Criba de Eratóstenes: Cómo colar números primos. Implementación en Java y VBA para Excel.

Walter Mora F.
wmora2@yahoo.com.mx
Escuela de Matemática
Instituto Tecnológico de Costa Rica

Introducción

La Criba de Eratóstenes es un algoritmo que permite hallar todos los números primos menores que un número natural dado n eliminando los números compuestos de la lista $\{2, 3, ..., n\}$. Es simple y razonablemente eficiente. En este trabajo se presenta un algoritmo (explicado en detalle) y la respectiva implementación. Al final se explica como manejar la memoria para el caso de números grandes.

Palabras claves: Números primos, algoritmo, criba de Eratóstenes.

1.1 Criba de Eratóstenes: Cómo colar números primos.

La criba[1] de Eratóstenes es un algoritmo que permite "colar" todos los números primos menores que un número natural dado n, eliminando los números compuestos de la lista $\{2, ..., n\}$. Es simple y razonablemente eficiente.

[1]Criba, tamiz y zaranda son sinónimos. Una criba es un herramienta que consiste de un cedazo usada para limpiar el trigo u otras semillas, de impurezas. Esta acción de limpiar se le dice cribar o tamizar.

Primero tomamos una lista de números $\{2,3,...,n\}$ y eliminamos de la lista los múltiplos de 2. Luego tomamos el primer entero después de 2 que no fue borrado (el 3) y eliminamos de la lista sus múltiplos, y así sucesivamente. Los números que permanecen en la lista son los primos $\{2,3,5,7,...\}$.

EJEMPLO 1.1 Primos menores que $n = 10$

Lista inicial	2	3	4	5	6	7	8	9	10
Eliminar múltiplos de 2	2	3	~~4~~	5	~~6~~	7	~~8~~	9	~~10~~
Resultado	2	3	5	7	9				
Eliminar múltiplos de 3	2	3	5	7	~~9~~				
Resultado	2	3	5	7					

Primer refinamiento: Tachar solo los impares

Excepto el 2, los pares no son primos, así que podríamos "tachar" solo sobre la lista de impares $\leq n$:

$$\{3,5,9,...,\} = \left\{2i+3 \ : \quad i = 0,1,... \left[\!\left[\frac{n-3}{2}\right]\!\right]\right\}$$

El último impar es n o $n-1$. En cualquier caso, el último impar es $2 \cdot \left[\!\left[\frac{n-3}{2}\right]\!\right] + 3$ pues,

Si n es impar, $n = 2k+1$ y $\left[\!\left[\frac{n-3}{2}\right]\!\right] = k-1 \implies 2(k-1)+3 = n.$

Si n es par, $n = 2k$ y $\left[\!\left[\frac{n-3}{2}\right]\!\right] = k-2 \implies 2(k-2)+3 = 2k-1 = n-1.$

Segundo refinamiento: Tachar de p_k^2 en adelante

En el paso $k-$ésimo hay que tachar los múltiplos del primo p_k desde p_k^2 en adelante.

Esto es así pues en los pasos anteriores se ya se tacharon $3 \cdot p_k$, $5 \cdot p_k, ..., p_{k-1} \cdot p_k$.

Por ejemplo, cuando nos toca tachar los múltiplos del primo 7, ya se han eliminado los múltiplos de $2,3$ y 5, es decir, ya se han eliminado $2 \cdot 7$, $3 \cdot 7$, $4 \cdot 7$, $5 \cdot 7$

y $6 \cdot 7$. Por eso iniciamos en 7^2.

Tercer refinamiento: Tachar mientras $p_k^2 \leq n$

En el paso $k-$ésimo hay que tachar los múltiplos del primo p_k solo si $p_k^2 \leq n$. En otro caso, nos detenemos ahí.

¿Porque?. En el paso $k-$ésimo tachamos los múltiplos del primo p_k desde p_k^2 en adelante, así que si $p_k^2 > n$ ya no hay nada que tachar.

