GUIA DE ESTUDIO: ARCHIVOS DIRECTOS

Introducción:

Una característica común de los tipos de datos que hemos visto hasta ahora es que la información que cargamos se mantiene vigente solamente mientras está corriendo el programa donde fueron declaradas las variables.
No importa cuanto esfuerzo nos lleve el ingresar una enorme cantidad de datos, esa información desaparece al finalizar el programa.

Sin embargo, en nuestro trato diario con la computadora, información de todo tipo (textos, figuras, sonidos, planillas de cálculo, etc.), es conservada, no sólo con independencia de que se esté corriendo un particular programa, sino, inclusive de que esté o no encendido el equipo.
.
Esta información que se encuentra en dispositivos de memoria mediata (también llamada memoria auxiliar) de nuestros ordenadores (disquetes, discos rígidos, etc.) que la mantienen en forma permanente es lo que llamamos archivos.

El lenguaje PASCAL, como todos los lenguajes de computación, tiene mecanismos para poder acceder a ese tipo de información, permitiendo declarar un tipo de variable estructurada que le permite vincularse con algunos de esos archivos.

Esto es lo que da una verdadera funcionalidad a los programas ya que se puede conservar la información obtenida más allá del tiempo de corrida del programa y, muy importante, intercambiar esa información con otros programas.

Para ello se define un nuevo tipo de datos estructurado llamado **FILE** (ARCHIVO). A continuación veremos algunas de sus características.

Definición de Archivo

Es un tipo de dato, que se puede guardar en memoria auxiliar para su uso posterior, mediante la ejecución de los programas adecuados, permitiendo almacenar grandes volúmenes de información.

Se trata de una estructura homogénea de datos consistente en una secuencia de elementos o componentes, todos del mismo tipo, ya sea simple o estructurado.

Por lo tanto para declarar una variable de tipo archivo es necesario definir previamente la naturaleza de sus componentes.

Observaciones:

1) Como la información contenida en el tipo de dato archivo se almacena en un dispositivo auxiliar (los datos obtenidos antes, durante y después del procesamiento no se pierden), necesita de la existencia de un “archivo físico” en el sistema operativo (que deberá ser creado según los modos y reglas de dicho sistema) y que será asociado con el archivo declarado en el programa.

2) A diferencia de los tipos de datos vistos hasta ahora no se establece en la declaración cual será el tamaño que se reserva.
En contraste, por ejemplo con los tipos estructurados **array** y **record** (arreglos y registros), que no reservan un espacio fijo en memoria.
Su tamaño solo estará limitado por el tamaño de la memoria auxiliar donde se guarda.

Tipos de Archivo:

Una primera consecuencia importante de esta interacción entre el Sistema Operativo y el lenguaje hace que tengamos que considerar dos tipos de archivos que están relacionados originariamente a los dispositivos físicos en los que se mantiene la información: a**rchivos secuenciales y archivos de acceso directo.**

Archivos Secuenciales:

Los archivos secuenciales, son los mas antiguos y su origen está vinculados con el soporte en cinta. En una cinta para leer cualquier parte de la misma se debe pasar por todos los datos anteriores (ej. Cuando queremos posicionarnos en una cinta de video).
En el curso no vamos a utilizar este tipo de archivo.[1]

Archivo de acceso directo:

Se dice que un archivo es de acceso u organización directa cuando para acceder a un registro n cualquiera no se tiene que pasar por los n-1 registros anteriores.
Por su definición, podemos rápidamente darnos cuenta que si los comparamos con los archivos secuenciales son mucho más rápidos al momento de recuperar datos.

Consideraciones:

- un archivo de acceso directo tiene que tener sus registros o renglones de un tamaño fijo o predeterminado de antemano,
- Un archivo de acceso directo permite posicionar el puntero de registros, en cualquier registro determinado sin necesidad de pasar por todos los registros anteriores.

De ahora en más al hablar de archivo, estaremos haciendo referencia a archivos de acceso directo.

