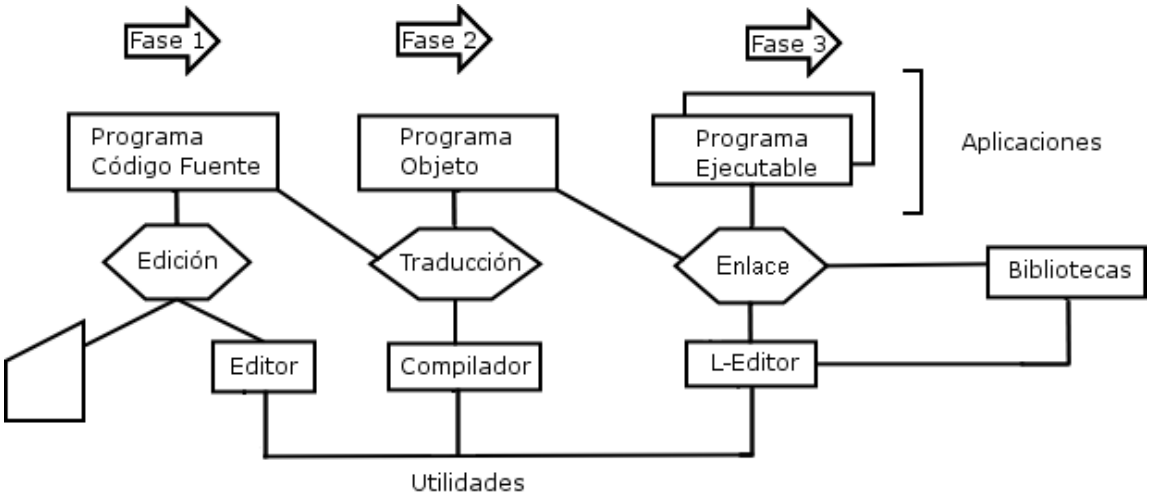
TURBO PASCAL 7

TUTORIAL

Por: **Pedro Luis Recuenco Andrés**

INDICE DEL TUTORIAL DE TURBO PASCAL 7.0

PROCESO GENERAL DE GENERACIÓN DE UN PROGRAMA



INTRODUCCIÓN DEL TUTORIAL DE TURBO PASCAL 7.0

La construcción de programas en Pascal se basa en módulos que guardan las siguientes reglas de construcción:

Program ***identificador*** **; {cabecera opcional en Turbo Pascal}**
Uses ***identificadores***
Label ***lista de etiquetas*** **; {sección de etiquetas}**
Const
definiciones de constantes
Type
declaración de tipos de datos definidos por el usuario
Var
declaración de variables
Procedure
definiciones de procedimientos
Function
definiciones de funciones
begin {cuerpo del programa}
sentencias
end.

Las cinco secciones de declaración -Label, Const, Type y Procedure y/o Function , así como la cláusula Uses y Program, no tiene que estar presentes en todos los programas. Turbo Pascal es muy flexible al momento de escribir las secciones de declaración, ya que se pueden hacer en cualquier orden (en Pascal estándar ISO si se requiere este orden). Sin embargo es conveniente seguir el orden establecido, le evitará futuros problemas.

Ejemplo:

```
Program MiPrimerPrograma; {cabecera}
Uses
Crt; {declaraciones}
Const
iva =0.10;
Type
cadena =string[35];
meses =1..12;
Var
sueldo :real;
numero :integer;
nombre :cadena;
Nmes :meses;
begin
ClrScr; {Limpia la pantalla}
Write ('Escribe tu nombre : ');
{Visualiza información en pantalla}
ReadLn(nombre);{Leer un dato del teclado}
WriteLn ('Bienvenido ', nombre);
{Visualiza información en pantalla}
Readkey; {Espera la pulsación de una tecla}
ClrScr
end.
```

TIPOS DE DATOS PREDEFINIDOS EN TURBO PASCAL.

