

Vetores Multidimensionais

Aula 11

Marcos Silvano Almeida

marcoossilvano@professores.utfpr.edu.br

Departamento de Computação

UTFPR Campo Mourão

Decorative wavy lines in light blue and white are positioned on the right side of the slide, overlapping the dark blue background and the course code.

BCC31A

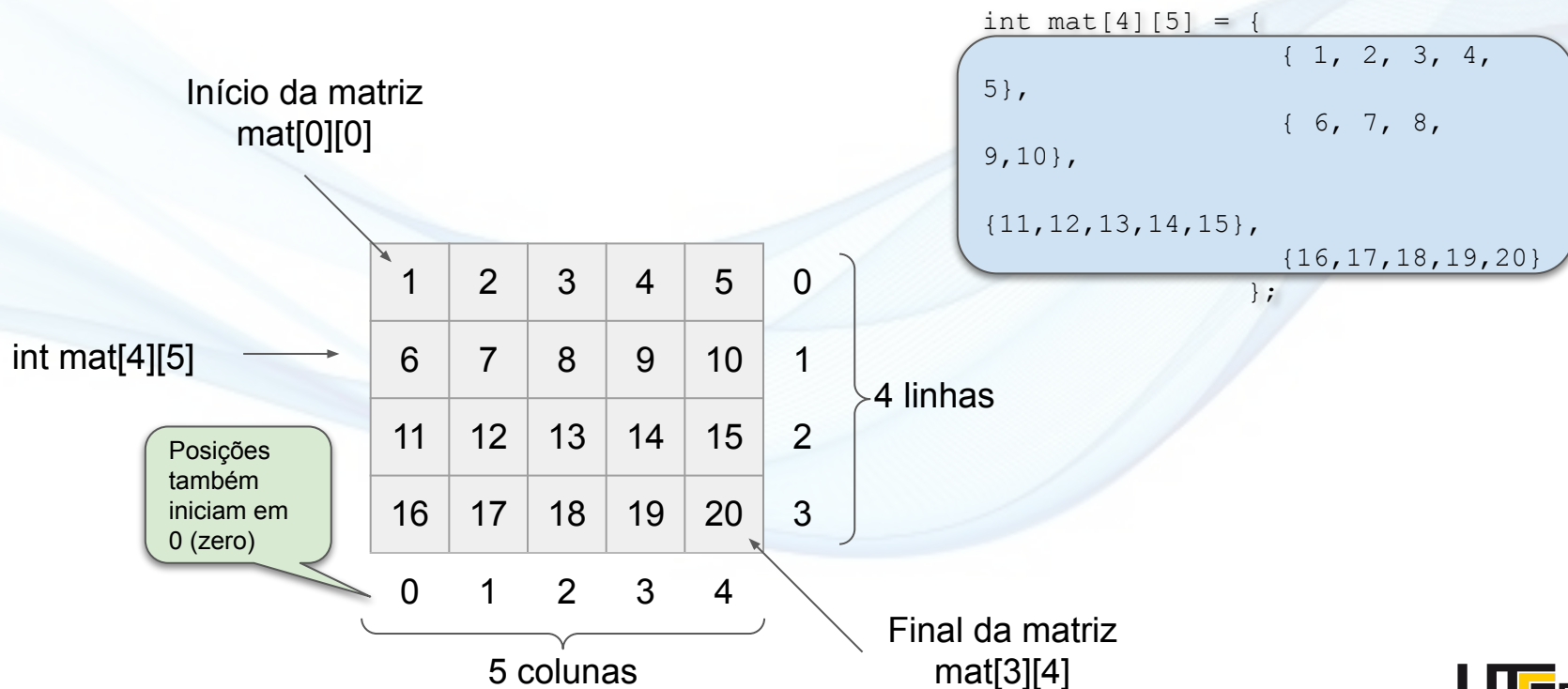
Questões sobre vetores e strings

- Não é possível **atribuir vetores**
 - Não existe cópia de vetores, apenas **inicialização**
 - É preciso criar função própria ou usar **memcpy()** (vetores) / **strcpy()** (strings)
- Não é possível **comparar** dois vetores pelos operadores =,<,<=,>,>=,!=
 - É necessário percorrer os dois vetores e comparar elemento a elemento
 - Para strings, existe **strcmp()**
- Um vetor não armazena seu tamanho
 - É necessário sempre guardá-lo em uma variável
 - Strings possuem um marcador especial ao final: NULL (0 ou '\0')
- Vetor passado como parâmetro à função é sempre uma referência (ponteiro)
 - Qualquer alteração dentro da função é refletida no vetor passado

