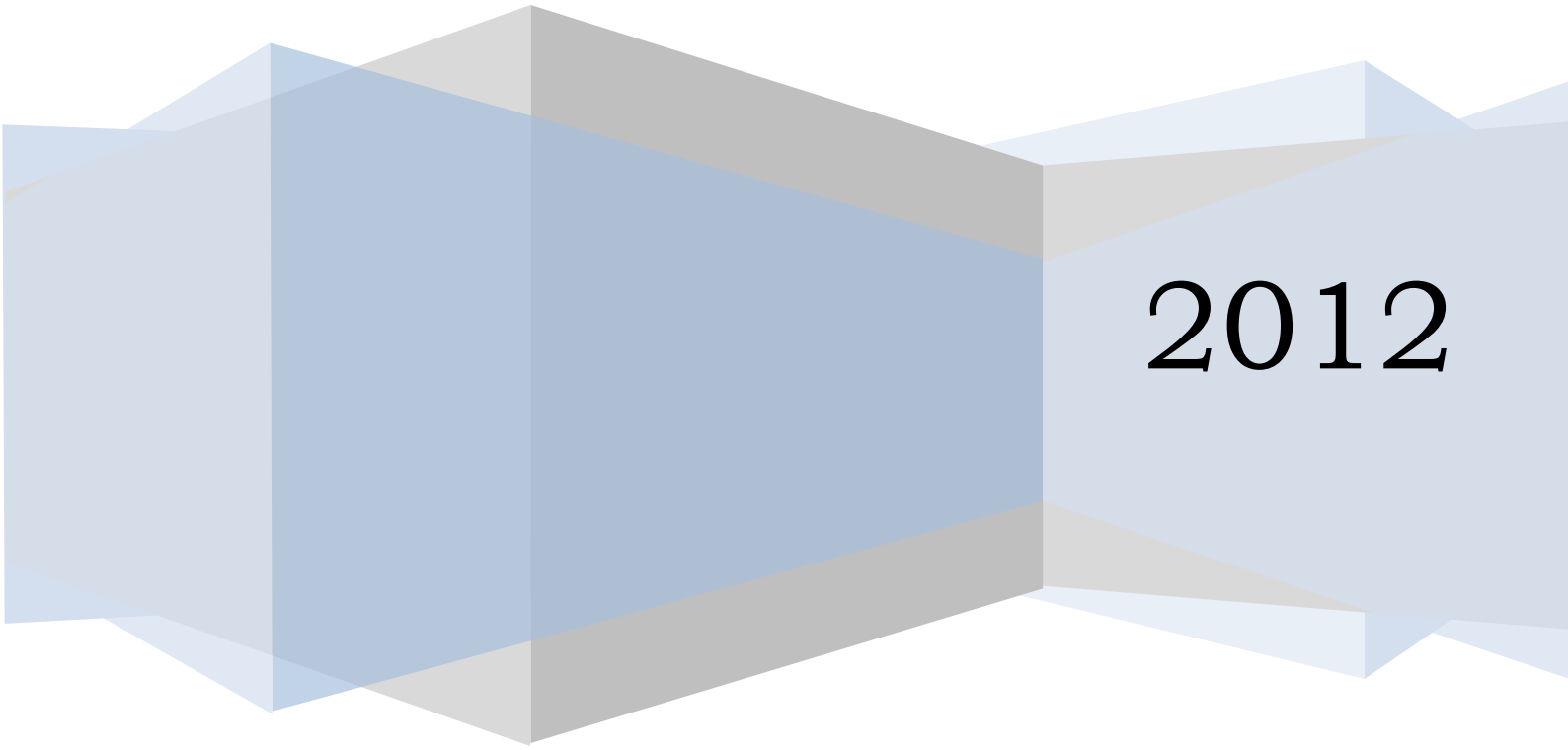
Flávio Augusto de Freitas
Algoritmos e Estruturas de Dados III usando C/C++
http://flavioaf.blogspot.com

C/C++

Tutorial 2 (usando Dev-C++ versão 4.9.9.2)



2012

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.

