Рекурсия

Язык Java поддерживает рекурсию. Рекурсия в программировании - это когда метод вызывает сам себя. В таком случае метод называют рекурсивным.

Рекурсивный метод:

static int factorial(int x){

    if (x == 1){

        return 1;

    }

    return x \* factorial(x - 1);

}

Вначале проверяется условие: если вводимое число не равно 1, то мы умножаем данное число на результат этой же функции, в которую в качестве параметра передается число x-1. То есть происходит рекурсивный спуск. И так дальше, пока не дойдем того момента, когда значение параметра не будет равно единице.

Рекурсивная функция обязательно должна иметь некоторый базовый вариант, который использует оператор return и который помещается в начале функции. В случае с факториалом это if (x == 0) return 1;. И все рекурсивные вызовы должны обращаться к подфункциям, которые в конечном счете сходятся к базовому варианту. Так, при передаче в функцию положительного числа при дальнейших рекурсивных вызовах подфункций в них будет передаваться каждый раз число, меньшее на единицу. И в конце концов мы дойдем до ситуации, когда число будет равно 0, и будет использован базовый вариант.

Когда метод вызывает сам себя, новым локальным переменным и параметром выделяется место в стеке и код метода выполняется с этими новыми начальными значениями. При каждом возврате из рекурсивного вызова старые локальные переменные и параметры удаляются из стека, и выполнение продолжается с момента вызова внутри метода.

Следует помнить, что рекурсивные методы требуют больше ресурсов для выполнения и даже может вызвать переполнение памяти при слишком больших значениях.

Рекурсивные методы часто используют в сортировке, а также в алгоритмах, связанных с искусственным интеллектом. В обычной практике рекурсия используется редко.

При использовании рекурсивных методов нужно смотреть, чтобы в программе был оператор if для выхода из рекурсивного метода без выполнения рекурсивного вызова. Иначе метод никогда не выполнит возврат.

Рассмотрим ещё один пример вывода первых элементов массива. Создадим отдельный класс с рекурсивным методом:

class Recursion {

int aValues[];

StringBuilder sb = new StringBuilder();

// Конструктор

Recursion(int i) {

aValues = new int[i];

}

// Рекурсивное отображение элементов массива

String printArray(int i) {

if (i == 0)

return ""; // не забываем про выход из метода

else

printArray(i - 1);

String output = "[" + (i - 1) + "] " + aValues[i - 1] + "\n";

sb.append(output);

return sb.toString();

}

}

public void onClick(View v) {

Recursion recursion = new Recursion(5);

int i;

for (i = 0; i < 5; i++)

recursion.aValues[i] = i;

textInfo.append(recursion.printArray(5));

}

Задачи

1. Дано натуральное число n. Выведите все числа от 1 до n.
2. Напишите метод, который проверяет, входит ли в массив заданный элемент или нет. Используйте перебор и двоичный поиск для решения этой задачи. Сравните время выполнения обоих решений для больших массивов (например, 100000000 элементов).
3. Найдите корень уравнения на отрезке [0;10] с точностью по x не хуже 0.001. Известно, что на этом промежутке корень единственный. Используйте для этого метод деления отрезка пополам (и рекурсию).
4. Реализуйте бинарное дерево поиска