

Лабораторная работа №3

«Создание классов. Создание экземпляров классов. Создание конструкторов»

1 Первая часть отчета

1.1 Упражнение 2.1

Требуется разработать класс расстояния, включающий в себя переменные для метров и сантиметров, методы для ввода с клавиатуры, установки значений, вывода на экран атрибутов. Далее в коде необходимо создать три экземпляра класса, атрибутам одного из экземпляров задать значения и продемонстрировать что атрибуты в таком случае меняются у конкретных экземпляров класса, а не объекта в целом.

Далее приведен листинг программы. На рисунке 1 результат работы.

Код упражнения 2.1

```
class Dist:
    meters, centimeters = 0, 0.0
    def set_dist(self, mt, ct):
        self.meters = mt
        self.centimeters = ct
    def get_dist(self):
        self.meters = int(input("Введите число метров: "))
        self.centimeters = float(input("Введите число сантиметров: "))
    def show_dist(self):
        print("{0} м {1} см".format(self.meters, self.centimeters))

dist1 = Dist()
dist2 = Dist()
dist3 = Dist()
dist2.set_dist(14, 25.)
dist3.get_dist()
print("dist1 = ", end = " ")
dist1.show_dist()
print("dist2 = ", end = " ")
dist2.show_dist()
```

```
print("dist3 = ", end = " ")  
dist3.show_dist()
```

```
Введите число метров: 44  
Введите число сантиметров: 28.33  
dist1 = 0 м 0.0 см  
dist2 = 14 м 25.0 см  
dist3 = 44 м 28.33 см
```

Рисунок 1 - результат работы упражнения 2.1

1.2 Упражнение 2.2

Требуется создать пустой класс, создать два его экземпляра. Далее необходимо создать атрибут объекта класса, и добавить атрибуты экземплярам класса. Вывести на экран атрибуты созданных экземпляров классов.

Далее приведен листинг программы. На рисунке 2 результат работы.

Код упражнения 2.2

```
class MyClass:  
    pass  
MyClass.x = 100  
obj1, obj2 = MyClass(), MyClass()  
obj1.y = 10  
obj2.y = 20  
print("obj1.x = {0} obj1.y = {1}".format(obj1.x, obj1.y))  
print("obj2.x = {0} obj2.y = {1}".format(obj2.x, obj2.y))
```

```
obj1.x = 100 obj1.y = 10  
obj2.x = 100 obj2.y = 20
```

Рисунок 2 - результат работы упражнения 2.2

1.3 Упражнение 2.3

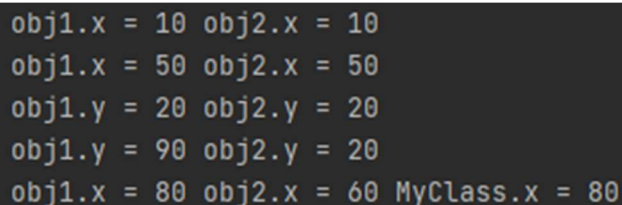
Требуется создать класс, содержащий атрибут класса, и атрибут экземпляра. Продемонстрировать на их примере разницу между атрибутом

экземпляра класса и атрибутом объекта класса присвоением разных значений атрибутам и вывода результатов на экран.

Далее приведен листинг программы. На рисунке 3 результат работы.

Код упражнения 2.3

```
class MyClass:
    x = 10
    def __init__(self):
        self.y = 20
obj1, obj2 = MyClass(), MyClass()
print("obj1.x = {0} obj2.x = {1}".format(obj1.x, obj2.x))
MyClass.x = 50
print("obj1.x = {0} obj2.x = {1}".format(obj1.x, obj2.x))
print("obj1.y = {0} obj2.y = {1}".format(obj1.y, obj2.y))
obj1.y = 90
print("obj1.y = {0} obj2.y = {1}".format(obj1.y, obj2.y))
obj2.x = 60
MyClass.x = 80
print("obj1.x = {0} obj2.x = {1} MyClass.x = {2}".format(obj1.x, obj2.x,
MyClass.x))
```



```
obj1.x = 10 obj2.x = 10
obj1.x = 50 obj2.x = 50
obj1.y = 20 obj2.y = 20
obj1.y = 90 obj2.y = 20
obj1.x = 80 obj2.x = 60 MyClass.x = 80
```

Рисунок 3 - результат работы упражнения 2.3

1.2 Упражнение 2.4

Требуется разработать класс расстояния с конструктором, принимающим значения метров и сантиметров. Данный класс должен включать в себя переменные для метров и сантиметров, методы для ввода с клавиатуры, установки значений, вывода на экран атрибутов. Далее в коде

необходимо создать три экземпляра класса, инициализировав один из них значениями отличными от нуля, а для другого ввести с клавиатуры.

