

**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ
КОММУНИКАЦИЙ**

Ордена Трудового Красного Знамени

**Федеральное государственное бюджетное образовательное учреждение
высшего образования**

«Московский технический университет связи и информатики»

Кафедра «Математическая кибернетика и информационные технологии»

Лабораторная работа №2

по дисциплине

Функциональное программирование

на тему

«Коллекции и функции высшего порядка»

Выполнил: студент группы

БВТ2201

Шамсутдинов Рустам

Москва 2025

Введение

Цель работы:

Закрепить базовый синтаксис Scala. Научиться использовать функции высшего порядка (map, filter, reduce). Понять преимущества иммутабельных коллекций.

Теоретические основы:

- Основы синтаксиса Scala: val и var, типы данных, выражения.
- Функции высшего порядка.
- Иммутабельные коллекции: List, Seq, Map, Set.
- Каррирование функций.

Задания:

1. Создать коллекции различных типов и выполнить над ними базовые операции.
2. Использовать map, filter и reduce для обработки числовых и строковых коллекций.
3. Написать функцию высшего порядка, принимающую другую функцию как аргумент.
4. Реализовать пример каррирования функции.
5. Сравнить работу с мутабельными и иммутабельными коллекциями.

Ход работы

Задание 1. Создать коллекции различных типов и выполнить над ними базовые операции.



```
run | debug
18 object CollectionsLab {
17   def main(args: Array[String]): Unit = {
16
15     println("\ntask 1:\n")
14
13     val list = List(1, 2, 3)
12     println(list)
11     println(list.drop(2))
10     println(list.dropWhile(_ < 2))
9     println(list.tail)
8     println(list.head)
7
6     val tuple = ("hi", 12, 4.5F)
5     println(tuple)
4     println(tuple(0))
3     println(tuple.last)
2
1
19 // println("\ntask 2:\n")
1
}
```

PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
sypoo@pop-os:~/mtuci/mtuci-scala/lab2/collections$ sbt
1
(hi,12,4.5)
hi
4.5

task 2:

[success] Total time: 0 s, completed Nov 12, 2025, 8:09:47 PM
[info] 35. Monitoring source files for root/run...
[info] Press <enter> to interrupt or '?' for more options.
[info] Build triggered by /home/sypoo/mtuci/mtuci-scala/lab2/collections/src/m
[info] compiling 1 Scala source to /home/sypoo/mtuci/mtuci-scala/lab2/collecti
[info] running CollectionsLab

