Министерство науки и высшего образования РФ

ФГАОУ ВО Пермский национальный исследовательский  
политехнический университет

Кафедра «Вычислительная математика, механика и биомеханика»

Отчёт по лабораторной работе № 2  
тема «Ветвящиеся алгоритмы»  
по дисциплине «Информатика»

Выполнил: студентка группы ПМ-23-1б Наговицына В. А.

Проверил: ст. пр. каф. ВММБ Ильиных Г.В.

Пермь, 2023

Оглавление

[Задание 1 4](#_Toc147913845)

[1.1 Постановка задачи 4](#_Toc147913846)

[1.2 Алгоритм решения 4](#_Toc147913847)

[1.3 Текст метода решения задачи 4](#_Toc147913848)

[1.4 Тестирование программы 5](#_Toc147913849)

[Задание 2 5](#_Toc147913850)

[2.1 Постановка задачи 5](#_Toc147913851)

[2.2 Алгоритм решения 6](#_Toc147913852)

[2.3 Текст метода решения задачи 6](#_Toc147913853)

[2.4 Тестирование программы 7](#_Toc147913854)

[Задание 3 7](#_Toc147913855)

[3.1. Постановка задачи 7](#_Toc147913856)

[3.2. Алгоритм решения 7](#_Toc147913857)

[3.3. Текст метода решения задачи 8](#_Toc147913858)

[3.4. Тестирование программы 8](#_Toc147913859)

[Задание 4 8](#_Toc147913860)

[4.1. Постановка задачи 8](#_Toc147913861)

[4.2. Алгоритм решения 9](#_Toc147913862)

[4.3. Текст метода решения задачи 9](#_Toc147913863)

[4.4. Тестирование программы 9](#_Toc147913864)

[Задание 5 9](#_Toc147913865)

[5.1. Постановка задачи 9](#_Toc147913866)

[5.2. Алгоритм решения 10](#_Toc147913867)

[5.3. Текст метода решения задачи 10](#_Toc147913868)

[5.4. Тестирование программы 10](#_Toc147913869)

[Задание 6 10](#_Toc147913870)

[6.1. Постановка задачи 10](#_Toc147913871)

[6.2. Алгоритм решения 11](#_Toc147913872)

[6.3. Текст метода решения задачи 11](#_Toc147913873)

[6.4. Тестирование программы 12](#_Toc147913874)

[Задание 7 12](#_Toc147913875)

[7.1. Постановка задачи 12](#_Toc147913876)

[7.2. Алгоритм решения 13](#_Toc147913877)

[7.3. Текст метода решения задачи 13](#_Toc147913878)

[7.4. Тестирование программы 14](#_Toc147913879)

[Задание 8 14](#_Toc147913880)

[8.1. Постановка задачи 14](#_Toc147913881)

[8.2. Алгоритм решения 15](#_Toc147913882)

[8.3. Текст метода решения задачи 15](#_Toc147913883)

[8.4. Тестирование программы 16](#_Toc147913884)

[Задание 9 16](#_Toc147913885)

[9.1. Постановка задачи 16](#_Toc147913886)

[9.2. Алгоритм решения 16](#_Toc147913887)

[9.3. Текст метода решения задачи 17](#_Toc147913888)

[9.4. Тестирование программы 17](#_Toc147913889)

# Задание 1

## 1.1 Постановка задачи

## Написать функцию, которая определяет, оканчивается ли число на чётную цифру. Написать функцию, которая определяет, является ли число квадратом целого числа. Создать программу, которая в бесконечном цикле получает от пользователя числа и проверяет их с помощью двух функций.

## 1.2 Алгоритм решения

## 1.3 Текст метода решения задачи

def task1():  
 while True:  
 k = 0  
 num = int(input('Введите число\n'))  
 if int(str(num)[-1]) % 2 == 0:  
 print('Число оканчивается на чётную цифру')  
 else:  
 print('Число оканчивается на нечётную цифру')  
 for sqr in range(1, 999):  
 if sqr \*\* 2 == num:  
 k += 1  
 if k > 0:  
 print('Число является квадратом целого числа')  
 elif k == 0:  
 print('Число не является квадратом целого числа')

## 1.4 Тестирование программы

Тестирование программы представлено на рисунке 1.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| а | б | в |

Рис. 1. Выполнение программы:  
а – первый запуск, б – второй запуск, в – третий запуск.

