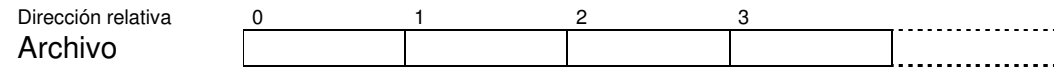
Clase 10: Archivo tipeado

El archivo de **texto**, consiste en una colección de caracteres (puede estar dividido en líneas), puede ser grabado, inspeccionado o modificado mediante el uso de un editor; también un programa Pascal puede escribir o leer del archivo. Se abre para operaciones de lectura o escritura.

El archivo **tipeado,** colección de datos de un mismo tipo (simple o estructurado): enteros, cadena, real, registro, arreglo, etc., sólo puede ser grabado y/o leído desde un programa Pascal, esto se debe a que la información que contiene no es de tipo caracter, sino la imagen de memoria del dato que se almacena (asociada a su tipo, de allí su nombre).
Se los llama tipeados o **binarios** y consisten en una secuencia de items denominados **componentes**. El lugar que ocupan respecto al origen del archivo se denomina *dirección relativa*. La primera dirección es la cero porque está en el origen.



Aunque esta descripción los asemeja al tipo arreglo unidimensional (**vector**), destacamos sus **diferencias**:

- ✓ El almacenamiento del archivo es en memoria secundaria, con lo cual permanece en memoria después de finalizada la ejecución del programa (Persistencia).
- ✓ El archivo no necesita establecer la cantidad máxima de componentes, dicha cantidad está sólo limitada por el espacio físico del dispositivo.
- ✓ Las componentes del archivo no se referencian mediante un índice.

Los archivos tipeados pueden ser procesados con **acceso**:
Secuencial, tomando componente a componente en el mismo orden en el que están almacenadas (y fueron grabadas)
Directo o aleatorio, posicionándose en la dirección relativa de una determinada componente (en cualquier lugar del archivo) y accediéndola para leer y/o grabar.

Se utilizan las mismas funciones y procedimientos con los que operan los archivos de texto, salvo los cambios que se detallan a continuación.
Recordar que a partir de la apertura del archivo un puntero o "flecha" está posicionado en una componente del mismo (componente actual).

a. **DECLARAR** el tipo archivo y la variable de tipo archivo

```
Type
  TipoArchivo = file of TipoComponente;  {de cualquier tipo, menos archivo}

Var
  ident: TipoArchivo;  {es obligatorio describir el tipo, ya que
                        no existe un tipo estándar}
```

b. **ABRIR** archivo

```
RESET(ident)   {el archivo debe existir, se posiciona al comienzo
                para leer o escribir sobre el archivo}
```

c. **LEER**

```
READ(ident, variable) {lee la componente actual sobre la variable,
                       avanza a la próxima componente del archivo}
```

d. **ESCRIBIR**

WRITE(ident, variable) *{graba la variable en la posición actual del archivo y avanza a la siguiente}*

e. **Cantidad de componentes** del archivo (función)

FILESIZE(ident) *{devuelve la cantidad de componentes del archivo}*

Si la cantidad de componentes es n las direcciones relativas están en el rango 0..n-1

f. **POSICIONARSE** sobre una determinada componente (procedimiento)

SEEK(ident, direccion) *{Se posiciona en la componente del archivo cuya dirección especifica el parámetro, 0 <= dirección < Filesize(ident) }*

Notar que:
- ✓ El tipo del archivo debe declararse en la sección *Type* ¿cuál es el motivo?
- ✓ No se utiliza *readln* ni *writeln* ¿por qué?

Parámetros de tipo archivo
Los archivos de texto y los tipeados son siempre parámetros variable, independientemente si el subprograma lee o escribe sobre los mismos. La variable de tipo archivo contiene la dirección de la próxima componente a ser leída o grabada y se modifica en cada operación de E/S.

