Trabalho Prático 1

Estudante: Tarcísio de Oliveira Rocha Nº de matricula: 2010016950

1. **Introdução:**

Criaremos no TP1 uma rede social, assim como o tweeter e facebook, onde os usuários tem disponível uma serie de operações que poderão ser solicitadas por cada usuário o nosso programa também o será. O programa consiste em coletar os dados de entrada em um arquivo entrada.txt, neste arquivo é disponibilizado o nome de cada usuário da rede, seu ID, e amigos que cada usuário tem. Em uma próxima linha são fornecidas as operações que os usuários da rede solicitaram, como por exemplo: adicionar um amigo, cancelar amizade, postar uma mensagem, etc. O TP1 visa reforçar as estruturas (Listas,Filas,Pilhas) vistas por nós em sala de aula. Dois TADs serão muito importantes para a implementação, o TAD usuário e TAD mensagem.

1. **Implementação:**

**Estrutura de Dados**

Para a implementação do trabalho foi criado um Tipo Abstrato de Dados usuário com a seguinte estrutura:

**typedef struct usuario{**

**int ID;**

**char nome[MAX\_NOME];**

**TipoLista amigos;**

**timeline LT[TIMELINE\_MAX];**

**int indice;**

**}usuario;**

A estrutura é composta por um ID (int) que é a identificação do usuário, o nome do usuário, sua lista de amigos, LT que é to tipo timeline onde ficará as mensagens a ser exibidas na sua timeline, e um inteiro índice, que irá conter o local do ultimo elemento do vetor LT[]. Os valores de MAX\_NOME, e TIMELINE\_MAX serão apresentados posteriormente.

**typedef struct mensagem{**

**int ID\_mensagem;**

**char texto[TWEETS];//tamanho maximo da mensagem**

**int t; // instante em que foi postada**

**int curtidas;//numero de curtidas**

**int ID\_user;**

**}mensagem;**

Essa estrutura é composta por um ID da mensagem(int), um texto que será a mensagem postada por um usuário, um inteiro t (tempo), curtidas e ID\_user que é o identificador do usuário que postou a mensagem.

**typedef struct timeline{**

**char text[TWEETS];**

**int ID\_msg;**

**}timeline;**

Essa estrutura armazena uma mensagens e o ID desta mensagem.

**typedef int TipoChave;**

**typedef struct TipoItem{**

**TipoChave ID\_amigo;**

**}TipoItem;**

**typedef struct TipoCelula \*TipoApontador;**

**typedef struct TipoCelula{**

**TipoItem Item;**

**TipoApontador Prox;**

**}TipoCelula;**

**typedef struct TipoLista{**

**TipoApontador Primeiro,Ultimo;**

**}TipoLista;**

Também foi criada uma estrutura para a lista encadeada como vista em sala, a única diferença é que o nome de chave, para ID\_amigo.

**Funções e Procedimentos**

**void FLVazia(TipoLista \*Lista) =>** essa função cria uma lista com zero elementos.

**int Vazia(TipoLista Lista)=>** testa se a lista está vazia.

**void Insere(TipoItem x, TipoLista \*Lista)=>** insere um nome item na lista.

**void Retira(TipoApontador p, TipoLista \*Lista , TipoItem \*Item)=>** retira um item da lista

**void Imprime(TipoLista Lista)=>** Imprime os elementos da lista.

**int procura\_usuario(usuario A[],int tamanho,int ID)=>** recebe um vetor do tipo usuário, o seu tamanho, e retorna a posição onde o ID é o mesmo, se não encontrado o ID retorna -1.

**int procura\_mensagem(mensagem A[],int indice,int ID)=>** recebe um vetor do tipo mensagem e retorna a posição do vetor onde a mensagem é encontrada, retorna -1 caso não seja encontrada.

**TipoApontador procura\_amigo(TipoLista \*L,TipoItem item)=>** procura um amigo na lista de amigos que tenha o mesmo item comparado e retorna um apontador para este amigo.

