# Функции: рекурсия

Урок №7

# Вспомним прошлое...

В какой момент программы мы можем вызвать функцию?

Ответ:
В любой момент в основном коде. Объявить функцию нужно в начале файла с кодом.

# Вспомним прошлое...

Можно ли вызвать функцию из
**другой** функции?

Ответ: да

# А что если вызвать функцию из самой функции?

# Сегодня с этим и будем разбираться

# Научимся писать бесконечные программы и программы, которые сами себя запускают и останавливают

Раскроем термин “рекурсия”, чтобы стать на шаг ближе к сообществу профессиональных айтишников

в сегодняшнем выпуске

# Правда:

# Функция может вызывать саму себя!

Функция, вызывающая сама себя позволяет реализовать рекурсивный алгоритм

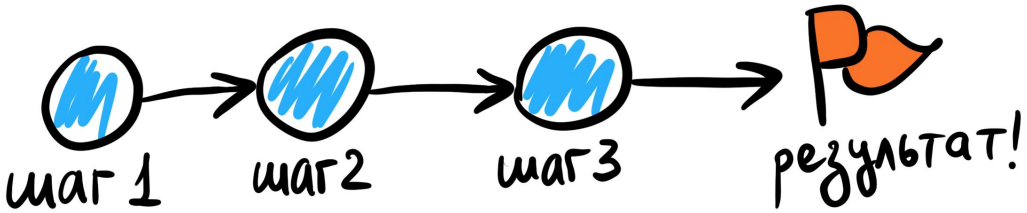
# Что такое “линейный алгоритм”?

Ответ: Алгоритм, который решает задачу, выполняя некоторые действия последовательно одно за другим. Такой алгоритм не имеет ветвлений и всегда получает результат, выполняя одинаковые шаги независимо от входных данных.

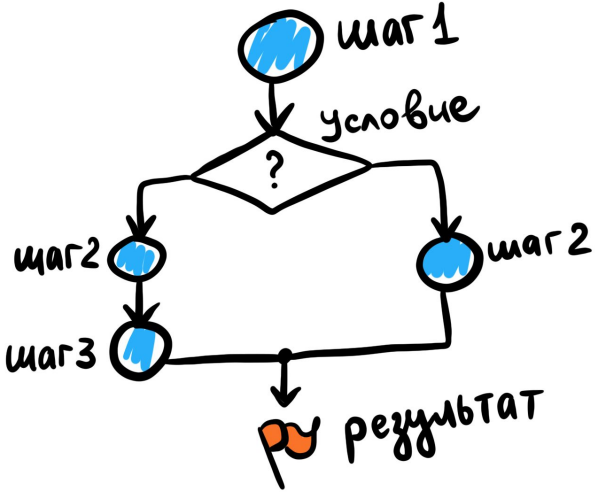
# Что такое “алгоритм с ветвлением”?

Ответ: Алгоритм, который выполняет разные шаги в зависимости от входных данных, т.к. в нём присутствует проверка(и) некоторого условия, от результата этой проверки зависит исход.

