Compilers Laboratory

DCC
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

ICEx Instituto de Ciências Exatas

UFMG

Introduction

Fernando M. Q. Pereira

fernando@dcc.ufmg.br

Is it profitable to develop Apps?

- In the world?
- In Brazil?
- Success stories?

1. Supercell: The Finnish mobile game publisher is around a $3 billion company. Most famous for its game Clash of Clans.

2. King: The Candy Crush legend is still generating a lot of money. King generated more than $1.9 billion in 2013.

3. GungHo Online: Puzzle & Dragons, brought in $650 million through Apple's App Store and $775 million through Google Play in 2013.

What were the most downloaded apps in 2019?


2019 Apps by Worldwide Downloads
App Store
Google Play
0
100M
200M
300M
400M
500M
600M
700M
800M
900M
WhatsApp
TikTok
Messenger
Facebook
Instagram
Likee
SHAREit
YouTube
Snapchat
Netflix
UC Browser
Spotify
YouTube Music
PicsArt
Uber
Wish
Amazon
Helo
Hotstar
Twitter
Top 5 New Apps
MV Master
Disney+
Jianying Vlog
Xrecorder
Sticker.ly
08:15
Maker
Disney+
Top 5 Publishers
Facebook
Google
ByteDance
Voodoo
Tencent
Source: SensorTower

And, what about in Brazil?

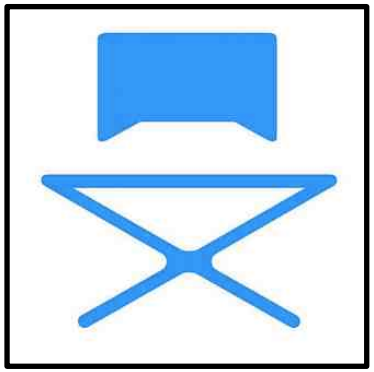
And, what about in Brazil?

Lista do Google dos melhores apps em 2018 tem 2 brasileiros

https://noticias.uol.com.br/tecnologia/noticias/redacao/2018/12/03/baixou-lista-do-google-dos-melhores-apps-em-2018-tem-2-brasileiros-veja.htm

Helton Simões Gomes, **12-03-2018**

GuiaBolso



Filmr

Dobra nº de apps que ganham R$ 1 mi no Brasil; qual a receita do sucesso?

Helton Simões Gomes
Do UOL, em São Paulo 08/11/2018 10h30

https://tecnologia.uol.com.br/noticias/redacao/2018/11/08/dobra-n-de-apps-que-ganham-r-1-mi-no-brasil-qual-a-receita-do-sucesso.htm

Brasil decola na indústria de apps e mercado acumula lucros nesta década

De um lado, o mercado consumidor do Brasil figura entre os principais do mundo; do outro, desenvolvedores nacionais assumem protagonismo numa bolada de US$ 25 bilhões. Série do Correio mostra o presente, o passado e o futuro dos aplicativos

https://www.correiobraziliense.com.br/app/noticia/tecnologia/2015/05/11/interna_tecnologia,482694/brasil-decola-na-industria-de-apps-e-mercado-acumula-lucros-nesta-deca.shtml

ECONOMIA & NEGÓCIOS

Os milionários dos aplicativos

Munidos de criatividade e espírito empreendedor, jovens brasileiros têm enriquecido na incipiente indústria de programas para smartphones e tablets

https://istoe.com.br/194216_OS+MILIONARIOS+DOS+APLICATIVOS/

Apps are implemented in which programming languages?

Apps are implemented in which programming languages?

- Native support
  - Java
  - Dart
  - C++ [*]
  - Kotlin
- Non-native support
  - C#
  - Python
  - JavaScript
  - Php
  - Lua
  - ...

*: we can't develop the entire app in C++

Why can't we simply stay with Java?

- When was Java created?
- What would you like to change in Java?

Why can't we simply stay with Java?

- Verbosity
- No REPL
- Poor support to functional programming

Why can't we simply stay with Java?

- No REPL

What's that?

Why can't we simply stay with Java?

- No REPL
  - Can you name each programming language below?

```
>>> 2 + 2
4
>>>
```

```
- 2 + 2;
val it = 4 : int
-
```

```
?- X is 2 + 2.
X = 4.
?-
```

```
# 2 + 2;;;
-: int = 4
#
```

And what's Kotlin?

