素数生成法の開発

０．目次

正整数ｎ以下の素数をすべて求める問題を考察する。
まず、素朴な方法を取り上げ改良していく。つぎに、「ふるい」という考え方を適用すると、劇的に改良されることを示す。

２．エラトステネスの方法

方法２．１

正整数 n 以下の素数をつぎの手順で生成する。

手順（０） 2 を素数とする。
手順（１） 数列 {3,5,7,･･･,n} を用意する。
手順（２） 数列の中の最小値pを素数とし、その数自身と
その数の倍数（2p,3p,4p,･･･）を数列から
取り除き、新しい数列を構成する。
手順（３） 新しい数列が空になるまで手順（２）を繰り返す。

方法２．２　エラトステネスの方法の改良

方法２．１の手順（２）、手順（３）を改良する。
2*p,3*p,4*p,...,(p-1)*pは、すでに数列から取り除かれて
いるはずだから省略できる。

手順（０） 2を素数とする。
手順（１） 数列 {3,5,7,･･･,n}を用意する。
手順（２） 数列の中の最小値pを素数とし、その数自身と
その数の倍数（p*p,p*(p+1),p*(p+2),･･･）を
数列から取り除き、新しい数列を構成する。
手順（３） p*p が n 以下の間、手順（２）を繰り返す。

方法２．３　エラトステネスの方法の改良

方法２．２の手順（２）を改良する。
pが奇数なので、p*(p+1),p*(p+3),p*(p+5),...は偶数になり
省略できる。

手順（０） 2を素数とする。
手順（１） 数列 {3,5,7,･･･,n}を用意する。
手順（２） 数列の中の最小値p素数とし、その数自身と
その数の倍数（p*p,p*(p+2),p*(p+4),･･･）を
数列から取り除き、新しい数列を構成する。
手順（３） p*p が n 以下の間、手順（２）を繰り返す。

方法２．４　エラトステネスの方法の改良

方法２．３の手順（２）を改良する。

手順（０）2を素数とする。
手順（１）数列 {3,5,7,...,n}を用意する。
手順（２）数列の中の最小値を素数pとし、その数自身と
その数の倍数を数列から取り除き、新しい数列を
構成する。
pが6k+1の形の素数とわかったとき、取り除く倍数を、
p*p,p*(p+4),p*(p+6),p*(p+10),p*(p+12),...とする。
pが6k+5の形の素数とわかったとき、取り除く倍数を、
p*p,p*(p+2),p*(p+6),p*(p+8),p*(p+12),...とする。
手順（３）　p*p が n 以下の間、手順（２）を繰り返す。

１．素朴な素数生成法

正整数ｎ以下の素数をすべて求める問題を考察する。まず、素朴な方法を取り上げ改良していく。

方法１．１

素朴な方法である。素数の候補（3以上の奇数）pを用意し、その候補に対して、3以上p-2以下の奇数dで割っていく。どの数でも割り切れない場合、素数とし、いずれかの数で割り切れた場合、合成数とする。

