**17 Функции высшего порядка с коллекциями**

[1. Введение](https://developer.android.com/codelabs/basic-android-kotlin-compose-higher-order-functions?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-3-pathway-1%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-higher-order-functions#0)

Ранее вы узнали о функциях высшего порядка, то есть функциях, которые принимают другие функции в качестве параметров и/или возвращают функцию, например repeat(). Функции высшего порядка особенно актуальны для коллекций, поскольку они помогают выполнять общие задачи, такие как сортировка или фильтрация, с меньшим количеством кода. Теперь, когда у вас есть прочная основа для работы с коллекциями, пришло время вернуться к функциям высшего порядка.

В этой лабораторной работе вы узнаете о различных функциях, которые можно использовать с типами коллекций, включая forEach(), map(), filter(), groupBy(), fold()и sortedBy(). В процессе вы получите дополнительную практику работы с лямбда-выражениями.

[2. forEach() и шаблоны строк с лямбда-выражениями](https://developer.android.com/codelabs/basic-android-kotlin-compose-higher-order-functions?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-3-pathway-1%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-higher-order-functions#1)

Стартовый код

В следующих примерах вы представите список меню печенья пекарни и воспользуетесь функциями более высокого порядка для форматирования меню различными способами.

Начните с настройки исходного кода.

1. Перейдите на [Kotlin Playground](https://developer.android.com/training/kotlinplayground) .
2. Над  функцией main() добавьте  класс Cookie. Каждый экземпляр Cookie представляет собой элемент меню с значками name, price и другой информацией о файле cookie.

class Cookie(  
    val name: String,  
    val softBaked: Boolean,  
    val hasFilling: Boolean,  
    val price: Double  
)  
  
fun main() {  
  
}

1. Под классом Cookie, за пределами main(), создайте список файлов cookie, как показано. Предполагается, что тип List<Cookie>.

class Cookie(  
    val name: String,  
    val softBaked: Boolean,  
    val hasFilling: Boolean,  
    val price: Double  
)  
  
val cookies = listOf(  
    Cookie(  
        name = "Chocolate Chip",  
        softBaked = false,  
        hasFilling = false,  
        price = 1.69  
    ),  
    Cookie(  
        name = "Banana Walnut",   
        softBaked = true,   
        hasFilling = false,   
        price = 1.49  
    ),  
    Cookie(  
        name = "Vanilla Creme",  
        softBaked = false,  
        hasFilling = true,  
        price = 1.59  
    ),  
    Cookie(  
        name = "Chocolate Peanut Butter",  
        softBaked = false,  
        hasFilling = true,  
        price = 1.49  
    ),  
    Cookie(  
        name = "Snickerdoodle",  
        softBaked = true,  
        hasFilling = false,  
        price = 1.39  
    ),  
    Cookie(  
        name = "Blueberry Tart",  
        softBaked = true,  
        hasFilling = true,  
        price = 1.79  
    ),  
    Cookie(  
        name = "Sugar and Sprinkles",  
        softBaked = false,  
        hasFilling = false,  
        price = 1.39  
    )  
)  
  
fun main() {  
  
}

Прокрутите список с помощью функции forEach()

Первая функция высшего порядка, о которой вы узнаете, — это функция forEach(). forEach() выполняет функцию, переданную в качестве параметра, один раз для каждого элемента коллекции. Это работает аналогично функции repeat()или циклу for. Лямбда-выражение выполняется для первого элемента, затем для второго элемента и т. д., пока оно не будет выполнено для каждого элемента коллекции. Сигнатура метода следующая:

forEach(action: (T) -> Unit)

forEach() принимает один параметр действия — функцию типа (T) -> Unit.

T соответствует любому типу данных, который содержит коллекция. Поскольку лямбда принимает один параметр, вы можете опустить имя и ссылаться на параметр с помощью it.

Используйте функцию forEach() для печати элементов в списке cookies.

1. В main() вызовите forEach() список cookies, используя конечный синтаксис лямбда. Поскольку конечная лямбда является единственным аргументом, вы можете опустить круглые скобки при вызове функции.

fun main() {  
    **cookies.forEach {  
          
    }**}

1. В теле лямбды добавьте println()оператор, который печатает it.

fun main() {

    cookies.forEach {

**println("Menu item: $it")**    }

}

1. Запустите свой код и наблюдайте за выводом. Все, что выводится, — это имя типа (Cookie) и уникальный идентификатор объекта, но не содержимое объекта.

