MANTENIMIENTO DE FICHEROS

1.- Algoritmos de mantenimiento de ficheros Secuenciales

Los algoritmos más importantes necesarios para el mantenimiento de un archivo secuencial son:

- Creación de un archivo.
- Inserción de un registro en un archivo ya creado.
- Supresión o borrado de un registro del archivo.
- Modificación de la información de un determinado registro del archivo.
- Consulta sobre el contenido de un registro del archivo.
- Consulta de todos los registros del archivo (recorrido).

1.1.- Creación de un archivo

Un archivo no existe en el sistema de archivos hasta el momento de su creación; para poder utilizar un archivo este tiene que existir. La operación o proceso de creación consistirá en la escritura de "todos" sus registros sobre el soporte, el archivo puede ser creado vacío.

ALGORITMO CREAR_ARCHIVO (nombre_f)

ENTORNO:

nombre_f: Cadena de caracteres que contiene el nombre del archivo a crear.

pfichero: Variable de tipo puntero a una estructura (de tipo FILE) que contiene las características de utilización del archivo.

buffer: Describe el área de memoria del programa de usuario utilizada como memoria de intercambio con el fichero.

Si es una estructura, su tipo será definido globalmente para que puede ser común a todas las funciones de una determinada aplicación.

El tipo: registro describirá una estructura formada por los campos campo1, campo2...

De entre todos ellos habrá uno que servirá para distinguir un determinado registro lógico en el fichero o archivo del resto, este campo se denominará campo de identificación del registro.

mas_datos: Switch (s/n) que nos servirá para el control de la cantidad de información que será introducida en el archivo.

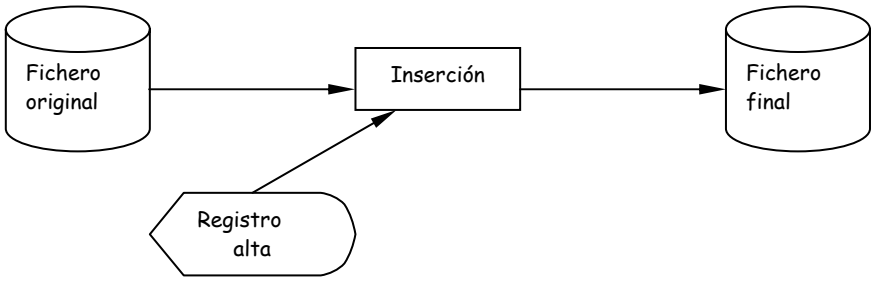
PROCESO:

```
Inicio
        mas_datos   ← 's'
        pf ichero   ←  < Abrirf (nombre_f, lectura) >
        si pfichero es nulo
        (no puede abrirse y por lo tanto no existirá y puede crearse)
                pfichero ←  < Abrirf (nombre_f, escritura) >
                si pfichero es nulo
                        < Visualizar ("Error, no puede abrirse") >
                        < Salir () >
                fin_si
                mientras mas_datos = 's'
                        (llenar buffer)
                        buffer   ←        Llenar_buffer(...)
                        < Escribir_en_fichero (buffer, pfichero) >
                        si (hay error en Escribir_en_fichero)
                                < Visualizar ("Error al escribir") >
                                < Salir () >
                        fin_si
                        < Visualizar ("¿Quieres introducir más datos?") >
                        < Leer (mas_datos) >
                fin_mientras
                < Cerrarf (pfichero) >
        sino
                < Visualizar ("El fichero ya existe") >
        fin_si
Fin
```

1.2.- Inserción de un registro en un archivo ya creado

Consistirá en añadir un registro en un archivo ya creado; esta inserción puede realizarse al final del archivo (añadir) o en una determinada posición sobre el soporte, si los registros están dispuestos siguiendo algún tipo de secuencia (los registros están ordenados).

El organigrama del proceso sería:



ALGORITMO INSERTAR_REGISTRO (nombre_f, alta)

ENTORNO:

nombre_f: Cadena de caracteres que contiene el nombre del fichero.

pfichero: Puntero a una estructura de tipo FILE que contiene las características del fichero.

buffer: Describe el área de intercambio memoria-archivo de programa de usuario.

nombre_f_aux: Cadena de caracteres que contiene el nombre del fichero auxiliar donde se efectuarán los cambios.

pficheroaux: Puntero a una estructura de tipo FILE que contiene las características del fichero auxiliar.

alta: Variable con las mismas características que buffer, que contiene los datos para dar de alta o insertar en el fichero.

