TEORIA 1

TEMAS de la CLASE

1. Concepto Informática
2. Etapas de resolución de un problema por computadora
3. Concepto de Algoritmo
4. Concepto de Programa
5. Concepto de Dato y Tipo de Dato
6. Clasificación de Tipos de Datos Simples definidos por el lenguaje (Entero y Real)
7. Declaración de Datos en Pascal
8. Estructura general de un programa en Pascal
9. Ejercitación

Concepto de Informática

La Informática es la ciencia que estudia el análisis y resolución de problemas utilizando computadoras.

Ciencia se relaciona con una metodología fundamentada y racional para el estudio y resolución de los problemas.

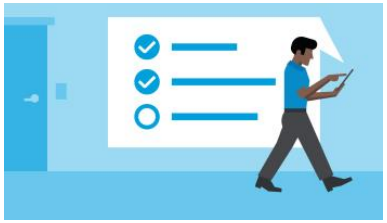
La ***resolución de problemas*** aplicaciones en áreas muy diferentes tales como biología, comercio, control industrial, administración, robótica, educación, arquitectura, etc.

Computadora máquina digital y sincrónica, con capacidad de cálculo numérico y lógico y de comunicación con el mundo exterior. Ayuda al hombre a realizar tareas repetitivas en menor tiempo y con mayor exactitud.

Etapas de resolución de un problema por computadora
Del Problema a la Solución

Cuando se resuelve un problema del mundo real utilizando una computadora es necesario atravesar una serie de etapas.

Etapas de resolución de un problema por computadora
Del Problema a la Solución



1er etapa: se sintetizan los requerimientos del problema y se simplifica el contexto y los datos a utilizar por el programa en la computadora.

Problema del Mundo Real

Análisis

Modelo

Etapas de resolución de un problema por computadora
Del Problema a la Solución

2da etapa: la descomposición funcional nos ayudará a reducir la complejidad, a distribuir el trabajo y en el futuro a reutilizar los módulos. Algoritmos.

Problema del Mundo Real

Análisis

Diseño

Algoritmos

Solución Modularizada

Etapas de resolución de un problema por computadora
Del Problema a la Solución

3er etapa: escribir algoritmos en un lenguaje de programación y elegir la representación de los datos.

Problema del Mundo Real

Análisis

Diseño

Implementación

Programa

Etapas de resolución de un problema por computadora
Del Problema a la Solución

4ta etapa: verificar que el programa conduce al resultado deseado, utilizando datos representativos del problema real

Problema del Mundo Real

Análisis

Diseño

Implementación

Verificación

¿Qué puede ocurrir?

Etapa de Análisis del Problema

- En una primera etapa, se analiza el problema en su contexto del mundo real. Deben obtenerse los **requerimientos** del usuario. El resultado del análisis del problema es un modelo preciso del ambiente del problema y del objetivo a resolver.

Etapa de Diseño de la solución

- Suponiendo que el problema es computable, a partir del modelo se debe diseñar una solución. En el paradigma procedural, esta etapa involucra entre otras tareas la modularización del problema, considerando la descomposición del mismo y los datos necesarios para cumplir su objetivo.
- Esta etapa involucra la especificación de los algoritmos:
- Cada uno de los módulos del sistema diseñado tiene una función que podemos traducir en un algoritmo (que puede no ser único). La elección del algoritmo adecuado para la función del módulo es muy importante para la eficiencia posterior del sistema de software.

Etapa de Implementación de la solución

- Esta etapa involucra la escritura de los programas. Los algoritmos definidos en la etapa de diseño se convierten en programas escritos en un lenguaje de programación concreto.

Etapa de Verificación de la solución

- Una vez que se tienen los programas escritos y depurados de errores de sintaxis, se debe verificar que su ejecución conduce al resultado deseado, utilizando datos representativos del problema real.

En resumen:

