

# PROGRAMACIÓN I-2022

Teoría – Ma. Virginia Ainçhil

**TEMAS  
de la  
CLASE**

- ① Presentación de la materia
- ② Concepto Informática
- ③ Etapas de resolución de un problema por computadora
- ④ Concepto de Algoritmo
- ⑤ Concepto de Programa
- ⑥ Concepto de Datos y Tipos de Datos
- ⑦ Clasificación de Tipos de Datos Simples definidos por el lenguaje
- ⑧ Declaración de Datos en Pascal
- ⑨ Estructura general de un programa en Pascal
- ⑩ Ejercitación

# Docentes

## **Teoría:**

Miércoles y viernes de 8 a 10 hs. (aula 11)

Sábados 12/03, 26/03, 23/04, 7/05 y 28/05 de 9 a 13 hs.  
(aula 11)

Profesores: Ma. Virginia Ainchil - Federico Cristina

**Práctica:** Miércoles y viernes de 10 a 12 hs. (varias aulas,  
distribuidos en comisiones)

JTP: César Estrebou

# Información de la Cátedra

En este curso trabajaremos con el entorno virtual Ideas.

**URL:** <http://ideas.info.unlp.edu.ar>

Deben solicitar inscripción al curso **“Programación I”**.

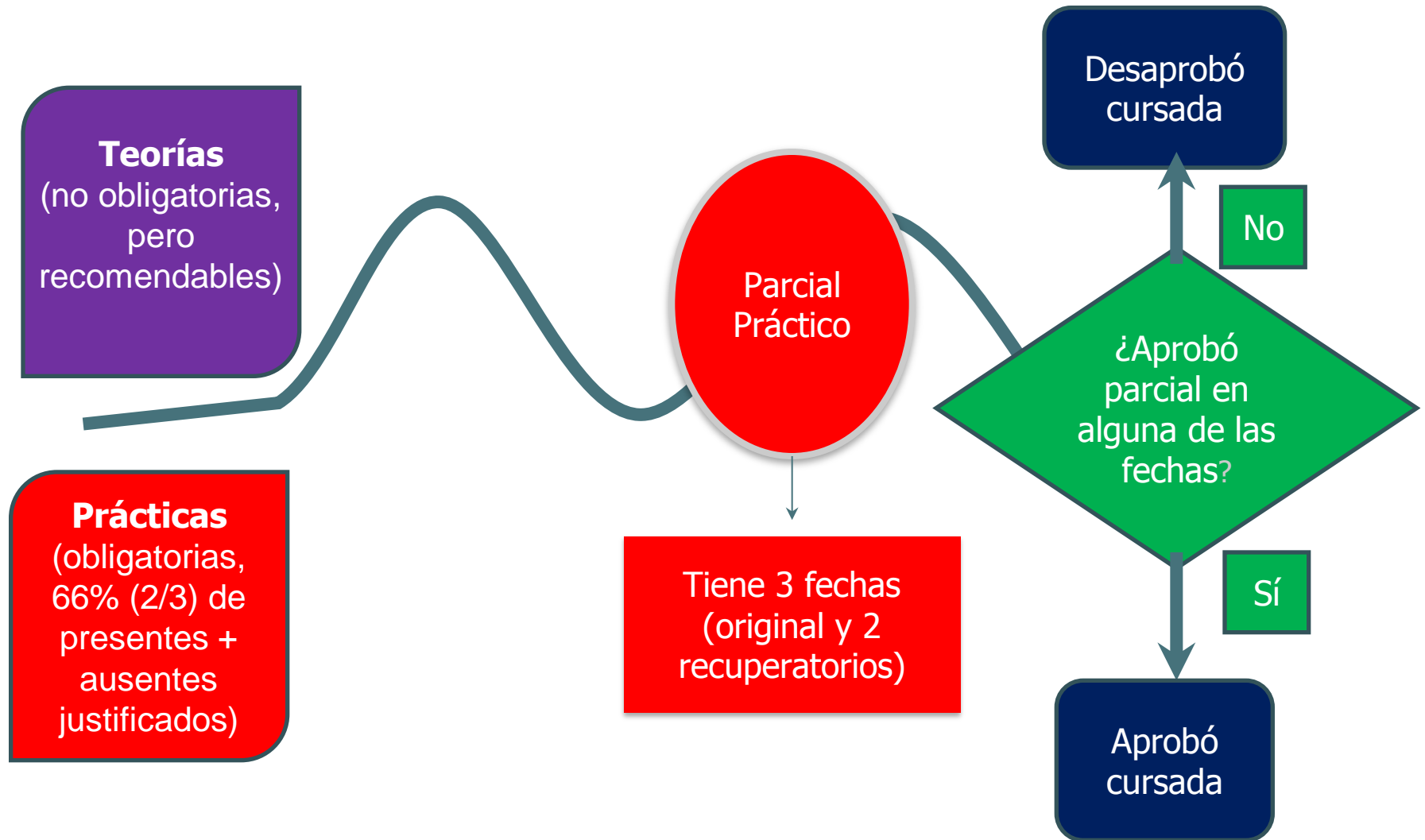
Dicho curso servirá para:

