MCTA028-15: Programação Estruturada

Aula 4: Vetores de Caracteres - `Strings` (Segunda Parte)

Wagner Tanaka Botelho
wagner.tanaka@ufabc.edu.br / wagtanaka@gmail.com
Universidade Federal do ABC (UFABC)
Centro de Matemática, Computação e Cognição (CMCC)

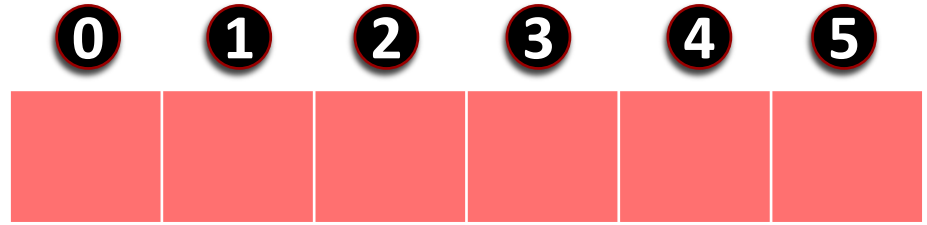
Vetores de Caracteres: `Strings`

Introdução

- **String:**
  - Sequência de caracteres, como uma palavra ou frase:
    - Armazenada na memória do computador na forma de um `array` do tipo `char`.

- Por ser um vetor de caracteres, sua declaração segue as regras de um vetor convencional:

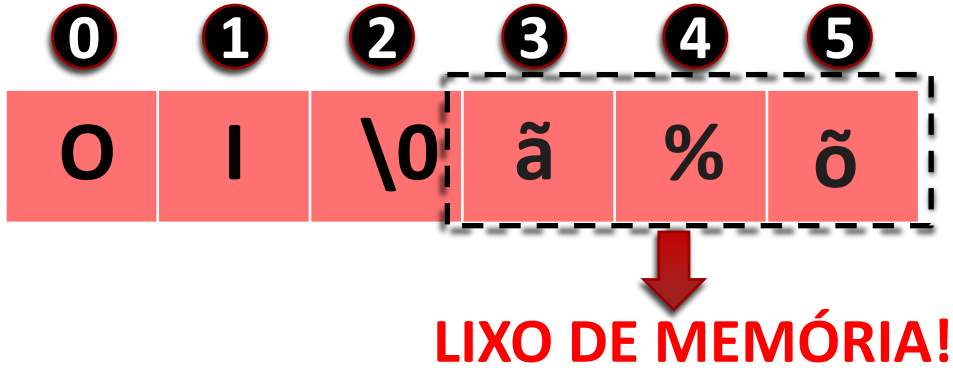
```
char str[6];
```



Cria na memória do computador uma `string` (*array* de caracteres) de nome **str** e tamanho igual a **6**.

As `strings` têm no elemento seguinte a ÚLTIMA letra da palavra/frase armazenada, um caractere "\0". Por que isso?

Pode-se definir uma `string` com tamanho maior do que a palavra armazenada. Por exemplo, uma `string` é definida com tamanho de 6 caracteres, mas utiliza-se apenas 2 caracteres para armazenar a palavra "**OI**". Neste caso, tem-se 4 posições NÃO utilizadas e que estão preenchidas com **LIXO** de **MEMÓRIA** (qualquer valor).



Assim, o caractere "**\0**" indica o **FIM** da sequência de caracteres e o início das posições restantes da `string` que **NÃO** estão sendo utilizadas nesse momento.

Como o caractere "**\0**" indica o final da string, isso significa que, em uma string definida com tamanho de 6 caracteres, apenas 5 estarão disponíveis.

