INF1301 Programação Modular
Período: 2018-2
Prof. Flavio Bevilacqua

1o. Trabalho

Data de divulgação: 15 de Agosto de 2018

Data de entrega: 05 de Setembro de 2018

1. Descrição do trabalho inicial

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

2. Descrição do primeiro trabalho

Neste trabalho deverá ser modificado o módulo exemplo **ARVORE** contido no diretório **Simples** do arcabouço de apoio ao teste automatizado. O arcabouço e alguns exemplos de seu uso estão no arquivo **arcaboucoteste-2-01.zip** que está disponível na aba **Software** da página da disciplina. Ao instalar (*dezipar*) este arquivo serão criados diversos diretórios. O módulo **ARVORE** e o respectivo módulo de teste específico (**TESTARV**) a que se refere o presente trabalho está no diretório **Simples**. A documentação do arcabouço encontra-se no diretório **Documentos**.

Inicialmente prepare-se para poder realizar o trabalho:

1. Baixe e instale ("*dezipe*") o arcabouço de apoio ao teste automatizado **arcaboucoteste-2-01.zip**. Este arquivo contém toda a documentação necessária e encontra-se na página da aba **Software**.
2. Ajuste a janela de linha de comando **CMD** de modo que tenha cerca de 50 linhas de pelo menos 100 caracteres cada (ideal 120). Ajuste o buffer da janela **CMD** para que tenha cerca de 500 linhas da mesma largura que a janela física.
3. Ajuste o arcabouço para que opere corretamente com a plataforma que você está utilizando, em especial a versão do compilador. Para isto é necessário regerar a biblioteca **ArcaboucoTeste.lib** contida no diretório **Arcabouc\Objetos**. Siga as instruções contidas no documento **ArcaboucoTeste-2-00-LeiaMe.pdf** que se encontra no diretório **Documentos** do arcabouço de apoio ao teste automatizado.
4. Estude o arcabouço, compile e teste o programa **ExemploSimples** assim como fornecido no diretório **Simples**. O objetivo disto é aprender a operar o arcabouço bem como ajustar as ferramentas de desenvolvimento a serem usadas, por exemplo o Visual Studio.
5. Estude o módulo de teste específico a ser alterado, bem como o módulo árvore a ser testado.

O objetivo do 1º. trabalho é modificar e testar o módulo **ARVORE** executando as seguintes tarefas:

- Modifique o módulo **ARVORE** de modo que possa operar com várias árvores ao mesmo tempo. Todas as árvores devem ter um elemento cabeça referenciado por uma elemento do tipo **ARV_tpArvore**. A referência para a cabeça identifica a árvore a ser manipulada. Cabe observar que o diretório **Instrum** contém uma solução para esta etapa do trabalho. Entretanto esta solução não interessa neste trabalho, uma vez que faz uso de uma série de características avançadas do arcabouço que serão utilizadas no 4º. Trabalho. Portanto, não copie simplesmente a solução. Se quiser use-a como fonte de inspiração, mas crie uma solução própria limitada aos requisitos do presente trabalho.
- Adicione a função **ARV_tpCondRet MarcarVisitado(ARV_tpArvore pArvore , ARV_tpModoVisita Modo)** que atribui o valor de **Modo** ao atributo **ModoVisita** no nó

corrente da árvore **pArvore**. Os modos de visita são pelo menos: **ARV_ModoDePai**, **ARV_ModoParaEsq**, **ARV_ModoParaDir**. Os modos de visita podem ser utilizados por funções de caminhamento para registrar o progresso da exploração. Note que em virtude do encapsulamento da estrutura da árvore através da cabeça deixou de ser possível o uso de caminhamento recursivo convencional. O caminhamento agora deverá ser realizado através da sucessiva atualização do modo de visita e do ponteiro para o nó corrente que se encontra na cabeça da árvore.

