Laboratório de Programação II

Matrizes

Universidade Federal de Juiz de Fora
Departamento de Ciência da Computação

Aula de Hoje

- Representação de matrizes
  - Vetor de vetores
  - Linear
- Aplicações com matrizes

Representação de matrizes

- Vetor de vetores

```
class Matriz2D
{
 private:
   int nl, nc;
   float **mat; // <--- elementos da matriz
   // ...
};
```

- **Linear**
  - Nesta aula iremos trabalhar apenas com essa representação!

```
class MatrizLin
{
 private:
   int nl, nc;
   float *vet; // <--- elementos da matriz
   // ...
};
```

TAD Matriz

```
class MatrizLin
{
 public:
   MatrizLin(int mm, int nn);
   ~MatrizLin();

   void set(int i, int j, float val);
   float get(int i, int j);

 private:
   int nl, nc; // numero de linhas e colunas
   float *vet; // vetor de tamanho nl*nc

   int detInd(int i, int j);
};
```

TAD Matriz - Representação linear

- Construtor e destrutor

```
MatrizLin::MatrizLin(int mm, int nn)
{
  nl = mm;
  nc = nn;
  vet = new float[nl*nc];
}
```

```
MatrizLin::~MatrizLin()
{
  delete [] vet;
}
```

TAD Matriz - Representação linear

```
int MatrizLin::detInd(int i, int j)
{
  if(i >= 0 && i < nl && j >= 0 && j < nc)
    return i*nc + j;
  else
    return -1; // indice invalido
};
```

TAD Matriz - Representação linear

```
float MatrizLin::get(int i, int j)
{
  int k = detInd(i, j);
  if(k != -1)
    return vet[k];
  else {
    cout << "Indice invalido!" << endl;
    exit(1);
  }
}
```

```
void MatrizLin::set(int i, int j, float val)
{
  int k = detInd(i, j);
  if(k != -1)
    vet[k] = val;
  else
    cout << "Indice invalido!" << endl;
}
```

Exercícios

1. Implemente uma operação do TAD MatrizLin para **imprimir a matriz**. Você deverá imprimir uma linha da matriz em cada linha da tela, com os elementos separados por vírgula.
2. Implementar uma função no programa principal que **determina se uma matriz quadrada é simétrica**. A função deve retornar `true` ou `false`.
3. Implemente uma função para **encontrar o maior valor da matriz**.

Exercícios

4. Implemente uma operação que **cria e retorna a transposta da matriz**. A matriz transposta deve ser **alocada de forma dinâmica** e retornada ao final.
   Utilize o seguinte protótipo:

```
MatrizLin* MatrizLin::transposta();
```

   Observações:

   - Essa é uma operação do TAD MatrizLin.
   - Essa operação não deve alterar a matriz original, apenas criar uma nova matriz e copiar os elementos de forma correta para esta.

Exercícios

5. Implemente uma função que realiza o **produto de uma matriz por um vetor**. O vetor será representado por um array de valores do tipo de dado float.
   Essa função deve ser implementada no arquivo `main.cpp` e deve possuir o seguinte protótipo:

```
float* prodMatVetor(MatrizLin *m, float *v);
```

   onde `m` é a matriz a ser multiplicada pelo vetor `v`. O resultado é um vetor (float *) que deve ser **alocado de forma dinâmica dentro da função** e retornado ao final.

- Exemplo:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1x + 2y + 3z \\ 4x + 5y + 6z \\ 7x + 8y + 9z \end{bmatrix}$$