Menu item: Cookie@5a10411

Menu item: Cookie@68de145

Menu item: Cookie@27fa135a

Menu item: Cookie@46f7f36a

Menu item: Cookie@421faab1

Menu item: Cookie@2b71fc7e

Menu item: Cookie@5ce65a89

Встраивание выражений в строки

Когда вы впервые познакомились с шаблонами строк, вы увидели, как  можно использовать символ доллара ($ ) с именем переменной для вставки его в строку. Однако это не работает должным образом в сочетании с оператором точки ( .) для доступа к свойствам.

1. При вызове forEach() измените тело лямбды для вставки $it.name в строку.

cookies.forEach {  
    println("Menu item: $it.name")  
}

1. Запустите свой код. Обратите внимание, что при этом вставляется имя класса Cookie и уникальный идентификатор объекта, за которым следует .name. Доступ к значению свойства name отсутствует.

Menu item: Cookie@5a10411.name

Menu item: Cookie@68de145.name

Menu item: Cookie@27fa135a.name

Menu item: Cookie@46f7f36a.name

Menu item: Cookie@421faab1.name

Menu item: Cookie@2b71fc7e.name

Menu item: Cookie@5ce65a89.name

Чтобы получить доступ к свойствам и встроить их в строку, вам понадобится выражение. Вы можете сделать выражение частью шаблона строки, заключив его в фигурные скобки.



Лямбда-выражение помещается между открывающей и закрывающей фигурными скобками. Вы можете получать доступ к свойствам, выполнять математические операции, вызывать функции и т. д., а возвращаемое значение лямбды вставляется в строку.

Давайте изменим код так, чтобы имя вставлялось в строку.

1. Заключите it.name в фигурные скобки, чтобы сделать его лямбда-выражением.

cookies.forEach {  
    println("Menu item: ${it.name}")  
}

1. Запустите свой код. Вывод содержит данные name каждого файла Cookie.