Tipos enteros

Nombre	Rango (desde...hasta)	Tamaño (bytes)	Formato
Integer	-32.768 a 32.767	2	Entero con signo
Word	0 a 65535	2	Entero sin signo
ShortInt	-127 a 127	1	Entero corto con signo
Byte	0 a 255	1	Entero corto sin signo
LongInt	-2.147.483.648 a 2.147.483.647	4	Entero largo con signo

Tipos reales

Nombre	Rango (desde...hasta)	Tamaño (bytes)	Cifra significativas
Real	2,9E-39 a 1,7E38	6	11-12
Single	1,5E-45 a 3,4E38	4	6-7
Double	5,0E-307 a 1,7E307	8	15-16
Extended	1,9E-4932 a 1,1E493210	10	19-20
Comp	-9,2E18 a 9,2E18	8	18-19

Tipo carácter (Char).

El tipo Char depende del código de caracteres empleado por la máquina. El más utilizado es el código ASCII. Una variable o constante tipo Char puede contener un solo carácter especificado entre apóstrofes. Ejem. 'a' 'M'

Tipo Lógico (Boolean).

Pueden tomar dos valores True (verdadero) False (falso). El identificador estándar boolean define una variable de este tipo.

Tipo Cadena (String).

Una cadena (string) es una secuencia de caracteres que tiene una longitud máxima de 255 caracteres. Los caracteres que componen la cadena se delimitan con apóstrofes. Ejem.

'abcd' longitud de la cadena 4
'' cadena vacía o nula.

Una variable de cadena se define utilizando la palabra reservada String y el tamaño fisico máximo que pueda alcanzar durante la ejecución del programa.

```
Ejem.
      Const
            Lonmax   80;
      Type
            Cadena = String[Lonmax];
      Var
            Varcad: Cadena; {*almacena hasta 80 caracteres*}
            Nombre: String; {*almacena hasta 255 caracteres*}
```

TIPOS DEFINIDOS POR EL USUARIO.

Existen dos diferentes tipos de datos simples definidos por el usuario: *enumerados* y *subrago.*

Tipos enumerados. Los tipos enumerados se componen de una lista de identificadores encerrados entre paréntesis y separados por comas.

```
Ejem.
        Type
                Estaciones = (primavera, verano, otoño, invierno);
                 Colores (rojo, amarillo, verde, azul, violeta);
```

Los tipos enumerados son ordinales ya que llevan asociado cada uno un número entero, empezando por el primero, al que se le asigna el O, al segundo un 1, y así sucesivamente, por lo que no es independiente el orden de declaración. Un valor de tipo enumerado no puede pertenecer a dos declaraciones de tipo distintas y no pueden leerse desde teclado, ni escribirse en pantalla.

Tipos subrango.

Es un subconjunto de un tipo ordinal (enteros, boolean, carácter y enumerado) que se especifica indicando el primero y el último elemento del conjunto.

```
Ejem. 1
        Type
             Identificador=primerelemento..ultimoelemento;

Ejem. 2
        Type
            Fecha= 1..31;
        Var
            Dia: Fecha;
```

Los tipos enteros, carácter, booleanos, enumerados y subrango se denominan *tipos ordinales*. Un tipo ordinal representa una secuencia ordenada de valores individuales, a los que se puede aplicar los conceptos de predecesor y sucesor. En cada tipo de datos ordinales hay un primer valor y un último valor.

OPERACIONES DE ENTRADA/SALIDA.

Sentencias de salida o escritura.

Write o Writeln (NombreArchivo, Nomvarl ,Nomvar2...):

Realizan conversión de tipos de datos antes de que la salida llegue al archivo de salida. Las variables o argumentos de esta sentencia pueden ser: expresiones. constantes o variables numéricas, de carácter, de cadena o booleanas.

