ВІДОКРЕМЛЕННИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ

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

Циклова комісія спеціальності «Інженерія програмного забезпечення»

**З В І Т**

**З НАВЧАЛЬНОЇ ПРАКТИКИ**

спеціальність 121 Інженерія програмного забезпечення

освітньо-кваліфікаційний рівень «фаховий молодший бакалавр»

|  |  |  |
| --- | --- | --- |
| ПІДСУМКОВА ОЦІНКА  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оцінка цифрою та прописом)  Керівники практики від коледжу:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. С. Нікітенко  (оцінка) (підпис)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т. І. Лумпова  (оцінка) (підпис)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ М. С. Ходжаєв  (оцінка) (підпис) |  | Виконав: |
| студента 2-го курсу групи ІПЗ-22 |
| Перевозник Назар Віталійович  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) | |  | |
| *Перевозник Назар Віталійович* | | | | |
| (П.І.Б студента.) | | | | |
| Спеціальність | *121 Інженерія програмного забезпечення* | Група | | *ІПЗ-22* |

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

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

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

Змн.

Арк.

№ докум.

Підпис

Дата

Арк.

4

Розроб.

Перевір.

Реценз.

В.Я

Н. Контр.

В.Я

Затверд.

Звіт з навчальної практики

Літ.

Аркушів

ІПЗ-21

ОМФК.0121 НП 20

**ЗМІСТ**

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

[ВСТУП 6](#_Toc167919157)

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

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

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

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

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

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

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

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

[3.1.2 Арифметичні операції 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 11](#_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 12](#_Toc167919173)

[3.2.4 Скріншот виконання програми 14](#_Toc167919174)

[**3.3 Виконання тестів** 14](#_Toc167919175)

[4 РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++" 14](#_Toc167919176)

[**4.1 Тема 1. Введення та виведення у базових консольних програмах** 14](#_Toc167919177)

[**4.2 Тема 2. Структури даних для управління станом програми** 15](#_Toc167919178)

[**4.3 Тема 3. Робота з файлами: запис та читання** 15](#_Toc167919179)

[**4.4 Тема 4. Формати файлів та їх обробка** 15](#_Toc167919180)

[**4..5 Тема 5. Системи контролю версій (VCS)** 15](#_Toc167919181)

[**4.6 Тема 6. Мова Markdown для опису проекті** 16](#_Toc167919182)

[**4.7 Розроблення ігрової програми** 16](#_Toc167919183)

[4.7.1 Код програми 16](#_Toc167919184)

[4.7.2 Скріншоти виконання програми 16](#_Toc167919185)

[5 РОЗДІЛ ”ВЕБ-ДИЗАЙН ТА HTML” 17](#_Toc167919186)

[**5.1 Мова розмітки HTML та структура Веб-сторінки** 17](#_Toc167919187)

[**5.2 Розмітка тексту за допомогою HTML. Зображення та посилання** 17](#_Toc167919188)

[**5.3 Каскадні таблиці стилів CSS та їх структура** 18](#_Toc167919189)

[**5.4 Спадковості і каскадування за допомогою CSS** 19](#_Toc167919190)

[**5.5 Практична частина** 20](#_Toc167919191)

[5.5.1 Постановки задачі 20](#_Toc167919192)

[5.5.2 HTML документ 20](#_Toc167919193)

[ВИСНОВКИ 23](#_Toc167919194)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ 24](#_Toc167919195)

[ДОДАТКИ 25](#_Toc167919196)

ОМФК.0121 НП 20

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

# ВСТУП

Автор:

Дата:

Мета:

Перелік вивчених тем:

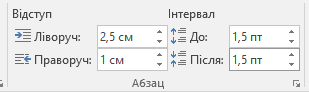
# 1 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ

# 2 РОЗДІЛ "ІНФОРМАТИКА"

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

1. Створити новий документ MS Word (орієнтація книжкова, розмір

А4, поля документа: верхнє –1.5, нижнє – 1.5, ліве – 2.5, праве – 1.0 см).

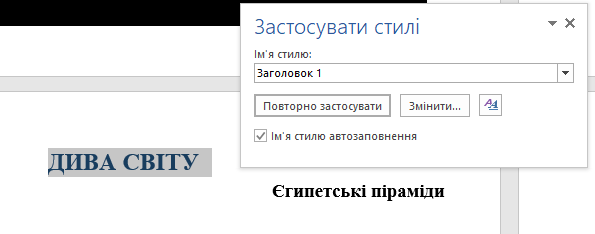
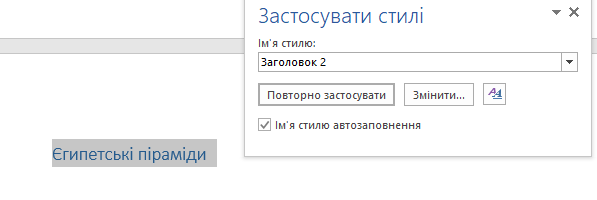


3. Відформатувати доданий текст: для заголовків першого рівня

обрати стиль «Заголовок 1», для заголовків другого рівня – стиль

«Заголовок 2», для основного тексту встановити шрифт Times New

Roman, 14 пт, інтервал між рядками 1.5 пт, відступ абзаца 1.25.

4. Змінити стиль заголовків першого рівня: шрифт Times New

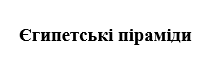
Roman, 18 пт, жирний, всі літери заглавні, колір темно-синій,

розташування посередині сторінки.



5. Змінити стиль заголовків другого рівня: шрифт Times New

Roman, 16 пт, жирний, колір чорний, розташування ліворуч.



6. Додати порожню сторінку на початок документа і створити на ній

зміст.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

7. Додати титульну сторінку документа та оформити її.

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

8. Вставити нумерацію сторінок документа наступним чином: на

непарних сторінках нумерація розташовується внизу сторінки

праворуч, на парних – внизу сторінки ліворуч.



9. Додати в текст документа картинки (фото) відповідно до його

змісту.

Изображение выглядит как текст, снимок экрана, пирамида

Автоматически созданное описание

10. Розширити розділ «7 НОВИХ ЧУДЕС ПРИРОДИ» додатковою

інформацією з мережі Інтернет. Виділити в ньому заголовки другого

рівня, вставити відповідні фотографії.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

11. Користуючись можливостями Word підпису об’єктів документа

вставити підписи до рисунків у форматі: Рисунок 1, Рисунок 2 і т.д:

Изображение выглядит как на открытом воздухе, природа, Водные ресурсы, водопад

Автоматически созданное описание Изображение выглядит как природа, пещера, Морская пещера, озеро

Автоматически созданное описание

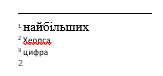
12. В тексті документа створити посилання на рисунки (не менше 5

посилань).

