



Laboratório de Computação I

Vetor de Caractere - String



Prof. Ivre Marjorie

Introdução

- ▶ Nessa aula vamos praticar o uso de vetor para armazenar caracteres
- ▶ Strings (cadeias de caracteres) são muito utilizadas para guardar:
 - ▶ nomes de arquivos
 - ▶ nomes de usuários
 - ▶ qualquer informação baseada em caracteres.
- ▶ A linguagem C utiliza **vetores** de **char** para armazenar uma cadeia de caracteres,
 - onde cada posição representa um caractere



Introdução

- ▶ A diferença básica entre strings e outros vetores é:
 - ▶ A linguagem de programação C indica o fim do vetor de strings através do acréscimo do caractere NULL ('\0') no final do String.
- ▶ Deve-se declarar sempre o vetor **com uma posição a mais** para armazenar o caractere nulo ('\0')
 - ▶ que **não** precisa ser armazenado manualmente, isso é feito automaticamente pelo compilador



Exemplo

- Para armazenar a palavra **CADEIA** deve-se declarar um vetor do tipo **char** com 7 posições (que ocuparão posições contíguas na memória)

char **palavra**[7]

índice	0	1	2	3	4	5	6	...
valor	C	A	D	E	I	A	\0	...
Posição Memória	863	864	865	866	867	868	869	...

- A variável **palavra**, quando é declarada pode ocupar qualquer posição na memória
- Entretanto, todas as posições do vetor ocupam espaços de memória adjacentes, sendo que cada caractere ocupa **1 byte**

Inicializando cadeias de caracteres

I- Inicialização no momento da declaração

```
char nome [ ] = {'P', 'r', 'o', 'g', 'r', 'a', 'm', 'a', '\0'};
```

- A variável **nome** recebeu as letras separadamente (inclusive o caractere nulo). Por isso, cada uma das letras está envolvida por apóstrofos (' ') – essa é a maneira de identificar um caractere isoladamente



Inicializando cadeias de caracteres

2- Inicialização no momento da declaração

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

- A variável **nome2** recebeu uma palavra, recebendo automaticamente o caractere nulo. Por isso, a palavra Programa está entre aspas (" ") – esta é a maneira de identificar uma cadeia de caracteres.



Inicializando cadeias de caracteres

3- Inicialização por meio da atribuição (*depois da declaração*)

```
char vet1[10], vet2[5];  
strcpy(vet1, "Programa");
```

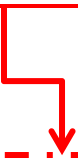
- A variável **vet1** recebeu um valor constante (a palavra Programa).



Inicializando cadeias de caracteres

4- Inicialização por meio da atribuição (*depois da declaração*)

```
char vet1[10], vet2[5];  
strcpy(vet1, vet2);
```



Usando duas strings, uma será copiada na outra. Ou seja, o conteúdo da variável **vet2** foi copiado na variável **vet1**.

Inicializando cadeias de caracteres

5- Inicialização por meio do teclado

```
char frase[100];  
  
printf("Digite um texto: ");  
  
scanf("%s", frase);
```

- O comando **scanf** consegue armazenar valores vindos do teclado na variável **frase**. No caso de uma cadeia de caracteres , esse comando consegue armazenar todos os símbolos digitados até a primeira ocorrência do **espaço em branco**.



Inicializando cadeias de caracteres

3. Inicialização por meio do teclado

```
char frase[100];  
printf("Digite um texto: ");  
gets(frase);
```

- A função **gets()** armazena na variável **frase** todos os símbolos digitados até a ocorrência do **ENTER**.
- Esta função exige a utilização da biblioteca **stdio.h**.



Imprimindo cadeias de caracteres

```
char frase[100];  
  
printf("\n Digite um texto: ");  
  
gets(frase);  
  
printf("\n A frase digitada foi: ");  
  
puts(frase);
```

- A função **puts()** é usada para imprimir uma cadeia de caracteres inicializada com o uso da função **gets()**.



Imprimindo cadeias de caracteres

```
char frase[100];  
  
printf("\n Digite um texto: ");  
  
gets(frase);  
  
printf("\n A frase digitada foi: %s", frase);
```

- Também é possível imprimir a cadeia de caracteres com o comando **printf**, nesse caso, vamos precisar usar o **%s** (string)



Funções da biblioteca string.h

- ▶ Como todas as cadeias de caracteres são variáveis compostas homogêneas (**vetor** ou **matriz**) deve-se utilizar funções específicas
- ▶ Essas funções fazem parte da biblioteca **string.h**
- ▶ Algumas delas:
 - ▶ strlen()
 - ▶ strcpy()
 - ▶ strcat()
 - ▶ strcmp()
 - ▶ strupr()
 - ▶ strlwr()



Funções da biblioteca string.h

I- Função: strlen()

```
int Tamanho;  
char str l[20];  
  
    Tamanho = strlen(str l);
```

- A função **strlen** retorna para a variável **Tamanho** o número de caracteres da cadeia **str l**.



