Programación I

"Lista de primos" en varios lenguajes de programación

Pseudocódigo

```
programa ListaPrimos

  función EsDivisible(a,b: enteros): booleano;
  inicio
    EsDivisible ← a mod b = 0
  fin

  función EsPrimo(n: entero) : booleano;
  variable divisor : entera
  inicio
    si n = 1 entonces
      EsPrimo ← falso
    si_no
      divisor ← 2;
      mientras (divisor < n) y no
                EsDivisible(n,divisor) hacer
        divisor ← divisor + 1;
      fin_mientras
      EsPrimo ← divisor ≥ n
    fin_si
  fin
```

```
variables i,n : enteras;
inicio
  leer n
  i ← 1
  mientras i ≤ n hacer
    si EsPrimo(i) entonces
      escribe i
    fin_si
    i ← i + 1
  fin_mientras
fin
```

Pascal

```
program ListaPrimos(input,output);

  function EsDivisible(a,b: integer): boolean;
  begin
    EsDivisible := a mod b = 0
  end;

  function EsPrimo(n: integer) : boolean;
  var divisor : integer;
  begin
    if n = 1 then EsPrimo := false else
    begin
      divisor := 2;
      while (divisor < n) and not
             EsDivisible(n,divisor) do
        divisor := divisor + 1;
      EsPrimo := divisor >= n
    end
  end;
```

```
var i,n : integer;
begin
  readln(input,n);
  for i := 1 to n do
    if EsPrimo(i) then
      write(output,i,' ');
end.
```

Eiffel

```
class ListaPrimos

  creation make;

  esPrimo(n: INTEGER) : BOOLEAN is
  require n > 0;
  local divisor : INTEGER;
  do
    if n == 1 then
      Result := False
    else
      from  divisor := 2
      until divisor >= n or n\\divisor = 0
      loop
        divisor := divisor+1
      end
      Result := divisor >= n
    end -- if
  end -- esPrimo

  make is
  local n,i: INTEGER
  do
    n := io.read_integer
    from  i := 1
    until i > n
    loop
      if esPrimo(i) then
        io.put_integer(i)
      end -- if
    end
  end -- make

end -- ListaPrimos
```

C

```c
#include <stdio.h>

#define NOESDIVISIBLE(a,b) ((a) % (b))

int EsPrimo(int n)
{
  int divisor;
  if(n == 1) return 0;
  divisor = 1;
  while(divisor++ < n) {
    if(! NOESDIVISIBLE(n,divisor)) return 0;
  }
  return 1;
}

int main(int argc, char* argv[])
{
   int i,n;
   scanf("%d",&n);
   for(i = 1; i <= n; i++) {
     if(EsPrimo(i)) printf("%d ",i);
   }
   return 0;
}
```

Java

```
import java.io.DataInputStream;
import java.io.IOException;

public class ListaPrimos {

    public static boolean esPrimo(int n) {
      if(n == 1) return false;
      int d;
      for(d = 2; d < n; d++) {
        if(n % d == 0) break;
       }
      return d >= n;
    }

    public static void main(String[] args) throws IOException {
      DataInputStream tec = new DataInputStream(System.in);
      int n = tec.readInt();
      for(int i = 1; i <= n; i++) {
        if(esPrimo(i)) System.out.print(i+" ");
      }
    }
}
```

Javascript en página HTML

```
function esPrimo(n) {
  if(i == 1) return false;
  for(d = 2; d < n; d++) {
    if(n % d == 0) return false;
  }
  return true;
}

function listaPrimos() {
  n = document.formu.entrada.value;
  txt = "<P ID='res'>";
  for(i = 1; i <= n; i++) {
    if(esPrimo(i)) txt = txt+i+" ";
  }
  res.outerHTML = txt+"</P>";
}
```

```
<HTML>
<HEAD>
  <SCRIPT>
  </SCRIPT>
</HEAD>
<BODY>
  <H1>Lista de primos</H1>
  <FORM NAME="formu">
    N = <INPUT TYPE="text" NAME="entrada">
    <INPUT TYPE="button" VALUE="Calcular!"
     ONCLICK="listaPrimos()">
  </FORM>
  <P ID="res">El resultado se escribe aqui</P>
</BODY>
</HTML>
```