[1] Sin embargo debemos mencionar dos muy importantes que se están utilizando desde el principio del curso :

a) El PASCAL considera al dispositivo estándar de ingreso de datos(en nuestro caso el teclado) como un archivo secuencial de sólo ingreso (sólo lectura)
b) De la misma manera, se toma el dispositivo estándar de salida (la pantalla) como un archivo secuencial que sólo permite salida (sólo escritura).

En estos casos no se considera necesario que los datos sean todos del mismo tipo

Operaciones básicas con archivos.

Declaración de un archivo.

La declaración de un archivo directo consta de dos pasos:

1) Declaración del tipo adecuado: se realiza con las palabras reservadas **FILE OF**, su sintaxis es la siguiente:

Nombredetipo = FILE OF TipoElementos;

Esta declaración se realiza en la sección correspondiente a la declaración de tipos.
Ejemplo:

```
TYPE
   Enteros= FILE OF Integer;          archivo de enteros

   Letra= ARRAY [1..100] of char;
   Caracters= FILE OF letra;          archivo de array
```

2) Declaración de una variable de archivo de un tipo de archivo declarado.

Ejemplo:

```
VAR
   Numeros: Enteros;

   Nombres: Caracters;
```

Normalmente no se desea crear archivos que puedan almacenar un solo tipo de datos ya que se requerirían varios archivos para poder registrar un conjunto de información. Por ejemplo para poder guardar los datos de un conjunto de personas sería necesario crear un archivo para los nombres, otro para apellidos, otro para la edad, etc. Para evitar este inconveniente conviene usar el tipo registro (**record**), que permiten grabar en un solo registro un grupo de datos que pueden ser de diferentes tipos.

Lógicamente, estos registros del tipo record deben ser declarados previamente a la declaración del tipo archivo.

Por ejemplo si se quiere crear un archivo para guardar el nombre, domicilio, edad y estado civil de un grupo de personas el primer paso a realizar es crear un registro que contenga todos estos campos:

```
TYPE
   Datos = RECORD
      Nombre : String[40];
      Domicili : String[60];
      Edad : 0..150;
      EstCivil : ('C','S','D','V');
   END;
```

Y a continuación declarar un archivo del tipo Datos y una variable del mismo tipo de los que se utilizarán en el archivo:

```
TYPE
      Archiv = FILE OF Datos;
VAR
      Personas : Archiv;
```

Asignación de un archivo.

Declarado la variable archivo, se debe como primera actividad, vincular su identificador en el programa con un archivo físico.
Esta operación asigna a un identificador de archivo un archivo real que es el que perdura en la memoria auxiliar.
El proceso de asignación establece una correspondencia entre la variable tipo archivo con un archivo externo. Dicha asignación debe realizarse de la siguiente forma:

```
ASSIGN (Archivo, 'Nombre.ext');
```

Donde Archivo es el nombre interno del archivo por el que se conoce el archivo dentro del programa (el declarado).

Nombre.txt es el nombre externo con el que es reconocido el archivo por el sistema operativo, es una cadena que puede ser una constante, una variable o estar escrita directamente en el programa. Naturalmente debe cumplir con todas las reglas para nombrar un archivo. Debe cumplir con las reglas de los identificadores que se utilizan en el sistema operativo.

El nombre externo debe estar creado (debe existir) en la memoria auxiliar, ANTES de hacer la asociación utilizando el ASSIGN. En el Anexo 1 se indican las diferentes formas de crearlo.

Continuando con el Ejemplo anterior:

```
BEGIN
      ASSIGN (Personas, 'Personas.dat' );
```

En este ejemplo, se puede observar que el nombre interno (el que se usa dentro del programa) del archivo es **Personas**, y el nombre externo **(el que se va a encontrar al explorar el sistema operativo**) es **Personas.dat.**

También es posible, indicar, al asociar el nombre interno con el nombre externo, una ubicación física determinada del archivo externo, por ejemplo:

```
ASSIGN(Personas, 'C:\Clientes\Personasl.dat')
```

Indica que los datos del archivo Personas se almacenarán en el archivo de memoria auxiliar Personas.dat, dentro del subdirectorio Clientes de la unidad C (sea cual fuera el dispositivo indicado con la letra C , generalmente un disco rígido).