Ejemplos:

```
Type

TR = record
   C1: real;
   C2: string[5];
End;

Tarch1 = file of word; {las componentes son enteros}
Tarch2 = file of TR;   {las componentes son registros}

Var

  A1: Tarch1;   A2: Tarch2;
```

Grabar y listar un archivo de componentes enteras positivas

```
Procedure Graba1(Var A1: arch1);
Var
  M: word;
Begin
  Rewrite (A1);  Readln(M);
  While M <> 0  do
  Begin
     Write(A1, M);  Readln(M)
  End;
  Close(A1)
End;
```

```
Procedure Lista1(Var A1: Tarch1);
Var
  M: word;
Begin
  Reset (A1);
  While Not Eof(A1)  do
  Begin
     Read (A1, M);  Write(M:6)
  End;
  Close(A1)
End;
```

```
Begin
   Assign(A1, 'Datos1.dat');
   Graba1(A1);
   Lista1(A1);
End.
```

Grabar y listar un archivo de componentes registro

```
Procedure Graba2(Var A2: Tarch2);
Var
   R: TR;
Begin
  Rewrite (A2);  Readln(R.C1);
  While R.C1 <> 0  do
  Begin
     Readln(R.C2);  Write(A2, R);
     Readln(R.C1)
  End;
  Close(A2)
End;
```

```
Procedure Lista2(Var A2: Tarch2);
Var
   R: TR;
Begin
  Reset (A2);
  While Not Eof(A2)  do
  Begin
     Read(A2, R);
     Write(R.C1 :6:2, R.C2)
  End;
  Close(A2)
End;
```

```
Begin
   Assign(A2,'Datos2.Dat');
   Graba2(A2);
   Lista2(A2);
End.
```

Importante: la mínima unidad de entrada salida al archivo es una componente completa, en este caso un registro. No es posible leer o escribir por campos.

Archivos con componente de tipo registro, algunas definiciones previas

Campo de secuencia: campo respecto al cual está ordenado el archivo.
Clave primaria: (*key* o llave) atributo que identifica unívocamente al registro (ej: patente, en el caso de un registro con datos de vehículos). Puede haber más de un campo clave.
Clave secundaria: atributo que identifica a un conjunto de registros, se trata de un valor (o un conjunto de valores) que se repite (ej: patente, en un archivo con datos de infracciones de vehículos)

Como se ve en los ejemplos, un atributo (patente) puede ser clave primaria en un archivo y secundaria en otro.

Ejemplo 1. **Proceso secuencial de un archivo y acceso aleatorio a otro.**

Se tiene un archivo *Tempe* con las marcas térmicas de distintas zonas, con el siguiente diseño de registro:

- Código de Zona (1..50)
- Temperatura Máxima
- Temperatura Mínima

En otro archivo *Zona* se grabaron los nombres de las 50 zonas, y se accede con el código de zona. Se pide recorrer el archivo *Tempe* y listar ***nombre y diferencia*** térmica de aquellas zonas con diferencia mayor a 10

```
Program LISTADO;
Type
   Cad20 = string[20];
   TR = record
      Cod: word;
      TMax, TMin: real;
   End;
 TarTemp = file of TR;
 TarZona = file of Cad20;

Procedure Listado(var Temp: TarTemp; var Zona:
TarZona);
Var
   R: TR;   Nom: Cad20;
Begin
  Reset(Temp); Reset(Zona);
  Writeln('ZONA   DIFERENCIA');
  While not Eof(Temp) do
  Begin
    Read(Temp, R);
    If R.TMax-R.TMin > 10 then
    Begin
      Seek(Zona, R.Cod-1);  Read(Zona, Nom);
      Writeln(Nom:30, R.TMax-R.TMin:8:2);
    End;
  End;
  Close(Temp); Close(Zona);
End;

Var
  ATemp: TarTemp;   AZona: TarZona;
Begin  {P.P.}
  Assign(ATemp, 'Tempe.dat');
```

```
  Assign(AZona, 'Zona.dat');
  Listado(ATemp, AZona);
End.
```

Ejemplo 2. Consulta individual de registros

Programa que a partir de la identificación de un alumno, muestra sus datos. La identificación coincide con la dirección, los datos de alumnos son:

- Nombre y Apellido
- Cantidad de materias
- Código y nota de cada una de las materias aprobadas

```
Program ConsultaDirecta;
Type
  TRM = Record
     CodMat, Nota: integer;
  end;
  Cad20 = string[20];
  TV = array[1..30] of TRM;       {arreglo de materias}
  TR = Record {identificacion es dirección del registro en el archivo, no se
  almacena}
      NyA:Cad20;
      CantMat :  Byte;
      VecMat :TV;
  End;
  TArch=File of TR;

Procedure Consulta(Var A:TArch);
Var
    i, Ident: integer;     R: TR;
Begin
  Reset(A);
  Writeln('Ingrese identificacion del alumno a consultar');
  Readln(Ident);
  While Ident <>0 do
  Begin
     Seek(A,Ident-1); Read(A, R);
     With R do
     Begin
       Writeln('Nombre, Apellido: ', NyA);
       Writeln('Cantidad de materias: ', CantMat);
         Writeln('Codigo de materia    Nota');
       For i := 1 to CantMat do
           Writeln(VecMat[i].CodMat:10, VecMat[i].Nota:5);
     End; {With}
     Writeln('Ingrese identificacion del alumno a consultar');
     Readln(Ident);
  End; {while}
  Close(A);
End; {Procedure}

Var
```

```
  A: TArch;
Begin {P.P.}
 Assign(A, 'Alumnos.dat');
 Consulta(A);
End.
```

Resolver el ejercicio 1 propuesto al final de Capítulo

Procesos sobre archivos ordenados

A partir de archivos ordenados es posible aplicar metodologías para mantenimiento de los mismos (incorporar, modificar y/o eliminar información), obtener nuevos archivos, generar informes que muestren resultados obtenidos a partir de los datos almacenados, etc.
Estos procesos consisten en:

- Enfrentamiento de Archivos (fusión, mezcla o apareo)
- Cortes de Control

Enfrentamiento de Archivos (fusión, mezcla o apareo)

Es un proceso por el cual dos o más archivos, **ordenados por el mismo campo de secuencia (clave primaria o secundaria)**, se enfrentan registro a registro para alguna tarea específica, obteniéndose como resultado listados y/o archivos.

Un proceso típico de esta metodología es denominado ABM y permite actualizar la información de un archivo, contempla:

- ALTAS: proceso que permite incorporar nuevos registros en un archivo
- BAJAS: proceso que permite eliminar registros de un archivo
- MODIFICACIONES: proceso que permite modificar la información almacenada en registros de un archivo (salvo la clave)

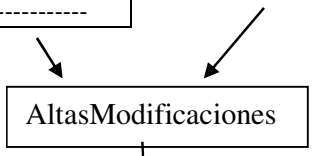
Ejemplo 3

Se tienen los datos de los ejemplares de una biblioteca, grabados en un archivo **Biblioteca** y en otro archivo **Compras** los datos de los títulos adquiridos (ambos del mismo tipo y orden)

Código de Libro {ANU4, campo de secuencia, clave primaria}
Cantidad
Descripción

Archivo **Biblioteca**		
Codigo	Cantidad	Descripcion
AAB2	6	xxxxxxxxxx
BZU7	0	xxxxxxxxxx
JUT5	20	xxxxxxxxxx
MKO9	8	xxxxxxxxxx
PTR3	5	xxxxxxxxxx
ZZZZ	--	----------

Archivo **Compras**		
Codigo	Cantidad	Descripcion
ABH5	12	xxxxxxxxxx
BZU7	10	----------
LJY6	9	xxxxxxxxxx
PTR3	5	----------
ZZZZ	--	----------



Archivo **Temp** (Biblioteca actualizado)		
Codigo	Cantidad	Descripcion
AAB2	6	xxxxxxxxxx
ABH5	12	xxxxxxxxxx
BZU7	10	xxxxxxxxxx
JUT5	20	xxxxxxxxxx
LJY6	9	xxxxxxxxxx
MKO9	8	xxxxxxxxxx
PTR3	10	xxxxxxxxxx
ZZZZ	--	----------

Se genera un tercer archivo **Temp,** con la información actualizada de Biblioteca, que reemplazará al original.

