

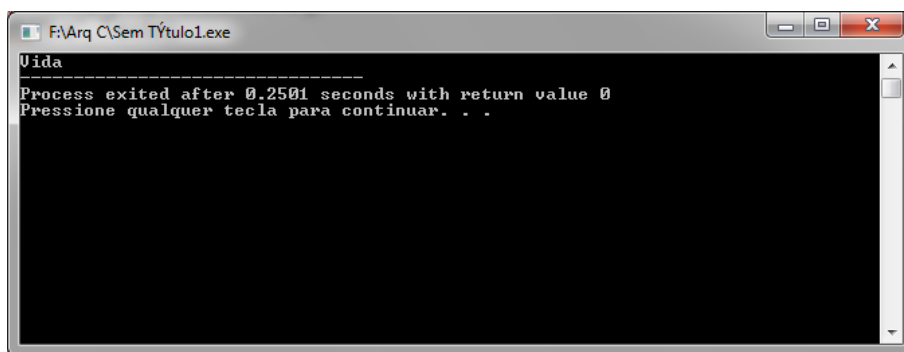
Strings

Strings na verdade são vetores de caracteres (**char**), ou seja, quando utilizamos, por exemplo, o comando **char texto[12]**; estamos declarando uma **String** chamada **texto**. Neste caso o texto armazenado pode conter no máximo 11 caracteres, pois o último elemento deve ser reservado para indicar o final da

String, para isto basta armazenar **'\0'** ao final da **String**. O fato de **'\0'** indicar o final da **String** é muito útil para programas que tratam **strings**. Agora vamos elaborar um programa onde criamos uma **String**, armazenamos a palavra “Vida” nela e depois exibimos na tela.

```
#include <stdio.h>
int main()
{
    char Texto[5];
    Texto[0] = 'V';
    Texto[1] = 'i';
    Texto[2] = 'd';
    Texto[3] = 'a';
    Texto[4] = '\0';
    printf("%s", Texto);
}
```

Executando temos:



Inicialmente criamos o **array** de caracteres chamado **Texto**, depois atribuímos, na ordem através do índice, uma letra da palavra “**Vida**” para cada um dos elementos do **array**, inserindo **'\0'** no último para indicar o final da **String**. Uma vez armazenada a **String** exibimos seu conteúdo na tela através do comando **printf("%s", Texto)**; observe que para mostrar a **String** utilizamos **“%s”**.

Neste programa atribuímos os valores caractere a caractere, quando o conteúdo da **String** já é conhecido uma forma mais simples de se atribuir valor é no momento em que criamos a **String** como vemos a seguir:

```
#include <stdio.h>
int main()
{
    char Texto[5] = {'V','i','d','a','\0'};
    printf("%s", Texto);
}
```

Este programa produz exatamente o mesmo resultado que o do programa anterior. É importante ressaltar que este tipo de atribuição você só pode fazer quando está declarando o vetor, se fizer o procedimento abaixo vai ocorrer erro na compilação

```
#include <stdio.h>

int main()
{
    char Texto[5];
Texto[5] = {'V','i','d','a','\0'};
    printf("%s", Texto);
}
```

Uma outra forma de atribuir valor, quando estamos declarando o vetor, é inserir a palavra entre aspas duplas.

```
#include <stdio.h>
int main()
{
    char Texto[5] = "Vida";
    printf("%s", Texto);
}
```

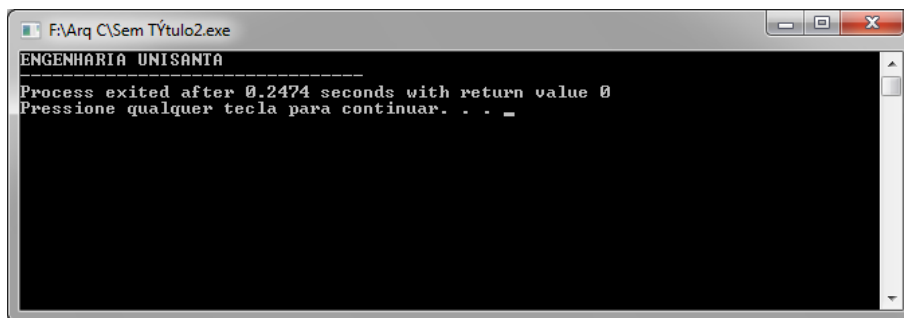
Neste caso o compilador vai se encarregar de colocar '\0' no final da **String**. Note que quando escrevemos uma **string** utilizamos o texto entre aspas duplas ("Vida") e quando estamos nos referindo a um único caractere (variável do tipo char) utilizamos as aspas simples ou apóstrofes ('V').

