**Практическая работа по Kotlin 2**

**Работа с циклами.**

[(7) Kotlin. Урок 9. Циклы while, do while, for в котлин - YouTube](https://www.youtube.com/watch?v=fYLOHDtTzPs)

**Теоретический материал**

## **Цикл for**

Цикл for обеспечивает перебор всех значений, поставляемых итератором. Он эквивалентен циклу foreach в таких языках, как C#.

**for** (item **in** collection) print(item)

Телом цикла может быть блок кода.

**for** (item: Int **in** ints) {

*// ...*

}

Как отмечено выше, цикл for позволяет проходить по всем элементам объекта, имеющего итератор, например,

* обладающего внутренней или внешней функцией iterator(), возвращаемый тип которой Iterator<>:
  + обладает внутренней или внешней функцией next()
  + обладает внутренней или внешней функцией hasNext(), возвращающей Boolean.

Все три указанные функции должны быть объявлены как operator.

Чтобы перебрать диапазон чисел, используйте выражение диапазона:

**for** (i **in** 1..3) {

println(i)

}

**for** (i **in** 6 downTo 0 step 2) {

println(i)

}

fun main() {

    val count = readLine().toString.toInt

    var sum = 0

    for (i in 1..count) {

        sum += i

    }

    print(sum)

}

Цикл for по диапазону или массиву компилируется в основанный на индексе цикл, который не создает объект итератора.

Если при проходе по массиву или списку необходим порядковый номер элемента, используйте следующий подход:

**for** (i **in** array.indices) {

println(array[i])

}

Также вы можете использовать библиотечную функцию withIndex.

**for** ((index, value) **in** array.withIndex()) {

println(**"the element at $index is $value"**)

}

## **Цикл while**

Тело циклов while и do-while выполняется до тех пор, пока их условие выполняется. Разница между ними заключается во времени проверки условия:

* while проверяет условие и, если оно истинно, выполняет тело, а затем возвращается к проверке условия;
* do-while выполняет тело и только затем проверяет условие. Если оно выполняется, цикл повторяется. Таким образом, тело do-while выполняется по крайней мере один раз независимо от условия.

**while** (x > 0) {

x--

}

**do** {

**val** y = retrieveData()

} **while** (y != **null**) *// y здесь доступно!*

## 

var sum = 1

while (sum < 1000) {

    sum = sum + (sum + 1)

}

## **Break и continue в циклах**

Kotlin поддерживает привычные операторы break и continue в циклах.

|  |
| --- |
| sum = 1    while (true) {      sum = sum + (sum + 1)      if (sum >= 1000) {          break      }  } |

Здесь условие цикла истинно (true), поэтому цикл будет повторятся бесконечно. Однако **оператор break** указывает на то, что цикл while завершится, когда сумма будет больше или равна 1000.

fun main() {

    var sum = 0

    for (row in 0 until 8) {

        if (row % 2 == 0) {

            continue

        }

         for (column in 0 until 8) {

            sum += row \* column

        }

    }     print("Результат: $sum")

}///Строка четная если результат деления номера строки на 2 равен 0. В этом случае continue заставляет **цикл for** переходить к следующей строке.

**Задания:**

1. Вывести на экран числа от 1 до 5.
2. Подсчитать сумму математической последовательности до того момента, пока значение не больше 1000. Сумму вывести на экран.
3. Напечатайте те натуральные числа, квадрат которых не превышает значения *n*.
4. Используя цикл while вычислить сумму чисел последовательности, заданную формулой

2 + 5 + ... + 3\*n-1

где значение ***n*** задается с клавиатуры.

Например, для n = 5 нужно вычислить сумму

s = 2 + 5 + 8 + 11 + 14

1. Создайте переменную counter со значением 0. Создайте другую переменную под названием roll, значением которой будет 0. Создайте цикл do-while.

Внутри цикла укажите, что переменная roll равна Random().nextInt(6), то есть выбор случайного числа между 0 и 5. Затем увеличьте counter на 1.

Под конец выведите После X бросков, roll равен Y, где Х является значением counter, а Y — значением roll. Укажите условие цикла, при котором цикл будет заканчиваться при выпаде первого 0.