Abrir archivos:

Una vez declarado y asignado un archivo ya es posible abrirlo.

En caso de querer abrir un archivo existente se utiliza el procedimiento **Reset**.

En caso de querer abrir un archivo nuevo (se crea y se abre) se utiliza el procedimiento **Rewrite.**

Ambos procedimientos están predefinidos. Si al utilizar el procedimiento **Rewrite** el archivo asignado ya existía se borrará el anterior y se creará uno nuevo, por lo mismo se debe tener cuidado al momento de abrir estos archivos.

La sintaxis es:

RESET (variabletipoarchivo);

REWRITE (variabletipoarchivo);

Lectura y escritura de archivos.

Una vez que se ha abierto el archivo, para la lectura y escritura en un archivo de acceso directo se utilizan los procedimientos **Read, Write y** la función **Eof**.

La sintaxis es:

READ (variabletipoarchivo, lista de elementos);

WRITE (variabletipoarchivo, lista de elementos);

EOF (variabletipoarchivo);

La función **EOF**( ) es una función de tipo lógico que indica si el fin del archivo se ha alcanzado, devolviendo TRUE en caso afirmativo y FALSE en caso contrario. Generalmente esta función se utiliza cuando se avanza en un archivo registro por registro, para determinar si ya se llegó al final del archivo.

Cerrar un archivo.

Para cerrar un archivo abierto se utiliza el procedimiento **Close**:
Sintaxis:

CLOSE (variabletipoarchivo);

Mantenimiento de un archivo.

Con el paso del tiempo es necesario hacer operaciones que permitan mantener actualizados los datos que se guardan en un archivo directo, por lo tanto, es necesario modificar o actualizar datos después que el archivo ha sido creado (modificar uno o más campos, borrar un registro, insertar un nuevo registro, o visualizar los valores de los campos de un registro). Para poder ejecutar las acciones detalladas es necesario utilizar una serie de procedimientos y funciones que se detallan a continuación.

Tamaño de un archivo.

La función **FileSize** regresa el tamaño actual de un archivo, es decir devuelve el número de registros contenidos en éste. Su sintaxis es la siguiente:

FILESIZE (variabletipoarchivo);

Al momento de abrir un archivo nuevo la función **FileSize** regresa el valor de 0, lo que significa que el archivo no tiene datos guardados en él.

Posicionamiento en el interior de un archivo.

La función **FilePos** devuelve la posición actual del archivo, el número de registro actual, su sintaxis es:

FILEPOS(variabletipoarchivo);

Para moverse a un registro determinado se utiliza la función **Seek**, que permite situarse en el número de registro que se desea. Tiene la siguiente sintaxis:

SEEK (variabletipoarchivo**,** númeroderegistro);

Sobre la base que cada uno de los registros de un archivo esta referenciado por un número específico comenzando desde el registro 0 y aumentando de 1 en 1.

Para moverse al final del archivo para agregar un nuevo registro se utiliza este mismo comando con el parámetro *númeroderegistro* como sigue:

SEEK (VariabletipoArchivo, **FILESIZE**(VariabletipoArchivo));

A continuación se presentan ejemplos que utilizan algunos de estos procedimientos y funciones.

