МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра инструментального и прикладного программного обеспечения (ИиППО)

Дисциплина «Программирование на языке Джава»

ОТЧЕТ
ПО ПРАКТИЧЕСКОМУ ЗАНЯТИЮ №5

Выполнил студент группы ИНБО-02-20 Лукьяненко Д.В.

Принял Степанов П.В.

Практическая работа выполнена «___»_____2021 г.

«_____» «___»_____2021 г.
Отметка о выполнении

Москва – 2021 г.

СОДЕРЖАНИЕ

Цель работы

Цель данной практической работы – изучение работы с рекурсией.

Репозиторий

Ссылка: https://github.com/neluckoff/mirea-java-lessons/tree/master/src/ru/luckoff/mirea/practice_5

Задание

Задания на рекурсию

1. Треугольная последовательность

Дана монотонная последовательность, в которой каждое натуральное число k встречается ровно k раз: 1, 2, 2, 3, 3, 3, 4, 4, 4, 4,…

По данному натуральному n выведите первые n членов этой последовательности. Попробуйте обойтись только одним циклом for.

2. От 1 до n

Дано натуральное число n. Выведите все числа от 1 до n.

3. От A до B

Даны два целых числа A и B (каждое в отдельной строке). Выведите все числа от A до B включительно, в порядке возрастания, если A < B, или в порядке убывания в противном случае.

4. Заданная сумма цифр

Даны натуральные числа k и s. Определите, сколько существует k-значных натуральных чисел, сумма цифр которых равна d. Запись натурального числа не может начинаться с цифры 0.

В этой задаче можно использовать цикл для перебора всех цифр, стоящих на какой-либо позиции.

5. Сумма цифр числа

Дано натуральное число N. Вычислите сумму его цифр.

При решении этой задачи нельзя использовать строки, списки, массивы (ну и циклы, разумеется).

6. Проверка числа на простоту

Дано натуральное число n>1. Проверьте, является ли оно простым. Программа должна вывести слово YES, если число простое и NO, если число составное. Алгоритм должен иметь сложность O(logn).

Указание. Понятно, что задача сама по себе нерекурсивна, т.к. проверка числа n на простоту никак не сводится к проверке на простоту меньших чисел. Поэтому нужно сделать еще один параметр рекурсии: делитель числа, и именно по этому параметру и делать рекурсию.

7. Разложение на множители

Дано натуральное число n>1. Выведите все простые множители этого числа в порядке неубывания 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.

12. Вывести нечетные числа последовательности

Дана последовательность натуральных чисел (одно число в строке), завершающаяся числом 0. Выведите все нечетные числа из этой последовательности, сохраняя их порядок.

В этой задаче нельзя использовать глобальные переменные и передавать какие-либо параметры в рекурсивную функцию. Функция получает данные, считывая их с клавиатуры. Функция не возвращает значение, а сразу же выводит результат на экран. Основная программа должна состоять только из вызова этой функции.

13. Вывести члены последовательности с нечетными номерами

Дана последовательность натуральных чисел (одно число в строке), завершающаяся числом 0. Выведите первое, третье, пятое и т.д. из введенных чисел. Завершающий ноль выводить не надо.

В этой задаче нельзя использовать глобальные переменные и передавать какиелибо параметры в рекурсивную функцию. Функция получает данные, считывая их с клавиатуры. Функция не возвращает значение, а сразу же выводит

результат на экран. Основная программа должна состоять только из вызова этой функции.

14. Цифры числа слева направо

Дано натуральное число N. Выведите все его цифры по одной, в обычном порядке, разделяя их пробелами или новыми строками.

При решении этой задачи нельзя использовать строки, списки, массивы (ну и циклы, разумеется). Разрешена только рекурсия и целочисленная арифметика

15. Цифры числа справа налево

Дано натуральное число N. Выведите все его цифры по одной, в обратном порядке, разделяя их пробелами или новыми строками.

