Resolución de Problemas y Algoritmos

Clase 6
Repetición incondicional

John von Neumann

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

Repaso: Repetición incondicional

FOR V:= *Exp1* TO *Exp2* DO *sentencia*

- *Exp1* y *Exp2* son expresiones cuyo valor debe pertenecer al mismo tipo que V (variable de control).
- Al comenzar a V se le asigna el valor de *Exp1* .
- Luego, V es incrementada automáticamente de a uno en cada repetición (hasta el valor de *Exp2*).
- 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 (copie los ejemplos del pizarrón)
- La *sentencia* se repetirá (*valor final* – *valor inicial* + 1) veces, o 0 veces si *valor final* < *valor inicial*

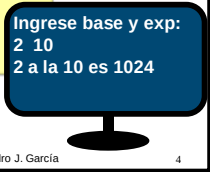
Problema propuesto

- **Escriba un programa para calcular un número natural elevado a una potencia (también natural).**
- **Ejemplos: 2^3=8 3^2=9 1^6=1 2^{10}=1024**
  2^3=2*2*2=8 3^2=3*3=9 1^6=1*1*1*1*1*1=1
- **Solución: multiplicar "base" "exponente" veces**

Algoritmo "potencia":
Leer base y exponente
Potencia ← 1
Repetir exponente veces
potencia ← potencia * base
Mostrar potencia

Un posible programa para "potencia"

```
PROGRAM CalculoPotencia;
{Calcula un número natural elevado a una potencia (también natural) }
VAR base, exponente, potencia, aux: integer;
BEGIN
writeln('Ingrese base y exp : ');
readln(base, exponente);
potencia := 1;
FOR aux := 1 TO exponente
   DO potencia:=potencia * base;
write(base, ' a la ', exponente);
writeln(' es ',  potencia);
END.
```



Problema propuesto

Factorial de un número N se denota con N!
y se define: N! = 1 * 2 * 3 * 4 * ..... * N
0! = 1

Ejemplos:
1! = 1
2! = 2 3! = 6 4! = 24 5! = 120
6! = 720
7! = 5.040
8! = 40.320
9! = 362.880
10! = 3.628.800

Escriba un programa para calcular el factorial de un número ingresado por el usuario.

Algoritmo propuesto

Problema: Escriba un programa para calcular el Factorial de un número N ingresado por el usuario.

Como N! = 1 * 2 * 3 * 4 * ..... * N
Por lo tanto tengo que hacer N multiplicaciones

Algoritmo factorial:
Leer N
factorial ← 1
factor ← 1
repetir N veces:
factorial ← factorial * factor
factor ← factor + 1
Mostrar factorial en pantalla

Programa para "factorial"

```
PROGRAM  CalcularFactorial;
 { calcula factorial de un número leido}
VAR  numero, factor, factorial: INTEGER;
BEGIN
writeln('ingrese número >= 0');
readln(numero);
factorial:=1;
FOR factor:=1 TO numero
   DO  factorial:=factorial * factor;
Writeln(' El factorial de ',numero, ' es ' , factorial);
END.
```


Problema para practicar

- **Escriba un programa que calcule el promedio de una cantidad conocida (e ingresada por el usuario) de números reales.**

```
Ingrese la cantidad de valores: 4
Ingrese un valor: 8.2
Ingrese un valor: 0.2
Ingrese un valor: -3.0
Ingrese un valor: 2.8
El promedio es: 2.05
```


Sentencia FOR-DOWNTO

La sentencia **FOR-DOWNTO** es análoga al **FOR-TO**, con la diferencia de que el valor de la variable de control se **decrementa de a uno** automáticamente en cada iteración.

```
FOR variable := expr1 DOWNTO  expr2
   DO  sentencia
```

```
FOR numero := 100 DOWNTO 90 DO write(numero);
```

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


Sentencia FOR-DOWNTO

```
FOR  V:= valor_inicial DOWNTO valor_final
DO   sentencia (simple o compuesta)
```

- Al comenzar a V se le asigna *valor_inicial*.
- Luego, V es derementada automáticamente de a uno en cada repetición (hasta llegar a *valor_final*).
- *valor_inicial* y *valor_final* son expresiones cuyo resultado debe pertenecer al mismo tipo que V.
- La *sentencia*, que puede ser compuesta, se repetirá (*valor_inicial* – *valor_final* + 1) veces.
- Si *valor_final* es mayor estricto a *valor_inicial* entonces se repetirá 0 veces.

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('----------');
FOR  letra:='G' DOWNTO 'A'
     DO   write(letra);
writeln('----------');
END.
```

```
Cuenta regresiva
9 8 7 6 5 4 3 2 1 0
----------
G F E D C B A
----------
```


Muy importante


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

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 la próxima clase).

