PARA COMPILAR

```
gcc DatosPersonalesB1.c comunes.c principal.c -o compara.exe
gcc DatosPersonalesB2.c comunes.c principal.c -o compara.exe
gcc DatosPersonalesT1.c comunes.c principal.c -o compara.exe
gcc DatosPersonalesB2.c comunes.c principal.c -o compara.exe
gcc DatosPersonalesB2.c comunes.c borradoFisico.c –o borradoFisico.exe
```

ESTUCTURA DE LOS DATOS EN MEMORIA

```
struct DatosPersonales
{
  char nombre[15];
  char apellido[15];
  long dni;
  float salario;
};
```

#DEFINE

```
#define MAX_LINEA 256
#define NUM_CAMPOS 4
#define BORRADO -1
```

T1

ESTUCTURA DE LOS DATOS EN DISCO: T1

```
Presupone que la información de cada
persona está en una única línea.

NO se permiten nombres ni apellidos
compuestos (con espacios en blanco)

ej.
antonio gomez 1111 1234.56
francisco perez 2222 987.56
lucia lozano 3333 1350.9
```

T2

ESTRUCTURA DE LOS DATOS EN DISCO: T2

```
Presupone que la información de cada
persona ocupa tres líneas, una por
campo.

SI se permiten nombres y apellidos
compuestos (con espacios en blanco)

ej.
antonio jose
gomez
1111
1234.56
francisco
del monte
2222
987.56
Lucia
lozano
3333
1350.9
```

B1 y B2

ESTRUCTURA DE LOS DATOS EN DISCO: B1 y B2

```
struct DatosPersonales
{
  char nombre[15];
  char apellido[15];
  long dni;
  };
```

NOTAS:

- No es obligatorio utilizar una marca de borrado, eso dependerá del enunciado del problema.
- En el caso B2 la marca de borrado consistirá en poner el campo dni a -1

void verFichero(char *fichero)

T1

```
void verFichero(char *fichero)
{
 FILE *pFichero;
 struct DatosPersonales persona;

 pFichero = fopen(fichero, "r"); /*
```

T2

```
void verFichero(char *fichero)
{
 FILE *pFichero;
 struct DatosPersonales persona;
 char linea[MAX_LINEA];
```

B1

```
void verFichero(char *fichero)
{
 FILE *pFichero;
 struct DatosPersonales persona;

 pFichero = fopen(fichero, "rb");
```

B2

```
void verFichero(char *fichero)
{
 FILE *pFichero;
 struct DatosPersonales persona;

 pFichero = fopen(fichero, "rb"); /*
```

```
abre fichero paa lectura */
 /* Comienza a leer datos del fichero
hasta que llega al final */
 //Tambien serviría un while
(fscanf(...)!=EOF)
 while (fscanf(pFichero, "%s %s %ld
%f", persona.nombre, persona.apellido,
&persona.dni,
&persona.salario)==NUM_CAMPOS)
 {
    escribirDatosPersonales(persona);
  }
 fclose(pFichero);
}
```

```
 pFichero = fopen(fichero, "r"); /*
abre fichero para lectura */
 /* Comienza a leer datos del fichero
hasta que llega al final */
 while (fgets(linea, MAX_LINEA,
pFichero)!=NULL)
  {
    //Procesamos nombre
    if(linea[strlen(linea)-1]=='\n')
      linea[strlen(linea)-1]='\0';
    strcpy(persona.nombre, linea);

    //Procesamos apellidos - se
admiten espacios en blanco
    fgets(linea, MAX_LINEA, pFichero);
     if(linea[strlen(linea)-1]=='\n')
      linea[strlen(linea)-1]='\0';
    strcpy(persona.apellido, linea);

    //Procesamos dni
    fgets(linea, MAX_LINEA, pFichero);
    sscanf(linea, "%ld",
&persona.dni);

        //Procesamos salario
    fgets(linea, MAX_LINEA, pFichero);
    sscanf(linea, "%f",
&persona.salario);
    escribirDatosPersonales(persona);
  }
 fclose(pFichero);}
```

```
 /* Comienza a leer datos del fichero
hasta que llega al final */
 while(fread(&persona,sizeof(struct
DatosPersonales),1,pFichero) == 1)
 /* Comprueba que no ha llegado al
final del fichero */
 {
    escribirDatosPersonales(persona);
/* Muestra el registro por pantalla */
 }
 fclose(pFichero);
}
```

```
abre fichero para lectura */
 /* Comienza a leer datos del fichero
hasta que llega al final */
 while(fread(&persona,sizeof(struct
DatosPersonales),1,pFichero) == 1)
  {
   /* Hay que comprobar que no esté
marcado como borrado */
   if (persona.dni != BORRADO)
    escribirDatosPersonales(persona);
/* Muestra el registro por pantalla */
  }
 fclose(pFichero);
}
```

