MANUAL DE INTRODUCCIÓN AL PASCAL

TEMA I INTRODUCCIÓN

La programación es el arte de solucionar problemas mediante un computador, utilizando programas. Las fases de la programación se pueden clasificar en tres fases:

1. **Planteamiento del problema.**
2. **Algoritmo.**
3. **Implementación en un lenguaje de programación.**

Los lenguajes de programación son reglas y símbolos para implementar algoritmos en las computadoras.

1.1TIPOS DE LENGUAJES DE PROGRAMACIÓN.

1. **Lenguajes de alto nivel** : Son lenguajes cuyas reglas son parecidas a la sintaxis humana. (Por ejemplo el PASCAL, BASIC, C, LISP, JAVA etc)

2. **Lenguajes de bajo nivel**:Son órdenes comprensibles por las máquinas (ENSAMBLADOR)

Los lenguajes pueden ser COMPILADOS o INTERPRETADOS .
En los lenguajes COMPILADOS, se produce la traducción de todo el programa a código máquina y generan un ejecutable.

Por el contrario, en los lenguajes INTERPRETADOS, la traducción del programa a código máquina se hace línea a línea, por lo que son más lentos.

Un caso especial es JAVA que es un lenguaje híbrido entre compilado e interpretado.

El PASCAL es un lenguaje que nace en 1968 de la mano de Niklaus Wirth y en 1983 se estandariza dando paso al ISO PASCAL.

Es un lenguaje estructurado, fácil de aprender y que actualmente hay compiladores gratuitos para muchas plataformas (Win,Linux,Commodore etc) se puede descargar desde http://www.freepascal.org/ y se puede obtener toda la información relativa al compilador.

FreePascal nace en 1996 bajo licencia GNU/GPL y es un buen comienzo para desarrollar programas en entornos visuales como Delphi y Kilyx (Win/Linux)
http://www.borland.com/

1.2 PROGRAMAS.

Un programa es un conjunto de reglas y símbolos escritos en un editor de texto con extensión generalmente .pas (Borland pascal) o .pp (freePascal) que al compilarlos obtendremos un ejecutable, (en DOS se obtiene un *.EXE)

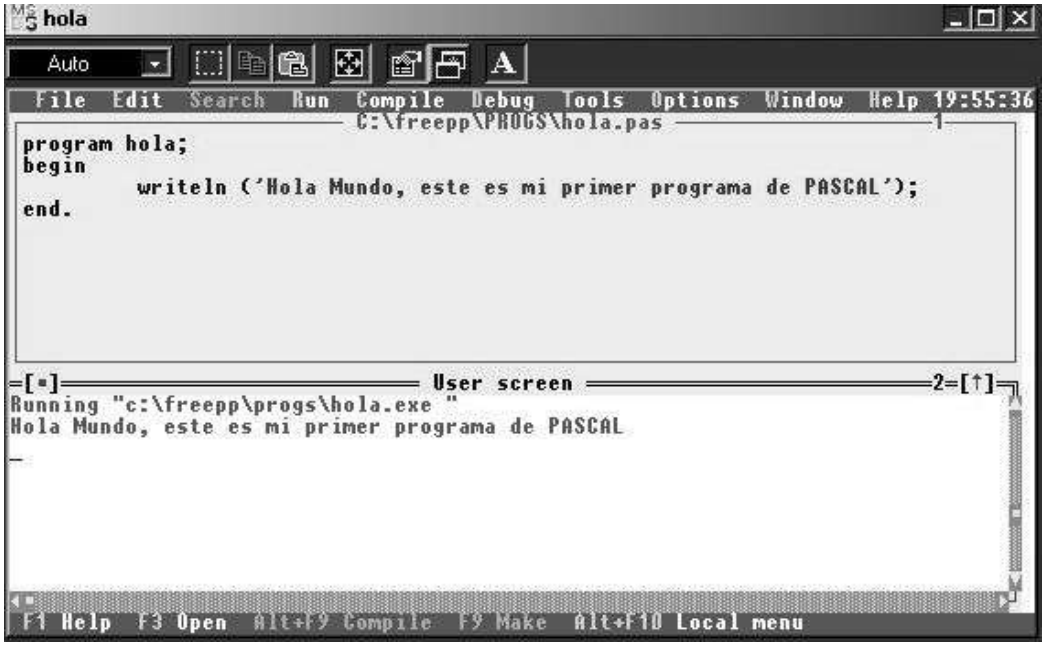
Este fichero de texto se compone de :

Componentes de programa	***Ejemplo***
CABECERA	PROGRAM ejemplo;
DECLARACIÓN DE CONSTANTES	CONST numero = 234;
DECLARACIÓN DE VARIABLES	VAR años : real;
BLOQUE DE PROGRAMA	BEGIN bloque de programa; END.

Veamos un ejemplo de un programa para calcular el área de un triángulo

```
PROGRAM AREATRI;
VAR
   base,altura,area : real;
BEGIN
            base:=2;
            altura:=4;
            area:=0.5*base*altura;
            WRITELN(area);
END.
```

El programa anterior calcula el área de un triángulo de base 2 y altura 4 utilizando la fórmula:
área=(base x altura) / 2
Veamos otro ejemplo:



En la imagen anterior tenemos la ejecución del típico programa 'Hola Mundo' en el entorno de freepascal para Win32.

