## СТРУКТУРА ПРОГРАММЫ

Точкой входа в программу на языке Kotlin является функция main. Именно с этой функции начинается выполнение программы на Kotlin, поэтому эта функция должна быть в любой программе на языке Kotlin.

fun main(){

    println("Hello Word")

}

## ПЕРЕМЕННЫЕ

val|var имя\_переменной: тип\_переменной

**ИЗМЕНЯЕМЫЕ И  
НЕИЗМЕНЯЕМЫЕ ПЕРЕМЕННЫЕ**

**val** – неизменяемая переменная (immutable variable) или переменная только для чтения (read-only). Можно присвоить значение такой переменной только один раз, но изменить его после первого присвоения уже нельзя.

**var** – изменяемая переменная (mutable variable). Можно многократно менять значения.

**ОПРЕДЕЛЕНИЕ КОНСТАНТ**

const val имя\_переменной = 100

Отличительной особенностью констант является то, что они на стадии компиляции должны иметь некоторое значение, и это значение изменить нельзя.

Это накладывает на использование констант ряд ограничений:

* нельзя изменить значение константы
* константа должна объявляться на самом верхнем уровне (вне класса/функции)
* тип данных константы должен соответствовать одному из примитивных (например, int) или типу string

## ТИПЫ ДАННЫХ

**ЦЕЛОЧИСЛЕННЫЕ ТИПЫ**

Byte: число от -128 до 127

Short: число от -32 768 до 32 767

Int

Long

val a: Byte = -10

val b: Short = 45

val c: Int = -250

val d: Long = 30000

**ЦЕЛОЧИСЛЕННЫЕ ТИПЫ БЕЗ ЗНАКА**

UByte: число от 0 до 255

UShort: число от 0 до 65 535

UInt: число от 0 до 232 - 1

ULong: число от 0 до 264-1

val a: UByte = 10U

val b: UShort = 45U

val c: UInt = 250U

val d: ULong = 30000U

**ШЕСТНАДЦАТЕРИЧНАЯ ЗАПИСЬ ЧИСЛА**

val num: Int = 0x0A1

**ДВОИЧНАЯ ЗАПИСЬ ЧИСЛА**

val num: Int = 0b0101

**ЧИСЛА С ПЛАВАЮЩЕЙ ТОЧКОЙ**

Float: число от -3.4\*1038 до 3.4\*1038

Double: число от ±5.0\*10-324 до ±1.7\*10308

val a: Double = 1.78

val pi: Float = 3.14F

Также тип Double поддерживает экспоненциальную запись:

val d: Double = 23e3

**ЛОГИЧЕСКИЙ ТИП BOOLEAN**

val isTrue: Boolean = true

**СИМВОЛЬНЫЙ ТИП**

val a: Char = 'A'

Также тип Char может представлять специальные последовательности, которые интерпретируются особым образом:

* \t: табуляция
* \n: перевод строки
* \': одинарная кавычка
* \": двойная кавычка
* \\: обратный слеш

**СТРОКОВЫЙ ТИП**

val name: String = "Соня"

val text: String = """

                        мяу

                        мяу

                        мяу

                   """

**ШАБЛОНЫ СТРОК (ИНТЕРПОЛЯЦИЯ)**

val name = "Соня"

val surname = "Мяу"

val welcome = "Hello, $name $surname"

**ПРЕОБРАЗОВАНИЯ ДАННЫХ К ОПРЕДЕЛЕННОМУ ТИПУ**

* toByte
* toShort
* toInt
* toLong
* toFloat
* toDouble
* toChar

**ТИП ANY**

Тип Any позволяет присвоить переменной данного типа любое значение

## ВЫВОД

print()

println() – необязательно должна принимать некоторое значения. Добавляет перевод на новую строку

