Experiment No. 7

Seating Arrangement Permutations Using Iteration and Recursion in C

Aim: Event Planning – Seating Arrangement Calculation

You are tasked with organizing a formal event—such as a wedding, conference, or banquet—and need to calculate the number of possible ways to arrange a given number of guests in a row. This requires computing permutations based on the total number of guests, which is mathematically represented by the factorial of the number. Develop a C program that calculates the factorial using both iterative and recursive methods to determine all possible unique seating arrangements. This mirrors real-world event planning challenges where optimal arrangements and guest management are essential.

Learning Outcomes:

After completing this experiment, students will be able to:

- Understand the mathematical concept of permutations and factorials.
- Implement both iterative and recursive approaches in C programming.
- Compare the efficiency and logic behind iteration and recursion.
- Enhance problem-solving skills for real-world applications like event organization.
- Develop the ability to simulate and analyze computational logic using control structures.

Theory:

In programming, factorials are used to calculate permutations and combinations. The factorial of a number n (n!) represents the total possible arrangements of n distinct items in a sequence. For example, if there are 5 guests, the total possible seating arrangements:

$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$.

In C, factorials can be calculated using two main methods:

1. Iteration – using loops (for/while) to multiply numbers sequentially.
2. Recursion – a function calling itself until a base condition is reached.

Example concepts used in this program:

```
// Iterative approach
int factorial_iterative(int n) {
    int fact = 1;
    for(int i = 1; i <= n; i++)
        fact *= i;
    return fact;
}

// Recursive approach
```

```
int factorial_recursive(int n) {
    if (n == 0 || n == 1)
        return 1;
    else
        return n * factorial_recursive(n - 1);
}
```

Both methods serve the same purpose but have different performance characteristics. Recursion offers elegant logic but can be slower due to function call overhead, whereas iteration is more memory-efficient.

Program Code:

Students are expected to implement this experiment by writing the complete C program for the Event Planning – Seating Arrangement Calculation using Iteration and Recursion in C.

Output:

Students are expected to execute the program and provide various sample outputs for the same code to demonstrate its functionality with different input values.

Conclusion: In this experiment, an Event Seating Management System was successfully developed using Iteration and Recursion in C.