Resolución de Problemas y Algoritmos

Clase 7
Repetición condicional

Fibonacci

Dr. Diego R. García

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

Uso de repetición en un algoritmo

Realizar un algoritmo para envasar todos los productos disponibles, usando estas primitivas:

Primitivas disponibles (ordenadas alfabéticamente):

- cerrar envase
- esperar por envase vacío
- hay productos
- poner producto en envase
- tomar producto

Algoritmo: envasar productos
Repetir mientras hay productos

- tomar producto
- esperar por envase vacío
- poner producto en envase
- cerrar envase

Motivación

Existen algoritmos donde:

- **se debe repetir una secuencia de acciones, pero**
- **no se sabe de antemano cuantas veces se van a repetir.**

Por ejemplo:

Algoritmo: cursada
Repetir
intentar cursar materia
hasta materia cursada

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

Resolución de Problemas y Algoritmos Dr. Diego R. García 3

Conceptos: sentencias repetitivas en Pascal

Repetición incondicional (repite un número fijo de veces):

```
FOR var := <ini> TO <fin> DO   <sentencia>

FOR var := <ini> DOWNTO <fin> DO  <sentencia>
```

Repetición condicional (depende de una condición):

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

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

Importante: consultar los diagramas sintácticos de Pascal (aquí) para conocer los detalles de la sintaxis de todas estas sentencias.

Resolución de Problemas y Algoritmos Dr. Diego R. García 4

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…}
```

Resolución de Problemas y Algoritmos Dr. Diego R. García 5

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

WHILE expresión booleana
DO sentencia ;
Otra sentencia siguiente;

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

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.

Resolución de Problemas y Algoritmos Dr. Diego R. García 6

Ejemplo de aplicación de WHILE

Validar el ingreso de datos y repetir el ingreso de datos mientras estos no sean correctos.

```
PROGRAM EjemploWhile;
{muestra una aplicación útil del WHILE}
VAR letra: char;
BEGIN
 writeln('Ingrese una letra mayúscula');
 readln(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;
 {… aquí estoy seguro que letra tiene una mayúscula y
      continúo con el resto del programa…}
```

Casos de prueba:

(letra < 'A') OR (letra > 'Z')

letra | ¿repite?

letra	(letra < 'A') OR (letra > 'Z')	¿repite?
A	FALSE	no
a	TRUE	sí
N	FALSE	no
n	TRUE	sí
$	TRUE	sí
1	TRUE	sí

Ejemplo de aplicación de WHILE

Validar el ingreso de datos y repetir el ingreso de datos mientras estos no sean correctos.

```
PROGRAM EjemploWhile;
{muestra una aplicación útil del WHILE}
VAR opcion:integer;
BEGIN
 writeln('Ingrese opción: (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…}
```

Casos de prueba:

(opcion<>1) and (opcion<>2)

opcion | ¿repite?

opcion	(opcion<>1) and (opcion<>2)	¿repite?
1	FALSE	no
0	TRUE	sí
2	FALSE	no
5	TRUE	sí

"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');
readln;
end.
```

tope	cont	cont<tope
?	?	?

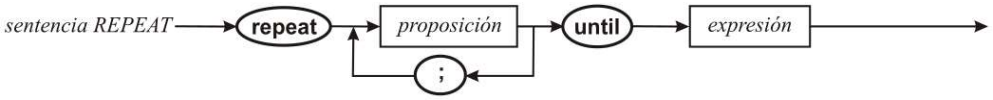
Repetición basada en condiciones

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 "Pulse la letra (S) para ejecutar nuevamente o
            cualquier otra letra para salir de la aplicación"
   leer (letra)
HASTA que se pulse letra distinta de 'S' ;
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 <expresión boolean>

<sentencia siguiente >
```

FALSE

TRUE

Si el resultado es FALSE vuelve a ejecutar la secuencia a partir de la palabra reservada REPEAT

Si el resultado es TRUE no vuelve a repetir y sigue en la sentencia siguiente a UNTIL

Resolución de Problemas y Algoritmos Dr. Diego R. García 11

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 lo indique}
 writeln('Pulse la letra (S) para ejecutar nuevamente o
          cualquier otra letra para salir de la aplicación');
   readln(letra);
  {se volverá a repetir  si presiona la tecla S }
UNTIL (letra<>'S') and (letra<>'s');
writeln('Muchas gracias por utilizar el programa.');
END.
```

Casos de prueba:

(letra<>'S') and (letra<>'s')

letra

¿repite?

