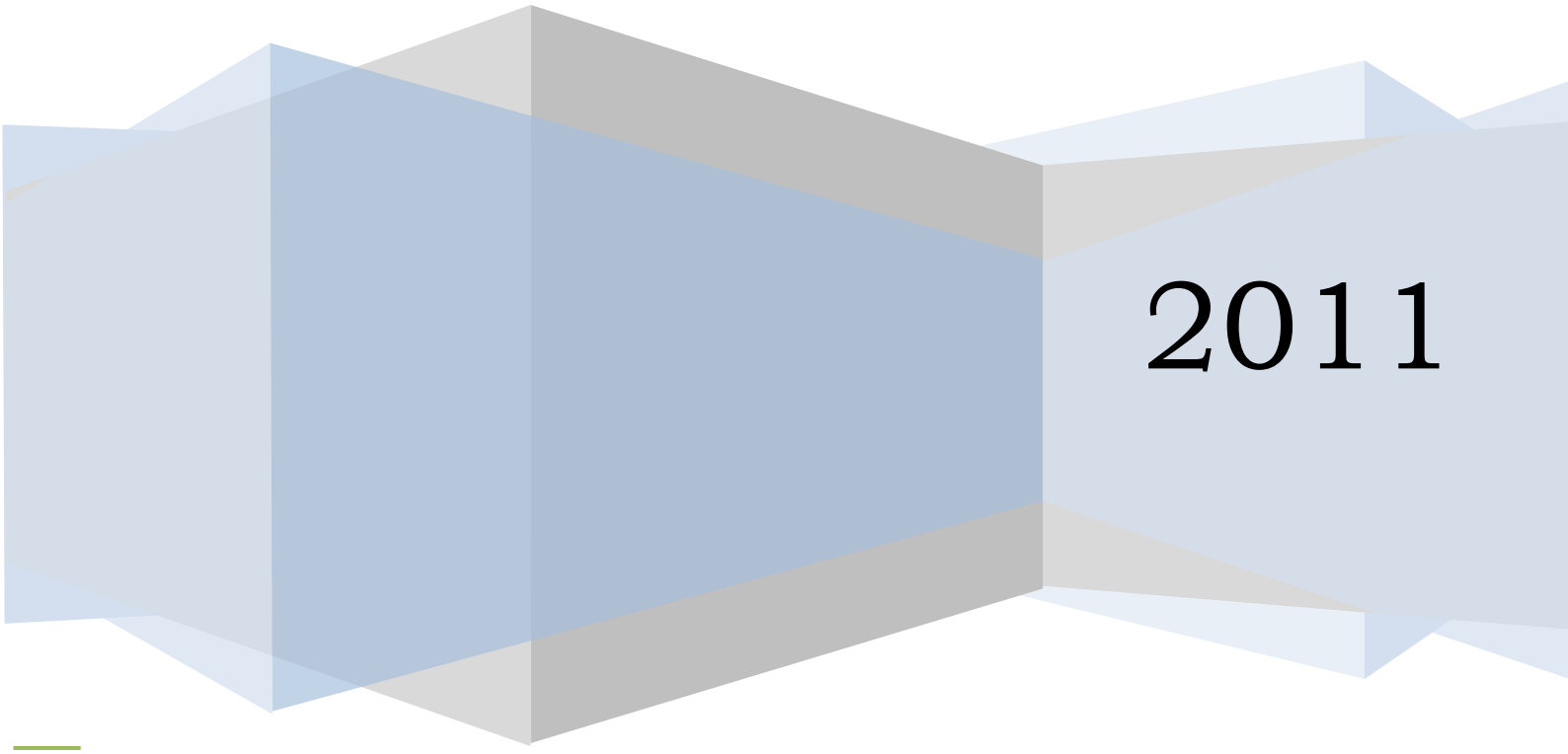
Flávio Augusto de Freitas
Introdução à Programação em Linguagem C/C++

