素数判定

N 以下の素数を全て求める処理を組む。

- 定義通りの処理
- エラトステネスのふるい
- 素数定理

プログラムの作成

N 以下の自然数 k について、k 以下の自然数を約数に持てば素数ではない。$m = 2$ から$\underline{k-1}$について、$mod(k, m)$ の値を調べれば k が素数かどうか判定できる。
プログラムでは、変数 isPrime が 1 か 0 かに応じて k が素数かどうかを判定している。このような変数をフラグと呼ぶ。
素数でないことがわかった段階で不要な j に関する処理を省略するため、break 文を使用した。break 文は、最も内側のループから抜け出す処理をする。

処理の流れ

1. $k = 2, 3$ は素数である。
2. $k = 4, 5, \ldots, N$ について、以下を繰り返す。
   - 2.1 $j = 2, 3, \ldots, k-1$ について、k を j で割った余りを見る。
   - 2.2 ある j について余りが 0 となれば、k は素数ではない。

プログラム

```
for k=[4:N]
  isPrime=1;
  for j=[2:____]
    if( mod(k,j) == 0)
      isPrime=0;
      break;
    endif
  endfor
  if ( isPrime == 1 )
    printf( "%d\n", k);
  endif
endfor
```

ふるい

定義の通りに書いたプログラムは、k を判定するために k 以下の自然数について約数かどうかを調べている。k 以下の素数について調べれば十分だから、効率を上げるために判定済みの素数を記録する。
素数かどうかを記録するために、N 個の要素をもつ配列 p を用意しよう。初期値として 1 を代入し、k が素数ではないとわかれば $p(k) = 0$ とする。

1. 配列 p([1:N]) を用意する。初期値として全ての要素に 1 を代入する。これは、1 から N までの各 k が素数であることを仮定する。
    1.1 各 m について、 m の倍数は素数ではない。m は素数として十分である。