

Vetores Multidimensionais

Aula 11

Marcos Silvano Almeida

marcoossilvano@professores.utfpr.edu.br

Departamento de Computação

UTFPR Campo Mourão

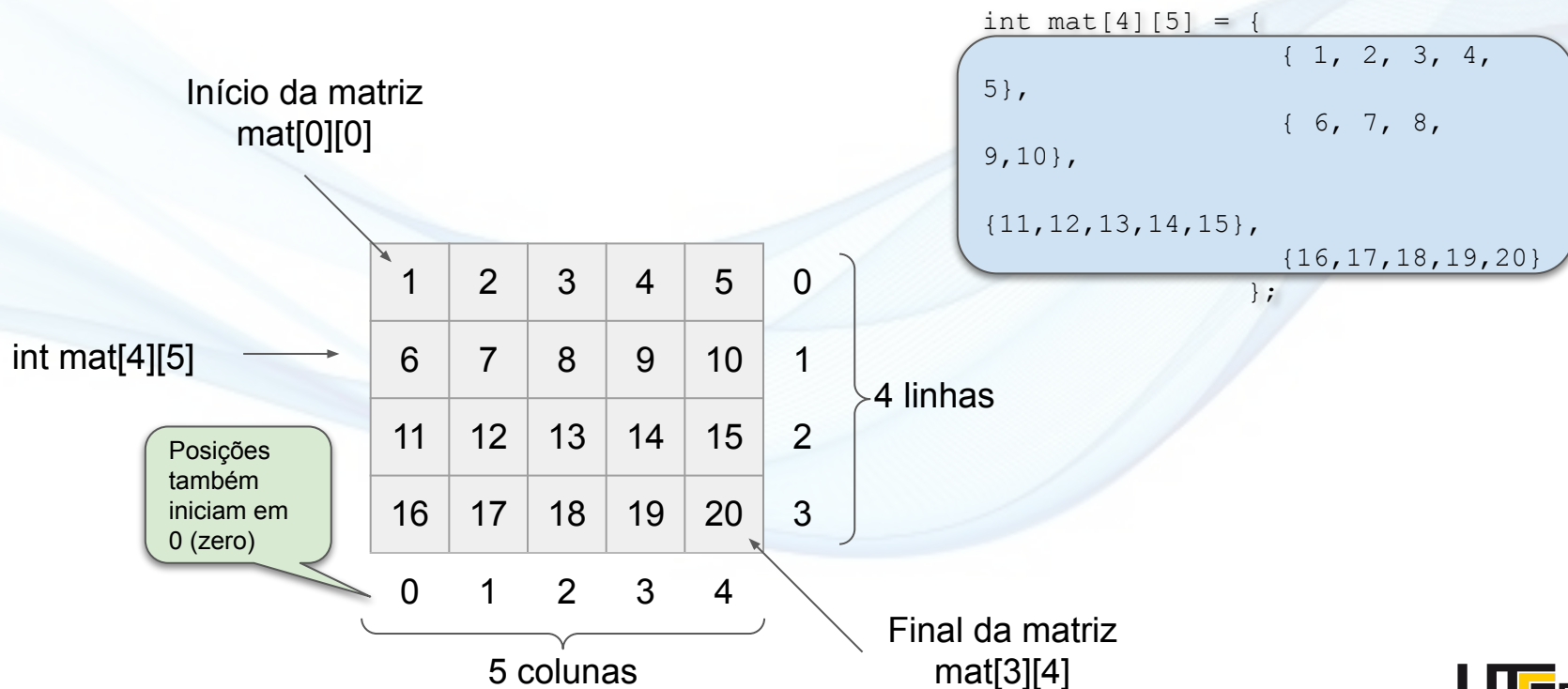
Questões sobre vetores e strings

- 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 um endereço
 - 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;
int cols = 5;
int mat[rows][cols] = {
    { 1, 2, 3, 4, 5}, // Não é permitido!
    { 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
um outro vetor.

Passando matriz para função

```
void print_matrix(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}  
    };  
  
    print_matrix(2, 5, mat);    // parâmetro m recebe o endereço de mat  
}
```

Lembre-se!

Todo vetor declarado como parâmetro de função **recebe o endereço** do vetor passado na chamada.