long contarRegistros(char *fichero)

```
long contarRegistros(char *fichero)
{
 FILE *pFichero;
 long nRegistros=0;
 struct DatosPersonales persona;

 pFichero = fopen(fichero, "r"); /*
abre fichero paa lectura */
 /* Comienza a leer datos del fichero
hasta que llega al final */
  //Tambien serviría un while
(fscanf(...)!=EOF)
  while (fscanf(pFichero, "%s %s %ld
%f", persona.nombre, persona.apellido,
                &persona.dni,
&persona.salario)==NUM_CAMPOS)

  {
    nRegistros++;
  }

 fclose(pFichero);
 return(nRegistros);
}
```

```
long contarRegistros(char *fichero)
{
 FILE *pFichero;
 long nRegistros=0;
 struct DatosPersonales persona;
 char linea[MAX_LINEA];

 pFichero = fopen(fichero, "r"); /*
abre fichero paa lectura */
 /* Comienza a leer datos del fichero
hasta que llega al final */
 while (fgets(linea, MAX_LINEA,
pFichero)!=NULL)
  {
    nRegistros++;
  }
 fclose(pFichero);
 return(nRegistros/NUM_CAMPOS);
}
```

```
long contarRegistros(char *fichero)
{
 FILE *pFichero;
 long numeroRegistros;

 /* Se abre para lectura el fichero */
 pFichero = fopen(fichero, "rb");

 /* Nos situamos al final del mismo */
 fseek(pFichero, 0, SEEK_END);

 /* ftell devuelve el numero de bytes
desde el principio del fichero
    hasta la posicion actual que es el
final del fichero. */
 numeroRegistros =
ftell(pFichero)/sizeof(struct
DatosPersonales);
 fclose(pFichero);
 return numeroRegistros;
}
```

```
long contarRegistros(char *fichero)
{
 FILE *pFichero;
 struct DatosPersonales  persona;
 long numeroRegistros=0;

 /* Se abre para lectura el fichero */
 pFichero = fopen(fichero, "rb");

 /* Nos situamos al final del mismo */
 while(fread(&persona,sizeof(struct
DatosPersonales),1,pFichero) == 1)
 {
   /*Usar fseek y ftell contaria
tambien los marcados para borrar*/
   /*solo contabilizamos si el
registro no esta marcado para borrar*/
   if(persona.dni !=BORRADO)
      numeroRegistros++;
 }
 fclose(pFichero);
 return numeroRegistros;
}
```

void anadirRegistro(char *fichero, struct DatosPersonales persona)

```
void anadirRegistro(char *fichero,
struct DatosPersonales persona)
{
 FILE *pFichero;

 pFichero = fopen(fichero, "a"); /*
abre el fichero para anadir */
 fprintf(pFichero, "%s %s %ld %.3f\n",
persona.nombre, persona.apellido,
            persona.dni,
persona.salario);  /* guarda los datos
en el fichero */
 fclose(pFichero);
}
```

```
void anadirRegistro(char *fichero,
struct DatosPersonales persona)
{
 FILE *pFichero;

 pFichero = fopen(fichero, "a"); /*
abre el fichero para anadir */
 fprintf(pFichero,
"%s\n%s\n%ld\n%.3f\n", persona.nombre,
persona.apellido,
            persona.dni,
persona.salario);  /* guarda los datos
en el fichero */
 fclose(pFichero);
}
```

```
void anadirRegistro(char *fichero, struct DatosPersonales persona)
{
 FILE *pFichero;
 /* abre el fichero para anadir */
 pFichero = fopen(fichero, "ab");
 /* guarda los datos en el fichero */
 fwrite(&persona, sizeof(struct DatosPersonales), 1, pFichero);
 fclose(pFichero);
}
```

