|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

Отчет по выполнению практической работы 5

**Тема. ПРОГРАММИРОВАНИЕ РЕКУРСИИ В JAVA**

Дисциплина Программирование на языке Джава

Выполнил

|  |  |
| --- | --- |
| студент | Болотов Михаил |
|  | Фамилия И.О. |
| группа | ИКБО-06-19 |
|  | Номер группы |

Москва 2020

**Содержание**

[​ Теория 2](#__RefHeading___Toc4128_811265792)

[​ Задание 2](#__RefHeading___Toc4136_811265792)

[​ Код 3](#__RefHeading___Toc35859_811265792)

[​ T7.java 3](#__RefHeading___Toc42585_811265792)

[​ T8.java 4](#__RefHeading___Toc42587_811265792)

[​ T9.java 4](#__RefHeading___Toc42589_811265792)

[​ T10.java 4](#__RefHeading___Toc42591_811265792)

[​ T11.java 5](#__RefHeading___Toc42593_811265792)

[​ Main.java 5](#__RefHeading___Toc42595_811265792)

[​ Скриншот 6](#__RefHeading___Toc3748_811265792)

[​ Заключение 6](#__RefHeading___Toc3750_811265792)

[​ Библиографический список 6](#__RefHeading___Toc3752_811265792)

# Теория

В контексте языка программирования рекурсия — это некий активный метод (или подпрограмма) вызываемый сам по себе непосредственно, или вызываемой другим методом (или подпрограммой) косвенно. В первую очередь надо понимать, что рекурсия — это своего рода перебор. Вообще говоря, всё то, что решается итеративно можно решить рекурсивно, то есть с использованием рекурсивной функции.

Так же, как и у перебора (цикла) у рекурсии должно быть условие остановки — базовый случай (иначе также, как и цикл, рекурсия будет работать вечно — infinite). Это условие и является тем случаем, к которому рекурсия идет (шаг рекурсии). При каждом шаге вызывается рекурсивная функция до тех пор, пока при следующем вызове не сработает базовое условие и не произойдет остановка рекурсии (а точнее возврат к последнему вызову функции). Всё решение сводится к поиску решения для базового случая. В случае, когда рекурсивная функция вызывается для решения сложной задачи (не базового случая) выполняется некоторое количество рекурсивных вызовов или шагов, с целью сведения задачи к более простой. И так до тех пор, пока не получим базовое решение.

Итак, рекурсивная функция состоит из:

* условие остановки или же базового случая или условия;
* условие продолжения или шага рекурсии — способ сведения сложной задачи к более простым подзадачам.

# Задание

7. Разложение на множители

Дано натуральное число n>1. Выведите все простые множители этого числа в порядке не убывания с учетом кратности. Алгоритм должен иметь сложность O(logn)

8. Палиндром

Дано слово, состоящее только из строчных латинских букв. Проверьте, является ли это слово палиндромом. Выведите YES или NO. При решении этой задачи нельзя пользоваться циклами, в решениях на питоне нельзя использовать срезы с шагом, отличным от 1.

9. Без двух нулей

Даны числа a и b. Определите, сколько существует последовательностей из a нулей и b единиц, в которых никакие два нуля не стоят рядом.

10. Разворот числа

Дано число n, десятичная запись которого не содержит нулей. Получите число, записанное теми же цифрами, но в противоположном порядке.

При решении этой задачи нельзя использовать циклы, строки, списки, массивы, разрешается только рекурсия и целочисленная арифметика.

Функция должна возвращать целое число, являющееся результатом работы программы, выводить число по одной цифре нельзя.

11. Количество единиц

Дана последовательность натуральных чисел (одно число в строке), завершающаяся двумя числами 0 подряд. Определите, сколько раз в этой последовательности встречается число 1. Числа, идущие после двух нулей, необходимо игнорировать. В этой задаче нельзя использовать глобальные переменные и параметры, передаваемые в функцию.

Функция получает данные, считывая их с клавиатуры, а не получая их в виде параметров.

