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 |
| Cеменової Валерії Сергіївни  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-21

ОМФК.0121 НП 20

Зміст

[ВСТУП 6](#_Toc167969935)

[1 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ПРИ РОБОТІ З КОМП’ЮТЕРОМ 7](#_Toc167969936)

[2 РОЗДІЛ "ІНФОРМАТИКА" 9](#_Toc167969937)

[**2.1** **Розроблення засобами MS Word документа складної структури** 9](#_Toc167969938)

[**2.2 Розв‘язок задач за допомогою засобів табличного редактора MS Excel** 9](#_Toc167969939)

[**2.3 Виконання** **практичних завдань за допомогою засобів табличного редактора MS Access** 9](#_Toc167969940)

[3 РОЗДІЛ "ПРОГРАМУВАННЯ С++" 10](#_Toc167969941)

[**3.1 Теоретичні відомості** 10](#_Toc167969942)

[3.1.1 Загальні поняття. Елементи мови С++ - константи, змінні, операції, перетворення типів 10](#_Toc167969943)

[3.1.2 Арифметичні операції 10](#_Toc167969944)

[3.1.3 Поняття вхідного та вихідного потоку, найпростіші математичні функції 10](#_Toc167969945)

[3.1.4 Програмування лінійних та розгалужених обчислювальних процесів 10](#_Toc167969946)

[3.1.5 Програмування циклічних обчислювальних процесів 10](#_Toc167969947)

[**3.2 Практичне завдання** 11](#_Toc167969948)

[3.2.1 Завдання 11](#_Toc167969949)

[3.2.2 Постановка задачі 11](#_Toc167969950)

[3.2.3 Код програми 11](#_Toc167969951)

[3.2.4 Скріншот виконання програми 11](#_Toc167969952)

[**3.3 Виконання тестів** 11](#_Toc167969953)

[4 РОЗДІЛ " ПРАКТИЧНА РОБОТА З С++" 12](#_Toc167969954)

[**4.1 Тема 1. Введення та виведення у базових консольних програмах** 12](#_Toc167969955)

[**4.2 Тема 2. Структури даних для управління станом програми** 16](#_Toc167969956)

[**4.3 Тема 3. Робота з файлами: запис та читання** 20](#_Toc167969957)

[**4.4 Тема 4. Формати файлів та їх обробка** 22](#_Toc167969958)

[**4..5 Тема 5. Системи контролю версій (VCS)** 22](#_Toc167969959)

[**4.6 Тема 6. Мова Markdown для опису проекті** 26](#_Toc167969960)

[**4.7 Розроблення ігрової програми** 29](#_Toc167969961)

[4.7.1 Код програми 29](#_Toc167969962)

[4.7.2 Скріншоти виконання програми 30](#_Toc167969963)

[5 РОЗДІЛ ”ВЕБ-ДИЗАЙН ТА HTML” 32](#_Toc167969964)

[**5.1 Мова розмітки HTML та структура Веб-сторінки** 32](#_Toc167969965)

[**5.2 Розмітка тексту за допомогою HTML. Зображення та посилання** 34](#_Toc167969966)

[Абзаци 34](#_Toc167969967)

[Цитати 35](#_Toc167969968)

[**5.3 Каскадні таблиці стилів CSS та їх структура** 36](#_Toc167969969)

[**5.4 Спадковості і каскадування за допомогою CSS** 38](#_Toc167969970)

[5.5.1 Постановки задачі 40](#_Toc167969971)

[5.5.2 HTML документ 40](#_Toc167969972)

[ВИСНОВКИ 46](#_Toc167969973)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ 47](#_Toc167969974)

[ДОДАТКИ 49](#_Toc167969975)

ОМФК.0121 НП 20

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

# ВСТУП

Автор: Семенова Валерія Сергіївна

Дата: 30.05.2024

Мета: отримати практичні навички написання коду, а саме коду гри, мовою С++

Перелік вивчених тем:

Тема 1. Введення та виведення у базових консольних програмах

Тема 2. Структури даних для управління станом програми

Тема 3. Робота з файлами: запис та читання

Тема 4. Формати файлів та їх обробка

Тема 5. Системи контролю версій (VCS)

Тема 6. Мова Markdown для опису проектів

# 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**

Об’єкт cout виводить дані на заданий пристрій.

Синтаксис:

cout << data[ << data ],  
де data – змінні, константи, вирази.

Изображение выглядит как текст, Шрифт, снимок экрана, документ

Автоматически созданное описание

Для cout використовуються різноманітні маніпулятори. Наприклад, маніпулятори hex і oct використовуються для виведення відповідно, 16-х та 8-х чисел.

Изображение выглядит как текст, чек, Шрифт, снимок экрана

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, белый, чек

Автоматически созданное описание

Маніпулятор установлює виведення 8 та 16–річних чисел, поки не зустрінеться інший маніпулятор oct, hex, dec.