- Utilize a árvore binária para representar árvores *n-árias*. Nestas árvores a referência **pEsquerda** significa referência para o primeiro filho da lista de filhos e **pDireita** representa a referência para o irmão à direita.
  - O que precisaria ser alterado a mais para que se pudesse caminhar em ordem pela esquerda ou pela direita?
- Implemente a função **void ARV_ExibirArvore(ARV_tpArvore pArvore)**. Esta função deve percorrer a árvore n-ária em ordem prefixada pela esquerda.
  - Ao entrar no primeiro filho da lista de irmãos imprime **" ("**
  - Ao entrar em qualquer nó imprime o caractere do nó antecedido de um caractere em branco.
  - Ao sair da lista de irmãos imprime **")"**
  - Entrar na raiz de uma árvore corresponde a entrar em uma lista de irmãos
- Altere o módulo **TESTARV** de teste do novo módulo **ARVORE** de modo que você possa testar as alterações introduzidas. O módulo de teste deve ser capaz de operar simultaneamente com até 10 árvores.
- Será fornecido um (ou mais) *script* de teste para testar estas alterações. O módulo árvore deverá operar zero erros com esse *script*.
- Redija o seu próprio *script* de teste. Procure redigir um *script* suficientemente rigoroso.
- Atualize toda a documentação contida nos módulos de definição.
- Teste usando os *scripts* fornecido junto com o enunciado do trabalho.

Para facilitar o desenvolvimento do trabalho, siga a seguinte ordem:

1. Altere e teste o módulo para poder operar com várias árvores. Procure sempre utilizar o *script* fornecido, mesmo que ele venha a falhar em todos os pontos ainda não implementados. Se quiser você pode criar cópias parciais, retirando os casos de teste que não funcionam por você ainda não ter implementado a correspondente funcionalidade.
2. Altere e teste o módulo de modo que contenha a função de apoio à exploração. Explore pedaços de árvores usando comandos de teste que testem esta função.
3. Altere e teste o módulo de modo que contenha a função de impressão. Crie as árvores necessárias para testar esta função utilizando as funções já disponíveis no módulo.
4. Teste com o *script* de teste fornecido. Verifique se tudo está correto. Se estiver, empacote (i.e. crie o arquivo **.zip** de entrega do trabalho) verifique se esse arquivo está correto e completo e envie-o para o instrutor.

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 o programa.
- Os arquivos de *script* de teste desenvolvidos pelo grupo.
- Os arquivos *batch* (**`.bat`**) que coordenam a execução dos testes. (veja o exemplo **`testatudo.bat`** no arcabouço).
- O *script* de composição dos construtos (**`.comp`**) e os arquivos *script* de **`make`** gerados pelo utilitário **`GMAKE`** disponível no arcabouço de teste.
- O programa executável (construto): **`TRAB1-1.EXE`**. Caso queiram entregar mais de um executável, estes deverão ter o nome no formato: **`TRAB1-2.EXE`**, **`TRAB1-3.EXE`**, e assim por diante.
- 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 módulo
- ♦ revisar código do módulo
- ♦ redigir script de teste
- ♦ revisar script de teste
- ♦ realizar os testes
- ♦ diagnosticar e corrigir os problemas encontrados

Observações:

- **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**: O arquivo **`ZIP`**, DEVERÁ CONTER SOMENTE OS ARQUIVOS RELACIONADOS A ESTE TRABALHO. Caso o arquivo enviado contenha outros arquivos que os acima enumerados (por exemplo: toda pasta do arcabouço, arquivos **`.bak`**, arquivos de trabalho criados pelo ambiente de desenvolvimento usado, etc.) o grupo **perderá 2 pontos**. Gaste um pouco de tempo criando um *diretório de distribuição* e um **`.bat`** que copia do *diretório de*

desenvolvimento para este diretório de distribuição somente os arquivos que interessam. Verifique se esta cópia está realmente completa!

- A mensagem de encaminhamento deve ter o assunto (*subject*) **`INF1301-Trab01-idGrupo`**. O tema **`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 10`**. Assegure-se que a versão do programa entregue é uma versão de produção, ou seja, sem dados e controles requeridos pelo *debugger*.

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.

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