Departamento de Ciência da Computação
Universidade de Brasília
Programação Sistemática – 2º/2015

Data de entrega: **02/Novembro/2015, 23:55h**

Trabalho II

Introdução

Nessa etapa do trabalho haverá uma reorganização de módulos e funcionalidades em componentes visando à melhor modularização programa. E, também a criação de um novo componente: Localidade.

É indispensável a documentação de todo os módulos usando a ferramenta Doxygen ou JavaDoc, testes abrangentes para **todas as funcionalidades de todos os módulos,** dessa vez, usando também a ferramenta gcov (C) ou EclEmma (Java) para verificar o grau de cobertura dos testes e análise estática usando o Splint(C) ou FindBugs (Java).

Componentização

Os novos componentes terão como objetivo principal gerenciar as três estruturas principais do sistema: Palestras, Palestrantes (já implementadas) e Localidades. Na Figura 1 ilustramos a proposta do modelo conceitual (arquitetura) da nova proposta.

Para cada um dos componentes, deverá ser criada uma interface (módulo de definição) e seus respectivos módulos de implementação, além de serem aplicadas as devidas regras de encapsulamento do código. Lembrando que cada módulo deverá ser possível a compilação independente.

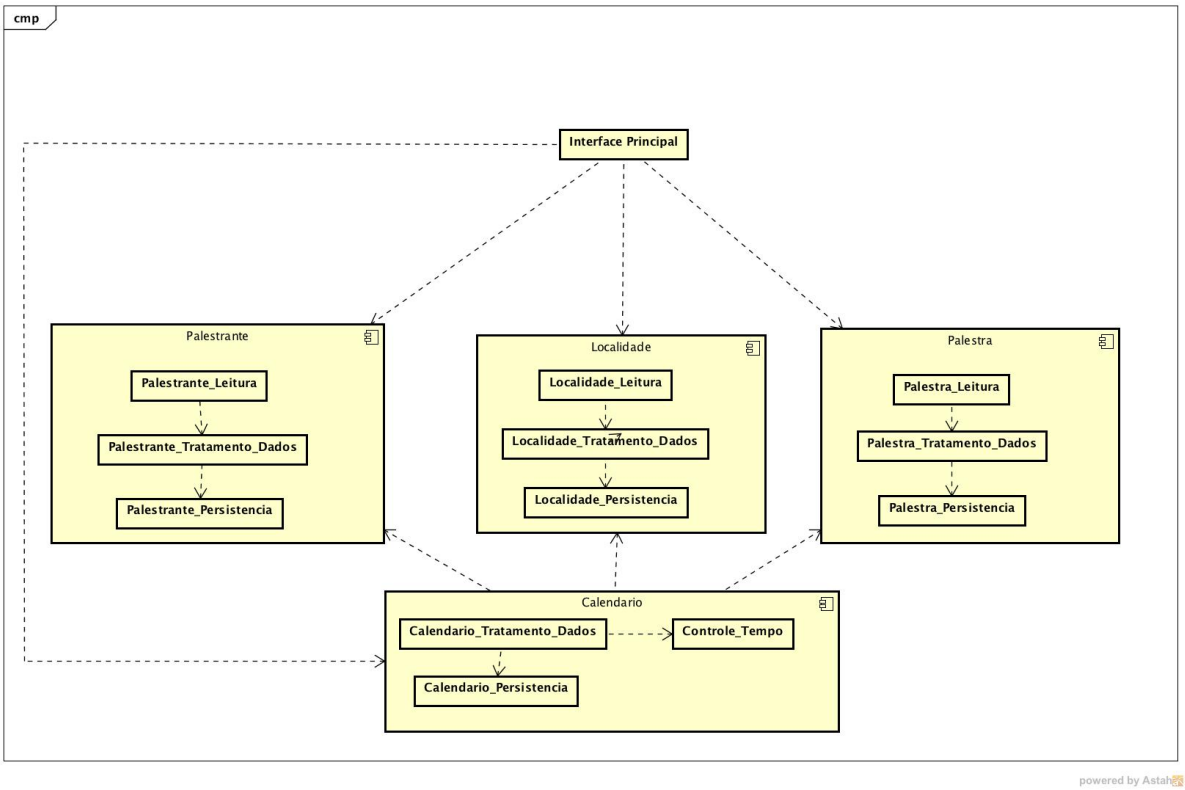

Figura 1 - Proposta do Arquitetura (Modelo Conceitual) da Nova Proposta

Parte das funcionalidades desses componentes já foram desenvolvidos na primeira etapa do trabalho, porém, será necessária uma refatoração dos módulos para melhor atender a nova proposta. A especificação da nova organização dos componentes devem ser a seguinte.

1. Componente de Palestras

Deve conter estruturas e funções para manipular dados das palestras com seus respectivos nomes, temas, duração. Muitas funções desse módulo já foram implementadas na parte 1 do trabalho poderão ser reutilizadas nesse novo componente.

Módulos Obrigatórios

- **Módulo de leitura de palestras**

  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.

  Saída: Conjuntos de dados organizados.