Formatos:

Write o Writeln (argumento)
Write o Writeln (argumcnto:m)
Write o Writeln (argumento:m:n)

Donde m es el número de caracteres que ocupará la salida ajustando la información a la derecha, y n sólo se emplea cuando el argumento es numérico real indica el número de caracteres que ocupará en la salida la parte decimal.

```
Ejem.
Mostrar la salida formateada para números decimales.

      Program formato;
      Var
            N:Real;
      Begin
            N:= 113;
            Writeln ('Formato por defecto',n);
            Writeln ('Anchura 8:1', n:8:1);
            Writeln ('Anchura 8:3', n:8:3)
      End.
```

Ejecución:

Formato por defecto _____ 3.33333333335E-01
Anchura 8:1 _____ 0.3
Anchura 8:3 _____ 0.333

Sentencia de entrada o lectura.

Read o Readln (NombreArchivo,Nomvarl ,Nomvar2..);

OPERADORES ARITMÉTICOS.

Operador	Función
-	Operador unario. Invierte el signo.
+	operador binario suma.
-	operador binario resta.
*	operador binario producto.
/	operador binario división real.
div	operador binario división entera.
mod	operador binario resto entero.

FUNCIONES PREDEFINIDAS.

Instrucción	Función
Abs(x)	Proporciona el valor absoluto de una variable numerica x.
ArcTan(x)	El arco cuya tangente es x.
Chr(x)	Devuelve el carácter ASCII de un entero entre O y 255.
Cos(x)	Proporciona el valor del coseno de x.
Exp(x)	La exponencial de x(eX).
Frac(x)	Parte decimal de x.
Int(x)	Parte entera de x.
Ln(x)	Logaritmo neperiano de x.
Odd(x)	True si x es impar, y false si es par.
Ord(x)	Ordinal de una variable tipo ordinal x.
Pred(x)	Ordinal anterior a la variable ordinal x.
Round(x)	Entero más próximo al valor x.
Succ(x)	Ordinal siguiente a la variable ordinal x.
Sin(x)	Seno de x.
Sqr(x)	Cuadrado de x.
Sqrt(x)	Raiz cuadrada de x, para x>=O.
Trunc(x)	Parte entera de x.

STRUCTURAS DE CONTROL.

Se denominan estructuras de control a aquellas que determinan qué instrucciones deben ejecutarse y qué número de veces. Existen dos tipos de estructuras de control: **alternativas** o de selección y **repetitivas** o de iteración.

ESTRUCTURAS ALTERNATIVAS.

Son aquellas que bifurcan o dirigen la ejecución de un programa hacia un grupo de sentencias u otro dependiendo del resultado de una condición. Las dos sentencias alternativas de Turbo Pascal son:

Sentencia alternativa simple IF-THEN-ELSE
Sentencia alternativa múltiple CASE-OF.

IF THEN ELSE.

```
Formato:

IF (expresión lógica o booleana) THEN
                                     Sentencial (simple o compuesta)
ELSE
    Sentencia2 (simple o compuesta);
```

```
Ejem.
IF n>O then Writeln (`Número positivo');
IF n>O then
    Writeln (`Número positivo')
ELSE
    Writeln (`Negativo o cero');
```

No puede existir un punto y coma inmediatamente antes de una palabra ELSE ya que sería interpretado como final de IF.

CASE OF

```
Formato:

CASE (expresión o variable) OF
                      (lista de constantes1):(sentencia1);
                      (lista de conslantes2):(sentencia2);
                      (lista de constantes3):(senteneia3);

                                        ...

                      (lista de constantesN):(sentenciaN);
ELSE (SENTENCIA)

...

END;
```

```
Ejem.

Program menu;
Var
    Numerodia: integer;
Begin
    Write('introduzca el ordinal de un día laborable de la semana:')
    Readín (numerodia);
    Write (`Hoy es ');
    Case numerodia Of
         1:Writeln ('Lunes');
         2:Writeln ('Martes');
         3:Writeln ('Miercoles');
         4:Writeln ('Jueves');
         5:Writeln ('Viernes');
         6:Writeln ('Sábado')
    Else
         Writeln ('¡¡¡Domingo!!! No es día laborable');
End.
```

ESTRUCTURAS REPETITIVAS.

