FUNDAMENTOS DE PROGRAMACIÓN – 1º ASI

IMPLEMENTACIÓN DE UNA **AGENDA DE CONTACTOS** UTILIZANDO ARCHIVOS DIRECTOS

(Incluye todos los ejercicios de archivos directos del Tema 4)

Modifica el programa de la **agenda de contactos** que hicimos en el tema 3. El programa debe disponer ahora de un **menú de opciones** que le permita:

1. ***Añadir nuevos contactos**. Asígnale un número de orden a cada contacto. El primero será el 1, el segundo el 2, etc. Añade ese número a la estructura de datos del contacto. Luego, graba los contactos en un archivo binario.*
2. ***Listar todos los contactos**. Hay que recorrer el archivo desde el principio, mostrando en la pantalla toda la información guardada en él.*
3. ***Modificar un contacto**. El programa nos pedirá un nombre y buscará los contactos que coincidan con ese nombre. Si encuentra alguno, pedirá al usuario que introduzca por teclado los nuevos datos de ese contacto y los grabará en el archivo, sustituyendo a los datos que hubiera antes.*
4. ***Borrar un contacto**. El programa pedirá un nombre y buscará los contactos que coincidan con él, eliminándolos del archivo mediante el procedimiento de marcarlos como borrados (no borrándolos físicamente)*
5. ***Buscar un contacto**. Existirán dos formas de búsqueda: por nombre (búsqueda secuencial) o por número de contacto (búsqueda directa)*

Además, hay que añadirle la posibilidad de **insertar** los contactos nuevos al final del archivo o **en los huecos** dejados libres al borrar otros contactos. También añadiremos la posibilidad de **ordenar** el archivo alfabéticamente, y la operación de **compactación**, que elimine realmente los huecos que se quedan al borrar registros.

PROPUESTA DE SOLUCIÓN

```
#include <stdio.h>
#include <conio.h>
// EJERCICIOS DE FUNDAMENTOS DE PROGRAMACION
// TEMA 4: FICHEROS
// AGENDA DE CONTACTOS (versión mejorada con menús y colorines)


#define ARCHIVO_DATOS "agenda.dat"

struct s_contacto                  // Declaración de la estructura del registro
{
   char nombre[100];
   char telef[10];
   char direc[100];
   char email[100];
   char borrado;
};

void borrar_pantalla(void);        // Prototipos de funciones
void pulsar_tecla(void);
int menu(void);
void insertar_datos_final(void);
void insertar_datos_hueco(void);
void listar_datos(void);
void borrar_registro(void);
void modificar_registro(void);
void buscar_por_nombre(void);
void buscar_por_numero(void);
void ordenar(void);
void compactar(void);
```

```
void introducir_datos_contacto(struct s_contacto* contacto);


int main(void)
{
   int opc;

   do
   {
      borrar_pantalla();
      opc = menu();
      switch (opc)
      {
        case 1: insertar_datos_final(); break;
        case 2: insertar_datos_hueco(); break;
        case 3: listar_datos(); break;
        case 4: modificar_registro(); break;
        case 5: borrar_registro(); break;
        case 6: buscar_por_nombre(); break;
        case 7: buscar_por_numero(); break;
        case 8: ordenar(); break;
      }
   }
   while (opc != 9);     // 9 es "salir"

   // Antes de salir del programa, compactamos el archivo de datos
   compactar();

   system("PAUSE");
   return 0;
}

// Borra la pantalla de texto
void borrar_pantalla(void)
{
   textbackground(RED); textcolor(YELLOW);
   clrscr();
   gotoxy(1,1);
}

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


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

   textcolor(WHITE); textbackground(RED);
   gotoxy(21,1); printf("Fundamentos de programacion - 1ASI");
   gotoxy(21,2); printf("EJERCICIO DE LA AGENDA DE CONTACTOS");

   do
   {
```

```
    // Muestra el menú
    textcolor(YELLOW); textbackground(GREEN);
    gotoxy(22,5);  printf("+------------------------------+\n");
    gotoxy(22,6);  printf("|             MENU             |\n");
    gotoxy(22,7);  printf("+------------------------------+\n");
    gotoxy(22,8);  printf("|      Insertar al final       |\n");
    gotoxy(22,9);  printf("|      Insertar en un hueco    |\n");
    gotoxy(22,10); printf("|      Listar agenda           |\n");
    gotoxy(22,11); printf("|      Modificar contacto      |\n");
    gotoxy(22,12); printf("|      Borrar contacto         |\n");
    gotoxy(22,13); printf("|      Buscar por nombre       |\n");
    gotoxy(22,14); printf("|      Buscar por número       |\n");
    gotoxy(22,15); printf("|      Ordenar                 |\n");
    gotoxy(22,16); printf("|      Salir                   |\n");
    gotoxy(22,17); printf("+------------------------------+\n");

    // Resalta la opción actualmente seleccionada
    gotoxy(22, 7 + opc); textcolor(WHITE); textbackground(BLUE);
    switch(opc)
    {
      case 1:   printf("|      Insertar al final       |\n"); break;
      case 2:   printf("|      Insertar en un hueco    |\n"); break;
      case 3:   printf("|      Listar agenda           |\n"); break;
      case 4:   printf("|      Modificar contacto      |\n"); break;
      case 5:   printf("|      Borrar contacto         |\n"); break;
      case 6:   printf("|      Buscar por nombre       |\n"); break;
      case 7:   printf("|      Buscar por número       |\n"); break;
      case 8:   printf("|      Ordenar                 |\n"); break;
      case 9:   printf("|      Salir                   |\n"); break;
    }

    // Leer el teclado y actualiza la opción activada
    tecla = getch();
    if (tecla == '+') { opc++; if (opc == 10) opc = 1; }
    if (tecla == '-') { opc--; if (opc == 0) opc = 9; }
  }
  while (tecla != 13);     // El código ASCII de "Return" es el 13

  return opc;
}

// Añade un contacto al final del archivo de datos
void insertar_datos_final(void)
{
  FILE *f;
  struct s_contacto nuevo;

  // Introducir por teclado los datos del nuevo contacto
  borrar_pantalla();
  printf("INSERTAR NUEVO CONTACTO\n");
  printf("Introduzca los datos del contacto.\n");
  introducir_datos_contacto(&nuevo);

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