# Задание 2

## Постановка задачи

По формуле вычислить значения функции f (x) в диапазоне значений x ∈ [a, b] и представить в виде графика:

## Алгоритм решения

## Текст метода решения задачи

def task2\_fucntion(x):  
 if x < 3:  
 return (x \*\* 3) - (3 \* x) + 9  
 else:  
 return 2 \* (x \*\* 2) + (5 \* x) - 6  
  
  
def task2():  
 a = int(input("введите а\n"))  
 b = int(input("введите b\n"))  
  
 x\_values = list(range(a, b + 1))  
 y\_values = [task2\_fucntion(x) for x in range(a, b + 1)]  
  
 plt.plot(x\_values, y\_values)  
 plt.show()

## Тестирование программы

Тестирование программы представлено на рисунке 2.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| а | б | в |

Рис. 2. Выполнение программы:  
а – первый запуск, б – второй запуск, в – третий запуск

# Задание 3

## 3.1. Постановка задачи

Написать функцию decimnal\_in\_new\_numeral\_system(number,base), которая переводит десятичную дробь из десятичной системы счисления в систему счисления с выбранным основанием. Написать программу, которая переводит заданную пользователем дробь в заданную пользователем систему счисления.

## 3.2. Алгоритм решения

Алгоритм решения задачи представлен в виде блок-схемы на рисунке 5.

## 3.3. Текст метода решения задачи

def task3():  
 def decimal\_in\_new\_numeral\_system(number, base):  
 result = ''  
 integer\_part = int(number) # целая часть числа  
 fraction\_part = number - integer\_part # дробная часть числа  
  
 # переводим целую часть числа в новую систему счисления  
 while integer\_part > 0:  
 remainder = integer\_part % base  
 result = str(remainder) + result  
 integer\_part = integer\_part // base  
  
 # разделитель для дробной части  
 result += '.'  
  
 # переводим дробную часть числа в новую систему счисления  
 precision = 10 # точность перевода (до 10 знаков)  
 while fraction\_part > 0 and precision > 0:  
 fraction\_part \*= base  
 int\_part = int(fraction\_part)  
 result += str(int\_part)  
 fraction\_part -= int\_part  
 precision -= 1  
  
 return result  
  
 decimal\_number = float(input("Введите десятичное число: "))  
 new\_base = int(input("Введите основание новой системы счисления: "))  
  
 result = decimal\_in\_new\_numeral\_system(decimal\_number, new\_base)  
 print(f"Результат перевода: {result} в систему счисления с основанием {new\_base}")

## 3.4. Тестирование программы

Тестирование программы представлено на рисунке 3.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| а | б | в |

Рис. 3. Выполнение программы:  
а – первый запуск, б – второй запуск, в – третий запуск.

# Задание 4

## 4.1. Постановка задачи

Даны две фигуры, организовать программу, которая даёт ответ, попала ли произвольно введённая точка в одну из фигур и если попала, то в какую? Графически отобразить точку и области. Алгоритм попадания точки в область заключается в следующем:

1. разложение фигур на составляющие их контур линии (прямые или дуги окружности)

2. составление уравнений для каждой линии

3. сравнение координат точки (x, y) с проекциями этой точки на каждую линию из контура. Если точка лежит внутри области, значит, она лежит выше всех нижних линий, ниже верхних, правее левых, левее правых. Если фигура невыпуклая, то её надо разбить на выпуклые фигуры (добавить ещё уравнение линии, разделяющей фигуры).

## 4.2. Алгоритм решения

## 4.3. Текст метода решения задачи

def task4():  
 figure\_1 = 0  
 figure\_2 = 0  
  
 # рисуем фигуры  
  
 def drawLines(axes):  
 line1 = Line2D([-1, 0], [3, 5], color="k")  
 line2 = Line2D([0, 2], [5, 4], color="k")  
 line3 = Line2D([2, 3], [4, 1], color="k")  
 line4 = Line2D([3, -2], [1, -3], color="k")  
 line5 = Line2D([-2, -1], [-3, -1], color="k")  
 line6 = Line2D([-3, 1], [1, 1], color="k")  
 axes.add\_line(line1)  
 axes.add\_line(line2)  
 axes.add\_line(line3)  
 axes.add\_line(line4)  
 axes.add\_line(line5)  
 axes.add\_line(line6)  
  