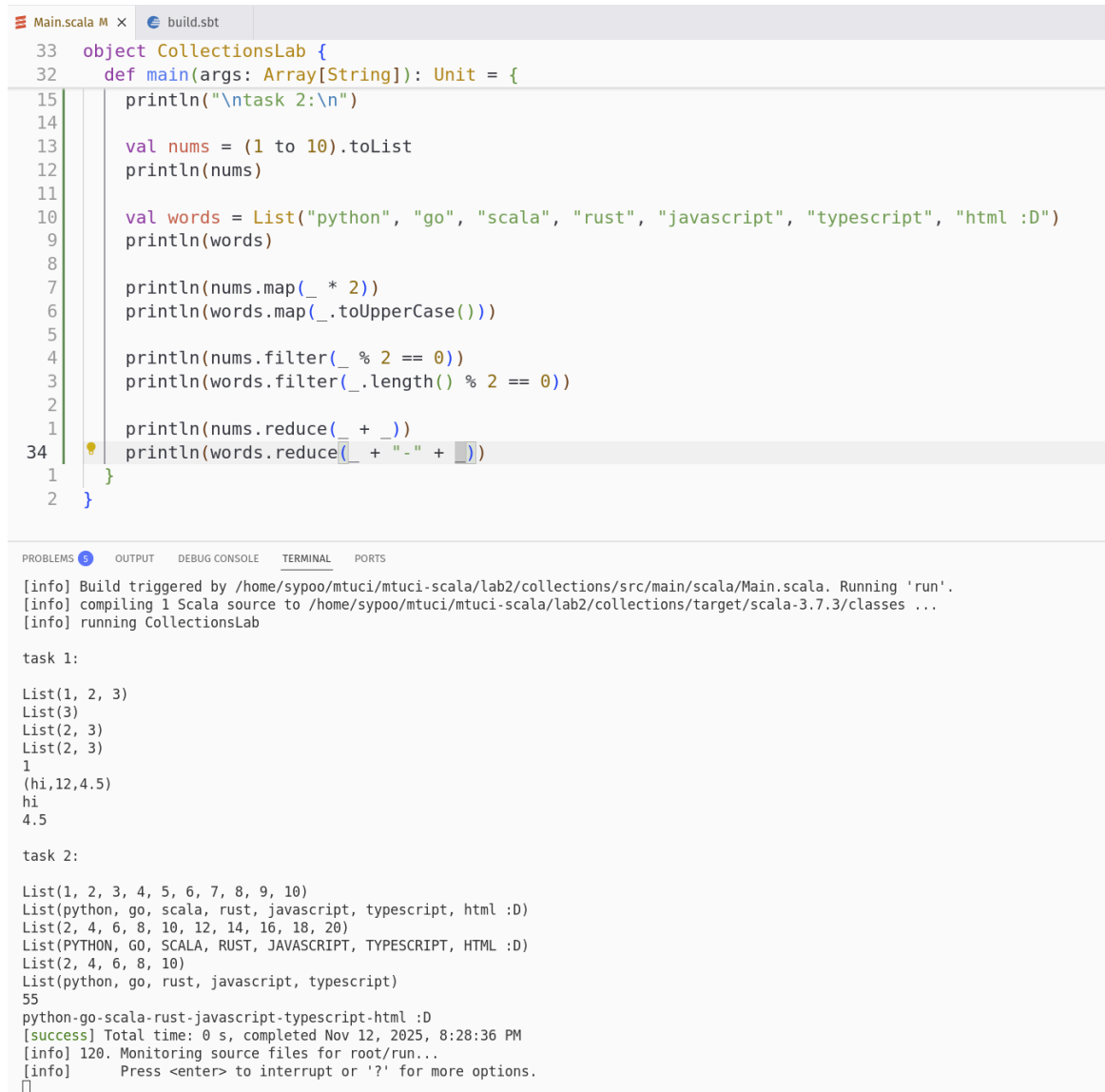
task 1:

List(1, 2, 3)
List(3)
List(2, 3)
List(2, 3)
1
(hi,12,4.5)
hi
4.5

[success] Total time: 0 s, completed Nov 12, 2025, 8:09:58 PM
[info] 36. Monitoring source files for root/run...
[info] Press <enter> to interrupt or '?' for more options.
□
```

Рис. 1 – базовые операции с коллекциями.

Задание 2. Использовать map, filter и reduce для обработки числовых и строковых коллекций.



The screenshot shows an IDE with a Scala file named `Main.scala` and a `build.sbt` file. The code defines an object `CollectionsLab` with a `main` function. It creates a list of numbers `nums` (1 to 10) and a list of words `words` ("python", "go", "scala", "rust", "javascript", "typescript", "html :D"). It then demonstrates the use of `map`, `filter`, and `reduce` on these collections.