```
   else
   {
      // Escribe el registro en el archivo y lo cierra
      fwrite(&nuevo, sizeof(struct s_contacto), 1, f);
      fclose(f);
      printf("Contacto insertado con ï¿½ito\n");
      pulsar_tecla();
   }
}


// Añade un contacto al final del archivo de datos
void insertar_datos_hueco(void)
{
   FILE *f;
   struct s_contacto nuevo, contacto;
   int n, hueco_encontrado;

   // Introducir por teclado los datos del nuevo contacto
   borrar_pantalla();
   printf("INSERTAR NUEVO CONTACTO\n");
   printf("Introduzca los datos del contacto.\n");
   introducir_datos_contacto(&nuevo);

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

   // Buscar un hueco donde insertar el dato
   hueco_encontrado = 0;
   while ((!feof(f)) && (hueco_encontrado == 0))
   {
      n = fread(&contacto, sizeof(struct s_contacto), 1, f);
      if ((n > 0) && (contacto.borrado == 'S'))      // Hemos encontrado un hueco
            hueco_encontrado = 1;
   }

   if (hueco_encontrado == 1)
   {  // Si se encontró un hueco, insertamos el dato en él
      fseek(f, -sizeof(struct s_contacto), SEEK_CUR);      // Hacemos retroceder el cursor
      fwrite(&nuevo, sizeof(struct s_contacto), 1, f);     // Escribimos el registro
   }
   else
   {   // Si no se han encontrado huecos, insertamos el dato al final del archivo
      printf("\n¡NO EXISTEN HUECOS!.\n");
      printf("El registro se insertará al final del archivo...\n");
      fseek(f, 0, SEEK_END);                               // Nos desplazamos al final
      fwrite(&nuevo, sizeof(struct s_contacto), 1, f);      // Escribimos el registro
   }

