

Algoritmos e Programação II

Strings em C

Profa. Dra. Eloize Seno





Strings

- String (ou cadeia de caracteres) em C é um vetor de caracteres.

- Na linguagem C **não existe** o tipo de dados string!

- Exemplo: criação de um vetor para armazenar um único nome:

```
char nome[40]; /* reserva um espaço  
para armazenar até 40 caracteres. */
```



Strings (cont.)

- **Terminador de strings:** toda string termina com o caracter null ('\0'), que é automaticamente inserido pelo compilador.
- Deve-se declarar sempre o vetor com uma posição a mais para armazenar o caractere nulo.
- Por exemplo, para armazenar a palavra **CADEIA** deve-se declarar um vetor do tipo char com sete posições: **char string[7];**

C	A	D	E	I	A	\0
0	1	2	3	4	5	6



Strings (cont.)

- Inicialização de strings no momento da declaração:

```
char nome[] = { 'S', 't', 'r', 'i', 'n', 'g', '\\0' };  
char nome[] = "String";
```

- **Obs:** Para armazenar em uma cadeia aspas, apóstrofos ou barra invertida, é preciso usar barra invertida antes.

– **Ex:** `char string[] = "caixa d\' agua";`

Lendo e imprimindo strings com gets e puts

- **gets**: lê toda a string até que a tecla ENTER seja digitada (inclui espaços em branco).

- Exemplo:

```
gets (nome) ;
```

- **puts**: imprime toda a string até encontrar o o caracter null ('\0');

- Exemplos:

```
puts ("O nome eh: \n") ;
```

```
puts (nome) ;
```

Obs: para usar os comandos **gets** e **puts**, use a biblioteca **stdio.h**



Lendo e imprimindo strings com gets e puts (cont.)

```
#include <stdio.h>
#include <stdlib.h>
#define tam 40 // define o tamanho máximo do
               vetor nome
int main ( )
{
    char nome[tam];
    /* Entrada de dados do vetor */
    puts("Por favor, qual o seu nome?");
    gets(nome);
    puts("Eu sou um computador PC, em que posso
        ajuda-lo?\n");
    puts(nome);
    system("Pause");
}
```



Vetor de Strings

- Um **vetor de strings** em C é uma **matriz bidimensional**.
 - 1ª dimensão: número máximo de strings a serem armazenadas;
 - 2ª dimensão: número máximo de caracteres a serem armazenados em cada string.
- Cada linha da matriz será tratada como um **vetor de caracteres**.



Vetor de Strings (cont.)

- Exemplo:

- ```
- char nomes[5][40]; /* reserva um
 espaço para armazenar até 5 nomes
 contendo até 40 caracteres cada.*/
```

|   | 0 | 1 | 2 | ... |    |   |    |    |   |    |  |  | 39 |
|---|---|---|---|-----|----|---|----|----|---|----|--|--|----|
| 0 | J | O | Ã | O   | \0 |   |    |    |   |    |  |  |    |
| 1 | M | A | R | I   | A  | N | A  | \0 |   |    |  |  |    |
| 2 | C | A | R | L   | O  | S | \0 |    |   |    |  |  |    |
| 3 | A | N | A |     | M  | A | R  | I  | A | \0 |  |  |    |
| 4 | J | O | S | É   | \0 |   |    |    |   |    |  |  |    |





# Vetor de Strings

- **Leitura e Impressão:** é preciso informar apenas o identificador da variável do tipo vetor de caracteres e o índice da linha na matriz que contém ou conterá a string.



# Programa exemplo: cria e imprime o vetor de nomes

```
#include<stdlib.h>
#include<stdio.h>
int main()
{
 int i;
 char nomes[5][40];

 for (i=0; i<5; i++)
 {
 printf("Entre com o nome da linha
 %d", i);
 gets(nomes[i]);
 }
```

continua →



## Programa exemplo: cria e imprime o vetor de nomes

```
for (i=0; i<5; i++)
{
 printf("\nO nome %d é\n", i);
 puts(nomes[i]);
}
system("Pause");
}
```



# Matriz de Strings

- Uma **matriz de strings** em C é uma **matriz tridimensional**.
  - 1ª dimensão: número de linhas da matriz;
  - 2ª dimensão: número de colunas da matriz;
  - 3ª dimensão: tamanho de cada string (número máximo de caracteres permitido).



