**ЛАБОРАТОРНА 8.2**

**ЕКЗАМЕН 1**

Основна функція **main** виконує наступні дії:

Оголошується **масив** символів **s** розміром **101**, який буде використовуватися для зберігання введеного рядка.

Користувачу виводиться повідомлення "**Enter** **string**:", і він може ввести рядок за допомогою функції **cin**.**getline(s,** **101**). Зауважте, що **cin**.**getline** використовується для зчитування рядка, щоб уникнути проблеми з пропуском пробілів.

Потім викликається функція **FirstLetter(s)**, яка перевіряє, чи починається хоча б одне слово в рядку **s** з літери **'a'**.

Залежно від результату перевірки, виводиться "**Yes**" або "**No**".

Функція **FirstLetter** приймає рядок **str** і перевіряє кожне слово в рядку на першу літеру **'a'**. Вона використовує функцію **strtok\_s** для розбиття рядка на окремі слова, використовуючи пробіл як роздільник.

Основна частина функції **FirstLetter** містить наступний цикл:

Змінна **str2** отримує перше слово з рядка str за допомогою **strtok\_s(str, " ", &nextToken).**

У циклі перевіряється, чи перша літера **str2** дорівнює **'a'**. Якщо так, то функція повертає **true**, означаючи, що знайдено слово з літерою **'a'**.

Викликається **strtok\_s(NULL, " ", &nextToken)**, щоб отримати наступне слово з рядка **str**.

Цей процес повторюється до тих пір, поки **strtok\_s** не поверне **NULL**, що означає, що всі слова в рядку були перевірені.

Якщо жодне слово з літерою **'a'** не знайдено, функція повертає **false**.

Загальною метою цього коду є перевірка, чи існує хоча б одне слово в рядку, яке починається з літери **'a'**.

**ВІДПОВІДІ НА ПИТАННЯ 8.2**

**ЕКЗАМЕН 1**

**1) Загальний синтаксис оголошення літерних рядків.**

Літерні рядки в C++ оголошуються за допомогою масивів символів, заключених у лапки. Наприклад, "Hello, World!" є оголошенням літерного рядка. char str1[] = "apple";

**2) Доступ до елементів літерних рядків.**

Елементи літерного рядка можна отримати за допомогою індексації. Наприклад, str[0] дає доступ до першого символу рядка str.

**3) Дії з літерними рядками.**

Літерні рядки можна об'єднувати, порівнювати, копіювати, вирізати підрядки, перетворювати регістр символів і т.д. Вони підтримують різні операції, такі як конкатенація (+), порівняння (==, !=), копіювання (strcpy), довжина (strlen) та інші.

**4) Функції роботи з символами та літерними рядками.**

Серед функцій, що працюють з символами та літерними рядками в C++, є strcpy (копіює один рядок в інший), strcat (додає один рядок до іншого), strlen (повертає довжину рядка), strcmp (порівнює два рядки), strchr (шукає перший входження символу у рядку) та інші.

**5) Загальні схеми циклів сканування літерних рядків для пошуку заданих символів.**

while (str2 != NULL)

{

if (str2[0] == 'a')

return true;

str2 = strtok\_s(NULL, " ", &nextToken);

}

**6) Загальні схеми рекурсивних підпрограм для опрацювання літерних рядків.**

Рекурсивні підпрограми для опрацювання літерних рядків можуть включати базовий випадок і рекурсивний виклик.

char\* nextToken;

char\* str2 = strtok\_s(str, " ", &nextToken);

**7) Внутрішня реалізація літерних рядків.**

Літерні рядки в C++ зберігаються як масиви символів, де кінцевий символ (NULL) \0 вказує на кінець рядка. Такий підхід відомий як нуль-термінований рядок.

**8) Передавання та опрацювання літерних рядків у функціях.**

Літерні рядки можуть передаватися у функції як аргументи, які можуть бути оголошені як const char\*. Функції можуть працювати з рядками, використовуючи функції роботи з символами та літерними рядками, які надає мова C++, або власні функції, розроблені користувачем.

**ЛАБОРАТОРНА 9.3**

**ЕКЗАМЕН 2**

Загальний опис функцій у даному коді:

**1. void Create(route\* n, const int N)**

Ця функція приймає вказівник на масив структур route і його розмір N. Вона виконує цикл for, пропонуючи користувачеві ввести дані про кожен маршрут: номер маршруту, початкову та кінцеву точки. Введені дані зберігаються відповідним членам структури route.