   printf("Contacto insertado con éxito\n");
   fclose(f);
   pulsar_tecla();
}
```

```
// Muestra por la pantalla el contenido del archivo de datos en forma tabular
void listar_datos(void)
{
   FILE *f;
   struct s_contacto contacto;
   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
       printf("NOMBRE               DIRECCIÓN            TELÉFONO    E-MAIL\n");
       printf("-------------------------------------------------------------------------------\n");
       while (!feof(f))
       {
           // Lee un registro del archivo
           n = fread(&contacto, sizeof(struct s_contacto), 1, f);
           // Muestra por la pantalla el contenido del registro (si no está marcado como borrado)
           if ((n>0) && (contacto.borrado == 'N'))
               printf("%-20s    %-20s    %-11s    %-15s\n",    contacto.nombre,    contacto.direc,
contacto.telef, contacto.email);
       } // while
       fclose(f);
       pulsar_tecla();
   } // else
}


// Elimina un registro del archivo de datos. Utiliza búsqueda por nombre.
void borrar_registro(void)
{
   FILE *f;
   struct s_contacto contacto;
   int encontrado, num_reg, n;
   char nombre[100];

   // Leer por teclado el nombre que se quiere borrar
   borrar_pantalla();
   printf("BORRAR REGISTRO\n");
   printf("Introduzca el nombre del contacto que desea borrar: ");
   gets(nombre);

   // Abrimos el archivo en modo directo
   f = fopen(ARCHIVO_DATOS, "r+b");
   if (f == NULL)
   {
      printf("Error al abrir el archivo de datos\n");
      pulsar_tecla();
   }
   else
   {
      // Recorre el archivo de forma secuencial buscando el registro que se quiere borrar
      encontrado = 0;
```

```
      num_reg = 0;
      while (!feof(f))
      {
         n = fread(&contacto, sizeof(struct s_contacto), 1, f);
         if (n>0) num_reg++;                                  // Contador del nº de registros leidos
         if ((n>0) && (strcmp(contacto.nombre, nombre) == 0)) // Hemos encontado el registro
         {
            // Hemos encontrado el registro que hay que borrar.
            encontrado++;
            // Lo marcamos como borrado
            contacto.borrado = 'S';
            // Colocamos el cursor justo al principio del registro actual
            fseek(f, (num_reg - 1) * sizeof(struct s_contacto), SEEK_SET);
            // Escribimos el registro (ya marcado como borrado) en el archivo
            fwrite(&contacto, sizeof(struct s_contacto), 1, f);
            // Volvemos a dejar el cursor al final del registro
            fseek(f, num_reg * sizeof(struct s_contacto), SEEK_SET);
         }
      }
      fclose(f);
      printf("Se econtraron y borraron %i registros\n", encontrado);
      pulsar_tecla();
   } //else
}

// Modifica el contenido de un registro. Utiliza búsqueda por nombre.
void modificar_registro(void)
{
   FILE *f;
   struct s_contacto contacto;
   int encontrado, num_reg, n;
   char nombre[100];

   // Leer por teclado el nombre que se quiere borrar
   borrar_pantalla();
   printf("MODIFICAR REGISTRO\n");
   printf("Introduzca el nombre del contacto que desea modificar: ");
   gets(nombre);

   // Abrimos el archivo en modo directo
   f = fopen(ARCHIVO_DATOS, "r+b");
   if (f == NULL)
   {
      printf("Error al abrir el archivo de datos\n");
      pulsar_tecla();
   }
   else
   {
      // Recorre el archivo de forma secuencial buscando el registro que se quiere modificar
      encontrado = 0;
      num_reg = 0;
      while (!feof(f))
      {
         n = fread(&contacto, sizeof(struct s_contacto), 1, f);
         if (n>0) num_reg++;                                  // Contador del nº de registros leidos
         if ((n>0) && (strcmp(contacto.nombre, nombre) == 0))   // Hemos encontado el registro
         {
            // Hemos encontrado el registro que hay que modificar.
            encontrado++;
            // Le pedimos al usuario que vuelva a introducir los datos por teclado
```