13. Створити виноски не менше ніж до трьох слів тексту (за власним

вибором).



14. Додати верхній колонтитул з назвою розділу тексту.



15. Оновити сторінку зі змістом документа.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

16. Для змісту встановити шрифт Times New Roman, 14 пт.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

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

а допомогою засобів табличного редактора MS Excel розв’язав наступні задачі:

1

Катети прямокутного трикутника дорівнюють 8см і 13см. Визначить гіпотенузу цього трикутника та площу. На скільки має змінитися менший катет заданого трикутника, щоб його гіпотенуза дорівнювала 22 см? Чому дорівнює площа такого трикутника?

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

2

Ціну 10 марок автомобілів, задану в $, перерахувати в грошових одиницях України та Польщі, використовуючи дані з інтернету про ціну автомобілів та курси валют. Визначить, яку суму необхідно сплатити автосалону, якщо він закупає по декілька екземплярів кожної моделі. (Марки автомобілів та їх кількість для автосалону обрати самостійно).

Изображение выглядит как текст, число, Шрифт, снимок экрана

Автоматически созданное описание

3

Типографія виготовляє зошити трьох видів: по 24 аркуші, по 48 аркушів і по 64 аркуші. Денний дохід від одного зошиту кожного виду 2 грн., 3 грн. і 5 грн. Необхідно визначити, за якої кількості виробів типографія отримає максимальний дохід, з урахуванням того, що:

* загальний обсяг виробництва — 300 зошитів на день;
* підприємству потрібно виготовити 50 зошитів по 24 аркуші для виконання існуючого замовлення;
* 40 зошитів по 48 аркушів — для виконання планового замовлення;
* зошитів по 64 аркуші необхідно виготовляти не більше ніж 40 одиниць на день.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

4

Засобами табличного процесора здійсніть статистичний аналіз температурного режиму двох весняних місяців (для введення початкових значень температури скористайтесь зразком таблиці на рисунку 2.18).

За веденими даними потрібно визначити:

* середнє значення температури за місяць;
* максимальну і мінімальну температури;
* значення температури, що найчастіше повторювалось;
* медіану температурних значень;

Використовуючи комбіновану діаграму виконайте візуалізацію значень температур за два місяці.

Изображение выглядит как текст, линия, диаграмма, График

Автоматически созданное описание

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

За допомогою засобів табличного редактора MS Access виконав наступні завдання:

1

Створіть структуру БД ГРУПА із таблицями Студенти, Дисципліни і Успішність. Властивості цих таблиць

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

2

В кожну таблицю введіть по п’ять записів, збережіть створену БД.

Изображение выглядит как текст, Шрифт, число, линия

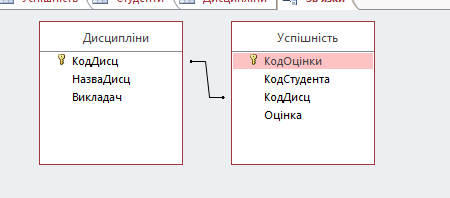
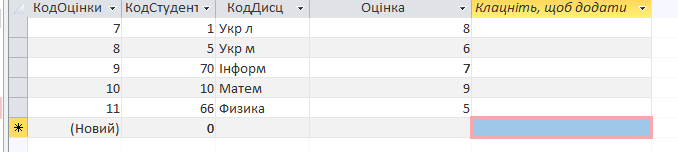
Автоматически созданное описаниеИзображение выглядит как текст, Шрифт, число, снимок экрана

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

3

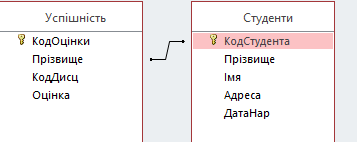
Відкрийте таблицю Успішність у режимі конструктора. Для поля КодДисц у списку типів даних виберіть ***Майстер*** ***підстановок***. Підставте замість числових кодів КодДисц таблиці Успішність відповідні назви дисциплін із таблиці Дисципліни. Для поля КодДисц уведіть підпис НазваДисц.



4

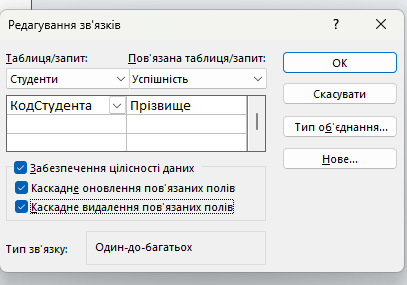
У таблиці Успішність для поля КодСтудент налаштуйте підстановку прізвищ з таблиці Студенти і введіть підпис Прізвище.

Изображение выглядит как текст, снимок экрана, Шрифт, число

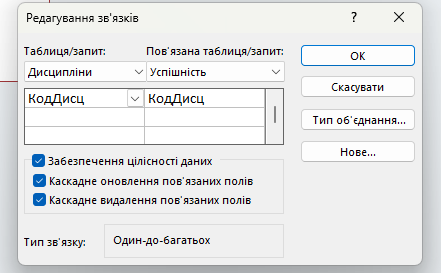
Автоматически созданное описание

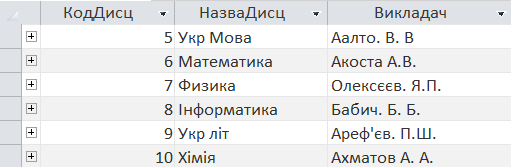
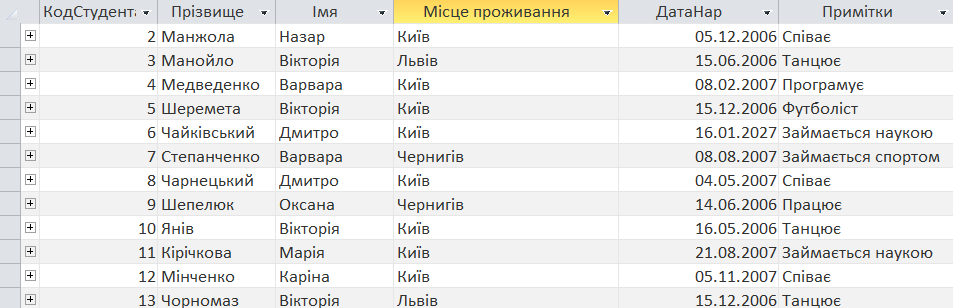
5

Відкрийте вікно ***Зв’язки***, додавши до нього таблиці Дисципліни, Успішність, Студенти. Зверніть увагу, що після налаштування підстановок між таблицями вже з’явилися зв’язки, для яких залишилося зазначити тип і умови цілісності. Викличте контекстне меню на лінії зв’язку, виберіть команду, у вікні ***Редагування зв’язків*** змініть тип зв’язку між таблицями Студенти та Успішність на **один-до-багатьох** і встановіть всі три прапорці. Це забезпечить цілісність даних і каскадне оновлення та видалення записів.