**2. void Print(route\* n, const int N)**

Ця функція приймає вказівник на масив структур route і його розмір N. Вона виводить на екран заголовок таблиці маршрутів, а потім у циклі виводить дані про кожен маршрут у вигляді таблиці з номерами, початковими та кінцевими точками та номерами маршрутів.

**3. void Menu(route\* n, const int N)**

Ця функція приймає вказівник на масив структур route і його розмір N. Вона відображає користувачеві меню з наступними пунктами: сортування маршрутів, пошук маршруту та вихід. Залежно від вибору користувача виконується відповідна операція.

**4. void Sort(route\* n, const int N)**

Ця функція приймає вказівник на масив структур route і його розмір N. Вона сортує масив маршрутів за зростанням їх номерів, використовуючи алгоритм сортування бульбашкою.

Умова **if (n[i1].nomer > n[i1 + 1].nomer)** перевіряє, чи поточний елемент **n[i1]** більший за наступний елемент **n[i1 + 1]** за полем **nomer**. Якщо умова виконується, тобто елементи не впорядковані, вони обмінюються місцями за допомогою змінної **tmp**.

Після завершення сортування масив буде впорядкованим за зростанням значень поля **nomer**

**5. int Search(route\* n, const int N, const int nomer)**

Ця функція приймає вказівник на масив структур route, його розмір N та номер маршруту nomer, який потрібно знайти. Вона здійснює пошук маршруту з вказаним номером у масиві n. Якщо маршрут знайдено, функція повертає його індекс в масиві n, в іншому випадку повертається -1.

**6. int main()**

Це головна функція програми. У ній виконуються наступні дії:

- Користувачеві пропонується ввести кількість маршрутів.

- Створюється динамічний масив структур route з введеним розміром.

- Викликаються функції Create() та Print() для створення та виведення маршрутів.

- Викликається функція Menu(), яка дозволяє користувачу вибирати певні операції з маршрутами до тих пір, поки він не вибере вихід з програми.

- Після виходу з циклу програми, пам'ять, виділена для масиву маршрутів, звільняється.

**ВІДПОВІДІ НА ПИТАННЯ 9.3**

**ЕКЗАМЕН 2**

**1) Загальний синтаксис оголошення масивів та переліків, структур і об’єднань.**

- Масиви оголошуються за допомогою квадратних дужок та вказанням типу елементів та розміру масиву. Наприклад, int numbers[5]; оголошує масив чисел типу int з розміром 5.

- Переліки оголошуються за допомогою ключового слова enum та перелічення значень. Наприклад, enum Color { RED, GREEN, BLUE }; оголошує перелік Color з трьома значеннями: RED, GREEN та BLUE.

- Структури оголошуються за допомогою ключового слова struct та визначаються полями, які можуть мати різні типи даних. Наприклад, struct Person { string name; int age; }; оголошує структуру Person з полями name типу string та age типу int.

- Об'єднання оголошуються за допомогою ключового слова union та мають різні поля, але використовують спільну область пам'яті. Наприклад, union Data { int number; float decimal; }; оголошує об'єднання Data з полями number типу int та decimal типу float.

**2) Доступ до елементів масивів та переліків, структур і об’єднань.**

- Елементи масивів та переліків доступні за допомогою їх індексів. Наприклад, numbers[0] дозволяє отримати перший елемент масиву numbers.

- До полів структур можна отримати доступ за допомогою оператора крапка (.). Наприклад, person.name дозволяє отримати значення поля name структури person.

- Для отримання значень полів об'єднань використовується той самий принцип, але об'єднання використовує спільну область пам'яті для всіх полів. Тому доступ до полів об'єднання дає значення останнього записаного поля.

**3) Дії з масивами та переліками, структурами і об’єднаннями.**

- З масивами можна виконувати різні операції, такі як читання та запис елементів, сортування, пошук тощо. Для цього використовуються цикли, функції та відповідні алгоритми.

- Переліки дозволяють працювати з набором значень і використовувати їх для визначення стану або варіантів в програмі. Можна порівнювати, перебирати та виконувати операції зі значеннями переліку.

- Структури дозволяють групувати різні типи даних в один складний тип. За допомогою структур можна створювати складні об'єкти з багатьма властивостями та виконувати операції з ними.

