1º ASI – Fundamentos de Programación

Ejercicio de Archivos Indexados (Videoclub)

Notas previas

Para simplificar la resolución del ejercicio se han asignado los códigos de película automáticamente. Cuando se inserta una nueva película, se busca un hueco en el archivo (de una película anteriormente borrada). Si se encuentra un hueco, se asigna ese código a la nueva película. Así no hay que modificar el archivo de índices. Y si no existe ningún hueco, se inserta la película al final del archivo de datos, asignándole el código de la última película más uno. De este modo, sólo hay que modificar la última entrada del archivo de índices, o agregar una nueva entrada si el segmento ya estaba lleno.

Con esta forma de asignar códigos a las nuevas películas se matan tres pájaros de un tiro:

- Sólo hay que tocar el último registro del archivo de índices (o añadir un registro nuevo si el último segmento estaba lleno)
- No hay que crear área de excedentes.
- El archivo de datos siempre se mantiene ordenado.

Nótese que esto puede hacerse por la naturaleza peculiar de los datos del fichero. En otros programas, esta asignación consecutiva de códigos quizás no pueda hacerse. Cada caso debe ser estudiado por separado.

Como siempre, se recomienda revisar atentamente el código del programa (a ser posible, después de haber intentado hacerlo) y preguntar cualquier cosa que no se entienda.

```
// ------------------------------------------------------------------------------------------------------------
// Definición de constantes
// ------------------------------------------------------------------------------------------------------------

#define ARCHIVO_DATOS "video.dat"
#define ARCHIVO_INDICE "video.idx"
#define TAM_SEG 3  // Tamaño del segmento

// ------------------------------------------------------------------------------------------------------------
// Definición de tipos de datos
// ------------------------------------------------------------------------------------------------------------
struct s_pelicula
{
   char titulo[50];
   char director[20];
   char reparto[200];
   char genero[20];
   char nacionalidad[10];
   int duracion;
   char borrado;
   int codigo;
};

struct s_indice
{
    long int num_segmento;
    long int direccion_primero;
    int clave_ultimo;
};


// ------------------------------------------------------------------------------------------------------------
// Prototipos de funciones
// ------------------------------------------------------------------------------------------------------------
void borrar_pantalla(void);
void pulsar_tecla(void);
int comprobar_archivo(void);
int menu(void);
void insertar_pelicula(int *n_regs);
void listar_datos(void);
void importar_datos(int *n_regs);
void buscar_por_codigo(void);
void reconstruir_indice(void);
void mostrar_indice(void);
```

```
void introducir_datos_pelicula(struct s_pelicula* peli);
void borrar_pelicula(void);
void modificar_pelicula(void);


                                                                                                                    //
-------------------------------------------------------------------------------------------------------------------
// Función main()
// -------------------------------------------------------------------------------------------------------------------
int main(void)
{
   int opc, n_regs;

   // Comprueba si el archivo de datos existe. Si no existe, lo crea.
   n_regs = comprobar_archivo();

   do
   {
      borrar_pantalla();
      opc = menu();
      switch (opc)
      {
        case 1: insertar_pelicula(&n_regs); break;
        case 2: buscar_por_codigo(); break;
        case 3: borrar_pelicula(); break;
        case 4: modificar_pelicula(); break;
        case 5: listar_datos(); break;
        case 6: mostrar_indice(); break;
        case 7: reconstruir_indice(); break;
      }
   }
   while (opc != 0);     // 0 es "salir"

   return 0;
}

// -------------------------------------------------------------------------------------------------------------------
// Borra la pantalla de texto
// -------------------------------------------------------------------------------------------------------------------
void borrar_pantalla(void)
{
    int i;
    for (i=0; i<40; i++)
       printf("\n");
}


// -------------------------------------------------------------------------------------------------------------------
// Muestra el texto "Pulse Intro para continuar" y espera a que el usuario pulse Intro
// -------------------------------------------------------------------------------------------------------------------
void pulsar_tecla(void)
{
   printf("Pulse Intro para continuar...\n");
   getchar();
}


// -------------------------------------------------------------------------------------------------------------------
// Muestra el menú y lee la opción del usuario. Devuelve el código de la opción elegida.
