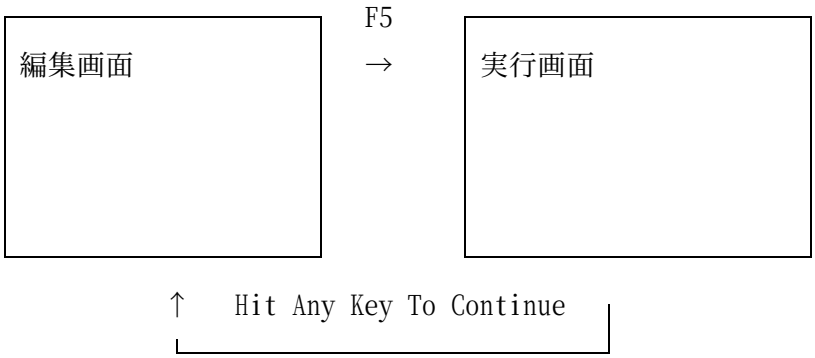
## ２－０　はじめに

今回は三回目。「まだ三回目」とも言えますが，三回目だって，プログラミング言語にピッタリあっていることなら，数学的なことを調べることもできます。今回は，整数のある話題に注目することにします。

また，前回の「ＫＥＹ－ＧＡＭＥ．ＢＡＳ」の「単語編」が「ＫＥＹ－２．ＢＡＳ」です。ここで入力する単語は，ＱＢＡＳＩＣのキーワードです。

まず，これを使って，指ならしをしてみましょう。今後も，折りにふれてトレーニングをしてみてください。

先週および，今の作業から推察できると思いますが，QBでは，次の二つの世界の使い分けをしています。




プログラムの入力および，編集作業をしてみて，これでいいかなと思ったら，それを「実行」してみる。そして，その結果を元にして，元の編集画面の中で再編集をする。

そういう作業の繰り返しです。

編集用のキーとして，次のものをよく使います。

| | |
|---|---|
| Delete | カーソルの部分の文字を消去する。 |
| BackSpace | カーソルの前の部分の文字を消去する。 |
| Enter | その部分を改行する。冒頭であれば，新しい行を加える。 |
| Insert | 挿入モードと上書モードの切り替え |

また，先週は FOR　文を使いましたが，リファレンスマニュアルを調べると，次のページのような記述があります。このような記述を読んだり，実際にサンプルプログラムを入力し，実行しながら，その意味を理解していくというのことが，プログラミングを覚えていくための一つの方法です。

このような日本語のものが欲しい人は，本屋さんに行って，Quick BASIC 関係の本を物色してみてください。また，たしかリファレンスマニュアル自体も翔泳社から出ていたように思います。

また，授業中などに，QBASICを使っているときに，調べたいと思うのも不思議なことではありません。そういうときには，該当する文字のところに，カーソルを合わせて（点滅する場所を合わせて），F1 キーを押してみましょう。すると，FOR に関する画面が出てきます。英語なので，使いにくいのは確かかもしれませんが，こういう形でマニュアルを参照することも可能です。

ちなみに，このようなマニュアルをオンラインマニュアルとか，オンラインヘルプと言います。また，日本語版のQuick BASIC をはじめ，各種の言語では，日本語でそのようなものが表示されるようになっています。

## ２－１　「入出力」という概念　－　INPUT と PRINT　－

プログラムあるいは，コンピュータというものは，勝手になにかをしてくれるものではありません。命令があれば，それを忠実に実行するというだけの存在です。そういうものとしてのプログラムの最も基本的な構成は，

入力→処理→出力

です。何が入力であり，何が出力なのかを理解すること，またそれを考えて設計することが最も基本的です。

もちろん，中には「入力がない」プログラムもあります。例えば，

```
FOR i = 1 to 10
    PRINT i;
NEXT
```

というプログラムでは，利用者が入力すべきものはありません。しかし，多くの場合，そういうプログラムは，あまり価値がありません。あるいは，プログラムの一部を変えながら使うことを想定しているものが多いと言ってもいいでしょう。例えば，このプログラムの場合，入力を明確化すると，次のようなプログラムになります。

```
INPUT LastNumber
FOR i = 1 to LastNumber
    PRINT i;
NEXT
```

