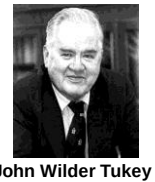


John Wilder Tukey

Resolución de Problemas y Algoritmos

Clase 5: repetición incondicional

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

Especificación de un algoritmo

- **Escriba un algoritmo para obtener exactamente 1 litro de agua en un bidón de 100 litros. Se dispone de una botella de medio litro (bot) y de las primitivas vaciar, llenar (con canilla) y trasvasar.**

Algoritmo 1a:
Vaciar(bidón)
Vaciar(bot)
Llenar (bot)
Trasvasar(bot,bidón)
Llenar (bot)
Trasvasar(bot,bidón)

Algoritmo 1b:
Vaciar(bidón)
Vaciar(bot)
REPETIR 2 VECES:
•Llenar (bot)
•trasvasar(bot,bidón)

Realizar la traza

Especificación de repetición en un algoritmo

- **Escriba un algoritmo para obtener exactamente 10 litros de agua en un bidón de 100 litros. Se dispone de una botella de medio litro y de las primitivas vaciar, llenar y trasvasar.**

Algoritmo 2:
Vaciar(bidón)
Vaciar(bot)
Repetir 20 veces:
• Llenar (bot)
• Trasvasar (bot, bidón)

Realizar la traza

Especificación de repetición en un algoritmo

- **Escriba otro algoritmo para obtener exactamente 80 litros de agua en un bidón de 100 litros. Se dispone de una botella de medio litro y de las primitivas vaciar, llenar y trasvasar.**

Algoritmo 3:
Vaciar(bidón)
Vaciar(bot)
Repetir 160 veces:
• Llenar (bot)
• Trasvasar (bot, bidón)

Realizar la traza

Especificación de repetición en un algoritmo

- **Escriba otro algoritmo para obtener exactamente N litros (0<N<100) litros de agua en un bidón de 100 litros. Se dispone de una botella de medio litro y de las primitivas vaciar, llenar y trasvasar.**

Algoritmo 4:
Vaciar(bidón)
Vaciar(bot)
Leer (N)
Repetir (2*N) veces:
• Llenar (bot)
• Trasvasar (bot, bidón)

Realizar la traza ¿cómo sería un algoritmo para 100 litros?

Problema propuesto (p)

- **Escriba un algoritmo y un programa para calcular "num" elevado a la "pot" (num y pot naturales).**
- **Ejemplos: $2^3=2*2*2=8$ $3^2=3*3=9$ $1^6=1*1*1*1*1*1=1$**
- **Solución: multiplicar "num" "pot" veces**

Algoritmo:
Leer num y pot
Potencia ← 1
Repetir pot veces
potencia ← potencia * num
Mostrar potencia

El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
"Resolución de Problemas y Algoritmos. Notas de Clase". **Alejandro J. García. Universidad Nacional del Sur. (c)1998-2013.**

Repetición incondicional en Pascal (p)

Las sentencias de un ciclo FOR-TO se ejecutan CERO o más veces dependiendo de *valor_inicial* y *valor_final*.

```
FOR V:= valor_inicial TO valor_final
DO  1 sentencia simple
    o compuesta ;
Otra sentencia siguiente;
```

La sentencia se ejecuta un número fijo de veces y luego continua en:

- La variable V se suele llamar variable de control.
- Al comenzar a V se le asigna *valor_inicial*.
- Luego, V es incrementada automáticamente de a uno en cada repetición (hasta llegar a *valor_final*).

