МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ

КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

ордена Трудового Красного Знамени федеральное государственное

бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

Практические работы

по дисциплине

«Проектный практикум»

|  |  |
| --- | --- |
|  | Выполнил: студент гр. БЭИ2202  Кулешов А. С.  Вариант 14  Проверил: к.п.н., доцент кафедры «Информатика» Гуриков С. Р. |

Москва, 2024 г.

**Практическая работа №1**

**«МЕТОД ПОШАГОВОЙ ДЕТАЛИЗАЦИИ»**

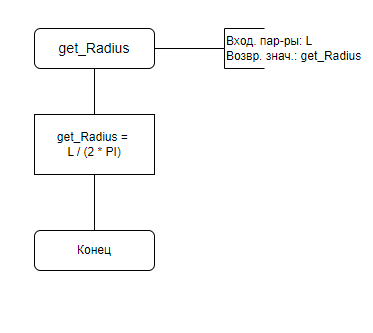
1. Индивидуальное задание на разработку проекта

Вычислить значение площади каждого из трёх кругов, ограниченных тремя окружностями, длины которых L1, L2, L3 известны. Используя формулы для окружностей:

**Назначение функции**: Вычислить площадь круга , предварительно вычислив R по формуле

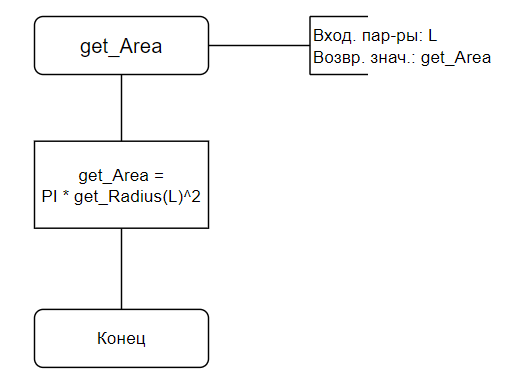
1. Схемы алгоритмов

Для функции, решающей поставленную задачу, была написана вспомогательная функция нахождения радиуса окружности get\_Radius(), алгоритм которой представлен на рисунке 1.



* + 1. Схема Функция вычисления радиуса

Cхема функции, которая решает задачу показана на рисунке 2.



* + 1. Cхема функции, которая решает задачу. Функция вычисления площади круга

Также был реализован алгоритм, который позволяет работать 3 тремя полями одновременно, см. на рисунке 3.

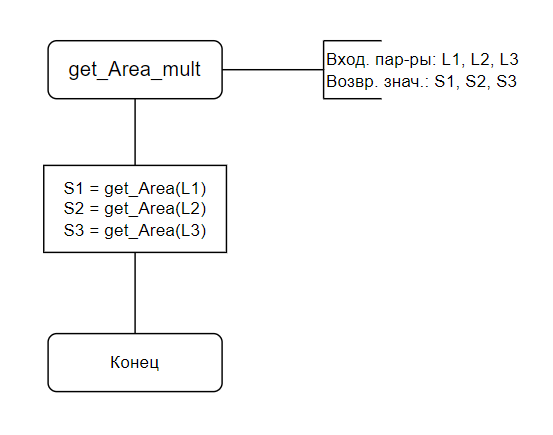


Рисунок 3 – Функция вычисления площади круга для нескольких кругов

Схема функции основной программы показана на рисунке 4.

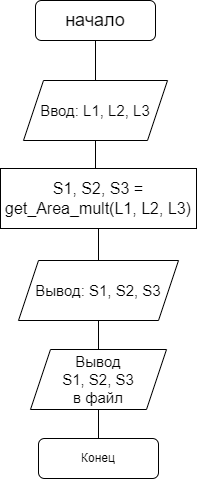


Рисунок 4 – Схема функции основной программы

1. Программный код

import math  
  
def get\_Radius(L):  
 return L / 2 \* math.pi  
  
def get\_Area(L):  
 return math.pi \* get\_Radius(L)\*\*2  
  
def get\_Area\_mult(L1, L2, L3):  
 return get\_Area(L1), get\_Area(L2), get\_Area(L3)  
  
