レベル		Tr	Tc	Td
与えられたプログラムの構文が認識できる	1–1	プログラムの構文（文法）が自分にとって馴染みがないものとして、プログラムの各部の記述を読んでその意味（この場合は文法構造）を理解する。	プログラムの構文に従って、プログラムの各部の結び付きを見出す。たとえば、繰り返しの始まりと終わりなど。	
与えられたプログラムの動作をトレースできる	1–2	プログラムの構文が自分にとって馴染みがないものとして、プログラムの各部の記述を読んでその意味（この場合は操作的意味）を理解する。	プログラムの実行経路に従って、プログラムの各部の結び付きを見出す。つまり、実行順に命令を結び付ける。たとえば、条件分岐の条件の真偽に従って、次に実行される命令を見出す。	
与えられたプログラムの動作が説明できる	2–1		プログラムの正しさを説明する場合など、与えられた仕様（目的・意図）に照らしてプログラムの動作を説明する場合は、仕様の各部とプログラムの各部の結び付きを見出す。	プログラムの動作そのものを説明するだけでなく、プログラムの仕様（目的・意図）を説明する場合は、プログラムに直接に示されていない仕様を見出す。
与えられたプログラムを異なる目的に沿って修正できる	2–2		プログラムの中で、与えられた目的に関連する部分を見出す。	異なる目的を達成するために、関連する部分をどのように書き替えるべきかを見出す。

レベル2-2 プログラムを修正する

- 正の整数 data が与えられて，data が素数かどうかを判定するプログラムを理解できるか？
- ２以上の整数 data が与えられて，data を素因数分解する

