

TEORIA 2

TEMAS de la CLASE

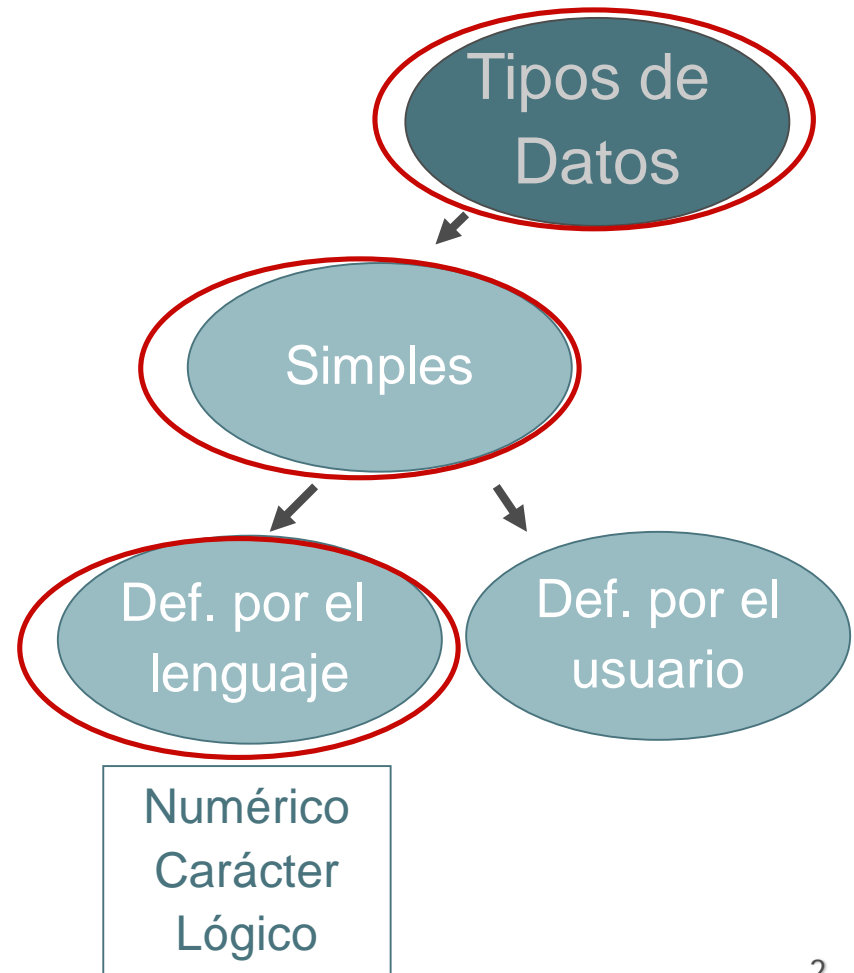
- 1 Tipo de Dato Lógico
- 2 Tipo de Dato Caracter
- 3 Concepto de Estructura de Control
- 4 Clasificación de las estructuras de control
 - Selección Simple
 - Selección Múltiple
 - Iterativa Precondicional

Clasificación de Tipos de datos

Recordemos la clasificación de los tipos simples y definidos por el lenguaje ya vista...

Recordemos también que los tipos de datos se caracterizan por:

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



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 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

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	å	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 Alt MÁS EL NUMERO	
17		41)	65	A	89	Y	113	q	137	ë	161	í	185		209	⌋	233	Θ		
18	‡	42	*	66	B	90	Z	114	r	138	è	162	ó	186		210	⌋	234	Ω	CORTESÍA DE: 	
19	‡‡	43	+	67	C	91	[115	s	139	ï	163	ú	187		211	⌋	235	δ		
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	Ä	166	°	190		214	⌋	238	€		
23	‡	47	/	71	G	95	_	119	w	143	Å	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: (ver en la tabla)

