Archivos o ficheros

Los archivos son tipos de datos estructurados, dinámicos, con la particularidad de que se almacena en memoria secundaria (los datos no se pierden al salir del programa)

Tipos de archivos:

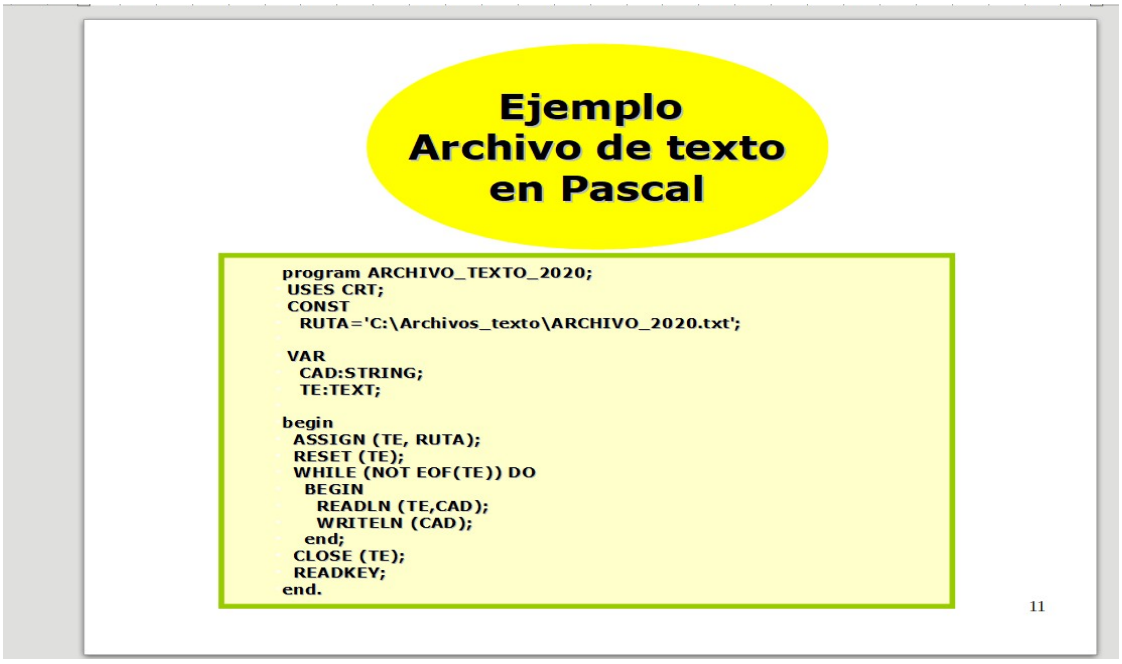
archivos de texto
archivos binarios

Archivos de texto:

Los archivos de texto se estructuran línea a línea.
Su acceso es secuencial. Es decir que para acceder al elemento M se debe haber pasado por los M-1 elementos anteriores. No se puede implementar el acceso directo, pues las líneas no siempre tienen la misma longitud.
La variable archivo mantiene un apuntador interno al elemento actual. Cuando se realiza un READ el apuntador pasa al siguiente elemento.

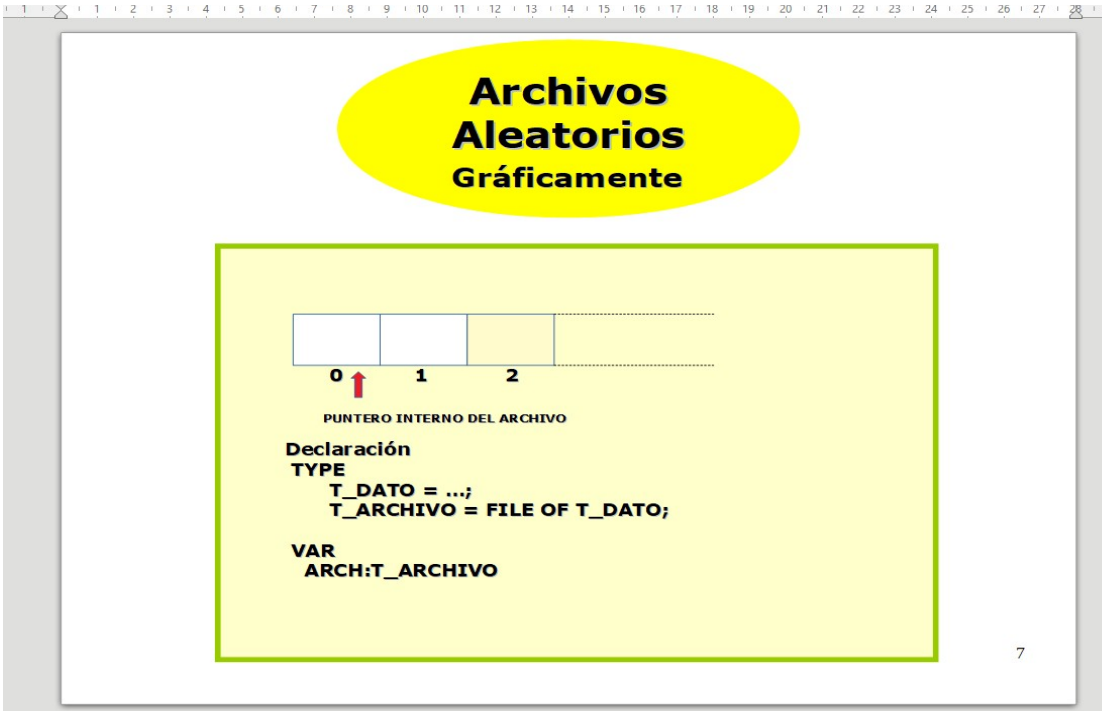
Type
archivo_de_texto= text;