 line7 = Line2D([2, 4], [-1, 1], color="k")  
 line8 = Line2D([4, 5], [-3, -4], color="k")  
 line9 = Line2D([5, 7], [-4, -3], color="k")  
 line10 = Line2D([7, 6], [-3, -1], color="k")  
 axes.add\_line(line7)  
 axes.add\_line(line8)  
 axes.add\_line(line9)  
 axes.add\_line(line10)  
  
 def drawArc(axes):  
 arc1 = matplotlib.patches.Arc((-1, 1), 4, 4, theta1=90, theta2=180)  
 arc2 = matplotlib.patches.Arc((-1, 1), 4, 4, theta1=270, theta2=360)  
 arc3 = matplotlib.patches.Arc((4, -1), 4, 4, theta1=0, theta2=90)  
 arc4 = matplotlib.patches.Arc((4, -1), 4, 4, theta1=180, theta2=270)  
 axes.add\_patch(arc1)  
 axes.add\_patch(arc2)  
 axes.add\_patch(arc3)  
 axes.add\_patch(arc4)  
 plt.text(0.6, -0.3, "")  
  
 # определим, попадает ли точка в фигуру, и в какую  
  
 def f1(x, y):  
 if 2 \* y + x > 10:  
 return False  
 elif y - 2 \* x > 5:  
 return False  
 elif (x - 1) \*\* 2 + (y - 1) \*\* 2 > 4:  
 return False  
 else:  
 return True  
  
 def f2(x, y):  
 if (x - 1) \*\* 2 + (y - 1) \*\* 2 < 4:  
 return False  
 elif y - 0.8 \* x < -1.4:  
 return False  
 elif y - 2 \* x > 1:  
 return False  
 else:  
 return True  
  
 def f3(x, y):  
 if (x - 4) \*\* 2 + (y + 1) \*\* 2 > 4:  
 return False  
 elif y + x < 1:  
 return False  
 elif y - x > -3:  
 return False  
 elif y - 0.5 \* x < -6.5:  
 return False  
 elif y + 2 \* x > 11:  
 return False  
 else:  
 return True  
  
 def f4(x, y):  
 if y - 0.8 \* x < -1.4:  
 return False  
 elif y + 3 \* y > 10:  
 return False  
 else:  
 return True  
  
 def f5(x, y):  
 if (x + 4) \*\* 2 + (y - 1) \*\* 2 > 4:  
 return False  
 elif y - x > -3:  
 return False  
 else:  
 return True  
  
 if \_\_name\_\_ == "\_\_main\_\_":  
 x = int(input('введите x \n'))  
 y = int(input('введите y \n'))  
  
 plt.xlim(-4, 8)  
 plt.ylim(-5, 6)  
 plt.grid()  
  
 axes = plt.gca()  
 axes.set\_aspect("equal")  
  
 drawLines(axes)  
 drawArc(axes)  
  
 plt.scatter(x, y, color='red')  
  
 plt.show()  
  
 if -4 < x <= 2 and y >= 1:  
 if f1:  
 figure\_1 += 1  
 elif -4 < x <= 2 and y < 1:  
 if f2:  
 figure\_1 += 1  
 elif 2 <= x < 4:  
 if f4:  
 figure\_1 += 1  
 elif f5:  
 figure\_2 += 1  
 elif x >= 4:  
 if f3:  
 figure\_2 += 1  
  
 if figure\_1 > 0:  
 print('Точка входит в первую фигуру')  
 elif figure\_2 > 0:  
 print('Точка входит во вторую фигуру')  
 else:  
 print('Точка не входит ни в одну фигуру')

## 4.4. Тестирование программы

Тестирование программы представлено на рисунке 4.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| а | б | в |

Рис. 4. Выполнение программы:  
а – первый запуск, б – второй запуск, в – третий запуск.

# Задание 5

## 5.1. Постановка задачи

## Написать программу, которая выводит на экран нечётные числа, сумма цифр которых больше заданного значения m. Изначально задается три трёхзначных числа.