- ('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.
```

Concepto de Estructura de control.

Todos los lenguajes de programación tienen un conjunto mínimo de instrucciones que permiten especificar el **control** del algoritmo que se quiere implementar.

Veremos que este conjunto **debe contener como mínimo:**

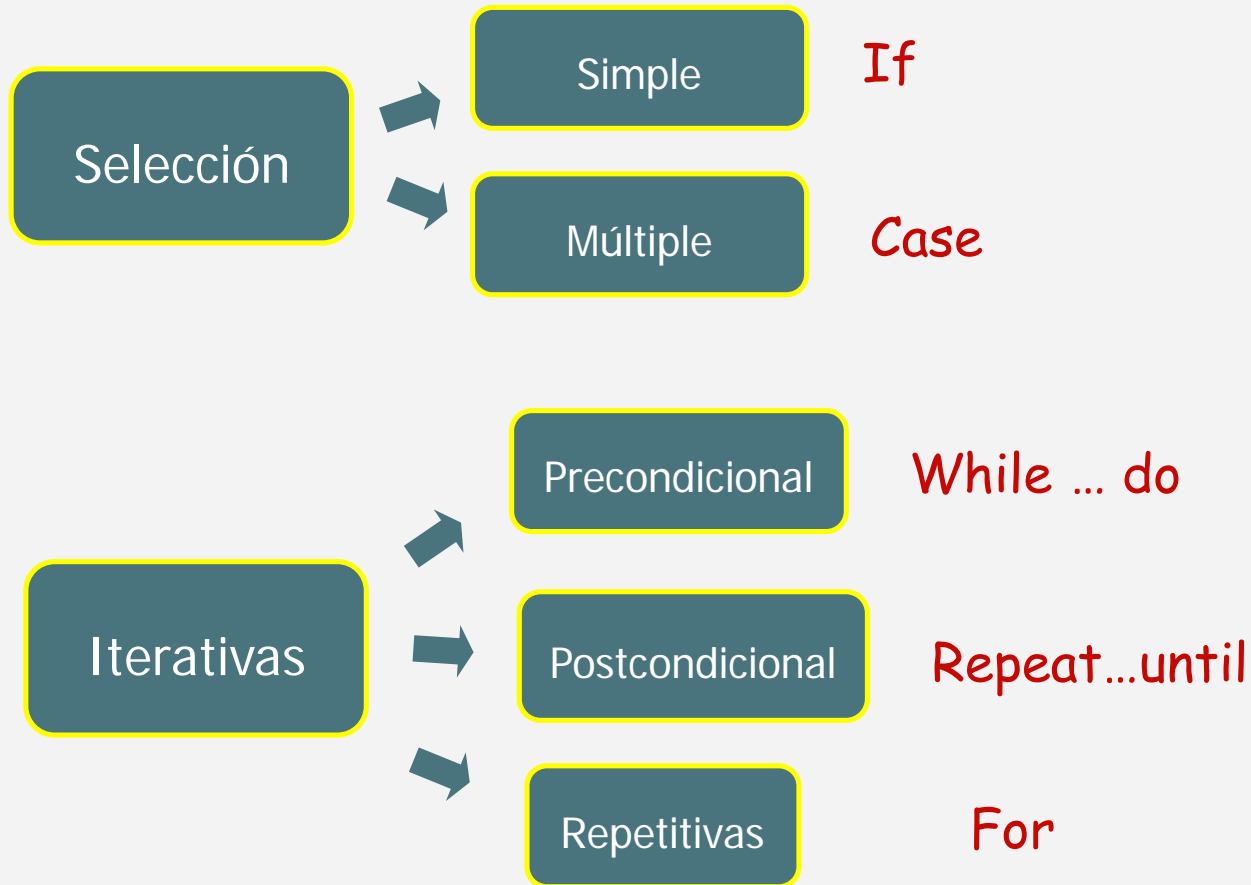
✓ **Selección**

✓ **iteración**

¿Para qué nos sirven las estructuras de control?