素数の候補　p	3, 5, 7, 9, 11, 13, 15, 17, 19, ･･･
除数　d	奇数で、3≦d≦p-2

●例

25 は、3で割り切れないが、5で割り切れるので合成数である。
29 は、3, 5, 7, …, 27で割り切れないので素数である。

●プログラム(prm111.c)

```
1  /* << prm111.c >> */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define N 1000000 /* 配列サイズの最大値。*/
5
6  int main() {
7    int a[N+1], /* 素数を保存する配列。*/
8        d,      /* 除数。*/
9        i,k,
10       n,      /* nの値。*/
11       p;      /* 素数の候補。*/
12
13   /* 正整数nの読み込み。*/
14   scanf("%d",&n);
15   if( n <= 0)||(n > N) ) { exit(0); }
16
17   /* 初期設定。2を素数として保存。*/
18   k = 1; a[k] = 2;
19   /* 素数候補の初期値。*/
20   p = 3;
21
22   /* 素数の探索。*/
23   while( p <= n ) {
24     /* 除数の初期値。*/
25     d = 3;
26     while( d <= p-2 ) {
27       if( p%d != 0 ) {
28         /* 除数の更新。*/
29         d = d + 2 ;
```

```
30        } else {
31          /* 割り切れた。*/
32          break ;
33        }
34      }
35
36      if( d >= p ) {
37        /* 素数候補p を素数として保存。*/
38        k++; a[k] = p ;
39      }
40      /* 素数候補pの更新。*/
41      p = p + 2;
42    }
43
44    /* 求めた素数を出力する。*/
45    printf("%d 以下の素数：%d 個 ¥n",n,k);
46    for( i=1; i<=k; i++ ) {
47      printf("%5d",a[i]);
48      if( i%10 == 0 ) { printf("¥n"); }
49    }
      printf("¥n");
}
```

実行結果

```
% cc prm111.c
% ./a.out
1000
1000 以下の素数：168 個
    2    3    5    7   11   13   17   19   23   29
   31   37   41   43   47   53   59   61   67   71
   73   79   83   89   97  101  103  107  109  113
＜＜途中省略＞＞
  811  821  823  827  829  839  853  857  859  863
  877  881  883  887  907  911  919  929  937  941
  947  953  967  971  977  983  991  997
```

実行時間（出力時間を除く）

```
実行時間測定
% time ./a.out
100000   10の5乗
100000 以下の素数：9592 個
4.0u 0.0s 0:06 60% 0+0k 0+0io 0pf+0w
% time ./a.out
1000000   10の6乗
1000000 以下の素数：78498 個
361.0u 0.0s 6:06 98% 0+0k 0+0io 0pf+0w
```

方法１．２

改良点：方法１．１において、除数dの範囲を小さくする。

素数の候補（3以上の奇数）pを用意し、その候補に対して、3以上、[$\sqrt{p}$]以下の奇数dで割っていく。どの数でも割りきれない場合、素数とし、いずれかの数で割り切れた場合、合成数とする。ここで、［x］は実数xの整数部分を意味する。

素数の候補　p	3, 5, 7, 9, 11, 13, 15, 17, 19, ･･･
除数　d	奇数で、3≦d≦[$\sqrt{p}$]

●例

25 は、5で割り切れるので合成数である。
29 は、3,5で割り切れないので素数である。

●［$\sqrt{p}$］以下の奇数dで割ればよい理由

［$\sqrt{p}$］より大きい奇数dで割り切れたとすると、p／dは［$\sqrt{p}$］以下の奇数となる。すなわち、［$\sqrt{p}$］以下の奇数の約数があることになる。つまり、3以上［$\sqrt{p}$］以下の奇数で割って調べれば十分である。

●プログラム(prm112.c)

```
1  /* << prm112.c >> */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define N 1000000 /* 配列サイズの最大値。*/
5
6  int main() {
7    int a[N+1], /* 素数を保存する配列。*/
8        d,      /* 除数。*/
9        i,k,
10       n,      /* nの値。*/
11       p;      /* 素数の候補。*/
12
13   /* 正整数nの読み込み。*/
14   scanf("%d",&n);
15   if( n <= 0)||(n > N) ) { exit(0); }
16
17   /* 初期設定。2を素数として保存。*/
18   k = 1; a[k] = 2;
19   /* 素数候補の初期値。*/
20   p = 3;
21
22   /* 素数の探索。*/
23   while( p <= n ) {
24     /* 除数の初期値。*/
```

```
25
26      d = 3;
27      while( d*d <= ④ p ) {
28        if( p%d != 0 ) {
29          /* 除数の更新。*/
30          d = d + 2;
31        } else {
32          /* 割り切れた。*/
33          break;
34        }
35      }
36
37      if( d*d > p ) {
38      /* 素数候補p を素数として保存。*/
39        k++; a[k] = p;
40      }
41      /* 素数候補pの更新。*/
42      p = p + 2;
43    }
44
45    /* 求めた素数を出力する。*/
46    printf("%d 以下の素数：%d 個 ¥n",n,k);
47    for( i=1; i<=k; i++ ) {
48      printf("%5d",a[i]);
49      if( i%10 == 0 ) { printf("¥n"); }
50    }
51    printf("¥n");
52  }
```