S	FALSE	sí
s	FALSE	sí
N	TRUE	no
n	TRUE	no
$	TRUE	no
1	TRUE	no

Resolución de Problemas y Algoritmos Dr. Diego R. García 12

"Repeat loop": repetición condicional

```
programa ejemplo2;
var tope, cont: integer;
begin
  Write('Ingrese un tope: ');
  readln(tope);
  cont := 0;
  REPEAT
    writeln(cont);
    cont:=cont+1;
  UNTIL cont >= tope;
  Writeln('press enter');
  readln;
end.
```

Las sentencias dentro de un REPEAT se ejecutan 1 (una) o más veces.

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

Si el resultado es false vuelve a repetir desde

FALSE

TRUE

Si el resultado es true sigue en

tope	cont	Cont>=tope
?	?	?

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 a 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 a 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);
```

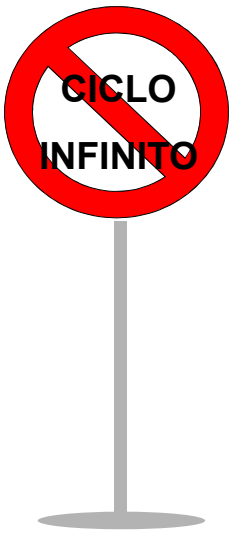
```
Write('ingrese nro. positivo: ');
REPEAT Read(num);
UNTIL num >= 0;
```

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 en las repeticiones no exista ningún caso en el cual el 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 = 3;
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<>3
DO v:=1;

Write(V);
```

MAL

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

MAL

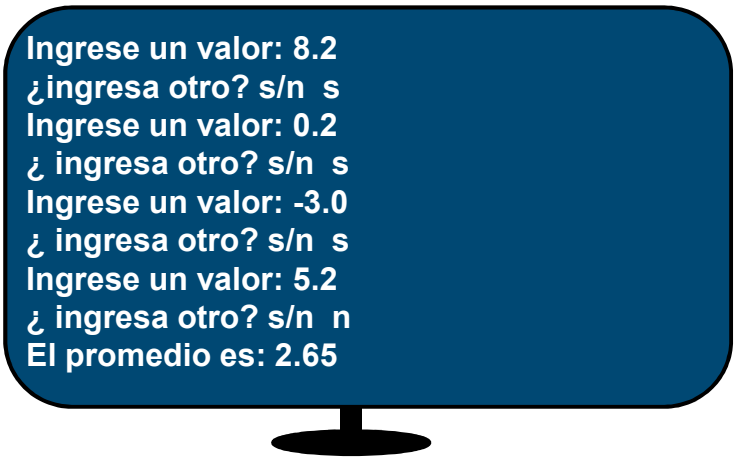
Algunos problemas clásicos:

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.**

Resolución de Problemas y Algoritmos Dr. Diego R. García 19

Problema propuesto para practicar

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



Resolución de Problemas y Algoritmos Dr. Diego R. García 20

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),

Resolución de Problemas y Algoritmos Dr. Diego R. García 21

Problemas propuestos (y posibles soluciones)

A continuación se incluyen algunos problemas para practicar repetición condicional:

1. **Indicar cuántas veces aparece un número (que llamaremos "buscado") en una secuencia de números terminada en -1.**
2. **Indicar cuántas veces aparece un símbolo ASCII (que llamaremos "buscado") en una oración terminada en un punto.**

También se incluyen posibles soluciones.
No obstante se sugiere en primer lugar intentar resolverlos siguiendo la metodología propuesta y luego acudir a la solución para comparar con su solución.

Resolución de Problemas y Algoritmos Dr. Diego R. García 22

Problema propuesto resulto (para comparar)

Indicar cuántas veces aparece un número (que llamaremos "buscado") en una secuencia de números terminada en -1.

Ejemplo: buscado=5 está 3 veces en: 12 5 26 41 5 15 0 5 -1

¿qué otros ejemplos significativos propone? (casos de prueba)

secuencia vacía: solo con -1, o buscado=33 en 12 5 3 4 -1

Solución: Recorrer la secuencia de principio a fin e ir contabilizando la cantidad de veces que aparece buscado.

Algoritmo:

leer(buscado) y cantidad ←0
leer un número de la secuencia (leído)
repetir mientras leído sea distinto de -1
  si leído = buscado entonces cantidad ← cantidad +1
  leer otro número de la secuencia (leído)
mostrar valor de cantidad

Resolución de Problemas y Algoritmos Dr. Diego R. García 23

Una posible implementación en Pascal

```
Program veces;  // Calcula cuantas veces aparece un nro. en una secuencia.
const terminador = -1;    // el -1 no pertenece a la secuencia, indica el final
var buscado,leido,cantidad:integer;
Begin
writeln(' ingrese número a buscar'); readln(buscado);
writeln(' ingrese una secuencia de números (use -1 para indicar el final): ');
cantidad:=0;
read(leido);  //leo el primero
WHILE leido <> terminador DO // controlo que no sea el finalizador
  begin
    if leido = buscado then cantidad:=cantidad+1; // encontré otro buscado
    read(leido); // leo el siguiente en la secuencia
  end;
readln; // lee el enter que se introdujo después del terminador
writeln('Cantidad de veces que está', buscado, '= ',cantidad);
readln; // espera otro enter para no cerrar la consola
end
```

Resolución de Problemas y Algoritmos Dr. Diego R. García 24

Problema propuesto

Indicar cuántas veces aparece un símbolo ASCII (que llamaremos "buscado") en una oración terminada en un punto.

