

Лекция №1 1
по дисциплине
«ТЕОРИЯ АЛГОРИТМОВ»

РЕКУРСИЯ

Преподаватель:
Золотоверх Д.О.

ОПРЕДЕЛЕНИЕ

Рекурсия – определение, описание, изображение какого-либо объекта или процесса внутри самого этого объекта или процесса.

Рекурсии свойственно создавать ситуацию, когда объект является частью самого себя.

В программировании прием может быть использован когда есть возможность поделить задачу на более простые аналогичные подзадачи.



ПРИМЕРЫ РЕКУРСИИ

Где встречается рекурсия

- Природа
- Фракталы
- Файловые системы
- Решение задач



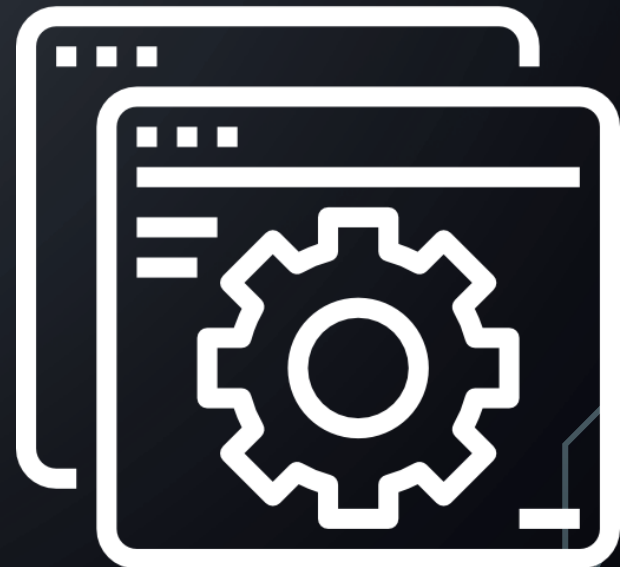
РЕКУРСИЯ В ПРОГРАММИРОВАНИИ

Рекурсией называется **прием** программирования, при котором **есть наличие функции, что вызывает сама себя.**

Такая функция называется **рекурсивной.**

Рекурсии в программировании свойственно иметь **предел** (**условие прекращения рекурсии**).

Такой механизм **необходим**, так как иначе выполнение программы невозможно.



РЕШЕНИЕ ЗАДАЧИ С ПОМОЩЬЮ РЕКУРСИИ

Перед вами **куча коробок**;

Некоторые коробки содержат **другие коробки внутри**;

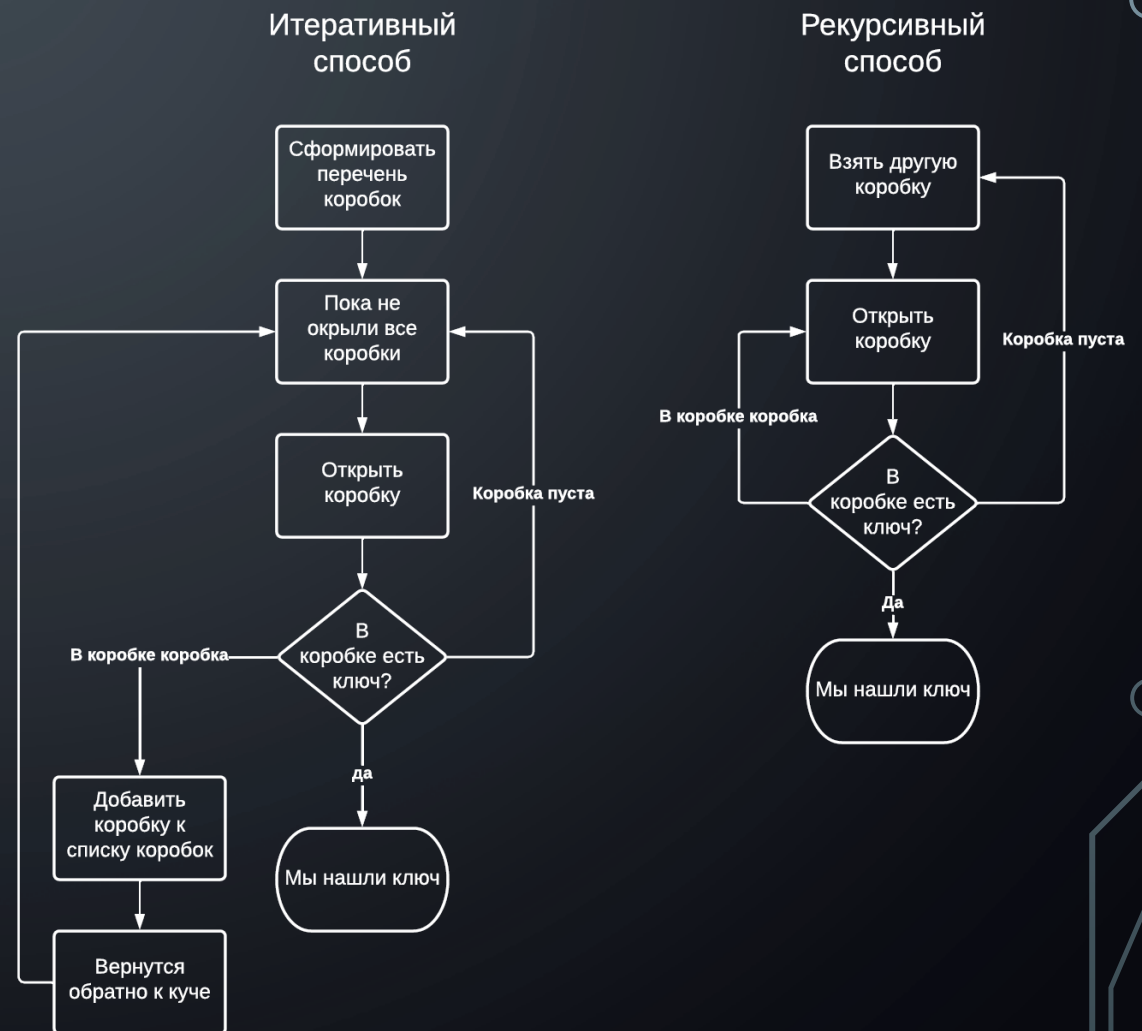
Некоторые коробки **пустые**;