実行結果：方法１．１と同じ

実行時間（出力時間を除く）

```
実行時間測定
% time ./a.out
100000   10の5乗
100000 以下の素数：9592 個
0.0u 0.0s 0:02 0% 0+0k 0+0io 0pf+0w
% time ./a.out
1000000   10の6乗
1000000 以下の素数：78498 個
0.0u 0.0s 0:03 0% 0+0k 0+0io 0pf+0w
% time ./a.out
10000000   10の7乗
10000000 以下の素数：664579 個
17.0u 0.0s 0:21 78% 0+0k 0+0io 0pf+0w
```

方法１．３

改良点：方法１．２において、候補pから3の倍数を除く。

素数の候補（3の倍数を除く奇数）pを用意し、その候補に対して、5以上［$\sqrt{p}$］以下の奇数dで割っていく。どの数でも割りきれない場合、素数とし、いずれかの数で割り切れた場合、合成数とする。

素数の候補 p	5, 7, 11, 13, 17, 19, 23, 25, ･･･
除数 d	奇数で、5≦d≦［$\sqrt{p}$］

●プログラム(prm113.c)

```
 1  /* << prm113.c >> */
 2  #include <stdio.h>
 3  #include <stdlib.h>
 4  #define N 1000000 /* 配列サイズの最大値。*/
 5
 6  int main() {
 7    int a[N+1], /* 素数を保存する配列。*/
 8        d,      /* 除数。*/
 9        dp,     /* p の増分（2または4となる）。*/
10        i,k,
11        n,      /* nの値。*/
12        p;      /* 素数の候補。*/
13
14    /* 正整数nの読み込み。*/
15    scanf("%d",&n);
16    if( n <= 0)||(n > N) ) { exit(0); }
17
18    /* 初期設定。2,3は、素数として保存。*/
19    a[1] = 2; a[2] = 3; k = 2;
20    /* 素数候補pの初期値。*/
21    p = 5;
22    /* 増分dpの初期値。*/
23    dp = 2;
24
25    /* 素数の探索。*/
26    while( p <= n ) {
27      /* 除数dの初期値。*/
28      d = 5;
29      while( d*d <= p ) {
30        if( p%d != 0 ) {
31          /* 除数dの更新。*/
32          d = d + 2;
33        } else {
34          /* 割り切れた。*/
35          break;
36        }
37      }
```

```
38
39      if( d*d > p) {
40        /* 素数候補p を素数として保存。*/
41        k++; a[k] = p;
42      }
43      /* 素数候補の更新。*/
44      p = p + dp;
45      /* 増分dpの更新。*/
46      dp = 6 - dp;
47    }
48
49    /* 求めた素数を出力する。*/
50    printf("%d 以下の素数：%d 個 ¥n",n,k);
51    for( i=1; i<=k; i++ ) {
52      printf("%5d",a[i]);
53      if( i%10 == 0 ) { printf("¥n"); }
54    }
55    printf("¥n");
56  }
```

実行結果：方法１．１と同じ

実行時間（出力時間を除く）

```
実行時間測定
% time ./a.out
100000   10の5乗
100000 以下の素数：9592 個
0.0u 0.0s 0:02 0% 0+0k 0+0io 0pf+0w
% time ./a.out
1000000   10の6乗
1000000 以下の素数：78498 個
0.0u 0.0s 0:03 0% 0+0k 0+0io 0pf+0w
% time ./a.out
10000000   10の7乗
10000000 以下の素数：664579 個
17.0u 0.0s 0:20 83% 0+0k 0+0io 0pf+0w
```

