MAC 2166 – Introdução à Computação

POLI - PRIMEIRO SEMESTRE DE 2007

Material Didático

Prof. Ronaldo Fumio Hashimoto

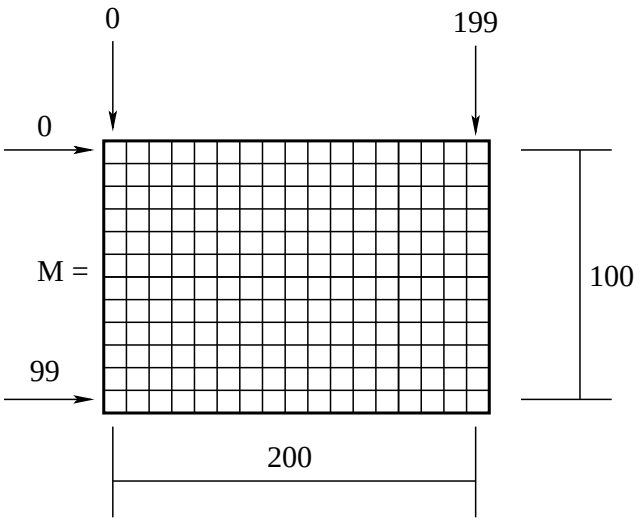
MATRIZES

Objetivo

O objetivo desta aula é introduzir o tipo **matriz**.

Matrizes

Matrizes são estruturas indexadas (em forma matricial) utilizadas para armazenar dados de um mesmo tipo: **int**, **char**, **float** ou **double**. O exemplo a seguir é de uma matriz de inteiros:



Declaração de Matrizes

A declaração de uma matriz é feita da seguinte forma:

<tipo_da_matriz> <nome_da_matriz> [<numero_de_linhas>][<numero_de_colunas>];

Exemplos:

- **int** M[100][200];

```
100 é o número de linhas!
200 é o número de colunas!
```

  A declaração acima aloca uma matriz com 100 linhas e 200 colunas na memória. Cada casa da matriz guarda um **int**.

- **float** x[20][30];

```
20 é o número de linhas!
30 é o número de colunas!
```

  A declaração acima aloca uma matriz com 20 linhas e 30 colunas na memória. Cada casa da matriz guarda um **float**.

Observação Importante:

1. Na **declaração de matriz**, o que está entre colchetes deve ser um **número constante**.
2. Assim, não é possível fazer algo deste tipo:

```
int n = 20, m = 30;
float x[n][m]; /* não é permitido declarar colocando variáveis entre colchetes */
```

   ou

```
int n, m;

printf ("Entre com n>0 e m>0: ");
scanf ("%d %d", &n, &m);

float x[n][m];
```

   O correto seria:

```
int n, m;
float x[20][30]; /* o correto é declarar sempre tamanhos fixos */
```

Uso de Matrizes

- São usados índices para acessar uma linha e uma coluna de uma matriz.
- Os índices são números naturais.
- O índice da **primeira** linha é sempre **zero**.
- O índice da **primeira** coluna é sempre **zero**.

Exemplo de Uso de Matrizes

- Exemplo:

```
1       # include <stdio.h>
2
3       int main () {
4         int A[10][80], i, j;
5
6         A[1][2] = 4; /* casa da linha 1 e coluna 2 recebe o inteiro 4 */
7         i = 2; j = 3;
8         A[i][j] = 5; /* casa de índice 2 do vetor v recebe o inteiro 3 */
9         A[A[i-1][j-1]][A[i][j]] = 10; /* vc saberia dizer qual casa da matriz A
10                                       * recebe o inteiro 10?
11                                       */
12
13        return 0;
14      }
```

Na Linha 4, a matriz `A` com 10 linhas e 80 colunas é declarada:

	0	1	2	3	4	5	...	78	79
0	?	?	?	?	?	?	...	?	?
1	?	?	?	?	?	?	...	?	?
2	?	?	?	?	?	?	...	?	?
3	?	?	?	?	?	?	...	?	?
⋮	⋮	⋮	⋮	⋮	⋮	⋮	...	⋮	⋮
9	?	?	?	?	?	?	...	?	?

