

LPG0001 – Linguagem de Programação

Cadeias de Caracteres (*Strings*)

Prof. Rui Jorge Tramontin Junior
Departamento de Ciência da Computação
UDESC / Joinville

Introdução

- Cadeias de caracteres (*strings*) não são tipos nativos da linguagem C;

Introdução

- Cadeias de caracteres (*strings*) não são tipos nativos da linguagem C;
- São implementadas como vetores (ou ponteiros) do tipo *char*;
 - Caractere '\0' marca o final do conteúdo;

Introdução

- Cadeias de caracteres (*strings*) não são tipos nativos da linguagem C;
- São implementadas como vetores (ou ponteiros) do tipo *char*;
 - Caractere `'\0'` marca o final do conteúdo;
- Utiliza-se o formato `"%s"`.

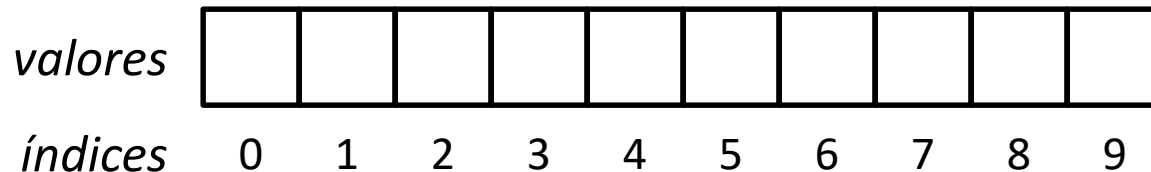
Exemplo: declaração, entrada e saída

```
char nome[10];
```

```
printf("Digite seu nome: ");
```

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

```
printf("Seu nome é %s\n", nome);
```



Exemplo: declaração, entrada e saída

```
char nome[10];
```

```
printf("Digite seu nome: ");
```

```
scanf("%s", nome); // suponha que foi digitado "João"
```

```
printf("Seu nome é %s\n", nome);
```

| | | | | | | | | | | |
|----------------|---|---|---|---|----|---|---|---|---|---|
| <i>valores</i> | J | o | ã | o | \0 | | | | | |
| <i>índices</i> | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

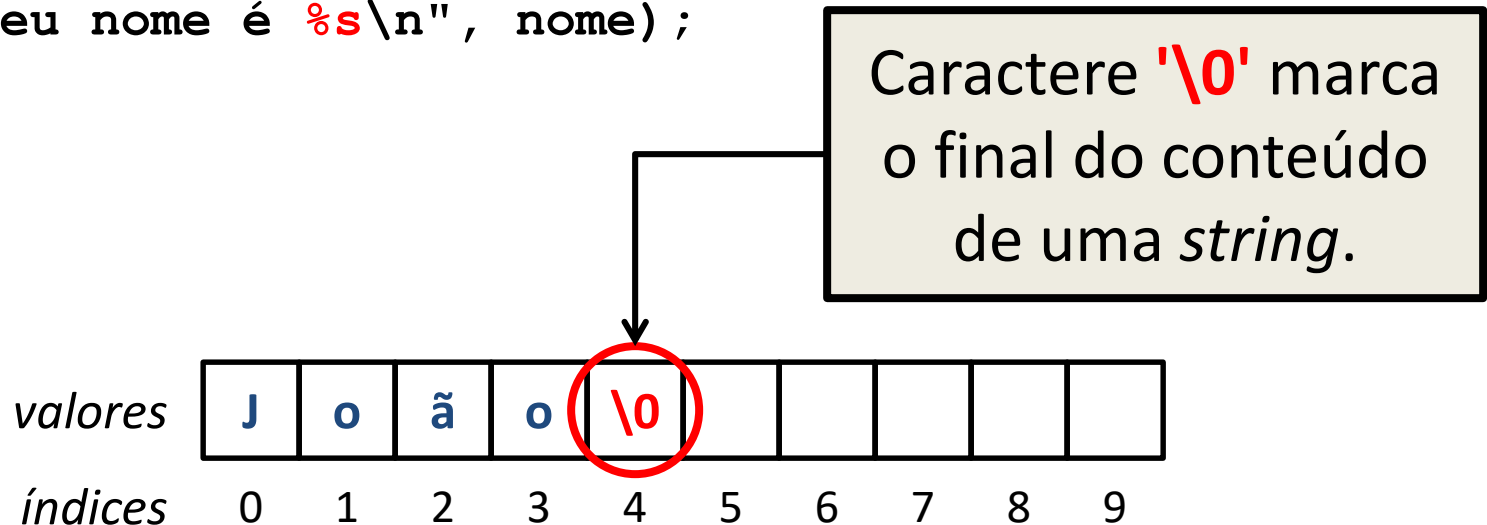
Exemplo: declaração, entrada e saída

```
char nome[10];
```

```
printf("Digite seu nome: ");
```

```
scanf("%s", nome); // suponha que foi digitado "João"
```

```
printf("Seu nome é %s\n", nome);
```



Inicialização de *strings*

- Vetores de caracteres (*strings*) podem ser inicializados de duas formas:

```
char nome1[] = { 'M', 'a', 'r', 'i', 'a' };
```

```
char nome2[] = "Maria";
```


