

UNIVERSIDADE DE BRASÍLIA

Introdução à Ciência da Computação

Disciplina: 113913



Prof: Luiz Augusto F. Laranjeira
luiz.laranjeira@gmail.com

Universidade de Brasília – UnB
Campus Gama



13. STRINGS



3. Variável String

- vetor do tipo *char* terminada pelo caractere `'\0'` (NULL)
- cada caractere de um string pode ser acessado individualmente
- vetor de tamanho $n \rightarrow$ string de tamanho $(n-1)$

Ex:

```
char mystring[10] = "exemplo";  
char mystring[10] = { "exemplo" };  
char mystring[10] = { 'e', 'x', 'e', 'm', 'p', 'l', 'o', '\0' };
```

```
printf ( "%s", mystring );  
printf ( "%c", mystring [ 0 ] );
```



3. Variável String

- Lendo uma string:
- **scanf** : lê o string até que um branco seja encontrado

Ex:

```
main ( )  
{
```

```
    char nome[40];
```

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

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

```
    printf ( "Bom dia %c", nome[0] );
```

```
}
```

Saída:

Digite seu nome: Jose Maria
Bom dia Jose



3. Variável String

- Lendo uma string:
- **scanf** : para ler uma string sem parar nos espaços usa-se

```
scanf("%[^\\n]s",string1);
```

Ex:

```
main ( )  
{
```

```
    char nome[40];
```

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

```
    scanf ( "%[^\\n]s", &nome[0] );    // scanf ( "%[^\\n]s", nome );
```

```
    printf ( "Bom dia %c", nome[0] );
```

```
}
```

Saída:

Digite seu nome: Jose Maria
Bom dia Jose Maria



3. Variável String

- Lendo uma string:
- **gets**
 - ✓ lê caracteres até encontrar '\n'
 - ✓ substitui '\n' por '\0'

Ex:

```
main ( )
```

```
{
```

```
    char nome[40];
```

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

```
    gets ( &nome[0] );    // ou gets(nome);
```

```
    printf ( "Bom dia %s", nome );
```

```
}
```

Saída:

Digite seu nome: Jose Maria
Bom dia Jose Maria





3. Variável String

- Imprimindo uma string:

✓ **printf**

✓ **puts**

Ex:

```
main ( )  
{
```

```
    char nome[40];
```

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

```
    gets ( &nome[0] );      // gets(nome);
```

```
    puts ( "Bom dia " );
```

```
    puts ( nome );
```

```
}
```

Saída:

Digite seu nome: Jose Maria

Bom dia

Jose Maria



3. Variável String

- Manipulando uma string:

✓ **strlen**

retorna o tamanho do string - não conta '\0'

Ex:

```
main ( )  
{
```

```
    char nome[40];
```

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

```
    gets ( &nome[ 0 ] );
```

```
    printf ("Tamanho (nome) = %d", strlen(&nome[ 0 ] ) );
```

```
}
```

Saída:

Digite seu nome: Jose Maria

Tamanho (nome) = 10




3. Variável String

- Manipulando uma string:

✓ **strcat (str1, str2)**

concatena str2 ao final de str1



Ex:
main ()
{

```
char    nome[40] = "Jose",  
char sobrenome[30] = "Maria";
```

```
strcat(nome, sobrenome);  
puts (sobrenome);  
puts (nome);
```

}

Saída:
Maria
JoseMaria

Cuidado:
str1 + str2 tem que
caber em str1



3. Variável String

- Manipulando uma string:

- ✓ **strcmp (str1, str2)**

- ✓ compara dois strings retornando:

- negativo se $str1 < str2$
- 0 se $str1 = str2$
- positivo se $str1 > str2$

- ✓ a comparação é feita por ordem alfabética





3. Variável String

- Manipulando uma string:

```
#include <stdio.h>
#include <string.h>
main ( )
{
    char    nome[40] = "Jose";
    char    sobrenome[30] = "Maria";

    if ( strcmp ( nome, sobrenome ) != 0)
        puts ( "os strings são diferentes" );
    else
        puts ( "os strings são identicos" );
}
```



