

Archivos

Introducción a la programación II

Definición de archivos y tipos de archivo

Un archivo es una colección de información almacenada como una unidad en memoria secundaria o externa.

Su tamaño no es fijo y está limitado por la cantidad de memoria secundaria. Así, no requiere tamaños predefinidos.

Cada archivo es referenciado por un identificador (su nombre).



Pendrive



Disco Duro Externo



Disco duro Magnético



Tarjetas Extraíbles



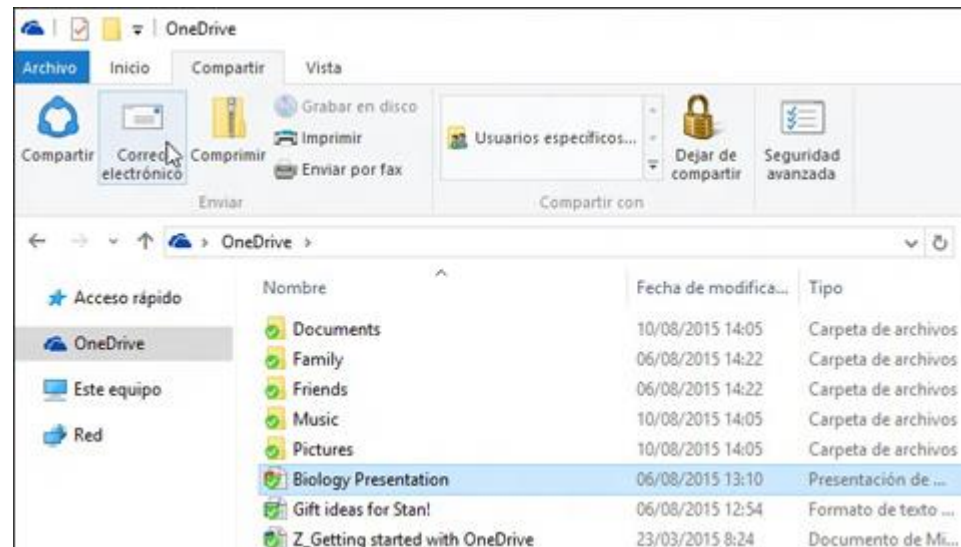
CD/DVD (Unidad Óptica)

Definición de archivos y tipos

Los elementos de un archivo pueden ser de cualquier tipo, simple o estructurado.

En Pascal existen 3 tipos de archivos:

- **archivos tipeados** (tipificados) o con tipo (**file of**). Son de **acceso aleatorio** a cualquier elemento a través de su posición.
- **archivos de texto** o **archivos secuenciales**, que exige una exploración secuencial desde el primer elemento.
- **no tipeados** (no tipificados) o sin tipo (file) (**No serán utilizados en la materia**)



Manejo de archivos

Para utilizar un archivo dentro de un programa es necesario:

- Declarar la variable asociada al archivo
- Abrir el archivo
- Leer y/o escribir en el archivo
- Cerrar el archivo

```
Program ...;
```

```
...
```

```
Procedure ...(...);
```

```
...
```

```
begin
```

```
...
```

```
//abrir archivos
```

```
//operar con archivos
```

```
//cerrar archivos
```

```
...
```

```
end;
```

```
...
```



La apertura y cierre de archivos debe mantenerse en el mismo bloque y alcance

Manejo de archivos de texto

Para declarar un archivo de texto se realiza:

```
var variablearchivo : text;
```

Una vez declarada la variable para poder utilizarla el primer paso es asignar a la misma el nombre de algún archivo:

```
assign (variablearchivo, NombreArchivo);
```

donde **NombreArchivo** es una cadena de caracteres que contiene el nombre del archivo, la unidad de disco donde se encuentra y el directorio. Por ejemplo:

```
var variablearchivo : text;  
begin  
...  
    assign(variablearchivo, '/ip2/prueba.dat');  
    // '/ip2/' se debe agregar para el entorno desde moodle  
...  
end;
```

Manejo de archivos de texto

Existen tres formas de abrir un archivo de texto:

reset(variablearchivo);

Procedimiento que abre un archivo existente para una operación de lectura. Si el archivo especificado no existe, se producirá un error de E/S.

rewrite(variablearchivo);

Procedimiento que crea y abre un nuevo archivo para una operación de escritura. Si el archivo especificado existe, se borra el contenido.

append(variablearchivo);

Procedimiento que abre un archivo existente para añadir datos al final del mismo. Si el archivo no existe, se produce un error de E/S; y si ya estaba abierto, primero se cierra y luego se reabre.

Manejo de archivos de texto

Para dar solución al error de E/S se suele definir una función de existencia de archivo:

```
function existearchivo(var variablearchivo:text):boolean;  
begin  
    {$I-}  
    reset(variablearchivo);  
    {$I+}  
    existearchivo:=(IOResult = 0);  
end;
```

¿Por qué una variable de tipo archivo debe ser pasada por referencia cómo parámetro incluso en una función?

Dado que es una variable asociada a la memoria secundaria, no se podría crear una copia de dicha memoria cuando se pasa por parámetro. Con lo cual, las variables de tipo archivo **SIEMPRE** se pasan como parámetros por referencia.

