

Cadeias de Caracteres

Professores(as):

Virgínia Fernandes Mota

João Eduardo Montandon de Araujo Filho

Leandro Maia Silva



- Uma cadeia de caracteres é uma sequência de caracteres justapostos e são fundamentais no desenvolvimento de programas computacionais.
- Exemplos de cadeias de caracteres (representadas internamente num programa):
 - Mensagem de e-mail;
 - Texto de um programa;
 - Nome e endereço em cadastro de clientes, alunos, etc...
 - Sequência genética. Um gene (ou o DNA de algum organismo) é composto de sequências dos caracteres A, T, G e C (nucleotídeos)

- Uma variável usada para armazenar um caractere é representada da seguinte maneira:
`char c;`
`c = 'a';`
- Podemos usar as funções **printf()** e **scanf()** (usando `%c`). E ainda as funções **getchar()** e **putchar()**.
- Se em uma variável do tipo `char` podemos armazenar somente um caractere, então para armazenar vários caracteres (ex: `jose`, `carro`) é necessário utilizar as cadeias de caracteres, representadas por vetores do tipo `caractere`.

Cadeia de caracteres

- Observe a declaração abaixo:
`char cidade[15];`
- A variável `cidade` é um vetor de caracteres (cadeia de caracteres).
- A variável `cidade` pode armazenar qualquer cadeia de até 14 caracteres.

0 1 2 3 4 5 6 7 8 9 10 11 12 13

B	E	L	O		H	O	R	I	Z	O	N	T	E
---	---	---	---	--	---	---	---	---	---	---	---	---	---

- 14 caracteres? Não são 15?

Cadeia de caracteres: Declaração

- Sintaxe para declaração de cadeia de caracteres:
char identificador[qtde de caracteres];
- Exemplo:
char nome[30];
char profissao[20];
- E como eu posso trabalhar com esse tipo de vetor?

Cadeia de caracteres: Manipulação

- Vamos atribuir à variável `nome`, criada anteriormente, o nome Jose.

```
nome = "Jose";
```

- Podemos ainda, obter um caracter qualquer da cadeia de caracteres da seguinte maneira:

```
char letra = nome[1]; // letra receberá o
```

- *Strings* são tipos especiais de cadeias de caractere em C e são terminadas, obrigatoriamente, pelo caractere nulo: `'\0'` (zero). Portanto, deve-se reservar uma posição para este caractere de fim de cadeia.
- Para ilustrar a declaração e a inicialização de *strings*, consideremos as seguintes declarações:

```
char s1[] = " "; //2 aspas duplas sem espaços entre elas  
char s2[] = "Belo Horizonte";  
char s3[81];  
char s4[81] = "Belo";
```

 - s1 armazena uma string vazia. Tem um único elemento: `'\0'` ;
 - s2 representa um vetor com 15 elementos (caracteres);
 - s3 representa uma cadeia de caracteres com até 80 caracteres e não é inicializada;
 - s4 também é dimensionada para conter até 80 caracteres e é inicializada com a cadeia Belo.

```
1 #include <stdio.h>
2
3 #define SUCESSO 0
4 #define TAMANHO_MAXIMO_STRING (19 + 1)
5
6 int main(int argc, char ** argv){
7     char s[TAMANHO_MAXIMO_STRING];
8     printf("Digite uma string: ");
9     scanf("%s", s);
10    printf("String digitada: %s", s);
11    return SUCESSO;
12 }
```

- Neste caso, a leitura será feita até encontrar um caractere branco: espaço (' '), tabulação ('\t') ou nova linha ('\n'). Assim, se digitarmos "Belo Horizonte", s conterá apenas "Belo".
- Não é necessário o & antes da variável s em scanf! Por quê?


```
1 #include <stdio.h>
2
3 #define SUCESSO 0
4 #define TAMANHO_MAXIMO_STRING    (19 + 1)
5
6 int main(int argc, char ** argv){
7     char s[TAMANHO_MAXIMO_STRING];
8     printf("Digite uma string: ");
9     gets(s); // Jeito ruim, professor triste :-(
10    fgets(s, TAMANHO_MAXIMO_STRING, stdin); // Agora sim \o/
11    puts(s);
12    return SUCESSO;
13 }
```

- Neste caso, se digitarmos Belo Horizonte , s conterá Belo Horizonte;
- gets(s): lê a string s a partir do teclado; MEGA PERIGOSA!
- fgets(s, tamanhoMaximo, stdin): Mesmo que gets, mas segura;
- puts(s): imprime uma string na tela seguida de nova linha;
- fputs(s, stdout): como o puts, mas não adiciona uma nova linha no fim.

