大数据第9周

1. Scala集成开发环境idea搭建

1.1下载解压idea

解压：tar -zxvf ideaIE-2019.3.3.tar.gz

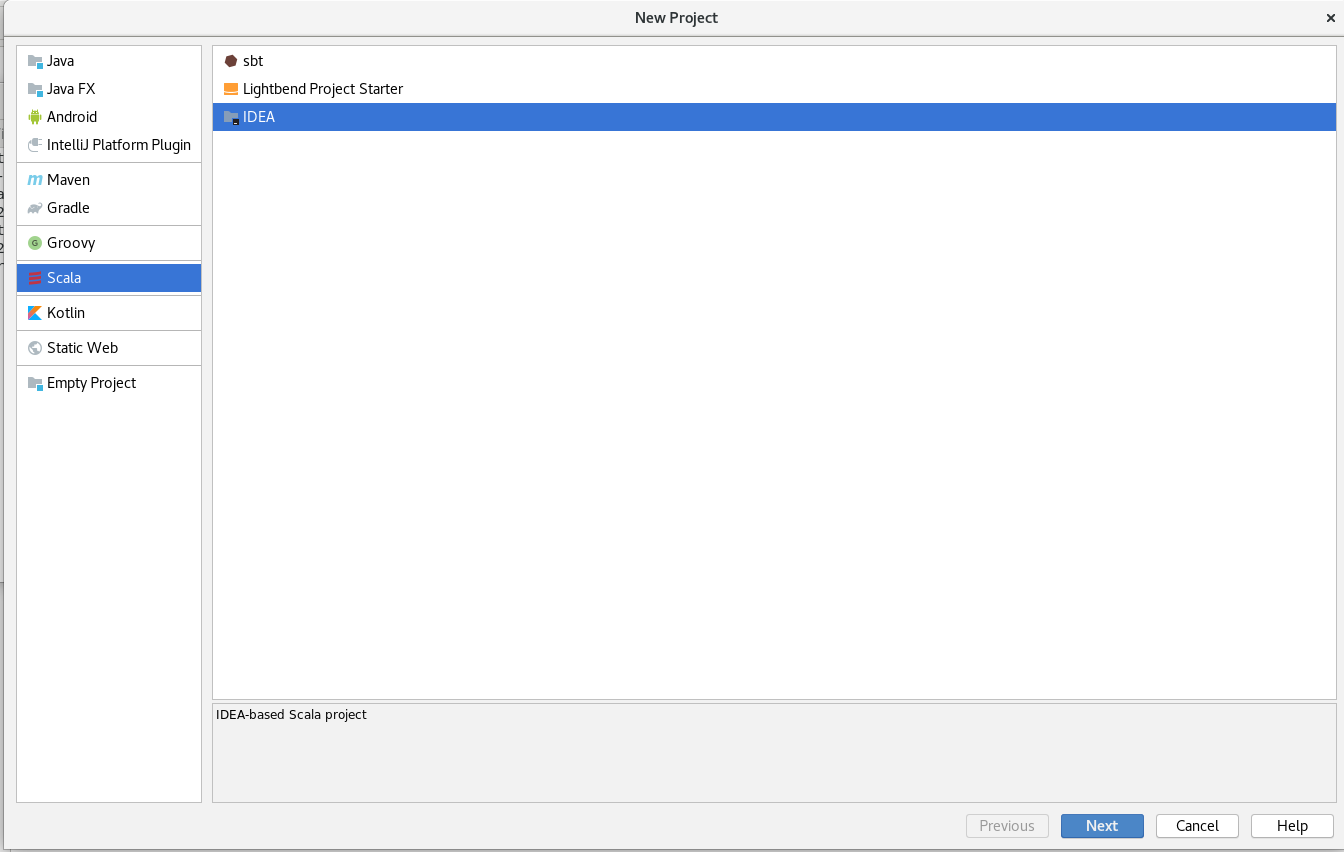
1.2初始化idea

进入bin目录，执行./idea.sh。

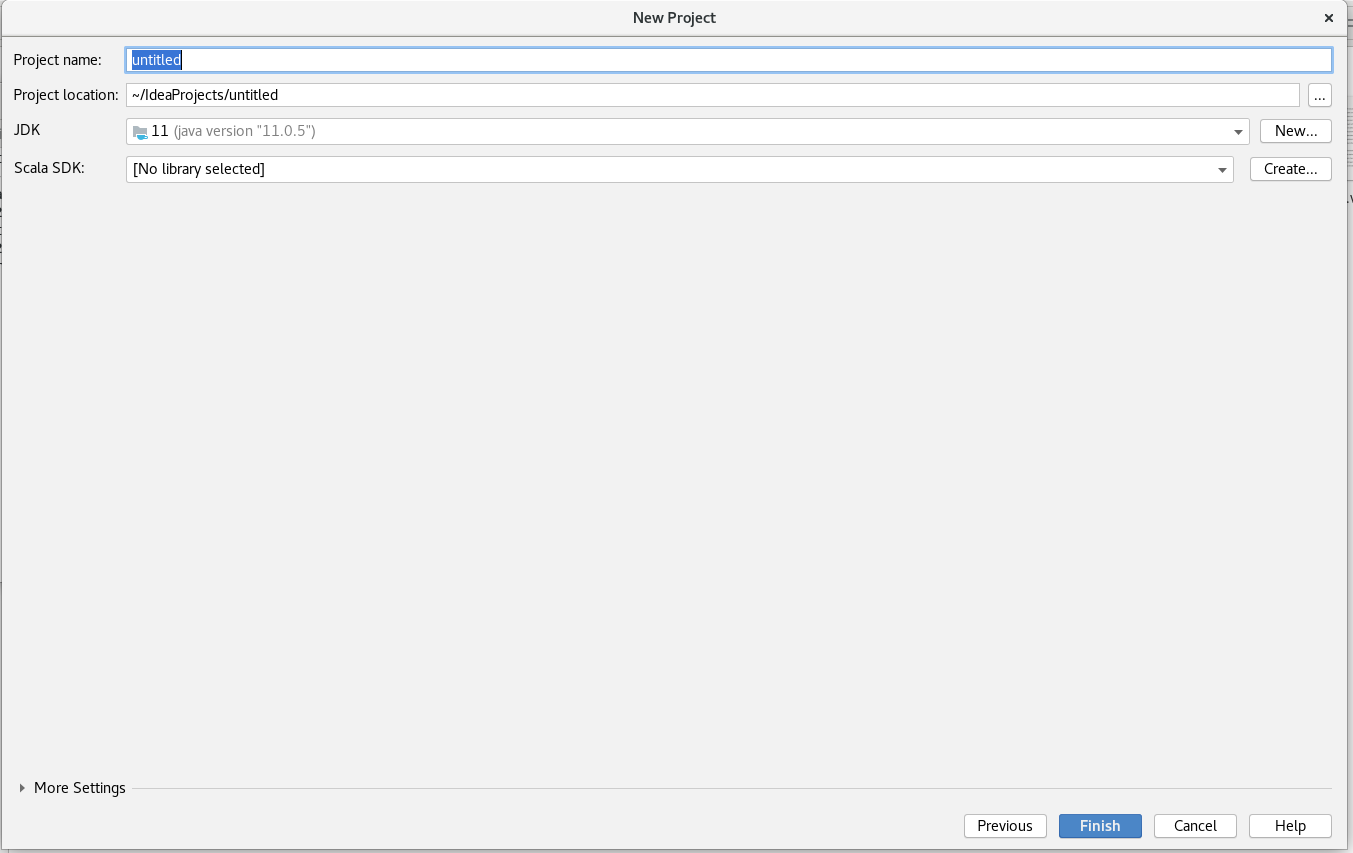
一路默认，到达：



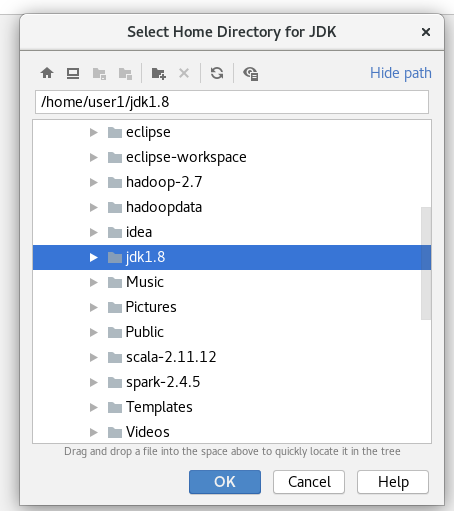
选择“Create New Project”。按下图选择，然后点击“Next”。



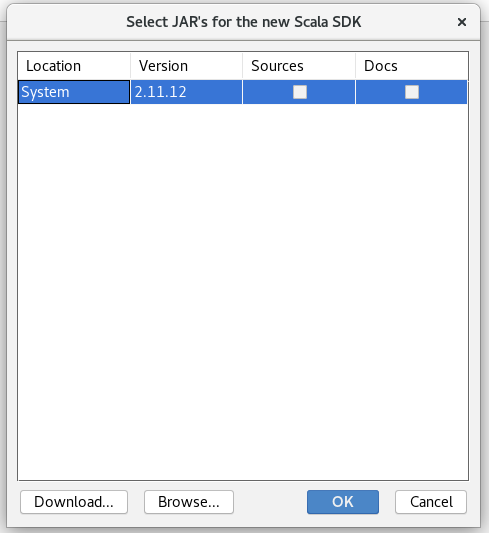
在“Project name”栏里输入名字，JDK栏要选择jdk1.8，因为scala2.11.12是基于jdk1.8的。由于我们要在hadoop框架下运行spark程序，所以需要尽量选择同一个jdk1.8的包。点击JDK栏右侧的“New”。