2 Entrada e Saída de Disco

Agora que já relembramos o uso do DevC++ e um pouquinho de linguagem C/C++, vamos ao que realmente interessa nesse tutorial, ou seja, trabalhar com arquivos em disco.

Se você ainda não está familiarizado com arquivos em disco, leia novamente o Tutorial 1.

3 Funções de Data e Hora

As funções de data e hora servem para obtermos o relógio atual do computador. Usaremos essas funções para obtermos dados de funcionamento e desempenho de nossos programas. Em algoritmos de ordenação essa é uma medida do desempenho e pode ser usada para comparar algoritmos entre si.

Por exemplo, em um algoritmo de ordenação o modo mais simples é obter a hora de início antes da chamada à função de ordenação e depois, a hora de término. Em seguida, calculamos o tempo decorrido e, assim, podemos comparar diversos métodos de ordenação e verificar qual é o mais rápido. Ou ainda, podemos ordenar o mesmo conjunto de dados para diferentes algoritmos, alterando-se a quantidade de dados em cada execução do algoritmo e depois compará-los entre si, conseguindo informações a respeito do desempenho dos algoritmos de acordo com a quantidade de dados a ser ordenada. Isto nos permitirá, por exemplo, decidir qual algoritmo se sai melhor com determinada quantidade de dados.

3.1 Obtendo a Hora Atual

```
// obtendo a hora local usando localtime
#include <stdio.h>
#include <time.h>

int main () {
  time_t rawtime;
  struct tm * timeinfo;

  time(&rawtime);
  timeinfo = localtime (&rawtime);
  printf("Tempo local: %s", asctime(timeinfo));
  return 0;
}
```

3.2 Obtendo o Tempo Decorrido

```
// obtendo tempo decorrido usando difftime
#include <stdio.h>
#include <time.h>

int main () {
  time_t start,end;
  char szInput[256];
  double dif;

  time(&start);
  printf("Por favor, digite seu nome completo: ");
  gets(szInput);
  time(&end);
  dif = difftime(end, start);
  printf("Oi %s.\n", szInput);
  printf("Você levou %.2lf segundos para digitar seu nome.\n", dif);

  return 0;
}
```

4 Lendo o Arquivo de Dados Fornecido no Blog

Baixe o arquivo clicando em http://flavioaf.blogspot.com/2012/02/10000-dez-mil-numeros-inteiros.html. Salve-o na Área de Trabalho (Desktop).

Este arquivo contém 10000 números inteiros aleatórios (randômicos), dispostos em 2000 linhas de 5 colunas cada. Cada número pode variar entre 0 e 99999.

Primeiramente, para trabalhar com esses dados precisaremos de um grande vetor de inteiros. Infelizmente o maior inteiro sem sinal em C/C++, unsigned int, pode armazenar valores até 65535. Precisamos usar um tipo maior, por exemplo, unsigned long int, que pode armazenar valores até 4294967296. Este tipo usa 4 bytes de espaço em memória. São então 4 x 10000 = 40000 bytes ou 40 KB, ou ainda, pouco mais de 39 KiB.