1.3 CONSTANTES.

Una constante es un valor fijo que no varia durante la ejecución del programa.
Así el número de los días de la semana, el valor PI, la conversión de kilómetros en metros, y otros más son valores constantes.
La definición de constantes en PASCAL se realiza de la siguiente forma:

```
CONST
        max=100;
        pi = 3.141592;
        curso = 'electronica_1_B';
```

Las cadenas de caracteres siempre van entre comillas simples, si los valores numéricos van precedidos de $ serán valores Hexadecimales, si empiezan por % son valores binarios.

```
CONST
       basedecimal = 10;
       base16 = $FF;
       base2 = %100101;
```

1.4 VARIABLES

Una variable, al igual que una constante se almacena en una posición de memoria, pero al contrario de las constantes cuyo valor no varia, el valor de una variable cambiará durante el transcurso de la ejecución de un programa.

La forma de definir las variables en PASCAL es como sigue:

```
VAR
    contador : Integer;
    nombre : String;
```

Cada variable es de un tipo determinado y hay que indicar el tipo en el bloque de programa VAR, si esto no ocurre, el compilador dará un error.

Existen tres tipos básicos de variables, las variables numéricas, las alfanuméricas y las booleanas.

Las variables numéricas son enteros y reales.
Las variables alfanuméricas son caracteres.
El tipo booleano pueden tener 2 valores (True y False)

Tipos numéricos.
Enteros

Tipo	***Rango***
Byte	0-255
Shortint	-128..127
Integer (mas utilizados)	-32768..32767
Word	0..65535
LongInt	-214783648.. 214783647
Cardinal	0..4294967296
Int64	-9223372036854775805.. 922337203685477580

Reales

Tipo	***Rango***
Single	1.5x10^-45..3.8^38
Real (más utilizados)	5x10^-324..1.7^308
Double	5.10^-328..1.7^308
Extended	1.9x10^-4951..1.1x10^4932
Comp	-2x10^64..2x10^63

Tipos alfanuméricos

Tipo	***Rango***
Char	1 Carácter
ShortString	255 Caracteres
String(n)	N caracteres

Tipo	*Rango*
AnsiString	Ilimitado a la memória
PChar	Null (Nulo)

Tipos Booleanos

Tipo	*Rango*
Boolean	True/false
ByteBool	Api Windows
WordBool	Api Windows
LongBool	Api Windows

Tanto los nombres de constantes como de variables no pueden ser palabras reservadas de Pascal.

Palabras reservadas Pascal

```
and             array           begin           case
const           div
do              downto          else            end
for             function
goto            if              in              label
mod             not
of              or              procedure       record
repeat          set
then            to              type            until
var             while
xor
```

Tipos Pascal

```
boolean                 byte            char
double                  file            integer
longint                 pointer         procedure
real                    string          text
word
```

Funciones y procedimientos Pascal

```
abs             append                  arctan
assign          blockread
blockwrite      close                   CloseGraph
clrscr          copy
cos             delay                   dispose
eof             exp
filepos         filesize                gettime
getdate         gotoxy
int             keypressed              length
ln              new
open            ord                     OutTextXY
read            readkey
```

```
readln          release                 reset
rewrite         round
seek            sin                     sqrt
str             trunc
upcase          val                     write
writeln
```

Colores

```
BLACK           BLUE            BROWN           CYAN
DARKGRAY        GREEN           LIGHTBLUE       LIGHTCYAN
LIGHTGRAY       LIGHTGREEN      LIGHTMAGENTA    LIGHTRED
MAGENTA         RED             WHITE           YELLOW
```

1.5 OPERADORES

Un operador es un símbolo que representa un operador matemático.

+Suma - Resta * Producto / División ** Potencia DIV División entera

mod Resto de división entera.

Ejemplos:

```
Program operación;
begin
        Writeln (3+4*5);
        Writeln ((15+3)*2);
        Writeln ((12/16)*(8/5));
End.
```

1.6OPERCIONES DE ENTRADA Y SALIDA

En los programas, muchas veces necesitamos interactuar con ellos, pasarles valores y que se nos presenten resultados, estos operaciones son de entrada (IN) y de salida (OUT)

SENTENCIAS DE SALIDA

Write y Writeln: Muestran en pantalla o escriben en ficheros los mensajes y los resultados de la computación del programa.

Ej: Write ('Hola');

Writeln (' Programa de ejemplo');

Write (VARIABLESALIDA);

Formatos de salida:

Podemos formatear la salida de forma que indiquen los valores enteros y decimales de los resultados de operaciones o de constantes y variables.

Ej

Program format;

```
Var
        num:real;
Begin
        num:=1237.67847362;
        writeln ('Sin formato = ', num);
        writeln ('formato 8:1 = ', num:8:1);
        writeln ('formato 4:5 = ', num:4:5);
End.
```

SENTENCIAS DE ENTRADA.

Para leer datos introducidos por teclado o por ficheros utilizamos READ y READLN

COMENTARIOS EN LOS PROGRAMAS

Muchas veces podemos introducir comentarios entre el código de programa, para aclarar el funcionamiento y poder mejorar este en las sucesivas versiones.