Problema Propuesto (p)

Escriba un programa para mostrar por pantalla los números entre dos topes ingresados.

```
PROGRAM ListaNumeros;
{muestra los números desde tope inferior a tope superior}
VAR topeinf,topesup,num:INTEGER;
BEGIN
writeln('Ingrese topes: ');
readln(topeinf, topesup);
writeln('Números');
FOR num:= topeinf TO topesup
   DO  write(num,' , ');
END.
```

```
Ingrese topes:
3 8
Números
3 , 4 , 5 , 6 , 7 , 8 ,
```


Repetición incondicional en Pascal

```
FOR V:= valor_inicial TO valor_final
DO  sentencia
```

- *valor_inicial* y *valor_final* son expresiones cuyo valor debe pertenecer al mismo tipo que V.
- La *sentencia* (que puede ser compuesta), se repitirá (*valor_final* – *valor_inicial* + 1) veces.

```
FOR V:= 1 TO 5 DO writeln(V); { 5 veces}
```

- Si *valor_final* es menor estricto a *valor_inicial* entonces se repetirá 0 veces.

```
FOR V:= 5 TO 1 DO write(V); {0 veces}
```


Repetición incondicional

- Al comenzar a V se le asigna el valor inicial y luego, V se incrementada automáticamente de a uno hasta llegar al valor final.

```
FOR V:= 1 TO 100
DO  <sentencia>
```
Aquí <sentencia> se repite 100 veces

```
FOR V:= 100 TO 199
DO  <sentencia>
```
Aquí <sentencia> se repite 100 veces

```
FOR V:= -10 TO -1
DO  <sentencia>
```
Aquí <sentencia> se repite 10 veces

```
FOR V:= 1 TO -2
DO  <sentencia>
```
Aquí <sentencia> se repite 0 veces

Escriba un programa para mostrar por pantalla los números entre dos topes ingresados en cualquier orden.

```
PROGRAM ListaNumeros;
{muestra los números desde tope inferior a tope superior}
VAR topeinf,topesup,num,aux:INTEGER;
BEGIN
writeln('Ingrese topes: '); readln(topeinf, topesup);
IF topesup < topeinf
Then begin {intercambio los valores de los topes}
       aux:=topeinf;
       topeinf:=topesup;
       topesup:=aux;
       end;
writeln('Números');
FOR num:= topeinf TO topesup
   DO  writeln(num,' , ');
END.
```

```
Ingrese topes:
8 3
Números
3 , 4 , 5 , 6 , 7 , 8 ,
```


Sentencia FOR-TO

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

- Tanto *exp1* como *exp2* pueden ser valores, variables, o expresiones siempre que sean del mismo tipo ordinal que V **(no puede ser tipo real).**
- Al comenzar a V se le asigna el valor de evaluar *exp1* y luego, V se incrementada automáticamente de a uno hasta llegar al valor de *exp2*.

```
N:=2;
FOR V := 1+1 TO  N DO writeln(N);
FOR V := N TO  N+N DO  writeln(N);
FOR V := N div 2 TO (N+10) - N + 2 DO  writeln(N);
```


El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
"*Resolución de Problemas y Algoritmos. Notas de Clase*". Alejandro J. García. Universidad Nacional del Sur. (c)1998-2013.

Escriba un programa para mostrar por pantalla los números entre dos topes ingresados, pero en este caso mejorando la salida por pantalla.

```
PROGRAM ListaNumeros;
{muestra los números desde tope inferior a tope superior
 a diferencia de las versiones anteriores no muestra la coma al
final de la lista, imprimiendo el último fuera del FOR}
VAR topeinf,topesup,num:INTEGER;
BEGIN
writeln('Ingrese topes: ');
readln(topeinf, topesup);
writeln('Números');
FOR num:= topeinf TO topesup-1
    DO   writeln(num,' , ');
writeln(' y ', topesup '. ');
END.
```

```
Ingrese topes:
3  8
Números
3 , 4 , 5 , 6 , 7, y 8 .
```


FOR-TO: ejemplo con CHAR

```
PROGRAM letras_y_codigos;
{muestra los códigos ASCII de algunas letras}
CONST  ult='J'; columnas=3;
VAR  letra: char;
BEGIN
FOR  letra:= 'A' TO ult DO
  BEGIN
  write(letra,'=', ord(letra),'   ');
  {baja de renglón cada "columnas" veces}
  if ((ord(letra)-ord('A')+1)  mod columnas) = 0
   then writeln;
  END
END.
```

```
A=65  B=66  C=67
D=68  E=69  F=70
G=71 H=72  I=73
J=74
```


Problema propuesto (p)

- **Escriba un algoritmo y un programa para calcular "num" elevado a la "pot" (num y pot naturales).**
- **Ejemplos: $2^3=8$ $3^2=9$ $1^8=1$**
- **Solución: multiplicar "num" "pot" veces**

Algoritmo:
- Leer num y pot
- Potencia ← 1
- Repetir pot veces
  - potencia ← potencia * num
- Mostrar potencia

Tarea: Escriba el programa en Pascal.