**int postarMensagem(mensagem post, usuario \*A, int tamanho,int user)=>** posta a mensagem na timeline de todos os amigos que o “user” tenha e na timeline do próprio user. Ela recebe a mensagem, o vetor contendo todos os usuários, o tamanho do vetor, o ID do usuário que postou a mensagem. A partir da função procura\_usuario(), encontra-se a posição de todos os usuários que são amigos de quem posta a mensagem, e então adiciona essa mensagem a timeline de todos os amigos.

**int iniciarAmizade(usuario \*A,TipoItem x)=>** recebe um ponteiro do tipo usuário e um item, e insere x na lista de amigos de A por meio da função Insere().

**int cancelarAmizade(usuario \*A,TipoItem Item)=>** recebe um usuário, e um item pertencente ao usuário que se queria cancelar a amizade. Procura-se a posição deste amigo pela função procura\_amigo(), e cancela a amizade pela função Retira(), que retira o amigo da lista de amigos.

**int curtirMensagem(int tempo,mensagem A[],int indice, int ID\_User, int**

**ID\_mensagem)=>** essa função procura a posição em que se encontra no vetor A[] o ID\_mensagem por meio da função procura\_mensagem(), depois de encontrar, damos um acréscimo nas curtidas e atualizamos o tempo.

**int exibirTimeline(FILE \*saida, timeline LT[], int indice)=>** Recebe uma variável do tipo timeline e imprime do ultimo elemento até o primeiro elemento da timeline.

**int verAmigos(TipoLista L)=>** essa função apenas chama a função Imprime(). (foi criada apenas para cumprir todas as especificações do enunciado).

**Programa Principal**

O programa principal cria duas variável do tipo File uma para a entrada e outra para saída, abre os 2 arquivos, entrada.txt e log.txt, na linha 12 coleta o numero de usuários da entrada; das linha 14 a 21 temos declarações de variáveis; na linha 24 inicializamos o índice dos usuários; na linha 33 temos um for que vai até o numero de usuarios, dentro deste for coletamos o ID do usuário e nome e amigos. Na linha 54 temos um while que vai até o fim do arquivo, dentro desse while temos condições “ifs” para as 5 operações:

POSTAR\_MENSAGEM ,INICIAR\_AMIZADE ,CANCELAR\_AMIZADE, CURTIR\_MENSAGEM , EXIBIR\_TIMELINE. Que poderão ser feitas pelo usuário, como descrito no enunciado. Mais detalhes sobre, vide código.

**Organização do Código, Decisões de Implementação e Detalhes Técnicos**

O código está dividido em três arquivos principais: myface.h, myface.c e main.c que roda o programa principal.

Defini valores específicos para algumas constantes:

#define TWEETS 140//dado do enunciado

#define MAX\_AMIGOS 5000//numero de amigos, mesmo numero que do facebook

#define MAX\_NOME 50//tamanho maximo do nome do usuario

#define MSG\_TAM 1000// tamanho maximo da mensagem

#define TIMELINE\_MAX 1000// tamanho Maximo para timeline, ou seja, quantidade de mensagens

///códigos como escritos no enunciado

#define POSTAR\_MENSAGEM 1

#define INICIAR\_AMIZADE 2

#define CANCELAR\_AMIZADE 3

#define CURTIR\_MENSAGEM 4

#define EXIBIR\_TIMELINE 5

De forma a tornar meu código mais legível o possível. Resolvi definir nome a cada uma das operações que poderão ser feitas pelo usuário, isso fez com que meu programa ficasse muito fácil de se entender.

O IDE utilizado foi o Codeblocks 10.05, compilador GCC no sistema operacional Windows XP. Para executá-lo, utilizar o ambiente do Codeblocks 10.05 importar os 3 arquivos enviados e clicar em executar. Se estiver utilizando o Linux, coloque o arquivo entrada.txt dentro da pasta, navegue no terminal Linux para o endereço da pasta e digite gcc main.c -o main.

**3. Análise de Complexidade**

A análise de complexidade será feita em função da variável n que representa o número de usuários da rede social.