Vetor de Strings

Vetor de strings = Matriz de caracteres

```
char vetor_strings[4][5] = {"C++", "Java", "C#", "Lua"};
```

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

4 strings

De até 4 chars + NULL

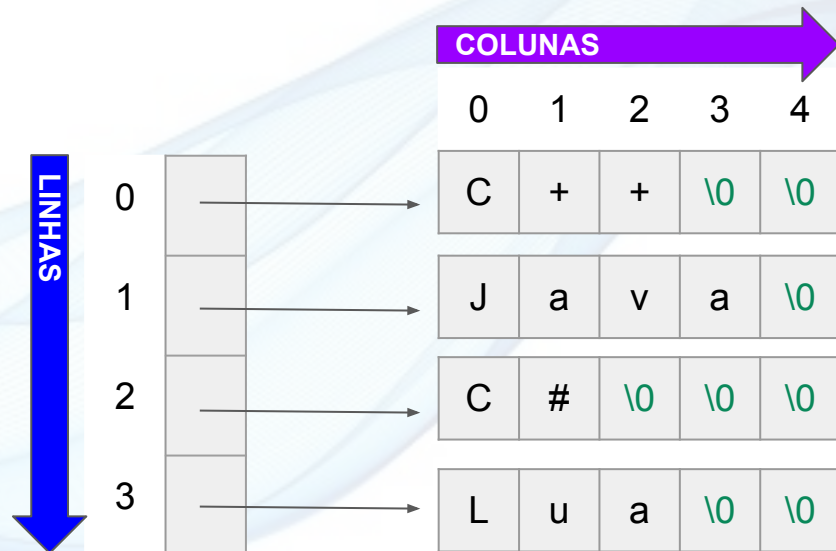
```
char vetor_strings[4][5] = {"C++", "Java", "C#", "Lua"};
```

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>
```

```
// imprimindo como vetor de strings
```

```
void print_string_vector(int n, int len, char v[n][len]) {
```

```
    // cada posição do vetor contém uma string
```

```
    for (int i = 0; i < n; i++) {
```

```
        printf("%s\n", v[i]);
```

```
    }
```

```
}
```

```
int main() {
```

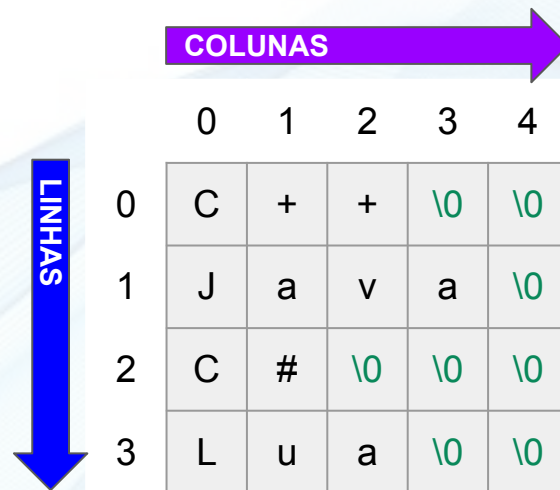
```
    // vetor de 4 strings de 5 chars (4 char + '\0')
```

```
    char string_vector[4][5] = {"C++", "Java", "C#", "Lua"};
```

```
    print_string_vector(4, 5, string_vector);
```

```
    return 0;
```

```
}
```



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

Imprimindo matriz de caracteres

```
#include <stdio.h>

void print_char_matrix (int rows, int cols, char v[rows][cols]) {
    // percorre cada linha
    for (int i = 0; i < rows; i++) {
        // percorre cada coluna/elemento da linha
        for (int j = 0; j < cols; j++) {
            printf("%c ", v[i][j]);
        }
        printf("\n");
    }
}

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

COLONAS

0 1 2 3 4

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

LINHAS

Vetor de strings

```
// vetor de 7 strings de até 9 chars (9 + \0)
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	\0	\0
1	A	b	a	c	a	x	i	\0	\0	\0
2	B	a	n	a	n	a	\0	\0	\0	\0
3	C	a	q	u	i	\0	\0	\0	\0	\0
4	L	a	r	a	n	j	a	\0	\0	\0
5	M	e	l	a	n	c	i	a	\0	\0
6	U	v	a	\0	\0	\0	\0	\0	\0	\0

7

10

Visualizando vetores de
três dimensões

Matriz de strings = Matriz 3D de caracteres