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.

Problema propuesto

Problema: Escriba un programa para calcular el Factorial de un número N ingresado por el usuario.
Solución: si N=0 0! = 1; si N>0 N! = 1 * 2 * 3 * 4 * ..... * N
Algoritmo:
- Leer N
- factorial ← 1
- factor ← 1
- repetir N veces:
  - factorial ← factorial x factor
  - factor ← factor + 1

Mostrar factorial en pantalla


```
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);
Readln; {espera que presione enter ….}
END.
```


El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
"Resolución de Problemas y Algoritmos. Notas de Clase". **Alejandro J. García. Universidad Nacional del Sur. (c)1998-2013.**

Conceptos: Hardware [Wikipedia]

Hardware corresponde a todas las partes tangibles de un sistema informático; sus componentes son: eléctricos, electrónicos, electromecánicos y mecánicos. Son cables, gabinetes, periféricos de todo tipo y cualquier otro elemento físico involucrado;

contrariamente, el soporte lógico que es intangible y es llamado ***software***.

Hardware es una palabra inglesa (literalmente: partes duras); como no posee una traducción adecuada, fue admitida por la Real Academia Española que lo define como: *«Conjunto de los componentes que integran la parte material de una computadora».*

Conceptos: Software [Wikipedia]

Se conoce como ***software*** *al equipamiento lógico o soporte lógico* de un sistema informático; comprende el conjunto de los componentes **lógicos** necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos, que son llamados ***hardware***.

Software es una palabra inglesa (literalmente: partes blandas o suaves); como en español no posee una traducción adecuada, fue admitida por la Real Academia Española que lo define como: *«Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora ».*

Software [Wikipedia]

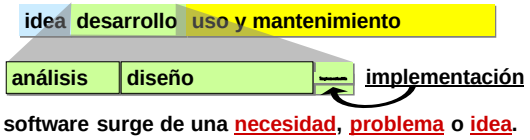
Existen varias definiciones similares aceptadas para software, pero probablemente la más formal sea la siguiente: « Es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación» [Extraído del estándar 729 del IEEE]

Considerando esta definición, el concepto de software va más allá de los programas de computación; también su documentación, los datos a procesar e incluso la información de usuario forman parte del software: es decir, *abarca todo lo intangible*, todo lo «no físico» relacionado.

Historia [Wikipedia]

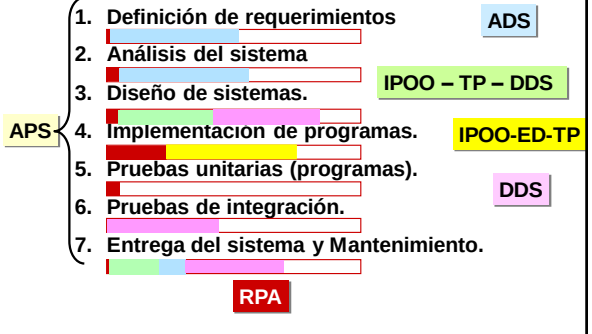
- John W. Tukey usó el termino "Software de Computación" (Computer Software) en un artículo de 1958 en el *American Mathematical Monthly*, aparentemente el primer uso del término.
- Mientras trabajaba con John von Neumann en los primeros diseños de computadoras, Tukey introdujo la palabra "bit" como contracción de "Dígito binario" (por sus siglas en inglés *Binary Digit*].
- *El concepto* de leer diferentes secuencias de instrucciones (programa) desde la memoria de un dispositivo para controlar los cálculos fue introducido por Charles Babbage alrededor de 1833.

