INF1413 Teste de Software
Período: 2017-1
Profs. Arndt von Staa

4o. Trabalho

Data de divulgação: **09 de junho (sexta-feira)**

Data de entrega: **21 de junho (quarta-feira)**

1. Descrição do trabalho

O trabalho visa exercitar a remoção de defeitos. A segunda parte visa estimar a eficácia do teste através da inserção de mutantes.

2. Descrição do quarto trabalho

2.1. Preparação:

1. Em anexo encontra-se um arquivo **`.zip`** que contém uma extração do arcabouço de teste Talisman. Descompacte o seu conteúdo para algum diretório, por exemplo: Trabalho4.
2. No pacote existe um diretório **`Docs`**. Este diretório contém uma documentação simples do arcabouço de teste e instruções para inicializar as variáveis de ambiente requeridas pelo compilador C++ do Visual Studio. Siga as instruções deste documento.
3. É necessário um compilador C++ – o pacote está configurado para o Visual Studio 2010 Professional. O pacote faz uso da biblioteca **`TalismanTestLib.lib`**. É possível que essa biblioteca venha a ser incompatível com o compilador que você esteja usando. Neste caso o remédio é utilizar um computador do laboratório dos cursos de graduação. Infelizmente não há solução melhor.
4. Ajuste a janela **`CMD`** para que tenha pelo menos 40 linhas e 80 colunas. Ao clicar com a tecla da direita do mouse abre-se um pop up que contém **`Properties`**. Ajuste a altura do buffer (número de linhas dele) para algo em torno de 2000, e o tamanho e a largura da janela. A largura do buffer deve ser igual à largura da janela.
5. No diretório **`...\Test\Batches`** existe um *batch file* prepara**`.bat`**. Execute este *batch file*. Serão excluídos todos os arquivos que virão a ser gerados. Serão gerados todos os *make files* necessário. Alguns avisos de "arquivo faltando" serão exibidos. Não tem problema, eles serão gerados durante a execução do *make file*. A seguir são compilados todos os módulos que constituem o programa **`tst-str`** . Estes módulos são ligados com a biblioteca **`TalismanTestLib.lib`** gerando o executável **`tst-str.exe`**. Não deveriam ocorrer erros de compilação. Ao terminar a compilação é aberta uma janela do **`NotePad`** contendo o log da compilação. Ao fechar a janela do **`NotePad`**, o programa **`tst-str.exe`** é executado com o script de teste **`tst-str-01.script`**. Não deveriam ocorrer erros de execução.
6. Compile agora o programa que servirá de cobaia para o trabalho.
    1. execute: g**`enmake tst-xmsg`** - vão aparecer algumas mensagens dizendo que alguns arquivos **`.inc`** ainda não existem, eles serão gerados ao compilar.
    2. execute: **`compile tst-xmsg`** - vai compilar os módulos adicionais que perfazem o trabalho. A seguir eles serão ligados com a biblioteca, gerando o programa **`tst-xmsg.exe`**. Não deveriam ocorrer erros de compilação.

3. execute: **`test xmsg 01 c`** - o programa vai ser executado, vão aparecer diversos erros de execução. O *batch file* **`test`** irá executar o programa **`tst-xmsg.exe`** usando o script **`tst-xmsg.script`**. Com existe o parâmetro "**`c`**" a execução efetuará a contagem de passagens. O arquivo que contém os nomes dos contadores é **`tst-xmsg.count`**.
4. Os passos 6.2 e 6.3 deverão ser repetidos durante a realização do trabalho. Se for necessário regerar o **`makefile`**, execute o passo 6.1. Se for necessário limpar o ambiente execute: **`cleanall`**

2.2. Trabalho:

O pacote contém os módulos **`XMESSAGE`**, **`XMSGBCD`**, **`XMSGBIN`**, **`XMSGSTR`**, **`XMSGTIME`**, e **`TST_XMSG.`** Além disso, estão disponíveis os arquivos **`XMESSAGE.COUNT`** que contém a declaração de todos os nomes de contadores de passagem já inseridos, e o arquivo **`tst-xmsg.script`** que contém o *script* de teste. Foram inseridos diversos defeitos nesses arquivos, sem marcação e sem que isso afete a compilação. Existem defeitos inseridos no script de teste. Exceto a correção desses defeitos e eventuais necessidade de adição de casos de teste, o script não deverá ser alterado. Todas as alterações deve estar devidamente assinaladas.

1. (**7 pontos**) Procure e elimine todos os defeitos até que o resultado do teste se torne OK. OBS. eu tenho outro log de teste para verificar se efetivamente foi corrigido. Os defeitos podem estar em qualquer um dos arquivos exceto nos arquivos .hpp Ao fazer isso redija, para cada defeito eliminado, um texto contendo:
    a. número da falha a ser eliminada, servirá como um id.
    b. descrição da falha observada
    c. estratégia a ser adotada para diagnosticar
    d. como foi realizada diagnose, descreva inclusive todos os becos sem saída e mudanças de estratégia possivelmente atingidos
    e. ao corrigir, referencie as linhas inserindo um comentário /* num ======= */ em que num é o número da falha.
    f. resultado. Mostre que o defeito efetivamente sanou a falha observada pelo teste.
2. (**3 pontos**) Crie três operadores de mutação e use-os para criar três mutantes no método **`AssembleMessage(...)`**. Documente os mutantes criados, indicando o operador de mutação e o local em que foi aplicado. Execute o teste TST-XMSG-01.script . Se necessário corrija o script de teste para assegurar que os três mutantes sejam mortos. É possível que o script de teste fornecido mate todos os mutantes. Obs. Caso o programa cancele ao executar, assegure-se que a causa é um mutante. Se for, evidentemente o mutante estará morto. Documente isso. Caso o programa não cancele, mas produza um resultado incorreto, verifique se o resultado incorreto é provocado por um dos mutantes inseridos. Caso não seja, identifique o porquê, documente e faça as correções necessárias para poder continuar a "matar" mutantes. Não se esqueça de adicionar ao relatório do trabalho a descrição dos mutantes inseridos e qual o caso de teste que os descobriu.

3. Entrega do Trabalho

O trabalho deve ser realizado em grupos de 2 ou 3 alunos.

Devem ser entregues os **relatórios, os códigos fonte (.cpp e .hpp), o script de teste final, e os executáveis finais (.exe)**, e, ainda, os resultados (logs) de teste gerados no decorrer da elaboração de cada um dos passos do trabalho.

Além dos arquivos acima, deve ser entregue um relatório de tempo despendido no trabalho. O relatório consta de uma tabela de registro de trabalho do grupo organizada como a seguir:

Data	Horas Trabalhadas	Quem trabalhou	Tipo Tarefa	Descrição da Tarefa Realizada

A coluna *Quem trabalhou* identifica 1 ou mais membros do grupo que realizaram a tarefa. Na descrição da tarefa redija uma explicação breve sobre o que foi feito. 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. **Crie e justifique a lista de naturezas de tarefas**.

4. Critérios de correção básicos

- Clareza da exposição contida no relatório.
- Completeza do relatório.
- Aderência às exigências de confecção do relatório.
- Obediência aos critérios de correção de trabalhos que estão anexados à descrição da disciplina.