- Comunicarnos vía una mensajería interna
- Descargar las prácticas y materiales de estudio
- Obtener información de interés de la cursada
- Descargar el reglamento y programa

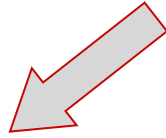
Una vez finalizado el curso, podrán encontrar información en:

Blog de la cátedra: <http://blogs.unlp.edu.ar/progra1/>

# Aprobación de la cursada

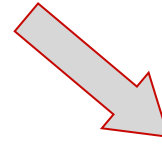


# Aprobación de la materia



## Examen final

- Se toma todos los meses y Uds. deben anotarse según el calendario académico.
- Programación I toma los días martes de la semana de finales (generalmente 8:30 hs.).
- La información se publica en el blog de la Cátedra y en la Cartelera Virtual.



## Régimen de promoción

- Realizar la autoevaluación propuesta en el entorno Ideas y obtener un 6 o más.
- Aprobar el parcial en primera o segunda fecha.
- Aprobar una evaluación breve en el mes de julio con nota 6 o más.

# Régimen de promoción de final

- ✓ Los alumnos que, **habiendo aprobado el parcial práctico (en primera o segunda fecha)**, no obtuvieran el mínimo de 6 en la evaluación breve, podrán rendir un examen recuperatorio, que abarcará los temas de la materia que el Profesor indique.

Los alumnos que aprueben la promoción y **se encuentren inscriptos en el curso bajo esta modalidad**, tendrán registrada su nota al final del curso.

# Examen final

- ✓ **Plazo de validez de la habilitación:** La habilitación para rendir Examen Final, tendrá una validez de cuatro (4) semestres, inmediatos siguientes al del curso realizado.
- ✓ Para rendir los exámenes finales existirán mesas examinadoras integradas por los Profesores de las Áreas/ Asignaturas. Los alumnos **deberán inscribirse en las fechas determinadas**, según el calendario académico.



# Concepto de Informática

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

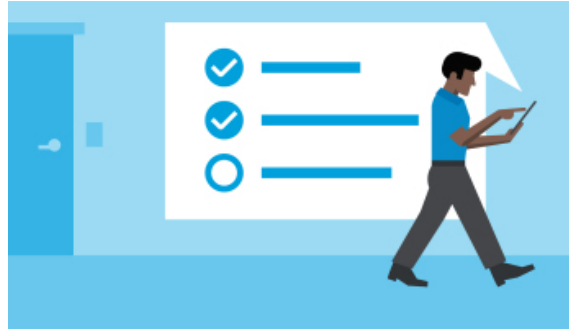
**Ciencia** se relaciona con una metodología fundamentada y racional para el estudio y resolución de los problemas.

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



**Modelo**

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

# 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 re-utilizar 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