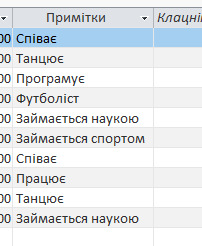
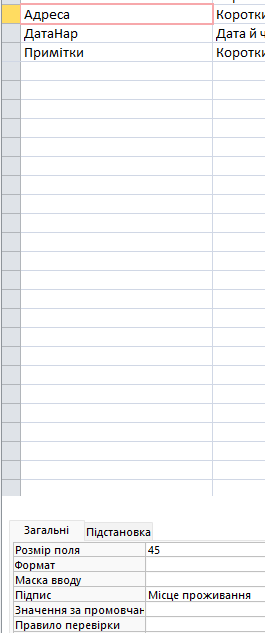
6

Зробіть подібні налаштування для зв’язку між таблицями Дисципліни та Успішність. Збережіть схему даних. Доповніть таблиці Студенти, Успішність до 10 записів, таблицю Дисципліни — до 8 записів.



7

У таблиці Студенти для поля Адреса встановіть підпис Місце проживання. У кінці таблиці додайте поле Примітки, у яке введіть дані, наприклад, «Займається спортом», «Співає», «Танцює», «Займається наукою». Збережіть зміни в таблиці.



8

Доповніть таблицю Дисципліни до 15 записів. Таблицю Успішність доповніть до 20 записів шляхом копіювання й вставляння.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

9

У нових записах вибірково замінить окремі прізвища на нові зі списку, поміняйте оцінки.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

10

Відкрийте таблицю Студенти, додайте нові записи (до 25) шляхом копіювання та вставляння.

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

11

У нових записах замініть прізвища на інші, кілька жіночих імен замініть на ім’я *Варвара*, необов’язкові поля Адреса, ДатаНар і Примітка можна не змінювати. За допомогою команди ***Замінити*** у таблиці змініть ім’я Вікторія на ім’я Варвара.

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание12

Впорядкуйте в таблиці Студенти дані одночасно за полями Прізвище та Імя за алфавітом.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

13

У таблиці Успішність виконайте фільтрування за виділеним власним прізвищем.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

14

Доповніть таблицю Студенти, щоб у ній були два-три записи з прізвищами, що починаються з літери «К» і «Г», та іменами, що закінчуються літерою «А» а також два-три прізвища із закінченням «енко», рік народження — 2007.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

# 3 РОЗДІЛ "ПРОГРАМУВАННЯ С++"

## **3.1 Теоретичні відомості**

C++ — це мова програмування загального призначення, яка підтримує різні парадигми програмування, зокрема процедурне, об'єктно-орієнтоване та узагальнене програмування. Вона була розроблена Б'ярне Страуструпом як розширення мови C. Ось основні концепції C++:

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

### Елементи мови:

### Константи:

### Фіксовані значення, які не змінюються під час виконання програми.

### Оголошуються з ключовим словом const.

### Наприклад: const int MAX\_SIZE = 100;.

### Змінні:

### Іменовані області пам'яті, де зберігаються значення.

### Оголошуються з вказанням типу.

### Наприклад: int age = 25;.

### Операції:

### Арифметичні: +, -, \*, /, %.

### Логічні: &&, ||, !.

### Порівняння: ==, !=, <, >, <=, >=.

### Присвоєння: =, +=, -=, \*=, /=, %=.

### Перетворення типів:

### Неявні: автоматичні перетворення типів компілятором (наприклад, з int в float).

### Явні: примусове перетворення типів за допомогою операторів або функцій (наприклад, static\_cast<int>(value)).

### Ці основні елементи мови C++ дозволяють створювати ефективні та структуровані програми

### 3.1.2 Арифметичні операції

 **Додавання**: +

 **Віднімання**: -

 **Множення**: \*

 **Ділення**: /

 **Ділення за модулем (остача)**: %

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

1. **Вхідний потік (cin)**:
   * Потік для отримання введених даних з клавіатури.
   * Використовується з оператором >>.
2. **Вихідний потік (cout)**:
   * Потік для виведення даних на екран консолі.
   * Використовується з оператором <<.

**Найпростіші математичні функції**

1. **Додавання**: a + b
2. **Віднімання**: a - b
3. **Множення**: a \* b
4. **Ділення**: a / b
5. **Модуль числа**: abs(x)
6. **Піднесення до степеня**: pow(x, n)
7. **Корінь квадратний**: sqrt(x)

### 3.1.4 Програмування лінійних та розгалужених обчислювальних процесів

1. **Лінійний процес**:
   * Виконується послідовно від початку до кінця без переривань.
   * Приклад: обчислення суми двох чисел.

**Програмування розгалужених обчислювальних процесів:**

1. **Розгалужений процес**:
   * Має різні шляхи виконання залежно від умов.
   * Використовує умовні оператори (if, else, switch).
   * Приклад: обчислення максимуму з двох чисел залежно від умови.

### 3.1.5 Програмування циклічних обчислювальних процесів

**Циклічний процес**:

* Виконується декілька разів до досягнення заданої умови.
* Використовується циклічний оператор (for, while, do-while).
* Приклад: обчислення суми чисел від 1 до N.

## **3.2 Практичне завдання**

Написати код на мові програмування C++ для розрахунку по формулам

### 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.*

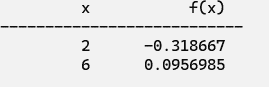
Коефіцієнти *a, b, c –* дійсні числа. Значення *a, b, c, xпоч*, *xкін*, *h* вводити з клавіатури. Передбачити перевірку допустимості введених значень.