Ejemplo de archivo de texto en Pascal:
//lee un archivo_2020.txt existente previamente en el directorio C:\Archivos_texto\



Archivos binarios

Los archivos binarios se estructuran de acuerdo a un tipo definido previamente. Los elementos son de igual tipo (o distintos pero se conoce con anterioridad su tamaño pues se debe declarar.)
Su acceso es directo (random), con la posición del elemento, pero también se puede acceder en forma secuencial.
La variable archivo, al igual que en los archivos de texto, mantiene un apuntador interno al elemento actual. Cuando se realiza un READ el apuntador pasa al siguiente elemento.



Ejemplos de archivos binarios en Pascal:

Archivos tipo file of

Ejemplo:

```
Type
        t_dato=record
                    Nom:string[50];
                    Tel: string[15];
                end;
         t_archivo = file of t_dato;
```

Archivos tipo file: no tienen definición de tipos

Ej: archivo = file;

Tipos de Organización:

- **Secuencial**

Es una sucesión de elementos, uno a continuación de otro, y para acceder al registro N se debe acceder previamente a los N-1 anteriores

- **Directa**

Cuando el orden físico no se corresponde con el orden lógico. Se accede a los elementos en forma directa mediante su posición

- **Secuencial indexada**

Consta de un archivo de índices y un archivo de datos

CLAVE	DIRECCION	CLAVE	DATOS
15	00	00	01
30	15		
			15
		15	
		29	30

Ejemplos:

. a partir de la dirección 00 se encuentra la clase 15
. a partir de la dirección 15 se encuentra la clave 30

Operaciones sobre archivos:

Asignar // assign (Varchivo, ruta)
Crear // rewrite (Varchivo)
Abrir // resert (Varchivo)
Cerrar // close (Varchivo)
Borrar // erase (Varchivo) el archivo debe estar cerrado previamente
Renombrar // rename (Varchivo_aux, ruta)
Actualizar
Clasificar u ordenar

Funciones y procedimientos para manejo de archivos en Pascal

Filesize (Varchivo) // devuelve la cantidad de registros que tiene almacenado el archivo
Filepos (Varchivo) // devuelve la posición en que se encuentra el puntero interno del archivo
IOResult // devuelve 0 si el archivo existe
EOF (Varchivo) // devuelve true si es fin de archivo
SEEK (Varchivo,pos) // posiciona el apuntador interno del archivo en la posición *pos*

Operaciones sobre registros (ABMC)

Alta:
Consiste en la adición de un nuevo registro.
Dependiendo de la lógica utilizada se podrá almacenar al principio en el medio o al final del archivo.
Para dar de alta se debe verificar previamente que el registro no exista.
Consultar si aprueba los datos a ser dados de alta y guardar los datos

Baja
Consiste en eliminar física (borrado y desaparición de todo el contenido de un registro) o lógicamente un registro (añadir un campo Activo de tipo booleano donde se establezca si está activo o no ese registro, no se borra físicamente)
Consultar si aprueba los datos a ser dados de baja y guardar los cambios si es baja lógica

Modificación:
Consiste en cambiar total o parcialmente el contenido del registro.
No se puede modificar el campo CLAVE
Para modificar se deben visualizar los datos del registro y dar la posibilidad de cambio
Luego, consultar si aprueba los datos a ser modificados y guardar las modificaciones