PROCESO:

```
Inicio
        pfichero ← < Abrirfichero(nombre_f, lectura) >
        si pfichero es nulo
                < Visualizar ("Error al abrir el fichero") >
                < Salir () >
        fin_si
        pficheroaux ← < Abrirfichero (nombre_f_aux,escritura) >
        si pficheroaux es nulo
                < Visualizar ("Error al abrir el fichero") >
                < Salir () >
        fin_si
        < Leerfichero (buffer, pfichero) >
        < realizar control de errores >
        mientras (no fin de fichero de pf y "memo_alta > buffer")
                < Escribirfichero (buffer, pficheroaux) >
```

< realizar control de errores >

< **Leerfichero** (**buffer**, **pfichero**) >

< realizar control de errores >

fin_mientras

< **Escribirfichero** (**alta, pficheroaux**) >

< realizar control de errores >

mientras no fin de fichero (pfichero)

< **Escribirfichero** (**buffer**, **picherofaux**) >

< realizar control de errores >

< **Leerfichero** (**buffer, pfichero**) >

< realizar control de errores >

fin_mientras

< **Cerrarf** (pfichero) >

< **Cerrar f**(pficheroaux) >

< **remove** (nombre_f) >

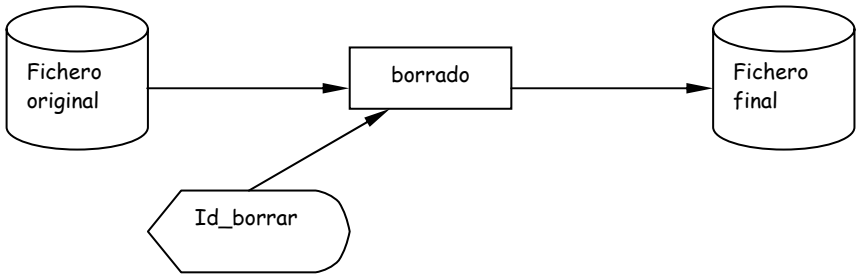
< **rename** (nombre_f_aux, nombre_f) >

Fin

1.3.- Supresión o borrado de un registro de un archivo

La supresión o borrado de un registro del archivo consiste en eliminar un registro de un archivo ya existente.

El organigrama del proceso será:



ALGORITMO BORRAR_REGISTRO (nombre_f, borrar)

ENTORNO:

nombre_f: cadena de caracteres que contiene el nombre del fichero.

pfichero: Puntero a una estructura de tipo FILE que contiene las características del fichero.

buffer: Describe el área de intercambio entre el programa y el archivo.

nombre_f_aux: Cadena de caracteres que contiene el nombre del fichero auxiliar donde se efectuarán los cambios.

pficheroaux: Puntero a una estructura de tipo FILE.

borrar: Variable que contiene el valor del campo que identifica la información que queremos borrar en el archivo.

PROCESO:

Inicio

pf ichero ← < **Abrirfichero** (nombre_f, **lectura**) >

si pfichero es nulo

< Visualizar ("Error al abrir el fichero") >

< Salir () >

fin_si

pficheroaux ← < **Abrirfichero** (nombre_f_aux, **escritura**) >

si pficheroaux es nulo

< Visualizar ("Error al abrir el fichero") >

< Salir () >

fin_si

< **Leerfichero** (**buffer, pfichero**) >

< realizar control de errores >

mientras no fin de fichero(pfichero)

si ("*borrar distinto de buffer*")

(si no es la información que queremos borrar la escribimos en el fichero auxiliar)

< **Escribirfichero**(**buffer, pficheroaux**) >

< realizar control de errores >

fin_si

< **Leerfichero** (**buffer, pfichero**) >

< realizar control de errores >

fin_mientras

< **Cerrarf** (pfichero) >

< **Cerrarf** (pficheroaux) >

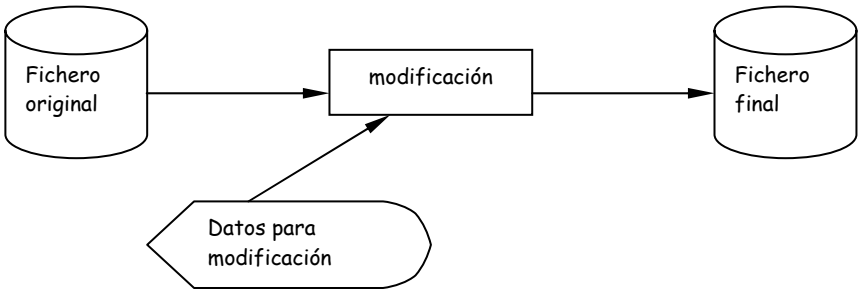
< **remove** (nombre_f) >

< **rename** (nombre_f_aux, nombre_f) >

Fin

1.4.- Modificación de la información de un registro en un fichero

El algoritmo correspondiente permite modificar toda o parte de la información correspondiente a un registro del archivo. El organigrama del proceso es:



ALGORITMO MODIFICACIÓN_REGISTRO (nombre_f, modifi)

ENTORNO:

nombre_f: cadena de caracteres que contiene el nombre del fichero.

pfichero: Puntero a una estructura de tipo FILE que contiene las características del fichero.

buffer: Describe el área de intercambio entre el programa y el archivo.

nombre_f_aux: Cadena de caracteres que contiene el nombre del fichero auxiliar donde se efectuarán los cambios.

pficheroaux: Puntero a una estructura de tipo FILE.

modifi: Representa la información que nos permita localizar los datos que se quiere modificar en el archivo.

PROCESO:

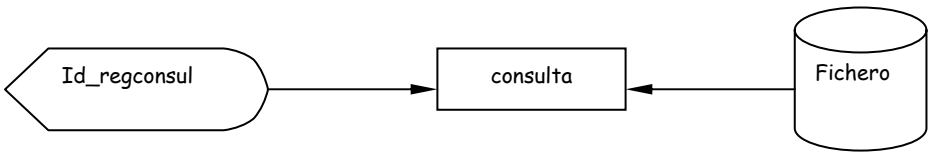
```
Inicio
        pf ichero  ←   < Abrirfichero (nombre_f, lectura) >
        si pf icheroes nulo
                < Visualizar ("Error al abrir el fichero") >
                < Salir () >
        fin_si
        pficheroaux ←   < Abrirfichero (nombre_f_aux, escritura) >
        si pficheroaux es nulo
                < Visualizar ("Error al abrir el fichero") >
                < Salir () >
        fin_si
        < Leerfichero (buffer, pfichero) >
```

```
< realizar control de errores >
mientras no fin de fichero(pfichero)
      si  (buffer no contiene la información que se va a modificar)
                < Escribirfichero (buffer,pficheroaux) >
                < realizar control de errores >
     sino
        (habría que implementar la función)
                 buffer  ←  < modificar_campos (buffer) >

                < Escribirfichero (buffer, pficheroaux) >
                < realizar control de errores >
     fin_si
    < Leerfichero (buffer, pfichero) >
    < realizar control de errores >
fin_mientras
< Cerrarf (pfichero) >
< Cerrarf(pficheroaux) >
< remove (nombre_f) >
< rename (nombre_f_aux, nombre_f) >
Fin
```

1.5.- Consulta de un registro

El diagrama del proceso será:



ALGORITMO CONSULTA_REGISTRO (nombre_f, consul)

ENTORNO:

nombre_f: cadena de caracteres que contiene el nombre del fichero.

pfichero: Puntero a una estructura de tipo FILE que contiene las características del fichero.

buffer: Describe el área de intercambio entre el programa y el archivo.

consul: Variable que contiene información sobre lo que se quiere consultar.

PROCESO:

Inicio

pfichero ← < **Abrirfichero**(nombre_f, **lectura**) >

si pficheroes nulo

< Visualizar ("Error al abrir el fichero") >

< Salir () >

fin_si

< **Leerfichero** (**buffer, pfichero**) >

< realizar control de errores >

mientras no fin de fichero(pfichero) y "*consul distinto buffer*"

(no tengo en buffer la información que quiero consultar)

< **Leerfichero** (**buffer,pfichero**) >

< realizar control de errores >

fin_mientras

si es fin de archivo(pfichero)

< Visualizar ("El registro no existe") >

sino

< Visualizar (buffer) >

fin_si

< **Cerrarf** (pfichero) >

Fin

1.6.- Recorrido de un fichero secuencial

El proceso consistirá en ir pasando por todos los registros del fichero y por ejemplo visualizando el contenido de estos registros por pantalla.

ALGORITMO RECORRIDO_SECUENCIAL (nombre_f)

ENTORNO:

nombre_f: cadena de caracteres que contiene el nombre del fichero.

pfichero: Puntero a una estructura de tipo FILE que contiene las características del fichero.

buffer: Describe el área de intercambio entre el programa y el archivo.

