Departamento de Ciência da Computação – Universidade de Brasília
Programação Sistemática – 1º/2015
Data de entrega: 09/07/2015, 08:00h.
Apresentação: 09/07/2015 – a partir de 09:00h.

Trabalho Prático – Parte 3

Introdução

A especificação para a terceira e última etapa do trabalho prático da disciplina de PS em 1/2015 consiste em implementar casos de teste das prévias etapas 1 e 2 do sistema para auxiliar na coleta das informações necessárias para o Sucupira a partir dos currículos Lattes dos professores da Pós Graduação em Informática (PPGI) do CIC.

Para todas as implementações das funções das interfaces de cada módulo (em Java ou em C) produzir primeiramente um grafo que representa o fluxo de execução de cada função, onde cada nó do grafo representa uma instrução do código. A partir do grafo gerado exercitar os critérios de seleção de **testes de caixa branca**: *instrução*, *aresta*, *decisão*, definindo casos de testes que garantam a cobertura de *todos os caminhos* para cada critério. Quando houver *repetições*, garantir que os casos de indução com base no critério do *arrasto* sejam cumpridas.

Segue logo abaixo a especificação para os módulos de tratamento de arquivos implementados em Java e para os módulos implementados em C.

Teste de arquivos externos

- O programa verifica se o arquivo está vazio e, ao invés de dar "segmentation fault" ele envia uma mensagem para o console do usuário informando "Arquivo <nome do arquivo.txt> vazio!". Para tanto, utiizar a função `ferror()`.
- Caso o arquivo contenha caracteres inválidos, o programa deve registrar em um respectivo arquivo de Log (Log-<nome-do-arquivo>.txt) os dados não processados, e prosseguir o processamento de leitura das demais informações do arquivo. Para os programas em C, utilizar tanto as funções `ferror()` e `fscanf()`. Para os programas em Java tratar os casos de `I/OException`, `NullPointerException` e `FileNotFoundException`.
- Implementar uma função que imprime os projetos e publicações de um professor para um ano dado como prompt ao usuário. Gerar um arquivo com apenas o nome de todos os projetos e outro para todas a publicações. Testar os casos em que essa função retorna corretamente os projetos e publicações para:
    - anos aleatoriamente escolhidos entre 2000 e 2015
    - anos antes de 1970 (CIC não havia sido criado)

    - anos após o ano corrente (2015)
    - Testar a geração de arquivos vazios: emiir a mensagem ”Não há publicações no período selecionado”, caso não constem publicações para o ano escolhido. E emitir ”Não há projetos no período selecionado”, caso não constem projetos para o ano escolhido. E não gerar arquivo vazio!
- Criar uma função que verifica se os arquivos gerados para todos os professores ao final do processamento (leitura e tratamento) corresponde ao nome dos professores do PPGI. A lista completa dos professores do PPGI está em http://ppgi.unb.br/curso/professors
- Criar uma função que verifica se o número de arquivos gerados para todos os professores ao final do processamento (leitura e tratamento) corresponde ao número de professores do PPGI.

Grafo de Relacionamento

- Testar que a lista de relacionamentos de cada professor contém apenas autores de artigo ou integrantes de projeto que são de fato professores do CIC.
- O grafo não pode ter auto-relacionamento: nome do próprio professor não pode constar na lista de professores com quem ele se relaciona em projeto ou publicação.
- O grafo não pode ter quantidade de nomes distintos de professores maior do que o número de professores do PPGI. Para tanto, utilize a função `strncmp()` para os primeiros 20 caracteres apenas.
    - Obs: para simplificar o teste do grafo de relacionamento pode-se gerar a lista de relacionamento a partir do nome completo dos professores, uma vez que existem vários nomes para citação.

Controle de qualidade das funcionalidades

A corretude do código deve ser assegurada utilizando as ferramentas de análise **estática e dinâmica**. Para a análise estática, utilize a ferramenta Splint (C) seguindo a análise para faltas de controle, armazenamento, interface e entrada e saída.

Para o teste, continue utilizando o CUnit. Depois de prontos os casos de tese, deve-se criar um módulo controlador de teste (disciplinado) usando o CUnit para testar se as principais funcionalidades e restrições dos módulos de armazenamento, tratamento e persistência dos dados atendem à especificação. O teste disciplinado deve seguir os seguintes passos:

1) Antes de testar, prepararem os módulos de implementação para tratar os casos solicitados no tratamento de arquivos externos e de grafos de relacionamento.

2) Produzir o grafo de cada função de cada módulo

3) Para cada critério de teste de caixa branca, estabelecer a massa de dados para exercitar cada critério.

4) Para o caso dos arquivos externos, estabelecer a massa de arquivos para exercitar cada caso de teste solicitado.

5) Antes de iniciar o teste: estabelecer o cenário do teste

6) Criar um módulo controlador de teste, usando a ferramenta CUnit para testar as principais funcionalidades de cada módulo.

7) Ao testar: produzir um laudo em que todas as discrepâncias encontradas são registradas. Esse laudo pode ser uma saída da execução do CUnit ou do JUnit. Somente termine o teste antes de completar o roteiro, caso observe que não vale mais a pena continuar executando o roteiro, uma vez que o contexto para o resto está danificado

Após a correção: repetir o teste a partir de 2 até o roteiro passar sem encontrar falhas.

Parte 2 - Escrita

Deverão ser entregues em conjunto com os códigos fonte:

- 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 casos de teste
- Revisar casos de teste
- Realizar os testes
- Diagnosticar e corrigir os problemas encontrados

Observações:
Dica: Preencha esta tabela de atividades ao longo do processo. Atividades de teste não são simples. 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.