Las estructuras de control permiten modificar el flujo de ejecución de las instrucciones de un programa.

Clasificación de las Estructuras de control.

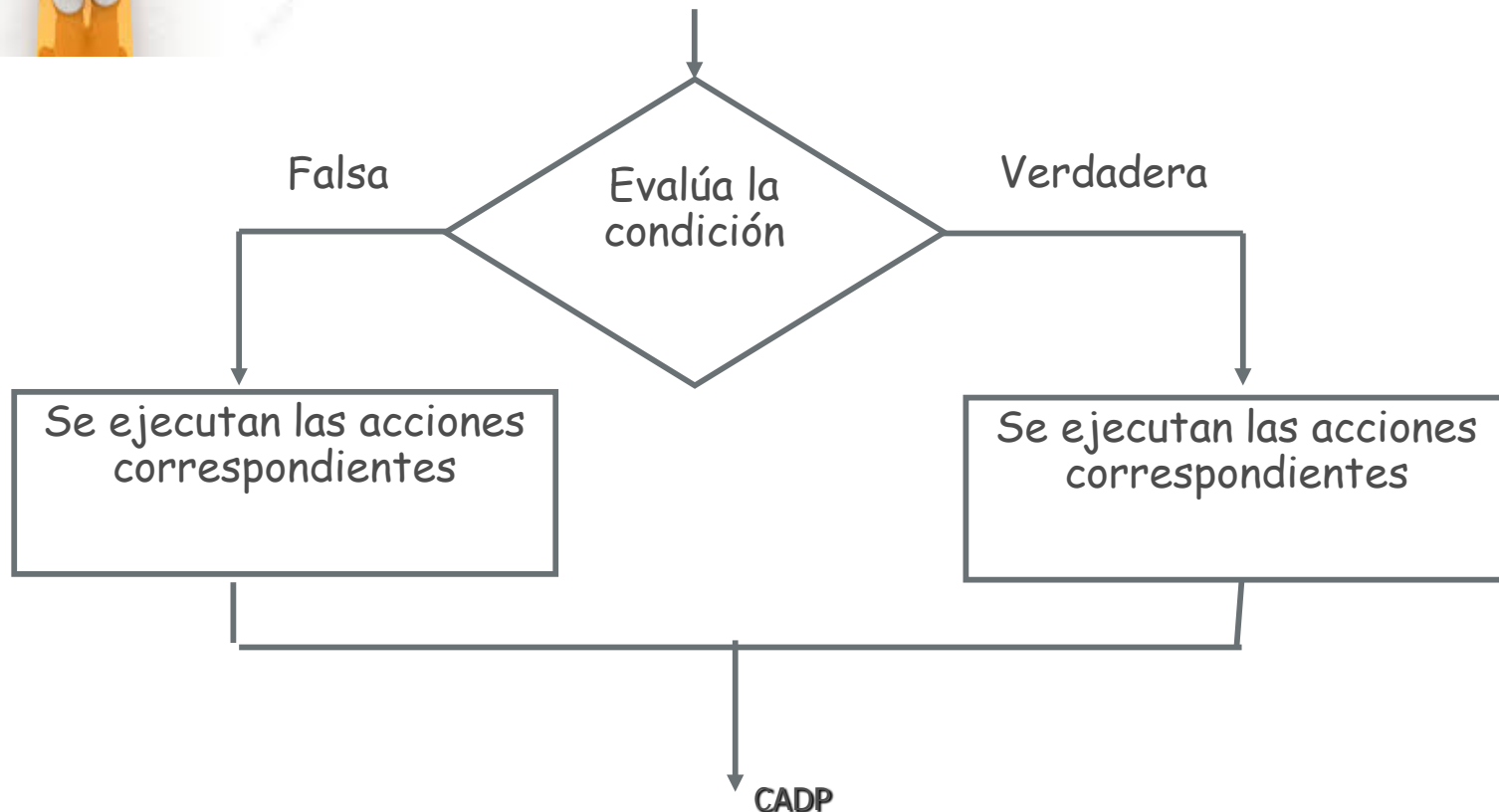


Estructura de control: selección simple (en Pascal)



Puede ocurrir que en un problema real sea necesario elegir una alternativa entre 2 posibles.

La estructura de selección simple se representa simbólicamente:



Estructura de control: selección simple (en Pascal)

