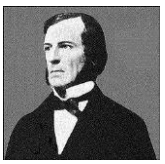
Resolución de Problemas y Algoritmos

Clase 3
Programación en Pascal.
Tipos de datos. Expresiones.



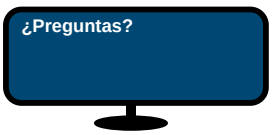
George Boole

Dr. Alejandro J. García
http:// cs.uns.edu.ar /~ajg

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Bahía Blanca - Argentina

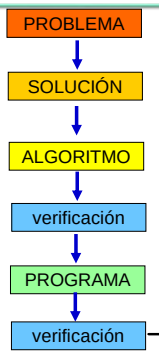
Conceptos de las clases anteriores

1. **Algoritmo.**
   Primitiva.
   Traza.
2. **Lenguaje de programación.**
   Pascal:
   - **Identificadores**
   - **Constantes y variables**
   - **Primitiva de asignación (:=)**
   - **Primitivas read y write**



Tengo una pregunta: ¿Cómo escribo el signo que abre en la consola?

Metodología general propuesta



En una traza de un programa en Pascal se llevará cuenta de los pasos y cambios realizados. Principalmente de cambios en los valores de las variables.

Primitiva de asignación (repaso)

En una asignación: variable := expresión

1) primero se **evalúa** la expresión de derecha y se obtiene un valor,
2) luego se **modifica el valor** de la variable, **perdiéndose el valor anterior.**

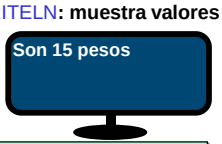
```
PROGRAM Asigna1;
CONST c = 2.3;
VAR a, b: REAL;
BEGIN
  a:= 1.5;
  a:= 2 + 1/2;
  b:= 2 * a + c;
  b:= a * -1;
END.
```

Traza de los valores almacenados en memoria para cada variable:

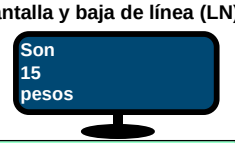
a	b
?	?

Primitivas para mostrar en pantalla

WRITE: **muestra valores en la pantalla**
WRITELN: **muestra valores en pantalla y baja de línea (LN)**




```
valor:=15;
write(´Son ´);
write(valor);
write(´ pesos.´);
```




```
Valor:=15;
writeln(´Son ´);
writeln(valor);
writeln(´ pesos. ´);
```

Observación: en el horario de la práctica se explicarán más detalles sobre estas primitivas (no se lo pierda) ☺

Formateo de salida en pantalla con WRITE

```
PROGRAM Asigna1;
CONST c = 2.3;
VAR a, b: REAL;
BEGIN
  a:= 1.5;
  a:= 2 + 1/2;
  b:= 2 * a + c;
  b:= a * -1;
  writeln(a,b);
  writeln('presione Enter');readln;
END.
```

Si no se especifica ningún formato, muestra reales en notación científica.

a	b
?	?
1.5	?
2.5	?
2.5	7.3
2.5	-2.5

El `readln` final espera un ENTER del usuario, evita que termine el programa y se cierre la consola (asi puedo ver los resultados)

Formateo de salida en pantalla con WRITE

```
PROGRAM Asigna1;
CONST c = 2.3;
VAR a, b: REAL;
BEGIN
  a:= 1.5;
  a:= 2 + 1/2;
  b:= 2 * a + c;
  b:= a * -1;
  writeln(a:10:3,b:20:1);
  writeln('presione Enter');readln;
END.
```

2.500 -2.5
presione Enter

El formato a:10:3 indica que mostrar el valor de a en 10 lugares con 3 decimales.

a	b
?	?
1.5	?
2.5	?
2.5	7.3
2.5	-2.5

Observación: en el horario de práctica se explicarán más detalles sobre estas primitivas (no se lo pierda) ☺

