



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«МИРЭА – Российский технологический университет»

**РТУ МИРЭА**

---

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

## **Отчет по Модулю 1 – Основы языка Котлин**

по дисциплине «Проектирования и разработка мобильных приложений на языке  
Котлин»

**Выполнили:**

Студенты группы ИКБО-65-23

Олефиоров Г.Г.

**Проверил:**

Егоров Н.С.

2025 г.

# 1.1. Практическая работа

Сamsung  
Innovation Campus

В начало Личный кабинет Мои курсы

Объявления

Анкета

От авторов

Благодарности

Обозначения в тексте

Правила работы с курсом

Общий форум

Документы

Модуль 1. Основы языка Kotlin

1.1. Введение в язык Kotlin

1.1. Теоретический тест

1.1. Практическая работа

1.2. Функции в Kotlin

1.2. Теоретический тест

1.2. Практическая работа

1.3. Null-безопасность

1.3. Теоретический тест

1.4. Объектно-ориентиров...

1.4. Теоретический тест

1.4. Практическая работа

1.5. Наследование. Интер...

1.5. Теоретический тест

1.5. Практическая работа

Мобильная разработка на Kotlin / 1.1. Практическая работа

1.1. Практическая работа

Тест начат

Состояние

Завершен

Прошло времени

Баллы

Оценка

пятница, 26 сентября 2025, 16:04

Завершены

пятница, 26 сентября 2025, 19:24

3 час. 19 мин.

2,00/2,00

10,00 из 10,00 (100%)

Вопрос 1

Верно

Баллов: 1,00 из 1,00

Отметить вопрос

Дана комната со сторонами a и b и ковер со сторонами m и n. Определить помещается ли ковер в комнату (ковер можно поворачивать, но нельзя складывать). На вход программе подается четыре вещественных числа, разделенных пробелом: a, b, m и n. На выходе необходимо напечатать "YES", если ковер помещается в комнату и "NO" в противном случае.

Правильным по форме, но не по содержанию является ответ вида

```
fun main() {
    println("YES")
}
```

Для примера:

Ввод	Результат
10.0 4.0 7.0 3.0	YES

Ответ: (штрафной режим: 0, 0, 20, ... %)

```
1 fun main() {
2     val input = readLine()!!.split(" ").map { it.toDouble() }
3     val a = input[0]
4     val b = input[1]
5     val m = input[2]
6     val n = input[3]
7
8     val roomMin = minOf(a, b)
9     val roomMax = maxOf(a, b)
10    val carpetMin = minOf(m, n)
11    val carpetMax = maxOf(m, n)
12
13    if (carpetMin <= roomMin && carpetMax <= roomMax) {
14        println("YES")
15    } else {
16        println("NO")
17    }
18 }
```

Навигация по тесту

1

2

Показать одну страницу

Закончить обзор

```
fun main() {
    val input = readLine()!!.split(" ").map { it.toDouble() }
    val a = input[0]
    val b = input[1]
    val m = input[2]
    val n = input[3]

    val roomMin = minOf(a, b)
    val roomMax = maxOf(a, b)
    val carpetMin = minOf(m, n)
    val carpetMax = maxOf(m, n)

    if (carpetMin <= roomMin && carpetMax <= roomMax) {
        println("YES")
    } else {
        println("NO")
    }
}
```

Сamsung  
Innovation Campus

В начало Личный кабинет Мои курсы

2025/26

GO

Баллы за эту попытку: 100/100.

Объявления

○ Анкета

От авторов

Благодарности

Обозначения в тексте

Правила работы с курсом

Общий форум

▼ Документы

▼ Модуль 1. Основы языка Kotlin

1.1. Введение в язык Kotlin

1.1. Теоретический тест

1.1. Практическая работа

1.2. Функции в Kotlin

1.2. Теоретический тест

1.2. Практическая работа

1.3. Null-безопасность

1.3. Теоретический тест

1.4. Объектно-ориентиров...

1.4. Теоретический тест

1.4. Практическая работа

1.5. Наследование. Интер...

1.5. Теоретический тест

1.5. Практическая работа

Вопрос 2

Верно

Баллов: 100 из 100

Отметить вопрос

ДНК состоит из 4 типов нуклеотидов: А (аденин), Т (тимин), G (гуанин), С (цитозин).

