

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

<http://flavioaf.blogspot.com>

# C/C++

Tutorial 16 (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>. Se alguém quiser adquirir mais conhecimentos e quiser aprofundar no assunto, visite <http://www.cplusplus.com/>.

Qualquer **exemplo de programa** nesta série de tutoriais implica **em um novo projeto**, dentro de uma **nova pasta**. Se não souber como criar uma nova pasta, volte e refaça o tutorial 1.

## 2 O QUE SÃO STRINGS?

Se ainda não estiver familiarizado com strings, veja o tutorial 3.

### 2.1 RELEMBRANDO

No tutorial 3 trabalhamos de uma maneira genérica com strings. Praticamente, só introduzimos o assunto. Entretanto, o uso de strings é vasto e podemos usá-las de diversos modos, criando rotinas e programas para os mais diversos usos:

## 3 ANÁLISE DE EXPRESSÕES E AVALIAÇÃO

Como escrever um programa que recebe como entrada uma string contendo uma expressão numérica, como  $(10 - 5) * 3$ , e calcular a resposta apropriada? Esse procedimento é chamado de *análise de expressões*, e é a espinha dorsal de todos os compiladores e interpretadores de linguagens, programas de planilha de cálculo e qualquer outra coisa que necessite converter expressões numéricas em uma forma que o computador possa usar.

### 3.1 VALORES POR EXTENSO

Para esquentar as coisas, vamos a um exemplo com strings bem mais simples. Que tal criarmos uma função que retorne o valor por extenso de um valor numérico qualquer.

Nossa função deve receber um valor, como 1537 e retornar o texto “mil quinhentos e trinta e sete”. Depois podemos estender nossa função para trabalhar com moedas. Aí um valor como 23484,45 retornará o texto “vinte e três mil, quatrocentos e oitenta e quatro reais e quarenta e cinco centavos”.

Vamos ao código:

Como esse é um programa didático, ele não tem muita precisão, já que usa o tipo **double** para o valor monetário e algumas operações aritméticas a fim de separar os valores da parte inteira e da parte centesimal. Infelizmente, por um problema de arredondamento e de precisão dos tipos float e double, às vezes o valor dos centavos fica diminuído de 1 centavo. Entretanto, volto a frisar, este é um programa didático e não profissional. Nossa intenção é mostrar ao visitante o que o C/C++ pode fazer.

### OBSERVAÇÃO

Não copie, cole e execute simplesmente o código. Procure executá-lo, testá-lo e ver o que ele pode fazer. Depois, tente ler o código e acompanhar o algoritmo, juntamente com os comentários, pois assim você poderá entender a lógica por trás da execução. Isso é de suma importância para o aprendizado de C/C++, bem como de qualquer linguagem de computador.

```
#include <cstdlib>
```

```
#include <iostream>
```

```
using namespace std;
```

```
char* extenso(int);
```

```
int main(int argc, char *argv[])  
{
```

```
    double n;  
    int v_reais;  
    int v_centavos;
```

```
    printf("Forneca um numero entre 0 e 999.99:  
");  
    scanf("%lf", &n);
```

```
    // separamos o valor lido em reais e centavos  
    v_reais = (int) n;  
    v_centavos = (int) (n * 100) % 100;
```

```
    printf("%d \\'real(is)\\\' %d \\'centavo(s)\\\'\\n\\n",  
v_reais, v_centavos);
```

```
    // geramos o valor por extenso  
    if(v_reais>0)  
        printf("%s", extenso(v_reais));  
    if(v_reais>1)  
        printf(" reais");  
    else
```

```

if(v_reais==1) printf(" real");
if((v_centavos>0) && (v_reais>0))
    printf(" e %s", extenso(v_centavos));
else
if(v_centavos>0)
    printf("%s", extenso(v_centavos));
if(v_centavos>1)
    printf(" centavos");
else
if(v_centavos==1) printf(" centavo");
printf("\n\n");

//for(n=0;n<=999;n++)    printf("%s    ",
extenso(n));

system("PAUSE");
return EXIT_SUCCESS;
}

char* extenso(int num) {
    char *unidades[10] = {
        "", "um", "dois", "tres", "quatro", "cinco",
"seis", "sete", "oito",
        "nove"
    };

    char *dezenas[19] = {
        "", "dez", "vinte", "trinta", "quarenta",
"cinquenta", "sessenta",
        "setenta", "oitenta", "noventa", "onze",
"doze", "treze", "quatorze",
        "quinze", "dezesesseis", "dezessete", "dezoito",
"dezenove"
    };

    char *centenas[10] = {
        "", "cento", "duzentos", "trezentos",
"quatrocentos", "quinhentos",
        "seiscentos", "setecentos", "oitocentos",
"novecentos"
    };

    char* s;
    int u, d, c;

    s = (char *) calloc(200, sizeof(char));

    strcpy(s, ""); // variável inicia vazia

    u = num % 10; // unidade
    num = (num - u) / 10;
    d = num % 10; // dezena
    c = (num - d) / 10; // centena

    if((c == 1) && (d == 0) && (u == 0)) { // caso
especial: valor igual a 100