{ Comentarios entre llaves }

(* Comentarios entre parentesis *)

// Comentarios hasta final de línea.

EJ Realizar un programa para calcular el área de un triángulo, leyendo la base y la altura y aplicando la fórmula área = (Base x Altura) /2

```
Program areatri;
(* Calcula el área de un triángulo utilizando la fórmula
base x altura / 2 *)

var
   base,altura : Real;
Begin
 Writeln ('Introduzca la base');
 Read (base);
 Writeln ('Introduzca altura');
 Read (altura);
 (* Se han introducido las variables *)
 Writeln ('Resultado =');
 Write ((base * altura * 0.5):8:3);
 Writeln (' en metros 2');
 // Se escribe el resultado
End.
```

Este programa calculará en metros cuadrados los valores introducidos cuando se pida la base y la altura.

1.7 TRABAJANDO CON CONSTANTES Y VARIABLES

Pascal, igual que otros lenguajes de programación, controla estrictamente las comprobaciones de los tipos de datos, es decir, no podemos mezclar tipos de variables o constantes que no tengan el mismo tipo.

```
Program errores;
VAR
        (* asignamos variables *)
        caracteres : string;
        numero : integer;
BEGIN
        caracteres := 10; //asignación errónea
        numero := 'a' ; // Numero debe ser entero;
        caracteres := numero ; // Tipos incompatibles
END.
```

Si hay errores de asignación, o necesitamos utilizar valores de tipos diferentes podemos recurrir a la transformación de tipos.

Esto se llama Typecasting y podemos amoldar el tamaño de las variables al trabajar con enteros o convertir caracteres a su valor numérico ASCII

EJEMPLO

```
PROGRAM formato;
VAR
        carácter : char;
BEGIN
        Write ('carácter');
        Readln ( carácter ) ;
        Writeln ('Valor ASCII = ' , integer (carácter));
END.
```

El typecasting (manejo de tipos) nos permite poder sumar strings como si fueran valores numéricos.

```
BEGIN
        Writeln ('El curso de elctrónica es ' + '  B1');
END.
```

FUNCIONES ESPECIALES

IntToStr Toma un entero y devuelve un string

```
program entstr;
uses
         Sysutils; //función externa donde está IntToStr
var

   cadena : string;
   ans : Integer;

Begin
        Writeln ('Cuantos años tienes?');
        Readln (ans);
        cadena := IntToStr(ans);
        Writeln(cadena);
        Writeln (ans);
End.
```

StrToInt Toma una cadena y devuelve un entero

```
PROGRAM SRINT;
uses Sysutils;
var
   a,b,total : string;
begin
           a:='100';
           b:='200';
           total:=IntToStr (StrToInt(a)+StrToInt (b));
           writeln (total);
end.
```

En el programa anterior las dos cadenas a y b las pasamos a un tipo entero, las sumamos y volvemos a convertirlos en cadenas

FloatToStr Toma un real y lo convierte en una cadena.

Ejercicios:

1.1- Realizar un programa que sume 3 números

1.2- Implementar un programa en Pascal que sume,reste,multiplique y divida dos números reales.

(Formatear las salidas)

1.3-Realizar un programa que calcule el área de un rectángulo con numeros enteros

1.4-Calcular el área y la longitud de una circunferencia usando reales

1.5-Calcular la velocidad de un automóvil v=e/t en Km/h si los valore e son en metros y t en segundos

1.6-Calcular el volumen de una esfera si V=4/3(pi+r^3)

1.7-Calcula z=(3x+6y+2)

1.8-Calcula z=x^2+y^2

1.9-Calcula y=(x+6)/2

1.10-Calcular una ecuación de segundo grado y=ax^2+bx+c

1.11-Calcular la hipotenusa de un triangulo si c^2=a^2+b^2

1.12-Realizar un programa que pase de Km a m,cm,mm

1.13-Realizar un programa que convierta de Faradios a milifaradios,microfaradios,nanofaradios y picofaradios

1.14-Realizar un programa que calcule la ley de Ohm Introduciendo el valor de V y de I

1.15-Realizar un programa que calcule la ley de Ohm introduciendo R y V

1.16-Implementar el cálculo de la resistencia total de un circuito serie de 3 resistencias (Debe de dibujar con caracteres ascii, el circuito)

1.17-Implementar el cálculo de la resistencia total de un circuito paralelo de 3 resistencias

1.18-Realizar un programa que realize la tabla del código de colores de las resistencias

1.19-Realizar un programa que convierta de decimal a binario

1.20-Realizar un progarma que convierta de binario a decimal.

TEMA I I ESTRUCTURAS DE CONTROL.

Se llaman estructuras de control , aquellas que realizan el control de un programa de forma que indican que instrucciones deben ejecutarse y cuantas veces tienen que repetirse.
Se utilizan para alterar la ejecución lineal de un programa.

IF ... THEN ...ELSE

Sirve para comprobar si es cierta o no una condición de programa.