Manejo de archivos de texto

Ejemplo: buscar en archivo:

```
function buscarenarchivo(var variablearchivo:text;elemento:string):boolean;  
begin  
    ...  
    buscarenarchivo:=...;  
end;
```

Ejemplo: ordenar archivo:

```
procedure ordenararchivo(var variablearchivo:text);  
begin  
    ...  
    //por inserción, selección, burbujeo;  
    ...  
end;
```


Manejo de archivos de texto

Para leer datos de un archivo abierto se realiza:

```
read (variablearchivo, elemento);  
readln (variablearchivo, elemento);
```

donde **elemento** es una variable que tiene un texto, o sea un char o un string.
Para escribir datos de un archivo abierto se realiza:


```
write (variablearchivo, elemento);  
writeln (variablearchivo, elemento);
```

Observación:

Los procedimientos **read**, **readln**, **write** y **writeln**, después de cada invocación el lugar actual del archivo se posiciona en el siguiente elemento.

```
Ejemplo: elemento:char;  
...  
read(variablearchivo,elemento);  
...
```

```
Ejemplo: elemento:string;  
...  
readln(variablearchivo,elemento);  
...
```

H o l a


Programación


Manejo de archivos de texto

Para verificar el estado actual del recorrido secuencial del archivo se utiliza:

eof (variablearchivo);

Función booleana que indica si el fin de archivo se ha encontrado; devuelve true si se encontró, false en caso contrario.

eofln (variablearchivo);

Función booleana que indica si la posición actual está en la marca de fin de línea.

Para cerrar un archivo de texto:

close(variablearchivo);

Observación: la apertura y cierre del archivo deberían estar en el mismo alcance a nivel de código.

Ejemplo moodle

//Ejemplo: escribir y leer un archivo de texto, visualizando su contenido por pantalla
//Para hacer andar este ejemplo en el entorno de ejercitación de moodle
//crear un fichero ejemplo.pas, copiar y pegar todo el contenido, guardar, depurar, ejecutar

Program ejemplo;

var **variablearchivo : text;**
 elemento:char;

begin

assign(variablearchivo, '/ip2/prueba.dat');

rewrite(variablearchivo);

writeln(variablearchivo,'Uno');

writeln(variablearchivo,'Dos');

close(variablearchivo);

reset(variablearchivo);

while not eof(variablearchivo) do

begin

read(variablearchivo,elemento);

write(elemento);

end;

close(variablearchivo);

end.

Manejo de archivos de texto

Ejemplo: transferir los datos de un archivo de texto origen a otro destino

```
Procedure transferir_dato(var origen,destino : text);  
var      elemento:char;  
begin  
    reset(origen);  
    rewrite(destino);  
    while not eof(origen) do  
    begin  
        readln(origen,elemento);  
        writeln(destino,elemento);  
    end;  
    close(origen);  
    close(destino);  
end;
```

Manejo de archivos de acceso aleatorio

Este tipo de archivos, también llamados archivos binarios, contienen datos de **tipo simple o estructurado**. Los elementos son accesibles directamente con solo conocer su posición.

Para declarar un archivo de acceso aleatorio se realiza:

type

tipoarchivo = file of <tipo simple o estructurado>;

var

variablearchivo : tipoarchivo;

elementodearchivo : <del mismo tipo con el que fue definido>;

Ejemplo: archivo de enteros

type

tipoarchivo = file of integer;

var

variablearchivo : tipoarchivo;

elementodearchivo : integer;

Manejo de archivos de acceso aleatorio

Ejemplo: archivo de tipo estructurado (arreglo de enteros)

const

MAX = 10;

type

arreglo = array[1..MAX] of integer;
tipoarchivo = file of **arreglo**;

var

variablearchivo : tipoarchivo;
elementodearchivo : **arreglo**;

Manejo de archivos de acceso aleatorio

Una vez declarada la variable para poder utilizarla el primer paso es asignar a la misma el nombre de algún archivo:

assign (variablearchivo, NombreArchivo);

Hay dos formas de abrir un archivo de acceso aleatorio:

reset(variablearchivo);

Procedimiento que abre un archivo existente para una operación de lectura. Si el archivo especificado no existe, se producirá un error de E/S.

rewrite(variablearchivo);

Procedimiento que crea y abre un nuevo archivo para una operación de escritura. Si el archivo especificado existe, se borra el contenido.

Observación: Con las mismas implicancias de la existencia y pasaje por parámetros de los archivos de texto.

Manejo de archivos de acceso aleatorio

Para leer datos de un archivo abierto se realiza:

read (variablearchivo, elemento);

donde elemento es una variable del mismo tipo de dato con el que fue definido el archivo.

Para escribir datos de un archivo abierto se realiza:

write (variablearchivo, elemento);

Para cerrar un archivo se usa:

close(variablearchivo);

Observación:

- La apertura y cierre del archivo deberían estar en el mismo alcance a nivel de código.
- Los procedimientos **read** y **write**, después de cada invocación el lugar actual del archivo se posiciona en el siguiente elemento.