L1, L2, L3 = 0, 0, 0  
  
try:  
 L1 = float(input("Введите длину окружности L1: "))  
 L2 = float(input("Введите длину окружности L2: "))  
 L3 = float(input("Введите длину окружности L3: "))  
 if (L1 < 0 or L2 < 0 or L3 < 0):  
 print("Введённые значения должны быть не меньше нуля")  
 quit()  
except ValueError:  
 print("Введённое значение должно быть числом")  
 quit()  
  
S1, S2, S3 = get\_Area\_mult(L1, L2, L3)  
  
print(f"Выходные данные:\nS1: {S1}, S2: {S2}, S3:{S3}")  
file = open("result.txt", "w", encoding="utf-8")  
file.write(f'Исходные данные:\n L1: {L1}, L2: {L2}, L3:{L3} \n')  
file.write(f'Выходные данные:\n S1: {S1}, S2: {S2}, S3: {S3}')  
file.close()

1. Результаты работы программы

На рисунке 5 представлен результат работы программы при выдуманных данных для проверки значения, представленного программой. Как видно, программа работает корректно.

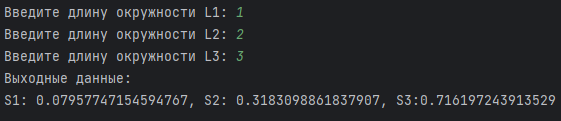


Рисунок 5 – Результаты работы программы

Результат записи в файл показан на рисунке 6.

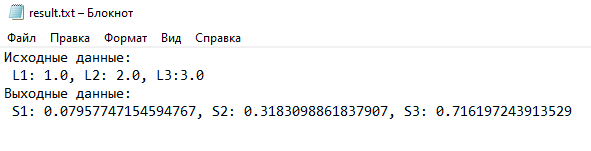


Рисунок 6 – Результаты записи в файл

Обработка исключений показана на рисунках 7-8.

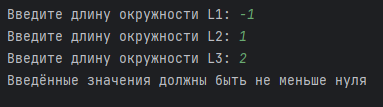


Рисунок 7 – Обработка исключений. Отрицательные числа

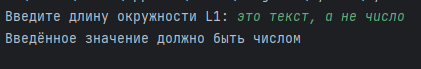


Рисунок 8 – Обработка исключений. Текст

1. Сравнительный анализ работы двух программ

Обе программы были созданы с одной целью - получение значения функции и обеспечение взаимодействия с пользователем.

Программа, написанная на языке C++, использует графический интерфейс пользователя, который основан на Windows Forms Application. В программе на CLI/C++ пользователь вводил данные в текстовые поля и получал ответ в другие текстовые поля, рисунок 9.

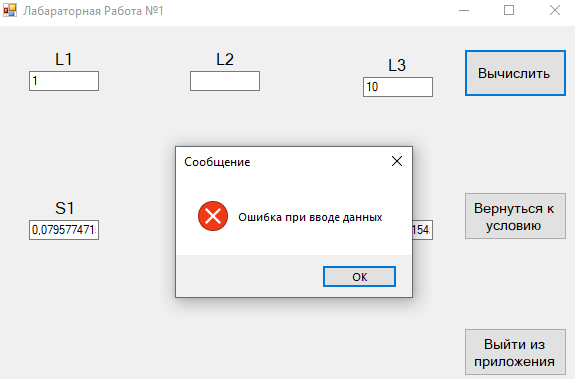


Рисунок 8 – Ввод и вывод данных в C++. Сообщение об ошибке

Программа на языке Python предлагает интерфейс в виде командной строки. В ней пользователь вводит значение переменной в консоль, и результат выводится также в консоль. Рисунок 9.