При решении этой задачи нельзя использовать строки, списки, массивы (ну и циклы, разумеется). Разрешена только рекурсия и целочисленная арифметика.

16. Количество элементов, равных максимуму

Дана последовательность натуральных чисел (одно число в строке), завершающаяся числом 0. Определите, какое количество элементов этой последовательности, равны ее наибольшему элементу.

В этой задаче нельзя использовать глобальные переменные. Функция получает данные, считывая их с клавиатуры, а не получая их в виде параметра. В программе на языке Python функция возвращает результат в виде кортежа из нескольких чисел и функция вообще не получает никаких параметров. В программе на языке C++ результат записывается в переменные, которые передаются в функцию по ссылке. Других параметров, кроме как используемых для возврата значения, функция не получает.

Гарантируется, что последовательность содержит хотя бы одно число (кроме нуля)

17. Максимум последовательности

Дана последовательность натуральных чисел (одно число в строке), завершающаяся числом 0. Определите наибольшее значение числа в этой последовательности.

В этой задаче нельзя использовать глобальные переменные и передавать какиелибо параметры в рекурсивную функцию. Функция получает данные, считывая их с клавиатуры. Функция возвращает единственное значение: максимум считанной последовательности. Гарантируется, что последовательность содержит хотя бы одно число (кроме нуля).

Выполнение работы

Мною были сделаны классы Main и Recursion, в одном я реализовал метод для каждой рекурсии, а в другом запуск программы с меню. Чтобы сильно не нагружать программу, я реализовал небольшое меню через switch case, чтобы после ввода номера рекурсии, вызывался номер с ней.

Весь код, результаты его работы и UML Диаграмма будут показаны на рисунках ниже.