- Exemplo: o programa a seguir imprime uma cadeia de caracteres, caractere por caractere:

```
1 #include <stdio.h>
2
3 #define SUCESSO 0
4 #define TAMANHO_MAXIMO_STRING (19 + 1)
5
6 int main(int argc, char ** argv){
7     char s[TAMANHO_MAXIMO_STRING];
8     int i;
9     printf("Digite uma string: ");
10    fgets(s, TAMANHO_MAXIMO_STRING, stdin);
11    for(i = 0; s[i] != '\0'; i++) {
12        printf("%c", s[i]);
13    }
14    return SUCESSO;
15 }
```

- O **for** acima equivale a `printf("%s",s);`

- Exemplo: o programa a seguir calcula e imprime o comprimento (número de caracteres) de uma cadeia:

```
1 #include <stdio.h>
2
3 #define SUCESSO 0
4 #define TAMANHO_MAXIMO_STRING (19 + 1)
5
6 int main(int argc, char ** argv){
7     char s[TAMANHO_MAXIMO_STRING];
8     int i, n = 0;
9     printf("Digite uma string: ");
10    fgets(s, TAMANHO_MAXIMO_STRING, stdin);
11    for(i = 0; s[i] != '\0'; i++) {
12        n++;
13    }
14    printf("\n O tamanho de %s eh: %d", s, n);
15    return SUCESSO;
16 }
```

- Exemplo: o programa a seguir faz uma cópia de uma cadeia, fornecida pelo usuário, para outra:

```
1 #include <stdio.h>
2
3 #define SUCESSO 0
4 #define TAMANHO_MAXIMO_STRING (19 + 1)
5
6 int main(int argc, char ** argv){
7     char destino[TAMANHO_MAXIMO_STRING],
8         origem[TAMANHO_MAXIMO_STRING];
9     int i;
10    printf("Digite uma string: ");
11    fgets(s, TAMANHO_MAXIMO_STRING, stdin);
12    for(i = 0; origem[i] != '\0'; i++)
13        destino[i] = origem[i];
14    destino[i] = '\0';
15    puts(destino);
16    return SUCESSO;
17 }
```

Funções para manipulação de *Strings*

- Existem várias funções em C para manipulação de strings. Essas funções estão declaradas no arquivo **string.h**. Entre elas pode-se destacar:
 - `strcpy(char destino[], char origem[])`: copia a string origem na string destino.
 - `strlen(char str[])`: retorna o tamanho da string str.
 - `strcat(char destino[], char origem[])`: Faz concatenação (junção) da string origem com a string destino. O resultado é armazenado na string destino.

- Criar um programa que receba como entrada uma string, seu tamanho (tam) e um caractere (procurado). A função deverá retornar a quantidade de vezes que o caractere procurado foi encontrado na string.

Strings: Exemplos

```
1 #include <stdio.h>
2
3 #define SUCESSO 0
4 #define TAMANHO_MAXIMO_STRING (19 + 1)
5
6 int main(int argc, char ** argv){
7     int tamanho, encontrados = 0, i = 0;
8     char s[TAMANHO_MAXIMO_STRING], procura_se;
9     printf("Digite sua string: ");
10    gets(s);
11    printf("Digite o caractere que deseja procurar: ");
12    scanf("%c", &procura_se);
13
14    tamanho = strlen(s);
15    while (i < tamanho){
16        if (s[i] == procura_se)
17            encontrados++;
18        i++;
19    }
20    printf("\n Foram encontrados %d caracteres %c na string %s",
21           encontrados, procura_se, s);
22    return SUCESSO;
23 }
```

- Criar um programa para verificar se a string *s2* está contida na string *s1*.

Ex: Se *s1* fosse *Ana Maria Silva* e *s2* fosse *Maria*, a função retornaria 1, pois *s2* está contido em *s1*.

Strings: Exemplos

```
1 #include <stdio.h>
2 #include <string.h>
3
4 #define SUCESSO 0
5 #define TAMANHO_MAXIMO_STRING (19 + 1)
6
7 int main(int argc, char ** argv){
8     char s1[TAMANHO_MAXIMO_STRING], s2[TAMANHO_MAXIMO_STRING];
9     int i, j, aux, tam1, tam2, estaContida;
10
11     printf("Digite sua string: ");
12     gets(s1);
13     printf("O que deseja procurar? ");
14     gets(s2);
15
16     tam1 = strlen(s1);
17     tam2 = strlen(s2);
18     estaContida = 0;
19     for(i = 0; i < tam1; i++){
20         aux=i;
21         for(j=0; j < tam2 && aux < tam1; j++){
22             if (s2[j] != s1[aux]) break;
23             aux++;
24         }
25         if (j == tam2) estaContida = 1;
26     }
27
28     if (estaContida) printf("A string foi encontrada!");
29     else printf("A string nao foi encontrada!");
30     return SUCESSO;
31 }
```

1. Fazer um programa para contar o número de espaços em brancos de uma string.
2. Fazer um programa para imprimir uma string, recebida como entrada do usuário, sem os espaços em branco.
3. Fazer um programa para contar o número de vogais numa string.
4. Escrever um programa para ler uma string (com mais de uma palavra) e faça com que a primeira letra de cada palavra fique em maiúscula. Para isso, basta subtrair 32 do elemento que deseja alterar para maiúsculo.
5. Escreva um programa que receba uma string de tamanho máximo 100, e verifique se é um palíndromo. Uma palavra é dita ser um palíndromo se a sequência de seus caracteres da esquerda para a direita é igual a sequência de seus caracteres da direita para a esquerda. Ex: arara, asa.

6. Um palíndromo, além de uma palavra, pode ser uma frase ou qualquer outra sequência de unidades que tenha a propriedade de poder ser lida tanto da direita para a esquerda como da esquerda para a direita. Em um palíndromo, normalmente são desconsiderados os sinais ortográficos, assim como o espaços entre palavras. Alguns exemplos de palíndromos são:

- Socorram-me, subi no onibus em Marrocos
- Anotaram a data da maratona
- Dammit, I'm mad!

Faça um programa que leia uma **frase** e determine se ela é um palíndromo. Use funções e/ou procedimentos.