方法１．４

改良点：方法１．３において、除数dとしてすでに得られた素数を使う。

素数の候補（3の倍数を除く奇数）pを用意し、その候補に対して、すでに得られた［$\sqrt{p}$］以下の素数dで割っていく。どの数でも割りきれない場合、素数とし、いずれかの数で割り切れた場合、合成数とする。

素数の候補 p	5, 7, 11, 13, 17, 19, 23, 25, ･･･
除数 d	すでに得られた［$\sqrt{p}$］以下の素数

●プログラム(prm114.c)

```
1  /* << prm114.c >> */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define N 1000000 /* 配列サイズの最大値。*/
5
6  int main() {
7    int a[N+1], /* 素数を保存する配列。*/
8        d,      /* 除数。*/
9        dp,     /* p の増分（2または4となる）。*/
10       i,k,
11       n,      /* nの値。*/
12       p;      /* 素数の候補。*/
13
14   /* 正整数nの読み込み。*/
15   scanf("%d",&n);
16   if( n <= 0)||(n > N) ) { exit(0); }
17
18   /* 初期設定。2,3は、素数として保存。*/
19   a[1] = 2; a[2] = 3; k = 2;
20   /* 素数候補pの初期値。*/
21   p = 5;
22   /* 増分dpの初期値。*/
23   dp = 2;
24
25   /* 素数の探索。*/
26   while( p <= n ) {
27     /* i番目の素数を除数dとする。*/
28     i = 2;
29     d = a[i];
30     while( d*d <= p ) {
31       if( p%d != 0 ) {
32         /* 除数dをi番目の素数とする。*/
33         i++; d = a[i];
34       } else {
35         /* 割り切れた。*/
36         break;
37       }
```

```
38      }
39
40      if( d*d > p ) {
41        /* 素数候補p を素数として保存。*/
42        k++; a[k] = p;
43      }
44      /* 素数候補pを更新する。*/
45      p = p + dp;
46      /* 増分dpを更新する。*/
47      dp = 6 - dp;
48    }
49
50    /* 求めた素数を出力する。*/
51    printf("%d 以下の素数：%d 個 ¥n",n,k);
52    for( i=1; i<=k; i++ ) {
53      printf("%5d",a[i]);
54      if( i%10 == 0 ) { printf("¥n"); }
55    }
56    printf("¥n");
57  }
```

実行結果：方法１．１と同じ

実行時間（出力時間を除く）

```
実行時間測定
% time ./a.out
100000   10の5乗
100000 以下の素数：9592 個
0.0u 0.0s 0:02 0% 0+0k 0+0io 0pf+0w
% time ./a.out
1000000   10の6乗
1000000 以下の素数：78498 個
0.0u 0.0s 0:02 0% 0+0k 0+0io 0pf+0w
% time ./a.out
10000000   10の7乗
10000000 以下の素数：664579 個
6.0u 0.0s 0:09 63% 0+0k 0+0io 0pf+0w
```