Na Linha 6, casa de linha 1 e coluna 2 da matriz `A` recebe o inteiro 4:

	0	1	2	3	4	5	...	78	79
0	?	?	?	?	?	?	...	?	?
1	?	?	4	?	?	?	...	?	?
2	?	?	?	?	?	?	...	?	?
3	?	?	?	?	?	?	...	?	?
⋮	⋮	⋮	⋮	⋮	⋮	⋮	...	⋮	⋮
9	?	?	?	?	?	?	...	?	?

Na Linha 8, casa de linha 2 e coluna 3 da matriz `A` recebe o inteiro 5:

	0	1	2	3	4	5	...	78	79
0	?	?	?	?	?	?	...	?	?
1	?	?	4	?	?	?	...	?	?
2	?	?	?	5	?	?	...	?	?
3	?	?	?	?	?	?	...	?	?
⋮	⋮	⋮	⋮	⋮	⋮	⋮	...	⋮	⋮
9	?	?	?	?	?	?	...	?	?

Na Linha 9, como `i=2`, `j=3`, temos que `A[i-1][j-1]=A[1][2]=3` e `A[i][j]=A[2][3]=5`. Assim, temos que `A[A[i-1][j-1]][A[i][j]]=A[3][5]=10`. Dessa forma, no comando da Linha 9, a linha 3 e coluna 5 da matriz `A` recebe o inteiro 10:

	0	1	2	3	4	5	...	78	79
0	?	?	?	?	?	?	...	?	?
1	?	?	4	?	?	?	...	?	?
2	?	?	?	5	?	?	...	?	?
3	?	?	?	?	?	10	...	?	?
⋮	⋮	⋮	⋮	⋮	⋮	⋮	...	⋮	⋮
9	?	?	?	?	?	?	...	?	?

Percorrimento de Matrizes

Percorrer uma matriz significa varrer a matriz de casa em casa a partir da linha 0 (zero) e da coluna 0 (zero). No percorrimento de uma matriz, é necessário saber o número de linhas e colunas que deve-se fazer este percorrimento. Este número normalmente é guardado em duas variáveis inteiras (no nosso exemplo, as variáveis `n` e `m`).

Muitos problemas em MAC2166 que envolvem matrizes têm como soluções o uso de um padrão para percorrimento de matrizes.

Para os exemplos desta seção, vamos considerar a seguinte declaração de matriz:

```
int A[20][30];
```

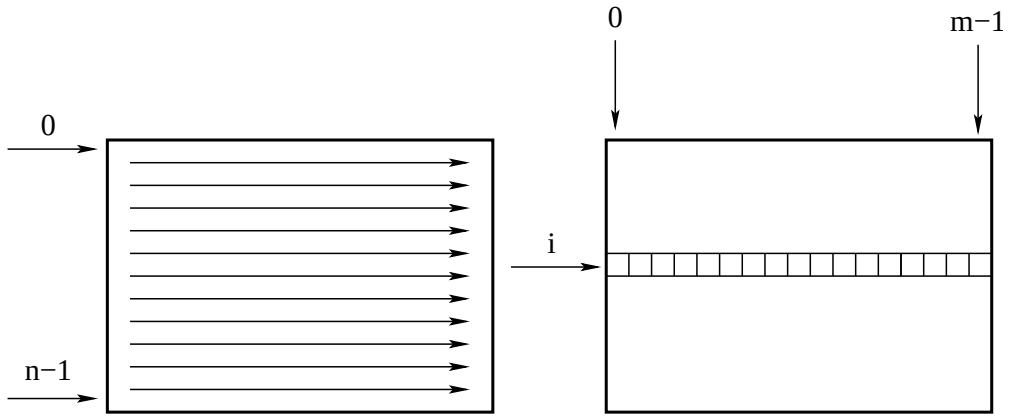
e as variáveis inteiras

```
int i, j, n, m, cont;
```

onde `n` e `m` são o número de linhas e colunas que devem ser consideradas na matriz `A`. É claro que neste caso, `n` tem que ser menor que 20 e `m` menor que 30.