EJEMPLO 1.2 Encontrar los primos menores que 20. El proceso termina cuando el cuadrado del mayor número confirmado como primo es < 20.

1. La lista inicial es $\{2,3,5,7,9,11,13,15,17,19\}$
2. Como $3^2 \leq 20$, tachamos los múltiplos de 3 desde $3^2 = 9$ en adelante:

$$\{2,3,5,7,\cancel{9},11,13,\cancel{15},17,19\}$$

3. Como $5^2 > 20$ el proceso termina aquí.
4. Primos < 20 : $\{2,3,5,7,11,13,17,19\}$

1.1.1 Algoritmo e implementación.

1. Como ya vimos, para colar los primos en el conjunto $\{2,3,...,n\}$ solo consideramos los impares:

$$\left\{2i+3 \ : \quad i=0,1,\ldots \left[\!\left[\frac{n-3}{2}\right]\!\right]\right\} = \{3,5,7,9,\ldots\}$$

2. Por cada primo $p = 2i+3$ (tal que $p^2 < n$), debemos eliminar los *múltiplos impares* de p menores que n, a saber

$$(2k+1)p = (2k+1)(2i+3), \quad k = i+1, i+2, \ldots$$

Observe que si $k = i+1$ entonces el primer múltiplo en ser eliminado es $p^2 = (2i+3)(2i+3)$, como debe ser.

Esto nos dice que para implementar el algoritmo solo necesitamos un arreglo (booleano) de tamaño "`quo(n-3,2)`". En Java se pone "`(n-3)/2`" y en VBA se pone "`(n-3)\2`".

El arreglo lo llamamos `EsPrimo[i], i=0,1,...,(n-3)/2.`

Cada entrada del arreglo "`EsPrimo[i]`" indica si el número $2i+3$ es primo o no.

Por ejemplo

`EsPrimo[0] = true` pues $n = 2\cdot 0+3 = 3$ es primo,

`EsPrimo[1] = true` pues $n = 2\cdot 1+3 = 5$ es primo,

`EsPrimo[2] = true` pues $n = 2\cdot 2+3 = 7$ es primo,

`EsPrimo[3] = false` pues $n = 2\cdot 3+3 = 9$ no es primo.

Si el número $p = 2i+3$ es primo entonces $i = (p-3)/2$ y

`EsPrimo[(p-3)/2] = true.`

Si sabemos que $p = 2i + 3$ es primo, debemos poner

```
EsPrimo[((2k+1)(2i+3) - 3)/2] = false
```

pues estas entradas representan a los múltiplos $(2k + 1)(2i + 3)$ de p. Observe que cuando $i = 0, 1, 2$ tachamos los múltiplos de $3, 5$ y 7; cuando $i = 3$ entonces $2i + 3 = 9$ pero en este momento `esPrimo[3]=false` así que proseguimos con $i = 4$, es decir, proseguimos tachando los múltiplos de 11.

En resumen: Antes de empezar a tachar los múltiplos de $p = 2i + 3$ debemos preguntar si `esPrimo[i]=true.`

Algoritmo 1.1: Criba de Eratóstenes

Entrada: $n \in \mathbb{N}$
Resultado: Primos entre 2 y n

```
1  máx = (n − 3)/2;
2  boolean esPrimo[i],   i = 1, 2, ...,máx;
3  for i = 1, 2, ..., máx do
4      esPrimo[i] =True;
5  i = 0;
6  while (2i + 3)(2i + 3) ≤ n do
7      k = i + 1;
8      if esPrimo(i) then
9          while (2k + 1)(2i + 3) ≤ n do
10             esPrimo[((2k + 1)(2i + 3) − 3)/2] =False;
11             k = k + 1;
12     i = i + 1;
13 Imprimir;
14 for j = 1, 2, ..., máx do
15     if esPrimo[j] =True then
16         Imprima j
```

