

Министерство образования Республики Беларусь
Учреждение образования Белорусский государственный университет
информатики и радиоэлектроники

Кафедра ЭВМ

Отчёт по лабораторной работе №2
“ Работа со списками и функциями ”

Проверил:

Выполнил:

Минск 2023

1. Цель

Работа со списками и функциями.

2. Краткие теоретические сведения

Остановимся на использовании функций для работы со списками
Примеры функций для работы со списками в Scala:

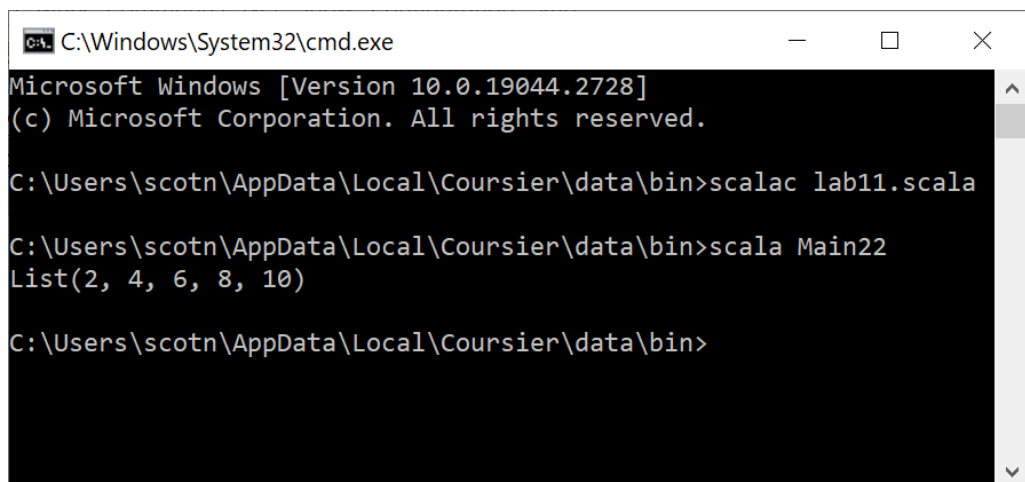
1. **Map (карта, отображение)** : эта функция применяет заданную функцию к каждому элементу списка и возвращает новый список с результатами.

```
val numbers = List(1, 2, 3, 4, 5)
val doubled = numbers.map(x => x * 2) // List(2, 4, 6, 8, 10)

object Main22 {

  def double(x: Int): Int = x * 2

  def main(args: Array[String]): Unit = {
    val myList = List(1, 2, 3, 4, 5)
    val doubledList = myList.map(double)
    println(doubledList) // Output: List(2, 4, 6, 8, 10)
  }
}
```



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.2728]
(c) Microsoft Corporation. All rights reserved.

C:\Users\scotn\AppData\Local\Coursier\data\bin>scalac lab11.scala

C:\Users\scotn\AppData\Local\Coursier\data\bin>scala Main22
List(2, 4, 6, 8, 10)

C:\Users\scotn\AppData\Local\Coursier\data\bin>
```

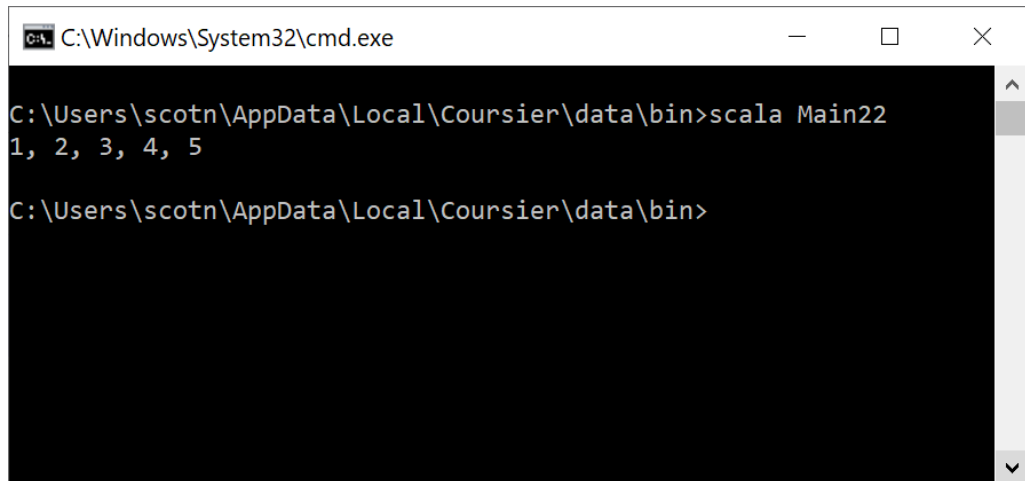
Поэлементный вывод списка:

```
object Main22 {

  def double(x: Int): Int = x * 2

  def main(args: Array[String]): Unit = {
    val myList = List(1, 2, 3, 4, 5)
    val doubledList = myList.map(double)
```

```
println(myList.mkString(", "))  
}  
}
```

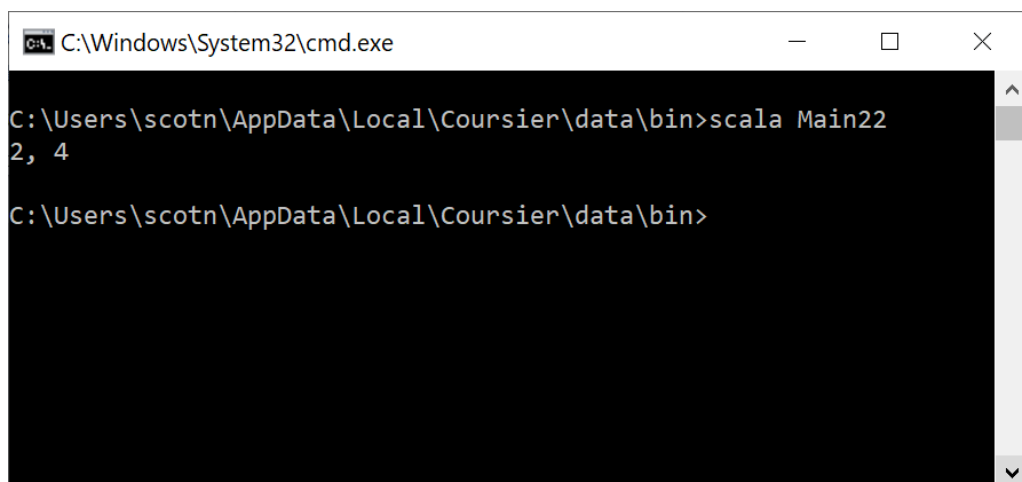


```
C:\Windows\System32\cmd.exe  
C:\Users\scotn\AppData\Local\Coursier\data\bin>scala Main22  
1, 2, 3, 4, 5  
C:\Users\scotn\AppData\Local\Coursier\data\bin>
```

2. filter: Эта функция отбирает элементы списка, удовлетворяющие заданному предикату.

Следующий пример показывает, как вывести четные элементы списка:

```
object Main22 {  
  
  def isEven(x: Int): Boolean = x % 2 == 0  
  
  def main(args: Array[String]): Unit = {  
    val myList = List(1, 2, 3, 4, 5)  
  
    val filteredList = myList.filter(isEven)  
    println(filteredList.mkString(", ")) // Output: List(2, 4)  
  
  }  
}
```



```
C:\Windows\System32\cmd.exe  
C:\Users\scotn\AppData\Local\Coursier\data\bin>scala Main22  
2, 4  
C:\Users\scotn\AppData\Local\Coursier\data\bin>
```

3. foldLeft: Эта функция последовательно применяется к элементам списка слева направо, накапливая результат. Сумму элементов списка можно найти таким образом

```
object Main22 {  
  
  def main(args: Array[String]): Unit = {  
    val myList = List(1, 2, 3, 4, 5)  
    val sum = myList.foldLeft(0)((ac_c, x) => ac_c + x)  
    println(sum) // Output: 15  
  
  }  
}
```

Здесь переменная `ac_c` играет роль аккумулятора. Первоначально ей присваивается значение 0:

```
myList.foldLeft(0)
```

4. zip: Эта функция объединяет два списка на примере словаря (dictionary) – ключ-значение.