# Линейный алгоритм

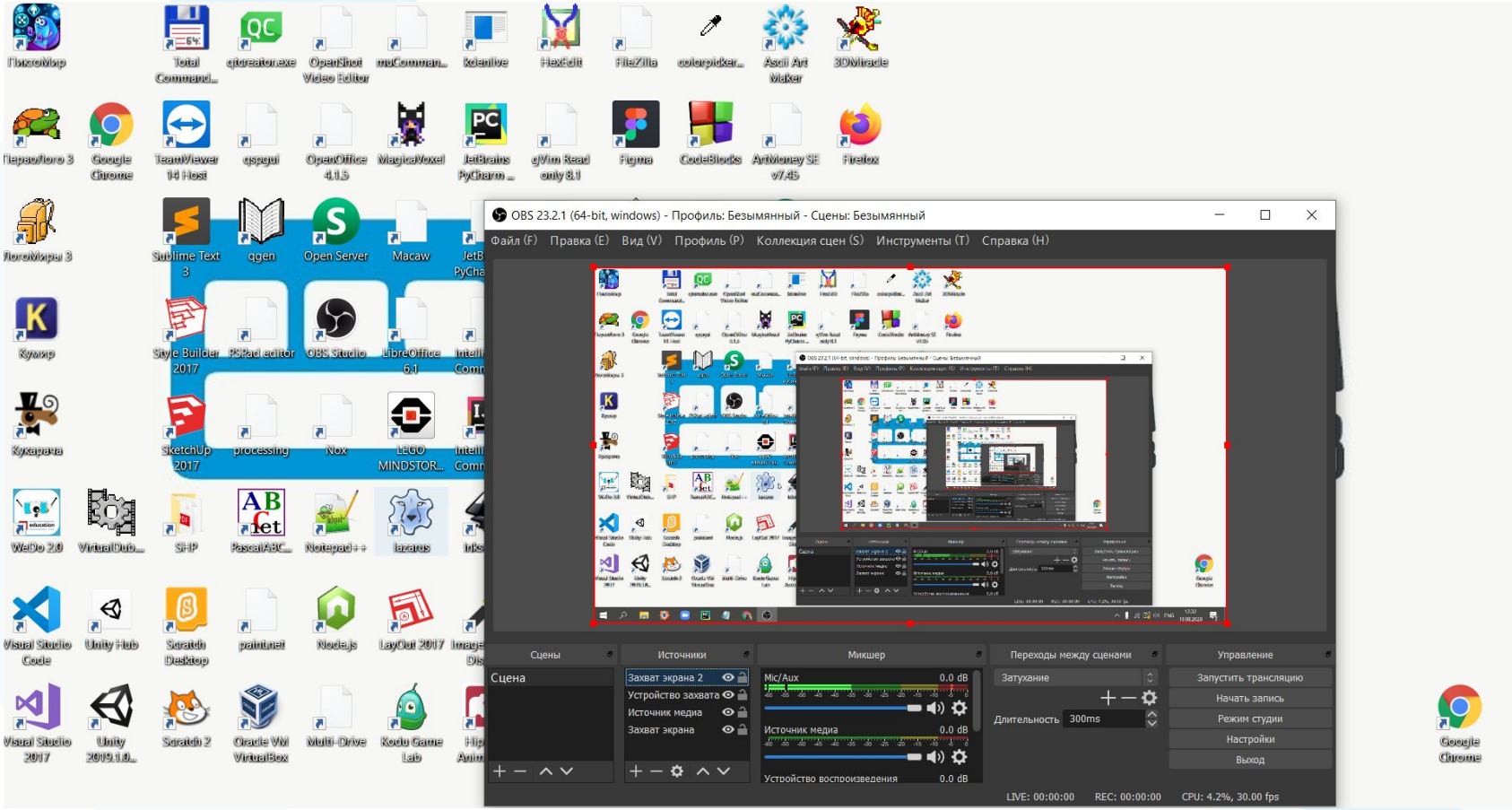



# Ветвление

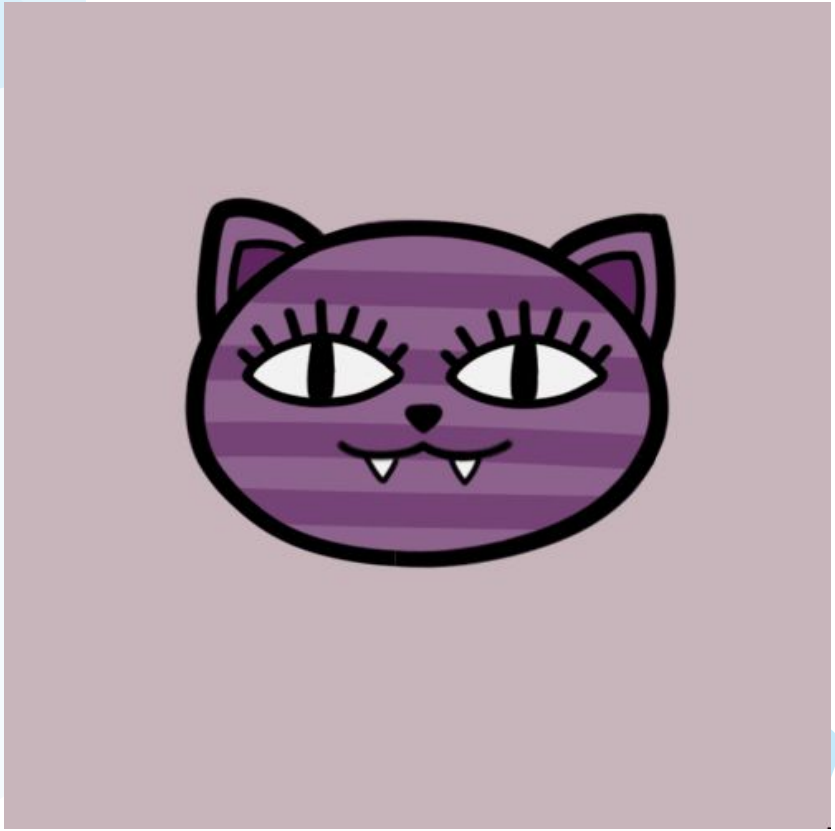


# Как выглядит рекурсия

# Экран показывает экран



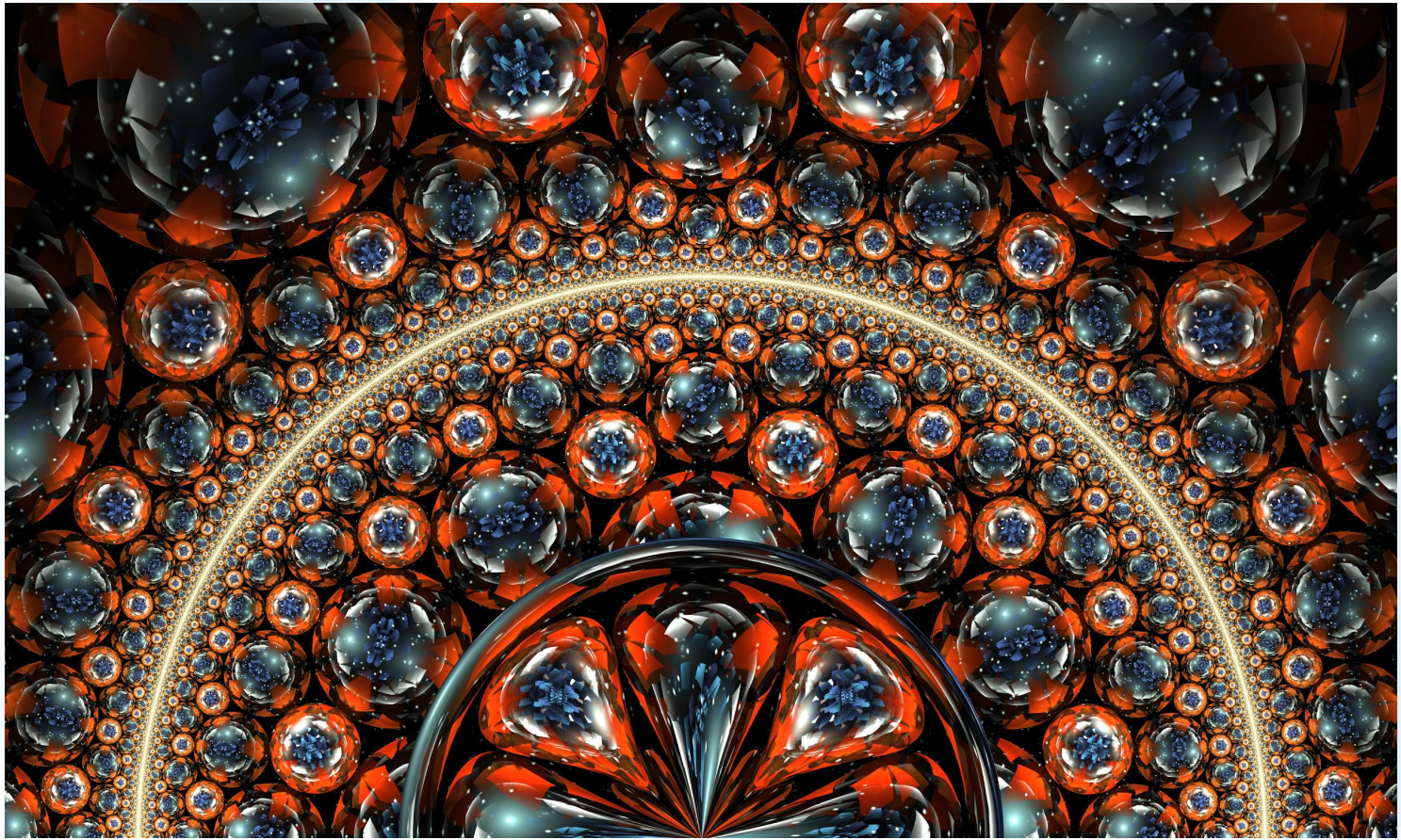
# Бесконечные гифки



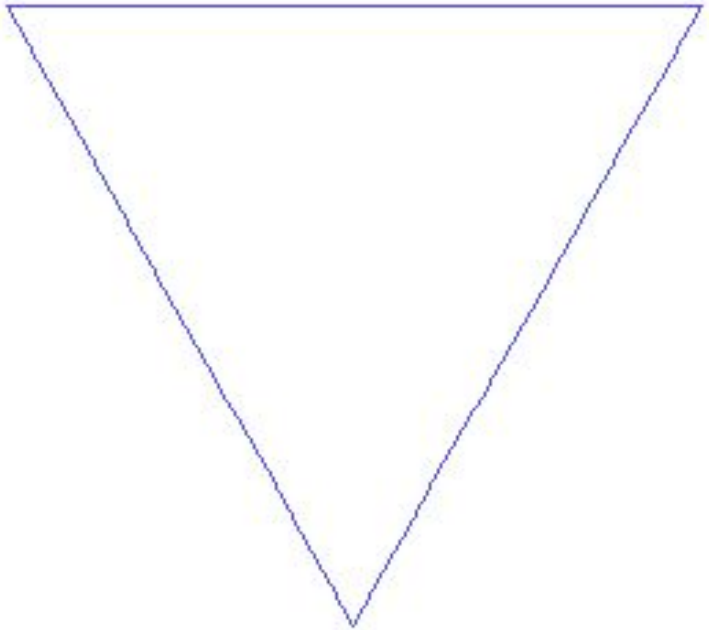
# Матрёшка

# Фрактальные рисунки

# Создано математически, а