```
int rows = 3;
```

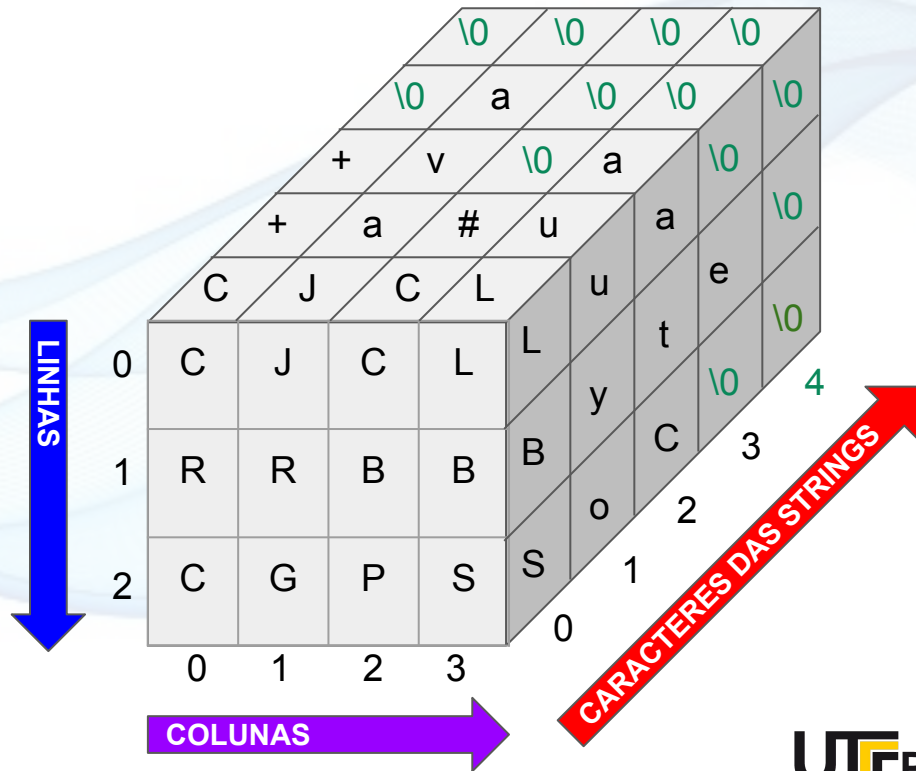
```
int cols = 4;
```

```
int len = 5;
```

```
char strings_matrix[rows][cols][len];
```

```
char strings_matrix[3][4][5] = {  
    {"C++", "Java", "C#", "Lua"},  
    {"RAM", "ROM", "Bit", "Byte"},  
    {"CPU", "GPU", "PPU", "SoC"}  
}
```

Visualizando como
forma geométrica 3D



Matriz de strings = Matriz 3D de caracteres

```
int rows = 3;
```

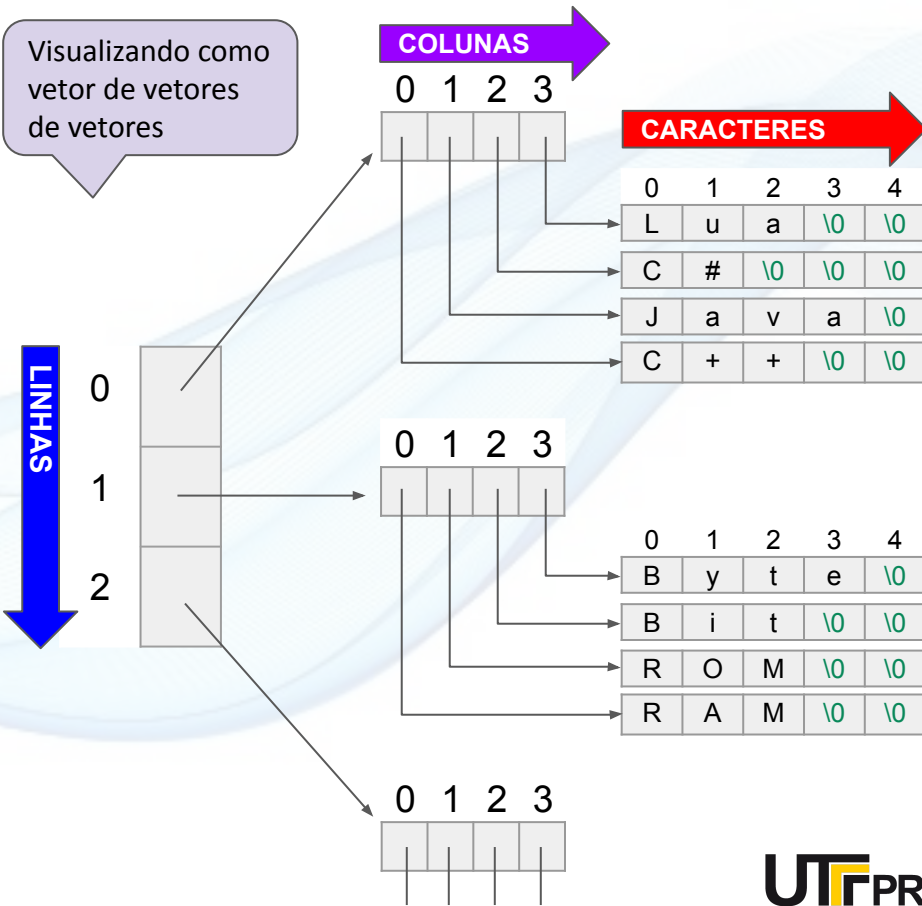
```
int cols = 4;
```

