INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Passo Fundo

Trabalho Etapa II - Lista

Professor: Maikon Cismoski dos Santos
Curso: Ciência da Computação
Disciplina: Estrutura de Dados I
Turma: 3M1

Enunciado

Redes sociais são popularmente conhecidas, onde usuários cadastrados na rede podem possuir diferentes amigos (também usuários da rede), que partilham objetivos e valores em comum.

Escreva um programa em C++ para gerenciar listas de amigos e de usuários de uma rede social. As operações que o programa deve ter são listadas a seguir:

- *addUsuario (ID, Nome, idade, sexo)* - Adiciona um usuário na rede social com seu identificador (ID), nome, idade e sexo.
- *addAmigo (ID1, ID2)* - Cria uma relação de amizade entre os dois usuários especificados nos parâmetros, ou seja, o usuário ID1 adiciona o usuário ID2 como amigo, assim como o usuário ID2 adiciona o usuário ID1.
- *removerAmigo (ID1, ID2)* - remove a relação de amizade entre dois usuários.
- *removerUsuario (ID)* - Remove o usuário com o ID passado por parâmetro da rede social.
- *imprimirUsuarios* - Imprime o nome e o identificador de todos os usuários cadastrados na rede social.
- *imprimirAmigos (ID)* - Imprime o nome e o identificador de todos os amigos do usuário com o ID passado por parâmetro.
- *imprimirMediaIdadeAmigos (ID)* – Imprime a média de idade dos amigos do usuário com o ID passado por parâmetro.
- *imprimirAmigosEmComum (ID1, ID2, sexo)* – Imprime o nome e o identificador de todos os amigos em comum entre dois usuários com base no parâmetro sexo:
    - Se sexo=0 – imprime todos os amigos do sexo masculino em comum entre os dois usuários passados por parâmetro;
    - Se sexo=1 – imprime todos os amigos do sexo feminino em comum entre os dois usuários passados por parâmetro;
    - Se sexo=2 – imprime todos os amigos entre os dois usuários passados por parâmetro, independente do sexo.

As operações citadas acima possuem as seguintes restrições:

- *addUsuario*:
    - Os ID são únicos, ou seja, não há ID repetidos.
    - Os usuários devem ser adicionados na lista de usuários em ordem crescente de ID. A lista não pode sofrer uma ordenação dos seus elementos.
    - Mostrar uma mensagem de erro quando o ID já está cadastrado na rede. O nome e o ID devem ser exibidos na mensagem de erro.
    - Mostrar uma mensagem quando um usuário é cadastrado com sucesso. O nome e o ID devem ser exibidos na mensagem.

- *addAmigo*:
    - Um usuário não pode ser amigo dele mesmo.
    - Caso um dos usuários passados por parâmetros não exista na rede social ou eles já possuam uma relação de amizade, a operação é abortada.
    - Se os usuários passados por parâmetro forem iguais, a operação deve ser cancelada.
    - Os amigos devem ser adicionados na lista de amigos em ordem crescente de ID de cada usuário. A lista não pode sofrer uma ordenação dos seus elementos.
    - Mostrar uma mensagem de erro quando não for possível criar uma relação de amizade entre os usuários passados por parâmetro. Os IDs devem ser exibidos na mensagem de erro.
    - Mostrar uma mensagem quando a operação é realizada com sucesso. Os nomes e os IDs dos usuários devem ser exibidos na mensagem.
- *removerAmigo*:
    - Se um dos usuários (ou ambos) não existir ou eles não sejam amigos, a operação é cancelada.
    - Mostrar uma mensagem de erro quando não for possível remover a relação de amizade entre os usuários passados por parâmetro. Os IDs devem ser exibidos na mensagem de erro.
    - Mostrar uma mensagem quando a operação é realizada com sucesso. Os nomes e os IDs dos usuários devem ser exibidos na mensagem.
- *removerUsuario* –
    - Quando um usuário é removido, a relação de amizade entre ele e seus amigos também deve ser removida.
    - Se o usuário não existir a operação é cancelada.
    - Mostrar uma mensagem de erro quando não for possível remover o usuário passado por parâmetro. O ID deve ser exibido na mensagem de erro.
    - Mostrar uma mensagem quando a operação é realizada com sucesso. O nome e o ID do usuário deve ser exibido na mensagem.
- *imprimirAmigosEmComum*:
    - Se um dos usuários (ou ambos) não existir, a operação é cancelada.
    - Mostrar uma mensagem de erro quando não for possível imprimir os amigos em comum entre os usuários passados por parâmetro. Os IDs devem ser exibidos na mensagem de erro.
    - Caso os usuários não tenham amigos em comum, apenas imprima o nome e os IDs dos usuários passados por parâmetro. Exemplo: “Amigos em comum entre Fulano (ID) e Ciclano (ID):\n”
- *imprimirMediaIdadeAmigos*:
    - Se o usuário não existir, a operação é cancelada.
    - Mostrar uma mensagem de erro quando não for possível imprimir a média de idade dos amigos do usuário passado por parâmetro. O ID deve ser exibido na mensagem de erro.
    - Caso o usuário não tenha amigos, apenas imprima o nome e o ID do usuário passado por parâmetro. Exemplo: “Média de idade dos amigos de Fulano (ID): \n”.
- *imprimirAmigos*:
    - Se o usuário não existir, a operação é cancelada.
    - Mostrar uma mensagem de erro quando não for possível imprimir os amigos do usuário passado por parâmetro. O ID deve ser exibido na mensagem de erro.
    - Caso o usuário não tenha amigos, apenas imprima o nome e o ID do usuário passado por parâmetro. Exemplo: “Amigos de Fulano (ID): \n”.