struct DatosPersonales registro_i(char *fichero, int i)

```
struct DatosPersonales registro_i(char
*fichero, int i)
{
 FILE *pFichero;
 long j;
 struct DatosPersonales persona;

 pFichero = fopen(fichero, "r");

 for(j=0; j<i; j++)
 {
    fscanf(pFichero, "%s %s %ld %f",
persona.nombre, persona.apellido,
&persona.dni, &persona.salario);
 }
 fclose(pFichero);

 /* devuelve el registro leido */
 return persona;
}
```

```
struct DatosPersonales registro_i(char
*fichero, int i)
{
 FILE *pFichero;
 long j;
 struct DatosPersonales persona;
 char nombre[MAX_LINEA];
 char apellidos[MAX_LINEA];
 char dni[MAX_LINEA];
 char salario[MAX_LINEA];

 pFichero = fopen(fichero, "r");

 for(j=0; j<i; j++)
 {
     fgets(nombre, MAX_LINEA,
pFichero);
     fgets(apellidos, MAX_LINEA,
pFichero);
     fgets(dni, MAX_LINEA, pFichero);
     fgets(salario, MAX_LINEA,
pFichero);
 }
 //Procesamos nombre
 if(nombre[strlen(nombre)-1]=='\n')
     nombre[strlen(nombre)-1]='\0';
 strcpy(persona.nombre, nombre);

 //Procesamos apellidos - se admiten
espacios en blanco
 if(apellidos[strlen(apellidos)-
1]=='\n')
    apellidos[strlen(apellidos)-
1]='\0';
 strcpy(persona.apellido, apellidos);

 //Procesamos dni
 sscanf(dni, "%ld", &persona.dni);
```

```
struct DatosPersonales registro_i(char
*fichero, long i)
{
 FILE *pFichero;
 struct DatosPersonales aux;

 /* Se abre para lectura el fichero */
 pFichero = fopen(fichero, "rb");

 /* Nos desplazamos al final del
registro i-1, desde el principio
 del fichero con la funcion fseek*/
 fseek(pFichero, (i-1)*sizeof(struct
DatosPersonales), SEEK_SET);

 /* Se lee el siguiente registro */
 fread(&aux,sizeof(struct
DatosPersonales),1,pFichero);

 fclose(pFichero);
 /* devuelve el registro leido */
 return aux;
}
```

```
struct DatosPersonales registro_i(char
*fichero, long i)
{
 FILE *pFichero;
 long cont=0;
 struct DatosPersonales persona;

 /* Se abre para lectura el fichero */
 pFichero = fopen(fichero, "rb");

 while(cont<i)
 {
   /*Usar fseek y ftell contaria
tambien los marcados para borrar*/
   fread(&persona,sizeof(struct
DatosPersonales),1,pFichero);
   if(persona.dni !=BORRADO)
      cont++;
 }

 printf("\ncont: %d", cont);
 fclose(pFichero);
 /* devuelve el registro leido */
 return persona;
}
```

```
 sscanf(salario, "%f",
&persona.salario);

 fclose(pFichero);

 /* devuelve el registro leido */
 return persona;
}
```

int buscarporDni(char *fichero, long dni_buscar, struct DatosPersonales *persona)

```
int buscarporDni(char *fichero, long
dni, struct DatosPersonales *persona)
{
 FILE *pFichero;
 int encontrado = 0; /* variables
auxiliares para la busqueda */
 int cont;
 struct DatosPersonales auxiliar;
 /* abre fichero para lectura */
 pFichero = fopen(fichero, "r");
 while (fscanf(pFichero, "%s %s %ld
%f", auxiliar.nombre,
auxiliar.apellido,
          &auxiliar.dni,
&auxiliar.salario) == NUM_CAMPOS)
  {
    if (auxiliar.dni == dni) /* ha
encontrado el registro */
     {
      encontrado = 1;
      *persona = auxiliar; /* almacena
en persona el registro encontrado */
     }
   }
 fclose(pFichero); /* se cierra el
fichero */
 return encontrado;
}
```

```
int buscarporDni(char *fichero, long
dni_buscar, struct DatosPersonales
*persona)
{
 FILE *pFichero;
 int salir = 0;
 int encontrado = 0; /* variables
auxiliares para la busqueda */
 int cont;
 struct DatosPersonales auxiliar;
 char nombre[MAX_LINEA];
 char apellidos[MAX_LINEA];
 char dni[MAX_LINEA];
 char salario[MAX_LINEA];

 pFichero = fopen(fichero, "r"); /*
abre fichero para lectura */

 while ((fgets(nombre, MAX_LINEA,
pFichero)!=NULL)&&!encontrado)
  {
    //Leemos apellidos - se admiten
espacios en blanco
    fgets(apellidos, MAX_LINEA,
pFichero);
    //Leemos dni
    fgets(dni, MAX_LINEA, pFichero);
    sscanf(dni, "%ld", &auxiliar.dni);
    //Leemos salario
    fgets(salario, MAX_LINEA,
pFichero);

    if (auxiliar.dni == dni_buscar) /*
ha encontrado el registro */
    {
      encontrado = 1;
      //Procesamos nombre
      if(nombre[strlen(nombre)-
1]=='\n')
        nombre[strlen(nombre)-1]='\0';
      strcpy(auxiliar.nombre, nombre);

      //Procesamos apellidos - se
admiten espacios en blanco
      if(apellidos[strlen(apellidos)-
1]=='\n')
        apellidos[strlen(apellidos)-
```

