Capitulo 1

1. Algoritmo – Procesador
2. Sintaxis Pascal
3. Tipos de datos, operadores
4. Estructura de un programa (Declaraciones y Sentencias ejecutables)
5. Errores sintácticos y semánticos
6. Ejemplos con estructura secuencial
7. Funciones del lenguaje Pascal

1. Un **algoritmo** es una secuencia ordenada y finita de pasos, exenta de ambigüedades que lleva a la solución de un problema dado. Cada paso describe una acción.

Se puede desarrollar un algoritmo para escribir los N primeros números naturales (para un valor de N dado), en cambio no es posible hacerlo para escribir todos los números naturales, pues los pasos de la solución son infinitos.

Las etapas para construir el algoritmos que resuelve un problema son :

a. Leer el enunciado del problema con atención, comprender la complejidad del mismo, determinar que resultados se quieren obtener y cuales son los datos que conocemos o sea nuestro punto de partida.
b. Inventar un ejemplo concreto para el problema planteado. (Ejemplo : N =5 → 1, 2, 3, 4, 5)
c. Resolverlo "manualmente", analizando que operaciones debemos efectuar sobre los datos, como están relacionados los mismos, como se condicionan entre ellos (comenzaría con el 1, lo escribo, luego le sumo uno, lo escribo ..., repito estas operaciones hasta que el numero que obtengo es igual a N)
d. Describir el algoritmo en un lenguaje apropiado (por ej: Pascal).

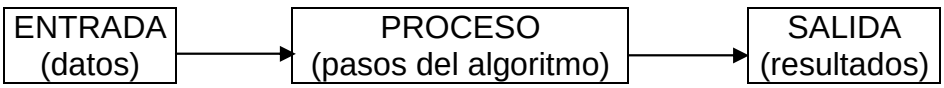
El lenguaje en el que se describe un algoritmo depende del procesador que lo va a ejecutar.

Procesador, entidad capaz de comprender un enunciado y ejecutarlo

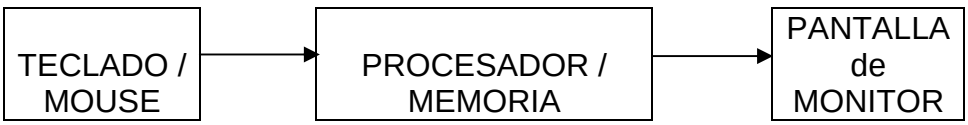
Una computadora es un procesador capaz de captar datos, operarlos y mostrar resultados

Para desarrollar algoritmos (programas) que ejecutara una computadora no basta con conocer el tipo de tareas puede comprender y realizar, sino también la sintaxis para expresarlas.

Como punto de partida, se considera un modelo computacional muy sencillo



La computadora que corresponde a este modelo es:



Los pasos que describe el algoritmo se realizaran en el orden en el que están escritos, deben estar dentro del repertorio que comprenda la computadora y ser eficaces para que produzcan la salida correcta para todas las posibles variaciones de la entrada. Incluso deben considerarse valores críticos para la entrada (ejemplo N=0) a tener en cuenta en el planteo de la solución.
Los procesos leen los datos, los operan y escriben los resultados.

Para evaluar el funcionamiento de un algoritmo hay que determinar el estado del mismo, se concibe como una instantánea que describe el antes y el después de la ejecución de uno o varios pasos.
Es de especial interés el estado inicial y final de un algoritmo.
El primero es la descripción de la entrada antes de que se ejecute el primer paso del algoritmo y recibe el nombre de precondición. En el ejemplo visto sería N entero positivo, N > 0. El estado final describe la entrada y la salida después que se ha ejecutado el algoritmo y recibe el nombre de poscondición. En general la entrada queda vacía, esto indica que una vez leídos los datos, no pueden leerse de nuevo. Los datos ingresan por lectura en el mismo orden en el que son tipeados en el teclado, o que han sido grabados en un archivo.
Los estados intermedios son las transformaciones que van sufriendo los datos para llegar a los resultados requeridos.
Estas especificaciones requieren del concepto de **variable,** nombre simbólico que se asocia a un valor durante la ejecución del algoritmo, pero que no puede determinarse (está vacía) en el momento que se construye o formula el algoritmo.

