Arrays (exemplos)

Array de 1 dimensão

Subprograma de base

- Subprograma para devolver um número inteiro entre dois limites de variação
  - nome: **lerValorValido**
  - argumentos (dados de entrada): dois valores inteiros, **inf** e **sup**
  - tipo de dados do resultado que é devolvido: **int**
  - subprograma em C:

```
int lerValorValido (int  inf, int  sup)
{
  int  num;
  do{
     printf("Insira um inteiro entre %d e %d: ", inf, sup);
     scanf("%d", &num);
  }while(num < inf || num > sup);
  return  num;
}
```

Subprograma de base

- Subprograma para construir um array de reais de tamanho superior a 0
  - nome: **lerArrayReais**
  - argumentos (dados de entrada):
    - um array **X** com elementos reais
    - um número inteiro **dim** (≥ 1) correspondente ao tamanho do array **X**
  - tipo de dados do resultado que é devolvido: nada (**void**)
  - subprograma em C:

```
void lerArrayReais (float X[], int  dim)
{
  int  k;
  for (k = 0; k <= dim-1; k = k + 1) {
     printf("Inserir um valor real: ");
     scanf("%f", &X[k]);
  }
}
```

Subprograma de base

- Subprograma para mostrar no monitor todos os elementos de um array de reais de tamanho superior a 0
  - nome: **mostrarArrayReais**
  - argumentos (dados de entrada):
    - um array **X** com elementos reais
    - um número inteiro **dim** (≥ 1) correspondente ao tamanho do array **X**
  - tipo de dados do resultado que é devolvido: nada (**void**)
  - subprograma em C:

```
void mostrarArrayReais (float X[], int  dim)
{
  int  k;
  for (k = 0; k <= dim-1; k = k + 1)
     printf("%7.2f ", X[k]);     // usando 7 casas, das quais 2 são decimais
}
```

Exemplo 1

- Implementar o subprograma **posicaoMaior** para determinar e devolver o índice do elemento de um array **X** com **dim** valores reais com maior valor
- Construir um programa em C que, usando as funções construídas antes,
  - **ler** um array **V** com **N** elementos do tipo real, em que N ≥ 1,
  - **determinar** o **índice** de um elemento de **V** com maior valor,
  - **calcular** o **maior** valor entre os elementos de **V**, e
  - **mostrar** no monitor os resultados obtidos

Exemplo 1

- Subprograma para determinar o índice do elemento de um array **X** com **dim** valores reais com maior valor
  - nome: **posicaoMaior**
  - argumentos:
    - um array **X** com elementos reais
    - um número inteiro **dim** (≥ 1), que é o tamanho de **X**
  - tipo de dados do resultado que é devolvido:
    - um valor **inteiro** associado ao índice do elemento do array **X** com maior valor

Exemplo 1

- Subprograma para determinar o índice do elemento de um array **X** com **dim** valores reais com maior valor
  - subprograma em C:

```
int posicaoMaior (float X[], int dim)
{
  int  k, posMaior;
  posMaior = 0;    // assumir que o maior está na primeira posição do array
  for (k = 1; k <= dim-1; k = k + 1)
     if (X[k] > X[pos_maior])
        posMaior = k;
  return posMaior;
}
```

Exemplo 1

- Programa principal em C

```
#include <stdio.h>
int lerValorValido (int inf, int sup);
void lerArrayReais (float X[], int dim);
int posicaoMaior (float X[], int dim);
main()
{
  float  V[50], maior;
  int  N, posMaior;
  N = lerValorValido(1, 50);
  lerArrayReais (V, N);
  posMaior = posicaoMaior(V, N);
  printf("O maior valor está na posição : %d.\n", posMaior);
  maior = V[posMaior];
  printf("O maior valor é : %f.\n", maior);
}
```

Exemplo 2

- Implementar um subprograma para inverter um array **X** com **dim** valores reais (trocar o primeiro elemento com o último, o segundo com o penúltimo, ...)
- Construir um programa em C que, usando as funções implementadas antes:
  - **ler** um array **V** com **N** valores reais, em que N ≥ 1,
  - **inverter** o array **V**, e
  - **mostrar** no monitor o array **V** obtido (invertido)

Exemplo 2

- Subprograma para inverter um array **X** com **dim** valores reais (trocar o primeiro elemento com o último, o segundo com o penúltimo, ...)
  - nome: **inverterArray**
  - argumentos:
    - um array **X** preenchido com elementos reais
    - um número inteiro **dim** (≥ 1), que é o tamanho de **X**
  - tipo de dados do resultado que é devolvido: nada (**void**)