Маніпулятори setw(), setprecision() і setfill() – це функції-члени які змінюють стан об’єкта cout.

setw() – вирівнює число виведення в межах заданої ширини.

setfill() – заповнює незаповнені позиції встановленим символом. При цьому формуються пробіли зліва.

setprecision(n) – маніпулятор виведення зазначеної кількості знаків n.

Изображение выглядит как текст, Шрифт, снимок экрана, алгебра

Автоматически созданное описание

Маніпулятори setiosflags і resetiosflags використовуються для установки певних глобальних прапорів, які клас C++ iostream використовує при визначенні поведінки по умовчанню при введенні та виведенні. На ці прапори посилаються як на змінні стану. Функція setiosflags() встановлює зазначені в ній прапори, а resetiosflags() очищує (або скидає) їх.

Изображение выглядит как текст, Шрифт, чек, снимок экрана

Автоматически созданное описание

Для виведення даних у класичному C використовується функція printf. Вона без змін перенесена у С++.

Синтаксис:

printf(char \*fotmat, <додаткові аргументи>)

Рядок формату складається із символів виведених без зміни та спеціальних символів формату, які задають перетворення даних. Для такого перетворення використовуються додаткові аргументи.

Специфікація перетворення формату має вигляд:

%[ прапори ] [ширина] [. точність] [ L|l ] тип, де прапори:

– (мінус) - вирівнювання по лівому краю поля;

+ (плюс) - виводиться знак числа + або –;  
ширина: визначає мінімальне число виведених символів, що

доповнюються пробілами або нулями.  
точність: визначає число знаків після десяткової точки для чисел із

плаваючою точкою. Для змінних цілого типу визначається максимальне число виведених цифр.

тип:  
d або i -десяткове ціле зі знаком (int);  
u – десяткове ціле без знака (unsigned int);  
x – шістнадцятирічне ціле без знака (unsigned int);  
f – виведення числа з плаваючою точкою (float, double);  
e – виведення числа з плаваючою точкою із використанням

експоненціальної форми (float, double);

c – виведення символу (char); s – виведення рядка;  
p – виведення по вказівці.

L | l – застосовується в комбінації із символом типу:  
Lf і Le - довгий із плаваючою точкою (long double); ld або li - довгий цілий (long int);  
lu – беззнаковий довгий цілий (unsigned long int).

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Введення даних здійснюється з використанням об’єкта cin. Синтаксис: cin >> змінна [ >> змінна ]...

При введенні рядків введення здійснюється тільки по одному слову. При введенні декількох змінних роздільник між ними - пробіл або <Enter>.

Изображение выглядит как текст, чек, Шрифт, снимок экрана

Автоматически созданное описание

Функція scanf є одною із багатьох функцій введення, які мають місце в зовнішніх бібліотеках. Кожній змінній, що вводиться у рядку функції scanf повинна відповідати специфікація (див. функцію виведення даних printf). Перед іменами змінних треба ставити символ &. Цей символ означає «взяти адресу».

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

Опис структури даних є важливою складовою розробки програмного забезпечення. Розглянемо кілька найпоширеніших структур даних та їхню реалізацію мовою C++. Типовим інструментом для представлення однорідних даних є масив.

**Масив** — згруповані за місцем розташування у пам'яті величини, що мають одну назву (ідентифікатор) і різні порядкові номери (індекси).Це поняття програмування відповідає математичним поняттям послідовності й таблиці (матриці).

Елемент масиву — одна з величин, що утворюють масив. Це поняття програмування відповідає математичному поняттю елемента послідовності чи матриці.

Індекс масиву — величина перелічуваного (зазвичай цілого) типу, яка (сукупність яких) вказує на конкретний елемент масиву. Це поняття програмування відповідає математичному поняттю номера елемента послідовності чи номеру рядка/стовпчика таблиці (матриці). У мові C++ найменше значення індексу — 0.  
Масив має такі властивості:

* тип (типи) елементів (масиву);
* назва (ідентифікатор) — одна й та сама для всіх елементів масиву;
* розмірність — кількість номерів (індексів), необхідних для визначення розташу­вання елемента у масиві;
* діапазони зміни номерів (індексів).

У мові C++ властивості елементів масиву не можна змінити протягом виконання програми. В деяких інших мовах це можливо. Cукупність розмірності й діапазонів називають формою масиву.

Звернення до значення елемента масиву у коді програми здійснюють, вказавши після його назви у квадратних дужках індекси у вигляді (арифметичних) виразів, що набувають значень з діапазопу порядкового типу. Наприклад, а[3] — 3-тій елемент масиву а, с[j] — j-ий елемент масиву с.

Елемент масиву може мати і тип простої (неструктурованої) величини і складений тип (масиву, рядка тощо), тобто може існувати масив масивів, масив рядків тощо. Глибину вкладеності (а значить, і кількість індексів) необмежено, але загальну довжину структури обмежено. У пам'яті ПК всі елементи масиву зберігають послідовно: при переході від молодших до старших адрес першим буде змінено індекс, записаний останнім (праворуч).

Під час розв'язування задач переважно використовують одновимірні та двовимірні масиви.

