**Лабораторная работа №3**

**Разработка программных модулей**

“Функции и рекурсия”

Студент-Алиев Тимур Заурович

Группа-VДКИП 111-прог

Преподаватель-Сибирев Иван Валерьевич

**Цель работы**

Изучение способов реализации алгоритмов с использованием рекурсии.

**Краткие теоретические сведения**

**Рекурсия** — это способ организации вычислительного процесса, при котором функция в ходе выполнения операторов обращается сама к себе. Примером может служить вычисление факториала:

double Faktorial\_R(int n) {

if (n < 2) return 1; // Условие окончания рекурсии

else

return n \* Faktorial\_R(n - 1); // Рекурсивный вызов

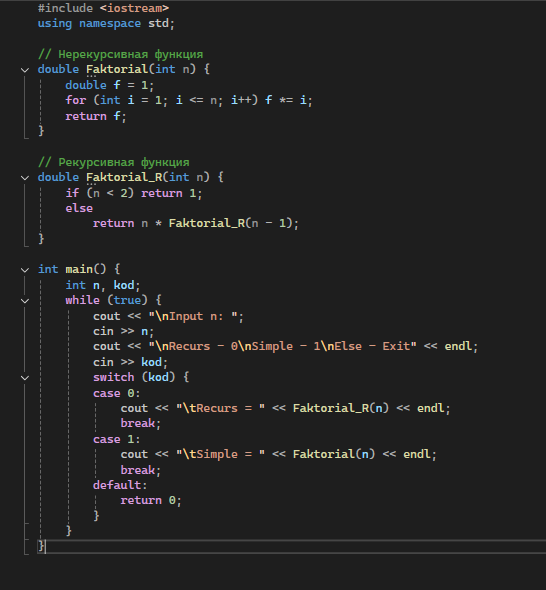
}

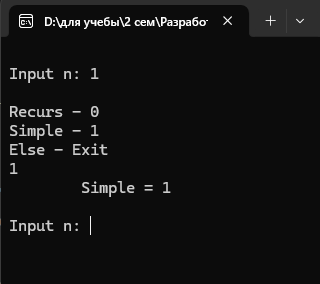
Правильно организованная рекурсивная функция должна:

1. Иметь условие окончания рекурсии (базовый случай).
2. Упрощать задачу на каждом шаге, приближаясь к базовому случаю.

**Пример выполнения задания**

**Задача:** Написать программу вычисления факториала положительного числа n*n*, используя рекурсивную и нерекурсивную функции.





<https://github.com/VinoDelJpg/Lab3>

**Блок-схема рекурсивной функции**

1. **Начало функции** — проверка условия n<2*n*<2.
2. Если условие истинно, возвращается 1.
3. Иначе — возвращается .

**Индивидуальные задания**

1. **Перевод числа в p-ичную систему счисления (p < 10)**  
   Алгоритм:
   * Если число N*N* меньше p*p*, вывести N*N*.
   * Иначе рекурсивно перевести N/p*N*/*p* в p-ичную систему и вывести остаток от деления N%p*N*%*p*.
2. **Двоичный поиск в упорядоченном массиве**  
   Алгоритм:
   * Найти средний элемент массива.
   * Если он равен c*c*, вернуть его индекс.
   * Если меньше c*c*, рекурсивно искать в правой половине.
   * Иначе — в левой половине.
3. **Нахождение НОД чисел M*M* и N*N* по теореме Эйлера**  
   Алгоритм:
   * Если M*M* делится на N*N*, вернуть N*N*.
   * Иначе рекурсивно найти НОД(N*N*, M%N*M*%*N*).
4. **Числа Фибоначчи**  
   Алгоритм:
   * Если n=0*n*=0, вернуть 0.
   * Если n=1*n*=1, вернуть 1.
   * Иначе вернуть Fb(n−1)+Fb(n−2)*Fb*(*n*−1)+*Fb*(*n*−2).
5. **Функция Аккермана**  
   Алгоритм:
   * Если m=0*m*=0, вернуть n+1*n*+1.
   * Если n=0*n*=0, вернуть A(m−1,1)*A*(*m*−1,1).
   * Иначе вернуть A(m−1,A(m,n−1))*A*(*m*−1,*A*(*m*,*n*−1)).
6. **Метод деления отрезка пополам для нахождения минимума функции**  
   Алгоритм:
   * Разделить отрезок пополам.
   * Вычислить значения функции в точках.
   * Рекурсивно сузить отрезок до половины, где функция меньше.
7. **Вычисление квадратного корня по рекуррентной формуле**  
   Алгоритм:
   * Начальное приближение x0=0.5(1+a)*x*0​=0.5(1+*a*).
   * Рекурсивно вычислять xn=0.5(xn−1+axn−1)*xn*​=0.5(*xn*−1​+*xn*−1​*a*​).
8. **Максимальный элемент в массиве**  
   Алгоритм:
   * Если массив из одного элемента, вернуть его.
   * Иначе рекурсивно найти максимум в подмассиве и сравнить с последним элементом.
9. **Вычисление вложенных корней**  
   Алгоритм:
   * Базовый случай: y(1)=1*y*(1)=1​.
   * Рекурсивно: y(n)=n+y(n−1)*y*(*n*)=*n*+*y*(*n*−1)​.
10. **Максимальный элемент в массиве методом деления пополам**  
    Алгоритм:
    * Разделить массив пополам.
    * Рекурсивно найти максимум в каждой половине.
    * Вернуть больший из двух максимумов.

**Контрольные вопросы**

1. **Что такое рекурсивная функция?**  
   Это функция, которая вызывает саму себя в процессе выполнения.
2. **Может ли рекурсивная функция содержать несколько операторов return?**  
   Да, если это необходимо для обработки разных условий (например, базовый случай и рекурсивный случай).
3. **Какие условия должны выполняться для корректной работы рекурсивной функции?**
   * Наличие базового случая.
   * Упрощение задачи на каждом шаге.
   * Гарантированное достижение базового случая.
4. **В чем преимущества и недостатки рекурсии?**
   * Преимущества: лаконичность кода, простота реализации для задач, которые естественно рекурсивны.
   * Недостатки: возможные проблемы с производительностью и переполнением стека.