Menu item: Chocolate Chip

Menu item: Banana Walnut

Menu item: Vanilla Creme

Menu item: Chocolate Peanut Butter

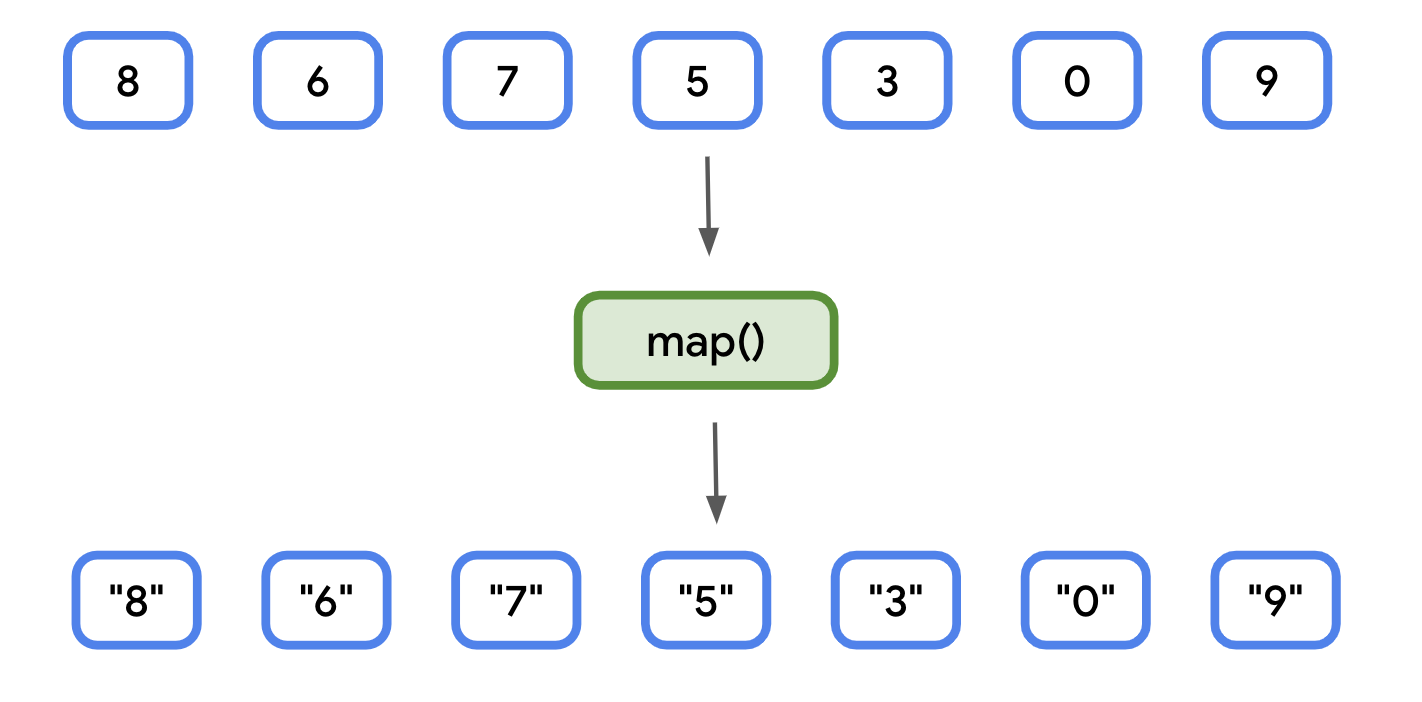
Menu item: Snickerdoodle

Menu item: Blueberry Tart

Menu item: Sugar and Sprinkles

## [3. map()](https://developer.android.com/codelabs/basic-android-kotlin-compose-higher-order-functions?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-3-pathway-1%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-higher-order-functions#2)

Функция map() позволяет преобразовать коллекцию в новую коллекцию с тем же количеством элементов. Например, map()можно преобразовать List<Cookie> в List<String> содержащий только файлы cookie name, при условии, что вы сообщите функции map(), как создать String из каждого Cookie элемента.



Допустим, вы пишете приложение, отображающее интерактивное меню пекарни. Когда пользователь переходит к экрану с меню файлов cookie, он может захотеть увидеть данные, представленные в логической форме, например, имя, за которым следует цена. Используя функцию map(), вы можете создать список строк, отформатированных с соответствующими данными (имя и цена) .

1. Удалите весь предыдущий код из main(). Создайте новую переменную с именем fullMenu и установите ее равной результату вызова map() списка cookies.

val fullMenu = cookies.map {  
      
}

1. В тело лямбды добавьте строку, отформатированную так, чтобы она включала символы it.name и it.price

val fullMenu = cookies.map {  
    "${it.name} - $${it.price}"  
}

1. Распечатайте содержимое fullMenu. Вы можете сделать это, используя forEach().fullMenu Возвращаемая коллекция имеет тип List<String>, а не List<Cookie>. Каждый  Cookie в  cookies соответствует String in fullMenu.

println("Full menu:")  
fullMenu.forEach {  
    println(it)  
}

1. Запустите свой код. Вывод соответствует содержимому списка fullMenu.

Full menu:

Chocolate Chip - $1.69

Banana Walnut - $1.49

Vanilla Creme - $1.59

Chocolate Peanut Butter - $1.49

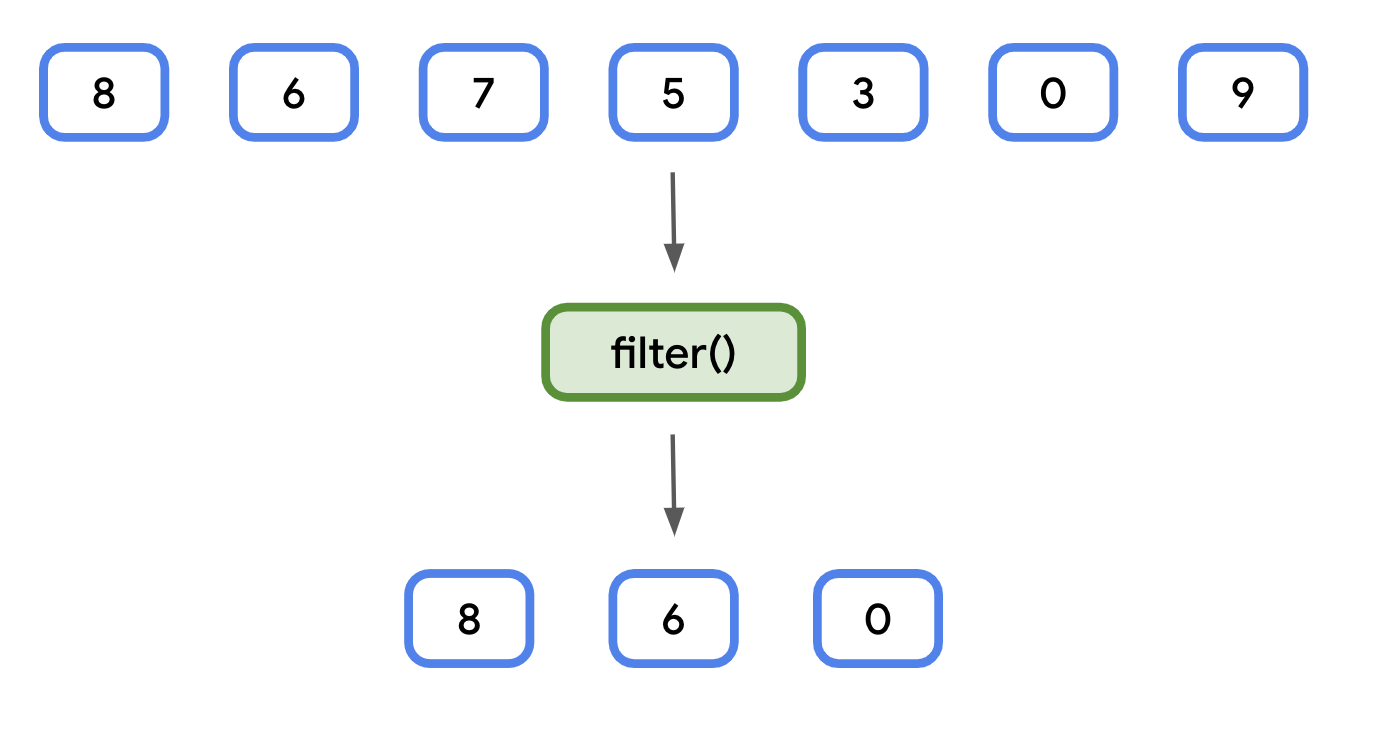
Snickerdoodle - $1.39

Blueberry Tart - $1.79

Sugar and Sprinkles - $1.39

## [4. filter()](https://developer.android.com/codelabs/basic-android-kotlin-compose-higher-order-functions?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-3-pathway-1%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-higher-order-functions#3)

Функция filter() позволяет создать подмножество коллекции. Например, если у вас есть список чисел, вы можете использовать его filter() для создания нового списка, содержащего только числа, делящиеся на 2.



В то время как результат функции map() всегда дает коллекцию одного и того же размера, filter() дает коллекцию того же размера или меньше, чем исходная коллекция. В отличие от map(), результирующая коллекция также имеет тот же тип данных, поэтому фильтрация List<Cookie> приведет к получению другого List<Cookie>.

Как map() и forEach(), filter() принимает в качестве параметра одно лямбда-выражение. Лямбда имеет один параметр, представляющий каждый элемент коллекции и возвращающий значение Boolean.

Для каждого предмета коллекции:

* Если результатом лямбда-выражения является true, то элемент включается в новую коллекцию.
* Если результат равен false, элемент не включается в новую коллекцию.

Это полезно, если вы хотите получить подмножество данных в своем приложении. Например, предположим, что пекарня хочет выделить свое мягкое печенье в отдельный раздел меню cookies. Вы можете сначала  составить список filter(), прежде чем печатать элементы.

1. В main() создайте новую переменную с именем softBakedMenu и присвойте ей результат вызова filter() списка cookies.

val softBakedMenu = cookies.filter {  
}

1. В тело лямбды добавьте логическое выражение, чтобы проверить, softBaked равно ли свойство файла cookie true. Поскольку softBaked это Boolean само по себе, тело лямбды должно содержать только it.softBaked.

val softBakedMenu = cookies.filter {  
**it.softBaked**}

1. Распечатайте содержимое softBakedMenu файла forEach().

println("Soft cookies:")  
softBakedMenu.forEach {  
    println("${it.name} - $${it.price}")  
}

1. Запустите свой код. Меню распечатано, как и раньше, но включает только мягкое печенье.

...

Soft cookies:

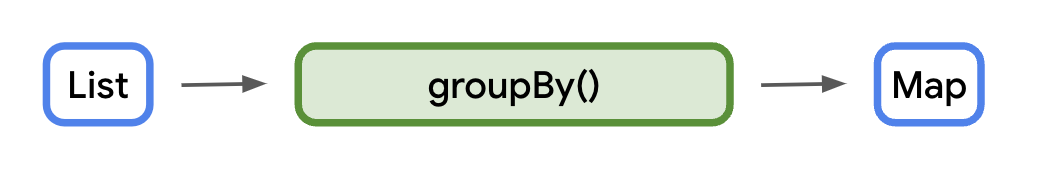
Banana Walnut - $1.49

Snickerdoodle - $1.39

Blueberry Tart - $1.79

## [5. groupBy()](https://developer.android.com/codelabs/basic-android-kotlin-compose-higher-order-functions?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-3-pathway-1%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-higher-order-functions#4)

Функцию groupBy()можно использовать для преобразования списка в карту на основе функции. Каждое уникальное возвращаемое значение функции становится ключом в результирующей карте. Значениями для каждого ключа являются все элементы коллекции, которые создали это уникальное возвращаемое значение.