Одновимірний масив інакше називають лінійним масивом або вектором. Кожному його елементу ставлять у відповідність один індекс. У математиці лінійному масиву відповідає поняття послідовності, а номеру члена послідовності — індекс масиву.

\Двовимірний масив інакше називають таблицею або матрицею. Кожному його елементу ставлять у відповідність два індекси: номер рядка та номер стовпчика. Прикладом унаочнення двовимірного масиву є шахівниця.

**Зв'язний список** — лінійно упорядкована структура (послідовність) даних, елементи якої — вузли — містить два види даних:

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

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).

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

**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):

Изображение выглядит как текст, Шрифт, чек, алгебра

Автоматически созданное описание

**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):

Изображение выглядит как текст, Шрифт, снимок экрана, чек

Автоматически созданное описание

**Хеш-таблиця (hash table)** — це структура даних для зберігання пар ключ-значення.

**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.

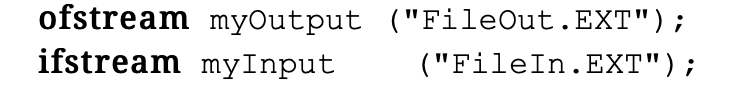
Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

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

У мові С++ введення/виведення описується як набір класів, описаний в заголовному файлі  iostream.h. Аналогами потоків stdin, stdout, stderr є класи cin, cout і cerr. Ці три потоки відкриваються автоматично. Потік cin пов'язаний з клавіатурою, а cout, cerr – з дисплеєм.

Файл <fstream.h> визначає класи ifstream і ofstream, за допомогою яких програма може виконувати операції файлового введення/виведення. Для відкриття файлу на введення/виведення оголошують об'єкт типу ifstream/  ofstream, передаючи конструктору цього об'єкта ім'я необхідного файлу:



Після того, як програма відкрила файл для введення або виведення, вона може читати або писати дані, використовуючи оператори: "<<" – для занесення (запису) в потік; ">>" – для вилучення (читання) з потоку.

Изображение выглядит как текст, Шрифт, чек, белый

Автоматически созданное описание

Більшість програм читають вміст файлу, поки не зустрінеться кінець файлу. Визначити кінець файлу можна за допомогою функції **eof().**

Для введення або виведення символів у файл або з файлу використовую функції **get()** і **put()**.

Изображение выглядит как текст, чек, Шрифт, белый

Автоматически созданное описание

Для зчитування цілого рядка використовують функцію **getline()**:

Изображение выглядит как текст, Шрифт, снимок экрана, белый

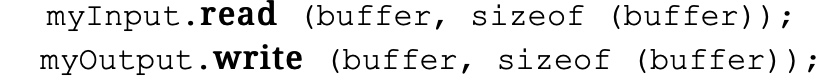
Автоматически созданное описание

Для перевірки помилок можна використовувати функцію **fail()**:

Изображение выглядит как текст, Шрифт, снимок экрана, белый

Автоматически созданное описание

Якщо програмам необхідно вводити або виводити такі дані, як структури або масиви, можна використовувати методи read() і write().



Якщо програма завершила роботу з файлом, його слід закрити за допомогою функції close().

Для того, щоб операції введення/виведення виконувалися не з початку файлу, можна використовувати інші режими відкриття файлів.

Наприклад,

    ifstream myFile ("Filename.txt", ios::out | ios::noreplace);

Для читання і запису даних будь-якого типу, тип яких може займати більше 1 байта, у файловій системі мови С є дві функції: **fread()** і **fwrite()**.

     size\_t **fread**(void \*buf,size\_t count,size\_t k,FILE \*pf);

     size\_t **fwrite**(const void \*buf,size\_t count,size\_t k,FILE \*pf);

Функція fread() повертає кількість прочитаних елементів. Якщо досягнуто кінець файлу або сталася помилка, то повернуте значення може бути менше, ніж лічильник. А функція fwrite() повертає кількість запи­саних елементів. Якщо помилки не було, то повернений результат буде дорівнює значенню лічильник. Одним з найбільш корисних застосувань функцій fread() і fwrite() є читання і запис даних користувача типів. Наприклад, якщо визначена структура

**struct** struct\_type {  
            float balance;  
            char name [80];  
      } Cust;

то наступний оператор записує вміст Сust у файл, на який вказує fp:

**fwrite** (&cust, sizeof (struct struct\_type), 1, fp);

В системі введення/виведення мови С є функції **fprintf()** і **fscanf()**:

      int  **fprintf** (FILE \* pf, const char \*str, ...);  
      int  **fscanf** (FILE \* pf, const char \*str, ...);

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

Формати файлів

Текстові формати

1 TXT - простий текстовий файл без форматування.

2 CSV - файл, де дані зберігаються у вигляді таблиці з роздільниками-комами.

3 JSON - текстовий формат для зберігання та передачі структурованих даних.

4 XML - розширюваний мова розмітки для зберігання та передачі даних.

Двійкові формати

1 BIN - загальний формат для зберігання будь-яких даних у вигляді байтів.

2 EXE - виконувані файли, що містять машинний код.

3 DLL - бібліотеки динамічних посилань, які містять код та дані для використання іншими програмами.

Файли різних форматів використовуються для зберігання різноманітної інформації. У C++ робота з файлами включає як текстові, так і двійкові файли. Текстові файли зберігають дані у вигляді тексту, тоді як двійкові файли зберігають дані у вигляді байтів, що дозволяє зберігати більш складні структури даних.