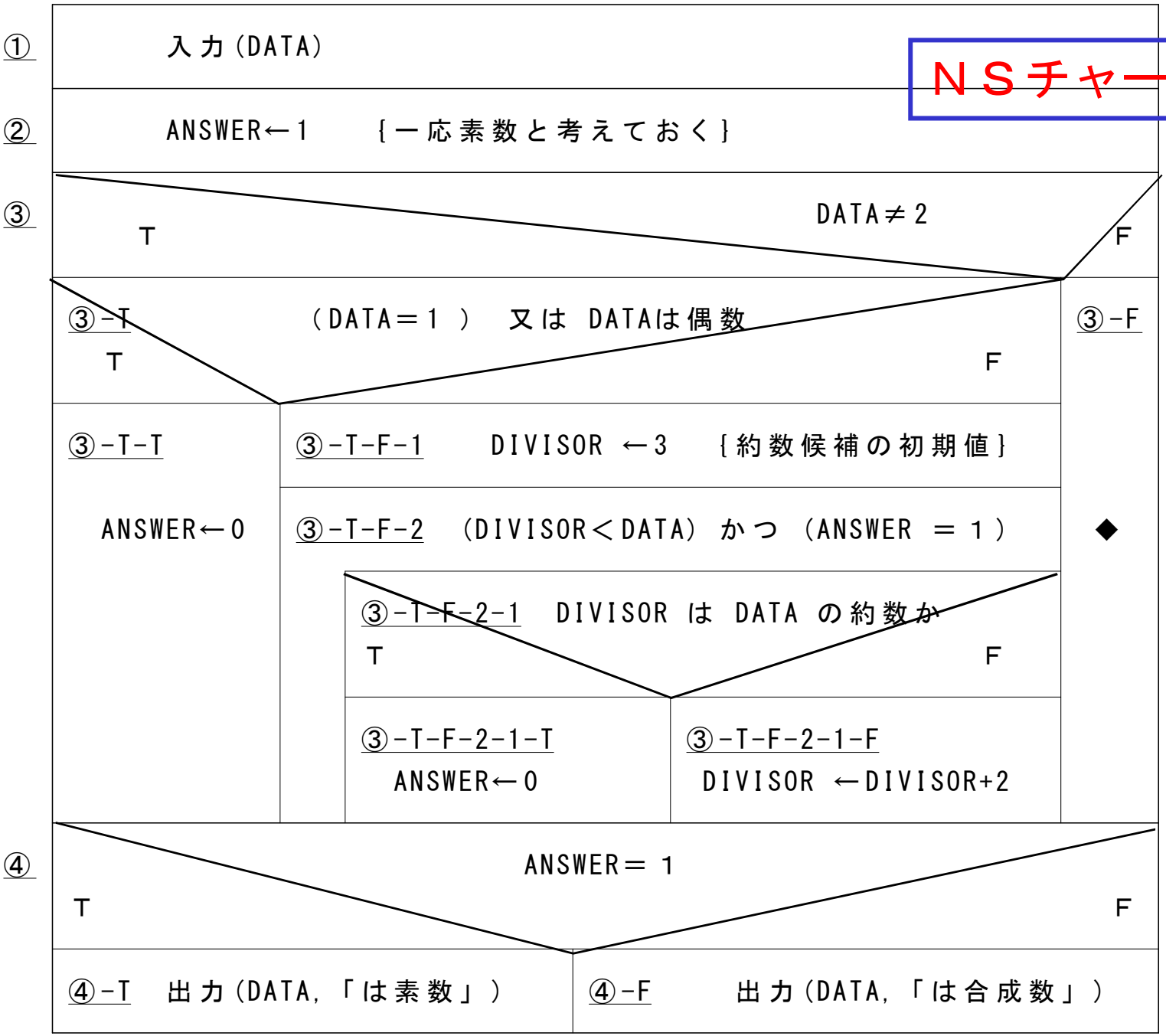
素数判定問題

- 入力:　1つの正整数 data
- 出力:　data が素数ならば「素数である」と素数でなければ「素数でない」

素数とは，1 より大きい自然数で，
正の約数が 1 と自分自身のみであるもののことである.
1は素数でないとする.

素数判定アルゴリズム

- 判定したいデータを入力し，それをdataとする
- 値dataをより小さい素数で順に割れるかどうか確認する
- data=2 → 素数である
- data=1　or　2以外の偶数 → 素数でない
- dataを 3 以上の素数で割り切れるかどうかをチェックする
  - 割り切れる素数がある → 素数でない
  - 割り切れる素数がない → 素数である


①
入力(DATA)
ＮＳチャート
②
ANSWER←1　{一応素数と考えておく}
③
T
DATA≠2
F
③-T
T
(DATA＝1)　又は　DATAは偶数
F
③-F
③-T-T
ANSWER←0
③-T-F-1　DIVISOR ←3　{約数候補の初期値}
③-T-F-2　(DIVISOR＜DATA) かつ (ANSWER ＝ 1)
◆
③-T-F-2-1　DIVISOR は DATA の約数か
T
F
③-T-F-2-1-T
ANSWER←0
③-T-F-2-1-F
DIVISOR ←DIVISOR+2
④
ANSWER＝ 1
T
F
④-T　出力(DATA,「は素数」)
④-F　出力(DATA,「は合成数」)

```
program CheckPrime1(input,output);
  var answer,data,divisor:integer;
  begin
    writeln('正の整数を入力してください');
    read(data);
    answer:=1;
    if data<>2 then
      if (data=1) or (data mod 2=0) then
        answer:=0
      else begin
        divisor:=3;
        while (divisor<data) and (answer=1) do
          if data mod divisor=0 then
            answer:=0
          else divisor:=divisor+2
        end;
    if answer=1 then writeln(data,'は素数である')
    else writeln(data,'は素数でない')
  end.
```

馴染みのない(？)
PASCAL風表現のプログラムが与えられ,
それを理解できるか？

```
program CheckPrime1(input,output);
  var answer,data,divisor:integer;
  begin
    writeln('正の整数を入力してください');
    read(data);
    answer:=1;
    if data<>2 then
      if (data=1) or (data mod 2=0) then
        answer:=0
      else begin
        divisor:=3;
        while (divisor<data) and (answer=1) do
          if data mod divisor=0 then
            answer:=0
          else divisor:=divisor+2
        end;
    if answer=1 then writeln(data,'は素数である')
    else writeln(data,'は素数でない')
  end.
```