1.1.1.1 Implementación en Java. Vamos a agregar un método a nuestra clase "`Teoria_Numeros`". El método recibe el número natural $n > 2$ y devuelve un

vector con los números primos $\leq n$. Para colar los números compuestos usamos un arreglo

```
boolean [] esPrimo = new boolean[(n-3)/2].
```

Al final llenamos un vector con los primos que quedan.

```
import java.math.BigInteger;
public class Teoria_Numeros
{   ...
 public static Vector HacerlistaPrimos(int n)
 {
            Vector        salida = new Vector(1);
            int k   = 1;
            int max = (n-3)/2;
            boolean[]   esPrimo  = new boolean[max+1];

            for(int i = 0; i <= max; i++)
                esPrimo[i]=true;

            for(int i = 0; (2*i+3)*(2*i+3) <= n; i++)
            {
              k = i+1;
              if(esPrimo[i])
              {
                while( ((2*k+1)*(2*i+3)) <= n)
                {
                    esPrimo[((2*k+1)*(2*i+3)-3)/2]=false;
                    k++;
                }
              }
             }
             salida.addElement(new Integer(2));
             for(int i = 0; i <=max; i++)
             { if(esPrimo[i])
               salida.addElement(new Integer(2*i+3));
             }
             salida.trimToSize();
```

```
        return salida;
}
public static void main(String[] args)
{
  System.out.println("\n\n");
 //-----------------------------------------------------------------
  int    n = 100;
  Vector primos;
         primos = HacerlistaPrimos(n);
  //Cantidad de primos <= n
  System.out.println("Primos <="+ n+": "+primos.size()+"\n");
  //imprimir vector (lista de primos)
  for(int p = 1; p < primos.size(); p++)
  {
      Integer num = (Integer)primos.elementAt(p);
      System.out.println(""+(int)num.intValue());
  }
//-----------------------------------------------------------------
  System.out.println("\n\n");
}}
```

1.1.1.2 Uso de la memoria En teoría, los arreglos pueden tener tamaño máximo `Integer.MAX_INT` $= 2^{31} - 1 = 2\,147\,483\,647$ (pensemos también en la posibilidad de un arreglo multidimensional!). Pero en la práctica, el máximo tamaño del array depende del hardware de la computadora. El sistema le asigna una cantidad de memoria a cada aplicación; para valores grandes de n puede pasar que se nos agote la memoria (veremos el mensaje "`OutOfMemory Error`"). Podemos asignar una cantidad de memoria apropiada para el programa "cribaEratostenes.java" desde la línea de comandos, si n es muy grande. Por ejemplo, para calcular los primos menores que $n = 100\,000\,000$, se puede usar la instrucción

```
C:\usrdir> java -Xmx1000m -Xms1000m Teoria_Numeros
```

suponiendo que el archivo "`Teoria_Numeros.java`" se encuentra en `C:\usrdir`.

Esta instrucción asigna al programa una memoria inicial (Xmx) de 1000 MB y una memoria máxima (Xms) de 1000 MB (siempre y cuando existan tales recursos de

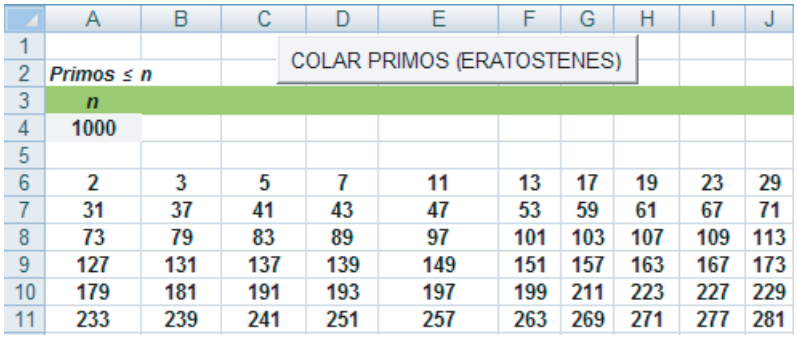
memoria en nuestro sistema).

