 1.
* Цитування: > текст
* Посилання: [текст](URL)
* Код: мова\_програмування \n код \n

**Використання:**

* GitHub
* Reddit
* Stack Overflow
* Блоги
* Нотатки

**Markdown - це зручний інструмент для створення чіткого, структурованого та читабельного тексту.**

* Стилізація тексту

|  |  |  |  |
| --- | --- | --- | --- |
| **Стиль** | **Синтаксис** | **Приклад** | **Вихідний результат** |
| Жирний | \*\* \*\* або \_\_ \_\_ | \*\*This is bold text\*\* | **This is bold text** |
| Курсивний | \* \* або \_ \_ | \_This text is italicized\_ | This text is italicized |
| Закреслення | ~~ ~~ | ~~This was mistaken text~~ | ~~This was mistaken text~~ |
| Жирний і вкладений курсив | \*\* \*\* and \_ \_ | \*\*This text is \_extremely\_ important\*\* | **This text is extremely**  **important** |
| Усі жирні та курсивні | \*\*\* \*\*\* | \*\*\*All this text is important\*\*\* | ***All this text is important*** |

* Посилання

Користувач може створити вбудоване посилання, помістивши текст посилання в квадратні дужки [ ], а потім увімкнувши URL-адресу в дужки ( ), або скористатися комбінацією клавіш Command+K, щоб створити посилання. Вибравши текст, можна вставити URL-адресу з буфера обміну, щоб автоматично створити посилання з виділеного тексту.

Також можна створити гіперпосилання Markdown, виділивши текст і скориставшись комбінацією клавіш Command+V. Якщо користувач хоче замінити текст посиланням, слід скористатися комбінацією клавіш Command+Shift+V.

Изображение выглядит как текст, Шрифт, снимок экрана, белый

Автоматически созданное описание

Зображення

Користувач може відобразити зображення, додавши ! і альтернативний текст в [ ]. Альтернативний текст – це короткий текстовий еквівалент інформації на зображенні. Потім слід помістити посилання на зображення в дужки ().

* Список

Невпорядкований список можна створити, поставивши перед одним або декількома рядками тексту -, \* або +.

Щоб упорядкувати список, перед кожним рядком слід додати номер.

## **4.7 Розроблення ігрової програми**

### 4.7.1 Код програми

#include <iostream>

#include <vector>

using namespace std;

// Функція для відображення ігрової дошки

void drawBoard(const vector<vector<string>>& board)

{

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

{

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

{

cout << board[row][col];

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

}

cout << endl;

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

}

}

// Функція для перевірки переможця

string checkWinner(const vector<vector<string>>& board)

{

// Перевірка рядків та стовпців

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

{

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

return board[i][0];

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

return board[0][i];

}

// Перевірка діагоналей

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

return board[0][0];

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

return board[0][2];

// Якщо переможця немає

return "";

}

int main()

{

vector<vector<string>> board(3, vector<string>(3, " "));

string currentPlayer = "X";

while (true)

{

drawBoard(board);

// Отримання ходу гравця

int row, col;

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

cin >> row >> col;

// Валідація введення

if (row < 0 || row >= 3 || col < 0 || col >= 3 || board[row][col] != " ") {

cout << "Invalid move. Please try again." << endl;

continue;

}

// Оновлення дошки ходом гравця

board[row][col] = currentPlayer;

// Перевірка переможця

string winner = checkWinner(board);

if (winner != "")

{

drawBoard(board);

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

break;

}

// Перевірка нічии

bool isDraw = true;

for (const auto& row : board) {

for (const auto& cell : row) {

if (cell == " ")

{

isDraw = false;

break;

}

}

}

if (isDraw)

{

drawBoard(board);

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

break;