```
33 object CollectionsLab {
32   def main(args: Array[String]): Unit = {
15     println("\ntask 2:\n")
14
13     val nums = (1 to 10).toList
12     println(nums)
11
10     val words = List("python", "go", "scala", "rust", "javascript", "typescript", "html :D")
9     println(words)
8
7     println(nums.map(_ * 2))
6     println(words.map(_.toUpperCase()))
5
4     println(nums.filter(_ % 2 == 0))
3     println(words.filter(_.length() % 2 == 0))
2
1     println(nums.reduce(_ + _))
34    println(words.reduce(_ + "-" + _))
1   }
2 }
```

The output of the program is shown in the terminal:

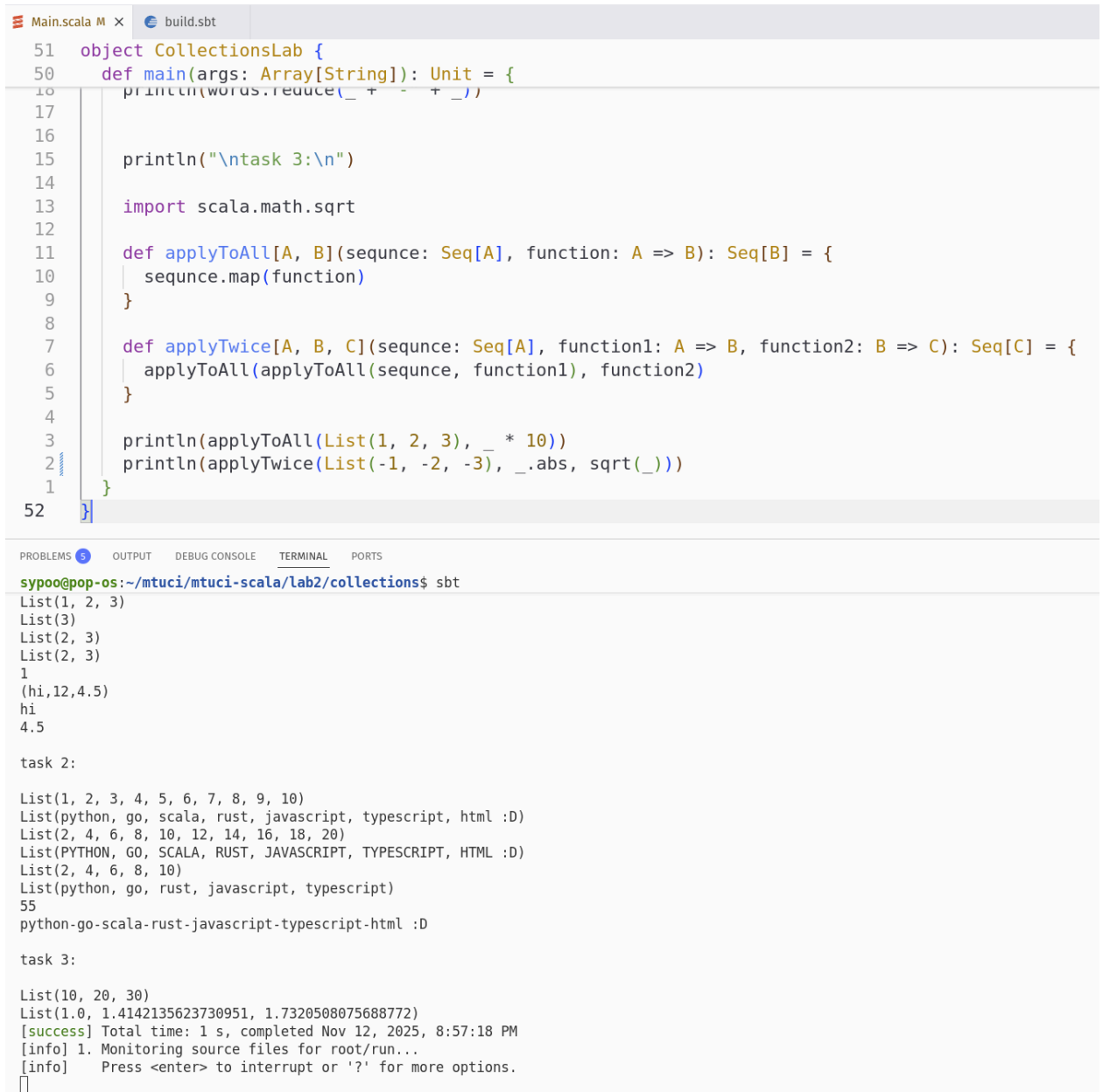
```
[info] Build triggered by /home/sypoo/mtuci/mtuci-scala/lab2/collections/src/main/scala/Main.scala. Running 'run'.
[info] compiling 1 Scala source to /home/sypoo/mtuci/mtuci-scala/lab2/collections/target/scala-3.7.3/classes ...
[info] running CollectionsLab