- Об'єднання дають можливість використовувати спільну область пам'яті для різних типів даних. Це може бути корисно в деяких ситуаціях, коли потрібно зберігати різні дані в одній пам'ятовій області.

**4) Ініціалізація масивів та переліків, структур і об’єднань.**

- Масиви та переліки можна ініціалізувати під час оголошення або пізніше за допомогою присвоєння значень окремим елементам.

- Структури можна ініціалізувати під час оголошення, вказавши значення полів в фігурних дужках. Також можна ініціалізувати структуру за допомогою конструктора.

- Об'єднання можуть мати лише одне активне поле, тому можна ініціалізувати об'єднання шляхом присвоєння значення одному з його полів.

**5) Внутрішня реалізація масивів та переліків, структур і об’єднань.**

Внутрішня реалізація масивів та переліків, структур і об'єднань у мові програмування С++ залежить від компілятора та реалізації самої мови. Зазвичай масиви та переліки реалізовані як блоки пам'яті з послідовним розташуванням елементів. Структури в С++ можуть мати внутрішню реалізацію, яка залежить від послідовності полів в пам'яті. Об'єднання використовують спільну область пам'яті для всіх полів.

**6) Передавання та опрацювання масивів та структур у функціях.**

- Масиви можна передавати у функції як покажчики на перший елемент або як посилання. Функції можуть модифікувати передані масиви.

- Структури можна передавати у функції як значення, як посилання або як константне посилання. Якщо структура передається як значення, то вона копіюється у локальну змінну функції.

- Функції можуть повертати масиви або структури як результат.

**7) Загальний синтаксис оголошення та формування динамічних масивів.**

- Динамічні масиви оголошуються за допомогою оператора new і вказанням типу елементів та розміру. Наприклад, int\* numbers = new int[5]; оголошує динамічний масив чисел типу int з розміром 5.

- Після використання динамічного масиву його слід звільнити з пам'яті за допомогою оператора delete[]. Наприклад, delete[] numbers; звільняє пам'ять, виділену для динамічного масиву numbers.

**8) Загальна схема фізичного впорядкування масиву.**

Загальна схема фізичного впорядкування масиву в мові програмування С++ залежить від реалізації компілятора та алгоритмів оптимізації. Зазвичай масиви зберігаються в послідовних блоках пам'яті, де кожен елемент має фіксований зсув відносно початку масиву.

**9) Загальна схема індексного впорядкування масиву.**

Загальна схема індексного впорядкування масиву в мові програмування С++ полягає у використанні індексів для доступу до елементів масиву. Кожен елемент масиву має унікальний індекс, який починається з 0 для першого елемента.

**10) Загальна схема послідовного пошуку.**

Загальна схема бінарного пошуку в масиві в мові програмування С++ передбачає використання відсортованого масиву та пошук середнього елемента. Порівнюючи шуканий елемент з середнім, виконується перехід до лівої або правої половини масиву до знаходження потрібного елемента або поки не буде виявлено його відсутність.

**11) Загальна схема бінарного пошуку.**

Вхідні дані: відсортований масив (або список) **arr** та шуканий елемент **target**.

**ЛАБОРАТОРНА 10.1**

**ЕКЗАМЕН 3**

Даний код є програмою, яка виконує пошук певних послідовностей символів у рядку, зчитаному з файлу "test.txt".

**Основна функція main має наступну структуру:**

1. Директива #define \_CRT\_SECURE\_NO\_WARNINGS вказує компілятору ігнорувати попередження, пов'язані з використанням функцій стандартної бібліотеки C.

2. Включаються необхідні заголовочні файли <iostream> і <fstream>, які надають функціонал для роботи зі введенням/виведенням та файлами в C++.

3. Використовується простір імен std.

4. Оголошується прототип функції bool search(const char\* s).

5. Функція main починає виконання.

6. Оголошується масив str розміром 101, який буде використовуватись для збереження рядка з файлу.

7. Відкривається файл "test.txt" за допомогою об'єкту ifstream з іменем f.

8. Зчитується перше слово з файлу за допомогою оператора >> та зберігається у масиві str.

9. Викликається функція search, передаючи їй масив str як аргумент.

10. Якщо результат функції search дорівнює true, виводиться "Yes".

11. Якщо результат функції search дорівнює false, виводиться "No".

12. Файл f закривається за допомогою методу close().

13. Чекається введення символу з консолі.