実行時間を厳密に測定する。

●プログラム(prm114t.c)

```
 1  /* << prm114t.c >> */
 2  #include <stdio.h>
 3  #include <stdlib.h>
 4  #include <time.h>
 5  #define N 1000000 /* 配列サイズの最大値。*/
 6
 7  int main() {
 8    int a[N+1], /* 素数を保存する配列。*/
 9        d,      /* 除数。*/
10        dp,     /* p の増分（2または4となる）。*/
11        i,k,
12        n,      /* nの値。*/
13        p,      /* 素数の候補。*/
14        t1,t2;  /* 測定用変数。*/
15
16    /* 正整数nの読み込み。*/
17    scanf("%d",&n);
18    if( n <= 0)||(n > N) ) { exit(0); }
19
20    /* 測定開始時刻：t1 */
21    t1 = clock();
22
23    /* 初期設定。2,3は、素数として保存。*/
24    a[1] = 2; a[2] = 3; k = 2;
25    p = 5;
26    dp = 2;
27
28    /* 素数の探索。*/
29    while( p <= n ) {
30      i = 2;
31      d = a[i];
32      while( d*d <= p ) {
33        if( p%d != 0 ) {
34          i++; d = a[i];
35        } else {
36          break;
37        }
38      }
39      if( d*d > p ) {
40        k++; a[k] = p; /* p を素数として保存。*/
41      }
42      p = p + dp;
43      dp = 6 - dp;
44    }
45
46    /* 測定終了時刻：t2 */
```

```
47    t2 = clock();
48
49    /* 測定時間を出力する。*/
50    printf("%d 以下の素数 %d個   生成時間%d ms¥n",
51            n,k,(t2-t1)/1000);
52  }
```

実行時間（出力時間を除く）

```
% cc prm114t.c
% ./a.out
100000   10の5乗
100000 以下の素数 9592個   生成時間10 ms
% ./a.out
1000000   10の6乗
1000000 以下の素数 78498個   生成時間320 ms
% ./a.out
10000000   10の7乗
10000000 以下の素数 664579個   生成時間6820 ms
```

２．エラトステネスの方法

方法２．１

正整数 n 以下の素数をつぎの手順で生成する。

手順（０） 2 を素数とする。
手順（１） 数列 {3,5,7,･･･,n} を用意する。
手順（２） 数列の中の最小値pを素数とし、その数自身と
その数の**倍数（2p,3p,4p,･･･）**を数列から
取り除き、新しい数列を構成する。
手順（３） 新しい数列が空になるまで手順（２）を繰り返す。

●例 n=29

2を素数とする。
１番目の数列 {3,5,7,9,11,13,15,17,19,21,23,25,27,29}
3を素数とし、3の倍数を削除する。

２番目の数列 {5,7,11,13,17,19,23,25,29}
5を素数とし、5の倍数を削除する。

３番目の数列 {7,11,13,17,19,23,29}
7を素数とし、7の倍数を削除する。

４番目の数列 {11,13,17,19,23,29}
11を素数とし、11の倍数を削除する。

５番目の数列 {13,17,19,23,29}
13を素数とし、13の倍数を削除する。

６番目の数列 {17,19,23,29}
17を素数とし、17の倍数を削除する。

７番目の数列 {19,23,29}
19を素数とし、19の倍数を削除する。

８番目の数列 {23,29}
23を素数とし、23の倍数を削除する。

９番目の数列 {29}
29を素数とし、29の倍数を削除する。

１０番目の数列 {}
空集合となり終了。

●プログラム(prm211.c)

```
/* << prm211.c >> */
#include <stdio.h>
#include <stdlib.h>
#define N 10000000 /* 配列サイズの最大値。*/

int main() {
  char x[N+1]; /* 数列を配列xで表現する。
                  すなわち、数列の要素iは、x[i]='0'、
                   数列の要素でなくなると、x[i]='1'とする。*/
  int count,   /* 素数の個数。*/
      i,k,
      n,       /* nの値。*/
      p;       /* 素数の候補。*/

  /* 正整数nの読み込み。*/
  scanf("%d",&n);
  if( n <= 0)||(n > N) ) { exit(0); }

  /* 初期設定。iが素数の候補である間x[i]='0'。
               iが素数pの倍数となる場合x[i]='1'。*/
  for( i=3; i<=n; i=i+2 ) { x[i] = '0'; }
  /* 素数候補pの初期値。*/
  p = 3;

  /* 素数の探索。*/
  while( p <= n ) {
    if( x[p] == '0' ) {
      /* pの倍数を削除。*/
      for( k=2*p; k<=n; k=k+p ) {
        x[k] = '1';
      }
    }
    /* 素数候補pの更新。*/
    p = p + 2;
  }

  /* x[i]='0' となる数 i は素数なので出力する。*/
  count = 1;
  printf("%5d",2);
  for( i=3; i<=n; i=i+2 ) {
    if( x[i] == '0' ) {
      count++; printf("%5d",i);
      if( count%10 == 0 ) { printf("¥n"); }
    }
  }
  printf("¥n%d 以下の素数：%d 個 ¥n",n,count);
}
```

