

Recuerde que estos son apuntes muy simplificados que deberá completar con la bibliografía recomendada

INTRODUCCIÓN AL TURBO PASCAL.

El lenguaje Turbo Pascal se caracteriza por ser un lenguaje de propósito general (puede usarse en un gran número de diversas aplicaciones), es un lenguaje procedimental, es estructurado (usa las estructuras repeat, for, while y no necesita go to) y recursivo, y posee una gran riqueza de tipos de datos.

Estructura de un programa haciendo uso del lenguaje Turbo Pascal:

PROGRAM	identificador	}	cabecera del programa
<hr/>			
USES	asocia una serie de rutinas de control de pantalla	}	Sección de declaraciones y definiciones
LABEL	declaración de etiquetas		
CONST	definición de constantes		
TYPE	declaración de tipos de datos definidos por el usuario		
VAR	declaración de variables		
<hr/>			
FUNCTION	Declaraciones y cuerpo de los subprogramas	}	Sección de subprograms
PROCEDURE			
<hr/>			
BEGIN	Sentencias o instrucciones	}	Cuerpo del programa
END.			
<hr/>			

OBJETOS DE UN PROGRAMA EN TURBO PASCAL.

IDENTIFICADORES: un identificador es un nombre dado a un elemento de programa, tal como una constante, una variable, un procedimiento, un programa, una función, una unidad, un campo de registro, etc.

CARACTERÍSTICAS DE LOS IDENTIFICADORES:

- Secuencia de caracteres de cualquier longitud, sólo los 63 primeros son significativos.
- Debe comenzar por una letra (A ► Z) y no puede contener blancos.
- Después del primer carácter se aceptan letras, dígitos y subrayado (_).
- No se pueden usar palabras reservadas.

Ejemplos válidos: NOMBRE_APELLIDO, IMPUESTO, NOTA2, H346

Ejemplos no válidos: NOMBRE APELLIDO, EJ?AB, 23ALX, 4NOM, &NOM

NOTA: El Turbo Pascal no distingue las letras mayúsculas de las minúsculas en los identificadores. Ejemplo: EDAD edad Edad son identificadores válidos e idénticos.

PALABRAS RESERVADAS: son palabras que tienen un significado especial y que no se pueden utilizar para otros propósitos. Las palabras reservadas no se pueden usar como identificador.

Ejemplo de palabras reservadas: ABSOLUTE, AND, ARRAY, BEGIN, CONST, DIV, DO, DOWNTON, ELSE, END, EXTERNAL, FILE, FOR, FORWARD, FUNCTION, GOTO, IF, IMPLEMENTATION, IN, INLINE, INTERFASE, INTERRUPT, LABEL, MOD, NIL, NOT, OF, OR, PAKED, PROCEDURE, PROGRAM, RECORD, REPEAT, SET, SHL, SHR, STRING, THEN, TO, TYPE, UNIT, UNTIL, USES, VAR, WHILE, WITH, XOR, etc.

DATOS: cifras originales que por si solas tienen poco significado. Son los conceptos básicos o elementales como el nombre de las cosas o personas, las cantidades, etc.

TIPOS DE DATOS:

Gráficamente los distintos tipos de datos del Turbo Pascal se muestran en la figura Nº 1. A continuación describiremos la mayoría de los que van a ser utilizados en el curso de Programación Digital.

DATOS NUMÉRICOS: conformados por dígitos, puede incluir un signo (+,-), un punto decimal (.) y un exponente. No pueden contener comas (,) o espacios en blanco.

DATOS NUMÉRICOS ENTEROS (INTEGER): No contienen ni punto decimal, ni exponente.

Ejemplo: 5 +5 -6 7343

Rango de valores posibles entre: -3276832767

Enteros largos (**LOGINT**), su rango de valores: -21474836482147483647

DATOS NUMÉRICOS REALES (REAL): Deben contener punto decimal (.) o un exponente o ambos.

Ejemplo: 0.0 -0.2 -314.63 0.000078 +32.45 32.45

En forma exponencial la base 10 se reemplaza por la letra **E**, manteniendo la forma siguiente: nE+-dd donde n = número decimal y dd = exponente entero (+,- o cero) .

