Capitulo 2: Estructura de un programa

1. Algoritmo – Procesador
2. Estructura de un programa (Declaraciones y Sentencias ejecutables)
3. Errores sintácticos y semánticos
4. Ejemplos con estructura secuencial

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 qué resultados se quieren obtener y cuáles 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 número que obtengo es igual a N)
d. Describir el algoritmo en un lenguaje apropiado (por ej: Pascal).

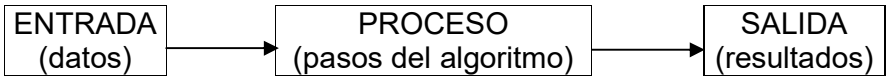
El lenguaje en el que se describe un algoritmo depende del procesador que lo va a ejecutar.

Procesador, entidad capaz de comprender un enunciado y ejecutarlo

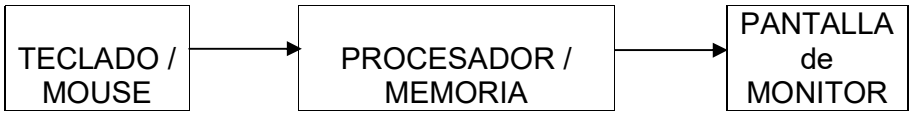
Una computadora es un procesador capaz de captar datos, operarlos y mostrar resultados

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. Estructura de un Programa: Un programa se compone de:

☞ **Sección declarativa:** donde se describen características de los objetos que utiliza el programa.

☞ **Sección ejecutable**: donde se describen las acciones que el programa realizará, estas sentencias 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

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.
El siguiente cuadro presenta la estructura básica, luego se incorporaran otras secciones.

PROGRAM <identificador>; {encabezamiento del programa} CONST <definiciones de constantes>; VAR <declaración de variables>;	**DECLARATIVA**
BEGIN <sentencias> END.	**EJECUTABLE**

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: program, begin, etc.

Identificador: es el nombre que le asignamos a un objeto (programa, constante, variable), debe comenzar con una letra y a continuación letras o dígitos o el carácter '_'.
Ej: Program *Determina_Triangulo*
No hay distinción entre mayúsculas y minúsculas.
Ej: Determina_Triangulo = determina_TRIANGULO
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.

Constante objeto de un programa cuyo valor no cambia.
Ej: 57, -32.79
Pueden tener un identificador asociado y se declaran en la sección
Const
Porcentaje= 15;
cada vez que aparezca Porcentaje en el código se sustituye por 15, la ventaja es que si se cambia el valor de la constante por ejemplo
Porcentaje=23, el código se actualiza automáticamente.

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
Pascal es un lenguaje fuertemente tipeado, todas las variables que se utilizan en un programa, deben ser declaradas especificando su nombre y tipo.

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 (verdadero y falso). Los dos primeros tienen a su vez varias posibilidades de rango y precisión. El programador puede declarar otros tipos para describir sus datos.
Se analizaran en una primera etapa los tipos numéricos entero y real, los otros tipos se verán más adelante.

REALES	***SINGLE***	*1.5E-45 a 3.4E38*
	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*

2.1.- Sección declarativa

Lo visto hasta ahora permite codificar la sección declarativa de un programa, de acuerdo a la estructura presentada anteriormente se especifican la secciones como sigue

CONST {opcional, se declara las constantes, identificándolas con un nombre}
id1 = v1; {id1, id2 identificadores}
id2 = v2; {v1, v2 valores}
…….

VAR {obligatorio declarar todas las variables que utiliza el programa}.
VAR
id1 : tipo1;
id2 : tipo2;
id3, id4 : tipo3; {cuando distintas variables tienen el mismo tipo}

Debe especificarse el nombre de la variable y su tipo, donde el nombre es un identificador y el tipo puede ser estándar o declarado previamente por el programador.

Para los siguientes incisos codificar la parte declarativa de un programa eligiendo identificadores y valores de constantes, identificadores y tipos de variables

1.- constante que represente el límite de velocidad en una autopista igual a 120 km/h.

