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 |
| Завадського Олександра Костянтиновича  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) | |  | |
| *Семенової Валерії Сергіївни* | | | | |
| (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](#_Toc168176310)

[1 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ 7](#_Toc168176311)

[2 РОЗДІЛ "ІНФОРМАТИКА" 9](#_Toc168176312)

[**2.1** **Розроблення засобами MS Word документа складної структури** 9](#_Toc168176313)

[**2.2 Розв‘язок задач за допомогою засобів табличного редактора MS Excel** 9](#_Toc168176314)

[**2.3 Виконання** **практичних завдань за допомогою засобів табличного редактора MS Access** 9](#_Toc168176315)

[3 РОЗДІЛ "ПРОГРАМУВАННЯ С++" 10](#_Toc168176316)

[**3.1 Теоретичні відомості** 10](#_Toc168176317)

[3.1.1 Загальні поняття. Елементи мови С++ - константи, змінні, операції, перетворення типів 10](#_Toc168176318)

[3.1.2 Арифметичні операції 10](#_Toc168176319)

[3.1.3 Поняття вхідного та вихідного потоку, найпростіші математичні функції 10](#_Toc168176320)

[3.1.4 Програмування лінійних та розгалужених обчислювальних процесів 10](#_Toc168176321)

[3.1.5 Програмування циклічних обчислювальних процесів 10](#_Toc168176322)

[**3.2 Практичне завдання** 11](#_Toc168176323)

[3.2.1 Завдання 11](#_Toc168176324)

[3.2.2 Постановка задачі 11](#_Toc168176325)

[3.2.3 Код програми 11](#_Toc168176326)

[3.2.4 Скріншот виконання програми 11](#_Toc168176327)

[**3.3 Виконання тестів** 11](#_Toc168176328)

[4 РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++" 12](#_Toc168176329)

[**4.1 Тема 1. Введення та виведення у базових консольних програмах** 12](#_Toc168176330)

[**4.2 Тема 2. Структури даних для управління станом програми** 16](#_Toc168176331)

[**4.3 Тема 3. Робота з файлами: запис та читання** 20](#_Toc168176332)

[**4.4 Тема 4. Формати файлів та їх обробка** 22](#_Toc168176333)

[**4..5 Тема 5. Системи контролю версій (VCS)** 22](#_Toc168176334)

[**4.6 Тема 6. Мова Markdown для опису проекті** 26](#_Toc168176335)

[**4.7 Розроблення ігрової програми** 26](#_Toc168176336)

[4.7.1 Код програми 26](#_Toc168176337)

[4.7.2 Скріншоти виконання програми 29](#_Toc168176338)

[5 РОЗДІЛ ”ВЕБ-ДИЗАЙН ТА HTML” 31](#_Toc168176339)

[**5.1 Мова розмітки HTML та структура Веб-сторінки** 31](#_Toc168176340)

[**5.2 Розмітка тексту за допомогою HTML. Зображення та посилання** 33](#_Toc168176341)

[Абзаци 33](#_Toc168176342)

[Цитати 35](#_Toc168176343)

[**5.3 Каскадні таблиці стилів CSS та їх структура** 36](#_Toc168176344)

[**5.4 Спадковості і каскадування за допомогою CSS** 38](#_Toc168176345)

[5.5.1 Постановки задачі 39](#_Toc168176346)

[5.5.2 HTML документ 40](#_Toc168176347)

[ВИСНОВКИ 45](#_Toc168176348)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ 46](#_Toc168176349)

[ДОДАТКИ 48](#_Toc168176350)

ОМФК.0121 НП 20

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

# ВСТУП

Автор: Завадський Олександр Костянтинович

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**

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

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

# 3 РОЗДІЛ "ПРОГРАМУВАННЯ С++"

## **3.1 Теоретичні відомості**

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

### 

### 3.1.2 Арифметичні операції

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

### 3.1.4 Програмування лінійних та розгалужених обчислювальних процесів

### 3.1.5 Програмування циклічних обчислювальних процесів

## **3.2 Практичне завдання**

### 3.2.1 Завдання

### 3.2.2 Постановка задачі

Вхідні дані

Вихідні дані

Блок-схема алгоритму роботи програми представлена на рис. ххх

Блок-схема

Рисунок хх - Блок-схема алгоритму роботи програми

### 3.2.3 Код програми

### 3.2.4 Скріншот виконання програми

Скріншот виконання програми представлена на рис. ххх

Скріншот

Рисунок хх - Скріншот виконання програми

## **3.3 Виконання тестів**

# 

# 4 РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++"

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

У 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**

У темі "Структури даних для управління станом програми" основні поняття включають опис найпоширеніших структур даних у C++ та їхні реалізації.

