Matrizes

Disciplina de Programação de Computadores I
Universidade Federal de Ouro Preto

Link das videoaulas (Aula 8 - partes 1 a 6):
https://www.youtube.com/playlist?list=PL1K9y5L0Vn9VypwpLuFe-yjpIlQ9l2IXt

Agenda

- Matrizes bidimensionais
- Matrizes multidimensionais
- Múltiplas strings com matrizes
- Ocultando a dimensão na declaração
- Leitura de strings com espaços: função fgets

Matrizes: motivação

- Suponha que queiramos ler 1 string de até 10 caracteres. Como sabemos, uma string em C é um vetor de caracteres:

```
char minhastring [10];
```

- E se quisermos ler 3 strings de 10 caracteres?

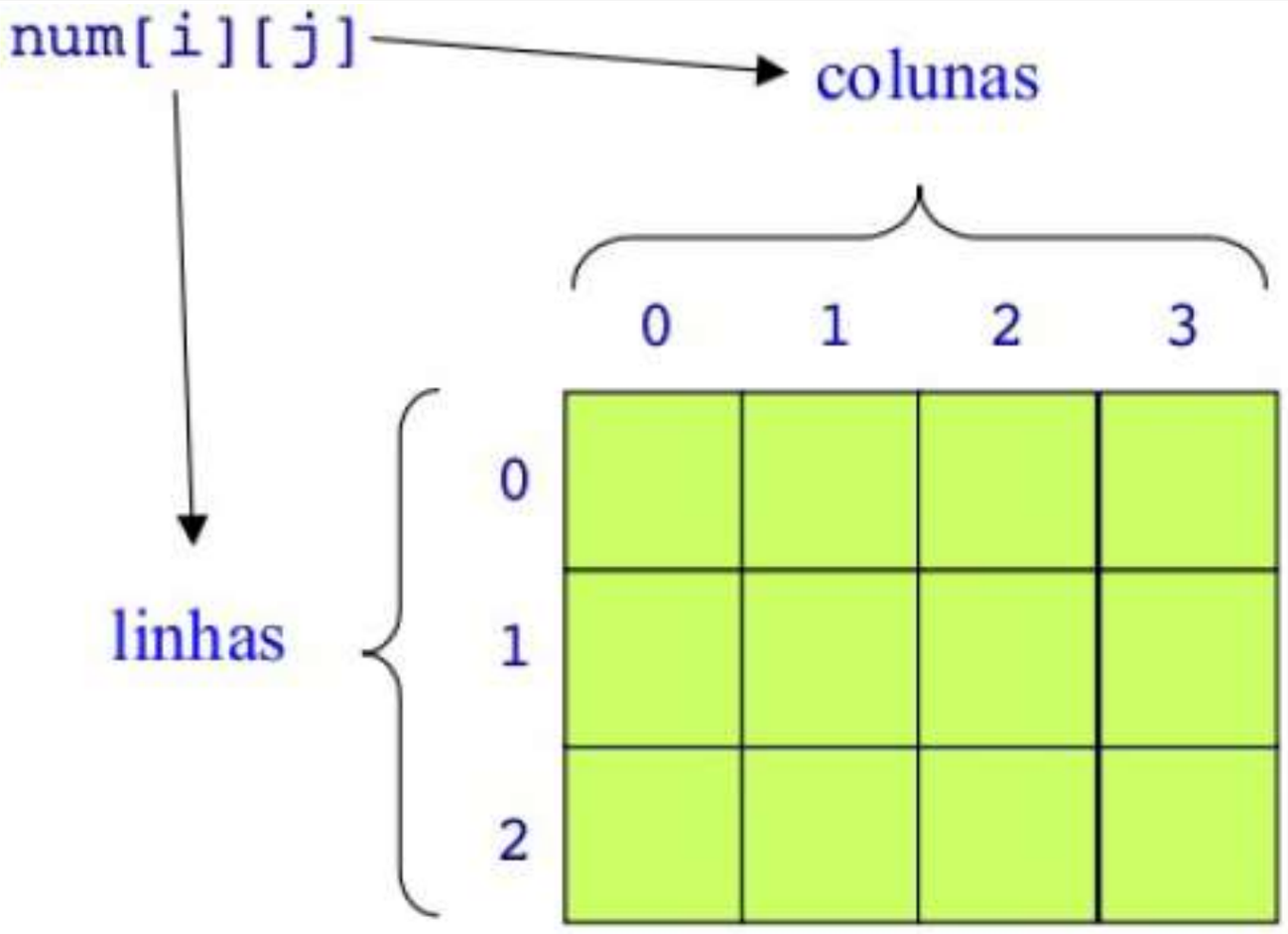
```
char minhastring1 [10];
char minhastring2 [10];
char minhastring3 [10];
```