4.1 Lendo e Imprimindo o Arquivo

```
#include <cstdlib>
#include <iostream>
#include <time.h>

using namespace std;

const char fPath[100] = "C:\\";     // altere isto
const char fName[20] = "10000int.txt";
```

```
int main(int argc, char *argv[]) {
   FILE *fp;
   int x;
   int n;
   int l;
   time_t start;
   time_t end;
   double dif;
   char fFullName[200];
   struct tm *timeinfo;

   time(&start);

   strcpy(fFullName, fPath);
   strcat(fFullName, fName);
   strcat(fFullName, "\0");

   fp=fopen(fFullName, "rt"); // rt? Tutorial 1

   if(fp != NULL)
      printf("Arquivo aberto com sucesso.\n");
   else printf("Nao foi possivel abrir o
arquivo.\n");

   for(l=0;l<1000;l++) {
      for(n=0;n<10;n++) {
         fscanf(fp, "%d", &x);
         printf("%d\t", x);
      }
   }
   fclose(fp);

   time(&end);
   dif = difftime(end, start);

   printf("\n\nProcessado o arquivo: %s\n",
fName);
   printf("Local: %s\n\n", fPath);
   timeinfo = localtime(&start);
   printf("Hora de inicio...: %s",
asctime(timeinfo));
   timeinfo = localtime(&end);
   printf("Hora de termino..: %s\n",
asctime(timeinfo));
   printf("Tempo decorrido..: %f segundos \n\n",
dif);

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

5 Arquivos Com Formatos Especiais

5.1 Gerando Um Arquivo

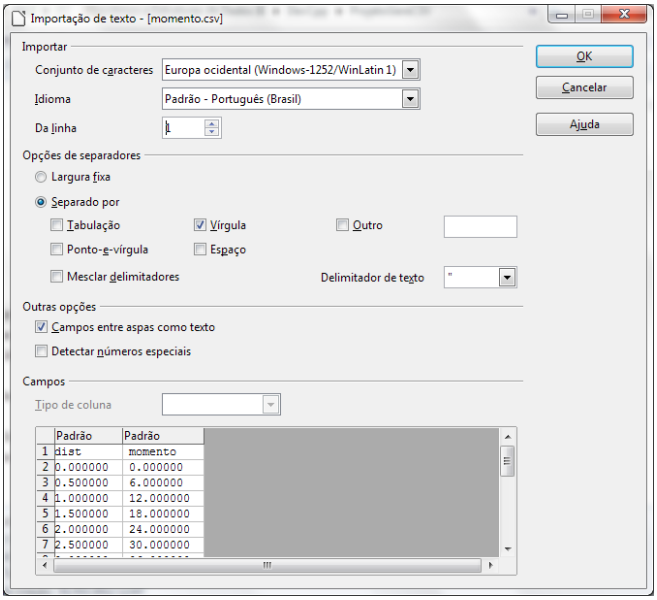
5.1.1 Gerando um Arquivo Para Importação em Planilha Eletrônica

O programa a seguir grava um arquivo de texto chamado **momento.csv** no disco. Digite-o no DevC++ e salve-o na sua pasta Área de Trabalho (Desktop).

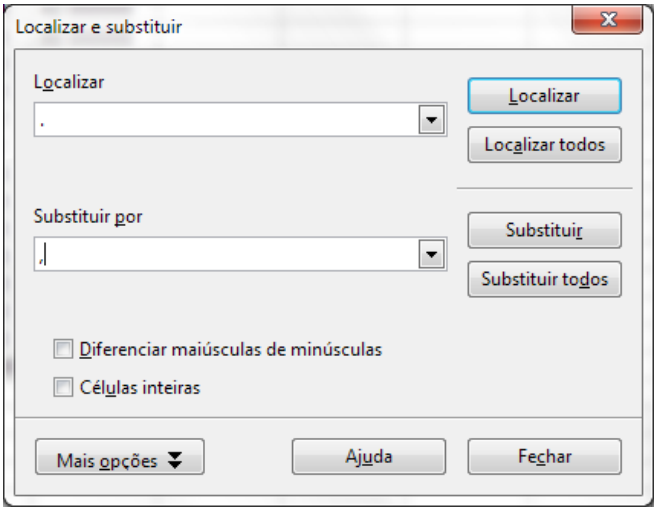
Note que o programa usa uma vírgula entre os valores gravados.

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

float momento(float f, float d) {
  return (f * d);
}

int main(void) {
  FILE *arq;
  float d, f, m;

  arq = fopen("momento.csv","w");
  if (arq == NULL) {
    printf("Erro ao criar o arquivo.\n");
    exit(1);
  }
  fprintf(arq,"%s, %s\n","dist","momento");
  f=12.0;
  for (d=0.0; d <= 10.0; d = d + 0.5) {
    m = momento(f, d);      // calcula
    fprintf(arq,"%f, %f\n",d,m);    // grava
  }
  fclose(arq);
  printf("Arquivo gravado com sucesso!\n");
  return (0);
}
```

Use o BrOffice.org Calc para abrir o arquivo recém-criado momento.csv. Deixe a janela de importação como na figura abaixo e clique no botão OK.