Ejemplo:

5.2x10 ⁻⁴ ➔ 5.2E-4 5.2e-4

 -6.784E-12 7.0E5 7.0E+5

DATOS CHARACTER (CHAR): sólo pueden contener un caracter y deben estar encerrados entre apóstrofes. Pueden ser una letra (A ➔ Z), un dígito (0 ➔ 9) o un caracter especial (\$,*, &, etc.).

Ejemplo: 'A' '*' '6' ' ' 'X'

NOTA: No se pueden utilizar apóstrofes cuando se introducen caracteres desde el terminal.

DATOS CADENA (STRING): es una secuencia de caracteres (letras, dígitos o caracteres especiales) escritos en una línea sobre el programa y encerrados entre apóstrofes, generalmente no mayor de 255 caracteres.

Ejemplo: 'DIOS SI EXISTE' '¿COMO ESTAS?' 'EDO. MERIDA'
'6457-AL3' '2X(3+D)-J'

DATOS LÓGICOS (BOOLEAN): pueden tomar sólo dos valores:

true ➔ significa verdadero

false ➔ significa falso

donde false < true



LOS DATOS DE UN PROGRAMA SE PUEDEN CLASIFICAR EN CONSTANTES Y VARIABLES.

CONSTANTES: son datos que no varían durante la ejecución o vida del programa. El dato simple se asocia a menudo a un IDENTIFICADOR que le proporciona un nombre al dato. El identificador se dice que es una constante si se le asigna un dato permanente.

Las constantes deben ser declaradas antes de su utilización.

Forma de definición de constantes:

CONST nombre = valor;
nombre1, nombre2, nombren = valor;

Donde: **CONST** es la palabra reservada para la declaración de constantes
nombre es un identificador que representa el nombre de la constante
valor es el dato efectivo que se asigna al nombre, el cual puede ser entero, real, carácter, cadena, lógico, ..., conjuntos o arreglos, ... o una expresión que se evalúa en tiempo de compilación.
; punto y coma es el elemento separador de sentencias.

Ejemplos: CONST

Pi = 3.141592;	→	Constante real
CUENTA = 632;	→	Constante entera
Min = 0;	→	Constante entera
SUMA = (2.5+40)/(3.5-4);	→	Constante real
COLOR = 'AZUL';	→	Constante cadena (string)
ANCHO,ALTO,LONG = 25;	→	Constantes enteras
SEC = 'X';	→	Constante caracter
COD = true;	→	Constante lógica
R = 25.E-7;	→	Constante real

NOTA: El tipo del valor asociado a la constante define el tipo de la constante. Los valores de las constantes tipo cadena o caracter deben ser asignados entre apóstrofes.

VARIABLES: puede decirse que una variable es un identificador cuyo valor puede cambiar durante la ejecución del programa. Cada variable debe ser declarada (definida) individualmente en la sección de declaración del programa antes de ser utilizadas.

Forma de definición de variables:

VAR nombre: tipo; o **VAR** nombre1, nombre2, ...nombren: tipo;

Donde: **VAR:** es la palabra reservada para la declaración de variables.
nombre: es el identificador que representa el nombre de la variable.
tipo: se refiere al tipo de dato contenido en la variable. Existen tantos tipos de variables como tipos de datos diferentes.
; punto y coma es el elemento separador de sentencias.

Ejemplos: VAR

EDAD: INTEGER;
SALARIO: REAL;
NOMB: STRING[30];
LETRA: CHAR;
NACIONALIDAD: CHAR;
CONDICION: BOOLEAN;

Las variables podrían tomar los valores siguientes:
EDAD 15, SALARIO 400000.00
NOMB 'Pedro Pérez'
LETRA 'x' NACIONALIDAD 'V'
CONDICION false

NOTA: Las definiciones de constantes deben preceder a las declaraciones de variables. Ejemplo: `CONST TITULO = 'LA CASA DE LA CULTURA';`

`FRAC = 0.18453;`
`VAR FILA, COLUMNA: INTEGER;`
`SB: REAL;`

COMENTARIOS: es cualquier frase encerrada entre llaves { } o entre los signos (* *) que puede acompañar a las instrucciones de un programa, sean estas de declaraciones o pertenecientes al cuerpo del mismo.