Vetores de Caracteres: Inicializando

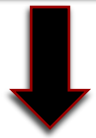
Vetores de Caracteres: Inicializando

- Uma `string` pode ser lida do teclado ou ser definida com um valor inicial;
- Para a sua inicialização:

```
char str [6] = { ‘O’, ‘I’, ‘\0’ };
```

- Também pode ser feita por meio de aspas duplas:

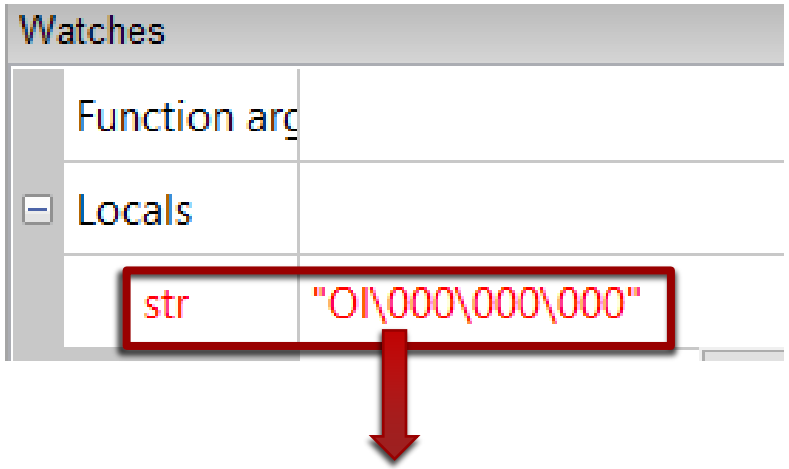
```
char str [6] = “OI”;
```



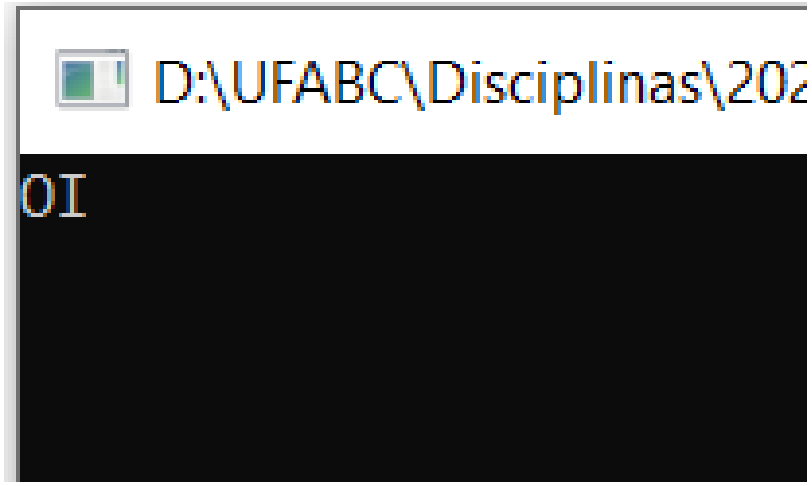
Possui a vantagem de já inserir o caractere **‘\0’** no **FINAL** da `string`.

Ex_12.c

```
1  #include <stdio.h>
2
3  void main(){
4
5      char str[6] = "OI";
6
7      printf("%s", str);
8  }
```



O caractere **'\0'** foi **INSERIDO** no **FINAL** da `string`.

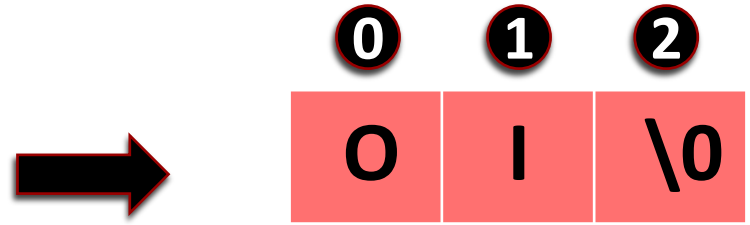


Vetores de Caracteres: Armazenando

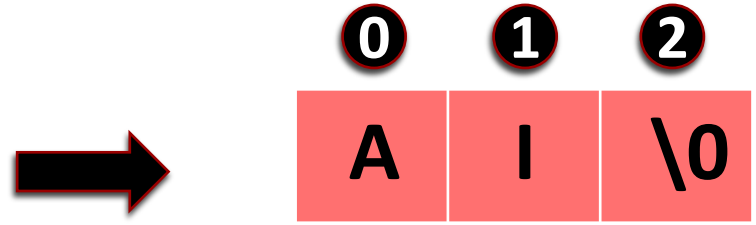
Vetores de Caracteres: Armazenando

- Cada caractere pode ser acessado individualmente por indexação como em qualquer outro vetor:

```
char str[3] = "OI";
```




```
str[0] = 'A';
```



IMPORTANTE!!