Масив є інструментом для представлення однорідних даних. Він складається з елементів, кожен з яких має індекс. Властивості масиву включають тип елементів, назву, розмірність і діапазони індексів.

У C++ індекси починаються з 0. Звернення до елемента масиву здійснюється за допомогою індексів у квадратних дужках.

Одновимірний масив, або вектор, має один індекс для кожного елемента. Двовимірний масив, або матриця, має два індекси: номер рядка та номер стовпчика.

**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.

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

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** — лінійно упорядкована структура (послідовність) даних, у яку можна вставити елемент або з якої можна вилучити елемент лише з одного кінця.

Стек нагадує стос аркушів паперу у вузькій шухляді: щоб узяти певний аркуш, потрібно спочатку прибрати всі аркущі над ним. Кажуть, що для стеку діє принцип: «Останній зайшов, перший вийшов» (англій­ською LIFO: Last In First Out).

**Изображение выглядит как снимок экрана, линия, дизайн

Автоматически созданное описание**

Основні операції зі стеком:

* вставлення елемента в кінець стеку;
* вилучення елемента з кінця;
* відображення вмісту останнього долученого елемента.

Наявні вбудовані функції для роботи з векторами дозволяють втілити ідею стеку. Контейнер stack безпосередньо втілює принцип: «Останній зайшов, перший вийшов» (англій­ською LIFO: Last In First Out):

**Черга** — лінійно упорядкована структура (послідовність) даних, у яку можна вставити елемент з одного кінця або з якої можна вилучити елемент з іншого кінця

Изображение выглядит как Шрифт, снимок экрана, текст, диаграмма

Автоматически созданное описание

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.

Изображение выглядит как текст, Шрифт, белый, дизайн

Автоматически созданное описание

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

**Set** — контейнер для збереження унікальних значень (кожне не більше одного разу), що втілює математичне поняття множини.

При додаванні нового елемента в даному контейнері буде проведено упорядкування елементів. Повторювані елементи буде вилучено.

## **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.

Такі сайти, як GitHub, Reddit та Stack Overflow використовують Markdown для полегшення обговорень між користувачами.

Ось декілька основних функцій мови Markdown, які були використані у оформлені мого репозиторію:

* Заголовки

Для того що б додати заголовок потрібно додати від одного до шести #-символів перед тестом заголовка. Кількість #-символів, яка використовується, визначатиме рівень ієрархії та розмір гарнітури заголовка. Наприклад:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Коли використовуєтсья два або більше заголовків, GitHub автоматично створює зміст, до якого можна отримати доступ, клацнувши в заголовку файлу. Кожна назва заголовка перерахована у змісті, і користувач може клацнути назву, щоб перейти до вибраного розділу.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

* 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 ().

Изображение выглядит как текст, мультфильм, графическая вставка, Анимация

Автоматически созданное описание

* Список

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

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Щоб упорядкувати список, перед кожним рядком слід додати номер.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

## **4.7 Розроблення ігрової програми**

### 4.7.1 Код програми

use serde::{Deserialize, Serialize};  
use std::fs::{self, File};  
use std::io::{self, Write};  
use std::path::Path;  
  
const WINS\_FILE: &str = "wins.json";  
const INVALID\_INPUT\_MSG: &str = "Invalid input! Please enter numbers between 0 and 2.";  
const OCCUPIED\_MSG: &str = "Occupied. Please choose another one.";  
const PLAYER\_1\_SYMBOL: char = 'O';  
const PLAYER\_2\_SYMBOL: char = 'X';  
  
