

ОТЧЕТ

По лр4

Дисциплина «Парадигмы и конструкции языков программирования»

Студент: Коваленко Е. Ю.

Группа: ИБМ3-34Б

Для выполнения лабораторной работы был взят код из ЛР1.

Модифицированный код программы с модульными тестами:

```
import sys
import math
import unittest

class BiquadraticSolver:

    @staticmethod
    def solve(a, b, c):
        if a == 0:
            raise ValueError("Коэффициент А не может быть 0")

        discr = b**2 - 4*a*c

        if discr < 0:
            return []

        sqrt_discr = math.sqrt(discr)
        y1 = (-b + sqrt_discr) / (2*a)
        y2 = (-b - sqrt_discr) / (2*a)

        roots = []

        for y in [y1, y2]:
            if y < 0:
                continue

            if abs(y) < 1e-10:
                roots.append(0.0)

            else: # y > 0
                sqrt_y = math.sqrt(y)
                roots.append(sqrt_y)
                roots.append(-sqrt_y)

        return sorted(set(round(root, 10) for root in roots))

    @staticmethod
    def get_coefficient(name, arg=None):
        while True:
            if arg is not None:
```

```

        value = arg

        arg = None

    else:

        value = input(f"Введите коэффициент {name}: ")

    try:

        return float(value)

    except ValueError:

        print(f"Ошибка: {name} должно быть числом")

        if arg is not None:

            arg = None

def run_interactive():

    solver = BiquadraticSolver()

    print("\n" + "="*50)

    print("РЕШЕНИЕ БИКВАДРАТНОГО УРАВНЕНИЯ")

    print("="*50)

    a = solver.get_coefficient('A')

    b = solver.get_coefficient('B')

    c = solver.get_coefficient('C')

    while a == 0:

        print("Ошибка: коэффициент A не может быть 0")

        a = solver.get_coefficient('A')

    roots = solver.solve(a, b, c)

    print("\n" + "-"*50)

    if not roots:

        print("Действительных корней нет")

    else:

        print(f"Найдено корней: {len(roots)}")

        for i, root in enumerate(roots, 1):

            print(f"Корень {i}: {root}")

    print("="*50)

```

TDD ТЕСТЫ

```
class TestBiquadraticSolver(unittest.TestCase):
```

```
    def test_1_four_roots(self):
```

```
        solver = BiquadraticSolver()
```

```
        roots = solver.solve(1, -13, 36)
```

```
        expected = [-3.0, -2.0, 2.0, 3.0]
```

```
        self.assertEqual(len(roots), 4)
```

```
        for r, e in zip(roots, expected):
```

```
            self.assertAlmostEqual(r, e, places=5)
```

```
        print("✓ TDD тест 1 пройден: 4 корня найдены правильно")
```

```
    def test_2_no_roots(self):
```

```
        solver = BiquadraticSolver()
```

```
        roots = solver.solve(1, 1, 1)
```

```
        self.assertEqual(roots, [])
```

```
        print("✓ TDD тест 2 пройден: корни отсутствуют как и ожидалось")
```

```
    def test_3_zero_a_error(self):
```

```
        solver = BiquadraticSolver()
```

```
        with self.assertRaises(ValueError):
```

```
            solver.solve(0, 2, 3)
```

```
        print("✓ TDD тест 3 пройден: ошибка при A=0 обработана правильно")
```

