Resolución de Problemas y Algoritmos

Clase 6
Sentencias repetitivas
(iteraciones, ciclos, o bucles)

Fibonacci

Dr. Alejandro J. García
http://cs.uns.edu.ar/~ajg

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

Conceptos y vocabulario

- En un lenguaje de programación, las sentencias repetitivas permiten repetir un cierto número de veces un grupo (bloque) de sentencias.
- En Pascal este grupo se identifica con una sentencia compuesta, esto es, una secuencia de sentencias entre *begin* y *end*.
- A la **repetición** también se la llama **iteración**.
- Al repetir (o iterar) una secuencia, esta secuencia vuelve a comenzar una y otra vez. Es por esto que también, en la jerga informática, se usan las palabra **ciclo**, **lazo** o **bucle**. *En inglés:* ***loop***.
- Por ejemplo: en la siguiente iteración, la secuencia de sentencias dentro del ciclo FOR se ejecutará 10 veces.

```
FOR V:= 1 TO 10 DO
  begin
   … secuencia de sentencias…
  end;
```

Ciclo FOR o bucle FOR.
En inglés: "FOR-DO Loop".

Sentencia FOR-TO (repaso)

```
FOR V:= exp1 TO exp2 DO sentencia
```

- ***exp1*** y ***exp2*** pueden ser valores, variables, o expresiones del mismo tipo ordinal que la variable V *(no puede ser tipo real).*
- El resultado de ***exp1*** y ***exp2*** deben poder calcularse justo antes de la ejecución del ciclo FOR. La variable **V** toma el valor inicial de evaluar ***exp1*** y luego, en cada iteración, **V** se **incrementa** automáticamente de a uno hasta llegar al valor de ***exp2***.

```
N:=3;
FOR V := N TO SQR(N)+N DO writeln(V);
read(num);
FOR V := num div 2 TO (num+10) - N DO writeln(V);
```


Sentencia FOR-DOWNTO

```
FOR V:= exp1 DOWNTO exp2 DO sentencia
```

- ***exp1*** y ***exp2*** pueden ser valores, variables, o expresiones del mismo tipo ordinal que la variable V *(no puede ser tipo real).*
- El resultado de ***exp1*** y ***exp2*** deben poder calcularse justo antes de la ejecución del bucle FOR-DOWNTO. La variable **V** toma el valor inicial de evaluar ***exp1*** y luego, en cada iteración, **V** se **decrementa** automáticamente de a uno hasta llegar al valor de ***exp2***.
- Si ***val1*** es el valor de ***exp1*** y ***val2*** el de ***exp2***, entonces la ***sentencia***, que puede ser compuesta, se repetirá (***val1*** – ***val2*** + 1) veces.
- Si ***val2*** es mayor estricto a ***val1*** entonces se repetirá 0 veces.

```
FOR numero := 10 DOWNTO 0 DO write(numero);
```

```
FOR letra:= 'Z' DOWNTO 'A' DO write(letra);
```


FOR-DOWNTO. Ejemplo.

```
PROGRAM CuentaRegresiva;
{Imprime en pantalla los dígitos de 9 a 0 en ese orden
 y las letras de G a A en ese orden}
VAR dig:integer; letra:char;
BEGIN
writeln('Cuenta regresiva');
FOR dig:=9 DOWNTO 0
  DO write(dig,' ');
writeln;
writeln('----------');
FOR letra:='G' DOWNTO 'A'
   DO write(letra,' ');
writeln;
writeln('----------');
END.
```

Cuenta regresiva
9 8 7 6 5 4 3 2 1 0

G F E D C B A
---

dig	letra
?	?

Repeticiones anidadas

```
FOR v:= 1 TO 3 DO
  writeln( v );
```

¿cuántas veces se ejecuta writeln(v)?

```
FOR v:= 1 TO 3 DO
  FOR h:= 1 TO 2 DO
    writeln( v, h);
```

¿cuántas veces se ejecuta writeln(v,h)?

```
FOR v:= 1 TO 3 DO
  FOR h:= 1 TO 2 DO
    FOR t:= 5 downto 1
    DO writeln(v, h, t);
```

Obs: usan diferentes variables de control ¿Por qué?