Análisis y Diseño

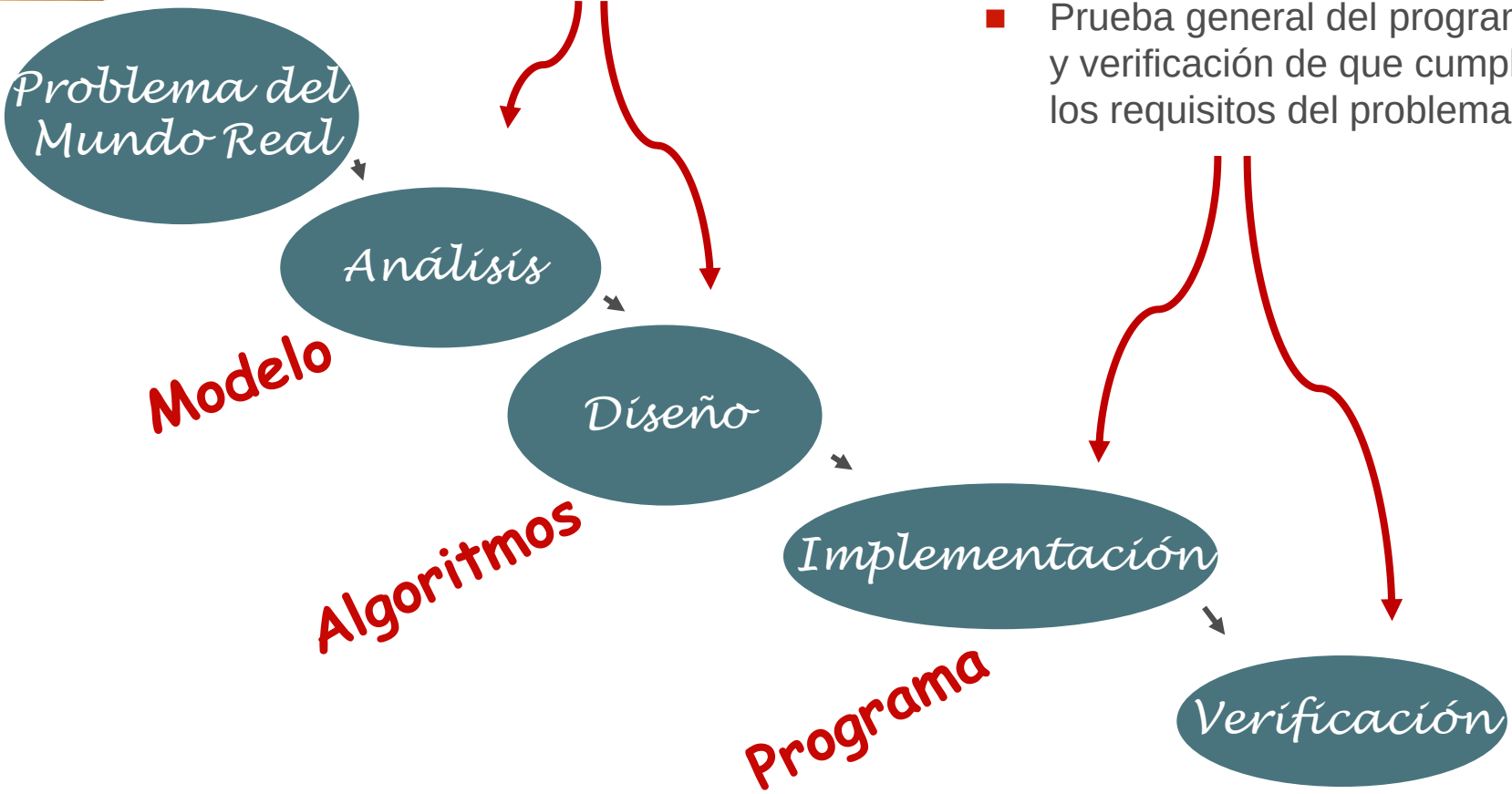
(*NO dependen del lenguaje*)

- Entender el problema
- Modelizarlo.
- Modularizar.
- Escribir los algoritmos.

Implementación

(*depende del lenguaje*)

- Codificación de los algoritmos en un lenguaje de programación.
- Prueba en ejecución de cada módulo.
- Prueba general del programa y verificación de que cumple los requisitos del problema.



Concepto de Algoritmo

Definiremos **algoritmo** como la **especificación rigurosa** de la secuencia de pasos (instrucciones) a realizar sobre un **autómata** para alcanzar un resultado deseado en un **tiempo finito**.

✓***Especificación rigurosa*** significa que debemos expresar un algoritmo en forma clara y unívoca.

✓Alcanzar el resultado en ***tiempo finito*** significa que suponemos que un algoritmo ***comienza y termina***. *Por lo tanto,* el número de instrucciones debe ser también finito.

Si el autómata es una computadora, tendremos que escribir el algoritmo en un lenguaje "entendible" y ejecutable por la máquina.

PROGRAMA

Concepto de Programa

Definiremos **programa** como el conjunto de instrucciones u órdenes **ejecutables** sobre una **computadora**, que permite cumplir con una función específica (dichas órdenes están expresadas en un lenguaje de programación concreto).

- ✓Normalmente los programas alcanzan su función objetivo en un **tiempo finito**.
- ✓ Los **programas de aplicación** constituyen el verdadero valor que da utilidad a las computadoras (programas WEB, de administración, cálculo, comunicaciones, control industrial, sistemas expertos etc.).
- ✓ Los **programas** se escriben en un **lenguaje de programación** siguiendo un conjunto de reglas sintácticas y semánticas.

Escribir un programa exige:

✓***Elegir la representación*** adecuada de los datos del problema**.**

✓***Elegir el lenguaje de programación*** a utilizar, según el problema y la máquina a emplear.

✓***Definir el conjunto de instrucciones*** (en el lenguaje elegido) cuya ejecución ordenada conduce a la solución.

Programa = Instrucciones + Datos

Programación Imperativa

El modelo que siguen los lenguajes de programación para **DEFINIR** y **OPERAR** la información, permite asociarlos a un paradigma de programación particular.

En esta primera parte del curso trabajaremos bajo el paradigma imperativo/procedural.

Utilizaremos el lenguaje de programación: **PASCAL**

Programa = Instrucciones + Datos

Las ***instrucciones*** (***acciones)*** representan las operaciones que ejecutará la computadora al interpretar el programa.

Los ***datos*** son los valores de información de los que se necesita disponer y, en ocasiones transformar, para ejecutar la función del programa.

Se escriben en un lenguaje de programación determinado

PASCAL

Cada lenguaje de programación tiene los propios

Concepto de Dato

Un **Dato** es una representación de un objeto del mundo real. Los datos permiten modelizar los aspectos del problema que se quieren resolver mediante un programa ejecutable en una computadora.

Datos constantes: datos que no cambian durante la ejecución del programa.