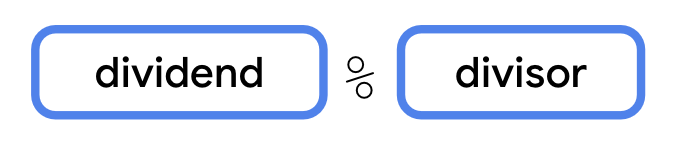
Тип данных ключей такой же, как тип возвращаемого значения функции, переданной в groupBy(). Тип данных значений — это список элементов из исходного списка.

**Примечание.** Значение не обязательно должно быть того же типа, что и список. Есть еще одна версия, [groupBy()](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/group-by.html" \t "_blank)которая может преобразовать значения в другой тип. Однако эта версия здесь не рассматривается.

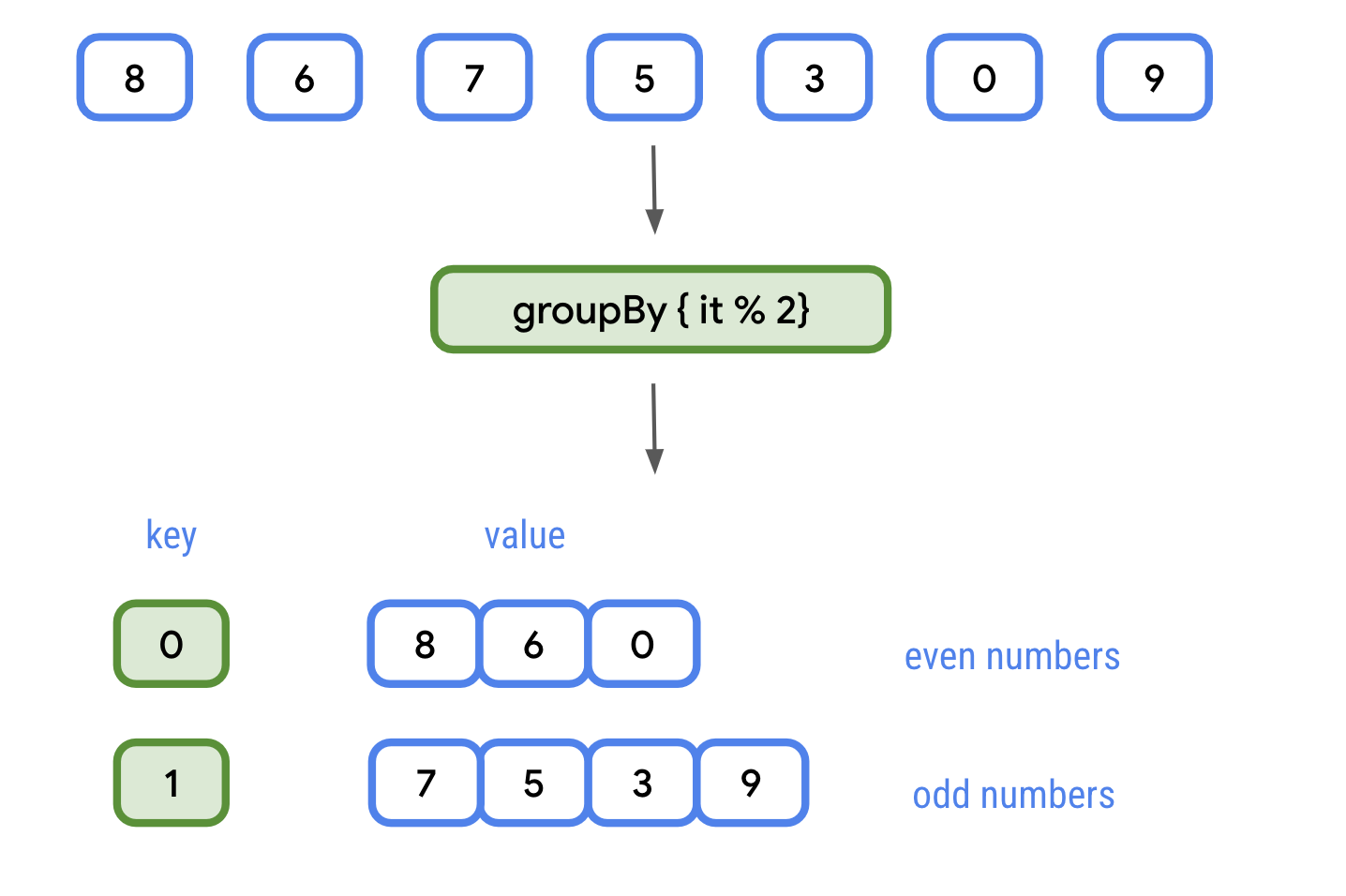
Это может быть сложно концептуализировать, поэтому давайте начнем с простого примера. Учитывая тот же список чисел, что и раньше, сгруппируйте их как нечетные или четные.