не вручную



# Пример того, как создаются такие рисунки



Рекурсивный алгоритм используется для решения математических задач

Иногда такое решение более красивое и понятное

# Идея рекурсии

**Ситуация:** вам прислали подарок на день рождения в посылке.

**Ваша цель:** открыть его и добраться сквозь упаковку до всех предметов.

# Зачем функции вызывать саму себя?

Имеется некоторая сложная задача.

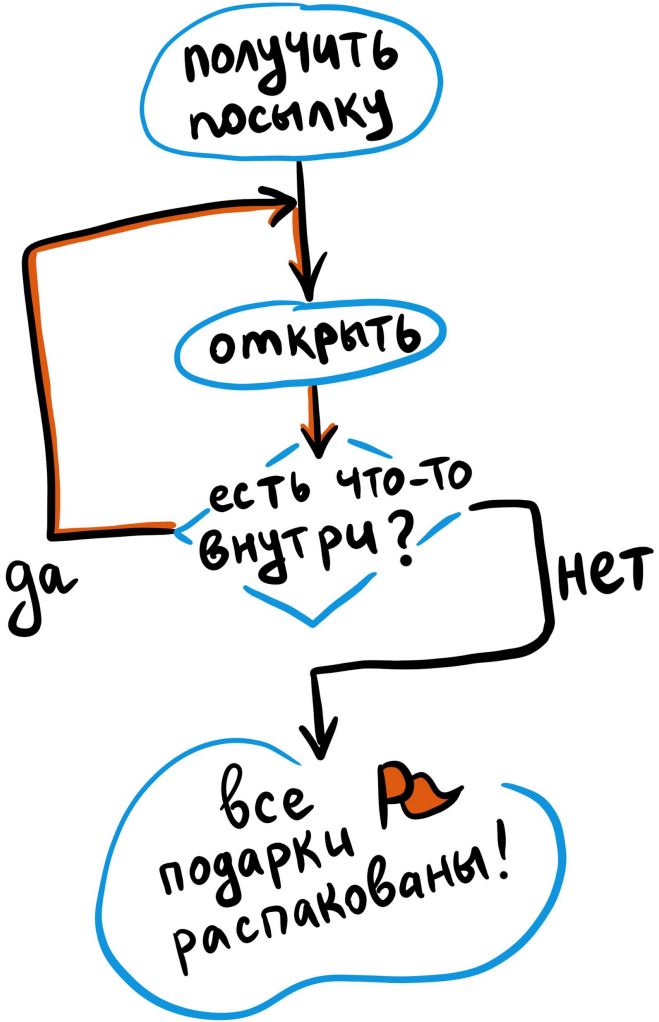
- Знаем, как решить аналогичную, но на более маленьком примере
- Можем найти взаимосвязь между маленьким примером и большим
- Получаем большой итоговый результат на основе маленького алгоритма, запущенного много раз.