A Biblioteca <string.h>

- A biblioteca <string.h> provê funções para facilitar o tratamento de strings.
- Algumas das funções da biblioteca <string.h>:

FUNÇÃO	DESCRIÇÃO
int strlen (char *s)	Retorna um inteiro com o tamanho da string s.
char* strstr (char *palheiro, char *agulha)	Essa função procura por *agulha em *palheiro, e retorna um ponteiro para a posição onde *agulha ocorre, ou NULL caso isso não aconteça.
int strcmp (char *s1, char *s2)	Retorna um número negativo (<0) se s1 for lexicograficamente menor do que s2, zero se forem iguais, ou um número positivo (>0) se s1 for lexicograficamente maior do que s1.

A Biblioteca <string.h>

- Mais funções da biblioteca <string.h>:

FUNÇÃO	DESCRIÇÃO
int strcasecmp (char *s1, char *s2):	Possui um funcionamento similar ao da strcmp, mas ela ignora as diferenças entre maiúsculas e minúsculas.
char* strcat (char *dest, char *fonte):	Concatena *fonte ao final de *dest, sendo que *dest precisa ter tamanho suficiente para essa operação. O retorno da função é um ponteiro para *dest.
char* strcpy (char *dest, char *fonte):	Copia a string em *fonte para a string em *dest, sendo que *dest deve ser grande o bastante para comportar *fonte.
char* strchr (char *s, char c):	Retorna um ponteiro para a primeira ocorrência do caractere c em s, ou NULL caso não haja.

Vetores de strings

- Vetores de strings são matrizes bidimensionais de caracteres.
- Imagine uma string. Ela é um vetor. Se fizermos um vetor de strings estaremos fazendo uma lista de vetores.
- Esta estrutura é uma matriz bidimensional de caracteres (char).
- Veja como definir esta estrutura no próximo slide.

Vetores de strings

- Definição de uma matriz bidimensional de caracteres (“vetor de strings”) em C:

```
char <nome> [<nro_strings>] [<comprimento_cada_string>];
```

- Aí surge a pergunta: como acessar uma string individual? É simples. Deve-se usar apenas o primeiro índice. Então, para acessar uma determinada string faça:

```
nome_da_variável [índice]
```

Vetores de strings – Exemplo 1

Exemplo:

Programa em C que lê 5 strings e as exibe na tela.

```
#include <stdio.h>

int main () {
    int i;
    char nome[5][30];

    for (i=0; i<5; i++) {
        printf ("\nDigite uma string: ");
        gets (nome[i]);
    }
    printf ("\nAs strings que voce digitou foram:\n\n");
    for (i=0; i<5; i++)
        printf ("%s\n", nome[i]);
    return 0;
}
```



3. Variável String – Exemplo 2

Escreva um programa que leia 5 nomes dados pelo usuário e, em seguida, os escreva em ordem alfabética.



Exemplo 2 - Solução

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

main()
{
    // Declarações de variáveis
    int i, j, k;
    char nomes[5][50];

    // Lendo os nomes
    for (i=0; i < 5; i++) {
        printf("Escreva o nome [%d]: ", i);
        gets(nomes[i]);
    }

    // Dando espaço para clareza na tela
    printf("\n\n\n");
```

```
    // Escrevendo os nomes em ordem alfabética
    for (k=0; k < 5; k++) {
        // Achando a 1a string não nula
        j = 0;
        while (strlen(nomes[j]) == 0) j++;

        // Localizando a string de ordem mais baixa
        for (i=j; i < 5; i++) {
            if (strlen(nomes[i]) > 0 &&
                strcmp (nomes[i], nomes[j]) <= 0) {
                j = i;
            }
        }