task 1:
List(1, 2, 3)
List(3)
List(2, 3)
List(2, 3)
1
(hi,12,4.5)
hi
4.5

task 2:
List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
List(python, go, scala, rust, javascript, typescript, html :D)
List(2, 4, 6, 8, 10, 12, 14, 16, 18, 20)
List(PYTHON, GO, SCALA, RUST, JAVASCRIPT, TYPESCRIPT, HTML :D)
List(2, 4, 6, 8, 10)
List(python, go, rust, javascript, typescript)
55
python-go-scala-rust-javascript-typescript-html :D
[success] Total time: 0 s, completed Nov 12, 2025, 8:28:36 PM
[info] 120. Monitoring source files for root/run...
[info] Press <enter> to interrupt or '?' for more options.
```

Рис. 2 – работа с map, filter и reduce.

Задание 3. Написать функцию высшего порядка, принимающую другую функцию как аргумент.



The screenshot shows an IDE with two tabs: 'Main.scala' and 'build.sbt'. The 'Main.scala' tab is active, displaying the following Scala code:

```
51 object CollectionsLab {  
50   def main(args: Array[String]): Unit = {  
18     println(words.reduce(_ + " " + _))  
17  
16  
15     println("\ntask 3:\n")  
14  
13     import scala.math.sqrt  
12  
11     def applyToAll[A, B](sequence: Seq[A], function: A => B): Seq[B] = {  
10       sequence.map(function)  
9     }  
8  
7     def applyTwice[A, B, C](sequence: Seq[A], function1: A => B, function2: B => C): Seq[C] = {  
6       applyToAll(applyToAll(sequence, function1), function2)  
5     }  
4  
3     println(applyToAll(List(1, 2, 3), _ * 10))  
2     println(applyTwice(List(-1, -2, -3), _.abs, sqrt(_)))  
1   }  
52 }
```

The bottom panel shows the execution output in the 'TERMINAL' tab. The prompt is 'sypoo@pop-os:~/mtuci/mtuci-scala/lab2/collections\$ sbt'. The output is as follows:

```
List(1, 2, 3)  
List(3)  
List(2, 3)  
List(2, 3)  
1  
(hi,12,4.5)  
hi  
4.5  
  
task 2:  
  
List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)  
List(python, go, scala, rust, javascript, typescript, html :D)  
List(2, 4, 6, 8, 10, 12, 14, 16, 18, 20)  
List(PYTHON, GO, SCALA, RUST, JAVASCRIPT, TYPESCRIPT, HTML :D)  
List(2, 4, 6, 8, 10)  
List(python, go, rust, javascript, typescript)  
55  
python-go-scala-rust-javascript-typescript-html :D  
  
task 3:  
  
List(10, 20, 30)  
List(1.0, 1.4142135623730951, 1.7320508075688772)  
[success] Total time: 1 s, completed Nov 12, 2025, 8:57:18 PM  
[info] 1. Monitoring source files for root/run...  
[info] Press <enter> to interrupt or '?' for more options.  
□
```

Рис. 3 – работа с функциями высшего порядка.

Задание 4. Реализовать пример каррирования функции.

```
15 println("\ntask 4:\n")
14
13 def normalMultiplyAndDouble(a: Int, b: Int): Int = 2 * a * b
12
11 def curriedMultiplyAndDouble(a: Int)(b: Int): Int = 2 * a * b
10
9 val normalResult = normalMultiplyAndDouble(5, 7)
8
7 val multiplyByTen = curriedMultiplyAndDouble(5)
6 val curiedResult = multiplyByTen(7)
5
4 val directCurriedCall = curriedMultiplyAndDouble(5)(7)
3
2 println(normalResult)
1
68 println(curiedResult)
2
3 println(directCurriedCall)
4
5 }
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
sypoo@pop-os:~/Mtuci/mtuci-scala/lab2/collections$ sbt
List(PYTHON, GO, SCALA, RUST, JAVASCRIPT, TYPESCRIPT, HTML :D)
List(2, 4, 6, 8, 10)
List(python, go, rust, javascript, typescript)
55
python-go-scala-rust-javascript-typescript-html :D

task 3:

List(10, 20, 30)
List(1.0, 1.4142135623730951, 1.7320508075688772)

task 4:

70
70
70
[success] Total time: 4 s, completed Nov 13, 2025, 2:09:43 AM
[info] 1. Monitoring source files for root/run...
[info] Press <enter> to interrupt or '?' for more options.
```

Рис. 4 – работа с каррированием функций.

Задание 5. Сравнить работу с мутабельными и иммутабельными коллекциями.

```
23 | println("\ntask 5:\n")
22 |
21 | val immutableNumbers = List(1, 2, 3)
20 | val newImmutableNumbers = immutableNumbers :+ 4
19 |
18 | println(immutableNumbers)
17 | println(newImmutableNumbers)
16 |
15 | import scala.collection.mutable.ListBuffer
14 |
13 | val buf = ListBuffer(1, 2, 3)
12 | buf += 4
11 |
10 | println(buf)
9 |
8 | val immutableMap = Map("a" -> 1)
7 | val newImmutableMap = immutableMap + ("b" -> 2)
6 |
5 | println(immutableMap)
4 | println(newImmutableMap)
3 |
2 | import scala.collection.mutable
1 | val mutableMap = mutable.Map("a" -> 1)
95 | mutableMap("b") = 2
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS

sypoo@pop-os:~/Mtuci/mtuci-scala/lab2/collections\$ sbt

List(1.0, 1.4142135623730951, 1.7320508075688772)

task 4:

70
70
70

task 5:

List(1, 2, 3)
List(1, 2, 3, 4)
ListBuffer(1, 2, 3, 4)
Map(a -> 1)
Map(a -> 1, b -> 2)
HashMap(a -> 1, b -> 2)
[success] Total time: 0 s, completed Nov 13, 2025, 2:34:53 AM
[info] 24. Monitoring source files for root/run...
[info] Press <enter> to interrupt or '?' for more options.

Рис. 5 – работа с мутабельными и иммутабельными коллекциями.

Заключение

Вывод: проделав работу, мы закрепили базовый синтаксис Scala, научились использовать функции высшего порядка (map, filter, reduce), а также поняли преимущества иммутабельных коллекций.

Контрольные вопросы

1. Назовите ключевые принципы функционального программирования. Чем они отличаются от императивного подхода?

Ключевые принципы ФП:

- Чистые функции (без побочных эффектов)
- Иммутабельность данных
- Функции высшего порядка
- Рекурсия вместо циклов
- Выражения вместо инструкций

Отличия от императивного подхода:

- ФП: что вычислять (декларативно)
- Императивное: как вычислять (последовательность команд)
- ФП: данные иммутабельны, создаются новые
- Императивное: данные мутабельны, изменяются состояния
- ФП: акцент на трансформации данных
- Императивное: акцент на изменении состояния

2. Что такое «чистые функции»? Приведите пример в Scala.

Чистая функция - функция, которая:

- Всегда возвращает одинаковый результат для одинаковых входных данных
- Не имеет побочных эффектов (не изменяет внешнее состояние)

// Чистая функция

```
def add(a: Int, b: Int): Int = a + b
```

```
// НЕ чистая функция
```

```
var counter = 0
```

```
def increment(): Int = {
```

```
    counter += 1 // побочный эффект
```

```
    counter
```

```
}
```

3. Объясните термин «иммутабельность». Почему она важна в ФП?

Иммутабельность - свойство объектов, которые нельзя изменить после создания.

