Resolución de Problemas y Algoritmos

Clase 7:
repetición condicional

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

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**

Alg.: subir una escalera:
mientras hay escalones
subir un escalón

Motivación

Formato de una clase:
1. **Saludar**
2. **Mientras existan dudas sobre temas anteriores: responder las preguntas**
3. **Explicar los temas nuevos hasta el fin de la clase**
4. **Despedirse**

Observe que aquí la repetición es un número fijo que pueda conocerse de antemano. Por ejemplo, el número de preguntas puede ser cero o más, y depende de el grupo de alumnos (una pregunta puede motivar otra en el momento).
El número de temas nuevos que se explican en clase es uno o más, también depende del tiempo disponible y no puede predecirse de antemano.

Motivación

```
ALGORITMO dar una clase
    Saludar
    Decir: "¿alguna pregunta sobre lo visto hasta ahora? "
    MIENTRAS hay preguntas
        -escuchar, pensar y contestar la pregunta
        -decir: " ¿otra pregunta? "
    FIN REPETIR
    REPETIR
        explicar un tema nuevo
    HASTA fin de la clase, o no quedan temas para explicar
    Despedirse
```


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 nuevamente 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.

```
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.

While: repetición condicional

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

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('fin del programa ');
end.
```


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 <condición>
<sentencia siguiente al repetir>
```

Si el resultado es false vuelve a

Si el resultado es true sigue en

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

- ¿Qué ocurre si la condición fuera cont=tope?
- 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

Conceptos: CICLOS INFINITOS ⊗

- **Una repetición condicional mal programada puede caer en una REPETICIÓN INFINITA.**

```
{…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.**

CICLOS INFINITOS ⊗

Una repetición que se realiza infinitas veces se denomina ciclo infinito

- **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 no puedan caer en ciclos infinitos.**

¿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 .
```

¿cómo estará implementado writeln?

```
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 .
```