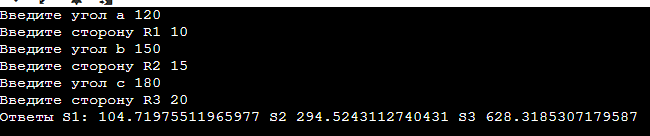


Рисунок 9 – Ввод и вывод данных в Python

В обеих программах предусмотрена обработка возможных ошибок при вводе данных. В программе на C++, если пользователь вводит некорректное значение, ему выводится сообщение об ошибке в диалоговом окне, рисунок 9. В программе на Python при вводе некорректного значения также выводится сообщение об ошибке, но в консоль.

Суммируя все выше перечисленное, разница обеих программ в интерфейсе, среде разработки, способах ввода и вывода данных и в языке программирования.

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 (дата обращения: 26.02.2024). – Режим доступа: по подписке.

2 ГОСТ 7.32-2017. Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления (введен в действие Приказом Росстандарта от 24.10.2017 N 1494-ст) — М., Стандартинформ, 2017.

**Практическая работа №2**

**«программирование алгоритмов разветвляющихся структур»**

1. Индивидуальное задание на разработку проекта

Запрограммировать систему кусочно-ломанной функции согласно индивидуальному варианту №14:

1. Программный код

Программный код основного файла main.py продемонстрирован ниже

from module import \*  
import logging  
logging.basicConfig(filename='log.txt', level=logging.INFO, encoding='utf-8')  
  
  
  
  
x, a, z, b, c = 0, 0, 0, 0, 0  
  
try:  
 x = float(input("Введите x: "))  
 a = float(input("Введите a: "))  
 z = float(input("Введите z: "))  
 b = float(input("Введите b: "))  
 c = float(input("Введите c: "))  
except ValueError as e:  
 print("Введённые значения должны быть числового типа")  
 logging.error(str(e))  
 quit()  
  
result = 0  
  
try:  
 result = func(x,a,z,b,c)  
except Exception as e:  
 print("Произошла ошибка при вычислениях")  
 logging.error(str(e))  
 quit()  
print()  
print(f"Выходные данные:")  
print(f"func(три знака после запятой): {'%.3f' % result}")  
print(f"func(ceil): {math.ceil(result)}")  
print(f"func(floor): {math.floor(result)}")  
print(f"func(round): {round(result)}")  
print(f"func(trunc): {math.trunc(result)}")  
  
file = open("result.txt", "w", encoding="utf-8")  
file.write(f'Исходные данные:\n x: {x}, a: {a}, z:{z}, b:{b}, c:{c} \n')  
  
file.write(f"Выходные данные:\n")  
file.write(f"func(три знака после запятой): {'%.3f' % result}\n")  
file.write(f"func(ceil): {math.ceil(result)}\n")  
file.write(f"func(floor): {math.floor(result)}\n")  
file.write(f"func(round): {round(result)}\n")  
file.write(f"func(trunc): {math.trunc(result)}\n")  
  
logging.info(f"Значение результирующей функции: {'%.3f' % result}")  
  
  
file.close()

Программный код файла модуля module.py продемонстрирован ниже

import math  
  
def func(x, a, z, b, c):  
  
 if x < 0 and z < 0:  
 m = x  
 if a > m:  
 m = a  
 m2 = z  
 if x \* a \*\* m < m2:  
 m2 = x \* a \*\* m  
 return math.log(x \* a + m2)  
 else:  
 if 0 <= x and x < 4 and 0 <= z and z < 3:  
 m = z  
 if (math.sqrt(x) > m):  
 m = math.sqrt(x)  
 if (abs(math.sqrt(x)) > m):  
 m = abs(math.sqrt(x))  
 return math.log(b \*\* 2) + m  
 else:  
 return b + c \* x

1. Результаты работы программы

На рисунках 1-3 представлены результаты работы программы при выдуманных данных для проверки значений, представленных программой. Каждый тест был подготовлен на соответствующую ветку решения. Как видно, программа работает корректно.