Ejemplo 1:

Se define la estructura de un archivo que va a guardar los datos de un grupo de alumnos y se permite la carga de dichos datos.

```
Program eje1(input, output);
type
        alumnos = RECORD
        legajo:integer;
        nombre:string[30];
        edad:16..150;
        end;
var
alumno:alumnos;
archivo: file of alumnos;
ba:char;

BEGIN
        assign(archivo, 'd:\pascal\alumnos.dat');
        rewrite(archivo);
        ba:='s';

        While ba='s' do
        BEGIN
                write('Ingrese el legajo del alumno : ');readln(alumno.legajo);
                write('Ingrese el nombre del alumno : ');readln(alumno.nombre);
                writeln('Ingrese la edad del alumno   : ');readln(alumno.edad);

                write(archivo, alumno);
                Write ('¨Ingresa datos de otro alumno?? ( s=SI   n=NO)');
                repeat
                        read (ba)
                until (ba = 's') or (ba = 'n');
        END;

        close(archivo);
        writeln('NUEVOS REGISTROS INSERTADOS´);
        readln;
END.
```

Ejemplo 2:

Ahora se pretende presentar la forma en que se realiza una búsqueda de un alumno, en el archivo cargado previamente (en el ejemplo 1), utilizando como dato de búsqueda la posición del registro donde fueron guardados sus datos.

```
Program eje2(input, output);
Type
        alumnos = RECORD
        legajo:integer;
        nombre:string[30];
        edad:16..150;
        end;
var
        archivo: file of alumnos;
        alumno:alumnos;
        pos:integer;
        ba: char;

BEGIN
        assign(archivo,'d:\pascal\alumnos.dat');
        reset(archivo);

         While ba='s' do
           BEGIN
              write('nro de registro a buscar');
              readln(pos);

              seek(archivo,pos-1);

              read(archivo,alumno);
              write('Legajo : ');writeln(alumno.legajo);
               write('Nombre : ');writeln(alumno.nombre);
              write('Edad : ');writeln(alumno.edad);
                  write ('¨Desea  datos de otro alumno?? ( s=SI   n=NO)');

                  repeat
                           read (ba)
                  until (ba = 's') or (ba = 'n');
        END;

        close(archivo);
END.
```

Ejemplo 3:
Se creará un archivo que contenga números enteros y luego los muestre, con la particularidad que, en la muestra solo aparecerán los números pares.

```
Program eje3(input,output);

Type
        Arc= File of integer;
var
Archivo: Arc;
Numero : Integer;
Fin:char;

Procedure Alta_num;
        Var
                Rta : Char;

        Begin
        Assign(Archivo,'d:\pascal\Numeros.dat');
        Rewrite(Archivo);
        Repeat
                write('Introduzca un número entero : ');readln(Numero);
                write(Archivo, Numero);

                writeln('¨ Desea guardar otro n£mero (S/N) : ');
                repeat
                        read (rta)
                until (rta = 'S') or (rta ='N');
        until rta='N';
        End;

Procedure Mues_num
        Begin
        Writeln('Números enteros y pares del archivo');
        Seek(Archivo,0);
        while not EOF(Archivo) do
                begin
                Read(Archivo, Numero);
                if Numero Mod 2 = 0 then writeln(Numero:6);
                end;
        End;
BEGIN
Alta_num;
Mues_num;
write ('FIN DEL PROGRAMA');
READLN (fin);
Close(Archivo)
END.
```

ANEXO 1

Operaciones para crear un archivo externo.

Existen al menos dos formas de crear un archivo externo para luego utilizar el Assign y asociar una variable declarada tipo archivo con dicho archivo externo (página 4). Se describen a continuación

1) **Desde el Pascal, como parte del programa. Este fragmento de Programa puede ser agregado al Ejemplo 1, para crear el archivo antes de insertar datos.**

Se utilizará la directiva del compilador **{$I}** .Esta directiva cuando esta activa (por defecto) verifica los errores de entrada y salida (querer abrir un archivo que no exista es un error de entrada y salida).

Entonces lo que hacemos es, desactivar el chequeo de error **{$I-}** e intentar abrir el archivo, si hay error es que no se pudo abrir porque no se encontró, entonces se crea.
Para saber si hubo un error se comprueba el valor de la function **IORESULT** que devuelve un entero que representa el estado de la ultima operación de entrada y salida, si dicho valor es diferente de 0 significa que hubo un error. El numero de error cuando no se encuentra un archivo es el 2.