```
            printf("Registro encontrado. Introduzca los nuevos datos:\n");
            introducir_datos_contacto(&contacto);
            // Colocamos el cursor justo al principio del registro actual
            fseek(f, (num_reg - 1) * sizeof(struct s_contacto), SEEK_SET);
            // Escribimos la estructura (que ya contiene los datos nuevos) en el archivo
            fwrite(&contacto, sizeof(struct s_contacto), 1, f);
            // Volvemos a dejar el cursor al final del registro
            fseek(f, num_reg * sizeof(struct s_contacto), SEEK_SET);
         }
      }
      fclose(f);
      printf("Se modificaron %i registros\n", encontrado);
      pulsar_tecla();
   } //else
}


// Busca un registro por nombre (búsqueda secuencial). Si lo encuentra, muestra su contenido.
void buscar_por_nombre(void)
{
   char nombre[100];
   struct s_contacto contacto;
   FILE *f;
   int encontrado, n;

   // Lee el nombre buscado por teclado
   borrar_pantalla();
   printf("BÚSQUEDA POR NOMBRE\n");
   printf("Introduzca el nombre que quiere buscar: ");
   gets(nombre);

   // Abre el archivo para lectura secuencial
   f = fopen(ARCHIVO_DATOS, "rb");
   if (f == NULL)
   {
      printf("Error al abrir el archivo\n");
      pulsar_tecla();
   }
   else
   {
       encontrado = 0;
       while (!feof(f))
       {
          n = fread(&contacto, sizeof(struct s_contacto), 1, f);
          if ((n>0) && (strcmp(contacto.nombre, nombre) == 0) && (contacto.borrado == 'N'))
          {  // Hemos encontrado el registro
             // Mostrar datos del registro por la pantalla
              encontrado = 1;
              printf("REGISTRO ENCONTRADO:\n");
             printf("Nombre: %s\n", contacto.nombre);
             printf("Dirección: %s\n", contacto.direc);
             printf("Teléfono: %s\n", contacto.telef);
             printf("e-mail: %s\n", contacto.email);
              pulsar_tecla();
           }
       }
       fclose(f);
       if (encontrado == 0)
       {
          printf("Registro no encontrado.\n");
```

```
            pulsar_tecla();
        }
    }
}


// Busca un registro por número (búsqueda directa). Si existe, muestra su contenido.
void buscar_por_numero(void)
{
   int numero;
   char aux[50];
   struct s_contacto contacto;
   FILE *f;
   int encontrado, n;

   // Lee el nombre buscado por teclado
   borrar_pantalla();
   printf("BÚSQUEDA POR NOMBRE\n");
   printf("Introduzca el número del registro que quiere buscar: ");
   gets(aux);
   numero = atoi(aux);

   // Abre el archivo para acceso directo
   f = fopen(ARCHIVO_DATOS, "r+b");
   if (f == NULL)
   {
      printf("Error al abrir el archivo\n");
      pulsar_tecla();
   }
   else
   {
       encontrado = 0;
       // Coloca el cursor en la posición del registro solicitado
       fseek(f, (numero - 1) * sizeof(struct s_contacto), SEEK_SET);
       // Lee el registro situado en esa posición
       n = fread(&contacto, sizeof(struct s_contacto), 1, f);
       // Comprueba que se haya leído algo
       if ((n>0) && (contacto.borrado == 'N'))
       {
          // Mostrar datos del registro por la pantalla
          encontrado = 1;
          printf("REGISTRO ENCONTRADO:\n");
          printf("Nombre: %s\n", contacto.nombre);
          printf("Dirección: %s\n", contacto.direc);
          printf("Teléono: %s\n", contacto.telef);
          printf("e-mail: %s\n", contacto.email);
          pulsar_tecla();
       }
       fclose(f);
       if (encontrado == 0)
       {
          printf("Registro no encontrado.\n");
          pulsar_tecla();
       }
    }
}

// Ordena el archivo de datos alfabéticamente (por el campo nombre) utilizando el
// método de la BURBUJA
void ordenar(void)
```

```
{
   FILE *f;
   struct s_contacto cj, cj1;
   int n, i, j, n_regs;

