TEORIA 2

TEMAS de la CLASE

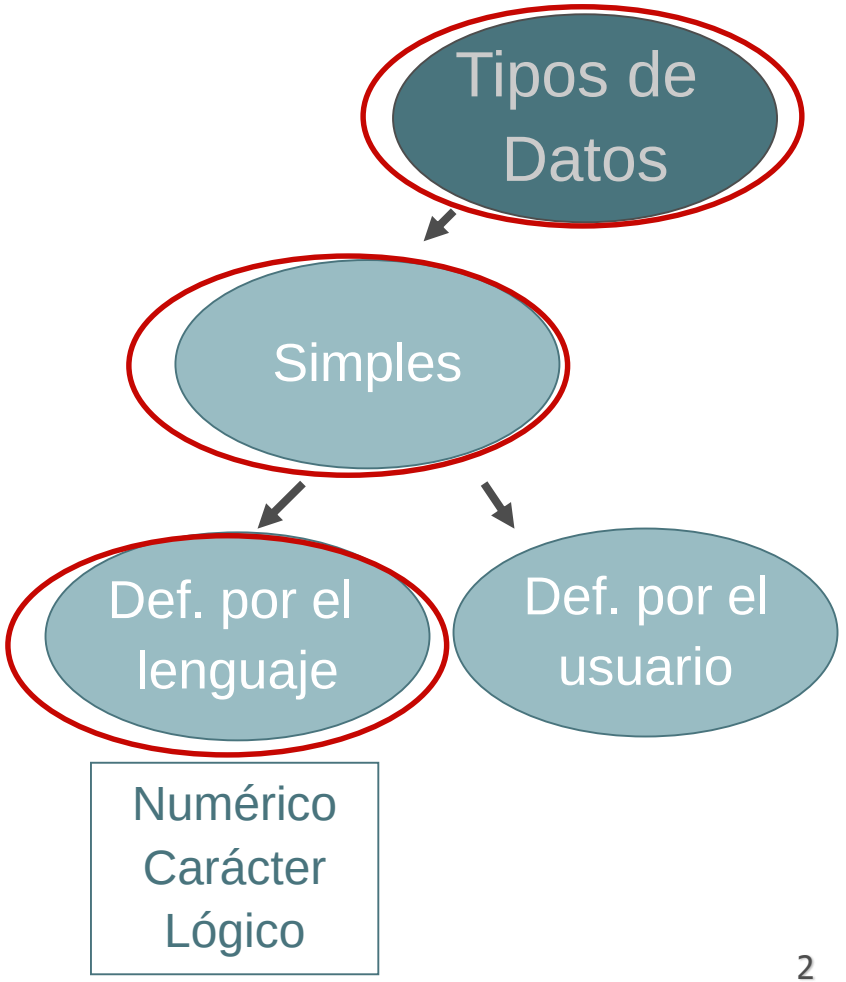
1. Tipo de Dato Lógico
2. Tipo de Dato Caracter
3. Concepto de Estructura de Control
4. Clasificación de las estructuras de control
   - Selección Simple
   - Selección Múltiple
   - Iterativa Precondicional

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*)

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

Existe un orden de precedencia para los operadores lógicos:

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

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

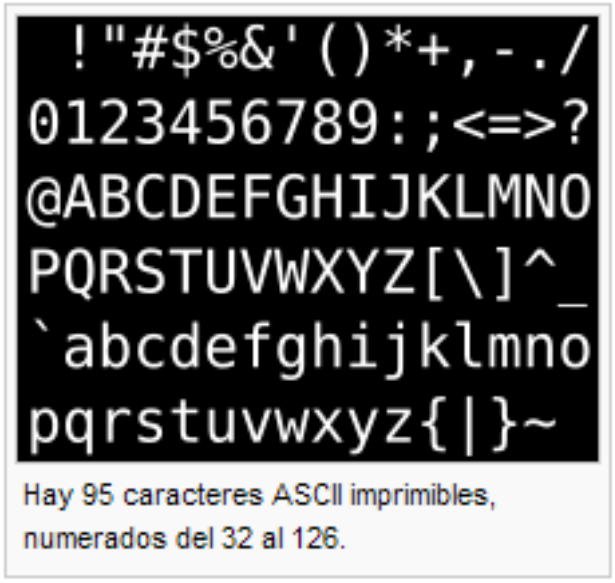
El **tipo de dato carácter** es un tipo de dato simple y ordinal.