実行結果

```
% cc prm211.c
% ./a.out
1000
    2    3    5    7   11   13   17   19   23   29
   31   37   41   43   47   53   59   61   67   71
   73   79   83   89   97  101  103  107  109  113
  127  131  137  139  149  151  157  163  167  173
  179  181  191  193  197  199  211  223  227  229
  233  239  241  251  257  263  269  271  277  281
  283  293  307  311  313  317  331  337  347  349
  353  359  367  373  379  383  389  397  401  409
  419  421  431  433  439  443  449  457  461  463
  467  479  487  491  499  503  509  521  523  541
  547  557  563  569  571  577  587  593  599  601
  607  613  617  619  631  641  643  647  653  659
  661  673  677  683  691  701  709  719  727  733
  739  743  751  757  761  769  773  787  797  809
  811  821  823  827  829  839  853  857  859  863
  877  881  883  887  907  911  919  929  937  941
  947  953  967  971  977  983  991  997
1000 以下の素数：168 個
```

実行時間（出力時間は除く）

```
実行時間測定
% ./a.out
100000   10の5乗
100000 以下の素数 9592個   生成時間0 ms
% ./a.out
1000000   10の6乗
1000000 以下の素数 78498個   生成時間20 ms
% ./a.out
10000000   10の7乗
10000000 以下の素数 664579個   生成時間610 ms
```

方法２．２　エラトステネスの方法の改良

方法２．１の手順（２）、手順（３）を改良する。
2*p,3*p,4*p,...,(p-1)*pは、すでに数列から取り除かれているはずだから省略できる。

手順（０）　2を素数とする。
手順（１）　数列 {3,5,7,･･･,n}を用意する。
手順（２）　数列の中の最小値pを素数とし、その数自身と
　　その数の**倍数 (p*p,p*(p+1),p*(p+2),･･･)** を
　　数列から取り除き、新しい数列を構成する。
手順（３）　p*p が n 以下の間、手順（２）を繰り返す。

●例 n=29

　2を素数とする。
１番目の数列 {3,5,7,9,11,13,15,17,19,21,23,25,27,29}
　3を素数とし、3*3,3*4,3*5,3*6,3*7,3*8,3*9を削除する。

２番目の数列 {5,7,11,13,17,19,23,25,29}
　5を素数とし、5*5を削除する。

３番目の数列 {7,11,13,17,19,23,29}
　7を素数とする。削除する要素がなくなったので、
　{11,13,17,19,23,29}を素数とし終了する。

●プログラム(prm221.c)

```
1  /* << prm221.c >> */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define N 10000000 /* 配列サイズの最大値。*/
5
6  int main() {
7    char x[N+1]; /* 数列を配列xで表現する。
8                    すなわち、数列の要素iは、x[i]='0'、
9                     数列の要素でなくなると、x[i]='1'とする。*/
10   int count,    /* 素数の個数。*/
11       i,k,
12       n,        /* nの値。*/
13       p;        /* 素数の候補。*/
14
15   /* 正整数nの読み込み。*/
16   scanf("%d",&n);
17   if( n <= 0)||(n > N) ) { exit(0); }
18
19   /* 初期設定。iが素数の候補である間x[i]='0'。
20                 iが素数pの倍数となる場合x[i]='1'。*/
21   for( i=3; i<=n; i=i+2 ) { x[i] = '0'; }
22   /* 素数候補pの初期値。*/
```

```
23    p = 3;
24
25    /* 素数の探索。*/
26    while( p*p <= n ) {
27      if( x[p] == '0' ) {
28        /* pの倍数を削除。*/
29        for( k=p*p; k<=n; k=k+p ) {
30          x[k] = '1';
31        }
32      }
33      /* 素数候補pの更新。*/
34      p = p + 2;
35    }
36
37    /* x[i]='0' となる数 i は素数なので出力する。*/
38    count = 1;
39    printf("%5d",2);
40    for( i=3; i<=n; i=i+2 ) {
41      if( x[i] == '0' ) {
42        count++; printf("%5d",i);
43        if( count%10 == 0 ) { printf("¥n"); }
44      }
45    }
46    printf("¥n%d 以下の素数：%d 個 ¥n",n,count);
47  }
```