Percorrimento por Linhas

Percorrimento de uma Linha:



Um padrão para percorrer uma linha `i` da matriz `A` é usar um comando de repetição (no caso, vamos usar o comando **for**) com uma variável inteira `j` para o índice das colunas da matriz `A`:

```
/* para uma linha fixa i */
for (j=0; j<m; j++) {
  /* comandos usando a matriz A[i][j] */
}
```

Exemplo:

```
cont = 0;
/* para uma linha fixa i */
for (j=0; j<m; j++) {
  A[i][j] = cont;
  cont++;
}
```

Percorrimento Completo da Matriz:

Um padrão para percorrer completamente a matriz `A` (isto é, as `n` linhas e as `m` colunas) por linhas é usar dois comandos de repetição (no caso, vamos usar o comando **for**) com duas variáveis inteiras `i` e `j`, um para percorrer as linhas e a outra para percorrer as colunas da matriz `A`:

```
for (i=0; i<n; i++) {
  for (j=0; j<m; j++) {
    /* comandos usando a matriz A[i][j] */
  }
}
```

O exemplo abaixo

```
for (i=0; i<n; i++) {
  for (j=0; j<m; j++) {
    A[i][j] = 0;
  }
}
```

inicializa as `n` linhas e as `m` colunas da matriz `A` com zero.

Observação sobre Percorrimento

Na declaração de uma matriz é definido um número fixo de linhas e colunas, uma vez que sempre deve-se colocar uma constante na definição do número de linhas e colunas da matriz. Por exemplo:

```
int A[20][30];
```

Mas, como podemos ver nos exemplos de percorrimento para ler e imprimir uma matriz, um usuário não necessariamente irá usar todas as linhas e colunas disponíveis da matriz. Note que no padrão de percorrimento por linhas deve sempre existir duas variáveis indicando quantas linhas e colunas da matriz estão sendo verdadeiramente usadas (variável `n` e `m` do padrão).

Assim, normalmente, em problemas de MAC2166 que envolvem matrizes deve-se sempre ter duas variáveis inteiras associadas à matriz que diz quantas linhas e colunas da matriz estão sendo usadas (por exemplo, variável inteira `n` e `m` associadas à matriz `A` nos exemplos de leitura e impressão de matrizes).

Leitura de uma Matriz

Para leitura de uma matriz, devemos ler elemento a elemento usando o padrão de percorrimento por linhas.

```c
#include <stdio.h>

int main () {
  float A[100][200];
  int i, j, n, m;

  printf ("Entre com 0<n<100: ");
  scanf ("%d" &n);

  printf ("Entre com 0<m<200: ");
  scanf ("%d" &m);

  /* percorrer a matriz A elemento a elemento
   * colocando o valor lido pelo teclado */
  for (i=0; i<n; i++) {
    for (j=0; j<m; j++) {
      printf ("Entre com A[%d][%d] = ", i, j);
      scanf ("%f", &A[i][j]);
    }
  }

  return 0;
}
```

Observe com cuidado a linha do programa utilizada para ler o elemento da linha `i` e coluna `j` da matriz `A`:

```c
scanf ("%f", &A[i][j]);
```

A linha `i` e a coluna `j` da matriz `A`, ou seja, `A[i][j]`, é utilizada da mesma forma que utizamos qualquer variável até o momento, ou seja, precedida pelo caractere '&'.

Impressão de uma Matriz

Para impressão de uma matriz, devemos imprimir elemento a elemento usando o padrão de percorrimento por linhas.

```c
# include <stdio.h>

int main () {
  float A[100][200];
  int i, j, n, m;

  printf ("Entre com 0<n<100: ");
  scanf ("%d" &n);

  printf ("Entre com 0<m<200: ");
  scanf ("%d" &m);

  /* percorrer a matriz A elemento a elemento
   * imprimindo o valor de cada casa */
  for (i=0; i<n; i++) {
    for (j=0; j<m; j++) {
      printf ("%f ", A[i][j]);
    }
    printf ("\n");
  }

  return 0;
}
```

Exercícios Comentados

Exercício 1

Faça um programa que leia um inteiro $n < 100$ e os elementos de uma matriz real $A_{n \times n}$ e verifica se a matriz A tem uma linha, coluna ou diagonal composta apenas por zeros.

Percorrimento de uma Linha de uma Matriz:

Para verificar se uma matriz `A` tem uma linha com todos elementos nulos, devemos percorrer uma linha `i` da matriz `A`.