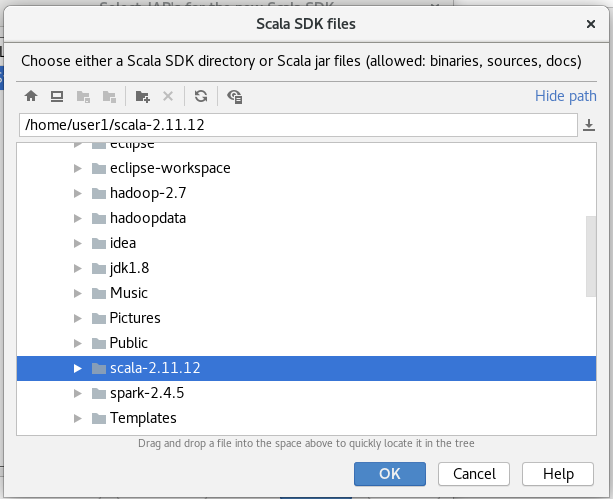
选中我们hadoop集群用的jdk1.8文件夹，点击下方“ok”。



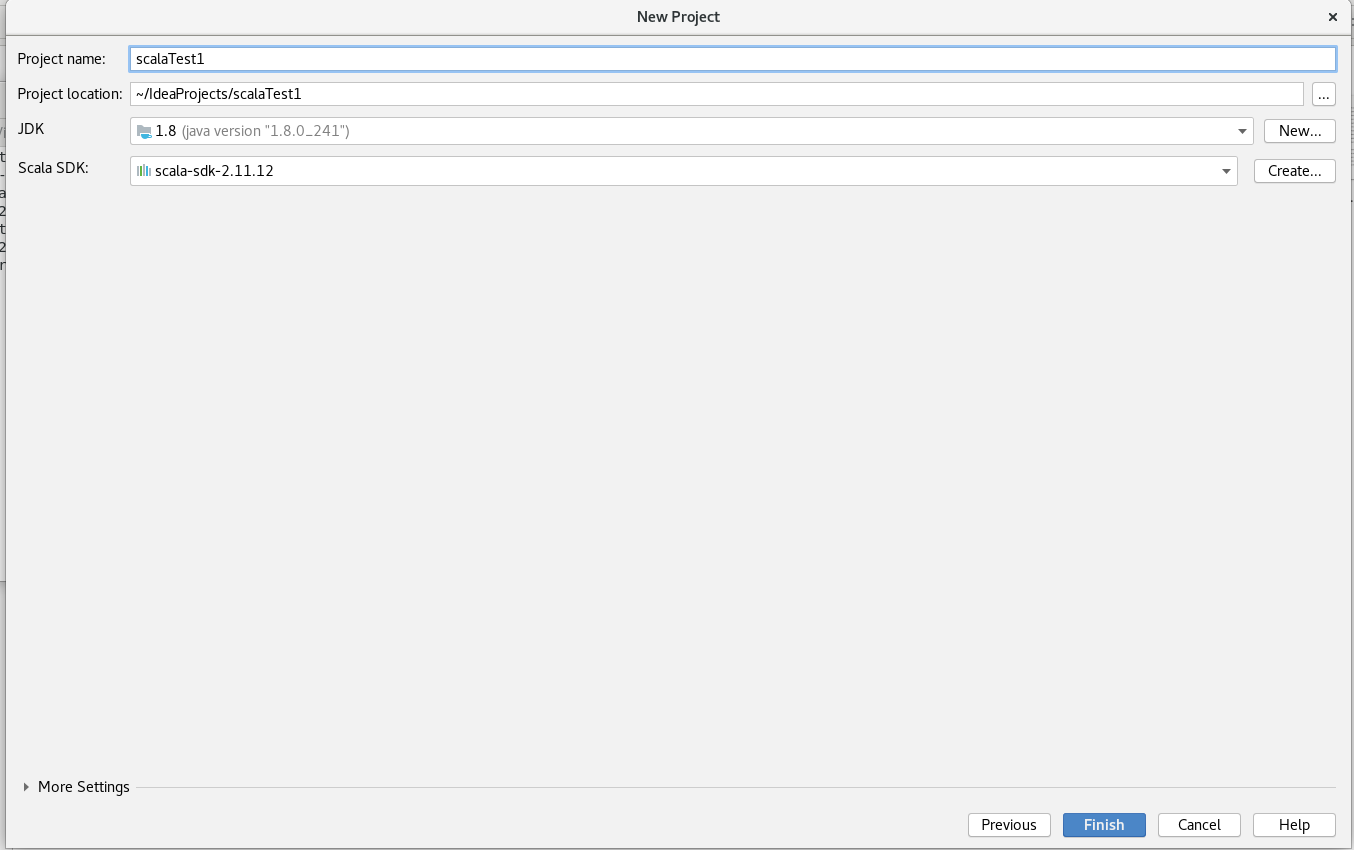
