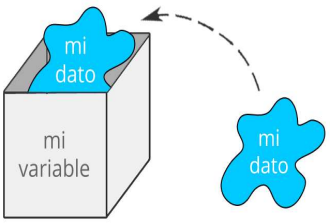
TECNOLOGÍAS EN EDUCACIÓN MATEMÁTICA

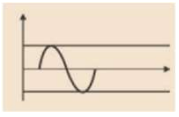
MODULO 12

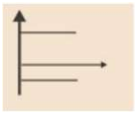
Dpto. de Ciencias e Ingeniería de la Computación
UNIVERSIDAD NACIONAL DEL SUR
Año 2019

Pascal Variables y Constantes

Los datos de un programa están representados por variables o constantes Tienen asociado un tipo.







Pascal Variables y Constantes

Tipo de dato en Pascal:

- define y restring el conjunto de valores posibles y el conjunto de operadores que se aplican sobre estos valores.
- Permite realizar **controles** para prevenir algunos errores.
- Si en una expresión un operador está asociado con operandos que no son compatibles con su tipo, el compilador reporta un error.
-

Integer
Real
Char
Boolean

Pascal Variables y Constantes

Una constante puede ser literal o quedar ligada a un identificador mediante una declaración.

```
…radio * 3.1415
write (‘Radio = ’)
…Ch = ‘.’
```

```
const
PI = 3.1415;
fin_texto = ’.’;
cartel = ‘El radio es: ’;
…
… radio * PI
 write(cartel)
…Ch = fin_texto
```

NO!!

Pascal Variables y Constantes

Las variables permiten que las instrucciones de un programa se ejecuten sobre diferentes valores.

En la declaración de una variable se establece:

- Nombre: cualquier identificador no reservado.
- Tipo: determina el conjunto de valores que puede tomar y en qué operaciones puede usarse.

En ejecución una variable tendrá:

- Valor: contenido en el conjunto establecido por el tipo.
- Locación: en memoria en la cual se almacena el valor.

Pascal Variables y Constantes

TIPO INTEGER: una computadora puede representar un conjunto finito de valores.

- El tipo *integer* brinda un subconjunto de los números enteros y un conjunto de operaciones definidas para este subconjunto de valores.
- El subconjunto está definido por el rango –MaxInt y MaxInt, donde MaxInt es una constante predefinida.
- Algunos operadores predefinidos del tipo integer son:

  + - * div mod sqr abs sqrt

Pascal Variables y Constantes

PREGUNTA:

Dadas las declaraciones

var x, y, M, N: integer;

¿Son válidas las expresiones?:

2+x*y

M mod N

sqr(x) div 2

abs(M)

Pascal Variables y Constantes

TIPO REAL: El tipo *real* incluye a un subconjunto de los números reales; aquellos que están dentro de un rango dado y con cierta precisión.

Algunos de los operadores predefinidos del tipo real son:

+ - * / abs sqr sin cos sqrt

Computan un valor de tipo real al aplicarse sobre un operando de tipo real.

trunc round

Se aplican sobre un argumento de tipo real y computan un número entero

Pascal Variables y Constantes

PREGUNTA: Dadas las declaraciones

```
var v1, v2: real;
```

¿Son válidas las expresiones?:

```
2.1 + v1/v2
sqr(v1)
abs(v2-v1)
sqrt(v1) + sqrt(v2*v2)
sin (v1)
v1 <= v2+1
```

Pascal Variables y Constantes

Tipo Booleano: El tipo boolean incluye solo a dos valores que son las constantes predefinidas *true* y *false* y los tres operadores lógicos **or**, **and** y **not**.

OJO!!!! El valor de las variables booleanas se puede imprimir pero no leer.

Los operadores relacionales <, >, =, <>, <=, >= se aplican a operandos de tipo integer, real o char y computan un valor de tipo Boolean

Pascal Variables y Constantes

```
const minimo = 10;
      maximo = 20;
var temp: integer;
    seSuspende: boolean;
…
seSuspende := (temp < minimo) or (temp > maximo);
```

Pascal no entiende: minimo < temp < maximo

Pascal Variables y Constantes

TIPO CHAR: El tipo ***char*** permite representar el conjunto de 256 letras mayúsculas y minúsculas, dígitos y otros caracteres.

Cada caracter está representado internamente como un número entero denotando su código ASCII.