Haskell

```
esPrimo n = testPrimo n 2 where
            testPrimo n d
              | d >= n            = True
              | n `mod` d == 0    = False
              | otherwise         = testPrimo n (d+1)

primos n = [x | x <- [2..n], esPrimo x]
```

Otra versión (criba de Eratóstenes):

```
criba (p:r) = p : criba [ n | n <- r, n `mod` p != 0]

primos n = take n (criba [2..])
```

Prolog

```
enRango(X,X,Max).
enRango(X,Min,Max) :-
  MinP is Min + 1,
  MinP =< Max,
  enRango(X,MinP,Max).

esDivisible(0,X).
esDivisible(X,Y) :-
  Xa is X-1,
  enRango(Y,2,Xa),
  Z is X mod Y,
  Z = 0.

esPrimoMenorQue(X,Lim) :-
  enRango(X,0,Lim),
  not(esDivisible(X,U)).
```

Significado:

enRango(X,Min,Max) = Min ≤ X ≤ Max

- **X ≤ X ≤ Max es cierto**
- **Min ≤ Max y Min+1 ≤ X ≤ Max ⇒ Min ≤ X ≤ Max es cierto**

esDivisible(X,Y) = Y divide a X

- **cualquier número divide a 0**
- **2 ≤ Y ≤ X-1 y resto(X/Y) = 0 ⇒ Y divide a X**

- **0 ≤ X ≤ Lim y ningun número divide a X ⇒ X es primo y menor o igual a Lim**

Prolog

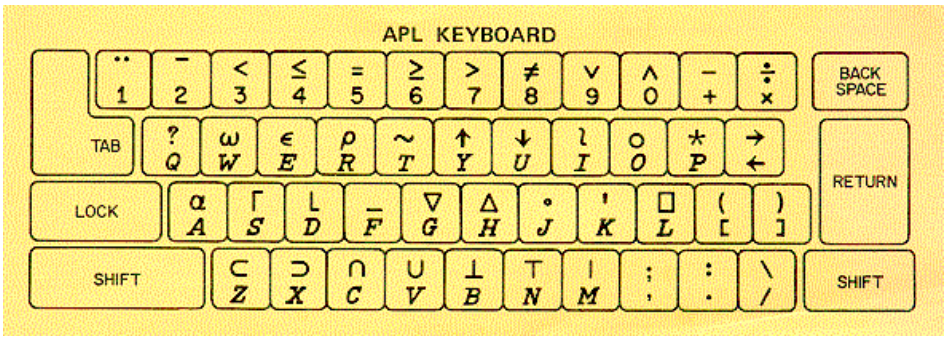
```
enRango(X,X,Max).
enRango(X,Min,Max) :-
  MinP is Min + 1,
  MinP =< Max,
  enRango(X,MinP,Max).

esDivisible(0,X).
esDivisible(X,Y) :-
  Xa is X-1,
  enRango(Y,2,Xa),
  Z is X mod Y,
  Z = 0.

esPrimoMenorQue(X,Lim) :-
  enRango(X,0,Lim),
  not(esDivisible(X,U)).
```

Ejemplo de uso:

```
?- esPrimoMenorQue(5,10).
Yes
?- esPrimoMenorQue(6,10).
No
?- esPrimoMenorQue(X,10).
X = 2;
X = 3;
X = 5;
X = 7;
?- esDivisible(12,X).
X = 2;
X = 3;
X = 4;
X = 6;
```

APL

$$(\sim N \in N \circ .\times N)/N \leftarrow 1 \downarrow \iota N$$



APL

$$(\sim N \in N \circ . \times N) / N \leftarrow 1 \downarrow \iota N$$

$\iota 9 =$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

$1 \downarrow$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | $=$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

$N \leftarrow$ | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

APL

$$(\sim N \in N \circ . \times N) / N \leftarrow 1 \downarrow \iota N$$

$N =$

2	3	4	5	6	7	8	9

$N \circ . \times N =$

4	6	8	10	12	14	16	18
6	9	12	15	18	21	24	27
8	12	16	20	24	28	32	36
10	15	20	25	30	35	40	45
12	18	24	30	36	42	48	54
14	21	28	35	42	49	56	63
16	24	32	40	48	56	64	72
18	27	36	45	54	63	72	81