このように，プログラム自身の中では明確になっていないとしても，何が入力であり，何が出力になっているのかを考えていく習慣をつけましょう。

## ２－２　代入としての式　－　総和を求める問題

入力から出力を作る「処理」において，最も基本的な役割を果たすのは様々な「式」です。それは，数学での計算が果たす役割と一緒です。しかし，数学での「式」の意味と，プログラムにおける「式」の意味は異なる部分がかなりあります。例えば，次の式を見てみましょう。

$$x = x + 1$$

これは数学的には「解なしの方程式」でしかありません。いわばナンセンスです。しかし，プログラムの中ではよく使われます。この意味は，

「ｘの値に１を加えたものを新たにｘの値とする」

ということです。よりコンピュータに密接に述べると，

「変数ｘという番地に記憶されている数値に１を加え，これを変数ｘの番地に書き込め」

ということです。つまり，＝の役割は，「等式」としての役割ではなく，「代入」ということです。また，ここで注意しておきたいのは，

「プログラムにおける『変数』というのは，数値が記憶される『箱』を指す」

ということです。

このことを念頭に置いて，次のプログラムの意味を考えてみましょう。

(1)

```
FOR i = 1 to 100
     sum = sum + i
NEXT
PRINT sum
```

(2)

```
FOR i = 1 to 100
     sum = sum + (i * i + 1)
NEXT
PRINT sum
```

(3)

```
FOR i = 1 to 201 STEP 2
     sum = sum + i
NEXT
PRINT sum
```

練習問題

(1) N を入力し，1 からN までの和を求めるプログラムを作れ。
その結果を下に表にせよ。

| N | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N までのk の和 | | | | | | | | | | | | | | |

(2) 1 からN までの和を求めるとき，初めて20000 を越えるのはN がいくつのときか

(3) 1 からN 個の奇数の和を求めるプログラムを作れ。
いろいろと調べて気がつくことはないか。

(4) 1 からN 個の$k^2$ の和を求めよ。
その結果を下に表にせよ。

| N | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N までの$k^2$ の和 | | | | | | | | | | | | | | |

(5) 1 からN 個の$k^3$ の和を求めよ。その結果を下に表にせよ。

| N | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N までの$k^3$ の和 | | | | | | | | | | | | | | |

何か気がつくことはあるか。

## ２－３　条件の判定としての式　－　IF THEN ～　と IF THEN　～ ELSE ～　－

もう一つの「式」の役割は，条件の明示とする判定である。そして，多くの場合，それはIF文との組み合わせで使われることが多い。例えば，

(1)

```
DO
    INPUT n
    IF n > 10 THEN PRINT n; "is big number."
LOOP until n = 0
```

(2)

```
DO
    INPUT n
    IF n > 10 THEN PRINT n; "is big number."ELSE PRINT n; "is small."
LOOP until n = 0
```

## ２－４　IF　ブロック

　また，プログラムを見やすくしたい場合や，処理する内容が多い場合には，次のようなIF　ブロックを使います。

例(3)

```
DO
    INPUT n
    IF n > 10 THEN
         PRINT n; "is big number."
    ELSE
         PRINT n; "is small."
    END IF
LOOP until n = 0
```

　ここで特徴的なのは，IFから始まるブロックの最後を ENDIF　で明示する部分ですが，このようなEND***という使い方は他でもよく使います。

## ２－５　これは何でしょう。

次のプログラムは何をするプログラムでしょう。

例(4)

```
INPUT n
FOR i = 2 to n-1
    IF (n MOD i) = 0 THEN
        check = -1
        EXIT FOR
    END IF
NEXT
IF check = -1 THEN
    PRINT n ; "is not prime number."
ELSE
    PRINT n ; "is a prime number."
END IF
```

例(5)　例(4)を少し修正すると，

```
DO
    INPUT n
    check = 0
    IF n = 0 then EXIT DO
    FOR i = 2 to n-1
```

```
        IF  (n MOD i) = 0 THEN
            check = -1
            EXIT FOR
        END IF
    NEXT
    IF check = -1 THEN
        PRINT n ; ”is not prime number.”
    ELSE
        PRINT n ; ”is a prime number.”
    END IF
LOOP
```

になります。これを使って，できるだけ多くの，そしてできるだけ大きなprime numberを見つけましょう。

prime number

(3) (2)を更に修正したいと思います。どんな観点から，どのように修正することができるでしょうか。また，これらを使うと，どんなことを調べることでできるでしょうか。

## ２－６　SELECT　文

IF文は，一つの条件を元に，二つの分岐をしますが，一つの式の結果から，いろいろな分岐をする場合には，SELECT　文を使う方が適してきます。

(1) 例

```
INPUT n
FOR i = 1 to n
    IF n MOD i = 0 then Yakusuu = Yakusuu + 1
NEXT
SELECT CASE Yakusuu
    CASE 1
        PRINT "n = 1, isn't it?"
    CASE 2
        PRINT n;"is a prime number"
    CASE ELSE
        PRINT n;"is not a prime number"
END SELECT
```

SELECT　文にはもう少し応用した使い方もありますが，必要なときに紹介することにし，ここでは上記の最も簡単な使い方を理解しておいてください。

## ２－７　問題

(1) n を入力し，n の約数をすべて表示するプログラムを作れ。

(2) 約数の個数が３個になる数を５つ見つけよ。

(3) 約数の個数が奇数個になる数と偶数個になる数にはどういう違いがあるか。