Вы можете проверить, является ли число нечетным или четным, разделив его на 2 и проверив, равен ли остаток 0 или 1. Если остаток равен 0, число четное. В противном случае, если остаток равен 1, число нечетное.

Этого можно добиться с помощью оператора по модулю ( %). Оператор по модулю делит делимое в левой части выражения на делитель в правой.



Вместо возврата результата деления, как оператор деления ( /), оператор по модулю возвращает остаток. Это делает его полезным для проверки того, является ли число четным или нечетным.



Функция groupBy() вызывается со следующим лямбда-выражением: { it % 2 }.

Полученная карта имеет два ключа: 0 и 1. Каждый ключ имеет значение типа List<Int>. Список ключа 0содержит все четные числа, а список ключа 1содержит все нечетные числа.

Реальным вариантом использования может быть приложение для фотографий, которое группирует фотографии по теме или месту, где они были сделаны. Для нашего меню выпечки давайте сгруппируем меню по тому, мягкое ли печенье.

Используйте groupBy() для группировки меню по softBaked свойству.

1. Удалите вызов filter()из предыдущего шага.

**Код для удаления**

val softBakedMenu = cookies.filter {  
    it.softBaked  
}  
println("Soft cookies:")  
softBakedMenu.forEach {  
    println("${it.name} - $${it.price}")  
}

1. Вызов groupBy() списка cookies, сохранение результата в переменной с именем groupedMenu.

val groupedMenu = cookies.groupBy {}

1. Передайте лямбда-выражение, которое возвращает it.softBaked. Тип возвращаемого значения будет Map<Boolean, List<Cookie>>.

val groupedMenu = cookies.groupBy { it.softBaked }

1. Создайте переменную softBakedMenu, содержащую значение groupedMenu[true], и crunchyMenu переменную, содержащую значение groupedMenu[false]. Поскольку результат индексации Map может иметь значение NULL, вы можете использовать оператор Элвиса ( ?:), чтобы вернуть пустой список.

val softBakedMenu = groupedMenu[true] ?: listOf()  
val crunchyMenu = groupedMenu[false] ?: listOf()

**Примечание.** В качестве альтернативы emptyList()создается пустой список, который может быть более читабельным.

1. Добавьте код для печати меню мягкого печенья, а затем меню хрустящего печенья.

println("Soft cookies:")  
softBakedMenu.forEach {  
    println("${it.name} - $${it.price}")  
}  
println("Crunchy cookies:")  
crunchyMenu.forEach {  
    println("${it.name} - $${it.price}")  
}

1. Запустите свой код. Используя groupBy()функцию, вы разделяете список на две части в зависимости от значения одного из свойств.

...

Soft cookies:

Banana Walnut - $1.49

Snickerdoodle - $1.39

Blueberry Tart - $1.79

Crunchy cookies:

Chocolate Chip - $1.69

Vanilla Creme - $1.59

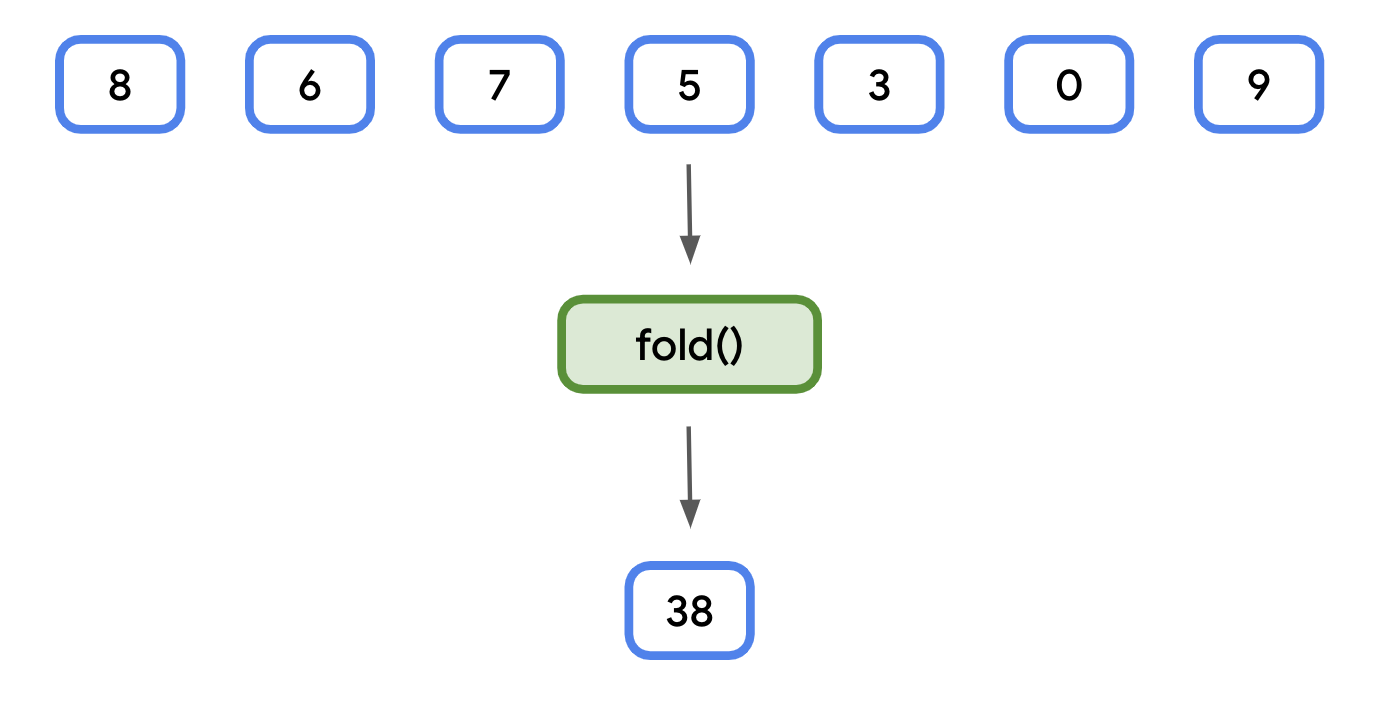
Chocolate Peanut Butter - $1.49

Sugar and Sprinkles - $1.39

**Примечание.** Если вам нужно разделить список только на две части, альтернативой является функция [partition()](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/partition.html" \t "_blank).

## [6. fold()](https://developer.android.com/codelabs/basic-android-kotlin-compose-higher-order-functions?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-3-pathway-1%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-higher-order-functions#5)

Функция fold() используется для создания одного значения из коллекции. Чаще всего это используется для таких вещей, как вычисление общей суммы цен или суммирование всех элементов в списке для нахождения среднего значения.



Функция fold()принимает два параметра:

* Начальное значение. Тип данных определяется при вызове функции (то есть начальное значение 0 подразумевается как Int).
* Лямбда-выражение, возвращающее значение того же типа, что и исходное значение.

Лямбда-выражение дополнительно имеет два параметра:

* Первый известен как аккумулятор. Он имеет тот же тип данных, что и исходное значение. Думайте об этом как о промежуточной сумме. Каждый раз, когда вызывается лямбда-выражение, аккумулятор равен возвращаемому значению предыдущего вызова лямбда-выражения.
* Второй тип того же типа, что и каждый элемент коллекции.

Как и другие функции, которые вы видели, лямбда-выражение вызывается для каждого элемента коллекции, поэтому его можно использовать fold() как краткий способ суммирования всех элементов.

Давайте воспользуемся fold() для расчета общей стоимости всех файлов cookie.

1. В main() создайте новую переменную с именем totalPrice и установите ее равной результату вызова fold() списка cookies. Передайте 0.0 начальное значение. Предполагается, что его тип Double.

val totalPrice = cookies.fold(0.0) {  
}

1. Вам нужно будет указать оба параметра для лямбда-выражения. Используйте total для аккумулятора и cookie для элемента коллекции. Используйте стрелку ( ->) после списка параметров.

val totalPrice = cookies.fold(0.0) {total, cookie ->  
}

1. В теле лямбды вычислите сумму total и cookie.price. Предполагается, что это возвращаемое значение total, и оно передается при следующем вызове лямбды.

val totalPrice = cookies.fold(0.0) {total, cookie ->  
    total + cookie.price  
}

1. Выведите значение totalPrice в формате строки для удобства чтения.

println("Total price: $${totalPrice}")

1. Запустите свой код. Результат должен быть равен сумме цен в cookies списке.

...

