Лабораторна робота №12

Рекурсивні функції в мові програмування С/С++

Мета. Придбати практичні навички щодо розроблення програм із використанням рекурсивних функцій та їх використанням на мові програмування С/С++. Розвиток навичок аналізу рекурсивних алгоритмів та їх реалізація в програмному коді.

Загальні відомості

Вступ до рекурсії

Рекурсія в програмуванні - це техніка, за допомогою якої функція викликає сама себе прямо чи опосередковано. Рекурсивні функції використовуються для вирішення задач, які можуть бути розділені на подібні підзадачі меншого розміру. Важливою концепцією рекурсії є "базовий випадок" - умова, при якій рекурсія припиняється, щоб уникнути нескінченного виклику функції.

Принцип роботи

Коли рекурсивна функція викликає сама себе, вона робить це з новим набором параметрів, приближаючись до базового випадку. Кожен виклик функції зберігається в стеку викликів до тих пір, поки не буде досягнуто базового випадку. Після цього стек починає розгортатись, повертаючи контроль до попередніх рівнів виклику, аж до першого виклику функції.

Переваги рекурсії:

Спрощення коду: Деякі алгоритми, такі як бінарний пошук або обхід дерева, набагато простіше реалізувати за допомогою рекурсії.

Природність розв'язання: Рекурсивні методи часто відповідають математичним означенням алгоритмів або структур даних, спрощуючи розуміння і аналіз.

Недоліки рекурсії:

Витрати пам'яті: Кожен виклик рекурсивної функції займає додатковий простір у стеку викликів, що може призвести до переповнення стеку при глибоких рекурсіях.

Швидкодія: Рекурсивні виклики можуть бути менш ефективними за ітеративні через додаткові витрати на виклик функцій та повернення з них.

Основні терміни:

Базовий випадок: Умова, за якої рекурсія припиняється.

Рекурсивний випадок: Частина функції, де відбувається виклик самої функції з новими параметрами.

Стек викликів: Структура даних, що зберігає інформацію про активні виклики функцій та їх параметри.

Основне завдання.

Написати програму на мові C/C++, яка використовує рекурсивну функцію для обчислення індивідуального завдання. Всі дані вводяться користувачем через консольний ввід, окрім елементів масиву які заповняються за допомогою генератора випадкових чисел (якщо це необхідно до індивідуального завдання).

Індивідуальне завдання

1. Рекурсивне обчислення факторіалу числа.
2. Рекурсивний алгоритм пошуку максимального елемента в масиві.
3. Рекурсивне обчислення n-го числа Фібоначчі.
4. Рекурсивне обернення рядка (перевертання рядка).
5. Рекурсивний алгоритм для обчислення суми цифр цілого числа.
6. Рекурсивний алгоритм для виведення елементів масиву в зворотному порядку.
7. Рекурсивне обчислення степеня числа.
8. Рекурсивний алгоритм для перевірки, чи є слово паліндромом.
9. Рекурсивний алгоритм для обчислення суми цифр в числі, що є степенем двійки.
10. Рекурсивний алгоритм для визначення, чи є масив впорядкованим за зростанням.
11. Рекурсивний алгоритм для визначення кількості нульових елементів у масиві.
12. Рекурсивне знаходження найменшого спільного кратного (НСК) двох чисел.
13. Рекурсивний алгоритм пошуку максимального елемента в масиві.
14. Рекурсивне обчислення n-го числа Фібоначчі.

Додаткові вимоги виконання завдання:

– звіт має бути виконаний згідно з вимогами до оформлення робіт;

Контрольні питання.

1. Що таке рекурсія у програмуванні? Наведіть визначення.
2. Як працює рекурсивна функція на прикладі обчислення факторіалу числа?
3. Які є основні компоненти рекурсивної функції?
4. Що таке базовий випадок у контексті рекурсивної функції і чому він необхідний?
5. Як рекурсія впливає на стек викликів у програмі? Що може статися при занадто глибокій рекурсії?
6. У чому полягають переваги та недоліки використання рекурсії замість ітерації?
7. Наведіть приклад задачі, яка ефективніше вирішується за допомогою рекурсії, а не ітерації.
8. Які стратегії можна використовувати для перетворення рекурсивного алгоритму на ітеративний?