UNIFEI	Universidade Federal de Itajubá Instituto de Engenharia de Sistemas e Tecnologias da Informação-IESTI

9º Laboratório de ECOP12
Profª. Thatyana de Faria Piola Seraphim

Neste laboratório trabalharemos com os métodos de caminhamento em árvore e de busca em uma árvore binária. Para isso, utilizaremos uma base de dados de CEP (baseCEP.txt) do Brasil de 2015.

a) Declare uma estrutura denominada "noCEP". Ela conterá as seguintes informações relacionadas com o CEP: número do cep, uf, cidade, logradouro e dois ponteiros para elementos do tipo noCEP.

```
typedef struct no{
   char cep[9];
   char uf[3];
   char cidade[31];
   char logradouro[61];
   struct no* esq;
   struct no* dir;
} noCEP;
```

b) Declare uma variável denominada **raiz** do tipo noCEP *, que armazena endereço para o primeiro elemento de uma árvore binária.

```
noCEP *raiz;
```

c) Após o item (c) adicione uma função chamada **inserirNo *iterativa*,** que inserirá um elemento do tipo no* na estrutura **raiz**. Esta função não retorna nada (void) e recebe um único parâmetro: um ponteiro para o arquivo em formato texto (*baseCep.txt*) que armazena as informações de CEP. O arquivo será aberto na função **main**.

Esta função deve ler uma linha do arquivo e armazenar os valores de cada informação de CEP *(cep, uf, cidade e logradouro)* em um elemento do tipo noCEP*. Este elemento, após ser alocado em memória e preenchido, deverá ser inserido na árvore binária.

Para realizar a leitura dos dados no arquivo, sugere-se utilizar as funções fgets e strtok da biblioteca C. A função fgets (*stdio.h*) seria utilizada para ler toda a linha do arquivo e a função strtok (*string.h*) separa uma string em partes a partir de um delimitador informado. Veja um exemplo abaixo de uso da função strtok:

```
char frase[] = "Gosto muito de C e C++";
char *parte;
//obtem o primeiro pedaço da string
parte = (char*)strtok(frase, " ");
while(parte != NULL){
  printf("%s\n", parte);
  parte = (char*)strtok(NULL, " ");
}
```

d) Crie uma função para impressão das informações de um único elemento do tipo noCEP*. A função **imprimeNoCEP** não retorna nada (void) e recebe como parâmetro um elemento do tipo noCEP*. Para o elemento recebido serão impressas as seguintes informações: *cep, uf, cidade e logradouro*.

e) Crie três funções para impressão das informações de **árvore binária,** de acordo com cada uma das formas de caminhamento vistas: pré-ordem, em-ordem e pós-ordem. Os referidos algoritmos serão *recursivos* e denominados, respectivamente, **imprimePreOrdem**, **imprimeEmOrdem** e **imprimePosOrdem**. As funções não retornam nada e recebe como parâmetro um ponteiro para noCEP (noCEP*). Imprima as informações dos **10 primeiros elementos** do tipo noCEP * visitados com uma chamada à função **imprimeNoCEP**, descrita no item (d).

f) Após o item (e), crie uma a função **buscarArvoreBinaria**. Esta função poderá ser ***recursiva*** ou **iterativa** e será responsável pera busca realizada sobre a estrutura árvore binária. Para isso, ela receberá dois parâmetros: um ponteiro para noCEP (noCEP *), correspondente ao nó da subárvore que está sendo analisado, e um vetor de caracteres correspondente ao número do *CEP* a ser procurado. Caso o *CEP* exista, é realizada a impressão dos dados do cep procurado: *cep, uf, cidade e logradouro*. Caso o *CEP* não seja encontrado, deverá ser impresso na tela uma mensagem de alerta ao usuário de que o CEP não foi encontrado.

g) Após o item (f), adicione a função main e acrescente o código para teste das funções até então implementados.

h) Implemente uma função chamada **destroiArvore** para liberar todo o espaço de memória alocado a cada inserção de um nó na árvore.