Son aquellas que crean un bucle (repetición continua de un conjunto de instrucciones) en la ejecución de un programa respecto de un grupo de sentencias en función de una condición.
Las tres sentencias repetitivas de Turbo Pascal son:

SENTENCIA WHILE

Indica al ordenador que se ejecuten una o más sentencias mientras se cumpla una determinada condición. La condición viene determinada por una variable o expresión booleana.

```
Formato:

WHILE condición DO
                  BEGIN
                          (sentencia1);
                              ...
                          (sentenciaN);
                  END;

WHILE condición DO
                  (sentencia);
```

Esta sentencia comprueba inicialmente si la condición es verdadera. Si la condición es verdadera se ejecutan las sentencias mientras la condición de su enunciado sea verdadera y finaliza cuando la condición es falsa. Dado que la condición puede ser falsa inicialmente, es decir antes de comenzar el bucle, habrá casos en que el bucle no se ejecute.

Características del bucle WHIILE:

Se ejecuta mientras la condición sea verdadera, y dentro del bucle debe existir, por lo menos, una sentencia que modifique el valor de la variable o expresión, de lo contrario se puede producir una situación de bucle infinito. Si la expresión lógica es falsa al comenzar el bucle, éste no se realizará.

Ejemplo.
Escribir los N primeros números naturales, donde N es un valor introducido por el usuario.

```
Program escribeenteros;
Var
      N,contador: integer;
Begin
      Write ('Introduzca numero maximo de enteros: ');
      Readln (N);
      Contador:=1;
      While contador<=N do
                        Begin
                                Write (contador:5);
                                Contador:=contador+1;
                        End;
      Writeln;
      Writeln ('Fin de programa. Contador = ',contador);
End.
```

SENTENCIA REPEAT UNTIL

Ejecuta las sentencias comprendidas entre las palabras reservadas REPEAT y UNTIL hasta que la expresión o variable sea verdadera.

```
Formato:

REPEAT
      begin
           (Sentencia);
           (Sentencia);
              ...
      end;

UNTIL condición;
```

Características del bucle REPEAT:

Se ejecutan siempre una vez, por lo menos, y la terminación del bucle se produce cuando el valor de la expresión lógica o condición de salida es verdadera. Se ejecuta hasta que la expresión es verdadera, es decir, se ejecuta mientras la expresión sea falsa.

```
Ejemplo.
El mismo que con la sentencia WHILE.

 Program escribeenteros;
 Var
    N,contador:integer;
 Begin
     Write ('Introduzca número máximo de enteros: ');
     Readin (N);
     Contador:= O;
     Repeat
           Contador:=contador+1;
           Write (contador:5)
     Until contador = N;
     Writeln ('Fin de programa. Contador = ',contador)
 End.
```

SENTENCIA FOR

Repite la ejecución de una o varias sentencias un número fijo de veces. previamente establecido. Necesita una variable de control del bucle que es necesariamente de tipo ordinal, ya que el bucle se ejecuta mientras la variable de control toma una serie consecutiva de valores de tipo ordinal, comprendidos entre dos valores extremos (inferior y superior).

```
Formato ascendente:


FOR variablecontrol:=valorinicial TO valorfinal DO
                                                (sentencia);

Formato descendente:

FOR variablecontrol:=valorincial DOWNTO valorfinal DO
                                                  (sentencia);

donde (sentencia) puede ser una sentencia simple o compuesta.
```

```
Ejemplo:
El mismo que con la sentencia WHILE.

Program escribeenteros;
Mar
      N,contador: integer;
Begin
      Write ('Introduzca numero maximo de enteros: ');
      Readln (N);
      For contador:=1 to n do
                              Write (contador:5);
      Writeln
End.
```

Características del bucle FOR:

Aunque a primera vista pueda resultar más atractivo FOR, existen limitaciones en su aplicación ya que en el bucle FOR siempre se incrementa o decrementa (de uno en uno) los valores de la variable de control de bucle y no de dos en dos o de tres en tres, o con valores fraccionarios.

El número de iteraciones de un bucle FOR siempre es fijo y se conoce de antemano:
Valor final - Valor inicial +1.

CUÁNDO UTILIZAR WHILE/REPEAT/FOR?