   // Abre el archivo en modo directo
   borrar_pantalla();
   f = fopen(ARCHIVO_DATOS, "r+b");
   if (f == NULL)
   {
      printf("Error al abrir el archivo de datos\n");
      pulsar_tecla();
      return;
   }

   // Contamos el número de registros del archivo
   fseek(f, 0, SEEK_END);
   n_regs = ftell(f) / sizeof(struct s_contacto);

   // Hacemos un doble bucle sobre el archivo para ejecutar una bubuja
   printf("ORDENANDO... Esta operación puede tardar unos segundos.\n");
   for (i = 0; i < n_regs; i++)
   {
      for (j = n_regs-1; j >= i; j--)
      {
          // Leemos el registro que ocupa la posición j del archivo
          fseek(f, j * sizeof(struct s_contacto), SEEK_SET);
          fread(&cj, sizeof(struct s_contacto), 1, f);
          // Leemos el registro que ocupa la posición (j-1) del archivo
          fseek(f, (j-1) * sizeof(struct s_contacto), SEEK_SET);
          fread(&cj1, sizeof(struct s_contacto), 1, f);

          // Comprobamos si están ordenados
          if ( strcmp(cj.nombre, cj1.nombre) < 0)
          {   // No están ordenados: hay que INTERCAMBIARLOS
              // Escibimos el registro que estaba en (j-1) en la posición j
              fseek(f, j * sizeof(struct s_contacto), SEEK_SET);
              fwrite(&cj1, sizeof(struct s_contacto), 1, f);
              // Escribimos el registro que estaba en j en la posición (j-1)
              fseek(f, (j-1) * sizeof(struct s_contacto), SEEK_SET);
              fwrite(&cj, sizeof(struct s_contacto), 1, f);
          } // if
       } // for j
   } // for i

   fclose(f);

   printf("El archivo ha quedado ordenado alfabéticamente.\n");
   pulsar_tecla();
}

// Compacta el archivo de datos, eliminando los huecos que pudieran existir
// después de haber hecho varias operaciones de borrado
void compactar(void)
{
   FILE *f_orig, *f_nuevo;
   struct s_contacto contacto;
   int n;

   borrar_pantalla();
```

```
   printf("COMPACTANDO... Esta operación puede tardar unos segundos.\n");

   // Abre el archivo en modo de lectura secuencial
   f_orig = fopen(ARCHIVO_DATOS, "rb");
   if (f_orig == NULL)
   {
      printf("Error al abrir el archivo de datos\n");
      pulsar_tecla();
      return;
   }

   // Abre el archivo nuevo, en el que se creará una copia del archivo de datos
   // eliminando los huecos
   f_nuevo = fopen("temporal", "wb");
   if (f_nuevo == NULL)
   {
      printf("Error al crear el archivo temporal\n");
      pulsar_tecla();
      return;
   }

   // Recorre el archivo de datos, creando una copia en el archivo temporal.
   // (Se copiarán todos los registros excepto los marcados como borrados)
   while (!feof(f_orig))
   {
       // Lee un registro del archivo
       n = fread(&contacto, sizeof(struct s_contacto), 1, f_orig);
       // Si no está borrado, lo pasa al archivo temporal
       if ((n>0) && (contacto.borrado == 'N'))
           fwrite(&contacto, sizeof(struct s_contacto), 1, f_nuevo);
   } // while
   fclose(f_orig);
   fclose(f_nuevo);

   // Elimina el archivo original y renombra el temporal, que a partir de ahora
   // será el nuevo archivo de datos (compactado)
   remove(ARCHIVO_DATOS);
   rename("temporal", ARCHIVO_DATOS);
   printf("Compactación terminada con éxito\n");
}

// Lee por teclado los datos de un contacto y los devuelve en el parámetro pasado por variable
void introducir_datos_contacto(struct s_contacto* contacto)
{
   printf("  Nombre: ");
   gets(contacto->nombre);
   printf("  Dirección: ");
   gets(contacto->direc);
   printf("  Teléfono: ");
   gets(contacto->telef);
   printf("  E-mail: ");
   gets(contacto->email);
   // Ponemos la marca de "no borrado"
   contacto->borrado = 'N';
}
```