Na atribuição de `strings` usam-se aspas duplas, enquanto na de caracteres usam-se aspas simples.

Armazenando Pelo Teclado

Armazenando Pelo Teclado: scanf( )

```
char str[10];
```

```
scanf("%s", str);
```

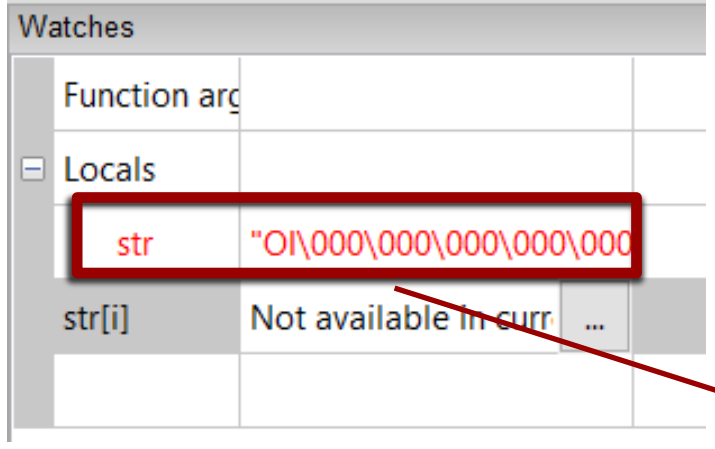
NÃO precisa do & antes do nome da variável. Os colchetes também NÂO são utilizados, pois TODA `string` será lida e não apenas UMA letra.

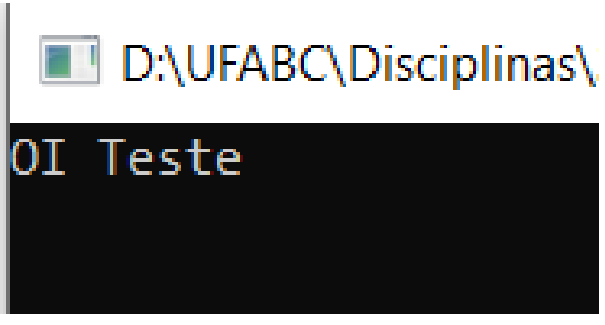
+ Para muitos casos, a função **scanf()** NÂO é a melhor opção para se ler uma string do teclado:

+ **`scanf()`**: lê apenas `strings` digitadas sem espaços, ou seja, palavras.

Ex_05.c

```
1  #include <stdio.h>
2
3  void main(){
4
5      char str[10];
6
7      scanf("%s", str);
8  }
```