Proceso de AltasModificaciones utiliza los archivos Biblioteca y Compras

El proceso de enfrentamiento de archivos se asemeja al de intercalación de dos arreglos ordenados, donde se compara un elemento de cada uno y el de menor valor es el procesado, incrementando el índice del arreglo del cual provino. Esto se repite hasta que uno o ambos arreglos finalicen. En el primer caso los elementos del arreglo que no terminó serán procesados en su totalidad.

PARA ARREGLOS ERA ASÍ

```
   V1          V2          V3
I := 1      J := 1      K := 0
```

```
WHILE (I<=N) AND (J<=M) DO
BEGIN
     K := K+1
```

Este proceso aplicado a dos archivos requiere que los mismos hayan sido **ordenados** con el **mismo criterio** (campo de secuencia). Los archivos se leen secuencialmente, la memoria almacena un registro de cada uno, el de clave menor es procesado (grabado en un archivo de salida, listando, etc) y se avanza leyendo otro registro del archivo del cual provino. Esto se repite hasta que uno o ambos archivos alcancen el final. En el primer caso los registros del archivo que no finalizó deben ser procesados en su totalidad.

Para **simplificar** el enfrentamiento y resolver dentro de un **único ciclo** el proceso de la totalidad de los registros de ambos archivos, se graba un **registro "centinela"** al final de los archivos, con un valor de clave mayor a todas. De esta forma al finalizar un archivo no se detiene el ciclo. **El archivo que finaliza primero deja en memoria su registro centinela con clave mayor a todas las pendientes de proceso en el otro archivo, dando "paso" a éstos hasta agotarlos** (siempre se procesa el registro con clave menor y se avanza en el archivo asociado).

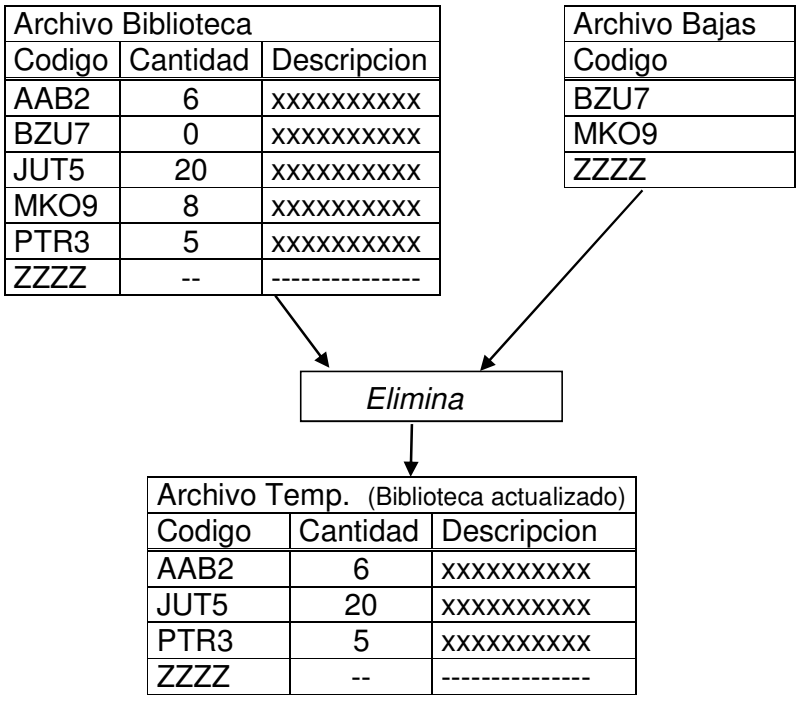
El ciclo finaliza cuando ambos archivos alcanzan el final y sus registros centinelas están en memoria.

```
While Not Eof(Biblioteca) or Not Eof(Compras) do
    If ... < ... then
      ...
   else
      If ... > ... then
        ...
      else        { por = }
```

```
Program ABM;
Type
  St4 = string[4];     St20 = string[20];
  TR = Record
    Codigo: St4;
    Cantidad: Word;
    Descripcion: String;
  End;
  TAr = File of TR;
  TArB = File of St4; { para almacenar código de libros, utilizado en el proceso de bajas}

Procedure AltasModificaciones(Var Biblioteca, Compras: TAr);
...
Procedure Elimina(Var Biblioteca: TAr; Var Bajas: TArB);
...

Var
  Biblioteca, Compras: TAr;
  Bajas: TArB;

Begin {P.P, se presupone que los registros centinela están grabados}
  Assign(Biblioteca, 'Biblio.dat');
  Assign(Compras, 'Compras.dat');
  Assign(Bajas, 'Bajas.dat');
  AltasModificaciones(Biblioteca, Compras);
  Elimina(Biblioteca, Bajas);
End.

{se considera compra de nuevos títulos (altas) y reposición (modificación)}
Procedure AltasModificaciones(Var Biblioteca, Compras: TAr);
Var
  Temp: Tar;
  RB,RC: TR;
Begin
  Assign(Temp, 'Temp.Dat');  Rewrite(Temp);
  Reset(Biblioteca);
  read(Biblioteca, RB); read(Compras, RC);
  While not Eof(Biblioteca) or not Eof(Compras) do
    If RB.Codigo < RC.Codigo Then
    Begin
       write(Temp, RB);
```

```
       read(Biblioteca, RB)
    End
    Else
      If RB.Codigo > RC.Codigo Then   {alta}
      Begin
        write(Temp, RC);
        read(Compras, RC);
      End
      Else { reposición, los códigos son iguales }
      Begin
        RB.Cantidad := RB.Cantidad + RC.Cantidad
        write(Temp,RB);
        read(Biblioteca, RB);
        read(Compras, RC);
      End; { fin del if y fin del while }
  write(Temp, RB); { para grabar el centinela en el archivo actualizado}
  Close(Biblioteca);  Close(Compras);  Close(Temp);
End; {Procedimiento}
```

