Universidade Federal de Minas Gerais

Programação e Desenvolvimento de Software II

Exercício I

Ana Carolina Condé Silva – 2018127238
Vinícius Belchior Bezerra – 2018104041

28 de Abril de 2019

1 Introdução

Neste trabalho propomos uma implementação de uma agenda virtual. Nela é possível cadastrar compromissos separados por mês, dia e horário (hora e minuto). Assim, foram utilizadas estruturas de dados pertinentes ao problema: vetores e listas encadeadas, efetuando uma modelagem através de Tipos Abstratos de Dados (TADs). As decisões de projeto, bem como detalhes de implementação, são apresentadas nos tópicos seguintes.

2 Estruturas de Dados

A estrutura de dados de maior destaque desse trabalho é a lista encadeada, justamente por proporcionar maior flexibilidade e melhor aproveitamento de memória, tendo em vista que esse tipo de estrutura suporta tamanhos dinâmicos sem comprometer sua integridade. Assim, foram utilizadas listas encadeadas para descrever a lista de meses e de compromissos. O impacto maior está na lista de compromissos, uma vez que a quantidade de elementos pode variar significativamente entre um dia e outro, assim como entre meses distintos. No entanto, também utilizamos vetores para a representação dos conjuntos de dias de cada mês, uma vez que a quantidade de dias não se altera dentro de um mesmo mês, aliado ao fato de termos que possibilitar o agendamento de compromissos em todos os dias de um mês.

3 Modelagem

Esse trabalho foi implementado de forma modular. As estruturas utilizadas são apresentadas na sequência.

- **Agenda**: módulo principal do trabalho, responsável por representar uma agenda de compromissos. Sua estrutura interna contém uma lista de meses, onde cada mês possui uma lista de compromissos para cada dia. Os meses são dispostos em ordem crescente na lista, isto é, de janeiro a dezembro.
- **Month**: estrutura para a representação de um mês. Possui como atributos o número de dias no mês, o qual é calculado em tempo de execução, um identificador para o mês e uma lista de compromissos para cada dia do mês. Vale ressaltar, entretanto, que, como sabemos de antemão a quantidade de dias existentes em um dado mês, utilizamos um vetor de tamanho fixo para armazenar as listas encadeadas que representam os compromissos. Assim como nos meses, os dias também são dispostos em ordem crescente, isto é, de 1 a 28 ou 30 ou 31, dependendo do mês.
- **Day**: estrutura para a representação de um dia. Trata-se de uma lista encadeada de compromissos, um novo tipo também criado para a execução do trabalho.

- **Commitment**: estrutura criada para a representação de um compromisso, contendo informações relevantes como *descrição*, *hora* e *minuto* de um compromisso em específico.

4 Execução

Para executar esse trabalho, basta utilizar o *Makefile* localizado no diretório raíz do projeto. Os comandos são apresentados abaixo:

```
$ make
$ make run
```

5 Funções

Esse trabalho contém implementações para as funções básicas requisitadas na especificação, as quais são brevemente apresentadas a seguir.

- **Abrir agenda**: permite ao usuário importar os dados previamente cadastrados na agenda, o que é possível através do salvamento do estado final da agenda em um arquivo em disco a cada término de execução. A função utilizada para isso é *loadAgenda*.
- **Inserir Compromisso**: permite a inserção de um compromisso na agenda. Para tanto, devem ser fornecidos o mês, dia, horário e descrição do compromisso, o qual será inserido ao fim da lista para aquela data (mês e dia). A função utilizada para isso é *insertCommitment*.
- **Remover Compromisso**: permite a remoção de um compromisso da agenda. Para tanto, devem ser especificados o mês, dia e horário do compromisso a ser removido. A função utilizada para isso é *removeCommitment*.
- **Listar Compromissos**: permite a listagem de todos os compromissos cadastrados para um dia de um mês. A função utilizada para isso é *showDay*.
- **Checar Compromisso**: permite a checagem de compromissos para uma data específica, isto é, para um mês, dia, hora e minuto. A função utilizada para isso é *showDayFilteredByDate*.
- **Fechar Agenda**: permite salvar o estado atual da agenda em um arquivo em memória secundária. Dessa forma, ao abrir a agenda novamente, os compromissos serão importados mesmo após o fechamento do programa, mantendo assim a integridade dos dados. A função utilizada para isso é *saveAgenda*.
- **Sair**: encerra a aplicação. Ao escolher essa opção, todas as alterações são descartadas e o arquivo de importação não é alterado.

6 Linguagem e Plataforma

Esse trabalho foi desenvolvido no sistema operacional *macOS Mojave*, entretanto também foi testado em ambiente Linux (distribuição *Ubuntu 16.04.6 LTS*). A linguagem utilizada foi *C++ 11* e, como utilizamos construções disponíveis apenas a partir da versão 11, tais como construtores e funções de *structs*, utilizamos uma *flag* para deixar explícito a escolha do compilador no *Makefile* (*g++* com *standard* da versão 11). Por fim, as implementações foram desacopladas de suas declarações, ou seja, os arquivos foram divididos em cabeçalhos *hpp* e arquivos *cpp*.

7 Conclusão

Através da implementação desse trabalho prático, foi possível obter um melhor entendimento sobre listas encadeadas, ficando claras as suas diferenças e peculiaridades em relação a vetores de tamanho fixo. Ainda, esse trabalho se mostrou positivo pois permitiu ao grupo implementar módulos separadamente para unificação posterior, permitindo maior contato com programação modular. Por fim, destaca-se também o aprendizado e aprimoramento do conhecimento de *C++*.