

# Estrutura de Dados I

Profª Amanda Tameirão

	Col. 0	Col. 1	Col. 2	Col. 3	Col. 4
Linha 0	(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
Linha 1	(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
Linha 2	(2,0)	(2,1)	(2,2)	(2,3)	(2,4)
Linha 3	(3,0)	(3,1)	(3,2)	(3,3)	(3,4)

Matrizes  
*e strings*

# Matriz unidimensional (Vetor)

- Possui apenas uma dimensão
- `tipo nome_vetor[linha];`
- Começam pelo índice 0 (zero)
- Podem ser inicializadas na declaração
- Atribuir valor
- Comando de entrada
- Comando de saída

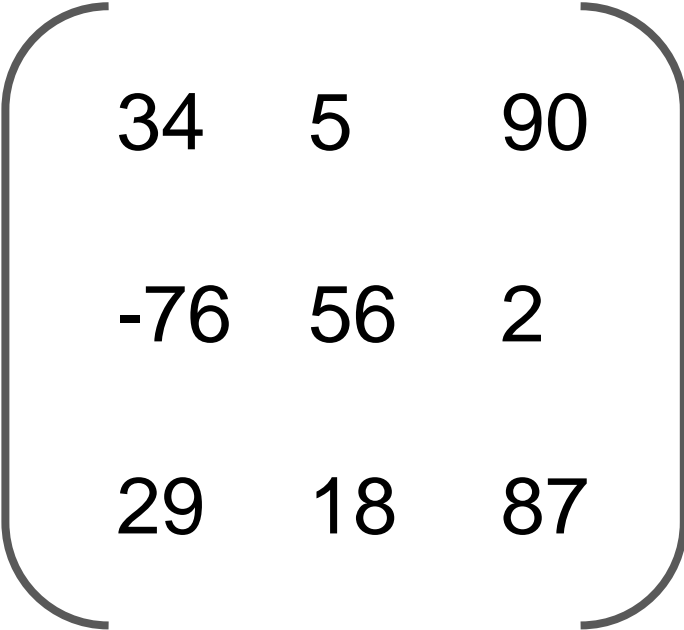
$$\begin{bmatrix} 34 \\ -76 \\ 29 \end{bmatrix}$$

# Matriz bidimensional

- Possui duas dimensões

`tipo nome_vetor[linha][coluna];`

- Começam pelo índice 0 (zero)
- Podem ser inicializadas na declaração
- Atribuir valor
- Comando de entrada
- Comando de saída



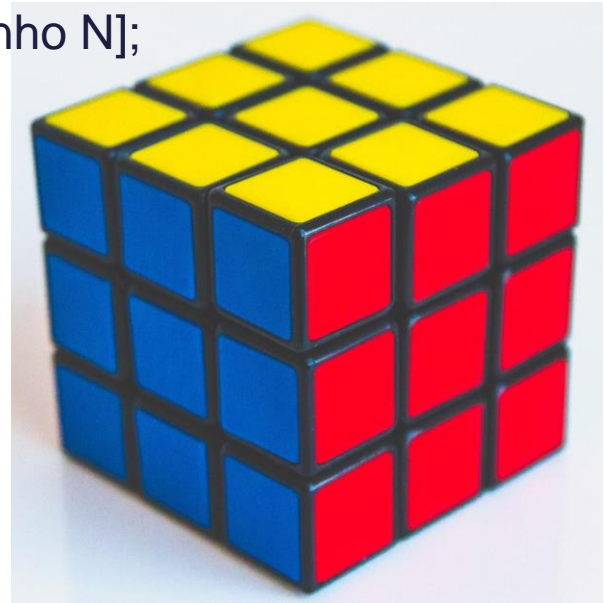
34	5	90
-76	56	2
29	18	87

# Matrizes Multidimensionais

- Possuem três ou mais dimensões

tipo nome\_matriz[tamanho1][tamanho 2]...[tamanho N];

- Começam pelo índice 0 (zero)



# Matrizes não dimensionadas

- Tamanho não especificado

```
tipo nome_vetor[] = {34, -76, 29};
```

```
tipo nome_matriz[][3] = {34, -76, 29, 5, 56, 18, 90, 2, 87};
```