Os usuários da rede social e as listas de usuários e amigos devem ser criadas com base do código baixo:

```
//pré-declaração da struct Usuario
struct Usuario;

struct No
{
    Usuario *dado;

    No()
    {
        dado = nullptr;
        prox = nullptr;
    }
};

struct Lista
{
    No* inicio;
    No* fim;
    int tamanho;
    Lista() //inicializa a lista!
    {
        inicio = nullptr;
        fim = nullptr;
        tamanho = 0;
    }
    ~Lista() //destroi a lista!
    {

        No *n = inicio;
        while(n)
        {
            No *aux = n;
            n = n->prox;

            delete aux;
        }
        inicio = nullptr;
        fim = nullptr;
        tamanho = 0;
    }
};
```

```
struct Usuario
{
    int ID;
    int idade;
    int sexo;
    std::string nome;
    Lista *amigos; //lista de amigos


    Usuario(int id, int ida, int sx, string nm)
    {
        ID = id;
        idade = ida;
        sexo = sx;
        nome =  nm;
        amigos = new Lista();
    }

    Usuario()
    {
        ID = 0;
        idade = 0;
        sexo = 0;
        nome =  "";
        amigos = nullptr;
    }

    ~Usuario()
    {

        if(amigos)
            delete amigos;
        amigos = nullptr;

        ID = 0;
        idade = 0;
        sexo = 0;
        nome =  "";
        amigos = nullptr;
    }
};
```

No que, os dados dos nós da lista de usuários da rede social são do tipo Usuário (lista de usuários). Da mesma forma, cada usuário possui uma lista de amigos, que também são usuários da rede (lista de usuários). Para não ocorrer duplicação de dados, quando um novo usuário é criado, deve-se alocar memória dinamicamente. Dessa forma, os nós de ambas as listas vão armazenar apenas os ponteiros para os usuários.

Entrada

O programa possui como entrada um arquivo texto com o nome *entrada.txt*. Um exemplo de um arquivo de entrada é ilustrado no quadro abaixo.

O arquivo armazena uma sequência de operações, sendo uma operação por linha. Além disso, cada operação é seguida de seus parâmetros, onde cada parâmetro é composto por uma única string e separados por espaços em branco.

O arquivo pode conter linhas em branco e comentários. Os comentários são identificados pelo caractere #, que indica que o conteúdo que segue este caractere na linha em que ele for encontrado é um comentário.

```
addUsuario 6 40 0 Joao #primeiro usuário da rede
addUsuario 4 25 0 Pedro
addUsuario 3 10 0 Paulo
addUsuario 7 30 1 Maria
addUsuario 30 50 0 Francisco
addUsuario 30 20 1 Mariana #não vai adicionar este usuário, pois o ID 30 já existe

addAmigo 6 4
addAmigo 6 4 #não cria uma nova amizade porque eles já são amigos
addAmigo 6 3
addAmigo 7 6
addAmigo 7 3
addAmigo 3 4
addAmigo 8 9 #não cria a amizade porque os usuários não existem

imprimirUsuarios

imprimirAmigos 3
imprimirAmigos 4
imprimirAmigos 6
imprimirAmigos 7
imprimirAmigos 30
imprimirAmigos 9 #Erro, o usuário com ID 9 não existe

imprimirMediaIdadeAmigos 6
imprimirMediaIdadeAmigos 4
imprimirMediaIdadeAmigos 3
imprimirMediaIdadeAmigos 7
imprimirMediaIdadeAmigos 30
imprimirMediaIdadeAmigos 9 #Erro, o usuário com ID 9 não existe

imprimirAmigosEmComum 6 3 2
imprimirAmigosEmComum 6 3 0
imprimirAmigosEmComum 6 3 1
removerAmigo 4 3
imprimirAmigosEmComum 6 3 2
imprimirAmigosEmComum 6 3 0
imprimirAmigosEmComum 6 3 1
imprimirAmigosEmComum 8 9 0 #os usuarios não existem, não imprime nada

removerUsuario 7

imprimirUsuarios
imprimirAmigos 3
imprimirAmigos 4
imprimirAmigos 6
imprimirAmigos 30
imprimirAmigos 7 #o usuário não existe mais, não imprime nada

removerUsuario 7 #o usuário não existe mais, não exclui nenhum usuário
removerAmigo 4 3 #a amizade não existe mais, não tem amizade para remover
removerAmigo 9 8 #usuários não existem, não tem amizade para remover

#imprimirUsuarios
```