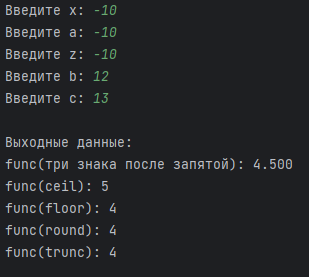


Рисунок 1 – Результаты работы программы для первого ветвления

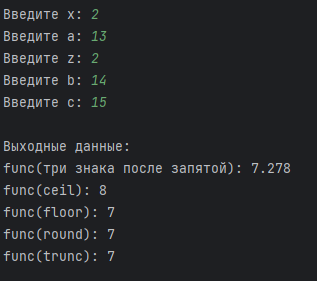


Рисунок 2 – Результаты работы программы для второго ветвления

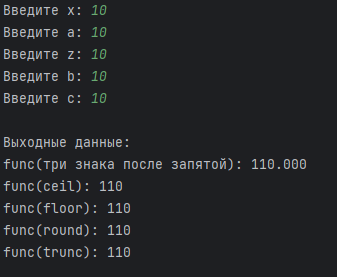


Рисунок 3 – Результаты работы программы для третьего ветвления

Результат записи в файл показан на рисунке 4.

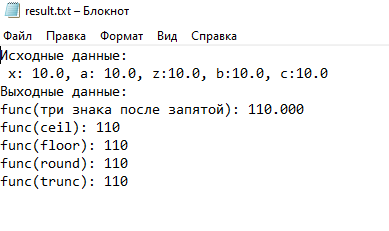


Рисунок 4 – Результаты записи в файл

Обработка исключений показана на рисунке 5.

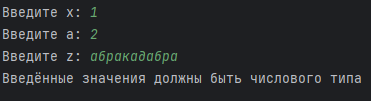


Рисунок 5 – Обработка исключений. Текст

Запись логов показана на рисунке 6.

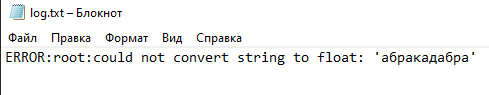


Рисунок 6 – Запись логов

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 (дата обращения: 26.02.2024). – Режим доступа: по подписке.

2 ГОСТ 7.32-2017. Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления (введен в действие Приказом Росстандарта от 24.10.2017 N 1494-ст) — М., Стандартинформ, 2017.

**Практическая работа №3**

**«Создание классов. создание эксземпляров классов. создание конструкторов»**

1.1 Задание на разработку программы

Разработать класс Dist, который содержит два поля meters и centimeters. Также реализовать методы:

* set\_dist(self, mt, ct) — в качестве аргументов принимает количество метров и сантиметров, задаёт значение соответствующим полям класса.
* get\_dist(self) — требует пользователя ввести данные со стандартного потока ввода, а затем изменяется соответствующие значения полей;
* show\_dist(self) — выводит в поток вывода данные о расстоянии.

Затем создать три объекта класса Dist с названиями dist1, dist2, dist3. Для объекта dist2 методом set\_dist() задать расстояние, используя аргументы и . Для объекта dist3 запросить расстояние у пользователя методом get\_dist(). Вывести на экран расстояния всех трех объектов методом show\_dist().

**1.2 Программный код**

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(f'{self.meters} м {self.centimeters} см')  
  
  
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()

**1.3 Результат работы программы**

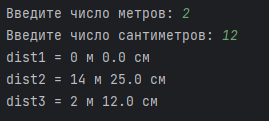


Рисунок 1 – Результат работы программы

2.1 Задание на разработку программы

Разработать пустой класс MyClass. В основной части программы создать атрибут x объекта класса MyClass. Создать два экземпляра класса MyClass с именами obj1 и obj2. Добавить обоим экземплярам атрибут y с помощью динамического создания атрибутов. Вывести на экран значения атрибутов x и y для обоих экземпляров класса MyClass.