IF (expresión lógica o booleana) THEN (Sentencias1)
ELSE (Sentencias2)
Ej:

```
Program positivo;
var
   n:real;
Begin
            Writeln ('Introduce un numero entero');
            Readln(n);
            IF n>0 THEN Writeln ('Numero positivo')//No lleva ; ya que if then
else es toda la sentencia
            ELSE Writeln ('Numero negativo o cero');
End.
```

Se puede utilizar tambien la estructura IF ... THEN

Ej Cálculo de la longitud de una circunferencia

```
Program longcir; //Long=2*pi*0.5
var
   r : real;
Begin
            Writeln ('Introduzca valor del radio ');
            Readln (r);
            IF r>=1 THEN //Si r>=1 realizamos el cálculo
                        begin
                                    Writeln ('Resultado :');
                                    Writeln (2*3.141592*r:6:3, ' metros');
                        end
            ELSE //Si no es mayor o igual a 1 mostramos mensaje
                         begin
                                     Writeln ('El radio debe ser >=1');
                                     Writeln ('Vuelva a intertarlo');
                        end;
End.
```

Los operadores lógicos booleanos son:

NOT	Negación
AND	Producto
OR	Suma
XOR	Or Exclusiva

Los operadores de relacción son:

= Igual
<> Distinto
> Mayor que
< Menor que
<= Menor o igual
>= Mayor o igual

```
//Programa para detectar si un numero es positivo,negativo o cero
Program positivo;
uses crt;// Para usar Clrscr (limpia pantalla)
var
   num : integer;
Begin
            Clrscr;
            Writeln ('Introduzca un numero entero');
            Readln (num);
            If num > 0 then writeln ('El numero es positivo');
           If num < 0 then writeln ('El numero es negativo');
            If num = 0 then writeln ('El numero es cero');
end.
```

2.1Implementar un programa que detecte si los tres numeros introducidos están en orden creciente.
2.2Escribir un programa que detecte si el número introducido es par o impar.
2.3Escribir un programa que pida una contraseña y detecte si es válida o no.
2.4Realizar un programa que lea 10 números, los multiplique y indique los múltiplos de 2 y de 3.

ESTRUCTURA FOR

Muchas veces, nos interesa que una sentencia o un grupo de ellas, se ejecuten un número determinado de veces, se utilizará un contador para determinar cuantas veces debe repetirse la secuencia.

Este bucle podrá ser ascendente o descendente.

Ascendente: FOR contador := Valorinicial TO Valorfinal DO
Descendente: FOR contador := Valorfinal DOWNTO Valorinicial DO

Siempre se utiliza una variable contador.

```
Program vsofor;
Uses crt; //Unit Crt para usar ClrScr
Var
           contador : Integer;
Begin
            ClrScr; //Borramos la pantalla
           For contador := 1 TO 5 DO
           Begin
                        Writeln('Valor de contador =',contador);
           End;
End.
```

Realizar un programa que calcule la suma de n números enteros siempre que los números sean >=0

```
Program Sumafor;
Uses crt;
Var
          suma,contador,numero,valor : Integer;
Begin
            suma := 0;
            Clrscr;//Borramos la pantalla
            Writeln ('Introduzca la cantidad de valores');
            Readln (numero);
            For contador := 1 to numero DO
                      Begin
                                  Writeln ('Introduzca valor',contador);
                                  Readln (valor);
                                  If valor >= 0 then
                                            Begin
                                            suma := suma + valor;
                                            End
                                  Else
                                            Begin
                                            Writeln ('El valor debe ser mayor o
igual que 0');
                                            Exit;//Salimos del programa si el
valor no cumple la condicion
                                            End;
                        End;
          Writeln ('El valor de la suma es =', suma);
End.
```

2.5Realizar un programa que incremente un contador cada vez que pulsemos la tecla 'INTRO',si pulsamos cualquier otra tecla mostrará el resultado del contador y saldrá del programa.

2.6Implementar un programa que utilice el bucle FOR y muestre en pantalla los números del 1 al 100 y indique qué valores son divisibles por 5

2.7Realizar un programa que genere la tabla de multiplicar de un número introducido por el teclado.

EL BUCLE WHILE .. DO

Este bucle se repetirá siempre hasta que la condición que lo ha generado, sea falsa.

WHILE Expresión booleana DO Sentencias

Ej

```
Program whiledo;
uses crt;
var
          cadena:string;
Begin
           ClrScr;
           cadena := 'a'; // Asignamos el valor a para empezar el bucle
           While cadena <> '.' Do // Con . finalizamos el bucle
                       Begin
                                  Writeln ('    Introduzca una cadena');
                                  Writeln ('_____');
                                  Writeln ('Introduzca ''.'' para finalizar');
                                  Readln (cadena);
                                  Writeln (Cadena);
                                  End;
          Writeln ('Finalización de programa');
End.
```

El bucle se ejecut hasta que se encuentre un punto.
Si la condición del bucle es falsa, el bucle WHILE..DO no se ejecuta.

REPEAT .. UNTIL

Es un bucle contrario al While .. Do , se ejecutará hasta que la condición sea cierta, el código del bucle se ejecuta por lo menos una vez, ya que la evaluación de la condición se realiza al final del bucle.

```
Program repunt;
Uses crt;
var
   nom:string;
Begin
     ClrScr;
     Repeat // Iniciamos el bucle
            Writeln ('Introduzca una cadena, ''.'' para terminar');
            Readln (nom);
            Writeln (nom);
     Until nom='.'; // Condición del bucle
End.
```

Podemos alterar la ejecución de los bucles con los comandos Break y Continue.
Break : Rompe la ejecución del bucle.
Continue: El bucle continua la ejecución en la próxima iteración.