Saída

O programa deve imprimir a saída no arquivo texto *saida.txt*, sendo um resultado por linha. Exemplo de saída:

```
O usuário Joao (6) foi adicionado na rede.
O usuário Pedro (4) foi adicionado na rede.
O usuário Paulo (3) foi adicionado na rede.
O usuário Maria (7) foi adicionado na rede.
O usuário Francisco (30) foi adicionado na rede.
Erro ao adicionar o usuário Mariana (30). O ID 30 já existe!
Os usuários Joao (6) e Pedro (4) se tornaram amigos.
Erro ao criar amizade dos usuários com IDs 6 e 4!
Os usuários Joao (6) e Paulo (3) se tornaram amigos.
Os usuários Maria (7) e Joao (6) se tornaram amigos.
Os usuários Maria (7) e Paulo (3) se tornaram amigos.
Os usuários Paulo (3) e Pedro (4) se tornaram amigos.
Erro ao criar amizade dos usuários com IDs 8 e 9!
Usuários da rede: Paulo (3), Pedro (4), Joao (6), Maria (7), Francisco (30)
Amigos de Paulo (3): Pedro (4), Joao (6), Maria (7)
Amigos de Pedro (4): Paulo (3), Joao (6)
Amigos de Joao (6): Paulo (3), Pedro (4), Maria (7)
Amigos de Maria (7): Paulo (3), Joao (6)
Amigos de Francisco (30):
Erro ao imprimir amigos do usuário com ID 9. O usuário não existe!
Média de idade dos amigos de Joao (6): 21.6667
Média de idade dos amigos de Pedro (4): 25
Média de idade dos amigos de Paulo (3): 31.6667
Média de idade dos amigos de Maria (7): 25
Média de idade dos amigos de Francisco (30):
Erro ao imprimir a média de idade dos amigos do usuário com ID 9!
Amigos em comum entre Joao (6) e Paulo (3) (todos os sexos): Pedro (4), Maria (7)
Amigos em comum entre Joao (6) e Paulo (3) (sexo masculino): Pedro (4)
Amigos em comum entre Joao (6) e Paulo (3) (sexo feminino): Maria (7)
Os usuários Pedro (4) e Paulo (3) não são mais amigos
Amigos em comum entre Joao (6) e Paulo (3) (todos os sexos): Maria (7)
Amigos em comum entre Joao (6) e Paulo (3) (sexo masculino):
Amigos em comum entre Joao (6) e Paulo (3) (sexo feminino): Maria (7)
Erro ao imprimir amigos em comum dos usuários com IDs 8 e 9!
O usuário Maria (7) foi excluído da rede.
Usuários da rede: Paulo (3), Pedro (4), Joao (6), Francisco (30)
Amigos de Paulo (3): Joao (6)
Amigos de Pedro (4): Joao (6)
Amigos de Joao (6): Paulo (3), Pedro (4)
Amigos de Francisco (30):
Erro ao imprimir amigos do usuário com ID 7. O usuário não existe!
Erro ao excluir o usuário com ID 7. O usuário não existe!
Erro ao remover a amizade. Não existe amizade entre os usuários com IDs 4 e 3!
Erro ao remover a amizade. Não existe amizade entre os usuários com IDs 9 e 8!
```

Orientações para o desenvolvimento e entrega do trabalho

I. **Não é permitido** o uso de bibliotecas para a implementação da estrutura de dados lista, tal como "#include <list>". Utilize os códigos desenvolvidos em aula para a implementação da estrutura de dados lista.
II. A entrega deverá ser feita no Moodle, tópico "Trabalho Etapa II – Lista".
III. A entrega do trabalho deve ser realizada **até às 23h:59min do dia 2 de outubro de 2021**. Os arquivos devem ser compactados em um único arquivo **ZIP**.
IV. Atenção, o projeto não pode passar de **50Mb**, pois o Moodle tem uma limitação para o tamanho do anexo.
V. Trabalho **individual.**
VI. Peso: **10 pontos.**