# Пример с подарками

Напишем функцию

**def распаковать(объект):**

- открыть упаковку объекта
- если внутри найден другой объект, то вызвать **распаковать (другой объект)**



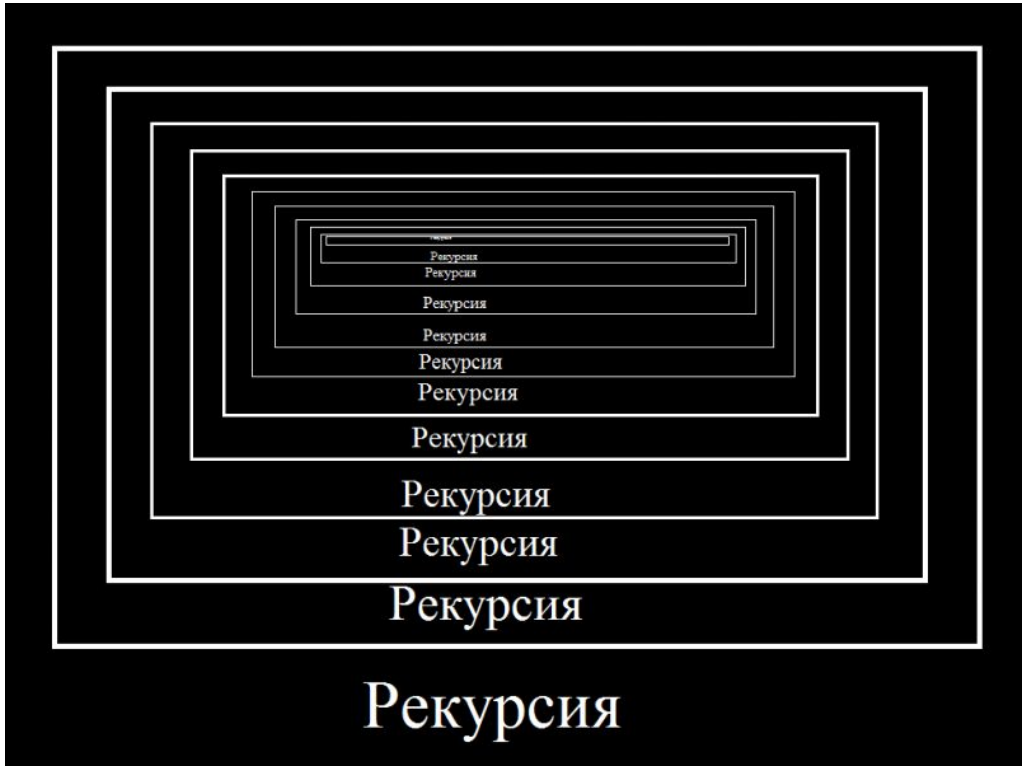
# Рекурсия

— это это алгоритм, который использует внутри сам себя.

В случае реализации:

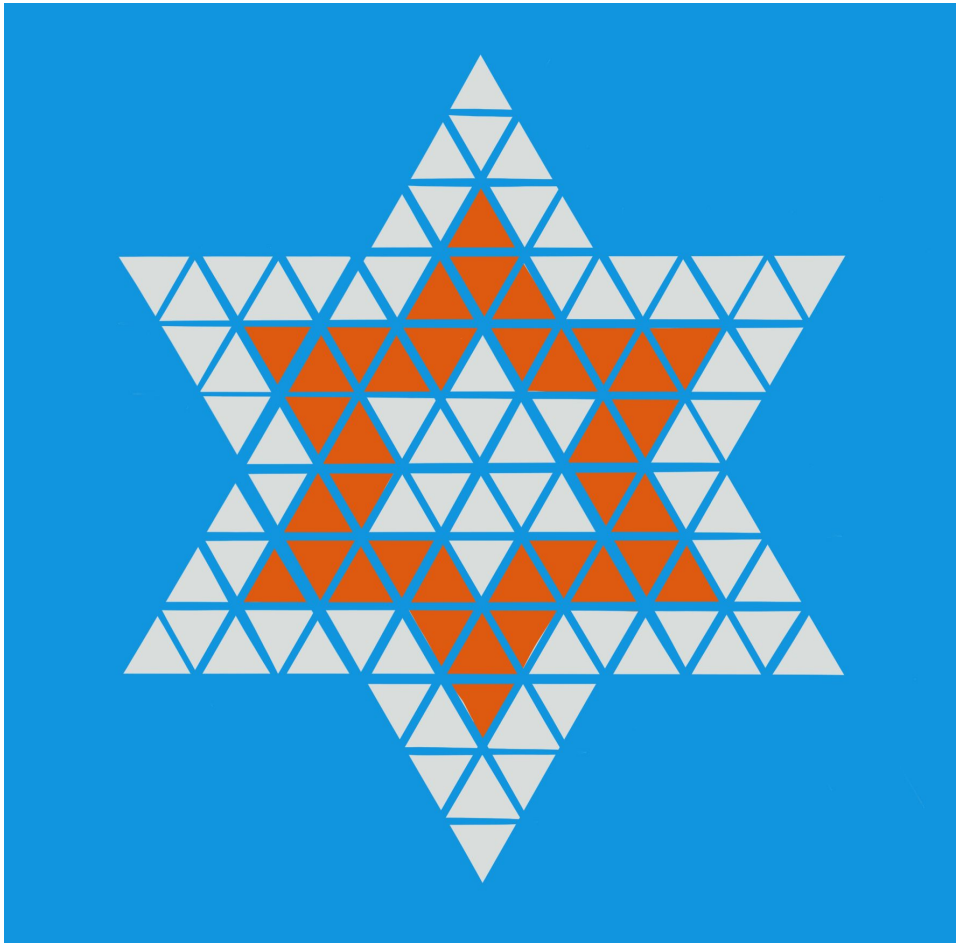
Это функция, которая вызывает сама себя

# Рекурсия - это



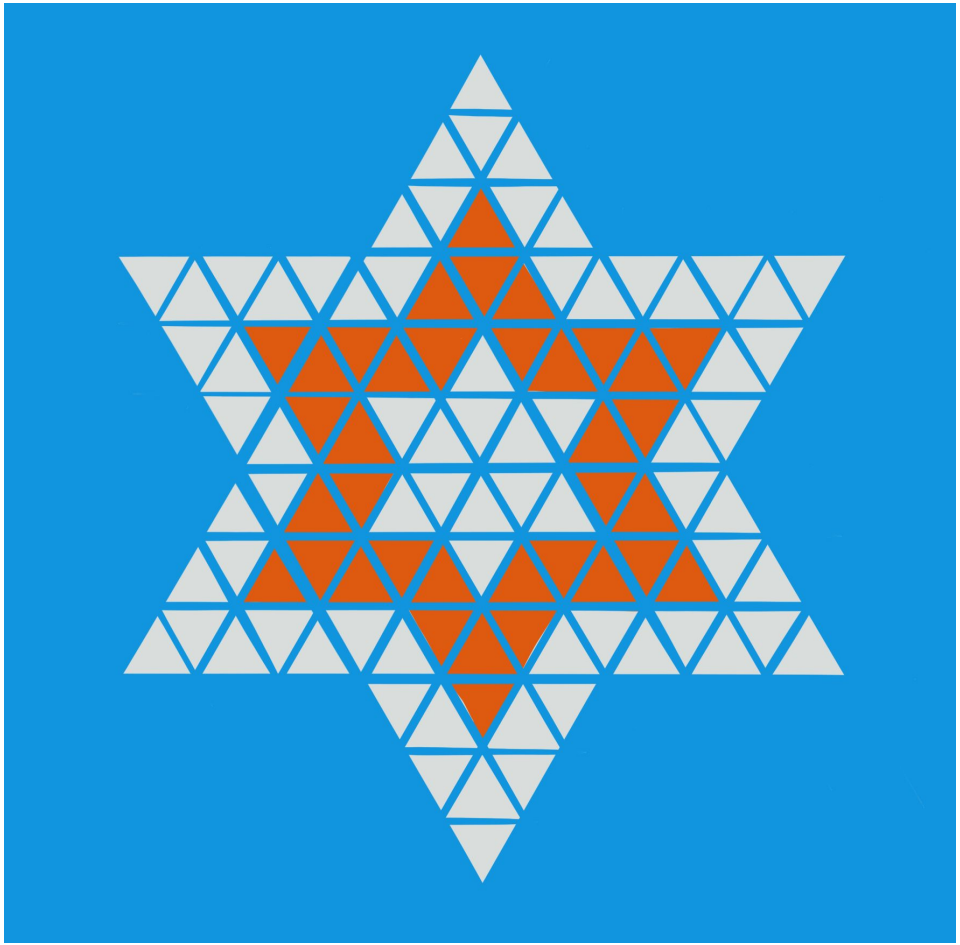
## Задача

Сколько треугольников плитки понадобиться, чтобы выложить следующий слой?