}

// Зміна гравця

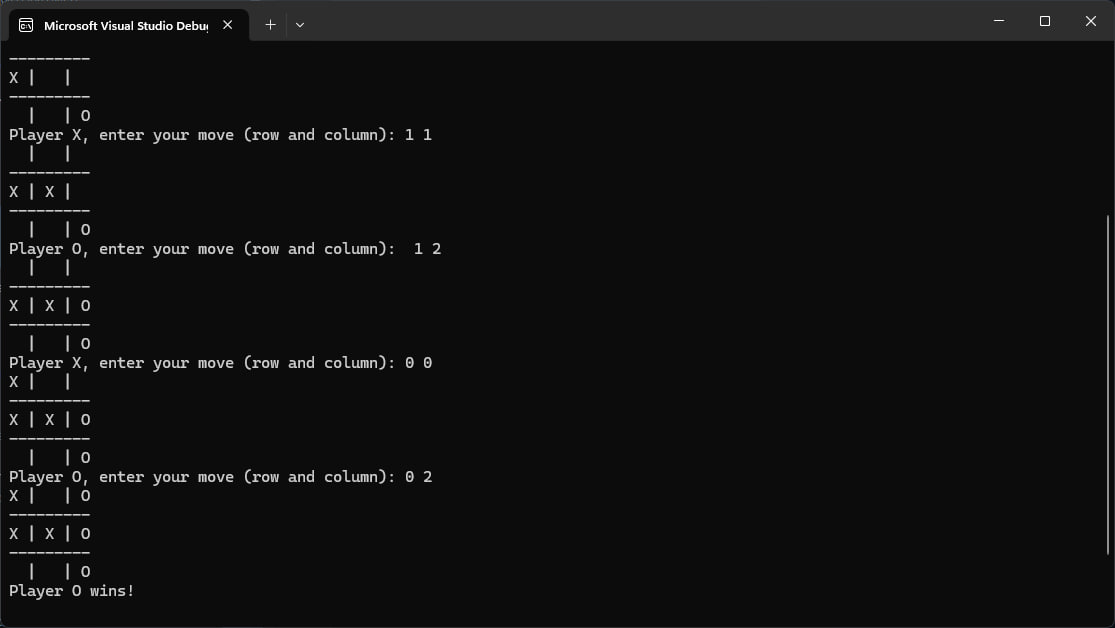
currentPlayer = (currentPlayer == "X") ? "O" : "X";

}

return 0;

}

### 4.7.2 Скріншоти виконання програми



# 5 РОЗДІЛ ”ВЕБ-ДИЗАЙН ТА HTML”

HTML (від англ. HyperText Markup Language – «мова гіпертекстової розмітки») – стандартна мова розмітки документів в мережі Інтернет. Біль- шість веб-сторінок містять опис розмітки на мові HTML (або XHTML). Мова HTML інтерпретується браузерами і відображається у вигляді документа в зручній для людини формі.

Мова HTML є додатком SGML (стандартної узагальненої мови роз- мітки) і відповідає міжнародному стандарту ISO 8879.

У всесвітній павутині HTML-сторінки, як правило, передаються брау- зерам від сервера по протоколах HTTP або HTTPS, у вигляді простого тексту або з використанням шифрування.

Мова HTML була розробленою британським вченим Тімом Бернерс-Лі приблизно в 1986-1991 роках в стінах Європейського Центру ядерних дослі- джень в Женеві (Швейцарія). Елементами є структурні і семантичні елементи – дескриптори. Дескриптори також часто називають «тегами». Також, в HTML внесена підтримка гіпертексту.

Таким чином, веб-сторінки створюються за допомогою мови розмітки гіпертексту HTML. Опис веб-сторінок міститься в HTML-програмі, який збе- рігається у звичайному текстовому файлі з розширенням htm чи html. Про- грами мовою HTML містять інструкції (коди), що називаються тегами. Всі теги мови HTML виділяються символами-обмежувачами < і >, між якими за- писується ідентифікатор (ім’я) тега і, можливо, його параметри. Більшість те- гів HTML використовується попарно, тобто для окремого тега (відкриваючо- го) в документі є відповідний закриваючий тег, який записується так само, як і відкриваючий, але з символом / перед іменем тега (закриваючі теги не вико- ристовують параметри).

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

HTML: Основні поняття та функціональність

Документ HTML оброблюється браузером і відображається на екрані у зручному для користувача вигляді. HTML надає можливості для:

- Створення структурованого документу шляхом маркування тексту, включаючи заголовки, абзаци, списки, таблиці, цитати тощо.

- Створення зв’язків між веб-сторінками за допомогою гіперпосилань.

- Створення інтерактивних форм.

- Включення зображень, аудіо, відео та інших медіа елементів.

Весь контент описується за допомогою спеціальних правил розмітки. Розмітка в HTML складається з чотирьох основних компонентів:

1. Елементи.

2. Базові типи даних (визначають тип вмісту атрибутів, такі як символьні дані, числа тощо).

3. Символьні мнемоніки.

4. Декларація типу документа (визначає версію HTML).

Мнемоніки — це спеціальні позначення (псевдоніми), які використовуються для вставки певних символів.

Спробуйте нові продукти:

В загальному, структура HTML-документа є такою:

*Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание*

Парою тегів <HTML>...</HTML> визначається початок і кінець документа. Розділ документа **HEAD** визначає його заголовок, розміщений між парою тегів <TITLE> і </TITLE>, а також мета-теги. Тег <META> містить різнорідну інформацію для броузерів, яка не відображається у вікні. Тег STYLE> пов’язаний з використанням таблиць стилів. Тег <SCRIPT> використовується для запису сценаріїв.