- Utilizar la sentencia o estructura FOR cuando se conozca el número de iteraciones, y siempre que la variable de control de bucle sea de tipo ordinal.
- Utilizar la estructura REPEAT-UNTIL cuando el bucle se realice por lo menos una vez.
- En todos los demás casos utilizar la sentencia WHILE.

PROCEDIMIENTOS Y FUNCIONES.

Pascal ofrece dos herramientas básicas para realizar programación descendente: los procedimientos (procedure) y las funciones (function), a los que nos referiremos genéricamente con el término de subprogramas. Turbo pascal incorpora además el concepto de unidad (unit), que permite aprovechar módulos independientes ya compilados.

Los parámetros

Los parámetros son canales de comunicación para pasar datos ente programas y subprogramas en ambos sentidos. Los parámetros van asociados a variables. constantes, expresiones, etc., y por tanto, se indican mediante los correspondientes identificadores o expresiones. Los parámetros que se utilizan en la llamada o invocación al subprograma se denominan parámetros actuales, reales o argumentos, y son los que entregan la información al subprograma. Los parámetros que la reciben en el subprograma se denominan parámetros formales o ficticios y se declaran en la cabecera del subprograma.

En una llamada a un subprograma tiene que verificarse que:

1. El número de parámetros formales debe ser igual al de actuales.
2. Los parámetros que ocupen el mismo orden en cada una de las Listas deben ser compatibles en tipo.

Estructura, declaración y empleo de un procedimiento

Se declaran inmediatamente después de las variables del programa principal, teniendo la precaución de que si un subprograma referencia o llama a otro, el referenciado debe declararse primero.

Declaración de procedimientos.

```
Cabecera                procedure nombreproced (lista de parámetros);

Declaraciones           const
Locales

                        Type...
                        Var...
                        <declaración de otros procedimientos y funciones>

Cuerpo                  begin

                        end; (*obseiwar; final de proeedimiento*)
```

Declaración de parámetros formales.

Se declaran encerrados entre paréntesis, indicando el identificador y el tipo correspondiente asociado a cada uno, separados por ':', y terminando en ';'. La palabra reservada VAR precediendo a un identificador de parámetro formal indica al compilador que el paso del parámetro es pro VARiable. Su ausencia u omisión indica que el paso de parámetro se realiza por VALOR.

```
Ejem.

Procedure Identificador (PF1 :tipol ;PF2:típo2; var PW:tipo3);
PFl y PF2 se pasan por valor.
PF3 se pasa por variable.
```

Llamada a un procedimiento.

Se realiza desde el programa principal indicando el identificador del procedimiento seguido de la lista de parámetros actuales encerrados entre paréntesis y separados por comas.

```
Ejem.

Identificador (PAl ,PA2,PA3)

Ejemplo.
Procedimiento para intercambiar los valores de dos variables.

Procedure intercambio (var pfl ,p12:integer);
Var
    Aux:integer; (variable local uso exclusivo en procedimiento}
Begin
      Aux:=pfl;
      Pfl:pf2;
      Pfl:      aux
End;

{La llamada a este procedimiento se haría ...}
{... por ejemplo desde el siguiente programa:}

Program Uno;
Uses crt;
Var
      Entero 1 ,entero2 : integer;

Procedure intercambio (var pfl ,p12:integer);
 ...

begin
      clrscr; {*borrado de pantalla*}
      Write ('introduzca 2 variables enteras: ');
      Readln(entero1,entero2);
      Writeln ('valores de las variables antes de la llamada');
      Writeln ('Entero 1 = ',enterol,'entero 2 = ',entero2);
      intercambio (enterol,entero2); {llamada al procedimiento}
      Writeln ('Valor de las variables después de la llamada');
      Writeln ('entero 1 = ',enterol,'entero 2 = ',entero2);
end;
```

Funciones o procedimientos?

Deben utilizarse funciones cuando solo tenga que devolverse un solo valor simple al programa llamador. En todos los demás casos utilizaremos: `procedimietos.`

LISTAS Y TABLAS (Arrays).

