**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №1

«Основные конструкции языка Python»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-32Б |  | преподаватель каф. ИУ5 |
| Мажитов В. |  | 1. Гапанюк Ю.Е. |
|  |  |  |

**Описание задания**

Разработать программу для решения биквадратного уравнения.

1. Программа должна быть разработана в виде консольного приложения на языке Python.
2. Программа осуществляет ввод с клавиатуры коэффициентов А, В, С, вычисляет дискриминант и ДЕЙСТВИТЕЛЬНЫЕ корни уравнения (в зависимости от дискриминанта).
3. Коэффициенты А, В, С могут быть заданы в виде параметров командной строки. Если они не заданы, то вводятся с клавиатуры в соответствии с пунктом 2.
4. Если коэффициент А, В, С введен или задан в командной строке некорректно, то необходимо проигнорировать некорректное значение и вводить коэффициент повторно пока коэффициент не будет введен корректно. Корректно заданный коэффициент - это коэффициент, значение которого может быть без ошибок преобразовано в действительное число.
5. Дополнительное задание 1. Разработайте две программы на языке Python - одну с применением процедурной парадигмы, а другую с применением объектно-ориентированной парадигмы.
6. Дополнительное задание 2. Разработайте две программы - одну на языке Python, а другую на любом другом языке программирования (кроме С++).

**Текст программы**

**Python, процедурная парадигма**

import sys

from dataclasses import dataclass

from math import sqrt

from typing import Callable, Optional, Set

class WrongInput(Exception):

pass

class NoSolutions(Exception):

pass

@dataclass

class Coefficients:

A: float

B: float

C: float

def parse\_float\_from\_string(inp: str, validator: Optional[Callable[[float], bool]] = None) -> float:

digit = float(inp)

if validator and not validator(digit):

raise ValueError()

return digit

def get\_float\_from\_terminal(prompt: str, validator: Optional[Callable[[float], bool]] = None) -> float:

while True:

try:

print(prompt)

inp = input()

return parse\_float\_from\_string(inp, validator)

except ValueError:

print("Неверный ввод")

def get\_coefficients() -> Coefficients:

if len(sys.argv) < 3:

a = get\_float\_from\_terminal("Введите коэффициент A", lambda x: x != 0)

b = get\_float\_from\_terminal("Введите коэффициент B")

c = get\_float\_from\_terminal("Введите коэффициент C")

else:

try:

args = sys.argv[1:]

a = parse\_float\_from\_string(args[0], lambda x: x != 0)

b = parse\_float\_from\_string(args[1])

c = parse\_float\_from\_string(args[2])

except ValueError:

print("Неверные аргументы командной строки")

exit()

return Coefficients(

A=a,

B=b,

C=c,

)

def solve(coefficients: Coefficients) -> Set[float]:

d = coefficients.B \*\* 2 - 4 \* coefficients.A \* coefficients.C

if d < 0:

raise NoSolutions()

roots: Set[float] = set()

possible\_roots = [

(-coefficients.B - sqrt(d)) / (2 \* coefficients.A),

(-coefficients.B + sqrt(d)) / (2 \* coefficients.A),

]

for root in possible\_roots:

if root >= 0:

roots.add(sqrt(root))

roots.add(-sqrt(root))

return roots

if \_\_name\_\_ == "\_\_main\_\_":

coefficients = get\_coefficients()

try:

print(solve(coefficients))

except NoSolutions:

print("Нет решений")

**Golang, объектно-ориентированная парадигма**

Файл main.go

package main

import (

"fmt"

"errors"

"lab1/service"

"lab1/domain"

)

func main() {

inputService := service.NewInputService()

var coefficients domain.Coefficients

var err error

for {

coefficients, err = inputService.Get()

if err != nil {

fmt.Println(err)

if errors.Is(err, domain.ErrWrongConsoleInput) {

break

}

} else {

break

}

}

solverService := service.NewSolverService()

roots, err := solverService.Solve(coefficients)

if err != nil {

fmt.Println(err)

return

}

fmt.Printf("Корни: %+v\n", roots)

}

Файл domain/coefficients.go

package domain

type Coefficients struct {

A float64

B float64

C float64

}

Файл domain/errors.go

package domain

import "errors"

var (

ErrWrongInput = errors.New("Неверный ввод")

ErrWrongConsoleInput = errors.New("Неверный ввод")

ErrNoSolutions = errors.New("Нет решений")

)

Файл domain/input.go

package domain

type Input struct {

A string

B string

C string

}

Файл service/input.go

package service

import (

"fmt"

"lab1/domain"

"os"

"strconv"

)

type InputService struct{}

func NewInputService() \*InputService {

return &InputService{}

}

func (\_ \*InputService) parseFloatFromString(input string) (float64, error) {

value, err := strconv.ParseFloat(input, 64)

if err != nil {

return 0, domain.ErrWrongInput

}

return value, nil

}

func (s \*InputService) getCoefficients(input domain.Input) (output domain.Coefficients, err error) {

output.A, err = s.parseFloatFromString(input.A)

if err != nil {

return

}

if output.A == 0 {

err = domain.ErrWrongInput

return

}

output.B, err = s.parseFloatFromString(input.B)

if err != nil {

return

}

output.C, err = s.parseFloatFromString(input.C)

if err != nil {

return

}

return

}

func getMethodBasedErr(isConsole bool) error {

if (isConsole) {

return domain.ErrWrongConsoleInput

} else {

return domain.ErrWrongInput

}

}

func (s \*InputService) Get() (domain.Coefficients, error) {

var input domain.Input

isConsole := len(os.Args) > 3

if isConsole {

input.A = os.Args[1]

input.B = os.Args[2]

input.C = os.Args[3]