- When was it created?
- Who created it?
- How does it look like?
- How can we run Kotlin programs?
- How can we install it?

Installing (Linux)

SDKMan

```
$ curl -s https://get.sdkman.io | bash
$ sdk install kotlin
```

Snap

```
$ sudo snap install --classic kotlin
```

Installing (OSX)

Homebrew

```
$ brew update
$ brew install kotlin
```

MacPorts

```
$ sudo port install kotlin
```

Installing (IDE)

How can we run Kotlin programs?

How can we run Kotlin programs?

```
fun main(args: Array<String>) {
  println("Hello, World!")
}
```

```
$ kotlinc hello.kt -include-runtime -d hello.jar
$ java -jar hello.jar
```

REPL

```
$ kotlinc
Welcome to Kotlin version 1.2.31 (JRE 1.8.0_65-b17)
Type :help for help, :quit for quit

>>> println("Hello, World!")
Hello, World!

>>> 2 + 2
4
```

Scripting

list_folders.kts

```
import java.io.File
val folders = File(args[0]).listFiles {
  file -> file.isDirectory()
}
folders?.forEach {
  folder -> println(folder)
}
```

```
$ kotlinc -script list_folders.kts ..
../FreqCounter
../Scripts
../Tests
```

Example: `PointReader.kt`

Implement a program that reads a csv file containing a list of 2D points, and prints the X coordinate of each point.

```
$ cat ~/Desktop/inst.csv
 X,  Y
 2,  3
-1,  4
-2, -1
 0,  3
 1,  2
```

Example of input

- How many loops do you need?
- How many lines of code?

Example: `PointReader.kt`

PointReader.kt

```
import java.io.File
fun main(args: Array<String>) {
  val file = File(args[0])
  val lines = file.readLines().drop(1)
  val xList = lines.map { it.split(",").get(0).toFloat() }
  xList.forEach { println(it) }
}
```

- Where are the types?
- Where are the loops?
- How to compile and run it?

Example: `PointReader.kt`

PointReader.kt

```
import java.io.File
fun main(args: Array<String>) {
  val file = File(args[0])
  val lines = file.readLines().drop(1)
  val xList = lines.map { it.split(",").get(0).toFloat() }
  xList.forEach { println(it) }
}
```

```
$ kotlinc PointReader.kt -include-runtime -d pointReader.jar
$ java -jar pointReader.jar ~/Desktop/inst.csv
2.0
-1.0
-2.0
0.0
1.0
```

Example: `PointReader.kt`

PointReader.kt

```
1  import java.io.File
2  fun main(args: Array<String>) {
3    val file = File(args[0])
4    val lines = file.readLines().drop(1)
5    val xList = lines.map { it.split(",").get(0).toFloat() }
6    xList.forEach { println(it) }
7  }
```

- **L4**: what's drop(1) doing? Why do we use it?
- **L5**: what's map?
- **L5**: what's 'it' in it.split(",")?
- **L5**: what's split(",")?

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSIDADE FEDERAL DE MINAS GERAIS
FEDERAL UNIVERSITY OF MINAS GERAIS, BRAZIL

BASIC SINTAX

What are primitive types?

- What are the primitive types in C?
- What about Java?

Primitive Types in C

- char
- short
- int
- long
- long long
- float
- double
- long double

- scalar types
  - signed
  - unsigned
  - int

```
signed long long x0 = 10;
signed long long x1 int = 10;
unsigned short = 10;
unsigned short int = 10;
```

Primitive Types in Java

- Integers
  - byte
  - chart
  - short
  - int
  - long
- Floating points
  - float
  - double
- boolean

Primitive Types in Kotlin

- Integers
  - Byte
  - Short
  - Int
  - Long
- Floating points
  - Float
  - Double

Can you explain each error that we got?

- Integers
  - Byte
  - Short
  - Int
  - Long
- Floating points
  - Float
  - Double

```
>>> val i: Int = 2
>>> val b: Byte = 8
>>> val b: Byte = 129
error: ...

>>> val d: Double = 129.0
>>> val d: Double = 129
error: ...

>>> val d: Double = 129.toDouble()
```

Type conversions must be explicit

Java

```
int n1 = 42;
long n2 = n1;
```

Kotlin

```
>>> val n1: Int = 42
>>> val n2: long = n1
error: ...
>>> val n2: Long = n1.toLong()
```

- toByte()
- toShort()
- toInt()
- toLong()
- toFloat()
- toDouble()
- toChar()

Why making conversions explicit?