Total price: $10.83

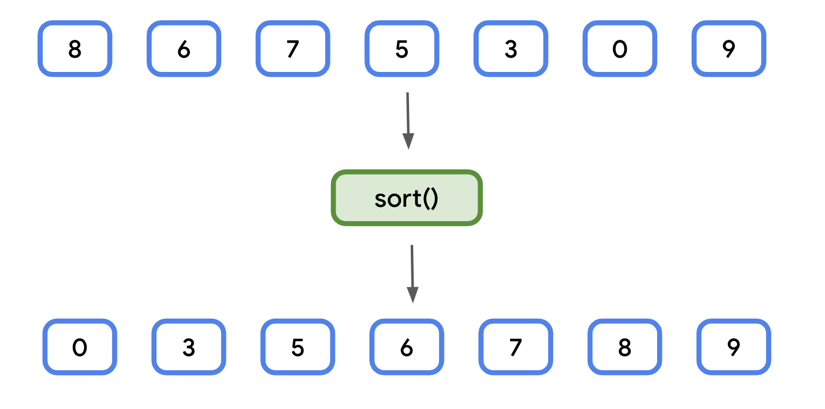
**Примечание:**fold() иногда называется reduce(). Функция fold()в Kotlin работает так же, как reduce()функция в JavaScript, Swift, Python и т. д. Обратите внимание, что в Kotlin также есть собственная функция под названием [reduce()](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/reduce.html" \t "_blank), где аккумулятор начинается с первого элемента коллекции, а не с начального значения, передаваемого как аргумент.

**Примечание.** В коллекциях Kotlin также есть [sum()](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/sum.html" \t "_blank)функция для числовых типов, а также [sumOf()](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/sum-of.html" \t "_blank)функция более высокого порядка.

## [7. sortedBy()](https://developer.android.com/codelabs/basic-android-kotlin-compose-higher-order-functions?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-3-pathway-1%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-higher-order-functions#6)

Когда вы впервые узнали о коллекциях, вы узнали, что эту sort()функцию можно использовать для сортировки элементов. Однако это не будет работать с коллекцией объектов Cookie. У класса Cookie есть несколько свойств, и Котлин не будет знать, по каким свойствам (name, price и т. д.) вы хотите выполнить сортировку.

Для этих случаев коллекции Kotlin предоставляют sortedBy() функцию. sortedBy() позволяет указать лямбду, возвращающую свойство, по которому вы хотите выполнить сортировку. Например, если вы хотите отсортировать по price, лямбда вернет it.price. Пока тип данных значения имеет естественный порядок сортировки (строки сортируются в алфавитном порядке, числовые значения сортируются в порядке возрастания), оно будет отсортировано так же, как коллекция этого типа.



Вы будете использовать sortedBy() для сортировки списка файлов cookie в алфавитном порядке.

1. В main() после существующего кода добавьте новую переменную с именем alphabeticalMenuи установите ее равной вызову sortedBy()в cookies списке.

val alphabeticalMenu = cookies.sortedBy {  
}

1. В лямбда-выражении верните it.name. Результирующий список по-прежнему будет иметь тип List<Cookie>, но отсортирован на основе name.

val alphabeticalMenu = cookies.sortedBy {  
    it.name  
}

1. Напечатайте имена файлов cookie в формате alphabeticalMenu. Вы можете использовать forEach()для печати каждого имени на новой строке.

println("Alphabetical menu:")  
alphabeticalMenu.forEach {  
    println(it.name)  
}

1. Запустите свой код. Имена файлов cookie печатаются в алфавитном порядке.

...

Alphabetical menu:

Banana Walnut

Blueberry Tart

Chocolate Chip

Chocolate Peanut Butter

Snickerdoodle

Sugar and Sprinkles

Vanilla Creme

**Примечание.** Коллекции Kotlin также имеют [sort()](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/sort.html" \t "_blank)функцию, если тип данных имеет естественный порядок сортировки.

[8. Заключение](https://developer.android.com/codelabs/basic-android-kotlin-compose-higher-order-functions?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-3-pathway-1%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-higher-order-functions#7)

Поздравляем! Вы только что увидели несколько примеров того, как функции высшего порядка можно использовать с коллекциями. Общие операции, такие как сортировка и фильтрация, могут выполняться в одной строке кода, что делает ваши программы более краткими и выразительными.

Краткое содержание

* Вы можете перебирать каждый элемент коллекции, используя forEach().
* Выражения можно вставлять в строки.
* map()используется для форматирования элементов коллекции, часто как коллекции другого типа данных.
* filter()может генерировать подмножество коллекции.
* groupBy()разбивает коллекцию на основе возвращаемого значения функции.
* fold()превращает коллекцию в одно значение.
* sortedBy()используется для сортировки коллекции по указанному свойству.

Дополнительно

<https://developer.android.com/codelabs/basic-android-kotlin-compose-higher-order-functions?continue=https%3A%2F%2Fdeveloper.android.com%2Fcourses%2Fpathways%2Fandroid-basics-compose-unit-3-pathway-1%23codelab-https%3A%2F%2Fdeveloper.android.com%2Fcodelabs%2Fbasic-android-kotlin-compose-higher-order-functions#8>