Los literales de tipo de caracter se denotan entre apóstrofes, e.g. 'A', 'B', …, 'a', 'b', 'c', …, "", '@', …, '0', '1', '2', …

Pascal Variables y Constantes

American Standard Code for Information Interchange

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

Pascal Variables y Constantes

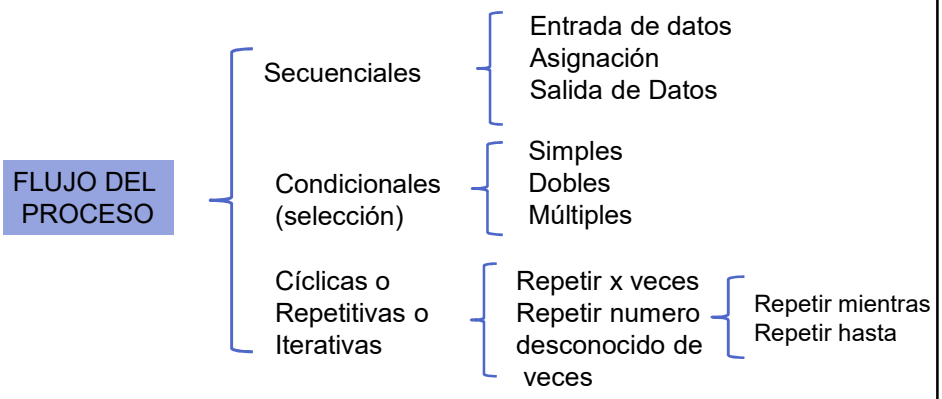
Los operadores **pred** y **succ** aplicados a un operando de tipo *char* computan un valor de tipo *char*.

El operador **chr** aplicado sobre un operando de tipo entero computa un valor de tipo *char*.

El operador **ord** aplicado sobre un operando de tipo *char* computa un valor de tipo integer

pred('z')='y'
succ('C')='D'
chr(35)='#'
ord('M')=77

Pascal Estructuras de Control



Pascal Estructuras de Control

Condicional Simple

Si condicion Entonces Accion1	If (condición) then Accion1

Si condicion
Entonces
Accion1
Accion2
	If (condición)
	then
	begin
	Accion1;
	Accion2;
	end;

Pascal Estructuras de Control

Si condicion
Entonces
  Accion1
Sino
  Accion2
	If (condición)
	 then
	  Accion1
	 else
	  Accion2;

Condicional Doble

Si condicion
Entonces
  Accion1
  Accion2
Sino
  Acciones3
  Accion4
	If (condición)
	then
	  begin
	  Accion1;
	  Accion2;
	  end
	else
	 begin
	  Accion3;
	  Accion4;
	 end;

Pascal Estructuras de Control

Condicional Multiple

En caso de DATO
  Valor1: accion1
  Valor2: accion2
  …
  ValorN: accionN
	Case a of
	  Valor1: accion1;
	  Valor1: accion2;
	  ….
	  ValorN: accionN;
	end;

Pascal Estructuras de Control

Iteración cuando conozco cantidad repeticiones

Para i: desde a hasta b
hacer
  acciones
	for i := a to b do
	  begin
	  accion1
	  accion2
	  end;

ITERACION CON CONTADOR

for variable de control := valor inicial to valor final do
instrucción simple o compuesta

- La *variable de control* debe ser de un **tipo ordinal**. Usaremos **enteros o caracteres**.
- El *valor inicial* y el *valor final* puede ser una constante, una variable o una expresión del mismo tipo que la variable de control.
- La instrucción simple o compuesta es el **bloque iterativo** y puede no ejecutarse.
- **El bloque iterativo no puede modificar el valor de la variable de control.**
- Tampoco deberían modificarse el resto de las variables que afectan al valor inicial y al valor final.

ITERACION CON CONTADOR

for variable de control := valor inicial to valor final do
instrucción simple o compuesta

```
program bucleFor;
var i: integer;
begin
   for i:= 1 to 5
   do
     writeln(i);
end.
```

```
program bucleFor;
var i: integer;
begin
   for i:= 0 to 5
   do
     writeln(i);
end.
```


ITERACION CON CONTADOR

for variable de control := valor inicial to valor final do
instrucción simple o compuesta

```
program bucleFor;
var i: integer;
begin
   for i:= 10 to 5
   do
     writeln (i);
end.
```

```
program bucleFor;
var i: integer;
begin
   for i:= 10 downto 5
   do
     writeln (i);
end.
```

ITERACION CON CONTADOR

for variable de control := valor inicial to valor final do
instrucción simple o compuesta

```
program bucleFor;
var i: integer;
begin
   for i:= -1 to 1
   do
     writeln (i);
end.
```

```
program bucleFor;
var i: char;
begin
   for i:= 'a' to 'm'
   do
     writeln (i, ': ord = ', ord(i));
end.
```

23

ITERACION CON CONTADOR

Problema: Calcular la suma de los dígitos de un número entero positivo N.

