LABORATÓRIO DE MANIPULAÇÃO DE ARQUIVOS

Profs. Eliane Santiago e Miryam Moraes

Objetivo: Trabalhar com manipulação de arquivos, strings, e estruturas não homogêneas.

Tarefa:

No final dos cursos de Bacharelado em Ciencia da Computação e Sistemas de Informação de uma Instituição de Ensino Superior (IES), cada aluno desenvolve um Trabalho de Curso (TC). Cada departamento (COMP, SI e outros) indica os melhores TCs daquele departamento. Os melhores TCs da IES são submetidos a uma nova comissão (composta inclusive por membros externos) a fim de escolher qual TC deve ser premiado. Sua tarefa é fazer um programa para ajudar a eleição do melhor TC da IES.

Requisitos funcionais:

1. Considere que todos os arquivos que vamos trabalhar:
    a. Estão no mesmo diretório que o .exe.
    b. Possuem "enter" no final.
    c. São lidos somente uma vez cada.
    d. Se não existirem, o programa deve avisar e encerrar.

2. Considere as seguintes siglas para os departamentos da IES: "BCC", "BSI", "ADS", "GTI", "BEC".

3. Considere que a TI fornece um arquivo **"professor.txt"** com dados de todos os professores da IES. No início do arquivo há a quantidade de professores. São providas as seguintes informações de cada professor, nessa ordem: código do professor (número inteiro), departamento (string sem espaços), idade (número inteiro) e nome completo (string com espaços). Exemplo:

```
3
123 BCC 35 Fulano de Tal
567 BSI 60 Jorge da Silva
381 GTI 42 Giovanna Bastos
```

   Obs: Considere que o arquivo está bem formado e todas as informações estão corretas.

4. Considere que a TI fornece um arquivo **"aluno.txt"** com dados de todos os alunos da IES que estão no último ano. No início do arquivo há a quantidade de alunos. São providas as seguintes informações de cada aluno, nessa ordem: código de matrícula (número inteiro), ano (número inteiro, nesse caso será sempre 3, correspondendo ao último ano), departamento (string sem espaços), idade (número inteiro) e nome completo (string com espaços). Exemplo:

```
4
123 3 BCC 22 Camilla Borba
234 3 BSI 25 Mauricio Oliveira
976 3 BCC 24 Maria Clara Diniz
723 3 BEC 23 Pedro Alves
```

Obs1: Considere que o arquivo está bem formado e todas as informações estão corretas.
Obs2: Um aluno pode ter código igual ao de um professor.

5. Considere que cada departamento fornece um arquivo **"TC_X.txt"** dados dos TCs indicados pelo departamento. Onde X pode ser "BCC", "BSI", "BEC", "GTI", ou "ADS" ou outro especificado no item 2 dos Requisitos Funcionais. No início do arquivo há a quantidade de TCs. São providas as seguintes informações de cada TC, nessa ordem: código do TC (número inteiro), código do aluno autor (número inteiro), código do orientador (número inteiro) e título do trabalho (string com espaços). Exemplo arquivo "TC_COMP.txt":

```
2
45 234 567 Simulacao de Software
12 723 123 Sistema Distribuído
```

Obs1: Considere que o TC só pode ser feito individualmente.
Obs2: Considere que o TC só pode ter um orientador (de qualquer departamento).
Obs3: Não repita código, por exemplo, é necessária apenas uma função para ler esse tipo de arquivo. O que muda é somente o nome do arquivo.
Obs4: Cada departamento deve prover seu arquivo, mesmo que com quantidade zero. Se o arquivo não existir, avise ao usuário e encerre o programa.

Ao ler esse arquivo é necessário fazer as seguintes verificações:

a. Verificar se o aluno existe, para isso ver se o código existe e se o departamento está correto. Ex: 234 é um aluno da COMP por isso está correto ele estar no arquivo "TC_COMP.txt"
b. Verificar se o professor existe, para isso basta ver se o código existe. O orientado pode ser de qualquer departamento.

6. Considere que a TI fornece um arquivo **"comissao.txt"** com dados de todos que podem votar na escolha do melhor TC. No início do arquivo há a quantidade de eleitores. São providas as seguintes informações de cada eleitor, nessa ordem: CPF (string sem espaços). Exemplo:

```
3
773.459.123-02
393.486.442-20
524.740.032-15
```

   Ao ler esse arquivo é necessário fazer as seguintes verificações:
   - ✓ O formato do CPF é xxx.xxx.xxx-yy onde x e y são dígitos entre 0 e 9.
   - ✓ Para validar yy utilize o algoritmo descrito em "Algoritmo para gerar CPF.pdf"
   - ✓ Ao encontrar um CPF errado, deve-se imprimir o erro na tela e encerrar o programa.
   - ✓ Não repita código, por exemplo, é necessária somente uma função para validar CPF. Essa função será útil depois.

7. Ao iniciar, o programa carrega os dados dos arquivos acima. Cada arquivo deve ser lido apenas uma vez.

8. O programa mostra o **MENU1**:
   a) Iniciar nova votação
   b) Continuar votação gravada
   Obs: O programa deve garantir que somente 'a' ou 'b' sejam selecionados, podendo ser maiúsculas ou minúsculas.