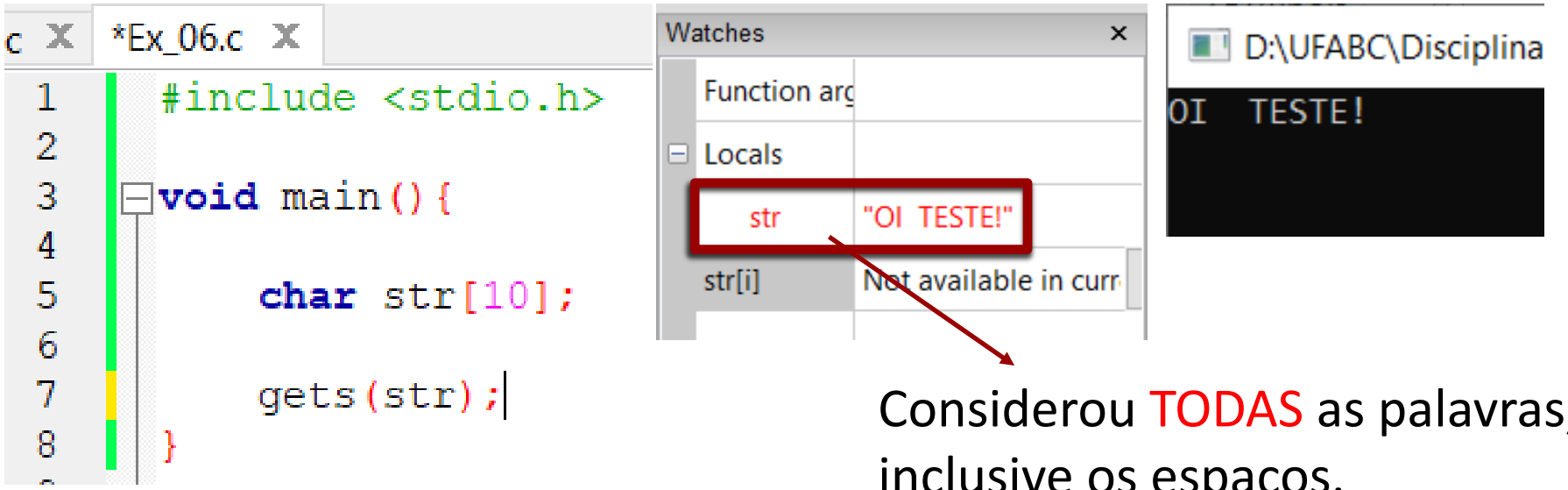


Considerou somente a primeira palavra antes do espaço.

Armazenando Pelo Teclado: gets()

- Uma alternativa mais eficiente para a leitura de uma `string` é a função `gets()`;
- O `gets()` faz a leitura do teclado considerando TODOS os caracteres digitados, incluindo os espaços, até encontra uma tecla enter:

```
char str[10];
gets(str);
```



Considerou TODAS as palavras, inclusive os espaços.

Vetores de Caracteres: Escrevendo na Tela - `printf()`

Escrevendo na Tela: `printf()`

- Para escrever uma `string` na tela utiliza-se a função **`printf()`** com o formato de dados "%s":

Ex_07.c

```
1  #include <stdio.h>
2
3  void main(){
4
5      char str[10];
6
7      printf("Digite uma frase: \n");
8      gets(str);
9
10     printf("Frase digitada: %s", str);
11 }
```

```
D:\UFABC\Disciplinas\2021-2025\Q1\PE\Aulas\04\Codigos\
Digite uma frase:
Oi Testando!
Frase digitada: Oi Testando!
Process returned 28 (0x1C)   execution time : 6
Press any key to continue.
```