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'



Hay 95 caracteres ASCII imprimibles, numerados del 32 al 126.

Una variable de tipo carácter ocupa 1 byte de memoria

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.
```

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.

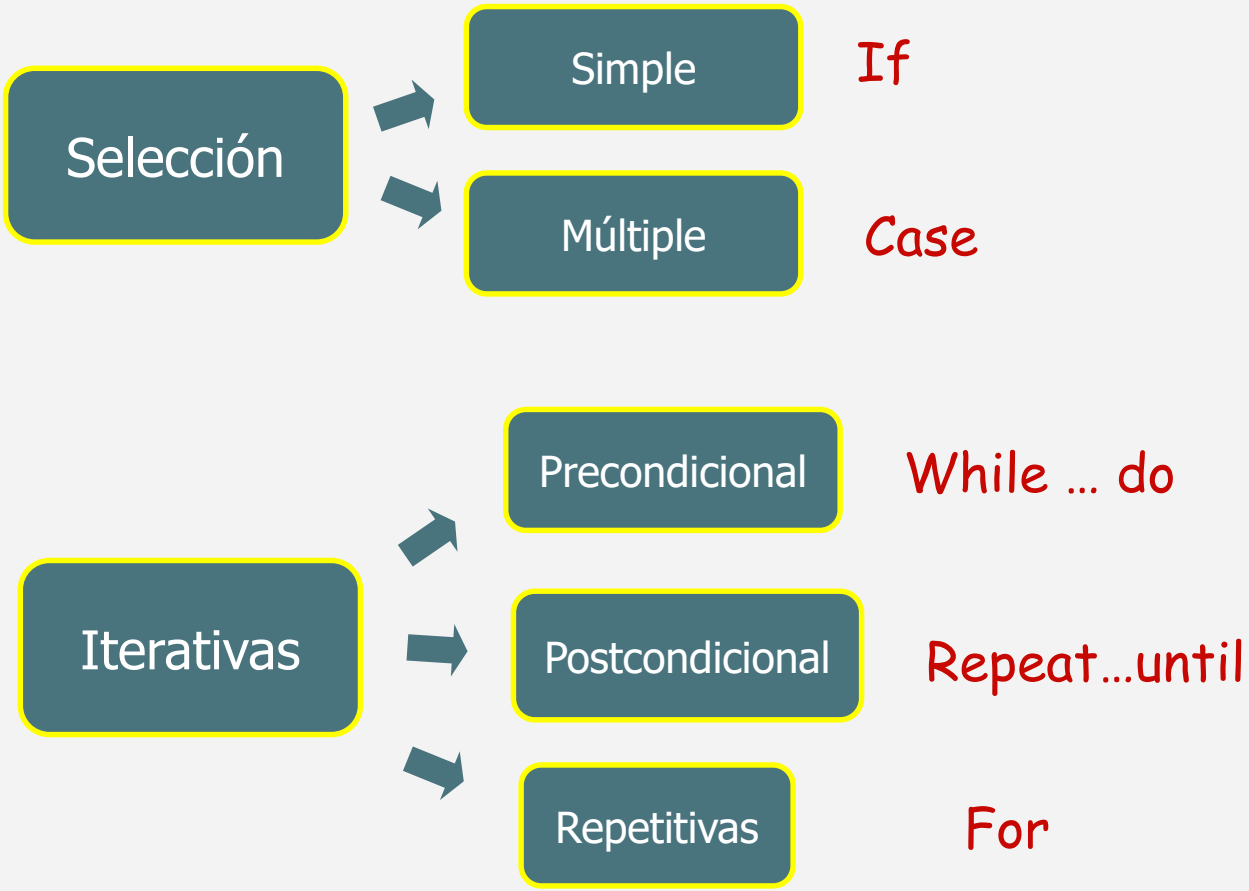
Veremos que este conjunto **debe contener como mínimo:**

- ✓ **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.

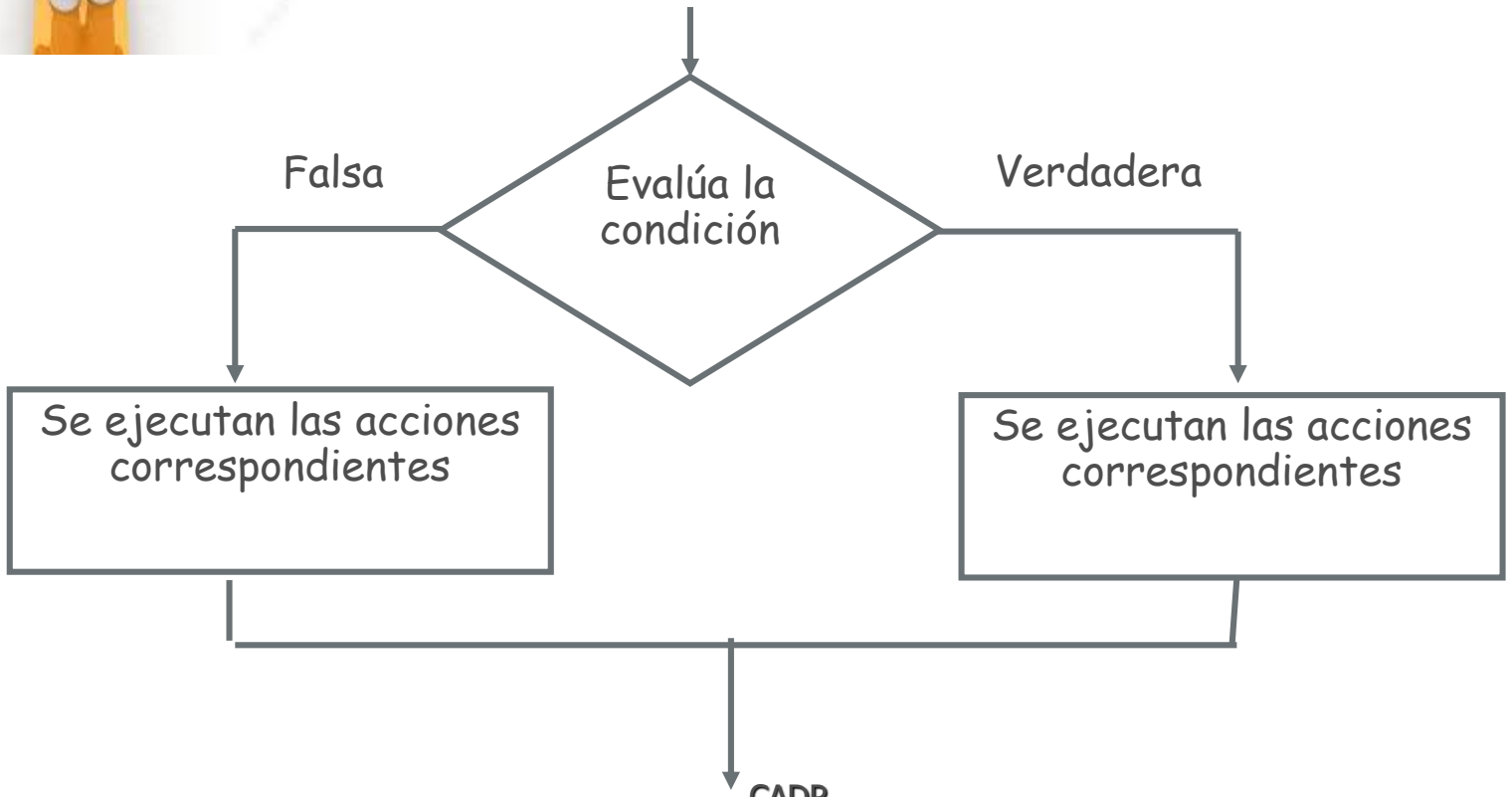