点击“Scala SDK”右侧的“Create”。



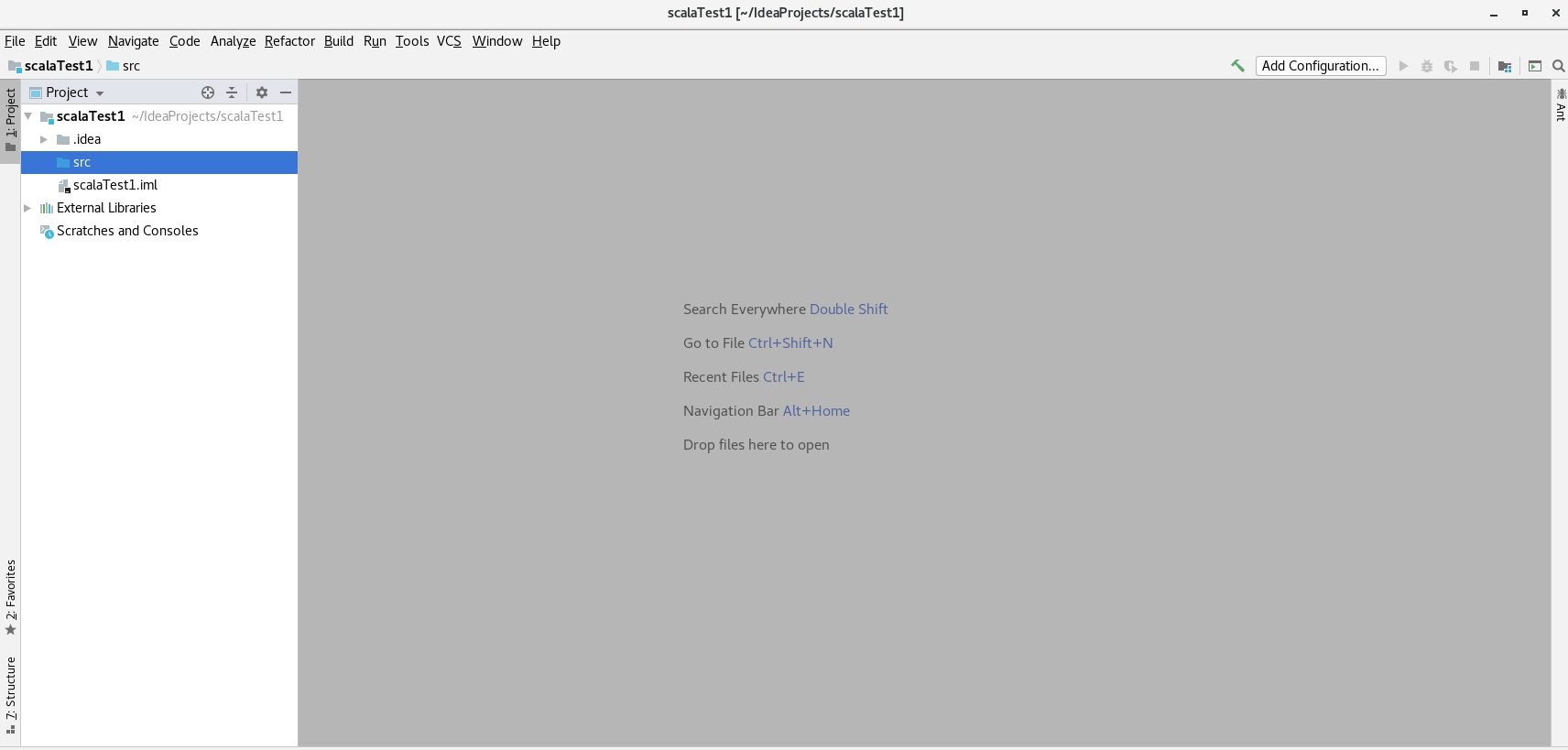
如果没有可选项，点击下方“Browse”。



