



# НЕЯВНЫЕ ПАРАМЕТРЫ

# СУММИРУЕМ СПИСКИ

```
def concatAll(list: List[String]): String =  
    list.foldLeft("")((x, y) => x + y)
```

```
def sumAll(list: List[Int]): Int =  
    list.foldLeft(0)((x, y) => x + y)
```

```
def forAll(list: List[Boolean]): Boolean =  
    list.foldLeft(true)((x, y) => x && y)
```

# СУММИРУЕМ СПИСКИ

```
scala> sumAll(List(1,2,3))
```

```
res0: Int = 6
```

```
scala> concatAll(List("hello", " ", "world", "!"))
```

```
res1: String = hello, world!
```

# ВЫНОСИМ ЗА СКОБКИ

```
trait Monoid[A]{  
  def empty: A  
  def combine(x: A, y: A): A  
}  
  
def combineAll[A](list: List[A])(monoid: Monoid[A]): A =  
  list.foldLeft(monoid.empty)(monoid.combine)
```

# ВЫНОСИМ ЗА СКОБКИ

```
object IntMonoid extends Monoid[Int]{  
  override def empty = 0  
  override def combine(x: Int, y: Int) = x + y  
}  
  
object StringMonoid extends Monoid[String]{  
  override def empty = ""  
  override def combine(x: String, y: String) = x + y  
}  
  
object BooleanMonoid extends Monoid[Boolean]{  
  override def empty = true  
  override def combine(x: Boolean, y: Boolean) = x && y  
}
```

# ВЫНОСИМ ЗА СКОБКИ

```
scala> combineAll(List(1,2,3))(IntMonoid)
```

```
res3: Int = 6
```

# НЕЯВНЫЕ ПАРАМЕТРЫ

```
def combineAll[A](list: List[A])(monoid: Monoid[A]): A =  
  list.foldLeft(monoid.empty)(monoid.combine)
```

# НЕЯВНЫЕ ПАРАМЕТРЫ

`implicit`

```
def combineAll[A](list: List[A])(monoid: Monoid[A]): A =  
  list.foldLeft(monoid.empty)(monoid.combine)
```



# НЕЯВНЫЕ ПАРАМЕТРЫ

`implicit`



```
def combineAll[A](list: List[A])(implicit monoid: Monoid[A]): A =  
  list.foldLeft(monoid.empty)(monoid.combine)
```

# НЕЯВНЫЕ ПАРАМЕТРЫ

```
object IntMonoid extends Monoid[Int]{  
  override def empty = 0  
  override def combine(x: Int, y: Int) = x + y  
}  
  
object StringMonoid extends Monoid[String]{  
  override def empty = ""  
  override def combine(x: String, y: String) = x + y  
}  
  
object BooleanMonoid extends Monoid[Boolean]{  
  override def empty = true  
  override def combine(x: Boolean, y: Boolean) = x && y  
}
```

# НЕЯВНЫЕ ПАРАМЕТРЫ

`implicit`

```
object IntMonoid extends Monoid[Int]{  
  override def empty = 0  
  override def combine(x: Int, y: Int) = x + y  
}  
  
object StringMonoid extends Monoid[String]{  
  override def empty = ""  
  override def combine(x: String, y: String) = x + y  
}  
  
object BooleanMonoid extends Monoid[Boolean]{  
  override def empty = true  
  override def combine(x: Boolean, y: Boolean) = x && y  
}
```

# НЕЯВНЫЕ ПАРАМЕТРЫ

implicit

→ implicit object IntMonoid extends Monoid[Int]{  
 override def empty = 0  
 override def combine(x: Int, y: Int) = x + y  
}

→ implicit object StringMonoid extends Monoid[String]{  
 override def empty = ""  
 override def combine(x: String, y: String) = x + y  
}

→ implicit object BooleanMonoid extends Monoid[Boolean]{  
 override def empty = true  
 override def combine(x: Boolean, y: Boolean) = x && y  
}

# НЕЯВНЫЕ ПАРАМЕТРЫ

```
object Monoid {  
  implicit object IntMonoid extends Monoid[Int]{  
    override def empty = 0  
    override def combine(x: Int, y: Int) = x + y  
  }  
  
  implicit object StringMonoid extends Monoid[String]{  
    override def empty = ""  
    override def combine(x: String, y: String) = x + y  
  }  
  
  implicit object BooleanMonoid extends Monoid[Boolean]{  
    override def empty = true  
    override def combine(x: Boolean, y: Boolean) = x && y  
  }  
}
```

# НЕЯВНЫЕ ПАРАМЕТРЫ

```
scala> combineAll(List(1, 2, 3))
```

```
res4: Int = 6
```

# ВЫВОД ТИПОВ

```
combineAll[A](List(1,2,3) : List[A])(monoid: Monoid[A])
```

# ВЫВОД ТИПОВ

```
combineAll[A](List(1,2,3) : List[A])(monoid: Monoid[A])
```



```
combineAll[A](List(1,2,3) : List[Int])(monoid: Monoid[A])
```