La función es la misma para B1 y B2.A1 ser el dni el campo que se marca para borrar, no haría falta comprobar si dni==BORRADO en el caso de ficheros B2

```
int buscarporDni(char *fichero, long dni, struct DatosPersonales *persona)
{
 FILE *pFichero;
 int encontrado = 0;
 struct DatosPersonales auxiliar;
 /* abre fichero para lectura */
 pFichero = fopen(fichero, "rb");
 while (fread(&auxiliar, sizeof(struct DatosPersonales), 1, pFichero) == 1)
  {
    if (auxiliar.dni == dni)
    /* ha encontrado el registro*/
     {
      encontrado = 1;
      *persona = auxiliar; /* almacena en persona el registro encontrado */
     }
   }
 fclose(pFichero); /* se cierra el fichero */
 return encontrado;
}
```

```
1]='\0';
      strcpy(auxiliar.apellido,
apellidos);

      //Procesamos el salario
      sscanf(salario, "%f",
&auxiliar.salario);

          //almacena en persona el
registro encontrado
      *persona = auxiliar;
    }
  }

 fclose(pFichero); /* se cierra el
fichero */
 return encontrado;
}
```

int mostrarporNombre(char *fichero, char *auxNombre)

```
int mostrarporNombre(char *fichero,
char *auxNombre)
{
 FILE *pFichero;
 struct DatosPersonales auxiliar;
 int encontrado = 0;

 pFichero = fopen(fichero, "r"); /*
abre fichero para lectura */
 //Tambien serviría un while
(fscanf(...)!=EOF),
 while (fscanf(pFichero, "%s %s %ld
%f", auxiliar.nombre,
auxiliar.apellido,
          &auxiliar.dni,
&auxiliar.salario) == NUM_CAMPOS)

  {
   if (strcmp(auxiliar.nombre,
auxNombre) == 0)
   /* se ha encontrado un registro con
ese nombre */
    {

escribirDatosPersonales(auxiliar); /*
se escriben sus datos */
     encontrado = 1;
    }
  }
 fclose(pFichero); /* se cierra el
fichero */
 return encontrado;
}
```

```
int mostrarporNombre(char *fichero,
char *auxNombre)
{
 FILE *pFichero;
 struct DatosPersonales auxiliar;
 int encontrado = 0;
 char linea[MAX_LINEA];

 pFichero = fopen(fichero, "r"); /*
abre fichero para lectura */

 while (fgets(linea, MAX_LINEA,
pFichero)!=NULL)
  {
    //Procesamos nombre
    if(linea[strlen(linea)-1]=='\n')
      linea[strlen(linea)-1]='\0';
    strcpy(auxiliar.nombre, linea);

   //Procesamos apellidos - se
admiten espacios en blanco
    fgets(linea, MAX_LINEA, pFichero);
     if(linea[strlen(linea)-1]=='\n')
      linea[strlen(linea)-1]='\0';
    strcpy(auxiliar.apellido, linea);

    //Procesamos dni
    fgets(linea, MAX_LINEA, pFichero);
    sscanf(linea, "%ld",
&auxiliar.dni);

    //Procesamos el salario
    fgets(linea, MAX_LINEA, pFichero);
    sscanf(linea, "%f",
&auxiliar.salario);

    //se ha encontrado un registro con
```

```
int mostrarporNombre(char *fichero,
char *auxNombre)
{
 FILE *pFichero;
 struct DatosPersonales auxiliar;
 int encontrado = 0;
  /* abre fichero para lectura */
 pFichero = fopen(fichero, "rb");
 while(fread(&auxiliar,sizeof(struct
DatosPersonales),1,pFichero) == 1)
  {
   if (strcmp(auxiliar.nombre,
auxNombre)==0)
   /* se ha encontrado un registro con
ese nombre */
    {

escribirDatosPersonales(auxiliar); /*
se escriben sus datos */
     encontrado = 1;
    }
  }
 fclose(pFichero); /* se cierra el
fichero */
 return encontrado;
}
```

```
int mostrarporNombre(char *fichero,
char *auxNombre)
{
 FILE *pFichero;
 struct DatosPersonales auxiliar;
 int encontrado = 0;
 /* abre fichero para lectura */
 pFichero = fopen(fichero, "rb");
 while(fread(&auxiliar,sizeof(struct
DatosPersonales),1,pFichero) == 1)
 /* se leen todos los registros */
  {
   if (strcmp(auxiliar.nombre,
auxNombre) == 0
       && auxiliar.dni!=BORRADO)
   /* se ha encontrado un registro con
ese nombre y no está marcado */
    {

escribirDatosPersonales(auxiliar); /*
se escriben sus datos */
     encontrado = 1;
    }
  }
 fclose(pFichero); /* se cierra el
fichero */
 return encontrado;
}
```

```
ese nombre
    if (strcmp(auxiliar.nombre,
auxNombre) == 0)
    {

escribirDatosPersonales(auxiliar); /*
se escriben sus datos */
                encontrado = 1;
    }
  }

 fclose(pFichero); /* se cierra el
fichero */
 return encontrado;
}
```

