



Universidade Federal de Viçosa

Universidade Federal de Viçosa Campus Florestal
CCF 251 - Algoritmos e Estruturas de Dados I

Trabalho Prático 1

Trabalho Prático 01 – AEDS 1

Gabriel Miranda - 3857

Felipe Dias - 3888

Mariana Souza - 3898

Sumário

1	Intrudução.....	3
2	Desenvolvimento.....	4
2.1	TADs.....	4
2.2	Listas.....	5
2.3	Leitura de Arquivo.....	5
2.4	Main.....	5
3	Execução.....	6
4	Conclusão.....	6

1 Intrudução

O presente trabalho tem como objetivo desenvolver um sistema automatizado para gerenciar o tempo para os professores do campus UFV-Florestal, esse sistema irá permitir a criação de uma agenda para cada professor, cada agenda terá os compromissos que cada professor poderá inserir, além disso, compromissos poderão ser mostrados para cada usuário e modificados como o desejado.

Para o desenvolvimento dos códigos, foram utilizados, o compilador GCC, a IDE CLion, Visual Studio Code e CodeBlocks, e a plataforma GitHub para cooperação entre os integrantes do grupo no processo de criação.

2 Desenvolvimento

Para alcançar o objetivo do trabalho prático proposto, desenvolver uma agenda para professores do campus, foi necessário iniciar pela implementação da modularização para uma melhor compreensão. Posteriormente, criou-se os TADs compromisso e agenda, e logo após, os integrantes do grupo decidiram a criação de mais dois TADs, professor e endereço.

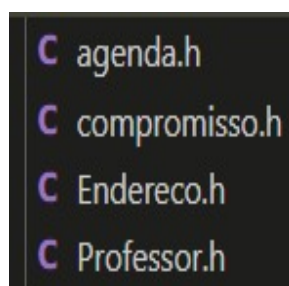
Junto a isso, foram desenvolvidas duas listas encadeadas para o compromisso e agenda, uma opção de Leitura de Arquivo e um menu para interação direta do usuário.

2.1 TADs

Houve a aplicação de quatro Tipos Abstratos de Dados. O compromisso receberá as funções para inicializar um compromisso usando alocação dinâmica, alterar a prioridade, retornar prioridade, verificar se o mesmo tem conflito e imprimir o compromisso. Foi criada modularização para data e hora do compromisso para verificação de casos de conflito entre dois compromissos, assim os evitando.

Na agenda temos as funções criar agenda, inserir, retornar e remover compromissos e imprimir agendas. No remover compromisso foi utilizada uma variável tratadora inteira recebendo a função, caso o tratador retorne 1, essa remoção será bem sucedida, caso contrário, a remoção não ocorrerá.

Em professor e endereço foram executadas as funções de inicializar e imprimir suas respectivas informações.

A imagem mostra um código-fonte em C para a declaração de Tipos Abstratos de Dados (TADs). O código está escrito em um editor com fundo escuro e letras coloridas (verde para comentários, amarelo para strings). As declarações são: agenda.h, compromisso.h, Endereco.h e Professor.h. Cada linha começa com um caractere 'C' em verde.

```
C agenda.h
C compromisso.h
C Endereco.h
C Professor.h
```

Imagem1. Declaração dos TADs.

2.2 Listas

A implementação de lista foi feita com listas encadeadas com cabeça para o compromisso e agenda, LCompromisso.h e LAgenda.h como o nome de seus arquivos. Contendo as funções inicializa lista compromisso e lista agenda, confere se uma lista é vazia, insere e imprime em ambas listas. Para a lista agenda foi aplicada a função procura a agenda do professor, essa irá fazer o login na agenda de determinado professor, retornando o endereço de memória da célula agenda que conterà o ID do professor. A função remove compromisso poderá remover um compromisso em qualquer posição da lista.

```
Tcelula_Agenda Procura_agenda_professor(TLagenda *lagenda, char *id_professor){
    Tcelula_Agenda *pAux;
    pAux = lagenda->primeiro->prox;
    while (pAux != NULL) {
        if(strcmp(pAux->Agenda->idProf, id_professor) == 0 ){
            return *pAux;
        }
        pAux = pAux->prox;
    }
}
```

Imagem2. Função procura agenda de um professor, arquivo “LAgenda.c”

2.3 Leitura de Arquivo

Outra opção para entrada de dados, a parte de leitura de arquivo foi projetada especificamente para o modelo padrão do arquivo de entrada. Sendo antecipada nela o tipo de conteúdo de cada linha do modelo “.txt” para a correta identificação de cada variável integrante da agenda ou de um compromisso.

Transformando os devidos caracteres lidos em tipos inteiros correspondentes através da função “atoi” da biblioteca “stdlib.h”. A leitura é adaptada para receber a quantidade total de professores e a quantidade de compromissos de cada um e fazer loops que permitam a leitura de todos.

2.4 Main

Para a execução, foram criadas, na main, um menu e um submenu contendo as opções para criar agenda, ler um arquivo, fazer login na agenda através de um ID, recuperar uma agenda e inserir, remover e imprimir compromissos. Também foram inseridas as chamadas das funções para realizar tais operações.

3 Execução

Para a compilação foi utilizado o comando “gcc main.c arquivos.c -o e”, gerando assim um executável (e.exe).

```
printf("|-----|\n");
printf("|  [1] - Criar Agenda:          |\n");
printf("|  [2] - Leia o arquivo:         |\n");
printf("|  [3] - Faca login:              |\n");
printf("|  [4] - Mostrar a agenda de todos os professores: |\n");
printf("|  [5] - Sair                    |\n");
printf("|-----|\n");
printf("Opcao: ");
```

Imagem3. Compilação no Terminal(Menu de opções)

```
printf("|-----|\n");
printf("|  [1] - Recuperar Agenda          |\n");
printf("|  [2] - Inserir Compromisso       |\n");
printf("|  [3] - Remover compromisso       |\n");
printf("|  [4] - Imprimir Compromissos     |\n");
printf("|  [5] - Imprimir Apenas um compromisso |\n");
printf("|  [6] - Retornar qtd compromissos |\n");
printf("|  [7] - Altera prioridade Compromisso |\n");
printf("|  [8] - Sair                     |\n");
printf("|-----|\n");
printf("Opcao: ");
```

Imagem4. Terminal(Submenu de opções)

4 Conclusão

Portanto, conclui-se que a essencialidade do conceito e uso de listas encadeadas para a boa prática e organização da programação, principalmente em códigos mais extensos e sem um limite máximo claro de componentes para listagem, impossibilitando um array. Juntamente ao encadeamento, tem-se quase inerente a importância da alocação dinâmica de memória, pois, além de ser um recurso necessário para a criação de novas células da lista, tal conceito contribui para o não desperdício de um possível array com muitos espaços que acabariam não sendo preenchidos por dados.