Код программы (процедурный подход решения):

```
import sys
import math


def get_coefficient(prompt):
    while True:
        try:
            value = float(input(prompt))
            return value
        except ValueError:
            print("Некорректное значение. Попробуйте еще раз.")


def calculate_D(a, b, c):
    return b ** 2 - 4 * a * c


def solve_biquadratic_equation(a, b, c):
    if a == 0:
        print("Коэффициент A не может быть равен нулю для биквадратного
уравнения.")
        return

    D = calculate_D(a, b, c)

    if D < 0:
        print("Действительных корней нет.")
        return

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

    roots = []
    if y1 >= 0:
        roots.extend([math.sqrt(y1), -math.sqrt(y1)])
    if y2 >= 0 and y2 != y1:
        roots.extend([math.sqrt(y2), -math.sqrt(y2)])

    if roots:
        print("Действительные корни:")
        for root in roots:
            print(f"{root:.4f}")
    else:
        print("Действительных корней нет.")


def main():
    args = sys.argv[1:]

    if len(args) >= 3:
        try:
            a = float(args[0])
        except ValueError:
            a = get_coefficient("Введите коэффициент A: ")
        try:
            b = float(args[1])
        except ValueError:
            b = get_coefficient("Введите коэффициент B: ")
        try:
            c = float(args[2])
        except ValueError:
            c = get_coefficient("Введите коэффициент C: ")
    else:
```

```
        a = get_coefficient("Введите коэффициент A: ")
        b = get_coefficient("Введите коэффициент B: ")
        c = get_coefficient("Введите коэффициент C: ")

    solve_biquadratic_equation(a, b, c)


if __name__ == "__main__":
    main()
```

Результат:

```
Введите коэффициент A: 1
Введите коэффициент B: -5
Введите коэффициент C: 4
Действительные корни:
2.0000
-2.0000
1.0000
-1.0000


Process finished with exit code 0
```

Код программы (ООП):

```
import sys
import math

class BiquadraticEquationSolver:
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c

    @staticmethod
    def get_coefficient(prompt):
        while True:
            try:
                value = float(input(prompt))
                return value
            except ValueError:
                print("Некорректное значение. Попробуйте еще раз.")

    def calculate_discriminant(self):
        return self.b ** 2 - 4 * self.a * self.c

    def solve(self):
```

```
        if self.a == 0:
            print("Коэффициент A не может быть равен нулю для биквадратного
уравнения.")
            return []

        D = self.calculate_discriminant()

        if D < 0:
            print("Действительных корней нет.")
            return []

        y1 = (-self.b + math.sqrt(D)) / (2 * self.a)
        y2 = (-self.b - math.sqrt(D)) / (2 * self.a)

        roots = []
        if y1 >= 0:
            roots.extend([math.sqrt(y1), -math.sqrt(y1)])
        if y2 >= 0 and y2 != y1:
            roots.extend([math.sqrt(y2), -math.sqrt(y2)])

        return roots

    def display_roots(self):
        roots = self.solve()
        if roots:
            print("Действительные корни:")
            for root in roots:
                print(f"{root:.4f}")
        else:
            print("Действительных корней нет.")

class Application:
    @staticmethod
    def get_coefficients_from_args_or_input(args):
        if len(args) >= 3:
            try:
                a = float(args[0])
            except ValueError:
                a = BiquadraticEquationSolver.get_coefficient("Введите
коэффициент A: ")
            try:
                b = float(args[1])
            except ValueError:
                b = BiquadraticEquationSolver.get_coefficient("Введите
коэффициент B: ")
            try:
                c = float(args[2])
            except ValueError:
                c = BiquadraticEquationSolver.get_coefficient("Введите
коэффициент C: ")
        else:
            a = BiquadraticEquationSolver.get_coefficient("Введите
коэффициент A: ")
            b = BiquadraticEquationSolver.get_coefficient("Введите
коэффициент B: ")
            c = BiquadraticEquationSolver.get_coefficient("Введите
коэффициент C: ")

        return a, b, c

    @staticmethod
    def main():
        args = sys.argv[1:]
        a, b, c = Application.get_coefficients_from_args_or_input(args)
```

```
        solver = BiquadraticEquationSolver(a, b, c)
        solver.display_roots()

if __name__ == "__main__":
    Application.main()
```

Результат:

```
Введите коэффициент A: 1
Введите коэффициент B: -5
Введите коэффициент C: 4
Действительные корни:
2.0000
-2.0000
1.0000
-1.0000


Process finished with exit code 0
```

Решение на языке Go:

```
package main

import (
    "errors"
    "fmt"
    "math"
    "os"
    "strconv"
)

type BiquadraticEquation struct {
    A, B, C float64
}

func (eq *BiquadraticEquation) calculateD() float64 {
    return eq.B*eq.B - 4*eq.A*eq.C
}

func (eq *BiquadraticEquation) Solve() {
    if eq.A == 0 {
        fmt.Println("Коэффициент A не может быть равен нулю для биквадратного
уравнения")
        return
    }

    D := eq.calculateD()
```

```
    if D < 0 {
        fmt.Println("Действительных корней нет.")
        return
    }

    // Решаем квадратное уравнение Ay^2 + By + C = 0
    y1 := (-eq.B + math.Sqrt(D)) / (2 * eq.A)
    y2 := (-eq.B - math.Sqrt(D)) / (2 * eq.A)

    roots := []float64{}

    if y1 >= 0 {
        roots = append(roots, math.Sqrt(y1), -math.Sqrt(y1))
    }
    if y2 >= 0 && y2 != y1 {
        roots = append(roots, math.Sqrt(y2), -math.Sqrt(y2))
    }

    if len(roots) == 0 {
        fmt.Println("Действительных корней нет.")
    } else {
        fmt.Println("Действительные корни:")
        for _, root := range roots {
            fmt.Printf("%.4f\n", root)
        }
    }
}

func getCoefficient(input string, prompt string) (float64, error) {
    if input == "" {
        fmt.Print(prompt)
        fmt.Scanln(&input)
    }
    value, err := strconv.ParseFloat(input, 64)
    if err != nil {
        return 0, errors.New("некорректное значение, повторите ввод")
    }
    return value, nil
}

func initializeEquation(args []string) *BiquadraticEquation {
    var a, b, c float64
    var err error

    if len(args) > 0 {
        a, err = getCoefficient(args[0], "Введите коэффициент A: ")
    } else {
        a, err = getCoefficient("", "Введите коэффициент A: ")
    }
    for err != nil {
        fmt.Println(err)
        a, err = getCoefficient("", "Введите коэффициент A: ")
    }

    if len(args) > 1 {
        b, err = getCoefficient(args[1], "Введите коэффициент B: ")
    } else {
        b, err = getCoefficient("", "Введите коэффициент B: ")
    }
    for err != nil {
        fmt.Println(err)
        b, err = getCoefficient("", "Введите коэффициент B: ")
    }
```

```
    if len(args) > 2 {
        c, err = getCoefficient(args[2], "Введите коэффициент C: ")
    } else {
        c, err = getCoefficient("", "Введите коэффициент C: ")
    }
    for err != nil {
        fmt.Println(err)
        c, err = getCoefficient("", "Введите коэффициент C: ")
    }

    return &BiquadraticEquation{A: a, B: b, C: c}
}

func main() {
    args := os.Args[1:]
    equation := initializeEquation(args)
    equation.Solve()
}
```

Результат:

```
<4 go setup calls>
Введите коэффициент A: 1
Введите коэффициент B: -5
Введите коэффициент C: 4
Действительные корни:
2.0000
-2.0000
1.0000
-1.0000


Process finished with the exit code 0
```