Exemplo:

Elabore um programa onde uma **String**, de no máximo 20 caracteres, receba o conteúdo de outra **String**.

```
#include <stdio.h>
int main()
{
    char string1[20] = "ENGENHARIA UNISANTA";
    char string2[20];
    for (int i = 0; i < 20; i++){
        string2[i] = string1[i];
    }
    printf("%s", string2);
}
```

Executando o programa temos:



Note que não podemos utilizar uma instrução do tipo **string2 = string1;** isto causaria um erro de compilação, temos que igualar elemento por elemento da **String**. Para facilitar a manipulação de Strings o C++ possui diversas funções pertencentes à biblioteca **string.h**, como veremos na sequência.

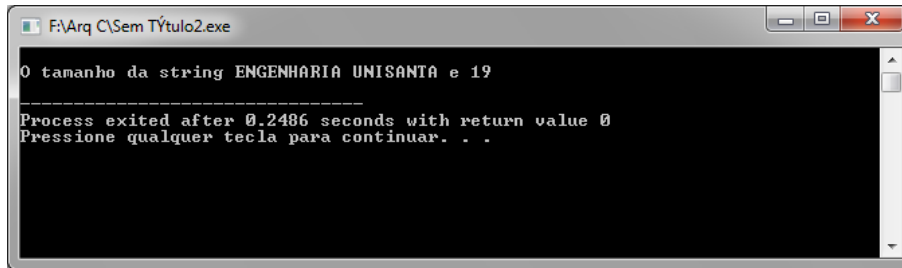
Funções da biblioteca string.h

- **strlen()**: a função retorna o comprimento da **String**.
Sintaxe: **strlen(string)**

Exemplo:

```
#include <stdio.h>
#include <string.h>
int main()
{
    char Texto[20] = "ENGENHARIA UNISANTA";
    printf("\nO tamanho da string %s e %d\n", Texto, strlen(Texto));
}
```

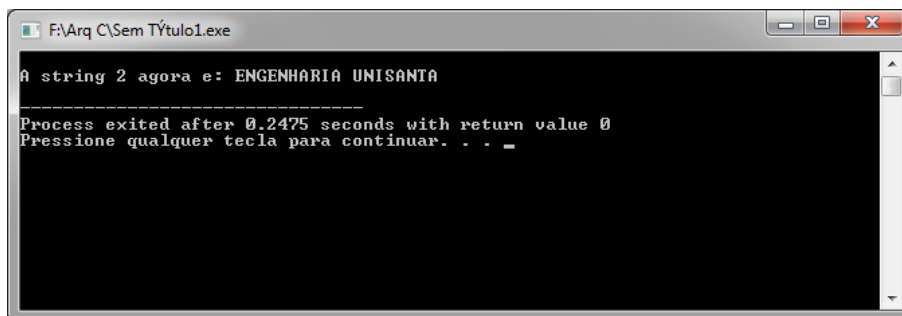
```
}
```



- **strcpy():** a função copia a string origem para a string destino.
Sintaxe: **strcpy(string destino, string origem)**

Exemplo:

```
#include <stdio.h>
#include <string.h>
int main()
{
    char string1[20] = "ENGENHARIA UNISANTA";
    char string2[20];
    strcpy(string2, string1);
    printf("\nA string 2 agora e: %s\n", string2);
}
```



