



LÓGICA DE PROGRAMAÇÃO

PROF^a. M.Sc. JULIANA H Q BENACCHIO

Manipulação de Strings

- String é utilizada para representação e manipulação de sequências de caracteres.
- Um literal de caractere é um valor inteiro representado como caractere entre aspas simples. Por exemplo, 'z' representa o valor inteiro de z

Manipulação de Strings

- O valor de um literal de caractere é o valor inteiro do caractere no conjunto de caracteres ASCII (*American Standard Code for Information Interchange*).

A = 65

a = 97

z = 122

é = 130

string

- É uma matriz unidimensional (vetor) do tipo **char** (caractere)
- Toda string deve terminar pelo caractere nulo ‘\0’, que tem valor decimal igual a 0.
- A declaração geral para uma string é:

```
char nome_da_string[tamanho];
```

string

- Por causa do terminador ‘\0’, é preciso declarar a string considerando um caractere mais longo que a string que se deseja armazenar.
- Por exemplo, para declarar uma matriz **str** que guarda uma string de 10 caracteres:

```
char str[11];
```

Manipulação de Strings

- Embora a linguagem C não tenha o tipo de dado string, ela permite constantes string.
- Uma **constante string** é uma lista de caracteres entre aspas. Não é preciso adicionar o nulo no final das constantes string manualmente, o compilador C faz isso automaticamente.

"Joao da Silva" (um nome)

"Foz, PR" (uma cidade e um estado)

Inicializando string

- Caracteres simples são colocados entres aspas simples e o conjunto é envolto por chaves

```
char nome[5] = {'A', 'n', 'a', '\0'};
```

```
char nome[5] = "Ana";
```

```
char nome[] = "Ana";
```

```
nome[1] = 'd';
```

Inicializando string

- Atribuição em variáveis NÃO é permitido!

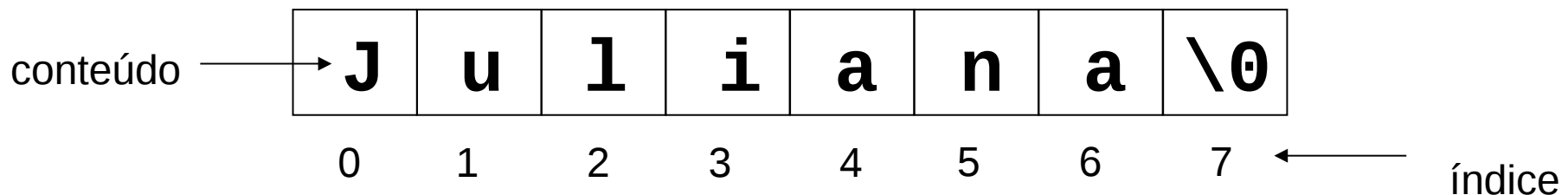
~~nome = "Ana";~~

Inicializando string

- Somente no momento da declaração

```
char nome[8] = "Juliana";
```

- Não é necessário adicionar o caractere nulo no final da string manualmente



Lendo e Imprimindo strings

- Funções de entrada:
 - `gets()` / `fgets()`
 - `scanf()`

- Funções de saída:
 - `puts()`
 - `printf()`

Lendo strings - scanf()

- Bastante limitada para leitura de strings
- Lê os caracteres digitados até encontrar qualquer espaço em branco
- Caractere de conversão '**%s**'
- Sua forma geral é:

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

```
scanf("%s", &nome_da_string[0]);
```

Lendo strings - gets()

- Lê os caracteres digitados até encontrar o caractere de nova linha '**\n**'
- Todos os caracteres anteriores ao '**\n**' são armazenado na string e por último é incluído o caractere '**\0**'
- Caracteres brancos como espaços e tabulações são aceitos como parte da string
- Sua forma geral é:

```
gets(nome_da_string);
```

Função `gets()` - *deprecated*

**Obsoleto
Descontinuado**

string1.c: In function 'main':

string1.c:7:2: warning: 'gets' is deprecated

(declared at /usr/include/stdio.h:638) [-Wdeprecated-declarations]

```
gets(nome);
```

```
^
```

/tmp/ccz430DP.o: na função `main':

string1.c:(.text+0x2e): aviso: the `gets' function is dangerous and should not be used.