### 3.2.2 Постановка задачі

Вхідні дані

5,4,8,2,8,4

Вихідні дані



Блок-схема алгоритму роботи програми представлена на рис. ххх

Блок-схема

Изображение выглядит как диаграмма, линия, оригами, шаблон

Автоматически созданное описание

### 3.2.3 Код програми

#include <iostream>

#include <iomanip>

#include <cmath>

using namespace std;

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

if (x + 10 > 0 && c > 3) {

return (b + 1) / (a \* x) - 0.5 - x;

} else if (x + 10 < 0 && b - c < 0) {

return (x - 2 \* a) / (b \* b) + x;

} else {

return sqrt(x \* x + 1) / (15 \* a \* b \* c);

}

}

int main() {

double a, b, c;

double xPoch, xKin, h;

// Введення коефіцієнтів та параметрів

cout << "Введіть значення a: ";

cin >> a;

cout << "Введіть значення b: ";

cin >> b;

cout << "Введіть значення c: ";

cin >> c;

cout << "Введіть початкове значення x: ";

cin >> xPoch;

cout << "Введіть кінцеве значення x: ";

cin >> xKin;

cout << "Введіть крок h: ";

cin >> h;

// Перевірка введених значень

if (h <= 0) {

cout << "Крок h має бути додатним числом." << endl;

return 1;

}

// Виведення заголовків таблиці

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

for (double x = xPoch; x <= xKin; x += h) {

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

cout << setw(10) << fixed << setprecision(4) << x << setw(10) << fixed << setprecision(4) << fx << endl;

}

return 0;

}

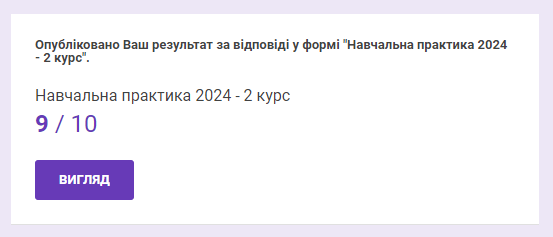
### 3.2.4 Скріншот виконання програми

Скріншот

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

## **3.3 Виконання тестів**



# 4 РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++"

## 

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

Введення та виведення даних здійснюється за допомогою бібліотеки iostream. Основні оператори для цього — cin для введення та cout для виведення даних.

#include <iostream> // Підключення бібліотеки для введення/виведення

int main() {

std::string name; // Оголошення змінної для зберігання імені

int age; // Оголошення змінної для зберігання віку

// Введення даних від користувача

std::cout << "Введіть ваше ім'я: ";

std::cin >> name;

std::cout << "Введіть ваш вік: ";

std::cin >> age;

// Виведення даних на екран

std::cout << "Привіт, " << name << "! Вам " << age << " років." << std::endl;

return 0;

}

1. **Підключення бібліотеки**:

#include <iostream>

Ця директива включає стандартну бібліотеку для введення та виведення даних.

1. **Функція main**:

int main() {

Це головна функція, з якої починається виконання програми.

1. **Оголошення змінних**:

std::string name; // Для зберігання імені

int age; // Для зберігання віку

Змінні name та age оголошені для зберігання відповідних даних.

1. **Введення даних**:

std::cout << "Введіть ваше ім'я: ";

std::cin >> name;

std::cout << "Введіть ваш вік: ";

std::cin >> age;

Оператор std::cout використовується для виведення тексту на екран, а оператор std::cin — для зчитування введених користувачем даних.

1. **Виведення даних**:

std::cout << "Привіт, " << name << "! Вам " << age << " років." << std::endl;

Оператор << використовується для виведення декількох частин рядка та змінних. std::endl додає символ нового рядка в кінці.

1. **Повернення значення з функції main**:

return 0;

Це вказує на успішне завершення програми.

### Загальні концепції:

* **Введення**: Використовується оператор >> з std::cin для зчитування даних з консолі.
* **Виведення**: Використовується оператор << з std::cout для виведення даних на консоль.
* **Типи даних**: У цьому прикладі використані змінні типу std::string для рядків та int для цілих чисел.

## 

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

Структури даних є важливою складовою будь-якої програми, оскільки вони дозволяють організувати та зберігати дані ефективним способом. Вибір відповідної структури даних допомагає забезпечити ефективність, зручність доступу та управління даними, що, у свою чергу, впливає на загальну продуктивність та надійність програми.

Масиви (Arrays) дозволяють зберігати послідовність елементів одного типу в суміжній області пам'яті. Вони корисні, коли необхідно зберігати фіксовану кількість елементів. Основна перевага масивів полягає у швидкому доступі до елементів за індексом. Проте їх розмір не можна змінити під час виконання програми.

**Переваги**: Прості у використанні, дозволяють швидкий доступ до елементів за індексом.

**Недоліки**: Фіксований розмір, що визначається при створенні.

Структури (struct) дозволяють об'єднувати різні типи даних під одним ім'ям. Це зручно для зберігання пов'язаних даних, таких як ім'я та вік особи. Структури дозволяють зберігати дані різних типів разом і надають простий спосіб їх організації.

Класи (class) подібні до структур, але також підтримують інкапсуляцію і можуть містити методи для роботи з даними. Вони дозволяють створювати об'єкти з приватними і публічними членами, що забезпечує кращу організацію та захист даних. Класи можуть включати конструктори, деструктори та різноманітні методи для взаємодії з внутрішнім станом об'єкта.

Вектори (std::vector) є динамічними масивами, які можуть змінювати свій розмір під час виконання програми. Вони забезпечують гнучкість у роботі з даними змінного розміру та мають зручні методи для додавання, видалення та доступу до елементів. Вектори є частиною стандартної бібліотеки C++ і автоматично управляють пам'яттю.

Списки (std::list) це двозв'язні списки, що дозволяють ефективно додавати і видаляти елементи в будь-якому місці. Вони особливо корисні, коли необхідно часто додавати і видаляти елементи, хоча доступ до елементів за індексом є повільнішим у порівнянні з масивами та векторами.

**Переваги**: Динамічний розмір, легке додавання та видалення елементів.

**Недоліки**: Доступ до елементів потребує послідовного проходження списку.

Мапи (std::map) є асоціативними контейнерами, що зберігають пари "ключ-значення". Вони забезпечують швидкий пошук, додавання та видалення елементів за ключем. Мапи корисні для зберігання даних, що потребують доступу за унікальним ключем.

Множини (std::set) є контейнерами, що зберігають унікальні елементи у впорядкованому вигляді. Вони забезпечують швидкий пошук і перевірку наявності елементів. Множини корисні, коли важливо зберігати лише унікальні значення та підтримувати їх впорядкованість.

Кожна з цих структур даних має свої сильні та слабкі сторони. Вибір конкретної структури залежить від вимог до управління станом програми та характеристик операцій, які необхідно виконувати над даними.

## 

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

Робота з файлами є важливою частиною багатьох програм, оскільки дозволяє зберігати дані на постійному носії і зчитувати їх для подальшої обробки. В залежності від мови програмування, існують різні способи реалізації запису та читання даних у файли.