CASE .. OF

Con CASE podemos bifurcar el programa dependiendo del valor de la variable.
Es ideal para realizar menús o restringir el valor de las variables de entrada.

CASE expresión OF
 Valor1 : Sentencias1;
 Valor2 : Sentencias2;
  .
  .
  .
 ValorN : SentenciasN;

```
// Menu de Cursos
Program cursos1;
Uses Crt;
Var
   cursos : Integer;
Begin
            ClrScr;
     Cursos :=4;
            While cursos<>3 DO // Bucle repeticion
            Begin
            Writeln ('Introducir cursos : 1=A 2=B 3=Salir');
            Readln (cursos);
            Case cursos OF
            1: Begin // Opcion 1 del menu
                     Writeln ('Curso primero');
                     Writeln ('HORARIO DE 8 A 13');
               End;
            2: Begin // Opcion 2 del menu
                     Writeln ('Curso segundo');
                     Writeln ('HORARIO DE 16 A 21');
               End;
          Else
               Begin
                    Writeln (); // Otras opciones (cadena vacia)
               End;
          End;
End;
End.
```

Otra opción más interesante es utilizar la función Readkey de la unidad Crt, que toma el carácter de la tecla pulsada.
Ej

```
Var
        Tecla:Char;
Begin
.....
Tecla := Readkey;
Case Tecla Of
'1' :Sentencias
'2' :Sentencias
.....
End.
```

2.8Escribe un programa que lea 10 numeros desde el teclado, los liste y escriba su suma.

2.9Realizar un programa que calcule el factorial de un número Ej 5!=5x4x3x2x1

2.10Realizar un programa que lea 10 números enteros y por cada número leído tendremos en pantalla su valor y tantos caracteres – como sea su valor

Ej 6 ----*-

8 ----*---

12 ----*----*--

2.11Realizar un programa en Pascal que lea por pantalla un número de 3 cifras y escriba el número en orden normal e inverso

ej 467 ---> 764

2.12Realizar un programa que lea el mes y año de nacimiento y diga cuantos años tienes y cuantos meses.

2.13Realizar un programa que liste el triángulo de Pascal (6 lineas)

```
          1
        1  1
      1  2  1
    1   3  3   1
  1    4   6   4   1
```

2.14Realizar un programa que mediante un menú, podamos sumar 2 números, restarlos, multiplicarlos y dividirlos.

TEMA III TIPOS DE DATOS ESTRUCTURADOS.

Los datos estructurados son conjuntos de datos que pueden estar definidos o podemos definirlos. Los arrays (vectores y matrices), registros y conjuntos son algunos de estos tipos.

Todos estos nuevos tipos de datos, que no son tipos de datos standar de Pascal, (booleanos,strings,integers ...) se declaran mediante la palabra reservada TYPE en la cabecera del programa.

TYPE T = (C1,C2,...Cn);

Ejemplos:

```
Type vehiculo = (tren,coche,bici,bus,camion);
Type moneda = (euro,dolar,yen);
```

La forma anterior de declaración de tipos es por enumeración, de forma que indicamos todos los valores separándolos por comas.
También podemos definir subrangos

```
Type subrang : min ... max
```

ARRAYS (Matrices)

Una matriz es un conjunto de elementos ordenados por filas y columnas.

La matriz puede ser :

y11	y12	y13
y21	y22	y23
y31	y32	y33

Tambien puede ser una matriz unidimensional:

Y =

y11	y12	y13

Se definen como las variables:
Ejemplos:

```
var
Unidimensional : array [1..n] of tipo de dato;
Bidimensional : array [1..n,1,,n] of tipo de dato;
Tridimensional : array [1..n,1..n,1..n] of tipo de dato;
```

Para almacenar valores en un array, se asigna mediante su fila y su columna:

Ej

```
var
a: array [1..5,1..5] of integer;
begin
a[1,3]:=7;
end.
```

Ej Introducir valores en una matriz unidimensional de 3 columnas.

```
Program matriz1;
var
   contador:Integer;
   valor:Integer;
   matriz: array [1..3] of Integer;
Begin
            For contador :=1 to 3 do
            begin
                        Writeln ('Introduce valor ',contador);
                        readln (valor);
                        matriz [contador]:=valor;
           end;
End.
```

Podemos acceder a los STRINGS mediante un array

frase := 'curso de pascal';
Writeln (frase[2]);
Writeln (frase[5]);

Ejercicios

3.1Realizar un programa que introduzca valores en una matriz unidimensional de 5 columnas y al finalizar la introducción muestre estos valores en pantalla de la forma y= [val1 ,v1l2, val3, ... valN]

3.2Realizar un programa para introducir datos en una matriz 2x2 (Integer) y muestre los valores por pantalla, una vez finalizada la introducción de los valores.

