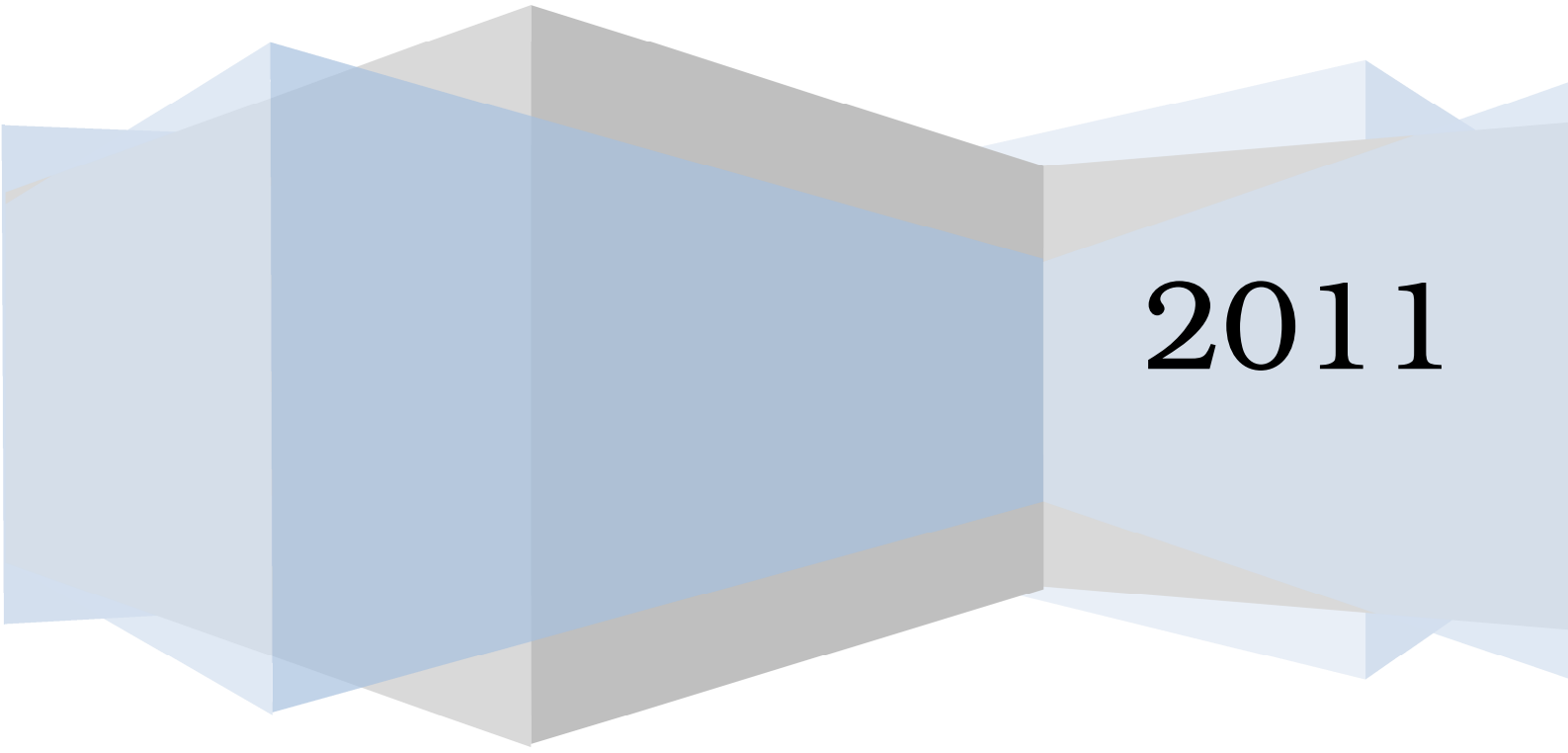
Flávio Augusto de Freitas
Introdução à Programação em Linguagem C/C++

http://flavioaf.blogspot.com

C/C++

Tutorial 3 (usando Dev-C++ versão 4.9.9.2)