Ora, nós já conhecemos o padrão para percorrer uma linha `i` da matriz `A`:

```c
/* para uma linha fixa i */
for (j=0; j<n; j++) {
  /* comandos usando a matriz A[i][j] */
}
```

Para contar quantos elementos nulos tem uma linha, podemos usar o padrão de percorrimento de uma linha da seguinte maneira:

```c
cont = 0;
/* para uma linha fixa i */
for (j=0; j<n; j++) {
  if (A[i][j] == 0)
    cont++;
}
```

Neste exemplo, o padrão conta quantos elementos nulos tem a linha `i`. Se quisermos saber se a linha `i` tem todos os elementos nulos, basta comparar se `cont` é igual a `n`. Assim:

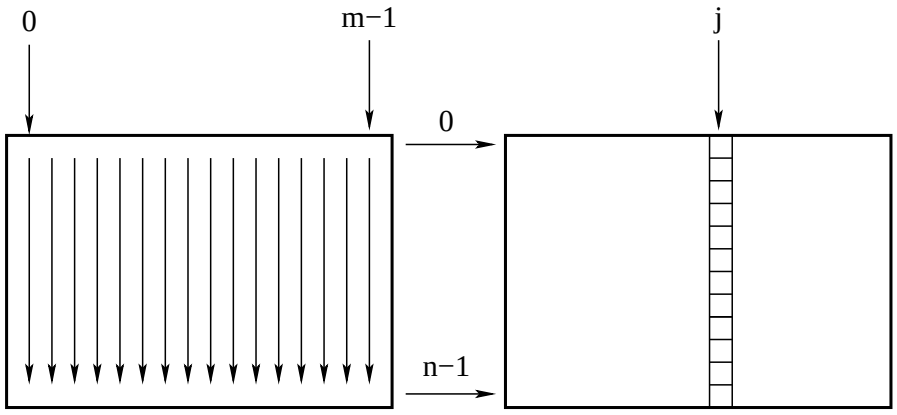
```
cont = 0;
/* para uma coluna fixa j */
for (i=0; i<n; i++) {
  if (A[i][j] == 0)
    cont++;
}
if (cont == n)
  printf ("A linha %d tem todos elementos nulos\n", i);
```

Assim, para verificar se uma matriz `A` tem uma linha com todos elementos nulos, devemos verificar cada linha `i` da matriz:

```
linha_nula = 0;
for (i=0; i<n; i++) {
  cont = 0;
  /* para uma linha i */
  for (j=0; j<n; j++) {
    if (A[i][j] == 0)
      cont++;
  }
  if (cont == n)
    linha_nula = 1;
}
if (linha_nula == 1)
  printf ("Matriz tem uma linha com todos elementos nulos\n",);
```

Percorrimento de uma Coluna de uma Matriz:

Para verificar se uma matriz `A` tem uma coluna com todos elementos nulos, devemos saber como percorrer uma coluna `j` da matriz `A`.



Um padrão para percorrer uma coluna `j` de uma matriz `A` é usar um comando de repetição (no caso, vamos usar o comando **for**) com uma variável inteira `i` para o índice das linhas da matriz `A`:

```
/* para uma coluna fixa j */
for (i=0; i<n; i++) {
  /* comandos usando a matriz A[i][j] */
}
```

Exemplos:

```
cont = 0;
/* para uma coluna fixa j */
for (i=0; i<n; i++) {
  A[i][j] = cont;
  cont++;
}

cont = 0;
/* para uma coluna fixa j */
for (i=0; i<n; i++) {
  if (A[i][j] == 0)
    cont++;
}
```

No último exemplo, o padrão conta quantos elementos nulos tem a coluna `j`. Se quisermos saber se a coluna `j` de uma matriz $A_{n\times n}$ tem todos os elementos nulos, basta comparar se `cont` é igual a `n`. Assim:

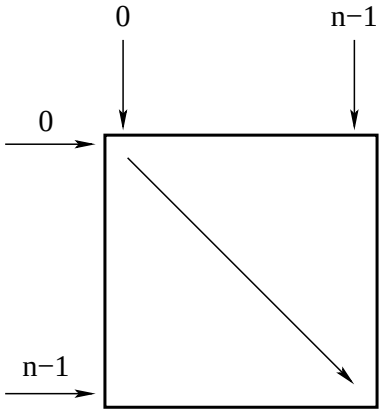
```
cont = 0;
/* para uma coluna fixa j */
for (i=0; i<n; i++) {
  if (A[i][j] == 0)
    cont++;
}
if (cont == n)
  printf ("A coluna %d tem todos elementos nulos\n", j);
```

Assim, para verificar se uma matriz quadrada `A` tem uma coluna com todos elementos nulos, devemos verificar cada coluna `j` da matriz:

```
coluna_nula = 0;
for (j=0; j<n; j++) {
  cont = 0;
  /* para uma coluna j */
  for (i=0; i<n; i++) {
    if (A[i][j] == 0)
      cont++;
  }
  if (cont == n)
    coluna_nula = 1;
}
if (coluna_nula == 1)
  printf ("Matriz tem uma coluna com todos elementos nulos\n",);
```

Percorrimento da Diagonal Principal de uma Matriz:

Para verificar se uma matriz quadrada $A_{n\times n}$ tem a diagonal principal com todos elementos nulos, devemos saber como percorrer esta diagonal da matriz `A`.



