INF1301 Programação Modular
Período: 2017-1
Prof. Alessandro Garcia
4o. Trabalho
Data de divulgação: 30 de maio (terça-feira)
Data de entrega: 26 de junho (segunda-feira)

1. Descrição do trabalho do período

O objetivo do trabalho do período é implementar um jogo de paciência Free Cell. Neste jogo as cartas de um baralho começam divididas em oito sequências embaralhadas com os valores e naipes visíveis. A medida que o jogo se desenrola, cartas são passadas de uma sequência para outra seguindo determinadas regras. Existem mais quatro espaços extras (Extra1, Extra2, Extra3, Extra4) para colocação de até uma carta e mais quatro espaços de sequências ordenadas (Naipe1, Naipe2, Naipe3 e Naipe4) destinadas às sequências ordenadas, uma para cada naipe. Ao longo do período, o grupo deverá implementar um jogo de paciência completo e operacional levando em conta todas as regras existentes. No quarto trabalho (T4) será concluída a implementação do jogo Free Cell, bem como adicionado instrumentações.

2. Descrição do quarto trabalho

Primeiramente, neste quarto trabalho deve ser concluída a implementação dos módulos do jogo Free Cell. Além disso, no presente trabalho, o módulo Lista será modificado para conter instrumentação e, a seguir, será testada a capacidade da instrumentação de identificar, em tempo de execução, possíveis falhas na estrutura de dados. O teste da instrumentação será automatizado. Deve ser utilizada a versão completa do arcabouço de teste. Esta, em adição à versão simplificada, possui o módulo **CONTA** para controle da cobertura de testes e o módulo **CESPDIN** para controle de problemas associados com alocação dinâmica de memória. Os *scripts* de teste a serem desenvolvidos neste trabalho devem controlar a cobertura e faltas associadas com alocação dinâmica de memória. Estude o exemplo contido na pasta "*Instrum*" do arcabouço.

Tarefas do 4o. trabalho:

- antes de mais nada, certifique-se que os problemas do trabalho T2 foram corrigidos.
- conclua a implementação do jogo Free Cell através da implementação dos módulos que não foram codificados no trabalho T2. A implementação deve estar aderente a especificação de requisitos, do modelo da arquitetura, do modelo físico, todos já entregues no T2.
- entregue novamente as especificações dos requisitos, o modelo da arquitetura e o modelo físico, reportando e justificando quaisquer modificações que foram necessárias em tais documentos após a entrega do T2.
- produza um documento (.txt, .doc, ou .pdf) que descreva como testar, passo a passo, o jogo como um todo, simulando a participação de um jogador usando o programa. Deve ser ilustrado, passo a passo, quais comandos devem ser acionados na interface para realização dos testes do programa via interface com o usuário.
- baixe e estude a documentação e os exemplos de código do arcabouço de teste. Estude os módulos **CESPDIN** e **CONTA**. Veja a descrição e as funções contidas no módulo **CESPDIN**, utilizado para o controle de acesso a espaços de dados dinâmicos. Você deve utilizar as funções do CESPDIN no código do seu programa para realizar o controle de acesso a espaços de dados dinâmicos.
- produza *scripts* de teste capazes de examinar o funcionamento do módulo Lista segundo o critério de completeza ***cobertura de arestas***. O módulo **CONTA** deve ser utilizado para a análise automatizada deste critério. Ao final do teste, nenhum dos contadores criados para o módulo Lista 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 uso de memória utilizando o módulo CESPDIN. O controle de uso de memória deve ser realizado com os comandos disponibilizados no arcabouço de teste.
- apresente o modelo físico (Aula 17) da estrutura Lista Genérica, disponibilizada pelo módulo LISTA. Esta estrutura deve ser transformada em uma estrutura *auto-verificável,* ou seja, a estrutura deve conter redundâncias para que seja possível verificar e corrigir erros estruturais durante a execução do programa. No desenho do modelo físico, todos os elementos (atributos e relacionamentos) redundantes devem estar em vermelho, enquanto que os demais estarão em preto. Implemente as assertivas estruturais do módulo Lista de modo que se tornem consistentes com o modelo físico auto-verificável. Altere o módulo Lista de modo que passe a conter funções de verificação e deturpação, assim possibilitando a verificação e teste das assertivas da estrutura auto-verificável. Para verificar a estrutura como um todo deve ser percorrida a lista de espaços alocados. Os espaços alocados pelo módulo Lista devem ser verificados. A estrutura de Lista auto-verificável, a ser desenvolvida neste trabalho, deve encadear todos os espaços alocados em uma lista de espaços alocados através do uso do CESPDIN.
- as funções verificadoras do módulo Lista também devem conter contadores de passagem controlando cada aresta de decisão existente em seu código. Caso o verificador faça uso de outras 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. **Pelo menos, 3 tipos de deturpações** devem ser implementadas. Estas funções interpretam as ações do comando de teste **=deturparlista** identificadas mais adiante.
- 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. A lista (incompleta) de possíveis ações de deturpação são:

  =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.

  outros outras ações que vocês identificarem necessárias para testar completamente o verificador

  =verificar <número de falhas esperado>

- 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 as funções 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.

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. Todos os módulos, funções, structs, e variáveis devem ser devidamente especificados segundo o padrão.

O trabalho deve ser enviado por e-mail em um único arquivo **.ZIP** (codificação do *attachment*: MIME). O arquivo **ZIP** deve conter:

- os arquivos de documentação de requisitos, modelos arquitetural e físico, bem como relato e justificativas das alterações feitas após o T2, caso foram necessárias.
- 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. Lembrar que é necessário ter um executável por módulo e um executável para o programa FreeCell todo. No caso do módulo LISTA, deve haver um executável para o teste sem instrumentação e um executável para cada teste do módulo com instrumentação e deturpação. Um dos programas executáveis deverá ser com a instrumentação ligada e o outro deverá ser com a instrumentação desligada.
- os arquivos **.COMP** necessários para geração dos arquivos **MAKE**.
- os arquivos **MAKE** e, se for o caso, **BUILD** necessários para gerar os programas executáveis.
- 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.
- arquivos *batch* (**.bat**) para compilar, testar e demais funcionalidades, assim como aqueles exemplos disponíveis no arcabouço; um dos arquivos *batch* deve encadear todos 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.
- um arquivo **LinguagemTeste.txt** (ou .doc, ou .pdf) contendo a documentação da linguagem de *scripts* de teste.
- tantos arquivos **RELATORnome.TXT** quantos forem os membros do grupo. O elemento nome deve identificar o membro do grupo. Estes arquivos devem conter uma tabela de registro de trabalho organizada conforme descrito nos enunciados de trabalho anteriores.

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 o relatório com a 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 completa!
- A mensagem de encaminhamento deve ter o assunto (*subject*) **INF1301-Trab04-idGrupo** ou **INF1628-Trab04-idGrupo** conforme o caso. 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**.

Observações:

- a mensagem de encaminhamento deve conter o assunto (*subject*) **INF1301 Trabalho 4 aaa-bbb-ccc** no qual
  - **aaa-bbb-ccc** identifica a composição do grupo (duas ou três letras por participante)
  - texto deve conter somente a lista dos alunos que compõem o grupo (formato: número de matrícula, nome e endereço e-mail).
- não deve ser acrescentado qualquer texto extra à mensagem.
- caso o instrutor não consiga ler o arquivo **zip** na sua máquina o grupo perde 1 ponto e terá a chance de resubmeter o trabalho em um arquivo correto.
- 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 Windows XP. O programa deve operar em uma janela CMD (DOS) sob este sistema operacional.

4. Critérios de correção básicos:

Leia atentamente a folha de critérios de correção entregue no primeiro dia de aula. Esta folha encontra-se em formato pdf, na página Web da disciplina.

São verificados ainda:

- modelos e assertivas não existem **–3**, assertivas não existem ou excessivamente incorretas **–2**; modelos e/ou assertivas com algumas incorreções **–1** ponto.
- documentação da linguagem *script*, não existe **-2**; inconsistente com o que está implementado **-2**; incompleta **-1**
- teste não é completo **–2**
- teste não utiliza contadores **–2**
- teste não controla vazamento de memória **–2**

Nos testes que envolvam deturpação os casos de teste que podem fazer o programa "voar" devem ser devidamente assinalados. Neste caso, o programa "voar" não será considerado falha, uma vez que era esperado ocorrer.

- 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:

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