```
If  (condición)  then begin
    Acciones_por_Condición_Verdadera;
end
    else begin
    Acciones_por_Condicion_Falsa;
end;
```

Caso Especial (sin else)

```
If  (condición)  then begin
    Acciones_por_Condición_Verdadera;
end;
```

Ejercicio



Se quiere comprar un terreno solo si su superficie es mayor a 900 mts².

Implementar un programa que calcule la superficie del terreno cuyas dimensiones se leen e informe si lo compra o no.

Algoritmo:

- Leer dimensiones
- Calcular superficie
- Si (superficie > 900) entonces
 - Mostrar Comprar
- sino
 - Mostrar No comprar

Program superficie;

var frente, fondo: real;

sup: real;

Begin

readln (frente); readln (fondo);

sup := frente * fondo;

if (sup > 900) then

write ('Comprar Terreno');

else

write ('No Comprar Terreno');

End.

Ejercicio



Escribir un programa que permita que se ingrese un tipo de impuesto y en función del tipo de impuesto se muestre a que caja debe dirigirse. Unicamente se cobran impuestos de los tipos "I" en la caja 1 y "M" en la caja 2.

Algoritmo:

- Leer impuesto
- Si es I -> Caja 1
sino -> Caja 2

```
Program Cobros;
```

```
var impuesto: char;
```

```
Begin
```

```
readln (impuesto);
```

```
if (impuesto = 'I') then
```

```
    write ('Ir a Caja 1');
```

```
else
```

```
    write ('Ir a Caja 2');
```

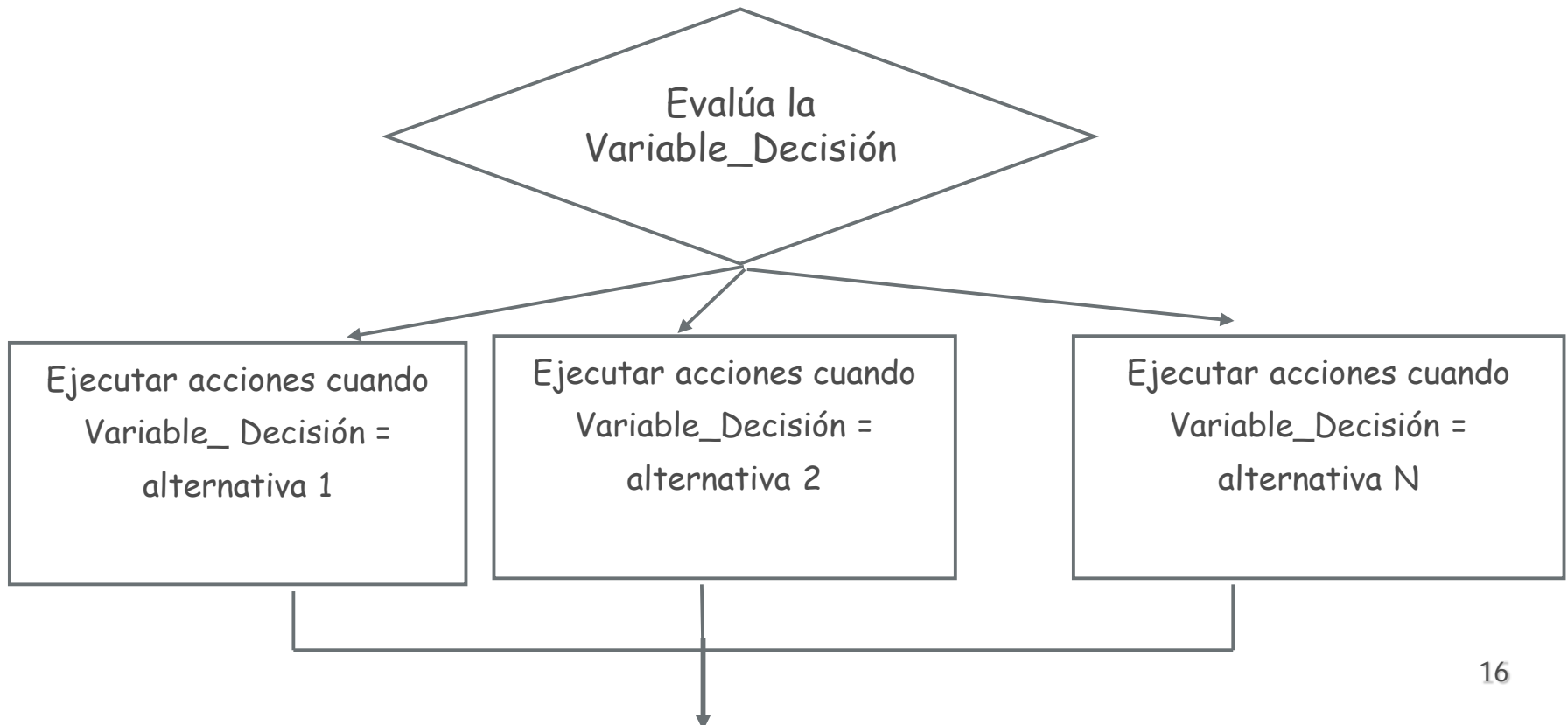
```
End.
```

Estructura de Control de selección múltiple



Puede ocurrir que en un problema real sea necesario elegir una alternativa entre varias posibles en función del problema a resolver.

La estructura de selección múltiple se representa simbólicamente:



Estructura de Control de selección múltiple (en Pascal)

Puede ocurrir que en un problema real sea necesario elegir una alternativa entre varias posibles en función del problema a resolver.