Ejemplos (significativos): que buscado esté en la secuencia
el símbolo 'a' está 2 veces en: La hora de RPA.
que buscado no esté: '2' no está en la secuencia anterior
que la secuencia sea vacía (solo un punto)

Solución: Recorrer la oración de principio a fin e ir contando la cantidad de veces que aparece el símbolo buscado.

Algoritmo:

leer(buscado) y cantidad ←0
leer un símbolo de la secuencia (leído)
repetir mientras leído sea distinto de '. '
si leido = buscado entonces cantidad ← cantidad +1
leer otro símbolo de la secuencia (leído)
mostrar valor de cantidad

Resolución de Problemas y Algoritmos Dr. Diego R. García 25

Una posible implementación en Pascal

```
Program veces; // Calcula cuantas veces aparece un char en una secuencia.
const terminador = '.' ;   // el terminador es un punto
var buscado,leido:char; cantidad: integer;
Begin
writeln(' ingrese símbolo a buscar'); readln(buscado);
writeln(' ingrese una secuencia de símbolos: ');
cantidad:=0;
read(leido);  //leo el primer carácter
WHILE leido <> terminador DO // controlo que no llegué al punto final
  begin
    if leido = buscado then cantidad:=cantidad+1; // encontré otro
    read(leido); // leo el siguiente en la secuencia
  end;
readln; // lee el enter que está después del punto, al final de la secuencia
writeln('Cantidad de veces que está', buscado, '= ',cantidad);
readln; // espera otro enter para no cerrar la consola
end.
```

Resolución de Problemas y Algoritmos Dr. Diego R. García 26

Cambiando el terminador

- **La solución anterior no permite que se busque el carácter "." (punto) en una secuencia.**
- **Se podría cambiar la constante "terminador" por otro carácter.**
- **Una alternativa es usar el "enter" como carácter terminador de la secuencia, ya que el "enter" igualmente también está en el buffer de entrada de donde la primitiva "read" obtiene los valores.**
- **Pero:**

  ¿cuál es el carácter que corresponde a "enter"?

¿Qué es ENTER?

En las máquinas de escribir mecánicas al finalizar un renglón había que hacer dos movimientos: (1) retorno de carro (2) nueva línea

En algunos sistemas operativos ENTER tiene asociados 2 caracteres:

(1) **ASCII 13**: retorno de carro (CR: carriage return)

(2) **ASCII 10**: nueva línea (LF: line feed)

Los símbolos ASCII 13 y 10 son caracteres de control y al imprimirlos en pantalla producen un efecto en lugar de mostrar algo visible. Vea por ejemplo:

```
Program uno;
Begin
WRITE(CHR(65));
WRITE(CHR(66));
End .
```

```
Program dos;
Begin
WRITE(CHR(65));
WRITE(CHR(13));
WRITE(CHR(10));
WRITE(CHR(66));
End .
```

```
Program tres;
Begin
WRITE(CHR(65));
WRITE(CHR(10));
WRITE(CHR(66));
End .
```

¿Cómo estará implementado writeln?

Una posible implementación en Pascal

```
Program veces; // Calcula cuantas veces aparece un char en una secuencia.
const terminador = chr(13);    // el terminador es el enter
var buscado,leido:char; cantidad: integer;
Begin
writeln(' ingrese símbolo a buscar'); readln(buscado);
writeln(' ingrese una secuencia de símbolos: ');
cantidad:=0;
read(leido);  //leo el primer carácter
WHILE leido <> terminador DO // controlo que no llegué al enter
  begin
    if leido = buscado then cantidad:=cantidad+1; // encontré otro
    read(leido); // leo el siguiente en la secuencia
  end;
writeln('Cantidad de veces que está', buscado, '= ',cantidad);
readln; // espera otro enter para no cerrar la consola
end.
```


Información adicional

Leonardo de Pisa, matemático italiano. (1170 -1250)

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

Información adicional

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.

Información adicional

- El **sistema de numeración decimal (base 10)** 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

Información adicional

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	ரு	சூ	எ	அ	கூ

Resolución de Problemas y Algoritmos Dr. Diego R. García 33