TECNOLOGÍAS EN EDUCACIÓN MATEMÁTICA

MODULO 13

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

Pascal Estructuras de Control

- FLUJO DEL PROCESO
  - Secuenciales
    - Entrada de datos
    - Asignación
    - Salida de Datos
  - Condicionales (selección)
    - Simples
    - Dobles
    - Múltiples
  - Cíclicas o Repetitivas o Iterativas
    - Repetir x veces
    - Repetir numero desconocido de veces
      - Repetir mientras
      - Repetir hasta

Pascal - Condicionales

```
if (a > 0)
then
  write(1);
writeln('fuera');
```

```
if (a > 0)
then
  write(0)
else
  write(1);
writeln('fuera');
```

```
if (a > 0)
then
  begin
    writeln(0);
    writeln(1)
  end
else
 begin
    writeln(2);
    writeln(3)
 end;
writeln('fuera');
```

Pascal – Condicionales
Indentacion y Anidamiento

```
IF (A  <  0)
THEN
     IF(B  =  0)
      THEN
          write(1)
ELSE
     write(2);
```

Correcto

```
IF (A  <  0)
THEN
  IF(B  =  0)
   THEN
    write(1)
   ELSE
    write(2);
```

La indentación afecta a la legibilidad pero no modifica el **significado** del programa. El **else** queda ligado al **if** interno.

Pascal – Condicionales-
Indentacion y Anidamiento

El else "colgante"

Siempre se asocia al último then

```
IF (A < 0)
THEN
  IF (B = 0)
  THEN
    write(1)
  ELSE
    write(2);
```

```
IF (A < 0)
THEN
  BEGIN
    IF(B = 0)
    THEN
      write(1)
  END
ELSE
  write(2);
```

Pascal – Condicional CASE

Se utiliza cuando una o varias instrucciones
mutuamente excluyentes
dependen de una **expresión** de un tipo **escalar**.

Se evalúa una **expresión escalar**
y de acuerdo al valor computado
se selecciona la instrucción que se va a ejecutar.

Escalar: es un tipo no estructurado
en el que está definida una relación de orden.
Integer y *char* son escalares.

Pascal – Condicional CASE

Escribir un segmento de programa que muestre el peaje a pagar según la cantidad de ruedas del vehículo:

	Peaje
▪ 2 ruedas	$ 0
▪ 4 ruedas	$ 5
▪ 6 ruedas	$ 8
▪ 8 ruedas	$ 12

¿Cuáles son los datos de entrada?

¿Cuáles son los datos de salida?

¿Cuáles pueden ser los casos de prueba?

Pascal – Condicional CASE

```
if (ruedas = 2)
then peaje := 0
else
  if (ruedas = 4)
  then peaje := 5
  else
    if (ruedas = 6)
    then peaje := 8
    else
      if (ruedas = 8)
      then peaje := 12;
```

Pascal – Condicional CASE

```
if (ruedas = 2)
then peaje := 0
else
  if (ruedas = 4)
  then peaje := 5
  else
    if (ruedas = 6)
    then peaje := 8
    else
      if (ruedas = 8)
      then peaje := 12;
```

```
readln(cantRuedas);
case cantRuedas of
    2 : peaje := 0;
    4 : peaje := 5;
    6 : peaje := 8;
    8 : peaje := 12;
  end;
writeln ('Peaje ',  peaje);
```

Pascal – Condicionales - EJERCITACION

Elija la estructura condicional adecuada y resuelva:

Escriba un programa que lea un número entero y muestre carteles para indicar si:

- Es múltiplo de 2
- Es múltiplo de 3 y 5
- No es múltiplo de 7 ni de 11.
- Es múltiplo de 5 y no es múltiplo de 7.

Pascal – Condicionales - EJERCITACION

Elija la estructura condicional adecuada y resuelva:

Escriba un programa que lea una letra y muestre la que sigue en forma circular.