Un lenguaje algorítmico
- ✓ tiene una sintaxis y un vocabulario limitado
- ✓ comprende solo el tipo de acciones que una computadora puede entender y realizar
- ✓ cada acción tiene una forma rigurosa de expresarse y una significación propia (sintaxis y semántica)

La descripción de un algoritmo en un lenguaje de programación constituye un programa

2. Sintaxis del lenguaje Pascal

Un programa se compone de

☞ **Declaraciones:** describen características de los objetos que utiliza el programa.

☞ **Sentencias:** o instrucciones ejecutables describen las acciones que el programa realizará. Pueden ser simples o compuestas.

SIMPLES	*COMPUESTAS*
No contienen otra sentencia Ejemplos : Write, Read	Grupo de sentencias simples encerradas entre un BEGIN y un END. Ejemplo: BEGIN sentencia 1; sentencia 2; sentencia n; END;

A continuación se describe algunos elementos del lenguaje y características generales

Delimitadores: se utilizan para agrupar o separar declaraciones o sentencias.
Ejemplos: ';' - 'BEGIN' - END'.

Comentarios: se utilizan para hacer aclaraciones en cualquier lugar del programa (son ignorados por el compilador). Se recomienda su uso para mayor legibilidad. Se encierran entre llaves o paréntesis y asterisco. Ejemplo: {esto es un comentario} , (*esto es un comentario*).

Palabras reservadas: son aquellas que sólo pueden usarse para un fin específico.
Ejemplos: begin, function, program, etc.

Identificador: es el nombre que le asignamos a un objeto (variable, constante, programa, subprograma), debe comenzar con una letra y a continuación letras o dígitos o el carácter '_'.
Ej: salario_bruto
No hay distinción entre mayúsculas y minúsculas.
Ej: SuMa = sumA
No hay restricción respecto a la longitud, pero debe considerarse la legibilidad así como también la facilidad para comprender y escribir el código. Es conveniente que el identificador esté asociado al rol de dicho objeto (Precio, Nombre, Promedio, SueldoProm, etc.)

3. Tipos de Datos

Los datos que opera un programa tienen características propias: la edad es un entero, precio es real, nombre es una cadena de caracteres, etc.
El lenguaje provee tipos estándar o simples para enteros, reales, caracteres y lógicos. Los dos primeros tienen a su vez varias posibilidades de rango y precisión. Los tipos simples permiten almacenar un único valor.
El programador puede declarar otros tipos para describir sus datos.
El tipo cadena permite almacenar una secuencia de caracteres y tratarla como una unidad de información.

SIMPLES (escalares estandar)	**REALES**	***SINGLE***	*1.5E-45 a 3.4E38*	**ORDINALES**
		REAL	*2.9E-39 a 1.7E38*	
		DOUBLE	*5.0E-324 a 1.7E308*	
		EXTENDED	*1.9E-4851 a 1.1E4932*	
	ENTEROS	***SHORTINT***	*-128 a 127*	
		BYTE	*0 a 255*	
		WORD	*0 a 65535*	
		INTEGER	*-32,768 a 32,767*	
		LONGINT	*-2,147,483,648 a 2,147,483,648*	
	CHAR			
	BOOLEAN			
DEFINIDOS por el USUARIO	**SUBRANGO**			
	ENUMERADOS			
STRING				
ESTRUCTURADOS	***ARRAY***			
	RECORD			
	SET			
	FILE			

A continuación se presenta una parte (la mas utilizada) de la tabla ASCII DOS, para la representación de caracteres, cada uno tiene una representación numérica, por ejemplo el carácter 'A' se codifica internamente con el número 65, el carácter blanco como el 32 . El número 31 no es representable como carácter y la representación del 30 coincide con la del 45.
Para visualizar un carácter debe mantener presionada la tecla **alt** mientras digita su respectivo código numérico.