Ejemplo: `VAR EDAD: INTEGER; {Edad del estudiante}`
`SALARIO: REAL; (* Sueldo del empleado *)`

EXPRESIONES Y OPERACIONES ARITMÉTICAS

Una **expresión aritmética** es un conjunto de datos (constantes o variables) o funciones unidos por operadores aritméticos.

OPERADORES ARITMÉTICOS BÁSICOS:

+ SUMA
- RESTA
*** MULTIPLICACIÓN**

Se usan con operandos enteros o reales. Si ambos son enteros el resultado es entero. Si alguno es real el resultado es real.

/ DIVISIÓN

El resultado siempre es real independientemente del operando

DIV DIVISIÓN ENTERA

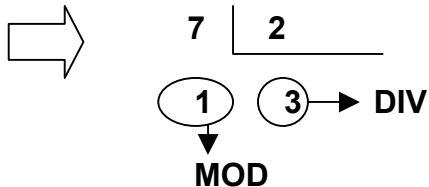
Cociente entero de a/b

MOD MÓDULO

Resto de a/b

NOTA: los operadores **div** y **mod** sólo se pueden utilizar con números enteros

Ejemplos: $7 \text{ div } 2 = 3$
 $7 \text{ mod } 2 = 1$



Otros ejemplos: $3 \text{ div } 5 = 0$, $15 \text{ div } 3 = 5$, $17 \text{ div } 3 = 5$, $3 \text{ div } -15 = 0$
 $7 \text{ mod } 5 = 2$, $-15 \text{ mod } 6 = -3$, $3 \text{ mod } 5 = 3$, $-5 \text{ mod } 3 = -2$,

18.5 div 3.0 = no válido

Si operando1 es 0, el resultado de la operación **div** o **mod** no está definido
Si operando2 es negativo, el resultado de la operación **mod** está indefinido

REGLAS PARA EVALUAR EXPRESIONES ARITMÉTICAS:

1. Todas las subexpresiones entre paréntesis se evalúan primero. Las subexpresiones con paréntesis anidados se evalúan de dentro hacia fuera. El paréntesis más interno se evalúa primero.
2. Prioridad de operadores aritméticos:
 * / **div** **mod** → **se evalúan primero**
 + - → **se evalúan de último**
3. Regla asociativa izquierda: los operadores con igual prioridad se evalúan de izquierda a derecha.

Ejemplos: $4+2*5 \rightarrow 4+10 \rightarrow 14$

$23*2\text{div}5 \rightarrow 46\text{div}5 \rightarrow 9$

$3+5*(10-(2+4)) \rightarrow 3+5*(10-6) \rightarrow 3+5*4 \rightarrow 3+20 \rightarrow 23$

$(9+3)*5\text{div}4\text{mod}7+1 \blacktriangleright 12*5\text{div}4\text{mod}7+1 \blacktriangleright 60\text{div}4\text{mod}7+1 \blacktriangleright 15\text{mod}7+1 \blacktriangleright 1+1 \blacktriangleright 2$

FÓRMULAS: en Pascal las fórmulas matemáticas se escriben en forma lineal.
Ejemplo:

$a + \frac{b}{c+d} \rightarrow A+B/(C+D)$

$b^2-4ac \rightarrow b*b-4*a*c$

FUNCIONES: una función es un subprograma que recibe como argumentos o parámetros datos de tipo numérico o no numérico (char, string, boolean u otros) y devuelve un resultado.

Las funciones pueden ser predefinidas o definidas por el usuario.