- Advantages?
- Disadvantages?
- Other programming languages?

Kotlin

```
>>> val n1: Int = 42
>>> val n2: long = n1
error: ...
>>> val n2: Long = n1.toLong()
```

Inference vs Type Annotations

PointReader.kt

```
import java.io.File
fun main(args: Array<String>) {
  val pFile = File(args[0])
  val lines = pFile.readLines()
  val pOnly = lines.drop(1)
  val pList = pOnly.map { it.split(",") }
  val xStrs = pList.map { it.get(0) }
  val xList = xStrs.map { it.toFloat() }
  xList.forEach { println(it) }
}
```

- Can you guess each type?

Inference vs Type Annotations

PointReader.kt

```
import java.io.File
fun main(args: Array<String>) {
  val pFile: File = File(args[0])
  val lines: List<String> = pFile.readLines()
  val pOnly: List<String> = lines.drop(1)
  val pList: List<List<String>> = pOnly.map { it.split(",") }
  val xStrs: List<String> = pList.map { it.get(0) }
  val xList: List<Float> = xStrs.map { it.toFloat() }
  xList.forEach { println(it) }
}
```

- What's the advantage of using type inference?
- What's the advantage of using annotations?

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSIDADE FEDERAL DE MINAS GERAIS
FEDERAL UNIVERSITY OF MINAS GERAIS, BRAZIL

LISTS AND RANGES

Which operations would you expect to find in Lists?

PointReader.kt

```
import java.io.File
fun main(args: Array<String>) {
  val pFile: File = File(args[0])
  val lines: List<String> = pFile.readLines()
  val pOnly: List<String> = lines.drop(1)
  val pList: List<List<String>> = pOnly.map { it.split(",") }
  val xStrs: List<String> = pList.map { it.get(0) }
  val xList: List<Float> = xStrs.map { it.toFloat() }
  xList.forEach { println(it) }
}
```

Basic List Operations

```
>>> val items = listOf(1, 2, 3, 4)
>>> items.first()
1
>>> items.last()
4
>>> items.drop(1)
[2, 3, 4]
>>> items.contains(2)
true
>>> items.get(2)
3
>>> items.indexOf(2)
1
>>> items.subList(1, 3)
[2, 3]
>>> items.dropLast(2)
[1, 2]
```

Ranges are a simple way to build lists and iterators

```
>>> val r = 1..10
>>> r
1..10
>>> r.first()
1
>>> r.last()
10
>>> r.toList()
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> r.drop(1)
[2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> r.contains(2)
true
>>> r.indexOf(3)
2
```

Lists and Ranges gives us Control Structures

```
>>> if (2 in 1..10) { println("oi") }
oi
>>> for (i in 1..4) println(i)
1
2
3
4
>>> for (i in listOf("a", "b", "cd", "efg"))
...   println(i)
a
b
cd
efg
```

High-Order Function: `map`

```
>>> val items = listOf(2, 3, 5, 7, 11, 13)
>>> items.map { x -> x * x }
[4, 9, 25, 49, 121, 169]
>>> items.map { x -> x.toString() + "!" }
[2!, 3!, 5!, 7!, 11!, 13!]
```

- What is `x -> x * x` ?
- And what is `x -> x.toString() + "!"` ?

Lambda Expressions

```
>>> val f = {a:Int, b:Int -> a + b}
>>> f(2, 3)
5

>>> val g = {a:Float, b:Float, c:Float -> a * b + c}
>>> g(2.76F, 3.14F, 1.12F)
9.7864

>>> val items = listOf(2, 3, 5, 7, 11, 13)
>>> items.map { x -> x * x }
[4, 9, 25, 49, 121, 169]
```

Anonymous Functions

```
>>> val h = fun(L:List<Int>, e:Int, d:Int):Int {
...   val aux = L.indexOf(e)
...   return if (aux > -1) e else d
... }

>>> h(listOf(1, 2, 3), 2, 10)
2

>>> h(listOf(1, 2, 3), 4, 10)
10
```

- What's the difference between using anonymous functions (with the `fun` keyword) and lambda blocks?
- Could you rewrite the same function h, this time as a lambda expression?