3.3Realizar un programa que sume 2 matrices 2x2 y muestre la pantalla las 2 matrices y el resultado de la suma.
Ejemplo

Tabla 1	*Tabla 2*	*Tabla 3*
1 0	2 1	3 1
2 3	1 3	3 6

3.4Escribir un programa que invierta una frase almacenada en un array
// Utilizar la función Length (Cadena)

CONJUNTOS (SET OF)

Un conjunto, es una relacción de elementos distintos entre sí , pero con una característica común, son del mismo tipo.
Por ejemplo la enumeración de los equipos de basket de la ACB sería un conjunto.
En Pascal, los conjuntos se representan como enumeraciones dentro de corchetes.
Ejemplos [12,3,45,6,8]
Type
Testudios = (electronica,mecanica,comercio,administracion);
Tconjestudios = set of Testudios;

Los conjuntos se pueden:
- Asignar Estudios:=[fisica,quimica,textil]
- Unir Estudios:=Estudios + [matematicas]
- Restar Estudios:=Estudios-[quimica]
- Interseccion Estudios:=Estudios*Asignaturas

REGISTROS O RECORDS

Un registro, es una estructura de datos muy flexible, básicamente es una colección de información referida a un objeto.
Por ejemplo, la información del DNI de una persona, tiene campos numéricos, gráficos, strings etc todo esto se almacena en un registro.
Los datos de una agenda telefónica serian otro ejemplo de registros.
Un registro en pascal, se declara de la siguiente forma:

```
Type Nombreregistro = Record
       variable1 : tipo1;
       variable2 : tipo2;
       .....................
       variableN : tipoN;
end;
```

Veamos un ejemplo concreto:

```
Program ficha1;
Uses Crt;
Type
          TClientes = record
          Nombre : String;
          Apellidos : String;
          Telefono : String;
          Direccion : String;
          end;
Var clientes : TClientes;
Begin
           Clrscr;
           Writeln ('Introducción de datos');
           Writeln ('Nombre :');
           Readln (clientes.nombre);
           Writeln ('Apellidos :');
           Readln (clientes.Apellidos);
           Writeln ('Telefono :');
           Readln (clientes.Telefono);
           Writeln ('Dirección :');
           Readln (clientes.Direccion);
           Clrscr;
           Writeln ('_____________________________');
           Writeln ('Nombre :',clientes.nombre);
           Writeln ('Apellidos :',clientes.Apellidos);
           Writeln ('Dirección :',clientes.Direccion);
           Writeln ('Telefono :',clientes.Telefono);
           Writeln;
           Writeln ('_________ FINAL DE REGISTRO ___________');
End.
```

Una opción interesante, es la declaración de arrays cuyos campos son registros

```
escola : array [1..10] of
        record
            nombre : array [1..15] of char;
            tel:String:
        end;
```

Ejercicios.

3.5Implementar un programa de notas para 5 alumnos con 3 asignaturas. El programa pedirá los nombres de los alumnos y las 3 notas y sacará por pantalla, los boletines de todos los alumnos.

3.6Escribir un programa que genere horarios de trenes entre 2 ciudades. El programa constará de 2 partes, una para modificar el horario y otra para visualizarlo

Horario de IDA
Ciudad A ..... Ciudad B
Salida X:X .....Llegada X:X
Salida X:X.....Llegada X:X
Salida X:X.....Llegada X:X
Horario de vuelta
Ciudad B .....Ciudad A
Salida X:X.....Llegada X:X
Salida X:X.....Llegada X:X
Salida X:X.....Llegada X:X

3.7Realizar un programa que dándole un valor, rellene un array con los 5 valores siguientes.

3.8Escribir un programa que rellene un array con los 10 primeros números pares y los sume.

3.9Escribir un programa que rellene un array con 5 notas, los muestre en pantalla y calcule la nota media.

3.10Realizar un programa que introduciendo valores en una matriz 3x3, muestre estos valores y muestre el valor mayor y su posición.

3.11Escribir un programa que muestre la tabla ASCII

3.12Escribe un programa para encriptar una cadena sumándo a cada carácter 3 a su valor1
Ej ABC ---------- CDE

TEMA IV ARCHIVOS/FICHEROS

Cuando los datos o resultados de un programa, deben ser almacenados para poder recuperarlos posteriormente, de forma que estos almacenes no sean volátiles, es decir no se pierdan cuando se desconecta la computadora, necesitaremos trabajar con ficheros.
Estos ficheros, pueden estar en discos flexibles, discos duros etc.
En Pascal podemos trabajar con 3 tipos de archivos:

- Archivos con tipo
- Archivos sin tipo
- Archivos de texto

Hay dos tipos posibles de accesoso a los datos:

- Acceso aleatorio(Se puede acceder a cualquier dato en un instante)
- Acceso secuencial(Se debe recorrer todos los datos hasta encontrar el que se busca)

ARCHIVOS DE TEXTO

Los archivos de texto son del tipo text

```
var archivotxt : text;
```

Para poder trabajar con el nombre de archivo reconocido por el sistema operativo, asignaremos un alias:

```
assign (archivotxt,'nombrefichero.txt');
```

Hay que tener cuidado con la asignación, ya que si Windows no es case sensitive, otros sistemas operativos si lo son y por lo tanto diferencian entre mayúsculas y minúsculas (Caso de sistemas UNIX).