int actualizarporDni(char* fichero, long dni)

```
int actualizarporDni(char* fichero,
long dni)
{
  FILE *pFichero1, *pFichero2;
  struct DatosPersonales aux;
  int encontrado = 0;

 /* Se abre para lectura el fichero
original */
 pFichero1 = fopen(fichero, "r");

 /* Fichero temporal para volcar los
registros que no se borran */
 pFichero2 = fopen("temporal", "w");

 /* Se recorre el fichero original y
los registros que no hay que borrar
    se pasan al fichero temporal */
  //Tambien serviría un while
(fscanf(...)!=EOF)
 while(fscanf(pFichero1, "%s %s %ld
%f",
       aux.nombre, aux.apellido,
&aux.dni, &aux.salario) == NUM_CAMPOS)
 {
   if (aux.dni == dni)
   {
         aux =
introducirDatosPersonales();
         encontrado = 1;
   }
   fprintf(pFichero2, "%s %s %ld
%.3f\n", aux.nombre, aux.apellido,
aux.dni, aux.salario);
 }

  /* Se cierran los ficheros */
  fclose(pFichero1);
  fclose(pFichero2);
```

```
int mostrarporNombre(char *fichero,
char *auxNombre)
{
 FILE *pFichero;
 struct DatosPersonales auxiliar;
 int encontrado = 0;
 char linea[MAX_LINEA];

 pFichero = fopen(fichero, "r"); /*
abre fichero para lectura */

 while (fgets(linea, MAX_LINEA,
pFichero)!=NULL)
  {
     //Procesamos nombre
    if(linea[strlen(linea)-1]=='\n')
      linea[strlen(linea)-1]='\0';
    strcpy(auxiliar.nombre, linea);

    //Procesamos apellidos - se
admiten espacios en blanco
    fgets(linea, MAX_LINEA, pFichero);
     if(linea[strlen(linea)-1]=='\n')
      linea[strlen(linea)-1]='\0';
    strcpy(auxiliar.apellido, linea);

    //Procesamos dni
    fgets(linea, MAX_LINEA, pFichero);
    sscanf(linea, "%ld",
&auxiliar.dni);

    //Procesamos el salario
    fgets(linea, MAX_LINEA, pFichero);
    sscanf(linea, "%f",
&auxiliar.salario);

    //se ha encontrado un registro con
ese nombre
    if (strcmp(auxiliar.nombre,
auxNombre) == 0)
```

```
int actualizarporDni(char* fichero, long dni)
{
 FILE *pFichero;
 struct DatosPersonales aux;
 int encontrado =0;

 /* Se abre para lectura y escritura */
 pFichero = fopen(fichero, "r+b");

 /* Se recorre el fichero */
 while(fread(&aux, sizeof(struct DatosPersonales),1,pFichero) == 1 )
  {
   /* Se comprueba si el registro tiene ese dni*/
   if (aux.dni == dni)
   {
     aux = introducirDatosPersonales();
     encontrado = 1;

    /* Nos situamos al principio del registro encontrado */
     fseek(pFichero, -(int)sizeof(struct DatosPersonales), SEEK_CUR);
     /* Se reescribe el registro */
     fwrite(&aux, sizeof(struct DatosPersonales), 1, pFichero);
     //La siguiente orden es necesario para que en windows
     //se actualicen los flags de FILE*
     //En Linux, no es necesario, pero se puede dejar
     //El estándar requiere fflush para hacer fread despues de fwrite
     fflush(pFichero);
    }
  }
  /* Se cierra el fichero*/
  fclose(pFichero);

  return(encontrado);
}
```

```
 /* Se borra el fichero original */
 remove(fichero);

 /* Se renombra el temporal con el
nombre del original */
 rename("temporal", fichero);

 return(encontrado);
}
```

```
    {

escribirDatosPersonales(auxiliar); /*
se escriben sus datos */
                encontrado = 1;
    }
  }

 fclose(pFichero); /* se cierra el
fichero */
 return encontrado;
}
```