Funções da biblioteca string.h

2- Função: strcpy()

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

- A função **strcpy** copia a cadeia **str2** na cadeia **str1**
- Sendo assim, a cadeia **str1** será substituída pela cadeia **str2**



Funções da biblioteca string.h

3- Função: strncpy()

```
strncpy (str1, str2, n);
```

- A função **strncpy** copia os **n** primeiros caracteres da cadeia **str2** para a cadeia **str1**



Funções da biblioteca string.h

4- Função: strcat()

```
strcat(cadeia1, cadeia2);
```

- A função **strcat** concatena a cadeia **cadeia2** na cadeia **cadeia1**, ou seja, **acrescenta** a cadeia **cadeia1** a cadeia **cadeia2**



Funções da biblioteca string.h

5- Função: strcmp()

Resultado = strcmp (cadeia1, cadeia2);

- Compara duas cadeias de caracteres e retorna um número inteiro para a variável **Resultado**, que pode ser:
 - ✓ **Zero:** se as duas cadeias forem iguais
 - ✓ **Um número menor que 0:** se a cadeia **cadeia1** for alfabeticamente menor que **cadeia2**
 - ✓ **Um número maior que 0:** se a cadeia **cadeia1** for alfabeticamente maior que **cadeia2**

** Essa função considera letras maiúsculas como sendo símbolos **diferentes** de letras minúsculas*

Funções da biblioteca string.h

6- Função: strcmp()

Resultado = strcmp (cadeia1, cadeia2);

- Compara duas cadeias de caracteres e retorna um número inteiro para a variável **Resultado**, que pode ser:
 - ✓ **Zero:** se as duas cadeias forem iguais
 - ✓ **Um número menor que 0:** se a cadeia **cadeia1** for alfabeticamente menor que **cadeia2**
 - ✓ **Um número maior que 0:** se a cadeia **cadeia1** for alfabeticamente maior que **cadeia2**

** Essa função considera letras maiúsculas e minúsculas como sendo símbolos iguais.*

Funções da biblioteca string.h

- 7- Função:** toupper(**caracter**) – converte o caracter em maiúsculo.
- 8- Função:**strupr(**string**) – converte a string para maiúsculo
- 9- Função:** tolower(**caracter**) – converte o caracter em minúsculo.
- 10- Função:** strlwr(**string**) – converte a string para minúsculo



Funções da biblioteca string.h

II - Função: toascii()

Valor = toascii(caractere);

- A função **toascii** retorna para a variável **Valor** o valor numérico que representa o **caractere** na tabela ASCII. Essa função exige a utilização da biblioteca `ctype.h`.



Funções da biblioteca string.h

```
caractere = int(número);
```

- A função **int** retorna para a variável **caractere** o caractere ASCII que é representado pelo **número**.





Exemplo 1

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

int main()
{
    char nome1[100], nome2[100];
    printf("Digite seu nome completo: ");
    scanf(" %s", nome1);
    printf("Digite seu nome completo: ");
    gets(nome2);
    gets(nome2); //usado duas vezes para resolver o problema de pular essa entrada de dados
    printf("\n***** Resultado com PRINTF *****");
    printf("\nNome1: %s ", nome1);
    printf("\nNome2: %s ", nome2);
    printf("\n\n***** Resultado com PUTS *****");
    printf("\n");
    puts(nome1);
    printf("\n");
    puts(nome2);
    return 0;
}
```



Exemplo 2

```
int main()
{
    char nome[40] = "Jose", sobrenome[30] = "Maria";
    strcat(nome, sobrenome);
    printf(" Sobrenome %s", sobrenome);
    printf("\n Nome %s", nome);
    return 0;
}
```





Exemplo 3

```
int main ()
{
    char nome[40] = "Jose", sobrenome[30] = "Jose";
    int teste;
    teste = strcmp (nome, sobrenome);
    if (teste != 0)
    {
        printf("As strings sao diferentes");
    }
    else
        printf("As strings sao identicas");
    return 0;
}
```





Exemplo 4

```
int main()
{
    char letra, maiuscula, minuscula;
    printf("\n\nDigite uma letra: ");
    scanf("%c",&letra);
    //toupper transforma em maiuscula
    maiuscula = toupper(letra);
    printf("\nMaiuscula: %c",maiuscula);
    //tolower transforma em minuscula
    minuscula = tolower(letra);
    printf("\nMinuscula: %c",minuscula);
    return 0;
}
```





Exercícios

- I. Faça um programa que solicite o nome do usuário, em seguida, imprimir na tela:
 - ▶ O nome do **usuário**
 - ▶ O nome do **usuário invertido**
 - ▶ A **quantidade** de caracteres digitados
 - ▶ O **número de caracteres** contidos no nome do usuário (sem considerar os espaços)
 - ▶ O **número de vogais** do nome do usuário





Exercícios

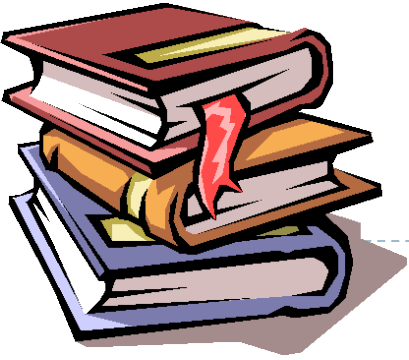
2. Faça um programa que receba uma frase, calcule e mostre a quantidade de palavras da frase digitada.
3. Faça um programa que inverta os caracteres de uma string.
4. Faça um programa que receba uma frase e uma palavra. Caso a frase contenha a palavra ESCOLA, substitua pela palavra digitada.

Exemplo:

Frase: EU MORO PERTO DE UMA ESCOLA. MAS
ESSA ESCOLA NÃO É A MELHOR.

Palavra: PADARIA

Resposta: EU MORO PERTO DE UMA **PADARIA**.
MAS ESSA **PADARIA** NÃO É A MELHOR.



Referência Bibliográfica

- ▶ MIZRAHI, Victorine Viviane. **Treinamento em linguagem C**. São Paulo: Pearson Prentice Hall, 2008. 2ª edição. Curso Completo. Capítulo 7.
- ▶ ASCENCIO, Ana Fernanda Gomes e CAMPOS, Edilene A. Veneruchi. **Fundamentos da Programação de Computadores – Algoritmos, Pascal, C/C++ e Java**. São Paulo: Pearson Prentice Hall, 2012. 3ª Edição.

