# **1.** **Основы Kotlin**

**Цель работы:** Освоить основы языка программирования Kotlin, включая базовые синтаксические конструкции, основные типы данных, операторы, работу с функциями и классами.

## **Теоретическая часть**

Kotlin – это современный, кроссплатформенный, статически типизированный, объектно-ориентированный язык программирования. Kotlin полностью совместим с Java и работает на JVM (Java Virtual Machine). Кроме того, существуют реализации для компиляции в JavaScript и нативный код с использованием Kotlin/Native. Также есть поддержка для Kotlin Multiplatform, которая позволяет писать код для нескольких платформ, включая Android, iOS, веб и серверные приложения. Kotlin является основным языком разработки Android-приложений, рекомендуемым компанией Google.

Название Kotlin произошло от острова Котлин, расположенного в Финском заливе рядом с Санкт-Петербургом, на котором располагается город-порт Кронштадт. Название было выбрано аналогично тому, как язык Java был назван в честь острова Ява.

### **Переменные и ветвление**

Переменные в Kotlin могут быть двух типов:

– val – неизменяемая переменная (аналог final в Java).

– var – изменяемая переменная.

Переменные имеют общие типы данных, которые имеются в других си-подобных языках: Int, Double, Float, Long, Boolean, String, Char а также коллекции (List, Set, Map). В Kotlin предусмотрены условные конструкции if-else и выражение when, которое имеет схожий синтаксис со switch-case выражением из си-подобных языков. Kotlin поддерживает циклы for, while и do-while.

Листинг 1.1 – Пример использования условных конструкций if-else и when в языке Kotlin

| val result = if (age > 18) "совершеннолетний" else "несовершеннолетний"  when (day) {  "Понедельник" -> println("Начало недели")  "Суббота", "Воскресенье" -> println("Выходной")  else -> println("Будний день")  } |
| --- |

Выше был приведен блок кода, который создает переменную типа String, в которую будет записан результат в зависимости от возраста (аналог тернарного оператора в Java). Также приведен пример использования конструкции when, которая в зависимости от дня недели выводит в консоль, является ли этот день рабочим или выходным.

### **Массивы**

Массив – это последовательность значений одного типа данных. Массив содержит несколько значений, называемых элементами. Элементы массива упорядочены и доступны с помощью индекса.

Индекс – это целое число, которое обозначает позицию элемента в массиве, отсчитываемую от начального элемента Это называется нулевой индексацией. Первый элемент массива имеет индекс 0, второй элемент имеет индекс 1, потому что он находится на одну позицию дальше, и так далее.

В памяти устройства элементы массива хранятся рядом друг с другом, поэтому доступ к элементу массива по его индексу быстрый. Вы можете получить доступ к любому элементу массива по его индексу и ожидать, что доступ к любому другому элементу займет примерно столько же времени.

Массив имеет фиксированный размер. Это означает, что вы не можете добавлять элементы в массив сверх этого размера. Попытка доступа к элементу с индексом 100 в массиве из 100 элементов вызовет исключение, поскольку максимальный индекс равен 99 (помните, что первый индекс равен 0, а не 1). Однако вы можете изменять значения индексов в массиве.

Чтобы объявить массив в коде, используйте функцию arrayOf()



Рис. 1.1. Синтаксис для создания массива (составлено авторами)

Функция arrayOf() принимает элементы массива в качестве параметров и возвращает массив соответствующего типа. Если вы передадите два аргумента в arrayOf(), результирующий массив будет содержать два элемента с индексами 0 и 1. Если вы передадите три аргумента, результирующий массив будет содержать 3 элемента с индексами от 0 до 2.

### **Функции**

Все программы в Kotlin начинают свою работу со стартовой функции main. Функция main по умолчанию является точкой входа в приложение, от которой будут запущены другие функции и подпрограммы.