У мові C++ для роботи з файлами використовується бібліотека <fstream>, яка надає засоби для створення, відкриття, запису та читання файлів. Розглянемо основні операції з файлами:

Запис у файл

#include <iostream>

#include <fstream> // Підключення бібліотеки для роботи з файлами

int main() {

std::ofstream fout; // Об'єкт для запису в файл

// Відкриття файлу для запису (створення нового файлу або очищення існуючого)

fout.open("example.txt");

// Перевірка, чи файл вдало відкрито

if (!fout.is\_open()) {

std::cerr << "Помилка відкриття файлу!" << std::endl;

return 1;

}

// Запис даних у файл

fout << "Це приклад запису в файл." << std::endl;

fout << "Це другий рядок у файлі." << std::endl;

// Закриття файлу

fout.close();

std::cout << "Дані були успішно записані у файл." << std::endl;

return 0;

}

Читання з файлу

#include <iostream>

#include <fstream> // для роботи з файлами

#include <string>

int main() {

std::ifstream inFile; // об'єкт для читання з файлу

// Відкриття файлу для читання

inFile.open("example.txt");

if (!inFile) { // перевірка на успішність відкриття файлу

std::cerr << "Неможливо відкрити файл для читання!" << std::endl;

return 1;

}

// Читання даних з файлу

std::string message;

std::getline(inFile, message);

// Виведення прочитаних даних на консоль

std::cout << "Прочитано з файлу: " << message << std::endl;

// Закриття файлу після читання

inFile.close();

return 0;

}

**Запис у файл** (ofstream):

* std::ofstream fout;: Створення об'єкту для запису у файл.
* fout.open("example.txt");: Відкриття файлу "example.txt" для запису. Якщо файл не існує, він буде створений.
* fout << "Текст для запису";: Запис даних у файл за допомогою оператора <<.
* fout.close();: Закриття файлу після завершення запису.

**Читання з файлу** (ifstream):

* std::ifstream fin;: Створення об'єкту для читання з файлу.
* fin.open("example.txt");: Відкриття файлу "example.txt" для читання.
* std::getline(fin, line): Читання рядка з файлу в змінну line.
* fin.close();: Закриття файлу після завершення читання.

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

Формати файлів є важливим аспектом обробки даних в програмах, оскільки вони визначають структуру і спосіб представлення інформації. В залежності від потреб програми та характеристик даних можуть використовуватися різні формати файлів. Розглянемо деякі основні формати і способи їх обробки.

### Основні формати файлів

1. **Текстові файли**

Це файли, у яких дані зберігаються у вигляді тексту, який можна прочитати і редагувати.

Простота в обробці, можуть містити різноманітні дані, такі як рядки тексту, числа тощо.

Конфігураційні файли, файли з даними у форматі CSV (Comma-Separated Values), файли з логами.

#include <iostream>

#include <fstream> // Бібліотека для роботи з файлами

int main() {

std::ofstream fout("example.txt"); // Запис у файл

if (!fout) {

std::cerr << "Не вдалося відкрити файл для запису!" << std::endl;

return 1;

}

fout << "Це приклад запису у текстовий файл." << std::endl;

fout.close();

std::ifstream fin("example.txt"); // Читання з файлу

if (!fin) {

std::cerr << "Не вдалося відкрити файл для читання!" << std::endl;

return 1;

}

std::string line;

while (std::getline(fin, line)) {

std::cout << line << std::endl;

}

fin.close();

return 0;

}

1. **CSV (Comma-Separated Values)**

Текстовий формат, в якому дані розділені комами (або іншими розділювачами) і можуть містити рядки і числа.

Легко читається і записується, підтримується багатьма програмами для обробки даних.

Імпорт і експорт даних з баз даних, таблиці для аналізу даних у програмах Excel, Google Sheets тощо.

Для обробки CSV файлів можна використовувати сторонні бібліотеки, такі як libcsv або робити ручне розбиття строк.

#include <iostream>

#include <fstream>

#include <sstream>

#include <vector>

int main() {

std::ofstream fout("data.csv"); // Запис у CSV файл

if (!fout) {

std::cerr << "Не вдалося відкрити файл для запису!" << std::endl;

return 1;

}

fout << "Name,Age,Salary" << std::endl;

fout << "John,25,35000" << std::endl;

fout << "Jane,30,45000" << std::endl;

fout.close();

std::ifstream fin("data.csv"); // Читання з CSV файлу

if (!fin) {

std::cerr << "Не вдалося відкрити файл для читання!" << std::endl;

return 1;

}

std::string line;

while (std::getline(fin, line)) {

std::vector<std::string> tokens;

std::stringstream ss(line);

std::string token;

while (std::getline(ss, token, ',')) {

tokens.push\_back(token);

}

for (const auto& token : tokens) {

std::cout << token << "\t";

}

std::cout << std::endl;

}

fin.close();

return 0;

}

**Переваги:**

1. **Простота**: Легкий у читанні та записі, особливо для табличних даних.
2. **Компактність**: Зазвичай займає менше місця порівняно з JSON та XML.
3. **Сумісність**: Широко підтримується багатьма програмами, включаючи електронні таблиці, такі як Microsoft Excel.

**Недоліки:**

1. **Обмежена структура**: Підходить тільки для табличних даних, не підтримує вкладені структури.
2. **Відсутність метаданих**: Не має вбудованої підтримки для опису типів даних або їх структури.
3. **Проблеми з форматом**: Може виникнути проблема з обробкою полів, що містять коми або нові рядки.

**Використання:**

* Експорт та імпорт табличних даних
* Робота з даними в електронних таблицях
* Прості списки даних

1. **Бінарні файли** є одним з форматів збереження даних, де інформація записується у вигляді послідовності байтів, що представляють собою числа у двійковій формі. Вони відрізняються від текстових файлів тим, що не мають вбудованої структури для розділення даних на окремі рядки або символи нового рядка.

Приклад запису та читання даних у бінарний файл:

#include <iostream>

#include <fstream>

#include <string>

struct Person {

std::string name;

int age;

double salary;

};

int main() {

// Запис даних у бінарний файл

std::ofstream fout("data.bin", std::ios::binary);

if (!fout) {

std::cerr << "Помилка відкриття файлу для запису!" << std::endl;

return 1;

}

Person p1 = {"John Doe", 30, 50000.0};

fout.write(reinterpret\_cast<char\*>(&p1), sizeof(Person));

fout.close();

std::cout << "Дані були успішно записані у бінарний файл data.bin." << std::endl;

// Читання даних з бінарного файлу

std::ifstream fin("data.bin", std::ios::binary);

if (!fin) {

std::cerr << "Помилка відкриття файлу для читання!" << std::endl;

return 1;

}

Person p2;

fin.read(reinterpret\_cast<char\*>(&p2), sizeof(Person));

fin.close();

std::cout << "Прочитані дані з бінарного файлу:" << std::endl;

std::cout << "Ім'я: " << p2.name << std::endl;

std::cout << "Вік: " << p2.age << std::endl;

std::cout << "Зарплата: " << p2.salary << std::endl;

return 0;

}

**JSON (JavaScript Object Notation) (відноситься до бінарних)**

Формат зберігання даних, який використовується для обміну інформацією між серверами та веб-додатками.

