Московский государственный технический университет
им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”

Дисциплина «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №1
«Основные конструкции языка Python»

Выполнил:
студент группы ИУ5-36Б
Галенко Алексей Владимирович
Проверил:
Гапанюк Юрий Евгеньевич

Москва, 2025

Задание:

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 A, B, C, вычисляет дискриминант и ДЕЙСТВИТЕЛЬНЫЕ корни уравнения (в зависимости от дискриминанта).
3. Коэффициенты A, B, C могут быть заданы в виде параметров командной строки (вариант задания параметров приведен в конце файла с примером кода). Если они не заданы, то вводятся с клавиатуры в соответствии с пунктом 2. Описание работы с параметрами командной строки.
4. Если коэффициент A, B, C 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 C++).

Code procedural.py

```
import sys
import math
def get_input_coefficient(args, index):
  try:
    return float(args[index])
  except (IndexError, ValueError):
    return None
def prompt_coefficient(name):
  while True:
    try:
      value = float(input(f"write coefficient {name}: "))
      return value
    except ValueError:
      print("error: coefficient must be float")
def solve_biquadratic(a, b, c):
  discriminant = b ** 2 - 4 * a * c
  if discriminant < 0:
    return []
  roots = set()
  y_first = (-b + math.sqrt(discriminant)) / (2 * a)
  y_second = (-b - math.sqrt(discriminant)) / (2 * a)
  if y_first >= 0:
    x_first = math.sqrt(y_first)
    roots.add(x_first)
    roots.add(-x_first)
  if y_second >= 0:
    x_second = math.sqrt(y_second)
    roots.add(x_second)
    roots.add(-x_second)
  return sorted(list(roots))
def main():
```

```
  args = sys.argv[1:]
  a = get_input_coefficient(args, 0)
  if a is None:
    a = prompt_coefficient('A')
  b = get_input_coefficient(args, 1)
  if b is None:
    b = prompt_coefficient('B')
  c = get_input_coefficient(args, 2)
  if c is None:
    c = prompt_coefficient('C')
  print(f"equation: {a}x^4 + {b}x^2 + {c} = 0")
  roots_real = solve_biquadratic(a, b, c)
  if not roots_real:
    print("no real roots")
  else:
    print(f"real roots: {roots_real}")
if __name__ == "__main__":
  main()
```

Code object.py

```
import sys
import math
class BiquadraticEquationInRealSolution:
  def __init__(self, a, b, c):
    self.a = a
    self.b = b
    self.c = c
    self.roots = []
  def solve(self):
    discriminant = self.b ** 2 - 4 * self.a * self.c
    if discriminant < 0:
      return
    temp_roots = set()
    y1 = (-self.b + math.sqrt(discriminant)) / (2 * self.a)
    y2 = (-self.b - math.sqrt(discriminant)) / (2 * self.a)
    if y1 >= 0:
      x1 = math.sqrt(y1)
      temp_roots.add(x1)
      temp_roots.add(-x1)
    if y2 >= 0:
      x2 = math.sqrt(y2)
      temp_roots.add(x2)
      temp_roots.add(-x2)
    self.roots = sorted(list(temp_roots))
  def display_results(self):
    print(
      f"equation: {self.a}x^4 + {self.b}x^2 + {self.c} = 0")
    if not self.roots:
      print("no real roots")
```

```
    else:
      print(f"real roots: {self.roots}")
def get_input_coefficient(args, index):
  try:
    return float(args[index])
  except (IndexError, ValueError):
    return None
def prompt_coefficient(name):
  while True:
    try:
      return float(input(f"write coefficient {name}: "))
    except ValueError:
      print("error: coefficient must be float")
def main():
  args = sys.argv[1:]
  a = get_input_coefficient(args, 0) or prompt_coefficient('A')
  b = get_input_coefficient(args, 1) or prompt_coefficient('B')
  c = get_input_coefficient(args, 2) or prompt_coefficient('C')
  equation = BiquadraticEquationInRealSolution(a, b, c)
  equation.solve()
  equation.display_results()
if __name__ == "__main__":
  main()
```

