# Resolución de Problemas y Algoritmos

**Clase 7:**
**Resolución de Problemas**

Alan M.Turing

**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

## Algoritmo

```
ALGORITMO de Ale para organizar la clase
    Saludar
    Decir: "¿alguna pregunta? "
    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
```



## Problema propuesto

**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**



## Una posible implementación en Pascal

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



## Conceptos: Uso de analogías (el valor de la experiencia)

- **Resolver un problema por analogía consiste en hacer uso de la solución encontrada para un problema ya resuelto antes (o parte de ella) para resolver un nuevo problema.**
- **De esta manera, se puede poner en práctica toda la experiencia obtenida en la solución de problemas previos.**
- **Resolver un problema por analogía consiste en encontrar una correspondencia entre los elementos involucrados en dos problemas: el problema cuya solución ya conocemos y el problema nuevo cuya solución buscamos.**
- **Usando esa correspondencia, hay buscar de adaptar la solución del problema ya resuelto para encontrar la solución del nuevo.**



## 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**

## Una posible implementación en Pascal

```
Program veces;              // Calcula la cantidad de veces que
const terminador = '.' ;    // aparece un símbolo en una secuencia
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 primero
WHILE leido <> terminador DO
  begin
    if leido = buscado then cantidad:=cantidad+1;
    read(leido); // leo el siguiente en la secuencia
  end;
readln; // lee el enter del final de la secuencia
writeln('Cantidad de veces que está', buscado, '= ',cantidad);
readln; // espera otro enter para no cerrar la consola
end.
```



## 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 la cantidad de veces que
const terminador = chr(13); //aparece un símbolo en una secuencia
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 primero
WHILE leido <> terminador DO
  begin
    if leido = buscado then cantidad:=cantidad+1;
    read(leido); // leo el siguiente en la secuencia
  end;
readln; // lee el enter del final de la secuencia
writeln('Cantidad de veces que está', buscado, '= ',cantidad);
readln; // espera otro enter para no cerrar la consola
end.
```



## Utilice ahora la experiencia…

- **La experiencia de resolver los problemas anteriores puede usarse en estos problemas que siguen a continuación.**
- **Para ello debe encontrar la correspondencia entre los elementos de algún problema ya resuelto y el que quiere resolver**

**IMPORTANTE:**

- **Tenga en cuenta que el nuevo problema puede tener características que requieran de una adaptación de la solución: como usar un repetir-hasta en lugar de un repetir-mientras si fuera más adecuado.**



## Problemas propuestos (resueltos)

**Los siguientes problemas están en mayor o menor medida resueltos, pero para practicar primero intente resolverlo sin mirar la solución.**

**Sin hacer trampa... ;)**

## Problema para propuesto

**Indicar si es que aparece un símbolo ASCII (que llamaremos "buscado") en una oración terminada en un punto.**

**Ejemplo: (buscado='a') oración: hola, estoy en clase de RPA.**

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

**Solución: ¿Es necesario llegar siempre al final de la oración?**

**Recorrer la oración hasta encontrar el símbolo o llegar al final.**

**Algoritmo:**

**leer(buscado) y encontrado ← falso**
**repetir**
**leer un símbolo de la secuencia (leído)**
**si leido = buscado entonces encontrado ← verdadero**
**hasta encontrado o símbolo = '. '**
**mostrar valor de cantidad**



## Problema propuesto: (dígito presente)

**Problema: Escriba un programa en Pascal para determinar si un dígito D aparece en un número entero N.**

Ej: si D = 3 y N=1234, entonces D aparece.
Ej: si D = 6 y N=661, entonces D aparece.
Ej: si D = 3 y N=661, entonces D NO aparece.

**Solución: *voy mirando* dígito a dígito y si alguno es igual a D es que aparece; si recorro todo el número y ninguno es igual a D entonces no aparece.**

**(pero ¿cómo puedo "mirar" un dígito particular?)**

**→ Para obtener el último dígito de N : N mod 10**

**→ Para eliminar el último dígito de N : N div 10**



## Dígito presente en un número

**Solución: *voy mirando* dígito a dígito y si alguno es igual a D es que aparece; si recorro todo el número y ninguno es igual a D entonces no aparece.**

**Observación: un número siempre tiene al menos un dígito (se repetirá 1 o más veces)**

**Algoritmo:**

**Si el último dígito de N es igual a D (N mod 10 = D)**
**entonces aparece**
**Eliminar el último dígito de N (N ← N div 10)**
**Repetir los dos pasos anteriores hasta que:**
**"N no tenga mas dígitos" o "descubrí que aparece"**



