Primer Recuperatorio Primer Parcial - Programación 2 - 26/02/2004

Se dispone de un archivo binario (**histor**) con información que responde a los siguientes campos :

clave : entero largo
descr : alfanumérico de 45 carácteres válidos
exist : entero largo
prepo : entero largo
pengr : entero largo

Se solicita un proceso que leyendo el archivo cargue en dos arrays bidimensionales **toda** la información de aquellos registros cuyo campo **descr** (compuesto de palabras) tenga alguna palabra que comience con una vocal (mayúscula o minúscula . Con los registros que no cumplan esta condición, se procederá a cargar en una cola dinámica sólo el campo **clave** . Una vez terminado el proceso, se procederá a generar un nuevo archivo (**actual**) con la información de aquellos registros cuya **clave** se ha cargado en la cola y a mostrar por pantalla, con los mensajes adecuados, la información almacenada en los arrays .

Los archivos deben ser convenientemente abiertos antes de comenzar el proceso y cerrados al terminar el mismo .

Para resolver lo pedido :

Desarrolle las funciones 'primitivas' de cola con los argumentos adecuados :

```
... inicializar_cola( ... );     /* inicializa los punteros que se emplean en esta
                                    estructura dinámica */
... encolar( ... );              /* inserta un nodo con información al final de la
                                    cola */
... cola_vacia( ... );           /* determina si no quedan más nodos con información
                                    en la cola */
... desencolar( ... );           /* recupera la información del primer nodo de la cola
                                    y elimina el nodo que la contenía */
```

Desarrolle además las siguientes funciones :

```
... es_vocal( ... );             /* determina si el argumento es una vocal */
... cant_vocales_inic( ... );    /* devuelve la cantidad de vocales al comienzo de
                                    cada palabra haciendo uso de la función anterior */
... mostrar( ... );              /* muestra la cantidad de registros y la informacion
                                    cargados en los array bidimensionales */
```

No emplear ni suponer variables globales .
Las funciones deben recibir punteros (consulte) .

```
/*** recp1_04.c ***/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>


#define  TAM               50

/*** archivos        ***/
typedef struct
{
   long           clave;
   char           descr[46];
   long           exist;
   long           prepo;
   long           pengr;
} t_reg;

/*****************************************************************************
 * esta funcion crea informacion de prueba en el archivo
 */
void creainfo(void)
{
   FILE          *fp          = fopen("histor", "wb");
   t_reg          datos[]     =
                        { { 1, "articulo uno",        11, 111, 1111 },
                          { 2, "segundo articulo",    22, 222, 2222 },
                          { 3, "tercer producto",     33, 333, 3333 },
                          { 4, "producto estrella",   44, 444, 4444 },
                          { 5, "quinto producto",     55, 555, 5555 } };
   fwrite(datos, sizeof(datos), 1, fp);
   fclose(fp);
}

/*** cola            ***/
typedef struct
{
   long           clave;
} t_info;

typedef struct s_nodo
{
   t_info         info;
   struct s_nodo *sig;
} t_nodo;

typedef struct
{
   t_nodo        *pri,
                 *ult;
} t_cola;

/*** prototipos      ***/
int cant_vocales_inic(const char *s);
int es_vocal(char c);
void mostrar(const char desc[][46], const long nros[][4], int cant);

void inicializar_cola(t_cola *p);
void encolar(t_cola *p, const t_info *d);
int cola_vacia(t_cola *p);
void desencolar(t_cola *p, t_info *d);

/*****************************************************************************
 * main
 */
void main(void)
```

```
{
   FILE          *ent,
                 *sal;
   t_reg          reg;

   char           descr[TAM][46];
   long           valores[TAM][4];
   int            fila             = 0;

   t_info         info;
   t_cola         cola;

   creainfo();  /* a los efectos de crear un archivo de prueba */

   inicializar_cola(&cola);

   if((ent = fopen("histor", "rb")) == NULL)
      exit(puts("Error en el archivo historicos"));

   if((sal = fopen("actual", "wb")) == NULL)
      exit(puts("Error en el archivo historicos"));

   fread(&reg, sizeof(t_reg), 1, ent);
   while(!feof(ent))
   {
      if(cant_vocales_inic(reg.descr))
      {
         if(fila == TAM)
            exit(puts("Se ha excedido el espacio de almacenamiento en array"));
         strcpy(descr[fila], reg.descr);
         valores[fila][0] = reg.clave;
         valores[fila][1] = reg.exist;
         valores[fila][2] = reg.prepo;
         valores[fila][3] = reg.pengr;
         fila++;
      }
      else
      {
         info.clave = reg.clave;
         encolar(&cola, &info);
      }
      fread(&reg, sizeof(t_reg), 1, ent);
   }

   rewind(ent);
   while(!cola_vacia(&cola))
   {
      desencolar(&cola, &info);
      fread(&reg, sizeof(t_reg), 1, ent);
      while(reg.clave != info.clave)
         fread(&reg, sizeof(t_reg), 1, ent);
      fwrite(&reg, sizeof(t_reg), 1, sal);
   }


   fclose(ent);
   fclose(sal);

   mostrar(descr, valores, fila);
}

/*****************************************************************************
 * determina si el argumento es una vocal
 */
int es_vocal(char c)
{
   if(c >= 'A' && c <= 'U')
      c += 32;
```

```
   return c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u';
}

/*****************************************************************************
 * devuelve la cantidad de vocales al comienzo de cada palabra
 */
int cant_vocales_inic(const char *s)
{
   int            vocal       = 0;

   if(*s == '\0')
      return 0;
   else
      if(es_vocal(*s))
         vocal++;
   s++;
   while(*s)
   {
      if(*(s - 1) == ' ' && es_vocal(*s))
         vocal++;
      s++;
   }
   return vocal;
}

/*****************************************************************************
 * muestra la cantidad de registros que se cargaron en los array
 *    bidimensionales y la informacion contenida en los mismos
 */
void mostrar(const char desc[][46], const long nros[][4], int cant)
{
   int            fila        = 0;

   printf("Hay %d elementos\n\n", cant);
   while(fila < cant)
   {
      printf("Clave : %ld\n"
             "Descr : %s\n"
             "exist : %ld\n"
             "prepo : %ld\n"
             "pengr : %ld\n\n",
             nros[fila][0],
             desc[fila],
             nros[fila][1],
             nros[fila][2],
             nros[fila][3]);
      fila++;

   }
}

 /*****************************************************************************
 */
void inicializar_cola(t_cola *p)
{
   p->pri = p->ult = NULL;
}

/*****************************************************************************
 */
void encolar(t_cola *p, const t_info *d)
{
   t_nodo        *nue         = (t_nodo *)malloc(sizeof(t_nodo));
   if(nue == NULL)
      exit(puts("Memoria insuficiente"));

   nue->info = *d;
   nue->sig  = NULL;
```

```
   if(p->pri == NULL)
      p->pri = nue;
   else
      p->ult->sig = nue;
   p->ult = nue;
}

/*****************************************************************************
 */
int cola_vacia(t_cola *p)
{
   return p->pri == NULL;
}

/*****************************************************************************
 */
void desencolar(t_cola *p, t_info *d)
{
   t_nodo        *elim        = p->pri;
   *d = p->pri->info;
   p->pri = p->pri->sig;
   if(p->ult == elim)
      p->ult = NULL;
   free(elim);
}
```