Como na diagonal principal temos que a linha é igual a coluna, um padrão para percorrer a diagonal principal de `A` é usar um comando de repetição (no caso, vamos usar o comando **for**) com uma variável inteira `i` para o índice das linhas e colunas da matriz `A`:

```
for (i=0; i<n; i++) {
  /* comandos usando a matriz A[i][i] */
}
```

Exemplos:

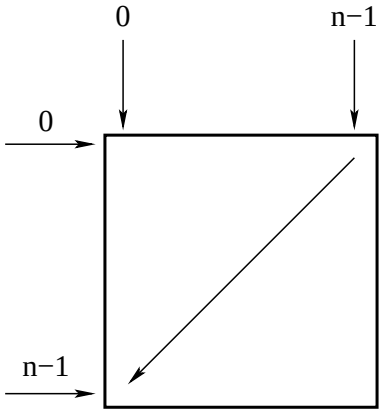
```
cont = 0;
for (i=0; i<n; i++) {
  A[i][i] = cont;
  cont++;
}

cont = 0;
for (i=0; i<n; i++) {
  if (A[i][i] == 0)
    cont++;
}
```

No último exemplo, o padrão conta quantos elementos nulos tem a diagonal principal. Se quisermos saber se a diagonal principal de uma matriz $A_{n\times n}$ tem todos os elementos nulos, basta comparar se `cont` é igual a `n`. Assim:

```
cont = 0;
for (i=0; i<n; i++) {
  if (A[i][i] == 0)
    cont++;
}
if (cont == n)
  printf ("A diagonal principal tem todos elementos nulos\n");
```

Percorrimento da Diagonal Secundária de uma Matriz:

Para verificar se uma matriz quadrada $A_{n\times n}$ tem a diagonal secundária com todos elementos nulos, devemos saber como percorrer esta diagonal da matriz `A`.



Como na diagonal secundária temos que a soma da linha com a coluna é igual a `n−1` (ou seja, para uma linha `i`, a coluna deve ser `n−1−i`), um padrão para percorrer a diagonal secundária de `A` é usar um comando de repetição (no caso, vamos usar o comando **for**) com uma variável inteira `i` para o índice das linhas e colunas da matriz `A`:

```
for (i=0; i<n; i++) {
  /* comandos usando a matriz A[i][n-1-i] */
}
```

Exemplos:

```
cont = 0;
for (i=0; i<n; i++) {
  A[i][n-1-i] = cont;
  cont++;
}
```

```
cont = 0;
for (i=0; i<n; i++) {
  if (A[i][n-1-i] == 0)
    cont++;
}
```

No último exemplo, o padrão conta quantos elementos nulos tem a diagonal secundária. Se quisermos saber se a diagonal secundária de uma matriz $A_{n\times n}$ tem todos os elementos nulos, basta comparar se `cont` é igual a `n`. Assim:

```
cont = 0;
for (i=0; i<n; i++) {
  if (A[i][n-1-i] == 0)
    cont++;
}
if (cont == n)
  printf ("A diagonal secundária tem todos elementos nulos\n");
```

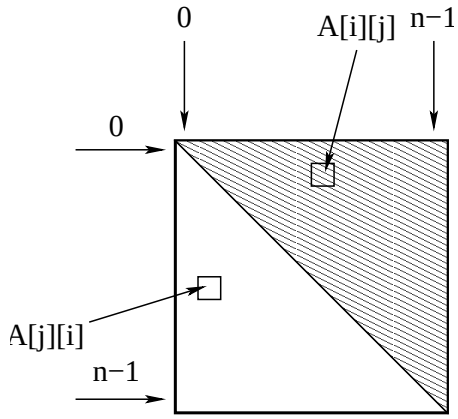
Juntando Tudo

Fica como exercício você fazer um programa que resolva o Exercício 1, ou seja, fazer um programa que leia um inteiro $n < 100$ e os elementos de uma matriz real $A_{n\times n}$ e verifica se a matriz A tem uma linha, coluna ou diagonal composta apenas por zeros.

