EXERCÍCIOS SOBRE RECURSÃO

Michael Móra

Exercícios

1. Modele e implemente um método recursivo que calcule o fatorial de um número n passado como parâmetro.

2. Modele e implemente um método recursivo que calcule o somatório de um número n (passado como parâmetro) até 0.

3. Modele e implemente um método recursivo que calcule o n-ésimo número da sequência de fibonacci.

4. Modele e implemente um método recursivo que calcule o somatório dos números inteiros entre os números k e j, passados como parâmetro.

5. Modele e implemente um método recursivo que recebe um String em retorna `true` se este String for um palíndrome, `false` caso contrário.

```
boolean isPal(String s)
```

6. Modele e implemente um método recursivo que recebe um inteiro zero ou positivo e retorna um String com o número em binário.

```
String convBase2(int n)
```

7. Modele e implemente um método recursivo que calcule o somatório dos números contidos em um ArrayList de inteiros, passado como parâmetro.

8. Modele e implemente um método recursivo para encontrar o maior elemento de um ArrayList.

```
int findBiggest(ArrayList<Integer> ar)
```

Ex.1 - Fatorial

`int fatorial(int n)`	
Ex.: n = -1 n = 0 n = 1 n = 10	
Situações de Erro:	n < 0 → exception
Situações de Parada (base da recursão):	n == 0 → 1 n == 1 → `1`

Recursão:	`n >= 2`
	
	`n == 2` → `2 * fatorial(1)`
	
	`n == 10` → `10 * fatorial(9)`

Ex.2 - Somatório (até 0)

`int somatorio(int n)`	
Ex.: n = -10 n = -1 n = 0 n = 1 n = 10	
Situações de Erro:	N/A
Situações de Parada (base da recursão):	n == 0 → 0

Recursão:	`n > 0 -> n + somatorio(n-1)`
	
	`n == 1 → 1 + 0`
	`1 + somatorio(0)`
	`n == 10 → 10 + 9 + 8 + … + 1 + 0`
	`10 + somatorio(9)`
	
	`n < 0 -> n + somatorio(n+1)`
	
	`n == -1 → -1 + 0`
	`-1 + somatorio(0)`
	`n == -10 → -10 + -9 + -8 + … + -1 + 0`
	`-10 + somatorio(-9)`

Ex.3 - Fibonacci

`int fibonacci(int n)`	
Ex.: n = 1 n = 10	
Situações de Erro:	n <= 0 → exception!
Situações de Parada (base da recursão):	n == 1 → 1 n == 2 → 1

Recursão:	`n > 2 -> Fibonacci(n-1) + fibonacci(n-2)`
	
	`n == 3` → `fibonacci(2) + fibonacci(1)`
	`1 + 1` → 2
	
	`n == 5` → `fibonacci(4) + fibonacci(3)`
	`3 + 2` → 5
	
	Sequência de Fibonacci:
	
	`Posição: 1 2 3 4 5 6 7 8 9 10 …`
	`Sequência: 1 1 2 3 5 8 13 21 34 55 …`

Ex.4 - Somatório (entre k e j)

`int somatorio(int k, int j)`	
Ex.: k = 1 j = 5; k = 5 j = 1; k = -1 j = 5; k = 5 j = -1; k = 5 j = 5;	
Situações de Erro:	N/A
Situações de Parada (base da recursão):	k == j → k

Recursão:	`k < j -> k + somatorio(k+1, j)`
	
	`k == 1, j == 5 → 1 + [2+3+4+5]`
	`1 + somatorio(2, 5)`
	
	`k > j -> somatorio(j, k)`
	`k == 5, j == 1 → somatorio(1, 5)`

Ex.5 - Palíndrome

`boolean isPal(String s)`	
Ex.: s = “arara” s = “cassac” s = “mesas” s = “mesam”	
Situações de Erro:	s == null → exception

Situações de Parada (base da recursão):	s.length() == 0 → true
	s.length() == 1 → `true`
	
	`“m” “e” “s” “a” “s”`
	`“m” != “s” → false`
	`s.chatAt(0) != s.charAt(s.size()-1) → false`

Recursão:	`“a” “r” “a” “r” “a”`
	`“a” == “a” → isPal(“rar”)`
	
	`“c” “a” “s” “s” “a” “c”`
	`“c” == “c” → isPal(“assa”)`
	
	`s.chatAt(0) == s.charAt(s.size()-1) →`
	`isPal(s.substring(1, s.length()-1)`

Ex.6 – Conversão Base 2

`String convBase2(int n)`	
Ex.: n = 0 n = 1	n = 2 (10) n = 5 (101)
Situações de Erro:	n < 0 → exception!
Situações de Parada (base da recursão):	n == 0 → `"0"` n == 1 → `"1"`

Recursão:	`n == 5 (101)`
	`convBase2(10) + "1"`
	
	`n == 6 (110)`
	`convBase2(11) + "0"`
	
	`n > 0 &&`
	`n != 0 &&`
	`n != 1 → convBase2(n div 2) + (n % 2)`
	
	(*) + é a concatenação de strings

Ex.7 – Somatório de Elementos do ArrayList

`int somaArray(ArrayList<Integer> ar)`	
Ex.: ar = [7,-2,23,40] ar = [70,-20,23,10]	
Situações de Erro:	ar == null → exception ar.size() == 0 → exception
Situações de Parada (base da recursão):	Ar == [10] → 10 ar.size == 1 → ar.get(0)

Recursão:	`ar ==[7,-2,23,40]`
	`ar.size() > 1` → 7 + somaArray([-2, 23, 40])
	
	`Como extrair [-2, 23, 40] de [7, -2, 23, 40]?`
	`a) ar.remove(0)`
	`b) arrayListCopy(ar, 1, novoAr, 0, ar.size()-1)`
	
	`Há solução sem copier o ArrayList?`

Ex.8 – Maior Elemento do ArrayList

`int findBiggest(ArrayList<Integer> ar)`	
Ex.: ar = [7,-2,23,40] ar = [70,-20,23,10]	
Situações de Erro:	ar == null → exception ar.size() == 0 → exception
Situações de Parada (base da recursão):	[10] → 10 ar.size == 1 → ar.get(0)

Recursão:	`[7,-2,23,40]`
	`7 < findBiggest([-2,23,40])`
	`→ 40`
	
	`[70,-20,23,10]`
	`70 > findBiggest([-20,23,10])`
	`→ 70`
	
	`p = ar.remove(0);`
	`if (p > findBiggest(ar))`
	`→ p`
	`else`
	`→ findBiggest(ar)`

Exercícios Adicionais

9. Implemente um método recursivo para determinar se um string ocorre dentro de outro.

```
boolean findSubStr(String str, String match)
```

10. Faça um método recursivo que determina o número de dígitos de um inteiro.

```
int nroDigit(int n)
```

11. Implemente um métodos que recebe um String e retorna um ArrayList com todas as permutações deste String.

```
ArrayList<String> permutations(String s)
```

Ex.:

```
cão -> [cão, coã, ãoc, ãco, oãc, oãc]
```