Primitiva de asignación

En una asignación: variable := expresión

1) primero se **evalúa** la expresión de derecha y se obtiene un valor,
2) luego se **modifica el valor** de la variable, **perdiéndose el valor anterior.**

```
PROGRAM Asigna2;
CONST c = 2.3;
VAR a, b: REAL;
BEGIN
  b:= 10;
  a:= 1;
  a:= a + 1;
  a:= a + 1;
  a:= a + 1;
END.
```

Traza de los valores almacenados en memoria para cada variable:

a	b
?	?
?	10
1	10
2	10
3	10
4	10

Primitiva de asignación

En una asignación: variable := expresión

1) primero se **evalúa** la expresión de derecha y se obtiene un valor,
2) luego se **modifica el valor** de la variable, **perdiéndose el valor anterior.**

```
PROGRAM Asigna3;
VAR a: REAL;
BEGIN
  a:= 1;
  a:= a + a;
  a:= a + a;
  a:= a + a;
  a:= a + a;
END.
```

Traza de los valores almacenados :

a
?
1
2
4
8
16

Conceptos: programa – código fuente

Un **programa** de computadoras *(computer program),* es un conjunto organizado de instrucciones que están escritas en un lenguaje de programación, y al ser ejecutadas por una computadora resuelven una tarea específica.

Cada lenguaje de programación nos brinda una manera de organizar las instrucciones de un programa.

Pascal es un lenguaje de programación secuencial e imperativo, pensado para crear programas que serán ejecutados secuencialmente en un solo procesador.

El texto de un programa de computadoras se conoce como **código fuente** *(source code).*

Comentarios en el código fuente

En cualquier lugar de un programa en Pascal, se pueden incluir **comentarios** encerrados entre llaves `{}`. También se puede comentar al final de una línea usando dos barras `//`. Estos comentarios serán ignorados en la ejecución el programa pero son muy útiles para los humanos que trabajan con ese código fuente.

```
PROGRAM Transforma;
{Este programa transforma un valor de temperatura
de la escala Celsius a la escala Fahrenheit.}
VAR cel,fah:REAL; //variables para las temperaturas
BEGIN
write('Ingrese temperatura en Celsius ');
readln(cel);
fah:= cel*9/5 + 32; {aquí realiza transformación}
writeln('En Fahrenheit es: ',fah:10:2);
readln; //  Espera a que el usuario presione ENTER
END.
```


Tipos de datos

- **En Pascal (y en otros lenguajes de programación) las variables deben ser declaradas indicando a que TIPO de DATO pertenecen.**
- **A continuación veremos cuatro tipos de dato que ya están predefinidos en Pascal:**
  - **INTEGER (enteros)**
  - **REAL (reales)**
  - **CHAR (caracteres)**
  - **BOOLEAN (lógico: verdadero o falso)**

Conceptos: tipos de datos

Tipo de Dato: define el conjunto de valores posibles que puede tomar una variable, y las operaciones que pueden aplicarse.

En Pascal los tipos de datos pueden ser:

1. **predefinidos** (ya tienen un conjunto de valores establecido y proveen operaciones para su uso)
2. **definidos por el programador** (el programador define los valores y las operaciones)

Tipo de dato predefinido de Pascal

Nombre: **INTEGER** (entero)

Valores: cualquier número entero entre un mínimo valor y un máximo definido por el compilador usado.

Constante predefinida: **MAXINT** (máximo valor INTEGER).

Operaciones predefinidas: Operadores aritméticos

+ (adición) – (substracción) * (multiplicación)

div (división entera) y **mod** (resto de la división entera)

Los operadores tienen la precedencia usual y los paréntesis () permiten cambiar el orden de evaluación.