Problema  
Solución

Problema del  
Mundo Real

Análisis

Diseño

Implementación

Programa

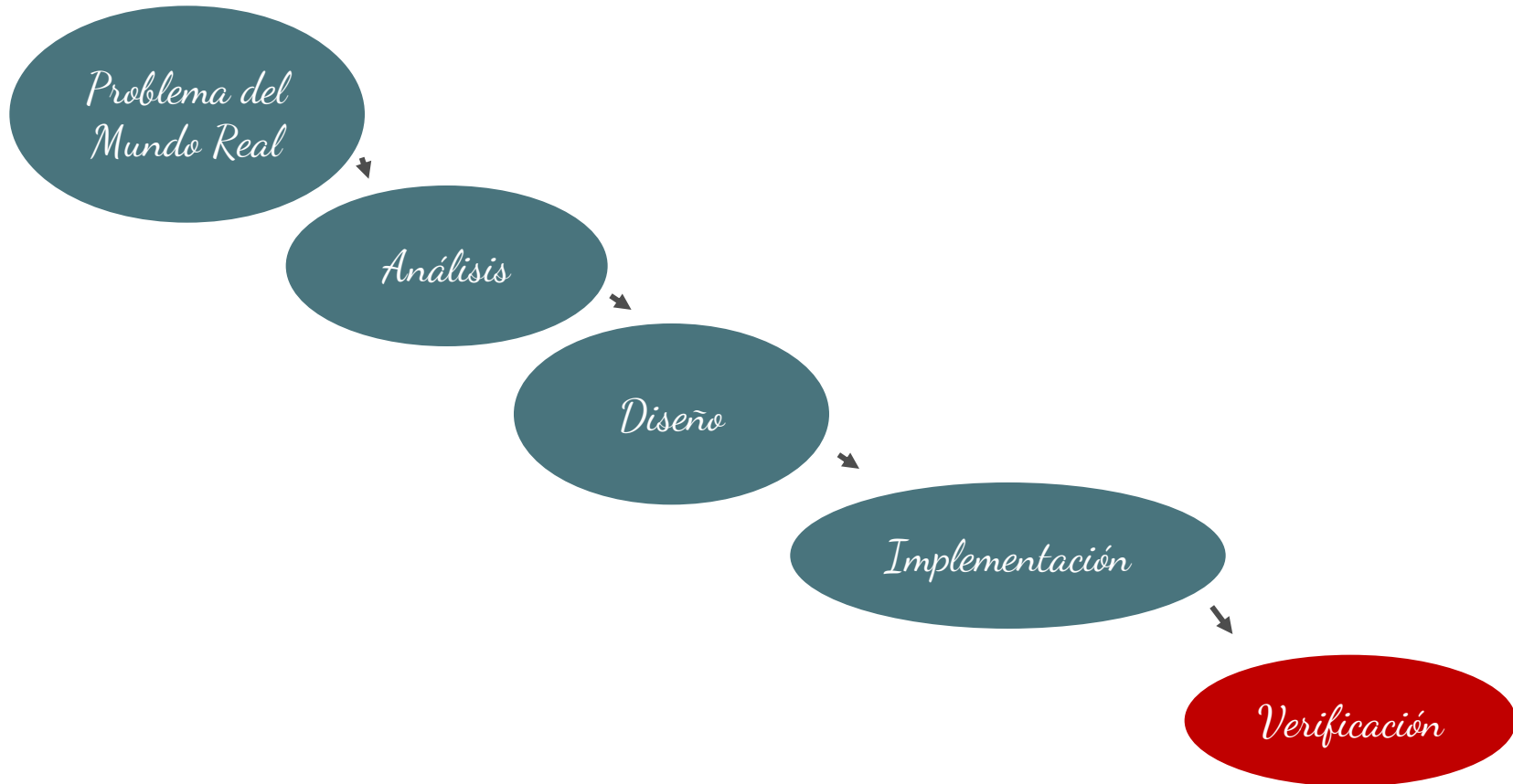
```
<?php·}·>>  
<?php·if(·$this->item->typ  
$this->item->linkparts[·op  
$this->item->linkparts[·va  
else·if(·document·getEleme  
> >> alert(·"<?php·echo·$Te  
}·<?php·}·>>  
<?php·if(·$this->item->typ
```

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

# Etapas de resolución de un problema por computadora

## Del Problema a la Solución

Problema  
Solución



# 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:

Problema  
Solución

## *Análisis y Diseño*

*(NO depende 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.

