Алгоритмизация и программирование

2. Ветвления

Глухих Михаил Игоревич
mailto: glukhikh@mail.ru

Ветвление

- Участок программы, выполняющий те или иные действия в зависимости от исходных данных (условия или ключа)

Простейший случай

```
fun max(m: Int, n: Int) = if (m > n) m else n
```

Максимум из двух

```
fun max(m: Int, n: Int) = if (m > n) m else n

// if..else.. – оператор ветвления
```

Максимум из двух

```
fun max(m: Int, n: Int) = if (m > n) m else n

// if..else.. – оператор ветвления
// m > n      – условие ветвления
```

Максимум из двух

```
fun max(m: Int, n: Int) = if (m > n) m else n

// if..else.. – оператор ветвления
// m > n      – условие ветвления
// if .. m    – основная ветвь
```

Максимум из двух

```
fun max(m: Int, n: Int) = if (m > n) m else n

// if..else.. – оператор ветвления
// m > n      – условие ветвления
// if .. m    – основная ветвь
// else .. n  – ветвь «иначе»
```

Каскадный if

```
fun quadraticRootNumber(
    a: Double, b: Double, c: Double
): Int {
    val d = discriminant(a, b, c)
    return if (d > 0.0) 2 else if (d == 0.0) 1 else 0
}
```

Число корней квадратного уравнения

```
// ax²+bx+c = 0
fun quadraticRootNumber(
    a: Double, b: Double, c: Double
): Int {
    val d = discriminant(a, b, c)
    return if (d > 0.0) 2 else if (d == 0.0) 1 else 0
}

// return if .. – вернуть результат if
```

Число корней квадратного уравнения

```
// ax²+bx+c = 0
fun quadraticRootNumber(
    a: Double, b: Double, c: Double
): Int {
    val d = discriminant(a, b, c)
    return if (d > 0.0) 2 else if (d == 0.0) 1 else 0
}

// return if .. – вернуть результат if
// if .. else if .. else – каскадный if
```

Операции в Котлине (сравнения)

- На больше / меньше
  - a <= b
  - a < b
  - a >= b
  - a > b
- На равенство: a == b
- На неравенство: a != b
- На включение: x in a..b
- На невключение: x !in a..b

Операции в Котлине (логические)

- Служат для проверки сложной комбинации условий

- И: condition1 && condition2
- ИЛИ: condition1 || condition2
- НЕ: !condition

Табличная форма ветвлений

```
fun quadraticRootNumber(
    a: Double, b: Double, c: Double
): Int {
    val d = discriminant(a, b, c)
    return when {
        d > 0.0  -> 2
        d == 0.0 -> 1
        else     -> 0
    }
}
```

Табличное ветвление по ключу

```
fun gradeNotation(grade: Int): String = when (grade) {
    5 -> "отлично"
    4 -> "хорошо"
    3 -> "удовлетворительно"
    2 -> "неудовлетворительно"
    else -> "несуществующая оценка $grade"
}
```

Табличное ветвление по ключу

```
fun gradeNotation(grade: Int): String = when (grade) {
    5 -> "отлично"
    4 -> "хорошо"
    3 -> "удовлетворительно"
    2 -> "неудовлетворительно"
    else -> "несуществующая оценка $grade"
}

// when (grade) { ... }
// Здесь grade – ключ или субъект ветвления
```

Цепочки if – биквадратное уравнение

- $ax^4 + bx^2 + c = 0$

Цепочки if – биквадратное уравнение

- $ax^4 + bx^2 + c = 0$
- $x^2 = (-b \pm sqrt(D)) / 2a$

Цепочки if – биквадратное уравнение

- $ax^4 + bx^2 + c = 0$
- $x^2 = (-b \pm \text{sqrt}(D)) / 2a$

- Рассмотрим алгоритм решения
- Упрощённая форма – ищем только **минимальный** корень из имеющихся

Цепочки if – биквадратное уравнение

- $ax^4 + bx^2 + c = 0$
- $x^2 = (-b \pm \text{sqrt}(D)) / 2a$

1. $a = 0 \rightarrow x = \pm \text{sqrt}(-c/b)$

Цепочки if – биквадратное уравнение

```
fun minBiRoot(a: Double, b: Double, c: Double): Double {
    if (a == 0.0) {
        if (b == 0.0) return Double.NaN
        val bc = -c / b
        if (bc < 0.0) return Double.NaN
        return -Math.sqrt(bc)
    }
}
```

Цепочки if – биквадратное уравнение

- $ax^4 + bx^2 + c = 0$
- $x^2 = (-b \pm sqrt(D)) / 2a$

1. $a = 0 \rightarrow x = \pm sqrt(-c/b)$
2. $D = b^2 - 4ac$

Цепочки if – биквадратное уравнение

```
fun minBiRoot(a: Double, b: Double, c: Double): Double {
    if (a == 0.0) {
        if (b == 0.0) return Double.NaN
        val bc = -c / b
        if (bc < 0.0) return Double.NaN
        return -Math.sqrt(bc)
    }
    val d = discriminant(a, b, c)
}
```

Цепочки if – биквадратное уравнение

- $ax^4 + bx^2 + c = 0$
- $x^2 = (-b \pm \text{sqrt}(D)) / 2a$

1. $a = 0 \rightarrow x = \pm \text{sqrt}(-c/b)$
2. $D = b^2 - 4ac$
3. $D < 0$?