// -------------------------------------------------------------------------------------------------------------------
int menu(void)
{
   int opc = 1;
   char aux[50];

   printf("Fundamentos de programación - 1º ASI\n");
   printf("EJERCICIO DEL VIDEOCLUB\n");
   printf("--------------------------------------\n");

   // Muestra el menú
   printf("Menú de opciones\n\n");
   printf("1. Añadir película\n");
   printf("2. Buscar película\n");
   printf("3. Borrar película\n");
   printf("4. Modificar película\n");
```

```
   printf("5. Listar películas\n");
   printf("6. Mostrar índice (para depuración)\n");
   printf("7. Reconstruir índice\n");
   printf("0. Salir\n\n");

   // Leer opción seleccionada
   printf("Seleccione opción --> ");
   gets(aux);
   opc = atoi(aux);

   return opc;
}

// ------------------------------------------------------------------------------------------------------------
// Comprueba si el archivo de datos existe. Si no existe, lo crea.
// Devuelve el código del último registro almacenado en el archivo.
// ------------------------------------------------------------------------------------------------------------
int comprobar_archivo(void)
{
    FILE *f;
    struct s_pelicula peli;
    int n_regs;

    f = fopen(ARCHIVO_DATOS, "r+");
    if (f == NULL)        // El archivo no existe: se crea y se pone el nÂº de registros a 0
    {
        f = fopen(ARCHIVO_DATOS, "a");
        if (f == NULL)
        {
                printf("Error al crear el archivo de datos\n");
                return 0;
        }
        n_regs = 0;
        fclose(f);
    }
    else                // El archivo sí existe: mirar cuál es el código del último registro
    {
        fseek (f, -sizeof(struct s_pelicula), SEEK_END);
        fread(&peli, sizeof(struct s_pelicula), 1, f);
        n_regs = peli.codigo;
        fclose(f);
    }

    return n_regs;
}


// ------------------------------------------------------------------------------------------------------------
// Añade una película al final del archivo de datos. El código de la nueva película se
// asigna automáticamente sumándole uno a la última película que exista en el archivo, así
// que la nueva película siempre se agregará al final del último segmento */
// ------------------------------------------------------------------------------------------------------------
void insertar_pelicula(int *n_regs)
{
   FILE *f, *f_ind;
   struct s_pelicula nuevo, peli;
   struct s_indice entrada_indice;
   int n, terminado = 0;

   // Introducir por teclado los datos del nuevo registro
   borrar_pantalla();
   printf("INSERTAR NUEVA PELICULA\n");
   printf("Introduzca los datos de la película\n");

   // Introducir por teclado los datos de la película
   introducir_datos_pelicula(&nuevo);

   // Abrir el archivo para añadir datos en él (modo secuencial, para agregar al final)
   f = fopen(ARCHIVO_DATOS, "r+b");
   if (f == NULL)
   {
      printf("Error al abrir el archivo de datos");
      pulsar_tecla();
   }
   else
```

```
   {
      // Buscamos un hueco donde insertar la película
      while ((!feof(f)) && (terminado == 0))
      {
         n = fread(&peli, sizeof(struct s_pelicula), 1, f);
         if ((n > 0) && (peli.borrado == '*'))         // Hemos encontrado un hueco
         {
             fseek(f, -sizeof(struct s_pelicula), SEEK_CUR);         // Retrocedemos
             nuevo.codigo = peli.codigo;                             // Reaprovechamos el código antiguo
             fwrite(&nuevo, sizeof(struct s_pelicula), 1, f);
             terminado = 1;
         }
      }

      if (terminado == 0)             // No se encontró ningún hueco
      {
         fseek(f, 0, SEEK_END);            // Escribiremos el registro AL FINAL del archivo