```
val a = List(1, 2, 3)  
val b = List("one", "two", "three")  
val zipped = a.zip(b) // List((1, "one"), (2, "two"), (3, "three"))
```

Потом возникает вопрос, как взять значение из пары в списке по ключу. Последовательно покажем решение этой задачи:

```
object Main22 {  
  def main(args: Array[String]): Unit = {  
    val a = List(1, 2, 3)  
    val b = List("one", "two", "three")  
    val zipped = a.zip(b) // List((1, "one"), (2, "two"), (3,  
    "three"))  
    println(zipped)  
    val filteredList = zipped.filter { case (a, _) => a == 1 }  
    println(filteredList)  
    val tup = filteredList.head  
    println(tup)  
    val secondItem = tup._2  
    println(secondItem)  
  }  
}
```

Обратимся к рекурсивным функциям. Со списками их показывать очень удобно.

6. head and tail: Эти функции возвращают голову и хвост списка соответственно.

```
val numbers = List(1, 2, 3, 4, 5)
```

```
val first = numbers.head // 1
val rest = numbers.tail // List(2, 3, 4, 5)
```

7. reverse: Эта функция возвращает список в обратном порядке.

```
val numbers = List(1, 2, 3, 4, 5)
val reversed = numbers.reverse // List(5, 4, 3, 2, 1)
```

3. Порядок выполнения работы

- Изучить теоретическую часть.
- Получить индивидуальное задание у преподавателя из прилагаемого списка.
- Создать и отладить приложение в среде Scala.
- Написать отчет.
- Защитить работу.

4. Индивидуальное задание

Вариант 1

1. Написать функцию для подсчета суммы элементов списка, значение которых не превосходит 10. Список задать самостоятельно.
2. Написать функцию для подсчета суммы первых трех элементов списка из 10 элементов. Список задать самостоятельно.
3. Написать функцию для отыскания (минимального) индекса максимального элемента списка. Список задать самостоятельно.
4. Написать функцию для проверки того, что список упорядочен по возрастанию. Список задать самостоятельно.
5. Написать функцию для проверки наличия одинаковых элементов в списке. Список задать самостоятельно. Функция возвращает значение Да или Нет.

Листинг программы:

Меню программы:

```
def main(args: Array[String]): Unit = {
  var continue = true
  while (continue) {
    print("Сделайте выбор: \n")
    println("1. Сумма чисел списка не более 10;")
    println("2. Сумма первых 3 элементов списка;")
    println("3. Поиск индекса максимального или")
    println("минимального элемента списка;")
  }
}
```

```

println("4. Проверка списка на упорядоченность по
возрастанию;")
println("5. Проверка на наличие одинаковых элементов
в списке;")
println("0. Выход.")
scala.io.StdIn.readLine() match {
  case "1" => sumList()
  case "2" => sumThreeFirst()
  case "3" => findMaxOrMinIndex()
  case "4" => sortedList()
  case "5" => sameInList()
  case "0" =>
    println("Завершение программы.")
    continue = false
  case _ => println("Параметр не найден!")
}
if (continue) {
  var continueChoice = ""
  while (continueChoice != "1" && continueChoice !=
"0") {
    println("Желаете выполнить еще операцию? (1 - да,
0 - выход):")
    continueChoice = scala.io.StdIn.readLine()
    if (continueChoice != "1" && continueChoice !=
"0") {
      println("Некорректный выбор. Пожалуйста,
введите 1 или 0.")
    }
  }
  if (continueChoice != "1") {
    println("Выход из программы.")
    continue = false
  }
}
}
}
}

```

Главное меню:

```

Сделайте выбор:
1. Сумма чисел списка не более 10;
2. Сумма первых 3 элементов списка;
3. Поиск индекса максимального или минимального элемента списка;
4. Проверка списка на упорядоченность по возрастанию;
5. Проверка на наличие одинаковых элементов в списке;
0. Выход.

```

1) Написать функцию для подсчета суммы элементов списка, значение которых не превосходит 10. Список задать самостоятельно.