Datos variables: datos que durante la ejecución del programa pueden cambiar.

En PASCAL cada dato debe tener asociado un tipo de dato

Definición de Tipo de Dato

Un *tipo de dato* es una clase de objetos de datos ligados a un conjunto de operaciones para crearlos y manipularlos.

Cada tipo de dato se caracteriza por presentar:

- ✓ Un rango de valores posibles
- ✓ Un conjunto de operaciones realizables sobre ese tipo
- ✓ Una representación interna

¿Para que sirven?

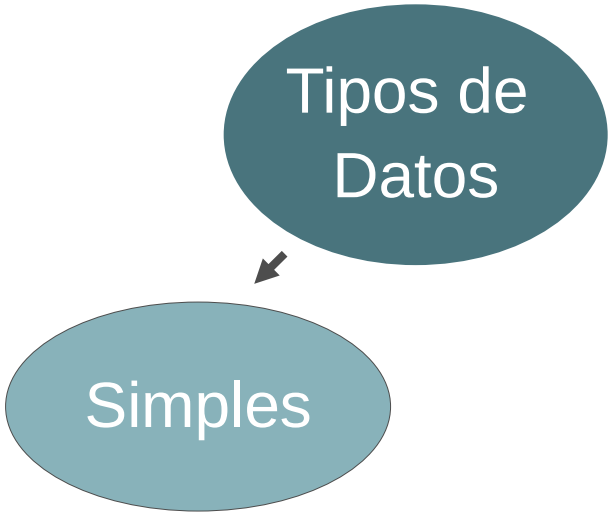
¿Qué valores puedo usar?

¿Qué operaciones puedo aplicar?

¿Cuánta memoria utiliza?

Clasificación de Tipos de Datos

Una primera clasificación



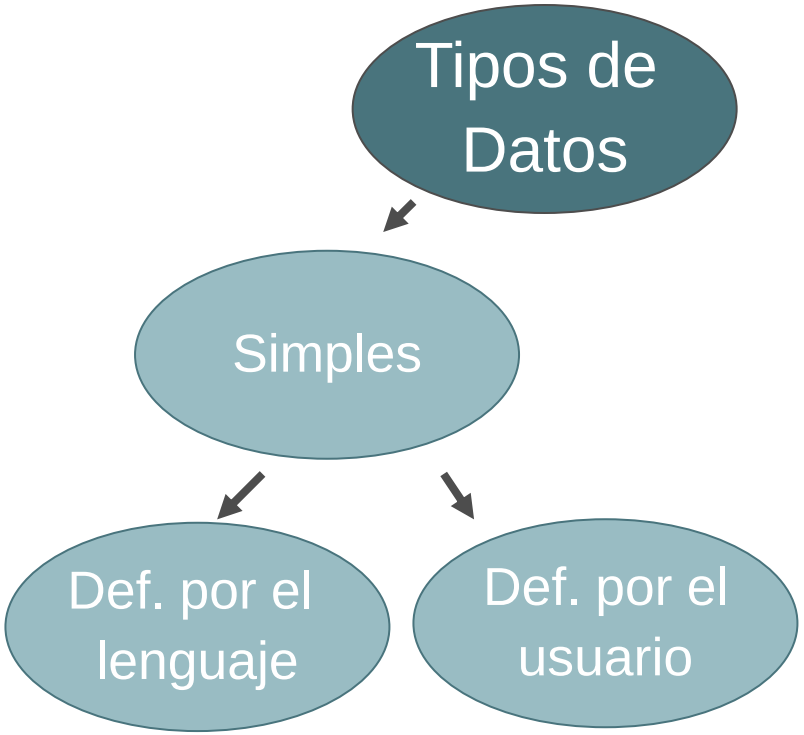
Los tipos de datos simples son aquellos que toman un único valor, en un momento determinado, de todos los permitidos para ese tipo.

Clasificación de Tipos de Datos Simples

A su vez, los tipos simples, pueden clasificarse en:

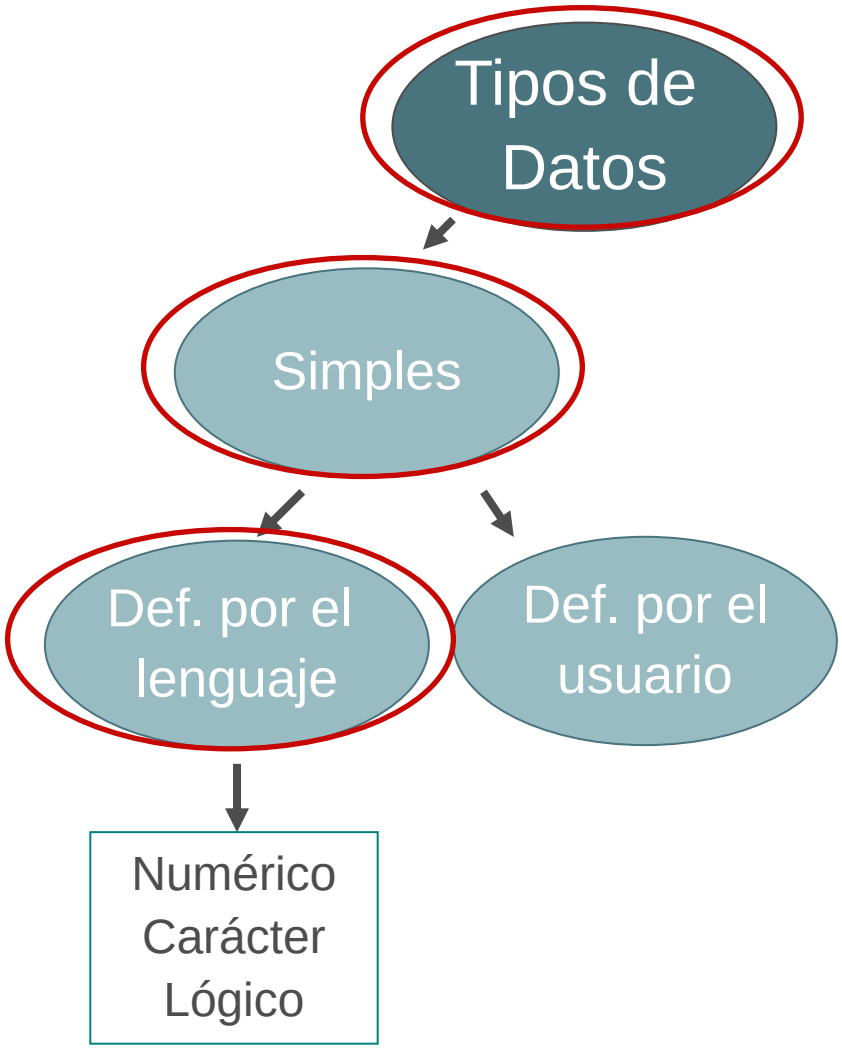
- Tipos de datos definidos por el lenguaje (primitivos o estándar) son provistos por el lenguaje y tanto la representación como sus operaciones y valores son reservadas al mismo.

- Tipos definidos por el usuario, permiten definir nuevos tipos de datos a partir de los tipos primitivos.



Clasificación de Tipos de Datos Simples Definidos por el lenguaje

Comenzaremos presentando los tipos simples y definidos por el lenguaje



Para cada tipo se presentarán:

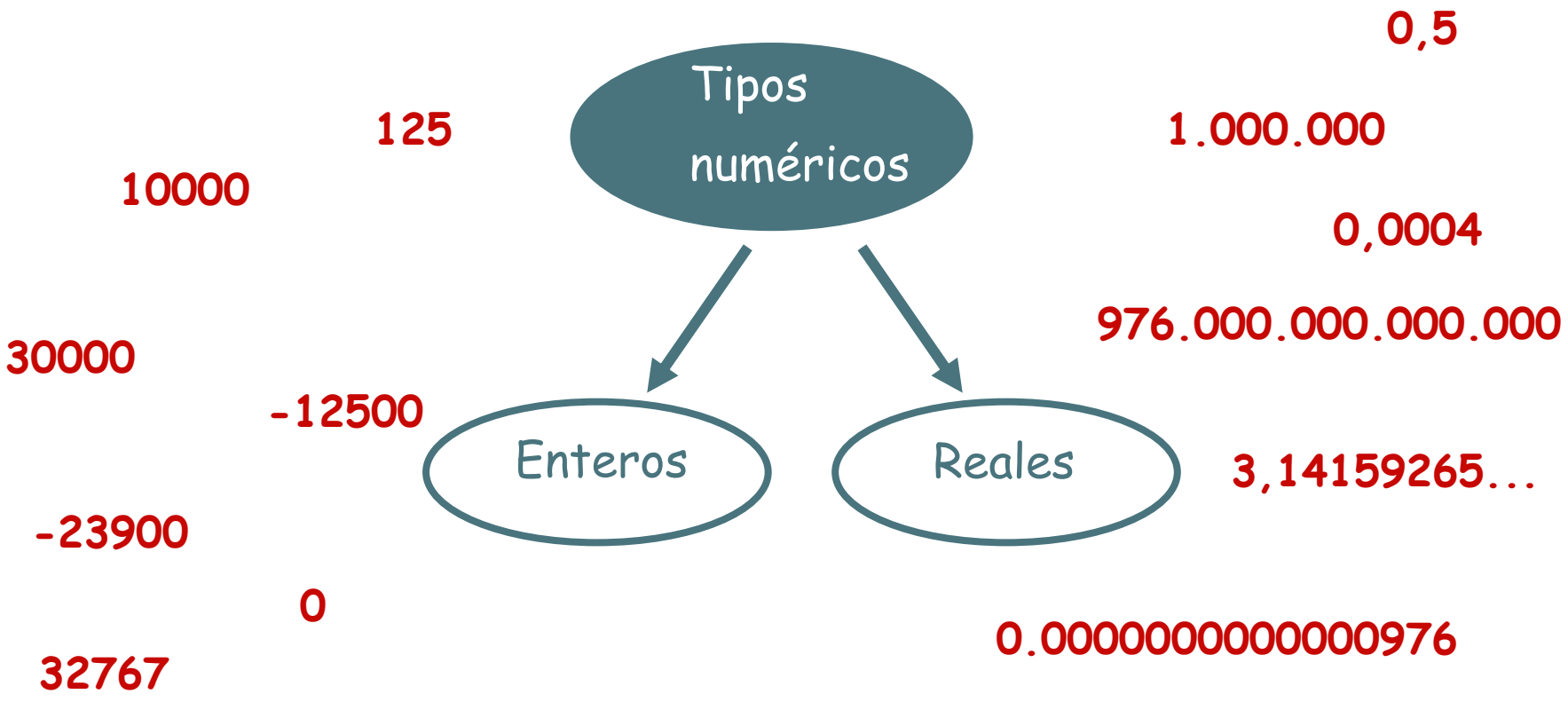
¿Valores posibles?