PROCESO:

```
Inicio
        pfichero ← < Abrirfichero (nombre_f, lectura) >
        si pfichero es nulo
                < Visualizar ("Error al abrir el fichero") >
                < Salir () >
        fin_si

        < Leerfichero (buffer, pfichero) >
        < realizar control de errores >
        mientras no fin de fichero(pfichero)
                < Visualizar (buffer) >
                < Leerfichero (buffer, pfichero) >
                < realizar control de errores >
        fin_mientras
        < Cerrarf (pfichero) >
Fin
```

2.- Algoritmos de mantenimiento de ficheros de organización directa

2.1.- Creación

El proceso de creación de un archivo con organización directa consiste en ir introduciendo los sucesivos registros, en el soporte que los va a contener, en la posición adecuada resultante de la aplicación del método HASH correspondiente y teniendo en cuanta que pueden producirse SINONIMOS.

ALGORITMO CREACIÓN (nombre_f) (Crea un archivo vacío)

(Crea un fichero vacío)

ENTORNO:

nombre_f: cadena de caracteres que contiene el nombre del fichero.

pf: Puntero a una estructura de tipo FILE que contiene las características del fichero.

PROCESO:

```
Inicio
        pf ← < AbrirFichero (nombre_f, lectura) >
        si pf es nulo
                < Visualizar ("El fichero no existe y puede ser creado") >
                pf ← < AbrirFichero (nombre_f, lectura/escritura) >
                si pf es nulo
```

```
                    < Visualizar ("Error al crear el fichero") >
                    < Salir () >
                fin_si
            < CerrarFichero (pf) >
            sino
                < Visualizar ("El fichero ya existe") >
            fin_si
        Fin
```

En cada registro de un archivo directo se suele incluir un campo (ocupado) que puede servir para distinguir un registro dado de baja o modificado de uno que existe o de otro que nunca contuvo información.

Dentro del proceso de creación del archivo podría considerarse una inicialización del campo ocupado para todos los registros del archivo directo. (s ó n, 1 ó 0, ...).

Por **ejemplo**:

```
struct tipo_registro registro;
...
Para i=1 mientras i<50 incremento de 1
    registro.ocupado ← 'n'
    < fwrite (&registro, sizeof (registro), 1, pf) >
    si error
        < Visualizar ("Error en escritura") >
    fin_si
fin_para
```

Además añadiremos un campo clave, preferentemente numérico que nos permita aplicar la "función Hash" con facilidad. En particular, podría ser un número de 1 a N.

2.2.- Alta

La operación de dar de alta un registro en un archivo con organización directa consiste en insertar un registro sobre el soporte en una determinada posición que se obtiene aplicando la función Hash elegida, a la clave. (Método de transformación de la clave).

ALGORITMO DE ALTA (nombre_f, reg_alta)

ENTORNO:

nombre_f: cadena de caracteres que contiene el nombre del fichero.

reg_alta: Estructura que contiene los datos del registro a dar de alta.

pf: Puntero a una estructura de tipo FILE que contiene las características del fichero.

posición: Valor correspondiente a la posición donde encontraremos un determinado registro en el fichero.

registro: Describe el lugar de intercambio entre el programa y el archivo.

Hash(): Función Hash para el cálculo de direcciones.

Tratamiento de sinónimos(): Función para el tratamiento de colisiones.

PROCESO:

```
Inicio
        pf ← < AbrirFichero (nombre_f, lectura/escritura) >
        si pf es nulo
                < Visualizar ("El fichero no existe y puede ser creado") >
                < Salir () >
        fin_si
        posición ← Hash (reg_alta.clave)
        < Posicionar (pf, posición, referencia) >
        < Control de errores()>
        < Leer_de_Fichero (registro, pf) >
        < Control de errores() >
        si registro.ocupado es 's'
                < Visualizar ("La posición está ocupada por otro registro) >
                < Tratamiento de sinónimos () >
        sino
                reg_alta.ocupado ← 's'
                < Posicionar (pf, posición, referencia) >
                < Control de errores()>
                < Escribir_en_Fichero(registro_alta, pf) >
                <Control de errores()>
        fin_si
        < CerrarFichero (pf) >
Fin
```

2.3. Baja

La operación de dar de baja un registro de un fichero con organización directa es una operación de baja lógica y consistirá en poner 'n' en el campo ocupado del registro que queremos dar de baja.