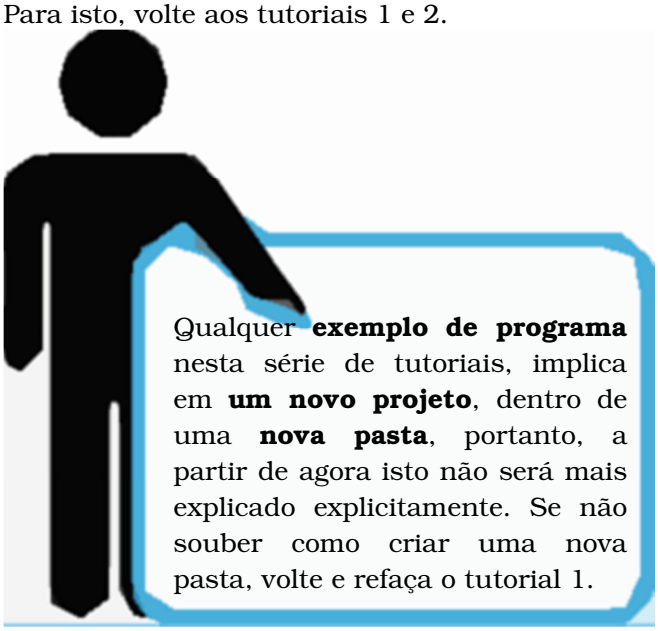


2011

1 INTRODUÇÃO

Esta série de tutoriais foi escrita usando o **Microsoft Windows 7 Ultimate** e o **Bloodshed Dev-C++** versão 4.9.9.2, que pode ser baixada em http://www.bloodshed.net.

A partir deste tutorial não será mais mostrado como criar um novo projeto, nem como compilar um programa e nem verificar quando há erros. Para isto, volte aos tutoriais 1 e 2.



2 PUTS() E PUTCHAR()

puts significa "put string" (colocar string), utilizado para "colocar" uma string na saída de dados. putchar significa "put char" (colocar caractere), utilizado para "colocar" um caractere na saída de dados.

São as funções mais simples do cabeçalho stdio.h. Ambas enviam à saída padrão os caracteres fornecidos a elas; putchar() manda apenas um caractere, e puts() manda uma sequência de caracteres (ou string). Exemplo:

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
  puts("Escrito com puts.");
  putchar('Z');

  system("PAUSE");
  return EXIT_SUCCESS;
}
```

Note que junto com a função puts devemos usar literais de string (com aspas duplas), e com putc devemos usar literais de caractere (com aspas simples). Se você tentasse compilar algo como putc ("T"), o compilador daria uma mensagem de erro. Lembre-se que "T" é diferente de 'T'.

Podemos também colocar caracteres especiais, como a tabulação (\t) e a quebra de linha (\n):

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
  puts("1a. linha\n2a. linha\te um grande espaço");
  putchar('\n'); // envia uma quebra de linha

  system("PAUSE");
  return EXIT_SUCCESS;
}
```

Este código resultaria em algo parecido com:

```
1a. linha
2a. linha        e um grande espaço
```

Note que o argumento deve ser uma sequencia de caracteres. Se você tentar, por exemplo, imprimir o número 42 desta maneira:

```
puts(42);
```

Na verdade o que o compilador tentará fazer é imprimir a sequência de caracteres que começa na posição 42 da memória (provavelmente ele irá alertá-lo sobre isso se você tentar compilar esse código). Se você tentar executar esse código, provavelmente ocorrerá uma falha de segmentação (erro que ocorre quando um programa tenta acessar memória que não lhe pertence). A maneira correta de imprimir o número 42 seria colocá-lo entre aspas duplas:

```
puts("42");
```

STRINGS

Strings (inglês) são cadeias ou sequências ordenadas de caracteres.