14. Функція main повертає 0, що означає успішне завершення програми.

**Функція bool search(const char\* s) виконує пошук певних послідовностей символів у рядку. Вона має наступну структуру:**

1. Приймає рядок s як вхідний параметр.

2. Виконує цикл for, який проходить через рядок s за допомогою індексу j.

3. У тілі циклу перевіряється, чи поточний символ s[j] та наступний символ s[j + 1] утворюють одну з трьох послідовностей: "aa", "bb" або "cc".

4. Якщо умова виконується, функція повертає true, позначаючи, що знайдено вказану послідовність.

5. Якщо цикл завершується без знаходження послідовності, функція повертає false.

Отже, ця програма зчитує рядок з файлу "test.txt" та перевіряє, чи міститься у ньому послідовність "aa", "bb" або "cc". Залежно від результату пошуку, виводиться "Yes" або "No".

**ВІДПОВІДІ НА ПИТАННЯ 10.1**

**ЕКЗАМЕН 3**

**1) Загальний синтаксис оголошення текстових файлів.**

Синтаксис оголошення текстового файлу в C++ включає використання об'єкту класу std::ifstream або std::ofstream з відповідним ім'ям для читання або запису в файл.

**2) Структура текстового файлу.**

Структура текстового файлу полягає у послідовному розташуванні символів, які можуть бути прочитані або записані у файлі. Файл складається зі слів, символів, рядків або будь-якої іншої структури, відповідно до його вмісту.

**3) Доступ до логічних рядків в текстовому файлі.**

Доступ до логічних рядків (ліній) у текстовому файлі відбувається шляхом читання файлу по одному рядку за раз за допомогою функції std::getline або оператора >>.

**4) Дії з текстовими файлами.**

Дії з текстовими файлами включають відкриття, закриття, читання та запис до файлу. Це включає в себе використання об'єктів std::ifstream для читання та std::ofstream для запису, а також використання методів таких як open, close, >>, << для взаємодії з файлом.

**5) Внутрішня реалізація текстових файлів.**

Внутрішня реалізація текстових файлів залежить від операційної системи та бібліотеки введення/виведення, але загалом текстові файли можуть бути збережені як послідовність символів, з врахуванням відповідного кодування символів.

**6) Передавання та опрацювання текстових файлів у підпрограмах.**

Передавання та опрацювання текстових файлів у підпрограмах включає передачу файлового об'єкту як аргумента підпрограми та використання методів читання/запису для взаємодії з файлом.

**7) Загальна схема послідовного пошуку у текстовому файлі.**

Загальна схема послідовного пошуку у текстовому файлі включає відкриття файлу, читання файлу по одному рядку за раз та порівняння рядка зі шуканою послідовністю або умовою пошуку. Процес повторюється.

**ЛАБОРАТОРНА 11.1**

**ЕКЗАМЕН 4**

Даний код є програмою на мові C++, яка працює з **бінарними файлами**. Давайте розберемо функції по черзі:

**1. `void CreateBIN(char\* fname)`:** Ця функція відповідає за створення бінарного файлу та запису чисел у нього. Вона отримує ім'я файлу (`fname`) як вхідний параметр. Функція відкриває файл за допомогою об'єкту `ofstream` з режимом `ios::binary` (бінарний режим запису). Якщо відкриття файлу не вдалося, виводиться повідомлення про помилку. У циклі користувачу пропонується ввести число (`x`), яке потім записується у бінарний файл за допомогою функції `write`. Після запису числа користувачу пропонується продовжити введення чисел. Цикл триває, доки користувач вводить 'y' або 'Y'. Коли цикл закінчується, файл закривається.

**2. `void PrintBIN(char\* filename)`:** Ця функція відповідає за читання та виведення чисел з бінарного файлу. Вона отримує ім'я файлу (`filename`) як вхідний параметр. Функція відкриває файл за допомогою об'єкту `ifstream` з режимом `ios::binary` (бінарний режим читання). Якщо відкриття файлу не вдалося, виводиться повідомлення про помилку. У циклі функція використовує функцію `read` для зчитування чисел з файлу у змінну `x`, а потім виводить значення `x`. Цей цикл продовжується, поки зчитування чисел з файлу вдається. Після завершення циклу файл закривається.