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

Що таке система контролю версій

**Система контролю версій** (СКВ від англ. Version Control System, VCS) -це спеціалізоване місце зберігання коду, система, що записує зміни у файл, чи набір файлів протягом часу і дозволяє повернутися пізніше до певної версії.

Інше формулювання звучить так:

**СКВ** - це програмне забезпечення, яке допомагає розробникам програм співпрацювати та вести повну історію своєї роботи. Воно може зберігати зміни файлів та модифікації вихідного коду. Щоразу, коли користувач вносить зміни в проект, СКВ приймає стан проекту та зберігає їх. Ці різні збережені стани проекту відомі як версії.

Системи **VCS** інколи називають інструментом **SCM** (source code management - управління початковим кодом) або **RCS** (revision control system - система управління редакціями).

**Контроль версії -** це практика відстеження змін програмного коду та управління ним. Системи контролю версії — це програмні інструменти, які допомагають командам розробників керувати змінами в вихідному коді з течією часу. В умовах праці вони допомагають команді розробників працювати швидше і ефективно. Системи контролю версії найбільш корисні для команд **DevOps**, оскільки допомагають скоротити час розробки та збільшити кількість успішних розгортань.

Програмне забезпечення контролю версій відстежує всі зміни, що вносяться в код, у спеціальній базі даних. При виявленні помилки розробники можуть повернутися назад і порівняти з попередніми версіями коду для виправлення помилок, зводячи до мінімуму проблеми для всіх учасників команди.

Усі системи контролю версій по суті вирішують 4 задачі.

1. **Доступ до коду.** Вихідники коду зберігаються у віддаленому репозиторії (сховищі даних), куди звертаються розробники, щоб забрати актуальну версію файлів або внести зміни. Так вибудовується командна технологія.

2. **log-ування змін у коді.** Відстеження **commit-**ів (внесення змін до коду) допомагає знайти хто, що і коли змінював, вирішити конфлікти при модифікуванні одних і тих же файлів, відкотитися на будь-який попередній стан.

3. **Розгалуження розробки.** Програмісти паралельно ведуть розробку нового функціоналу у окремих гілках, не торкаючись працездатності старого.

4. **Підтримка версії продуктів.** При випуску оновлень програмних продуктів ми позначаємо релізні версії, наприклад, за допомогою **tag**-ів, щоб зафіксувати їх у цьому стані, для дебагу або ретроспективи.

На сьогодні існує розподіл СКВ на три головні категорії:

-  **Локальні;**

-  **клієнт-серверні (централізовані);**

-  **розподілені (децентралізовані).**

Локальні системи контролю версій

Кожен користувач як метод контролю версій застосовує копіювання файлів в окремі каталоги, але такий підхід є недосконалим, оскільки не дозволяє відслідковувати всі важливі зміни та відповідно уникнути різник помилок в роботі з файлами, що погіршує якість та продуктивність праці і особливо в складних проектах.

Изображение выглядит как текст, Шрифт, число, снимок экрана

Автоматически созданное описание

Для того, щоб вирішити цю проблему, програмісти давно розробили локальні СКВ з простою базою даних, яка зберігає записи про всі змінив файлах, здійснюючи тим самим контроль ревізій.

Централізовані системи контролю версій

Для вирішення проблеми необхідності тісної взаємодії з іншими розробниками проекту, були розроблені централізовані системи контролю версій (ЦСКВ). Такі системи, як **CVS, Subversion** і **Perforce**, використовують єдиний сервер, що містить всі версії файлів, і кілька клієнтів, які отримують файли з цього централізованого сховища. Застосування ЦСКВ було стандартом багато років.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Централізована СКВ має переваги над іншими СКВ, особливо над локальним. Наприклад, всі розробники проекту певною мірою знають, чим займається кожен із них. Адміністратори мають повний контроль над тим, хто і що може робити і також набагато простіше адмініструвати ЦСКВ, ніж оперувати локальними базами даних на кожному клієнті.

Мінуси – єдина точка відмови представлена централізованим сервером.

Якщо цей сервер вийде з ладу на годину, протягом цього часу ніхто не зможе використовувати контроль версій для збереження змін, над якими працює, а також ніхто не зможе обмінюватися цими змінами з іншими розробниками.

Якщо жорсткий диск, на якому зберігається центральна БД, пошкоджений, а своєчасні бекапи відсутні, ви втратите весь проект разом з історією проекту, не враховуючи одиничних знімків репозиторію, які збереглися на локальних машинах розробників.

Локальні СКВ страждають від тієї самої проблеми: коли вся історія проекту зберігається в одному місці, ви ризикуєте втратити все.