Ваша программа получает на вход строку вида ATGCSCTCTCTC и должна посчитать число нуклеотидов каждого типа (вывести числа через пробел в порядке как в строке выше).  
Правильным по форме, но не по содержанию является ответ вида

```
fun main() {  
    println("0 0 0 0")  
}
```

Для примера:

Ввод	Результат
ATGCSCTCTCTC	1 4 1 5

Ответ: (штрафной режим: 10, 20, ... %)

```
1 fun main() {  
2     val DNK = readln()  
3     val countA1 = DNK.count { it == 'A' }  
4     val countA = DNK.split("A").lastIndex  
5     val countT = DNK.split("T").lastIndex  
6     val countG = DNK.split("G").lastIndex  
7     val countC = DNK.split("C").lastIndex  
8     println("$countA1 $countT $countG $countC")  
9  
10 }
```

Ввод	Ожидаемый	Получено
✓ ATGCSCTCTCTC	1 4 1 5	1 4 1 5 ✓

Прошли все тесты! ✓

Навигация по тесту

1

2

Показать одну страницу

Закончить обзор

```
fun main() {  
    val DNK = readln()  
    val countA1 = DNK.count { it == 'A' }  
    val countA = DNK.split("A").lastIndex  
    val countT = DNK.split("T").lastIndex  
    val countG = DNK.split("G").lastIndex  
    val countC = DNK.split("C").lastIndex  
    println("$countA1 $countT $countG $countC")  
}
```

## 1.2. Практическая работа

Сamsung  
Innovation Campus

В начало Личный кабинет Мои курсы

Яндекс

2025/26 ГО

Обозначения в тексте

Правила работы с курсом

Общий форум

Документы

Модуль 1. Основы языка K...

1.1. Введение в язык Kotlin

1.1. Теоретический тест

1.1. Практическая работа

1.2. Функции в Kotlin

1.2. Теоретический тест

1.2. Практическая работа

1.3. Null-безопасность

1.3. Теоретический тест

1.4. Объектно-ориентиров...

1.4. Теоретический тест

1.4. Практическая работа

1.5. Наследование. Интерф...

1.5. Теоретический тест

1.5. Практическая работа

1.6. Обобщенное програм...

1.6. Теоретический тест

Мобильная разработка на Kotlin / 1.2. Практическая работа

1.2. Практическая работа

Тест начат

воскресенье, 28 сентября 2025, 17:30

Состояние

Завершены

Завершен

воскресенье, 28 сентября 2025, 17:31

Прошло времени

1 мин. 1 сек.

Баллы

1,00/1,00

Оценка

10,00 из 10,00 (100%)

Вопрос 1

Верно

Баллов: 1,00 из 1,00

Отметить вопрос

Напишите функцию countVowels подсчитывающую число гласных английских букв (a, e, i, o, u) в строке, у которая передаётся в виде параметра

Ответ: (штрафной режим: 10, 20, ... %)

```
1 fun countVowels(s:String):Int {
2     var res: Int = 0
3     val countA: Int = s.split("a").lastIndex
4     res += countA
5     val countI: Int = s.split("i").lastIndex
6     res += countI
7     val countU: Int = s.split("u").lastIndex
8     res += countU
9     val countE: Int = s.split("e").lastIndex
10    res += countE
11    val countO: Int = s.split("o").lastIndex
12    res += countO
13    return res
14 }
```

Тест	Ожидаемый	Получено

Навигация по тесту

1

Закончить обзор

```
fun countVowels(s:String):Int {
    var res: Int = 0
    val countA: Int = s.split("a").lastIndex
    res += countA
    val countI: Int = s.split("i").lastIndex
    res += countI
    val countU: Int = s.split("u").lastIndex
    res += countU
    val countE: Int = s.split("e").lastIndex
    res += countE
    val countO: Int = s.split("o").lastIndex
    res += countO
    return res
}
```

## 1.4. Практическая работа

Личный кабинет Мои курсы

20/25/26

Баллы 2,00/2,00

Оценка 10,00 из 10,00 (100%)

Вопрос 1

Верно

Баллов: 1,00 из 1,00

Отметить вопрос

Необходимо разработать класс для управления роботом (Robot), содержащий поля координат `x, y` (тип `Int`) и направления `direction`. Для направления уже определён заранее тип `Direction`:

```
enum class Direction {  
    UP, DOWN, RIGHT, LEFT  
}
```