**2.2 Программный код**

class MyClass:  
 pass  
  
  
MyClass.x = 100  
obj1, obj2 = MyClass(), MyClass()  
obj1.y = 10  
obj2.y = 20  
  
print(f'obj1.x = {obj1.x} obj1.y = {obj1.y}')  
print(f'obj2.x = {obj2.x} obj2.y = {obj2.y}')

**2.3 Результат работы программы**



Рисунок 2 – Результат работы программы

3.1 Задание на разработку программы

Разработать класс MyClass со статическим атрибутом x, по умолчанию равным 10, и динамическим атрибутом y, инициализируемым значением в конструкторе класса. Создать два объекта класса MyClass с именами obj1 и obj2. Вывести на экран значения атрибута x для обоих объектов. Затем изменить значение статического атрибута x класса MyClass на и вывести значения атрибута x для обоих объектов. Затем изменить значения атрибута y объекта obj1 на и вывести значения атрибутов y для обоих объектов. Установить значение атрибута x объекта obj2 равным и значение статического атрибута x класса MyClass равным . Вывести на экран значения атрибута x для обоих объектов и для класса MyClass.

**3.2 Программный код**

class MyClass:  
 x = 10  
  
 def \_\_init\_\_(self):  
 self.y = 20  
  
  
obj1, obj2 = MyClass(), MyClass()  
print(f'obj1.x = {obj1.x} obj2.x = {obj2.x}')  
MyClass.x = 50  
print(f'obj1.x = {obj1.x} obj2.x = {obj2.x}')  
print(f'obj1.y = {obj1.y} obj2.y = {obj2.y}')  
obj1.y = 90  
print(f'obj1.y = {obj1.y} obj2.y = {obj2.y}')  
obj2.x = 60  
MyClass.x = 80  
print(f'obj1.x = {obj1.x} obj2.x = {obj2.x} MyClass.x = {MyClass.x}')

**3.3 Результат работы программы**

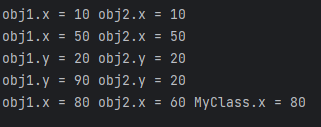


Рисунок 3 – Результат работы программы

4.1 Задание на разработку программы

Разработать класс Dist. Определить метод инициализации, который принимает два аргумента: метры (mt) и сантиметры (ct). В методе инициализации установить начальные значения атрибутов meters и centimeters объекта класса. Класс должен иметь следующие методы:

* get\_dist(self) — запрашивает у пользователя ввод количества метров и сантиметров, устанавливает их соответствующие значения внутренним переменным объекта;
* show\_dist(self) — выводит на экран расстояние в формате «метры и сантиметры».

Создать три объекта класса Dist с названиями dist1, dist2, dist3, используя метод инициализации. Для dist1 и dist2 установить аргументы, равные нулю. Для объекта dist2 установить аргументы равные и . Для объекта dist3 запросить расстояние у пользователя методом get\_dist(). Вывести на экран расстояния всех трех объектов методом show\_dist().

**4.2 Программный код**

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(f'{self.meters} м {self.centimeters} см')  
  
  
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()

**4.3 Результат работы программы**

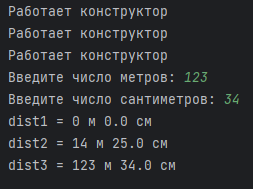


Рисунок 4 – Результат работы программы

5.1 Индивидуальное задание на разработку программы

Запрограммировать систему кусочно-ломанной функции согласно индивидуальному варианту №14:

Для вычисления значения кусочно-заданной функции написать функции для каждой ветви системы. Для каждой из них создать свой модуль.

Для выполнения задания следует разработать класс Branches с методом-конструктором, который должен выводить строку «Работает конструктор». Внутри класса должны быть реализованы следующие методы:

* input(self) — реализует ввод данных с потока ввода для вычисления значения кусочно-заданной функции;
* get\_branch1(self) — обращается к модулю, вызывая функцию вычисления ветки «»;
* get\_branch2(self) — обращается к модулю, вызывая функцию вычисления ветки «»;
* get\_branch3(self) — обращается к модулю, вызывая функцию вычисления ветки «».

В основной части программы, кроме ввода данных, осуществить проверку условий для каждой ветви программы с вызовом соответствующего метода и вывести полученный результат.

