INF1301 Programação Modular
Período: 2015-2
Profs. Flavio Bevilacqua
4o. Trabalho
Data de divulgação: 4 de novembro (quarta-feira)
Data de entrega: 30 de novembro (segunda-feira)

ATENÇÃO !!!
NÃO VALERÁ PARA ESTE TRABALHO A REGRA DE PERDER UM PONTO POR DIA DE ATRASO. COM ISSO, QUEM ENTREGAR O TRABALHO APÓS AS 7:00 DO DIA 1/12/2015 (TERÇA) FICARÁ COM NOTA ZERO NESTE TRABALHO.

1. Descrição do quarto trabalho

A estrutura utilizada neste trabalho será a Lista Duplamente Encadeada Genérica.

Tarefas do 4o. trabalho:

- Baixe e estude a documentação e os exemplos de código do arcabouço de teste. Estude os módulos CESPDIN e CONTA e o conteúdo da pasta “instrum”.

- A estrutura auto-verificável, a ser desenvolvida neste trabalho, deve encadear todos os espaços alocados em uma lista de espaços alocados. Veja a descrição e as funções contidas no módulo CESPDIN, utilizado para o controle de acesso a espaços de dados dinâmicos.

- Para verificar a estrutura como um todo deve ser percorrida a lista de espaços alocados. Os espaços alocados pelo módulo Lista devem ser verificados.

- Transforme o modelo físico da Lista Duplamente Encadeada Genérica para estrutura auto-verificável. No desenho do novo modelo todos os elementos (atributos e relacionamentos) redundantes devem estar em vermelho, enquanto que os demais estarão em preto.

- Reveja e complete as assertivas estruturais do conjunto de módulos de modo que se tornem consistentes com o modelo auto-verificável.

- Altere o módulo Lista de modo que passe a conter funções de verificação e de deturpação, implementando as assertivas da estrutura auto-verificável.

- verificador do módulo Lista deve conter contadores de passagem controlando cada aresta de decisão existente em seu código. Caso o verificador faça uso de funções de verificação, estas também devem conter contadores de passagem.

- O deturpador deve ser capaz de selecionar elementos específicos da estrutura de dados e torná-los não válidos segundo alguma das assertivas. Estas funções interpretam as ações do comando de teste =deturparlista identificadas mais adiante.

- Produza scripts de teste capazes de examinar o funcionamento do verificador do módulo Lista. Nestes scripts os verificadores devem ser testados segundo o critério de completeza cobertura de arestas. Ao final do teste nenhum contador deverá conter zero. Ou seja, o teste deve ser completo segundo o critério cobertura de arestas, sendo que este critério será controlado pela instrumentação inserida.

- Controle o vazamento de memória. O controle de vazamento de memória deve ser realizado com os comandos disponibilizados no arcabouço de teste.

- Adicione comandos de teste ao módulo de teste específico de modo que o deturpador possa ser utilizado. Para utilizar esse comando será necessário posicionar o elemento corrente no nó a ser deturpado para, depois, adulterar o seu conteúdo. Considere a lista de ações de deturpação a seguir:

=deturparlista <ação>

<ação> = 1 elimina o elemento corrente da lista.

<ação> = 2 atribui NULL ao ponteiro para o próximo nó.

<ação> = 3 atribui NULL ao ponteiro para o nó anterior.

<ação> = 4 atribui lixo ao ponteiro para o próximo nó

<ação> = 5 atribui lixo ao ponteiro o nó anterior.

<ação> = 6 atribui NULL ao ponteiro para o conteúdo do nó.

<ação> = 7 altera o tipo de estrutura apontado no nó.

<ação> = 8 desencadeia nó sem liberá-lo com free

<ação> = 9 atribui NULL ao ponteiro corrente

<ação> = 10 atribui NULL ao ponteiro de origem.

=verificar <número de falhas esperado>

Verifica a estrutura de dados Lista como um todo, percorrendo a lista de espaços alocados. O verificador deve listar todas as falhas estruturais que for capaz de encontrar, retornando o número total de falhas encontradas (0 => não encontrou falhas). É possível que, para determinadas deturpações, a função “voe”. Coloque um comentário antes do comando =verificar no script de teste alertando o fato. Antes dos casos de teste que possam “voar” convém terminar e reinicializar o módulo Conta de modo que a totalização de contagem dos testes realizados até o ponto de cancelamento não seja perdida.

- Produza os scripts de teste de modo que o conjunto de casos teste percorra todas as arestas do código do verificador. Mostre que isto de fato ocorreu através de contadores inseridos no verificador. Caso um determinado script “voe”, particione-o em vários de modo que cada

condição que leve a um cancelamento seja testada por um script individual. Explique a razão para o script voar.

- O programa deve testar completamente a função de verificação sem deturpações e posteriormente, em outros scripts, utilizando deturpações.

- Produza um documento (.txt, .doc, ou .pdf) descrevendo cada comando de teste disponibilizado para testar o verificador.

2. 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:

- Um arquivo (.doc, .ppt ou .pdf) contendo o modelo do Lista genérica auto-verificável e as correspondentes assertivas estruturais.
- Os arquivos fonte dos diversos módulos que compõem o programa.
- Os programas executáveis: TRAB4-i.EXE, em que i é um identificador ordinal do programa. Um dos programas executáveis deverá ser com a instrumentação ligada e o outro deverá ser com a instrumentação desligada.
- Todos os scripts de teste. Em virtude de o deturpador poder deixar as estruturas em estado incorreto, ou até "voar", é possível que sejam necessários vários scripts de teste. Nos testes que envolvam deturpação, o caso de teste que faça o programa "voar" deve ser assinalado indicando o porquê do problema. Neste caso, o programa "voar" não será considerado falha, uma vez que era esperado ocorrer.
- O arquivo de declaração dos contadores.
- Arquivo de totalização dos contadores. O script de teste utilizado para verificar estruturas corretas deve conter um comando que zere todos os contadores. Dessa forma a seqüência de teste corretamente totalizará as contagens, mesmo que o teste seja efetuado repetidas vezes.
- Um arquivo batch (.bat) que encadeia os testes a serem realizados e finaliza imprimindo o conteúdo do arquivo de totalização de contadores.
- Um arquivo LEIAME.TXT contendo a explicação de como utilizar os programas e os scripts de teste.
- 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-Trab04-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 7**. Assegure-se que a versão do programa entregue é uma versão de produção, ou seja, sem dados e controles requeridos pelo debugger (versão release). **CABE RESSALTAR QUE SE O PROGRAMA FOR EXECUTADO PELO PROFESSOR E NECESSITAR DE BIBLIOTECAS ADICIONAIS DESNECESSÁRIAS, O TRABALHO PERDERÁ 8 PONTOS MESMO TENDO RODADO NA MÁQUINA DOS COMPONENTES DO GRUPO. SUGESTÃO: ENVIE UM EXECUTÁVEL ANTES DO DIA DA ENTREGA PARA SER TESTADO NA MÁQUINA DO PROFESSOR.**

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!