BDD ТЕСТЫ

```
def run_bdd_tests():
```

```
    print("\n" + "="*50)
```

```
    print("BDD ТЕСТЫ (Behavior-Driven Development)")
```

```
    print("="*50)
```

```
    solver = BiquadraticSolver()
```

```
tests_passed = 0
```

```
total_tests = 3
```

```
print("\nСценарий 1: Решение уравнения с действительными корнями")
```

```
print("Дано: уравнение  $x^4 - 5x^2 + 4 = 0$ ")
```

```
print("Когда: вызываем метод solve(1, -5, 4)")
```

```
print("Тогда: должны получить 4 корня")
```

```
roots = solver.solve(1, -5, 4)
```

```
if len(roots) == 4:
```

```
    print(f"✓ УСПЕХ: найдено {len(roots)} корней: {roots}")
```

```
    tests_passed += 1
```

```
else:
```

```
    print(f"✗ ОШИБКА: ожидалось 4 корня, получено {len(roots)}")
```

```
print("\nСценарий 2: Решение уравнения без действительных корней")
```

```
print("Дано: уравнение  $x^4 + 2x^2 + 5 = 0$ ")
```

```
print("Когда: вызываем метод solve(1, 2, 5)")
```

```
print("Тогда: не должно быть действительных корней")
```

```
roots = solver.solve(1, 2, 5)
```

```
if not roots:
```

```
    print("✓ УСПЕХ: корней нет (ожидаемый результат)")
```

```
    tests_passed += 1
```

```
else:
```

```
    print(f"✗ ОШИБКА: ожидалось 0 корней, получено {len(roots)}")
```

```
print("\nСценарий 3: Решение уравнения с одним корнем")
```

```
print("Дано: уравнение  $x^4 = 0$ ")
```

```
print("Когда: вызываем метод solve(1, 0, 0)")
```

```
print("Тогда: должен быть один корень  $x = 0$ ")
```

```

roots = solver.solve(1, 0, 0)
if roots == [0.0]:
    print("✓ УСПЕХ: найден один корень x = 0")
    tests_passed += 1
else:
    print(f"✗ ОШИБКА: ожидался корень [0.0], получено {roots}")

print("\n" + "-"*50)
print(f"BDD тесты пройдены: {tests_passed}/{total_tests}")
print("="*50)

return tests_passed == total_tests

```

```

def run_all_tests():
    print("\n" + "="*60)
    print("ЗАПУСК ВСЕХ ТЕСТОВ")
    print("="*60)

    print("\n1. TDD ТЕСТЫ (Test-Driven Development):")
    print("-"*40)

    suite = unittest.TestLoader().loadTestsFromTestCase(TestBiquadraticSolver)
    runner = unittest.TextTestRunner(verbosity=0, stream=sys.stdout)
    tdd_result = runner.run(suite)
    tdd_passed = tdd_result.wasSuccessful()

    print("\n2. BDD ТЕСТЫ (Behavior-Driven Development):")
    print("-"*40)

    bdd_passed = run_bdd_tests()

    print("\n" + "="*60)

