BCC 201 - Introdução à Programação

Estruturas Homogêneas II (Matrizes)

Guillermo Cámara-Chávez
UFOP

Exercícios I

Escreva um programa que calcule a soma de todos os valores positivos e a quantidade de números negativos

Exercícios II

```
#define n 5
void Insere(int M[n][n], int, int);
int Soma(int M[n][n], int, int);
int NumNeg(int M[n][n], int, int);
int main()
{
    int M[n][n], somaPos, negativos;
    Insere(M, n, n);
    somaPos = Soma(M, n, n);
    negativos = NumNeg(M, n, n);
    cout << "Soma de positivos:" << somaPos;
    cout << "Numero de negativos: " << negativos;
    return 0;
}
```

Exercícios III

```
int Soma(int M[n][n], int lin, int col)
{
    int i, j, soma = 0;
    for (i = 0; i < lin; i++)
        for (j = 0; j < col; j++)
            if (M[i][j] > 0)
                soma += M[i][j];
    return soma;
}
```

Exercícios IV

```
int NumNeg(int M[n][n], int lin, int col)
{
    int i, j, cont = 0;
    for (i = 0; i < lin; i++)
        for (j = 0; j < col; j++)
            if (M[i][j] < 0)
                cont++;
    return cont;
}
```

Exercícios V

Leia uma matriz 20×20. Leia também um valor X. O programa deverá fazer uma busca desse valor na matriz e, ao final escrever a localização (linha e coluna) ou uma mensagem de "não encontrado".

Exercícios VI

```
#define n 20
void Insere(int M[n][n], int, int);
bool Procura(int M[n][n], int, int, int);

int main()
{
    int M[n][n], X;
    Insere(M, n, n);
    cout << "Inserir valor: \n";
    cin >> X;
    if (Procura(M, n, n, X) == 1)
        cout << "Valor encontrado";
    else
        cout << "Valor nao encontrado";
    return 0;
}
```

Exercícios VII

```
bool Procura(int M[n][n], int lin, int col, int valor)
{
    int i, j;
    for (i = 0; i < lin; i++)
        for (j = 0; j < col; j++)
            if (M[i][j] == valor)
            {
                cout << i <<", " << j << endl;
                return true;
            }
    return false;
}
```

Exercícios VIII

Leia duas matrizes 20×20 e escreva os valores da primeira que ocorrem em qualquer posição da segunda.

Exercícios IX

```
#define n 20
void Insere(int M[n][n], int, int);
bool Procura(int M[n][n], int, int, int);
void MostraProcurados(int A[n][n], int, int,
                      int B[n][n], int, int);

int main()
{
    int A[n][n], B[n][n];
    Insere(A, n, n);
    Insere(B, n, n);
    MostraProcurados(A,n,n,B,n,n);
    return 0;
}
```

Exercícios X

```
void MostraProcurados(int A[n][n], int linA, int colA,
                      int B[n][n], int linB, int colB)
{
    int i, j;
    for (i = 0; i < linA; i++)
        for (j = 0; j < colA; j++)
            if (Procura(B, linB, colB, A[i][j]) == true)
                cout << A[i][j] << endl;

}
```

Exercícios XI

Dada uma matriz de $m \times n$, imprimir o número de linhas e o número de colunas nulas da matriz. Uma coluna/linha é nula, se todos os elementos são zeros. Criar os seguintes procedimentos:

- Inicializar a matriz com zeros
- Mostrar o conteúdo da matriz
- Inserir dados na matriz
- Indicar o número de linhas e colunas nulas

Exercícios XII

```c
#define n 5
#define m 6
void Insere(int M[m][n], int, int);
void IniMatriz(int M[m][n], int, int, int);
void Mostra(int M[m][n], int, int);
void NumLinhasNulas(int M[m][n], int, int);
void NumColunasNulas(int M[m][n], int, int);
int main()
{
    int A[m][n];
    Insere(A, m, n);
    IniMatriz(A, m, n, 0);
    Mostra(A, m, n);
    NumLinhasNulas(A, m, n);
    NumColunasNulas(A, m, n);
    return 0;
}
```

Exercícios XIII

```
void IniMatriz(int M[m][n], int lin, int col, int valor)
{
    int i, j;
    for (i = 0; i < lin; i++)
        for (j = 0; j < col; j++)
            M[i][j] = valor;
}
```

Exercícios XIV

```
void NumLinhasNulas(int M[m][n], int lin, int col)
{
    int i, j, somaLinha;
    for (i = 0; i < lin; i++)
    {
        somaLinha = 0;
        for (j = 0; j < col; j++)
            somaLinha += M[i][j];
        if (somaLinha == 0)
            cout << "Linha " << i << "nula \n";
    }
}
```

Exercícios XV

```
void NumColunasNulas(int M[m][n], int lin, int col)
{
    int i, j, somaColuna;
    for (j = 0; j < col; j++)
    {
        somaColuna = 0;
        for (i = 0; i < lin; i++)
            somaColuna += M[i][j];
        if (somaColuna == 0)
            cout << "Coluna "<< j << " nula";
    }
}
```

Exercícios XVI

```
void Mostra(int M[m][n], int lin, int col)
{
    int i, j;
    for (i = 0; i < lin; i++)
    {
        for (j = 0; j < col; j++)
            cout << M[i][j] << " ";
        cout << endl;
    }
}
```

Exercícios XVII

Simular o funcionamento de uma matriz através de um vetor. Implementar os procedimentos que permitem inserir valores, mostrar o conteúdo da matriz

Exercícios XVIII

```
void InserirMat(int* mat, int lin, int col);
void MostrarMat(int* mat, int lin, int col);
int main()
{
    int M[12]; // a matriz será de 3x4
    InserirMat(M, 3, 4);
    MostraMat(M, 3, 4);
    return 0;
}
```

Exercícios XIX

```
void InserirMat(int* mat, int lin, int col){
    int i, j;
    for (i = 0; i < lin; i++)
        for (j = 0; j < col; j++)
        {
            cin >> mat[i*col+j];
        }
}
void MostrarMat(int* mat, int lin, int col){
    int i, j;
    for (i = 0; i < lin; i++)
    {
        for (j = 0; j < col; j++)
            cout << mat[i*col+j] << " ";
        cout << endl;
    }
}
```

Exercícios propostos I

1. Uma matriz simétrica é uma matriz quadrada onde seus elementos, considerando uma reflexão segundo a diagonal principal, são iguais, isto é, $m = n$ (quadrada) e $a_{ij} = a_{ji}$. Elabore uma função para verificar se uma matriz quadrada de ordem n e elementos inteiros é simétrica.
2. Uma matriz é triagular superior quando sua banda inferior é nula (todos os elementos da banda inferior são iguais a zero). Elabore uma função para verificar se uma matriz quadrada de ordem n e elementos inteiros é triangular superior.

FIM