**3. `void ProcessBIN(char\* fname, char\* outnameodd, char\* outnameeven)`:** Ця функція відповідає за обробку бінарного файлу та розділення чисел на файли з парними та непарними числами. Вона отримує ім'я вхідного файлу (`fname`), а також імена вихідних файлів для парних (`outnameeven`) та непарних (`outnameodd`) чисел. Функція відкриває вхідний файл для читання та два вихідних файли для запису за допомогою об'єктів `ifstream` та `ofstream` з режимом `ios::binary`. У циклі функція зчитує числа з вхідного файлу за допомогою `read`, а потім перевіряє, чи є число парним чи непарним. Залежно від результату перевірки, число записується у відповідний вихідний файл за допомогою функції `write`.

**4. `int main()`:** Це головна функція програми. Вона викликається при запуску програми. У цій функції користувачу пропонується ввести ім'я вхідного файлу за допомогою `cin`. Потім викликається функція `CreateBIN` для створення та заповнення бінарного файлу. Після цього викликається функція `PrintBIN` для виведення вмісту вхідного файлу. Користувачу також пропонується ввести імена вихідних файлів для парних та непарних чисел. Потім викликається функція `ProcessBIN` для обробки вхідного файлу та розділення чисел на відповідні вихідні файли. Нарешті, викликаються функції `PrintBIN` для виведення вмісту вихідних файлів з парними та непарними числами.

**ВІДПОВІДІ НА ПИТАННЯ 11.1**

**ЕКЗАМЕН 4**

**1) Загальний синтаксис оголошення бінарних файлів.**

- Для читання: `ifstream ім'я\_об'єкту(ім'я\_файлу, режими)`

- Для запису: `ofstream ім'я\_об'єкту(ім'я\_файлу, ios::in out app binary)`

**2) Доступ до компонентів бінарних файлів.**

Компоненти бінарних файлів у С++ можна доступатися за допомогою методів та операцій, таких як read, write, seekg, seekp, tellg, tellp, eof, fail, тощо. Зокрема, read та write використовуються для читання з файлу та запису в файл в бінарному вигляді.

**3) Дії з бінарними файлами.**

- Відкриття файлу (ifstream для читання або ofstream для запису).

- Зчитування та запис даних у бінарний файл за допомогою методів read та write.

- Переміщення покажчика у файлі за допомогою методів seekg та seekp.

- Отримання поточної позиції покажчика у файлі за допомогою методів tellg та tellp.

- Перевірка на кінець файлу (eof) та наявність помилок (fail).

- Закриття файлу.

**4) Внутрішня реалізація бінарних файлів.**

Внутрішня реалізація бінарних файлів залежить від операційної системи та бібліотеки вводу/виводу, яку використовує мова програмування. Наприклад, у С++ використовується стандартна бібліотека вводу/виводу iostream, яка надає інтерфейс для роботи з бінарними файлами на різних платформах.

**5) Передавання та опрацювання бінарних файлів у підпрограмах.**

Для передавання та опрацювання бінарних файлів у підпрограмах в С++ можна передавати об'єкти класу ifstream та ofstream як аргументи функцій. Операції зчитування та запису виконуються в середині підпрограми з використанням методів цих об'єктів.

**6) Загальна схема послідовного пошуку у бінарному файлі.**

Загальна схема послідовного пошуку у бінарному файлі включає такі етапи:

- Відкриття бінарного файлу для читання (ifstream).

- Читання записів з файлу у циклі до досягнення кінця файлу (eof).

- Порівняння зчитаного запису з шуканим значенням.

- Якщо знайдено шукане значення, виконується відповідна обробка.

- Після закінчення пошуку файл закривається.

**ЛАБОРАТОРНА 12.6**

**ЕКЗАМЕН 5**

**1. Функція enqueue:**

Ця функція додає новий елемент до черги. Вона приймає посилання на початок черги (first), посилання на кінець черги (last) і значення для додавання (value). Алгоритм дії функції наступний:

- Створюється новий об'єкт типу Elem з інформацією value і вказівником link на NULL.

- Якщо черга не є порожньою (тобто last != NULL), то вказівник link останнього елемента черги оновлюється так, що він вказує на новий елемент.

- Вказівник last оновлюється, щоб вказувати на новий елемент.

- Якщо черга була порожня (тобто first == NULL), то вказівник first оновлюється так, щоб вказувати на новий елемент.