Lambda Expressions are Anonymous Functions

```
>>> val h = {L:List<Int>, e:Int, d:Int ->
                    if (L.indexOf(e) > -1) e else d}

>>> h(listOf(1, 2, 3), 2, 10)
2

>>> h(listOf(1, 2, 3), 4, 10)
10
```

- When is it mandatory to use anonymous functions?

What's the type of each LX and EX?

```
>>> val items = listOf(2, 3, 5, 7)

>>> val L0 = items.map { x ->
kotlin.math.ln(x.toDouble()) }
>>> L0.javaClass
???
>>> val E0 = L0.first()
>>> E0.javaClass
???

>>> val L1 = items.map { x -> x > 4 }
>>> val E1 = L1.first()
>>> E1.javaClass
???

>>> val L2 = items.map { x -> listOf(x) }
>>> val E2 = L2.first()
>>> E2.first().javaClass
???
```

What's the type of each LX and EX?

```
>>> val items = listOf(2, 3, 5, 7)

>>> val L0 = items.map { x ->
kotlin.math.ln(x.toDouble()) }
>>> L0.javaClass
class java.util.ArrayList
>>> val E0 = L0.first()
>>> E0.javaClass
double

>>> val L1 = items.map { x -> x > 4 }
>>> val E1 = L1.first()
>>> E1.javaClass
boolean

>>> val L2 = items.map { x -> listOf(x) }
>>> val E2 = L2.first()
>>> E2.first().javaClass
class java.util.Collections$SingletonList
```

The single parameter `it`

```
>>> items.map
{ kotlin.math.ln(it.toDouble()) }
[0.69, 1.10, 1.61, 1.94]

>>> items.map { it > 4 }
[false, false, true, true]

>>> items.map { listOf(it) }
[[2], [3], [5], [7]]
```

- When can we use `it`?

`for` vs `forEach`

```
>>> val list = (1..1000).toList()
```

- What's the code above doing?

```
>>> var x:Int = 0
>>> kotlin.system.measureNanoTime{for (i in list) {x += i} }
469793
```

- And this program above, what's it doing?

```
>>> var x:Int = 0
>>> kotlin.system.measureNanoTime{ list.forEach{ x += it} }
397783
```

- Finally, what is the `forEach` doing in the code above?
- Which one is faster, `forEach` or `for`, for lists?

Inference vs Type Annotations

PointReader.kt

```
import java.io.File
fun main(args: Array<String>) {
3   val pFile = File(args[0])
4   val lines = pFile.readLines()
5   val pOnly = lines.drop(1)
6   val pList = pOnly.map { it.split(",") }
7   val xStrs = pList.map { it.get(0) }
8   val xList = xStrs.map { it.toFloat() }
9   xList.forEach { println(it) }
}
```

- Can you explain now what is doing each line above?

High-Order Function: `fold`

```
>>> listOf(1, 2, 3).fold(0) { acc, element -> acc + element }
6

>>> listOf("a", "b", "c").fold("") { acc, element -> acc + element }
abc

>>> listOf(true, false, true).fold(0) { x, e -> if (e) x + 1 else x }
2
```

- What is `fold` good for?

Fill up the question marks

```
>>> val L = listOf(2, 3, 5, 7, 11)
>>> L.fold(0) { x, e -> if (e > x) e else x }
??

>>> (L.fold(0) { x, e -> x + e }) / L.size
??
```

Fill up the question marks

```
>>> val L = listOf(2, 3, 5, 7, 11)
>>> L.fold(0) { x, e -> if (e > x) e else x }
11

>>> (L.fold(0) { x, e -> x + e }) / L.size
5
```

- What's this weird syntax: `L.fold(0){x,e->x+e}`?

The Last Parameter

```
>>> L.fold(0) { x, e -> x + e }
28

>>> L.fold(0, { x, e -> x + e })
28
```

- If the last parameter is a function, then it can be passed outside parentheses.