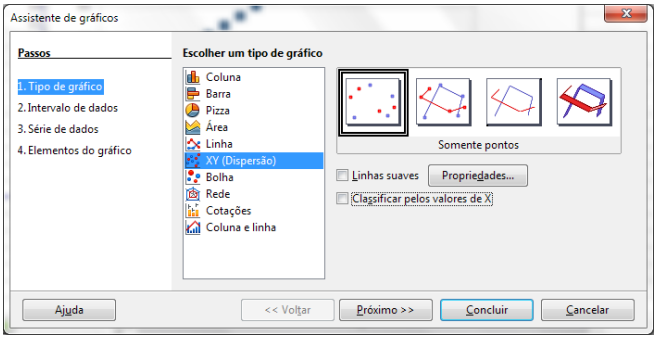


Ainda, o DevC++ usa um ponto no lugar de uma vírgula para separar a parte inteira da parte decimal de um número, o que também deve ser informado ao BrOffice.org Calc. Assim que os dados aparecerem na planilha, clique no menu Editar e escolha a opção Localizar e substituir.... No campo Localizar digite **.** (ponto) e no campo Substituir por digite **,** (vírgula). Use a figura abaixo como referência.



Clique no botão Substituir todos e depois no botão Fechar.

No BrOffice.org Calc selecione os dados arrastando o mouse sobre eles, desde a célula A1 até a célula B22. Clique no menu Inserir e escolha Gráfico.... Deixe a janela Assistente de gráficos como na figura a seguir e clique no botão Concluir.



Salve a pasta de trabalho, e feche o BrOffice.org Calc.

5.1.2 Gerando um Arquivo Para Exibição em um Browser

Você já deve saber como criar um arquivo contendo marcas (tags) HTML para ser aberto em um browser (navegador). Se não souber, dê uma olhada em http://www.freewebs.com/aprenderhtml/.

Agora verá uma adaptação do programa anterior para salvar os dados em uma tabela dentro de uma página HTML. As marcas <table> e </table> delimitam uma tabela. Dentro delas, isto é, entre elas, as marcas <tr> e </tr> (de table row) delimitam uma linha da tabela. Dentro de uma linha, as marcas <td> e </td> delimitam uma célula (coluna). Você pode usar também as marcas <th> e </th> para delimitar uma célula de uma linha de cabeçalho da tabela com um formato diferente do corpo. O trecho abaixo mostra o código HTML para criar uma tabela com uma linha de cabeçalho e uma linha de dados, ambas com duas colunas. O parâmetro border dentro da marca <table> indica a largura desejada para a borda da tabela.

```
<table border="1">
  <tr>
    <th>Dist</th> <th>Momento</th>
  </tr>
  <tr>
    <td>0.0</td><td>0.0</td>
  </tr>
</table>
```

O programa a seguir grava um arquivo de texto chamado **momento.htm** no disco. Digite-o no DevC++ e salve-o na sua pasta Área de Trabalho (Desktop).

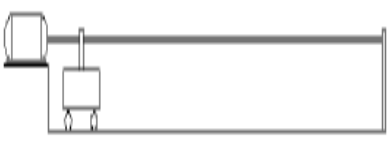
Execute o programa. Usando o atalho Meu Computador, localize o ícone do arquivo **momento.htm** que seu programa acabou de criar. Dê um duplo clique nele para abri-lo no navegador Web padrão.

Veja que os dados aparecem dentro de uma tabela. Clique com o botão direito do mouse em algum ponto da página e selecione a opção Exibir código fonte, para ver o código da página. Lembre que esse conteúdo foi escrito pelo programa em linguagem C. Feche a janela de exibição do código fonte e a janela do Navegador.