5.2 Программный код модулей

Программный код модуля branch1:

import math  
  
def branch(x, a, z, b, c):  
 print("Ветвь x < 0 и z < 0")  
 m = x  
 if a > m:  
 m = a  
 m2 = z  
 if x \* a \*\* m < m2:  
 m2 = x \* a \*\* m  
 return math.log(x \* a + m2)

Программный код модуля branch2:

import math  
  
def branch(x, a, z, b, c):  
 print("Ветвь 0 <= x < 4 и 0 <= z < 3")  
 m = z  
 if (math.sqrt(x) > m):  
 m = math.sqrt(x)  
 if (abs(math.sqrt(x)) > m):  
 m = abs(math.sqrt(x))  
 return math.log(b \*\* 2) + m

Программный код модуля branch3:

import math  
  
def branch(x, a, z, b, c):  
 print("Ветвь иного случая")  
 return b + c \* x

5.3 Программный код основной части программы

import branch1  
import branch2  
import branch3  
  
  
class Branches :  
 def \_\_init\_\_(self, x, a, z, b, c):  
 self.x = x  
 self.a = a  
 self.z = z  
 self.b = b  
 self.c = c  
 print('Работает конструктор')  
  
 def input(self):  
 values = map(float, input('Введите значения (x, a, z, b, c), разделив их пробелами: ').split())  
 self.x, self.a, self.z, self.b, self.c = values  
  
 def get\_branch1(self):  
 return branch1.branch(self.x, self.a, self.z, self.b, self.c)  
  
 def get\_branch2(self):  
 return branch2.branch(self.x, self.a, self.z, self.b, self.c)  
  
 def get\_branch3(self):  
 return branch3.branch(self.x, self.a, self.z, self.b, self.c)  
  
  
branching = Branches(0,0,0,0,0)  
try:  
 branching.input()  
except:  
 print("Произошла ошибка ввода")  
 exit(-1)  
res = 0  
  
if branching.x < 0 and branching.z < 0:  
 res = branching.get\_branch1()  
else:  
 if 0 <= branching.x and branching.x < 4 and 0 <= branching.z and branching.z < 3:  
 res = branching.get\_branch2()  
 else:  
 res = branching.get\_branch3()  
  
print(f'Результат ветвления: {round(res, 5)}')

**4.4 Результат работы программы**

Результат работы программы на первой ветви показан на рисунке 5. Набор входных данных идентичен тому же в предыдущей работе, результаты совпадают.

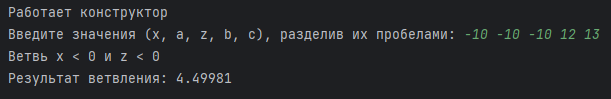


Рисунок 5 – Результат работы программы на первой ветви

Результат работы программы на второй ветви показан на рисунке 6. Набор входных данных идентичен тому же в предыдущей работе, результаты совпадают.

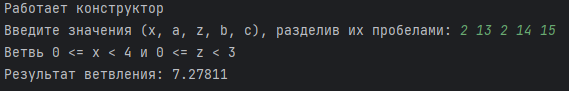


Рисунок 6 – Результат работы программы на второй ветви

Результат работы программы на третьей ветви показан на рисунке 7. Набор входных данных идентичен тому же в предыдущей работе, результаты совпадают.

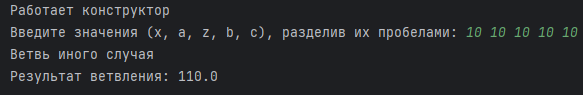


Рисунок 7 – Результат работы программы на третьей ветви

Результат работы программы на некорректно введённых данных показан на рисунке 8.

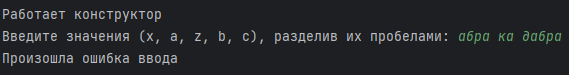


Рисунок 8 – Результат работы программы при неверных входных данных

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 (дата обращения: 12.02.2024). – Режим доступа: по подписке.