実行結果：方法２．１と同じ

実行時間（出力時間は除く）

```
実行時間測定
% ./a.out
100000   10の5乗
100000 以下の素数 9592個   生成時間0 ms
% ./a.out
1000000   10の6乗
1000000 以下の素数 78498個   生成時間10 ms
% ./a.out
10000000   10の7乗
10000000 以下の素数 664579個   生成時間340 ms
```

方法２．３　エラトステネスの方法の改良

方法２．２の手順（２）を改良する。
pが奇数なので、p*(p+1),p*(p+3),p*(p+5),...は偶数になり省略できる。

手順（０）　2を素数とする。
手順（１）　数列 {3,5,7,･･･,n}を用意する。
手順（２）　数列の中の最小値p素数とし、その数自身と
　　その数の**倍数（p*p,p*(p+2),p*(p+4),･･･）**を
　　数列から取り除き、新しい数列を構成する。
手順（３）　p*p が n 以下の間、手順（２）を繰り返す。

●例 n=29

2を素数とする。
１番目の数列 {3,5,7,9,11,13,15,17,19,21,23,25,27,29}
3を素数とし、3*3,3*5,3*7,3*9を削除する。

２番目の数列 {5,7,11,13,17,19,23,25,29}
5を素数とし、5*5を削除する。

３番目の数列 {7,11,13,17,19,23,29}
7を素数とする。削除する要素がなくなったので、
{11,13,17,19,23,29}を素数とし終了する。

●プログラム(prm231.c)

```
1  /* << prm231.c >> */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define N 10000000 /* 配列サイズの最大値。*/
5
6  int main() {
7    char x[N+1]; /* 数列を配列xで表現する。
8                    すなわち、数列の要素iは、x[i]='0'、
9                     数列の要素でなくなると、x[i]='1'とする。*/
10   int count,    /* 素数の個数。*/
11       i,k,
12       n,        /* nの値。*/
13       p;        /* 素数の候補。*/
14
15   /* 正整数nの読み込み。*/
16   scanf("%d",&n);
17   if( n <= 0)||(n > N) ) { exit(0); }
18
19   /* 初期設定。iが素数の候補である間x[i]='0'。
20                 iが素数pの倍数となる場合x[i]='1'。*/
21   for( i=3; i<=n; i=i+2 ) { x[i] = '0'; }
22   /* 素数候補pの初期値。*/
23   p = 3;
```

```
24
25    /* 素数の探索。*/
26    while( p*p <= n ) {
27      if( x[p] == '0' ) {
28        /* pの倍数を削除。*/
29        for( k=p*p; k<=n; k=k+2*p ) {
30          x[k] = '1';
31        }
32      }
33      /* 素数候補pの更新。*/
34      p = p + 2;
35    }
36
37    /* x[i]='0' となる数 i は素数なので出力する。*/
38    count = 1;
39    printf("%5d",2);
40    for( i=3; i<=n; i=i+2 ) {
41      if( x[i] == '0' ) {
42        count++; printf("%5d",i);
43        if( count%10 == 0 ) { printf("¥n"); }
44      }
45    }
46    printf("¥n%d 以下の素数：%d 個 ¥n",n,count);
47  }
```

