

① 予備知識

エラトステネスの篩とは、次の手順によって素数表を生成する手法です。

Step 1. 2 以外の 2 の倍数 (4, 6, 8, ...) を全て消す。

	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32	33	34	

Step 2. 3 以外の 3 の倍数を全て消す。

	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32	33	34	

Step 3. 5 以外の 5 の倍数を全て消す。

	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32	33	34	

.....

ふるい落とされずに最後まで残った数が素数になります。

② 素数表を作る

自然数 N を引数にとり、エラトステネスの篩により N までの素数表を生成する関数 `eratosthenes` を作ってみたいと思います。

```

1  eratosthenes(N) := block([p: 2, L],
2      L: makelist(1, i, 1, N),
3      while(p <= sqrt(N)) do (
4          if(L[p] = 1) then (
5              j: p,
6              while(j + p <= N) do (
7                  j: j + p,
8                  L[j]: 0
9              )
10         ),
11         p: p + 1
12     ),

```

- eratosthenes.mac -

```

13   for i: 2 thru N do
14       if L[i] = 1 then
15           print(i)
16   );

```

全ての要素を1で埋め尽くしたリスト L を用意し (2行目)、整数 j をふるい落とす作業を、リスト L の j 番目の要素 $L[j]$ に0を代入する操作として実装しています (8行目)。すなわち、引数 N の平方根 \sqrt{N} までの全ての自然数 p に対して (3行目)、既にふるい落とされているかチェックし (4行目)、ふるい落とされていなければ (このとき、 p は素数です) 5行目から8行目までの処理が実行されます。変数 j は、数列と見なすと初項 p (5行目)、公差 p の等差数列 (7行目) ですから、「 p 以外の全ての p の倍数」に対して、8行目の処理が実行されることとなります。

13行目以降の for 文は、最後までふるい落とされなかった数 (素数) i を出力する処理です。

```

(%i1) load("eratosthenes.mac");
(%o1)                                     eratosthenes.mac
(%i2) eratosthenes(100);
2
3
5
7
11

(中略)

83
89
97
(%o2)                                     done

```

100 までの素数を出力させてみました。

③ 素因数分解

素数表を利用した素因数分解関数を定義してみたいと思います。自然数 M に対して、前節のエラトステネスの篩を用いて、平方根 $N = \sqrt{M}$ までの素数表を作成し、各素数で M を割っていきます。

```

1  factor_e(M) := block([p: 2, L,                                     -factor_e.mac -
2      N: floor(sqrt(M)), S: []],
3      load("numericalio"),
4      L: makelist(1, i, 1, N),

```

```

5  while(p <= sqrt(N)) do (
6      if(L[p] = 1) then (
7          k: p,
8          while(k + p <= N) do (
9              k: k + p,
10             L[k]: 0
11         )
12     ),
13     p: p + 1
14 ),
15 with_stdout("eratosthenes_output.txt",
16     for i: 2 thru N do
17         if L[i] = 1 then
18             print(i)
19     ),
20     for i in read_list("eratosthenes_output.txt") do
21         while(remainder(M, i) = 0) do (
22             S: cons(i, S),
23             M: M / i
24         ),
25         if M # 1 then S: cons(M, S),
26         S
27 );

```

今回の関数 `factor_e` の 4 行目から 14 行目までと、16 行目から 18 行目までは前節の関数 `eratosthenes` と全く同じです。15 行目の関数 `with_stdout` で、前節では画面（標準出力）へ出力していた素数表をファイル `eratosthenes_out.txt` へ出力しています。

20 行目から 24 行目までの `for` 文が素因数分解を実行している部分です。20 行目の関数 `read_list` によって、素数表 `eratosthenes_out.txt` から素数を 1 つずつ取り出していますが、この関数 `read_list` を利用するために、3 行目で `numericalio.mac` を読み込んでいます。

```

(%i3) load("factor_e.mac");
(%o3)                                     factor_e.mac
(%i4) factor_e(1234567890);
(%o4)                                     [3803, 3607, 5, 3, 3, 2]
(%i5) apply("*", %);
(%o5)                                     1234567890

```

作成日：平成 20 年 8 月 23 日～8 月 24 日
ソフトウェア：Maxima 5.15.0cvs & CMU Common Lisp Snapshot 2008-08 (19E)