Exercício 2

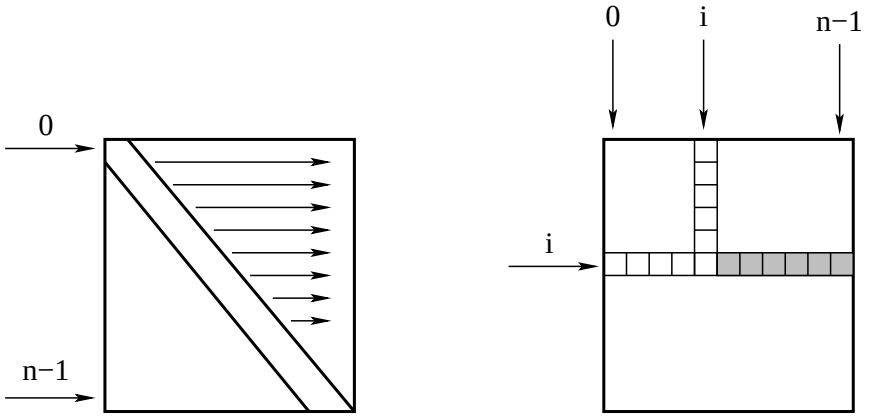
Dado $n < 200$ e uma matriz real $A_{n\times n}$, verificar se A é simétrica.

Uma matriz $A_{n\times n}$ é simétrica se, e somente se, A é igual a sua transposta, ou seja, $A = A^t$.



Neste caso, temos que verificar se cada $A[i][j]$ é igual a $A[j][i]$ como indicado na figura. Note que devemos percorrer somente uma parte da matriz, no caso, a parte superior da matriz.

Percorrimento da Parte Superior de Matrizes por Linha



Para uma linha `i`, temos que começar a percorrer as colunas a partir da coluna `i+1` até a última coluna `n−1`.

Assim, um padrão para percorrer uma linha `i` da parte superior de uma matriz `A` é usar um comando de repetição (no caso, vamos usar o comando **for**) com uma variável inteira `j` para o índice das colunas da matriz `A`:

```
/* para uma linha fixa i */
for (j=i+1; j<n; j++) {
  /* comandos que fazem algo com A[i][j] */
}
```

Exemplo:

```
cont = 0;
/* para uma linha fixa i */
for (j=i+1; j<n; j++) {
  A[i][j] = cont;
  cont++;
}
```

Um padrão para percorrer completamente a parte superior da matriz `A` por linhas é usar dois comandos de repetição (no caso, vamos usar o comando **for**) com duas variáveis inteiras `i` e `j`, um para percorrer as linhas e a outra para percorrer as colunas da matriz `A`:

```
for (i=0; i<n; i++) {
  /* para uma linha fixa i */
  for (j=i+1; j<n; j++) {
    /* comandos que fazem algo com A[i][j] */
  }
}
```

Exemplo:

```
cont = 0;
for (i=0; i<n; i++) {
  /* para uma linha fixa i */
  for (j=i+1; j<n; j++) {
    A[i][j] = cont;
    cont++;
  }
}
```

Assim, para verificar se uma matriz real $A_{n\times n}$, verificar se A é simétrica, podemos fazer:

```
simetrica = 1;
for (i=0; i<n; i++) {
  /* para uma linha fixa i */
  for (j=i+1; j<n; j++) {
    if (A[i][j] != A[j][i])
      simetrica = 0;
  }
}
printf ("Matriz ");
if (simetrica == 0) {
  printf ("nao ");
}
printf ("eh Simetrica\n");
```

Solução Completa:

```c
# include <stdio.h>

# define MAX 100

int main () {
  int i, j, n, simetrica = 1;
  float A[MAX][MAX];

  printf ("Entre com 0<n<100: ");
  scanf ("%d" &n);

  /* percorrer a matriz A elemento a elemento
  * colocando o valor lido pelo teclado */
  for (i=0; i<n; i++) {
    for (j=0; j<n; j++) {
      printf ("Entre com A[%d][%d] = ", i, j);
      scanf ("%f", &A[i][j]);
    }
  }

  /* verificando se eh simetrica */
  for (i=0; i<n; i++) {
    /* para uma linha fixa i */
    for (j=i+1; j<n; j++) {
      if (A[i][j] != A[j][i])
      simetrica = 0;
    }
  }

  /* Impressao da Resposta Final */
  printf ("Matriz ");
  if (simetrica == 0) {
    printf ("nao ");
  }
  printf ("eh Simetrica\n");
  return 0;
}
```

Note que neste exercício definimos uma constante `MAX` usando o "comando" `define`. Observe que `MAX` é uma constante e não uma variável. Mais sobre definição de constantes, veja o material didático **Alguns Detalhes da Linguagem C**.