#[derive(Serialize, Deserialize, Debug)]  
struct Wins {  
 X: i32,  
 O: i32,  
}  
  
impl Wins {  
 fn new() -> Self {  
 Wins { X: 0, O: 0 }  
 }  
}  
  
fn read\_wins\_from\_file() -> Wins {  
 if Path::new(WINS\_FILE).exists() {  
 let data = fs::read\_to\_string(WINS\_FILE).expect("Unable to read file");  
 serde\_json::from\_str(&data).expect("Unable to parse JSON")  
 } else {  
 Wins::new()  
 }  
}  
  
fn write\_wins\_to\_file(wins: &Wins) {  
 let data = serde\_json::to\_string(wins).expect("Unable to serialize data");  
 fs::write(WINS\_FILE, data).expect("Unable to write file");  
}  
  
fn check\_winner(current\_player: i32, player1: &mut i32, player2: &mut i32, wins: &mut Wins) {  
 if current\_player % 2 == 0 {  
 println!("Player 2 wins!");  
 \*player2 += 1;  
 wins.X += 1;  
 } else {  
 println!("Player 1 wins!");  
 \*player1 += 1;  
 wins.O += 1;  
 }  
}  
  
fn annulate(board: &mut Vec<Vec<char>>) {  
 for row in board.iter\_mut() {  
 for cell in row.iter\_mut() {  
 \*cell = ' ';  
 }  
 }  
}  
  
fn print\_board(board: &Vec<Vec<char>>) {  
 for (i, row) in board.iter().enumerate() {  
 for (j, cell) in row.iter().enumerate() {  
 print!("{} ", cell);  
 if j < 2 {  
 print!("| ");  
 }  
 }  
 println!();  
 if i < 2 {  
 println!("---------");  
 }  
 }  
}  
  
fn print\_score(player1: i32, player2: i32) {  
 println!("Player 1: {} | Player 2: {}", player1, player2);  
}  
  
fn main() {  
 let mut wins = read\_wins\_from\_file();  
 let mut player1 = 0;  
 let mut player2 = 0;  
 let mut board = vec![vec![' '; 3]; 3];  
 let mut current\_player = 1;  
  
 loop {  
 println!("Enter the field (row and column): ");  
 io::stdout().flush().unwrap();  
  
 let mut input = String::new();  
 io::stdin().read\_line(&mut input).unwrap();  
  
 let mut iter = input.trim().split\_whitespace();  
 let row: usize = match iter.next() {  
 Some(value) => match value.parse() {  
 Ok(num) => num,  
 Err(\_) => {  
 println!("{}", INVALID\_INPUT\_MSG);  
 continue;  
 }  
 },  
 None => {  
 println!("{}", INVALID\_INPUT\_MSG);  
 continue;  
 }  
 };  
  
 let column: usize = match iter.next() {  
 Some(value) => match value.parse() {  
 Ok(num) => num,  
 Err(\_) => {  
 println!("{}", INVALID\_INPUT\_MSG);  
 continue;  
 }  
 },  
 None => {  
 println!("{}", INVALID\_INPUT\_MSG);  
 continue;  
 }  
 };  
  
 if row >= 3 || column >= 3 {  
 println!("{}", INVALID\_INPUT\_MSG);  
 continue;  
 }  
  
 if board[row][column] != ' ' {  
 println!("{}", OCCUPIED\_MSG);  
 continue;  
 }  
  
 if current\_player % 2 == 0 {  
 board[row][column] = PLAYER\_2\_SYMBOL;  
 } else {  
 board[row][column] = PLAYER\_1\_SYMBOL;  
 }  
  
 print\_board(&board);  
  
 for i in 0..3 {  
 if (board[i][0] == board[i][1] && board[i][1] == board[i][2] && board[i][0] != ' ')  
 || (board[0][i] == board[1][i] && board[1][i] == board[2][i] && board[0][i] != ' ')  
 {  
 check\_winner(current\_player, &mut player1, &mut player2, &mut wins);  
 print\_score(player1, player2);  
 annulate(&mut board);  
 current\_player = 1;  
 write\_wins\_to\_file(&wins);  
 continue;  
 }  
 }  
  
 if (board[0][0] == board[1][1] && board[1][1] == board[2][2] && board[0][0] != ' ')  
 || (board[0][2] == board[1][1] && board[1][1] == board[2][0] && board[0][2] != ' ')  
 {  
 check\_winner(current\_player, &mut player1, &mut player2, &mut wins);  
 print\_score(player1, player2);  
 annulate(&mut board);  
 current\_player = 1;  
 write\_wins\_to\_file(&wins);  
 continue;  
 }  
  
 let mut draw = true;  
 for row in &board {  
 for cell in row {  
 if \*cell == ' ' {  
 draw = false;  
 }  
 }  
 }  
 if draw {  
 println!("Draw!");  
 annulate(&mut board);  
 current\_player = 1;  
 continue;  
 }  
  
 current\_player += 1;  
 }  
}

# ВИСНОВКИ

# 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>

# ДОДАТКИ