El archivo **Bajas**, almacena los códigos de libro que se desean eliminar del archivo biblioteca (sus componentes son de tipo cadena, clave primaria ordenadas como Biblioteca)




```
{se eliminan registros }
Procedure Elimina(Var Biblioteca: TAr; Var Bajas: TArB);
Var
  Temp: TAr;
  RB: TR;    R: St4;
Begin
  Assign(Temp, 'Temp.Dat');
  Reset(Biblioteca);
```

```
  Reset(Bajas);
  Rewrite(Temp);
  read(Biblioteca, RB);
  read(Bajas, R);
  While Not Eof(Biblioteca) or Not Eot(Bajas) do
    If RB.Codigo < R Then
    Begin
      write(Temp, RB);
      read(Biblioteca, RB);
    End
    Else
      If RB.Codigo > R Then
        read(Bajas, R) { R Baja incorrecta}
      Else     {son iguales los códigos, no grabo en TEMP}
      Begin
        Read(Biblioteca, RB);
        Read(Bajas, R);
      End;
  write(Temp, RB); { para grabar el centinela en el archivo actualizado}
  Close(Biblioteca);   Close(Bajas);   Close(Temp);
End; {Procedure}
```

Resolver los ejercicios 2 y 3 propuestos al final de Capítulo

CORTES de CONTROL 1er y 2do nivel

Son programas que generan listados a partir de archivos ordenados y cuyo campo de secuencia es clave secundaria. Esto implica que la información se agrupa en "bloques" pertenecientes a un mismo valor de clave y el proceso consiste en controlar esos bloques, listando la información obtenida a ese nivel (totales, promedios, máximos).
Si el control se realiza sobre un solo campo de secuencia, se llama corte de 1er. Nivel, si se controlan dos campos (ej: materia y dentro de cada una de ellas por matrícula), corte de 2do. Nivel y así sucesivamente. La información que se obtiene del archivo está en los distintos niveles que se realiza el control.

