# Algorítmica y Programación II

## Archivos

# Archivos

- Estructuras de datos homogénea guardada en dispositivo secundario.

- Nos interesa avanzar sobre aquellos que su estructura es un **conjunto de elementos (registros) del mismo tipo**.

# Archivos

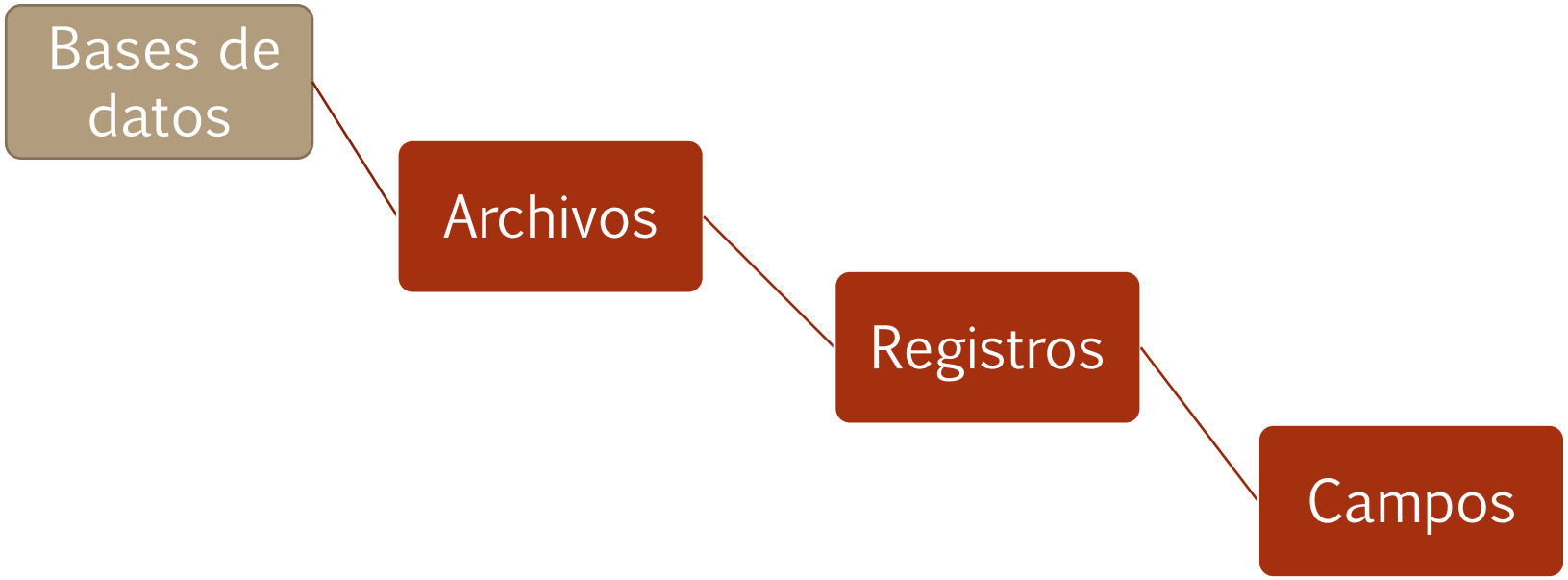
- Características
  - Pueden utilizarlos varios programas
  - La información es permanente
  - Los tiempos de acceso se miden en milisegundos.
  - No hace falta que estén definidos previamente en la memoria auxiliar.
  - El nexo entre el archivo y el programa es el S.O.
  - El tamaño será relativo a los datos que alberga.

# Archivos

- Operaciones:
  - Vincular (descriptor lógico con archivo físico).
  - Abrir el archivo (varios modos).
  - Leer elementos de un archivo.
  - Escribir nuevos elemento o modificar existentes.
  - Cerrar el archivo.
  - Borrar el archivo.

# Archivos

› Jerarquización y relación de composición.



# Archivos

## › Organización

Según la forma en que se guardan los elementos :

- Secuencial: se guarda cada elemento de forma consecutiva.
- Directa (aleatoria): el orden físico no coincide con el orden lógico.

  Los elementos se recuperan por su posición dentro del archivo.

  Debe existir una función que relacione el nro. de elemento con la posición dentro del archivo.

# Archivos

- Organización
  - Secuencial indexada: es una mezcla de las anteriores.

    Se accede directamente a un índice y desde allí secuencialmente al elemento buscado.

    Ej: Un contacto, primero accedo con la letra de su apellido y luego, secuencialmente, recorro hasta encontrar el adecuado.

# Archivos

## › Acceso

Según el modo en que se organice el contenido, se consideran dos tipos de acceso:

- Secuencial: se accede en orden, desde el 1er elemento al último elemento.
- Directo: se accede directamente al elemento sin la necesidad de haber pasado por los anteriores.