# Код

### T7.java

package dev.ky3he4ik.pr.pr5;  
  
import java.util.Scanner;  
  
public class T7 {  
  
 private static void findMult(int n, int i) {  
 if (i > n / 2) {  
 System.out.println(n);  
 return;  
 }  
 if (n % i == 0) {  
 System.out.println(i);  
 findMult(n / i, i);  
 } else  
 findMult(n, i + 1);  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 findMult(scanner.nextInt(), 2);  
 }  
}

### T8.java

package dev.ky3he4ik.pr.pr5;  
  
import java.util.Scanner;  
  
public class T8 {  
 private static boolean isPalindrom(String s, int shift) {  
 if (shift >= s.length() / 2)  
 return true;  
 return s.charAt(shift) == s.charAt(s.length() - 1 - shift) && isPalindrom(s, shift + 1);  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 System.out.println(isPalindrom(scanner.nextLine(), 0) ? "YES" : "NO");  
 }  
}

### T9.java

package dev.ky3he4ik.pr.pr5;  
  
import java.util.Scanner;  
  
public class T9 {  
 private static int cntSeq(int a, int b) {  
 if (a > b + 1)  
 return 0;  
 if (a == 0 || b == 0)  
 return 1;  
 return cntSeq(a, b - 1) + cntSeq(a - 1, b - 1);  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 System.out.println(cntSeq(scanner.nextInt(), scanner.nextInt()));  
 }  
}

### T10.java

package dev.ky3he4ik.pr.pr5;  
  
import java.util.Scanner;  
  
public class T10 {  
 private static int reverse(int n, int t) {  
 if (n == 0)  
 return t;  
 return reverse(n / 10, t \* 10 + n % 10);  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 System.out.println(reverse(scanner.nextInt(), 0));  
 }  
}

### T11.java

package dev.ky3he4ik.pr.pr5;  
  
import java.util.Scanner;  
  
public class T11 {  
 private static int oneCount() {  
 Scanner in = new Scanner(System.in);  
 int n = in.nextInt();  
 if (n == 0) {  
 n = in.nextInt();  
 if (n == 0)  
 return 0;  
 else  
 return oneCount() + (n == 1 ? 1 : 0);  
 } else  
 return oneCount() + (n == 1 ? 1 : 0);  
 }  
  
 public static void main(String[] args) {  
 System.out.println(oneCount());  
 }  
}

### Main.java

package dev.ky3he4ik.pr.pr5;

public class Main {

public static void main(String[] args) {

T7.main(args);

T8.main(args);

T9.main(args);

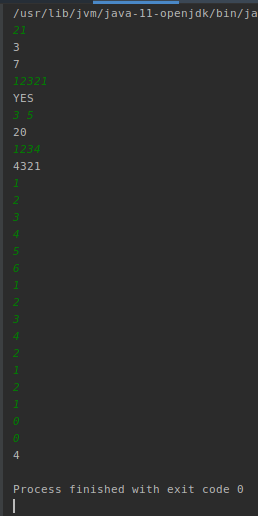
T10.main(args);

T11.main(args);

}

}

# Скриншот



# Заключение

В данной практической работе я освоил на практике разработку и программирование рекурсивных алгоритмов на языке Java.

# Библиографический список

1. Зорина Н.В. Курс лекций по Объектно-ориентированному программированию на Java, МИРЭА, Москва, 2016
2. Программирование на языке Java: работа со строками и массивами. Методические указания. [Электронный ресурс] : Учебно-методические пособия — Электрон. дан. — СПб. : ПГУПС, 2015. — 24 с.
3. Кожомбердиева, Г.И. Программирование на языке Java: создание графического интерфейса пользователя: учеб. пособие. [Электронный ресурс] : Учебные пособия / Г.И. Кожомбердиева, М.И. Гарина. — Электрон. дан. — СПб.: ПГУПС, 2012. — 67 с.
4. Вишневская, Т.И. Технология программирования. Часть 1. [Электронный ресурс] / Т.И. Вишневская, Т.Н. Романова. — Электрон. дан. — М. : МГТУ им. Н.Э. Баумана, 2007. — 59 с.