Результаты вывести на экран.

Далее приведен листинг программы. На рисунке 4 результат работы.

Код упражнения 2.4

```
class Dist:
    def __init__(self, mt, ct):
        self.meters = mt
        self.centimeters = ct
        print("Работает конструктор")
    def get_dist(self):
        self.meters = int(input("Введите число метров: "))
        self.centimeters = float(input("Введите число сантиметров: "))
    def show_dist(self):
        print("{0} м {1} см".format(self.meters, self.centimeters))
dist1 = Dist(0, 0.0)
dist2 = Dist(14, 25.)
dist3 = Dist(0, 0.0)
dist3.get_dist()
print("dist1 = ", end = " ")
dist1.show_dist()
print("dist2 = ", end = " ")
dist2.show_dist()
print("dist3 = ", end = " ")
dist3.show_dist()
```

```

Работает конструктор
Работает конструктор
Работает конструктор
Введите число метров: 44
Введите число сантиметров: 28.33
dist1 = 0 м 0.0 см
dist2 = 14 м 25.0 см
dist3 = 44 м 28.33 см

```

Рисунок 4 - результат работы упражнения 2.4

2 Вторая часть отчета

2.1 Задание

Разработать классы с методом-конструктором для расчета своего варианта лабораторных работ №1 и №2 с применением методов класса. Конструктор принимает в качестве аргумента входные параметры расчетных функций.

Для расчета кусочно-ломанной функции необходимо создать три метода, для расчета каждой из ветвей. В основной части программы осуществить проверку ветвей условия и вызвать соответствующие методы.

2.2 Решение ЛР №1 с помощью класса с конструктором

Для решения этой задачи требуется рассчитать значение арифметического выражения методом класса, где входные значения для функции заполняются с помощью конструктора. На рисунке 5 представлен результат работы программы, на рисунке 6 результат из ЛР №1 для сравнения выходных данных.

Листинг 1 – код решения ЛР №1

```

import math
class lab1:
    def __init__(self, x, y):
        self.output = (1 - math.e**(x*y))/(0.7 * math.log10(math.fabs(1 - x**2)))
x = float(input("Введите x "))
y = float(input("Введите y "))
result = lab1(x, y).output

```

```

c = math.ceil(result)
t = math.trunc(result)
f = math.floor(result)
r = round(result, 3)
print("Результат работы программы: " + str(result))
print("Округленный до большего результат: " + str(c))
print("Усеченный до целого результат: " + str(t))
print("Округленный до меньшего результат: " + str(f))
print("Округленный с точностью до трех знаков результат: " + str(r))
f = open("lab1result.txt", "w")
f.write("Результат работы программы: " + str(result) + "\n")
f.write("Округленный до большего результат: " + str(math.ceil(result)))
f.write("\nУсеченный до целого результат: " + str(math.trunc(result)))
f.write("\nОкругленный до меньшего результат: " + str(math.floor(result)))
f.write("\nОкругленный с точностью до трех знаков результат: " +
str(round(result, 3)))
f.close()

```

```

Введите x 13
Введите y 0.54321
Результат работы программы: -748.1858229900707
Округленный до большего результат: -748
Усеченный до целого результат: -748
Округленный до меньшего результат: -749
Округленный с точностью до трех знаков результат: -748.186

```

Рисунок 5 – Результат работы программы из листинга 1

```

Введите x 13
Введите y 0.54321
Результат работы программы: -748.1858229900707
Округленный до большего результат: -748
Усеченный до целого результат: -748
Округленный до меньшего результат: -749
Округленный с точностью до трех знаков результат: -748.186

```

Рисунок 6 - Результат из ЛР №1

2.3 Решение ЛР №2 с помощью класса с конструктором

Для решения лабораторной работы №2 требуется рассчитать значение кусочно-ломанной функции с помощью трех методов класса, с использованием конструктора. Выбор метода будет происходить в основном коде программы при выборе ветви системы. На рисунке 7 изображена блок-схема программы.