**Funções e Procedimentos**

**void FLVazia(TipoLista \*Lista) =>** O(1).

**int Vazia(TipoLista Lista)=>** O(1).

**void Insere(TipoItem x, TipoLista \*Lista)=>** O(1).

**void Retira(TipoApontador p, TipoLista \*Lista , TipoItem \*Item)=>** O(1)

**void Imprime(TipoLista Lista)=>** O(1).

**int procura\_usuario(usuario A[],int tamanho,int ID)=>** faz-se um for até o tamanho Maximo de usuários, O(n).

**int procura\_mensagem(mensagem A[],int indice,int ID)=>** estimando que as mensagens tenham a mesma ordem de grandeza que usuarios**.** O(n).

**TipoApontador procura\_amigo(TipoLista \*L,TipoItem item)=>** O(n). dado o fato de que todos possam ser amigos na rede.

**int postarMensagem(mensagem post, usuario \*A, int tamanho,int user)=>** O(n). Dado o fato de que todos possam ser amigos na rede.

**int iniciarAmizade(usuario \*A,TipoItem x)=>** O(1). Mesmo caso para insere().

**int cancelarAmizade(usuario \*A,TipoItem Item)=>** O(1). Mesmo caso para retira().

**int curtirMensagem(int tempo,mensagem A[],int indice, int ID\_User, int**

**ID\_mensagem)=>** O(n). depende do numero de amigos do usuário.

**int exibirTimeline(FILE \*saida, timeline LT[], int indice)=>** O(n). exibir tameline depende do índice que por sua vez depende do numero de usuários na rede.

**int verAmigos(TipoLista L)=>** O(1). Essa função apenas chama Imprime, elas são O(1), partindo do pressuposto que o printf tem custo muito baixo.

­

**Programa Principal**

No programa principal tempos varias atribuições de variáveis que não convém ao caso, começando então na linha 24 nossa análise.

Na linha 24 temos a inicialização dos índices, ela é O(n), pois depende do numero de usuarios. Na Linha 42 temos um while que vai até o numero de amigos do usuário que pode ser até (n-1), como esse while se encontra dentro de um loop for que vai até o numero de usuários, temos O(n^2). A partir da linha 53 temos uma serie de condições que chamam as funções ante-hora analizadas. Das condições: 1 postar mensagem O(n). 2 iniciar amizade O(1). 3 Cancelar amizade O(1). 4 Curtir mensagem O(n^2) pois temos um “arreda do meio da implemetação”[vide codigo]. 5 exibir timeline O(n).

Complexidade geral do programa: **O(n^2).**

1. **Testes**

Desde o inicio do processo, tive uma preocupação em registrar tudo o que fazia, para que o leitor desta documentação, tenha facilidade em entender minhas ideia, e as soluções por mim apresentadas:

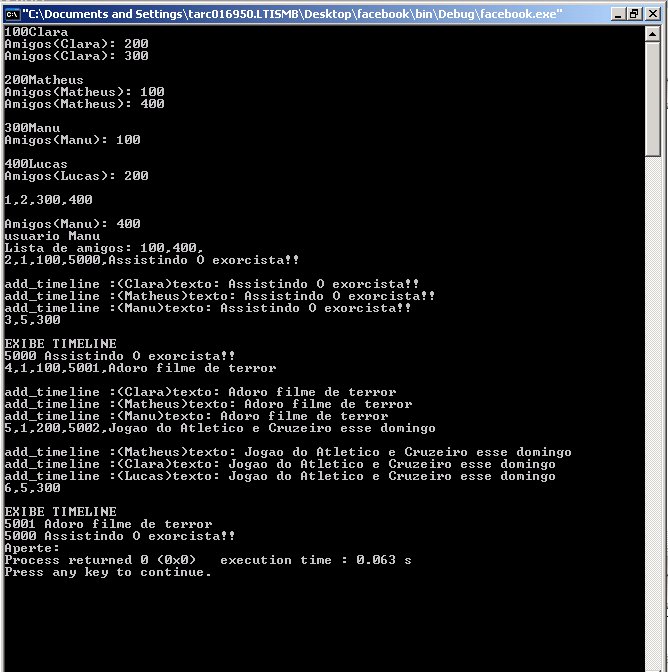


Figura1.