2.- variable para almacenar el peso de una persona

3.- variable para almacenar la edad de una persona

4.- variable para almacenar la cantidad de titulares de una tarjeta de crédito

5.- variable para almacenar una suma de dinero adeudada

6.-variable/s para almacenar la hora (ej. 12:30:10 , 12.502777)

2.2.-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

Operadores

Cada tipo de dato, tiene sus propios operadores con las cuales se construyen expresiones, tomando las siguientes variables:
L, N, M, P : integer;
A, B, E :real;

OPERADORES ARITMETICOS	TIPOS de OPERANDOS	TIPO de RESULTADO	EJEMPLOS de EXPRESIONES
* , / , + , -	entero, real	entero, real	A * B / (6 - L) , N + 5
DIV (cociente entero) MOD (resto entero)	entero	entero	P MOD 2 , M DIV N

Orden de prioridad :

1. * , / , DIV , MOD
2. + , -

- ✓ 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, por ejemplo si se desea elevar al cuadrado un valor X, en vez de calcular X*X se invoca la función Sqrt (X).
Al final del capítulo se presenta una tabla de las funciones más utilizadas en Pascal.

☞ ASIGNACION

Es una operación que permite almacenar un valor en una variable

VARIABLE **:=** <expresión>

Ejemplos :

```
A := 3;
B := A * 10;
C := A + B;
```

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 que:

- La variable es real y la expresión es entera

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.

☞ LECTURA

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 .

Importante: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

☞ ESCRITURA

Write(lista de salida)

{escribe los elementos de la lista en la pantalla 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;                    S A L I D A
                                          C O L U M N A S
A:= 3.456  ;
C:=10 ;                    1 2 3 4 5 6 7 8 9 10
Write(A: 8:2) ;                     3 . 4 6
Write(C:10) ;                                 1 0
Write(A2:4) ;              3 . 4 5 6 0
```

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

3. 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

4. Ejemplos de programas Pascal. Determinar que problemática resuelven.

4.1.-

```
Program Uno;
Var
   Largo, Ancho, Diag, Sup : real;
Begin
Writeln('Ingrese largo y ancho de un rectángulo`);
Readln (Largo, Ancho);
Sup:= Largo * Ancho;
Writeln('la superficie es ',Sup);
Diag:= sqrt( sqr(Largo) + sqr(Ancho));
Writeln('la diagonal es ', Diag);
End.
```

4.2.-

```
Program Dos;
Const
   Mazo=40;
Var
   Jugadores, Restan, Tocan : byte;
Begin
Writeln('Ingrese cant. de jugadores); Readln (Jugadores);
Tocan:= Mazo DIV Jugadores;
Restan:= Mazo MOD Jugadores;
Writeln('cada jugador recibe ', Tocan, 'cartas');
Writeln('sobran ', Restan, 'cartas');
End.
```

Para los programas anteriores realizar las siguientes modificaciones:

7.- En el programa *Uno* agregar el cálculo del perímetro.

8.- En el programa Dos considerar que la cantidad de cartas del mazo es variable.

Para los siguientes problemas desarrollar un programa Pascal

9.-A partir del peso inicial y final que registró una persona en un tratamiento para adelgazar, calcular e informar el porcentaje que perdió con respecto al peso inicial.

10.-Realizar un algoritmo que permita calcular el volumen de un cilindro. Se ingresarán por teclado la altura del cuerpo y el radio de la base.

11.-Escribir un programa para calcular la velocidad y el consumo por kilómetro de un automóvil. Ingresar por teclado la distancia (en kilómetros), el tiempo (en horas) y la cantidad de gasoil empleado.

Fórmulas :
Velocidad = kilómetros / horas
consumo por km = litros / kilómetros

12.-Calcular e imprimir el sueldo bruto y neto de un empleado. Considerar la paga de $8 la hora, un descuento del 11% previsional y el 5% para cobertura médica. La cantidad de horas trabajadas es un dato de entrada.

13.- Un supermercado otorga puntos para canjear por premios, por las compras que realiza en tres rubros. Otorga 1punto cada $3 en alimentos,$2 en limpieza y $5 en otros. Ingresar los tres importes e informar los puntos obtenidos. (si en algún rubro no realizó compras, dicho importe es cero)

14.-Leer dos enteros, intercambiar el contenido de ambas variables y mostrarlas por pantalla.

☞ Funciones aritméticas

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