APL

$$(\sim N \in N \circ . \times N) / N \leftarrow 1 \downarrow \iota N$$

2	3	4	5	6	7	8	9

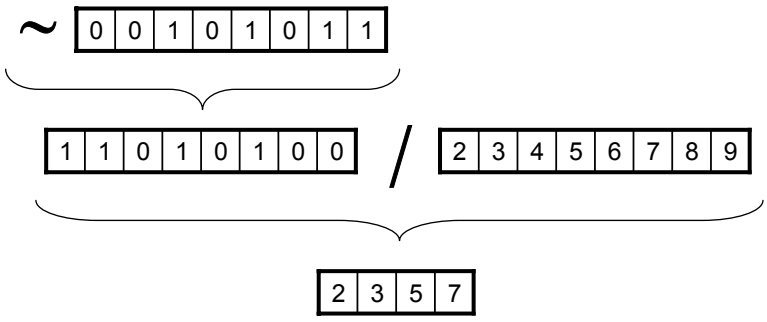
$\in$

4	6	8	10	12	14	16	18
6	9	12	15	18	21	24	27
8	12	16	20	24	28	32	36
10	15	20	25	30	35	40	45
12	18	24	30	36	42	48	54
14	21	28	35	42	49	56	63
16	24	32	40	48	56	64	72
18	27	36	45	54	63	72	81

0	0	1	0	1	0	1	1

APL

$$(\sim N \in N \circ . \times N) / N \leftarrow 1 \downarrow \iota N$$



BrainFuck

18+

```
+++>++>>>+[>>,[>++++++<[[->]<<]<[>]>]>-[<<++++++>>-[<<----->>-
    [->]<]] <[<-<[<]+<+[>]<<+>->>>]<]<[<]>[-[>+++++++<-]>[<+>-
    ]+<<<+++>+> [- [<<+>->- [<<[-]>>- [<<++>+>- [<<-->-
    >>+++<- [<<+>+>>--<- [<<->->- [<<++++>+>>+<- [>-<- [<<->-
    >- [<<->>- [<<+++>>>-<- [<<---->>>++<- [<<++>>>+<- [>[-]<-
    [<<->>>+++<- [<<->>>--<- [<<++++>+>>+<- [<<[-]>->>++<-
    [<<++++++>+>>--<- [<->>++< [<<->>- ]]]]]]]]]]]]]]]]]]]]<[-
    >>[<<+>>-]<<<[>>>+<<<-]<[>>>+<<<-]]>>] >[-[---[-
    <]]>]>[+++[<++++++>--]>]+<++[[>++++++<-]<]>>[-.>]
+++>++>>>+[>>,[>++++++<[[->]<<]<[>]>]>-[<<++++++>>-[<<----->>-
    [->]<]] <[<-<[<]+<+[>]<<+>->>>]<]<[<]>[-[>+++++++<-]>[<+>-
    ]+<<<+++>+> [- [<<+>->- [<<[-]>>- [<<++>+>- [<<-->-
    >>+++<- [<<+>+>>--<- [<<->->- [<<++++>+>>+<- [>-<- [<<->-
    >- [<<->>- [<<+++>>>-<- [<<---->>>++<- [<<++>>>+<- [>[-]<-
    [<<->>>+++<- [<<->>>--<- [<<++++>+>>+<- [<<[-]>->>++<-
    [<<++++++>+>>--<- [<->>++< [<<->>- ]]]]]]]]]]]]]]]]]]]]<[-
    >>[<<+>>-]<<<[>>>+<<<-]<[>>>+<<<-]]>>] >[-[---[-
    <]]>]>[+++[<++++++>--]>]+<++[[>++++++<-]<]>>[-.>]
```

BrainFuck

- Existe un número (ilimitado) de celdas de memoria
- Un puntero selecciona la celda sobre la que se trabaja
- Sólo existen 8 comandos (1 caracter cada uno):

Caracter	Significado
>	puntero a celda siguiente
<	puntero a celda anterior
+	incrementar (en uno) contenido celda
-	decrementar (en uno) contenido celda
.	escribir valor celda
,	leer valor y almacenar en celda
[	si celda es cero, omitir comandos entre **[ ]**
]	si celda no es cero, saltar atrás a **[** correspondiente