Елементи являють собою базові компоненти розмітки HTML. Кожен елемент має назву і задається за допомогою тегів, що записуються у кутових дужках. Вміст (контент) елемента записують між відкриваючим і закриваючим тегами. Також елементи можуть мати атрибути (властивості), які задаються всередині відкриваючого тегу.

Значення атрибута записують у подвійних лапках. Деякі елементи, можуть не мати атрибутів, інші ж мають обов'язкові атрибути, без яких вони не зможуть виконувати свої функції. Існують також елементи, які не мають вмісту (контенту) і задаються за допомогою одиночних тегів.

За своїм призначенням та дією елементи HTML можна розділити на три великі категорії: структурні, презентаційні та елементи для інтерактивної взаємодії з користувачем.

Структурні. Елементи структурної розмітки застосовуються задля опису семантики тексту та задання структури документу. В свою чергу, в залежності від змісту документа, структурні елементи поділяються на: текст, списки, таблиці, гіперпосилання та додаткові об'єкти (зображення, аплети та ін). Вони не

зазначають ніякого спеціального (візуального) відтворення тексту, проте більшість браузерів мають стандартні стилі форматування для кожного елемента. Для подальшого стилізування тексту рекомендується використовувати Каскадні таблиці стилів (CSS).

Презентаційні. Елементи презентаційної (візуальної) розмітки застосовуються задля опису зовнішнього вигляду документа та не зазначають при цьому функцій тексту, до якого вони застосовуються. Більшість з цих елементів є такими, що не рекомендується застосовувати у розмітці. В залежності від свого змісту, елементи презентаційної розмітки поділяють на: елементи додавання каскадних таблиць стилів, візуальні елементи тексту (шрифти, кольори, насиченість та нахил щрифту, лінії та інші) та фрейми.

Інтерактивної взаємодії. Елементи інтерактивної взаємодії дозволяють користувачам впливати на вміст або вигляд документу. Вони поділяються на два види: форми для введення данних користувачами та скрипти для активних документів.

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

Почнемо з найпростішого тега <p>, за допомогою якого створюються абзаци. За замовчуванням абзаци починаються з нового рядка і мають вертикальні відступи, якими можна управляти за допомогою стилів.

Заголовки і підзаголовки

Для створення структури великих текстів зазвичай використовуються заголовки. У текстових редакторах є можливість виділити частину тексту  як заголовок. В мові HTML для виділення заголовків передбачено ціле сімейство тегів: від <h1> до <h6>. Тег <h1> (headline 1) позначає найважливіший заголовок (заголовок верхнього рівня), а тег <h6> позначає підзаголовок самого нижнього рівня.

На практиці нечасто зустрічаються тексти, в яких зустрічаються підзаголовки нижче третього рівня. Тому самими часто використовуваними тегами заголовків є: <h1>, <h2> і <h3>. Варто відзначити, що пошукові системи надають особливе значення заголовкам.

Маркований (ненумерований) список

Списки часто використовуються в різних документах. Іноді, щоб зробити список, користувач просто нумерує рядки тексту. Такий підхід не є хорошим, оскільки в документі відсутня логічна сутність «список». У HTML існує сімейство тегів для створення списків: невпорядкованих, упорядкованих і списків визначень.

Невпорядковані (або марковані) списки створюються за допомогою тега <ul> (unordered list), який може містити всередині себе теги <li> (list item), що позначають «елемент списку».

Впорядкований (нумерований) список

Впорядкований список створюється за допомогою тега <ol> (ordered list), який містить всередині себе теги <li>. Якщо елементи неупорядкованого списку за замовчуванням позначаються маркерами, то елементи впорядкованого списку нумеруються. Нумеровані або впорядковані списки використовуються для елементів, які повинні йти в певному порядку, наприклад, покрокові інструкції. Вони функціонують так само, як і марковані списки, але визначаються за допомогою тега <ol>. Замість маркерів браузер автоматично додає номери перед елементами списку, тому вам не потрібно проставляти їх вручну. Це дозволяє легко змінювати порядок елементів списку без повторної нумерації.

Для зміни маркерів і номерів списків можна використовувати властивість list-style-type у таблиці стилів CSS. Наприклад, для маркованих списків можна змінити форму маркера з кола на квадрат або незаштриховане коло, додати власне зображення або прибрати маркер повністю. Для нумерованих списків можна замінити традиційні арабські цифри римськими (I., II., III. або i., ii., iii.), літерами (A., B., C. або a., b., c.) та іншими типами нумерації.

Важливість. Теги <strong> и <b>

Тег <strong> позначає важливість зазначеного тексту, тоді як тег <b> (bold) використовується для виділення тексту без додавання йому особливої важливості. Найкраще відмінності між цими тегами відчують користувачі зі спеціальними налаштуваннями ОС, особливо ті, хто має проблеми з зором. Коли увімкнена функція читання тексту, програма буде інтонаційно виділяти слова, позначені тегом <strong>. Так само відбувається і з тегами <em> (emphasis) та <i>. Тег <em> також виділятиметься інтонацією під час читання. Візуально теги <strong> і <b> виглядають однаково, роблячи текст жирним. В HTML5 тег <b> отримав новий сенс. Раніше він просто робив текст напівжирним і використовувався лише для візуального форматування..