A linguagem C, ao contrário de outras linguagens de programação, não possui um tipo de dados correspondente às strings; no lugar, usam-se vetores (e ponteiros, como veremos mais adiante). Em C, strings são vetores de caracteres terminados pelo caractere nulo ('\0'). Por exemplo:

```
char nome[]={'F','l','a','v','i','o','\0'};
```

No entanto, escrever strings dessa maneira é muito trabalhoso; por isso, foi criada uma notação abreviada que equivale à notação acima e elimina a necessidade de colocar o caractere terminador:

```
char nome[] = "Flavio";
```

Assim como nos vetores, podemos acessar e modificar elementos individuais de uma string. Podemos também diminuir o tamanho de uma string: uma vez que a única marcação do tamanho é o terminador \0, colocar um terminador em outro local determinará o novo final da string. No entanto, aumentar o tamanho da string é mais difícil; isso ficará para outra seção.

Atenção ao se usar acentos numa string. Como existem diferentes formas de codificar caracteres acentuados, o tratamento de uma string do tipo:

```
char nome[] = "Flávio";
```

pode ser diferente de uma máquina para outra. Neste tutorial não serão tratados acentos.

Funções da biblioteca padrão

A biblioteca padrão fornece várias funções úteis para manipular strings. A seguir mostraremos algumas delas. Para usá-las, você deve incluir o cabeçalho string.h no início dos seus arquivos.

strlen

strlen retorna o tamanho, em caracteres, de uma string dada. Na verdade o que strlen() faz é procurar o terminador de string e calcular a distância dele ao início da string. Por exemplo:

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
  char nome[] = "Flavio Freitas";
  int s = strlen(nome);
  // s conterá o valor 14
  printf("%s tem %d caracteres\n", s, nome);

  system("PAUSE");
  return EXIT_SUCCESS;
}
```

strcpy

strcpy copia o conteúdo de uma string para outra e coloca um terminador de string. Sua sintaxe é strcpy (destino, origem).

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
  char n1[] = "Clarice Lispector";
  char n2[] = "Flavio Freitas";

  printf("n1: %s - n2: %s\n", n1, n2);

  strcpy (n1, n2);
  // agora n1 conterá "Flavio Freitas"

  printf("n1: %s - n2: %s\n", n1, n2);

  system("PAUSE");
  return EXIT_SUCCESS;
}
```

Tome cuidado com strcpy(), pois se a string a ser copiada for maior que a string de destino, provavelmente você gravará dados em lugares indesejados — um problema conhecido como estouro de buffer. Para evitar esse problema, use a função strncpy, que recebe um terceiro argumento que corresponde ao número máximo de caracteres a serem copiados:

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
  char msg[] = "Bom dia!";
  char nome[] = "Maria da Silva";

  printf("msg: %s - nome: %s\n", msg, nome);

  strncpy (msg, nome, strlen(msg));
  // agora msg conterá "Maria da"

  printf("msg: %s - nome: %s\n", msg, nome);

  system("PAUSE");
  return EXIT_SUCCESS;
}
```

strcat

strcat concatena duas strings, adicionando o conteúdo da segunda ao final da primeira, além do terminador (\0). Note que a primeira string deve ter espaço suficiente para conter a segunda, para que não ocorra um "estouro de buffer". Por exemplo:

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
  char nome[50] = "Maria";
  char sobrenome[] = " da Silva";

  strcat (nome, sobrenome);
  // agora nome contém "Maria da Silva"

  printf("nome: %s\n", nome);

  system("PAUSE");
  return EXIT_SUCCESS;
}
```

Analogamente à função strncpy, existe também a função strncat, onde o número máximo de caracteres a serem copiados é o terceiro argumento.

strcmp

Se você tentar criar duas strings com o mesmo conteúdo e compará-las como faria como números, verá que elas "não são iguais". Isso ocorre porque, na verdade, o que está sendo comparado são os endereços de memória onde estão guardadas as strings. Para comparar o conteúdo de duas strings, você deve usar a função strcmp (ou suas variantes):

```
int strcmp (char *s1, char *s2);
```

O valor de retorno é:

- menor que zero se s1 for menor que s2;
- igual a zero se s1 e s2 são iguais;
- maior que zero se s1 for maior que s2.