Manejo de archivos de acceso aleatorio

Para realizar el acceso aleatorio es necesario posicionarse y analizar si se está dentro de los límites de donde hay elementos (no estar fuera de rango). Las posiciones en el archivo van de **0** a **N-1**, donde **N** es la cantidad de elementos.

eof (variablearchivo); //función booleana que indica si es el fin de archivo

Para posicionarse se utiliza:

seek(variablearchivo,posicion);

Procedimiento que me posiciona en dicha posición dentro del archivo.

filepos(variablearchivo);

Función que devuelve un entero con la posición actual.

filesize(variablearchivo);

Función que devuelve un entero con el número de elementos.

Ejemplo moodle

//Ejemplo: escribir y leer un archivo de texto,
visualizando su contenido por pantalla
//Para hacer andar este ejemplo en el entorno de
ejercitación de moodle
//crear un archivo ejemplo.pas, copiar y pegar todo
el contenido, guardar, depurar, ejecutar

Program ejemplo;

type

integerfile = file of integer;

**Procedure crear_archivo(var variablearchivo :
integerfile);**

begin

**rewrite(variablearchivo);
write(variablearchivo,1);
write(variablearchivo,2);
close(variablearchivo);**

end;

**Procedure leer_archivo(var variablearchivo :
integerfile);**

var elemento:integer;

begin

reset(variablearchivo);

while not eof(variablearchivo) do

begin

**read(variablearchivo,elemento);
writeln(elemento);**

end;

close(variablearchivo);

end;

var variablearchivo : integerfile;

begin

**assign(variablearchivo, './ip2/prueba.dat');
crear_archivo(variablearchivo);
leer_archivo(variablearchivo);**

end.

Manejo de archivos de acceso aleatorio

Ejemplo: modificar el contenido del registro ubicado en pos de un archivo de enteros incrementando en uno su valor.

```
...
type
    tipoarchivo = file of integer;
...
Procedure modificar_registro(var inventario : tipoarchivo; pos : integer);
begin
    reset(inventario);
    seek(inventario,pos);
    read(inventario,elemento);
    seek(inventario,pos);
    write(inventario,elemento+1);
    close(inventario);
end.
...
```

Ejemplo moodle

//Ejemplo: escribir, modificar y leer un archivo de texto,
visualizando su contenido por pantalla

//Para hacer andar este ejemplo en el entorno de ejercitación de moodle

//crear un fichero ejemplo.pas, copiar y pegar todo el contenido,
guardar, depurar, ejecutar

Program ejemplo;

type

integerfile = file of integer;

Procedure crear_archivo(var variablearchivo : integerfile);

begin

rewrite(variablearchivo);

write(variablearchivo,1);

write(variablearchivo,2);

close(variablearchivo);

end;

Procedure leer_archivo(var variablearchivo : integerfile);

var elemento:integer;

begin

reset(variablearchivo);

while not eof(variablearchivo) do

begin

read(variablearchivo,elemento);

writeln(elemento);

end;

close(variablearchivo);

**Procedure modificar_registro(var variablearchivo : integerfile; pos
: integer);**

var elemento:integer;

begin

reset(variablearchivo);

seek(variablearchivo,pos);

read(variablearchivo,elemento);

seek(variablearchivo,pos);

write(variablearchivo,elemento+1);

close(variablearchivo);

end;

var

begin

variablearchivo : integerfile;

assign(variablearchivo, '/ip2/prueba.dat');

crear_archivo(variablearchivo);

leer_archivo(variablearchivo);

modificar_registro(variablearchivo,1);

leer_archivo(variablearchivo);

end.

Manejo de archivos de acceso aleatorio

Ejemplo: crear un archivo de posiciones y verificar en la lectura si fue bien guardado

```
...
Procedure crear_archivo_pos(var archivo : tipoarchivo);
var
    pos:integer;
begin
    rewrite (archivo);
    for pos:=0 to MAX-1 do
        Write (archivo,filepos(archivo));
    close (archivo);

end;
...
Function verificar_contenido(var archivo : tipoarchivo):boolean;
var
    pos,elemento:integer;
begin
    pos:=-1;    elemento:=pos;
    reset (archivo);
    while not (eof(archivo)) and (elemento=pos) do
    begin
        pos:=filepos(archivo);
        read (archivo,elemento);

    end;
    close(archivo);
    verificar_contenido:= (elemento=pos);

end;
...
```

Manejo de archivos de acceso aleatorio

Añadir un elemento en un archivo

```
...  
seek(inventario, filesize(inventario));  
write(inventario, elemento);  
...
```

Borrar un elemento en una posición **pos** de un archivo

```
...  
reset(inventario);  
for i:=pos+1 to filesize(inventario)-1 do  
begin  
    seek(inventario, i);  
    read(inventario, elemento);  
    seek(inventario, i-1);  
    write(inventario, elemento);  
end;  
close(inventario);  
...
```

Opciones

Deja al final el elemento repetido.

Poner un valor inválido en el elemento a borrar.

Crear una copia sin el elemento que se repite, borrar el archivo, y renombrar la copia.