matrizes

INF1005 -- Programação I -- 2016.1
Prof. Roberto Azevedo
razevedo@inf.puc-rio.br

matrizes

tópicos

- declaração (alocação estática)
- operações com matrizes

referência

- Capítulo 8 da apostila
- Capítulo 6 do livro

vetores bidimensionais: matrizes

- Uma **matriz** representa um conjunto **bi-dimensional** de valores na memória do computador

- Pode ser visto como uma **tabela de variáveis** de mesmo tipo que ocupa uma região contínua de memória.

- Similar a variáveis simples e vetores, **matrizes devem ser declaradas** para que o espaço de memória apropriado seja reservado.

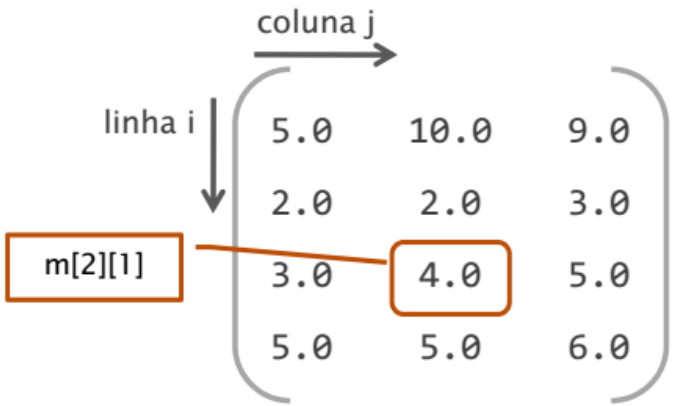
vetores bidimensionais: matrizes

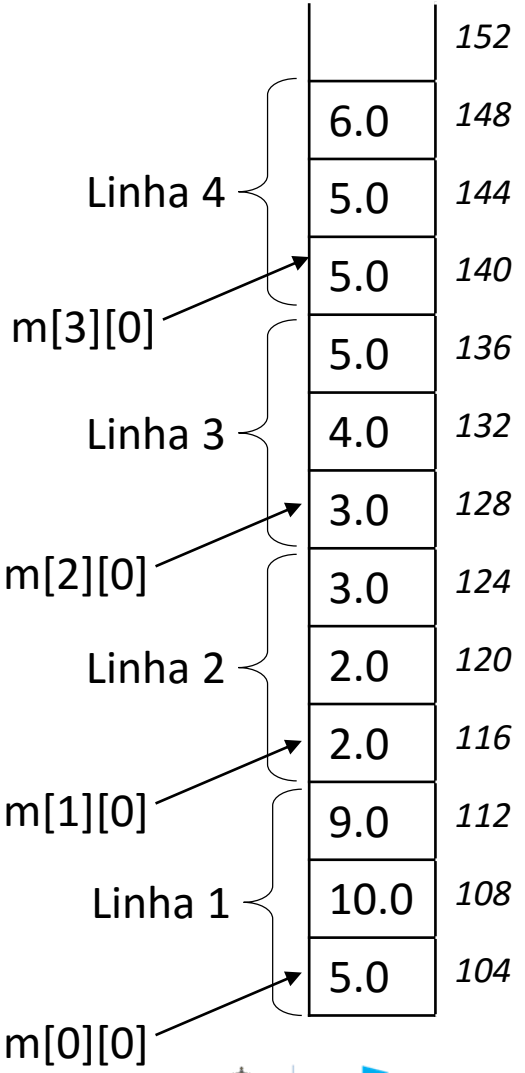
declarando com [linhas] e [colunas]:

```
float m[4][3];
```

já inicializando:

```
float m[4][3] = {
        {5.0, 10.0, 9.0},
        {2.0,  2.0, 3.0},
        {3.0,  4.0, 5.0},
        {5.0,  5.0, 6.0}
};
```





PUC RIO

DEPARTAMENTO DE INFORMÁTICA PUC-RIO

vetores bidimensionais: matrizes

outras formas de inicializar

```
float m[4][3] = { 5.0, 10.0, 9.0, 2.0, 2.0, 3.0, 3.0, 4.0, 5.0, 5.0,  5.0, 6.0};

float m[ ][3] = { 5.0, 10.0, 9.0, 2.0, 2.0, 3.0, 3.0, 4.0, 5.0, 5.0,  5.0, 6.0};

float m[ ][3] = {
                    {5.0, 10.0, 9.0},
                    {2.0,  2.0, 3.0},
                    {3.0,  4.0, 5.0},
                    {5.0,  5.0, 6.0}
};
```

indexação

Usamos indexação M[i][j] dupla para acessar elementos de M

- i indexa na dimensão primeira dimensão (linha)
- j indexa na segunda dimensão (coluna)
- $M[i][j]$ válido sse $0 \leq i < n$ e $0 \leq j < m$, para uma matriz $n \times m$.