## Задача

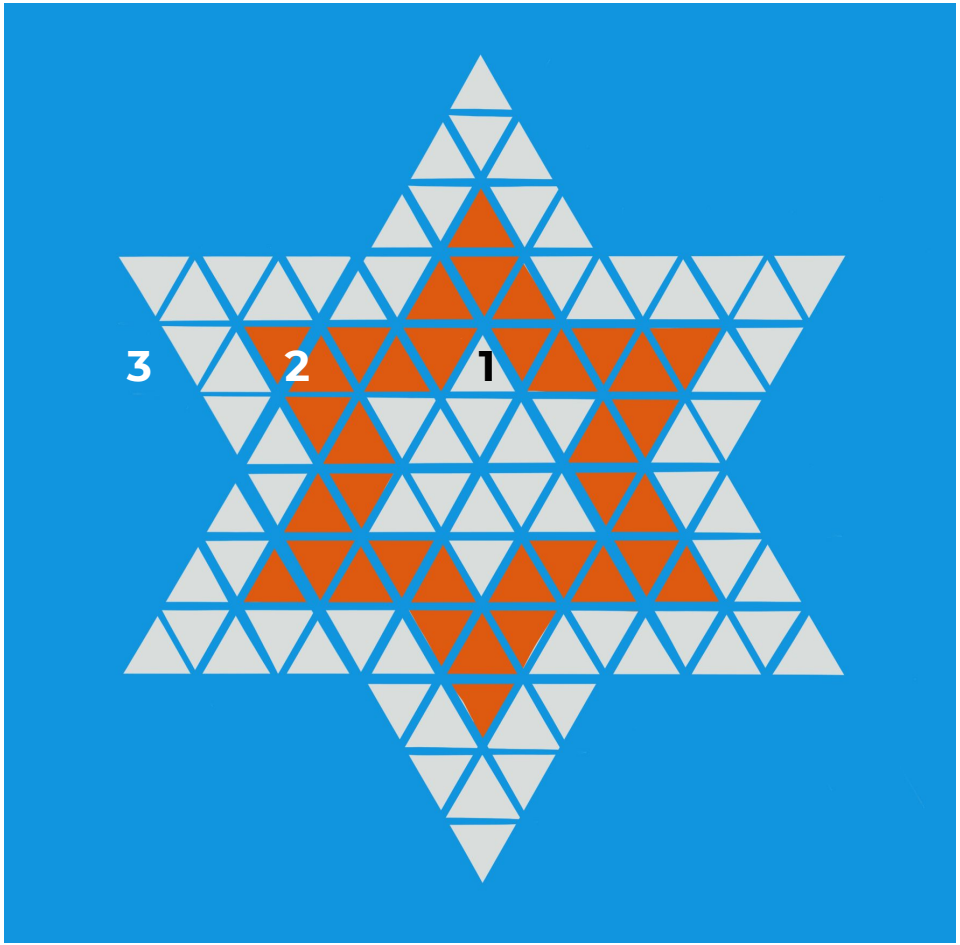
Сколько треугольников плитки понадобиться, чтобы выложить 101 слой?



## Решение

Знаем количество Δ в стартовой фигуре 1.