Цепочки if – биквадратное уравнение

```
fun minBiRoot(a: Double, b: Double, c: Double): Double {
    if (a == 0.0) {
        if (b == 0.0) return Double.NaN
        val bc = -c / b
        if (bc < 0.0) return Double.NaN
        return -Math.sqrt(bc)
    }
    val d = discriminant(a, b, c)
    if (d < 0.0) return Double.NaN
}
```

Цепочки if – биквадратное уравнение

- $ax^4 + bx^2 + c = 0$
- $x^2 = (-b \pm sqrt(D)) / 2a$

1. $a = 0 \rightarrow x = \pm sqrt(-c/b)$
2. $D = b^2 - 4ac$
3. $D < 0$?
4. $y_{1,2} = (-b \pm sqrt(D)) / 2a$

Цепочки if – биквадратное уравнение

```
fun minBiRoot(a: Double, b: Double, c: Double): Double {
    if (a == 0.0) {
        if (b == 0.0) return Double.NaN
        val bc = -c / b
        if (bc < 0.0) return Double.NaN
        return -Math.sqrt(bc)
    }
    val d = discriminant(a, b, c)
    if (d < 0.0) return Double.NaN
    val y1 = (-b + Math.sqrt(d)) / (2 * a)
    val y2 = (-b - Math.sqrt(d)) / (2 * a)
}
```

Цепочки if – биквадратное уравнение

- $ax^4 + bx^2 + c = 0$
- $x^2 = (-b \pm \text{sqrt}(D)) / 2a$

1. $a = 0 \rightarrow x = \pm \text{sqrt}(-c/b)$
2. $D = b^2 - 4ac$
3. $D < 0$?
4. $y_{1,2} = (-b \pm \text{sqrt}(D)) / 2a$
5. $y_3 = \max(y_1, y_2)$

Цепочки if – биквадратное уравнение

```
fun minBiRoot(a: Double, b: Double, c: Double): Double {
    if (a == 0.0) {
        if (b == 0.0) return Double.NaN
        val bc = -c / b
        if (bc < 0.0) return Double.NaN
        return -Math.sqrt(bc)
    }
    val d = discriminant(a, b, c)
    if (d < 0.0) return Double.NaN
    val y1 = (-b + Math.sqrt(d)) / (2 * a)
    val y2 = (-b - Math.sqrt(d)) / (2 * a)
    val y3 = Math.max(y1, y2)
}
```

Цепочки if – биквадратное уравнение

- $ax^4 + bx^2 + c = 0$
- $x^2 = (-b \pm sqrt(D)) / 2a$

1. $a = 0 \rightarrow x = \pm sqrt(-c/b)$
2. $D = b^2 - 4ac$
3. $D < 0$?
4. $y_{1,2} = (-b \pm sqrt(D)) / 2a$
5. $y_3 = max(y_1, y_2)$
6. $y_3 < 0$?

Цепочки if – биквадратное уравнение

```
fun minBiRoot(a: Double, b: Double, c: Double): Double {
    if (a == 0.0) {
        if (b == 0.0) return Double.NaN
        val bc = -c / b
        if (bc < 0.0) return Double.NaN
        return -Math.sqrt(bc)
    }
    val d = discriminant(a, b, c)
    if (d < 0.0) return Double.NaN
    val y1 = (-b + Math.sqrt(d)) / (2 * a)
    val y2 = (-b - Math.sqrt(d)) / (2 * a)
    val y3 = Math.max(y1, y2)
    if (y3 < 0.0) return Double.NaN
}
```

Цепочки if – биквадратное уравнение

- $ax^4 + bx^2 + c = 0$
- $x^2 = (-b \pm \text{sqrt}(D)) / 2a$

1. $a = 0 \rightarrow x = \pm \text{sqrt}(-c/b)$
2. $D = b^2 - 4ac$
3. $D < 0$?
4. $y_{1,2} = (-b \pm \text{sqrt}(D)) / 2a$
5. $y_3 = \max(y_1, y_2)$
6. $y_3 < 0$?
7. $x = \pm \text{sqrt}(y_3)$

Цепочки if – биквадратное уравнение

```
fun minBiRoot(a: Double, b: Double, c: Double): Double {
    if (a == 0.0) {
        if (b == 0.0) return Double.NaN
        val bc = -c / b
        if (bc < 0.0) return Double.NaN
        return -Math.sqrt(bc)
    }
    val d = discriminant(a, b, c)
    if (d < 0.0) return Double.NaN
    val y1 = (-b + Math.sqrt(d)) / (2 * a)
    val y2 = (-b - Math.sqrt(d)) / (2 * a)
    val y3 = Math.max(y1, y2)
    if (y3 < 0.0) return Double.NaN
    return -Math.sqrt(y3)
}
```

Тесты

- Необходимо проверить все ветви алгоритма

1. $0x^4 + 0x^2 + 1 = 0$
2. $0x^4 + 1x^2 + 2 = 0$
3. $0x^4 + 1x^2 - 4 = 0$
4. $1x^4 - 2x^2 + 4 = 0$
5. $1x^4 + 3x^2 + 2 = 0$
6. $1x^4 - 3x^2 + 2 = 0$

Упражнения к лекции

- См. lesson2/task1 и lesson2/task2 в обучающем проекте
- Решите хотя бы одно из заданий
- Протестируйте решение с помощью готовых тестов
- Добавьте ещё хотя бы один тестовый случай
- Добавьте коммит в свой репозиторий
- Создайте Pull Request и убедитесь в правильности решения