Matrizes: vetores bidimensionais

Matrizes: vetores bidimensionais

- Podemos definir vetores de múltiplas dimensões (*vetor de vetores*)
- Caso mais comum: vetor bidimensional \Rightarrow matriz



Inicialização e acesso

- Na declaração da matriz
 - Primeira dimensão pode ser omitida (demais devem ser explícitas)

```
void main() {  
    printf("MATRIZES E ELEMENTOS\n");  
  
    int mat[ ][5] = { { 1, 2, 3, 4, 5}, // <- 4 x 5 elementos  
                     { 6, 7, 8, 9,10},  
                     {11,12,13,14,15},  
                     {16,17,18,19,20}  
                   };  
  
    int lin = 4;  
    int col = 5;  
    printf("primeiro elemento: %d\n", mat[0][0]);  
    printf("ultimo elemento: %d\n", mat[lin-1][col-1]);  
}
```

Inicialização

- Não é permitido declarar matrizes de tamanho **indefinido** + inicialização.

```
int rows = 4;          // PROBLEMA!  
int cols = 5;  
int mat[rows][cols] = {  
    { 1, 2, 3, 4, 5},  
    { 6, 7, 8, 9,10},  
    {11,12,13,14,15},  
    {16,17,18,19,20}  
};
```

```
int rows = 4;          // OK!  
int cols = 5;  
int mat[rows][cols];
```

Pode declarar matriz de tamanho indefinido, desde que não haja inicialização.

Impressão da Matriz

```
void main() {  
    int mat[4][5] = {  
        { 1, 2, 3, 4, 5},  
        { 6, 7, 8, 9,10},  
        {11,12,13,14,15},  
        {16,17,18,19,20}  
    };  
    // percorre e imprime a matriz  
    for (int i = 0; i < 4; i++) {  
        for (int j = 0; j < 5; j++) {  
            printf(" %2d", mat[i][j]);  
        }  
        printf("\n");  
    }  
}
```

Uma matriz é um **vetor de vetores**:
Cada elemento do primeiro vetor é
outro vetor.

Passando matriz para função

```
void printMatrix(int rows, int cols, int m[rows][cols]) {  
    for (int i = 0; i < rows; i++) {  
        for (int j = 0; j < cols; j++) {  
            printf(" %2d", m[i][j]);  
        }  
        printf("\n");  
    }  
}  
  
void main() {  
    int mat[2][5] = {  
        { 1, 2, 3, 4, 5},  
        {11,12,13,14,15}  
    };  
    printMatrix(4, 5, mat);  
}
```

Todo vetor declarado como parâmetro de função **aponta** para o vetor passado na chamada.
Logo, podemos modificar o vetor "externo" pelo código da função.

Vetor de Strings

Vetor de strings = Matriz de caracteres

```
int n = 4;
```

```
int tamString = 5;
```

```
char vetor_de_strings[n][tamString];
```

Visualizando como
uma matriz (forma
geométrica 2D)

```
{"C++", "Java", "C#", "Lua"}
```

cada string é um vetor de **chars**

```
{  
  {'C', '+', '+', '\\0', '\\0'},  
  {'J', 'a', 'v', 'a', '\\0'},  
  {'C', '#', '\\0', '\\0', '\\0'},  
  {'L', 'u', 'a', '\\0', '\\0'}  
}
```

		COLUNAS				
		0	1	2	3	4
LINHAS	0	C	+	+	\\0	\\0
	1	J	a	v	a	\\0
	2	C	#	\\0	\\0	\\0
	3	L	u	a	\\0	\\0

Vetor de strings = Matriz de caracteres

```
int n = 4;
```

```
int tamString = 5;
```