Erros Comuns

Ao desenvolver seus programas com matrizes, preste atenção com relação aos seguintes detalhes:

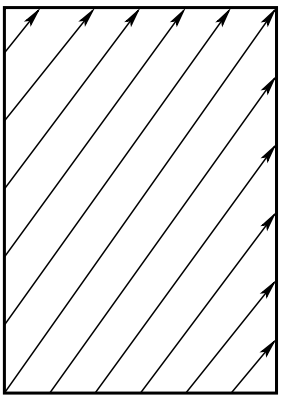
- **índices inválidos:** tome muito cuidado, especialmente dentro de um **while** ou **for**, de não utilizar índices negativos ou maiores que o tamanho máximo designado para as linhas e colunas da matriz.
- A **definição do tamanho das linhas e colunas da matriz** se faz na declaração da matriz. Os tamanhos das linhas e colunas são constantes; só mudando a sua declaração é que podemos alterar estes tamanhos. Isso significa que podemos estar "desperdiçando" algum espaço da memória por não estar usando todas as casas da matriz. Não cometa o erro de ler `n` e `m`, onde `n` e `m` seriam os tamanhos das linhas e colunas da matriz, e tentar "declarar" a matriz em seguida.

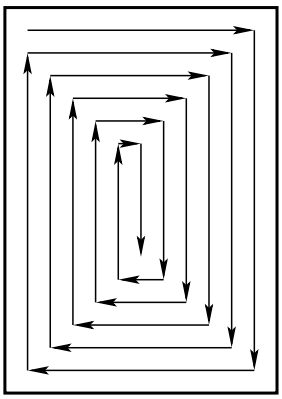
Percorrimento de Matrizes

Muitos problemas em MAC2166 que envolvem matrizes têm como soluções o uso de um padrão para percorrimento de matrizes. Nesta aula aprendemos:

- Percorrimento de uma Linha de uma Matriz.
- Percorrimento Completo da Matriz por Linhas.
- Percorrimento de uma Coluna de uma Matriz.
- Percorrimento Completo da Matriz por Colunas.
- Percorrimento da Diagonal Principal de uma Matriz.
- Percorrimento da Diagonal Secundária de uma Matriz.
- Percorrimento da Parte Superior de Matrizes por Linha.

Há muitas outras forma de percorrer matrizes. Por exemplo:





Exercícios Recomendados

1. Escreva um programa que, dadas duas matrizes $A_{m \times n}$ e $B_{n \times p}$, calcula a matriz $C_{m \times p}$ que é o produto de A por B.
2. Imprimir as n primeiras linhas do triângulo de Pascal.

3. Um jogo de palavras cruzadas pode ser representado por uma matriz $A_{m \times n}$ onde cada posição da matriz corresponde a um quadrado do jogo, sendo que 0 (zero) indica um quadrado branco e -1 indica um quadrado preto. Indicar na matriz as posições que são início de palavras horizontais e/ou verticais nos quadrados correspondentes (substituindo os zeros), considerando que uma palavra deve ter pelo menos duas letras. Para isso, numere consecutivamente tais posições.

   Exemplo: Dada a matriz:

$$\begin{pmatrix} 0 & -1 & 0 & -1 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & -1 & -1 \end{pmatrix}$$

   A saída deverá ser:

$$\begin{pmatrix} 1 & -1 & 2 & -1 & -1 & 3 & -1 & 4 \\ 5 & 6 & 0 & 0 & -1 & 7 & 0 & 0 \\ 8 & 0 & -1 & -1 & 9 & 0 & -1 & 0 \\ -1 & 10 & 0 & 11 & 0 & -1 & 12 & 0 \\ 13 & 0 & -1 & 14 & 0 & 0 & -1 & -1 \end{pmatrix}$$