```
Case variable decision of  
    alternativa 1 : Acciones;  
    alternativa 2 : Acciones;  
    .....  
    else otras acciones  
End;
```

Ejercicio



Escribir un programa que permita que se ingrese un tipo de impuesto y en función del tipo de impuesto se muestre a que caja debe dirigirse. Se cobran impuestos de los tipos “I”, “M”, “P” y “S”, en las cajas 1 a 4 respectivamente. Para otros impuestos no hay cajas disponibles.

Algoritmo:

- Leer impuesto
- Verificar tipo
- Si es I -> Caja 1
 sino si es M -> Caja 2
 sino si es P ...

```
case impuesto of
    'I': writeln ('Dirigirse a Caja 1');
    'M': writeln ('Dirigirse a Caja 2');
    'P': writeln ('Dirigirse a Caja 3');
    'S': writeln ('Dirigirse a Caja 4');
    else write  ('Acá no se cobra ese
                impuesto')
end;
```

Ejercicio



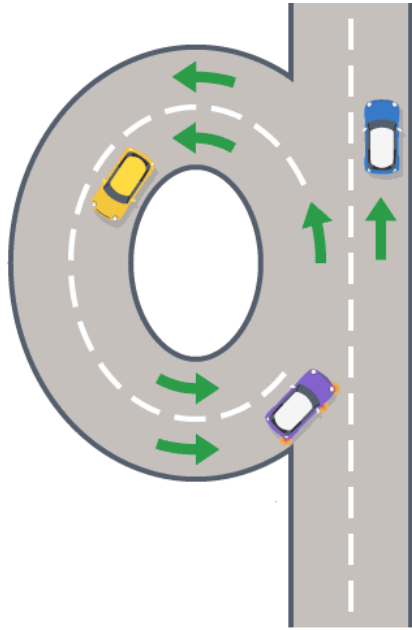
Escribir un programa que permita que se ingrese un tipo de impuesto y en función del tipo de impuesto se muestre a que caja debe dirigirse. Se cobran impuestos de los tipos "I", "M", "P" y "S", en las cajas 1 a 4 respectivamente. Para otros impuestos no hay cajas disponibles.

```
Program pagoimpuesto;  
  var impuesto : char;  
  
begin  
  readln (impuesto);  
  case impuesto of  
    'I': writeln ('Dirigirse a Caja 1');  
    'M': writeln ('Dirigirse a Caja 2');  
    'P': writeln ('Dirigirse a Caja 3');  
    'S': writeln ('Dirigirse a Caja 4');  
    else write ('Acá no se cobra ese impuesto')  
  end;  
end.
```

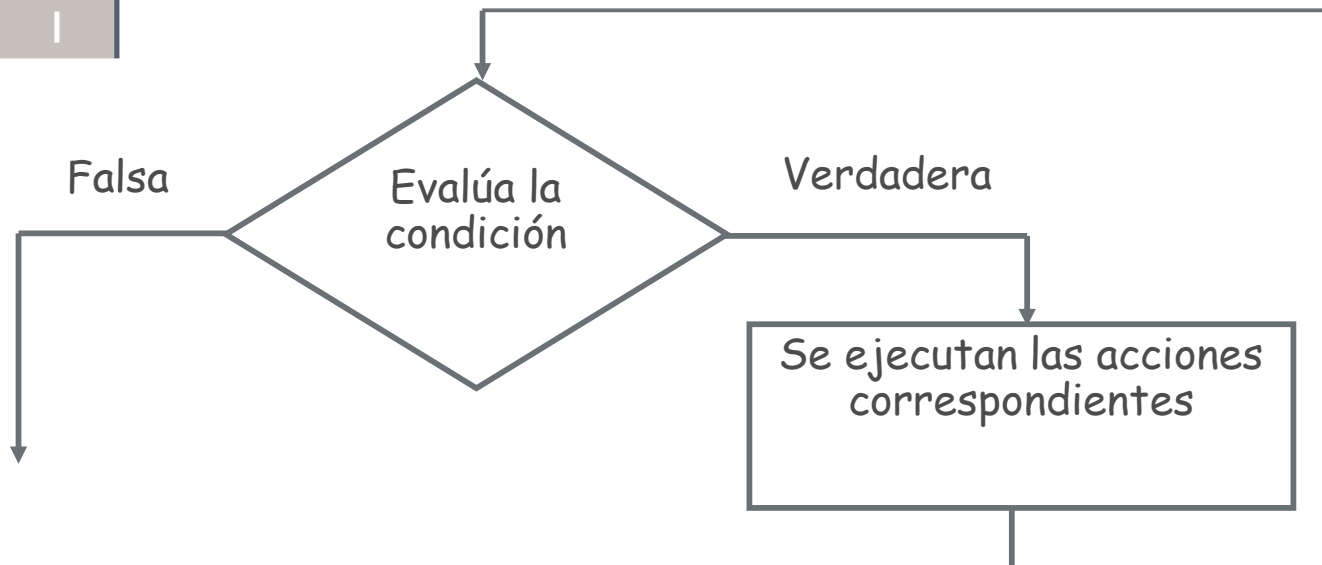
Estructuras de Control iterativas

- Puede ocurrir que se desee ejecutar un bloque de instrucciones desconociendo el número exacto de veces que se ejecutan.
- Para estos casos existen en la mayoría de los lenguajes de programación estructurada las **estructuras de control iterativas condicionales**.
- Como su nombre lo indica las acciones se ejecutan dependiendo de la evaluación de la condición.
- Estas estructuras se clasifican en **pre-condicionales** y **post-condicionales**.

Estructura de Control iterativa precondicional



- Las **estructuras de control iterativas precondicionales** primero evalúan la condición y si es verdadera se ejecuta el bloque de acciones. Dicho bloque se puede ejecutar 0, 1 ó más veces.
- Importante: el valor inicial de la condición debe ser conocido o evaluable antes de la evaluación de la condición.



Estructura de Control iterativa precondicional (en Pascal)

■ Las **estructuras de control iterativas precondicionales** primero evalúan la condición y si es verdadera se ejecuta el bloque de acciones. Dicho bloque se puede ejecutar 0, 1 ó más veces.

■ **Importante**: el valor inicial de la condición debe ser conocido o evaluable antes de la evaluación de la condición.