http://flavioaf.blogspot.com

C/C++

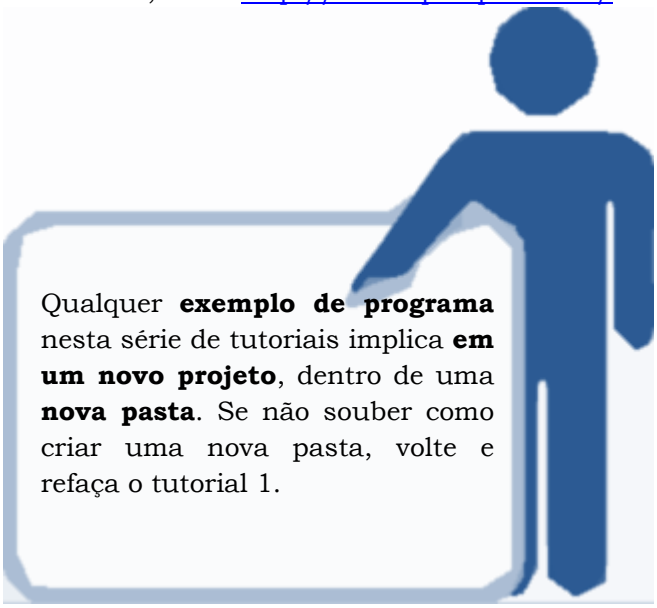
Tutorial 14 (usando Dev-C++ versão 4.9.9.2)



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 FUNÇÕES?

Podemos definir uma função como um bloco de código que possui um identificador que permite sua reutilização.

Confuso? Vamos a um exemplo:

Vamos supor que sua aplicação calcule uma função que precise do resultado de vários fatoriais. Por exemplo:

$$s = \sum_{i=0}^{100} \frac{(100 - i)!}{2 \times i!}$$

EQUAÇÃO 1: O SOMATÓRIO ACIMA É O MESMO QUE $s = \frac{(100-0)!}{2\times 0!} + \frac{(100-1)!}{2\times 1!} + \frac{(100-2)!}{2\times 2!} + \cdots + \frac{(100-100)!}{2\times 100!}$, ONDE I = 0, 1, 2, ..., 100

Você precisará calcular 101 vezes (0 a 100) o fatorial de 100 – i e o fatorial de i, além de calcular a divisão e acumular o resultado em uma variável.

Então, ficaria assim em algoritmo:

Passo 1: Zerar a variável s

Passo 2: Iniciar i com 0

Passo 3: Calcular o fatorial de 100 – i

Passo 4: Calcular o fatorial de i

Passo 5: Calcular (100 – i)! / (2 * i!)

Passo 6: Somar o resultado do Passo 5 à variável s

Passo 7: Incrementar i

Passo 8: Voltar ao Passo 3 até que i seja igual a 100

Passo 9: Imprimir s

Ok, mas o que é um fatorial?

Fatorial é uma função matemática definida assim por n! (leia-se fatorial de n ou n fatorial):

$$n! = \begin{cases} 1, se\ n = 0 \\ 1, se\ n = 1 \\ n \times (n-1) \times (n-2) \times \cdots \times 1, se\ n > 1 \end{cases}$$

Por exemplo, 3! = 3.2.1 = 6, 5! = 5.4.3.2.1 = 120, 10! = 10.9.8.7.6.5.4.3.2.1 = 3628800, e assim por diante.

Agora imagine tendo que calcular os fatoriais de 100, 99, 98, ..., 0 do Passo 3 do nosso algoritmo, sendo que para cada vez que o passo é executado você teria de fazer os cálculos 100!, 99! etc. Cada um desses fatoriais envolve a multiplicação de todos os inteiros até o 1, ou seja, para 100! temos: 100.99.98.97. ... .1. Seu programa ficará enorme. Mais, se o somatório definido anteriormente fosse i = 0 .. 1000, ou i = 0 .. 100000. Ficaria quase impossível codificar se não usarmos funções.

Por exemplo, **seja o problema**:

Escreva um programa na linguagem C para receber um número N, inteiro, digitado pelo usuário. Caso o número digitado seja ímpar e menor que 10, imprimir o fatorial desse número. O cálculo do fatorial deve ser feito por uma função, que recebe o valor digitado por parâmetro e retorna o valor do fatorial calculado. Caso N seja par e maior ou igual a dez, passar esse valor para outra função por parâmetro e calcular e retornar a soma dos inteiros de 1 a N. Imprimir o valor retornado pela função.

Solução

```
#include <stdio.h>

int fatorial (int a) {
  int i, valor = 1;
  for(i=1; i<=a; i++)
    valor = valor * i;
  return valor;
}

int somatoria (int a) {
  int i, soma = 0;
  for (i=1; i<=a; i++)
    soma = soma + i;
  return soma;
}

int main() {
  int num;
  scanf("%d", &num);
  if( (num %2 != 0) && (num < 10) )
    printf("%d", fatorial(num) );
  else if( (num % 2 == 0) && (num >= 10) )
      printf("%d", somatoria(num));

  return 0;
}
```

Aqui há três funções: **main**, **somatoria** e **fatorial**. As três funções retornam valores inteiros. **somatoria** e **fatorial** recebem parâmetros inteiros. Somente a função **main** não recebe valor algum.