```
PROGRAM digito_presente;
            {indica si está un dígito en un número}
VAR numero, digito, cant_veces : INTEGER; esta: BOOLEAN;
BEGIN
writeln('ingrese Num y Dig'); readln(numero, digito);
esta:=false; {asumo que no está}
REPEAT
    IF (numero mod 10)  = digito {comparo con el último dígito}
        THEN  esta:=true;
    numero := numero div 10;  {elimino el último dígito}
UNTIL (numero = 0) OR esta;   {hasta recorrer todo o esta}
IF esta {también puede ser "esta=true" pero es redundante}
then writeln(digito, ' aparece en ', numero)
else writeln(digito, ' no aparece en ', numero);
END.
```



## Problema propuesto

**Problema: Escribir un programa para ver si un número N es primo (*Definición: un número es primo si es un entero positivo mayor que 1, que es divisible solamente por si mismo y la unidad*)**

**Ejemplos: 2 , 3, 7, 11 y 2003: son primos**
**4 no es primo es divisible por 2**
**2001 no es primo, es divisible por 3**
**121 no es primo, es divisible por 11**

**Una solución: si N tiene un divisor distinto de N o 1 entonces NO es primo, de lo contrario es primo.**

**¿Algoritmo?**



## Algoritmo para "esPrimo"

```
ALGORITMO EsPrimo (algoritmo 1)
COMIENZO
 leer(N)
 CantDivisores ← 0 { asumo  que cantidad de divisores de N es 0 }
 Divisor ← 2  { el primer divisor a considerar es 2  ... }
 REPETIR MIENTRAS (Divisor <  N) y (CantDivisores = 0)
     SI N // Divisor = 0 {Si Divisor divide a N, incremento CantDivisores}
         ENTONCES CantDivisores ← CantDivisores + 1
     Divisor ← Divisor + 1 {paso al próximo Divisor .. }
 fin repetir mientras
 SI (CantDivisores=0) y (N > 1)
   ENTONCES   'N es primo'
   SINO       'N no es primo'
 FIN ALGORITMO
```

```
PROGRAM EsPrimo; {a partir del algoritmo 1}
VAR N,cantDivisores,Divisor:integer;
BEGIN
  writeln('Ingrese un número'); read(N);
  CantDivisores := 0; { asumo  que cantidad de divisores de N es 0 }
  Divisor := 2;  { el primer divisor a considerar es 2  ... }
  WHILE (Divisor <  N) and (CantDivisores = 0)  DO
  BEGIN
    IF N mod Divisor = 0 {Si Divisor divide a N, incremento CantDivisores}
       THEN CantDivisores := CantDivisores + 1 ;
    Divisor := Divisor + 1; {paso al próximo Divisor .. }
  END {while}
   IF (CantDivisores=0) and (N > 1)
    THEN   writeln('Es nro. Primo')
     ELSE  writeln('NO es nro. Primo')
END.
```

**Observación: en realidad no necesito "contar" los divisores**



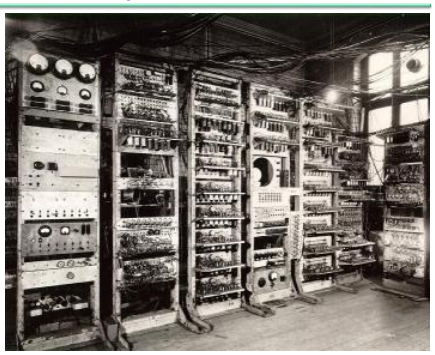
```
ALGORITMO EsPrimo (algoritmo 2, otra solución)
COMIENZO
  leer(N)
  SI (N > 1)
  ENTONCES  EsPrimo ← Verdadero { asumo  que es primo}
  SINO EsNroPrimo ← Falso
  Divisor ← 2  { el primer divisor a considerar es 2  ... }
  REPETIR MIENTRAS (Divisor <  N) y (esPrimo = verdadero)
       SI N // Divisor = 0 {Si Divisor divide a N, NO ES PRIMO}
            ENTONCES esPrimo ← falso
       Divisor ← Divisor + 1 {paso al próximo Divisor .. }
  fin repetir mientras
FIN ALGORITMO
```

**Tarea 1: Implemente en Pascal este algoritmo usando WHILE.**
**Tarea 2: Luego implemente usando REPEAT-UNTIL**



## Primeras computadoras: Mark 1

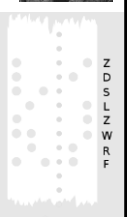
**Mark 1 (1949).**
**Manchester UK.**
**Contenía 4050 válvulas y consumía 25.000 watts.**





## Manchester Mark 1 y Alan Turing

- **Su desarrollo comenzó en 1948 y estuvo completamente operable para octubre de 1949. Contenía 4050 válvulas y consumía 25.000 watts.**
- **Alan Turing fue nombrado director of Computing Machine Laboratory at the University of Manchester en1948. Desarrolló un esquema de codificación que permitía que programas y datos sean escritos y leidos de una cinta de papel.**
- **Mark 1 no tenía lenguaje ensamblador, los programas debían ser escritos en binario.**
- **Mark 1 no tenía sistema operativo.**






## Alan M. Turing, (23/6/1912 - 7/6/1954)

**Matemático, lógico, científico de la computación, criptógrafo y filósofo británico.**

**Proporcionó una influyente formalización de los conceptos de algoritmo y computación, hoy denominada:** *la máquina de Turing*.

**También contribuyó de forma particular e incluso provocativa a la Inteligencia Artificial con su artículo Computing Machinery and Intelligence, el cual comienza así:** I propose to consider the question, "Can machines think?" This should begin with definitions of the meaning of the terms "machine" and "think"….

**http://es.wikipedia.org/wiki/Alan_Turing**




## ¡A disfrutar lo más posible!