実行結果：方法２．１と同じ

実行時間（出力時間は除く）

```
実行時間測定
% ./a.out
100000   10の5乗
100000 以下の素数 9592個   生成時間0 ms
% ./a.out
1000000   10の6乗
1000000 以下の素数 78498個   生成時間10 ms
% ./a.out
10000000   10の7乗
10000000 以下の素数 664579個   生成時間220 ms
```

方法２．４　エラトステネスの方法の改良

方法２．３の手順（２）を改良する。

手順（０）2を素数とする。
手順（１）数列 {3,5,7,...,n}を用意する。
手順（２）数列の中の最小値を素数pとし、その数自身とその数の
倍数を数列から取り除き、新しい数列を構成する。
pが6k+1の形の素数とわかったとき、取り除く倍数を、
p*p,p*(p+4),p*(p+6),p*(p+10),p*(p+12),...とする。
pが6k+5の形の素数とわかったとき、取り除く倍数を、
p*p,p*(p+2),p*(p+6),p*(p+8),p*(p+12),...とする。
手順（３）p*p が n 以下の間、手順（２）を繰り返す。

●プログラム(prm241.c)

```
1  /* << prm241.c >> */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define N 10000000 /* 配列サイズの最大値。*/
5
6  int main() {
7
8    char x[N+1]; /* 数列を配列xで表現する。
9                    すなわち、数列の要素iは、x[i]='0'、
10                    数列の要素でなくなると、x[i]='1'とする。*/
11   int count,    /* 素数の個数。*/
12       dq,       /* 増分。*/
13       i,k,
14       n,        /* nの値。*/
15       p,        /* 素数の候補。*/
16       pq,       /* pの倍数。*/
17       q;
18
19   /* 正整数nの読み込み。*/
20   scanf("%d",&n);
21   if( n <= 0)||(n > N) ) { exit(0); }
22
23   /* 初期設定。iが素数の候補である間x[i]='0'。
24                 iが素数pの倍数となる場合x[i]='1'。*/
25   for( i=3; i<=n; i=i+2 ) { x[i] = '0'; }
26   p = 3;
27   /* 3の倍数を削除。*/
28   for( k=p*p; k<=n; k=k+2*p ) { x[k] = '1'; }
29   /* 素数候補pの初期値。*/
30   p = 5;
31
32   /* 素数の探索。*/
33   while( p*p <= n ) {
```

```
34      if( x[p] == '0' ) {
35        /* 増分dqの更新。*/
36        if( p%6 == 1 ) {
37          dq = 2;
38        } else {
39          dq = 4;
40        }
41        q = p; pq = p*q;
42        /* pの倍数pqを削除。*/
43        do {
44          x[pq] = '1';
45          dq = 6 - dq;
46          q = q + dq;
47          pq = p*q;
48        } while( pq <= n );
49      }
50      /* 素数候補pの更新。*/
51      p = p + 2;
52    }
53
54    /* x[i]='0' となる数 i は素数なので出力する。*/
55    count = 1;
56    printf("%5d",2);
57    for( i=3; i<=n; i=i+2 ) {
58      if( x[i] == '0' ) {
59        count++; printf("%5d",i);
60        if( count%10 == 0 ) { printf("¥n"); }
61      }
62    }
63    printf("¥n%d 以下の素数：%d 個 ¥n",n,count);
64  }
```

実行結果：方法２．１と同じ

実行時間（出力時間は除く）

```
実行時間測定
% ./a.out
100000   10の5乗
100000 以下の素数 9592個   生成時間0 ms
% ./a.out
1000000   10の6乗
1000000 以下の素数 78498個   生成時間10 ms
% ./a.out
10000000   10の7乗
10000000 以下の素数 664579個   生成時間180 ms
```