Uma particularidade é que todas as funções devem ser declaradas antes de **main**, assim **main** fica sabendo que as funções existem, senão haverá um erro ao compilar o programa.

Há um modo de usar funções após o **main**, usando *protótipos de funções*, que não serão tratadas neste tutorial.

3 Exercícios Resolvidos

1. Escreva um programa que leia um inteiro não-negativo n e imprima a soma dos n primeiros números primos.

```
#include <stdio.h>

int testaPrimo (int a) {
  int i, total = 0;
  for (i=1; i<= a; i++)
    if (a % i == 0) total++;
  if (total==2)
    return 1;
  else
    return 0;
}

int main() {
  int valor = 2, quantidade, soma = 0, cont =
0;
  scanf("%d", &quantidade);

  while (cont<quantidade) {
    if ( testaPrimo(valor) ) {
      soma = soma + valor;
      cont++;
    }
    valor++;
  }

  printf("A soma e´ %d.", soma);

  return 0;
}
```

2. Faça uma função que receba uma palavra de até 100 caracteres e imprima o número de aparições de uma letra escolhida pelo usuário.

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

int contaCaracteres (char *palavra, char letra);

int main() {
  char texto[100], letra;
  int total;
  scanf("%s", texto);
  getchar();
  scanf("%c", &letra);
  total = contaCaracteres (texto, letra);
  printf("A letra %c aparece %d vezes na palavra %s.", letra, total, texto);
  return 0;
}

int contaCaracteres (char *palavra, char letra)
{
  int tamanho, i, total=0;
  tamanho = strlen(palavra);
  for (i=0; i < tamanho; i++) {
    if (palavra[i] == letra)
      total++;
  }
  return total;
}
```

3. Escreva uma função desenhaQuadrado que exibe um quadrado sólido (o mesmo número de linhas e colunas). O caractere utilizando para preencher o quadrado e o valor do lado são passados como argumentos para a função. Por exemplo, se o caractere for x e o valor do lado for 5, a função deverá exibir

```
xxxxx
xxxxx
xxxxx
xxxxx
xxxxx
```

```
#include <stdio.h>

void desenhaQuadrado (char a, int b);

int main() {
  int lado;
  char letra;
  scanf("%c", &letra);
  scanf("%d", &lado);
  desenhaQuadrado(letra, lado);
  return 0;
}

void desenhaQuadrado (char a, int b) {
  int i, j;
  for (i=0; i < b; i++) {
    for (j=0; j < b; j++) {
      printf("%c ", a);
    }
    printf("\n");
  }
}
```

4 EXERCÍCIOS PROPOSTOS

a) Escreva uma função que receba um vetor de inteiros não ordenados e um valor inteiro que será pesquisado no vetor. A função deve retornar o elemento do vetor que está mais próximo do valor inteiro. É possível que haja mais de um elemento que esteja com igual proximidade do elemento pesquisado. Se for necessário, a função pode ser receber também o tamanho do vetor como parâmetro (válido para implementações em C). Se o vetor estivesse ordenado, haveria diferença na implementação? Haveria alguma vantagem? Reflita sobre isso.

b) Escreva uma função que leia um vetor de inteiros ordenados de forma não decrescente e que imprima somente os números que não sejam repetidos.

c) Faça uma função para ler um vetor. Este procedimento deve receber o número de elementos do vetor e retornar o vetor lido. Faça também um procedimento para mostrar os elementos de um vetor. Este procedimento deve receber o vetor e o número de elementos deste vetor. Faça um algoritmo e um programa que leia 2 vetores A (com 5 elementos) e B (com 5 elementos) utilizando o procedimento de leitura de vetor. O algoritmo/programa deverá fazer com que o vetor C receba os elementos do vetor A multiplicados pelos elementos correspondentes do vetor B. Por fim o algoritmo/programa deverá chamar o procedimento que mostra os elementos de um vetor para mostrar os elementos dos vetores A, B e C.

d) Escreva uma função int remove_dup(float v[], int n) receba um vetor e verifique a existência de elementos duplicados. Caso não existam elementos duplicados retorne zero. Caso existam, remova estes elementos (deixando apenas um) e retorne o número de elementos removidos.

e) Escreva uma função void insert(float v[], int n, float valor, int pos) que faça a inserção de valor na posição pos do vetor v, deslocando os demais elementos.

5 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.