¿cuántas veces se ejecuta writeln(v,h,t)?

Repeticiones anidadas: ejemplo

```
PROGRAM combinaLetras;
{muestra las 27 combinaciones
de las letras A B y C}
VAR L1,L2,L3: CHAR;
BEGIN
FOR  L1 := 'A'  TO   'C'  DO
    FOR L2 :=  'A'  TO   'C'  DO
       FOR L3 :=   'A'   TO    'C'   DO
              writeln (L1,L2,L3);
END.
```

```
AAA
AAB
AAC
ABA
ABB
ABC
...
```

L1	L2	L3
?	?	?
A	A	A
A	A	B
A	A	C
A	B	A

- **Problema propuesto: Un dominio automotor (patente) es una combinación de 3 letras y 3 dígitos. Escriba un programa que muestre TODAS las patentes posibles. ¿Cuántas son?**

Sentencias FOR-TO anidadas. Ejemplo.

```
PROGRAM FOR_anidados;
{ejemplo de bucles FOR anidados donde el ciclo
interno tiene un límite (cant) que va cambiando en
cada iteración}
VAR Letra: char; i,cant: integer;
BEGIN
cant:=1; {cantidad de letras por renglón}
FOR  Letra := 'A'  TO  'E'
DO
  BEGIN  {imprime un renglón de "Letra"s}
    FOR i := 1 TO cant DO write(Letra);
    writeln; {baja de renglón}
    cant:=cant+1;{incrementa la cantidad por renglón}
  END;
END.
```

```
A
BB
CCC
DDDD
EEEEE
```

Realice la traza.


```
PROGRAM FOR_anidados2;
VAR    cant, num, pri, ult : integer;
BEGIN {otro ejemplo de límites que cambian}
writeln('ingrese 2 dig.'); readln(pri, ult);
{primera mitad incrementando}
FOR  cant := pri  TO ult DO
    BEGIN
    FOR num := pri TO  cant DO write(num);
    writeln;
    END;
{segunda mitad decrementando}
FOR  cant := ult-1  DOWNTO pri DO
    BEGIN
    FOR num := pri TO  cant DO write(num);
    writeln;
    END;
END.
```

```
ingrese 2 dig.:
2 5

2
23
234
2345
234
23
2
```


Problema para practicar

- **Problema propuesto: Escriba un programa que pida al usuario un valor 0 < N < 10 y dibuje una forma como la que sigue (por ejemplo para N=4):**

```
ingrese N: 4

1
22
333
4444
333
22
1
```

1. Entender el problema
2. Buscar solución:
   3. Buscar ejemplos particulares
   4. Dividir el problema en partes
5. Escribir algoritmo
6. Verificar con una traza.
7. Escribir programa
8. Verificar con una traza.

Observaciones sobre repetición incondicional

FOR V:= *Exp1* TO (DOWNTO) *Exp2* DO *sentencia*

- Tanto en un ciclo FOR-TO como en FOR-DOWNTO, el valor de *Exp1* (llamado *valor inicial*) y el de *Exp2* (llamado *valor final*) deben poder calcularse al comenzar la repetición, de lo contrario es un error de programación.

```
PROGRAM Incorrecto;
{¿Por qué es incorrecto? }
VAR v, inicio, tope: integer;
BEGIN
tope:=10;
FOR  v := inicio  TO  tope
DO    writeln (v);
END.
```

MAL

V	inicio	tope
?	?	?
		10

Error de programación: inicio sin valor

Muy importante

FOR V:= <expresión1> TO (DOWNTO) <expresión2> DO <Bloque de Sentencias>

Ya sea para un ciclo FOR-TO o FOR-DOWNTO, en RPA será considerado error de programación:

1) **Cambiar el valor de la variable de control V, dentro del bloque de sentencias de un ciclo FOR.**
2) **Cambiar el valor de cualquier variable de <expresión1> o <expresión2>, dentro del bloque de sentencias de un FOR, con motivo de que cambien los límites de la iteración.**

Si surge la necesidad de hacerlo es porque tendría usar una repetición condicional con **REPEAT** o **WHILE** (que veremos en breve…).

```
…
FOR  V:= 1 TO 100
DO begin
   writeln(V);
   V:= V + 5;
end;
…
```

MAL

```
…
FOR  V:= 1 TO 100 DO
begin
   writeln(V);
   if V=12 then V:= 100;
end;
…
```

MAL

Error de programación

Importante: es un error de programación intentar controlar "manualmente" la variable de control o los límites de un for.

Lazarus dice: *Error: Illegal assignment to for-loop variable "v"*
Estos es: Error: asignación ilegal a la variable de control "v"

Error de programación

```
…
ultimo:= 100;
FOR  V:= 1 TO ultimo DO
begin
   writeln(V);
   if V = 12 then ultimo:=13;
end;
…
```

MAL

Es un error pensar que cambiando el valor del límite, la cantidad de repeticiones de un FOR puede cambiar o dejar de repetir.

Importante: es un error de programación intentar controlar "manualmente" la variable de control o los límites de un for.

Usos permitidos de la variable de control

- **La variable de control si puede usarse y modificarse en otras partes del programa que estén "afuera" del ciclo FOR.**
- **Como muestra el ejemplo 1, si es posible cambiar la variable de control V antes o después del ciclo FOR.**
- **En el ejemplo 2 se muestra que la misma variable de control V también usarse como variable de control para otro ciclo FOR siempre que uno no esté anidado en el otro.**

```
{ejemplo 1: válido}
V:= 9;
writeln(V);
FOR  V:= 1 TO 3
DO   writeln(V);
V:= 8; writeln(V);
```

```
{ej 2: válido}
FOR  V:= 1 TO 3
DO   writeln(V);
FOR  V:= 4 TO 9
DO   writeln(V);
V:= 4; writeln(V);
```

```
{ej 3: inválido}
FOR  V:= 1 TO 3 DO
begin
   FOR  V:= 4 TO 9
   DO   writeln(V);
end;
```

MAL

Motivación

Hay muchos problemas en los cuales se debe repetir una secuencia de acciones pero no se conoce de antemano el número de veces que se repetirá.

Por ejemplo:

Algoritmo: llenar bidón
Repetir
  trasvasar jarrito
hasta bidón lleno

Algoritmo:
Repetir mientras hay productos
- **tomar producto**
- **esperar por envase vacío**
- **poner producto en envase**
- **cerrar envase**

Algoritmo: subir una escalera:
mientras hay escalones
  subir un escalón

Repetición basada en condiciones

Por ejemplo, considere que se quiere validar el ingreso de datos y se quiere repetir el ingreso de datos mientras estos no sean correctos.

```
mostrar('Ingrese una letra mayúscula')
leer(letra)
{ validación de los datos ingresados por el usuario }
MIENTRAS  ( letra < 'A') o (letra > 'Z') {i.e. ,no sea mayúscula}
     mostrar(' Error, ingrese una letra mayúscula')
     leer(letra)
fin repetir
{… Datos validados: si llega a este punto
     es porque letra tiene una mayúscula…}
```


Conceptos: sentencias repetitivas en Pascal

Repetición incondicional:

```
FOR <ini> TO <fin> DO
  <sentencia>
FOR <ini> DOWNTO <fin> DO
  <sentencia>
```

Repetición condicional:

```
WHILE <condición> DO
  <sentencia>
```

Repetición condicional:

```
REPEAT
  <sentencias>
UNTIL  <condición>
```

Repetición condicional en Pascal: WHILE

La sentencia de un ciclo WHILE se ejecutará 0 (cero) o más veces dependiendo del resultado (true o false) que se obtiene al evaluar la expresión booleana que representa la condición.