- **strncpy():** Copia os primeiros **n** caracteres da strings de origem para a string destino. Se o final da string de origem (que é sinalizada por um caractere nulo) for encontrado antes que os **n** caracteres tenham sido copiados, o destino será preenchido com '\0'.
Sintaxe: **strncpy(string destino, string origem, número de caracteres)**

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

```
int main ()
{
    char str1[20]= "ENGENHARIA UNISANTA";
    char str2[20];
```

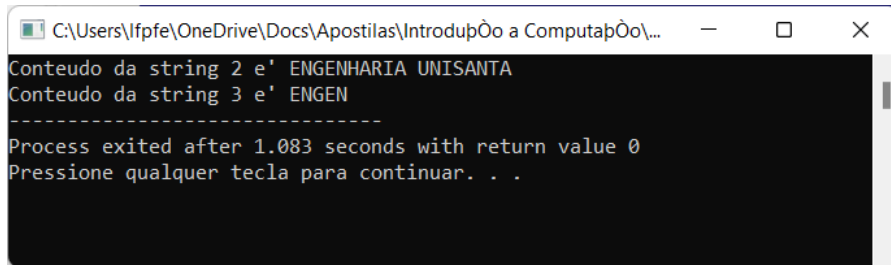
```

char str3[20];

strncpy ( str2, str1, 20 );
strncpy ( str3, str1, 5 );

printf("Conteudo da string 2 e' %s",str2);
printf("\nConteudo da string 3 e' %s",str3);
return 0;
}

```



```

C:\Users\lfpfe\OneDrive\Docs\Apostilas\Introdução a Computação\...
Conteudo da string 2 e' ENGENHARIA UNISANTA
Conteudo da string 3 e' ENGEN
-----
Process exited after 1.083 seconds with return value 0
Pressione qualquer tecla para continuar. . .

```

Agora vamos atribuir um valor inicial para a strings 3.

```

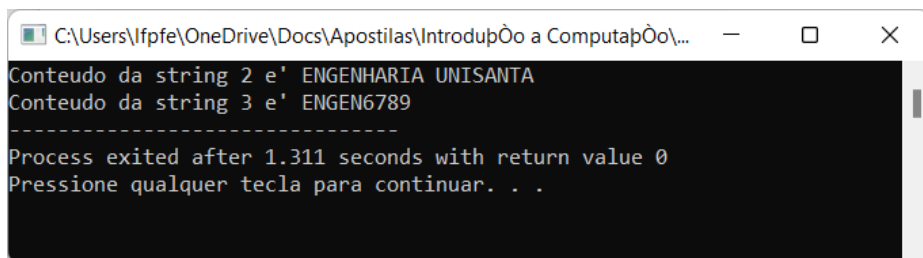
#include <stdio.h>
#include <string.h>

int main ()
{
    char str1[20]= "ENGENHARIA UNISANTA";
    char str2[20];
    char str3[20] = "123456789";

    strncpy ( str2, str1, 20 );
    strncpy ( str3, str1, 5 );

    printf("Conteudo da string 2 e' %s",str2);
    printf("\nConteudo da string 3 e' %s",str3);
    return 0;
}

```



```

C:\Users\lfpfe\OneDrive\Docs\Apostilas\Introdução a Computação\...
Conteudo da string 2 e' ENGENHARIA UNISANTA
Conteudo da string 3 e' ENGEN6789
-----
Process exited after 1.311 seconds with return value 0
Pressione qualquer tecla para continuar. . .

```

Como podemos observar os primeiros 5 caracteres foram substituídos pelo conteúdo inicial da string 1, mas como não é inserido o caractere nulo '\0' no final a strings não foi finalizada e continuou com o resto dos valores antigos.

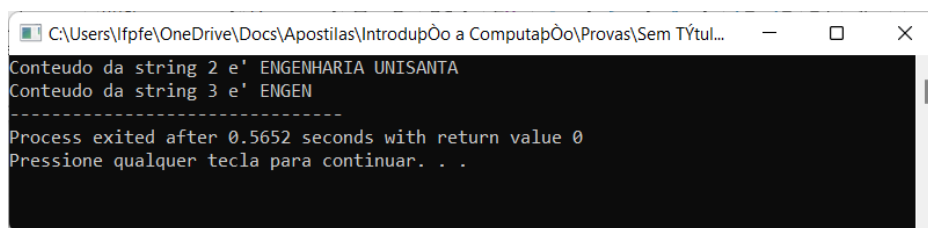
Neste caso se desejar copiar os caracteres e finalizar a string, o caractere nulo deve ser inserido manualmente.

