Theory: Basic literals

8 minutes 0 / 5 problems solved

All programming languages give you the ability to describe basic data: booleans, numerics or floating-point numbers. We need these building blocks to work with data, define constants and form expressions. When you want to define the mathematical constant Pi to calculate the area of a circle, you have to enter its value (say, 3.14159 is enough for our computation). This representation of a fixed value in source code is called a **literal**.

Here we'll consider literals forming the basic set of types. The **type** describes a class of values we can use (say, floating-point numbers), and the **literal** defines one specific value from this class (3.14159). For each type, we can define the allowed literals and how to use them correctly. Let's start with the type with only one allowed literal and then move on to more complicated types.

§1. Unit

Scala has one specific type with only one existing value: **Unit**. For example, the main function of any program returns Unit:

```
object Main {
  def main(args: Array[String]): Unit = {
     println("Returning Unit")
  }
}
```

You can understand the Unit type literally as void, that is, it corresponds to no value. You can use a special literal to define the value of Unit: `()`. For example, we could write a program like this:

```
object Main {
  def main(args: Array[String]): Unit = {
     () // Here we return Unit
  }
}
```

§2. Booleans

Scala gives you the ability to work with boolean literals. There are two of them, `true` and `false`:

```
scala> true
res1: Boolean = true
scala> 2 + 2 == 5
res2: Boolean = false
```

§3. Integer numbers

Scala allows using a wide variety of numerics. Let start with **integers**:

```
scala> 1
res1: Int = 1
```

To be precise, we can use Byte (8-bit, signed), Short (16-bit, signed), Char (16-bit, unsigned), Int (32-bit, signed) numbers, but all of these numeric types use digits when defining literals.

If 32-bit is not enough, we can switch to signed Long (64-bit):

```
scala> -4L
res2: Long = -4
```

Note that we have to use the L postfix for the literal here.

Current topic:

Basic literals

Topic depends on:

× Overview of the basic program

Topic is required for:

Values and variables

Table of contents:

We can use floating-point numbers Float (32-bit) and Double (64-bit):

```
scala> 123.456f
res3: Float = 123.456
scala> -789.0
res4: Double = -789.0
```

Note that here we have to use the `f` postfix for the Float literal.

§4. Characters

Characters are 16-bit unsigned integers that have representation as symbols. We can define it directly in quotation marks like here:

```
scala> 'a'
res1: Char = a
scala> 98.toChar
res2: Char = b
```

Note that in this case, there should be only one character. The sequence of characters forms Strings.

Characters in Scala have some convenient methods like `isDigit`, `isLetter`, `isWhitespace` that return the boolean check result:

```
scala> 'a'.isDigit
res17: Boolean = false
scala> 'a'.isLetter
res18: Boolean = true
```

§5. Conclusion

Now you know the basic Scala literals! You can use them to define data in your programs. Scala does not limit the value types to those described above, so you can define another, but let's play with it later.

Report a typo

22 users liked this theory. 2 didn't like it. What about you?

Start practicing

Comments (0) Hints (0) Useful links (0) Show discussion