*Problema del  
Mundo Real*

*Análisis*

*Diseño*

*Implementación*

*Verificación*

**Modelo**

**Algoritmos**

**Programa**

# 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).

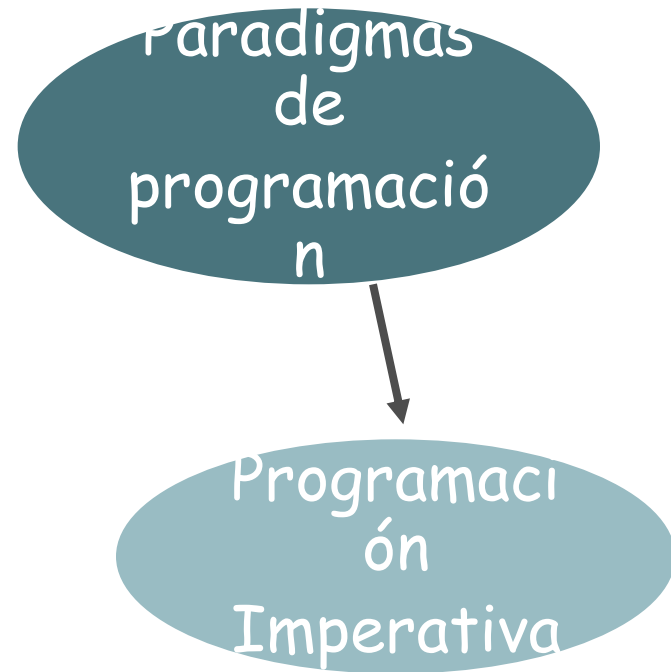
- ✓ Normalmente los programas alcanzan su función objetivo en un **tiempo finito**.
- ✓ Los **programas de aplicación** constituyen el verdadero valor que da utilidad a las computadoras (programas WEB, de administración, cálculo, comunicaciones, control industrial, sistemas expertos etc.).
- ✓ Los **programas**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

# 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.

Lenguaje de programación: *PASCAL*



# 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**

# Programa = Instrucciones + Datos

```
graph TD; A[Programa = Instrucciones + Datos] --> B[Las instrucciones (acciones) representan las operaciones que ejecutará la computadora al interpretar el programa.]; A --> C[Los datos son los valores de información de los que se necesita disponer y en ocasiones transformar para ejecutar la función del programa.]; B --> D[Se escriben en un lenguaje de programación determinado]; C --> E[Cada lenguaje de programación tiene los propios]; D --> F[PASCAL];
```

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

Se escriben en un lenguaje de programación determinado

**PASCAL**

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

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.

## Los tipos de datos se caracterizan por:

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

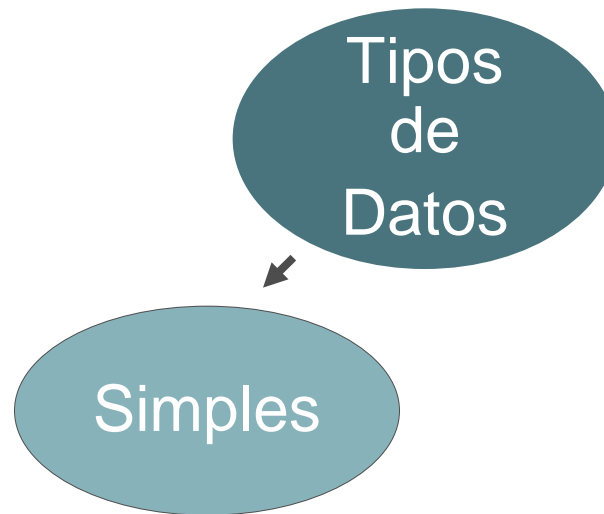
¿Qué valores puedo usar?

¿Qué operaciones puedo aplicar?

¿Cuánta memoria utiliza?