- **Módulo de tratamento de dados de palestras**

  Com os dados lidos, faz manipulações necessárias, aloca e preenche as estruturas com essas informações e verifica a validade das informações associadas às palestras conforme dados de localidade, palestrante e calendário.

  Entrada: Saída do módulo de leitura.

  Saída: Dados prontos para serem alocados nas estruturas.

- **Módulo de persistência de dados de palestras**

  Com as informações das estruturas prontas para seu uso final, faz um canal de comunicação com o módulo calendário. Enviando suas informações, e recebendo dados quanto ao sucesso na alocação da palestra.

  Entrada: Dados tratados, informações do módulo calendário.

  Saída: Estrutura palestra organizada.

2. Componente de Palestrantes

Deve conter estruturas e funções para manipular dados dos palestrantes. Muitas funções desse módulo já foram implementadas na parte 1 do trabalho, então, basta organizá-las em módulo de definição e implementação.

Módulos Obrigatórios

- **Módulo de leitura de palestrantes**

  Deve conter todas as funções para a leitura correta do arquivo, separando cada informação de maneira correta e conveniente.

  Entrada: Arquivo com informações dos palestrantes.

  Saída: Conjuntos de dados organizados.

- **Módulo de tratamento de dados de palestrantes**

  Com os dados lidos, faz manipulações necessárias, aloca, preenche as estruturas com essas informações e e verifica a validade das informações dos seus palestrantes.

  Entrada: Saída do módulo de leitura.

  Saída: Dados prontos para serem alocados nas estruturas.

- **Módulo de persistência de dados de palestrantes**

Com as informações das estruturas prontas para seu uso final, faz um canal de comunicação com o módulo calendário. Enviando suas informações, e recebendo dados quando o sucesso na alocação do palestrante.

Entrada: Dados tratados, informações do módulo calendário.

Saída: Estrutura palestrante organizada.

3. Componente de Localidade

Deve conter as estruturas e funções necessárias para criação e manipulação de dados relativos aos locais onde as palestras serão alocadas com respectivos palestrantes. Os campos de localidade deverão ser os seguinte: Pilha de Palestrantes (já palestrados), Endereço, Horário e Responsável (nome e contato email e telefone).

Módulos Obrigatórios

- **Módulo de leitura de localidades**

Deve conter todas as funções para a leitura correta do arquivo, separando cada informação de maneira correta e conveniente.

Entrada: Arquivo com informações dos locais.

Saída: Conjuntos de dados organizados.

- **Módulo de tratamento de dados de localidades**

Com os dados lidos, faz manipulações necessárias, aloca, preenche as estruturas com essas informações e verifica a validade das informações das localidades. Deve também disponibilizar funções de manipulação da pilha de palestrantes que ministraram palestra naquela localidade.

Entrada: Saída do módulo de leitura.

Saída: Dados prontos para serem alocados nas estruturas.

- **Módulo de persistência de dados de localidades**

Com as informações das estruturas prontas para seu uso final, faz um canal de comunicação com o módulo calendário. Enviando suas informações, e recebendo dados quando o sucesso na alocação do local.

Entrada: Dados tratados, informações do módulo calendário.

Saída: Estrutura local organizada.

4. Componente Calendário

Esse componente é central no sistema, uma vez que ele acopla os demais componentes de palestrantes, localidades e palestras. Deve juntar as informações dos outros componentes de maneira correta, e alocando as palestras de modo que não existam choques de horários para um mesmo dia em uma mesma localidade ou que hajam localidades com palestras sem palestrantes ou vice-versa. Na função de alocação das palestras e palestrantes em suas localidades, deverá ser implementado um algoritmo de *rodízio entre os palestrantes* a fim de garantir que haja uma rotatividade entre os palestrantes disponíveis para os horários de uma determinada localidade.

- **Módulo de tratamento de dados de calendário**

Aloca um local para uma palestra de um palestrante de maneira correta e sem os conflitos citados anteriormente

Entrada: Estruturas Palestra, Palestrante e Local.

Saída: Estrutura devidamente alocada e preenchida com as informações de entrada. (Sugestão: Estrutura mês).

- **Módulo de persistência de dados de calendário**

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.

- **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 linguagem 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.

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) ou FindBugs (Java) seguindo a análise para faltas de controle, armazenamento, interface e entrada/saída.

Para o teste, continue utilizando o CUnit (C) ou JUnit (Java), criando um módulo controlador de teste (disciplinado) para testar se as principais funcionalidades e restrições dos componentes especificados. 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.

Entrega do trabalho e correção

Nessa etapa do trabalho a documentação será avaliada com mais rigor que na anterior. Mantenham a organização e lembrem-se sempre de registrar todas as informações sobre atividades individuais de desenvolvimento, compilação, execução e funcionamento de cada módulo e seus componentes. Sugerimos uma organização de pastas para os códigos fontes. Separando os componentes e módulos de definição e implementação da seguinte forma:

$local$/src (arquivos fontes)
$local$/bin (arquivos executáveis)
$local$/doc (documentação gerada pelo JavaDoc ou Doxygen)
$local$/relatório (atividades individuais registradas)

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...). E uma explicação de como utilizar o(s) programa(s).
3. 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.

Excelente trabalho!