```

```

        strcat(s, "cem"); // caso especial
        return s;
    }
    else
        if((c >= 1) && ((d != 0) || (u != 0))) { // valores
entre 101 e 999
            strcat(s, centenas[c]);
            if(d > 0) strcat(s, " e ");
            strcat(s, dezenas[d]);
            if(u > 0) strcat(s, " e ");
            strcat(s, unidades[u]);
            return s;
        }
    else
    {
        if((c == 0) && (d > 0) && (u == 0)) { //
valores como 10, 20, ..., 90 etc.
            strcat(s, dezenas[d]);
            if(u > 0) {
                strcat(s, " e ");
                strcat(s, unidades[u]);
            }
            return s;
        }
    else
        if((d > 0) && (d != 1) && (u > 0)) { // valores
entre 21 e 99
            strcat(s, dezenas[d]);
            strcat(s, " e ");
            strcat(s, unidades[u]);
            return s;
        }
    else
        if((c > 0) && (d == 0) && (u == 0)) { // caso
especial: centenas redondas como 500, 600 etc.
            strcat(s, centenas[c]);
            return s;
        }
    else
        if((d == 1) && (u > 0)) {
            strcat(s, dezenas[d*10 + u - 1]); // caso
especial: valores entre 11 e 19
            return s;
        }
    else
    {
        strcat(s, unidades[u]); // valores entre 0 e 9
        return s;
    }
}
}
}
}

```

## 4 EXERCÍCIOS RESOLVIDOS

1. Escreva um programa para copiar uma string para outra.

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

int main() {
    char var1[]="Sample string";
    char var2[40];
    char var3[40];
    strcpy (var2,var1); // copia var1 para var2
    strcpy (var3,"sucesso"); // copia o texto para var3
    printf ("str1:  %s\nstr2:  %s\nstr3: %s\n",var1,var1,var3);
    return 0;
}
```

2. Escreva um programa para copiar 5 caracteres de uma string para outra.

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

int main() {
    char var1[] = "Exemplo de strncpy";
    char var2[6];
    strncpy (var2,var1,5); // copia 5 caracteres
    var2[5]='\0'; // obrigatório indicar o fim
    printf("%s",var2);
    return 0;
}
```

3. Escreva um programa para copiar um bloco de caracteres de uma string para outra.

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

int main() {
    char string0[] = "Movimentando blocos";
    memmove (string0+10,string0+5,5);
    printf("%s\n",string0);
    return 0;
}
```

4. Idem exercício 3, mas usando outro comando.

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

int main() {
    char string0[]="Copiar bloco";
    char string1[20];
    char string2[30];
    memcpy (string1,string0,strlen(string0)+1);
    memcpy (string2,"Copia feita",16);
    printf("1:%s\n2:%s\n3:%s\n", string0, string1, string2);
    return 0;
}
```

5. Escreva um programa para “quebrar” uma string em partes.

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

int main() {
    char str[] = "Um exemplo de strtok";
    strtok (str,"o");
    printf("%s\n",str);
    return 0;
}
```

6. Escreva um programa para localizar uma string dentro de outro.

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

int main() {
    char frase[40] = "Exemplo de strstr";
    char substituido[40] = "strstr";
    char substituto[40] = "---strstr---";
    strncpy(strstr(frase, substituido), substituto,
    strlen(substituto));
    printf("%s\n",frase);
    return 0;
}
```

