Exercício 2
Programação E Desenvolvimento De Software II

Marcela Garcia Raposo - 2017000528
Vitor Oliveira Moraes - 2017074335
Rodrigo Couto Leite Mendes - 2017098170

1 Introdução

Neste exercício prático, vamos simular o funcionamento de uma agenda virtual, capaz de armazenar compromissos em diferentes horários para todos os dias de cada mês do ano.
Para sua implementação, vamos utilizar um lista encadeada de meses e outra de compromissos, onde cada mês tem um vetor fixo de dias e cada dia possui um lista de compromissos.

2 Estruturas De Dados

A principal estrutura de dados utilizada neste exercício é a lista encadeada. Ela funciona de forma a simular um vetor, mas tal que seu tamanho é dinâmico, ou seja, sempre é possível aumentar ou diminuir a quantidade de elementos de forma que não irá comprometer a integridade de toda a estrutura. Para encontrar um dado elemento porém, precisamos percorrer a lista por completo e novos elementos geralmente são adicionados no final.

3 Modelagem

Na implementação do trabalho prático, foram utilizadas 4 estruturas diferentes, explicadas abaixo:

- **Compromisso**: uma estrutura que contém uma *string* (que é a descrição do compromisso) dois inteiros que representam as horas e os minutos e um ponteiro para o próximo compromisso, que seria a próxima célula da lista.
- **Dia**: uma estrutura que contém um ponteiro para o compromisso inicial e o compromisso final (que não necessariamente estão ordenados) e serve como encapsulador da lista de compromissos.
- **Mes**: uma estrutura que contém dois inteiros *d* e *t* que armazenam o número do mês e a quantidade de dias, respectivamente. Possui também um vetor de dias e um ponteiro para o próximo mês.

- **Agenda**: uma estrutura que armazena um ponteiro para o mês inicial e o mês final. Essa lista é criada em ordem de acordo com os meses do ano.

4 Funções

As funções do programa são todas para atender as seis operações básicas listadas na especificação, segue abaixo uma enumeração delas e como foi implementado em código:

- **Operação "Abrir Agenda"**: se dá pela função *abre_agenda*, uma função do tipo ponteiro para agenda que constrói um agenda nova, alocando a lista de meses e em cada mês alocando a tabela de dias correspondente, retornando a agenda.
- **Operação "Inserir Compromisso"**: se dá pela função *leitura_compromisso* que recebe um ponteiro para a agenda e após fazer as leituras de entrada adequadas, insere um novo compromisso no horário e data corretos, mas verificando primeiro se existe colisão.
- **Operação "Remover Compromisso"**: se dá pela função *remove_compromisso* que recebe um ponteiro para a agenda e depois de fazer a leitura da entrada do usuário, busca pela existência de um compromisso que tenha mesma data e horário (já que colisões não são permitidas) e se ele existir, será removido. Se não, exibirá uma mensagem de erro.
- **Operação "Listar Compromissos"**: se dá pela função *lista_compromissos* recebe um ponteiro para a agenda, e depois da leitura da entrada, lista todos os compromissos de um dado mês, percorrendo todas as listas de compromissos.
- **Operação "Verifica Compromisso"**: se dá pela função *verifica* que recebe um ponteiro para a agenda, e depois de ler a entrada do usuário faz a busca pelo compromisso compatível, e avisa se o compromisso existe ou não.
- **Operação "Fecha Agenda"**: se dá por duas funções, *fecha_agenda* e *destroi_agenda*, onde as duas recebem ponteiros para a agenda e a primeira salva os dados em arquivo e a segunda desaloca as estruturas que compõem a agenda.
- **Operação "Sair"**: simplesmente chama a função *destroi_agenda* (se a agenda ainda não foi encerrada) e encerra o programa.

5 Desenvolvimento e Organização

O programa foi desenvolvido em ambiente *Windows 10* utilizando a linguagem *C++* e o compilador *g++*consistindo em um arquivo *.hpp* e dois *.cpp*.

6 Conclusão

Por essa exercício foi possível aprender sobre a construção de estruturas dinâmicas encadeadas como as listas e sobre como modelar um problema real em um computacional e resolve-lo de forma razoavelmente eficiente, e também sobre o trabalho de desenvolvimento de software em equipe.