Por ejemplo si N= 605 la suma es 11.

Usamos un acumulador de sumas:

0+5 (d_1 o sea dígito en la posición 1 de N)
… +0 (d_2)
… +6 (d_3)

Estrategia para la resolución: Acumulo el último dígito de N, lo elimino, acumulo el último dígito de N, lo elimino, así hasta que N no tenga más dígitos.

ITERACION CON CONTADOR

Algoritmo SumaDigitos
DE: N
DS: suma
Comienzo
suma <- 0
Mientras (N>0) hacer
    suma ← suma + N mod 10
    N ← N div 10
Fin

```
Program SumaDigitos;
var N, suma: integer;
begin
writeln('Ingrese un numero:');
read(N);
suma := 0;
  while (N > 0) do
  begin
    suma := suma + N mod 10;
    N := N div 10;
  end;
writeln('La suma es:', suma);
end.
```

Operaciones básicas:
N *mod* 10 (consulto el último dígito de N)
N:=N *div* 10 (elimino el último dígito de N)

ITERACION CON CONTADOR

Problema: Determinar si un dígito d está en un número entero positivo N.

Por ejemplo
si d=3 y N= 605 d no está en N.
si d=3 y N= 305 d está en N.
si d=3 y N= 6035 d está en N.
si d=3 y N= 6053 d está en N.
si d=3 y N= 6303 d está en N.

ITERACION CON CONTADOR

Problema: Determinar si un dígito d está en un número entero positivo N.

```
Algoritmo dEstaenN
DE: N, d (enteros)
DS: esta (lógico)

esta ← Falso
mientras (N > 0) Y (esta=Falso) hacer
        Si (N mod 10 = d)
          entonces
            esta ← Verdadero
          sino
            N ← N div 10;
```

ITERACION CON CONTADOR

```
{Procesamiento}
  esta:= false;
  while (N > 0) and (not esta)
  do
     if (N mod 10 = d)
      then
        esta := true
      else
        N := N div 10;
```

(esta=false)

ITERACION CON CONTADOR

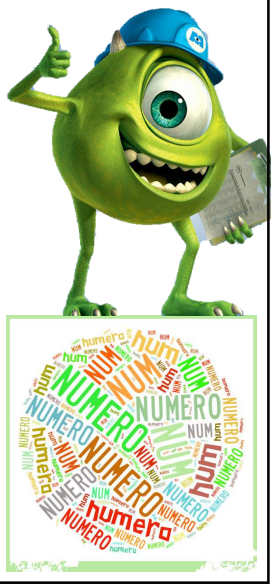
```
program dEstaenN;
{Decide si un dígito d está en N }
var N, d: integer;
    esta: boolean;
begin
{Entrada}
  write ('Ingrese N ');
  readln (N);
  write ('Ingrese el dígito ');
  readln (d);
…
```

ITERACION CON CONTADOR

```
program dEstaenN;
…
begin
{Entrada} …
{Procesamiento} …
{Salida}
  if   (esta=true)
  then
    writeln(d, ' está en ', N)
  else
    writeln(d, ' no está en ', N)  ;
end.
```

CUIDADO!! N fue modificado en el procesamiento!!!
Agregar donde corresponda:
aux:=N
Y luego:
N:=aux
para que estos carteles se muestren correctamente.

```
program dEstaenN; {Decide si un dígito d está en N }
var N, d, aux: integer;
    esta: boolean;
begin
  write ('Ingrese N ');    readln (N);
  write ('Ingrese el dígito ');  readln (d);
  aux:=N;
  esta := false;
  while (N > 0) and (esta=false)
  do
       if (N mod 10 = d)
        then            esta := true
        else           N := N div 10;
  if   esta
  then     writeln(d, ' está en ', aux)
  else     writeln(d, ' no está en ', aux)  ;
end.
```



ESTRUCTURAS DE CONTROL

Iteración cuando NO conozco cantidad repeticiones

Mientras Condicion accion	While Condicion accion

Mientras Condicion Acción 1 Accion 2 Accion n	While Condicion Begin Acción 1; Accio n 2; Accion n End:

ESTRUCTURAS DE CONTROL

EJERCICIO: Leer los límites de un intervalo cerrado [a, b] y un número x y determinar si x está en el intervalo.