構文構造が理解できるか？

```
program CheckPrime1(input,output);
  var answer,data,divisor:integer;
  begin
    writeln('正の整数を入力してください');
    read(data);
    answer:=1;
    if data<>2 then
      if (data=1) or (data mod 2=0) then
        answer:=0
      else begin
        divisor:=3;
        while (divisor<data) and (answer=1) do
          if data mod divisor=0 then
            answer:=0
          else divisor:=divisor+2
        end;
    if answer=1 then writeln(data,'は素数である')
    else writeln(data,'は素数でない')
  end.
```

変数 answer の役割を理解できるか？
0 と 1 が代入されている意味は？

```
program CheckPrime1(input,output);
  var answer,data,divisor:integer;
  begin
    writeln('正の整数を入力してください');
    read(data);
    answer:=1;
    if data<>2 then
      if (data=1) or (data mod 2=0) then
        answer:=0
      else begin
        divisor:=3;
        while (divisor<=data div 2) and (answer=1) do
          if data mod divisor=0 then
            answer:=0
          else divisor:=divisor+2
        end;
    if answer=1 then writeln(data,'は素数である')
    else writeln(data,'は素数でない')
  end.
```

与えられた尺度でより良いプログラムが書ける	4-2

与えられた尺度という基準において、候補となるプログラム（の構成要素）の中から上位ないし下位のものを選択する。

divisor<data

効率の悪いループを改善できるか？

$\sqrt{N}$ 以下の素因数がある

- 整数Nが合成数（素数でない）の場合
- １でない最小の素因数を p とする

$$N = p \times q \quad (p \leq q)$$

$$N = p \times q \geq p \times p$$

$$\sqrt{N} \geq p$$

```
program CheckPrime1(input,output);
  var answer,data,divisor:integer;
  begin
    writeln('正の整数を入力してください');
    read(data);
    answer:=1;
    if data<>2 then
      if (data=1) or (data mod 2=0) then
        answer:=0
      else begin
        divisor:=3;
        while (divisor<=sqrt(data)) and (answer=1) do
          if data mod divisor=0 then
            answer:=0
          else divisor:=divisor+2
        end;
    if answer=1 then writeln(data,'は素数である')
    else writeln(data,'は素数でない')
  end.
```

与えられた尺度でより良いプログラムが書ける	4-2

整数data に約数があるならば
data の平方根以下のものがある

```
program CheckPrime1(input,output);
  var answer,data,divisor:integer;
  begin
    writeln('正の整数を入力してください');
    read(data);
    answer:=1; {繰り返し制御のため}
    if data<>2 then
      if (data=1) or (data mod 2=0) then
        answer:=0
      else begin
        divisor:=3;
        while (divisor*divisor<=data) and (answer=1) do
          if data mod divisor=0 then
            answer:=0
          else divisor:=divisor+2
        end;
    if answer=1 then writeln(data,'は素数である')
    else writeln(data,'は素数でない')
  end.
```

標準関数を使わずにできることもある

(divisor<=sqrt(data))

素因数分解

- 任意の正の整数値が一つ与えられて，その値の素因数分解を求めるプログラムを書きなさい.
- 出力例 376=2*3**3*7
- 素数判定プログラムを参考にする