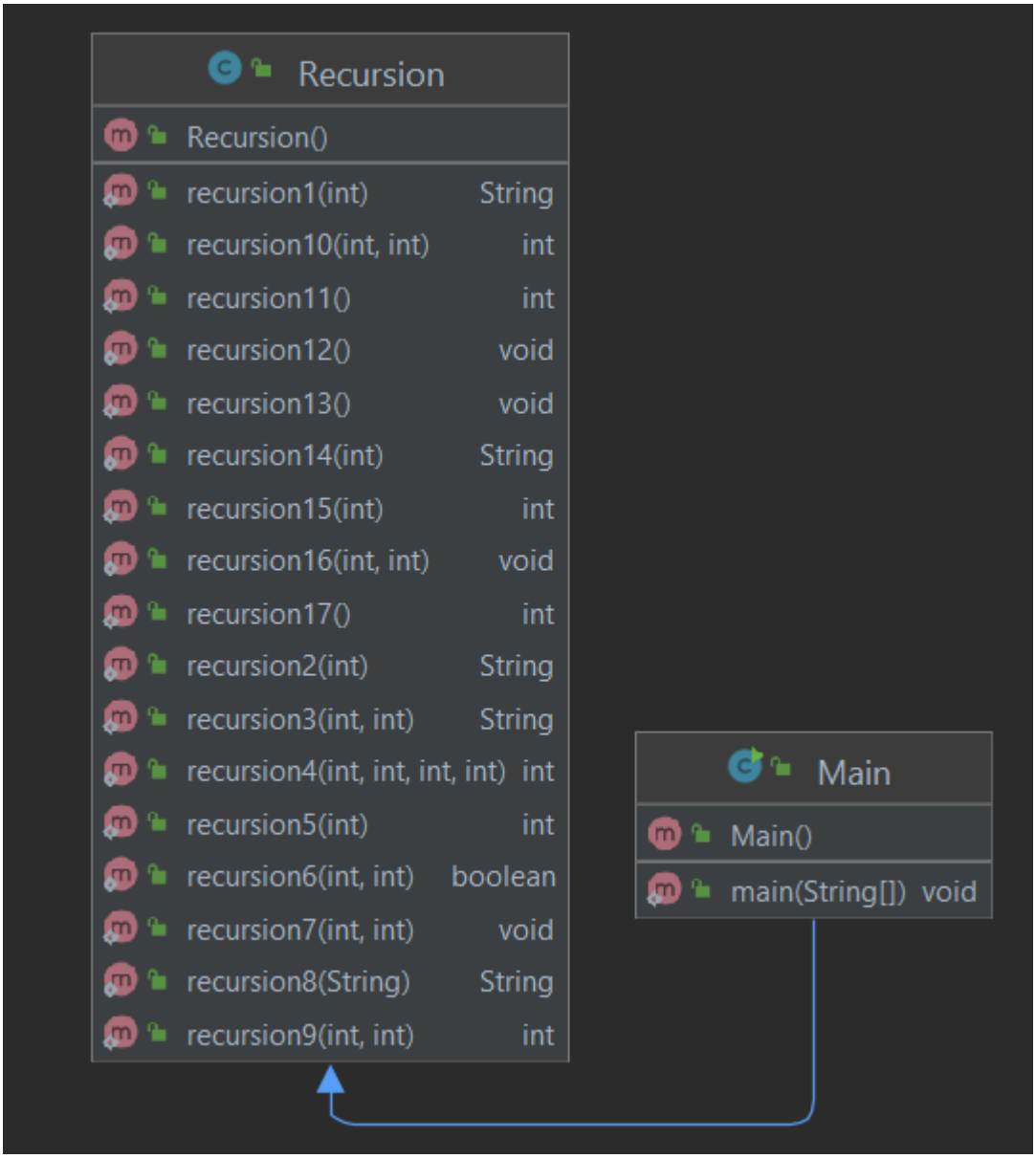

Рисунок 1 – Полученная UML Диаграмма

Код выполненной работы

Здесь в нескольких скриншотах можно увидеть как выглядит код выполненного задания и результат его работы.

```
public static String recursion1(int n) {
    int s = 0;
    int j = 0;
    if (n == 1) {
        System.out.print("1");
    } else {
        for (int i = 1; s < n; i++) {
            s += i;
            j = i;
        }
        System.out.print(recursion1(--n) + " " + j);
    }
    return "";
}
```

Рисунок 2 – Треугольная последовательность

```
public static String recursion2(int n) {
    if (n == 1) {
        return "1";
    }
    return recursion2( n: n - 1) + " " + n;
}
```

Рисунок 3 – От 1 до n

```java
public static String recursion3(int a, int b) {
    if (a > b) {
        if (a == b) {
            return Integer.toString(a);
        }
        return a + " " + recursion3( a: a - 1, b);
    } else {
        if (a == b) {
            return Integer.toString(a);
        }
        return a + " " + recursion3( a: a + 1, b);
    }
}
```

Рисунок 4 – От A до B

```java
public static int recursion4(int len, int sum, int k, int s) {
    if (len == k) {
        if (sum == s) {
            return 1;
        } else {
            return 0;
        }
    }
    int c = (len == 0 ? 1 : 0);
    int res = 0;
    for (int i = c; i < 10; i++) {
        res += recursion4( len: len + 1,  sum: sum + i, k, s);
    }
    return res;
}
```

Рисунок 5 – Заданная сумма цифр

```java
public static int recursion5(int n) {
    if (n < 10) {
        return n;
    }
    else {
        return n % 10 + recursion5( n: n / 10);
    }
}
```

Рисунок 6 – Сумма цифр числа

```java
public static boolean recursion6(int n, int i) {
    if (n < 2) {
        return false;
    }
    else if (n == 2) {
        return true;
    }
    else if (n % i == 0) {
        return false;
    }
    else if (i < n / 2) {
        return recursion6(n,  i: i + 1);
    } else {
        return true;
    }
}
```

Рисунок 7 – Проверка числа на простоту

```java
public static void recursion7(int n, int k) {
    if (k > n / 2) {
        System.out.println(n);
        return;
    }
    if (n % k == 0) {
        System.out.println(k);
        recursion7( n: n / k, k);
    }
    else {
        recursion7(n, ++k);
    }
}
```