Costuma parecer estranho dizer que uma string é menor ou maior que outra; na verdade essa comparação é entre a primeira letra que difere nas duas strings. Assim, se tivermos s1 = "abc" e s2 = "abd", diremos que s2 é maior que s1, pois na primeira posição em que as duas strings diferem, a letra em s2 é "maior".

É importante notar que a comparação feita por strcmp distingue maiúsculas de minúsculas. Isto é, as strings "ABC" e "abc" não são iguais para essa função.

As variantes mais usadas de strcmp são:

strncmp - compara apenas os n primeiros caracteres das duas strings, sendo n um terceiro argumento.

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
  int igual;
  char s1[20], s2[20];

  printf("Digite uma palavra (ex. tarugo): ");
  scanf("%s", &s1); printf("\n\n");

  printf("Digite outra palavra (ex. tartaruga): ");
  scanf("%s", &s2); printf("\n\n");

  igual = !strcmp(s1, s2);

  printf("%s eh %s %s\n\n", s1, igual ? "igual a" : "diferente de", s2);

  // Comparando os 3 primeiros caracteres
  igual = !strncmp(s1, s2, 3);
  printf("%s e %s sao %s ate\' o 3o. caractere\n\n", s1, s2, igual ? "iguais" : "diferentes");

  // Comparando os 4 primeiros caracteres
  igual = !strncmp(s1, s2, 4);
  printf("%s e %s sao %s ate\' o 4o. caractere\n\n", s1, s2, igual ? "iguais" : "diferentes");

  system("PAUSE");
  return EXIT_SUCCESS;
}
```

stricmp - compara duas strings sem distinção entre maiúsculas e minúsculas. A sintaxe é igual à de strcmp. Essa função não faz parte da biblioteca padrão, mas é comumente encontrada como extensão particular de várias delas.

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
  int igual;
  char s1[20], s2[20];

  printf("Digite uma palavra (ex. tarugo): ");
  scanf("%s", &s1); printf("\n\n");

  strcpy(s2, s1);
  igual = !strcmp(s1, s2);
  printf("Usando strcmp: %s eh %s %s\n\n", s1,
         igual ? "igual a" : "diferente de", s2);

  for(int i = 0; s2[i] != '\0'; i++)
    s2[i] = toupper(s2[i]);

  igual = !strcmp(s1, s2);
  printf("Usando strcmp: %s eh %s %s\n\n", s1,
         igual ? "igual a" : "diferente de", s2);

  igual = !stricmp(s1, s2);
  printf("Usando stricmp: %s eh %s %s\n\n", s1,
         igual ? "igual a" : "diferente de", s2);

  system("PAUSE");
  return EXIT_SUCCESS;
}
```

7 Exercícios Propostos

a) Crie um programa para ler um string e imprimir cada caractere em um linha. Exemplo: “teste” deverá ficar
   t
   e
   s
   t
   e
b) Faça um programa para ler dois strings e informar qual dos dois têm mais caracteres.
c) Escreva um programa para ler um string e convertê-lo todo para minúsculo. Exemplo: “Isto eh um String” deverá ficar “isto eh um string”.
d) Escreva outro programa para ler um string e convertê-lo todo para maiúsculo. Exemplo: “Isto eh um string” deverá ficar “ISTO EH UM STRING”.
e) Crie um programa para ler um string e converter as iniciais para maiúsculo e o restante para minúsculo. Exemplo: “isto eh um string” deverá ficar “Isto Eh Um String”.
f) Programe um código para ler dois strings, cada um com exatamente 10 caracteres, que transfira o 3º, 4º e 5º caracteres do string 2 para o string 1.
g) Crie um código que leia um string e inverta-o. Exemplo: “roma” deverá ficar “amor”.
h) Escreva um programa que leia um string e verifique se ele é palíndromo. Exemplo: “osso” é palíndromo, pois o inverso – “osso” -é o mesmo string.

8 Terminamos

Terminamos por aqui. Saia do Dev-C++ e corra para o próximo tutorial.