Primero se evalua la expresión booleana

si el resultado es TRUE se ejecuta la sentencia que sigue al DO

```
WHILE expresión booleana
DO   sentencia ;
Otra sentencia siguiente;
```

TRUE

FALSE

Una vez ejecutada la sentencia que sigue al DO, se vuelve a evaluar la expresión booleana

si el resultado es FALSE, saltea (no ejecuta) la sentencia del DO y sigue en la siguiente al while.

"While loop": repetición condicional

Las sentencias dentro de un WHILE se ejecutan 0 (cero) o más veces.

Casos de prueba: tope con -1, 0, 2

Obs: si la sentencia del while se repitió N veces, la expresión cont < tope se evaluó N+1 veces.

Mientras cont < tope sea true ejecuta:

Si cont < tope es false sigue en:

```
programa ejemplo;
var tope, cont: integer;
begin
Write('Ingre un tope: ');
readln(tope);
cont := 0;
WHILE cont < tope
DO   Begin
       writeln(cont);
       cont:=cont+1;
     End;
Writeln('press enter');
end.
```

tope	cont	cont<tope
?	?	?
1	0	T
	1	F

Sentencia simple vs. compuesta

```
N := 0;
WHILE  N < 3  DO
       N := N + 1;
writeln(N);
```

Importante: Si en un WHILE queremos que se repita más de una sentencia (un **bloque** de sentencias), debemos usar la sentencia compuesta con BEGIN-END

```
N := 0;
WHILE  N < 3  DO
       N := N + 1;
       writeln(N);
```

```
N := 0;
WHILE  N < 3  DO
    BEGIN
        N := N + 1;
        writeln(N);
    END;
```



```
PROGRAM EjemploWhile; {muestra una aplicación útil del WHILE}
VAR letra: char; opcion:integer;
BEGIN
  writeln('Ingrese una letra mayúscula');  read(letra);
  { validación de los datos ingresados por el usuario}
  WHILE ( letra < 'A') or (letra > 'Z')  DO
  BEGIN writeln('Incorrecto.  Ingrese letra mayúscula');
        read(letra);
  END;
  writeln('Ingrese una opción: ');
  write(' (1)- pasar a minúscula. (2)- mostrar código ASCII ');
  read(opcion);
{ validación de los datos ingresados por el usuario}
 WHILE ( opcion <> 1) and ( opcion <> 2)  DO
  BEGIN  write(' Incorrecto. Ingrese 1 o 2');  read(opcion);
  END
{… resto del programa…}
```


Repetición basada en condiciones

Por ejemplo, se quiere permitir al usuario repetir la ejecución del programa hasta que el decida.

```
REPETIR

 { esta parte del programa se repetirá hasta que el
                usuario indique fin}
   mostrar "Quiere ejecutar nuevamente  (S) si  (N) no"
   leer (letra)
HASTA ( letra = 'N') o ( letra = 'n');
{se dejará de repetir si presiona la tecla N (may. o min.)}

mostrar " Muchas gracias por utilizar el programa"
```


Repetición condicional en Pascal

Las sentencias dentro de un REPEAT-UNTIL se ejecutan 1 o más veces dependiendo del resultado (true o false) que se obtiene al evaluar la expresión booleana que representa la condición.

```
REPEAT
   <sentencia 1>
   <sentencia 2>
     . . .
   <sentencia  n>
UNTIL <exp. boolean>
```

FALSE

Si el resultado es false vuelve a ejecutar la secuencia a partir de la palabra reservada repeat

TRUE

Si el resultado es true sigue en

<sentencia siguiente al repetir>

Repetición condicional en Pascal

```
Write('Tope='); read(tope);
cont := 0;
REPEAT
 writeln(cont);
 cont:=cont+1;
UNTIL cont >= tope;
Writeln('----------');
```

Si el resultado es false vuelve a

Si el resultado es true sigue en

- **Tarea: escriba una vesión de la "validación de datos" pero con REPEAT-UNTIL en lugar de while.**