ACII	0	1	2	3	4	5	6	7	8	9
30	-		blanco	¡	“	#	$	%	&	‘
40	(	)	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	¿	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	|	}	~		Ç	ü

Como se ve en la tabla, existe un orden entre los caracteres y es posible compararlos en forma aislada ‘A’ < ‘h’.
Notar que es diferente la representación del valor entero 5 (internamente un 5 en binario) del carácter ‘5’ (internamente un 53 en binario)

Un tipo string es una secuencia de hasta 255 caracteres, delimitada entre apóstrofos.
Es posible limitar el número de caracteres de una string especificando entre corchetes dicha cantidad, por ejemplo string [n], con n entero menor a 255. La longitud exacta de la cadena almacenada se obtiene con la función length(…), dicho valor coincide con la posición del último carácter. Es posible el acceso individual a un carácter de la cadena, indicando a continuación del nombre de la variable, su posición entre corchetes.
Se puede establecer una relación entre cadenas, ya que al estar conformadas por caracteres el orden de estos determina el resultado de la relación. Por ejemplo ‘Papa’ > ‘PAPA’.

El tipo boolean toma valores TRUE o FALSE, estos valores no pueden ser ingresado (lectura), si pueden ser visualizados (escritura).

Los tipos integer, boolean, char se los conoce como ordinales. Un tipo de datos es ordinal porque el conjunto de valores que representa se puede contar, es decir, se puede establecer una relación uno a uno entre sus elementos y el conjunto de los números naturales. Esta relación es de orden porque se puede establecer un predecesor y un sucesor para cada valor del tipo ordinal. Esta característica será importante en la implementación de ciertas estructuras de control (Case, For).

Los tipos estructurados permiten almacenar un a colección de datos, a diferencia de los simples que almacenan un único dato

Variable , objeto de un programa cuyo valor puede “variar”, es una celda de memoria donde se almacena información.
Tiene dos atributos o características propias : un identificador y un tipo;
Es importante que el identificador o nombre de las variables informe acerca de su contenido, esto hace más fácil la comprensión de un algoritmo. Se deben establecer criterios para la elección de los identificadores, adoptaremos que las palabras que lo integran comiencen con mayúscula Ej: SalarioBruto

No confundir identificadores o nombres de variables con valores de tipo cadena, estos últimos están delimitados por apóstrofos y no son equivalentes las mayúsculas y minúsculas.
Ej : ‘PERA’ <> ‘peRa'
Pascal es un lenguaje fuertemente tipeado, todas las variables que se utilizan en un programa, deben ser declaradas especificando su nombre y tipo.

Constante objeto de un programa cuyo valor no cambia.

Operadores

Cada tipo de dato, tiene sus propios operadores, tomando las siguientes variables:
C, D, Z, Y : integer;
A, B, E :real;
Car :char;
Cad :string;

OPERADORES		TIPOS	EJEMPLOS
ARITMETICO	* , / , (MOD , DIV),+ , -	entero, real	A * B / (6 - C) , D MOD 5
ALFANUMERICO	+	char, string	Cad + 'ena' , Car + 'A'
RELACIONAL	< , > , <= , >= , = , <>	todos	2<= E , 'Papa'> Cad , Z +Y * 5 <> 0
LOGICO	NOT , AND , OR	boolean	(2<= E) AND (Cad'> 'Pala')

- ✓ Orden de prioridad : aritmético y alfanumérico – lógico – relacional
- ✓ Dicha prioridad se altera utilizando paréntesis
- ✓ Si en una expresión el orden de precedencia de los operadores es el mismo, se resuelve de izquierda a derecha.

Existen funciones definidas por el lenguaje que facilitan los cálculos, a continuación se presentan alguna de las más utilizadas en Pascal.

