**Практическая работа №5**

**Рекурсия, рекурсивные алгоритмы**

1. Изучите материал в презентации.

**Ответ: изучен**

1. **Ответ:**

**Рекурсия** (широком смысле) — это определение объекта посредством ссылки на себя.

1. **Ответ:**

**Зеркала друг напротив друга**

Когда два зеркала размещены друг напротив друга, они создают эффект бесконечного повторения изображений, где каждое отражение содержит отражение предыдущего.

**Русские матрёшки**

Матрёшка — это серия деревянных фигур, каждая из которых содержит внутри себя уменьшенную копию.

**Ветки деревьев**

Дерево растёт так, что каждая ветвь делится на меньшие ветки, которые, в свою очередь, делятся дальше.

**Фракталы в природе**

Пример — листья папоротника или снежинки. Узор повторяется на каждом уровне увеличения.

**Семейное дерево**

Семейное дерево отображает связь поколений. Каждый предок порождает своих потомков, которые в свою очередь имеют своих.

1. **Ответ:**

**Рекурсивный алгоритм** — это алгоритм, в определении которого содержится прямой или косвенный вызов этого же алгоритма

1. **Ответ:**

**Рекурсивную триаду** — это этапа для решения задач рекурсивными методам: параметризация, базовый случай и декомпозиция.

1. **Ответ:**

Для оценки трудоемкости рекурсивных алгоритмов строится **полное дерево рекурсии**.

**Глубина рекурсивных вызовов**- наибольшее одновременное количество рекурсивных обращений функции, определяющее максимальное количество слоев рекурсивного стека, в котором осуществляется хранение отложенных вычислений.

**Объем рекурсии** - количество вершин полного рекурсивного дерева без единицы

1. **Ответ:**

**Рекурсивный стек**

Для каждого текущего обращения формируется локальный слой данных стека (при этом совпадающие идентификаторы разных слоев стека независимы друг от друга и не отождествляются).

Завершение вычислений происходит посредством восстановления значений данных каждого слоя в порядке, обратном рекурсивным обращениям.

Количество рекурсивных обращений ограничено размером области памяти, выделяемой под программный код.

1. Пример рекурсивной процедуры:

def factorial(n):

if n == 0 or n == 1: # базовый случай

return 1

else:

return n \* factorial(n - 1) # рекурсивный случай

**Рекурсивная триада:**

Базовый случай: в примере базовый случай наступает при n=0n = 0n=0 или n=1n = 1n=1, когда возвращается 1.

Рекурсивный случай: В примере это выражение n×factorial(n−1)n \times factorial(n - 1)n×factorial(n−1).

Прогрессивное приближение к базовому случаю: Каждый рекурсивный вызов уменьшает аргумент nnn, приближаясь к базовому случаю.

**Полное дерево рекурсии:**

factorial(4)

/ \

factorial(3) 4 \*

/ \

factorial(2) 3 \*

/ \

factorial(1) 2 \*

/ \

1. 1

**Глубина дерева**

глубина рекурсивного дерева равна 4

**Объем рекурсии**

Объем рекурсии - O(h), глубина это h и она равна 4. Поэтому объем рекурсии будет O(4).

1. **Ответ:** суть мема в рекурсии. Мем обыгрывает, создавая визуальный цикл, Рекурсия указывает на саму себя, что отражает бесконечный и самореференциальный характер рекурсии.