Ejemplo con REPEAT-UNTIL

```
PROGRAM EjemploRepeat; {muestra una aplicación útil del repeat}
VAR letra: char;
BEGIN
REPEAT
   { esta parte del programa se repetirá hasta que
   el  usuario indique fin}
  writeln('Quiere ejecutar nuevamente el programa');
  write(' (S) si  (N) no'); readln(letra);
  {se dejará de repetir si presiona la tecla N (may. o min.)}
UNTIL ( letra = 'N') or ( letra = 'n');
writeln('Muchas gracias por utilizar el programa.');
END.
```


Conceptos: Diferencias REPEAT y WHILE

REPEAT-UNTIL

- si condición es **falsa** sigue repitiendo.
- si condición es **verdadera** deja de repetir.
- **repite 1 o más veces**: siempre ejecuta al menos una vez la secuencia interna al repetir

WHILE

- si condición es **verdadera** sigue repitiendo
- si condición es **falsa** deja de repetir
- **repite 0 o más veces**: puede no ejecutar nunca la secuencia interna al repetir

- **Ejercicio propuesto para practicar: escriba las diferencias y similitudes entre las tres sentencias repetitivas FOR, WHILE y REPEAT.**

Repeticiones anidadas (algunos ejemplos)

```
REPEAT
  WHILE <condición>
  DO  WHILE <condic.>
      DO   <sent. >
UNTIL <condición>
```

```
WHILE <condición>
DO  FOR  v:= … TO … DO
      <sentencia >
```

```
REPEAT
   REPEAT
      <sentencia >
   UNTIL <condición>
UNTIL <condición>
```

El límite está en la imaginación del programador ☺

Conceptos: repetición condicional vs. incondicional

- **La repetición condicional (REPEAT o WHILE) depende de una condición (expresión boolean).**
- **La repetición incondicional (FOR) se ejecuta un número fijo de veces que se conoce antes de comenzar a repetirse la sentencia.**
- **Toda vez que se puede usar una repetición incondicional (FOR), el código podría reescribirse para usarse una repetición condicional (while o repeat).**
- **Pero no todas las repeticiones condicionales (while o repeat) pueden reescribirse para usar una incondicional (FOR).**

Repetición condicional es MÁS GENERAL

```
a:=1;
FOR  v:=1 TO 5
DO a:=a*v;
Write(a);
```

```
a:=1;  v:=1;
REPEAT
   a:=a*v;
   v:=v+1;
UNTIL v > 5
Write(a);
```

```
a:=1;  v:=1;
WHILE v <= 5
DO BEGIN
   a:=a*v;
   v:=v+1;
END
Write(a);
```

```
Write('ingrese nro. positivo: ');
Read(num);
WHILE num < 0 DO Read(num);
```

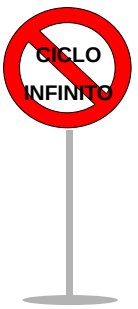
```
Write('ingrese nro. positivo: ');
REPEAT Read(num);
UNTIL num >= 0;
```

```
FOR ?? TO ??
DO
```

Concepto: CICLO INFINITO ☹

Una repetición que se realiza infinitas veces se denomina ciclo infinito

- **En RPA un ciclo infinito en un programa será considerado un ERROR GRAVE de programación.**
- **Cuando realice la traza de sus programas debe asegurarse que sus repeticiones que no exista ningún caso en que su programa pueda caer en un ciclo infinito.**



Conceptos: CICLOS INFINITOS ☹

Una repetición condicional mal programada puede caer en una REPETICIÓN INFINITA, lo cual es un error grave de programación

```
{...OK...}
v:=1; w:=1;
REPEAT
   v:=v+1;
UNTIL V = 9;
Write(V);
```

```
{ 1. MAL  }
v:=1; w:=1;
REPEAT
   v:=v+1;
UNTIL w = 0;
Write(V);
```
MAL

```
{ 2. MAL }
v:=1; w:=1;
WHILE V<>9
DO v:=1;

Write(V);
```
MAL

```
{ 3. MAL }
v:=1; w:=1;
REPEAT
   v:= w-1;
UNTIL V = 9;
Write(V);
```
MAL

