Programando em Java
(Matrizes, ou vectores bi-dimensionais)

Mestrado Integrado em Engenharia Informática FCT UNL
http://ctp.di.fct.unl.pt/miei/ip/

Corpo Docente 2020/2021
António Ravara, Artur Miguel Dias, Bernardo Toninho,
Ema Vieira, Inês Fernandes, Margarida Mamede,
Miguel Monteiro, Rui Nóbrega

Matrizes: vectores bi-dimensionais

Um vector em Java pode conter elementos de qualquer tipo. Em particular, os elementos de um vector podem ser também de tipo vector.

Chamamos **matriz** a um vector de vectores em que todos os vectores internos têm o mesmo comprimento. Trata-se do caso mais útil e mais usado dos vectores de vectores. Estes slides são dedicados ao caso especial das matrizes.

Declaração de matrizes de dado `tipo`:

```
type[][] matrix;
```

matriz que guarda valores de tipo *type*

Exemplos:

```
int[][] matInt;
// matriz de inteiros
BankAccount[][] matBA;
// matriz de contas bancárias
// (objectos da classe BankAccount)
```

Criação de vectores bi-dimensionais

Declaração de matriz de dado `tipo`:

```
type[][] matrix;
```

os elementos de `matrix` são elementos de tipo *type*

Para criar a matriz, é preciso definir quantas linhas e quantas colunas terá

Criação de matrizes:

```
matrix = new type[nRows][nCols]
```

assumindo que `nRows` e `nCols` são constantes ou variáveis inicializadas

Exemplos de declaração e criação:

```
int[][] matInt = new int[3][5];
// matriz de inteiros com 3 linhas e 5 colunas
private static final int NR = 3;
private static final int NC = 5;
boolean[][] matBol = new boolean[NR][NC];
// matriz de booleanos com 3 linhas e 5 colunas
```

Inicialização de vectores bi-dimensionais

Declaração de matriz de dado tipo: `type[][] matrix;`

Criação de matriz: `matrix = new type[nRows][nCols]`

assumindo que `nRows` e `nCols` são constantes ou variáveis inicializadas

Numa matriz como `matrix`, obtém-se:

1. o número de linhas escrevendo `matrix.length`
2. o número de colunas escrevendo `matrix[0].length` (assumindo que `matrix.length > 0`, ou seja que `matrix[0]` existe)

Inicialização de matriz (em geral, de vectores de vectores):

1. um elemento na linha `i` e coluna `j` com o valor `val` de tipo *type*

   ```
   matrix[i][j] = val;
   ```

2. um vector de tamanho `n` (número de colunas) na linha `i` com os valores `val_1` a `val_n` de tipo *type*

   ```
   matrix[i] = {val_1,...,val_n}
   ```

3. uma matriz de `n` por `m` de valores do tipo *type*

   ```
   matrix = {{val_11,...,val_1n},...,{val_m1,...,val_mn}}
   ```

Determinar o máximo duma matriz

Algoritmo (considerando matrizes com pelo menos uma linha e uma coluna):
- percorrer a matriz por linhas (1º a 1ª, depois a 2ª, etc, até à ultima)
- comparar o valor na posição a analisar (1ª da 1ª linha, 2ª da 1ª linha, etc, até ao fim da linha, passando depois para a linha de baixo, se existir) com o máximo encontrado até ao momento
- se a posição em análise tem um valor maior que o máximo encontrado até então, o valor nessa posição passa a ser o máximo

Implementação:

- usa-se uma variável auxiliar para guardar o máximo encontrado até então (inicializado ao 1º valor da 1ª linha)
- faz um ciclo para percorrer todas as linhas da matriz
- o corpo do ciclo referido é um ciclo para percorrer as colunas

Determinar o máximo duma matriz

Suponha-se que existe uma variável `matrix` com uma matriz de inteiros, de `rows` linhas por `cols` colunas (considera-se `rows` e `cols` **positivos**)

Inicialização: usa-se uma variável auxiliar (pode-se chamar `max`) para guardar o máximo encontrado, inicializada com o valor da matriz na posição (0,0)

```
int max = matrix[0][0];
```

Ciclos (`for`, pois sabe-se o número de linhas/colunas):