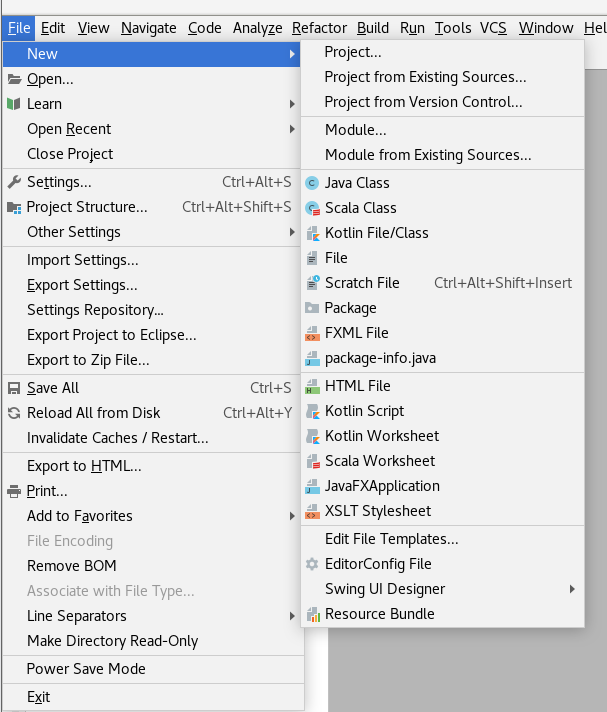
选中我们使用的scala文件夹，然后点击下方“OK”。可看到如下正确配置的界面，点击下方的“Finish”，就建立了第一个scala project。