Ciclo de vida del software



- **El software surge de una necesidad, problema o idea.**
- **Luego se desarrolla e implementa.**
- **Finalmente se usa durante un tiempo y de ser necesario se modifica. (según la experiencia el 60% del ciclo de vida lo constituye esta etapa)**
- **Es importante utilizar metodologías y técnicas que simplifiquen el desarrollo y faciliten el mantenimiento (algunas de ellas vamos a comenzar a ver en RPA)**

El desarrollo de software y algunas materias



El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
"*Resolución de Problemas y Algoritmos. Notas de Clase*". Alejandro J. García. Universidad Nacional del Sur. (c)1998-2013.

Conceptos: Pautas para un buen estilo de programación

- Es sumamente importante que un algoritmo o programa sea **claro**, y puede ser **entendido rápidamente** por una **persona** que lo lee.
- Para lograr esto hay que tener en cuenta una serie de **pautas** de "**buen estilo**" para la confección de programas:

1. **Uso de nombres representativos.**
2. **Indentación (en inglés indent).**
3. **Comentarios en el código.**

   {comentarios en Pascal entre llaves}

1. Uso de nombres representativos

☹

```
IF (x1 > x2)
THEN
IF (x1 > w)
THEN z8:= x1
ELSE  z8 := w
ELSE IF (x2 > w)
THEN z8 := x2
ELSE  z8 := w
```

☺

```
IF (num1 > num2)
THEN
IF (num1 > num3)
THEN máximo := num1
ELSE máximo := num3
ELSE IF (num2 > num3)
THEN máximo := num2
ELSE máximo := num3
```


2. Indentación

😐

```
IF (num1 > num2)
THEN
IF (num1 > num3)
THEN máximo := num1
ELSE máximo := num3
ELSE IF (num2 > num3)
THEN máximo := num2
ELSE máximo := num3
```

☺

```
IF (num1 > num2)
THEN
   IF (num1 > num3)
      THEN máximo := num1
      ELSE máximo := num3
ELSE
   IF (num2 > num3)
      THEN máximo := num2
      ELSE máximo := num3
```


3. Comentarios en el código

😐

```
IF (num1 > num2)
THEN
   IF (num1 > num3)
      THEN máximo := num1
      ELSE máximo := num3
ELSE
   IF (num2 > num3)
      THEN máximo := num2
      ELSE máximo := num3
```

☺

```
{ calcula el máximo entre 3
  números: num1, num2 y num3}
IF (num1 > num2)
THEN
   IF (num1 > num3)
      THEN máximo := num1
      ELSE máximo := num3
ELSE {… caso num1 <= num2….}
   IF (num2 > num3)
      THEN máximo := num2
      ELSE máximo := num3
{… la variable máximo ahora tiene
  el mayor valor de los 3….}
```


Compare

☹

```
IF (x1 > x2)
THEN
IF (x1 > w)
THEN z8:= x1
ELSE  z8 := w
ELSE IF (x2 > w)
THEN z8 := x2
ELSE  z8 := w
```

☺

```
{ calcula el máximo entre 3
  números: num1, num2 y num3}
IF (num1 > num2)
THEN
   IF (num1 > num3)
      THEN máximo := num1
      ELSE máximo := num3
ELSE {… caso num1 <= num2….}
   IF (num2 > num3)
      THEN máximo := num2
      ELSE máximo := num3
{… la variable máximo ahora tiene
  el mayor valor de los 3….}
```


Cuando escribe en papel también puede usar líneas para agrupar bloques

😐

```
IF (num1 > num2)
THEN
   IF (num1 > num3)
      THEN máximo := num1
      ELSE máximo := num3
ELSE
   IF (num2 > num3)
      THEN máximo := num2
      ELSE máximo := num3
```

☺

```
IF (num1 > num2)
THEN
   IF (num1 > num3)
      THEN máximo := num1
      ELSE máximo := num3
ELSE
   IF (num2 > num3)
      THEN máximo := num2
      ELSE máximo := num3
```


El uso total o parcial de este material está permitido siempre que se haga mención explícita de su fuente:
***"Resolución de Problemas y Algoritmos. Notas de Clase".* Alejandro J. García. Universidad Nacional del Sur. (c)1998-2013.**