Clasificación de las Estructuras de control.



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

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

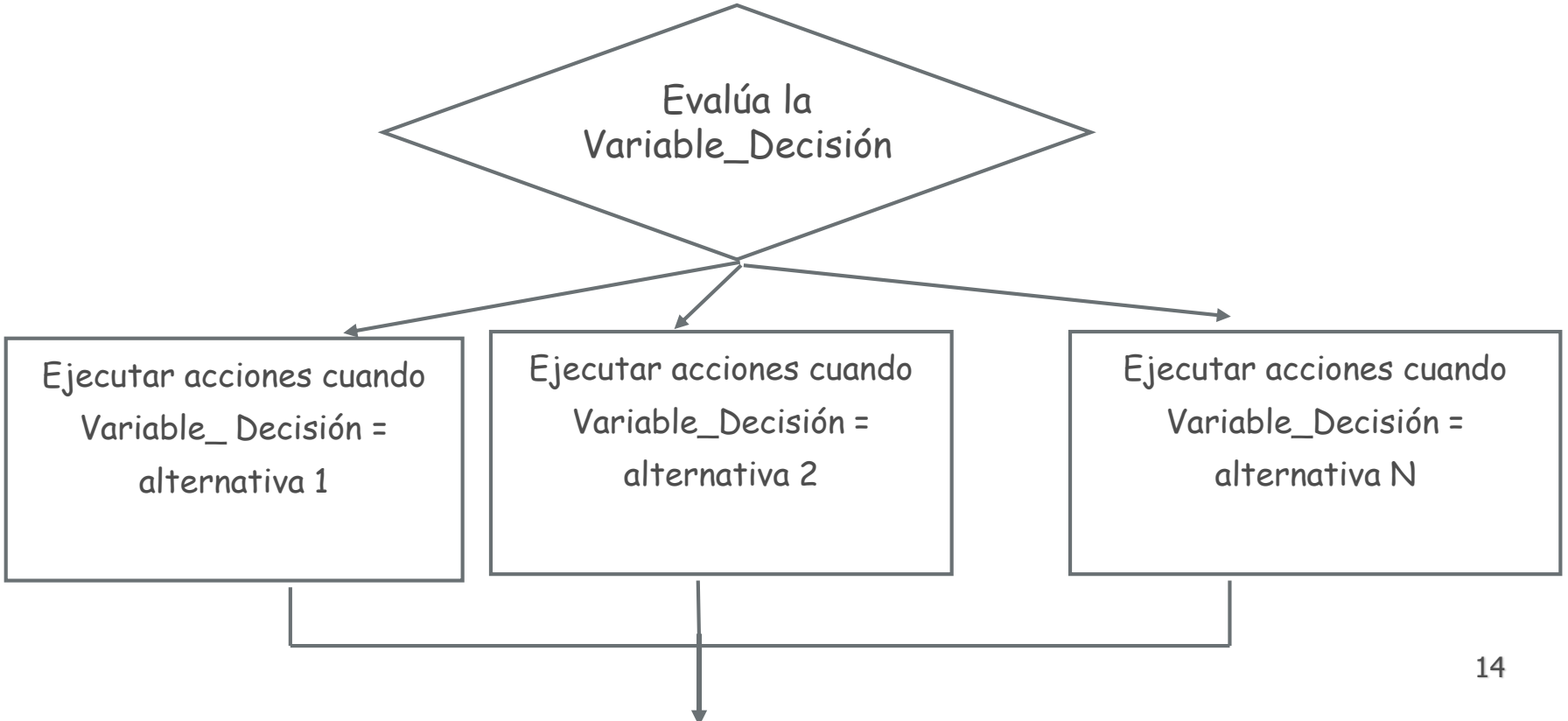
```
Program Cobros;
  var impuesto: char;