Рисунок 8 – Разложение на множители

```java
public static String recursion8(String s) {
    if (s.length() == 1) {
        return "YES";
    } else {
        if (s.substring(0, 1).equals(s.substring(s.length() - 1, s.length()))) {
            if (s.length() == 2) {
                return "YES";
            }
            return recursion8(s.substring(1, s.length() - 1));
        } else {
            return "NO";
        }
    }
}
```

Рисунок 9 – Палиндром

```java
public static int recursion9(int a, int b) {
    if (a > b + 1) {
        return 0;
    }
    if (a == 0 || b == 0) {
        return 1;
    }
    return recursion9(a,  b: b - 1) + recursion9( a: a - 1,  b: b - 1);
}
```

Рисунок 10 – Без двух нулей

```java
public static int recursion10(int n, int i) {
    return (n==0) ? i : recursion10( n: n/10,  i: i*10 + n%10 );
}
```

Рисунок 11 – Разворот числа

```java
public static int recursion11() {
    Scanner in = new Scanner(System.in);
    int n = in.nextInt();
    if (n == 1) {
        int m = in.nextInt();
        if (m == 1) {
            return recursion11() + n + m;
        } else {
            int k = in.nextInt();
            if (k == 1) {
                return recursion11() + n + m + k;
            } else {
                return n + m + k;
            }
        }
    } else {
        int m = in.nextInt();
        if (m == 1) {
            return recursion11() + n + m;
        } else {
            return n + m;
        }
    }
}
```

Рисунок 12 – Количество единиц

```java
public static void recursion12() {
    Scanner in = new Scanner(System.in);
    int n = in.nextInt();
    if (n > 0) {
        if (n % 2 == 1) {
            System.out.println(n);
            recursion12();
        } else {
            recursion12();
        }
    }
}
```

Рисунок 13 – Вывести нечетные числа последовательности

```java
public static void recursion13() {
    Scanner in = new Scanner(System.in);
    int n = in.nextInt();
    if (n > 0) {
        int m = in.nextInt();
        System.out.println(n);
        if (m > 0) {
            recursion13();
        }
    }
}
```

Рисунок 14 – Вывести члены последовательности с нечетными номерами

```java
public static String recursion14(int n) {
    if (n < 10) {
        return Integer.toString(n);
    }
    else {
        return recursion14( n: n / 10) + " " + n % 10;
    }
}
```

Рисунок 15 – Цифры числа слева направо

```java
public static int recursion15(int n) {
    if (n < 10) {
        return n;
    }
    else {
        System.out.print(n % 10 + " ");
        return recursion15( n: n / 10);
    }
}
```

Рисунок 16 – Цифры числа справа налево

```java
public static void recursion16(int max, int count) {
    Scanner in = new Scanner(System.in);
    int n = in.nextInt();
    if (n > 0) {
        if (n > max) {
            recursion16(n,  count: 1);
        }
        else if (n == max) {
            recursion16(max, ++count);
        }
        else {
            recursion16(max, count);
        }
    } else {
        System.out.println(count);
    }
}
```

Рисунок 17 – Количество элементов, равных максимуму

```java
public static int recursion17() {
    Scanner in = new Scanner(System.in);
    int n = in.nextInt();
    if (n == 0) {
        return 0;
    }
    else {
        return Math.max(n, recursion17());
    }
}
```

Рисунок 18 – Максимум последовательности

```java
public class Main extends Recursion {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        while(true) {
            System.out.print("Input a number: ");
            int num = in.nextInt();
            switch (num) {
                case 1:
                    System.out.println("Number 1");
                    System.out.println(recursion1( n: 5));
                    break;
                case 2:
                    System.out.println("Number 2 \n" + recursion2( n: 5));
                    break;
                case 3:
```

Рисунок 19 – Реализация меню в Main

Вывод

В результате выполнения данной практической работы я научился работать с огромным количеством разновидностей рекурсии.