Переноси та розділювачі. Теги <br> и <hr>

Іноді виникає потреба вставити перенесення рядка в тексті, не створюючи при цьому абзацу. Це особливо корисно при розмітці віршів або текстів пісень. У HTML для цього передбачено одиночний тег <br> (break). Однак, цей тег іноді використовують для розбиття тексту на "псевдоабзаци", що є поганою практикою. Для розмітки абзаців використовуйте тег <p>.

Одиночний тег <hr> (horizontal rule) використовується для створення горизонтальної лінії-розділювача.

Цитати

У HTML є кілька тегів для позначення цитат:

- <blockquote> використовується для виділення довгих цитат, які можуть складатися з кількох абзаців. Цей тег виділяє цитату як окремий блок тексту з відступами.

- <q> (quote) призначений для виділення коротких цитат у тексті. Текст всередині цього тега автоматично обрамляється лапками.

- <cite> використовується для виділення джерела цитати – назви твору, але не автора цитати.

Посилання

Посилання зазвичай виглядає як підкреслений текст, при натисканні на який ви переходите на іншу сторінку, відкриваєте зображення або починаєте завантаження файлу. Якщо уявити Інтернет як величезну мережу з безліччю вузлів, то посилання є нитками, що з'єднують всі ці вузли. Посилання створюються за допомогою дуже простого і короткого тегу <a>. Наприклад:

<a href="URL">Текст посилання</a>

Адреси можуть бути різними: абсолютними, відносними, такими, що вказують на сторінку, файл, зображення або містять якір і так далі. Тому посилання можуть вести на будь-який об'єкт в Інтернеті – на веб-сторінку, файл чи зображення. Якщо натиснути на посилання, що веде на файл, браузер запропонує завантажити його. Проте, якщо браузер підтримує цей тип файлів, їх вміст відкриється прямо у браузері. Найчастіше це стосується зображень. Останнім часом браузери навчилися відкривати .pdf файли та багато інших форматів.

**Робота з картинками.**

Щоб додати зображення на веб-сторінку, використовується одиночний тег <img> з атрибутом src, який вказує адресу картинки. Наприклад:

```html

<img src="logo.png">

```

Найпоширенішими форматами зображень у мережі є JPEG і PNG. Для управління шириною або висотою зображення використовуються атрибути width і height. Наприклад:

```html

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

```

У цьому прикладі зображенню задано ширину 100 пікселів. Якщо у користувача відключені зображення або їх неможливо завантажити, браузер відобразить альтернативний текст. Це особливо корисно для меню, створеного за допомогою зображень, щоб користувачі зрозуміли, куди веде кожен пункт. Задання альтернативного тексту за допомогою атрибуту alt є гарною практикою, хоча і не обов'язковою:

```html

<img src="logo.png" alt="Опис зображення">

```

Посилання можна створювати не тільки за допомогою тексту, але й за допомогою зображень. Для цього тег <img> обертається в тег <a>. Посилання-зображення часто використовуються в галереях, де на зменшену версію зображення ставиться посилання на повнорозмірну картинку.

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

Каскадні таблиці стилів — це набір параметрів форматування, який застосовується до елементів веб-сторінки для управління їх виглядом і

положенням.

CSS надає можливість створювати правила, які легко змінювати, редагувати і застосовувати до усіх визначених нами елементів.

Властивість описує елемент, що вводиться.

Значення визначають (природу) параметри властивостей.

Існує два основні способи підключення таблиці стилів:

1. Додавання CSS в HTML-тег.

У цьому способі CSS додається в HTML- документ за допомогою HTML-атрибуту style у середині будь-якого HTML-тегу, що знаходиться у контейнері <body>.

1. Спосіб другий: Установка стилю для тегів в HTML-документі.

CSS додається в HTML-документ за допомогою HTML-тегу <style> в середині контейнеру <head>. В ньому описуються всі стилі, що будуть використані.

Однією з основних дій, яку можна використовувати за допомогою CSS, є робота з кольором та фоном. Властивість color описує колір елемента. Приклад:

H1 { color: #ff0000; } Відображення у браузері:

Властивість background-color описує колір фону елемента. Приклад:  
body { background-color: #FFCC66; } h1{color:#000000;background-color:

#FC9804;text-align:center;}

Відображення у браузері:

CSS-властивість background-image використовується для додавання фонового зображення. Приклад: body { background-image: url(1.gif); }

Блокування фонового зображення [background- attachment]. Властивість background-attachment визначає, чи фіксується фоновий малюнок, або прокручується разом з вмістом сторінки.

