

|  |  |
| --- | --- |
| **Министерство образования и науки**  **Российской Федерации**  **Государственное образовательное учреждение высшего образования «МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ**  **ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  **(МАДИ)»** |  |

**Кафедра «Высшая математика»**

**Отчет по дисциплине «Цифровая грамотность»**

**Лабораторная работа №1**

**«Стандарт представления чисел с плавающей точкой IEEE754»**

**Выполнил:**

Учебная группа 1бПМ\_\_\_\_\_\_\_\_\_  
ФИО Греча К. П. \_

**Принял:**

Должность Старший преподаватель

Звание \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ФИО \_ Кутейников И. А.\_\_\_\_\_

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023г.

**Цель:**Написать код для перевода чисел в нормализованный и денормализованный вид с использованием норм IEEE754.

**Алгоритм:**

import java.util.Scanner;  
  
public class Main  
{  
 private static void exponentNum(String inNum)// вывод числа в экспоненциальном виде в десятичной СС  
 {  
 boolean f = true;  
 String sign = Double.*parseDouble*(inNum) >= 0 ? "" : "-";  
 inNum = Double.*toString*(Math.*abs*(Double.*parseDouble*(inNum)));  
 boolean isNumBiggerThanOne = Float.*parseFloat*(inNum) >= 1;  
 String mantissa = "";  
  
 int count\_non\_significant\_symbols = 0;  
 for (int i = 0; i < inNum.length(); i++)  
 {  
 if (inNum.charAt(i) == '0')  
 count\_non\_significant\_symbols++;  
 else if (inNum.charAt(i) == '.') {  
 count\_non\_significant\_symbols++;  
 f = false;  
 }  
 else  
 break;  
 }  
  
 int start\_index = count\_non\_significant\_symbols;  
 int count = count\_non\_significant\_symbols > 0 ? count\_non\_significant\_symbols - 1 : -1;  
 for (int i = start\_index; i < inNum.length(); i++)  
 {  
 if (inNum.charAt(i) != '.')  
 mantissa += inNum.charAt(i);  
 else  
 f = false;  
  
 if (f && isNumBiggerThanOne)  
 count++;  
 else if (f) {  
 count--;  
 }  
 }  
  
 if (isNumBiggerThanOne)  
 {  
 System.*out*.println("Normalized: " + sign + mantissa.charAt(0) + "." + mantissa.substring(1) + "E+" + count);  
 System.*out*.println("Denormalized: " + sign + "0." + mantissa + "E+" + (count + 1));  
 }  
 else  
 {  
 System.*out*.println("Normalized: " + sign + mantissa.charAt(0) + "." + mantissa.substring(1) + "E-" + count);  
 System.*out*.println("Denormalized: " + sign + "0." + mantissa + "E-" + (count - 1));  
 }  
 }  
  
 //Метод Binary для перевода дробной части числа в двоичную систему счисления:  
 static String Binary(double fract, int integer\_len){  
 String fract\_bin = "";  
 int i = 0;  
 while (Math.*floor*(fract) != fract)  
 {  
  
 if (i == (23 - integer\_len))// нужно чтобы мантисса была не больше 23 бит  
 break;  
  
 // когда мы переводили целые числа в двоичную СС то делили на 2  
 // и брали остатки и записывали в обратном порядке  
 // здесь мы умножаем на 2 и берем целую часть от получившегося числа  
 // до тех пор, пока у нас не получится число у которого дробная часть == .0  
 fract\_bin += (int) Math.*floor*(fract \* 2) % 2;  
 fract \*= 2;  
 i++;  
 }  
  
 if (fract\_bin.isEmpty())  
 fract\_bin = "0";  
  
 return fract\_bin;  
 }  
  
  
 private static int findExponenta(String binary\_num)// нахождение экспоненты  
 {  
 int count = -1;  
 for (int i = 0; i < binary\_num.length(); i++)  
 {  
 if (binary\_num.charAt(i) != '.')  
 count++;  
 else  
 return count;  
 }  
 return 0;  
 }  
  
  
 public static String IEEE754(double num)// метод конвертации в IEEE754(binary32)  
 {  
 // считываем знак числа и продолжаем вычисления уже для модуля числа  
 String sign = num > 0 ? "0" : "1";  
 num = Math.*abs*(num);  
  
 int integer = (int) Math.*floor*(num); // целая часть числа  
 int exp = 0;  
 String moved\_exp = "";  
 double fractional = num - integer; // дробная часть числа  
  
 String integer\_bin = Integer.*toBinaryString*(integer); // целая часть числа в 2-ой сс  
 String fract\_bin = *Binary*(fractional, integer\_bin.length()); // дробная часть числа в 2-ой сс  
  
 String mantissa = integer\_bin + fract\_bin; // мантисса  
 String binary\_num = integer\_bin + "." + fract\_bin; // число в 2-ой сс  
  
 exp = *findExponenta*(binary\_num);  
 moved\_exp = Integer.*toBinaryString*(exp + 127);  
  
 String binaryStr = sign + moved\_exp + mantissa.substring(1); // само число в представлении IEEE754  
  
 // если длина binaryStr меньше 32 -> дополняем нулями до 32 символов строку  
 if (binaryStr.length() < 32)  
 {  
 for (int i = binaryStr.length(); i < 32; i++)  
 binaryStr += "0";  
 }  
  
 System.*out*.println("Binary: "+ binaryStr);  
 System.*out*.println("\t\t" + binary\_num);  
 return binaryStr;  
 }  
  
 public static void IEEE754\_hex(String num)  
 {  
 System.*out*.println("Hexadecimal(шестнадцатиричный): " + *HexString*(num));  
 }  
  
  
 private static String byteHex(String byteStr)//метод byteToHex для перевода 4-битных блоков в 16-ричную систему  
 {  
 int decimal = Integer.*parseInt*(byteStr, 2); // переводим из 2-ой в 10-ю  
 return Integer.*toHexString*(decimal); // возвращаем переведенное значение из 10-ой в 16-ю  
 }  
  
 private static String HexString(String binaryStr)//метод toHexString для конвертации двоичной строки в 16-ричную  
 {  
 String hexStr = "";  
 // берём каждые 4 бита (т.к. значение идет от 0000 (0) до 1111 (15)) и переводим их в 16-ую систему  
 for (int i = 0; i <= binaryStr.length() - 4; i+=4)  
 {  
 hexStr += *byteHex*(binaryStr.substring(i, i+4));  
 }  
 return hexStr;  
 }  
 public static void main(String[] args)  
 {  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.print("Enter number: ");  
 String inNum = in.next(); // ввод числа  
  
 *exponentNum*(inNum);  
 String out = *IEEE754*(Double.*parseDouble*(inNum));  
 *IEEE754\_hex*(out);  
 }  
}

**Результат:**

Enter number: **28489**

Normalized: **2.84890E+4**

Denormalized: **0.284890E+5**

Binary: **01000110110111101001001000000000**

**110111101001001.0**

Hexadecimal(шестнадцатиричный): **46de9200**

**Заключение:**В процессе выполнения лабораторной работы мы научились переводить числа в нормализованный и денормализованный вид с использованием IEEE754.