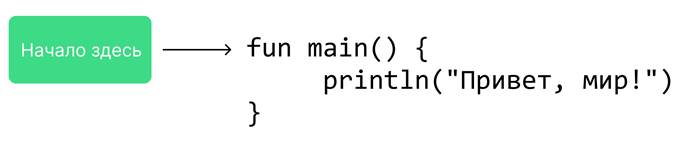


Рис. 1.2. Точка входа в программу (составлено авторами)

Функции определяются с помощью ключевого слова fun. Затем задаётся имя функции, чтобы ее можно было вызывать. Также функция может принимать входные данные, которые передаются ​​при вызове. Функция использует эти входные данные для выполнения задачи для которой она написана. Функции могут не принимать входных данных. Инструкции для выполнения задачи описываются в теле функции.



Рис. 1.3. Синтаксис функций в Kotlin (составлено авторами)

### 

### **Классы**

Важной частью Kotlin являются классы *–* большую часть времени вы будете работать с ними и с их экземплярами. Все типы данных в Kotlin являются классами, которые описывают, как можно использовать тот или иной тип данных.

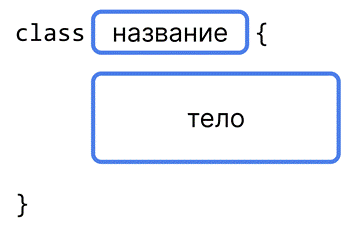


Рис. 1.4. Структура класса в Kotlin (составлено авторами)

Классы в Kotlin принято именовать с большой буквы. В теле класса могут содержаться функции и переменные, необходимые для его логики. Класс может содержать вложенные классы или объекты. Также как и в других языках у любых членов класса может быть переопределен модификатор доступа. В Kotlin существует всего 3 модификатора доступа:

— private;

— protected;

— internal.

Public в Kotlin не существует в привычном виде, так как это модификатор доступа по умолчанию (если вы не указали явно один из трех выше).

Конструктор в классе может быть указан явно через ключевое слово constructor внутри тела. Класс может иметь несколько конструкторов с разными наборами параметров.

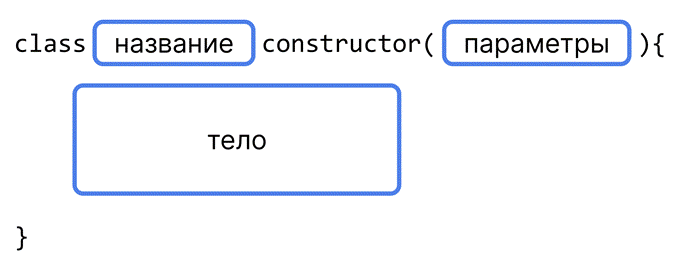


Рис. 1.5. Структура класса c конструктором в Kotlin (составлено авторами)

Либо ключевое слово constructor можно опустить и использовать класс без него, указав только входные параметры, как вы это делали, например, с функциями.

Наследование в классе можно выполнить, указав через двоеточие после параметров название класса-родителя.

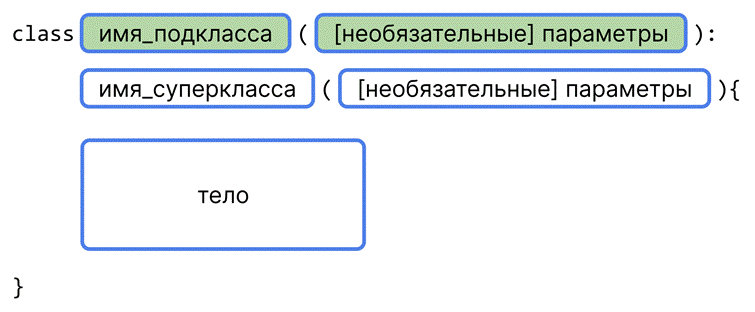


Рис. 1.6. Структура класса c наследованием в Kotlin (составлено авторами)

## **Контрольное задание для самопроверки**

Создайте новый проект в Intellij IDEA и выполните задания. Создайте отчет о результатах проделанной работы. Исходный код загрузите в удаленный репозиторий.