} else {

fmt.Print("Введите коэффициенты через пробел: ")

\_, err := fmt.Scanf("%s %s %s", &input.A, &input.B, &input.C)

if err != nil {

return domain.Coefficients{}, getMethodBasedErr(isConsole)

}

}

coefficients, err := s.getCoefficients(input)

if err != nil {

return domain.Coefficients{}, getMethodBasedErr(isConsole)

}

return coefficients, nil

}

Файл service/solver.go

package service

import (

"lab1/domain"

"math"

)

type SolverService struct{}

func NewSolverService() \*SolverService {

return &SolverService{}

}

func (s \*SolverService) Solve(coefficients domain.Coefficients) ([]float64, error) {

rootSet := map[float64]struct{}{}

d := math.Pow(coefficients.B, 2) - 4\*coefficients.A\*coefficients.C

if d < 0 {

return []float64{}, domain.ErrNoSolutions

}

possibleRoots := []float64{

(-coefficients.B - math.Sqrt(d)) / (2 \* coefficients.A),

(-coefficients.B + math.Sqrt(d)) / (2 \* coefficients.A),

}

for \_, root := range possibleRoots {

if root < 0 {

continue

}

rootSet[-math.Sqrt(root)] = struct{}{}

rootSet[math.Sqrt(root)] = struct{}{}

}

roots := make([]float64, 0, len(rootSet))

for root := range rootSet {

roots = append(roots, root)

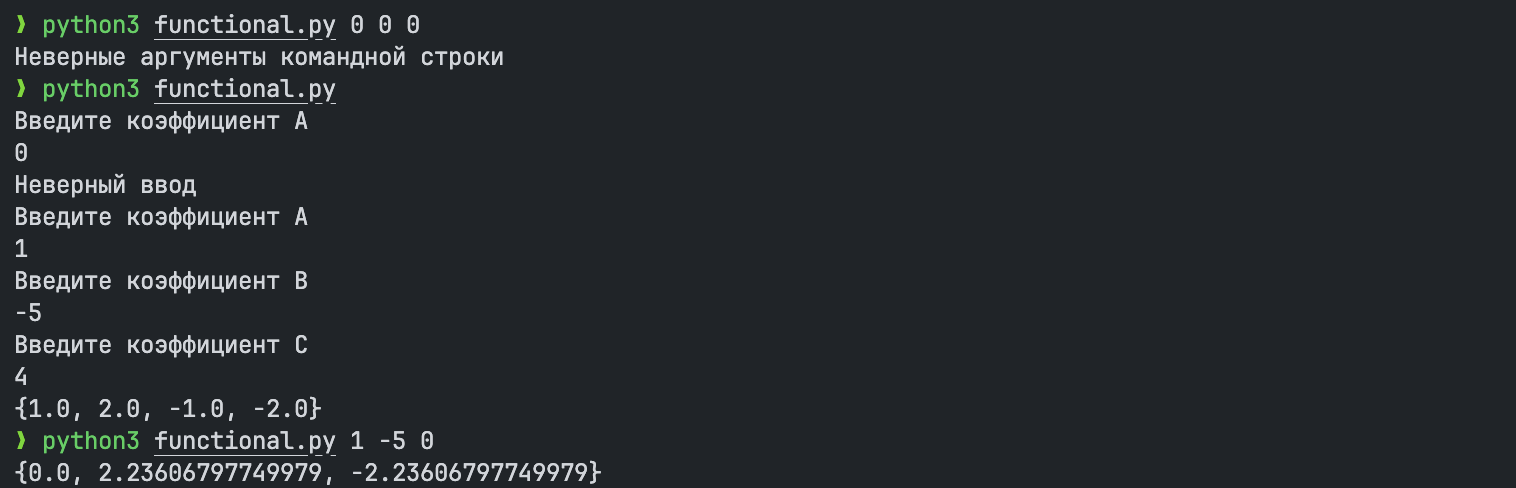
}

return roots, nil

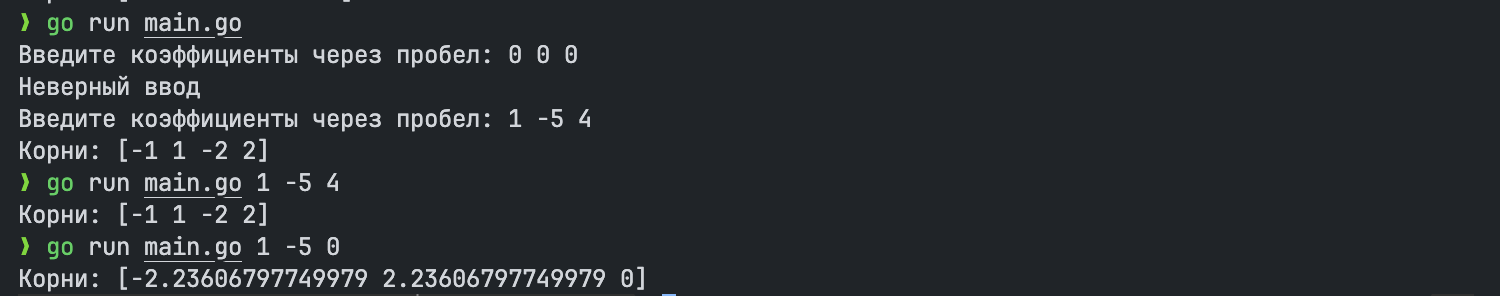
}

**Примеры выполнения программы**

1. **Python, процедурная парадигма**

****

1. **Golang, объектно-ориентированная парадигма**

****