A Figura 1, apresenta um teste de entrada e saída de dados, isso me ajudou a ver se realmente estava programando corretamente, como podem ver, além disso essa figura me da informação de todas as operações pedidas no enunciado. Esse teste foi feito pela entrada do enunciado.

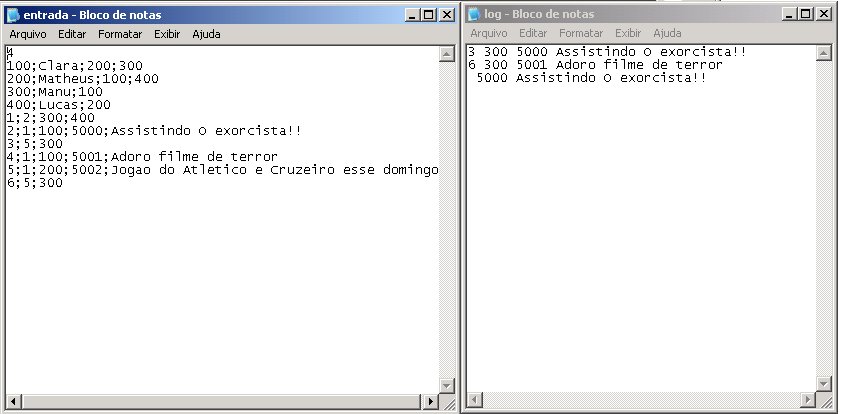


Figura 2.

A Figura 2 é a minha primeira tentativa de gerar o arquivo de saída ok, mais ainda não está bom.

Figura3

Tempo gasto para o trabalho pratico 1:

Comecei no mesmo dia em que o trabalho foi disponibilizado (quinta-feira, 29 de setembro de 2016, 10:57:01), então foram mais ou menos 21dias disponíveis dos quais pude trabalhar 13 dias, programei todas as tardes e noites no laboratório do dcc no primeiro andar, e alguns dias de manha também, não pude programar aos sábados e domingos pois meu computador queimou, comprei outro e este queimou novamente. São um total de 13 dias, 6 horas todos os dias (média) = 78 horas. Desta vez tive a ajuda do fórum da disciplina.

**6. Conclusão**

A implementação do trabalho transcorreu sem maiores problemas e os resultados a meu ver foram os esperados, todas as funções criadas foram testadas por uma serie de printf que me deram mais noção do acerto ou erro que estava tendo em algumas funções. As principais dificuldades encontradas foram com relação à utilização de um EOF, pois tive que aprende-lo para criar um loop até o fim do arquivo.“Tive duvidas de como tirar o caractere “;” quando na leitura do arquivo de entrada, depois de lido o ID do usuário eu estava pegando todo o nome deste além do ID dos seus amigos e salvando em uma variável char[], resolvi o problema utilizando a máscara [^;]. Além disso ocorreu um erro quando declarei uma variável do tipo char com o nome “mensagem”, isso fez com que meu programa parasse de funcionar já que eu tinha uma estrutura com o mesmo nome, é até cômico, mais demorei um tempo pra perceber isso. Tentei explicitar minhas ideias o mais claro possível, todo o código foi bem identando e comentado, e organizei o código do programa como aprendido em sala de aula.

**Referências:**

<http://pt.stackoverflow.com/questions/70140/como-%C3%A9-feito-o-loop-de-fim-de-arquivo>

<https://virtual.ufmg.br/20162/pluginfile.php/228395/mod_resource/content/1/cap3.pdf>

<https://www.vivaolinux.com.br/topico/C-C++/Ler-nomes-com-espaco>

**Anexos**

Listagem dos programas:

myface.h

myface.c

main.c