7. Escrever um programa para procurar repetições em uma frase.

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

int main() {
    int i;
    char frase[] = "Frase";
    char frase1[] = "Fraser";
    printf ("As letras de frase sao iguais a de
    frase1 ate a posicao %d.\n", strspn(frase,
    frase1));
    return 0;
}
```

## 4 SOLUÇÃO DOS EXERCÍCIOS PROPOSTOS DO TUTORIAL 15

- a) Escreva um programa usando função para calcular uma expressão numérica passada na forma *número1 operador número2*, onde operador pode ser qualquer um de: +, -, \*, /, que o usuário fornecer. Os números também são fornecidos pelo usuário.

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

using namespace std;

float fazOp(float, char, float);

int main(int argc, char *argv[])
{
    float a, b;

    printf("Numero 1: ");
    scanf("%f", &a);
    putchar('\n');

    printf("Numero 2: ");
    scanf("%f", &b);
    putchar('\n');

    printf("\nResultados para %f e %f:\n", a,
    b);
    printf("\tsoma.....: %f\n", fazOp(a, '+',
    b));
    printf("\tdiferenca.....: %f\n", fazOp(a, '-',
    b));
    printf("\tproduto.....: %f\n", fazOp(a, '*',
    b));
    printf("\tdivisao.....: %f\n", fazOp(a, '/',
    b));
}
```

```
printf("\n\nTentando o operador
\\@\':\n");
printf("\toperacao invalida: %f\n", fazOp(a,
'@', b));
```

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

```
float fazOp(float n1, char op, float n2) {
    switch(op) {
        case '+': return n1 + n2; break;
        case '-': return n1 - n2; break;
        case '*': return n1 * n2; break;
        case '/': return n1 / n2; break;
        default:
            printf("\tERRO: Operador %c
            invalido!\n\n", op);
            return 0;
            break;
    }
}
```

- b) Faça uma função que verifique se um número inteiro é perfeito ou não. Um número inteiro é dito perfeito quando ele é igual a soma dos seus divisores excetuando-se ele próprio. (Exemplo: 6 é perfeito,  $6 = 1 + 2 + 3$ , que são seus divisores, exceto o próprio 6). A função deve retornar um valor booleano. Use a função para testar os primeiros 200 números naturais.

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

```
using namespace std;
```

```
int numPerfeito(int);
```

```
int main(int argc, char *argv[])
{
    int a;
```

```
    printf("Verificar se um numero e
    perfeito.\n");
    printf("\tDigite um numero inteiro: ");
    scanf("%d", &a);
    putchar('\n');
    if(numPerfeito(a))
        printf("E perfeito!\n\n");
    else printf("Nao e perfeito!\n\n");
```

```
system("PAUSE");
system("CLS");
```

```

printf("Verificando os primeiros 200
naturais.\n");
for(a=1;a<=200;a++) {
    if(a%5==0) {
        system("PAUSE");
        system("CLS");
    }
    if(numPerfeito(a))
        printf(" >>>> E perfeito!\n");
    else printf(" >>>> Nao e perfeito!\n");
}

system("PAUSE");
return EXIT_SUCCESS;
}

```

```

int numPerfeito(int n) {
    int i;
    int soma = 0;

    printf("\nDivisores: ");
    for(i=1;i<=n;i++) {
        if(n%i==0) {
            printf("%d; ", i);
            if(i<n) soma += i;
        }
    }

    printf("\n\nSoma dos divisores de %d,
exceto %d = %d\n\n", n, n, soma);
    return soma==n;
}

```

- c) Faça uma função que recebe um valor inteiro e verifica se o valor é positivo ou negativo. A função deve retornar um valor booleano.

```

#include <cstdlib>
#include <iostream>
#include <stdbool.h>

using namespace std;

bool numPositivo(int);

int main(int argc, char *argv[])
{
    int a;

    printf("Verificar se um numero e positivo
ou negativo.\n");
    printf("\tDigite um numero inteiro: ");
    scanf("%d", &a);
    putchar('\n');
    if(numPositivo(a))
        printf("E positivo!\n\n");
}

```

```

else printf("Nao e positivo!\n\n");

system("PAUSE");
return EXIT_SUCCESS;
}

```

```

bool numPositivo(int n) {
    return n>=0;
}

```

- d) Escreva uma função que recebe por parâmetro um valor positivo N e retorna o valor de S.

$$S = 1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/N.$$

```

#include <cstdlib>
#include <iostream>

```

```

using namespace std;

```

```

//S = 1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/N

```

```

double calculaS(double);

```

```

int main(int argc, char *argv[])
{
    double a;

```

```

    printf("Calcula somas do tipo S = 1 + 1/2 +
1/3 + 1/4 + 1/5 + 1/N\n");
    printf("Forneca N: ");
    scanf("%lf", &a);

```

```

    printf("Soma S = %lf\n\n", calculaS(a));