En todo caso hay que tener en cuenta los siguientes datos

n	Primos $\leq n$
10	4
100	25
1 000	168
10 000	1 229
100 000	9 592
1 000 000	78 498
10 000 000	664 579
100 000 000	5 761 455
1 000 000 000	50 847 534
10 000 000 000	455 052 511
100 000 000 000	4 118 054 813
1 000 000 000 000	37 607 912 018
10 000 000 000 000	346 065 536 839

1.1.1.3 Implementación en Excel. Para la implementación en Excel usamos un cuaderno como el de la figura (1.1).

El número n lo leemos en la celda $(4,1)$. El código VBA incluye una subrutina para imprimir en formato de tabla, con `ncols` columnas. Este último parámetro es opcional y tiene valor default 10. También incluye otra subrutina para limpiar las celdas para que no haya confusión entre los datos de uno y otro cálculo.



	A	B	C	D	E	F	G	H	I	J
1				COLAR PRIMOS (ERATOSTENES)						
2	*Primos ≤ n*									
3	*n*									
4	1000									
5										
6	2	3	5	7	11	13	17	19	23	29
7	31	37	41	43	47	53	59	61	67	71
8	73	79	83	89	97	101	103	107	109	113
9	127	131	137	139	149	151	157	163	167	173
10	179	181	191	193	197	199	211	223	227	229
11	233	239	241	251	257	263	269	271	277	281

Figura 1.1 Primos $\leq n$.

Imprimir en formato de tabla

Para esto usamos la subrutina

```
Imprimir(ByRef Arr() As Long, fi, co, Optional ncols As
                                    Variant).
```

La impresión inicia en la celda "`(fi,co)`". Para imprimir en formato de tabla usamos `Cells(fi + k, co + j)` con el número de columnas `j` variando de 0 a `ncols-1.` Para reiniciar `j` en cero actualizamos `j` con `j = j Mod ncols.` Para cambiar la fila usamos `k.` Esta variable aumenta en 1 cada vez que `j` llega a `ncols-1.` Esto se hace con división entera: `k = k + j \ (ncols - 1)`

Subrutina para borrar celdas

Para esto usamos la subrutina

```
LimpiaCeldas(fi, co, ncols).
```

Cuando hacemos cálculos de distinto tamaño es conveniente borrar las celdas de los cálculos anteriores para evitar confusiones. La subrutina inicia en la celda `(fi,co)` y borra `ncols` columnas a la derecha. Luego pasa a la siguiente fila y

hace lo mismo. Prosigue de igual forma hasta que encuentre la celda `(fi+k,co)` vacía.

```
Option Explicit
    Private Sub CommandButton1_Click()
    Dim n, ncols
    n = Cells(4, 1)
    ncols = Cells(4, 3)
    Call Imprimir(ERATOSTENES(n), 6, 1, ncols)
End Sub

' Imprime arreglo en formato de tabla con "ncols" columnas,
' iniciando en la celda (fi,co)
Sub Imprimir(ByRef Arr() As Long, fi, co, Optional ncols As Variant)
Dim i, j, k
    ' Limpia celdas
    ' f        = fila en que inicia la limpieza
    ' co       = columna en q inicia la limpieza
    ' ncols    = número de columnas a borrar
Call LimpiaCeldas(fi, co, ncols)
If IsMissing(ncols) = True Then
            ncols = 10
End If
'Imprimir
j = 0
k = 0
For i = 0 To UBound(Arr)
    Cells(fi + k, co + j) = Arr(i)
    k = k + j \ (ncols - 1) 'k aumenta 1 cada vez que j llegue a ncols-1
    j = j + 1
    j = j Mod ncols         'j=0,1,2,...,ncols-1
Next i

End Sub


Function ERATOSTENES(n) As Long()
Dim i, j, k, pos, contaPrimos
Dim max As Long
Dim esPrimo() As Boolean
```

```
Dim Primos() As Long
max = (n - 3) \ 2 ’ División entera
ReDim esPrimo(max + 1)
ReDim Primos(max + 1)
For i = 0 To max
    esPrimo(i) = True
Next i
contaPrimos = 0
Primos(0) = 2 ’contado el 2
j = 0
While (2 * j + 3) * (2 * j + 3) <= n
    k = j + 1
    If esPrimo(j) Then
        While (2 * k + 1) * (2 * j + 3) <= n
            pos = ((2 * k + 1) * (2 * j + 3) - 3) \ 2
            esPrimo(pos) = False
            k = k + 1
        Wend
    End If
j = j + 1
Wend

For i = 0 To max
   If esPrimo(i) Then
        contaPrimos = contaPrimos + 1 ’3,5,...
        Primos(contaPrimos) = 2 * i + 3
    End If
Next i

ReDim Preserve Primos(contaPrimos) ’Cortamos el vector
ERATOSTENES = Primos()
End Function


Private Sub LimpiaCeldas(fi, co, nc)
Dim k, j
k = 0
While LenB(Cells(fi + k, co)) <> 0   ’ celda no vac\’ia
      For j = 0 To nc
          Cells(fi + k, co + j) = ""  ’ borra la fila hasta nc columnas
      Next j
      k = k + 1
```

```
Wend
End Sub
```

