Resolución de Problemas y Algoritmos

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

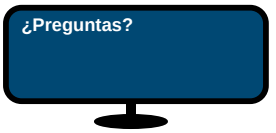
George Boole

Dr. Diego R. García

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

Conceptos de las clases anteriores

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



Sobre el trabajo profesional futuro

Un informático debe tener la capacidad de resolver problemas que pueden ser de:

- ✓ **muy gran escala**: *por ejemplo mantener en órbita a la Estación Espacial Internacional (ISS).*
- ✓ **gran escala**: por ejemplo desarrollar un sistema de manejo de materias y exámenes (como SIU-Guarani).
- ✓ **pequeña escala**: como *por ejemplo,*
  - *validar identidad de un usuario mediante su nombre y clave;*
  - *controlar que una fecha ingresada en un formulario web sea correcta;*
  - transformar grados Celsius a Fahrenheit.

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 de Pascal

Se pueden incluir **de una o más líneas** encerrados entre llaves `{ }` y también comentar al final de una línea usando dos barras `//`.

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

Programación de computadoras a bajo nivel

Acme

Chimtel

Maple

Achepe

- **Cada CPU de una computadora es capaz de ejecutar un único lenguaje llamado lenguaje máquina.**
- **Generalmente, cada marca de CPU tiene su propio lenguaje máquina.**

¿Tengo que aprender el lenguaje máquina de cada CPU? Afortunadamente NO (como verá a continuación)

Programación de computadoras a alto nivel

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

Conceptos: compilación

Un **compilador** es un programa que traduce un programa, que está escrito en un lenguaje de programación, a otro lenguaje de programación.

- Este proceso de traducción se conoce como **compilación**.
- Un **compilador** permite traducir el código fuente de un programa, a otro lenguaje (típicamente lenguaje de máquina) permitiendo generar **código ejecutable**.

El **código ejecutable** es una secuencia de código que la computadora puede ejecutar directamente al ser invocado, (sin necesidad que el compilador esté presente.)
Generalmente archivos de extensión EXE o COM en Windows.

Edición / Compilación / Ejecución

Edición: (por ejemplo con "Notepad") → **Compilación** (por ejemplo con "Free Pascal") → **Ejecución (por ej. en PC con Windows)**

Prueba.PAS **Código fuente** escrito en Pascal

Prueba.EXE **Código ejecutable** por la computadora

Existen aplicaciones (como Lazarus) que brindan un **entorno de programación,** donde se puede editar, compilar (con Free Pascal) y ejecutar programas en Pascal dentro del mismo entorno.

Conceptos: Tipo de dato (Data type)

Tipo de Dato: define el conjunto de valores posibles que puede tomar un elemento de un programa (ej. variable), las operaciones que pueden aplicarse, y cual es la representación interna para su almacenamiento en memoria.

Un tipo de dato (o simplemente un tipo) le indica al compilador cómo el programador tiene intención de usar los datos.

En Pascal los tipos de datos pueden ser:

1. **Predefinidos** (tienen un conjunto de valores y operaciones que ya está pre-establecido). Por ejemplo hoy veremos:
   - INTEGER (enteros)
   - REAL (reales)
   - CHAR (caracteres)
   - BOOLEAN (lógico: verdadero o falso)
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 y un máximo definido por el compilador usado.

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

Operaciones predefinidas: del tipo INTEGER

+ (adición) – (substracción) * (multiplicación)
div (división entera) y **mod** (resto de la división entera).

Operadores relacionales para comparar: =, >, <, >=, <=, <>

"distinto de"

"mayor o igual"

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

Funciónes predefinidas: Cada compilador define las propias, por ejemplo, **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 EjemploInteger; {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); // constante predefinida
    N1 := 2000 mod 2;
    N2 := 2000 div 2 ;
    N3 := SQR(SQR(3));
    N4 := MAXINT;
    form:= 15; //valor de formateo (Ancho que ocupa, justificado a derecha)
    writeln(n1:form, n2:form, n3:form, n4:form);
    N5 := 1+ N4 ;  //¿qué valor toma N5?
    writeln(N5);  //¿por qué será este valor?
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)

"es posterior en el código ASCII"