Розташування фонового малюнка [background-position]. Є багато способів встановити значення background-position. Тим не менше, всі вони представляють собою набір координат. Координати можна вказувати у відсотках ширини екрана, у фіксованих одиницях (пікселі, сантиметри, і т.п.), або можна використовувати слова top, bottom, center, left і right.

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

Каскадні таблиці стилів — це набір параметрів форматування, який застосовується до елементів веб-сторінки для управління їх виглядом і положенням.

CSS надає можливість створювати правила, які легко змінювати, редагувати і застосовувати до усіх визначених нами елементів.

Властивість описує елемент, що вводиться.

Значення визначають (природу) параметри властивостей.

Існує два основні способи підключення таблиці стилів:

1. Додавання CSS в HTML-тег.

У цьому способі CSS додається в HTML- документ за допомогою HTML-атрибуту style у середині будь-якого HTML-тегу, що знаходиться у контейнері <body>.

1. Спосіб другий: Установка стилю для тегів в HTML-документі.

CSS додається в HTML-документ за допомогою HTML-тегу <style> в середині контейнеру <head>. В ньому описуються всі стилі, що будуть використані.

Однією з основних дій, яку можна використовувати за допомогою CSS, є робота з кольором та фоном. Властивість color описує колір елемента. Приклад:

H1 { color: #ff0000; } Відображення у браузері:

Властивість background-color описує колір фону елемента. Приклад:  
body { background-color: #FFCC66; } h1{color:#000000;background-color:

#FC9804;text-align:center;}

Відображення у браузері:

CSS-властивість background-image використовується для додавання фонового зображення. Приклад: body { background-image: url(1.gif); }

Блокування фонового зображення [background- attachment]. Властивість background-attachment визначає, чи фіксується фоновий малюнок, або прокручується разом з вмістом сторінки.

Розташування фонового малюнка [background-position]. Є багато способів встановити значення background-position. Тим не менше, всі вони представляють собою набір координат. Координати можна вказувати у відсотках ширини екрана, у фіксованих одиницях (пікселі, сантиметри, і т.п.), або можна використовувати слова top, bottom, center, left і right.

**5.5 Практична частина**

### 5.5.1 Постановки задачі

Основною задачею було створити веб-сайт у вигляді резюме. Основною метою цього завдання було вивчення та розбір мови розмітки HTML та структури Веб-

сторінки, знайомство з CSS та каскадуванням стилів, а також вивчення та застосування їх основних функцій. Мій проєкт наповнений ненумерованими списками, відформатованим текстом, а саме текстом зі зміненим кольором, розміром шрифту, стилю і т.д. Також у моїй роботі є такий контент як фото, активні посилання та елементи оформлення об’єктів для кращого сприйняття їх персоною, яка буде переглядати мій веб-сайт.

### 5.5.2 HTML документ

<!DOCTYPE html>

<html lang="uk">

<head>

    <meta charset="UTF-8">

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

    <title>Резюме</title>

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

</head>

<body>

<div class="container">

    <div class="photo">

        <img src="image.jpg" class="" alt="Фото">

    </div>

    <div class="content">

        <h1>Свідерок Олександр</h1>

        <h2>Студент</h2>

        <div class="section">-

            <h3>Контактна інформація</h3>

            <hr>

            <div class="info">

                <div>

                    <p><strong>Email:</strong> sashaitkids@gmail.com</p>

                    <p><strong>Телефон:</strong> +38 (097) 782-5003</p>

                </div>

                <div>

                    <p><strong>Місто:</strong> Київ</p>

                    <p><strong>Країна:</strong> Україна</p>

                </div>

            </div>

        </div>

        <div class="section">

            <h3>Досвід в професійних клубах</h3>

            <hr>

            <p><strong>ФК «Динамо» (2021-2023):</strong> Центральний захисник,  1 місце в м.Київ</p>

            <p><strong>ФК "Колос" (2018-2021):</strong> Правий захисник, 1 місце у Київській обл. </p>

        </div>

        <div class="section">

            <h3>Освіта</h3>

            <hr>

            <p><strong>Оптико-механічний коледж Київського національного університету імені Тараса Шевченка :</strong> Інженерія програмного забеспечення, 2022-2024</p>

            <p><strong>Крюківщинська загальноосвітня школа:</strong> Базова середня освіта, 2013-2022</p>

        </div>

        <div class="section">

            <h3>Навички</h3>

            <hr>

            <p>Володіння мовами,

                Розробка сайтів та модифікацій для ігр.