Ejemplo de algunas de las funciones predefinidas:

FUNCION	EFEECTO	TIPO DE PARÁMETRO	TIPO DE RESULTADO
Abs(x)	Calcula valor absoluto de x	Entero o real	Entero o real
* Arctan(x)	Calcula arcotangente de x	Entero o real	Real

* Cos(x)	Calcula coseno de x	Entero o real	Real
Exp(x)	Calcula exponencial de x (e^x)	Entero o real	Real
Frac(x)	Devuelve parte decimal de x	Real	Real
Int(x)	Devuelve parte entera de x	Real	Real
Ln(x)	Calcula logaritmo natural de x	Entero o real	Real
Pi	Devuelve el valor de Pi (3.1415...)	Real	Real
Round(x)	Redondea el valor de x al entero positivo más próximo. Round(-x) = Round(x)	Entero o real	Entero
* Sin(x)	Calcula seno de x	Entero o real	Real
Sqr(x)	Calcula cuadrado de x	Entero o real	Entero o real
Sqrt(x)	Calcula raíz cuadrada de x ($x \geq 0$)	Entero o real	Real
Trunc(x)	Suprime la parte decimal de x	Real	Entero
Log10(x)	Logaritmo base 10	Entero o real	Real

NOTA1: las funciones marcadas con un * significa que el argumento es siempre en radianes.

NOTA2: la expresión X^Y se escribe en Turbo Pascal de la siguiente manera:
Exp(Y*Ln(x))

Ejemplos de funciones predefinidas:

Trunc(5.2) = 5 Trunc(5.99) = 5 Trunc(-3.14) = -3 Round(4.44) = 4

Round(18.5) = 19 Round(-7.15) = -7 Round(0.7) = 1 Abs(-63) = 63

Abs(3.97) = 3.97 Frac(28.437) = 0.437 Int(45.438) = 45.0

Exp(4.5) = $e^{4.5}$ = 2.798282^{4.5}

Otras funciones que utilizaremos en el curso son las siguientes:

Función UPCASE: cambia las letras **minúsculas** a letras **MAYÚSCULAS**. Si las letras ya están en MAYÚSCULAS las deja igual.

forma: **UPCASE (s);** donde s es una expresión tipo **char**

Ejemplo: UPCASE('a') \longrightarrow 'A'
 UPCASE('A') \longrightarrow 'A'

Función RANDOM: devuelve un número pseudoaleatorio.

donde **n** debe ser una expresión entera de valor mayor que 0, de ser 0 o negativo se produce un error.

n es opcional

forma: **RANDOM (n)**

Si no existe **n** la función devuelve un número pseudoaleatorio en el rango:

$0 \leq \text{número} < 1$

Si **n** existe la función devuelve un número entero pseudoaleatorio en el rango:

$0 \leq \text{número} < n$

Función KEYPRESSED: es una función de valor lógico que devuelve true (verdadero) si se ha pulsado una tecla en el teclado.

Forma: **KEYPRESSED**

nota: esta función no detecta teclas de desplazamiento o alternativas como shift, alt, numlock,

EXPRESIONES Y OPERACIONES ENTRE CADENAS DE CARACTERES.

Una **expresión cadena** es un conjunto de datos tipo string (constantes o variables) unidos por el operador de concatenación **+** o la función **concat**.

Ejemplo:

'Universidad de' + ' Los Andes' → 'Universidad de Los Andes'

Función concat: permite concatenar una secuencia de cadenas

forma: **concat (x1,x2,...,xn)** donde **x1,x2....xn** son cadenas o variables string.

Si la cadena resultante es mayor que 255 caracteres, los caracteres después del 255 son truncados.

Ejemplo:

Suponiendo que **A** ← tuviese el valor de 'programacion' y **B** ← 'digital'

Concat (A,B) daría como cadena resultante 'programaciondigital'

Concat (A, ' ', B) → 'programacion digital'

Concat (A, ' analogica') → 'programacion analogica'

Ver ejemplos de lo expuesto en estos apuntes y demás explicaciones en clase.

NOTA: Los conceptos anteriores son tomados básicamente de:

- Programación en Turbo/ Borland. Pascal 7. Luis Joyanes Aguilar
- Programación con Lenguaje Turbo Pascal. F.J. Sanchis Llorca.