# ВЫВОД ТИПОВ

```
combineAll[A](List(1,2,3) : List[A])(monoid: Monoid[A])
```



```
combineAll[A](List(1,2,3) : List[Int])(monoid: Monoid[A])
```



```
combineAll[Int](List(1,2,3) : List[Int])(monoid: Monoid[A])
```

# ВЫВОД ТИПОВ

```
combineAll[A](List(1,2,3) : List[A])(monoid: Monoid[A])
```



```
combineAll[A](List(1,2,3) : List[Int])(monoid: Monoid[A])
```



```
combineAll[Int](List(1,2,3) : List[Int])(monoid: Monoid[A])
```

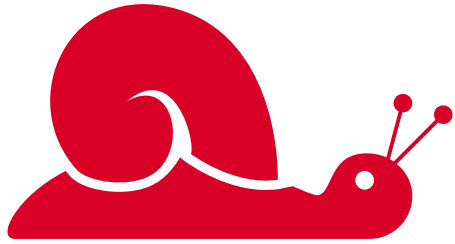


```
combineAll[Int](List(1,2,3) : List[Int])(monoid: Monoid[Int])
```

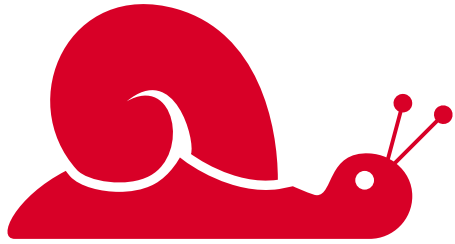
implicit



implicit

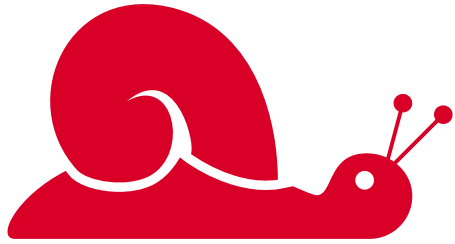


implicit

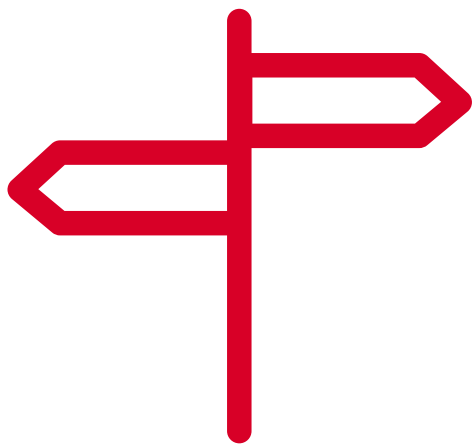


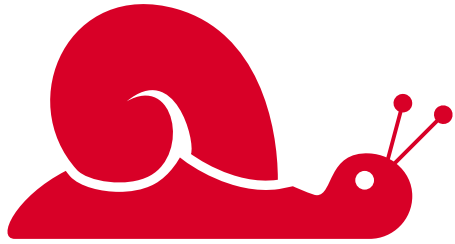
implicit



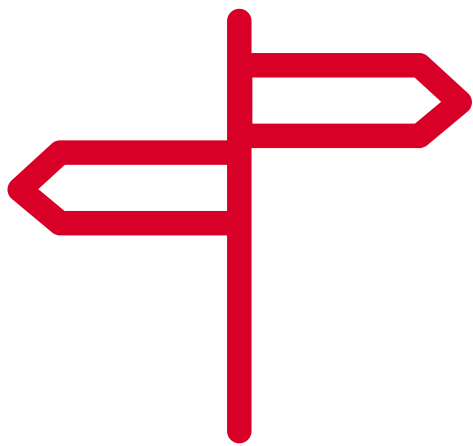


implicit





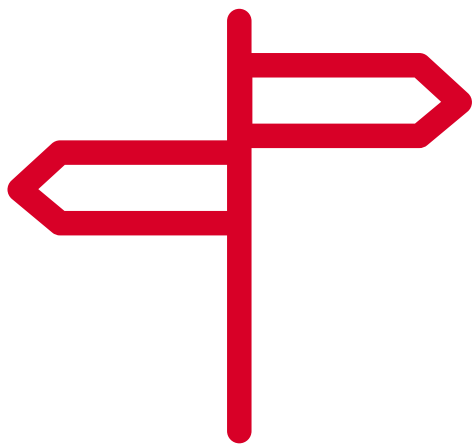
implicit



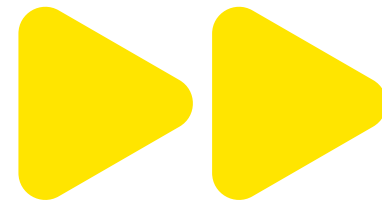




implicit



**В этом разделе  
мы изучили модификаторы**



**В следующем изучим  
обобщенные типы**