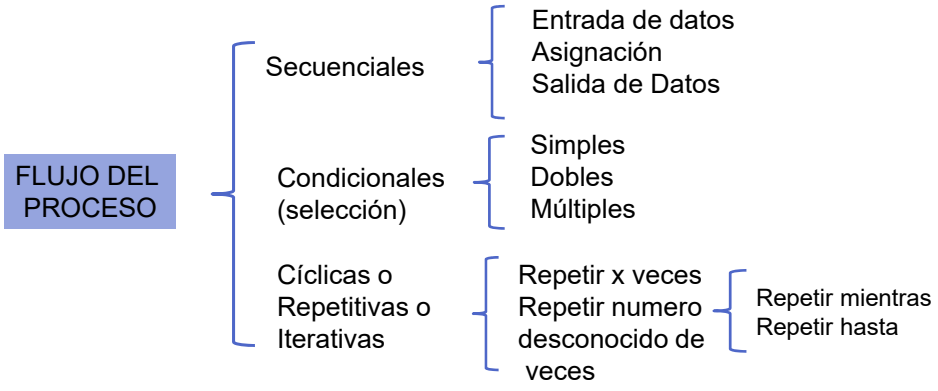
Es decir, la que sigue a la 'A' es la 'B', la que sigue a la 'Z' es la 'A.

Considere minúsculas y mayúsculas.

Si el usuario ingresa un carácter que no es una letra el programa muestra un mensaje de error.

Elija casos de prueba adecuados para verificar su solución.

Pascal Estructuras de Control



Pascal – Iteración – Con Contador

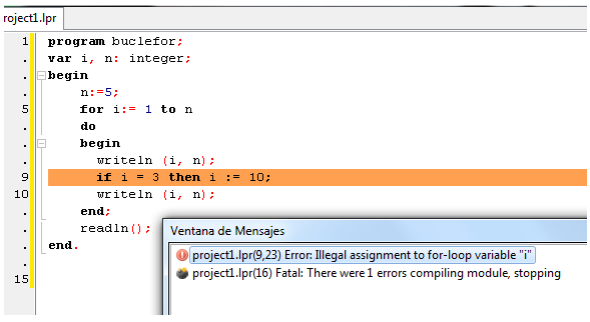
for *variable de control* := *valor inicial* to valor final do
instrucción simple o compuesta

- 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.

Pascal – Iteración – Con Contador

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

```
program bucleFor;
var i, n: integer;
begin
   n:=5;
   for i:= 1 to n
   do
   begin
     writeln (i, n);
     if i = 3 then
          i := 10;
     writeln (i, n);
   end;
end.
```

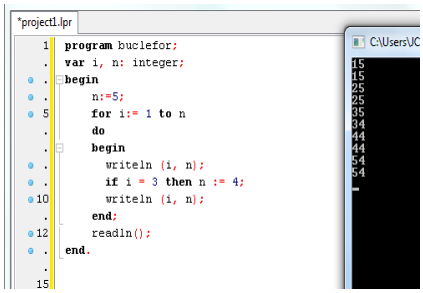


Error al compilar

Pascal – Iteración – Con Contador

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

```
program bucleFor;
var i, n: integer;
begin
   n:=5;
   for i:= 1 to n
   do
   begin
     writeln (i, n);
     if i = 3 then n := 4;
     writeln (i, n);
   end;
end.
```

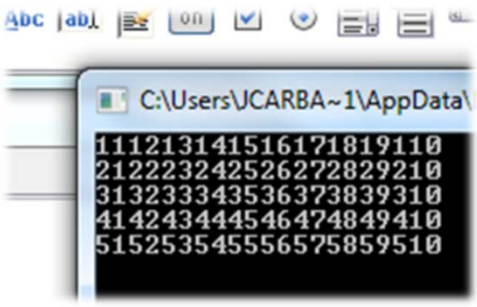


Oscurece la legibilidad del programa

Pascal – Iteración – Con Contador

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

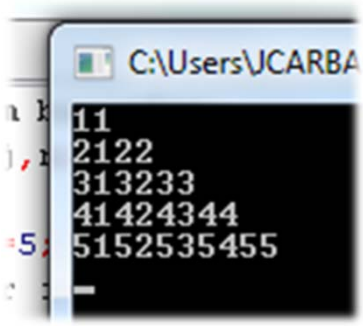
```
program bucleFor;
var i, j, n: integer;
begin
   n:=5;
   for i:= 1 to n
   do
   begin
     for j:= 1 to 10
     do
       write (i,  j);
     writeln;
   end;
end.
```