```

object Main2 {
  private def sumList(): List[Int] = {
    println("Введите количество элементов списка:")
  }
}

```

```

    val count = scala.io.StdIn.readInt()
    def inputValuesRecursion(i: Int, acc: List[Int]):
List[Int] = {
        if (i <= count) {
            println(s"Введите элемент №$i:")
            val inputValue = scala.io.StdIn.readInt()
            inputValuesRecursion(i + 1, inputValue :: acc)
        } else {
            acc.reverse
        }
    }
    val inputValues = inputValuesRecursion(1, Nil)
    if (inputValues.nonEmpty) {
        println("Весь список: " + inputValues)
        println("Сумма чисел не более 10: ")
        def sumRecursion(values: List[Int], acc: Int): Int =
{
            values match {
                case Nil => acc
                case head :: tail =>
                    if (head < 10) {
                        sumRecursion(tail, acc + head)
                    } else {
                        sumRecursion(tail, acc)
                    }
            }
        }
        val sum = sumRecursion(inputValues, 0)
        println(sum)
        inputValues
    } else {
        println("Список пуст")
        Nil
    }
}

```

Работа функции:

```

1
Введите количество элементов списка:
5
Введите элемент №1:
1
Введите элемент №2:
2
Введите элемент №3:
3
Введите элемент №4:
4
Введите элемент №5:
5
Весь список: List(1, 2, 3, 4, 5)
Сумма чисел не более 10:
15
Желаете выполнить еще операцию? (1 - да, 0 - выход):

```

2) Написать функцию для подсчета суммы первых трех элементов списка из 10 элементов. Список задать самостоятельно.

```
private def sumThreeFirst(): List[Int] = {
  println("Введите количество элементов списка (не менее
3):")
  val count = scala.io.StdIn.readInt()
  if (count < 3) {
    println("Для данного задания требуется 3 или более
элемента!")
    return Nil
  }
  val inputValues = (1 to count).map { i =>
    println(s"Введите элемент №$i:")
    scala.io.StdIn.readInt()
  }.toList
  if (inputValues.nonEmpty) {
    println("Весь список: " + inputValues)
    println("Сумма первых 3 элементов: ")
    var sum = 0
    for (i <- 0 until 3) {
      if (i < inputValues.length) {
        sum += inputValues(i)
      }
    }
    println(sum)
    inputValues
  } else {
    println("Список пуст")
    Nil
  }
}
```

Работа функции:

```
2
Введите количество элементов списка:
4
Введите элемент №1:
1
Введите элемент №2:
2
Введите элемент №3:
3
Введите элемент №4:
4
Весь список: List(1, 2, 3, 4)
Сумма первых 3 элементов:
6
Желаете выполнить еще операцию? (1 - да, 0 - выход):
```


3) Написать функцию для отыскания (минимального) индекса максимального элемента списка. Список задать самостоятельно.

```
private def findMaxOrMinIndex(): Unit = {
  println("Введите количество элементов списка:")
  val count = scala.io.StdIn.readInt()
  val inputValues = (1 to count).map { i =>
    println(s"Введите элемент №$i:")
    scala.io.StdIn.readInt()
  }.toList
  if (inputValues.nonEmpty) {
    println("Весь список: " + inputValues)
    println("Выберите операцию:")
    println("1. Поиск индекса максимального элемента")
    println("2. Поиск индекса минимального элемента")
    val choice = scala.io.StdIn.readInt()
    val result = choice match {
      case 1 =>
        val (maxElement, maxIndex) = findMax(inputValues,
        0, 0, inputValues.head, 0)
        (maxElement, maxIndex + 1)
      case 2 =>
        val (minElement, minIndex) = findMin(inputValues,
        0, 0, inputValues.head, 0)
        (minElement, minIndex + 1)
      case _ =>
        println("Некорректный выбор.")
        return
    }
    val (extremeValue, extremeIndex) = result
    println(s"Индекс искомого элемента ($extremeValue):
    $extremeIndex")
  } else {
    println("Список пуст")
  }
}

private def findMax(list: List[Int], currentIndex: Int,
maxIndex: Int, currentMax: Int, globalMax: Int): (Int, Int) = {
  if (currentIndex < list.length) {
    val currentValue = list(currentIndex)
    if (currentValue > currentMax) {
      findMax(list, currentIndex + 1, currentIndex,
currentValue, globalMax)
    } else {
      findMax(list, currentIndex + 1, maxIndex,
currentMax, globalMax)
    }
  } else {
    if (currentMax > globalMax) {
      (currentMax, maxIndex)
    } else {
      (globalMax, maxIndex)
    }
  }
}
```

