


Strings

1

- Vetor de char terminada por null `'\0'`
- O caracter `'\0'` tem valor decimal 0
- `char endereco[50];`
49 caracteres + o caracter null
- `scanf ("%s", endereco);`
inclui `\0` quando o espaço é incluído
- `endereco == &endereco[0]`
nome do vetor é o endereço inicial

www.gomeshp.com




Strings

2

- Inicialização:
`static char nome[] = { 'A', 'n', 'a', '\0' };`
`static char nome[] = { "Ana" };`
`static char nome[5];`
- *A inicialização é importante para que a string não contenha "lixo de memória".*

www.gomeshp.com




Strings

3

- **Leitura**
 - `scanf ("%s", frase);`
finaliza com espaço => uma palavra
 - `gets (frase);`
finaliza com enter => uma frase
- **Saída**
 - `printf ("%s", frase);` => não salta linha
 - `puts (frase);` => salta linha
 - `puts (&frase[3]);` => mostra a partir do 4o.

www.gomeshp.com



Funções p/ Strings


4

- **strlen (endereço);**
retorna o tamanho da string armazenada a partir do endereço até um caracter antes do null.

```

#include <stdio.h>
#include <conio.h>
#include <string.h>
void main( ) {
    static char frase [20] = "Tamanho da string";
    printf ("%d\n", strlen ( frase ));           // 17
    printf ("%d\n", strlen ( &frase[4] ));       // 13
    getch ( );
}
    
```

www.gomeshp.com



Funções p/ Strings

5


- **strcpy (string1, string2);**
copia a *string2* para a *string1*

- não é realizado teste do tamanho da *string1*
- a *string2* não é alterada

```

// incluir bibliotecas
void main( ) {
    static char s1[20], s2[20];
    gets (s2);
    strcpy (s1, s2);
    puts (s1);
    getch( );
}
    
```

www.gomeshp.com



Funções p/ Strings

6


- **strcat (string1, string2);**
concatena a *string2* ao final da *string1*

- não é realizado teste do tamanho da *string1*
- a *string2* não é alterada

```

// incluir bibliotecas
void main( ) {
    static char frase [20];
    static char palavra[10];
    gets (frase);
    scanf ("%s", palavra);
    strcat (frase, palavra);
    puts (frase);
    getch( ); }
    
```

www.gomeshp.com



Funções p/ Strings

7

- **strcmp (string1, string2);**
compara a *string1* com a *string2*;


$< 0 \Rightarrow str1 < str2$
 $0 \Rightarrow str1 == str2$
 $> 0 \Rightarrow str1 > str2$

```

if ( senha == entrada )
    // endereços de memória sempre diferentes
    // &senha[0] == &entrada[0]

if ( strcmp (senha, entrada) == 0 )
    
```

www.gomeshp.com



Funções p/ Strings


8

- **strcmp (string1, string2);**
compara a *string1* com a *string2*;

```

// incluir bibliotecas
void main( ) {
    char nome1[20], nome2[20];
    gets (nome1);
    gets (nome2);
    if ( strcmp (nome1, nome2) < 0 )
        { puts (nome1); puts (nome2); }
    else { puts (nome2); puts (nome1); }
    getch ( );
}
    
```

www.gomeshp.com



Funções p/ Strings


9

- **strlwr (string);**
transforma toda a *string* para letras minúsculas;

```

// incluir bibliotecas
void main( ) {
    char frase[20];
    gets (frase);           // Linguagem C
    strlwr (frase);
    puts (frase);           // linguagem c
    getch ( );
}
    
```

www.gomeshp.com



Funções p/ Strings


10

- **strupr (string);**
transforma toda a *string* para letras maiúsculas;

```

// incluir bibliotecas
void main( ) {
    char frase[20];
    gets (frase);      // Estudo de Strings
    strupr (frase);
    puts (frase);      // ESTUDO DE STRINGS
    getch ( );
}
    
```

www.gomeshp.com



Funções p/ Strings


11

- **strrev (string);**
inverte todos os caracteres da *string*, exceto o caracter \0 (null);

```

// incluir bibliotecas
void main( ) {
    char frase[20];
    gets (frase);      // Funções sobre Strings
    strrev (frase);
    puts (frase);      // sgnirtS erbos seõçnuF
    getch ( );
}
    
```