ALGORITMO DE BAJA (nombre_f, clave_baja)

ENTORNO:

nombre_f: cadena de caracteres que contiene el nombre del fichero.

pf: Puntero a una estructura de tipo FILE que contiene las características del fichero.

posición: Valor correspondiente a la posición donde encontraremos un determinado registro en el fichero.

registro: Describe el lugar de intercambio entre el programa y el archivo.

clave_baja: Variable que guarda el código del registro a dar de baja.

Hash(): Función Hash para el cálculo de direcciones.

PROCESO:

```
Inicio
        pf  ←   < AbrirFichero (nombre_f, lectura) >
        si pf es nulo
                < Visualizar ("El fichero no existe y puede ser creado") >
                < Salir () >
        fin_si
        posición  ←  < Hash (clave_baja)>
        < Posicionar (pf, posición, referencia) >
        <Control de errores()>
        < Leer_de_Fichero (registro,  pf) >
        < Control de errores()>
         si registro.ocupado  es 's'
                 Habría que comprobar  que no se trata de un sinónimo:
                  si   (registro.clave == clave_baja)
                     registro.ocupado  ←   'n'
                < Posicionar (pf, posición, referencia) >
                < Control de errores()>
                < Escribir_en_Fichero (registro, pf) >
                < Control de errores()>
        sino
```

< Visualizar ("Error el registro no existe") >

fin_si

< **CerrarFichero** (pf) >

Fin

2.4.-Consulta

ALGORITMO DE CONSULTA (nombre_f, clave_consulta)

ENTORNO:

nombre_f: cadena de caracteres que contiene el nombre del fichero.

pf: Puntero a una estructura de tipo FILE que contiene las características del fichero.

posición: Valor correspondiente a la posición donde encontraremos un determinado registro en el fichero.

registro: Describe el lugar de intercambio entre el programa y el archivo.

clave_consulta: Variable que guarda el código del registro que se va a consultar.

PROCESO:

Inicio

pf ← < **AbrirFichero** (nombre_f, lectura) >

si pf es nulo

< Visualizar ("El fichero no existe") >

< Salir () >

fin_si

posición ← < **Hash** (clave_consulta)>

< **Posicionar** (pf, posición, referencia) >

<Control de errores()>

< **Leer_de_Fichero** (registro, pf) >

< Control de errores()>

si registro.ocupado es 's'

(Habría que comprobar que no se trata de un sinónimo, comprobando

si clave_consulta == registro . clave)

< Visualizar (todos los campos) >

sino

< Visualizar ("El registro no existe") >

fin_si

< **CerrarFichero** (pf) >

Fin

2.5.- Visualización

ALGORITMO DE VISUALIZACIÓN (nombre_f)

ENTORNO:

nombre_f: cadena de caracteres que contiene el nombre del fichero.

pf: Puntero a una estructura de tipo FILE que contiene las características del fichero.

registro: Describe el lugar de intercambio entre el programa y el archivo.

Contador:

PROCESO:

Inicio

pf ← < **AbrirFichero** (nombre_f, lectura) >

si pf es nulo

< Visualizar ("El fichero no existe") >

< Salir () >

fin_si

< **Posicionar** (pf, 0, ppio del archivo) >

<Control de errores()>

< **Leer_de_Fichero** (registro, pf) >

< Control de errores()>

mientras no sea fin fichero de pf

si registro.ocupado es 's'

(Habría que comprobar que no se trata de un sinónimo, comprobando si clave_consulta == registro . clave)

< Visualizar (todos los campos) >

fin_si

< **Leer_de_Fichero** (registro, pf) >

< Control de errores()>

fin_mientras

fin_si

< **CerrarFichero** (pf) >

<Control de errores()>

fin

Fin

2.5.- Borrado

ALGORITMO DE BORRADO (nombre_f)

ENTORNO:

nombre_f : Cadena de caracteres que contiene el nombre del archivo que queremos borrar.

PROCESO:

Inicio

<remove(nombre_f)>

fin

fin

2.6.- Función Hash

ALGORITMO FUNCIÓN Hash(valor_clave, posición)

ENTORNO:

Valor_clave: Recibe el valor de la clave para la que quiere calcularse el desplazamiento en el archivo.

posición: dirección obtenida al aplicar un valor clave a la función hash().

PROCESO:

Inicio

posicion ← (valor_clave – huecos_iniciales)*tamaño de registro

devolver posicion

fin

fin