INF1301 Programação Modular
Período: 2015-1
Prof. Flavio Bevilacqua
1o. Trabalho

Data de divulgação: *9 de março (segunda-feira)*
Data de entrega: *25 de março (quarta-feira)*

1. Descrição do trabalho inicial

O objetivo deste primeiro trabalho é ajudar o aluno a familiarizar-se com os conceitos básicos da modularização de programas e da utilização do arcabouço de teste automatizado.

2. Descrição do primeiro trabalho

O arcabouço de teste automatizado presente no site da disciplina (www.inf.puc-rio.br/~inf1301, aba `software`) possui um exemplo de projeto utilizando uma estrutura de árvore binária que armazena caracteres. Crie um diretório no qual você deseja instalar o arcabouço. Descompacte o arquivo `arcaboucoteste-2-02.zip` assegurando que sejam gerados os sub-diretórios contidos no arquivo `.zip`. Isto criará toda a estrutura de diretórios utilizada e copiará tanto o arcabouço como os exemplos de uso. São copiados também os *scripts* de construção (`.make`) e teste do próprio arcabouço. Você encontrará a documentação nesta estrutura de diretórios.

O projeto a ser utilizado neste trabalho está contido na pasta "`Simples`". Crie no Visual Studio uma nova *workspace* e inclua os arquivos `.c` e `.h` deste projeto para que possa trabalhar no ambiente. Você agora terá à disposição duas formas de trabalhar, uma através do Visual Studio e outra através de janelas de comando (programa CMD do Windows)

Como este trabalho é para você se familiarizar com as ferramentas e métodos de trabalho a serem utilizados no restante do período, é imprescindível que você consiga executar o projeto, rodar o programa utilizando o *script* fornecido e veja os resultados alcançados. Os passos a serem seguidos são:

1. Execute o projeto `Arvore` tal como se encontra no arcabouço. Nesta etapa você precisará configurar corretamente o arcabouço em sua máquina e somente deve seguir para o passo adiante caso consiga executar o programa a partir do *script* de teste e gerar os resultados no *log* de saída. Procure fazê-lo usando tanto a janela de comando como o Visual Studio. Dica: o Visual Studio vem configurado para "*precompiled headers*". Desligue isto! Como? Percorra a configuração do projeto (*properties*).

2. Pode ser que a compilação não seja bem sucedida. Em geral isto se deve a uma incompatibilidade da versão da biblioteca (`ArcaboucoTeste.lib`) e a versão do ambiente que você está utilizando. Estude o documento `Leia.me` que vem junto com o arcabouço para solucionar este problema, i.e. compilar uma versão compatível.

3. Gere uma versão modificada do *script* "forçando" erros na estrutura de modo que o *log* de saída aponte as falhas. Neste novo script alterado pelo grupo, deverão aparecer erros propositais e o grupo deverá acrescentar comentários em cada linha alterada de forma que evidencie que o erro foi proposital. Esta execução deverá apresentar um *log* com erros.

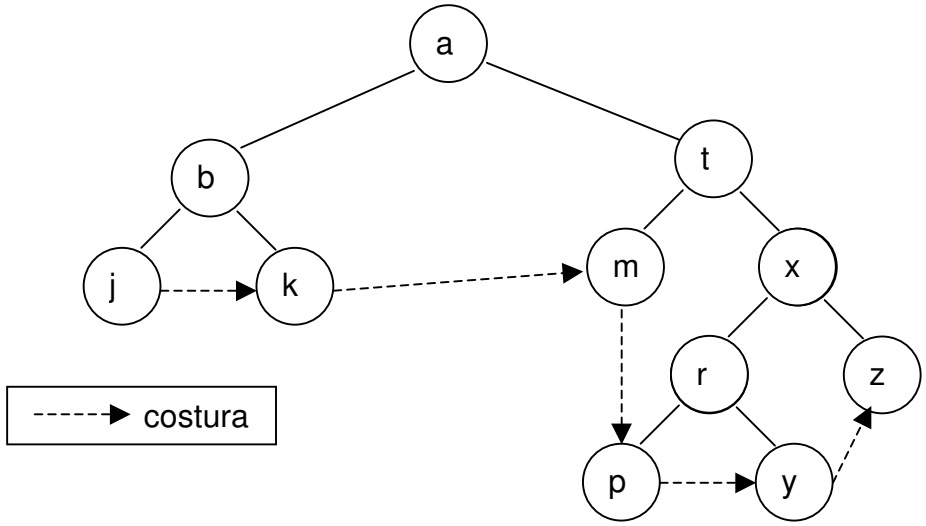
4. Gere uma versão modificada do script "forçando" erros e depois recuperando-os. O *log* de saída deverá aparecer com zero erros e cada linha de script alterada deverá ter um comentário do grupo informando o que foi feito.