Розподілені системи контролю версій

Розподілені СКВ (РСКВ) вже долають головні недоліки локальних та централізованих СКВ. У РСКВ (таких як Git, Mercurial, Bazaar або Darcs) клієнти не просто завантажують знімок всіх файлів (стан файлів на певний момент часу) - вони повністю копіюють репозиторій. У цьому випадку, якщо один із серверів, через який розробники обмінювалися даними, помре, будь-який репозиторій клієнта може бути скопійований на інший сервер для продовження роботи. Кожна копія репозиторію є повним бекапом всіх даних.

Изображение выглядит как текст, Шрифт, дизайн, снимок экрана

Автоматически созданное описание

Ще однією з переваг РСКВ є те, що більшість з таких систем можуть одночасно взаємодіяти з декількома віддаленими репозиторіями. Завдяки цьому ви можете працювати з різними групами людей, застосовуючи різні підходи одночасно в рамках одного проекту. Це дозволяє застосовувати відразу кілька підходів у розробці, наприклад, ієрархічні моделі, що не можливо в централізованих системах.

Історія розвитку систем контролю версій

Найвідоміші СКВ за технічними ознаками історично поділяють на три покоління.

Системи контролю версій (VCS) першого покоління відслідковували зміни в окремих файлах, а редагування підтримувалося лише локально та одним користувачем за один раз. Системи будувалися на припущенні, що всі користувачі заходитимуть по своїх облікових записах на той самий загальний вузол Unix.

У VCS другого покоління з'явилася підтримка мережі, що призвело до централізованих сховищ з офіційними версіями проектів. Це був значний прогрес, оскільки кілька користувачів могли одночасно працювати з кодом, роблячи commit-и в той самий центральний репозиторій. Однак для commit-ів був потрібний доступ до мережі.

Третє покоління складається з розподілених VCS, де копії репозиторію вважаються рівними, немає центрального репозиторію. Це відкриває шлях для commit-ів, гілок та злиття, які створюються локально без доступу до мережі та переміщуються в інші репозиторії за необхідності.

## **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.

Реалізації Markdown різними мовами програмування включені (або наявні як плаґіни) у численні системи керування вмістом.

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

Хоча Markdown є полегшеною мовою розмітки яку легко читати та редагувати звичайними текстовими редакторами, існують спеціально розроблені редактори, які дозволяють попередньо переглядати зі стилями.

Є безліч таких редакторів, які наявні для всіх основних платформ. Існує плаґін підсвітки синтаксису для Markdown, вбудований у gedit та Vim. Текстовий редактор Atom має підсвітку синтаксису Markdown, є також можливість легко встановлювати пакунки для розширення функціоналу мови: наприклад, пакунок markdown-preview-plus дозволяє відтворювати також синтаксис LaTeX, а markdown-writer допомагає з форматуванням списків, таблиць тощо.

 Markdown із LaTeX також використовуються в IPython-записниках в Jupyter та Google Colaboratory для форматування тексту. Ці вебоболонки для IPython можуть використовуватись як текстові редактори, оскільки там є можливість створювати блоки тексту окремо від блоків коду та можливість завантажувати сторінку, наприклад, у PDF.

Реалізації Markdown існують для багатьох різних фреймворків, платформ та мов.

• Система документації джерельних текстів Doxygen підтримує Markdown з додатковими можливостями.

• RStudio, IDE для R підтримує C++ обгортку функцій для реалізації markdown Sundown.

• IntelliJ IDEA, IDE для Java, містить плаґін для підтримки Markdown.

• Існують формат і програма MultiMarkdown з ширшим синтаксисом та можливостями експорту, ніж в традиційній Markdown.

Існує багато іншого відкритого програмного забезпечення реалізації Markdown, доступного в мережі.

**Markdown Extra** - це легка мова розмітки на основі Markdown реалізована в PHP (початково), Python та Ruby. Вона містить функції недоступні в звичайному синтаксисі Markdown. Markdown Extra підтримується системами керування вмістом, наприклад Drupal, TYPO3 та MediaWiki.

Додає наступні функції до Markdown:

• Markdown всередині блоків HTML

• Елементи з атрибутами id/class

• "обгороджені" блоки коду (англ. Fenced code blocks)

• Таблиці

• Списки означень (dl)

• Зноски

• Абревіатури

Заголовки

HTML-заголовки створюються розміщенням кількох символів решітки перед текстом заголовка відповідно до бажаного рівня (HTML підтримує 6 рівнів заголовіків), наприклад:

Изображение выглядит как текст, Шрифт, снимок экрана, белый

Автоматически созданное описание

Перші два рівні заголовків також мають альтернативний синтаксис:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Параграфи

Параграф є одним чи декількома послідовними рядками тексту, які розділяються одним чи декількома порожніми рядками. Звичайні параграфи не повинні мати відступи або табуляцію:

Изображение выглядит как текст, Шрифт, белый, снимок экрана

Автоматически созданное описание

Список

Изображение выглядит как текст, снимок экрана, Шрифт, белый

Автоматически созданное описание

Виділений текст

Изображение выглядит как текст, Шрифт, снимок экрана, белый

Автоматически созданное описание

Код

Для включення коду (відформатованого в моноширинний шрифт), ви можете або оточити вбудований код зворотніми лапками (`), наприклад, `деякий код`, або відділити декілька рядків коду щонайменше чотирма пробілами, як тут:

Изображение выглядит как текст, Шрифт, снимок экрана, белый

Автоматически созданное описание

Останній варіант за допомогою пробілів дозволяє зберігати і показувати синтаксис Markdown.