1.1.2 Primos entre m y n.

Para encontrar todos los primos entre m y n (con $m < n$) procedemos como si estuvieramos colando primos en la lista $\{2,3,...,n\}$, solo que eliminamos los múltiplos que están entre m y n : Eliminamos los múltiplos de los primos p para los cuales $p^2 \leq n$ (o también $p \leq \sqrt{n}$), que están entre m y n.

Múltiplos de p entre m y n

Para los primos p inferiores a $\sqrt{n}$, buscamos el primer múltiplo de p entre m y n.

Si $m-1 = pq+r,\ 0 \leq r < p \implies p(q+1) \geq m$

Así, los múltiplos de p mayores o iguales a m son

$$p(q+1),\ p(q+2),\ p(q+3),\ ... \quad \text{con} \quad q = \text{quo}(m-1,p)$$

EJEMPLO 1.3 Para encontrar los primos entre $m = 10$ y $n = 30$, debemos eliminar los múltiplos de los primos $\leq \sqrt{30} \approx 5$. Es decir, los múltiplos de los primos $p = 2,3,5$.

Como $10-1 = 2\cdot 4+1$, el 2 elimina los números $2(4+k) = 8+2k,\ k \geq 1$; es decir $\{10,12,...,30\}$

Como $10-1 = 3\cdot 3+0$, el 3 elimina los números $3(3+k) = 9+3k,\ k \geq 1$; es decir $\{12,15,18,21,24,27,30\}$

Como $10-1=5\cdot 1+4$, el 5 elimina los números $5(1+k)=5+5k$, $k\geq 1$; es decir $\{10,15,20,25.\}$

Finalmente nos quedan los primos $11,13,17,19,23,29$.

1.1.2.1 Algoritmo. Como antes, solo consideramos los impares entre m y n. Si ponemos

$$min=\text{quo}(m+1-3,2) \text{ y } max=\text{quo}(n-3,2)$$

entonces $2\cdot min+3$ es el primer impar $\geq m$ y $2\cdot max+3$ es el primer impar $\leq n$. Así, los impares entre m y n son los elementos del conjunto $\{2\cdot i+3 : i=min,...,max\}$

Como antes, usamos un arreglo booleano `esPrimo(i)` con $i=min,...,max$. `esPrimo(i)` representa al número $2\cdot i+3$.

EJEMPLO 1.4 Si $m=11$ y 20, $[\![(m+1-3)/2]\!]=4$ y $[\![(n-3)/2]\!]=8$. Luego $2\cdot 4+3=11$ y $2\cdot 8+3=19$.

Para aplicar el colado necesitamos los primos $\leq\sqrt{n}$. Esta lista de primos la obtenemos con la función `Eratostenes(isqrt(n)).` Aquí hacemos uso de la función `isqrt(n)` (algoritmo **??**).