- o de fora percorre as linhas
- o de dentro percorre as colunas
- o corpo do ciclo de dentro (que é o corpo do de fora) compara os elementos da matriz com o máximo já encontrado

```
for(int i = 0; i < rows; i++)
  for(int j = 0; j < cols; j++)
    if(matrix[i][j] > max)
      max = matrix[i][j];
```

Determinar o máximo duma matriz

```
public class Matrix {
    private int rows, cols;
    private int[][] matrix;

    // Inicialização das variáveis de instância - matriz não vazia
    // @pre matrix.length > 0 && matrix[0].length > 0
    public Matrix(int[][] matrix){...}

    public int maxMatrix(){
        int max = matrix[0][0];
        for(int i = 0; i < rows; i++)
            for(int j = 0; j < cols; j++)
                if(matrix[i][j] > max) max = matrix[i][j];
        return max;
    }
}
```

Determinar o máximo duma matriz

E se se pretende-se pesquisar por colunas, i.e., percorrer as colunas da matriz, em vez das linhas?

Basta trocar os os limites dos ciclos e os índices da matriz na condição do `if`:

- o ciclo de fora percorre as colunas `for(int i = 0; i < cols; i++)`
- o de dentro percorre as linhas `for(int j = 0; j < rows; j++)`
- o corpo do ciclo de dentro (que é o corpo do de fora) compara os elementos da matriz com o máximo já encontrado

```
if(matrix[j][i] > max)
  max = matrix[j][i];
```

Pesquisa linear em matriz

Suponha-se que existe uma variável `matrix` com uma matriz de inteiros, de `rows` linhas por `cols` colunas; procura-se o valor `elem`

Inicialização: usa-se uma variável auxiliar booleana (pode-se chamar `found`) para assinalar se o valor foi encontrado, inicializada a `false` (antes de se iniciar a pesquisa, não se encontrou)

Ciclos (`while`, agora, pois não se irá necessariamente até ao fim da linha/coluna):

- o de fora percorre as linhas
- o de dentro percorre as colunas
- o corpo do ciclo de dentro (que faz parte do corpo do de fora) compara os elementos da matriz com o elemento que se procura

```
while(i < rows && !found) {
  j = 0;
  while(j < cols && !found)
    if(matrix[i][j] == elem)
      found = true;
    else j++;
  i++;
}
return found;
```

Pesquisa linear em matriz

Alternativa: alterar os índices quando se encontra, para terminar os ciclos;
se no fim o índice das linhas ultrapassou o limite, encontrou-se o valor `elem`

```
int i = 0;
while(i < rows) {
    int j = 0;
    while(j < cols)
        if(matrix[i][j] == elem) {
            i = rows;
            j = cols;
        }
        else j++;
    i++;
}
return i == rows+1;
```

Número de linhas crescentes na matriz

Suponha-se que existe uma variável `matrix` com uma matriz de inteiros, de `rows` linhas por `cols` colunas

Inicialização: usa-se uma variável auxiliar (pode-se chamar `count`) para contar as linhas crescentes, inicializada a `0` (antes de se iniciar a pesquisa, não se encontrou nenhuma)

Ciclos:

- o de fora percorre todas as linhas
- o de dentro percorre as colunas enquanto os valores forem crescentes
- o corpo do ciclo de fora vê também se o de dentro foi até ao fim; se sim, a linha é crescente

```
int j, count = 0;
for(int i = 0; i < rows; i++) {
  j = 0; while(j < cols-1 &&
   matrix[i][j] < matrix[i][j+1])
      j++;
  if (j == cols-1) count++;
}
```

Coluna com maior média

Problema: devolver o índice da coluna de uma matriz de inteiros que tem o maior valor médio

Inicialização: usa-se uma variável auxiliar (`max`) para guardar o máximo encontrado, inicializada ao menor valor possível (para não afectar o resultado), e outra (`res`) para guardar o índice da coluna que tem esse máximo, inicializada a `0`

pode-se usar a constante `Integer.MIN_VALUE` do pacote `java.lang` da classe `Integer`; o seu valor é `-2^31 = -2147483648`

```
int max = Integer.MIN_VALUE; int res = 0;
for(int i = 0; i < cols; i++) {  // soma linhas e ve e' a
    int sum = 0;                 // maior media
    for(j = 0; j < rows; j++) sum += matrix[j][i];
    if(max < sum/cols) {max = sum/cols; res = i;}
}
return res;
```