Vetores de Caracteres: Percorrendo Cada Posição

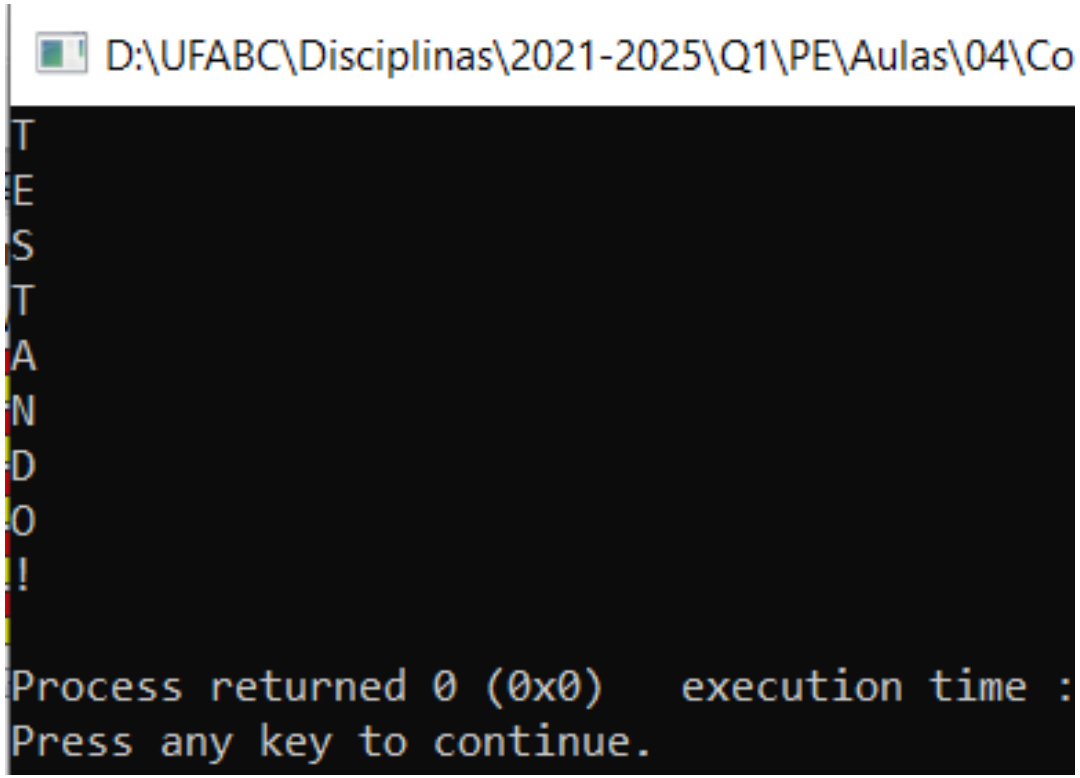
Ex_14.c

```c
#include <stdio.h>

int main(){
    int i=0;
    char str[10] = "TESTANDO!";

    for (i = 0; str[i]!='\0'; i++){
        printf("%c\n", str[i]);
    }
}
```

Percorre cada posição até encontrar o caractere **‘\0’** no **FINAL** da string.



Vetores de Caracteres: Manipulando `Strings`

Vetores de Caracteres: Manipulando `Strings`

- **`strlen()`**:
  - Obter o tamanho de uma `string`.

*Ex_08.c

```
1  #include<stdio.h>
2
3  void main(){
4      char str[3] = "OI";
5      int tam = 0;
6
7      tam = strlen(str);
8
9      printf("O tamanho eh: %d", tam);
10 }
```

D:\UFABC\Disciplinas\2021-2025\Q1\PE\Aulas\04\Cc

```
O tamanho eh: 2
Process returned 15 (0xF)   execution time
Press any key to continue.
```

Vetores de Caracteres: Manipulando `Strings`

- **`strcpy()`**:
  - Copiar o conteúdo de uma `string` para outra `string`.

Ex_09.c

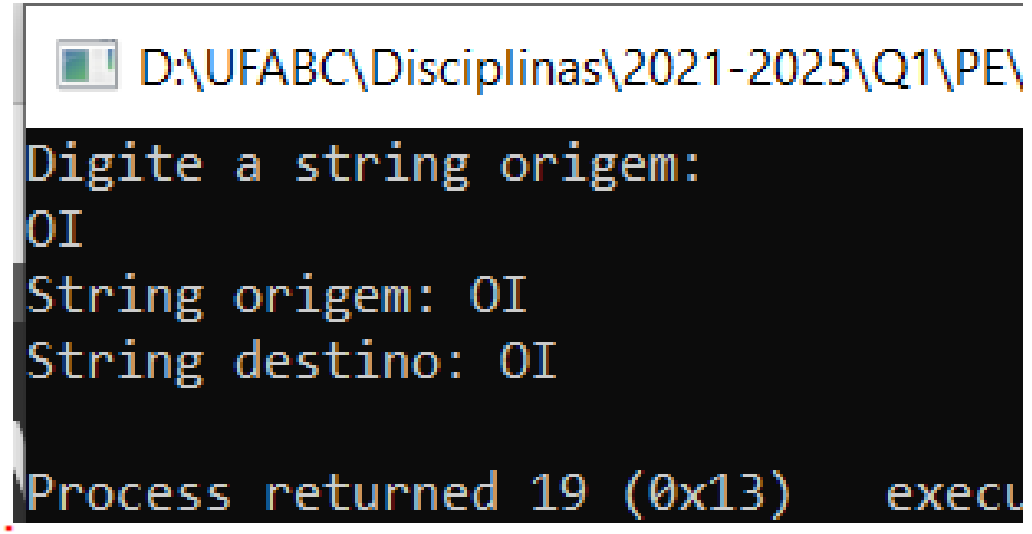
```c
#include <stdio.h>

void main(){
    char str_Origem[3], str_Destino[3];

    printf("Digite a string origem: \n");
    gets(str_Origem);

    strcpy(str_Destino, str_Origem);

    printf("String origem: %s\n", str_Origem);
    printf("String destino: %s\n", str_Destino);
}
```