         nuevo.codigo = (*n_regs) + 1;     // El código es el siguiente del último asignado
         fwrite(&nuevo, sizeof(struct s_pelicula), 1, f);
         // Al insertar al final, tenemos que actualizar el índice
         if (nuevo.codigo % TAM_SEG == 0)
         {    // Crear nueva entrada en el índice
              f_ind = fopen(ARCHIVO_INDICE, "ab");
                   if (f_ind == NULL) { printf("Película insertada, pero no puedo actualizar el índice\n");
return; }
              entrada_indice.clave_ultimo = nuevo.codigo;
              entrada_indice.direccion_primero = ftell(f);
              entrada_indice.num_segmento = nuevo.codigo / TAM_SEG;
              fwrite(&entrada_indice, sizeof(struct s_indice), 1, f_ind);
              fclose(f_ind);
         }
         else
         {    // Modificar última entrada del índice
              f_ind = fopen(ARCHIVO_INDICE, "r+b");
                   if (f_ind == NULL) { printf("Película insertada, pero no puedo actualizar el índice\n");
return; }
              fseek(f_ind, -sizeof(struct s_indice), SEEK_END);
              fread(&entrada_indice, sizeof(struct s_indice), 1, f_ind);
              entrada_indice.clave_ultimo = nuevo.codigo;
              fseek(f_ind, -sizeof(struct s_indice), SEEK_END);
              fwrite(&entrada_indice, sizeof(struct s_indice), 1, f_ind);
              fclose(f_ind);
         }
      }

      fclose(f);
      printf("Película insertada con el código %i\n", nuevo.codigo);
      (*n_regs)++;
      pulsar_tecla();
   }
}


// ------------------------------------------------------------------------------------------------------------
// Lee por teclado los datos de una película y los devuelve en el parámetro pasado por variable.
// Asigna todos los campos EXCEPTO EL CODIGO, que se asigna automáticamente en la función
// de inserción de la película.
// ------------------------------------------------------------------------------------------------------------
void introducir_datos_pelicula(struct s_pelicula* peli)
{
   char aux[50];

   printf("  Título: ");
   gets(peli->titulo);
   printf("  Director: ");
   gets(peli->director);
   printf("  Reparto: ");
   gets(peli->reparto);
   printf("  Género: ");
   gets(peli->genero);
   printf("  Nacionalidad: ");
   gets(peli->nacionalidad);
   printf("  Duración (minutos): ");
   gets(aux);
   peli->duracion = atoi(aux);
```

```
   // Ponemos la marca de "no borrado"
   peli->borrado = '-';
}


// ------------------------------------------------------------------------------------------------------------
// Localiza una película por su código
// ------------------------------------------------------------------------------------------------------------
void buscar_por_codigo(void)
{
    FILE *f, *f_ind;
    struct s_pelicula peli;
    struct s_indice entrada_indice;
    int cod_buscado, n, cont;
    char aux[50];

    f = fopen(ARCHIVO_DATOS, "r+b");
    if (f == NULL) { printf("Error al abrir archivo de datos\n"); return; }
    f_ind = fopen(ARCHIVO_INDICE, "rb");
    if (f_ind == NULL) { printf("Error al abrir archivo de índices\n"); return; }

    borrar_pantalla();
    printf("BUSCAR UNA PELÍCULA POR CÓDIGO\n");
    printf("Introduzca el código buscado\n");
    gets(aux);
    cod_buscado = atoi(aux);

    // Buscamos entrada del indice
    entrada_indice.clave_ultimo = -9999;
    while ((!feof(f_ind)) && (entrada_indice.clave_ultimo < cod_buscado))
        fread(&entrada_indice, sizeof(struct s_indice), 1, f_ind);

    if (cod_buscado <= entrada_indice.clave_ultimo)
    {
        // Buscamos película
        fseek(f, entrada_indice.direccion_primero, SEEK_SET);
        cont = 0;
        do
        {
                fread(&peli, sizeof(struct s_pelicula), 1, f);