Operadores relacionales: =, >, <, <> (distinto), >= (mayor o igual) y <= (menor o igual)

Función predefinida: **SQR** (devuelve el cuadrado (square) de un entero). Ejemplo: SQR(3) = 9. SQR(SQR(3)) =81

Realice una traza y luego pase a la máquina

```
PROGRAM Ejemplo1; {Algunos ejemplos para el tipo entero}
VAR  N1,N2,N3,N4,N5:INTEGER;
       form: INTEGER; {para el "formateo" en pantalla}
BEGIN
    writeln('Máximo entero: ', MAXINT);
    N1 := 2000 mod 2;
    N2 := 2000 div 2 ;
    N3 := SQR(SQR(3));
    N4 := MAXINT;
    form:= 15; //valor para el formateo
    writeln(n1:form, n2:form, n3:form, n4:form);
    N5 := 1+ N4 ;  //¿qué valor toma N5?
    writeln(N5);  //¿por qué será este valor?
END.
```


Tipo de dato predefinido de Pascal

Nombre: **REAL** (real)

Valores: Se pueden expresar con punto decimal (3.5459), o en notación científica: Por ej. $3.5*10^{-3}$ = 0.0035 en Pascal es 3.5E-3 y $1.28*10^{8}$ = 128000000 es 1.28E8

Es un subconjunto de los números reales en dos sentidos: (1) tiene mínimo y máximo, y (2) tiene una "precisión" máxima. No se cumple que "entre dos números reales existe siempre otro número real".

Operaciones predefinidas: +, –, *, / (división)

operadores relacionales: =, >, <, <>, >=, <=

Funciones predefinidas: ver a continuación

Algunas funciones predefinidas para REAL

Funciones trigonométricas: **SIN, COS y TAN**. Dado un valor de un ángulo (en radianes), devuelven su seno, coseno o tangente.
Ejemplos: SIN(0) = 0, COS(0) = 1

Función raíz cuadrada (square root) **SQRT**
Ejemplo: SQRT(4) = 2.0

Función de redondeo ROUND: dado un valor real, devuelve el entero más cercano.
Ejemplos: ROUND(2.9) = 3 ROUND(2.3) = 2

Función truncado TRUNC: dado un valor real, devuelve el entero que resulta de eliminar la parte decimal.
Ejemplos: TRUNC (2.9) = 2 TRUNC(2.3) = 2

Realice una traza y luego pase a la máquina

```
PROGRAM Ejemplo2; {Algunos ejemplos con el tipo real}
VAR  N1,N2,N3:INTEGER;
       R1,R2: REAL;
BEGIN
    R1 :=  20 mod 2; // Todo entero es un real
    R2 := MAXINT + 1; // Los reales tienen un rango más amplio
    N1 := TRUNC(2.5);
    N2 := ROUND(2.5);
    N3 := ROUND(3.5); // ¿Por qué redondea así?
    write(R1:5:2, R2:25:2,N1:5,N2:5,N3:5);
    readln; //mantiene consola abierta
END.
```

El código ASCII

American Standard Code for Information Interchange (ASCII)

Está formado por 256 símbolos, aquí se muestran algunos:

				32		33	!	34	"	35	#	36	$	37	%	38	&	39	'
40	(	41	)	42	*	43	+	44	,	45	-	46	.	47	/	48	0	49	1
50	2	51	3	52	4	53	5	54	6	55	7	56	8	57	9	58	:	59	;
60	<	61	=	62	>	63	?	64	@	65	A	66	B	67	C	68	D	69	E
70	F	71	G	72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U	86	V	87	W	88	X	89	Y
90	Z	91	[	92	\	93	]	94	^	95	_	96	`	97	a	98	b	99	c
100	d	101	e	102	f	103	g	104	h	105	i	106	j	107	k	108	l	109	m
110	n	111	o	112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124	|	125	}	126	~	127		128	Ç	129	ü
130	é	160	á	161	í	162	ó	163	ú	164	ñ	165	Ñ			168	¿		

Tipo de dato predefinido de Pascal

Nombre: **CHAR** (caracter)

Valores: es el conjunto de los 256 caracteres del código **ASCII** (American Standard Code for Information Interchange)

Operaciones predefinidas: (relacionales) =, >, <, <>, >=, <=

Funciones predefinidas: **(ver a continuación)**

¿Cómo se diferencia en Pascal entre una variable cuyo identificador es A y el símbolo ASCII A ?

- Para indicar un **valor** de tipo CHAR, se utilizan las comillas simples. Ej : ´a´, ´? ´, ´ + ´, ´ ´, etc.
- En Pascal: ´**A**´ es un valor del tipo char, pero en cambio **A** es un identificador creado por un programador.