            </p>

        </div>

    </div>

    <div class="buttons">

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

        <a href="https://www.instagram.com/\_sanxs.kk\_?igsh=cmdocXZxanBmZzA2&utm\_source=qr" target="\_blank">Instagram</a>

      </div>

</div>

</body>

</html>

**CSS**

body {

    font-family: Arial, sans-serif;

    margin: 0;

    padding: 0;

    background-color: #f4f4f4;

}

.container {

    width: 80%;

    margin: 20px auto;

    background: white;

    padding: 20px;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

    border-radius: 10px;

}

.photo {

    text-align: center;

}

.photo img {

    width: 150px;

    height: 150px;

    border-radius: 100px;

    border: 5px solid #007BFF;

}

.content {

    margin-top: 20px;

}

h1, h2, h3 {

    text-align: center;

    color: #333;

}

.section {

    margin: 20px 0;

    padding: 0 20px;

}

.section p {

    line-height: 1.6;

}

.buttons {

    text-align: center;

    margin-top: 20px;

}

.buttons a {

    display: inline-block;

    margin: 10px;

    padding: 10px 20px;

    color: white;

    background-color: #007BFF;

    text-decoration: none;

    border-radius: 5px;

    transition: background-color 0.3s;

}

.buttons a:hover {

    background-color: #0056b3;

}

.info {

    display: flex;

    justify-content: space-between;

    margin: 0 20px;

}

.info div {

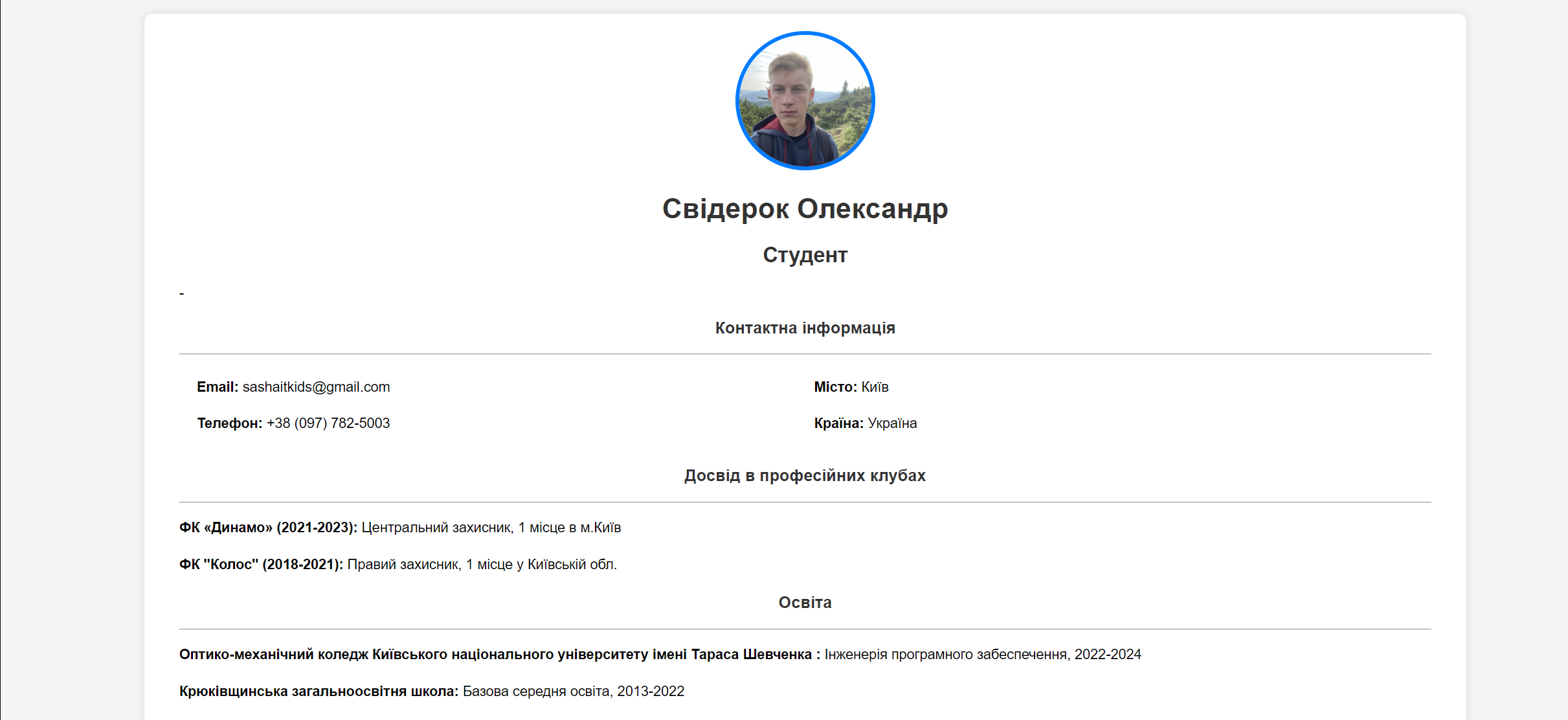
    flex: 1;