Concepto y clasificación de estructuras

Las estructuras se clasifican de acuerdo a varios criterios.

- Respecto al número de componentes, las estructuras se clasifican en estáticas (el número de componentes es fijo) y dinámicas (el número de componentes varía durante la ejecución del programa ya que se pueden crear y destruir las variables durante el desarrollo del mismo.
- Respecto al tipo de componentes las estructuras suelen formarse o bien por combinación de datos de distinto tipo (registros) o bien por la repetición de datos del mismo tipo (arrays,conjuntos archivos, listas, árboles, etc.). Dentro de éstas el acceso puede hacerse por posicion, ya sea de forma secuencial o bien directa, o bien por contenido o clave.

La estructura Array.

Es una estructura homogénea de datos de tamaño constante accediendo a cada uno de sus elementos mediante un identificador comun y uno o varios índices.

- Todos los elementos del array son del mismo tipo.
- El número de ellos no varía durante la ejecución del programa.
- Accedemos a un elemento de la estructura mediante un identificador común, el nombre del array, y con el valor que toman uno o varios indices. Al número de indices necesarios para designar un elemento del array se le denomina dimensión del array.
- El número máximo de valores posibles que puede tomar cada indice se denomina rango de esa dimensión o indice. Los valores han de ser consecutivos, por lo que el indice ha de ser de un tipo ordinaL

Declaración de tipos y variables array

La declaración más general de un array es la siguiente:

```
Type
   Rango1 = tipoordinal1;
   Rango2 = tipoordinal2;
            ...
   rangon = tipoordinaln;
   tipobase = (*cualquier predefinido o definido por el usuario*)
   tipoarray = array [rango1,rango2,...,rangon] of tipobase,
```

Ejemplo: Array para almacenar las notas correspondiente a todos los alumnos de un colegio. Suponiendo lo siguiente:

Numero de cursos 5
Grupos por curso 3
Numero de evaluaciones 3
Numero de asignaturas 6
Numero de alumnos por curso 20

```
Const
  Numcurso=5;
  Numasig=6;
  Numalum=20;

Type
  Cursos=1.. nurneurso;
  Grupos='A'..'C';
  Eval=(primera,segunda,tercera);
  Asign=1.. numasin;
  Alum=1. .numalum;
  Tiponotas=array[cursos,grupos,eval,asign,alum] of real;

Var
  Notas:tiponotas;
  Curso:cursos;
  Grupo:grupos;
  Evaluacion:eval;
  Materia:asign;
  Alumno:alum;
```

Con los elementos de un array podemos realizar las mismas operaciones que el tipo base al que pertenecen.

STRING: Cadenas de caracteres

Las cadenas de caracteres son arrays especiales.
Una cadena de caracteres (string) consiste en una serie o secuencia de caracteres cuyo número (longitud) puede estar entre 0 y 255.
Se puede definir la longitud de la cadena poniendo string [n] donde n está entre 0 y 255

Longitud fisica: corresponde al maximo numero de caracteres que puede almacenar.
Longitud lógica: corresponde al numero de caracteres que tiene en un instante determinado.

OPERACIONES Y TRATAMIENTO DE CADENAS

Operaciones con cadenas.

Las dos operaciones basicas son *comparación y concatenación*.

Operador de concatenación (+)
Se utiliza para reunir varias cadenas en una sola

Ejemplo

```
Cad1:='esto es un ejemplo';
Cad2:='de concatenacion de cadenas';
Cadr:=cad1+cad2;
Write (cadr);
```

Se visualizaría *esto es un ejemplo de concatenacion de cadenas*

La funcion **concat** realiza la misma funcion que el operador de concatenacion. La sintáxis es:

Function concat (cad1,cad2,...:string): string;

La cadena vacia o nula se representa con dos caracteres apóstrofes seguidos ' '. El acceso a los elementos de una cadena individualmente se hace como si fuera un array.

Ejemplo.

```
Cad1:='ejemplo';
```

Para referirnos al primer elemento pondriamos cad1[1] que seria la letra e.

La funcion **Length** proporciona la longitud logica de una cadena de caracteres.

