**Практическая работа №5**

**Рекурсия, рекурсивные алгоритмы**

*2. Дайте определение рекурсии (в широком смысле):*

**Рекурсия в широком смысле** - это определение объекта посредством ссылки на себя.

*3. Приведите примеры рекурсии в жизни и опишите их (4-5 примеров, можно фото):*



Рис 1 – Матрешки

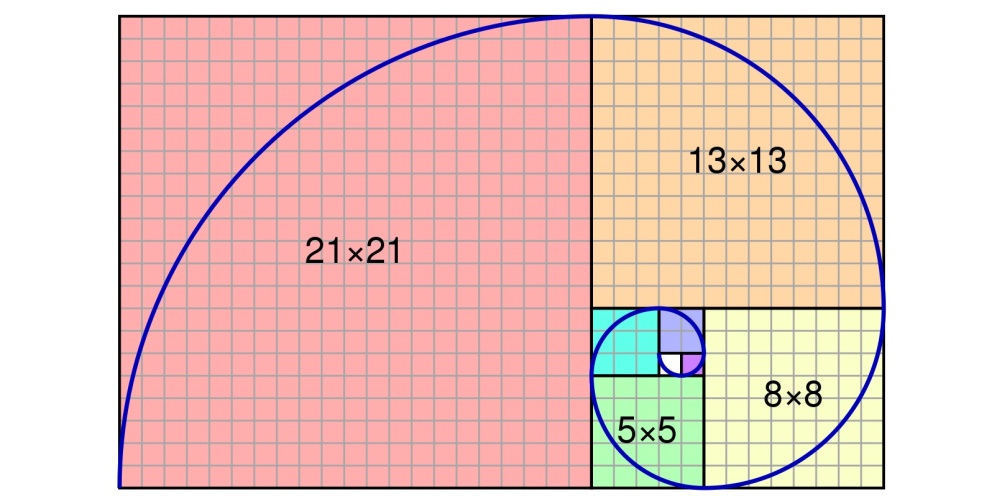


Рис 2 – Золотое сечение

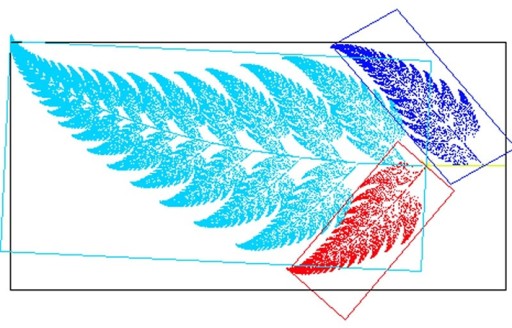


Рис 3 – Структура листа папоротника



Рис 4 – Ракушка

*4. Дайте определение рекурсивного алгоритма:*

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

**Прямая рекурсия** – подразумевает прямое обращение подпрограммы к самой себе в своем теле с измененными параметрами.

**Косвенная рекурсия** – подразумевает вызов других подпрограмм, в теле которых присутствует обращение к исходной подпрограмме с измененными параметрами.

**Линейная рекурсия** – в теле подпрограммы происходит только один вызов той же подпрограммы.

**Ветвящаяся рекурсия** – в теле подпрограммы может происходить несколько прямых или косвенных вызовов той же подпрограммы.

*5. Что такое рекурсивная триада:*

**Рекурсивная триада** – этапы, необходимые для разработки рекурсивного алгоритма:

**Параметризация** – выделение параметров, которые используются для описания условия задачи, а затем в решении.

**База рекурсии** – простой случай поставленной задачи, решение которого очевидно, на основе которого можно решить более сложные вариации задачи.

**Декомпозиция** – общие случаи задачи, которые необходимо преобразовать/выразить через простой случай.

*6. Что такое полное дерево, глубина и объем рекурсии:*

**Дерево рекурсии –** структура, строящаяся по принципу дерева, которая применяется для оценки трудоемкости рекурсивных алгоритмов.

**Глубина рекурсивных вызовов –** максимальное количество уровней вложенности рекурсивных вызовов, достигнутых во время выполнения программы.

**Объем рекурсии** – общее количество выполненных рекурсивных вызовов, включая уровни вложенности.

*7. Как называется область памяти, выделяемая для хранения всех промежуточных значений локальных переменных? Опишите ее:*

Так как рекурсивные подпрограммы несколько раз вызывают сами себя, то сам процесс выполнения этих подпрограмм на каждом звене не прерывается, а замораживается, до окончания выполнения вызванной подпрограммы. Область памяти, предназначенная для хранения всех промежуточных значений работы подпрограммы, называется **рекурсивным стеком**.

Для каждого нового рекурсивного обращения формируется новый слой стека (количество слоев строго ограничено) Когда слои стека накладываются друг на друга, т.е. подпрограммы вызывают самих себя, происходит **рекурсивный подъем**.