# Clasificación de Tipos de Datos

## Una primera clasificación

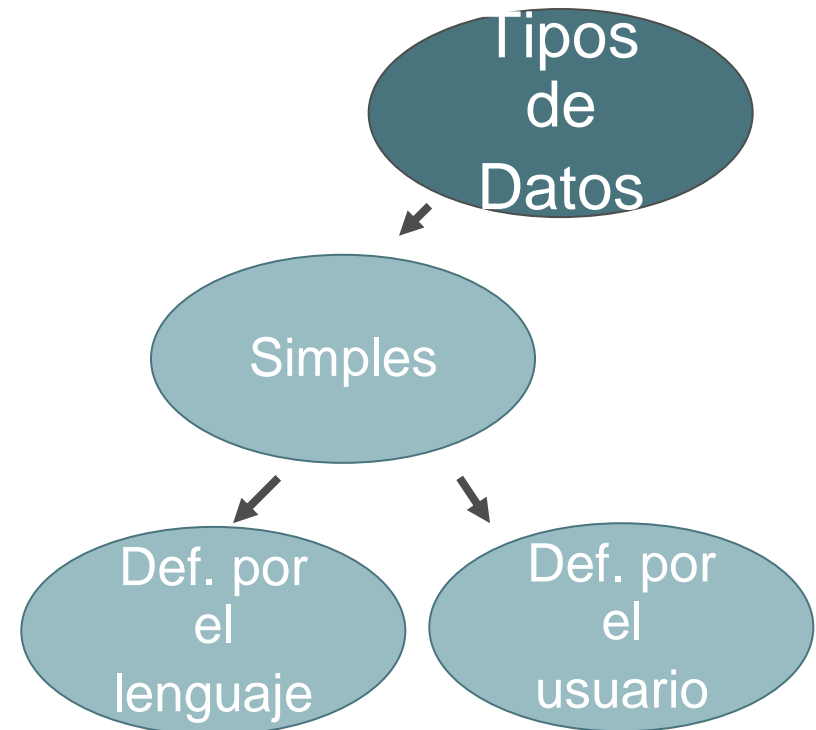


**Los tipos de datos simples** son aquellos que toman un único valor, en un momento determinado, de todos los permitidos para ese tipo.

# Clasificación de Tipos de Datos Simples

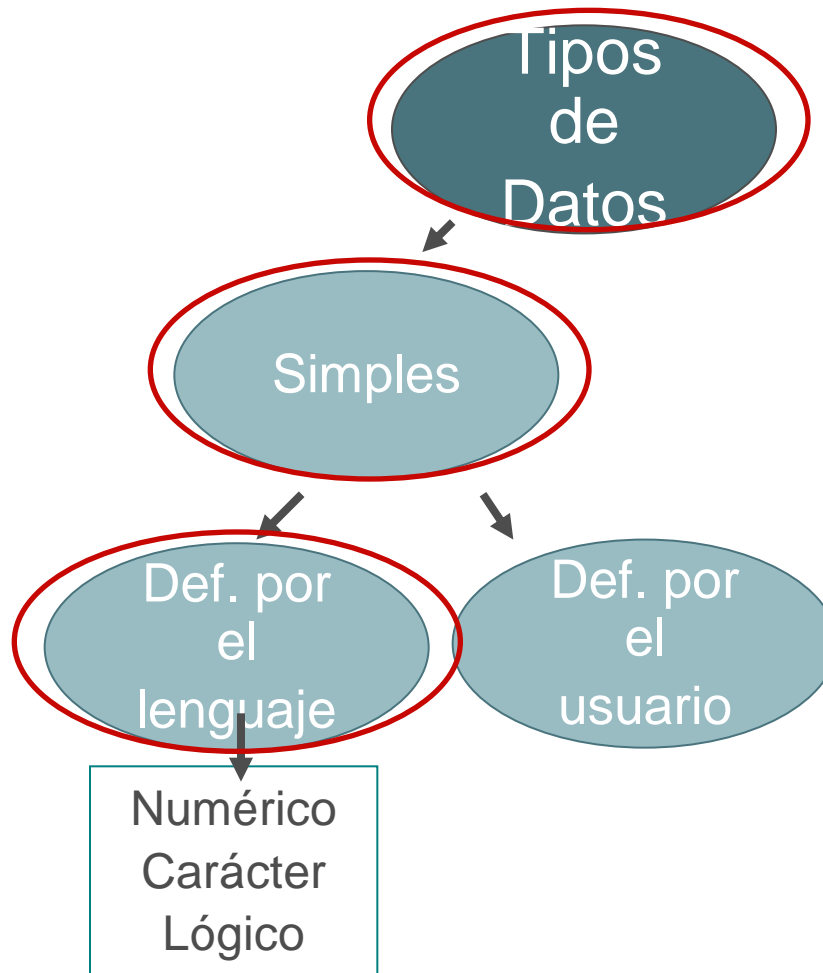
A su vez, los tipos simples, pueden clasificarse en:

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



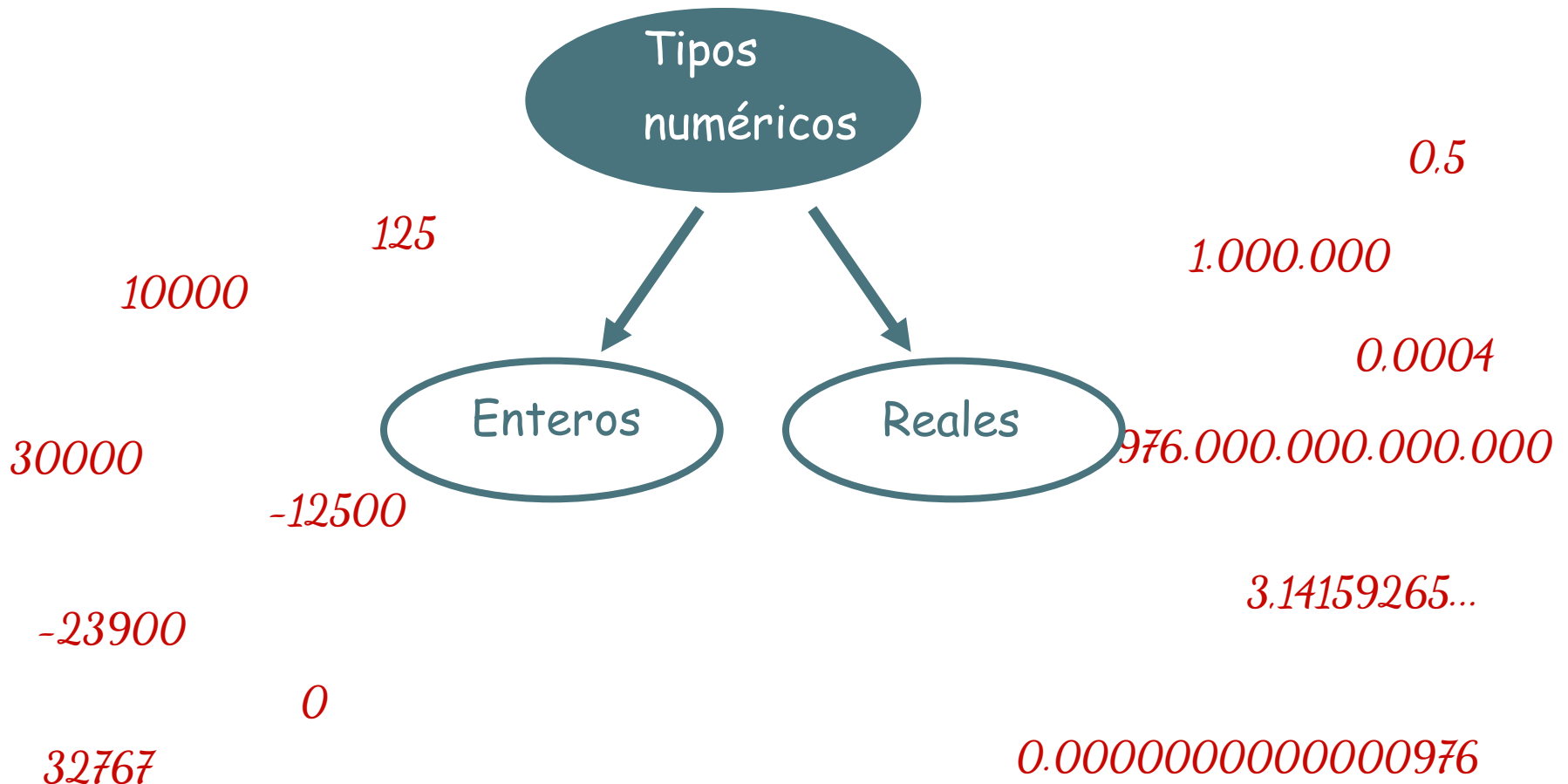
# Clasificación de Tipos de Datos Simples Definidos por el lenguaje

Comenzaremos presentando los tipos simples y definidos por el lenguaje



# 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 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 otro 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  $+2^{15}$ . Entonces podrán representarse los enteros entre -32768 y +32767

Trabajaremos con la idea que la ocupación en memoria es de 2 bytes

Diagrama de valores enteros en memoria (2 bytes):

32767    ~~-50000~~    30000    0    -23900    -12500    ~~100000~~

# 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).

La ocupación en memoria es de 6 bytes

$976.000.000.000.000 \rightarrow 9,76 \times 10^{14}$

$0.00000000000000976 \rightarrow 9,76 \times 10^{-14}$

# Operaciones del tipo de dato numérico

## ■ Operaciones aritméticas

Las operaciones válidas para el tipo de dato numérico son:

- ✓ suma (+),
- ✓ resta (-),
- ✓ multiplicación (\*),
- ✓ división (/),
- ✓ división entera (div)
- ✓ módulo (mod).