        // Escrevendo um nome na tela
        puts(nomes[j]);
        // Zerando a string do nome já escrito na tela
        nomes[j][0] = '\0';
    }
    system("pause");
    return(0);
} // main
```


Exercícios

- **ATENÇÃO:** Estes exercícios deverão ser feitos SEM a utilização das funções de <string.h> (exceto strlen).

1. Faça um programa em C que leia duas strings e concatene-as em uma terceira string, mostrando o resultado na tela. Você pode definir o tamanho das strings, lembrando que o usuário poderá informar strings de tamanhos diferentes.

2. Uma prova de 10 questões com cinco alternativas (A, B, C, D e E) de múltipla escolha foi aplicada em uma turma. Faça um programa que leia o gabarito, o número de alunos e as respostas de cada aluno, e diga a maior nota obtida.

EXEMPLO DE ENTRADA:

Gabarito: ABCDEABCDE

Nro alunos: 3

Respostas aluno 1: ABCAABAAAA

Respostas aluno 2: ABCDABAAAA

Respostas aluno 3: ABCDEABCDD

EXEMPLO DE SAÍDA:

Maior nota obtida: 9

Exercícios

- **ATENÇÃO:** Estes exercícios deverão ser feitos SEM a utilização das funções de <string.h> (exceto strlen).

3. Uma forma rudimentar de compactação pode ser a representação da multiplicidade de caracteres repetidos. Escreva um programa que leia uma cadeia de caracteres e para cada um deles que possua adjacentes iguais a ele, imprima o número de repetições e apenas uma cópia dele, e imprima apenas o caractere sem o número, caso naquela ocorrência não haja repetições.

Exemplo de entrada: AAABCCCCCBBBCCAAA

Exemplo de saída: 3AB5C3B2C3A

4. Faça um programa que leia, via teclado, duas strings formadas apenas por letras e espaços brancos. O programa deve imprimir uma lista das palavras que aparecem simultaneamente nas duas strings. Pode-se supor que em cada uma das strings não há palavras repetidas.

Exercícios

- ATENÇÃO: Estes exercícios deverão ser feitos SEM a utilização das funções de <string.h> (exceto strlen).

5. Crie um programa que leia o nome de uma pessoa (20 caracteres) e o escreva de trás para frente. Você pode usar a função strlen para descobrir o tamanho do nome digitado.

6. Faça programa para copiar n caracteres da string s1 na string s2, começando pela posição i da string s1. O usuário deve informar s1, n, e i.

7. Faça um programa para remover n caracteres de uma string s, a partir da posição i. O usuário deve informar s, n, e i.

8. Faça um programa que procura uma palavra (*pvelha*) em uma frase (*fr*) e a substitui por outra palavra (*pnova*). Supõe-se que a frase seja formada apenas por palavras e espaços brancos, podendo haver qualquer quantidade de espaços em branco entre as palavras.

Exercícios

- **ATENÇÃO:** Estes exercícios deverão ser feitos **SEM** a utilização das funções de `<string.h>` (exceto `strlen`).

9. Faça um programa que leia, via teclado, uma string formada apenas por letras e espaços brancos. O programa deve gerar outra string, formada por essas palavras, na mesma ordem, mas cada uma delas escrita de maneira invertida.

Exemplo:

Entrada: abcd ghi rstu vwxyz

Saída: dcba ihg utsr zyxwv

Obs: Não se esqueça de utilizar o comando `scanf("%[^\n]s%c",string)` ou o comando `gets(string)` para permitir a leitura de espaços em branco.

Desafios

1. Escreva um programa para descompactar a sequência compactada gerada pelo programa do exercício 3.

2. Um erro comum de digitação é colocar as mãos no teclado uma coluna à direita da posição correta. Com isso o 'Q' será digitado como 'W', o 'J' como 'K' e assim por diante. Escreva um programa que decodifique uma mensagem digitada dessa maneira. A entrada do seu programa conterá uma linha escrita dessa forma, e ela pode conter caracteres maiúsculos, espaços, números e sinais de pontuação. Seu programa deve fazer a correção em função do layout de teclado abaixo. Espaços em branco na entrada devem ser reproduzidos na saída.

Exemplo de entrada: **O S, GOMR YPFSU/**

Exemplo de saída: **I AM FINE TODAY.**