## **4.7 Розроблення ігрової програми**

### 4.7.1 Код програми

#include <iostream>  
using namespace std;  
void winner(int player) {  
 if (player % 2 == 0) {  
 cout << "Player 2 wins!" << endl; // Якщо player % 2 == 0, то виграє гравець 2  
 } else {  
 cout << "Player 1 wins!" << endl; // Якщо player % 2 == 1, то виграє гравець 1  
 }  
}  
int main() {  
 char mass[3][3] = { {' ', ' ', ' '}, {' ', ' ', ' '}, {' ', ' ', ' '} };  
 int player = 1;  
  
 while(true){  
 int a, b;  
 cout << "Enter the field (row and column): ";  
 cin >> a >> b;  
  
 // Перевірка границь масива  
 if (a < 0 || a >= 3 || b < 0 || b >= 3) {  
 cout << "Invalid input. Please enter values between 0 and 2." << endl;  
 continue;  
 }  
  
 // Перевірка, чи порле пусте  
 if (mass[a][b] != ' ') {  
 cout << "Field already occupied. Please choose another one." << endl;  
 continue;  
 }  
  
 // Встановлення мітки гравця  
 if (player % 2 == 0) {  
 mass[a][b] = 'X';  
 } else {  
 mass[a][b] = 'O';  
 }  
  
 // Відображення поля  
 for(int i = 0; i < 3; i++) {  
 cout << '|';  
 for(int j = 0; j < 3; j++) {  
 cout << mass[i][j] << '|';  
 }

cout << endl;  
 }  
  
 // Перевірка, чи немає перемоги  
 for (int i = 0; i < 3; i++) {  
 if ((mass[i][0] == mass[i][1] && mass[i][1] == mass[i][2] && mass[i][0] != ' ') ||  
 (mass[0][i] == mass[1][i] && mass[1][i] == mass[2][i] &&

mass[0][i] != ' ')) {  
 winner(player);  
 return 0;  
 }  
 }  
 if ((mass[0][0] == mass[1][1] && mass[1][1] == mass[2][2] && mass[0][0] != ' ') ||  
 (mass[0][2] == mass[1][1] && mass[1][1] == mass[2][0] && mass[0][2] != ' ')) {  
 winner(player);  
 return 0;  
 }  
  
 // Перевірка, чи немає нічиєї  
 bool draw = true;  
 for (int i = 0; i < 3; i++) {  
 for (int j = 0; j < 3; j++) {  
 if (mass[i][j] == ' ') {  
 draw = false;  
 break;  
 }  
 }  
 }  
 if (draw) {  
 cout << "Draw!" << endl;  
 return 0;  
 }  
  
 // Перехід коду до наступного ігрока  
 player++;  
 }  
  
 return 0;  
}

### 4.7.2 Скріншоти виконання програми

Початок програми:

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Середина програми:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Кінець програми:

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

# 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**

Документ HTML оброблюється браузером та відтворюється на екрані у звичному для людини вигляді.

HTML надає засоби для:  
• створення структурованого документу шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше; •  створення зв’язків між веб-сторінками через гіперпосилання; •  створення інтерактивних форм; •  включення зображень, звуку, відео та ін.

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

•  елементів;

•  базових типів даних (визначають тип вмісту атрибутів: символьні дані,

числа,..);

•  символьних мнемонік;

•  декларації типу документа (оголошує версію HTML).

Мнемоніки – спеціальні позначення (псевдоніми), які використовуються для вставки деяких символів.

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**

# Абзаци

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 може містити всередині себе теги <li>. Якщо елементи неупорядкованого списку за замовчуванням позначаються маркерами, то елементи впорядкованого списку - нумеруються. Нумеровані або впорядковані списки призначені для елементів, які слідують в певному порядку, наприклад покрокові інструкції. Вони функціонують так само, як і описані раніше марковані списки, за винятком того, що визначені за допомогою тега <ol>, та замість маркерів браузер автоматично вказує номера перед елементами списку, тому вам не потрібно проставляти їх у вихідному документі. Це дозволяє легко міняти місцями елементи списку без повторної нумерації.

Для зміни маркерів і номерів списків варто використовувати властивість list-style-type таблиці стилів CSS. Наприклад для маркованих списків Ви можете змінити форму маркера з кола на квадрат або незаштриховане коло, на власне зображення, або видалити маркер повністю. Для нумерованих списків - замінити традиційні арабські цифри римськими (I., II., IІІ. або i., ii., iii.), літерами (А., В., С., або а., b., с.) та іншими типами нумерації.