int borrarporDni(char *fichero, long dni)

```
int borrarporDni(char *fichero, long
dni)
{
 FILE *pFichero1, *pFichero2;
 struct DatosPersonales aux;
 int borrado = 0;

 /* Se abre para lectura el fichero
original */
 pFichero1 = fopen(fichero, "r");

 /* Fichero temporal para volcar los
registros que no se borran */
 pFichero2 = fopen("temporal", "w");

 /* Se recorre el fichero original y
los registros que no hay que borrar
    se pasan al fichero temporal */
 //Tambien serviría un while
(fscanf(...)!=EOF)
 while(fscanf(pFichero1, "%s %s %ld
%f",
       aux.nombre, aux.apellido,
&aux.dni, &aux.salario) == NUM_CAMPOS)
  {
   if (aux.dni != dni)
    fprintf(pFichero2, "%s %s %ld
%.3f\n", aux.nombre, aux.apellido,
aux.dni, aux.salario);
   else
    borrado = 1;
  }

  /* Se cierran los ficheros */
  fclose(pFichero1);
  fclose(pFichero2);

 /* Se borra el fichero original */
 remove(fichero);

 /* Se renombra el temporal con el
nombre del original */
 rename("temporal", fichero);
```

```
int borrarporDni(char *fichero, long
dni_buscar)
{
 FILE *pFichero1, *pFichero2;
 struct DatosPersonales persona;
 char linea[MAX_LINEA];
 int borrado = 0;

 /* Se abre para lectura el fichero
original */
 pFichero1 = fopen(fichero, "r");

 /* Fichero temporal para volcar los
registros que no se borran */
 pFichero2 = fopen("temporal", "w");

 /* Se recorre el fichero original y
los registros que no hay que borrar
    se pasan al fichero temporal */

  while (fgets(linea, MAX_LINEA,
pFichero1)!=NULL)
  {
    //Procesamos nombre
    if(linea[strlen(linea)-1]=='\n')
      linea[strlen(linea)-1]='\0';
    strcpy(persona.nombre, linea);

    //Procesamos apellidos - se
admiten espacios en blanco
    fgets(linea, MAX_LINEA,
pFichero1);
     if(linea[strlen(linea)-1]=='\n')
      linea[strlen(linea)-1]='\0';
    strcpy(persona.apellido, linea);

    //Procesamos dni
    fgets(linea, MAX_LINEA,
pFichero1);
    sscanf(linea, "%ld",
&persona.dni);

    //Procesamos salario
```

```
int borrarporDni(char *fichero, long
dni_buscar)
{
 FILE *pFichero1, *pFichero2;
 struct DatosPersonales aux;
 int borrado=0;

 /* Se abre para lectura el fichero
original */
 pFichero1 = fopen(fichero, "rb");

 /* Fichero temporal para volcar los
registros que no se borran */
 pFichero2 = fopen("temporal", "wb");

 /* Se recorre el fichero original y
los registros que no hay que borrar
    se pasan al fichero temporal */

 while(fread(&aux, sizeof(struct
DatosPersonales),1,pFichero1) == 1 )
  {
   /* Se comprueba si el registro
tiene ese dni*/
   if (dni_buscar!=aux.dni)
     fwrite(&aux, sizeof(struct
DatosPersonales), 1, pFichero2);
   else
     borrado = 1;
  }
  /* Se cierran los ficheros */
  fclose(pFichero1);
  fclose(pFichero2);

 /* Se borra el fichero original */
 remove(fichero);

 /* Se renombra el temporal con el
nombre del original */
 rename("temporal", fichero);
 return(borrado);
}
```

```
int borrarporDni(char *fichero, long
dni)
{
 FILE *pFichero;
 struct DatosPersonales aux;
 int encontrado = 0;

 /* Se abre para lectura y escritura
el fichero */
 pFichero = fopen(fichero, "r+b");

 /* Se recorre el fichero original y
los registros a borrar se
    reescriben marcandolos como
borrados */
 while((fread(&aux, sizeof(struct
DatosPersonales),1,pFichero) == 1
)&&!encontrado)
  {
   /* Se comprueba si el registro
tiene ese nombre y no está marcado */
   if (aux.dni == dni)
    {
     /* Se marca como borrado */
     aux.dni = BORRADO;
     /* Nos situamos al principio del
registro encontrado */
     fseek(pFichero, -
(int)sizeof(struct DatosPersonales),
SEEK_CUR);
     /* Se reescribe el registro */
     fwrite(&aux, sizeof(struct
DatosPersonales), 1, pFichero);
     //La siguiente orden es necesario
para que en windows
     //se actualicen los flags de
FILE*
     //En Linux, no es necesario, pero
se puede dejar
     fflush(pFichero);
     /* Se ha borrado al menos un
registro */
     encontrado = 1;
```

```
 return(borrado);
}
```

```
    fgets(linea, MAX_LINEA,
pFichero1);
    sscanf(linea, "%f",
&persona.salario);

        //el registro no contiene
dni_buscar
    if (persona.dni != dni_buscar)
       fprintf(pFichero2,
"%s\n%s\n%ld\n%.3f\n", persona.nombre,
persona.apellido, persona.dni,
persona.salario);
    else
      borrado = 1;
  }

  /* Se cierran los ficheros */
  fclose(pFichero1);
  fclose(pFichero2);

  /* Se borra el fichero original */
  remove(fichero);

  /* Se renombra el temporal con el
nombre del original */
  rename("temporal", fichero);
  return(borrado);
}
```

```
    }
  }
  /* Se cierra el fichero */
  fclose(pFichero);
  return encontrado;
}
```