Using `reduce` instead of `fold`.

```
>>> L.reduce { x, e -> if (e > x) e else x }
11

>>> L.reduce { x, e -> if (e < x) e else x }
2
```

- What's the difference between `fold` and `reduce`?
- What will happen in the program below?

```
emptyList<Int>().reduce { x, e -> if (e < x) e else x }
```

Example: `Centroid.kt`

Implement a program that reads a csv file containing a list of 2D points, and prints the *centroid* of these points.

```
$ cat ~/Desktop/inst.csv
 X,  Y
 3,  3
 0,  4
-1, -2
 1,  3
 2,  2
```

Example of input

- What's a *centroid*?
- Which operations will we need?

Example: `Centroid.kt`

Centroid.kt

```
import java.io.File
fun main(args: Array<String>) {
  val file = File(args[0])
  val lines = file.readLines()
  val points = lines.drop(1)
  val xL = points.map { it.split(",").get(0).toFloat() }
  val xAvg = xL.fold(.0) { x,e -> x+e } / xL.size.toFloat()
  val yL = points.map { it.split(",").get(1).toFloat() }
  val yAvg = yL.fold(.0) { x,e -> x+e } / yL.size.toFloat()
  print("Centroid = (${xAvg}, ${yAvg})")
}
```

- What is the meaning of `${xAvg}, ${yAvg}`?
- Can you think on ways of improving this program?

External function

Centroid2.kt

```
1   import java.io.File
2   fun avg(list: List<Float>): Float {
3       return list.fold(.0F) { x,e -> x+e }/list.size.toFloat()
4   }
5   fun main(args: Array<String>) {
6       val file = File(args[0])
7       val lines = file.readLines()
8       val points = lines.drop(1)
9       val xList = points.map { it.split(",").get(0).toFloat() }
10      val xAvg = avg(xList)
11      val yList = points.map { it.split(",").get(1).toFloat() }
12      val yAvg = avg(yList)
13      print("Centroid = (${xAvg}, ${yAvg})")
    }
```

- Should we move lines 9 and 11 to within `avg`?

External function

Centroid3.kt

```
import java.io.File
fun avg2(list: List<String>, index: Int): Float {
    val fList = list.map { it.split(",").get(index).toFloat() }
    return fList.fold(.0F) { x,e -> x+e }/fList.size.toFloat()
}
fun main(args: Array<String>) {
    val file = File(args[0])
    val lines = file.readLines()
    val points = lines.drop(1)
    val xAvg = avg2(points, 0)
    val yAvg = avg2(points, 1)
    print("Centroid = (${xAvg}, ${yAvg})")
}
```

- Which function is more reusable, `avg` or `avg2`?

The Segregation Principle

Each function should have a well-defined obligation

- Which version is more reusable?

```
fun avg2(list: List<String>, index: Int): Float {
  val fList = list.map {
    it.split(",").get(index).toFloat()
  }
  val rVal = fList.fold(.0F) {
    x, e -> x + e
  }
  return rVal / fList.size.toFloat()
}
```

```
fun avg1(list: List<Float>): Float {
  val rVal = list.fold(.0F) {
    x, e -> x + e
  }
  return rVal / list.size.toFloat()
}
```

Using the Kotlin Library

Centroid4.kt

```
import java.io.File
fun main(args: Array<String>) {
  val file = File(args[0])
  val lines = file.readLines()
  val points = lines.drop(1)
  val xList = points.map { it.split(",").get(0).toFloat() }
  val xAvg = xList.average()
  val yList = points.map { it.split(",").get(1).toFloat() }
  val yAvg = yList.average()
  print("Centroid = (${xAvg}, ${yAvg})")
}
```

- What are the advantages of using the library? And disadvantages?
- Can we improve this program even further?

Unstructured data

- What is the meaning of this line:
  `it.split(",").get(1).toFloat()` ?
- How do we know that `get(1)` will give back something?
- What if the file has a bad format?
  - How many things can go wrong in this case?