Exemplo 2

- Subprograma para inverter um array **X** com **dim** valores reais (trocar o primeiro elemento com o último, o segundo com o penúltimo, ...)
  - subprograma em C:

```
void inverterArray (float X[], int dim)
{
  int  k;
  float  aux;
  for (k = 0; k < dim/2; k = k + 1)
  {
     aux = X[k];
     X[k] = X[dim-1-k];
     X[dim-1-k] = aux;
  }
}
```

Exemplo 2

- Programa principal em C

```
#include <stdio.h>
int lerValorValido (int inf, int sup);
void lerVetor (float X[], int dim);
void mostrarArrayReais (float X[], int dim);
void inverterArray (float X[], int dim);
main()
{
  float  V[50];
  int  N;
  N = lerValorValido(1, 50);
  lerArrayReais(V, N);
  inverterArray(V, N);
  printf("Array invertido:\n");
  mostrarArrayReais(V, N);
}
```

Array de 2 dimensões

Subprograma de base

- Subprograma para construir/preencher um array com **LIN** linhas e **COL** colunas de valores inteiros (LIN e COL ≥ 1)
  - nome: **lerArray2D**
  - argumentos (dados de entrada):
    - um array **X** com elementos reais
    - dois números inteiros, **LIN** e **COL** (≥ 1) correspondentes à dimensão do array **X**
  - tipo de dados do resultado que é devolvido: nada (**void**)

Subprograma de base

- Subprograma para construir/preencher um array com **LIN** linhas e **COL** colunas de valores inteiros (LIN e COL ≥ 1)

```
void lerArray2D (int X[][maxCol], int LIN, int COL)
{
  int  i, j;
  for (i = 0; i <= LIN-1; i = i + 1)
  {
     printf("Inserir a linha %d da matriz.\n", i);
     for (j = 0; j <= COL-1; j = j + 1)
     {
        printf("Inserir um valor inteiro: ");
        scanf("%d", &X[i][j]);
     }
  }
}
```

Subprograma de base

- Subprograma para mostrar no monitor um array com **LIN** linhas e **COL** colunas de valores inteiros (LIN e COL ≥ 1)
  - nome: **mostrarArray2D**
  - argumentos (dados de entrada):
    - um array **X** com elementos reais
    - dois números inteiros, **LIN** e **COL** (≥ 1) correspondentes à dimensão do array **X**
  - tipo de dados do resultado que é devolvido: nada (**void**)

Subprograma de base

- Subprograma para mostrar no monitor um array com LIN linhas e COL colunas de valores inteiros (LIN e COL ≥ 1)

```
void mostrarArray2D (int X[][maxCol], int LIN, int COL)
{
  int  i, j;
  for (i = 0; i <= LIN-1; i = i + 1)
  {
     printf("Linha %d da matriz: ", i);
     for (j = 0; j <= COL-1; j = j + 1)
        printf("%d ", X[i][j]);
     printf("\n");
  }
}
```

Exercicio 1

- Implementar o subprograma **numNulosMatriz** que calcula o número de elementos nulos de uma matriz (array de 2 dimensões) **X** de inteiros com **LIN** linhas e **COL** colunas
- Implementar o subprograma **matrizTransposta** que determina a matriz transposta de uma matriz **X** de ordem **N** (N linhas e N colunas)
- Construir um programa em C que, usando as funções anteriores:
  - **ler** uma matriz **MAT** com **N** linhas e **N** colunas ($1 \leq N \leq maxCol$),
  - **determinar** a quantidade de nulos de **MAT**,
  - **escrever** no monitor a quantidade de elementos nulos de **MAT**,
  - **determinar** z matriz transposta de **MAT**, e
  - **mostrar** a matriz transposta de **MAT**

Exercício 1

- Subprograma para calcular o número de elementos nulos de uma matriz (array de 2 dimensões) **X** de inteiros com **LIN** linhas e **COL** colunas
  - nome: **numNulosMatriz**
  - argumentos (dados de entrada):
    - um array **X** com elementos inteiros
    - dois números inteiros, **LIN** e **COL** (≥ 1) correspondentes à dimensão do array **X**
  - tipo de dados do resultado que é devolvido:
    - um valor **inteiro** correspondente à quantidade de elementos nulos da matriz X

Exercício 1

- Subprograma para calcular o número de elementos nulos de uma matriz (array de 2 dimensões) **X** de inteiros com **LIN** linhas e **COL** colunas
  - subprograma em C

```
int numNulosMatriz (int X[][maxCol], int LIN, int COL)
{
  int  i, j, nulos;
  nulos = 0;
  for (i = 0; i <= LIN-1; i = i + 1)
     for (j = 0; j <= COL-1; j = j + 1)
        if (X[i][j] == 0)
           nulos = nulos + 1;
  return  nulos;
}
```