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.

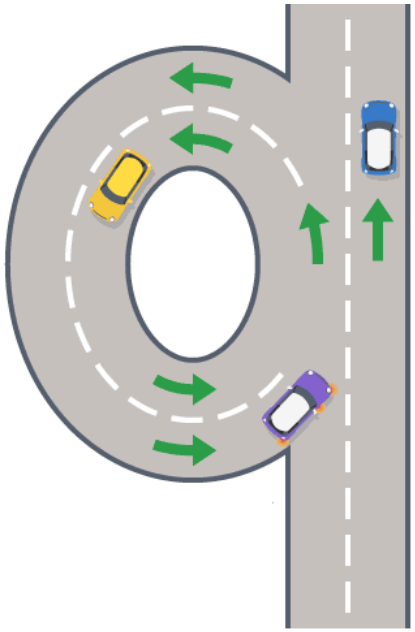
```
Program pagoimpuesto;
   var impuesto : char;

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

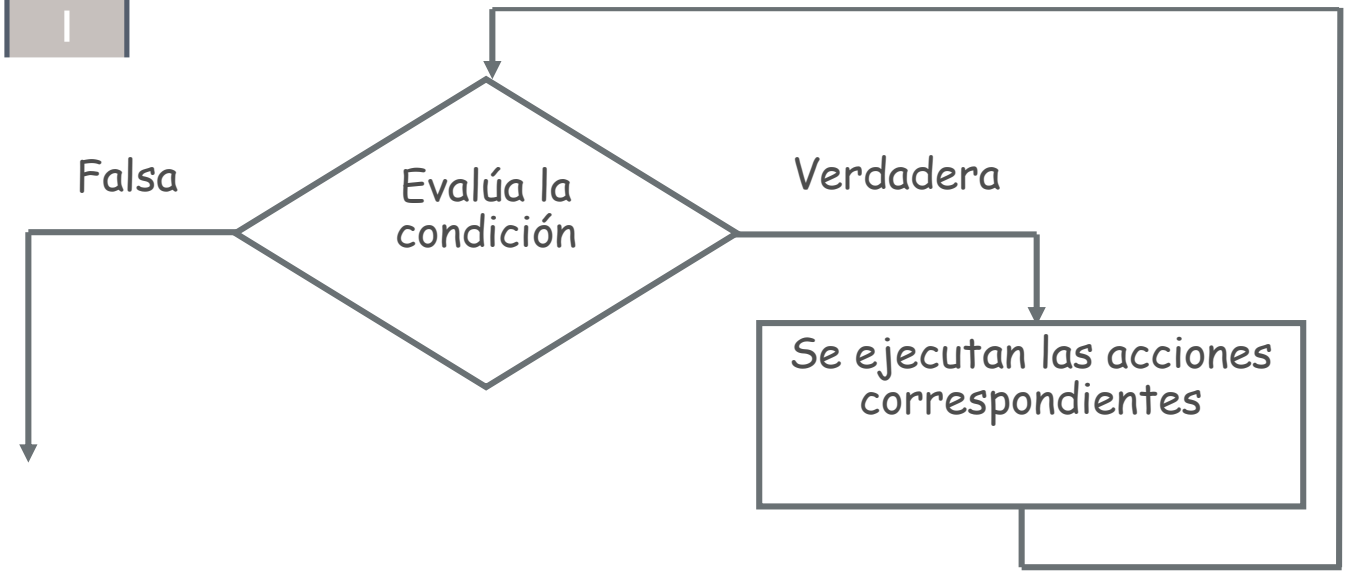
- Puede ocurrir que se desee ejecutar un bloque de instrucciones desconociendo el número exacto de veces que se ejecutan.
- Para estos casos existen en la mayoría de los lenguajes de programación estructurada las **estructuras de control iterativas condicionales.**
- Como su nombre lo indica las acciones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

Si 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.
```

Resolver

Si en el problema anterior, 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