x está en el intervalo [a, b] si a <= x <= b

Pascal no admite una expresión como
a <= x <= b

```
(x >= a) and (x <= b)

(a <= x) and (x <= b)
```

```
program intervalo;
var a, b, x: integer;
 pertenece: boolean;
begin
  writeln ('Ingrese los límites del intervalo ');
  readln (a,  b);
  writeln ('Ingrese el valor de x:');
  readln (x);

  if ((a<=x) and (x<=b))
        then pertenece := true
        else pertenece:=false;

  if   (pertenece = true)
     then writeln (x, ' pertenece al intervalo')
     else writeln (x,' no pertenece al intervalo');
end.
```

Algoritmo INTERVALO
DE: a, b, x (enteros)
DS: pertenece (lógico)
SI ((a<=x) Y (x<=b))
entonces
    pertenece ← Verdadero
sino
    pertenece ← Falso

Antes del ELSE no va punto y coma.

```
program intervalo;
var a, b, x: integer;
 pertenece: boolean;
begin
  writeln ('Ingrese los límites del intervalo ');
  readln (a,  b);
  writeln ('Ingrese el valor de x:');
  readln (x);

 if ((a<=x) and (x<=b))
        then pertenece := true
        else pertenece:=false;

 if   (pertenece = true)
     then writeln (x, ' pertenece al intervalo')
     else writeln (x,' no pertenece al intervalo');
end.
```

Algoritmo INTERVALO
DE: a, b, x (enteros)
DS: pertenece (lógico)

Leer los datos de entrada

Antes del ELSE no va punto y coma.

Mostrar los datos de salida

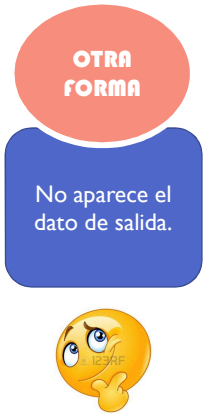
CONDICIONAL: IF

```
program intervalo;
var a, b, x: integer;     pertenece: boolean;
begin
  writeln ('Ingrese los límites del intervalo ');
  readln (a,  b);
  writeln ('Ingrese el valor de x:');
  readln (x);

  pertenece:= (a<=x) and (x<=b);

  if   (pertenece = true)
     then writeln ('x pertenece al intervalo')
     else writeln ('x no pertenece al intervalo');
end.
```

OTRA FORMA

CONDICIONAL: IF

```
program intervalo;
var a, b, x: integer;
begin
  writeln ('Ingrese los límites del intervalo ');
  readln (a,  b);
  writeln ('Ingrese el valor de x:');
  readln (x);

  if   ((a<=x) and (x<=b))
       then writeln ('x pertenece al intervalo')
       else writeln ('x no pertenece al intervalo');
end.
```



CONDICIONAL: CASE

Escribir un programa que leer una expresión aritmética binaria en notación prefija y calcular el resultado.
En la notación prefija el operador +, -, * o / precede a sus operandos.

¿Cuáles son los datos de entrada?

¿Cuáles son los datos de salida?

¿Cuáles pueden ser los casos de prueba?

CONDICIONAL: CASE

op es un caracter, x e y son enteros.
sol es entera

```
case op of
  '+': sol := x+y;
  '-': sol := x-y;
  '*': sol := x*y;
  '/': sol := x div y;
end;
```

CONDICIONAL: CASE

```
write ('Ingrese operador operando operando');
readln (op, x, y);
if (op <> '+') and (op <> '-') and
   (op <> '*') and (op <> '/')
then
  writeln ('El operador no existe')
else
  begin
    ...
  end
```

CONDICIONAL: CASE

```
program evaluaExpresion;
var
  x, y, sol: integer;
  op: char;
begin
  write ('Ingrese operador operando operando ');
  read (op, x, y);
  if (op<>'+') and (op<>'-') and (op<>'*') and (op<>'/')
  then writeln ('El operador no existe')
  else
    begin
        case op of
        '+': sol := x+y;
        '-': sol := x-y;
        '*': sol := x*y;
        '/': sol := x div y;
        end;
      writeln ('La solucion es ',sol);
   end;
  readln;
  readln;
end.
```

TECNOLOGÍAS EN EDUCACIÓN MATEMÁTICA

FIN MODULO 12

Dpto. de Ciencias e Ingeniería de la Computación
UNIVERSIDAD NACIONAL DEL SUR

Año 2019