Exercício 1

- Subprograma para calcular a matriz transposta de uma matriz **X** de inteiros com **M** linhas e **N** colunas
  - nome: **matrizTransposta**
  - argumentos (dados de entrada):
    - um array **X** com elementos inteiros preenchida
    - um array **XT** de inteiros vazia (receberá a matriz transposta de X)
    - dois números inteiros, **M** e **N**, correspondentes à dimensão dos arrays **X** e **XT**
  - tipo de dados do resultado que é devolvido: nada (**void**)

Exercício 1

- Subprograma para calcular a matriz transposta de uma matriz **X** de inteiros com **LIN** linhas e **COL** colunas
  - subprograma em C:

```
void  matrizTransposta (int X[][maxCol], int XT[][maxCol], int M, int N)
{
  int  i, j;
  for (i = 0; i <= M-1; i = i + 1)
     for (j = 0; j <= N-1; j = j + 1)
        XT[j][i] = X[i][j];
}
```

Exercício 1

- Programa principal em C

```
#include <stdio.h>
#define maxLin 10
#define maxCol 10
void lerArray2D (int X[][maxCol], int LIN, int COL);
void mostrarArray2D (int X[][maxCol], int LIN, int COL);
int lerValorValido (int inf, int sup);
int numNulosMatriz (int X[][maxCol], int LIN, int COL);
void matrizTransposta (int X[][], int XT[][], int M, int N);
...
```

Exercício 1

- Programa principal em C

```
...
main()
{
  int  X[maxLin][maxCol], XT[maxCol][maxLin], M, N, nulos;
  M = lerValorValido(1, maxLin);
  N = lerValorValido(1, maxCol);
  lerArray2D(X, M, N);
  nulos = numNulosMatriz(X, M, N);
  printf("A quantidade de nulos da matriz é: %d.\n", nulos);
  matrizTransposta(X, XT, M, N);
  printf("Matriz transposta\n");
  mostrarArray2D(XT, N, M);
}
```

Exercício 2

- Implementar o subprograma **matrizProduto** que determine a matriz produto de duas matrizes de inteiros, **X** (**L1xC1**) e **Y** (**L2xC2**)
- Construir um programa em C que, usando as funções anteriores:
  - **leia** duas matrizes de inteiros, **A** (**M1xN1**) e **B** (**M2xN2**) (com N1 = M2),
  - **determine** a matriz produto entre **A** e **B**, e
  - **escreva** a matriz produto determinada

Exercício 2

- Subprograma para calcular a matriz produto entre duas matrizes **X** (M1 linhas e N1 colunas) e **Y** (M2 linhas e N2 colunas) (com M2 = N1)
  - nome: **matrizProduto**
  - argumentos (dados de entrada):
    - uma matriz **X** com elementos inteiros preenchida
    - uma matriz **X** com elementos inteiros preenchida
    - uma matriz **XY** de inteiros vazia (receberá a matriz produto ente **X** e **Y**)
    - dois números inteiros, **M1** e **N1**, correspondentes à dimensão da matriz **X**
    - dois números inteiros, **M2** e **N2**, correspondentes à dimensão da matriz **Y**
  - tipo de dados do resultado que é devolvido: nada (**void**)

Exercício 2

- Subprograma para calcular a matriz produto entre duas matrizes **X** (M1 linhas e N1 colunas) e **Y** (M2 linhas e N2 colunas) (com M2 = N1)
  - subprograma em C:

```
void matrizProduto (int X[][maxCol], int Y[][maxCol], int XY[][maxCol],
int M1, int N1, int M2, int N2)
{
  int   i, j, k, S;
  for (i = 0; i <= M1-1; i = i + 1)
     for (j = 0; j <= N2-1; j = j + 1)
     {
        S = 0;
        for (k = 0; k <= N1-1; k = k + 1)
           S = S + X[i][k] * Y[k][j];
        XY[i][j] = S;
     }
}
```

Exercício 2

- Programa principal em C:

```
#include <stdio.h>
#define  maxLin  20
#define  maxCol  20
void lerArray2D (int X[][maxCol], int LIN, int COL);
void mostrarArray2D (int X[][maxCol], int LIN, int COL);
int lerValorValido (int inf, int sup);
void matrizProduto (int X[][maxCol], int Y[][maxCol], int XY[][maxCol],
int M1, int N1, int M2, int N2);
...
```

Exercício 2

- Programa principal em C (cont.):

```
...
main()
{
  int  A[maxLin][maxCol], B[maxLin][maxCol], P[maxLin][maxCol];
  int  M1, N1, M2, N2;
  M1 = lerValorValido(1, maxLin);
  N1 = lerValorValido(1, maxCol);
  lerArray2D(A, M1, N1);
  M2 = N1;
  N2 = lerValorValido(1, maxCol);
  lerArray2D(B, M2, N2);
  matrizProduto(A, B, P, M1, N1, M2, N2);
  printf("Matriz produto obtida:\n");
  mostrarArray2D(P, M1, N2);
}
```