                cont++;
        }
        while ((cont < TAM_SEG) && (peli.codigo != cod_buscado));

        // Película encontrada: mostrar
        if ((peli.codigo == cod_buscado) && (peli.borrado == '-'))
        {
               printf("DATOS DEL REGISTRO:\n");
                        printf("%-3i  %s  (%s),  %s,  %s,  %i  min\n",  peli.codigo,  peli.titulo,  peli.director,
peli.nacionalidad, peli.genero, peli.duracion);
               printf("Reparto: %s\n", peli.reparto);
        }
        else
               printf("Película no encontrada en su segmento\n");
    } // if
    else
       printf("Película no encontrada en el índice\n");

    pulsar_tecla();
    fclose(f);
    fclose(f_ind);
}

// ------------------------------------------------------------------------------------------------------------
// Muestra el contenido del archivo de datos por la pantalla
// ------------------------------------------------------------------------------------------------------------
void listar_datos(void)
{
   FILE *f;
   struct s_pelicula peli;
   int n;

   // Abre el archivo en modo de lectura secuencial
   borrar_pantalla();
   f = fopen(ARCHIVO_DATOS, "rb");
```

```
   if (f == NULL)
   {
      printf("Error al abrir el archivo de datos");
      pulsar_tecla();
   }
   else
   {
       // Escribe en la pantalla la cabecera de la lista
       while (!feof(f))
       {
           // Lee un registro del archivo
           n = fread(&peli, sizeof(struct s_pelicula), 1, f);
           // Muestra por la pantalla el contenido del registro (si no está marcado como borrado)
           if ((n>0) && (peli.borrado == '-'))
           {
               printf("--------------------------------------------------------------------------------\n");
                        printf("%-3i  %s  (%s),  %s,  %s,  %i  min\n",  peli.codigo,  peli.titulo,  peli.director,
peli.nacionalidad, peli.genero, peli.duracion);
               printf("Reparto: %s\n", peli.reparto);
           }
       } // while
       fclose(f);
       printf("--------------------------------------------------------------------------------\n");
       pulsar_tecla();
   } // else

}

// ------------------------------------------------------------------------------------------------------------
// Reconstruye el archivo índice
// ------------------------------------------------------------------------------------------------------------
void reconstruir_indice(void)
{
    FILE *f, *f_ind;
    struct s_pelicula peli;
    struct s_indice ind;
    int registro, segmento, dir_inicial, n;


    borrar_pantalla();
    printf("Reconstruyendo el índice...\n");

    // Abrimos el archivo de datos en modo secuencial
    f = fopen(ARCHIVO_DATOS, "rb");
    if (f == NULL)
    {
        printf("No se puede abrir el archivo de datos. Operación cancelada.\n");
        pulsar_tecla(); return;
    }

    // Creamos un nuevo archivo de índices
    f_ind = fopen("temp", "wb");
    if (f_ind == NULL)
    {
        printf("No se puede crear el nuevo archivo de índices. Operación cancelada.\n");
        pulsar_tecla(); return;
    }

    // Recorremos secuencialmente el archivo de datos
    segmento = 1;       // Contador de segmentos
    registro = 0;       // Contador de registros en el segmento
    dir_inicial = 0;    // Dirección inicial del segmento
    while (!feof(f))
    {
        n = fread(&peli, sizeof(struct s_pelicula), 1, f);
        if (n > 0)
        {
           registro++;
           if (registro % TAM_SEG == 0)        // Cambiar de segmento
           {
                // Creamos una entrada en el índice y la grabamos
                ind.num_segmento = segmento;
                ind.direccion_primero = dir_inicial;