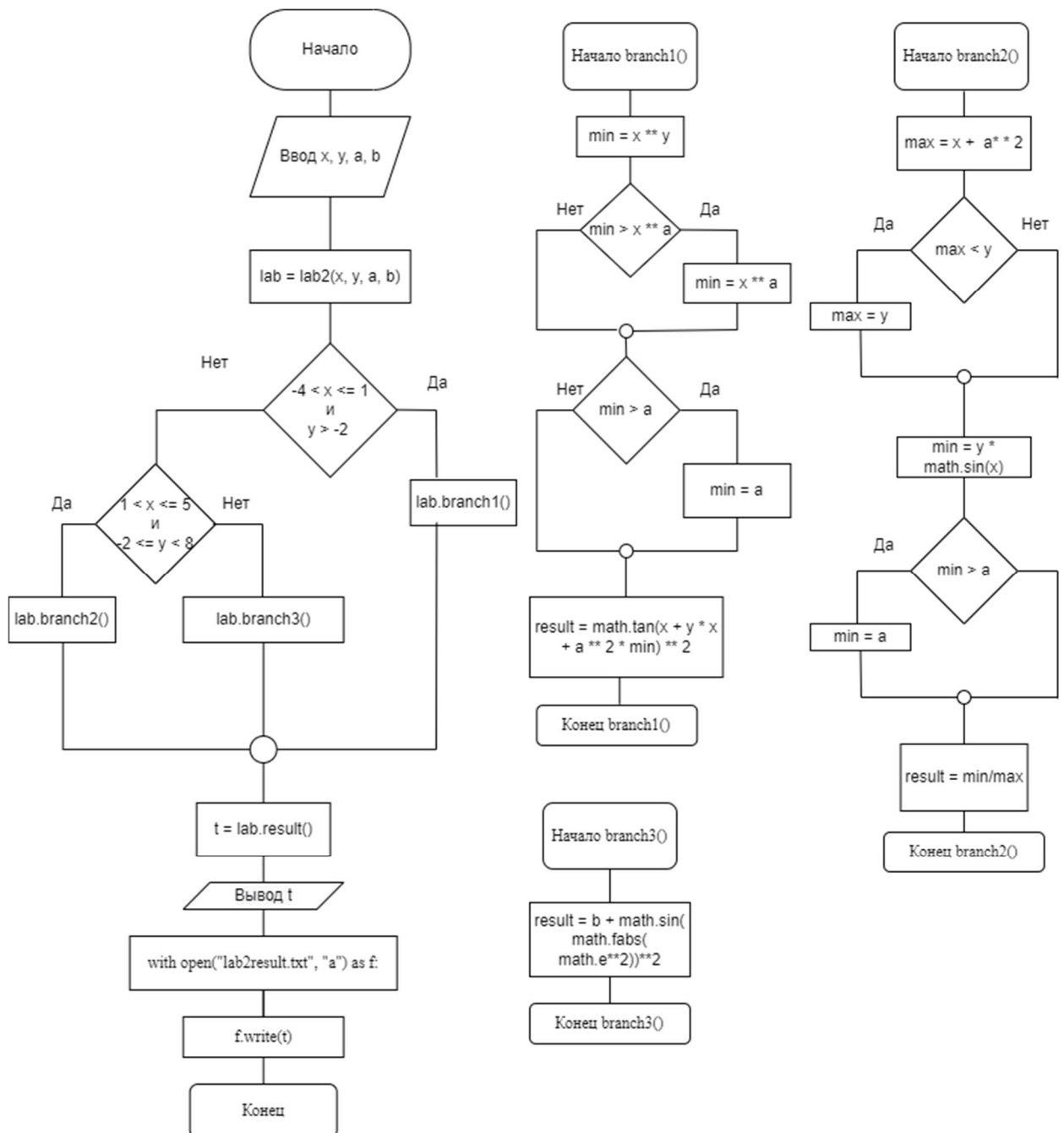


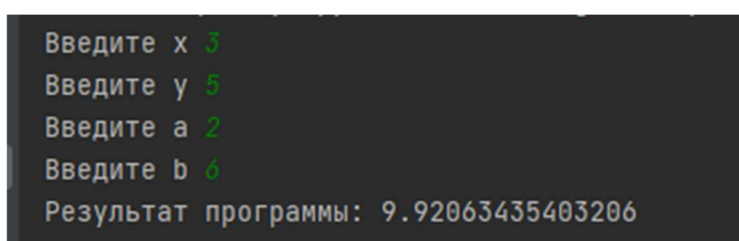
Рисунок 7 - Блок-схема программы для решения кусочно-ломанной функции методами класса