После окончания работы последнего обращения, происходит возобновление работы слоя стека (т.е. обращения к подпрограмме) находящегося под ним. С этого момента вызванные процедуры постепенно начинают завершаться, т.е. происходит **рекурсивный спуск**.

*8. Приведите пример рекурсивной процедуры/функции. Постройте полное дерево рекурсии, определите глубину и объем рекурсии. Укажите рекурсивную триаду:*

**Функция – вычисление факториала.**

**function** factorial(i:integer): integer;

**begin**

**if** i < 2 **then**

factorial := 1

**else**

factorial := i \* factorial(i-1);

**end**;

Код функции

**Глубина рекурсии**: глубина рекурсии при вычислении факториала числа n, будет составлять n.

**Объем рекурсии:** т.к. рекурсия в данной функции линейная, то объем рекурсии будет равен ее глубине, т.е. при вычислении факториала от n числа, объем рекурсии будет равен n.

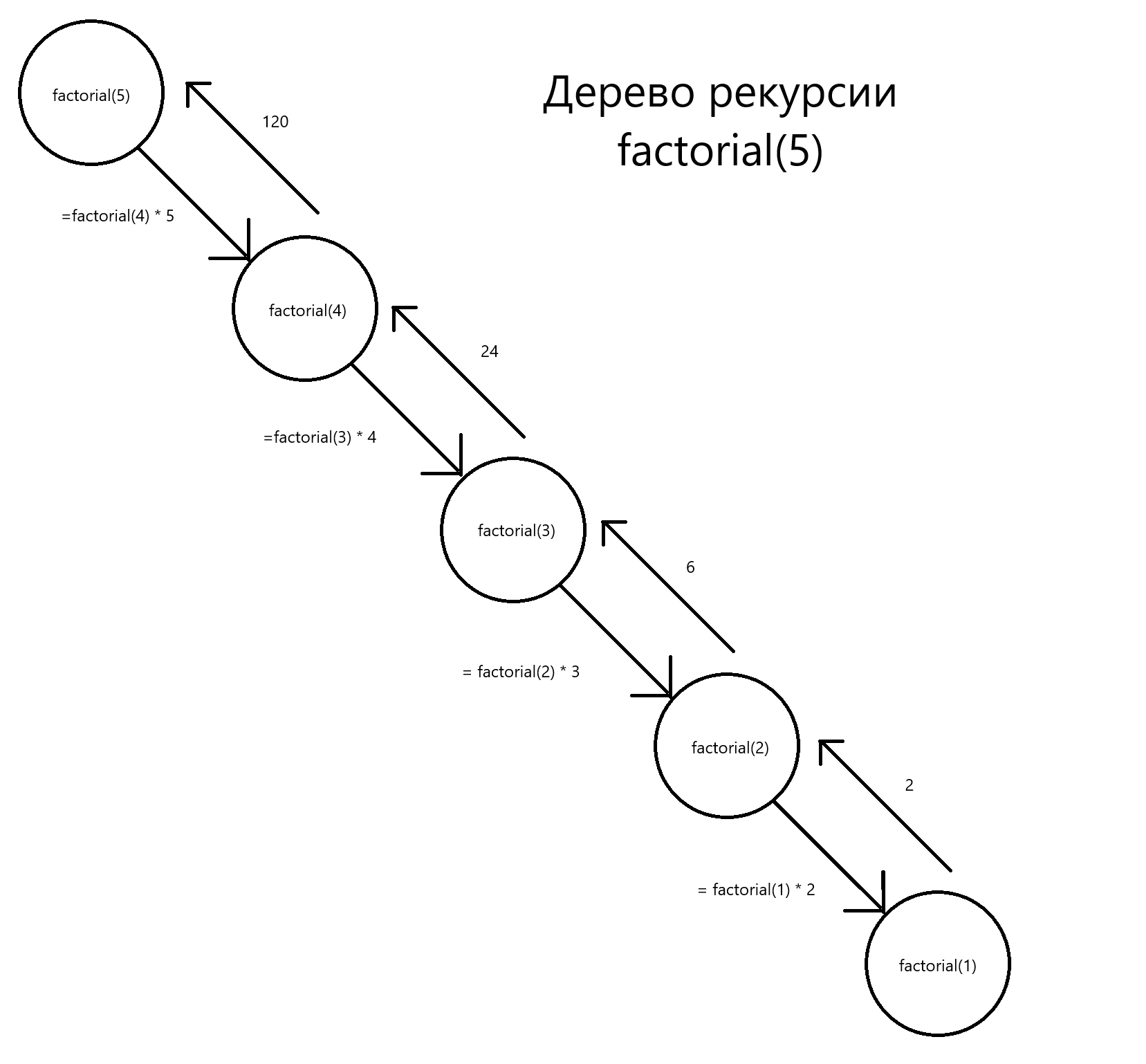


Рис 5. – Дерево рекурсии

**Рекурсивная триада:**

**Параметризация**: используется один параметр – i. Он отвечает за то число, факториал которого необходимо вычислить. В каждом новом рекурсивном вызове данный параметр перемножается с его последующим значением, которое определяется новым рекурсивным вызовом. При новом вызове в функцию передается значение i, уменьшенное на 1, таким образом будет происходит подсчет произведений чисел до того момента, пока число i не достигнет базового случая.