¿Operaciones?

¿Representación en memoria?

Tipo de dato numérico

El **tipo de dato numérico** permite representar el conjunto de los valores numéricos que puede referirse a los números enteros o a los números reales:



Tipo de Dato numérico: ENTERO

El tipo de dato entero es un tipo de dato simple y ordinal.

Los elementos son valores enteros del tipo: ....., -3, -2, -1, 0, 1, 2, 3, ...

Dado que una computadora tiene memoria finita, la cantidad de valores enteros que se pueden representar sobre ella son finitos, por esto se deduce que existe un número entero máximo y un número entero mínimo.

Hay sistemas de representación numérica que utilizan 16 dígitos binarios (bits) para almacenar en memoria cada número entero, permitiendo un rango de valores enteros entre -2^{15} y $(+2^{15} - 1)$. Entonces podrán representarse los enteros entre -32768 y +32767

La ocupación en memoria es de 2 bytes

Tipo de Dato numérico: REAL

El tipo de dato real es una clase de dato numérico que permite representar números decimales. Es un tipo de dato simple.

✓ El tipo de dato real tiene una representación finita de los números reales.

✓ Se utiliza la notación exponencial o científica, utilizada por cualquier tipo de calculadora, y consiste en definir cada número como una *mantisa* (parte decimal) y un *exponente* (posición de la coma).

976.000.000.000.000 -> 9,76 x 10^{14}

0.000000000000976 -> 9,76 x 10^{-14}

La ocupación en memoria es de 6 bytes

Operaciones del tipo de dato numérico

Operaciones aritméticas

Las operaciones válidas para el tipo de dato numérico son:

- ✓**suma** (+)
- ✓**resta** (-)
- ✓**multiplicación** (*)
- ✓**división real** (/)
- ✓**división entera** (div)
- ✓**módulo** (mod)

Ejemplos	Resultado
3+4	7
2,5 / 2	1,25
10 div 3	3
10 mod 3	1 (devuelve el resto de la división)

En general:

Operador	Tipo de Operando	Tipo de resultado
Suma (+)	Entero ó Real	Entero ó Real
Resta (-)	Entero ó Real	Entero ó Real
Multiplicación (*)	Entero ó Real	Entero ó Real
División (/)	Real	Real
Div	Entero	Entero
Mod	Entero	Entero

Orden de precedencia de las operaciones con números

Las expresiones que tienen dos o más operandos requieren reglas matemáticas que permitan determinar el orden de las operaciones.

El orden de precedencia para la resolución de la expresión matemática es:

En primer lugar: ***** y ***/***

Luego: **+** y **-**

Por último: **div** y **mod**

- 6 + 8 * 5 = 6 + 40 = 46
- (6 + 8) * 5 = 14 * 5 = 70
- 5 * 2 + 7 + 4 * 3 = 10 + 7 + 12 = 17 + 12 = 29
- 5 * (2 + 7 + 4) * 3 = 5 * 13 * 3 = 195
- 5 + 10 mod 4 = 3
- 10 div 2 * 3 = 1
- (5 * 4) div 4 = 5

Se pueden utilizar paréntesis para alterar el orden de precedencia

Operaciones del tipo de dato numérico

- Operación de asignación (:=)

- Operaciones de comparación

Además de los operadores matemáticos mencionados, el tipo de dato numérico posee operadores relacionales que permiten comparar valores.

- ✓ **igualdad** (=),
- ✓ **desigualdad** (<>)
- ✓ **orden** (<, <=, >, >=)

Su **resultado** es Verdadero o Falso

- 0 < 7 (verdadero)
- 5 <= 7 (verdadero)
- 5 > 7 (falso)
- 5 <> 7 (verdadero)
- 5 >= 2 (verdadero)
- Pensemos: 5.0 = 5 ???

Tipo de dato lógico

El **tipo de dato lógico** permite representar datos que pueden tomar solamente uno de dos valores. Este tipo de dato se conoce también como tipo de dato boolean. Es un tipo de dato simple y ordinal.

Valores permitidos

- **verdadero** (*true*)
- **falso** (*false*)

Se utiliza en situaciones donde se representan dos alternativas de una condición.

¿Juan es mayor que María?

¿Edad de Juan?

¿Edad de María?

Una variable de tipo lógico ocupa 1 byte de memoria

Operaciones con el dato lógico

Las operaciones permitidas son:

- **asignación** *(:=)*
- **negación** (*not*)
- **conjunción** (*and*)
- **disyunción** (*or*)

El resultado de estas operaciones es el correspondiente a las conocidas tablas de verdad.

Existe un orden de precedencia para los operadores lógicos:

1. **operador *or***
2. **operador *and***
3. **operador *not***

Ejemplos

- **(18 < 14)** ***falso***
- **(25 / 2.5 = 10)** ***verdadero***
- **(28 > 13)** ***verdadero***
- ***not* (8 > 3)** ***falso***
- **(17 > 2)** ***and* (19 = 33)** ***falso***
- **(17>4)** ***or* (19=33)** ***verdadero***

Tipo de Dato Caracter

El **tipo de dato carácter** es un tipo de dato simple y ordinal.

Un dato de tipo carácter contiene solo un carácter.