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

$$3+4$$

$$2.5/3$$

$$10 \text{ div } 4$$

$$10 \text{ mod } 4$$



# 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, ya conocido, es:

1ro: Operadores  $*$ ,  $/$

2do: Operadores  $+$ ,  $-$

3ero: Operadores **div** y **mod**.

*Se pueden utilizar paréntesis para alterar el orden de precedencia*

$$\cdot 6 + 8 * 5 = 6 + 40 = 46$$

$$\cdot (6 + 8) * 5 = 14 * 5 = 70$$

$$\cdot 5 * 2 + 7 + 4 * 3 = 10 + 7 + 12 = 17 + 12 = 29$$

$$\cdot 5 * (2 + 7 + 4) * 3 = 5 * 13 * 3 = 195$$

# 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 ( $\neq$ )

✓ orden ( $<$ ,  $\leq$ ,  $>$ ,  $\geq$ )

Su **resultado** es Verdadero o Falso

▪ $0 < 7$	Resultado: verdadero
▪ $5 \leq 7$	Resultado: verdadero
▪ $5 > 7$	Resultado: falso
▪ $5 \neq 7$	Resultado: verdadero
▪ $5 \geq 2$	Resultado: verdadero

# ¿Cómo 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:

```
var  
    variable1: tipo A;  
    variable2, variable3: tipoB;
```

# Declaración de variables numéricas en Pascal

En Pascal, para representar los datos numéricos se cuenta con:

- **integer** → dato numérico entero
- **real** → dato numérico real

## Var

```
cantidad: integer; {cantidad puede contener un nro. entero}  
total_cobrado: real; {total_cobrado puede contener número real }
```

Recordar que cuando se asigna el tipo de dato a una variable queda determinado: Valores posibles , Operaciones permitidas y Representación en memoria

# Estructura general de un Programa en Pascal

Program nombre;

Const

.....

Var

....

begin      {Cuerpo del programa}

.....

....      {Instrucciones ejecutables}

End.

Zona de  
Declaración de  
constantes

Zona de  
Declaración de  
variables

Zona de  
Instrucciones  
ejecutables

# Ejemplo de un Programa en Pascal

**Program** nombre;

**Const**

num = 5;

Zona de  
Declaración de  
constantes

**Var**

cantidad: integer;

totalCobrado: real;

Zona de  
Declaración de  
variables

**Begin** { *Cuerpo del programa*}

cantidad := 4;

totalCobrado := 10,50;

**End.**

Zona de  
Instrucciones  
ejecutables



# Ejercicio 1



Escribir un programa que calcule la superficie de un terreno que tiene 15,5 mts. de frente y 60 mts. de largo e informe el resultado.

Algoritmo:

- Calcular superficie
- Mostrar resultado

¿Cómo se calcula la superficie?

¿Cómo se muestra el resultado?

# 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 ('El resultado es: ', b);
```

## ■ 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.

# Resolución del ejercicio 1



Escribir un programa que calcule la superficie de un terreno que tiene 15,5 mts. de frente y 60 mts. de largo e informe el resultado.

**Algoritmo:**

- Calcular superficie
- Mostrar resultado

**Program** superficie;

var sup: real;

**Begin**

sup := 15,5 \* 60;

**write** ('La superficie es: ', sup);

**End.**

# Ejercicio 2



Escribir un programa que calcule la superficie de un terreno cuyas dimensiones se leen de teclado e informe el resultado.

Algoritmo:

- Leer dimensiones
- Calcular superficie
- Mostrar resultado

*¿Como se leen las dimensiones?*

# 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);
```

## ■ **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.

# Resolución del ejercicio 2



Escribir un programa que calcule la superficie de un terreno cuyas dimensiones (frente y fondo) se leen de teclado e informe el resultado.

## Algoritmo:

- Leer dimensiones
- Calcular superficie
- Mostrar resultado

**Program** superficie;

**var** frente, fondo: real;

        sup: real;

**Begin**

**readln** (frente);

**readln** (fondo);

        sup := frente \* fondo;

**write** ('La superficie es: ', sup);

**End.**

# Ejercicio 3



Si se quiere alambrar ese terreno ¿Cuántos metros lineales de alambrado se necesitan?

Modificar el programa anterior, para que también resuelva e informe la cantidad de metros necesarios para alambrarlo

# 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 posibles

- **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*)

Existe un orden de precedencia para los operadores lógicos:

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

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

## 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 lógico en Pascal

**Program** nombre;

**Const**

max = 10;

**Var**

exito: boolean;

resultado: boolean;

**Begin** { *Cuerpo del programa*}

exito := 8 < max;

resultado := false;

.....