2 Букунов, С. В. Объектно ориентированное программирование на языке Python : учебное пособие / С. В. Букунов, О. В. Букунова. — Санкт-Петербург : Санкт-Петербургский государственный архитектурно-строительный университет, ЭБС АСВ, 2020. — 119 c. — ISBN 978-5-9227-1128-9. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/117194.html

3 ГОСТ 7.32-2017. Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления (введен в действие Приказом Росстандарта от 24.10.2017 N 1494-ст) — М., Стандартинформ, 2017.

**Практическая работа №4**

**«СОЗДАНИЕ МЕНЮ. ИНКАПСУЛЯЦИЯ**

1 Задание на разработку программы

Необходимо разработать алгоритм разветвляющей структуры, рассчитывающий значение функции, по формуле:

Для вычисления значения кусочно-заданной функции написать функции для каждой ветви системы. Для каждой из них создать свой модуль.

Для выполнения задания следует разработать класс Branches с методом-конструктором, который должен выводить строку «Работает конструктор». Внутри класса должны быть реализованы следующие методы:

* input(self) — реализует ввод данных с потока ввода для вычисления значения кусочно-заданной функции;
* \_\_get\_branch1(self) — обращается к модулю, вызывая функцию вычисления ветки «»;
* \_\_get\_branch2(self) — обращается к модулю, вызывая функцию вычисления ветки «»;
* \_\_get\_branch3(self) — обращается к модулю, вызывая функцию вычисления ветки «».
* get\_value(self) — совершает проверку ветвления, вызывая соответствующую функцию ветвления.

В основной части программы, кроме ввода данных, осуществить проверку условий для каждой ветви программы с вызовом соответствующего метода и вывести полученный результат. Также нужно реализовать меню для работы с программой. Продемонстрировать доступ к закрытым атрибутам с помощью создания свойств.

1. Программный код

Программный код модуля branch1:

import math  
  
def branch(x, a, z, b, c):  
 print("Ветвь x < 0 и z < 0")  
 m = x  
 if a > m:  
 m = a  
 m2 = z  
 if x \* a \*\* m < m2:  
 m2 = x \* a \*\* m  
 return math.log(x \* a + m2)

Программный код модуля branch2:

import math  
  
def branch(x, a, z, b, c):  
 print("Ветвь 0 <= x < 4 и 0 <= z < 3")  
 m = z  
 if (math.sqrt(x) > m):  
 m = math.sqrt(x)  
 if (abs(math.sqrt(x)) > m):  
 m = abs(math.sqrt(x))  
 return math.log(b \*\* 2) + m

Программный код модуля branch3:

import math  
  
def branch(x, a, z, b, c):  
 print("Ветвь иного случая")  
 return b + c \* x

Программный код модуля main.py:

import branch1  
import branch2  
import branch3  
  
class Branches :  
 def \_\_init\_\_(self, x, a, z, b, c):  
 self.x = x # Открытый атрибут  
 self.\_\_a = a  
 self.\_\_z = z  
 self.\_\_b = b  
 self.\_\_c = c  
 # a,z,b,c - закрытые атрибуты  
 print('Работает конструктор')  
  
 def input(self):  
 values = map(float, input('Введите значения (x, a, z, b, c), разделив их пробелами: ').split())  
 self.x, self.a, self.z, self.b, self.c = values  
  
 def \_\_get\_branch1(self):  
 return branch1.branch(self.x, self.a, self.z, self.b, self.c)  
  
 def \_\_get\_branch2(self):  
 return branch2.branch(self.x, self.a, self.z, self.b, self.c)  
  
 def \_\_get\_branch3(self):  
 return branch3.branch(self.x, self.a, self.z, self.b, self.c)  
  
 def getA(self):  
 return self.a  
  
 @property  
 def valueOfA(self):  
 return self.a  
  
 def get\_value(self):  
 if self.x < 0 and self.z < 0:  
 return self.\_\_get\_branch1()  
 else:  
 if 0 <= self.x and self.x < 4 and 0 <= self.z and self.z < 3:  
 return self.\_\_get\_branch2()  
 else:  
 return self.\_\_get\_branch3()  
  