Ejemplo.

```
Longitud:=length(cad1);
```

La variable longitud tomaria el valor 7.

Funciones de tratamiento de cadenas.

Instrucción	Función
COPY	Extrae una subcadena de caracteres de otra cadena de caracteres Copy (cadl,po,num) Po: primera posición del caracter a extraer. Num: número de caracteres que se extraen. Po y Num deben ser enteros
POS	Determina si una cadena es subcadena de otra, en caso afirmativo devuelve la posición donde comienza la subcadena, en caso negativo devolvería cero. Pos (subcadena,cadena)
DELETE	Suprime el numero de caracteres que le digamos de una cadena a partir de la posición que le indiquemos Delete (Cad,Po,Num)
INSERT	Inserta una cadena de caracteres en otra a partir de una posicion dada. Insert (subcadena,destino,posicion) Subcadena: Cadena a insertar Destino: Cadena donde se va a insertar Posicion: Lugar a partir del cual se va a insertar
UPCASE	Devuelve el caracter mayúscula Upcase(x) Donde x es una variable de tipo char.
STR	Convierte un valor numerico a la correspondiente cadena de caracteres que lo representa. Str(valor,cad)
VAL	Procedimiento inverso a Str, es decir, devuelve el valor numerico de una cadena. Val(cad,variable,codigo) Cad: la cadena a convertir en valor numerico. Variable: el numero que se obtenga. Codigo: cero si se ha podido convertir.

MÉTODOS DE ORDENACIÓN

Método del Quickshort

```
Program Quicksort;
uses crt;
type
    vector=array [1..10] of integer;
const
     lista:vector=(8,5,6,3,1,4,2,7,10,9);
var
   k:integer;
   longitud:integer;

procedure rapido (var a:vector;n:integer);
    procedure partir (primero,ultimo :integer);
    var
       i,j,central:integer;
       procedure intercambiar (var m,n:integer);
       var
          aux:integer;
       begin
            aux:=m;
            M:=n;
            N:=aux;
       end;
    begin
         i:=primero;
         j:=ultimo;
         central:=a[(primero+ultimo) div 2];
         repeat
               while a[i]central do
                                   j:=j-1;
               if i<=j then
                       begin
                            intercambiar (a[i],a[j]);
                            i:=i+1;
                            j:=j-1;
                       end;
         until i>j;
         if primero<j then
                     partir(primero,j);
         if i<ultimo then
                    partir(i,ultimo);
    end;
begin
     partir (1,n);
end;
begin
     clrscr;
     k:=0;
     write ('Este es el vector original: ');
     repeat
           begin
                write (lista[k],' ');
                k:=k+1;
           end;
     until k=11;
     writeln;
     rapido (lista,10);
     k:=0;
     write ('Este es el vector ordenado: ');
     repeat
           begin
                write (lista[k],' ');
                k:=k+1;
           end;
     until k=11;
     readln;
end.
```

REGISTROS Y ARCHIVOS

Registros

Un registro es una estructura heterogénea de datos, denominados campos y a los que accedemos por nombre. Al igual que cualquier otro dato, el tipo registro (Record) antes de poder se utilizado debe ser declarado en la sección de tipos.

La única operación (a parte de la lectura) que se puede realizar con una variable registro como tal es la asignación, es decir, se pueden copiar todos los campos de una variable registro a otra variable registro del mismo tipo. Además un registro puede ser pasado como parámetro a una función o procedimiento.

Archivos

Un archivo es una estructura homogénea de datos consistente en una secuencia de elementos llamados registros, todos del mismo tipo, ya sea simple o estructurado. Un archivo se almacena en un dispositivo auxiliar (discos, cintas, etc), de forma que los datos obtenidos antes, durante y después del procesamiento de los datos, no se pierden. Para declarar una variable archivo es necesario definir previamente la naturaleza de sus registros.

```
Ejemplo:

  Type
        Tiporegistro = record
                        Campo 1 :tipo1;

                          .
                          .

                        Campo N : tipoN;
         End;

         Tipoarchivo = file of tiporegistro.
  Var

    Archivo1 : tipoarchivo;
    Registro1 : tiporegistro;
```