Підтримує структуризовані дані, такі як об'єкти і масиви, легко читається людиною і має добру підтримку в більшості мов програмування.

Конфігураційні файли, обмін даними між веб-сервісами, зберігання налаштувань і структурованих даних.

#include <iostream>

#include <fstream>

#include <nlohmann/json.hpp>

using json = nlohmann::json;

int main() {

// Створення об'єкту JSON

json j;

j["name"] = "John";

j["age"] = 30;

j["city"] = "New York";

j["married"] = true;

// Запис JSON у файл

std::ofstream fout("data.json");

fout << j.dump(4); // 4 - кількість пробілів для відступу (опціонально)

fout.close();

std::cout << "Дані були успішно записані у файл data.json." << std::endl;

// Читання JSON з файлу

std::ifstream fin("data.json");

json j\_read;

fin >> j\_read;

fin.close();

// Виведення даних з JSON

std::cout << "Ім'я: " << j\_read["name"] << std::endl;

std::cout << "Вік: " << j\_read["age"] << std::endl;

std::cout << "Місто: " << j\_read["city"] << std::endl;

std::cout << "Шлюб: " << (j\_read["married"] ? "Так" : "Ні") << std::endl;

return 0;

}

**Json** може відноситися до бінарних файлів або використовуватись у бінарному контексті:

 **Бінарні представлення JSON**: Існують бінарні формати, які базуються на JSON, такі як BSON (Binary JSON). BSON використовується, наприклад, в MongoDB для зберігання даних. BSON зберігає ті ж самі дані, що і JSON, але у бінарному форматі, що дозволяє швидше читати і записувати дані, а також зменшити розмір файлу.

 **Зберігання бінарних даних в JSON**: Іноді в JSON можуть зберігатися бінарні дані, закодовані у вигляді тексту, зазвичай за допомогою Base64. Це робиться для передачі бінарних даних (наприклад, зображень або файлів) у форматі JSON. При цьому сам JSON залишається текстовим, але вміст включає закодовані бінарні дані.

 **Використання JSON у бінарних файлах**: Деякі системи можуть використовувати JSON як частину бінарних файлів. Наприклад, JSON може бути включений у бінарний файл як метадані або конфігураційні дані. У такому випадку сам файл є бінарним, але містить в собі JSON.

**Переваги:**

1. **Легкість читання та запису**: Зрозумілий, легко читається і редагується.
2. **Гнучкість структури**: Підтримує вкладені структури, такі як масиви та об'єкти.
3. **Підтримка багатьох мов програмування**: Широко підтримується багатьма мовами програмування та платформами.

**Недоліки:**

1. **Відносно великий розмір**: Через використання текстового формату дані можуть займати більше місця порівняно з бінарними форматами.
2. **Відсутність схем**: Немає вбудованої підтримки схем для валідації структури даних, хоча можна використовувати JSON Schema.

**Використання:**

* API та веб-служби
* Зберігання конфігурацій
* Логування та аналітика

**Приклад:**

**{**

**"employees": [**

**{**

**"name": "John Doe",**

**"age": 30,**

**"position": "Developer"**

**},**

**{**

**"name": "Jane Smith",**

**"age": 25,**

**"position": "Designer"**

**}**

**]**

**}**

1. **XML (eXtensible Markup Language)**

Формат файлів, який використовується для представлення структурованих даних у вигляді тексту, що легко читається людиною.

Підтримує ієрархічні структури даних, добре підходить для представлення складних об'єктів з багатьма взаємозв'язками.

Конфігураційні файли, обмін даними між різними системами, опис даних для веб-сервісів.ї

**Приклад**:

#include <iostream>

#include <fstream>

#include "pugixml.hpp"

int main() {

// Створення нового XML документа

pugi::xml\_document doc;

pugi::xml\_node root = doc.append\_child("root");

// Додавання елементів у XML

pugi::xml\_node person = root.append\_child("person");

person.append\_child("name").append\_child(pugi::node\_pcdata).set\_value("John");

person.append\_child("age").append\_child(pugi::node\_pcdata).set\_value("30");

// Запис XML у файл

doc.save\_file("data.xml");

std::cout << "Дані були успішно записані у файл data.xml." << std::endl;

// Читання XML з файлу

pugi::xml\_document doc\_read;

if (doc\_read.load\_file("data.xml")) {

// Виведення даних з XML

pugi::xml\_node root\_read = doc\_read.child("root");

pugi::xml\_node person\_read = root\_read.child("person");

std::cout << "Ім'я: " << person\_read.child\_value("name") << std::endl;

std::cout << "Вік: " << person\_read.child\_value("age") << std::endl;

} else {

std::cerr << "Помилка читання файлу XML!" << std::endl;

return 1;

}

return 0;

}

**Переваги*:***

1. **Гнучкість структури**: Підтримує складні та вкладені структури даних.
2. **Схеми та валідація**: Підтримує схеми (XSD) для валідації структури та типів даних.
3. **Сумісність**: Широко використовується в різних галузях, особливо в інтеграційних проектах.

**Недоліки:**

1. **Складність та вербозність**: Може бути важким для читання та написання через велику кількість тегів.
2. **Розмір файлів**: Зазвичай займає більше місця через надмірне використання тегів.

**Використання:**

* Веб-служби (SOAP)
* Конфігураційні файли
* Обмін даними між різними системами

**Приклад:**

<employees>

<employee>

<name>John Doe</name>

<age>30</age>

<position>Developer</position>

</employee>

<employee>

<name>Jane Smith</name>

<age>25</age>

<position>Designer</position>

</employee>

</employees>

**Коли б я використовував кожен формат:**

* **JSON:** Ідеально підходить для веб-розробки та API, де важлива легкість читання і підтримка складних структур даних.
* **CSV:** Найкраще підходить для простих табличних даних, експорту та імпорту даних у програми для роботи з електронними таблицями та базами даних.
* **XML:** Використовав б там, де потрібна гнучкість у визначенні власних структур даних і необхідна підтримка схем валідації, наприклад, у великих корпоративних додатках та веб-сервісах.

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

Системи контролю версій (VCS) є невід'ємною частиною сучасного програмістського середовища. Вони надають засоби для ефективного керування версіями файлів, спільної роботи над проектами та відстеження змін.

**Основні поняття в системах контролю версій:**

**Репозиторій (Repository)**: це централізоване або розподілене сховище, де зберігаються всі дані проекту, включаючи всі версії файлів, історію змін, гілки (branches) і теги (tags).