Nombre	**Tipo del argumento**	**Resultado**	**Tipo del resultado**
Abs(x)	Real o Integer	el valor absoluto del argumento	Real o Integer
Frac(x)	Real	la parte decimal del argumento	Real
Int(x)	Real	la parte entera del argumento	Real
Round(x)	Real o Integer	el entero más próximo al argumento	Integer
Sqr(x)	Real o Integer	el cuadrado del argumento	Real
Sqrt(x)	Real o Integer	la raíz cuadrada del argumento	Real
Trunc(x)	Real	la parte entera del argumento	Integer
Pi	no posee	3.1415926535897932385	Real
Upcase(x)	char	la mayúscula del argumento si éste es una letra minúscula, sino devuelve el mismo caracter	char
Odd(x)	entero	el valor lógico True si el argumento de la función es impar y False si es par	boolean
Random [(n)]	entero (opcional)	devuelve un número aleatorio. Sin argumento el número aleatorio real entre 0 y 1. Con argumento el número aleatorio entero entre 0 y n -1	Real o Integer
Length (S)	S: cadena de caracteres	la longitud lógica de S	byte

4.Estructura de un Programa

Un programa en Pascal consta de un encabezamiento y un bloque dividido en secciones; las primeras son declarativas, y la última ejecutable. Todas las secciones excepto la última pueden estar vacías.

PROGRAM <identificador>; {encabezamiento del programa} USES <identificadores>; CONST <definiciones de constantes>; TYPE <declaración de tipos de datos def. por el usuario>; VAR <declaración de variables>; <definiciones de procedimientos y funciones>;	**D E C L A R A T I V A**
BEGIN <sentencias> END.	**EJECUTABLE**

4.1.Descripción de la sección Declarativa

4.1.1.USES declara las unidades (módulos compuestos por subprogramas compilados independientemente). Permite al programa utilizar cualquier subprograma incluido en las unidades que declara 'usar'.

4.1.2.CONST declara las constantes, identificándolas con un nombre.

```
CONST
  n1 = v1;  {n1, n2 identificadores}
  n2 = v2;  {v1, v2 números, caracteres, cadenas, expresiones o identificadores de constantes}
```

4.1.3.TYPE en esta sección se declaran los tipos de datos definidos por el programador.

```
TYPE
  i1 = t1;  {i1, i2 identificadores de los nuevos tipos}
  i2 = t2;  {t1, t2 indican la estructura del tipo que se está definiendo }
```

4.1.4.VAR en esta sección es obligatorio declarar todas las variables que utiliza el programa. Debe especificarse el nombre de la variable y su tipo, donde el nombre es un identificador y el tipo puede haber sido declarado en la sección de tipos previamente.

```
VAR
 nombre1 : tipo1;
 nombre2 : tipo2;
    ...
 nombre3, nombre4 : tipo3; {cuando distintas variables tienen el mismo tipo}
```

4.1.5.FUNCTION Y PROCEDURE son las palabras reservadas que identifican a las funciones y procedimientos que se declaran y desarrollan (condición necesaria para poder utilizarlos). Estos subprogramas son módulos de programación autónomas que pueden ser invocadas (o activadas) desde la sección de sentencias ejecutables o por otros procedimientos o funciones.
Las funciones estándar o predefinidas no se declaran.

4.2..Sentencias de la sección Ejecutable

Una variable recibe valor de dos formas **asignación y lectura**, en ambos casos si tiene un valor almacenado previamente lo pierde. Su contenido se puede visualizar mediante **escritura**

Acción → Asignación
Acción → Lectura
Acción → Escritura

Estas son las acciones básicas que constituyen los "pasos de un algoritmo". Las sentencias o instrucciones de un programa desencadenan las respectivas acciones

4.2.1. ASIGNACION

Ejemplos : A **:=** B * C
Cad **:=** ´casa´

VARIABLE **:=** <expresión>

El valor situado a la derecha se almacena en la variable de la izquierda, o sea que almacenamos un dato en la posición de memoria asociada al nombre de esa variable.
El tipo de la expresión debe ser igual al de la variable, salvo dos excepciones :

- La variable es real y la expresión es entera
- La variable es cadena y la expresión es carácter

La variable de la izquierda pierde el valor que almacena, al recibir otro por asignación.
Las variables que figuran a la derecha de la asignación no pierden su valor.