Operaciones básicas con archivos:

Instrucción	Operación
ASSIGN	Este procedimiento asigna un archivo lógico con su archivo fisico correspondiente. Después de la asignación, cualquier operación sobre la variable archivo afectará al archivo Dos correspondiente. Assign (Vararch,nomarch);
RESET	Procedimiento que abre un archivo para lectura posicionando el puntero de lectura del archivo en el primer elemento del archivo, y poniendo la variable booleana EOF asociada al archivo a False, o en la marca de fin de archivo sí el archivo esta vacio, en cuyo caso la variable EOF toma el valor True. No se puede modificar el contenido de ningún registro. RESET (nomvararchivo);
IORESULT	Función que devuelve el número del tipo de error cometido en el tratamiento de archivos. Si no hay ningún error devuelve O.
REWRITE	El procedimiento Rewrite abre un archivo para escritura destruyendo el contenido del arhivo si este ya existe. No es posible ver datos de un archivo que está abierto con este procedimiento, ya que borra los datos existentes.
READ	Este procedimiento se utiliza para introducir el contenido de un registro del archivo en una variable de memoria definida del mismo tipo de dato que el registro leido. READ (nomvararchivo,nomvarreg);

EOF	En la lectura del último registro el salto del puntero posiciona éste sobre la marca de fin de archivo, colocando la función lógica EOF "fin de archivo" asociada a cada archivo a verdadero.
	
	EOF (vararchivo)
	
	Ejemplo:
	
	While not eof(pruebas) do
	Begin
	Read (pruebas,info);
	Write (info);
	End;

WRITE	El procedimiento write escribe en un registro del archivo el contenido de una variable de memoria definida del mismo tipo. WRITE (nomvararchivo,nomvarreg);

Archivos de acceso directo

Están formados por registros del mismo formato y longitud por lo que permiten el acceso a un registro especifico mediante un número asociado al mismo, que se denomina su número de registro lógico. El número asociado es de tipo longint y se asigna al primer registro lógico el valor O. Para que un archivo pueda ser tratado por posicionamiento o acceso directo debe residir obligatoriamente en un dispositivo de almacenamiento de este tipo.

La declaración de un archivo de acceso directo es idéntica a la de otros archivos y sólo se distingue de ellos por las funciones de posicionamiento en un registro. El contenido de un archivo directo se almacena en disco bajo forma binaria comprimida y no es visualizable directamente en pantalla, como los archivos de texto, con la orden TYPE de DOS o con editores.

Las principales operaciones de archivos de acceso directo que se usan en Pascal son:

RESET Abrir archivo existente.

REWRITE Abrir un archivo nuevo.
Son dos procedimientos para abrir el archivo de acceso directo, ya sea con un procedimiento u otro, el archivo se abrirá para lectura y escritura.

FILESIZE Tamaño del archivo en formato longint, indica el número de registros almacenados. Si el fichero está vacio devuelve el valor O.

SEEK Permite seleccionar un registro específico del archivo por su número de registro, para su uso en una operación de lectura o escritura.

READ Lectura del registro actual.

WRITE Escritura sobre el registro actual.

CLOSE Cerrar el archivo.

Tratamiento de archivos desde Turbo Pascal

El Turbo Pascal permite manipular archivos y directorios en disco de modo similar al sistema operativo Ms-DOS. Pudiendose realizar las siguientes operaciones con archivos y directorios:

```
Erase(nomvararchivo)
Rename(nomvararchivo,'nombrenuevoarchivo')
Chdir (directorio)
Mkdir (directorio)
Rmdir (directorio)
Getdir(unidad,camino)

  Unidad = 0 unidad de arranque
  Unidad = 1 A:
  Unidad = 2 B:
  Unidad = 3 C:

  Camino contiene el directorio actual.
```

`Getdir` obtiene la unidad y el camino del directorio actual de una unidad y lo almacena en las variables unidad, de tipo **`byte`** y cambio de tipo String.