Важливість. Теги <strong> и <b>

Тег <strong> визначає важливість зазначеного тексту. Тег <b> (bold) призначений для виділення тексту без додання йому особливої ​​важливості.

Найкраще відмінності цих тегів будуть помітні людям, які використовують спеціальні налаштування ОС, зокрема, сліпим і слабозорим. Коли вони включать функцію читання тексту, то «читалка» буде інтонацією виділяти слова з тегом <strong>. Те ж саме стосується і тегів <em> (emphasis) та <i>. Тег <em> «читалка» виділятиме інтонацією. Візуально обидва тега однакові – вони виділяють текст жирним шрифтом. Відзначимо, що новий сенс тегу <b> надали в HTML5. Раніше це був тег, який просто робить текст напівжирним. Тобто він був призначений лише для візуального форматування.

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

Іноді виникає необхідність вставити в текст перенесення рядка, не створюючи при цьому абзац. Наприклад, при розмітці віршів або текстів пісень. Для цього в HTML передбачений одиночний тег <br>(break). Іноді цей тег використовується для розбиття тексту на «як би абзаци», що є поганим підходом. Використовуйте для розмітки абзаців тег <p>.

Одиночний тег <hr> (horisontal rule) використовується для того, щоб створити горизонтальну лінію-розділювач.

# Цитати

У HTML існує декілька тегів для позначення цитат:

• <blockquote> призначений для виділення довгих цитат, які можуть складатися з кількох абзаців. Тег виділяє цитату як окремий блок тексту з відступами.

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

• <cite> використовується для того, щоб виділити джерело цитати – назву твору, але не автора цитати.

Посилання

Посилання, зазвичай, виглядає як підкреслена частина тексту, клацаючи на якій ви переходите на іншу сторінку, відкриваєте зображення або починаєте завантажувати файл. Якщо уявити, що Інтернет це величезна мережа з безліччю вузлів, то посилання будуть нитками, що з'єднують всі вузли цієї мережі. Посилання створюються за допомогою дуже простого і короткого тегу <a> Наприклад:

<a href="Саме посилання"> Назва</a>

Адреси бувають різні: абсолютні, відносні, ті що вказують на сторінку, на файл, зображення, ті що містять якір і так далі. А значить і посилання може вказувати на будь-який об'єкт в Інтернеті – на будь-яку веб-сторінку, на будь-який файл. Якщо клацнути по посиланню, що веде на файл, то браузер запропонує його завантажити. Однак, якщо браузер вміє обробляти файли цього типу, то вміст файлу відкриється прямо в браузері. Найчастіше так відбувається із зображеннями. Останнім часом

браузери навчилися відкривати .pdf файли і багато інших.

Робота з картинками.

Щоб додати на сторінку зображення, потрібно використовувати одиночний тег <img> (image) з атрибутом src (source), в якому вказана адреса картинки. наприклад:

<img src = "logo.png">

До речі, найпоширенішими форматами зображень у мережі є JPEG і PNG.

Щоб управляти шириною або висотою зображення, потрібно

використовувати атрибути width і height. Як ось, наприклад:

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

В прикладі зображенню задана ширина 100px (пікселів).

Якщо у користувача відключені зображення чи їх неможливо завантажити, то в браузері відображається альтернативний текст. Наприклад, якщо меню зроблено за допомогою зображень, то альтернативний текст допоможе зрозуміти, куди веде кожен пункт. Загалом, задавати альтернативний текст хороший стиль програмування, хоча і не є обов`язковим. Альтернативний текст зображення задається за допомогою атрибуту alt (alternative)

Посилання можна робити не лише за допомогою тексту, але і за допомогою зображень. Для цього потрібно обернути тег <img> в тег <a>.

Часто посилання-зображення використовуються в галереях, коли 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.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

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>.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

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.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

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:

Изображение выглядит как текст, снимок экрана, веб-страница, программное обеспечение

Автоматически созданное описание

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

#FC9804;text-align:center;}

Відображення у браузері:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

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**

Ієрархічне дерево

HTML-документ являє собою ієрархічне дерево. Це означає, що у кожного елемента (крім кореневого) є тільки один батько, тобто елемент, усередині якого він розташовується. У кореневого розділу батько відсутній. Як приклад, найпростіша сторінка:

<html>

<head> </ head>

<body>

<p> Текст документа </ p>

         <p class = "text"> Виділена <span> рядок </ span> </ p>

</ body>

</ html>

Для цієї сторінки можна намалювати таке ієрархічне дерево:

Изображение выглядит как диаграмма, текст, линия, Шрифт

Автоматически созданное описание

Воно схематично відображає структуру вкладеності елементів. В даному прикладі видно, що в елемента span батьком є p.text, а у p.text батько - Body. Ієрархічна структура документа визначає основи концепції наслідування.

Наслідування

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

Наслідування властивості

До спадкоємною властивостей відносяться в першу чергу властивості, що визначають параметри відображення тексту:

font-size, font-family, font-style, font-weight, color, text-align, text-transform, text-indent, line-height, letter-spacing, word-spacing, white-space, direction і т. д.