## ВВВОД

Для ввода с консоли применяется встроенная функция readLine(). Она возвращает введенную строку. Стоит отметить, что результат этой функции всегда представляет объект типа String. Соответственно введеную строку мы можем передать в переменную типа String:

print("Введите имя: ")

val name = readLine()

println("Ваше имя: $name")

## ОПЕРАЦИИ С ЧИСЛАМИ

* + (сложение)

val x = 5

val y = 6

val z = x + y

println(z)

* - (вычитание)

val x = 5

val y = 6

val z = x - y

* \* (умножение)

val x = 5

val y = 6

val z = x \* y

* / (деление)

val x = 60

val y = 10

val z = x / y

При этом если в операции деления оба операнда представляют целые числа, то результатом тоже будет целое число, а если в процессе деления образовалась дробная часть, то она отбрасывается

* %: возвращает остаток от целочисленного деления двух чисел

val x = 65

val y = 10

val z = x % y  // z = 5

* ++ (инкремент): увеличивает значение на единицу

Префиксный инкремент возвращает увеличенное значение

var x = 5

val y = ++x

println(x)      // x = 6

println(y)      // y = 6

Постфиксный инкремент возвращает значение до увеличения на единицу

var x = 5

val y = x++

println(x)      // x = 6

println(y)      // y = 5

* -- (декремент): уменьшает значение на единицу

Префиксный декремент возвращает уменьшенное значение:

var x = 5

val y = --x

println(x)      // x = 4

println(y)      // y = 4

Постфиксный декремент возвращает значение до уменьшения на единицу:

var x = 5

val y = x--

println(x)      // x = 4

println(y)      // y = 5

## ОПЕРАЦИИ ОТНОШЕНИЯ

* > (больше чем)

val a = 11

val b = 12

val c : Boolean = a > b

println(c) // false - a меньше чем b

* < (меньше чем)

val a = 11

val b = 12

val c = a < b // true

* >= (больше чем или равно)

val a = 11

val b = 12

val c = a >= b // false

val d = 11 >= a // true

* <= (меньше чем или равно)

val a = 11

val b = 12ц

val c = a <= b // true

val d = 15 <= a // false

* == (равно)

val a = 11

val b = 12

val c = a == b // false

val d = b == 12 // true

* != (не равно)

val a = 11

val b = 12

val c = a != b      // true

## ЛОГИЧЕСКИЕ ОПЕРАЦИИ

* and

val a = true

val b = false

val c = a and b // false

* or: возвращает true, если хотя бы один из операндов равен true.

val a = true

val b = false

val c = a or b // true

val d = (11 < 5) or (9 > 10) // false

* xor: возвращает true, если только один из операндов равен true. Если операнды равны, возвращается false

val a = true

val b = false

val c = a xor b // true

val d = a xor (90 > 10) // false

* !: возвращает true, если операнд равен false

val a = true

val b = !a // false

val c = !b // true

В качестве альтернативы оператору ! можно использовать метод not():

val a = true

val b = a.not() // false

val c = b.not() // true

* in: возвращает true, если операнд имеется в некоторой последовательности.

val a = 5

val b = a in 1..6 // true - число 5 входит в последовательность от 1 до 6

val c = 4

val d = c in 11..15 // false - число 4 НЕ входит в последовательность от 11 до 15

## УСЛОВНАЯ КОНСТРУКЦИЯ IF ELSE

val a = 10

if(a == 10) {

    println("a равно 10")

}

else if(a == 9){

    println("a равно 9")

}

else if(a == 8){

    println("a равно 8")

}

else{

    println("a имеет неопределенное значение")

}

Если при определении возвращаемого значения надо выполнить еще какие-нибудь действия, то можно заключить эти действия в блоки кода:

val a = 10

val b = 20

val c = if (a > b){

println("a = $a")

a

} else {

println("b = $b")

b

}

## КОНСТРУКЦИЯ WHEN

when(объект){

    значение1 -> действия1

    значение2 -> действия2

    значениеN -> действияN

}