Para cada primo p_i en la lista,

1. si $m\leq p_i^2$, tachamos los múltiplos impares de p_i como antes,

```
1 if m ≤ p_i^2 then
2 |  k = (p_i − 1)/2;
3 |  while (2k + 1)p_i ≤ n do
4 |  |  esPrimo[((2k + 1)p_i − 3)/2] =False;
5 |  |  k = k + 1;
```

Note que si $k = (p_i - 1)/2$ entonces $(2k+1)p_i = p_i^2$

2. si $p_i^2 < m$, tachamos desde el primer múltiplo impar de p_i que supere m :

Los múltiplos de p_i que superan m son $p_i(q+k)$ con $q = \text{quo}(m-1, p)$. De esta lista solo nos interesan los múltiplos impares. Esto requiere un pequeño análisis aritmético.

Como p_i es impar, $p_i(q+k)$ es impar solo si $q+k$ es impar. Poniendo $q_2 = \text{rem}(q, 2)$ entonces $(2k+1-q_2 \ + \ q)$ es impar si $k = q_2,\ q_2+1, \ldots$. En efecto,

$$2k+1-q_2 \ + \ q = \begin{cases} 2k+1 \ + \ q & \text{si} \quad q \quad \text{es par. Aquí} \quad k = q_2 = 0, 1, \ldots \\ 2k \ + \ q & \text{si} \quad q \quad \text{es impar. Aquí} \quad k = q_2 = 1, 2, \ldots \end{cases}$$

Luego, los múltiplos impares de p_i son los elementos del conjunto

$$\{(2k+1-q_2 \ + \ q) \cdot p \ : \ q_2 = \text{rem}(q, 2) \quad \text{y} \quad k = q2, q2+1, \ldots\}$$

La manera de tachar los múltiplos impares de p_i es

```
1 if p_i^2 < m then
2     q = (m − 1)/p ;
3     q_2 = rem(q, 2) ;
4     k = q_2 ;
5     mp = (2k + 1 − q_2 + q) · p_i ;
6     while mp ≤ n do
7         esPrimo[(mp − 3)/2] =False;
8         k = k + 1;
9         mp = (2k + 1 − q_2 + q) · p_i
```

Ahora podemos armar el algoritmo completo.

Algoritmo 1.2: Colado de primos entre m y n.

Entrada: $n, m \in \mathbb{N}$ con $m < n$.
Resultado: Primos entre m y n

```
Primo() = una lista de primos ≤ √n ;
min = (m + 1 − 3)/2; max = (n − 3)/2 ;
esPrimo[i], i = min, ..., max ;
for j = min, ..., max do
    esPrimo[j] =True;
np = cantidad de primos en la lista Primos;
Suponemos Primo(0) = 2;
for i = 1, 2, ..., np do
    if m ≤ p_i^2 then
        k = (p_i − 1)/2 ;
        while (2k + 1)p_i ≤ n do
            esPrimo[((2k + 1)p_i − 3)/2] =False;
            k = k + 1;
    if p_i^2 < m then
        q = (m − 1)/p ;
        q_2 = rem(q, 2) ;
        k = q_2 ;
        mp = (2k + 1 − q_2 + q) · p_i ;
        while mp ≤ n do
            esPrimo[(mp − 3)/2] =False;
            k = k + 1;
            mp = (2k + 1 − q_2 + q) · p_i
Imprimir;
for j = min, ..., max do
    if esPrimo[j] =True then
        Imprima 2 ∗ i + 3
```

1.1.2.2 *Implementación en Excel.*

Para la implementación en Excel usamos un cuaderno como el de la figura (1.2).

m y n los leemos en la celdas $(4,1)$, $(4,2)$. Como antes, el código VBA hace referencia a las subrutinas para imprimir en formato de tabla y limpiar las celdas

(sección 1.1.1.3).