Los caracteres que reconocen las computadoras no son estándar. Sin embargo, este conjunto de valores se normalizó, entre otros, por un estándar llamado ASCII, el cual permite establecer un **orden de precedencia** entre los mismos.

Valores permitidos

- ✓ **Caracteres especiales**:'!', '#', '$', '%', ...
- ✓ **Dígitos**: '0', '1', '2', ..., '8', '9'
- ✓ **Letras mayúsculas**: 'A', 'B', 'C', ..., 'Y', 'Z'
- ✓ **Letras minúsculas**: 'a', 'b', 'c', ..., 'y', 'z'

Una variable de tipo carácter ocupa 1 byte de memoria

Tipo de Dato Caracter

Cada elemento del código ASCII tiene asociado un valor entero entre 1 y 255 que permite establecer el orden de precedencia

Caracteres de control ASCII

DEC	HEX	Simbolo ASCII	
00	00h	NULL	(carácter nulo)
01	01h	SOH	(inicio encabezado)
02	02h	STX	(inicio texto)
03	03h	ETX	(fin de texto)
04	04h	EOT	(fin transmisión)
05	05h	ENQ	(enquiry)
06	06h	ACK	(acknowledgement)
07	07h	BEL	(timbre)
08	08h	BS	(retroceso)
09	09h	HT	(tab horizontal)
10	0Ah	LF	(salto de linea)
11	0Bh	VT	(tab vertical)
12	0Ch	FF	(form feed)
13	0Dh	CR	(retorno de carro)
14	0Eh	SO	(shift Out)
15	0Fh	SI	(shift In)
16	10h	DLE	(data link escape)
17	11h	DC1	(device control 1)
18	12h	DC2	(device control 2)
19	13h	DC3	(device control 3)
20	14h	DC4	(device control 4)
21	15h	NAK	(negative acknowle.)
22	16h	SYN	(synchronous idle)
23	17h	ETB	(end of trans. block)
24	18h	CAN	(cancel)
25	19h	EM	(end of medium)
26	1Ah	SUB	(substitute)
27	1Bh	ESC	(escape)
28	1Ch	FS	(file separator)
29	1Dh	GS	(group separator)
30	1Eh	RS	(record separator)
31	1Fh	US	(unit separator)
127	20h	DEL	(delete)

Caracteres ASCII imprimibles

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
32	20h	espacio	64	40h	@	96	60h	`
33	21h	!	65	41h	A	97	61h	a
34	22h	"	66	42h	B	98	62h	b
35	23h	#	67	43h	C	99	63h	c
36	24h	$	68	44h	D	100	64h	d
37	25h	%	69	45h	E	101	65h	e
38	26h	&	70	46h	F	102	66h	f
39	27h	'	71	47h	G	103	67h	g
40	28h	(	72	48h	H	104	68h	h
41	29h	)	73	49h	I	105	69h	i
42	2Ah	*	74	4Ah	J	106	6Ah	j
43	2Bh	+	75	4Bh	K	107	6Bh	k
44	2Ch	,	76	4Ch	L	108	6Ch	l
45	2Dh	-	77	4Dh	M	109	6Dh	m
46	2Eh	.	78	4Eh	N	110	6Eh	n
47	2Fh	/	79	4Fh	O	111	6Fh	o
48	30h	0	80	50h	P	112	70h	p
49	31h	1	81	51h	Q	113	71h	q
50	32h	2	82	52h	R	114	72h	r
51	33h	3	83	53h	S	115	73h	s
52	34h	4	84	54h	T	116	74h	t
53	35h	5	85	55h	U	117	75h	u
54	36h	6	86	56h	V	118	76h	v
55	37h	7	87	57h	W	119	77h	w
56	38h	8	88	58h	X	120	78h	x
57	39h	9	89	59h	Y	121	79h	y
58	3Ah	:	90	5Ah	Z	122	7Ah	z
59	3Bh	;	91	5Bh	[	123	7Bh	{
60	3Ch	<	92	5Ch	\	124	7Ch	|
61	3Dh	=	93	5Dh	]	125	7Dh	}
62	3Eh	>	94	5Eh	^	126	7Eh	~
63	3Fh	?	95	5Fh	_	elCodigoASCII.com.ar		