  
  
  
choice = None  
while choice!="0":  
 print("Меню 0 - Выйти, 1 - Выполнить программу")  
 choice = input("Сделайте выбор ")  
 print()  
  
 if choice == "0":  
 print("Заканчиваем работу")  
 elif choice =="1":  
 print("Введите исходные данные ", end="")  
 branching = Branches(0, 0, 0, 0, 0)  
 try:  
 branching.input()  
 except:  
 print("Произошла ошибка ввода")  
 continue  
  
 #print(branching.\_\_a) - раскоментировать, для получения ошибки  
 print(f'Результат ветвления: {round(branching.get\_value(), 5)}')  
 print(f'Значение а через @property: {branching.valueOfA}')  
 print(f'Значение а через getter: {branching.getA()}')  
 else:  
 print("Такого пункта в меню нет:", choice)

1. Тестирование программы

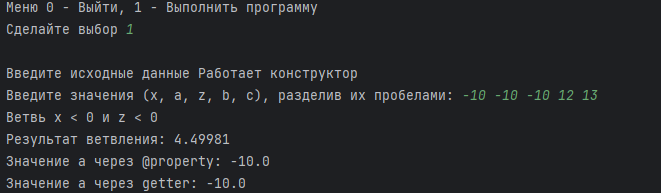


Рисунок 1 – Результат работы программы на первой ветви

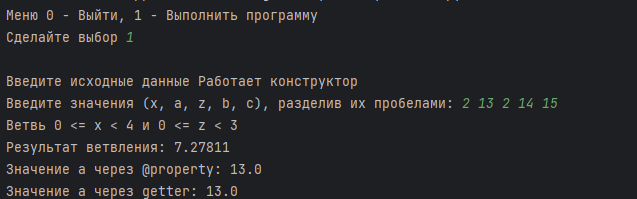


Рисунок 2 – Результат работы программы на второй ветви

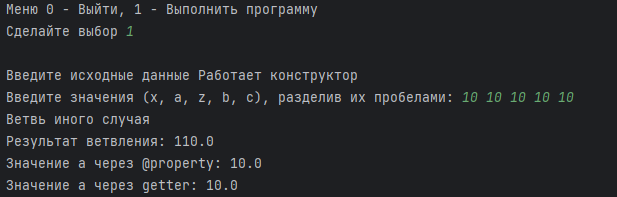


Рисунок 3 – Результат работы программы на третьей ветви

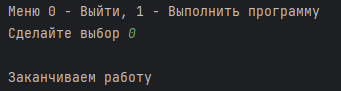


Рисунок 4 – Результат работы меню при выходе

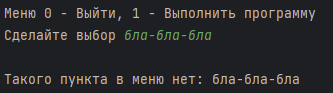


Рисунок 5 – Результат работы меню при неверной опции

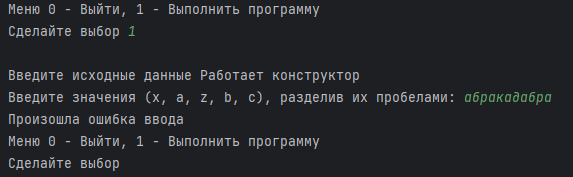


Рисунок 6 – Результат работы программы при неверном вводе

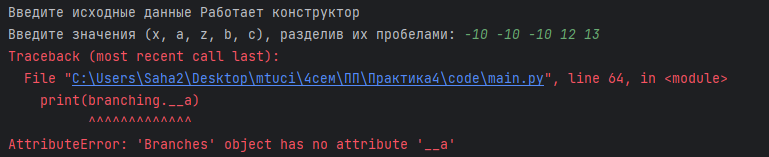


Рисунок 7 – Ошибка при обращении к закрытому атрибуту

Как видно из рисунков 1-3 результат работы программы совпадает с результатом, полученным в ходе выполнения третьей практической работы.

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 (дата обращения: 26.02.2024). – Режим доступа: по подписке.

2 ГОСТ 7.32-2017. Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления (введен в действие Приказом Росстандарта от 24.10.2017 N 1494-ст) — М., Стандартинформ, 2017.