Також до спадкоємною властивостей відносяться list-style, cursor, visibility, border-collapse і деякі інші. Але вони використовуються значно рідше.

Ці властивості можна і потрібно задавати через предків, слідуючи семантиці документа. Наприклад, параметри тексту часто не змінюються в межах окремих блоків сторінки: меню, основного змісту, інформаційних панелей. Тому загальні параметри тексту (колір, розмір, гарнітура) зазвичай вказують в стилях самих блоків.

Каскадність

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

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

CSS-правила існують як мінімум в трьох різних місцях:

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

2.                в атрибуті style;

3.                в стандартних стилях відображення, вбудованих в браузер.

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

1.                важливість;

2.                специфічність;

3.                порядок вихідного коду.

**5.5 Практична частина**

### 5.5.1 Постановки задачі

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

**НTML**

<!DOCTYPE HTML> #Semenova Valeriia

<html lang="en">.

<head>

<meta charset="UTF-8">

<meta name="viewpirt" content="width=device-width,minitial-scale=1.0">

<title>Valeriia Semenova</title> #задаємо назву веб-сайту

<link rel="stylesheet" href="практика\_сайт.css"> #посилання на файл css

</head>

<body> #основна частина «тіло» коду

<div class="container"> #застосовуємо стиль

<header class="imagine"> #відділяємо верхню частину сторінки + стиль

<div>

<img src="photo2.jpg" alt="" id="centered"> #додаємо фото

</div>

<div class="center"> #додаємо текст і застосовуємо стиль

<h1>Семенова Валерія</h1>

<h4>студентка 2-го курсу</h3>

</div>

</header>

<h2 class="mood">Загальна інформація</h2> #додаємо текст + стиль

<hr> #добавляємо горизонтальну лінію

<p>Стресостійка фахівчиня, з вмінням гарно комунікувати з людьми, критичним мисленням, творчими здібностями та

клієнтоорієнтованістю. Загалом маю 2 роки досвіду роботи адміністратором в закладі та декілька розроблених

самостійно проєктів, таких як розробка Telegram-боту, наукова робота та проєкт у сфері Data Analytics.</p> #додаємо параграф з текстом

<p>Маю досвід роботи у стресовий ситуаціях, завжди вкладаюсь у час поставлений для виконання завдання, відкрита до нових

складних завдань, які вимагають швидкого навчання та пошуку оптимального рішення задачі. </p>

<h2 class="mood">Освіта</h2>

<hr>

<ul> #ненумерований список

<li> <b>2012-2022 Базова середня освіта з відзнакою</b>

<br> Боярський академічний ліцей "Гармонія"

</li>

<li><b>2022-дотепер Молодший бакалавр</b>

<br> Опико-механічний фаховий коледж імені Тараса Шевченка <br> Спеціальність:

121 Інженерія програмного забезпечення</li>

</ul>

<h2 class="mood">Досвід роботи</h2>

<hr>

<ul>

<li><b>Адміністратор (2021-2023)</b>

<br> Місце роботи:вейк-парк "Піратська бухта" м.Київ

</li>

<li><b>Data Analytics (березень-квітень 2023)</b>

<br> Розробка аналітичного дашборду, а саме проведення аналітики

кінофільмів за жанрами, країнами, в яких ці фільми були випущені, та мови, якими вони були озвучені.

</li>

<li><b>Python-розробник (січень-квітень 2024)</b>

<br> Розробка Telegram-боту. Написаний мовою Python, цей бот має всі

навички Штучного Інтелекту та гарний інтерфейс, що дозволяє зручно ним користуватись.

</li>

</ul> #ненумерований список

<footer> #відділяємо нижню частину веб-сторінки

<ul>

<li><div>

<a href="mailto:lerasem47@gmail.com" target="\_blank">Email <img src="post.png" class="im"></a>

</div></li>

<div class=""><li>

<a href="https://www.instagram.com/vvaalllerriaa/" target="\_blank">Instagram <img src="instagram.png" class="im"></a>

</li></div>

</ul>

</footer>

</div>

</body>

**CSS**

ul {

list-style-type: none;

}

a {

margin: 10px;

padding: 10px;

text-decoration: none;

text-align: center;

width: 90%;

height: 25px;

display: block;

border: 4px solid rgb(255, 215, 165);

text-decoration: none;

border-radius: 20px;

}

a:hover,

a:visited {

background-color: rgba(217, 184, 143, 0.881);

color: black;

}

.container {

background-color: rgba(218, 203, 186, 0.387);

width: 95vw;

margin: 0 auto;

}

.mood {

color: rgb(93, 82, 68);

}

.imagine {

text-align: center;

}

#centered {

border-radius: 100%;

height: 150px;

width: 150px;

}

.center {

color: rgb(93, 82, 68);

text-align: center;

}

.im {

height: 20px;

}

**Изображение выглядит как текст, снимок экрана, Веб-сайт, веб-страница

Автоматически созданное описание**

**Изображение выглядит как текст, веб-страница, Веб-сайт, программное обеспечение

Автоматически созданное описание**

# 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