Code procedural.c

```
#include <stdio.h>  // For input/output functions like printf
and scanf
#include <stdlib.h> // For string to number conversion (atof)
#include <math.h>   // For math functions, specifically sqrt
(square root)
// Function to prompt the user for a coefficient
// &coeff is a "pointer", it allows the function to modify the
variable in main
void prompt_input_coefficient(const char* name, double* coeff) {
  printf("Enter coefficient %s: ", name);
  // Infinite loop until a valid number is entered
  while (scanf("%lf", coeff) != 1) {
    printf("Error: Invalid input. Please try again: ");
    // Clear the input buffer to prevent the program from getting
stuck on invalid input
    while (getchar() != '\n');
  }
}
int main(int argc, char *argv[]) {
  double a, b, c;
  if (argc == 4) {
    printf("Coefficients taken from command-line arguments.\n");
    a = atof(argv[1]);
    b = atof(argv[2]);
```

```
    c = atof(argv[3]);
  } else {
    printf("No coefficients were passed, please enter them
manually.\n");
    prompt_input_coefficient("A", &a);
    prompt_input_coefficient("B", &b);
    prompt_input_coefficient("C", &c);
  }
  printf("\nSolving equation: %.2fx^4 + %.2fx^2 + %.2f = 0\n", a,
b, c);
  // --- Solving Logic ---
  double discriminant = b * b - 4 * a * c;
  if (discriminant < 0) {
    printf("No real roots because the discriminant is negative.
\n");
    return 0;
  }
  double yFirst = (-b + sqrt(discriminant)) / (2 * a);
  double ySecond = (-b - sqrt(discriminant)) / (2 * a);
  double roots[4];
  int count = 0;
  if (yFirst >= 0) {
    double xFirst = sqrt(yFirst);
    if (xFirst == 0) {
      roots[count++] = 0;
    } else {
      roots[count++] = xFirst;
      roots[count++] = -xFirst;
    }
  }
  if (ySecond >= 0) {
    double xSecond = sqrt(ySecond);
    if (yFirst != ySecond) {
      if (xSecond == 0) {
        if (yFirst != 0) {
          roots[count++] = 0;
        }
      } else {
        roots[count++] = xSecond;
        roots[count++] = -xSecond;
      }
    }
  }
  // --- Solving Logic ---

  // --- Output Result ---
  if (count == 0) {
    printf("No real roots found.\n");
  } else {
    printf("Found real roots:\n");
    for (int i = 0; i < count; i++) {
      printf("%.2f ", roots[i]);
```

```
    }
    printf("\n");
  }
  // --- Output Result ---

  return 0;
}
```

Test procedural.py

```
aleksejgalenko@MacBook-Pro-Aleksej-3 Lab_1 % python3 procedural.py 3 2 1
equation: 3.0x^4 + 2.0x^2 + 1.0 = 0
no real roots
```

```
aleksejgalenko@MacBook-Pro-Aleksej-3 Lab_1 % python3 procedural.py 3 -4 1
equation: 3.0x^4 + -4.0x^2 + 1.0 = 0
real roots: [-1.0, -0.5773502691896257, 0.5773502691896257, 1.0]
```

```
aleksejgalenko@MacBook-Pro-Aleksej-3 Lab_1 % python3 procedural.py 3
write coefficient B: -4
write coefficient C: 1
equation: 3.0x^4 + -4.0x^2 + 1.0 = 0
real roots: [-1.0, -0.5773502691896257, 0.5773502691896257, 1.0]
```

```
aleksejgalenko@MacBook-Pro-Aleksej-3 Lab_1 % python3 procedural.py
write coefficient A: 3
write coefficient B: -2
write coefficient C: 1
equation: 3.0x^4 + -2.0x^2 + 1.0 = 0
no real roots
```

Test object.c

```
aleksejgalenko@MacBook-Pro-Aleksej-3 Lab_1 % python3 object.py 3 -4 1
equation: 3.0x^4 + -4.0x^2 + 1.0 = 0
real roots: [-1.0, -0.5773502691896257, 0.5773502691896257, 1.0]
```

```
aleksejgalenko@MacBook-Pro-Aleksej-3 Lab_1 % python3 object.py 3 4 1
equation: 3.0x^4 + 4.0x^2 + 1.0 = 0
no real roots
```

```
aleksejgalenko@MacBook-Pro-Aleksej-3 Lab_1 % python3 object.py 3
write coefficient B: -4
write coefficient C: 1
equation: 3.0x^4 + -4.0x^2 + 1.0 = 0
real roots: [-1.0, -0.5773502691896257, 0.5773502691896257, 1.0]
```

```
aleksejgalenko@MacBook-Pro-Aleksej-3 Lab_1 % python3 object.py
write coefficient A: 3
write coefficient B: -6
write coefficient C: 2
equation: 3.0x^4 + -6.0x^2 + 2.0 = 0
real roots: [-1.2559260603991087, -0.6501151673437363, 0.6501151673437363, 1.2559260603991087]
```

Test procedural.c

```
aleksejgalenko@MacBook-Pro-Aleksej-3 Lab_1 % ./procedural
No coefficients were passed, please enter them manually.
Enter coefficient A: 3
Enter coefficient B: -4
Enter coefficient C: 1

Solving equation: 3.00x^4 + -4.00x^2 + 1.00 = 0
Found real roots:
1.00 -1.00 0.58 -0.58
```

```
aleksejgalenko@MacBook-Pro-Aleksej-3 Lab_1 % ./procedural 3 -4 1
Coefficients taken from command-line arguments.

Solving equation: 3.00x^4 + -4.00x^2 + 1.00 = 0
Found real roots:
1.00 -1.00 0.58 -0.58
```

```
aleksejgalenko@MacBook-Pro-Aleksej-3 Lab_1 % ./procedural 3
No coefficients were passed, please enter them manually.
Enter coefficient A: 3
Enter coefficient B: -4
Enter coefficient C: 1

Solving equation: 3.00x^4 + -4.00x^2 + 1.00 = 0
Found real roots:
1.00 -1.00 0.58 -0.58
```

```
aleksejgalenko@MacBook-Pro-Aleksej-3 Lab_1 % ./procedural
No coefficients were passed, please enter them manually.
Enter coefficient A: 2
Enter coefficient B: 2
Enter coefficient C: 2

Solving equation: 2.00x^4 + 2.00x^2 + 2.00 = 0
No real roots because the discriminant is negative.
```