0.2 Aula 21 - Indo além de vetores

Nessa aula o objetivo é ir além de vetores, primeiro entendendo que em Júlia o que se pode fazer com vetores é bem flexível, em seguida vamos ver que vetores podem ter várias dimensões. Veremos ao final alguns algoritmos com matrizes.

0.2.1 Indo além de vetores

Conforme já vimos antes, os vetores ocupam várias posições de memória, uma forma de se inicializar um vetor é através da função zeros. Que cria um vetor com valores nulos.

```
zeros(1)
zeros(100)
zeros(Int8, 3)
```

A novidade é que agora o zeros pode criar também vetores de mais de uma dimensão, ou matrizes.

```
zeros(3,3)
zeros(Int, 2, 2)
```

Duas outras funçÕes interessantes são a ones() com comportamento similar a zeros(), e a rand() que cria vetores ou matrizes com números aleatórios.

O comando fill() serve para fazer isso com um valor específico.

```
ones(3, 3)
rand(4, 4)
rand(1: 10, 5, 5)
fill(42, 3, 3, 3)
```

Também é possível criar matrizes diretamente.

```
m = [1 2 3; 4 5 6; 7 8 9]
```

O comando length() funciona para matrizes, mas devolve o seu tamanho total, mas temos também os comandos ndims() e size().

A linguagem Julia ainda oferece vários açucares sintáticos, vejamos o exemplo abaixo.

```
a = [1 2; 3 4]
b = [1 0; 0 1]
a * b
a .* b
```

Mas, lembrando que o objetivo da disciplina é ensinar algoritmos, vamos ver alguns.

Uma função que imprime uma matriz quadrada m.

```
function imprimeMatriz(m)
    a = size(m)
    soma = 0
    for i in 1:a[1]
       for j in 1:a[2]
          print(" m[", i, ", ", j, "] = ", m[i,j])
        end
        println()
    end
end
```

Faça uma função que recebe uma matriz quadrada e devolve a soma dos seus elementos.

```
function somaMatriz(m)
    a = size(m)
    soma = 0
    for i in 1:a[1]
       for j in 1:a[2]
          soma = soma + m[i, j]
        end
     end
     return soma
  end
```

Uma matriz quadrada é um quadrado mágico se a soma dos elementos de cada linha, de cada coluna e das diagonais é sempre igual. Faça uma função que dada uma matriz quadrada m, verifica se ela é um quadrado mágico.

```
function QuadradoMagico(m)
    if length(m) == 1
      return true
    end
    a = size(m)
    if length(a) != 2 || a[1] != a[2]
      return false
    end
    somaP = 0
    for i in 1:a[1]
      somaP = somaP + m[i, i]
    end
```

```
    soma = 0
    for i in 1:a[1]
       soma = soma + m[i, a[1] - i + 1]
    end
    if somaP != soma
       return false
    end
    for i in 1:a[1]
       somaL = 0
       somaC = 0
       for j in 1:a[2]
          somaL = somaL + m[i, j]
          somaC = somaC + m[j, i]
       end
       if somaL != somaP || somaC != somaP
          return false
       end
    end
    return true
end
```

Dadas duas matrizes m e n, faça uma função que devolva o produto delas (sem usar o * para matrizes).

```
function multiplica(a, b)
   dima = size(a)
   dimb = size(b)
   if dima[2] != dimb[1]
     return -1
   end
   c = zeros(dima[1], dimb[2])
   for i in 1:dima[1]
      for j in 1:dimb[2]
         for k in 1:dima[2]
            c[i, j] = c[i, j] + a[i, k] * b[k, j]
         end
      end
   end
   return c
 end
```

Uma matriz quadrada de tamanho n é um quadrado latino se em cada linha e coluna aparecem todos os valores de 1 a n. Faça uma função que dada uma matriz quadrada verifica se ela é um quadrado latino.