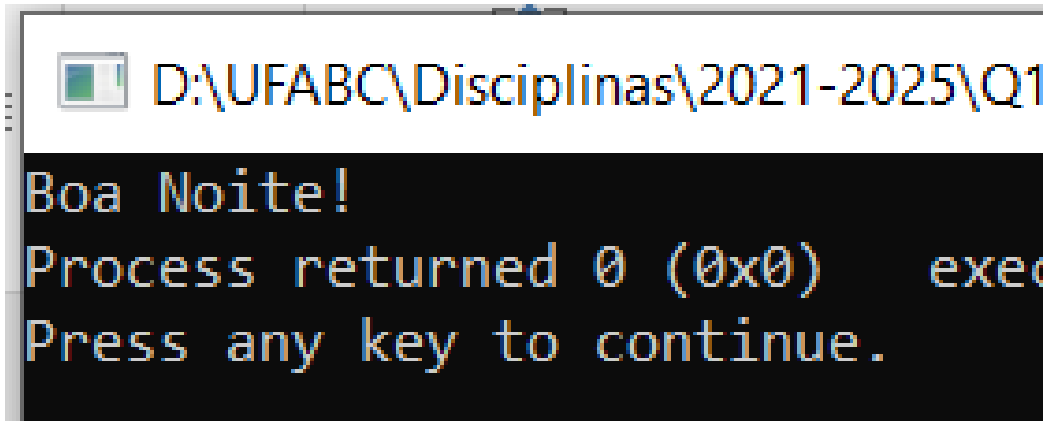


Vetores de Caracteres: Manipulando `Strings`

- **`strcat()`**:
  - Copiar uma `string` para o **FINAL** de outra `string`.

*Ex_10.c

```
1   #include <stdio.h>
2
3   int main(){
4       char str_1[10] = "Boa ";
5       char str_2[10] = "Noite!";
6
7       strcat(str_1, str_2);
8
9       printf("%s",str_1);
10  }
```



Copia a sequência de caracteres contida em str_2 para o FINAL da str_1

Vetores de Caracteres: Manipulando `Strings`

Ex_11.c

```c
#include <stdio.h>

int main(){

    char str_1[10], str_2[10];
    int result_Comp=0;

    printf("Entre com a string 01: \n");
    gets(str_1);

    printf("Entre com a string 02: \n");
    gets(str_2);

    result_Comp = strcmp(str_1, str_2);

    if(result_Comp == 0){
        printf("Strings IGUAIS!!!\n");
    }
    else{
        printf("Strings DIFERENTES!!!\n");
    }
}
```

- **`strcmp()`**:
  - Comparando duas `strings`.

Compara se a str_1 é IGUAL a str_2.

+ **result_Comp = zero:** significa que as strings str_1 e str_2 são IGUAIS;
+ **result_Comp = 1:** significa que as strings str_1 e str_2 são DIFERENTES.

Referências

- Slides do Prof. Luiz Rozante;
- SALES, André Barros de; AMVAME-NZE, Georges. Linguagem C: roteiro de experimentos para aulas práticas. 2016;
- BACKES, André. Linguagem C Completa e Descomplicada. Editora Campus. 2013;
- SCHILDT, Herbert. C Completo e Total. Makron Books. 1996;
- DAMAS, Luís. Linguagem C. LTC Editora. 1999;
- DEITEL, Paul e DEITEL, Harvey. C Como Programar. Pearson. 2011.