Comandos parea trabajar con ficheros:

- RESET (Archivo); // Abierto para lectura
- REWRITE(Archivo); // Borramos el fichero para poder sobreescribir en él.
- APPEND(Archivo); // El cursor se situa al final para añadir nuevos datos.
- CLOSE(Archivo); // Cerramos el fichero para no perder datos

Ejemplo de programa para trabajar con ficheros

```
Program fichero;
var archivotxt:Text;
Begin
            Assign(archivotxt,'fichero.txt');
            //Asignamos archivotxt --> fichero.txt
            Rewrite(archivotxt);
            //Abrimos el archivo borrando todo
            Writeln (archivotxt,'Prueba de escritura en fichero.txt');
            //Escribimos la frase en el archivo
            Close(archivotxt);
            //Cerramos el archivo para no perder datos
            Writeln ('Final de programa');
End.
```

El procedimiento básico es asignar una variable el nombre del fichero reconocido por el Sistema Operativo.
-Abrir el fichero con Reset,Rewrite o Append
-Escribir o leer en el fichero
-Cerrar el fichero para no perder los datos.

Hay dos funciones interesenates en el trabajo con archivos:

- EOF (End of file) Final de fichero
- EOLN (End of line) Final de linea

Ej While NOT EOF do

-SEEK (Se posiciona dentro del fichero)
Ej SEEK (archivo,5); (El cursor se coloca en la posición 5 del fichero)

Ejercicios:
4.1Escribir un programa que escriba ficheros de texto con readln y writeln
4.2Escribir un programa que lea un fichero de texto fichero.txt y lo escriba en pantalla
4.3Diseña un programa en Pascal que escriba en un fichero de texto el triangulo de Pascal

```
      1
    1  1
  1   2   1
1   3    3   1 etc (como mínimo 6 lineas)
```

4.4Escribir un programa que lea un fichero de texto, lo muestre en pantalla y numere cada línea del mismo

---

ARCHIVOS CON TIPO

Podemos trabajar con ficheros de un tipo determinado.
Se utiliza la sentencia FILE OF tipo de archivo.

Ejemplo :

```
Type
Inventario = record
                nombre : array [1..25] of char;
                descripcion: array [1..100] of char:
                cantidad : Integer;
End;
Var
                fichero:file of Inventario;
                temporal:Inventario;
Begin
                .....;
                .....;
End.
```

En los ficheros de tipo, hay que trabajar con variables temporales antes de escribir en ellos.

Otras funciones interesantes:

- filesize(archivo) //Indica el tamaño del archivo.
- filepos(archivo) //Indica la posición del cursor en el fichero.

Ejercicios:

4.5Crear una agenda simple con 4 campos, nombre, apellido1, apellido2, telefono y se grabe en un archivo llamado agenda.txt

4.6Realizar un menú para el programa anterior de forma que se puedan añadir, leer o borrar toda la agenda.

ARCHIVOS SIN TIPO

No tienen ningun formato definido, la diferencia es que hay que utilizar las funciones BlockWrite y BlockRead en vez de write y read, además hay que indicar la cantidad de bytes que se tienen que leer o escribir, por defecto 128 bytes.
Se suele utilizar la función sizeof (Variable) para determinar la cantidad de bytes.
Ej:

```
fichero:file;
Assig(file,'fichero.fic');
Blockread(fichero,variable,tamañoenbytes);
```

Otras funciones referentes a los archivos:

- ERASE (Borramos archivos) // Ej erase (archivo)
- RENAME (Renombramos archivos) // Permite renombrar archivos

- Fileexists (Nomarchivo) //Indica la existencia de un archivo (If NOT fileexists (Nomarchivo) THEN ...
- Readkey ; // Necesita CRT,La función sin argumentos, esperará a que se pulse una tecla.

4.7Realizar un programa que lea ficheros de texto y cuente letras.

4.8Implementar un programa para contar palabras dentro de un fichero de texto.

4.9Realizar un programa en Pascal que lea un número de 2 digitos y muestre por pantalla su transcripción en letras.

TEMA V FUNCIONES Y PROCEDIMIENTOS.

Muchas veces, resolver problemas complejos, se puede realizar mejor dividiendo el problema en pequeños problemas que sean más fáciles de resolver.

Otra posibilidad en un programa es que se repitan bloques de sentencias de forma que cada vez tengamos que teclearlas, par estos tipos de problemas, Pascal propone dos elementos: Las funciones y los Procedimientos.

FUNCIONES

Las funciones son bloques de programa que devuelven siempre un tipo de datos determinado.
Por ejemplo, una función de la unidad Sysutils es FileExists:

If FileExists (Nombredefichero) then //Devuelve TRUE o FALSE

Para declarar una función, siempre hay que hacerlo antes del bloque principal BEGIN --- END
La forma de declaración es:

FUNCTION Nombredefuncion (PARÄMETROS) : Resultado ;

Ejemplo //Cálculo de longitudes de circunferencia.

```
Program longitud;
uses crt;
const
            PI = 3.141592;
var
            radio : real;
// Empieza la función
function long (r:real):real;
(* long = nombre de la funcion
la declaración de la variable r:real será el radio
y por último el resultado es real *)
            begin
                  long:=2*PI*r; // Se calcula la longitud 2*PI*R
            end;
//Termina la función
Begin
            Writeln ('Introduzca radio ');
            Readln (radio);
            Writeln ('Resultado ',long(radio));//Llamada a la función long
            //se le pasa como parámetro el rádio
            Readkey; //Espera hasta que se pulsa una tecla
End.
```

Ejercicios

5.1Realizar un programa que determine si un valor entero A es mayor que otro B. Utilizar una función para realizar la comprobación.