Обратите внимание, что мы используем тип `enum` для указания направления. Для управления роботом определите методы `turnLeft()`, `turnRight()`, `stepForward()`. Конструктор получает параметры (`x, y, direction`). Предусмотрите вывод состояния робота методом `toString()` в виде

`x: $x, y: $y, dir: $direction`

Ответ: (штрафной режим: 10, 20, ... %)

```
1 class Robot(var x: Int, var y: Int, var direction: Direction) {  
2  
3     fun turnLeft() {  
4         direction = when (direction) {  
5             Direction.UP -> Direction.LEFT  
6             Direction.LEFT -> Direction.DOWN  
7             Direction.DOWN -> Direction.RIGHT  
8             Direction.RIGHT -> Direction.UP  
9         }  
10    }  
11  
12    fun turnRight() {  
13        direction = when (direction) {  
14            Direction.UP -> Direction.RIGHT  
15            Direction.RIGHT -> Direction.DOWN  
16            Direction.DOWN -> Direction.LEFT  
17            Direction.LEFT -> Direction.UP  
18        }  
19    }  
20  
21    fun stepForward() {  
22        when (direction) {  
23            Direction.UP -> y++  
24            Direction.DOWN -> y--  
25            Direction.RIGHT -> x++  
26            Direction.LEFT -> x--  
27        }  
28    }  
29  
30    override fun toString(): String {  
31        return "x: $x, y: $y, dir: $direction"  
32    }  
33 }
```

Навигация по тесту

1 2

Показать одну страницу

Закончить обзор

```
class Robot(var x: Int, var y: Int, var direction: Direction) {
```

```
    fun turnLeft() {  
        direction = when (direction) {  
            Direction.UP -> Direction.LEFT  
            Direction.LEFT -> Direction.DOWN  
            Direction.DOWN -> Direction.RIGHT  
            Direction.RIGHT -> Direction.UP  
        }  
    }
```

```
    fun turnRight() {  
        direction = when (direction) {  
            Direction.UP -> Direction.RIGHT  
            Direction.RIGHT -> Direction.DOWN  
            Direction.DOWN -> Direction.LEFT  
            Direction.LEFT -> Direction.UP  
        }  
    }
```

```
    fun stepForward() {  
        when (direction) {  
            Direction.UP -> y++  
            Direction.DOWN -> y--  
            Direction.RIGHT -> x++  
            Direction.LEFT -> x--  
        }  
    }
```

```
    override fun toString(): String {  
        return "x: $x, y: $y, dir: $direction"  
    }
```

```

    }
}fun countVowels(s:String):Int {
    var res: Int = 0
    val countA: Int = s.split("a").lastIndex
    res += countA
    val countI: Int = s.split("i").lastIndex
    res += countI
    val countU: Int = s.split("u").lastIndex
    res += countU
    val countE: Int = s.split("e").lastIndex
    res += countE
    val countO: Int = s.split("o").lastIndex
    res += countO
    return res
}

```

начало Личный кабинет Мои курсы

2025/26 10

Баллы за эту попытку: 1,00/1,00.

## Вопрос 2

Верно

Баллов: 1,00 из 1,00

Отметить вопрос

Используя класс для управления роботом (`Robot`), разработанный в предыдущем задании, реализуйте функцию `moveRobot(r: Robot, toX: Int, toY: Int)`, перемещающую объект типа `Robot` с помощью методов `turnLeft()`, `turnRight()`, `stepForward()` в заданную точку (`toX`, `toY`).

В качестве решения сообщите только код функции `moveRobot`, классы `Direction` и `Robot` уже определены.