Pascal – Iteración – Con Contador

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

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



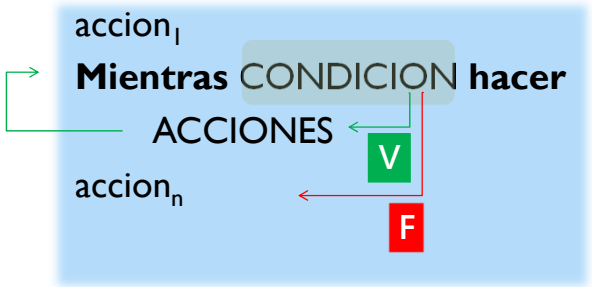
Pascal – Iteración – WHILE

while *expresión lógica* do
instrucción simple o compuesta

- La instrucción simple o compuesta es el **bloque iterativo** y puede no ejecutarse.
- Cuando la expresión lógica computa falso se produce la **condición de corte**.
- El bloque iterativo debe modificar al menos una de las variables involucradas en la expresión lógica para permitir que la **condición de corte** se produzca.
- Si la condición de corte no se verifica nunca, se produce un **bucle infinito**.
- Debe analizarse cuidadosamente la consistencia entre la inicialización de las variables, la condición de corte y el bloque iterativo.

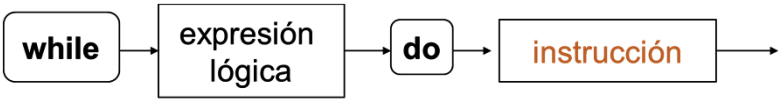
Pascal – Iteración – WHILE

while *expresión lógica* do
    instrucción simple o compuesta



Pascal – Iteración – WHILE

while *expresión lógica* do
    instrucción simple o compuesta



La instrucción del *while* es el **bloque iterativo**.

Cuando la expresión lógica es falsa
se produce la **condición de corte** de la iteración.

En el bloque iterativo debe modificarse
al menos una de las variables de la expresión lógica,
para permitir que la expresión lógica se haga falsa y el bucle termine.

Pascal – Iteración – WHILE

while *expresión lógica = V* do
    instrucción simple o compuesta

```
program bucleWhile;
var i, n: integer;

begin
   n:=5;   i:=1;
   while (i < n)
   do
     begin
       writeln (i, n);
       i:= i+1;
     end;
end.
```

```
program bucleWhile;
var i, n: integer;

begin
   n:=5;   i:=n;
   while (i < n)
   do
     begin
       writeln (i, n);
       i := i+1;
     end;
end.
```

Pascal – Iteración – WHILE

Problema: Determinar si un número N entero positivo es primo.

Un número natural N es un número primo si tiene **exactamente dos divisores**, 1 y N.

Una alternativa intuitiva para decidir si un número es primo es hallar el conjunto de sus divisores.

Una alternativa más **eficiente** es hallar, si existe, un divisor distinto a 1 y a N, porque en ese caso ya podemos asegurar que N no es primo.

Pascal – Iteración – WHILE

Problema: Determinar si un número N entero positivo es primo.

- Si 2 es un divisor de N (N>2) entonces N no es primo, no importa si tiene o no otros divisores.
- Análogamente si 3, 4, …, son divisores de N, entonces N no es primo (si N es distinto del divisor).

Para generalizar vamos a computar *N mod d* con d tomando valores 2, 3, 4 …

- ¿Cuándo terminamos?

Asumimos que N es primo hasta que encontremos un valor para d tal que N mod d = 0.

Pascal – Iteración – WHILE

Problema: Determinar si un número N entero positivo es primo.

```
…
  d:=2;
  esprimo := true;
  while (d < N) and (esprimo)
  do
      if (N mod d = 0)
         then
                  esprimo := false
         else
                  d := d + 1;
…
```

(esprimo=true)

Hacer trazas para n igual a 4, 5, 19 y 45.

Pascal – Iteración – WHILE

Problema: Determinar si un número N entero positivo es primo.

```
Program Nesprimo;
{determina si un número N ingresado por consola es primo,
asumiendo N > 1}
var N, d: integer;
    esprimo: boolean;
begin
{entrada}
  write ('ingrese N ');
  readln (N);
…
{salida}
  if esPrimo
    then     writeln(N, ' es primo ')
    else     writeln(N, ' no es primo');
end.
```

COMPLETAR!!!!!