```
While (condición) do  
  begin  
    Acciones a realizar  
  end;
```

Ejercicio



En el ejercicio de los terrenos, si se piensa que hay varios terrenos y se quiere informar la superficie de cada uno, ¿Cómo lo resolveríamos sabiendo que si el largo es 0 se termina el cálculo?

Escribir un programa que permita ingresar los datos (frente y fondo) de los terrenos e informe la superficie de cada uno. El ingreso de los datos del terreno finaliza cuando se ingresa el fondo igual a 0.

Algoritmo:

- Leer fondo
 - Mientras el fondo sea distinto de 0
 - Leer frente
 - Calcular superficie
 - Mostrar superficie
 - Leer fondo
- Fin mientras

```
Program VariosTerrenos;  
  var frente, fondo, sup : real;  
  
begin  
  readln (fondo);  
  while (fondo <> 0) do begin  
    readln (frente);  
    sup:= frente * fondo;  
    writeln ('La superficie es: ', sup);  
    readln (fondo);  
  end;  
end.
```

Ejercicio



Si volvemos al caso del pago de impuestos y ahora se piensa que hay varios clientes que van a pagar un impuesto, escribir un programa que permita ingresar el tipo de impuesto a pagar por cada cliente y en función del tipo de impuesto se muestre a que caja debe dirigirse. El ingreso de impuestos finaliza cuando se ingresa el impuesto '@'.

Algoritmo:

- Leer impuesto
 - Mientras el impuesto sea distinto de '@'
 - Verificar tipo
 - Si es I -> Caja 1
 - sino si es M -> Caja 2
 - sino si es P ...
 - Leer impuesto
- Fin mientras

```
Program PagoVariosImpuestos;  
  var impuesto : char;  
  
begin  
  readln (impuesto);  
  while (impuesto <> '@') do begin  
    case impuesto of  
      'I': writeln ('Dirigirse a Caja 1');  
      'M': writeln ('Dirigirse a Caja 2');  
      'P': writeln ('Dirigirse a Caja 3');  
      'S': writeln ('Dirigirse a Caja 4');  
      else write ('Acá no se cobra ese impuesto')  
    end;  
    readln (impuesto);  
  end;  
end.
```


Para entregar



En el problema de los terrenos, reescriba el programa para que además informe cuántas superficies se calcularon.

```
Program VariosTerrenos;  
  var frente, fondo, sup : real;  
  
begin  
  readln (fondo);  
  while (fondo <> 0) do begin  
    readln (frente);  
    sup:= frente * fondo;  
    writeln ('La superficie es: ', sup);  
    readln (fondo);  
  end;  
end.
```

1. Analice los datos y tipos de datos necesarios
2. Reescriba el programa en Pascal

Resolver



Si en el problema del pago de impuestos, además se pide que:

1. Informe el total facturado por las cuatro cajas.
2. Informe el valor promedio de los impuestos pagados.

1. Escriba el algoritmo
2. Analice los datos y tipos de datos necesarios
3. Escriba el programa en Pascal