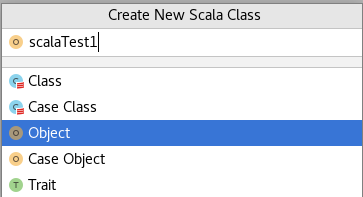
初始化后到达如下界面：



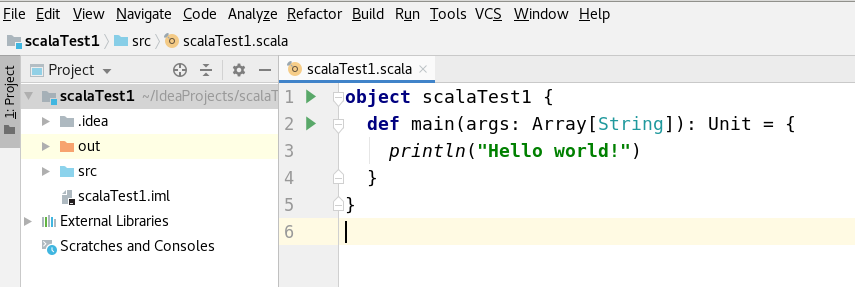
左键点击左侧的src，然后主菜单选择“File”，“New”：



选择“Scala Class”。然后，填入名字，选择“Object”。回车。



输入一段程序：



在程序上点右键，选择“Run”，可以看到执行结果。

1. scala语言：表达式和条件式

2.1表达式

返回值：不使用return，而是最后一句执行的就是返回语句。

2.2 if else表达式

Scala只有if else形式，没有if、else if、else if、else形式。

Scala没有？表达式，而要使用if else来代替。

2.3匹配模式

类似其他语言的case语句。

通配模式匹配：使用“other”，或者“\_”，当所有选项都没有选中时执行这个语句。

模式哨卫匹配：在匹配表达式中使用if表达式。

用模式变量类型匹配。

2.4 for循环

2.4.1没有yield的for循环：

object scalaTest1 {

def main(args: Array[String]): Unit = {

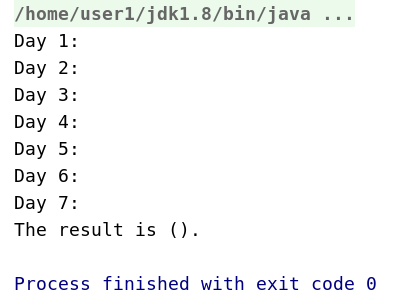
val res=for(x <- 1 to 7) println(s"Day $x:")

println(s"The result is $res.")

}

}

运行结果：



从结果可以看出，没有yield就没有返回值。

* + 1. yield的for循环：

object scalaTest1 {

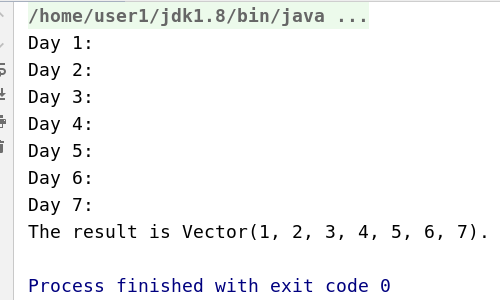
def main(args: Array[String]): Unit = {

val res=for(x <- 1 to 7) yield {println(s"Day $x:");x}

println(s"The result is $res.")