                ind.clave_ultimo = peli.codigo;
                fwrite(&ind, sizeof(struct s_indice), 1, f_ind);
```

```
                // Iniciamos el nuevo segmento
                segmento++;
                dir_inicial = ftell(f);
            }
        } // if
    } // while

    // Si el último segmento no está completo, no se habrá grabado
    // su entrada en el archivo de índices, así que lo haremos ahora
    if (registro % TAM_SEG != 0)
    {
        // Creamos una entrada en el índice y la grabamos
        ind.num_segmento = segmento;
        ind.direccion_primero = dir_inicial;
        ind.clave_ultimo = peli.codigo;
        fwrite(&ind, sizeof(struct s_indice), 1, f_ind);
    }

    // Cerramos y renombramos el índice
    fclose(f);
    fclose(f_ind);
    remove(ARCHIVO_INDICE);
    rename("temp", ARCHIVO_INDICE);
    printf("Proceso concluido con éxito.\n");
    pulsar_tecla();
}


// ------------------------------------------------------------------------------------------------------------
// Muestra el contenido del archivo de índices
// (SOLO CON PROPÓSITOS DE DEPURACIÓN. Esta función debería desaparecer de la versión definitiva del programa)
// ------------------------------------------------------------------------------------------------------------
void mostrar_indice(void)
{
   FILE *f;
   struct s_indice entrada_indice;
   int n;

   // Abre el archivo en modo de lectura secuencial
   borrar_pantalla();
   f = fopen(ARCHIVO_INDICE, "rb");
   if (f == NULL)
   {
      printf("Error al abrir el archivo de índices");
      pulsar_tecla();
   }
   else
   {
       printf("SEG.  DIR.  CLAVE\n");
       printf("-----------------\n");
       while (!feof(f))
       {
           // Lee un registro del archivo
           n = fread(&entrada_indice, sizeof(struct s_indice), 1, f);
           // Muestra por la pantalla el contenido del registro
           if (n>0)
           {
                  printf("%-3i    %-5i  %-4i\n", entrada_indice.num_segmento, entrada_indice.direccion_primero,
entrada_indice.clave_ultimo);
           }
       } // while
       fclose(f);
       pulsar_tecla();
   } // else

}


// ------------------------------------------------------------------------------------------------------------
// Busca una película y, si existe, permite que el usuario modifique cualquiera de sus campos (excepto el
código)
// ------------------------------------------------------------------------------------------------------------
void modificar_pelicula(void)
{
```

```
    FILE *f, *f_ind;
    struct s_pelicula peli;
    struct s_indice entrada_indice;
    int cod_buscado, n, cont;
    char aux[50];

    f = fopen(ARCHIVO_DATOS, "r+b");
    if (f == NULL) { printf("Error al abrir archivo de datos\n"); return; }
    f_ind = fopen(ARCHIVO_INDICE, "rb");
    if (f_ind == NULL) { printf("Error al abrir archivo de índices\n"); return; }

    borrar_pantalla();
    printf("MODIFICAR UNA PELÍCULA\n");
    printf("Introduzca el código de la película:\n");
    gets(aux);
    cod_buscado = atoi(aux);

    // Buscamos entrada del indice
    entrada_indice.clave_ultimo = -9999;
    while ((!feof(f_ind)) && (entrada_indice.clave_ultimo < cod_buscado))
        fread(&entrada_indice, sizeof(struct s_indice), 1, f_ind);

    if (cod_buscado <= entrada_indice.clave_ultimo)
    {
        // Buscamos película
        fseek(f, entrada_indice.direccion_primero, SEEK_SET);
        cont = 0;
        do
        {
                fread(&peli, sizeof(struct s_pelicula), 1, f);
                cont++;
        }
        while ((cont < TAM_SEG) && (peli.codigo != cod_buscado));

        // Película encontrada: mostramos los datos y pedimos su modificación. Si se pulsa "Intro", los dejamos
sin cambios.