Compilação terminada com sucesso.

Função `gets()` - *deprecated*

- A função **`gets`** pode gerar um grande problema para o programador que a usa: como essa função não limita o número de caracteres a serem lidos da entrada padrão (**`stdin`**), pode haver vazamento de memória – ***buffer overflow***.
- A solução é usar **`fgets`**, que limita o *buffer* de leitura.

Lendo strings - fgets()

- Similar ao **gets**, lê os caracteres digitados até encontrar o caractere de nova linha '**\n**'
- Porém o **fgets()** adiciona a quebra de linha ('**\n**') no final da string
- Sua forma geral é:

```
fgets(string, quant, file_stream);
```

Lendo strings - fgets()

- **file stream** → fluxo de arquivo
- Pode ser um arquivo de texto, ou no caso, **stdin**, que é o arquivo que representa a entrada padrão (teclado)
- Uma forma de utilização seria:

```
fgets(str, 255, stdin);
```


Lendo strings - fgets()

- Operador **sizeof**, é um operador em tempo de compilação unário que retorna o tamanho, em bytes, da variável em parênteses
- Exemplo:

```
fgets(str, sizeof(str), stdin);
```

Imprimindo strings - puts()

- Complemento da função **gets()**
- Imprime uma única string por vez
- Reconhece o '**\0**' como fim da string, por isso que cada string impressa por **puts()** termina por um caractere de nova linha.
- Ou seja, a função **puts()** pula de linha sozinha
- Sua forma geral é:

```
puts(string);
```

Imprimindo strings - printf()



- Caractere de conversão ' %s '
- Sua forma geral é:

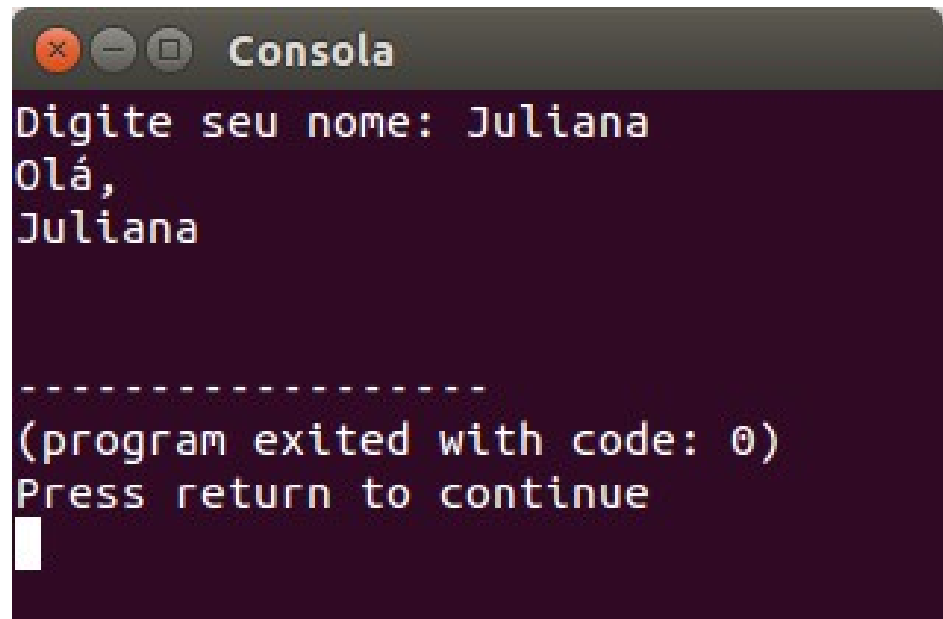
```
printf("%s", string);
```

- É possível imprimir mais de uma string na mesma linha

```
printf("%s %s\n", string1, string2);
```

Lendo e Imprimindo strings

```
int main(){
    char nome[50];
    printf("Digite seu nome: ");
    fgets(nome, sizeof(nome), stdin);
    puts("Olá, ");
    puts(nome);
    return 0;
}
```