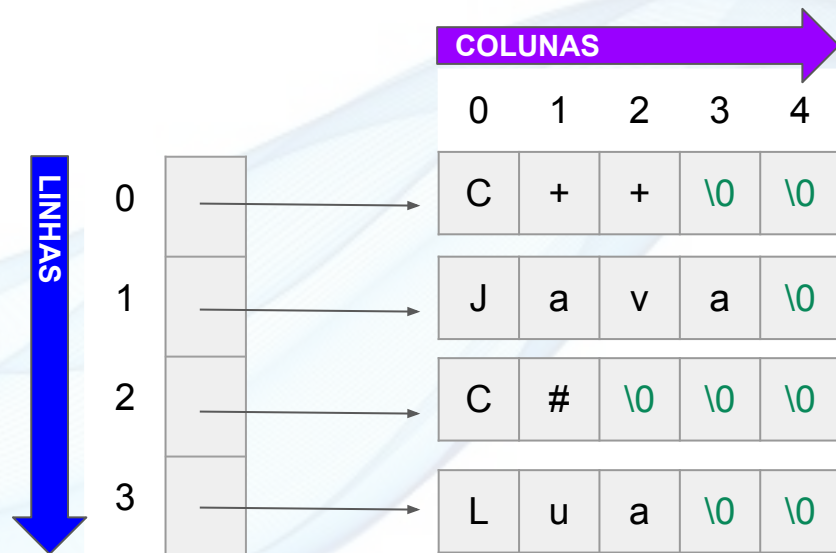
```
char vetor_de_strings[n][tamString];
```

Visualizando como
vetor de vetores

```
{"C++", "Java", "C#", "Lua"}
```

cada string é um vetor de **chars**

```
{  
  {'C', '+', '+', '\\0', '\\0'},  
  {'J', 'a', 'v', 'a', '\\0'},  
  {'C', '#', '\\0', '\\0', '\\0'},  
  {'L', 'u', 'a', '\\0', '\\0'}  
}
```

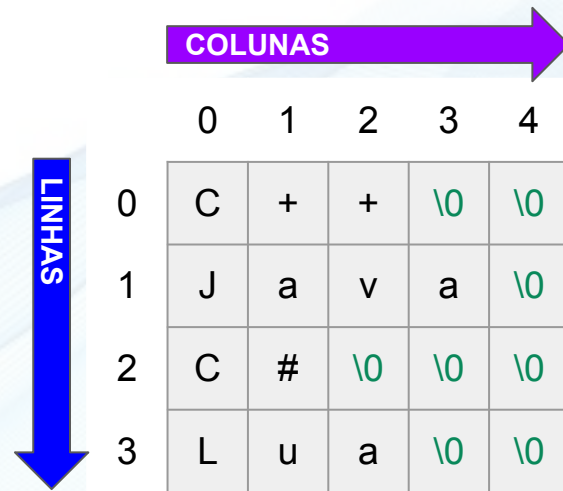


Imprimindo vetor de strings

```
#include <stdio.h>
#include <string.h>
```

```
void printStringVector(int n, int len, char v[n][len]) {
    // percorre vetor de strings
    for (int i = 0; i < n; i++) {
        printf("%s\n", v[i]);
    }
}

int main() {
    // vetor de 4 strings de 5 chars (4 char + '\0')
    char stringVector[4][5] = {"C++", "Java", "C#", "Lua"};
    printStringVector(4, 5, stringVector);
    return 0;
}
```



		COLUNAS				
		0	1	2	3	4
LINHAS	0	C	+	+	\0	\0
	1	J	a	v	a	\0
	2	C	#	\0	\0	\0
	3	L	u	a	\0	\0

Vetores de strings

```
// vetor de 7 strings de até 10 chars
char fruits[7][10] = {
    "Abacate", "Abacaxi", "Banana",
    "Caqui", "Laranja", "Melancia", "Uva"
};
// procurando uma palavra na lista
int pos = -1;
for (int i = 0; i < 7; i++) {
    if (strcmp(fruits[i], "Caqui") == 0) {
        pos = i;
    }
}
if (pos != -1) {
    printf("Caqui está na posição %d\n", pos);
}
```

char fruits[7][10]

0	A	b	a	c	a	t	e	\0		
1	A	b	a	c	a	x	i	\0		
2	B	a	n	a	n	a	\0			
3	C	a	q	u	i	\0				
4	L	a	r	a	n	j	a	\0		
5	M	e	l	a	n	c	i	a	\0	
6	U	v	a	\0						

7

10

Referências

- Algoritmos e Programação
 - Marcela Gonçalves dos Santos
 - Disponível pelo Moodle
- Estruturas de Dados, Waldemar Celes e José Lucas Rangel
 - PUC-RIO - Curso de Engenharia
 - Disponível pelo Moodle
- Linguagem C, Silvio do Lago Pereira
 - USP - Instituto de Matemática e Estatística
 - Disponível pelo Moodle
- Curso Interativo da Linguagem C
 - <https://www.tutorialspoint.com/cprogramming>