ASCII extendido

DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo
128	80h	Ç	160	A0h	á	192	C0h	└	224	E0h	Ó
129	81h	ü	161	A1h	í	193	C1h	┴	225	E1h	ß
130	82h	é	162	A2h	ó	194	C2h	┬	226	E2h	Ô
131	83h	â	163	A3h	ú	195	C3h	├	227	E3h	Ò
132	84h	ä	164	A4h	ñ	196	C4h	─	228	E4h	õ
133	85h	à	165	A5h	Ñ	197	C5h	┼	229	E5h	Õ
134	86h	å	166	A6h	ª	198	C6h	ã	230	E6h	µ
135	87h	ç	167	A7h	º	199	C7h	Ã	231	E7h	þ
136	88h	ê	168	A8h	¿	200	C8h	╚	232	E8h	Þ
137	89h	ë	169	A9h	®	201	C9h	╔	233	E9h	Ú
138	8Ah	è	170	AAh	¬	202	CAh	╩	234	EAh	Û
139	8Bh	ï	171	ABh	½	203	CBh	╦	235	EBh	Ù
140	8Ch	î	172	ACh	¼	204	CCh	╠	236	ECh	ý
141	8Dh	ì	173	ADh	¡	205	CDh	═	237	EDh	Ý
142	8Eh	Ä	174	AEh	«	206	CEh	╬	238	EEh	¯
143	8Fh	Å	175	AFh	»	207	CFh	¤	239	EFh	´
144	90h	É	176	B0h	░	208	D0h	ð	240	F0h	≡
145	91h	æ	177	B1h	▒	209	D1h	Ð	241	F1h	±
146	92h	Æ	178	B2h	▓	210	D2h	Ê	242	F2h	‗
147	93h	ô	179	B3h	│	211	D3h	Ë	243	F3h	¾
148	94h	ò	180	B4h	┤	212	D4h	È	244	F4h	¶
149	95h	ò	181	B5h	Á	213	D5h	ı	245	F5h	§
150	96h	û	182	B6h	Â	214	D6h	Í	246	F6h	÷
151	97h	ù	183	B7h	À	215	D7h	Î	247	F7h	¸
152	98h	ÿ	184	B8h	©	216	D8h	Ï	248	F8h	°
153	99h	Ö	185	B9h	╣	217	D9h	┘	249	F9h	¨
154	9Ah	Ü	186	BAh	║	218	DAh	┌	250	FAh	·
155	9Bh	ø	187	BBh	╗	219	DBh	█	251	FBh	¹
156	9Ch	£	188	BCh	╝	220	DCh	▄	252	FCh	³
157	9Dh	Ø	189	BDh	¢	221	DDh	¦	253	FDh	²
158	9Eh	×	190	BEh	¥	222	DEh	Ì	254	FEh	■
159	9Fh	ƒ	191	BFh	┐	223	DFh	▀	255	FFh	

Operaciones con caracteres

- **Asignación (:=)**

- **Comparación (>, <, =…)**

dos valores de tipo carácter se pueden comparar por =, <>, >, <, >=, <=. El resultado es un valor lógico (verdadero/falso) que depende del orden establecido en el código ASCII

Ejemplos: (ver en la tabla el valor entero asociado a cada caracter)

- ('b' = 'B') *falso*
- ('c' < 'Z') *falso*
- ('c' < 'z') *verdadero*
- ('X' > '5') *verdadero*
- (' ' < 'H') *verdadero*
- ('4' = 4) no puede evaluarse
- ('@' > '$') *verdadero (ya que la ubicación de $ es 36 y la del @ es 64)*

¿Como se identifican los datos del programa?

IDENTIFICADORES

Para identificar los datos de un programa (constante / variable) se utilizan nombres descriptivos. Esa identificación permite conocer su dirección real en la memoria y el valor que contiene.

En Pascal, los identificadores están formados por letras, dígitos en cualquier orden y algunos símbolos especiales (excepto el primer carácter que **debe ser una letra**).

flores1

papeles

Total_cuadras

pasos

cantidad

Es obligatorio declarar cada uno de los datos que utiliza el programa

Declaración de constantes y variables en Pascal

En **Pascal**, las constantes deben ser declaradas antes de ser usadas.

Siguiendo su notación, la declaración de constantes **en Pascal** se realiza mediante la palabra clave **const**, de la forma:

```
const  nombre = valor
```

donde nombre es el identificador que representa el nombre de la constante. El tipo de dato de la constante queda definido implícitamente por el valor que se le asigna.

```
const
   N = 25          {N se asume entero }
   pi = 3.1416     {pi se asume  real }
```

Declaración de variables

Las variables en Pascal se declaran utilizando la palabra clave **var**, de la forma:

```
var  variable : tipo;
```

Si hay varias variables del mismo tipo, se pueden declarar separadas por coma y se le asocia el tipo de dato, una sola vez.

Por ejemplo:

```
program ejemplo;

const
  …
var
  numero: integer; car: char;
  resultado1, resultado2: real; cumple: boolean;
begin
  …
end.
```

Declaración de variables numéricas en Pascal

En Pascal, para representar los tipos de datos vistos se cuenta con:

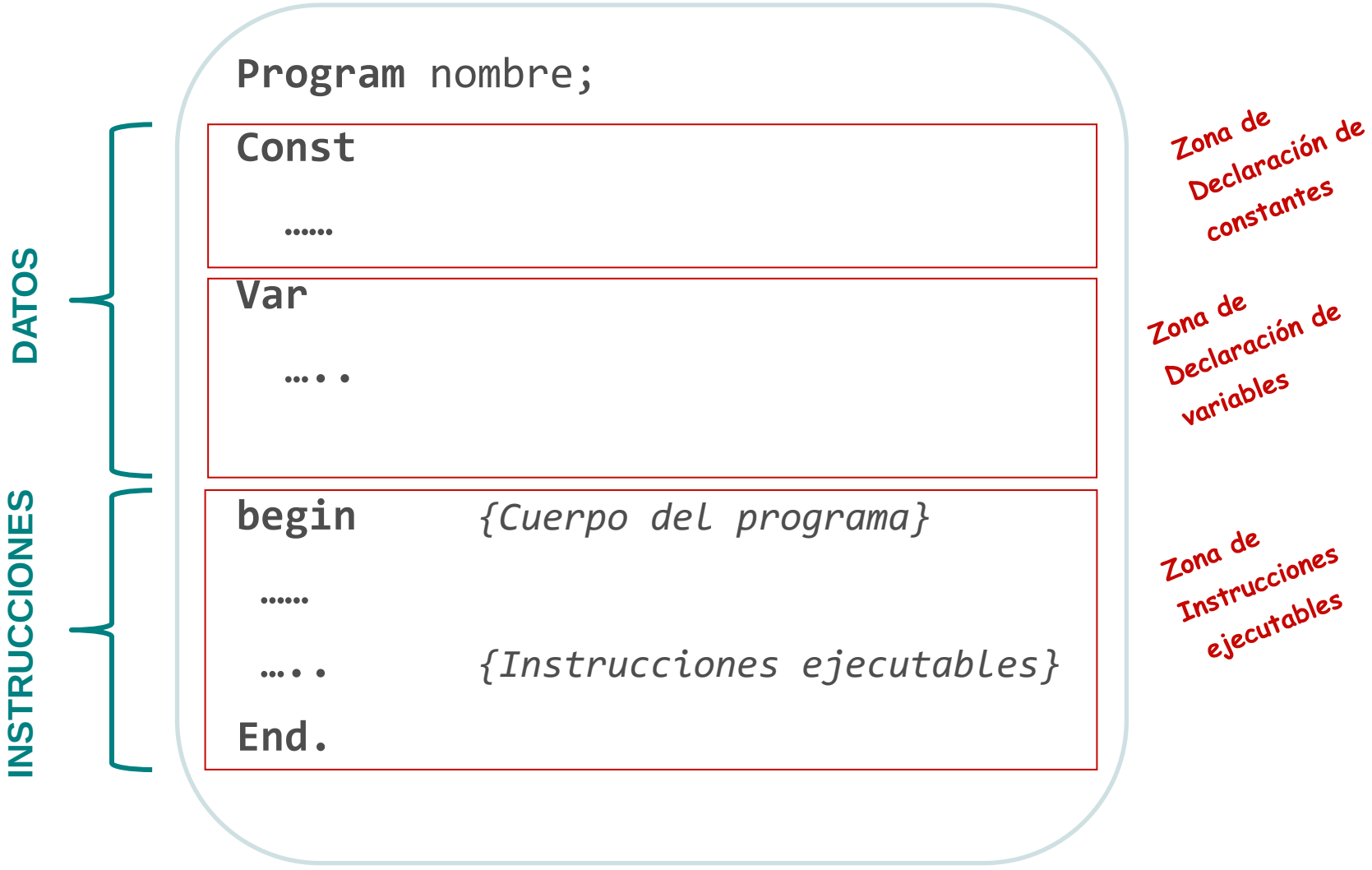
- **integer** ➡ dato numérico entero
- **real** ➡ dato numérico real
- **boolean** ➡ dato lógico
- **char** ➡ dato caracter

```
Var
   cantidad:  integer;     {cantidad puede contener un nro. entero}
   total_cobrado: real;    {total_cobrado puede contener número real}
   exito: boolean;         {exito puede contener un valor logico}
   letra: char;            {letra puede contener un valor del código ASCII}
```

Cuando se asigna el tipo de dato a una variable queda determinado:

Valores posibles / Operaciones permitidas / Representación en memoria

Estructura general de un Programa en Pascal



Ejemplos de programas en Pascal

```
Program uno;
Const
  num = 5;
Var
  cantidad:  integer;
  total_cobrado: real;
Begin  { Cuerpo del programa}
  cantidad := 4;
  total_cobrado := 10,50;
End.
```

```
Program dos;
Const
  max = 10;
 Var
   exito: boolean;
   resultado: boolean;
Begin  { Cuerpo del programa}
  exito := 8 < max;
  resultado := false;
End.
```

```
Program tres;
Const
   fin = '*';
 Var
   resultado: boolean;
   letra1, letra2 : char;
Begin  { Cuerpo del programa}
 letra1:= 'A';
 letra2:= 'm';
 resultado := letra 1 < letra2;
End.
```

Operación de entrada en Pascal: READ

- **Read**

se usa para leer datos (por defecto desde teclado) y asignarlos a las variables correspondientes.

```
Read (a, b, c);
Read (a);
Read (b);
Read (c);
```

```
program ejemplo;
var
  numero: integer; car: char; precio: real;
begin
  read(numero);
  read(char);
  readln(precio)
end.
```

- **También puede utilizarse la instrucción Readln**

Es similar a la instrucción Read pero hace que la siguiente sentencia Read comience leyendo en la siguiente línea.

Operación de salida en Pascal: WRITE

- **Write**

se usa para mostrar el contenido de una variable, por defecto en pantalla. Pueden ser de tipo entero, real, char. Los datos a mostrar si son más de uno deben ir separados por coma.

```
Write (a, b);
Write (a);
Write (b);
```

```
program ejemplo;
var
  numero: integer; car: char; precio: real;
begin
  numero := 100;
  read(char);
  write(numero);
  writeln(char);
end.
```

- **También puede utilizarse la instrucción Writeln**

Es similar a la instrucción Write, pero hace que la siguiente sentencia Write muestre el contenido de la variable en la siguiente línea.

EJERCITACIÓN DE LA CLASE 1

Un centro de deportes quiere procesar la información de los 4 tipos de actividades que ofrece: 1 (Musculación), 2 (Spinning), 3 (Cross Fit) y 4 (Libre).

Para ello, se debe leer y guardar el precio mensual de cada actividad.

Al finalizar, informar el promedio de cuota mensual.