```
int len = 5;
```

```
char strings_matrix[rows][cols][len];
```

```
char strings_matrix[3][4][5] = {  
    {"C++", "Java", "C#", "Lua"},  
    {"RAM", "ROM", "Bit", "Byte"},  
    {"CPU", "GPU", "PPU", "SoC"}  
}
```

Visualizando como
vetor de vetores
de vetores

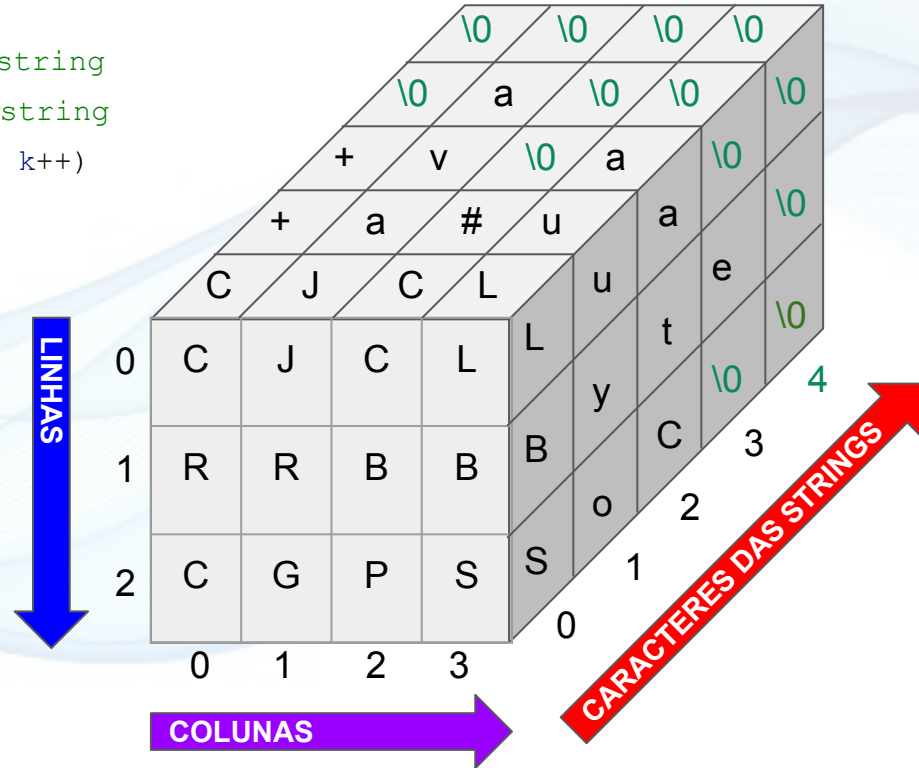


```

void print_string_matrix(int rows, int cols, int len, char v[rows][cols][len]) {
    // percorre as linhas da matrix
    for (int i = 0; i < rows; i++) {
        // percorre os elementos de cada linha
        for (int j = 0; j < cols; j++) {
            // percorre os caracteres de cada string
            // > cada célula (i,j) contém uma string
            for (int k = 0; v[i][j][k] != '\0'; k++)
                printf("%c", v[i][j][k]);
            printf(" ");
        }
        printf("\n");
    }
}

void main() {
    // matriz 3x4 de strings de até 4 chars
    char stringMatrix[3][4][5] = {
        {"C++", "Java", "C#", "Lua"},
        {"RAM", "ROM", "Bit", "Byte"},
        {"CPU", "GPU", "PPU", "SoC"}
    };
};

```



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>