5.2Implementar un programa que liste la temperatura de cada mes del año y la media de temperaturas implementada con la función media.

PROCEDIMIENTOS

Son subprogramas pero que al contrario que las funciones, no hay que especificar el tipo de salida, las funciones son llamadas en una sentencia pero los procedimientos son llamados mediante una llamada al procedimiento.

Ej

```
Program autor;
Procedure proautor;//Empieza el procedimiento
          Begin
          Writeln ('Programaciò en Pascal');
          Writeln ('Curs 1er B Electrònica');
          End;
           //Finaliza el procedimiento
Begin
          proautor;//Llamada al procedimiento
End.
```

Ejemplo de procedimientos para determinar si un valor entero es mayor que otro.

```
Program mayor1;
Uses crt;
Var val1,val2 : Integer;

Procedure mayor(a,b:Integer);
//Se inicia el procedimiento
(* nombre del procedimiento = mayor
datos de entrada a,b del tipo Integer *)
          var resultado:String;// Se utiliza una variable local
          Begin
                    If a > b then Resultado := 'A > B';
                    If a < b then Resultado := 'A < B';
                    If a = b then Resultado := 'A = B';
                    Writeln (Resultado);
          End;
// Final del procedimiento
(* Se inicia el cuerpo del programa principal *)
Begin
           Writeln ('Introducir valor A , valor B ');
           Read (val1,val2);// Leemos los dos enteros separados pos espacio
           mayor (val1,val2);// Llamada al procedimiento
           Readkey;// Esperamos a pulsar una tecla
End.
```

5.3Escribir un programa que calcule el valor máximo de un vector de 5 valores, utilizando un procedimiento.

5.4Modificar el programa anterior para que se puedan sumar 2 vectores, se indique el valor mayor de la suma y la posición de este valor en el vector resultado.
Utilizar procedimientos

5.5Escribir un procedimiento para intercambiar el valor de 2 variables
Si A=3 y B=5 ---> A=5 B=3

5.6Escribir un procedimiento para calcular determinantes de matrices 2x2

5.7Escribir un programa para calcular determinantes 3x3, utilizar un procedimiento e utilizarlo para calcular sistemas de 3 ecuaciones de 3 incógnitas

VARIABLES LOCALES Y VARIABLES GLOBALES

Las variables locales son aquellas que se utilizan sólo en una función o un procedimiento, su rango está limitado sólo al uso de la función.
Las variables globales son las que se declaran en la cabecera del programa principal y por tanto, su rango se extiende a todo el programa, incluyendo funciones y procedimientos.
Hay que tener cuidado a la hora de manejar este tipo de variables y procurar no utilizar los mismos nombres, ya que podria dar lugar a errores.

Ej

```
Program global;
var
A,B,C,D >: TIPO ---> Variables globales

function local
        var d,e,f : tipo --> Variables locales
Begin
        .....
        .....
End.
```

PUNTEROS

Hasta ahora hemos trabajado con variables estáticas, por lo que el compilador gestiona la memória de estas variables, pero cuando queremos acceder a posiciones de memoria, o modificar el espacio de memoria de las variables, necesitamos los punteros.

Un puntero es una variable que apunta a una posición de memoria.

```
Var
   puntero : ^Integer;
```

Para poder utilizar los punteros, necesitamos reservar la memoria necesaria para estas variables.
Pascal implementa una función para reservar memoria a los punteros; New(Puntero);
Para poder liberar la memoria de los punteros utilizaremos la función Dispose(Puntero);

Por último necesitamos una variable que apunte a la posición de memoria

```
        Puntero^:=5;
```

En este caso, la variable puntero apunta a la dirección de memoria 5.

Ej

```
Program puntero1;
var
    Puntero: ^Integer; // Utilizamos punteros enteros
Begin
   New(Puntero); // Asignamos espacio de memoria a la variable puntero
   Puntero^ := 10; // Apuntamos a la dirección 10
   Dispose (Puntero); // Liberamos la memoria
End.
```

Si no reservamos memoria para el puntero, éste tendrá el valor NIL
Ej: IF puntero = NIL THEN ...

Si empleamos el símbolo @ delante de una variable estática, obtendremos su posición de memoria.

Otro tipo de punteros, son los punteros de tipo POINTER, de forma que no está definido el tipo. Para reservar memoria a estos tipos de punteros utilizaremos Getmem y para liberar la memoria utilizamos Freemem

Ej:

```
Program puntero2;
var
     puntero : pointer ;
     punteroentero : ^Integer;
Begin
    Getmem (puntero,sizeof(integer));
    punteroentero : = puntero;
    punteroentero^ : = 10;
    Freemem (puntero,sizeof(integer));
End.
```