Algunos problemas clásicos:

1. **La condición de corte es errónea.**
2. **La variable de la condición no se modifica.**
3. **La variable de la condición se modifica mal.**

Problema Propuesto

- **Escriba un programa que calcule el promedio de de números reales ingresados por el usuario.**

```
Ingrese un valor: 8.2
¿otro? s/n  s
Ingrese un valor: 0.2
¿otro? s/n  s
Ingrese un valor: -3.0
¿otro? s/n  s
Ingrese un valor: 5.2
¿otro? s/n  n
El promedio es: 2.65
```


Fibonacci (1170 -1250)

Leonardo de Pisa, matemático italiano.
El apodo de su padre era *Bonacci* (bien intencionado) y él recibió el apodo *Fibonacci: filius* (hijo de) Bonacci.

Su padre era comerciante, y Fibonacci de joven vivió en África, donde estudió con los matemáticos árabes más destacados de ese tiempo.

Allí aprendió el sistema de numeración árabe (decimal).

Consciente de la superioridad de este sistema comparado con el romano, en 1202, a los 32 años de edad, publicó lo que había aprendido en el **Liber Abaci** (libro del ábaco o libro de los cálculos), mediante el cuál se introdujo en Europa el sistema decimal que reemplazaría al romano.

http://es.wikipedia.org/wiki/Leonardo_de_Pisa

Una página del libro *Liber Abaci* en la Biblioteca Nazionale di Firenze mostrando en el recuadro de la derecha la secuencia de Fibonacci en números Romanos y numeros Arábigos.

Sistema de numeración decimal (base 10)

- El sistema de numeración decimal se considera uno de los avances más significativos de las matemáticas.
- La mayoría de los historiadores coinciden en afirmar que tuvo su origen en la India (Tamil) en 300 aC (pero también se especula que tuviera sus orígenes en China).
- Este sistema de numeración llegó a Oriente Medio hacia el año 670. al-Jwarizmi escribió el libro "Acerca de los cálculos con los números de la India" cerca de el año 825.
- En Europa se utilizaban los números Romanos, pero Fibonacci, que había estudiado en Bugía (en la actual Argelia), contribuyó a la difusión por Europa del sistema arábigo con su libro *Liber Abaci*, publicado en 1202.

http://es.wikipedia.org/wiki/Números_arábigos

Evolución de los símbolos de los dígitos

1200	**Europeo**	0	1	2	3	4	5	6	7	8	9
	Arábico-Índico	٠	١	٢	٣	٤	٥	٦	٧	٨	٩
670	**Arábico-Índico Oriental** (Persa y Urdu)	٠	١	٢	٣	۴	۵	۶	٧	٨	٩
	Devanagari (Hindi)	०	१	२	३	४	५	६	७	८	९
300aC	**Tamil (India)**		க	உ	ரு	சு	ரூ	கூ	எ	அ	கை

Sucesión de Fibonacci

La **sucesión de Fibonacci** es una sucesión infinita de números naturales que inicia con 0 y 1, y a partir de ahí cada elemento es la suma de los dos anteriores:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, …
Cada elemento de esta sucesión se llama **número de Fibonacci**.

```
anterior ←0    mostrar(0)
ultimo ←1      mostrar(1)
repetir
    nuevo ← ultimo + anterior
    mostrar(nuevo)
    anterior ← ultimo
    ultimo ← nuevo
hasta nuevo > tope
```

La sucesión fue descrita por Fibonacci, en su libro *Liber Abaci,* como la solución a un problema de la cría de conejos.

Antes de que Fibonacci escribiera su trabajo, la sucesión de los números de Fibonacci había sido descubierta por matemáticos indios tales como Gopala (antes de 1135) y Hemachandra (1150),