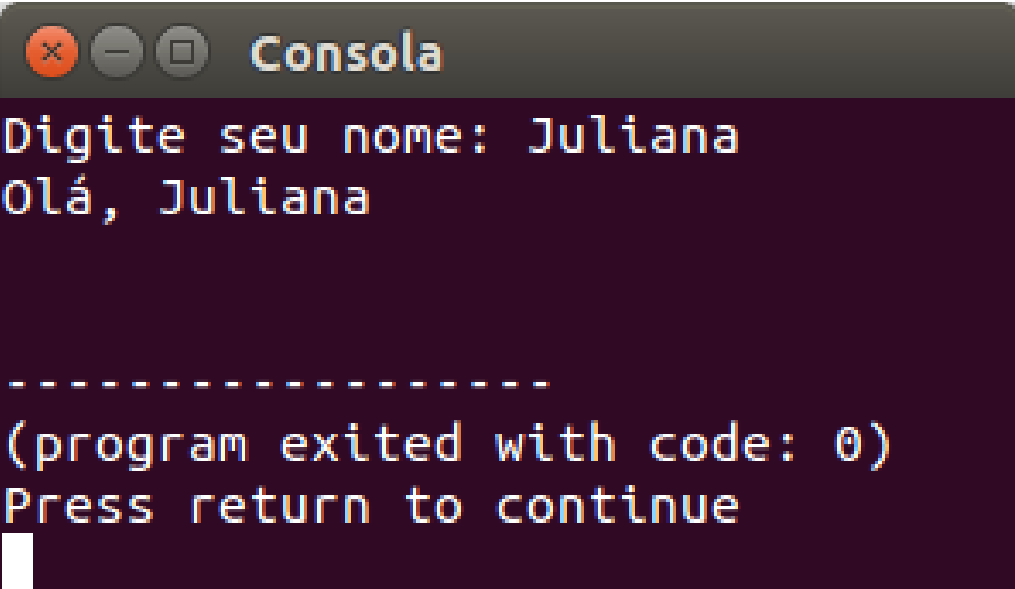


```
Consola
Digite seu nome: Juliana
Olá,
Juliana

-----
(program exited with code: 0)
Press return to continue
```

Lendo e Imprimindo strings

```
int main(){
    char nome[50];
    printf("Digite seu nome: ");
    fgets(nome, sizeof(nome), stdin);
    printf("Olá, %s \n", nome);
    return 0;
}
```



```
Consola
Digite seu nome: Juliana
Olá, Juliana

-----
(program exited with code: 0)
Press return to continue
```

Manipulação de strings

Funções de manipulação:

#include <string.h>

strlen

strcpy

strcat

strcmp

Função `strlen()`

```
tamanho = strlen(string);
```

- Retorna o tamanho da string fornecida
- O caractere nulo '`\0`' não é contado
- Isto quer dizer que, de fato, o comprimento do vetor da string deve ser um a mais que o inteiro retornado por `strlen()`

Função strlen()

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

int main()
{
    int tam;
    tam = strlen("teste");
    printf("A string tem %d caracteres\n", tam);
    return 0;
}
```


Função strlen()

```
#include <stdio.h>
#include <string.h>
int main(){
    char nome[50];
    int tam;
    printf("Digite seu nome: ");
    fgets(nome, sizeof(nome), stdin);
    printf("Olá, %s", nome);
    tam = strlen(nome) - 1;
    printf("Seu nome tem %d caracteres\n", tam);
    return 0;
}
```

Função strcpy()

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

- Copia o conteúdo da string2 para a string1
- A string str1 deve ser grande o suficiente para conter a string str2.

Função `strcat()`

```
strcat(str1, str2);
```

- Concatena duas strings, isto é, junta uma string ao final de outra
- Copia a segunda string no final da primeira e esta combinação gera uma nova primeira string
- A segunda string não é alterada

Função strcmp()

```
result = strcmp(string1, string2);
```

- Compara duas strings
- Compara a string1 com a string2 e retorna:

Valor	Significado
< 0	s1 é menor que s2
= 0	s1 é igual a s2
> 0	s1 é maior que s2

Exercícios

1) Leia um texto pela entrada padrão com no máximo 99 caracteres. Em seguida imprima o número de caracteres digitados (Dica: percorra o vetor até encontrar o caractere terminador '\0') - Não utilizar a função `strlen`!

2) Declare duas strings com capacidade para 20 caracteres. Leia pela entrada padrão a primeira string. Em seguida, copie o texto da primeira string para a segunda. Imprima no final o conteúdo das duas strings Não utilizar a função `strcpy`!