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

int main ()
{
    char str1[20]= "ENGENHARIA UNISANTA";
    char str2[20];
    char str3[20] = "123456789";

    strncpy ( str2, str1, 20 );
    strncpy ( str3, str1, 5 );
    str3[5] = '\0';

    printf("Conteudo da string 2 e' %s",str2);
    printf("\nConteudo da string 3 e' %s",str3);
    return 0;
}
```



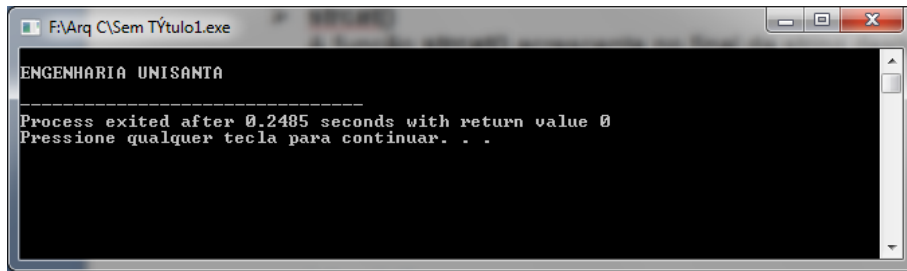
```
C:\Users\lfpfe\OneDrive\Docs\Apostilas\Introdução a Computação\Provas\Sem Títul...
Conteudo da string 2 e' ENGENHARIA UNISANTA
Conteudo da string 3 e' ENGEN
-----
Process exited after 0.5652 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

- **strcat():** a função **strcat()** acrescenta no final da string destino o conteúdo da string origem.

Sintaxe: **strcat(string destino, string origem)**

Exemplo:

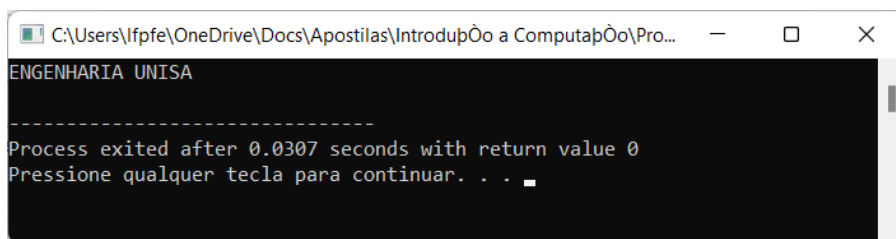
```
# include <stdio.h>
# include <string.h>
int main()
{
    char string1[20] = "ENGENHARIA ";
    char string2[20] = "UNISANTA";
    strcat(string1,string2);
    printf("\n%s\n", string1);
}
```



- **strncat():** a função **strncat()** acrescenta no final da string destino os **n** primeiros caracteres da string origem.
Sintaxe: **strncat(string destino, string origem, número de caracteres)**

Exemplo:

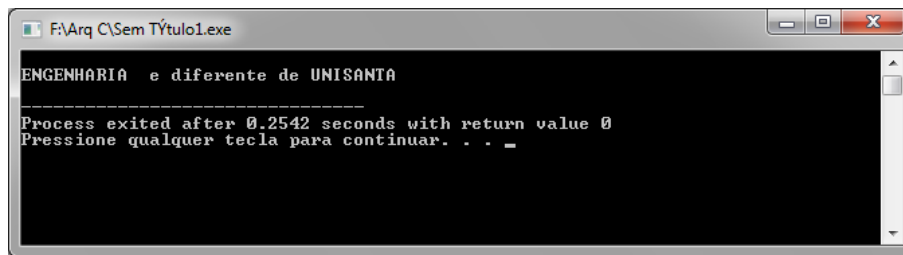
```
#include <stdio.h>
#include <string.h>
int main()
{
    char string1[20] = "ENGENHARIA ";
    char string2[20] = "UNISANTA";
    strncat(string1,string2,5);
    printf("%s\n", string1);
}
```



- **strcmp():** a função compara duas **Strings** se elas forem idênticas a função retorna 0, caso contrário retorna -1.