5. Crie uma função no módulo `Arvore` que “costure” todas as folhas das árvores seguindo uma ordem alfabética. O grupo deverá criar um campo adicional (ponteiro) em cada nó para que realize esta “costura”. O grupo também deverá criar um campo Chave (alfanumérico) a ser utilizado na costura em ordem.

6. Crie uma função que apresente a sequência de caracteres da “costura” com “printf”. Esta função deverá percorrer a árvore através dos ponteiros deste novo campo

7. Crie um ou mais scripts para testar a função da costura e a função de apresentação da sequência.

Segue abaixo o exemplo de uma “costura” respeitando a ordem alfabética das folhas.



3. Entrega do Trabalho

O trabalho deve ser feito em grupos de dois ou três alunos. Os programas devem ser redigidos em "C". Não será aceita nenhuma outra linguagem de programação. Todos os programas devem estar em conformidade com os padrões dos apêndices de 1 a 10 do livro-texto da disciplina. Em particular, os módulos e funções devem estar devidamente especificados.

Recomenda-se fortemente a leitura do exemplo e do texto explanatório do arcabouço. Além de mostrar como implementar um teste automatizado dirigido por *script*, ilustra também as características de um programa desenvolvido conforme os padrões do livro.

O trabalho deve ser enviado por e-mail em um único arquivo `.zip` (codificação do *attachment*: MIME). Veja os *Critérios de Correção de Trabalhos* contidos na página da disciplina para uma explicação de como entregar.

O arquivo ZIP deverá conter:

- os arquivos-fonte dos diversos módulos que compõem os programas.
- os arquivos de *script* de teste desenvolvidos pelo grupo,
- os arquivos `.log` contendo os relatórios de execução dos testes
- os programas executáveis: **TRAB1-1.EXE, TRAB1-2.EXE**, **TRAB1-3.EXE**, e assim por diante que implementam as soluções requeridas.
- um arquivo **LEIAME.TXT** contendo a explicação de como utilizar o(s) programa(s).

- Tantos arquivos **RELATORIO-nome.TXT** quantos forem os membros do grupo. O tema **nome** deve identificar o membro do grupo ao qual se refere o relatório. Estes arquivos devem conter uma tabela de registro de trabalho organizada como a seguir:

Data, Horas Trabalhadas, Tipo Tarefa, Descrição da Tarefa Realizada

Na descrição da tarefa redija uma explicação breve sobre o que o componente do grupo fez. Esta descrição deve estar de acordo com o Tipo Tarefa. Cada Tipo Tarefa identifica uma natureza de atividade que deverá ser discriminada explicitamente, mesmo que, durante uma mesma sessão de trabalho tenham sido realizadas diversas tarefas. Os tipos de tarefa são:

- estudar
- especificar os módulos
- especificar as funções
- revisar especificações
- projetar
- revisar projetos
- codificar
- revisar código
- planejar teste
- revisar plano de teste
- testar
- corrigir

Dica: Preencha esta tabela de atividades ao longo do processo. **NÃO DEIXE PARA ÚLTIMA HORA, POIS VOCÊ NÃO SE LEMBRARÁ DO QUE FEZ TAL DIA, TAL HORA**. Com relatórios similares a esse você aprende a planejar o seu trabalho.

Importante: Ao entregar o arquivo ZIP, este **SOMENTE DEVERÁ CONTER OS ARQUIVOS RELACIONADOS A ESTE TRABALHO**. Caso existam outros arquivos incluídos ao compactar toda pasta do arcabouço e enviar (por exemplo: os arquivos **.bak**, os de trabalho criados pelo ambiente de desenvolvimento usado, etc.) o grupo **perderá 2 pontos**. GASTE UM TEMPO SELECIONANDO TODOS OS ARQUIVOS QUE VOCÊ EFETIVAMENTE PRECISA ENVIAR PARA PODER INSPECIONAR, MODIFICAR, USAR E RECOMPILAR OS PROGRAMAS ENTREGUES.

Observações:

- A mensagem de encaminhamento deve ter o assunto (*subject*) **INF1301-Trab01-*idGrupo*** conforme o caso. O ***idGrupo*** deve ser formado pelas iniciais dos nomes dos membros do grupo. O texto da mensagem deve conter somente a lista de alunos que compõem o grupo (formato: número de matrícula, nome e endereço do e-mail). Perde-se **2 pontos** caso não seja encaminhado desta forma. Mais detalhes podem ser encontrados no documento *Critérios de Correção dos Trabalhos* disponível na página da disciplina.

- O programa será testado utilizando o programa compilado fornecido. Deve rodar sem requerer bibliotecas ou programas complementares. O sistema operacional utilizado durante os testes será o Windows 7.

4. Critérios de correção básicos

Leia atentamente o documento *Critérios de Correção dos Trabalhos* disponível na página da disciplina. Muitas das causas para a perda substancial de pontos decorrem meramente da falta de cuidado ao entregar o trabalho. Este documento também pode ser encontrado no anexo à descrição da disciplina distribuída na primeira aula.

Não deixem para a última hora.
Este trabalho dá trabalho!