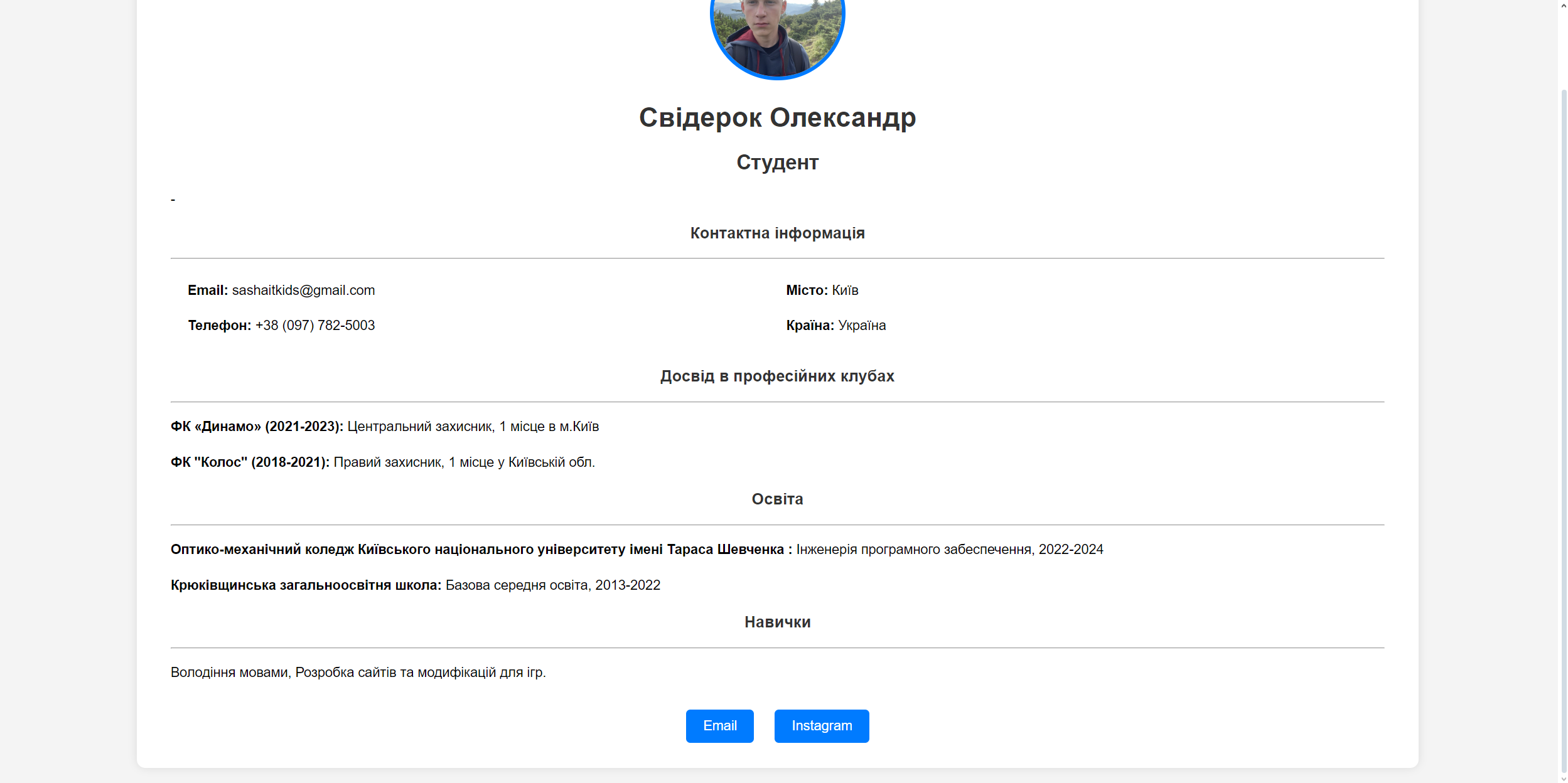
}

.info div:not(:last-child) {

    margin-right: 20px;

}

****

****

# ВИСНОВКИ

Під час практичних занять ми здобули базові знання з HTML, CSS, C++ та інформатики, що є основою для будь-якої діяльності в галузі інформаційних технологій.

Ми навчилися створювати структуру веб-сторінок за допомогою HTML та додавати стилі за допомогою CSS. Це дозволяє створювати естетично привабливі та функціональні веб-сайти.