- E se quisermos ler 300 strings de 10 caracteres?

Matriz: Definição

- Matriz: variável composta homogênea multidimensional.
    - Formada por uma sequência de variáveis, todas do mesmo tipo, com o mesmo nome, e alocadas sequencialmente na memória.
- A forma geral da declaração de uma matriz bidimensional (mais frequente) é muito parecida com a declaração de um vetor:
    - tipo_variável nome_variável [altura][largura];
- Também formada por uma sequência de variáveis do mesmo tipo.
    - Pense em vários vetores.
- Os componentes são identificados por um conjunto de índices.
- Utiliza-se um índice para cada dimensão.

Matrizes



- Qual o valor de i e j?

Matriz bidimencional: Declaração

- Uma matriz de 2 dimensões pode ser declarada assim:

tipo nome [dim1] [dim2];

- dim1 e dim2 são números inteiros ou variáveis do tipo int.
- Esta declaração cria dim1 x dim2 variáveis do tipo tipo.
- As variáveis criadas pelo vetor são acessadas por:
  - nome[0][0]
  - nome[0][1]
  - ...
  - nome[1][0]
  - nome[0][1]
  - ...
  - nome_do_vetor[dim1 -1][dim2 -1]
- O compilador não verifica se os valores para as dimensões são válidos.

Matriz: Matriz na memória

```
int m [ 4 ] [ 4 ];
```

m[0][0]	m[0][1]	m[0][2]	m[0][3]	m[1][0]	m[1][1]
m[1][2]	m[1][3]	m[2][0]	m[2][1]	m[2][2]	m[2][3]
m[3][0]	m[3][1]	m[3][2]	m[3][3]		

Matrizes

- **Declaração:**
  - Na declaração de uma variável matriz especificamos:
    - o tipo dos componentes da matriz
    - o nome da variável
    - o número de componentes que formam cada dimensão da matriz
    - os elementos do vetor (opcional)

```
tipo nome[dimensão1][dimensão2]...[dimensãoN];
```

- Exemplos:

```
int notas[2][6];
float y[2][4][3];
char mat [4][3] [= { { 'A', 'E', 'I' },
                     { '6', '5', '4' },
                     { '#', '@', '=' },
                     { 'm', 'n', 'z' }
                   };
```

Matriz: Preenchimento

- Podemos utilizar laços encaixados para preencher matrizes, sendo um laço para cada dimensão.

- Preencher a matriz: int m [3][4]; com 1’s

```
for (int i = 0; i < 3; i++)
        for (int j = 0; j < 4; j++)
                m[ i ][ j ] = 1;
```

Matriz: Inicialização

- Assim como vetores unidimensionais, matrizes podem ser inicializadas junto à sua declaração.

```
int vetor2D[3][4] =
{
    {1, 2, 3, 4},
    {1, 2, 3, 4},
    {1, 2, 3, 4}
};
```

```
int vetor3D[2][3][4] =
{
    {
        {1, 2, 3, 4},
        {1, 2, 3, 4},
        {1, 2, 3, 4}
    }
,
    {
        {1, 2, 3, 4},
        {1, 2, 3, 4},
        {1, 2, 3, 4}
    }
};
```

Matriz n-dimensional: Declaração

- Uma matriz de n dimensões pode ser declarada assim:

tipo nome [dim1] [dim2] [dim3] ... [dimN];

- dim1, dim2, ..., dimN são números inteiros ou variáveis do tipo int.
- Esta declaração cria dim1 x dim2 x ... x dimN variáveis do tipo tipo.
- As variáveis criadas pelo vetor são acessadas por:
  - nome[0][0][0]...[0]
  - nome[0][0][0]...[1]
  - nome[0][0][0]...[2]
  - ...
  - nome_do_vetor[dim1 -1][dim2 -1] ... [dimN -1]
- O compilador não verifica se os valores para as dimensões são válidos.