Funciones predefinidas del tipo CHAR

CHR: permite obtener un caracter cualquiera a partir de su código ASCII. Ej: chr(65) = ´A´; chr(33) = ´!´.

ORD: dado un caracter cualquiera, devuelve su código ASCII. Ejemplos: ord(´A´) = 65, ord(´!´) = 33.

SUCC: retorna el siguiente ASCII si es que existe. Ejemplos: succ(´A´) = ´B´ , succ(´0´) = '1'

PRED: retorna el anterior ASCII si es que existe. Ejemplos: pred(´B´) = ´A´, pred(´A´) = '@'

```
PROGRAM Espaniol;
BEGIN
write(chr(160),chr(130),chr(161),chr(162),chr(163));
write(chr(164),chr(165),chr(168));
END.
```


Tipo de dato predefinido de Pascal

Nombre: **BOOLEAN** (Booleano)

Valores: (solamente dos) **true**, **false**.

Obs: en Free Pascal son identificadores reservados

Operadores predefinidos: (identificadores reservados): **and** (conjunción "y"), **or** (disyunción "o") y **not** (negación "no")

Operadores relacionales: =, >, <, <>, >=, <=

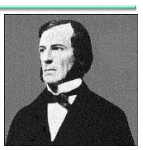
Observe que los operadores relacionales retornan un valor de tipo BOOLEAN (true o false).

Por ejemplo:

5 > 3 retorna true

5 <> 5 retorna false

George Boole (1815-1864)



Matemático y filósofo Inglés. Inventor del "álgebra de Boole", la base de la aritmética computacional moderna. Es considerado uno de los fundadores del campo de las Ciencias de la Computación.

En 1854 publicó "*An Investigation of the Laws of Thought*" donde desarrolla un sistema de reglas que permite expresar, manipular y simplificar, problemas lógicos y filosóficos cuyos argumentos admiten dos estados (verdadero o falso) por procedimientos matemáticos.

Se podría decir que es el padre de las operaciones lógicas, y gracias a su álgebra hoy en día podemos manipular operaciones lógicas.

Conceptos: Expresiones Lógicas

Así como hay infinitos valores numéricos, hay sólo dos valores lógicos (también llamados valores de verdad) : **verdadero** (**true**) y **falso** (**false**)

Los operadores lógicos: **and**, **or** y **not**, reciben valores de verdad y retornan valores de verdad.

Con los operadores lógicos se pueden construir **expresiones lógicas** que retornarán un valor de verdad.

Ej: `(num > 0) and (num < 10) or not (num = 5)`

A diferencia de los operadores numéricos, para definir el resultado de un operador lógico alcanza una tabla (llamada Tabla de Verdad)

Tablas de verdad

Una tabla de verdad para un operador lógico, muestra explícitamente el resultado de cada entrada posible.

Sea A una expresión lógica cualquiera (esto es, su resultado es verdadero o falso), la tabla de verdad de la negación es la siguiente:

A	no A
verdadero	**falso**
falso	**verdadero**

Por ejemplo, A podría representar "es un número par", "es la letra J ", "es un número entre 1 y 100"

Tabla de verdad para la conjunción "y"

A	B	A y B
verdadero	**verdadero**	**verdadero**
verdadero	**falso**	**falso**
falso	**verdadero**	**falso**
falso	**falso**	**falso**

Por ejemplo: para representar las condiciones necesarias para realizar una llamada puedo usar a expresión **"tengo señal" y "tengo saldo"**

Tabla de verdad para la disyunción "o"

A	B	A o B
verdadero	**verdadero**	**verdadero**
verdadero	**falso**	**verdadero**
falso	**verdadero**	**verdadero**
falso	**falso**	**falso**

Por ejemplo, para indicar las condiciones para trasladarme hasta la Universidad puedo utilizar la expresión **"voy caminando" o "voy en colectivo"**

Precedencia de los operadores lógicos

- La precedencia es: **no**, **y**, **o** **(not, and, or)**
- Los paréntesis cambian el orden de evaluación