```
…
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 muestran estos ejemplos a continuación, si es posible cambiar V antes o después del ciclo FOR, y también usarse para otro FOR que no esté anidado.

```
V:= 9; writeln(V);
FOR  V:= 1 TO 3
DO   writeln(V);
V:= 8; writeln(V);
```

```
FOR  V:= 1 TO 3
DO   writeln(V);
FOR  V:= 3 DOWNTO 1
DO   writeln(V);
V:= 4; writeln(V);
```


Problemas para practicar

- **Escriba un programa que dados tres valores enteros V1, V2 y N ingresados por el usuario, muestre y cuente cuantos enteros hay entre V1 y V2 que sean múltiplos de N.**

```
Ingrese dos valores enteros: 7 30
Ingrese un divisor: 6
Entre 7 y 30, son múltiplos de 6:
12, 18, 24, 30
En total son 4 múltiplos.
```


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:= 1 TO 5
    DO writeln(v, h, t);
```

Obs: usan diferentes variables de control ¿Por qué?

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


```
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
...
```

- **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 letras;
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 anidados;
VAR   cant, num, pri, ult : integer;
BEGIN
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
```


Computadora con arquitectura von Neumann

Una computadora es un sistema digital con tecnología microelectrónica compuesta por:

1- CPU (Unidad Central de Proceso)

2- Memoria

3- Dispositivos de Entrada y Salida

Todo interconectado (por "buses")

CPU ⟷ bus ⟷ **Memoria:** programa y datos

Esta arquitectura, y el concepto de programa almacenado en memoria, (ideados en 1945) se le atribuyen al matemático húngaro: John von Neumann

Representación de información: bit y byte

- Un **bit** es la unidad mínima para representar información en un sistema informático. Permite almacenar dos valores diferentes (generalmente representados con 0 y 1).
- **Byte** se utiliza como unidad en la medida de capacidad de almacenamiento de un dispositivo.
- Un byte equivale a 8 bits y permite $2^8 = 256$ valores diferentes.
- Históricamente se usó como el número de bits necesario para codificar caracteres en una computadora y por esa razón se convirtió en la porción más pequeña direccionable y por ende en la unidad de memoria en la mayoría de las computadoras.
- Byte proviene de bite (en inglés "mordisco"), como la cantidad más pequeña de datos que un procesador podía "morder" a la vez.

Almacenamiento en memoria

Las variables de los siguientes tipos usan un espacio en memoria que es fijo que no cambia en ejecución (pero puede variar de un compilador a otro).

En Free Pascal (Lazarus) generalmente ocupan:

Tipo:	Tamaño:
Char	1 byte
Boolean	1 byte
Integer	4 bytes
Real	6 bytes

1 byte = 8 bits

Computadora con arquitectura von Neumann

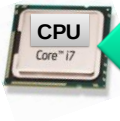
Una computadora es un sistema digital con tecnología microelectrónica compuesta por:

1- CPU (Unidad Central de Proceso)

2- Memoria

3- Dispositivos de Entrada y Salida

Todo interconectado (por "buses")



bus ⟷ **Memoria principal (RAM)** *contiene: programas en ejecución y datos* ⟷ bus

Memoria secundaria (ej: disco rígido, pen-drive, dvd, unidad de estado sólido, "la nube")

Contiene: Archivos de programas, textos, datos, fotos, música, etc.

Memoria RAM

RAM concreta y real: módulo 4Gb DDR3

abstracción de bajo nivel (secuencia de bytes)

0 0 1 0 1 0 1 1
1 1 1 0 1 0 0 1
0 1 1 1 0 1 0 1
1 1 1 0 1 0 0 1
1 1 0 1 1 1 1 0
1 0 1 1 1 1 1 1
1 1 1 1 1 0 1 0
1 1 1 0 1 0 0 0

abstracción de alto nivel (una traza de RPA)

NUM:	123
ES_PAR:	FALSE
LETRA:	"A"
PRECIO:	12.02

Memoria secundaria (disco duro)

Sistema de grabación magnética con uno o más platos o discos rígidos, unidos por un mismo eje que gira a gran velocidad

Memoria secundaria (solid state drive)

Memoria secundaria (SSD - solid state drive)

Una unidad de estado sólido o SSD (acrónimo de solid-state drive) es un dispositivo de almacenamiento de datos que usa una memoria no volátil, como la memoria flash, o una memoria volátil como SDRAM, para almacenar datos, en lugar de los platos giratorios magnéticos encontrados en los discos duros convencionales.

En comparación con los discos duros tradicionales, las SSD son menos sensibles a los golpes, son prácticamente inaudibles, y son más rápidas.

A veces se traduce erróneamente en español la "D" de SSD como "disk" cuando, en realidad, representa la palabra "drive", que podría traducirse como unidad o dispositivo.

A partir de 2010, la mayoría de los SSDs utilizan memoria flash basada en compuertas NAND, que retiene los datos sin alimentación.

CONCEPTOS: Valores de variables en Pascal

- **Las variables de los tipos de datos SIMPLES vistos hasta el momento en esta materia tiene las siguientes características:**
  - **(1) Residen en memoria principal (RAM,** *random-access memory o memoria de acceso aleatorio***).**
  - **(2) Los valores que contienen no perduran cuando termina la ejecución del programa. (El espacio de memoria utilizado por esas variables es liberado y usado por otros programas)**
  - **(3) La cantidad de memoria que usan es fija (no cambia en ejecución) y el compilador usa este dato para reservar lugar en memoria.**

Problemas 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. Escribir programa
7. Verificar con una traza.