Matriz bidimensional de inteiros

```
int main(){
    int matriz[5][7], valor = 0;
    for (int i = 0; i < 5; ++i)
        for (int j = 0; j < 7; ++j)
            matriz[ i ][ j ] = valor++;

    for (int i = 0; i < 5; ++i){
        printf("[");
        for (int j = 0; j < 7; ++j)
            printf(" %d ", matriz[ i ][ j ]);
        printf("]\n");
    }
    return 0;
}
```

Saída:

```
[  0  1  2  3  4  5  6 ]
[  7  8  9 10 11 12 13 ]
[ 14 15 16 17 18 19 20 ]
[ 21 22 23 24 25 26 27 ]
[ 28 29 30 31 32 33 34 ]
```

Matriz bidimensional de caracteres

- Matriz para armazenar 300 strings de 10 caracteres:

```
char strings [ 300 ] [ 10 ];
```

- Preencimento:

```
for (int i = 0; i < 300; i++)
    scanf(“%s”, strings [ i ]);
```

- Preencimento:

```
for (int i = 0; i < 300; i++)
    printf("string[%d] = %s \n", i, string[i]);
```

Vetor não dimensionado (I)

- Ao declararmos um vetor, podemos omitir a sua primeira dimensão.

- Utilizamos esta propriedade ao declararmos e inicializarmos um vetor unidimensional:

```
int vetor[] = {1,2,3,4,5};
```

- Ou ao passarmos um vetor unidimensional como parâmetro de sub-rotina:

```
void subrotina (int vetor[ ], int tamanho_vetor);
```

Matriz multidimensional não dimensionada (II)

- No caso de uma matriz multidimensional, só podemos omitir o tamanho da primeira dimensão:

```
int matriz2D[ ][4] =
{
    {1, 2, 3, 4},
    {1, 2, 3, 4}
};

void subrotina (int matriz[ ][10], int qtde_linhas);
```

Vetor multidimensional não dimensionado (III)

- Não podemos omitir as demais dimensões do vetor multidimensional, mesmo se fornecermos a primeira dimensão!

```
int matriz2D[3][] =
{
    {1, 2, 3, 4},
    {1, 2, 3, 4},
    {1, 2, 3, 4}
};
```

Declarações Erradas!

```
void subrotina (int matriz[5][], int qtde_linhas);
```

Vetor multidimensional como parâmetro de sub-rotina

- Ao passarmos um vetor de qualquer dimensão como parâmetro de sub-rotina, o mesmo poderá ser alterado dentro da sub-rotina.

```
void zeraMatriz(int matriz[2][2]) {
    int i, j;
    for (i = 0; i < 2; i++)
        for (j = 0; j < 2; j++)
            matriz[i][j] = 0;
}

int main(){
    int mat[2][2] = { {0,1} , {2,3} };
    zeraMatriz(mat);
    return 0;
}
```

Referências Bibliográficas

- Material de aula da disciplina Algoritmos, UFJF: https://sites.google.com/site/algoritmosufjf
- Material de aula do Prof. Ricardo Anido, da UNICAMP: http://www.ic.unicamp.br/~ranido/mc102/
- Material de aula da Profa. Virgínia F. Mota: https://sites.google.com/site/virginiaferm/home/disciplinas
- DEITEL, P; DEITEL, H. *C How to Program*. 6a Ed. Pearson, 2010.

Agradecimentos

- Professores do Departamento de Ciência da Computação da UFJF que gentilmente permitiram a utilização das videoaulas elaboradas por eles.