Laboratório de AED 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

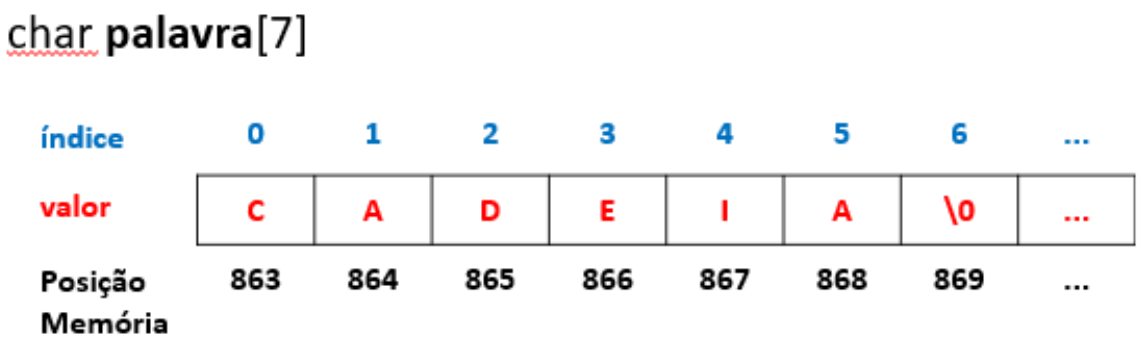
Introdução

- 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 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)



- 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**

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( )

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("\n\Minuscula: %c",minuscula);
    return 0;
}
```

Exercícios

1. Faça um programa que receba uma frase, calcule e mostre a quantidade de consoantes da frase digitada. O programa deverá contar consoantes maiúsculas e minúsculas.

2. Faça um programa que receba uma frase, calcule e mostre a quantidade de palavras da frase digitada.

3. Faça um programa para criptografar uma frase dada pelo usuário (a criptografia troca as vogais da frase por *).

Exercícios

4. Faça um programa que receba uma frase com letras minúsculas e converta somente a primeira letra de cada palavra da frase para maiúscula.

5. Faça um programa que receba uma frase e um caracter e verifique em que posição da frase o caracter digitado aparece pela última vez

Exercícios

6. Faça um programa que receba um nome e gere como saída o nome digitado e seu login. Lembre-se de respeitar as letras maiúsculas e minúsculas, já que o login será sempre com letras minúsculas. A regra de geração do login é: a primeira letra do nome e, caso exista apenas um sobrenome, deve-se acrescenta-lo; caso existam dois sobrenomes, deve-se gerar a primeira letra do nome, mais a primeira letra do primeiro sobrenome, seguido do último sobrenome; caso existam três ou mais sobrenomes, deve-se proceder como na situação anterior, considerando o nome, o primeiro sobrenome e o último sobrenome.

Exemplos:
Nome: Pedro Hansdorf
Login: phansdorf

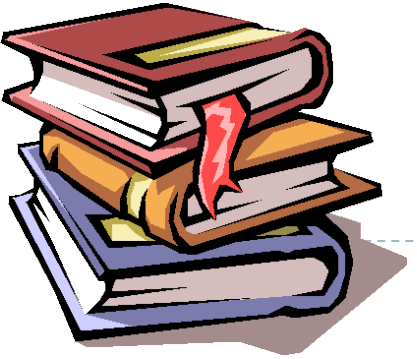
Nome: Robson Soares Silva
Login: rssilva

Nome: Jaqueline Oliveira Fernandes Espanhola
Login: joespanhola

Exercícios

7. Implemente uma função que receba como entrada uma *string* e indique se essa *string* corresponde a um palíndromo. Palíndromos são palavras, grupos de palavras ou versos em que o sentido é o mesmo, quer se leia da esquerda para a direita ou da direita para a esquerda. São exemplos de palíndromos: "reviver", "radar", "arara", "Acaiaca", "A grama é amarga", "Ame o poema", "Socorram-me em Marrocos", etc.

8. Dada um *string* ***S*** e um caractere ***C***, escreva um método recursivo que responda verdadeiro ou falso para a questão: o caractere ***C*** existe no *string* ***S***? Implemente também um programa para testar seu método recursivo. Dica: transforme o *string* ***S*** em um vetor de caracteres antes de invocar o método recursivo.



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; CAMPOS, Edilene Aparecida Veneruchi de. Fundamentos da programação de computadores. São Paulo: Pearson, 2012. ISBN 978856457416 8.