**Базовый случай**: базовым случаем является факториалы от 0 или 1, значение которых известны и равны 1.

**Декомпозиция**: общий случай, при котором необходимо умножить текущее значение параметра i на результат рекурсивного вызова функции.

*9. Приведите примеры мемчиков на рекурсии. Опишите их (желательно сделать свой).*

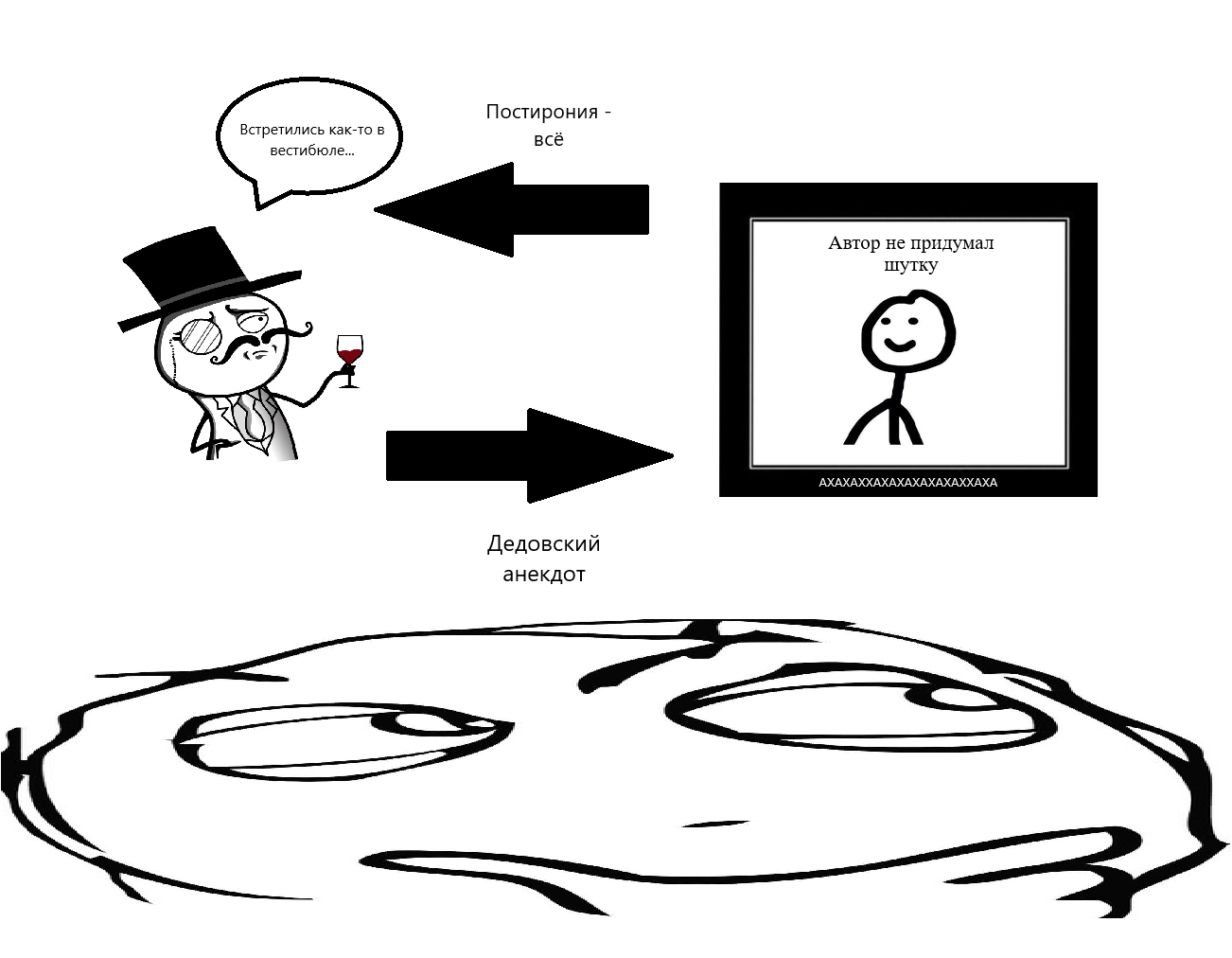


Рис 6. – Творческая часть работы