1) Пользователь вводит целое положительное число. Вычислить сумму первой и последней цифры этого числа и вывести результат на экран. Использовать ввод с клавиатуры readln(). Есть несколько способов:

— преобразовать строку в число, используя toInt(), и отделить цифры, используя остаток от деления на 10 “%” и целочисленное деление без остатка “/”

— получить первый символ строки с помощью функции first(), и последний **–** с помощью last().

2) Пользователь поочередно вводит с клавиатуры любые числа (подтверждая ввод каждого нажатием “Enter”) до тех пор, пока не будет введено число “0”. После ввода нуля необходимо вывести на экран количество введенных чисел, их общую сумму и среднее арифметическое. Использовать ввод с клавиатуры readln().

3) Программа “Угадай число”. Программа “загадывает” рандомное число A (используем (0..10).random() ), пользователь вводит свой вариант – число B. Если число В > A, вывести сообщение “Много” и повторить ввод; если В < А вывести сообщение “Мало” и повторить ввод; иначе – “Угадал”.

4) Задается количество простых чисел n (1n). Необходимо вывести на экран простые числа, начиная с 2.

Пример вывода для n-ного количества простых чисел:

1-ое число: 2

2-ое число: 3

3-е число: 5

…

n-ное число: x

5) Дан массив из целых чисел. Вывести на экран элементы массива, которые больше соседних (предыдущего и следующего) элементов массива. Если элемент первый или последний, то его не выводить. Выполнить с использованием:

— цикла for;

— цикла while;

— оператора forEach.

Дан массив с элементами. Найдите произведение элементов этого массива, а также min и max элемент. Выполнить с использованием:

— цикла for;

— цикла while;

— оператора forEach;

— функции reduce();

— функций min() max().

### 

6) Решение квадратного уравнения ax2 + bx + c = 0.

— Написать функцию sqr(n), которая в качестве параметра принимает число n и возвращает квадрат этого числа.

— Написать функцию discriminant(a, b, c), которая в качестве параметров принимает коэффициенты a, b и c заданного уравнения и возвращает рассчитанный дискриминант.

— Написать функцию rootsNumber(a, b, c), которая в качестве параметров принимает коэффициенты a, b и c заданного уравнения. Данная функция в зависимости от дискриминанта должна определить и вернуть количество корней заданного квадратного уравнения. Реализовать с помощью конструкции when().

— Написать функцию quadraticRoot(a, b, c), которая в качестве параметров принимает коэффициенты a, b и c заданного уравнения. Данная функция должна вывести на экран корни заданного квадратного уравнения. При реализации использовать ранее написанные функции.

7) Создать класс, который содержит одномерный массив чисел (массив передается в конструктор). В классе создать методы:

— для вычисления суммы положительных элементов;

— для вычисления произведения элементов массива;

— для вычисления среднего арифметического значения элементов массива.

Создать объект класса и вывести на экран результаты работы методов.

8) Создать класс Vector, который описывает вектор в трехмерном пространстве. Конструктор в качестве параметров принимает координаты x, y, z. В классе создать методы:

- для вычисления длины вектора ;

- для вычисления скалярного произведения векторов 𝑥1∗𝑥2+𝑦1∗𝑦2 +𝑧1∗𝑧2.. В качестве параметра метод принимает объект класса **Vector;**

- добавить инфиксную запись между двумя векторами в предыдущий пункт. Используйте ключевое слово **infix;**

- описать функцию скалярного произведения векторов, чтобы выражение приняло вид: Vector(1.0,2.0,3.0) \* Vector(3.0,2.0,1.0);

- вне класса создать функцию скалярного произведения, которая в качестве параметров принимает два объекта типа Vector.

### 

9) Создать суперкласс Vehicle, описывающий поведение любого транспортного средства.

В классе Vehicle создать два свойства с инициализацией значений по умолчанию, которые будут переопределены в классах наследниках: скорость и название.

В классе Vehicle создать два метода start() и stop() с действием по умолчанию, которые будут переопределены в классах наследниках. Методы должны выводить на экран начало движения и остановку. Пример: “Автомобиль начал движение со скоростью 50 км/ч”.

Создать три класса-наследника от Vehicle, описывающих три типа транспортных средств: лодка, самолет и танк.