		Td
与えられたプログラムを異なる目的に沿って修正できる	2-2	異なる目的を達成するために、関連する部分をどのように書き替えるべきかを見出す。

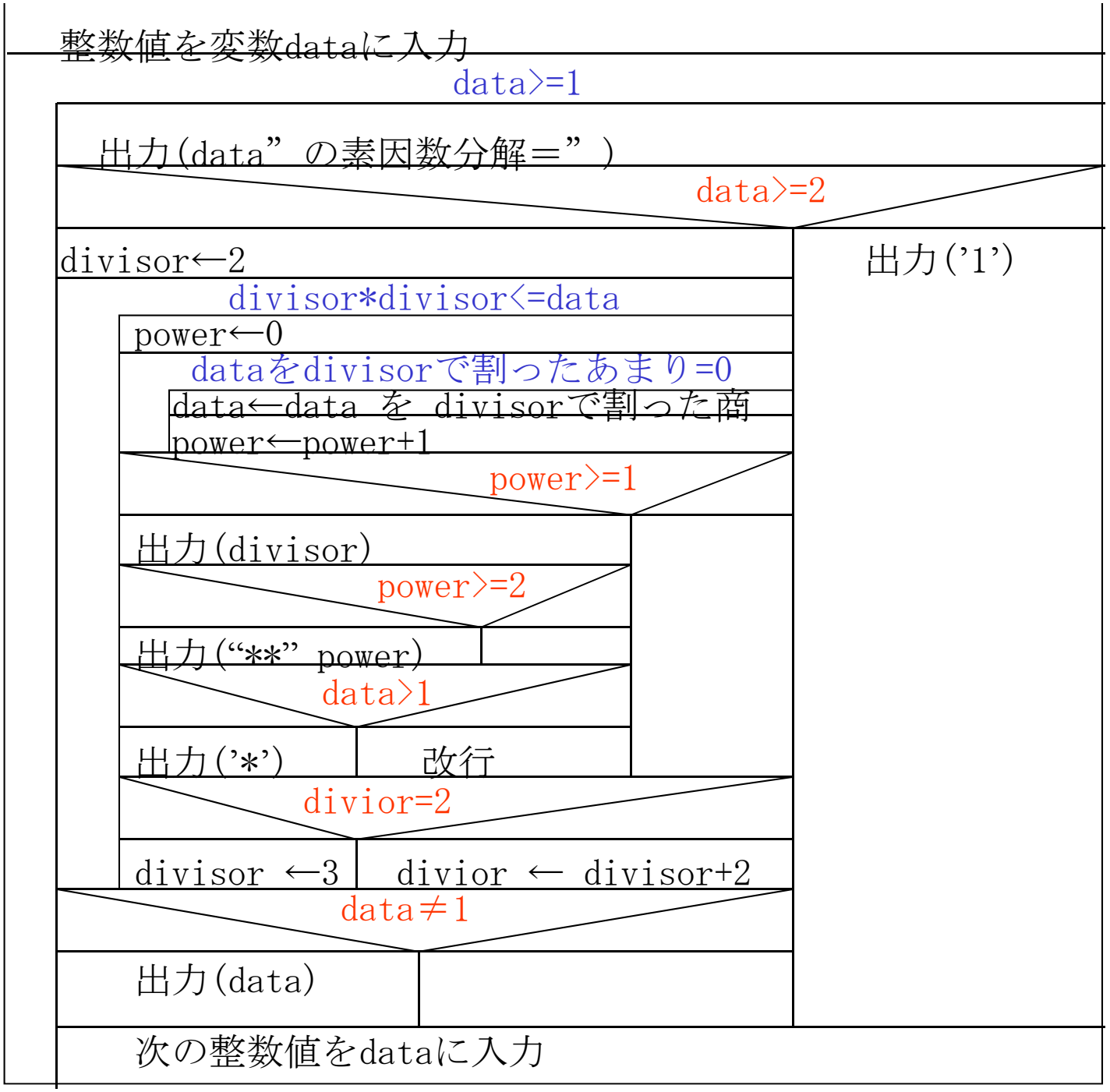
素因数分解

- 例えば，247401の素因数分解はどうなるか？
- 小さい素数から順に割っていく
- 求め方は知っているはず！
- それをプログラムとして書けるか？

3	247401
3	82467
3	27489
7	9163
7	1309
11	187
	17

247401＝
3×3×3×7×7×11×17

NSチャート
素因数分解



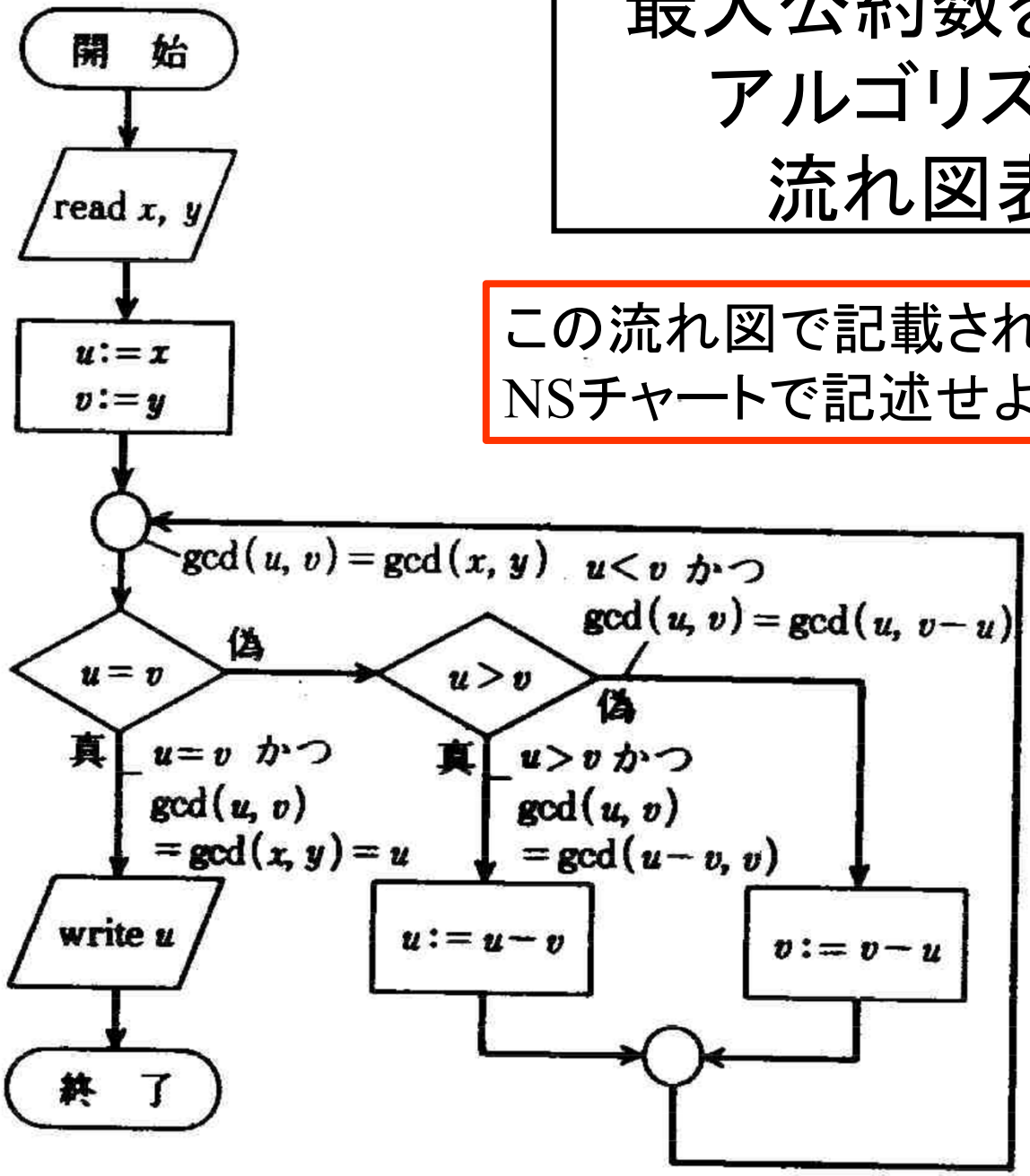
```
program PrimeFactorization (input,output);
  var data,divisor,power:integer;
  begin
    writeln('正の整数を入力してください');
    read(data); write(data,' =' ); divisor:=2;
    while divisor*divisor<=data do begin
      power:=0;
      while data mod divisor=0 do begin
        data:=data div divisor; power:=power+1
      end;
      if power>=1 then begin
        write(divisor);
        if power>=2 then write(' **' ,power);
        if data>1 then write(' *' );
      end;
      if divisor=2 then divisor:=3
      else divisor:=divisor+2
   end;
   if data<>1 then writeln(data)
  end.
```

2-1 与えられたアルゴリズムの動作を説明できる

- あるスタイルのアルゴリズム記述を，他のスタイルのアルゴリズム記述に書き直せるか.
  - フローチャート表現 → NSチャート表現

最大公約数を求めるアルゴリズムの流れ図表示

この流れ図で記載されたアルゴリズムを
NSチャートで記述せよ



最大公約数を求めるアルゴリズムのNSチャート表示

1973年頃のI. Nassi と
B. Shneidermanによる考えを
N. Chapinが整理した
図式（チャート）

構造化プログラミングの3つの
基本形（連接，選択，反復）を
明確に表現する