4.2.2LECTURA

Readln (lista de variables separadas por comas)
{lee los datos del teclado almacenando en la lista de variables según el orden en que se ingresan}

Los datos que alimentan las variables de la lista, son tipeados en el teclado, deben estar separados por blancos (o comas).

<u>Importante:</u>No tiene sentido asignar un valor a una variable inmediatamente antes o después de haber leído sobre la misma, ya que lo pierde quedando el correspondiente a la última operación

4.2.3.ESCRITURA

Write(lista de salida)
{escribe los elementos de la lista y queda posicionado en la misma línea}

Writeln(lista de salida)
{escribe los elementos de la lista y queda posicionado al comienzo de la línea siguiente}

Escritura con formato

Es posible establecer, para cada elemento de la lista de salida, el *ancho* del campo donde se escribe. Basta colocar después del elemento el carácter **':'** y la cantidad de columnas del campo de escritura.

Writeln(elemento: *Ancho*);

Ancho es un entero que indica la cantidad de columnas del campo en el que se escribe el elemento. Es utilizado para enteros, cadena, caracteres.

Para los números reales se debe especificar la cantidad de *decimales* deseada .

Writeln(*elemento real* : A*ncho: Decimales*);

Ancho es un entero que especifica la cantidad total de dígitos del número real (contando el signo, parte entera, punto decimal y dígitos decimales)

Decimales es un entero que especifica la cantidad de dígitos decimales con los que se escribirá el número real.

Ejemplos:

A: real; C: integer;

A:= 3.456 ;
C:=10 ;
Write(A: 8:2) ;
Write(C:10) ;
Write(A2:4) ;

S A L I D A

C O L U M N A S

1	2	3	4	5	6	7	8	9	10
				3	.	4	6		
								1	0
3	.	4	5	6	0				

Si no se especifica formato en los números reales, se visualizan en notación científica.

5. Errores de programa

Sintácticos provienen de no respetar la sintaxis del lenguaje, por ejemplo

Writeln (A ; es el resultado),

Semánticos provienen de la lógica de la solución, por ejemplo si se desea promediar tres valores

A + B+ C / 3

6.Ejemplos de problemas y su solución mediante programas Pascal

1.-Conociendo el largo y ancho de un rectángulo, informar el perímetro y la superficie

```
Program Uno;
Var
   Largo, Ancho, Per, Sup : real;
Begin
Writeln('Ingrese largo y ancho de un rectángulo`);
Readln (Largo, Ancho);
Per:= 2*(Largo + Ancho);
Sup:= Largo * Ancho;
Writeln('El perímetro es ', Per :8:2, 'y la superficie es ',Sup:8:2);
End.
```

2.-Conociendo la cantidad de artículos y el importe total de una compra, informar el precio unitario

```
Program Dos;
Var
   Importe, PrecioUnitario: real;
   Cantidad: byte;
Begin
Writeln('Ingrese la cantidad de unidades que compró');  Readln(Cantidad);
Writeln('Ingrese el importe que pagó');  Readln(Importe);
PrecioUnitario := Importe/Cantidad;
Writeln ('El  precio unitario es ', PrecioUnitario  )
End.
```

3.-A partir de dos puntos en el plano, calcular e informar la distancia entre ambos.

```
Program Tres;
Var
   X1 , Y1 , X2 , Y2 : real;
Begin
Writeln('Ingrese coordenadas del primer punto'); Readln (X1, Y1);
Writeln('Ingrese coordenadas del segundo punto'); Readln (X2, Y2);
Writeln('La distancia entre ambos puntos es', sqrt( sqr( X2 - X1 ) + sqr( Y2 - Y1 ) ));
End.
```