```

```

    system("PAUSE");
    return EXIT_SUCCESS;
}

```

```

double calculaS(double n) {
    double denominador = 1.0;
    double soma = 0.0;

    while(denominador<=n) {
        soma += 1.0/denominador;
        denominador+=1.0;
    }

    return soma;
}

```

- e) Escreva uma função que recebe por parâmetro um valor inteiro e positivo N e retorna o valor de S.

$$S = 1 + 1/1! + 1/2! + 1/3! + 1/N!$$

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

```
using namespace std;
```

```
//S = 1 + 1/1! + 1/2! + 1/3! + 1/N!
```

```
double calculaS(double);
double calculaFatorial(double);
```

```
int main(int argc, char *argv[])
{
    double a;
```

```
    printf("Calcula somas do tipo S = 1 + 1/1! + 1/2! + 1/3! + 1/N!\n");
```

```
    printf("Para N > 1000 a soma S se aproxima do numero 'e', base dos logaritmos naturais.\n");
```

```
    printf("Alias, numeros como e, sen(x), cos(x), tan(x), log(x) etc. sao calculados assim.\n\n");
```

```
    printf("Forneca N: ");
```

```
    scanf("%lf", &a);
```

```
    printf("Soma S = %0.20lf\n\n", calculaS(a));
```

```
    system("PAUSE");
```

```
    return EXIT_SUCCESS;
```

```
}
```

```
double calculaS(double n) {
    double denominador = 0.0;
    double soma = 0.0;
```

```
    while(denominador<=n) {
```

```
        soma +=
```

```
        1.0/calculaFatorial(denominador);
```

```
        denominador+=1.0;
```

```
    }
```

```
    return soma;
```

```
}
```

```
double calculaFatorial(double n) {
    double fat = 1.0;
    double i = 1.0;
```

```
    if((n==0.0) || (n==1.0)) return 1.0;
```

```
    while(i<=n)
```

```
        fat*=i++;
```

```
    return fat;
```

```
}
```

- f) Escreva uma função que recebe por parâmetro um valor inteiro e positivo N e retorna o valor de S.

$$S = 3/4 + 5/5 + 7/6 + \dots + (2n + 1) / (n + 3)$$

```
#include <cstdlib>
```

```
#include <iostream>
```

```
using namespace std;
```

```
//S = 3/4 + 5/5 + 7/6 + ... + (2n + 1) / (n + 3)
```

```
double calculaS(double);
```

```
int main(int argc, char *argv[])
{
```

```
    double a;
```

```
    printf("Calcula somas do tipo S = 3/4 + 5/5 + 7/6 + ... + (2n + 1) / (n + 3)\n\n");
```

```
    printf("Forneca n: ");
```

```
    scanf("%lf", &a);
```

```
    printf("Soma S = %0.20lf\n\n", calculaS(a));
```

```
    system("PAUSE");
```

```
    return EXIT_SUCCESS;
```

```
}
```

```
double calculaS(double n) {
```

```
    double numerador;
```

```
    double denominador;
```

```
    double soma = 0.0;
```

```
    double i = 1.0;
```

```
    while(i<=n) {
```

```
        numerador = 2*i + 1;
```

```
        denominador = i + 3;
```

```
        soma += numerador/denominador;
```

```
        i+=1.0;
```

```
    }
```

```
    return soma;
```

```
}
```

## 5 EXERCÍCIOS PROPOSTOS

1. Elabore um programa que receba uma linha de texto e conte as vogais apresentando o respectivo histograma na seguinte forma:

Exemplo:

Linha de texto passada: “Na próxima quarta-feira é feriado.”

a: \*\*\*\*\* (6)

e: \*\*\* (3)

i : \*\*\* (3)

o: \*\* (2)

u: \* (1)

2. Implemente um programa que receba uma linha de texto, retire os espaços em excesso existentes deixando apenas um espaço entre as várias palavras.
3. Implemente um programa que receba um nome e apresente apenas o apelido e o 1º nome na seguinte forma:

Apelido, 1º nome

Exemplo:

Flávio Augusto de Freitas

Freitas, Flávio

## 6 TERMINAMOS

Terminamos por aqui. O que está esperando, saia do Dev-C++ e corra para pegar o próximo tutorial em <http://flavioaf.blogspot.com>. Siga o blog, assim você fica sabendo das novidades no momento em que forem publicadas. Seguindo o blog você se mantém sempre atualizado de qualquer lançamento novo.