Видим слои 1, 2 и 3:

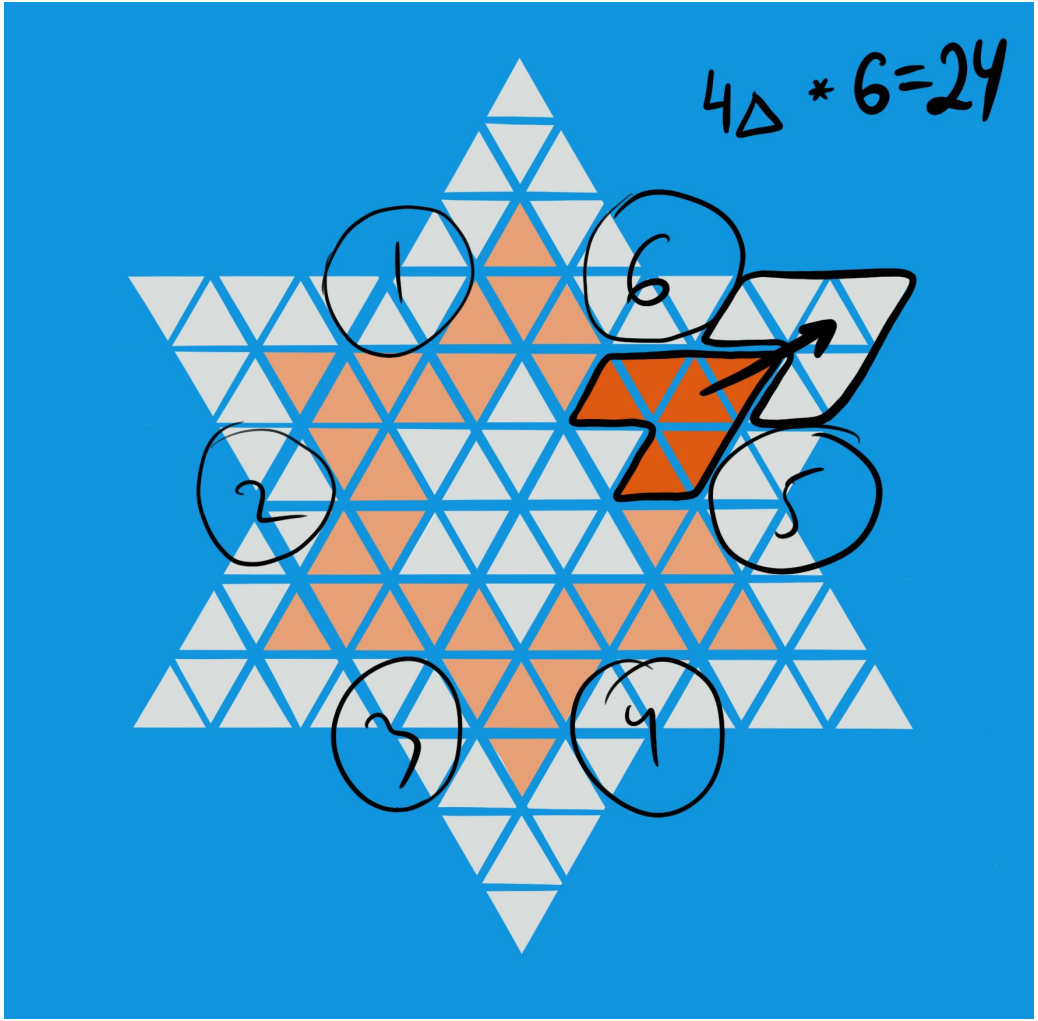


## Решение

Находим повторяющиеся части.

## Решение

Находим закономерность прироста Δ в каждом следующем слое.



## Решение

Получаем формулу:

Δ в слое **1** = 12

Δ в слое **2** = 12 + 24 = 36

Δ в слое **3** = 36 + 24 = 60

Δ в слое **4** = 60 + 24 = 84

Δ в слое **N** = Δ в слое **N-1** + 24

# Реализация на Питоне

```
def count(n):
    #n - это номер слоя
    if n == 1:
        return 12
    else:
        return count(n-1) + 24
#для 101 слоя вызов такой
print(count(101))
```

Крайний случай
(базовое значение = 12)

Шаг рекурсии

# Чек-лист для рекурсии

1. Найти закономерность получения следующего шага - **как углубляться в рекурсию**
2. Указать в функции момент остановки**!!!**
   (когда функция НЕ должна вызывать себя)
   **и знать значение базового случая**

# Чек-лист для рекурсии

1. Найти закономерность получения следующего шага - **как углубляться в рекурсию**

```
count(n-1) + 24
```

2. Указать в функции момент остановки!!!

   (когда функция НЕ должна вызывать себя)

   **и знать значение базового случая**

```
return 12
```

## Домашнее задание в EduApp

## Внимание!
## Творческий конкурс!