www.gomeshp.com




Programas

12

```

// incluir bibliotecas
void main( ) {
    char frase[45];
    int x, qtda = 0;
    clrscr ( );
    printf ( "Digite uma frase: " );    gets (frase);
    for ( x = 0; x < strlen (frase); x++ )
        if ( toupper ( frase[x] ) == 'A' )    qtda++;
    printf ( "\nA frase possui %d A's.", qtda );
    getch ( );
}
    
```

www.gomeshp.com



Programas


13

```

// incluir bibliotecas
void main ( ) {
    static char password [5] = {"Amor"};
    char senha [5];
    int x = 1;
    clrscr ( );
    do {
        printf ( "\n %da. Digite a senha: ", x );
        gets ( senha );
    } while ( x++ < 3 && strcmp ( senha, password ) );
    if ( strcmp ( senha, password ) == 0 )
        printf ( "\n\n Senha correta.\n" );
    else printf ( "\n\n Senhas incorretas.\n" );
    getch ( );
}

```

www.gomeshp.com



Matriz de Strings

14


- Declaração:


```
static char nomes [4][40];
```
- Inicialização:


```
static char nomes [4][40] =
{ "Ana Maria", "Beatriz", "Cláudio", "Dênia" } ;
```
- Leitura dos quatro nomes:


```
for ( x = 0; x < 4; x++ )
    gets ( nomes [x] );    // gets ( &nomes [x][0] );
```

www.gomeshp.com




Exercícios

15

1. Ler o nome (máximo 30 caracteres) e o endereço (máximo 50 caracteres) do aluno. Mostrar os dados de entrada na mesma linha separados por hífen.
2. Ler dois nomes com no máximo 20 caracteres cada e mostrá-los em ordem alfabética.
3. Ler duas palavras com no máximo 10 caracteres cada e mostrar se são iguais ou diferentes.
4. Ler o primeiro nome e o último nome de uma pessoa com no máximo 15 caracteres cada. Criar uma 3a. string (máximo 30 caracteres) que contém o primeiro e o último nome, separados por espaços. Mostrar esta string.
5. Ler uma frase com no máximo 30 caracteres e mostrar a quantidade de caracteres e mostrar a frase em letras maiúsculas.
6. Ler uma frase com no máximo 20 caracteres e copiar para outra string (cópia). Mostrar o conteúdo da string copia e depois em letras maiúsculas.

www.gomeshp.com



Exercícios

16

7. Ler uma palavra com no máximo 10 caracteres e mostrá-la na ordem inversa.

8. Ler uma frase (máximo 45 caracteres) e mostrar a quantidade de a's (maiúsculos e minúsculos).


9. Ler o nome de uma pessoa com no máximo 50 caracteres e mostrar o primeiro nome e a primeira letra.

10. Definir uma senha para seu programa com 5 caracteres. Pedir ao usuário que adivinhe sua senha (ler a senha). Permitir três chances. Assim, que o usuário acertar, mostrar mensagem de acerto. Se não acertar, mostrar mensagem de erro. Assim que acertar, o programa deve ser finalizado.

11. Ler o nome de três estados com no máximo 15 caracteres cada e mostrá-los em ordem alfabética.

12. Ler uma frase e mostrar a primeira palavra em letras maiúsculas e a última palavra em letras minúsculas.

www.gomeshp.com



Exercícios

17

13. Fazer um programa para declarar e inicializar uma matriz com o nome de cinco amigos. Mostrar os nomes na tela.

14. Elaborar um programa para armazenar a naturalidade de oito pessoas fornecidas como entrada. Mostrar a quantidade de pessoas nascidas em Brasília.

15. Criar um programa para ler os dados de entrada de seis pessoas: o bairro onde mora (matriz *Bmora*) e o bairro onde trabalha (matriz *Btrab*) - (no máximo 30 caracteres cada); e mostrar quantas pessoas moram e trabalham no mesmo bairro.

www.gomeshp.com