**End.**

# Tipo de Dato Carácter

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

**TABLA DE CARACTERES DEL CÓDIGO ASCII**

1	␣	25	↓	49	1	73	I	97	a	121	y	145	æ	169	—	193	±	217	⌈	241	±
2	⦿	26		50	2	74	J	98	b	122	z	146	Æ	170	—	194	⌋	218	⌋	242	±
3	♥	27		51	3	75	K	99	c	123	{	147	Ô	171	—	195	⌋	219	⌋	243	±
4	♦	28	—	52	4	76	L	100	d	124		148	Ö	172	—	196	⌋	220	⌋	244	±
5	♣	29	↔	53	5	77	M	101	e	125	}	149	ò	173	—	197	⌋	221	⌋	245	±
6	♠	30	▲	54	6	78	N	102	f	126	~	150	û	174	«	198	⌋	222	⌋	246	±
7		31	▼	55	7	79	O	103	g	127	⌘	151	ù	175	»	199	⌋	223	⌋	247	±
8		32		56	8	80	P	104	h	128	Ç	152	ÿ	176		200	⌋	224	α	248	±
9		33	!	57	9	81	Q	105	i	129	ü	153	Ö	177		201	⌋	225	β	249	±
10		34	"	58	:	82	R	106	j	130	é	154	Ü	178		202	⌋	226	Γ	250	±
11		35	#	59	;	83	S	107	k	131	â	155	Ç	179		203	⌋	227	π	251	±
12		36	\$	60	<	84	T	108	l	132	ä	156	£	180		204	⌋	228	Σ	252	±
13		37	%	61	=	85	U	109	m	133	à	157	¥	181		205	⌋	229	σ	253	±
14		38	&	62	>	86	V	110	n	134	â	158	℞	182		206	⌋	230	μ	254	±
15		39	'	63	?	87	W	111	o	135	ç	159	f	183		207	⌋	231	τ	255	±
16	▶	40	(	64	@	88	X	112	p	136	ê	160	á	184		208	⌋	232	Φ	PRESIONA LA TECLA	
17		41	)	65	A	89	Y	113	q	137	ë	161	í	185		209	⌋	233	Θ	<b>Alt</b>	
18	‡	42	*	66	B	90	Z	114	r	138	è	162	ó	186		210	⌋	234	Ω	MÁS EL NUMERO	
19	‡‡	43	+	67	C	91	[	115	s	139	ï	163	ú	187		211	⌋	235	δ	CORTESÍA DE:	
20	¶	44	,	68	D	92	\	116	t	140	î	164	ñ	188		212	⌋	236	∞		
21	§	45	-	69	E	93	]	117	u	141	ì	165	Ñ	189		213	⌋	237	φ		
22	—	46	.	70	F	94	^	118	v	142	Ë	166	°	190		214	⌋	238	€		
23	‡	47	/	71	G	95	~	119	w	143	Ä	167	º	191		215	⌋	239	∩		
24	†	48	0	72	H	96	`	120	x	144	É	168	¿	192		216	⌋	240	≡		

# 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:

- ('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)*

# Tipo de dato carácter en Pascal

```
Program nombre;
```

```
Const
```

```
    fin = '*';
```

```
Var
```

```
    resultado: boolean;
```

```
    letra1, letra2 : char;
```

```
Begin  { Cuerpo del programa}
```

```
    letra1:= 'A';
```

```
    letra2:= 'm';
```

```
    resultado := letra 1 < letra2;
```

```
    .....
```

```
End.
```

# Tipos de lenguajes

- ✓ Algunos lenguajes, como Pascal, exigen que se especifique a qué tipo pertenece cada una de las variables.
- ✓ Verifican que el tipo de los datos asignados a esa variable se correspondan con su definición.
- ✓ Esta clase de lenguajes se denomina ***fuertemente tipados*** (strongly typed).

# Tipos de lenguajes

- ✓ Otra clase de lenguajes, determina el tipo acorde al primer valor que se le asigne, se denomina *auto tipados* (**self typed**).
- ✓ Existe una tercera clase de lenguajes que permiten que una variable tome valores de distinto tipo durante la ejecución de un programa. Esta se denomina *dinámicamente tipados* (**dynamically typed**).