Pascal – Iteración – REPEAT

```
repeat
      instrucción simple o compuesta
until expresión lógica = V
```

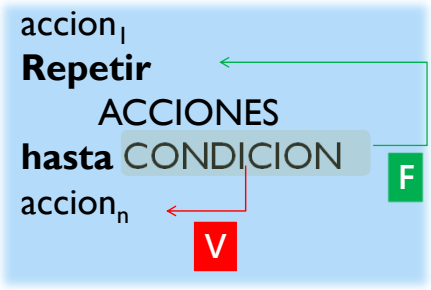
- La instrucción o secuencia de instrucciones es el **bloque iterativo** y se ejecuta por lo menos una vez.
- Cuando la expresión lógica computa **verdadero** se produce la **condición de corte**.
- El bloque iterativo debe modificar al menos una de las variables involucradas en la expresión lógica para garantizar que la condición de corte se produzca.
- Si la condición de corte no se verifica nunca, se produce un **bucle infinito**.
- Debe analizarse cuidadosamente la consistencia entre la inicialización de las variables, la condición de corte y el bloque iterativo.

Pascal – Iteración – REPEAT

repeat
    instrucción simple o compuesta
until *expresión lógica = V*

Repetir
    ACCIONES
hasta CONDICION

Expresión Lógica



Pascal – Iteración – REPEAT

repeat
    instrucción simple o compuesta
until *expresión lógica = V*

```
program enIntervalo2;
var a, b, num: integer; encontre: boolean;
begin
repeat
   writeln('Ingrese un valor positivo para a: ');
   readln(a)
until (a>=0);
repeat
   writeln('Ingrese un valor para b (mayor que a): ');
   readln(b)
until (b>a);
….
```

Se puede utilizar para validar los datos.

Pascal – Iteración - EJERCITACION

Problema 1: A partir de dos enteros, a y b (con a<b), calcular la cantidad de números en el intervalo [a, b] que son múltiplos de 2 y de 3.

Ejemplos:

a=12, b=20 ➔ la cantidad es 2 (el 12 y el 18).
a=0, b=2 ➔ la cantidad es 0.

¿Qué estructura de control deberíamos utilizar?

Pascal – Iteración - EJERCITACION

Problema 1: A partir de dos enteros, a y b (con a<b), calcular la cantidad de números en el intervalo [a, b] que son múltiplos de 2 y de 3.

```
cant:=0;
for num := a   to   b   do
  if (num mod 2=0) and (num mod 3=0)
       then
              cant:=cant + 1;
```

Pascal – Iteración - EJERCITACION

```
Program enIntervalo;
var a, b, cant: integer;
begin
writeln('Ingrese los valores de a y b: ');
readln(a, b);
cant:=0;
for num := a   to   b   do
  if (num mod 2=0) and (num mod 3=0)
        then
                cant:=cant + 1;
writeln('La cantidad de números que cumplen con la condición es: , cant);
end.
```

Pascal – Iteración - EJERCITACION

Problema 2: A partir de dos enteros positivos, a y b (con a<b) determinar si hay algún número en el intervalo [a, b] que sea múltiplo de 2 y de 3.

Ejemplos:

a=12, b=20 ➔ la respuesta es TRUE (el 12).
a=100, b=300 ➔ la respuesta es TRUE (el 102).
a= 0, b=2 ➔ la respuesta es FALSE.

¿Qué estructura de control deberíamos utilizar?

Pascal – Iteración - EJERCITACION

Problema 2: A partir de dos enteros positivos, a y b (con a<b) determinar si hay algún número en el intervalo [a, b] que sea múltiplo de 2 y de 3.

```
encontre:=false;
for num := a   to   b   do
  if (num mod 2=0) and (num mod 3=0)
        then
                encontre:=true;
```

Pascal – Iteración - EJERCITACION

Problema 2: A partir de dos enteros positivos, a y b (con a<b) determinar si hay algún número en el intervalo [a, b] que sea múltiplo de 2 y de 3.

¡¡OPTIMIZAMOS!!

```
encontre:=false;
num:=a;
while (num<=b) and (not encontre) do
 begin
  if (num mod 2=0) and (num mod 3=0)
        then
                encontre:=true;
  num:=num+1;
 end;
```

TECNOLOGÍAS EN EDUCACIÓN MATEMÁTICA

FIN MODULO 12

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