Листинг 2 – код решения ЛР №2

```
import math
import logging
class lab2:
    def __init__(self, x, y, a, b):
        self.x = x
        self.y = y
        self.a = a
        self.b = b
        self.result = 0
    def branch1(self):
        min = self.x ** self.y
        if min > math.e ** self.x:
            min = math.e ** self.x
        if min > a:
            min = a
        self.result = math.tan(self.x + self.y * self.x + self.a ** 2 * min) ** 2
    def branch2(self):
        max = self.x + self.a ** 2
        if max < self.y:
            max = self.y
        min = self.y * math.sin(x)
        if min > self.a:
            min = self.a
        try:
            self.result = max / min
        except Exception as e:
            print("Деление на ноль!")
            logging.error(str(e))
            exit()
    def branch3(self):
        self.result = self.b + math.sin(math.fabs(math.e ** self.x)) ** 2
logging.basicConfig(filename="log.txt", level=logging.DEBUG)
x = float(input("Введите x "))
y = float(input("Введите y "))
a = float(input("Введите a "))
b = float(input("Введите b "))
lab = lab2(x, y, a, b)
if -4 < x <= 1 and y > -2:
    lab.branch1()
elif 1 < x <= 5 and -2 <= y < 8:
    lab.branch2()
else:
```

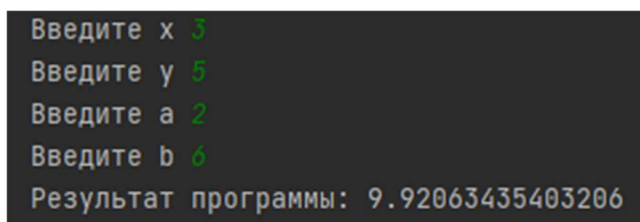
```
lab.branch3()
t = lab.result
logging.info(str(t))
print("Результат программы: " + str(t))
try:
    with open("lab2result.txt", "a") as f:
        f.write("Результат работы программы: " + str(t) + "\n")
except Exception as e:
    logging.error(str(e))
```

Далее на рисунках 8 – 9 приведены результаты кода из листинга 2 и результаты из ЛР №2 соответственно, из которых можно сделать вывод о соответствии результатов.



```
Введите x 3
Введите y 5
Введите a 2
Введите b 6
Результат программы: 9.92063435403206
```

Рисунок 8 - результат работы программы из листинга 2



```
Введите x 3
Введите y 5
Введите a 2
Введите b 6
Результат программы: 9.92063435403206
```

Рисунок 9 - результаты из ЛР №2

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) Гуриков, С. Р. Основы алгоритмизации и программирования на Python : учебное пособие / С.Р. Гуриков. — Москва : ИНФРА-М, 2022. — 343 с. — (Высшее образование: Бакалавриат). - ISBN 978-5-16-017142-5. - Текст : электронный. - URL: <https://znanium.com/catalog/product/1356003> . – Режим доступа: по подписке. + библиотека МТУСИ

2) Дроботун, Н. В. Алгоритмизация и программирование. Язык Python : учебное пособие / Н. В. Дроботун, Е. О. Рудков, Н. А. Баев. — Санкт-Петербург : Санкт-Петербургский государственный университет промышленных технологий и дизайна, 2020. — 119 с. — ISBN 978-5-7937-1829-5. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/102400.html>

3) Шелудько, В. М. Основы программирования на языке высокого уровня Python : учебное пособие / В. М. Шелудько. — Ростов-на-Дону, Таганрог : Издательство

Южного федерального университета, 2017. — 146 с. — ISBN 978-5-9275-2649-9. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: <https://www.iprbookshop.ru/87461.html> (дата обращения: 17.10.2021). — Режим доступа: для авторизир. пользователей

4) Шелудько, В. М. Язык программирования высокого уровня Python. Функции, структуры данных, дополнительные модули : учебное пособие / В. М. Шелудько. — Ростов-на-Дону, Таганрог : Издательство Южного федерального университета, 2017. — 107 с. — ISBN 978-5-9275-2648-2. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: <https://www.iprbookshop.ru/87530.html> (дата обращения: 17.10.2021). — Режим доступа: для авторизир. пользователей

5) Коломейченко, А. С. Информационные технологии : учебное пособие для вузов / А. С. Коломейченко, Н. В. Польшакова, О. В. Чеха. — 2-е изд., перераб. — Санкт-Петербург : Лань, 2021. — 212 с. — ISBN 978-5-8114-7564-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/177030>