```

private def findMin(list: List[Int], currentIndex: Int,
minIndex: Int, currentMin: Int, globalMin: Int): (Int, Int) = {
  if (currentIndex < list.length) {
    val currentValue = list(currentIndex)
    if (currentValue < currentMin) {
      findMin(list, currentIndex + 1, currentIndex,
currentValue, globalMin)
    } else {
      findMin(list, currentIndex + 1, minIndex,
currentMin, globalMin)
    }
  } else {
    if (currentMin < globalMin) {
      (currentMin, minIndex)
    } else {
      (globalMin, minIndex)
    }
  }
}

```

Работа функции:

```

3
Введите количество элементов списка:
3
Введите элемент №1:
1
Введите элемент №2:
2
Введите элемент №3:
3
Весь список: List(1, 2, 3)
Выберите операцию:
1. Поиск индекса максимального элемента
2. Поиск индекса минимального элемента
1
Индекс искомого элемента (3): 3
Желаете выполнить еще операцию? (1 - да, 0 - выход):

```

4) Написать функцию для проверки того, что список упорядочен по возрастанию. Список задать самостоятельно.

```

private def isSortedAsc(list: List[Int]): Boolean = {
  isSortedAscRecursive(list)
}

private def isSortedAscRecursive(list: List[Int], index:
Int = 0): Boolean = {
  if (index < list.length - 1) {
    if (list(index) <= list(index + 1)) {
      isSortedAscRecursive(list, index + 1)
    } else {
      false
    }
  } else {
    true
  }
}

```

```

}

private def sortedList(): Unit = {
  println("Введите количество элементов списка:")
  val count = scala.io.StdIn.readInt()
  val inputValues = (1 to count).map { i =>
    println(s"Введите элемент №$i:")
    scala.io.StdIn.readInt()
  }.toList
  if (inputValues.nonEmpty) {
    println("Весь список: " + inputValues)
    val isSorted = isSortedAsc(inputValues)
    if (isSorted) {
      println("Список упорядочен по возрастанию")
    } else {
      println("Список не упорядочен по возрастанию")
    }
  } else {
    println("Список пуст")
  }
}

```

Работа функции:

```

4
Введите количество элементов списка:
3
Введите элемент №1:
1
Введите элемент №2:
2
Введите элемент №3:
3
Весь список: List(1, 2, 3)
Список упорядочен по возрастанию
Желаете выполнить еще операцию? (1 - да, 0 - выход):

```

5) Написать функцию для проверки наличия одинаковых элементов в списке. Список задать самостоятельно. Функция возвращает значение Да или Нет.

6)

```

private def hasDuplicates(list: List[Int]): Boolean = {
  hasDuplicatesRecursive(list, Set.empty)
}

private def hasDuplicatesRecursive(list: List[Int],
seenSet: Set[Int]): Boolean = {
  list match {
    case Nil => false
    case head :: tail =>
      if (seenSet.contains(head)) {
        true
      } else {
        hasDuplicatesRecursive(tail, seenSet + head)
      }
  }
}

```

```

    }
  }
}

private def sameInList(): Unit = {
  println("Введите количество элементов списка:")
  val count = scala.io.StdIn.readInt()
  val inputValues = (1 to count).map { i =>
    println(s"Введите элемент №$i:")
    scala.io.StdIn.readInt()
  }.toList

  if (inputValues.nonEmpty) {
    println("Весь список: " + inputValues)
    val hasDupe = hasDuplicates(inputValues)
    if (hasDupe) {
      println("В списке есть одинаковые элементы")
    } else {
      println("В списке нет одинаковых элементов")
    }
  } else {
    println("Список пуст")
  }
}

```

Работа функции:

```

5
Введите количество элементов списка:
3
Введите элемент №1:
1
Введите элемент №2:
2
Введите элемент №3:
3
Весь список: List(1, 2, 3)
В списке нет одинаковых элементов
Желаете выполнить еще операцию? (1 - да, 0 - выход):

```

5. Вывод

Познакомились с созданием и работой со списками и функциями.