Важность в ФП:

- Потокбезопасность (нет гонки данных)
- Предсказуемость поведения
- Упрощение отладки
- Возможность кеширования результатов
- Облегчение композиции функций

4. В чем разница между val и var? Когда что использовать?

val - иммутабельная ссылка (нельзя переназначить)

var - мутабельная ссылка (можно переназначить)

```
val name = "Alice" // Нельзя изменить
```

```
var age = 25 // Можно изменить: age = 26
```

Когда что использовать:

- Используйте val по умолчанию
- Используйте var только когда действительно нужно изменять значение

5. Что такое case-классы? Какие преимущества они дают?

Case-классы - специальный тип классов с автоматической реализацией полезных методов.

Преимущества:

- Автоматические equals/hashCode
- Автоматический toString
- Копирование с изменением (copy)
- Возможность использования в pattern matching
- Компаньон-объект с apply

```
case class Person(name: String, age: Int)
```

```
val person = Person("Alice", 25) // Не нужен 'new'
```

6. Как работает pattern matching в Scala? Приведите пример.

Pattern matching - мощный механизм сопоставления с образцом, похожий на switch, но более выразительный.

```
def describe(x: Any): String = x match {
```

```
  case 1 => "один"
```

```
  case "hello" => "приветствие"
```

```
  case List(1, 2, 3) => "список 1,2,3"
```

```
  case Person(name, age) => s"Человек: $name, $age лет"
```

```
  case _ => "что-то другое"
```

```
}
```

7. Что такое функция высшего порядка? Приведите пример из стандартной библиотеки Scala.

Функция высшего порядка - функция, которая принимает другие функции как параметры или возвращает функцию как результат.

```
// exists - проверяет, существует ли элемент, удовлетворяющий условию
```

```
val hasEven = List(1, 3, 5, 7).exists(_ % 2 == 0) // false
```

```
val hasNegative = List(1, -2, 3).exists(_ < 0) // true
```

```
// takeWhile - берет элементы из коллекции, пока условие истинно
```

```
val numbers = List(2, 4, 6, 7, 8, 10)
```

```
val evenPrefix = numbers.takeWhile(_ % 2 == 0) // List(2, 4, 6)
```

```
// sortWith - сортирует коллекцию с использованием функции сравнения
```

```
val names = List("Alice", "Bob", "Charlie")
```

```
val sortedByLength = names.sortWith(_.length < _.length) // List("Bob", "Alice",  
"Charlie")
```

```
// collect - комбинирует map и filter, применяя частичную функцию
```

```
val mixed = List(1, "hello", 2, "world", 3)
```

```
val onlyInts = mixed.collect { case x: Int => x * 2 } // List(2, 4, 6)
```

8. Объясните, как работают map, filter и reduce для коллекций.

map - преобразует каждый элемент коллекции

filter - оставляет только элементы, удовлетворяющие условию

reduce - агрегирует все элементы в один результат

```
List(1, 2, 3).map(_ * 2) // List(2, 4, 6)
```

```
List(1, 2, 3, 4).filter(_ > 2) // List(3, 4)
```

```
List(1, 2, 3, 4).reduce(_ + _) // 10
```

9. Что такое каррирование? Как оно реализуется в Scala?

Каррирование - преобразование функции от многих аргументов в последовательность функций от одного аргумента.

```
// Обычная функция
```

```
def add(x: Int, y: Int): Int = x + y
```

```
// Каррированная версия
```

```
def curriedAdd(x: Int)(y: Int): Int = x + y
```

```
// Частичное применение
```

```
val addFive = curriedAdd(5)
```

```
val result = addFive(3) // 8
```

10. Что такое алгебраические типы данных (ADT)? Приведите пример.

ADT - композитные типы, создаваемые комбинацией других типов через:

- Произведение (product types) - например, case-классы
- Суммы (sum types) - например, sealed traits

```
// ADT пример: представление арифметических выражений
```

```
sealed trait Expr
```

```
case class Number(n: Int) extends Expr
```

```
case class Add(left: Expr, right: Expr) extends Expr
```

```
case class Multiply(left: Expr, right: Expr) extends Expr
```

```
// Использование
```

```
val expression = Add(Number(2), Multiply(Number(3), Number(4)))
```