Universidade Federal de Minas Gerais

Exercício I – Programação e Desenvolvimento de Software II

Felipe Santos Netto Malta – 2018092833
Gabriel Soares Garrocho de Almeida – 2018019290
João Rafael Lemos Oliveira – 2018019273

28 de Abril de 2019

1 Introdução

Este trabalho possui como objetivo a implementação de uma agenda de compromissos, de forma a trabalhar o conteúdo de listas encadeadas. Para tanto, utilizamos uma lista encadeada para o armazenamento dos meses, cada mês contendo um vetor de listas encadeadas que representa as listas de compromissos para cada dia daquele mês.

2 Estruturas de Dados Utilizadas

Como já citado, foram utilizados vetores de tamanho fixo e listas encadeadas. Como o número de dias em um mês é fixo e para todo mês é necessário mapear os compromissos, um vetor de tamanho fixo é suficiente para essa representação, onde seu tamanho corresponde ao número de dias em um mês específico – nesse trabalho, utilizamos 28, 30 e 31 dias, de acordo com o mês em questão (excluímos anos bissextos). O número de compromissos em cada dia, entretanto, é imprevisível e pode sofrer variações significativas de um mês para outro e mesmo de um dia para outro. Nesse contexto, utilizar listas encadeadas para esse caso é uma boa estratégia, provendo um melhor aproveitamento de recursos de memória.

3 Estrutura do Projeto

De forma a tornar o projeto modular, foram implementadas quatro estruturas, como listado abaixo.

- **AppointmentItem**: uma estrutura que representa um compromisso. Seus campos são *hour* (inteiro), *minute* (inteiro) e *description* (string), compondo as informações de um compromisso – hora, minuto e descrição, respectivamente.
- **AppointmentList**: uma estrutura que representa uma lista de compromissos, contendo um apontador para o primeiro nó, um apontador para o último nó e um inteiro para o armazenamento do tamanho da lista encadeada em um dado momento.
- **AgendaItem**: uma estrutura que representa um mês, contendo um identificador do mês, um vetor de listas de compromissos e um inteiro representando a quantidade de dias daquele mês.
- **Agenda**: uma estrutura que representa a agenda, possuindo um ponteiro para o mês inicial e um ponteiro para o mês final, além de um inteiro contendo o número de nós presentes na lista em um dado momento.

4 Execução

Para a execução do programa desenvolvido, basta utilizar o *Makefile* criado no diretório raíz do projeto, gerando um arquivo executável de nome **tp1**. Os comandos necessários para isso estão ilustrados abaixo:

```
$ make
$ ./tp1
```

5 Funções Implementadas

As funções implementadas nesse trabalho são as funções básicas propostas na especificação, as quais são listas a seguir:

- **Abrir Agenda**: a agenda só pode ser utilizada após ser explicitamente aberta, bloqueando todas as funções enquanto essa opção não for selecionada. Ao abrir a agenda, um arquivo, de nome *agenda.txt*, localizado na pasta *data* é lido. Esse arquivo possui todas as informações referentes aos compromissos previamente cadastrados no sistema, que são então carregados. Cada linha desse arquivo possui a descrição, em texto claro, de um compromisso, no seguinte formato: ***mês|dia|hora|minuto|descrição***. A principal função utilizada para esse processo é *open_agenda*.
- **Inserir Compromisso**: insere um compromisso na agenda com base no mês, dia, hora e minuto fornecidos pelo usuário através de um menu interativo. A principal função utilizada para tal é *push_appointment*.
- **Remover Compromisso**: remove um compromisso da agenda com base no mês, dia, hora e minuto fornecidos pelo usuário através de um menu interativo. As principais funções utilizadas para isso são *search_node_agenda* e *delete_appointment*.
- **Listar Compromissos**: lista todos os compromissos de um mês e dia informados pelo usuário através de um menu interativo. A função utilizada nesse processo é *print_appointment_list*.
- **Verificar Compromisso**: verifica se existe um compromisso em um mês, dia, hora e minuto. As principais funções utilizadas para isso são *search_node_agenda* e *print_appointment_list_by_date*.
- **Fechar Agenda**: salva as alterações realizadas em memória secundária – arquivo *agenda.txt*, de forma a possibilitar futuras alterações mesmo após fechar o programa. A função utilizada para isso é *close_agenda*.
- **Sair**: descarta todas as alterações realizadas e fecha o programa.

6 Tecnologias

Esse trabalho foi desenvolvido inteiramente na linguagem *C++*. Além disso, como foram utilizadas algumas construções do *C++ 11*, como *nullptr*, por exemplo, foi utilizado o compilador *g++* com a *flag -std=c++11*, como é possível observar no *Makefile*. Além disso, o código foi modularizado em arquivos de cabeçalho (*hpp*) e de implementação (*cpp*), sendo tais arquivos de código dispostos no diretório *src*. Por fim, vale ressaltar que o trabalho foi desenvolvido e testado em ambiente Linux.

7 Conclusão

Esse trabalho foi importante para o aprimoramento dos conceitos vistos em sala de aula. As diferenças entre vetores de tamanho fixo e listas encadeadas, bem como os momentos onde cada estrutura se aplica melhor, foram consolidadas ao realizar esse trabalho, permitindo uma melhor familiarização do grupo tanto com essas estruturas quanto com a linguagem *C++*.