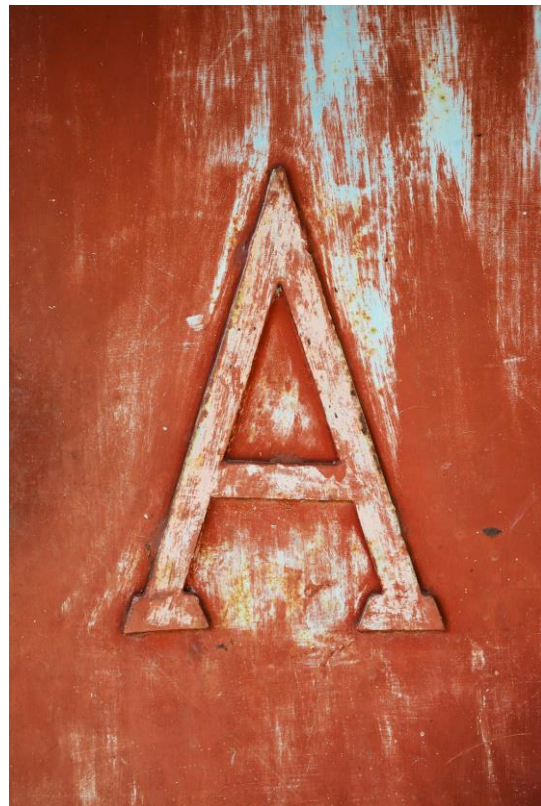


# Strings



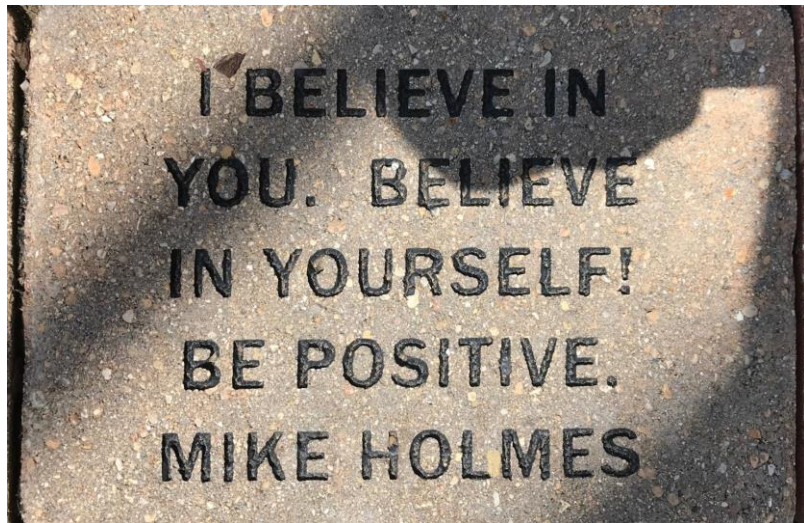
# Relembrando Caractere

- Ocupa 1 byte
- Formatado com %c
- Permite saber seu decimal com %d



# Strings

- Permite armazenar mais de um caractere
- Formado com um vetor de char
- Finalizado com `\0` - Caractere nulo
- Ao declarar, considere que o caractere nulo sempre ocupará uma posição.





# Strings - Ocupação na memória

- Pode ocupar todo espaço declarado

E	s	t	r	u	t	u	r	a	\0
---	---	---	---	---	---	---	---	---	----

- Pode ocupar menos que o espaço declarado

T	e	s	t	e	\0	?	?	?	?
---	---	---	---	---	----	---	---	---	---

- Não permita ultrapassar o espaço declarado



# Manipulação de Strings

# Comando de entrada

- `gets(string)`
- `fgets(string, tamanho, stdin)`
- `scanf("%s", string)` - Este só recebe até digitar enter ou espaço  
`scanf("%[^\n]", string)` - Este só recebe até digitar enter (incluindo espaços)



# Formatação de saída

- Para exibir utilize %s





A word cloud of string.h functions. The words are arranged in a roughly circular pattern, with some overlapping. The colors of the words include shades of green, blue, red, yellow, and brown. The functions listed are: strlen, strcpy, strerror, strtok, strncat, strspn, strcasecmp, strncmp, strcspn, strchr, strncpy, strrchr, and strcat.

strlen  
strcpy  
strerror  
strtok  
strncat  
strspn  
strcasecmp  
strncmp  
strcspn  
strchr  
strncpy  
strrchr  
strcat

**Funções da biblioteca string.h**

# Contar caracteres válidos

- **strlen**

```
int strlen(string)
```

Retorna a quantidade de caracteres válidos



# Copiar uma string em outra

- `strcpy`

```
void strcpy(string_destino, string_origem)
```

Substitui o conteúdo da string de destino com o conteúdo da string de origem.

# Copiar alguns caracteres de uma string em outra

- **strncpy**

`void strncpy(string_destino, string_origem, quantidade de caracteres)`

Substitui o conteúdo da string de destino com o conteúdo da string de origem, com a quantidade de caracteres indicados.

# Concatenar uma string em outra

- **strcat**

```
void strcat(string_destino, string_origem)
```

Concatena o conteúdo da string de destino com o conteúdo da string de origem.

# Concatenar alguns caracteres de uma string em outra

- **strncat**

`void strncat(string_destino, string_origem, quantidade de caracteres)`

Concatena o conteúdo da string de destino com o conteúdo da string de origem, com a quantidade de caracteres indicados.

# Comparar duas strings - lexicograficamente

- **strcmp**

`int strcmp(primeira_string, segunda_string)`

Retorna                      -1 - se a primeira for menor

0 - se forem iguais

1 - se a segunda for menor

# Exercícios de fixação

