



**TECNOLOGIA E INOVAÇÃO
EM PROL DA INDÚSTRIA**



Técnico em Informática

Programação de Aplicativos– 140h

Prof^a: Francisleide Almeida

Strings x Caractere

- Para escrever uma string usa-se aspas duplas;
- Para guardar valor em uma variável tipo caractere usa-se aspas simples;
- Na variável tipo Caractere, guarda apenas 1 byte;
- Na variável tipo String guarda mais de 1 byte

Strings

- Como em C as strings não são um tipo básico, a única forma de representar caracteres é recorrendo a um vetor de caracteres
- Uma string é um vetor de caracteres, mas nem todo vetor de caracteres é uma string;

Strings

- Na linguagem C strings são vetores de caracteres que possuem um caractere que indica o término de seu conteúdo, o caractere nulo '\0' (contra barra zero).

Declaração de Strings

- Como a string possui o caractere nulo para delimitar o final do seu conteúdo, o tamanho da string deve ser definido com um caractere a mais do que será efetivamente necessário.

Sintaxe:

`char identificador-da-string [tamanho+1];`

Declaração de Strings

- Ex:

`char vetc [6];`

- **vetc** é um vetor de caracteres (string) de tamanho 6. Pode receber uma palavra de no máximo 5 letras

Declaração de Strings

- Uma string pode ser inicializada na sua declaração com uma sequência de caracteres entre chaves e separadas por vírgula.

```
char vetc[6]= {'T', 'e', 'x', 't', 'o', '\0'};
```

- *Lembre-se que o compilador só reconhecerá um caractere se este estiver entre aspas simples, logo usar uma atribuição do tipo {t,e,x,t,o,\0} ou {texto\0} **irá gerar um erro de compilação.***

- Uma string pode também ser inicializada por uma sequência de caracteres entre aspas duplas. Neste caso, não é necessário o uso de aspas simples e vírgulas, o compilador C coloca automaticamente o '\0' no final.

```
char vetc[6] = "Texto";
```

- Assim como vetores e matrizes, na inicialização de uma string o seu tamanho pode ser omitido.

```
char vetc[ ] = "Texto";
```

```
/* vetor não-dimensionado, o compilador  
coloca automaticamente o '\0' no final */
```

Leitura de Strings

- *Utilizando a função **scanf()***
- A sintaxe para receber uma string por meio da *scanf()* é:

scanf(“%s”, nome_da_string);

OBS.: não é necessário colocar o operador &, pois o nome da string em si já é um endereço de memória.

Leitura de Strings

- **Exemplo:** Crie um aplicativo em C que peça ao usuário seu nome, armazene em uma String, peça o sobrenome, armazene em outra string e exiba o nome do usuário de maneira formal (Sobrenome, Nome).

Leitura de Strings

- **Exemplo:** Crie um aplicativo em C que peça ao usuário seu nome, armazene em uma String, peça o sobrenome, armazene em outra string e exiba o nome do usuário de maneira formal (Sobrenome, Nome).

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char nome[21], sobrenome[21];

    printf("Primeiro nome: ");
    scanf("%s", nome);

    printf("Ultimo sobrenome: ");
    scanf("%s", sobrenome);

    printf("Ola senhor %s, %s. Bem-vindo ao curso de linguagem C.\n", sobrenome, nome);

    system("pause");
}
```


- O que aconteceria se fosse digitado um nome composto no 'nome' ou no 'sobrenome'?
- A *scanf()* vai simplesmente cortar seu nome composto. Essa função pega tudo **até** encontrar um espaço em branco, caractere *new line* `\n`, tab ou ENTER.

• *Funções para Manipulação de Strings*

```

#include <stdio.h>
#include <stdlib.h>
int main ()
{
    char nome[] = "fulano";
    char sobrenome[] = "de tal";
    char nomeCompleto[] = nome + sobrenome;
    int i=0;

    while(nomeCompleto[i] != '\0'){
        printf("%c",nomeCompleto[i]);
        nomeCompleto[i++];
    }

    return(0);
    system("pause");
}
    
```

Leitura de Strings

- *Funções para Manipulação de Strings*

```
#include <stdio.h>
#include <stdlib.h>
int main ()
{
    char nome[] = "fulano";
    char sobrenome[] = "de tal";
    char nomeCompleto[] = nome + sobrenome;
    int i=0;

    while(nomeCompleto[i] != '\0'){
        printf("%c", nomeCompleto[i]);
        nomeCompleto[i++];
    }

    return (0);
    system("pause");
}
```

PORQUE O PROGRAMA NÃO COMPILA?

Um erro muito comum no uso de string em C esta sendo cometido na linha 7. char nomeCompleto[] = nome + sobrenome.

String não poder ser concatenadas utilizando o operador +. Existe uma diretiva em C que implementa diversas funções de manipulação de valores em string.

Manipulação de Strings

- **Get String (gets):**
- A função **gets()** lê os caracteres do dispositivo padrão de entrada (teclado) para o seu argumento – um vetor do tipo **char** – até que um caractere de nova linha ou o indicador de fim de arquivo seja encontrado. Um caractere **NULL** ('\0') será adicionado ao vetor quando a leitura terminar. Sua forma geral é:

gets (nome_da_string);

Manipulação de Strings

- **Get String (gets)**

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char string[100];

    printf("Digite o seu nome: ");
    gets(string);
    printf ("\n\n Ola %s\n",string);

    system("pause");
}
```

- Como o primeiro argumento da função **printf()** é uma string também é válido fazer:

`printf (string); /* isto simplesmente imprimirá a string. */`

- **Get String (gets)**

- Não é uma função segura, pois o tamanho da string não é especificado. A função *scanf()* pega tudo até aparecer o primeiro espaço em branco, e pára. Já a *gets()* não, ela pega tudo até aparecer uma *new line* \n, inclusive nada. Ou seja, se você der um ENTER, a *gets()* vai armazenar esse enter na string


```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    char nome[31], sobrenome[31], nascimento[11];
    int idade;

    printf("Nome: ");
    gets(nome);

    printf("Sobrenome: ");
    gets(sobrenome);

    printf("Idade: ");
    scanf("%d", &idade);

    printf("Data de nascimento: ");
    gets(nascimento);

    printf("\nNome completo: %s %s\n", nome, sobrenome);
    printf("Idade: %d\n", idade);
    printf("Data de nascimento: "); puts(nascimento);

    system("pause");
}
```