Для примера:

Тест	Результат
<pre>val r = Robot(0,1,Direction.UP) moveRobot(r, 3, 7) println("\${r.x} \${r.y}")</pre>	3 7

Ответ: (штрафной режим: 10, 20, ... %)

```

1 fun moveRobot(r: Robot, toX: Int, toY: Int) {
2     val dx: Int = toX - r.x
3     if (dx != 0) {
4         when {
5             (dx > 0) -> turnToDir(r, Direction.RIGHT)
6             (dx < 0) -> turnToDir(r, Direction.LEFT)
7         }
8         repeat(myabs(dx)) { r.stepForward() }
9     }
10    val dy: Int = toY - r.y
11    if (dy != 0) {
12        when {
13            dy > 0 -> turnToDir(r, Direction.UP)
14            dy < 0 -> turnToDir(r, Direction.DOWN)
15        }
16        repeat(myabs(dy)) { r.stepForward() }
17    }
18 }
19
20 private fun myabs(i: Int): Int {
21     return if (i >= 0) i else -i
22 }
23
24 private fun turnToDir(r: Robot, targetDir: Direction) {
25     while (r.direction != targetDir) {
26         r.turnRight()
27     }
28 }

```

## Навигация по тесту



Показать одну страницу

Закончить обзор

```

fun moveRobot(r: Robot, toX: Int, toY: Int) {
    val dx: Int = toX - r.x
    if (dx != 0) {
        when {
            (dx > 0) -> turnToDir(r, Direction.RIGHT)
            (dx < 0) -> turnToDir(r, Direction.LEFT)
        }
        repeat(myabs(dx)) { r.stepForward() }
    }
    val dy: Int = toY - r.y
    if (dy != 0) {
        when {
            dy > 0 -> turnToDir(r, Direction.UP)
            dy < 0 -> turnToDir(r, Direction.DOWN)
        }
        repeat(myabs(dy)) { r.stepForward() }
    }
}

```

```

    }

private fun myabs(i: Int): Int {
    return if (i >= 0) i else -i
}

private fun turnToDir(r: Robot, targetDir: Direction) {
    while (r.direction != targetDir) {
        r.turnRight()
    }
}
}

```

## 1.5. Практическая работа

Samsung Innovation Campus

В начало Личный кабинет Мои курсы

2025/26

Г0

1.1. Практическая работа

1.2. Функции в Kotlin

1.2. Теоретический тест

1.2. Практическая работа

1.3. Null-безопасность

1.3. Теоретический тест

1.4. Объектно-ориентиров...

1.4. Теоретический тест

1.4. Практическая работа

1.5. Наследование. Интер...

1.5. Теоретический тест

1.5. Практическая работа

1.6. Обобщенное програм...

1.6. Теоретический тест

1.7. Множества, списки и а...

1.7. Теоретический тест

Пробный тест по Модулю 1

Контрольный тест по Мод...

Балл преподавателя за М...

Модуль 2. Основы разрабо...

2.1. Первое Android прило...

2.1. Теоретический тест

2.2. Жизненный цикл акти...

Завершен

воскресенье, 28 сентября 2025, 23:55

Прошло времени

26 сек.

Баллы

1,00/1,00

Оценка

10,00 из 10,00 (100%)

Вопрос 1

Верно

Баллов: 1,00 из 1,00

Отметить вопрос

Определены два класса: `Point` (точка с координатами  $x, y$ ) и `CloudOfPoints` (набор точек `Point`).

```

internal class Point(var x: Int, var y: Int)
internal class CloudOfPoints(val points: ArrayList<Point>)

```

Дополните классы новой функциональностью – реализуйте интерфейс `Movable` (метод `move(dx, dy)` добавляет  $dx$  к координате  $x$ , а  $dy$  к координате  $y$ ). Необходимо написать фрагмент программы: полное описание классов с реализованным интерфейсом.

```

interface Movable {
    fun move(dx: Int, dy: Int)
}

// Ваш фрагмент будет помещен здесь

```

Для примера:

Результат

Ответ: (штрафной режим: 10, 20, ... %)

```

1 internal class Point(var x: Int, var y: Int) : Movable {
2     override fun move(dx: Int, dy: Int) {
3         x += dx
4         y += dy
5     }
6 }
7
8 internal class CloudOfPoints(val points: ArrayList<Point>) : Movable {
9     override fun move(dx: Int, dy: Int) {
10         for (point in points) {
11             point.move(dx, dy)
12         }
13     }
14 }

```

Навигация по тесту

1

Закончить обзор

```

internal class Point(var x: Int, var y: Int) : Movable {
    override fun move(dx: Int, dy: Int) {
        x += dx
        y += dy
    }
}

internal class CloudOfPoints(val points: ArrayList<Point>) : Movable {
    override fun move(dx: Int, dy: Int) {
        for (point in points) {
            point.move(dx, dy)
        }
    }
}

```