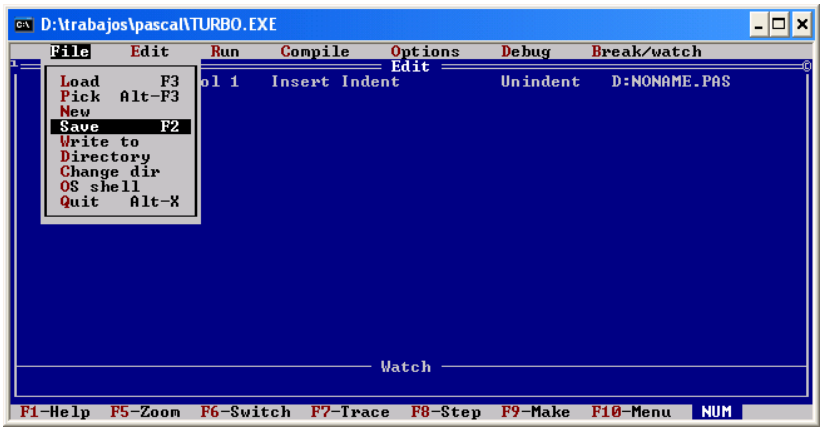
Por último se debe volver a activar la directiva **{$I+}**

A continuación se describe las líneas del Programa Pascal:

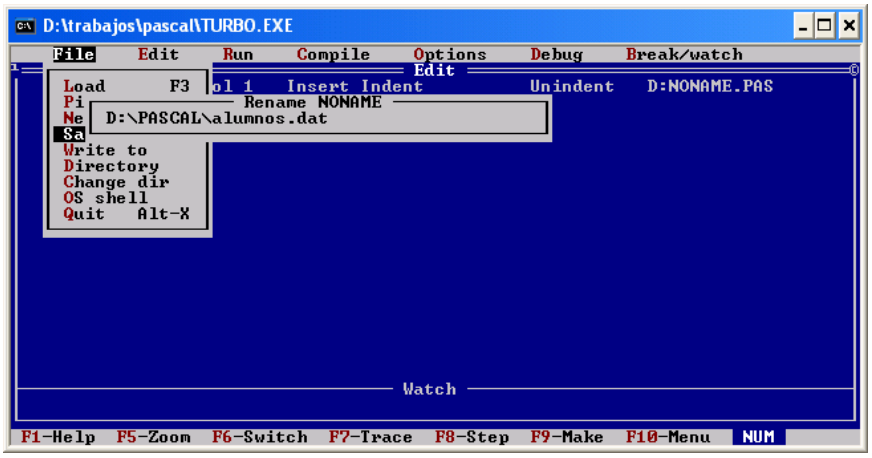
```
Begin

 assign(archivo,'alumnos.dat');

    {$I-}

    reset(archivo);

    If ioresult =2 then  rewrite(archivo);

    {$I+}

  end;
```

Esto debe ser expresado como un procedimiento e invocado al inicio del programa.

2) **Una forma fácil de crear un archivo externo es generar un archivo vacío. Para ello, ejecute el Pascal y desde la barra de Menú seleccione la opción SAVE:**



A continuación escriba el nombre que desee para el archivo externo, archivo que va a contener los datos ingresados:



Ahora ejecute el programa Pascal deseado (en este texto el Ejemplo 1 o Ejemplo 2).

Otras operaciones de gestión de archivos.

Pascal tiene operaciones que permiten manipular archivos y directorios en disco como el sistema operativo MS-DOS(aunque el ambiente es WINDOWS).
Se pueden realizar las siguientes operaciones con archivos y directorios:

- **Erase**(nomvararchivo) — Borra un archivo
- **Rename** (nomvararchivo,'nombrenuevoarchivo') — Renombra
- **Chdir** (directorio) — Cambia Directorio
- **Mkdir** (directorio) — Crea un Directorio
- **Rmdir** (directorio) — Borra un Directorio

Agradezco a Claudia Dania, Danilo Monti y Raúl Kantor sus valiosas observaciones durante la confección de este apunte.