## Matriz de Strings (cont.)

- Exemplo:
  - `char nomes[5][3][40]; /* reserva um espaço para armazenar até 15 nomes contendo até 40 caracteres cada.*/`



## Matriz de Strings (cont.)

- **Leitura e Impressão:** é preciso informar o identificador da variável do tipo matriz de strings, o índice da linha e o índice da coluna na matriz que contém ou conterá a string.



# Programa exemplo: cria e imprime a matriz de nomes

```
#include<stdlib.h>
#include<stdio.h>
int main()
{
 int i, j;
 char nomes[5][3][40];

 for (i=0; i<5; i++)
 {
 for (j=0; j<3; j++)
 {
 printf("Entre com o nome da linha
 %d coluna %d", i, j);
 gets(nomes[i][j]);
 }
 }
}
```

continua →

# Manipulando Cadeias de Caracteres

- Biblioteca **string.h**: oferece um conjunto de funções para manipulação de strings.
- Funções para atribuição (ou cópia) de valores, concatenação e comparação de strings, entre outras.





# Atribuição (ou cópia) de Cadeias de Caracteres

- O operador de atribuição **=** não funciona com strings!
- A atribuição de strings é feita pela função: **strcpy()**
- **strcpy(str1, str2);** // copia a cadeia str2 para str1  
Ex: 

```
char str1[10], str2[10];
strcpy(str1, "Programa");
strcpy(str2, str1);
```



# Concatenação de Cadeias de Caracteres

- Função: **strcat()**  
**strcat(str1, str2);** /\* concatena a cadeia str2 com a cadeia str1 \*/
- Função: **strncat()**  
**strncat(str1, str2, n);** /\* concatena os n primeiros caracteres da cadeia str2 com a cadeia str1 \*/
- Exemplo:  

```
char str1[] = "eu adoro";
char str2[] = "programar";
strcat(str1, str2); /* str1 contém a
string Eu adoro programar */
```



# Comparação de Cadeias de Caracteres

- Função: **strcmp()**  
    resultado = strcmp(str1, str2);
- A função **strcmp()** compara duas strings e retorna:
  - 0, se as *strings* forem iguais,
  - um nº menor que 0, caso str1 seja alfabeticamente menor que str2;
  - um nº maior que 0, se str1 for alfabeticamente maior que str2;



## Comparação de Cadeias de Caracteres (cont.)

- **Atenção:** a função **strcmp()** considera letras maiúsculas símbolos diferentes de letras minúsculas.
- As funções **stricmp()** e **strcmpi()** consideram letras maiúsculas e minúsculas iguais:  
    **resultado = stricmp(str1,str2);**  
    **resultado = strcmpi(str1,str2);**



## Outras Funções Úteis

- Descobrindo o número de caracteres da cadeia (comprimento):  
    resultado = **strlen(str1)**; // retorna o tamanho de str1
- Revertendo uma string:  
    **strrev(str1)**; // inverte a string armazenada em str1



## Outras Funções Úteis (cont.)

- Conversão de maiúsculas em minúsculas:  
**strlwr(str1);**
- Conversão de minúsculas em maiúsculas:  
**strupr(str1);**



## Outras Funções Úteis (cont.)

- Conversão de cadeia de caracteres em valor numérico inteiro:

**numero = atoi(cadeia);**

**Atenção:** a variável numero deve ser do tipo **int**.

- Conversão de cadeia de caracteres em valor numérico real:

**numero = strtod(cadeia);**

**Atenção:** a variável numero deve ser do tipo **float**.

**Obs:** as funções **atoi** e **strtod** exigem a inclusão da biblioteca **stdlib.h**



# Exercícios

- 1) Faça um programa que receba uma palavra e verifique se ela é um palíndromo, ou seja, se escrita do fim para o começo é igual à palavra escrita do começo para o fim. Exemplo: RENNEN, ANA, MIRIM, OVO, etc.
- 2) Faça um programa que leia um nome e imprima as 4 primeiras letras.
- 3) Faça um programa que leia 10 nomes, ordene-os de forma crescente e mostre-os na tela.
- 4) Altere o programa 1) para imprimir todos os nomes que contenham uma letra qualquer fornecida pelo usuário.





# Exercícios

- 5) Faça um programa que leia um nome e escreva o numero de letras que ele possui.
- 6) Faça um programa que leia o nome e o sobrenome de uma pessoa separadamente. O programa deve juntar as duas strings em uma só e escrever na tela: a nova string, o seu número de caracteres, a sua primeira e a última letra.
- 7) Faça um programa que leia um nome completo e o imprima de duas formas: com todas as letras em minúsculas e com todas as letras em maiúsculas.