Ejemplo 4 CORTE de 1er. NIVEL

Se tiene un archivo MULTAS, con los **importes** por infracciones de tránsito registrados en diferentes **zonas** por distintos vehículos (**patentes**), el diseño de registro es:

ZONA (cadena de 4) {campo de secuencia, clave secundaria}
PATENTE
IMPORTE

Se desea calcular por zona, cantidad de infracciones e importe total de las mismas.

ARCHIVO MULTAS

ZONA	PATENTE	IMPORTE
A34	CAN180	80
A34	CAN180	100
A34	CAN180	90
A34	DEX901	50
A34	DEX901	40
C78	CAN180	30
C78	CAN180	100
C78	RUN100	80
C78	RUN100	100
C78	RUN100	60
ZZZ	------	

➔

LISTADO

ZONA	***CANT.INFRACCIONES***	***IMPORTE TOTAL***
A34	5	360
C78	5	370
TOTAL	*10*	

El bloque correspondiente a la ZONA **A34**, genera una línea de listado con Cantidad de Infracciones 5 y Total 360, y acumula en el Total la cantidad de infracciones.

El proceso consiste en leer y operar los registros de cada bloque, para ello se debe verificar que el campo de secuencia se mantenga con el mismo valor (A34), para lo cual se utiliza una variable de control, si el registro pertenece al bloque se procesa contando la infracción y sumando el importe (según cada caso realizando el proceso pertinente).

Cuando el campo de secuencia no coincide con el de la variable de control, indica que finaliza el bloque y se produce una ruptura o ***corte,*** desencadenando una serie de acciones finales de acuerdo a la índole del problema y asociadas a dicho bloque.

Los archivos deben tener grabado un registro "centinela", con un valor ('ZZZ') mayor al del campo de secuencia del último bloque, la función de este registro (la información que contiene no se procesa) es producir la ruptura o corte del último bloque.

Cada vez que comienza el proceso de un nuevo bloque se inicializan variables (entre ellas la que controla el campo de secuencia).
Se describe, en forma general, lo dicho mediante el siguiente algoritmo:

```
Títulos/ Iniciar variables generales
Read(archivo, registro)
While  Not  Eof(archivo) do
   Comienzo de bloque: Subtítulos/Inicializar variables de bloque
   While mismo bloque do
      Procesar registro
      Read(archivo, registro)
   Fin bloque: Acciones finales del bloque (escribir resultados del bloque,
   comparar, acumular, etc.)
Acciones finales del programa (Escribir resultados generales)
```

procesa un bloque

Cuando se especifica que el listado debe tener línea de detalle, debe listarse la información contenida en cada uno de los registros procesados.

```
Program Corte1;
Type
  St3 = string[3]; St6 = string[6];
  TReg = record
     Zona: St3;
     Patente:  St6;
     Importe: real;
  End;
  TArch = file of TReg;

Procedure ListadoCorte1(var Ar: TArch);
Var
  R: TReg;
  Total, Cont: word;
  SumImp: real;
  ZonaAct: St3;
begin
  reset(Ar); read(Ar, R); Total := 0;
  writeln('Zona   Cantidad Infracciones     Importe Total');
  while not Eof(Ar) do
  begin
    Cont := 0; SumaImp := 0;                {comienza el bloque de ZONA}
    ZonaAct := R.Zona;
    while ZonaAct = R.Zona do
    begin
       Cont := Cont + 1;
       SumaImp := SumaImp + R.Importe;      {procesa registros ZONA}
       read(Ar, R);
    end;
    writeln(ZonaAct, Cont, SumaImp:8:2);    {finaliza bloque ZONA}
    Total := Total + Cont;
  end;
  close(Ar);
  writeln('Total = ', Total);
end;

Var
   Ar: TArch;
Begin
  Assign(Ar, 'Multas.dat');
  ListadoCorte1 (Ar);
End.
```

Ejemplo 5 - CORTE de 2do. NIVEL

A partir del mismo archivo MULTAS del ejercicio anterior y teniendo en cuenta el siguiente orden:

ZONA {1er. campo de secuencia, clave secundaria}
PATENTE {2do. campo de secuencia, **clave secundaria**}
#IMPORTE

Se desea obtener un listado que informe en cada zona el total por patente y la patente con mayor importe de infracciones.

