Departamento de Ciência da Computação – Universidade de Brasília
CIC 113956 Programação Sistemática – 1º/2015
Data de entrega: 21/Setembro/2015, 23:59h

Trabalho Prático - Parte I

Introdução

Planejamento e organização são práticas indispensáveis no cotidiano das pessoas. Um estudante, em geral, consegue se organizar e gerenciar suas atividades com uma simples agenda e anotações. Porém, para eventos ou atividades de grande escala, esse tipo de organização não é eficaz, devido ao grande número de informações que estão envolvidos. Um bom exemplo é a distribuição de locais e horários de palestras ou disciplinas e seus respectivos palestrantes e professores em uma universidade.

Como programadores, estamos sempre interessados em resolver problemas. A especificação do trabalho prático da disciplina Programação Sistemática será a criação de um sistema para armazenagem, organização, distribuição de palestras pela UnB. O programa deve ser implementado em dois componentes: o primeiro deverá ler arquivos fornecidos, tratar seus dados e distribuí-los em estruturas convenientes para a criação de um calendário; o segundo é responsável pela gerência de uma agenda feita através dos dados e rodízio de palestrantes. Sempre aplicando **todos** os conceitos aprendidos durante o andamento da disciplina.

Componente que lê, trata e aloca os dados

Funcionamento do Componente:

Essa parte do sistema deverá receber, ler e distribuir os dados em suas respectivas estruturas. Os dados serão disponibilizados em arquivos de texto (.txt) e possuem todas as informações das palestras e palestrantes.

Estrutura do Arquivo 1 (Palestrantes.txt):

Nome: Palestrante1.
Disponibilidade: Ter, 01/09/2015, 10:00-12:00; Qua, 00/09/2015, 8:00-9:30.
Nome: Palestrante2.
Disponibilidade: Dom, 12:00-14:00; Qui, 03/09/2015, 7:00-9:30; Seg, 00/10/2015, 14:00-16:30.

Obs.: As datas representadas com dia 0 significam disponibilidade para o dia da semana citado para o mês todo.

Estrutura do Arquivo 2 (Palestras.txt):

Nome: NomePalestra1.
Palestrante: Palestrante1.
Tema: TemaPalestra1.
Local: Local1.
Duracao: 1:30h.

Nome: NomePalestra2.
Palestrante: Palestrante2.
Tema: TemaPalestra2.
Local: Local2
Duracao: 1:00h.

Note que cada campo representa um elemento da estrutura que será usada para armazenamento das informações dos palestrantes/palestras. Ou seja, é obrigatório o uso desses campos nos TADs do trabalho. Lembramos que o módulo que trata a localidade (campo Local no exemplo acima) será tratado na próxima especificação do trabalho.

Modularização

Para cada um dos módulos desse componente, deverá ser criado um módulo de declaração e um módulo de implementação, além de serem aplicadas as devidas regras de encapsulamento do código. Lembrando que cada módulo deverá ser possível a compilação independente.

Módulos Obrigatórios:

- Módulo de leitura:

Deve conter todas as funções para a leitura correta do arquivo, separando cada informação de maneira correta e conveniente.
Entrada: Arquivo com informações das palestras e palestrantes.
Saída: Conjuntos de dados organizados.

- Módulo de tratamento de dados:

Com os dados lidos, faz manipulações necessárias (p.e Tratamento de maiúsculas e minúsculas, conversão da hora lida para um tipo mais fácil para manipulação, etc.). Aloca e preenche as estruturas com essas informações.
Entrada: Saída do módulo de leitura.
Saída: Dados prontos para serem alocados nas estruturas.

- Módulo calendário:

Contém a codificação necessária para a criação e manipulação de uma estrutura calendário. Essa estrutura é o TAD com nível mais alto de abstração podendo depender de outras estruturas mais simples, que podem ser implementadas em outros módulos.
Entrada: Saída do módulo de tratamento de dados.
Saída: Estrutura calendário devidamente preenchida e pronta para utilização.

- Módulo de controle de tempo:

Como será necessário trabalhar com horas e datas, deve ser criado um módulo que manipula esse tipo de informação (somar, fazer consultas, etc.). Também, é importante que exista uma referência temporal atualizada e correta. Por isso, a biblioteca *time.h* ou java.util.Calendar deverão ser utilizadas nesse módulo, conforme a inguagem de programação escolhida.
Referência:
Versão C (time.h): http://www.cplusplus.com/reference/ctime/
Versão Java (java.util.Calendar):
http://docs.oracle.com/javase/7/docs/api/java/util/Calendar.html
Entrada: Informações de data e hora, consultas.

Saída: Resultado das operações.

- Módulo de persistência de dados:

Com as informações do calendário gerado, cria um arquivo texto. Deve ser um relatório organizado e claro, já que irá ser lido diretamente pelo usuário.

Entrada: Estrutura calendário preenchida.

Saída: Arquivo com as informações do calendário, visando à compreensão do usuário final. Esse arquivo deve ser organizado internamente de maneira cronológica. Também deve existir a opção de gerar somente o arquivo do mês selecionado.

Exemplo de arquivo de saída (CalendarioPalestras.txt):

Calendario de Palestras (05/2015 – 08/2015)

Mes 05/2015 – Maio

Dia 01
Palestra1 (Palestrante1): 8:00-10:00.
Palestra2 (Palestrante2): 14:00-15:30.
...
PalestraN (PalestranteN): HH:MM-HH:MM.

Dia 02
Palestra1 (Palestrante1): 9:00-10:55.
Palestra2 (Palestrante2): 13:00-14:40.
...
PalestraN (PalestranteN): HH:MM-HH:MM.

...

Mes 06/2015 – Junho
...

Controle de qualidade das funcionalidades

Deve-se criar um módulo controlador de teste (disciplinado) usando o CUnit (C) ou JUnit (Java) para testar se as principais funcionalidades e restrições dos módulos atendem a especificação. O teste disciplinado deve seguir os seguintes passos:

1. Antes de testar: produzir um roteiro de teste;
2. Antes de iniciar o teste: estabelecer o cenário do teste;
3. Criar um módulo controlador de teste, usando a ferramenta CUnit (C) ou JUnit (Java)para testar as principais funcionalidades de cada módulo;
4. Ao testar: produzir um laudo em que todas as discrepâncias encontradas são registradas. Esse laudo pode ser uma saída da execução do CUnit (C) ou JUnit (Java). 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;
5. Após a correção: repetir o teste a partir de 2 até o roteiro passar sem encontrar falhas.

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

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.

Entrega do trabalho:

1. O trabalho deve ser entregue com todos os arquivos necessários para a sua compilação, comprimidos em uma pasta (.zip).
2. No arquivo LEIAME.TXT, incluir toda a informação necessária para a compilação e, também, as informações do sistema em que o trabalho foi desenvolvido e compilado. (p.e. Desenvolvido e compilado no Ubuntu 64 bits...).
3. Lembre-se de aplicar os conceitos aprendidos na disciplina.