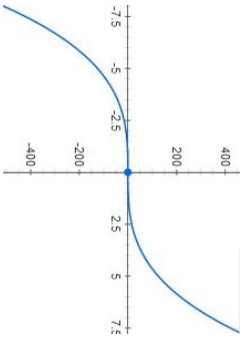
```
#include <cstdlib>
#include <iostream>

using namespace std;

float momento(float f, float d) {
  return (f * d);
}

int main(void) {
  FILE *arq;
  float d, f, m;
  arq = fopen("momento.htm","w");
  if (arq == NULL) {
    printf("Erro ao criar o arquivo.\n");
    exit(1);
  }
  fprintf(arq,"<html>\n\n<head>\n");
  fprintf(arq,"<title>Momento</title>\n");
  fprintf(arq,"</head>\n<body>\n");
  fprintf(arq,"<table border=1>\n");
  fprintf(arq,"<tr>\n");
  fprintf(arq,"<th>Dist.</th><th>Momento</th>\n");
  fprintf(arq,"</tr>\n");
  /* fica calculando o momento e gravando */
  f = 12.0;
  for (d=0.0; d <= 10.0; d = d + 0.5) {
    m = momento(f, d); /* calcula */
    fprintf(arq,"<tr>\n");
    fprintf(arq,"<td>%f</td><td>%f</td>\n",d,m);
    fprintf(arq,"</tr>\n");
  }
  fprintf(arq,"</table>\n</body>\n</html>\n");
  fclose(arq);
  printf("Arquivo gravado com sucesso!\n");
  system("PAUSE");
  return (0);
}
```

6 Exercícios

1. Um carro desliza sobre trilhos movido por um parafuso acoplado a um motor. Sabendo que o passo da rosca do parafuso é igual a 2 mm/volta e que a velocidade do motor é de 1800 rpm (lembre que 1800 rpm = 30 voltas por segundo), escreva um programa em linguagem C que calcule a posição (distância da origem) do carro em 11 instantes de tempo iniciando com t = 0s e terminando em t = 60s e grave os pares de valores (instante de tempo, posição do carro) em um arquivo de texto chamado **movimento.csv**. Considere que o carro está na origem no instante t = 0s. Cada par de valores deve ser gravado em uma linha do arquivo e os valores devem ser separados por um caractere de tabulação (ASCII 9 ou '\t').



2. Escreva outro programa que grave os valores calculados em uma página HTML chamada **movimento.htm**. A página deve apresentar os dados dentro de uma tabela. Lembre que 1800 rpm = 30 voltas por segundo. Usando o BrOffice.org Calc, produza um gráfico do movimento do carro no período de tempo considerado pelo seu programa.
3. Escreva um programa que gere um gráfico da função x^3, de forma que fique com um aspecto parecido com a figura a seguir. O gráfico deve ser gerado em um arquivo denominado **grafico.txt**. No lugar da linha azul você deverá usar o símbolo *****. A variável *x* deve variar de -8 a 8, em intervalos 2 décimos. *X* varia de cima para baixo no arquivo gerado. Cada linha do arquivo gerado terá no máximo 80 colunas. Os eixos também deverão ser traçados. A dificuldade neste exercício está em ter de fazer tudo ao mesmo tempo, ou seja, traçar a função, desenhar os eixos e escrever os valores junto aos eixos nos lugares corretos só pode ser realizado de forma contínua. Não há como retornar e fazer cada parte em separado. Seu código deve ser parametrizado, de tal forma que seja possível traçar o gráfico de qualquer função solicitada, por exemplo, $x^3 - x^5$. Se precisar, digite a função, por exemplo, x^3 - x^5 na busca do Google para ter uma noção de como o gráfico ficará.



7 Terminamos

Terminamos por aqui. Clique no menu Arquivo, depois clique na opção Sair. Corra para o próximo tutorial.