**2. Функція dequeue:**

Ця функція видаляє перший елемент з черги і повертає його значення. Вона також приймає посилання на початок черги (first) і посилання на кінець черги (last). Алгоритм дії функції наступний:

- Створюється тимчасовий вказівник tmp, який вказує на другий елемент черги (first->link).

- Значення першого елемента зберігається в змінній value.

- Перший елемент черги видаляється з пам'яті за допомогою оператора delete.

- Вказівник first оновлюється, щоб вказувати на другий елемент.

- Якщо черга була порожня (тобто first == NULL), то вказівник last також оновлюється на NULL.

- Значення value повертається як результат функції.

**3. Функція read:**

Ця функція відповідає за читання значень з файлу і додавання їх до черги. Вона приймає посилання на початок черги (first) і посилання на кінець черги (last). Алгоритм дії функції наступний:

- Відкривається файл з ім'ям "test.txt" за допомогою об'єкту ifstream.

- В циклі while виконується наступне:

- Зчитується значення з файлу у змінну symbol.

- Викликається функція enqueue, яка додає значення symbol до черги.

- Після завершення циклу файл закривається.

**4. Функція print:**

Ця функція виводить елементи черги на екран. Вона приймає початковий вказівник черги (L). Алгоритм дії функції наступний:

- В циклі while виконується наступне:

- Виводиться значення info поточного елемента черги.

- Вказівник L оновлюється, щоб вказувати на наступний елемент черги.

- Після завершення циклу виводиться символ нового рядка.

**5. Функція count:**

Ця функція обробляє чергу і виводить статистичні дані, такі як мінімальний та максимальний елементи, їх позиції в черзі та середнє арифметичне. Вона приймає початковий вказівник черги (L). Алгоритм дії функції наступний:

- Ініціалізуються змінні min, max, min\_pos, max\_pos і k зі значенням 0.

- В циклі while виконується наступне:

- Збільшується значення k на 1.

- Перевіряється, чи поточний елемент черги є меншим за min. Якщо так, то оновлюється min і min\_pos.

- Перевіряється, чи поточний елемент черги є більшим за max. Якщо так, то оновлюється max і max\_pos.

- Вказівник L оновлюється, щоб вказувати на наступний елемент черги.

- Після завершення циклу перевіряється, чи існує останній елемент черги (L != NULL).

- Якщо так, то оновлюється k на k + 1.

- Перевіряється, чи поточний елемент черги є меншим за min. Якщо так, то оновлюється min і min\_pos.

- Перевіряється, чи поточний елемент черги є більшим за max. Якщо так, то оновлюється max і max\_pos.

- Виводяться статистичні дані, такі як мінімальний елемент і його позиція, максимальний елемент і його позиція, а також середнє арифметичне між мінімальним і максимальним значеннями.

Основна функція **main** викликає вищезгадані функції для читання з файлу, виведення на екран та обробки черги, а також виводить вміст черги після виконання усіх операцій.

**ВІДПОВІДІ НА ПИТАННЯ 12.6**

**ЕКЗАМЕН 5**

**1) Загальний синтаксис оголошення елемента черги та вказівника на нього.**

- Структура Elem має поля link (вказівник на наступний елемент черги) та info (інформація елемента черги).

**2) Загальні схеми створення черги (добавлення елемента), знищення черги (вилучення елемента).**

- Добавлення елемента: За допомогою функції enqueue, створюється новий елемент, який додається до кінця черги.

- Вилучення елемента: За допомогою функції dequeue, перший елемент черги вилучається, а черга зміщується.

**3) Загальні схеми обходу черги.**

- Для обходу черги можна використати цикл, який пройде через кожен елемент черги, використовуючи вказівник на початок черги.

**4) Доступ до елементів черги.**

- Доступ до елементів черги здійснюється за допомогою поля info кожного елемента черги.

**5) Дії із чергою.**

- Додавання елемента в чергу (enqueue).

- Вилучення елемента з черги (dequeue).

- Обхід черги.

- Перевірка порожністі черги.

- Отримання першого елемента черги без вилучення.

**6) Внутрішня реалізація черги.**

- Внутрішня реалізація черги базується на однозв'язному списку, де кожен елемент містить вказівник на наступний елемент.

**7) Передавання та опрацювання черг у функціях.**

- Черга може передаватись у функції за допомогою посилань або вказівників на початок і кінець черги.

- Опрацювання черги здійснюється за допомогою декількох функцій, які можуть використовувати передані посилання або вказівники для зміни черги.