**Коміт (Commit)** представляє собою конкретний стан проекту в певний момент часу. Він включає в себе зміни до файлів, які були зроблені, а також повідомлення про коміт, що пояснює, що було змінено.

**Гілка (Branch)** є окремою лінією розвитку в репозиторії. Вона дозволяє здійснювати розділення роботи між різними функціональними частинами проекту, вносити зміни незалежно від головної (master) гілки та зливати зміни пізніше.

**Злиття (Merge)** використовується для об'єднання змін з однієї гілки в іншу. Це дозволяє врахувати нові зміни та інтегрувати їх у вже існуючу лінію розвитку.

**Конфлікт (Conflict)** виникає, коли система контролю версій не може автоматично вирішити розбіжності між двома версіями файлів під час злиття. Це вимагає ручного втручання користувача для вирішення суперечностей.

**Відгалуження (Fork)**відбувається, коли користувачі або команди створюють копію репозиторію з метою незалежної роботи над проектом. Це часто відбувається у відкритих віджетах для внесення внесків у проект.

**Популярні системи контролю версій:**

**Git** є найбільш поширеною та потужною системою контролю версій, яка підтримує розподілене сховище, гілки, злиття, відгалуження та багато інших функцій. Він підтримується широким спектром платформ і використовується в багатьох великих проектах.

Основні команди:

1. **Створення нового репозиторію**

git init

Ініціалізує новий Git репозиторій у поточній директорії.

1. **Клонування існуючого репозиторію**

git clone <url>

Клонує репозиторій з віддаленого сервера до вашої локальної машини.

1. **Перевірка статусу**

git status

Показує поточний статус репозиторію, включаючи зміни, що ще не додані до комітів.

1. **Додавання змін до індексу**

git add <file>

git add .

Додає зміни у файл або всі файли до індексу (staging area).

1. **Коміт змін**

git commit -m "Опис змін"

Створює новий коміт з описом змін.

1. **Перегляд історії комітів**

git log

Виводить журнал комітів.

**Subversion (SVN)** був популярним до появи Git і використовує централізовану модель сховища. Він також підтримує версіонування файлів, гілки та злиття.

**Mercurial** є альтернативою Git, також підтримує розподілену модель роботи та має свої унікальні особливості.

**Застосування систем контролю версій:**

**Історія змін**: Зберігання всіх змін до файлів, що дозволяє повертатися до попередніх версій.

**Спільна робота**: Багатокористувацький доступ до одного і того ж репозиторію для спільної роботи над проектами.

**Ефективність управління кодом**: Зручне керування версіями програмного коду, документацією та іншими ресурсами проекту.

**Гнучкість**: Можливість ефективного вирішення конфліктів, керування гілками та відгалуженнями для різних розробок та виправлень.

Загалом, системи контролю версій є незамінним інструментом для будь-якого розробника або команди розробників, що працюють над будь-яким проектом, який потребує керування версіями, спільної роботи та ефективного управління змінами.

## 

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

Мова Markdown є простим інструментом для написання форматованого тексту з мінімальними тегами розмітки. Вона часто використовується для написання README файлів у проектах програмного забезпечення, веб-сайтів, блогів та інших документаційних матеріалів. Ось основні конструкції та приклади використання мови Markdown для опису проекту:

### Заголовки

Для створення заголовків використовується символ # з пробілом після нього. Кількість символів # відповідає рівню заголовка (від 1 до 6):

# Заголовок рівня 1

## Заголовок рівня 2

### Заголовок рівня 3

### Абзаци та текст

Простий текст відображається як звичайний абзац. Для розділення абзаців використовується порожній рядок:

Це перший абзац.

Це другий абзац.

### Списки

Markdown підтримує нумеровані та марковані списки:

Маркований список:

- Пункт 1

- Пункт 2

- Вкладений пункт

- Пункт 3

Нумерований список:

1. Перший пункт

2. Другий пункт

1. Вкладений пункт

3. Третій пункт

### Посилання

Для вставки посилань використовується синтаксис [текст посилання](URL):

Це [посилання на Google](https://www.google.com/).

### Зображення

Для вставки зображень використовується синтаксис ![альтернативний текст](URL\_зображення):

![Logo](https://example.com/logo.png)

### Код

Для відображення коду використовуються або акуратно відступи від тексту, або блоки коду з акуратним відступом з чотирьох пробілів або табуляції:

Щоб вбудувати код, використовуйте \`код\` для коротких уривків коду, або блок коду:

```python

def hello():

print("Hello, World!")

```

### Таблиці

Таблиці створюються з використанням синтаксису для стовпців та рядків:

| Заголовок1 | Заголовок2 |

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

| Клітина1 | Клітина2 |

| Клітина3 | Клітина4 |

### Цитати

Цитати використовуються за допомогою символу >:

> Це цитата. Можна використовувати кілька рядків.

### Жирний та курсив

Жирний текст відзначається подвійними зірочками \*\*жирний\*\*, а курсив одинарними \_курсив\_:

\*\*Це жирний текст\*\* і \_це курсив\_.

Ці конструкції дозволяють створювати чітку, зрозумілу та зручну для читання документацію для будь-якого проекту, використовуючи простий текстовий формат. Markdown використовується не тільки для README файлів у проектах, а й для створення блогів, статей на GitHub, розмітки у Wiki та іншого текстового контенту в інтернеті.

## 

## **4.7 Розроблення ігрової програми**

### 4.7.1 Код програми

#include <iostream>

using namespace std;

const int SIZE = 3;

char board[SIZE][SIZE] = {{'1', '2', '3'}, {'4', '5', '6'}, {'7', '8', '9'}};

void displayBoard() {

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

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

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

}

cout << endl;

}

}

bool isWinner(char player) {

// Check rows and columns

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

if ((board[i][0] == player && board[i][1] == player && board[i][2] == player) ||

(board[0][i] == player && board[1][i] == player && board[2][i] == player)) {

return true;

}

}

// Check diagonals

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

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

return true;

}

return false;

}

bool isDraw() {

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

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

if (board[i][j] != 'X' && board[i][j] != 'O') {

return false;

}

}

}

return true;

}

void makeMove(char player) {

int move;

cout << "Player " << player << ", enter your move (1-9): ";

cin >> move;

int row = (move - 1) / SIZE;

int col = (move - 1) % SIZE;

if (board[row][col] != 'X' && board[row][col] != 'O') {

board[row][col] = player;

} else {

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

makeMove(player);

}

}

int main() {

char currentPlayer = 'X';

bool gameWon = false;

while (!gameWon && !isDraw()) {

displayBoard();

makeMove(currentPlayer);

gameWon = isWinner(currentPlayer);

if (gameWon) {

displayBoard();

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

} else if (isDraw()) {

displayBoard();

cout << "The game is a draw!" << endl;

}

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

}

return 0;

}

### 4.7.2 Скріншоти виконання програми

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

# 5 РОЗДІЛ ”ВЕБ-ДИЗАЙН ТА HTML”

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

Основи HTML

HTML (HyperText Markup Language) - це основна мова для створення веб-сторінок та веб-додатків. Вона використовується для структуризації вмісту та визначення елементів на сторінці. HTML складається з набору елементів, які представлені у вигляді тегів.