struct DatosPersonales* ficheroAVector(char* fichero, long* nEle)

```
struct DatosPersonales*
ficheroAVector(char* fichero, long*
nEle)
{
   FILE *pFichero;
   struct DatosPersonales persona, *V;
   long i=0;
   * nEle = contarRegistros(fichero);
   V = reservarVector(*nEle);
   pFichero = fopen(fichero, "r");
      /*Leemos todo el fichero*/
   //Tambien serviría un while
(fscanf(...)!=EOF)
   while (fscanf(pFichero, "%s %s %ld
%f", persona.nombre, persona.apellido,
                    &persona.dni,
&persona.salario)==NUM_CAMPOS)
   {
      V[i]=persona;
      i++;
   }
   fclose(pFichero);
   return(V);
}
```

```
struct DatosPersonales*
ficheroAVector(char* fichero, long*
nEle)
{
   FILE *pFichero;
   struct DatosPersonales persona, *V;
   long i=0;
   char linea[MAX_LINEA];

   *nEle = contarRegistros(fichero);
   V = reservarVector(*nEle);
   pFichero = fopen(fichero, "r");

   /*Leemos todo el fichero*/
   while (fgets(linea, MAX_LINEA,
pFichero)!=NULL)
   {
    //Procesamos nombre
    if(linea[strlen(linea)-1]=='\n')
      linea[strlen(linea)-1]='\0';
    strcpy(persona.nombre, linea);

    //Procesamos apellidos - se
admiten espacios en blanco
    fgets(linea, MAX_LINEA, pFichero);
     if(linea[strlen(linea)-1]=='\n')
      linea[strlen(linea)-1]='\0';
```

```
struct DatosPersonales*
ficheroAVector(char* fichero, long*
nEle)
{
   struct DatosPersonales* V;
   FILE* pFichero;

   *nEle = contarRegistros(fichero);
   V = reservarVector(*nEle);
   pFichero = fopen(fichero, "rb");

   /*Leemos todo el fichero (mas
rapido que elemento a elemento)*/
   fread(V, sizeof(struct
DatosPersonales),*nEle,pFichero);
   fclose(pFichero);
   return(V);
}
```

```
struct DatosPersonales*
ficheroAVector(char* fichero, long*
nEle)
{
   struct DatosPersonales* V;
   FILE* pFichero;
   long registros, i;

   //Contamos registros, incluidos los
borrados
   //contarRegistros solo cuenta los
activos, por lo que no es valida
   pFichero = fopen(fichero, "rb");
   fseek(pFichero, 0, SEEK_END);
   registros =
ftell(pFichero)/sizeof(struct
DatosPersonales);

   V = reservarVector(registros);

   /*Leemos todo el fichero (mas
rapido que elemento a elemento)*/
   fseek(pFichero, 0, SEEK_SET);
   fread(V, sizeof(struct
DatosPersonales), registros,
pFichero);
   fclose(pFichero);
```

	strcpy(persona.apellido, linea);
	
	//Procesamos dni
	fgets(linea, MAX_LINEA, pFichero);
	sscanf(linea, "%ld", &persona.dni);
	
	//Procesamos salario
	fgets(linea, MAX_LINEA, pFichero);
	sscanf(linea, "%f", &persona.salario);
	
	V[i]=persona;
	i++;
	}
	
	fclose(pFichero);
	return(V);
	}
			//Nos quedamos solamente con los registros no borrados
			for(i=0, *nEle=0; i<registros; i++)
			{
			printf("\ncopiando registro: %ld nEle: %ld", i, *nEle);
			if(V[i].dni!=BORRADO)
			{
			V[*nEle] = V[i];
			(*nEle)++; //Parentesis necesarios
			}
			}
			V= (struct DatosPersonales*) realloc(V, (*nEle)*sizeof(struct DatosPersonales));
			return(V);
			}

void vectorAFichero(char* fichero, struct DatosPersonales* V, long nEle)			

void vectorAFichero(char* fichero, struct DatosPersonales* V, long nEle)
{
FILE* pFichero;
long i=0;
 	
pFichero = fopen(fichero, "w");
for (i=0; i<nEle; i++)
fprintf(pFichero, "%s %s %ld %.3f\n", V[i].nombre, V[i].apellido, V[i].dni, V[i].salario);
fclose(pFichero);
}
	void vectorAFichero(char* fichero, struct DatosPersonales* V, long nEle)
	{
	FILE* pFichero;
	long i=0;
	
	pFichero = fopen(fichero, "wt");
	for (i=0; i<nEle; i++)
	fprintf(pFichero, "%s\n%s\n%ld\n%.3f\n", V[i].nombre, V[i].apellido, V[i].dni, V[i].salario);
	fclose(pFichero);
	}
		void vectorAFichero(char* fichero, struct DatosPersonales* V, long nEle)
		{
		FILE* pFichero;
		
		pFichero = fopen(fichero, "wb");
		fwrite(V, sizeof(struct DatosPersonales), nEle, pFichero);
		fclose(pFichero);
		}	

int borrarMarcados(char *fichero)			

			int borrarMarcados(char *fichero)
			{
			FILE *pFichero1, *pFichero2;
			struct DatosPersonales aux;
			int borrados = 0;
			