Contenido del archivo **histor** :

histor

```
000000  01 00 00 00 61 72 74 69  63 75 6C 6F 20 75 6E 6F  ....articulo uno
000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000030  00 00 CC CC 0B 00 00 00  6F 00 00 00 57 04 00 00  ........o...W...
000040  02 00 00 00 73 65 67 75  6E 64 6F 20 61 72 74 69  ....segundo arti
000050  63 75 6C 6F 00 00 00 00  00 00 00 00 00 00 00 00  culo............
000060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000070  00 00 CC CC 16 00 00 00  DE 00 00 00 AE 08 00 00  ................
000080  03 00 00 00 74 65 72 63  65 72 20 70 72 6F 64 75  ....tercer produ
000090  63 74 6F 00 00 00 00 00  00 00 00 00 00 00 00 00  cto.............
0000a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0000b0  00 00 CC CC 21 00 00 00  4D 01 00 00 05 0D 00 00  ....!...M.......
0000c0  04 00 00 00 70 72 6F 64  75 63 74 6F 20 65 73 74  ....producto est
0000d0  72 65 6C 6C 61 00 00 00  00 00 00 00 00 00 00 00  rella...........
0000e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0000f0  00 00 CC CC 2C 00 00 00  BC 01 00 00 5C 11 00 00  ....,.......\...
000100  05 00 00 00 71 75 69 6E  74 6F 20 70 72 6F 64 75  ....quinto produ
000110  63 74 6F 00 00 00 00 00  00 00 00 00 00 00 00 00  cto.............
000120  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000130  00 00 CC CC 37 00 00 00  2B 02 00 00 B3 15 00 00  ....7...+.......
000140
```

Contenido del archivo **actual** :

actual

```
000000  03 00 00 00 74 65 72 63  65 72 20 70 72 6F 64 75  ....tercer produ
000010  63 74 6F 00 00 00 00 00  00 00 00 00 00 00 00 00  cto.............
000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000030  00 00 CC CC 21 00 00 00  4D 01 00 00 05 0D 00 00  ....!...M.......
000040  05 00 00 00 71 75 69 6E  74 6F 20 70 72 6F 64 75  ....quinto produ
000050  63 74 6F 00 00 00 00 00  00 00 00 00 00 00 00 00  cto.............
000060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000070  00 00 CC CC 37 00 00 00  2B 02 00 00 B3 15 00 00  ....7...+.......
000080
```

Salida por pantalla :

Rec1_04

```
Hay 3 elementos

Clave : 1
Descr : articulo uno
exist : 11
prepo : 111
pengr : 1111

Clave : 2
Descr : segundo articulo
exist : 22
prepo : 222
pengr : 2222

Clave : 4
Descr : producto estrella
exist : 44
prepo : 444
pengr : 4444

_
```