Структура HTML-документа

Типовий HTML-документ складається з наступних частин:

DOCTYPE декларація: Вказує тип документа і версію HTML.

Елемент <html>: Кореневий елемент, що містить весь вміст HTML-документа.

Елемент <head>: Містить метадані про документ (наприклад, заголовок, стилі, метаінформацію).

Елемент <body>: Містить весь вміст, який відображається на веб-сторінці (тексти, зображення, посилання тощо).

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

HTML надає різноманітні теги для форматування тексту:

* **Заголовки**: <h1> до <h6>

<h1>Головний заголовок</h1>

<h2>Підзаголовок</h2>

* **Параграфи**: <p>

html

Копіювати код

<p>Це параграф тексту.</p>

* **Жирний текст**: <strong>

html

Копіювати код

<p>Це <strong>жирний текст</strong>.</p>

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

CSS (Cascading Style Sheets) - мова стилів для оформлення HTML-документів.

**Структура CSS**

CSS правило складається з селектора та набору декларацій:

селектор {

властивість: значення;

}

**Селектори**

* **Теги**: p { color: blue; }
* **Класи**: .class-name { font-size: 20px; }
* **ID**: #id-name { background-color: yellow; }
* **Комбіновані**: div p { margin: 10px; }

**Типи CSS**

1. **Вбудовані**: <p style="color: red;">Текст</p>
2. **Внутрішні**: В межах тегу <style> в HTML:

<style>

p { color: blue; }

</style>

1. **Зовнішні**: У файлі .css, підключається через <link>:

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

**Каскадність та успадкування**

* **Каскадність**: Стилі переписуються залежно від специфічності та порядку.
* **Успадкування**: Деякі властивості успадковуються від батьківських елементів.

**Приклад:**

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

/\* styles.css \*/

p {

color: green;

}

CSS дозволяє створювати красиві та структуровані веб-сторінки за допомогою стилів.

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

**Спадковість**

Деякі CSS властивості передаються від батьківських елементів до дочірніх.

body {

font-family: Arial, sans-serif;

}

p {

color: blue;

}

**Каскадування**

CSS правила застосовуються за пріоритетом селекторів і порядком визначення.

p {

color: blue; /\* менш специфічний \*/

}

p.special {

color: red; /\* більш специфічний \*/

}

**Пріоритет селекторів**

1. Вбудовані стилі (inline): style="color: green;"
2. ID селектори: #id
3. Класові селектори: .class
4. Селектори тегів: p

## **5.5 Практична частина**

Зробити сайт

### 5.5.1 Постановки задачі

Зробити сайт про себе з посиланнями на свої соц мережі

5.5.2 HTML документ

**HTML**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Перевозник Назар - CV</title>

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

    <script src="https://kit.fontawesome.com/5366f83049.js" crossorigin="anonymous"></script>

</head>

    <meta charset="utf-8">

    <style>

     \* {

      margin: 0;

      padding: 0;

     }

    </style>

</head>

<body>

    <div class="container">

        <div class="header">

            <img src="фото 3.4.jpg" alt="Profile Image" class="profile-img">

            <h1>Perevoznyk Nazar</h1>

            <h2>Student</h2>

        </div>

        <div class="summary">

            <h3>Summary</h3>

            <hr style="color: rgb(0, 0, 0)">

            <p>I do not have any work experience at the moment, but I am a college student studying Python, C++, HTML, and Css. I have strong analytical thinking, attention to detail, perseverance, communication skills, creativity, self-organization, and a willingness to cooperate. I am getting better every day, learning new features and avoiding fewer and fewer mistakes</p>

        </div>

        <br/>

        <div class="education">

            <h3>Education</h3>

            <hr style="color: rgb(0, 0, 0)">

            <p>2013-2022. Secondary general education school №66</p>

            <p>2022-2026. Optical and Mechanical College of Taras Shevchenko Kyiv National University</p>

        </div>

        <br/>

        <div class="experience">

            <h3>Hobbies</h3>

            <hr style="color: rgb(0, 0, 0)">

            <div class="job">

                <p>

                    My hobbies are writing music, in particular writing instrumentals and playing the piano. I love learning different languages, including English, Polish and French. I am a big fan of music and movies. I also collect speed puzzles, including rubik's cubes.</p>

            </div>

        </div>

        <div class="contact">

        <a href="mail to: nazarcheekk@gmail.com" class="btn mail" target="\_blank"> <i class="fa-regular fa-envelope"></i> Mail</a>

        <br/>

        <a href="https://www.instagram.com/nazarcheekk/" class="btn instagram" target="\_blank"> <i class="fa-brands fa-instagram"></i> Instagram</a>

    </div>

</body>

</html>

**CSS**

body {

    font-family: Arial, sans-serif;

    margin: 0;

    padding: 0;

    background-color: #F2F3FB;

}

.container {

    width: 70%;

    margin: 0 auto;

    padding: 20px;

    background-color: #F2F3FB;

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

}

.header {

    text-align: center;

    margin-bottom: 20px;

}

.profile-img {

    width: 150px;

    height: 200px;

    border-radius: 50%;

}

h1, h2 {

    margin: 10px 0;

}

h3 {

    border-bottom: 2px solid #F2F3FB ;

    padding-bottom: 5px;

    margin-bottom: 10px;

}

p {

    line-height: 1.7;

    margin-bottom: 10px;

}

.experience .job {

    margin-bottom: 30px;

}

.contact {

    text-align: center;

    margin-top: 20px;

}

.btn {

    display: inline-block;

    margin: 5px;

    padding: 20px 450px;

    border: 1px solid black;

    border-radius: 5px;

    color: black;

    background-color: none;

    text-decoration: none;

    position: relative;

    padding-left: 40px;

}

.icon1{

    position: relative;

    cursor: pointer;

}

.icon1 {

        content: "\f0e0";

        font-weight: 900;

        position: absolute;

        left: 10px;

        top: 50%;

        transform: translateY(-50%);

    }

.btn.email::before {

    content: "";

    width: 200px;

}

.btn.instagram::before {

    content: "";

    width: 200px;

}

.btn:hover {

    background-color: black;

    color: white;

}

# ВИСНОВКИ

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ

1. ГОСТ 2.105-95 ЕСКД

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

# ДОДАТКИ