Количество коробок заранее **не известно**;

В одной из коробок **ключ**;

Необходимо найти **ключ**.

Данную задачу можно решить с помощью **итеративного** и **рекурсивного** способов.



СТРУКТУРА РЕКУРСИИ

Рекурсивная функция всегда должна иметь **хотя бы две части**:

- **условие прекращения рекурсии** (терминальная ветвь);
- **рекурсивный вызов** (рекурсивная ветвь).

Иногда некоторым рекурсивным функциям свойственно:

- наличие нескольких **альтернативных ветвей**;
- наличие **параллельной рекурсии**.



РЕКУРСИЯ В C++

Как и любой другой язык программирования C++ поддерживает рекурсию.

Большинство языков ограничивают количество вызовов рекурсии (глубина рекурсии).

Python: 1000 JavaScript: 10000-100000

В C++ глубина рекурсии не ограничена количеством вызовов, она ограничена количеством памяти.

```
void recurse()  
{  
    .. .. .  
    recurse();  
    .. .. .  
}
```

```
int main()  
{  
    .. .. .  
    recurse();  
    .. .. .  
}
```

Рекур-
сивный
вызов

Вызов
функции

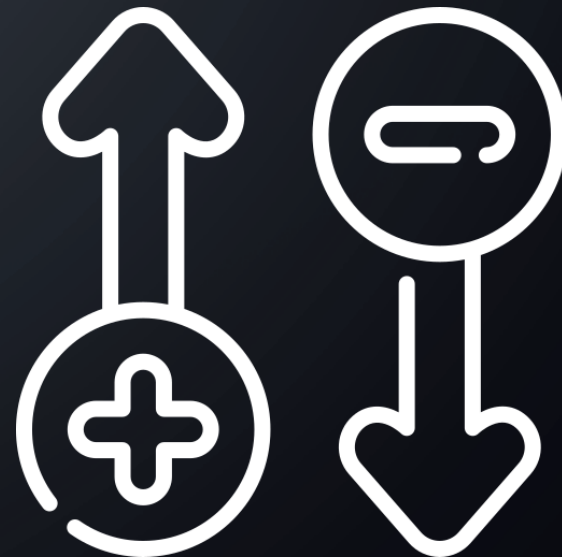
ПРЕИМУЩЕСТВА И НЕДОСТАТКИ

Преимущества:

- код **короче** и **чище**, более элегантный;
- рекурсия необходима в задачах, касающихся **структур данных** и **продвинутых алгоритмов**, таких как обход графа и дерева.

Недостатки:

- программы с рекурсией **занимают много места** на стеке по сравнению с итеративной программой;
- используют **больше процессорного времени**.
- **сложная отладка** по сравнению с эквивалентной итеративной программой.



ФАКТОРИАЛ ЧИСЛА С ПОМОЩЬЮ РЕКУРСИИ

Факториал — определённая на **множестве неотрицательных целых чисел**.

Факториалом числа n является **произведение всех натуральных чисел от 1 до n** .

Реализовать факториал можно с помощью рекурсии:

- 1) Осуществив рекурсивный вызов
- 2) Завершив выполнение терминальным условием

```
int factorial(int n) {  
    // рекурсивная ветвь  
    if (n > 1) {  
        // рекурсивный вызов  
        return n * factorial(n - 1);  
    } else {  
        // терминальная ветвь  
        return 1;  
    }  
}
```

ЧИСЛА ФИБОНАЧЧИ С ПОМОЩЬЮ РЕКУРСИИ

Числа Фибоначчи или ряд Фибоначчи — последовательность натуральных чисел.

Каждое следующее число — это сумма двух предыдущих.

Пример вывода для первых 10 чисел:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55

Реализовать факториал можно с помощью рекурсии:

- 1) Осуществив рекурсивный вызов (необходимо наличие параллельной рекурсии)
- 2) Завершив выполнение терминальным условием

```
int fib(int n)
{
    // терминальная ветвь
    if (n <= 1){
        return n;
    // рекурсивная ветвь
    } else {
        // рекурсивный вызов
        // характерным есть наличие
        // параллельной рекурсии
        return fib(n-1) + fib(n-2);
    }
}
```

СПАСИБО ЗА ВНИМАНИЕ!