¿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 Castellano;
BEGIN {algunos caracteres del castellano}
write(chr(160),chr(130),chr(161),chr(162));
write(chr(163), chr(164),chr(165),chr(168));
END.
```


Información adicional

George Boole (1815-1864)

Matemático y filósofo Inglés.

Es considerado como uno de los fundadores de las Ciencias de la Computación.

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

Ese álgebra de dos valores hoy es conocida como "álgebra de Boole", la cual se transformó en la base de la aritmética computacional moderna.

Gracias a su álgebra, hoy en día, podemos manipular operaciones lógicas que son la base de las computadoras y de la informática.

Tipo de dato predefinido de Pascal

Nombre: **BOOLEAN** (Booleano o lógico)

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

Obs: en Free Pascal (y en Lazarus) son identificadores reservados

Operadores predefinidos:

and (conjunción "y"),
or (disyunción "o")
not (negación "no")
Operadores relacionales: =, >, <, >=, <=, <>

"mayor o igual"

"distinto de"

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

Por ejemplo: 5 > 3 retorna **true**, y 5 <> 5 retorna **false**

Conceptos: Expresiones Lógicas

- Los operadores relacionales retornan un valor de verdad: `5 > 3` retorna `true`, `5 <> 5` retorna `false`
- Los operadores lógicos: `and`, `or` y `not`, reciben valores de verdad y retornan valores de verdad.
- Los operadores lógicos permiten construir **expresiones lógicas** que retornarán un valor de verdad.
- Ejemplo: considere `num` de tipo entero:

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

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

Tablas de verdad

Una **tabla de verdad** para un operador lógico, muestra explícitamente el resultado para cada valor 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:

`not A` ← `En Pascal`

A	no A
true	false
false	true

Ejemplo: A podría representar "tengo señal de wi-fi"

En este caso, si "tengo señal de wi-fi" es verdadero, entonces "**no** tengo señal de wi-fi" es falso.

Tablas de verdad

Sean A y B dos expresiones lógicas cualquiera, las tablas de verdad de la conjunción (y) y de la disyunción (o) son:

`Pascal`

A	B	`A and B` A y B	`A or B` A o B
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

Ejemplos:

- "A y B" podría representar las condiciones para realizar una llamada con la expresión **"tengo señal y tengo saldo"**
- "A o B" podría representar las condiciones para utilizar Internet con la expresión **"hay red wi-fi o hay red de datos móviles"**

Precedencia de los operadores lógicos

Los operadores pueden **combinarse**.

La **precedencia** (orden de evaluación) es: **no, y, o (not, and, or)**

Los paréntesis cambian el **orden** de evaluación, por ejemplo,

- Compare estas dos expresiones para diferentes casos de prueba:

hay wi-fi o hay red-datos y no (batería = 0)

(hay wi-fi o hay red-datos) y no (batería = 0)

Considere este caso de prueba:

hay wifi = verdadero, **hay red-datos** = falso, **batería =** 0

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

Tenga esta tabla siempre "a mano" porque le será de mucha utilidad cuando esté programando condiciones en Pascal.

Tabla 12.1: Precedencia de operadores en Free Pascal

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

Operador	Precedencia	Categoría
not	La más alta (primero)	Unario
*** / div mod and**	segundo	binario
+ - or	tercero	binario
= <> < > <= >=	La más baja (último)	relacional

Para alterar el orden de precedencia en la evaluación se utilizan los () paréntesis.

Expresiones Lógicas Equivalentes

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

Importante:

- 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 posibles.

Ejemplos:

A > 0 es equivalente a not (A <= 0)

Ya que, en todos los casos que A>0 es verdadero, not(A<=0) también lo será.

not (A or B) no es equivalente a (not A or not B)

ya que, si **A**=true y **B**=false:

not (A or B) es false**,** en cambio **(not A or not B)** es verdadero

Expresiones lógicas equivalentes

¿Por qué son importantes las expresiones equivalentes?

Porque la misma condición puede escribirse de diferentes maneras, y hay que buscar la más adecuada

Ejemplo: La condición ***"El número entero N tiene un solo dígito"*** puede representarse con cualquiera de estas cuatro 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 > -10) and (N < 10)

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

N div 10 = 0

¿Cuál usaría?

Realice una traza y luego pase a la máquina

```
PROGRAM EjemploBool; {Algunos ejemplos con el tipo Boolean}
VAR   caracter: CHAR;
      es_mayuscula, es_minuscula, es_letra: BOOLEAN;
      otra: BOOLEAN; // para mostrar una expresión equivalente
BEGIN
  Write(' Ingrese un carácter ');  read(caracter);
  es_mayuscula := (caracter >= 'A') and (caracter <= 'Z');
  es_minuscula := (caracter >= 'a') and (caracter <= 'z');
  es_letra:= es_mayuscula or es_minuscula;
  writeln('¿ Es ', caracter, ' una letra?  ', es_letra);
  otra := (ORD(caracter) >= ORD('A')) and (ORD(caracter) <= ORD('Z'));
  readln;
END.
```

Expresiones equivalentes

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 (booleano)**
- **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 ☺

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

Funciones predefinidas para reales/enteros en Pascal

Función	Descripción	Recibe	Retorna
abs	Valor absoluto	real o integer	El mismo tipo recibido
arctan	arctan en radianes	real o integer	real
cos	cosine de un radián	real o integer	real
exp	***e*** a una potencia	real o integer	real
ln	Algoritmo natural	real o integer	real
round	redondeo	real	integer
sin	Seno de un radián	real o integer	real
sqr	Cuandrado (square)	real o integer	El mismo tipo recibido
sqrt	square root (raíz cuad.)	real o integer	real
trunc	truncado	real o integer	integer

http://wiki.freepascal.org/Standard_Functions

Realice una traza y luego pase a la máquina

```
PROGRAM EjemploReal; {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.
```


Conceptos: tipos ordinales en Pascal

De los 4 tipos simples predefinidos que hemos visto, INTEGER,CHAR y BOOLEAN son tipos ordinales (REAL no es ordinal).

Los tipos ordinales tienen estas características:

- **Tienen un primer y último elemento.**
- **Tienen un orden, y para cada elemento está definido el siguiente** (a excepción del último) **y el anterior** (a excepción del primero).

Esto permite utilizar sobre los tipos ordinales las operaciones predefinidas:

- **succ() retorna el siguiente (excepto del último)**
- **pred() retorna el anterior (excepto del primero)**
- **ord() retorna el número de orden**

Continuará