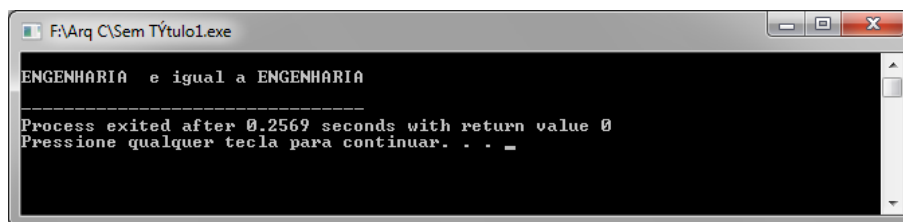
Exemplo:

```
#include <stdio.h>
#include <string.h>
int main()
{
    char string1[20] = "ENGENHARIA ";
    char string2[20] = "UNISANTA";
    if(strcmp(string1,string2))
        printf("\n%s e diferente de %s\n",string1, string2);
    else
        printf("\n%s e igual a %s\n",string1, string2);
}
```



```
F:\Arq C\Sem Título1.exe
ENGENHARIA e diferente de UNISANTA
-----
Process exited after 0.2542 seconds with return value 0
Pressione qualquer tecla para continuar. . . _
```

Substituindo no código `char string2[20] = "UNISANTA";` por `char string2[20] = "ENGENHARIA ";` temos:



```
F:\Arq C\Sem Título1.exe
ENGENHARIA e igual a ENGENHARIA
-----
Process exited after 0.2569 seconds with return value 0
Pressione qualquer tecla para continuar. . . _
```

- **gets():** esta função lê uma string do teclado.
Sintaxe: **gets(string);**

Exemplo:

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

```
int main()
```

```
{
```

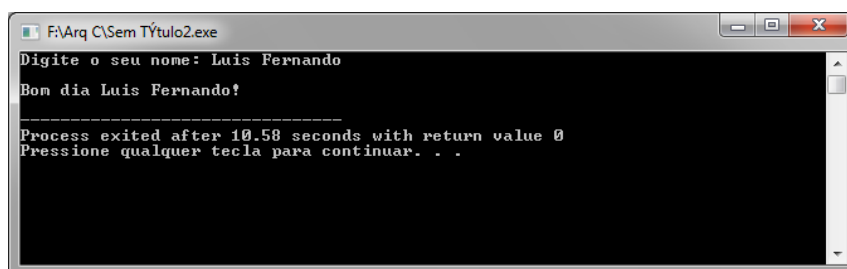
```
    char nome[50];
```

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

```
    gets(nome);
```

```
    printf("\nBom dia %s!\n",nome);
```

```
}
```



```
F:\Arq C\Sem Título2.exe
Digite o seu nome: Luis Fernando
Bon dia Luis Fernando!
-----
Process exited after 10.58 seconds with return value 0
Pressione qualquer tecla para continuar. . . _
```

Observe que não foi preciso incluir a biblioteca **string.h**, isto se deve ao fato de **gets()** ser uma função da **stdio.h**.