Мова програмування C++ дозволила нам зрозуміти основи програмування, такі як змінні, цикли, умови, функції та об'єктно-орієнтоване програмування. Це важливо для розробки ефективного та надійного програмного забезпечення. За допомогою практичного завдання ми підтвердили наші знання.

Вивчення інформатики допомогло нам зрозуміти основи роботи комп'ютерів, алгоритмів, структур даних та мереж, ми удосконалили знання про табличні процеси та правильного оформлення тексту. Ці знання є необхідними для вирішення складних технічних задач.

**Загальний висновок**

Отримані знання з HTML, CSS, C++ та інформатики створюють міцну основу для подальшого навчання та розвитку у сфері ІТ. Вони відкривають перед нами безліч можливостей для професійного зростання та реалізації власних проектів.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ

1. ДСТУ 1.5-93 Загальні вимоги до побудови, викладу, оформлення та змісту.
2. Мова програмування С++: конспект лекцій / О. В. Галкін, М. М. Верес. — К.: ДП “Вид. дім “Персонал”, 2017. — 260 с. — Біблі- огр.: с. 249.: <https://maup.com.ua/assets/files/lib/book/c_plisplus.pdf>
3. «ОСНОВИ ПРОГРАМУВАННЯ на C++» Зеленський О.С., Лисенко В.С. – Кривий Ріг: Державний університет економіки і технологій, 2023.-269 с.: <https://dspace.duet.edu.ua/jspui/bitstream/123456789/831/1/НП%20Osnovy_C%2B%2B.pdf>
4. Формальні та фактичні параметри, хеш-таблиця (словник), масив, список, стек, черга мовою C++: <https://www.google.com/url?Sa=i&url=https%3A%2F%2Fwww.kievoit.ippo.kubg.edu.ua%2Fkievoit%2F2016%2F65_C%2B%2B%2Findex.html&psig=aovvaw0e_izvl6jk9vdsho-jqp2y&ust=1717139073705000&source=images&cd=vfe&opi=89978449&ved=0cacqrpomahckewiah9ui7lsgaxuaaaaahqaaaaaqba>
5. Міністерство освіти і науки україни Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» «Управління версіями програмних засобів проекту»: <https://ipze.kpi.ua/wp-content/uploads/2024/02/04-Upravlinnia_versiiamy_prohramnykh_zasobiv_proektu.pdf>
6. Матеріал з Вікіпедії — вільної енциклопедії. «Markdown»: <https://uk.wikipedia.org/wiki/Markdown>
7. MDN Web Docs - an open-source, collaborative project documenting Web platform technologies : <https://developer.mozilla.org/en-US/>
8. Міністерство освіти і науки україни державний вищий навчальний заклад «Ужгородський національний університет» математичний факультет, кафедра кібернетики і прикладної математики «Основи HTML та CSS»: <https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/40974/1/Основи%20HTML.pdf>
9. Міністерство освіти і науки України державний вищий навчальний заклад “Ужгородський національний університет” математичний факультет, кафедра системного аналізу і теорії оптимізації «Вступ до web-програмування. Основи html»: <https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/32871/1/Вступ%20до%20WEB-програмування.%20Основи%20HTML.pdf>
10. Е-helper.com.ua «Лекція 2. Розмітка тексту. Зображення. Посилання.»: <http://e-helper.com.ua/node/949>
11. «CASCADING STYLE SHEETS (CSS) - КАСКАДНІ ТАБЛИЦІ СТИЛІВ»: <https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=http://vv-steshyn.edu.kpi.ua/katalog/files/Lekciya-3.-CSS-kaskadni-tablici-stiliv-68.pdf&ved=2ahUKEwi5utTekqKGAxXD4AIHHaW3DtcQFnoECBIQAQ&usg=AOvVaw3_uxBU96i5-43dUKH8RfG4>
12. Е-helper.com.ua «Лекція 5. Наслідування та каскадування. Оформлення тексту за допомогою CSS»: <http://e-helper.com.ua/node/952>
13. А. Каплун, Ю. В. Баришев, А. В. Остапенко «ТЕХНОЛОГІЯ ПРОГРАМУВАННЯ ЛАБОРАТОРНИЙ ПРАКТИКУМ»: <https://web.posibnyky.vntu.edu.ua/fitki/14baryshev_tehnologiya_programuvannya/08.html>
14. MDN Web Docs - an open-source, collaborative project documenting Web platform technologies : <https://developer.mozilla.org/en-US/>
15. «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>
16. Е-helper.com.ua «Лекція 2. Розмітка тексту. Зображення. Посилання.»: <http://e-helper.com.ua/node/949>
17. Міністерство освіти і науки україни державний вищий навчальний заклад «Ужгородський національний університет» математичний факультет, кафедра кібернетики і прикладної математики «Основи HTML та CSS»: <https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/40974/1/Основи%20HTML.pdf>

# ДОДАТКИ