```

```
print("ИТОГИ ТЕСТИРОВАНИЯ:")
print(f"TDD тесты: {'ПРОЙДЕНЫ' if tdd_passed else 'НЕ ПРОЙДЕНЫ'}")
print(f"BDD тесты: {'ПРОЙДЕНЫ' if bdd_passed else 'НЕ ПРОЙДЕНЫ'}")
```

```
if tdd_passed and bdd_passed:
    print("✓ ВСЕ ТЕСТЫ ПРОЙДЕНЫ УСПЕШНО!")
else:
    print("X НЕКОТОРЫЕ ТЕСТЫ НЕ ПРОЙДЕНЫ")
```

```
print("="*60)
```

```
return tdd_passed and bdd_passed
```

```
def run_tdd_tests_only():
```

```
    print("\n" + "="*50)
    print("ЗАПУСК TDD ТЕСТОВ")
    print("="*50)
```

```
    suite = unittest.TestLoader().loadTestsFromTestCase(TestBiquadraticSolver)
    runner = unittest.TextTestRunner(verbosity=2)
    result = runner.run(suite)
```

```
    print("\n" + "-"*50)
    print(f"TDD тесты: {'ПРОЙДЕНЫ' if result.wasSuccessful() else 'НЕ ПРОЙДЕНЫ'}")
    print("="*50)
```

```
    return result.wasSuccessful()
```

```
def run_bdd_tests_only():
```

```
    print("\n" + "="*50)
```

```
print("ЗАПУСК BDD ТЕСТОВ")
```

```
print("="*50)
```

```
result = run_bdd_tests()
```

```
print("\n" + "-"*50)
```

```
print(f"BDD тесты: {'ПРОЙДЕНЫ' if result else 'НЕ ПРОЙДЕНЫ'}")
```

```
print("="*50)
```

```
return result
```

```
def show_menu():
```

```
    print("\n" + "="*60)
```

```
    print("РЕШЕНИЕ БИКВАДРАТНЫХ УРАВНЕНИЙ И ТЕСТИРОВАНИЕ")
```

```
    print("="*60)
```

```
    print("1. Решить биквадратное уравнение (интерактивный режим)")
```

```
    print("2. Запустить ВСЕ тесты (TDD + BDD)")
```

```
    print("3. Запустить только TDD тесты (3 теста)")
```

```
    print("4. Запустить только BDD тесты (3 сценария)")
```

```
    print("5. Запустить быстрый демо-тест")
```

```
    print("0. Выход")
```

```
    print("="*60)
```

```
while True:
```

```
    try:
```

```
        choice = input("\nВыберите действие (0-5): ")
```

```
        if choice in ['0', '1', '2', '3', '4', '5']:
```

```
            return int(choice)
```

```
        else:
```

```
            print("Ошибка: введите число от 0 до 5")
```

```
    except ValueError:
```



```
print("Ошибка: введите число от 0 до 5")
```

```
def quick_demo():
```

```
    print("\n" + "="*50)
```

```
    print("БЫСТРАЯ ДЕМОНСТРАЦИЯ")
```

```
    print("="*50)
```

```
    solver = BiquadraticSolver()
```

```
    print("\nПример 1:  $x^4 - 13x^2 + 36 = 0$ ")
```

```
    print("Ожидается: 4 корня (-3, -2, 2, 3)")
```

```
    roots = solver.solve(1, -13, 36)
```

```
    print(f"Получено: {roots}")
```

```
    print("\nПример 2:  $x^4 + x^2 + 1 = 0$ ")
```

```
    print("Ожидается: нет действительных корней")
```

```
    roots = solver.solve(1, 1, 1)
```

```
    print(f"Получено: {roots}")
```

```
    print("\nПример 3:  $x^4 = 0$ ")
```

```
    print("Ожидается: один корень  $x = 0$ ")
```

```
    roots = solver.solve(1, 0, 0)
```

```
    print(f"Получено: {roots}")
```

```
    print("\n" + "="*50)
```

```
    print("ДЕМОНСТРАЦИЯ ЗАВЕРШЕНА")
```

```
    print("="*50)
```

```
def main():
```

```
if len(sys.argv) > 1:
    if sys.argv[1] == "--test":
        run_all_tests()
    elif sys.argv[1] == "--tdd":
        run_tdd_tests_only()
    elif sys.argv[1] == "--bdd":
        run_bdd_tests_only()
    elif sys.argv[1] == "--demo":
        quick_demo()
    elif sys.argv[1] == "--interactive":
        run_interactive()
    elif len(sys.argv) >= 4:
        try:
            solver = BiquadraticSolver()
            a = float(sys.argv[1])
            b = float(sys.argv[2])
            c = float(sys.argv[3])

            if a == 0:
                print("Ошибка: коэффициент А не может быть 0")
                return

            roots = solver.solve(a, b, c)

            if not roots:
                print("Действительных корней нет")
            else:
                print(f"Найдено корней: {len(roots)}")
                for root in roots:
                    print(root)
        except ValueError:
            print("Ошибка: коэффициенты должны быть числами")
```