Ejercitación (abarca los siguientes temas)

- Tipos de datos, Variables y constantes , Expresiones
- Entrada, salida y asignación
- Prueba de escritorio

Para los siguientes problemas desarrollar un programa Pascal .
Codificar la parte declarativa con constantes, variables y tipos adecuados, eligiendo identificadores representativos

Ej 1) En una pinturería informan que para obtener el color "gris mara" se debe mezclar 9.5 litros de Negro y 4.5 l de Blanco. Se quiere asesorar a un cliente con las cantidades que necesita mezclar para obtener el color si:

a) Tiene N litros de pintura Blanca. ¿Cuánto debe comprar de pintura Negra? .
b) Tiene M litros de pintura Negra. ¿Cuánto debe comprar de pintura Blanca?.

Ej 2) Sea la siguiente sucesión: $a_n = a_1 + 3\ (n-1)$, indica que es el n-ésimo término , y $a_1=1$. Se pide:

a) ¿Cuál es el K-ésimo término, siendo K un valor ingresado por teclado?
b) ¿Cuál es la diferencia entre los términos k y (k+1)?

Ej 3) Dada una cantidad X de horas trabajadas, calcular e imprimir el sueldo bruto y neto de un empleado. Considerar la paga de $200 la hora, un descuento del 11% previsional y el 5% para cobertura médica.

Ej 4) Dada las siguientes declaraciones en Pascal

```
Const
    LimInf = 10;
    LimSup = 255;
Var
    X,Y : real;
    N,M : integer;
    Cadena: string;
    Car : char;
    Mayus, Cumple, Ok : boolean;
```

Analizar si son sintáctica y semánticamente correctas, o si son redundantes, las siguientes sentencias:	Evaluar cuál es el resultado de cada asignación si Cumple=True, N=20,M=1,X=5.8,Y=12.7 y car='B'

a) Mayus := upcase(car)=car;
b) Cumple := (X<=LimInf) and (X>=LimSup);
c) Mayus := 'A' = car or car = ´B' or TRUE;
d) Cumple :=length(cadena)>LimSup;
e) M:= N / LimInf;
f) OK:=LimInf<M<LimSup;
g) Ok:=Odd (n * (n - 1))
	a) Ok:=Not cumple or (N>LimInf) and (X<>Y);
	b) X:= N / 3 + M*Y;
	c) Mayus:= Cumple or (car='S') and (odd(N));
	d) N:= trunc (X)+LimInf ;

Ej 5) Dado un terreno de 10 x 30 se quiere embaldosar una superficie de N x M. Calcular e informar:

a) Qué superficie representa del total del terreno.
b) Si cada baldosa es de 0.25 x 0.25. ¿Cuántas se deben comprar para hacer el trabajo?

Ej 6) Hacer un programa para cada ítem donde: Dado un entero N de más de 3 cifras

a) hallar la cifra que se encuentra en posición de las centenas.
b) quitarle las 3 últimas cifras

Ej 7) Se dispone de dos mazos de cartas españolas. Cada carta tiene los siguientes atributos:

- Color del mazo (Rojo, Azul)
- Palo (O,C,E,B)
- Numero(1..12)

Considerando que una carta está representada en tres variables, una por cada atributo, se pide expresar las siguientes situaciones con operadores relacionales y lógicos.

a) Carta AS de copa de cualquier mazo
b) Carta figura del mazo rojo, cualquier palo
c) Dos cartas iguales de diferente mazo
d) Dos cartas consecutivas del mismo palo sin importar el mazo
e) Dos cartas suman doce, de distinto palo y mazo
f) Dos cartas de igual número, en el mismo mazo o del mismo palo
g) Dos cartas de distinto número en el mismo mazo y palo
h) La primera carta mayor que la segunda (el mazo rojo tiene más peso que el azul, cuando coinciden en el mazo, el palo determina supremacía en el orden en que se describieron, por último define el número)