Funções da biblioteca ctype.h

Existe também no C++ uma biblioteca que faz o tratamento dos caracteres que é a **ctype.h**. Dentre as funções encontradas nesta biblioteca há aquelas que

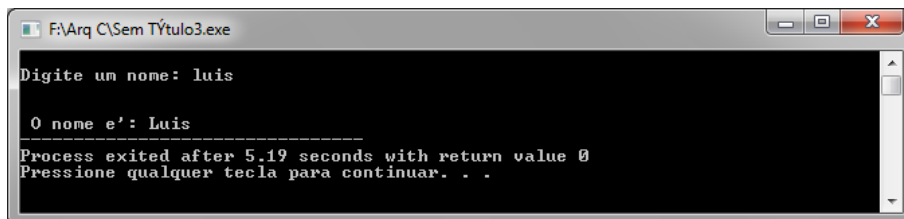
modificam o estado da letra (maiúsculas e minúsculas) e até mesmo funções que servem para descobrir se o que foi digitado é um ponto, vírgula, letra, número, espaço, ctrl, etc. A sintaxe dos comandos é bem simples basta inserir o comando desejado e entre parênteses o caractere a ser verificado ou alterado. A seguir temos uma lista com as principais funções desta biblioteca.

- **toupper** - esta função retorna o caractere inserido no formato maiúsculo, se o caractere já for maiúsculo ou não for letra ele não será modificado.
- **tolower** - esta função retorna o caractere inserido no formato minúsculo, se o caractere já for minúsculo ou não for letra ele não será modificado.
- **isalnum** - verifica se o caractere ou inteiro passado como parâmetro é alfanumérico. Isso inclui todos os números e as letras do alfabeto, tanto maiúsculas quanto minúsculas. Esta função retorna 1 quando encontra uma letra maiúscula, 2 quando for minúscula, 4 quando for um número e 0 para outros caracteres.
- **isalpha** - verifica se o caractere ou inteiro passado como parâmetro é alfabético. Isso inclui todas as letras do alfabeto, tanto maiúsculas quanto minúsculas. Esta função retorna 1 quando encontra uma letra maiúscula, 2 quando for minúscula e 0 quando não se tratar de uma letra.
- **isdigit** - verifica se o caractere ou inteiro passado como parâmetro é um dígito. Isso inclui todos os números. Esta função retorna 1 quando encontra número e 0 para outros caracteres.
- **ispunct** - verifica se o caractere ou inteiro passado como parâmetro é uma pontuação. Isso inclui qualquer tipo de pontuação como . , ? ! ^ ' { } ~ : ;. Porém, não é capaz de verificar se uma letra é acentuada. Esta função retorna 16 quando encontra símbolos e 0 para outros caracteres.
- **isspace** - verifica se o caractere ou inteiro passado como parâmetro é um espaço em branco. Esta função retorna 8 quando encontra espaços e 0 para outros caracteres.
- **islower** - verifica se o caractere ou inteiro passado como parâmetro é uma letra minúscula. Esta função retorna 2 quando encontra letras minúsculas e 0 para outros caracteres.
- **isupper** - verifica se o caractere ou inteiro passado como parâmetro é uma letra maiúscula. Esta função retorna 1 quando encontra letras maiúsculas e 0 para outros caracteres.
- **iscntrl** - verifica se o caractere ou inteiro passado como parâmetro é um caractere de comando. Isso inclui CTRL, ALT, ENTER, BACKSPACE, etc. Esta função retorna 32 quando se tratar de um caractere de controle e 0 para outros caracteres.

- **isxdigit** - verifica se o caractere ou inteiro passado como parâmetro é compatível com um número hexadecimal. Isso inclui todos os números (0 - 9) e qualquer letra entre A e F (não importa se minúsculo ou maiúsculo). Esta função retorna 1 se tratar de um número hexadecimal e 0 para outros caracteres.

Exemplo - Entre com um nome, escrito em minúsculas, e altere a primeira letra dele para maiúscula.

```
#include<stdio.h>
#include<ctype.h>
int main()
{
    char nome[30];
    printf("\nDigite um nome: ");
    gets(nome);
    nome[0] = toupper(nome[0]);
    printf("\n\n O nome e': %s",nome);
}
```



Se quisermos transformar toda a string devemos percorrer todos os caracteres dela.

```
#include <stdio.h>
#include <ctype.h>
int main()
{
    char string1[20] = "Engenharia";
    int i = 0;
    char aux = ' ';
    do{
        aux = string1[i];
        string1[i] = toupper(string1[i]);
        i++;
    } while (aux != '\0');
    printf("%s",string1);
}
```