ARCHIVO MULTAS		
ZONA	PATENTE	IMPORTE
A34	CAN180	80
A34	CAN180	100
A34	CAN180	90
A34	DEX901	50
A34	DEX901	40
C78	CAN180	30
C78	CAN180	100
C78	RUN100	80
C78	RUN100	100
C78	RUN100	60
ZZZ		

➔

LISTADO

ZONA A34

PATENTE	*TOTAL*
CAN180	270
DEX901	90

Patente con mayor importe CAN180

ZONA C78

PATENTE	TOTAL
CAN180	130
RUN100	240

Patente con mayor importe RUN100

Dentro de cada "bloque zona" se forman los "bloques patente" donde se agrupan las infracciones de la misma patente en dicha zona. El control de una zona implica el control de los bloques de patente que contiene. Por lo tanto se repite la necesidad de utilizar una variable que permita verificar para cada registro si se trata de la misma patente.

Como se indica en el ejemplo, para cada campo de secuencia que se desea controlar se tratan los bloques que se forman, implementando tres etapas: inicialización, proceso y finalización. En el caso de querer controlar más de un campo de secuencia estas etapas se anidan. Dentro del proceso del 1er. campo de secuencia se realizan la inicialización, proceso y finalización del bloque determinado por el 2do. campo de secuencia.

```
Program Corte2;
Type
  St3 = string[3]; St6 = string[6];
  TReg = record
     Zona: St3;
     Patente: St6;
     Importe: real;
  End;
  TArch = file of TReg;

Procedure ListadoCorte2(var Ar: TArch);
Var
```

```
   R: TReg;
   Max, SumImp: real;
   ZonaAct: St3;
   PatenteAct, MaxPatente: St6;
begin
  reset(Ar); read(Ar, R);
  while not Eof(Ar) do
  begin
    writeln('ZONA ', R.Zona);
    writeln('ZONA      TOTAL');
    Max := 0;                                   {comienza el bloque de ZONA}
    ZonaAct:= R.Zona;
    while ZonaAct = R.Zona  do
    begin
      PatenteAct := R.Patente;                {comienza bloque PATENTE}
      SumImp := 0;
      while(ZonaAct = R.Zona)and(PatenteAct = R.Patente) do {proc bloque ZONA}
      begin
        SumImp := SumImp + R.Importe;         {procesa registro PATENTE}
        read(Ar, R);
      end;
      writeln(PatenteAct, SumImp);
      if Max < SumImp then                    {finaliza bloque PATENTE}
      begin
        Max := SumImp;
        MaxPatente := PatenteAct;
      end;
    end;
    writeln('Patente con mayor importe', MaxPatente);   {finaliza bloque ZONA}
  end;
  close(Ar):
end;

Var
  Ar: TArch;
Begin
  Assign(Ar, 'Multas.dat');
  ListadoCorte2(Ar);
End.
```

Resolver los ejercicios 4 y 5 propuestos al final de Capítulo

Ejercitación

Para cada problema deberá plantearse un juego de datos, luego grabarlos generando el archivo de prueba y listar el archivo para verificar que no haya errores en la carga.

1.- Crear un archivo ALUMNOS, con los datos de los alumnos de Programación A. El diseño de registro es:

DNI
APELLIDOyNOMBRE
NotaTP1, Par1, NotaTP2, Par2

Teniendo en cuenta el cálculo de la nota final:

NF = 0.30 * Par1 + 0.10 * TP1 + 0.40 * Par2 + 0.20 * TP2

Desarrollar un programa que lea el archivo y muestre los datos de los alumnos que promocionaron, la cantidad de alumnos que regularizaron y el porcentaje de desaprobados.
A un parcial no rendido o un TP no entregado le corresponde nota cero.