Compare:

tomo502 o tomo500 y tengo_tarjeta

(tomo502 o tomo500) y tengo_tarjeta

Calcule el resultado de ambas expresiones con:

tomo502=verdadero, t**omo500**=falso, **tengo_tarjeta**=falso

Compare: no A y B con no (A y B)

Pruebe con: **A** = falso **B**= falso

y luego con **A**= verdadero **B**=falso

Precedencia de los operadores en Pascal

Table 12.1: Precedence of operators in Free Pascal

http://www.freepascal.org/docs-html/ref/refch12.html

Operator	Precedence	Category
not	Highest (first)	Unary operators
*** / div mod and**	Second	Multiplying operators
+ - or	Third	Adding operators
= <> < > <= >=	Lowest (Last)	relational operators

Expresiones Lógicas Equivalentes

Dos expresiones lógicas son equivalentes si para todos los casos donde una es verdadera la otra también es verdadera. Ejemplos:

(tomo502 o tomo500) y tengo_tarjeta

es equivalente **a**

(tomo502 y tengo_tarjeta) o (tomo500 y tengo_tarjeta)

no (A o B) no es equivalente a **(no A o no B)**

Observación: con un ejemplo alcanza para mostrar que no es equivalente, sin embargo para mostrar que sí es equivalente hay que mostrar para todos los casos.

Expresiones lógicas equivalentes

¿Por qué son importantes las expresiones equivalentes?

'La misma condición puede escribirse de diferentes maneras, por ejemplo, "*El número entero N tiene un solo dígito*" puede representarse con las siguientes tres expresiones equivalentes:

(N=1) or (N=2) or (N=3) or (N=4) or (N=5) or (N=6) or (N=7) or (N=8) or (N=9) or (N=0) or (N=-1) or (N=-2) or (N=-3) or (N=-4) or (N=-5) or (N=-6) or (N=-7) or (N=-8) or (N=-9)

(N>= -9) and (N<=9)

N div 10 = 0

Expresiones numéricas y lógicas

- **Al introducir la primitiva de asignación, se mostró que el lado derecho al símbolo ":=" es una expresión que da un valor.**
- **Las expresiones indican ("expresan") como calcular adecuadamente un valor.**
- **Saber construir correctamente expresiones es muy importante porque:**
  - **se utilizan de muchas maneras en un algoritmo (no solo en asignaciones)**
  - **hay expresiones de muchos tipos de valores (no solo numéricos)**

Operadores y valores

- **Operadores numéricos: (ej, + - / * mod div) Toman números y tienen un número por resultado**
- **Operadores relacionales: (ej. = > < <> >= <=) Relacionan dos datos del mismo tipo y tienen un resultado que es verdadero o falso.**
- **Operadores lógicos: (ej. and or not) Toman valores del conjunto { verdadero, falso } y su resultado es un valor verdadero o falso.**

Expresiones numéricas y lógicas

Tarea, escriba expresiones para:

- **Un número N es mayor a 10**
- **N es mayor a 10 y menor a 100.**
- **N tiene a lo sumo 4 dígitos.**
- **N tiene 4 dígitos (exactamente).**
- **N tiene dos o cuatro dígitos.**
- **N es un número impar.**
- **N es divisible por 7 y divisible por 11 y tiene dos dígitos.**

Hay más ejercitación vea el práctico ☺

Realice una traza y luego pase a la máquina

```
PROGRAM Ejemplo3; {Algunos ejemplos con el tipo Boolean}
VAR   R1: REAL;
      es_par, es_positivo, mayor_a_maxint: BOOLEAN;
      condicion:BOOLEAN;
BEGIN
Write('ingrese un número'); read(R1);
Es_par :=  (TRUNC(R1) mod 2) <> 1;
es_positivo := R1 >= 0;
Mayor_a_maxint := R1 > MAXINT;
Condicion:= es_par and es_positivo and not mayor_a_maxint;
writeln('Resultado de la expresión lógica:  ', condicion);
readln;
END.
```


Continuará