INF1301 Programação Modular
Período: 2018-1
Prof. Alessandro Garcia
4o. Trabalho
Data de divulgação: 25 de maio (sexta-feira)
Data de entrega: 29 de junho (sexta-feira)

1. Descrição do trabalho do período

O objetivo do trabalho do período é especificar, projetar e implementar o jogo Truco. Neste jogo, dois, quatro ou seis jogadores são divididos em duas equipes, utilizando-se um baralho de **40 cartas**. Em cada **"mão"**, cada jogador recebe três cartas. Cada mão pode ter até 3 rodadas e pode valer 1, 3, 6 ou 12 pontos. A partida termina quando uma das equipes somar 12 pontos. Ao longo do período, o grupo deverá desenvolver o jogo Truco completo e operacional, levando em conta todas as regras existentes. No trabalho T2, o objetivo foi especificar os requisitos com base nas regras do jogo Truco, criar a arquitetura e modelo físico do programa, implementar e/ou testar alguns módulos básicos do jogo. Os módulos desenvolvidos foram testados individualmente. Os outros módulos (por exemplo, o módulo principal que administra todos os outros), especificados na arquitetura do grupo e permitem jogar uma partida completa serão desenvolvidos no trabalho T4. Além disso, no T4, também serão incorporados instrumentos aos módulos do programa de forma a tornar o programa de Truco mais confiável.

2. Descrição do trabalho T4

Primeiramente, neste quarto trabalho deve ser concluída a implementação e teste de **todos os módulos do jogo Truco**, de tal forma que seja possível a realização de partidas entre jogadores. Devem ser adicionadas assertivas de entrada e saída, pelo menos, nos módulos BARALHO e LISTA. Além disso, no presente trabalho, os módulos LISTA e BARALHO serão modificados para conter instrumentação que apoia a análise da cobertura de testes (através do uso do módulo CONTA do arcabouço). Esta análise será feita através de scripts com comandos associados com o módulo CONTA. Deve ser utilizada a versão completa do arcabouço de teste; esta versão, em adição à versão simplificada, possui o módulo **CONTA** para controle da cobertura de testes. Os *scripts* de teste a serem desenvolvidos neste trabalho devem satisfazer a cobertura por **arestas** (conforme descrito abaixo) dos módulos LISTA e BARALHO. Estude o exemplo contido na pasta "*Instrum*" do arcabouço.

Lista detalhada das tarefas do 4o. trabalho:

- conclua a implementação do progama do jogo Truco 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. Estes documentos devem ser re-entregues no T4. Quando houver necessidade de alterar a especificação e modelos, as alterações podem ser feitas. Entregue novamente as especificações dos requisitos, o modelo da arquitetura e o modelo físico, reportando as modificações feitas.
- para possibilitar o teste do programa de Truco, forneça um documento que guie um testador a testar o programa via Interface, de forma que possa ser certificado que o jogo está funcionando por completo. Esse documento pode ser redigido em um arquivo Word ou .txt. Importante lembrar que este documento é adicional aos scripts de teste individual de cada módulo.

- inclua novas assertivas de entrada e saída nas funções do módulo LISTA (se já não tiverem sido feitas no T2) e nas funções do módulo BARALHO.
- baixe e estude a documentação e os exemplos de código do arcabouço de teste. Estude o módulo **CONTA**.
- produza *scripts* de teste capazes de examinar o funcionamento dos **módulos LISTA e BARALHO,** segundo o critério de completeza ***cobertura de arestas***. Quando possível de serem exercitados pelos testes, devem ser também incluídos contadores no código de assertivas de entrada ou saída. O módulo **CONTA** deve ser utilizado para a análise automatizada deste critério. Ao final do teste nenhum dos contadores criados para os módulos **LISTA e BARALHO** deverá conter zero. Ou seja, o teste deve ser completo segundo o critério cobertura de arestas, sendo que este critério será controlado pelos contadores inseridos e os comandos de teste do módulo **CONTA** nos scripts.
- para implementação das assertivas, pode ser que os módulos façam uso de funções verificadoras (funções destinadas a implementar uma ou mais assertivas). Tais funções também devem ter conter contadores de passagem controlando cada aresta existente em seu código. Em resumo, caso o módulo faça uso de funções de verificação, estas também devem conter contadores de passagem. Para possibilitar o teste de algumas assertivas (sugere-se apenas para o módulo LISTA), pode ser feito uso de uma função deturpadora, que deve ser capaz de selecionar elementos específicos da estrutura de dados e torná-los não válidos segundo alguma das assertivas. Um máximo de 3 tipos de deturpações são suficiente. A função deturpadora interpreta as ações do comando de teste **=deturparlista** identificadas mais adiante. Para isso, 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 de ações de deturpação para assertivas do LISTA a seguir é incompleta:

  =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 associadas com as deturpações implementadas. Mostre que isto de fato ocorreu através de contadores inseridos no código das assertivas exercitadas pelas deturpações. Dica: 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 sem deturpações e posteriormente, em outros *scripts*, utilizando deturpações.

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 globais 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. Um dos programas executáveis deverá ser com a instrumentação ligada e o outro deverá ser com a instrumentação desligada.
- os *scripts* **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.
- 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.
- um arquivo **LinguagemTeste.txt** (ou .doc, ou .pdf) contendo a documentação da linguagem *script* 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 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. 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**. 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 XP**. Assegure-se que a versão do programa entregue é uma versão de produção, ou seja, sem dados e controles requeridos pelo *debugger*.

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!