Inicialização de *strings*

- Vetores de caracteres (*strings*) podem ser inicializados de duas formas:

```
char nome1[] = { 'M', 'a', 'r', 'i', 'a' };
```

```
char nome2[] = "Maria";
```

| | | | | | | |
|---------|---|---|---|---|---|----|
| valores | M | a | r | i | a | \0 |
| índices | 0 | 1 | 2 | 3 | 4 | 5 |

Manipulação de *strings*

- *Strings* podem ser manipuladas diretamente, caractere a caractere;

Manipulação de *strings*

- *Strings* podem ser manipuladas diretamente, caractere a caractere;
- Exemplo: mostrar a *string* no console:

```
printf("%s\n", nome) ;
```

Manipulação de *strings*

- *Strings* podem ser manipuladas diretamente, caractere a caractere;
- Exemplo: mostrar a *string* no console:

```
printf("%s\n", nome);
```

- ou:

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

Manipulação de *strings*

- Manipulação *char* a *char* tende a ser um processo trabalhoso, mas nem sempre há uma função pronta;

Manipulação de *strings*

- Manipulação *char* a *char* tende a ser um processo trabalhoso, mas nem sempre há uma função pronta;
- Todavia, pode-se utilizar a biblioteca *string.h*:
 - `strlen()` – conta a quantidade de caracteres;
 - `strcpy()` – atribuição de *strings*;
 - `strcmp()` – comparação entre *strings*;
 - `strcat()` – concatenação de *strings*;
 - ente outras funções

Função *strlen()*

- Retorna a quantidade de caracteres de uma *string*, contando a partir do índice 0 até encontrar o `'\0'`;
- Protótipo:

```
int strlen( char str[] );
```

Função *strlen()*

```
char nome[10] = "João";  
int n = strlen( nome ); // quantos caracteres...  
printf("%s tem %d caracteres.\n", nome, n); // 4
```


Função *strlen()*

```
char nome[10] = "João";  
int n = strlen( nome ); // quantos caracteres...  
printf("%s tem %d caracteres.\n", nome, n); // 4  
  
// Percorrendo a string...  
int i;  
for(i=0; i < n; i++)  
    printf("%c\n", nome[i]);
```

Função *strlen()*

```
char nome[10] = "João";  
int n = strlen( nome ); // quantos caracteres...  
printf("%s tem %d caracteres.\n", nome, n); // 4  
  
// Percorrendo a string...  
int i;  
for(i=0; i < n; i++)  
    printf("%c\n", nome[i]);  
  
// strlen() é diferente de sizeof()!  
int k = sizeof( nome ); // quantos bytes...  
printf("%s ocupa %d bytes.\n", nome, k); // 10
```

Função *strlen()*

- Sua implementação é bem simples;
- Basta percorrer até encontrar o **'\0'**, usando um contador;

```
int cont = 0;  
while( str[cont] != '\0' ){  
    cont++;  
}  
return cont;
```

Função *strcpy()*

- Faz a atribuição do conteúdo da *string* **orig** para a *string* **dest**;
- Protótipo:

```
void strcpy( char dest[], char orig[] );
```

Função *strcpy()*

```
char str1[10] = "João";
```

```
char str2[10];
```

```
// Não é possível fazer atribuição!
```

```
str2 = str1; // Erro de compilação!
```

Função *strcpy()*

```
char str1[10] = "João";
```

```
char str2[10];
```

```
// Não é possível fazer atribuição!
```

```
str2 = str1; // Erro de compilação!
```

```
// str2 recebe o conteúdo de str1...
```

```
strcpy( str2 , str1 );
```

```
printf("str2 = %s\n", str2);
```

Função *strcpy()*

- Sua implementação consiste em percorrer a *string* **orig** e copiar todos os caracteres para **dest**;

```
int i;  
for(i=0; orig[i] != '\0' ; i++) {  
    dest[i] = orig[i];  
}  
dest[i] = '\0';
```

Função *strcmp()*

- Compara o conteúdo (ordem alfabética) de duas *strings* **str1** e **str2**, e retorna:
 - 0: **str1** e **str2** são iguais;
 - Valor positivo: **str1** maior que **str2**;
 - Valor negativo: **str1** menor que **str2**;
- Protótipo:

```
int strcmp( char str1[], char str2[] );
```


Função *strcmp()*

```
char str1[] = "Maria";  
char str2[] = "Maria";  
char str3[] = "Rui";  
  
int n = strcmp( str1 , str2 );  
// Saída: 0, pois str1 == str2.  
printf("%d\n", n);
```

Função *strcmp()*

```
char str1[] = "Maria";  
char str2[] = "Maria";  
char str3[] = "Rui";
```

```
int n = strcmp( str1 , str2 );  
// Saída: 0, pois str1 == str2.  
printf("%d\n", n);
```

```
n = strcmp( str1 , str3 );  
// Saída: valor negativo, pois str1 < str3.  
printf("%d\n", n);
```

Exemplos Práticos

- Comparação usando *strcmp()*;
- Concatenação usando *strcat()*.

Exercícios em aula

1. Dados uma *string* *s* e um *char* *c*, verifique se *s* contém *c*;
2. Verifique se uma *string* contém somente dígitos;
3. Converta uma *string* que contém somente dígitos para um valor inteiro (variável *int*).