else:

```
print("Использование:")  
  
print(" python biquadratic_solver_easy_test.py --test    # Запуск всех тестов")  
print(" python biquadratic_solver_easy_test.py --tdd     # Запуск TDD тестов")  
print(" python biquadratic_solver_easy_test.py --bdd     # Запуск BDD тестов")  
print(" python biquadratic_solver_easy_test.py --demo    # Быстрая демонстрация")  
print(" python biquadratic_solver_easy_test.py --interactive # Интерактивный режим")  
print(" python biquadratic_solver_easy_test.py A B C     # Решить уравнение")  
print(" python biquadratic_solver_easy_test.py          # Меню выбора")
```

else:

while True:

choice = show_menu()

if choice == 0:

print("\nВыход из программы. До свидания!")

break

elif choice == 1:

run_interactive()

elif choice == 2:

run_all_tests()

elif choice == 3:

run_tdd_tests_only()

elif choice == 4:

run_bdd_tests_only()

elif choice == 5:

quick_demo()

input("\nНажмите Enter для продолжения...")

if __name__ == "__main__":

main()

```
===== RESTART: C:\Users\User\Desktop\лр4.py =====

=====
РЕШЕНИЕ БИКВАДРАТНЫХ УРАВНЕНИЙ И ТЕСТИРОВАНИЕ
=====
1. Решить биквадратное уравнение (интерактивный режим)
2. Запустить ВСЕ тесты (TDD + BDD)
3. Запустить только TDD тесты (3 теста)
4. Запустить только BDD тесты (3 сценария)
5. Запустить быстрый демо-тест
0. Выход
=====

Выберите действие (0-5): 1

=====
РЕШЕНИЕ БИКВАДРАТНОГО УРАВНЕНИЯ
=====
Введите коэффициент A: 2
Введите коэффициент B: 4
Введите коэффициент C: 6

-----
Действительных корней нет
=====

Нажмите Enter для продолжения...|

Выберите действие (0-5): 3

=====
ЗАПУСК TDD ТЕСТОВ
=====
test_1_four_roots (__main__.TestBiquadraticSolver.test_1_four_roots) ... ✓ TDD
тест 1 пройден: 4 корня найдены правильно
ok
test_2_no_roots (__main__.TestBiquadraticSolver.test_2_no_roots) ... ✓ TDD тест
2 пройден: корни отсутствуют как и ожидалось
ok
test_3_zero_a_error (__main__.TestBiquadraticSolver.test_3_zero_a_error) ... ✓
TDD тест 3 пройден: ошибка при A=0 обработана правильно
ok

-----
Ran 3 tests in 0.091s

OK

-----
TDD тесты: ПРОЙДЕНЫ
=====

Нажмите Enter для продолжения...|
```

Выберите действие (0-5): 4

```
=====
ЗАПУСК BDD ТЕСТОВ
=====
```

```
=====
BDD ТЕСТЫ (Behavior-Driven Development)
=====
```

Сценарий 1: Решение уравнения с действительными корнями
Дано: уравнение $x^4 - 5x^2 + 4 = 0$
Когда: вызываем метод `solve(1, -5, 4)`
Тогда: должны получить 4 корня
✓ УСПЕХ: найдено 4 корней: `[-2.0, -1.0, 1.0, 2.0]`

Сценарий 2: Решение уравнения без действительных корней
Дано: уравнение $x^4 + 2x^2 + 5 = 0$
Когда: вызываем метод `solve(1, 2, 5)`
Тогда: не должно быть действительных корней
✓ УСПЕХ: корней нет (ожидаемый результат)

Сценарий 3: Решение уравнения с одним корнем
Дано: уравнение $x^4 = 0$
Когда: вызываем метод `solve(1, 0, 0)`
Тогда: должен быть один корень $x = 0$
✓ УСПЕХ: найден один корень $x = 0$

```
-----
BDD тесты пройдены: 3/3
=====
```

```
-----
BDD тесты: ПРОЙДЕНЫ
=====
```

Нажмите Enter для продолжения...