Unstructured data

- What is the meaning of this line:
  `it.split(",").get(1).toFloat()` ?
- How do we know that `get(1)` will give back something?
- What if the file has a bad format?
  - How many things can go wrong in this case?

```
$ cat ~/Desktop/inst.csv
 X,  Y
 3,  3
 0
```

Example of bad input

`IndexOutOfBoundsException`

Unstructured data

- What is the meaning of this line:
  `it.split(",").get(1).toFloat()` ?
- How do we know that `get(1)` will give back something?
- What if the file has a bad format?
  - How many things can go wrong in this case?

```
$ cat ~/Desktop/inst.csv
 X,  Y
 3,  3
 0, frog
```

Another example of bad input

`NumberFormatException`

Unstructured data

- What is the meaning of this line:
  `it.split(",").get(1).toFloat()` ?
- How do we know that `get(1)` will give back something?
- What if the file has a bad format?
  - How many things can go wrong in this case?
- How can we improve this program to ease its understanding?

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSIDADE FEDERAL DE MINAS GERAIS
FEDERAL UNIVERSITY OF MINAS GERAIS, BRAZIL

OBJECTS AND CLASSES

The `Point` Class

PointReader2.kt

```
class Point(val x: Float, val y: Float) {
    override fun toString(): String {
        return "(${x}, ${y})"
    }
}

fun line2Point(line: String): Point {
    val list = line.split(",")
    val x = list.get(0).toFloat()
    val y = list.get(1).toFloat()
    return Point(x, y)
}
```

- Where is the constructor?
- How to convert a file into points?

Mapping lines into points

PointReader2.kt

```
fun main(args: Array<String>) {
  val file = File(args[0])
  val lines = file.readLines()
  val lDrop = lines.drop(1)
  val points = lDrop.map { line2Point(it) }
  points.forEach { println(it) }
}
```

```
$ kotlinc PointReader2.kt -include-runtime -d pointReader.jar
$ java -jar pointReader.jar ~/Desktop/inst.csv
(3.0, 3.0)
(0.0, 4.0)
(-1.0, -2.0)
(1.0, 3.0)
(2.0, 2.0)
```

Back into Centroids

- Can you use the new class `Point` to solve the problem of finding centroids?

Back into Centroids

- Can you use the new class `Point` to solve the problem of finding centroids?

```
fun main(args: Array<String>) {
  val file = File(args[0])
  val lines = file.readLines()
  val lDrop = lines.drop(1)
  val points = lDrop.map { line2Point(it) }
  val xAvg = points.map{it.x}.average()
  val yAvg = points.map{it.y}.average()
  print("Centroid = (${xAvg}, ${yAvg})\n")
}
```

Example: `RandomPairs.kt`

Implement a program that reads a txt file containing a list of students, and organizes the students into random teams. The number of students per team is a command line argument.

```
$ cat s.txt
ARTUR
BRENO
CAIO
IVO
JOÃO
JOSE
MATHEUS
PEDRO
THAIS
VINICIUS
```

Example of input

```
$ java -jar randomPairs.jar s.txt 3
> [JOSE, ARTUR, CAIO]
> [VINICIUS, JOÃO, MATHEUS]
> [PEDRO, BRENO, THAIS]
> [IVO]

$ java -jar randomPairs.jar s.txt 4
> [PEDRO, CAIO, IVO, ARTUR]
> [THAIS, MATHEUS, JOÃO, JOSE]
> [BRENO, VINICIUS]
```

Example of outupt

Example: `RandomPairs.kt`

RandomPairs.kt

```
import java.io.File
fun main(args: Array<String>) {
  val file = File(args[0])
  val teamSize = args[1].toInt()
  val lines = file.readLines()
  val shuffled = lines.shuffled()
  val chunked = shuffled.chunked(teamSize)
  chunked.forEach {
    println("> $it")
  }
}
```

- What would happen if we do not inform enough command line arguments?

Checking input arguments

- Modify the program, so that it reports a missing argument, and asks users to use the expected syntax.

```
$ java -jar randomPairs.jar s.txt
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:
1 at RandomPairsKt.main(RandomPairs.kt:5)
```

Missing argument

Checking the number of arguments in the command line

RandomPairs.kt

```
import java.io.File
fun main(args: Array<String>) {
  if (args.size < 2) {
    System.err.println("Syntax: cmd <file.txt> <teamSize>")
  } else {
    val file = File(args[0])
    val teamSize = args[1].toInt()
    val lines = file.readLines()
    val shuffled = lines.shuffled()
    val chunked = shuffled.chunked(teamSize)
    chunked.forEach {
      println("> $it")
    }
  }
}
```

```
$ java -jar randomPairs.jar s.txt
Syntax: cmd <file.txt> <teamSize>
```