Recordar que:

- Para promocionar la materia deberá tener aprobados los dos Trabajos Prácticos y los dos parciales, con NF >= 7. Dicha nota será la definitiva
- Para regularizar la materia deberá tener aprobados los dos parciales y al menos un Trabajo Práctico con 4 <= NF < 7.
- Cualquier otra situación, lleva a desaprobar la cursada

Desarrollar un procedimiento que a partir de las 4 notas devuelva el promedio y la situación del alumno (P, R ó D)

2.- Dado un archivo CREDITOS que contiene información sobre créditos de clientes, cuyo diseño es:

Cod Cliente {ANU5, campo de secuencia, clave primaria}
Cuotas Totales
Cuotas Pagadas

Y otro archivo PAGOS con información de los clientes que han pagado una cuota el mes en curso, con el siguiente diseño:

Cod Cliente {ANU5, campo de secuencia, clave primaria}

Se pide enfrentar ambos archivos para obtener un tercero con los créditos actualizados (cuotas pagadas), en caso que hayan pagado todas las cuotas eliminarlos del nuevo archivo.

Por fin de proceso informar la cantidad de clientes que completaron el pago del crédito y la cantidad que no pagaron la cuota el mes en curso.

3.- Una empresa de servicios médicos, almacena en un archivo AFILIADOS los siguientes datos:

DNI { campo de secuencia, clave primaria}
Plan (1..4)
Fecha de alta (aaaa/mm/dd)

En otro archivo NOVEDADES se han grabado solicitudes de ingreso o de cambio de plan (solo el próximo superior o inferior) con el mismo diseño:

DNI { campo de secuencia, clave primaria}
Plan (1..4)
Fecha de alta o cambio (aaaa/mm/dd)

Se pide enfrentar ambos archivos para actualizar AFILIADOS en un nuevo archivo, considerar que los cambios de plan incorrectos no serán tenidos en cuenta y se generará un listado de los mismos.
Indicar al final del proceso la cantidad de afiliados que no cambiaron el plan (ya sea que no lo solicitaron o que no se pudo realizar)

4.- Una droguería vende medicamentos a distintas farmacias, registra dichas operaciones en el archivo VENTAS de la siguiente forma:

- Cod.Farmacia {ANU5, campo de secuencia, clave secundaria}
- Cod.Medicamento {ANU3}
- Cant.Envases
- VentaLibre (S/N)

A partir del archivo listar para cada farmacia:
El código de la Farmacia, la Cantidad de medicamentos de venta libre y la cantidad con receta, el código de medicamento con más envases vendidos.
Por fin de proceso, el promedio de medicamentos (no de envases) vendido por farmacia.

El listado deberá tener el siguiente formato:

```
Cod.Farmacia    Cant.Envases V/L     Cant.Envases C/R         Cod.Medicamento más envases
   xxxxx             999                  999                         xxx
   xxxxx             999                  999                         xxx
   .....             ...                  ...                         ...
Promedio de medicamentos vendidos por farmacia  99999
```

5.- En el archivo NOTAS, cada alumno de la universidad registra los resultados de exámenes en diferentes materias:

- DNI {ANU8, 1er. campo de secuencia, clave secundaria}
- Cod. Materia {ANU3, 2do. campo de secuencia, clave secundaria}
- Nota

Considerando que un alumno aprueba si obtiene promedio mayor a 5 y no más de un examen desaprobado (nota<4), emitir el siguiente listado:

```
Documento:  xxxxxxxx
Materia         Cantidad de exámenes        Nota Promedio         Aprobó(S/N)
   xxx                  99                     99.99                 x
   xxx                  99                     99.99                 x
   ...                  ..                     .....                ..
Aprobó 99 materias

Documento:  xxxxxxxx
Materia         Cantidad de exámenes        Nota Promedio         Aprobó(S/N)
   xxx                  99                     99.99                 x
   xxx                  99                     99.99                 x
   ...                  ..                     .....                ..
Aprobó 99 materias
.....
El alumno con documento xxxxxxxx fue el que obtuvo la mayor cantidad de aplazos (99)
```