9. Ao selecionar **"a) Iniciar nova votação"** de **MENU1**, o programa mostra o **MENU2**:
   a) Entrar com voto
   b) Suspender votação
   c) Concluir votação
   Obs: O programa deve garantir que somente 'a', 'b' ou 'c' sejam selecionados, podendo ser maiúsculas ou minúsculas.

10. Se a opção for **"a) Entrar com voto"** de **MENU2**, então o usuário entra com o número do CPF do eleitor que deseja votar.
    a. Se o CPF for inválido, o programa informa o erro, mas permite que entre com outro CPF.
    b. Se o CPF for válido, mas não pertencer ao conjunto de CPFs da comissão, o programa informa o erro e permite que entre com outro CPF.
    c. Se o CPF for válido, pertencer à comissão, mas já tiver votado, o programa informa isso e permite que entre com outro CPF.
    d. Se o CPF for válido, pertencer à comissão e ainda não tiver votado, então o usuário entra com o seu voto (código do TC).
    e. Se o código do TC for inválido (não existir em nenhum departamento), o

programa informa o erro, mas permite que entre com outro voto.

f. Se o código do TC for válido, o TC recebe mais um voto. O programa também deve guardar qual foi o voto daquele usuário.
g. Por fim, o programa apresenta novamente o **MENU2**.

11. Se a opção for **"b) Suspender votação"** de **MENU2**, então o programa gera o arquivo de saída **"parcial.txt"** com todos os votos que ocorreram até o momento. No início do arquivo há a quantidade de pessoas que já votaram. São providas as seguintes informações de cada voto, nessa ordem: CPF do eleitor (string sem espaços), e código do TC votado (número inteiro). Exemplo:

```
1
773.459.123-02 45
```

Atenção: só aparece o arquivo quem já votou!
Em seguida, o programa encerra.

12. Se a opção for **"c) Concluir votação"** de **MENU2**, então o programa gera o arquivo de saída **"resultado.txt"** com todos os votos que ocorreram até o momento. Exemplo:

```
TC vencedor
Codigo: 45
Titulo: Simulacao de Software
Aluno: Mauricio Oliveira
Depto aluno: BCC
Orientador: Jorge da Silva
Depto orientador: BSI

Eleitores que votaram
773.459.123-02 45
524.740.032-15 45

Eleitores que não votaram
393.486.442-20
```

Obs: Se houver empate, informe todos os TCs vencedores.

13. Se a opção for **"b) Continuar votação gravada"** de **MENU1**, então o programa lê o arquivo **"parcial.txt"** com os votos salvos anteriormente.
    a. Se o arquivo não existir, informe ao usuário e mostre novamente o **MENU1**.
    b. Se o arquivo existir, contabilize os votos existentes e mostre o **MENU2**.

Requisitos não funcionais:

14. Utilize boas práticas de programação.

15. Defina os protótipos das funções.

16. Como variáveis globais, deixe apenas vetores e sua respectiva dimensão. Evite outras variáveis globais.

17. Não precisa se preocupar com acentuação nas palavras.

18. Utilize as seguintes estruturas não homogêneas e os seguintes vetores/dimensões:

```
#define max 50
#define maxNome 60
#define maxSigla 10


typedef struct Pessoa Pessoa;
struct Pessoa {
      char nome[maxNome];    //nome da pessoa
      int idade;             //idade da pessoa
};

typedef struct Professor Professor;
struct Professor {
      Pessoa pes;    //assim professor tem nome e idade
      int codigo;    //código do professor
      char depto[maxSigla];    //departamento onde trabalha
};

typedef struct Aluno Aluno;
struct Aluno {
      Pessoa pes;       //assim aluno tem nome e idade
      int matricula;    //número de matrícula do aluno
      int ano;          //ano no curso.
      char depto[maxSigla];    //departamento onde estuda
};

typedef struct TC TC;
struct TC {
      int codigo;
      int autor;      //equivalente à matrícula do aluno autor
      int orientador; //equivalente ao código do professor
      char titulo[maxNome];    //título do TC
      int qtdeVotos;           //somatório dos votos recebidos
};

typedef struct Eleitor Eleitor;
struct Eleitor {
      char cpf[15];
      bool votou;  //true se já votou; false se não votou
      int codigoTC;
};
```

19. Declare globalmente os seguintes vetores e dimensões:

```
Professor docentes[max];     //vetor de professores
int qtdeDocentes;            //qtde de professores no vetor

Aluno formandos[max];       //vetor de alunos
int qtdeFormandos;          //qtde de alunos no vetor

TC listaTCs[max];     //vetor de TCs
int qtdeTCs;         //qtde de TCs no vetor

Eleitor comissao[max];     //vetor de eleitores
int qtdeEleitores;        //qtde de eleitores no vetor
```

Entregar no MS-Teams: código fonte, arquivo txt com o nome dos membros grupo (mesmo Grupo da APS).