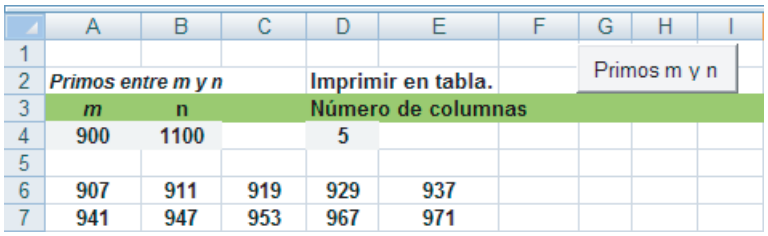


Figura 1.2 Primos $\leq n$.

En VBA Excel podemos declarar un arreglo que inicie en *min* y finalice en *max*, como el algoritmo. Por eso, la implementación es muy directa.

```
Option Explicit
Private Sub CommandButton1_Click()
Dim n, m, ncols
m = Cells(4, 1)
n = Cells(4, 2)
ncols = Cells(4, 4)
Call Imprimir(PrimosMN(m, n), 6, 1, ncols)
End Sub

Sub Imprimir(ByRef Arr() As Long, fi, co, Optional ncols As Variant)
...
End Sub

Function ERATOSTENES(n) As Long()
...
End Sub

Function isqrt(n) As Long
Dim xk, xkm1
If n = 1 Then
   xkm1 = 1
End If
If n > 1 Then
```

```
    xk = n
    xkm1 = n \ 2
    While xkm1 < xk
         xk = xkm1
         xkm1 = (xk + n \ xk) \ 2
    Wend
End If
isqrt = xkm1
End Function
’ m < n
Function PrimosMN(m, n) As Long()
Dim i, j, k, pos, contaPrimos, mp, q, q2
Dim min, max
Dim esPrimo() As Boolean
Dim primo() As Long
Dim PrimosMaN() As Long

min = Int((m + 1 - 3) \ 2)
max = Int((n - 3) \ 2)

ReDim esPrimo(min To max)
ReDim PrimosMaN((n - m + 2) \ 2)
For i = min To max
    esPrimo(i) = True
Next i

primo = ERATOSTENES(isqrt(n))

For i = 1 To UBound(primo)          ’primo(1)=3
    If m <= primo(i) * primo(i) Then
       k = (primo(i) - 1) \ 2
       While (2 * k + 1) * primo(i) <= n
       esPrimo(((2 * k + 1) * primo(i) - 3) \ 2) = False
       k = k + 1
       Wend
    End If
    If primo(i) * primo(i) < m Then
       q = (m - 1) \ primo(i)  ’p(q+k)-> p*k
       q2 = q Mod 2
       k = q2
       mp = (2 * k + 1 - q2 + q) * primo(i)  ’m\’ultiplos impares
        While mp <= n
```

```
            esPrimo((mp - 3) \ 2) = False
            k = k + 1
            mp = (2 * k + 1 - q2 + q) * primo(i)
       Wend
    End If
Next i

If m > 2 Then
    contaPrimos = 0
Else
    contaPrimos = 1
    PrimosMaN(0) = 2
End If

For i = min To max
   If esPrimo(i) Then
        PrimosMaN(contaPrimos) = 2 * i + 3
        contaPrimos = contaPrimos + 1 '3,5,...
    End If
Next i

If 1 <= contaPrimos Then
ReDim Preserve PrimosMaN(contaPrimos - 1)
Else
ReDim PrimosMaN(0)
End If

PrimosMN = PrimosMaN()
End Function
```

Bibliografía

[1] Lindsay N. Childs. *A Concrete Introduction to Higher Algebra.* Springer-Verlag New York, 1995.

[2] Hans Riesel. *Prime Numbers and Computer Methods for Factorization.* Springer; 2 edition. 1994.

[3] Paulo Ribenboim. *The New Book of Prime Number Records.* 3rd ed. Springer. 1995.

[4] R. Sedgewick, K. Wayne. *Introduction to Programming in Java. An Interdisciplinary Approach.* Addison-Wiley. 2008.