# Archivos

› Tipo de archivo

- De texto: organización y acceso secuencial. Conjunto de caracteres.
- Tipados: organización directa, acceso secuencial o directo.
- Sin Tipo: su organización y acceso será secuencial.
  - Las lecturas y escrituras de realizan dependiendo el tipo de archivo.
  - Las lecturas y escrituras se hacen a nivel de byte

# Archivos

› Archivos de texto (pascal)

```
Type
     TArchTextoT = file of text;
     TArchInt    = file of integer;
     TARchTextoC = file of char;
Var
     archivo : TArchTextoC;
     arch_tex: TextFile;
```

Un archivo de texto estará compuesto por de una serie de caracteres separados por marcas, fin de línea y fin de archivo.

# Archivos

## › Archivos de texto (pascal)

- El retorno de carro (CR), que consiste en empujar el *carro* (el cilindro horizontal donde se apoya el papel) hacia la izquierda.
- El salto de línea (LF), que consiste en girar el *carro* un poco para que el papel se desplace una línea.

**CRLF** es simplemente CR seguido de LF, por tanto su codificación es en hexadecimal **0D 0A**, o **13 10** en decimal.

# Archivos

## › Archivos de texto (pascal)

- **file of char** is just a simple sequence of single byte characters and you can only read or write a single character at a time. That is, you can only call **Read(F, C)** or **Write(F, C)** where C is variable of type **char**.
- **TextFile** offers much more functions, and represents the usual concept of a text file. You can use Read, Readln, Write, Writeln to read/write from a text file a number of standard types, like strings, integers and floating-point values. Line endingsare also automatically handled: when reading, various line endings are recognized; when writing, the current OS line endings are used.

# Archivos

## › Archivos tipados (pascal)

- La **organización** de este tipo de archivo es **directa**. Por lo que admite acceso secuencial o directa.

- La numeración de **los registros empieza con el número CERO** , por lo que al elemento_1 se le llamará registro 0, al elemento_2 registro 1, y así sucesivamente hasta llegar al fin de archivo.

# Archivos – Tipados - declaración

```
Type
    Tpersona = record
        nombre,
        apellido:String;
    end;
    TArchPersonas = File of TPersona;
Var archivoPersonas:TArchPersonas;
```

# Archivos

## › Operaciones

- `AssignFile(file, ‘ruta+archivo.dat')`
  Relaciona el descriptor del archivo físico con su variable.

- `Reset(file)`
  Abre un archivo existente para su procesamiento (lectura\escritura), coloca el apuntador de registro en el primer registro (0).

- `Close(file)`
  Cierra el archivo.

# Archivos

- `Rewrite(file)`
  Crea un nuevo archivo (**o sobre-escribe en uno existente**) y lo abre para procesamiento con el apuntador de registro colocado en el registro 0.
- `Read(file, variable)`
  Lee el elemento del archivo y lo asigna a la variable.
- `Write(file, variable)`
  Escribe el elemento en el archivo según el apuntador.
- `Erase(file)`
  Borra físicamente el archivo.

# Archivos

- `Filesize(file)`
  Devuelve la cantidad de registros que posee el archivo.
- `FilePos(file)`
  Devuelve el número de registro donde actualmente se encuentra el cursor de lectura o escritura.
- `EOF(file)`
  Devuelve True si encontró la marca de fin de archivo.
- `Seek(file, Pos)`
  Coloca el cursor de E/S, en Pos.
- `Rename(file, 'ruta+archivo.dat')`
  Renombra el archivo.

# Archivos

## › Eliminar registros

Cuando se requiere eliminar registros, que ya no se utilizan de un archivo, se puede proceder de dos maneras:

- **Forma lógica:** El espacio no se libera. En ese registro se pude poner la información de otro registro o dejarla inutilizada.
- Ventaja: si no se regraba información, y se borra por error, la información del registro borrado puede ser recuperada.
- Desventaja: puede generar archivos muy grandes, con poca información relevante.

# Archivos

## › Eliminar registros

- **Forma física:** se recupera realmente el espacio que el archivo ocupa en el disco. Para ello hay que hacer **corrimientos o generar un archivo nuevo** que contenga sólo los registros con información significativa.
- Desventaja: tiempo de proceso. La información del registro borrado no puede ser recuperada

# Archivos - Ejemplo

```
function esArchivo(rutaNombre : String) : boolean;
var F : File…; pudoAbrir:boolean;
begin
  {$I-}
  Assign(F,rutaNombre);
  Reset(F);
  {$I+}
  pudoAbrir:=(IoResult=0);
  if pudoAbrir then
    Close(f);
end;
```

# TDA - Bibliografía


- Introducción a la programación. Maria, Abásolo y Francisco Perales.
- Introducción a la programación con pascal. Rafael, Llavori.
- Algoritmos, Datos y programas. Armando, De Gusti.
- http://wiki.freepascal.org/File_Handling_In_Pascal