## 5.2. Алгоритм решения

## 5.3. Текст метода решения задачи

def task5():  
 try:  
 num1 = int(input("Введите первое трехзначное число: "))  
 num2 = int(input("Введите второе трехзначное число: "))  
 num3 = int(input("Введите третье трехзначное число: "))  
  
 m = int(input("Введите число m: "))  
 numbers = []  
  
 if not (100 <= num1 <= 999 and 100 <= num2 <= 999 and 100 <= num3 <= 999):  
 print('Одно или несколько чисел не являются трёхзначными.')  
 else:  
 if num1 % 2 != 0 and ((num1 % 10) + ((num1 // 10) % 10) + (num1 // 100) > m):  
 numbers.append(num1)  
 if num2 % 2 != 0 and ((num2 % 10) + ((num2 // 10) % 10) + (num2 // 100) > m):  
 numbers.append(num2)  
 if num3 % 2 != 0 and ((num3 % 10) + ((num3 // 10) % 10) + (num3 // 100) > m):  
 numbers.append(num3)  
  
 if len(numbers) == 0:  
 print("Ни одно из введенных чисел не удовлетворяет условиям.")  
 else:  
 print("Нечетные числа, сумма цифр которых больше", m, ": ", numbers)  
  
 except ValueError:  
 print("Ошибка ввода. Введите трехзначные числа и значение суммы цифр целыми числами.")

## 5.4. Тестирование программы

Тестирование программы представлено на рисунке 5.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| а | б | в |

Рис. 5. Выполнение программы:  
а – первый запуск, б – второй запуск, в – третий запуск.

# Задание 6

## 6.1. Постановка задачи

У студента межсессионная задолженность по информатике, осталось сдать labs лабораторных работ. Он приходит на consultations\_per\_month консультаций в месяц, при этом сдаёт лабораторную работу на консультации с attempts раза. Написать программу, которая с помощью цикла определяет количество месяцев до сдачи предмета. Учесть, что задаётся начальное количество попыток после каждой сданной лабораторной работы уменьшается на единицу, но не стать меньше 1.

## 6.2. Алгоритм решения

## 6.3. Текст метода решения задачи

def task6():  
 labs = int(input('введите labs \n'))  
 consultations\_per\_month = int(input('введите consultations\_per\_month \n'))  
 attempts = int(input('введите attempts \n'))  
 months = 0  
 while labs > 0 and attempts > 0:  
 if consultations\_per\_month <= attempts:  
 attempts -= consultations\_per\_month  
 labs -= 1  
 else:  
 attempts = 0  
 months += 1  
 if labs > 0:  
 print("Не хватает попыток для сдачи всех лабораторных работ")  
 else:  
 print(months)

## 6.4. Тестирование программы

Тестирование программы представлено на рисунке 6.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| а | б | в |

Рис. 6. Выполнение программы:  
а – первый запуск, б – второй запуск, в – третий запуск.

# Задание 8

## 8.1. Постановка задачи

## Дан ряд:

## S =

## Посчитать сумму первых n членов ряда

## 8.2. Алгоритм решения

## 8.3. Текст метода решения задачи

def task8():  
 n = int(input("введите n\n"))  
 x = float(input("введите x\n"))  
 sum = float(0)  
 for i in range(0, n + 1):  
 sum += ((x \*\* 2) + n)  
 print("сумма = {:.4f}".format(sum))

## 8.4. Тестирование программы

Тестирование программы представлено на рисунке 8.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| а | б | в |

Рис. 8. Выполнение программы:  
а – первый запуск, б – второй запуск, в – третий запуск.

# Задание 9

## 9.1. Постановка задачи

Посчитать выражения и провести проверку с помощью электронных таблиц:

)

9.2. Алгоритм решения

## 9.3. Текст метода решения задачи

def task9():  
 result1 = 0  
 for i in range(1, 8 + 1):  
 for j in range(1, i + 1):  
 result1 += (j + i) \*\* 2  
 print("result 1 = {:.4f}".format(result1))  
  
 result2 = 1  
 for i in range(1, 5 + 1):  
 for j in range(1, i + 1):  
 result2 \*= j  
 print("result 2 = {:.4f}".format(result2))  
  
 result3 = 0  
 for i in range(1, 8 + 1):  
 result3\_2 = 1  
 for j in range(1, 2 \* i + 1):  
 for k in range(1, 2 + 1):  
 if (2 \* (j \*\* i) - (k \*\* j)) == 0: # при определенных значениях получается 0  
 print(j, " ", k) # из-за этого ответ всегда = 0  
 result3\_2 \*= (2 \* (j \*\* i) - (k \*\* j))  
 result3 += result3\_2  
 print("result 3 = {:.4f}".format(result3))

## 9.4. Тестирование программы

Тестирование программы представлено на рисунке 9.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| а | б | в |

Рис. 9. Выполнение программы:  
а – первый запуск, б – второй запуск, в – третий запуск.