Consulta:
Tiene por fin, mostrar el contenido de un elemento del archivo

Ejemplo de un archivo random para una agenda

```
UNIT ARCHIVOS;
INTERFACE
CONST
RUTA='C:\ALGORITMOS\PERSONAS.DAT'
TYPE
T_DATO = RECORD
        NOM: STRING[60];
        TEL:STRING[15];
        ESTADO:BOOLEAN; //baja lógica
    END;
T_ARCHIVO = FILE OF T_DATO;
PROCEDURE CREAR(VAR ARCH:T_ARCHIVO);
PROCEDURE ABRIR(VAR ARCH:T_ARCHIVO);
PROCEDURE CERRAR(VAR ARCH:T_ARCHIVO);
```

```
IMPLEMENTATION
PROCEDURE CREAR(VAR ARCH:T_ARCHIVO);
BEGIN
  ASSIGN(ARCH,RUTA);
  REWRITE(ARCH);
END;

PROCEDURE ABRIR(VAR ARCH:T_ARCHIVO);
BEGIN
  ASSIGN(ARCH,'C:\ARCHIVO.DAT');
  RESET(ARCH);
END;
{Assign(ARCH, Ruta); //donde Ruta es una constante
o bien: Assign(ARCH, C:\'Archivo.dat')
//Para chequear si el archivo está creado y no de error al utilizar RESET, se puede utilizar las directivas al compilador:
{$I-} //orden al compilador que deshabilite el control de IO
Reset(ARCH);
{$I+} //orden al compilador que habilite el control de IO
if IOResult  <> 0 then Rewrite(ARCH); {si no existe lo crea}}


PROCEDURE CERRAR(VAR ARCH:T_ARCHIVO);
  BEGIN
    CLOSE(ARCH);
  END;
BEGIN
END.

UNIT ARCH_REG;
INTERFACE
USES ARCHIVOS, CRT;
PROCEDURE LEE_REGISTRO(VAR ARCH:T_ARCHIVO; POS:CARDINAL; VAR REG:t_dato);
PROCEDURE GUARDA_REGISTRO(VAR ARCH:T_ARCHIVO; var POS:CARDINAL; REG:t_dato);
PROCEDURE  MUESTRA_registro(r: t_dato);
PROCEDURE  CARGA_registro(VAR r: t_dato);
```

```
PROCEDURE LISTADO(VAR ARCH:T_ARCHIVO);
PROCEDURE LISTADO2 (VAR ARCH:T_ARCHIVO);
```

IMPLEMENTATION

```
PROCEDURE LEE_REGISTRO(VAR ARCH:T_ARCHIVO; POS:CARDINAL; VAR REG:t_dato);
BEGIN
  SEEK(ARCH, POS);
  READ(ARCH, REG);
END;

PROCEDURE GUARDA_REGISTRO(VAR ARCH:T_ARCHIVO; var POS:CARDINAL; REG:t_dato);
BEGIN
  SEEK(ARCH, POS);
  WRITE(ARCH, REG);
END;

PROCEDURE CARGA_registro(VAR r: t_dato);
begin
  with r do
  begin
        WRITE('NOMBRE:  ');
        READLN (NOM);
       WRITE('NRO_TELEFONO:');
       READLN (TEL);
    ESTADO:= TRUE;
  end;
PROCEDURE  MUESTRA_registro(r: t_dato);
begin
  with (r) do
  begin
       WRITELN('NOMBRE:  ',NOM);
       WRITE('NRO_TEL: ', TEL);
  end;
end;
```

```
PROCEDURE LISTADO(VAR ARCH:T_ARCHIVO);
VAR
 REG:t_dato;
BEGIN
  RESET(ARCH);
  WHILE NOT(EOF(ARCH)) DO
    BEGIN
     READ(ARCH, REG);
    if reg.ESTADO = TRUE then
     MUESTRA_REGISTRO(REG);
    END;
END;

PROCEDURE LISTADO(VAR ARCH:T_ARCHIVO);
VAR
 REG:t_dato;
BEGIN
   RESET(ARCH);
  WHILE NOT(EOF(ARCH)) DO
    BEGIN
     READ(ARCH, REG);
     if reg.ESTADO = TRUE then
     MUESTRA_REGISTRO(REG);
    END;
END;
```

```
PROCEDURE LISTADO2 (VAR ARCH:T_ARCHIVO);
VAR
 REG:t_dato;
I:CARDINAL;
BEGIN
     FOR I:= 0 TO FILESIZE(ARCH)-1 DO
     BEGIN
      LEE_REGISTRO (ARCH,I,REG); //SEEK(ARCH,POS); READ(ARCH, REG);
      if reg.ESTADO = TRUE then
      MUESTRA_REGISTRO(REG);
   END;
 END;
BEGIN
END.
```