importante: “Nome de Matriz”

Como matrizes são vetores (de vetores), valem as mesmas regras

- Em expressões, “M” (nome da matriz) sempre avalia para &M[0]
- M[i][j] é equivalente a ***(* (M + i) + j)**

indexação: exemplo

zerando os elementos

```
int m[4][3];
int linhas = 4, cols = 3;
int i, j;

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

acessando os elementos de uma matriz

leitura dos elementos de uma matriz

```
int m[4][3];
int linhas = 4, cols = 3;
int i, j;

for (i = 0; i < linhas; i++)
{
   for (j = 0; j < cols; j++)
   {
      scanf ("%d", &m[i][j]);
   }
}
```

acessando os elementos de uma matriz

imprimindo os elementos de uma matriz

```
int m[4][3];
int linhas = 4, cols = 3;
int i, j;

for (i = 0; i < linhas; i++)
{
   for (j = 0; j < cols; j++)
   {
     printf ("%d ", m[i][j]);
   }
   printf("\n");
}
```

matrizes como parâmetros de função

alternativas de declaração

```
void func (int M[10][20); /* ou */

void func (int M[ ][20]); /* ou */

void func (int (*M)[20]);
```

dentro de func, nos três casos, M comporta-se como **int (*)[20]** (ponteiro para vetor de 20 ints)

Chamada

```
int M[10][20];
...
func (M); /* ou */
func (&M[0]);
```

matrizes como parâmetros de função

```
void imprime_matriz (int linhas, int cols, float mat[4][3]); /* ou */

void imprime_matriz (int linhas, int cols, float (*mat)[3]); /* ou */

void imprime_matriz (int linhas, int cols, float mat[ ][3]);
```

```
void imprime_matriz (int linhas, int cols, float mat[ ][3])
{
   int i, j;
   for (i=0; i < linhas; i++)
      for (j=0; j < cols; j++)
         printf("linha %d, coluna %d: elemento %f\n", i, j, mat[i][j]);
}
```

exemplo com matriz

Entrada: arquivo com as três notas obtidas por cada aluno

7.5	8.5	7.8
8.4	9.2	6.8
9.1	10.0	9.5
4.0	5.2	4.3
4.3	6.0	5.8

Objetivo: ler as notas do arquivo e armazená-la na memória do computador para que, posteriorment seja possível processarmos as notas.

exemplo com matriz

Solução 1

declarar três vetores, um para cada nota:

```
float p1[N];
float p2[N];
float p3[N];
```

onde N é o número máximo de alunos (e.g. #define N 50)

Solução 2

declara uma matriz com todas as notas de todos os alunos

```
float notas[N][3];
```

as notas do i-ésimo aluno são representadas por notas[i][0], notas[i][1] e notas[i][2]

- **Todos os dados estão armazenados em uma única estrutura!**

exemplo com matriz (cont.)

```
#define N 50
int le_notas (double[][3], int);
double calcula_media (double[][3], int);

int main (void)
{
   double notas[N][3]; /* matriz 50x3 com as notas */
   double m; /* media da turma */
   int n; /* num linhas preenchidas por le_notas */
   n = le_notas (notas, N); /* le arquivo e preenche notas */
   m = calcula_media (notas, n);
   printf ("Media da turma: %.2g\n", m);
   return 0;
}

double calcula_media (double notas[][3], int n)
{
   /* retorna a media da turma */
}
```

exemplo com matriz (cont.)

```
double calcula_media (double notas[][3], int n)
{
   int i, j;
   double soma;
   for (i = 0; i < n; i++){ /* para cada linha i */
      for (j = 0; j < 3; j++) { /* para cada coluna j */
         soma = soma + mat[i][j];
      }
   }

   return soma / (3 * n);
}
```

exercícios: funções algébricas

escreva as seguintes funções. Em todos os casos assuma que as matrizes usadas são matrizes N x N, em que N é uma costante definida via macro-processador (e.g. #define N 25)

int simetrica **(**double A**[ ][**N**]);**

- retorna 1 se A é simétrica ou 0 caso contrário
- A é simétrica sse A[i][j] == A[j][i], para todo i, j

void criar_transposta **(**double A**[ ][**N**],** double B**[ ][**N**]);**

armazena em B a matriz transposta de A

- B é a transposta de A sse B[i][j] == A[j][i], para todo i, j

exercícios: funções algébricas

```
void transpor (double A[ ] [N]);
```

- transpõe a matriz A
- ou seja, troca A[i][j] por A[j][i], para todo i < j

```
void multiplica_escalar (double A[ ][N], double x);
```

- multiplica A por x
- ou seja, transforma A[i][j] em (A[i][j]) * x, para todo i, j