}

}



从运行结果可以看到返回值是一个Vector。

2.4.3迭代器哨卫

就是在循环中有if语句。

object scalaTest1 {

def main(args: Array[String]): Unit = {

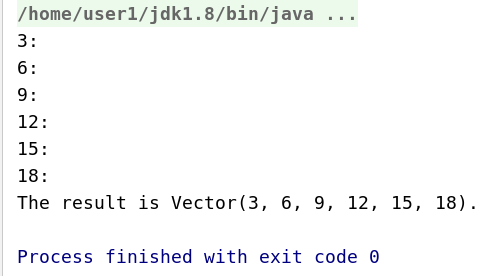
val res=for(i <- 1 to 20 if i%3==0) yield {println(s"$i:");i}

println(s"The result is $res.")

}

}

运行结果：



2.4.4 for循环中的值绑定

第一段程序：

object scalaTest1 {

def main(args: Array[String]): Unit = {

val res=for(i <-0 to 8 )yield {var pow=1<<i;pow}

println(res)

}

}

第二段程序（for循环中的值绑定）

object scalaTest1 {

def main(args: Array[String]): Unit = {

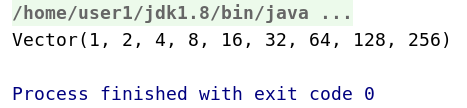
val res=for(i <-0 to 8； pow=1<<i;)yield pow

println(res)

}

}

上两段程序的运行结果是一样的：



* 1. while/do while循环

1. 函数

3.1 无参数、不指定返回类型的函数

定义方式1：

object scalaTest1 {

def main(args: Array[String]): Unit = {

def fHi="hi"

val res=fHi

println(res)

}

}

定义方式2：加空括弧

object scalaTest1 {

def main(args: Array[String]): Unit = {

def fHi()="hi"

val res=fHi()

println(res)

}

}

两种定义方式结果是没有区别的，加空括号仅仅是为了看起来是一个函数。

3.2 指定返回类型的函数

用冒号指定返回类型。例如：

def fHi:String="hi"

3.3指定参数类型和返回类型的函数

object scalaTest1 {

def div(x:Int,y:Int):Float={val tmp:Float=x;tmp/y}

def main(args: Array[String]): Unit = {

val res=div(3,5)

println(res)

}

}

运行结果：0.6

3.4递归函数

递归函数可以提高代码的利用率，但有“栈溢出”危险，所以需要仔细处理终点条件。

object scalaTest1 {

def power(x:Int,n:Int):Long={if(n>=1) x\*power(x,n-1) else 1}

def main(args: Array[String]): Unit = {

val res=power(2,3)

println(res)

}

}

计算结果是：8。

3.5调用函数时使用命名参数

就是调用函数时，参数的传递采用：“形参=实参”的形式，这样好处是：参数传递不用按照形参顺序。

3.6有默认参数值的函数

函数定义时可以给形参一个值，作为默认参数，一般情况下，有默认值的形参放到形参表的尾部。

如果有默认值的形参不在形参表的尾部，函数调用时如果使用默认值，就需要使用命名参数调用形式。

第一段程序（正常默认参数位置）：

object scalaTest1 {

def sum(x:Int,y:Int,z:Int=3):Long={x+y+z}

def main(args: Array[String]): Unit = {

val res=sum(2,4,6)

println(res)

}

}

第二段程序（由于默认参数在第一个，所以传参时，需要传三个参数，或者使用“形参=实参”形式）：

object scalaTest1 {

def sum(x:Int=1,y:Int,z:Int):Long={x+y+z}

def main(args: Array[String]): Unit = {

val res=sum(y=2,z=4)

println(res)

}

}

3.7 vararg参数

就是可以输入个数可变的参数。定义方法是在形参类型后加“\*”。

object scalaTest1 {

def sum(x:Int\*):Long={var total=0; for(i<-x) total+=i;total}

def main(args: Array[String]): Unit = {

val res=sum(2,4,5,6,7,8)

println(res)

}

}

结果是：32。