MAC122 – Princípios de Desenvolvimento de Algoritmos
Segundo Semestre de 2013 – BMAC – IMEUSP – Prof. Marcilio
EXERCÍCIO PROGRAMA III – Entregar até 01/Dez/2013

Dado um arquivo texto onde cada linha contém dados de uma pessoa. Os dados são os seguintes:

```
char IDENT[8];  /* 1 - número da identidade */
char NOME[40];  /* 2 - nome                 */
char DATAN[10]; /* 3 - data do nascimento   */
char CEP[9];    /* 4 - CEP da residência    */
```

Os campos estão separados por vírgula no arquivo. O nome tem tamanho máximo 40. Os demais campos tem tamanho fixo.

```
32343245,José de Castro,16/11/1958,05454-600
43456790,Maria das Dores e Silva,30/10/1973,04632-900
22589349,Antonio Paixão dos Santos,12/03/2001,09872-813
...
```

O programa deve ler este arquivo e classificá-lo crescente por um dos campos usando um algoritmo $O(n^2)$ ou O(log n). Para isso o programa deve solicitar além do nome do arquivo destino, o campo e o algoritmo. Exemplo:

```
Entre com o nome do arquivo destino: meuarquivo
Entre com o campo de classificação (I, N, D ou C): N
Entre com o método de classificação (Q ou L): L
```

Neste exemplo o programa classifica o arquivo por nome, usando o método logarítmico colocando-o no arquivo de nome **meuarquivo.txt**.

Como o nome possui letras maiúsculas, minúsculas e acentuadas, é conveniente ao comparar transformar todas para maiúsculas sem acento para que a ordem fique correta. Mas o conteúdo original do nome deve ser mantido.

Para tornar a classificação rápida, é melhor ler o arquivo todo e colocá-lo numa matriz TAB[MAX][70]. Note que cada linha tem no máximo 70 caracteres. Supor MAX 10.000.

Escreva duas funções de classificação de TAB[][]:

- **MétodoQ(char TAB[][70], int NL, char Campo)** – classifica a matriz TAB de NL linhas pelo campo **Campo**. Escolher entre os algoritmos seleção, bolha, inserção.

- **MétodoL(char TAB, int NLinhas, char Campo)** – classifica a matriz TAB de NL linhas pelo campo **Campo**. Escolher entre os algoritmos quick, merge e heap.

Escrever também uma função que verifique se a tabela está ordenada:

VerifClass(char TAB[][70], int NL, char campo[]) – verifica se **TAB[i][] ≤ TAB[i+1][]** comparando **campo** que pode ser **"I", "N", "D" ou "C"**

O programa fica então:

```
entre com o nome do arquivo origem ArqO;
while (1) {
    entre com:
        nome do arquivo destino ArqD;
        campo (I, N, D ou C);
        método de classificação (Q ou L);
    carregue ArqO em TAB[][];
    classifique TAB pelo método escolhido;
    Mostre quanto tempo levou e o número de trocas;
    VerifClass(TAB, Nreg, campo)
    Gravar TAB[][] em ArqD;
}
```

Para calcular o tempo gasto na classificação use a função clock como no exemplo abaixo:

```
#include <time.h>
. . .
double time1, time2, timedif;
  time1 = (double) clock() / (double)CLOCKS_PER_SEC;
  /* trecho do programa a cronometrar.....
  time2 = (double) clock() / (double)CLOCKS_PER_SEC;
  timedif = time2 - time1;
```

O programa abaixo gera um arquivo para ser testado pelo seu programa.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>
#define NUSP 91827364

void GeraPessoa(char * p) {
  static char n1[20][10] = {"Felícia", "Catuló", "Ósmund",
"Artêmio", "Senízio", "Anisião", "Lâmpera", "Ligúria",
"Marrento", "Túlia", "Antônio", "Marcos", "José", "Abel",
"Alfredo", "Carla", "João","Moacir","Orlando","Paulo"};
  static char n2[20][10] = {"Cartuxo", "Olambro", "Rômulo",
"Âmbulo", "Átomon", "Adrianov", "Ênio", "Élcio", "Ceni",
"Ôstego", "Úmbria", "Paula", "Ciranda", "Abelato", "Alfinet",
"Tchucar", "Jojoba","Mococa","Alves",""};
  static char n3[20][10] = {"Sereno","Soterno", "Monções",
"Óscaran", "Tôpovi", "Chrevsky", "Juliern", "Barentin", "Otelo",
"Nerengue", "Almeida", "Arcanjo", "Souza", "Araujo", "Monteiro",
"Cardoso","Lima", "Silva","Matias", ""};
  static char n4[20][10] = {"Lésmia", "Mântega", "Casas",
"Lorentão", "Melkioz", "Thunder", "Ésquiso", "Flâmula", "Champs",
"Ontup", "Castrões",
"Lemíntece","Setúbal","Matos","Silva","Santos", "Neves",
"Cabral", "Gonzaga", "Costa"};
```

```
    int i,j,k;
    char dd[20];
    /* Gera dados de uma pessoa */
    /* monta IDENT */
    for (j=0;j<8;j++) p[j] = rand() % 10 + '0';
    p[8] = 0;
    /* virgula */
    strncat(p, ",", 1);
    /* primeiro nome */
    j = rand() % 20;
    strncat(p, n1[j], 10);
    strncat(p, " ", 1);
    /* segundo nome */
    j = rand() % 20;
    strncat(p, n2[j], 10);
    if (strcmp(n2[j], "") != 0) strncat(p, " ", 1);
    /* terceiro nome */
    j = rand() % 20;
    strncat(p, n3[j], 10);
    if (strcmp(n3[j], "") != 0) strncat(p, " ", 1);
    /* quarto nome */
    j = rand() % 20;
    strncat(p, n4[j], 10);
    /* data do nascimento */
    i = rand() % 29 + 1; j = rand() % 12 + 1; k = rand() % 63 +
1950;
    sprintf(dd, ",%02d/%02d/%04d", i, j, k);
    strncat(p, dd, 11);
    /* CEP */
    i = rand() % 100000; j = rand() % 1000;
    sprintf(dd, ",%05d-%03d", i, j);
    strncat(p, dd, 10);
}

int main() {
    FILE * f;
    char nomearq[20], pess[100];
    int NReg, i;
    /* Letras com acento Unicode */
    setlocale(LC_ALL, ""); srand(NUSP);
    /* nome do arquivo a ser criado */
    puts("Qual o nome do arquivo que deseja criar ?");
    gets(nomearq);
    /* acrescenta o .txt */
    strcat(nomearq, ".txt");
    /* abrir arquivo */
    f = fopen(nomearq, "w");
    /* Quantos registros tem o arquivo ? */
    puts("Quantos registros terá o arquivo ?");
    scanf("%d", &NReg);
```

```
    /* Grava NReg registros e fecha */
    for (i = 1; i <= NReg; i++) {
        GeraPessoa(pess); fputs(pess, f); fputs("\n", f);
    }
    fclose(f);
    system("pause"); return 0;
}
```