        if ((peli.codigo == cod_buscado) && (peli.borrado == '-'))
        {
               printf("Modifique el registro (pulse Intro para dejar igual):\n");
               printf("Código: %i\n", peli.codigo);
               printf("Título (%s): \n", peli.titulo);
               gets(aux); if (strcmp(aux,"") != 0) strcpy(peli.titulo, aux);
               printf("Director (%s): \n", peli.director);
               gets(aux); if (strcmp(aux,"") != 0) strcpy(peli.director, aux);
               printf("Reparto (%s): \n", peli.reparto);
               gets(aux); if (strcmp(aux,"") != 0) strcpy(peli.reparto, aux);
               printf("Nacionalidad (%s): \n", peli.nacionalidad);
               gets(aux); if (strcmp(aux,"") != 0) strcpy(peli.nacionalidad, aux);
               printf("Género (%s): \n", peli.genero);
               gets(aux); if (strcmp(aux,"") != 0) strcpy(peli.genero, aux);
               printf("Duración (%i): \n", peli.duracion);
               gets(aux); if (strcmp(aux,"") != 0) peli.duracion = atoi(aux);

               printf("¿Confirma la modificación del registro? (S/N)");
               gets(aux);
               if ((aux[0] == 's') || (aux[0] == 'S'))
                {
                       // Retrocedemos un registro en el fichero de datos
                       fseek(f, -sizeof(struct s_pelicula), SEEK_CUR);
                       // Sobreescribimos el registro con los nuevos datos
                       fwrite(&peli, sizeof(struct s_pelicula), 1, f);
                       printf("Registro modificado\n");
               }
               else
                       printf("Operación cancelada\n");
        }
        else
                printf("Película no encontrada en su segmento\n");
    } // if
    else
       printf("Película no encontrada en el índice\n");

    pulsar_tecla();
    fclose(f);
    fclose(f_ind);
```

```
}

// ------------------------------------------------------------------------------------------------------------
// Busca una película y, si existe, la marca como borrada
// ------------------------------------------------------------------------------------------------------------
void borrar_pelicula(void)
{
    FILE *f, *f_ind;
    struct s_pelicula peli;
    struct s_indice entrada_indice;
    int cod_buscado, n, cont;
    char aux[50];

    f = fopen(ARCHIVO_DATOS, "r+b");
    if (f == NULL) { printf("Error al abrir archivo de datos\n"); return; }
    f_ind = fopen(ARCHIVO_INDICE, "rb");
    if (f_ind == NULL) { printf("Error al abrir archivo de índices\n"); return; }

    borrar_pantalla();
    printf("BORRAR UNA PELÍCULA\n");
    printf("Introduzca el código de la película:\n");
    gets(aux);
    cod_buscado = atoi(aux);

    // Buscamos entrada del indice
    entrada_indice.clave_ultimo = -9999;
    while ((!feof(f_ind)) && (entrada_indice.clave_ultimo < cod_buscado))
        fread(&entrada_indice, sizeof(struct s_indice), 1, f_ind);

    if (cod_buscado <= entrada_indice.clave_ultimo)
    {
        // Buscamos película
        fseek(f, entrada_indice.direccion_primero, SEEK_SET);
        cont = 0;
        do
        {
                fread(&peli, sizeof(struct s_pelicula), 1, f);
                cont++;
        }
        while ((cont < TAM_SEG) && (peli.codigo != cod_buscado));

        // Película encontrada: pedimos confirmación y borramos
        if ((peli.codigo == cod_buscado) && (peli.borrado == '-'))
        {
               printf("DATOS DEL REGISTRO:\n");
                         printf("%-3i  %s  (%s),  %s,  %s,  %i  min\n",  peli.codigo,  peli.titulo,  peli.director,
peli.nacionalidad, peli.genero, peli.duracion);
               printf("Reparto: %s\n\n", peli.reparto);
              printf("¿Está seguro de que desea borrar la película? (S/N)");
              gets(aux);
              if ((aux[0] == 's') || (aux[0] == 'S'))
              {
                     // Retrocedemos un registro en el fichero de datos
                     fseek(f, -sizeof(struct s_pelicula), SEEK_CUR);
                     // Sobreescribimos el registro marcando la película como borrada
                     peli.borrado = '*';
                     fwrite(&peli, sizeof(struct s_pelicula), 1, f);
                     printf("Registro borrado\n");
              }
              else
                     printf("Operación cancelada\n");
        }
        else
               printf("Película no encontrada en su segmento\n");
    } // if
    else
       printf("Película no encontrada en el índice\n");

    pulsar_tecla();
    fclose(f);
    fclose(f_ind);
}
```