			/* Se abre para lectura el fichero original */
			pFichero1 = fopen(fichero, "rb");
			
			/* Se abre para escritura el fichero temporal para volcar los registros que no se borran */
			pFichero2 = fopen("temporal", "wb");
			
			/* Se recorre el fichero original y los registros no marcados se pasan al fichero temporal */
			while(fread(&aux,sizeof(struct DatosPersonales),1,pFichero1) == 1)

```
 {
  if(aux.dni != BORRADO)
   /* Si no está marcado el registro
se pasa al temporal */
   fwrite(&aux,sizeof(struct
DatosPersonales),1,pFichero2);
  else
   /* al menos hay uno marcado */
   borrados++;
 }

 /* Se cierran los ficheros */
 fclose(pFichero1);
 fclose(pFichero2);

 /* Se borra el fichero original */
 remove(fichero);

 /* Se renombra el temporal con el
nombre del original */
 rename("temporal", fichero);
 return borrados;
}
```

struct DatosPersonales introducirDatosPersonales()

```
struct DatosPersonales introducirDatosPersonales()
{
 struct DatosPersonales aux;

 /*Introduccion de datos */
 printf("  DNI : ");
 scanf("%ld", &aux.dni);
 getchar();

 printf("Nombre : ");
 fgets(aux.nombre, MAX_LINEA, stdin);
 aux.nombre[strlen(aux.nombre)-1]='\0';

 printf("Apellido :");
 fgets(aux.apellido, MAX_LINEA, stdin);
 aux.apellido[strlen(aux.apellido)-1]='\0';

 printf("Salario:");
 scanf("%f", &aux.salario);
 getchar();

 /* devolucion de los datos de la persona */
 return aux;
}
```

void escribirDatosPersonales(struct DatosPersonales aux)

```
void escribirDatosPersonales(struct DatosPersonales aux)
{
 printf("DNI: %ld\n\tNombre: %s  Apellido: %s  Salario: %.3f\n",
         aux.dni, aux.nombre, aux.apellido, aux.salario);
}
```

int existeFichero(char *fichero)

```
int existeFichero(char *fichero)
{
 FILE *pFichero;

 pFichero = fopen(fichero, "r"); /* abre fichero para lectura */
 if (pFichero == NULL) /* el fichero no existe */
  {
   return 0;
  }
 else /* el fichero existe */
  {
   fclose (pFichero);
   return 1;
  }
}
```

long contarBytes(char *fichero)

```
long contarBytes(char *fichero)
{
  FILE* f;
  long tam;

  if((f=fopen(fichero, "r"))==NULL)
  {
    fprintf(stderr, "\nError: no puedo abrir <%s>", fichero);exit(-1);
  }
  if(fseek(f, 0L, SEEK_END))
  {
    fprintf(stderr, "\nError: no puedo usar <%s>", fichero); exit(-1);
  }
  tam = ftell(f);
  fclose(f);
  return(tam);
}
```

struct DatosPersonales* reservarVector(long nEle)

```
struct DatosPersonales* reservarVector(long nEle)
{
   struct DatosPersonales* V=NULL;

   if((V=(struct DatosPersonales*)malloc(nEle*sizeof(struct DatosPersonales)))==NULL)
   {
      printf("\nError: No se ha podido reservar la memoria");
      exit(-1);

   return(V);
}
```

void liberarVector(struct DatosPersonales V)**

```
void liberarVector(struct DatosPersonales** V)
{
   free(*V);
   *V=NULL;
}
```

void incrementarSalarios(char* fichero, float incremento)

```
void incrementarSalarios(char* fichero, float incremento)
{
   struct DatosPersonales * V;
   long nEle;
   FILE* pFichero;
   long i;

   /*Para reducir los accesos a disco trabajamos con un vector*/
   V = ficheroAVector(fichero, &nEle);

   /*Actualizamos los datos en el vector*/
   for(i=0; i<nEle; i++)
     V[i].salario += incremento;

   /*Escribimos el vector en el fichero*/
   vectorAFichero(fichero, V, nEle);

   /*liberamos el vector*/
   liberarVector(&V);
}
```