応用数学　　1998.5.7/8
３．判断・分岐・使いやすくするための工夫　　飯島

３－０　はじめに

前回のテキストの最後の方は消化できなく，「次回持ってきてね」ということにしましたが，忘れた人もいるかもしれないし，また授業内容としては，それ以前にしておく方がいいこともあるので，前回の内容も，以下に再録することにしました。その代わり，今回の内容はおそらく全部は消化できません。そこは「自習してね」ということで。

３－１　マウスの利用

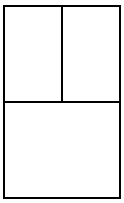
ＱＢＡＳＩＣでは，基本的にキーボードだけですべての作業をすることができますが，慣れるまでは，どのキーを押すのか，あるいはキーがどこにあるのか，戸惑うことがよくあると思います。たとえば，ファイルを読み込むときには，Alt キーと矢印キーのEnterキーの他にTAB キーが不可欠ですが，最初は戸惑います。

そういうのを解消するための手段としては，「マウス」を使う手があります。

なお，マウスは，手の動きをそのまま反映するので，使いやすい面がある一方，マウスの動きを目で追い，手の微妙な動きが必要になるので，熟練者にとっては，かえって「アバウトなキー操作をしても一定の同じ動作をしてくれる」キーボードの方が使いやすく疲れない面もあります。

ＱＢＡＳＩＣ（やWindows 3.1)では，使うのはほとんど左ボタンだけです。（なお，Windows95 では，右ボタンを使うこともあります。）

・クリック　　：ボタンを押すのをクリックと言います。
・ダブルクリック：ボタンを続けて２回押すのをダブルクリックと言います。
　「ファイルを選択する」ときなど，これを使います。
　続けて早く押さないと「クリック２回」と認識してしまいます。
　うまくできないときは，「ＯＫ」を押すようにしましょう。

左ボタン  右ボタン

３－２　「暴走」を止めるには

プログラムを実行させてしまうと，「止めたくなる」ことがよくあります。たとえば，

```
FOR i = 1 to 10000
    PRINT i;
NEXT
```

のようなプログラムもそうですし，また，

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

のようなプログラムでは，つい入力のときに，100000なんて入力してしまうと，

「止まらないよう」

ということになります。あるいは，「12345678の約数をすべて求めたい」というようなとき，アルゴリズムがヘタクソだと，時間がやたらかかります。

そういうときは，最後までさせるよりも，途中で停止し，考え直せる方がいいわけです。さて，そういうときには，

Ctrl + Pause　（CtrlキーとPause キーを「一緒に」押す）

で停止します。キー入力を要求するような場合には，これらを押した後，何らかのキーを押すと停止することもあります。

（Pause キーのみでも，一見停止しますが，何か別のキーを触るとそのまま再開します。Ctrlキーを一緒に押すことをおわすれなく。）

またＦ５を押すと，「中断箇所からの再開」になります。

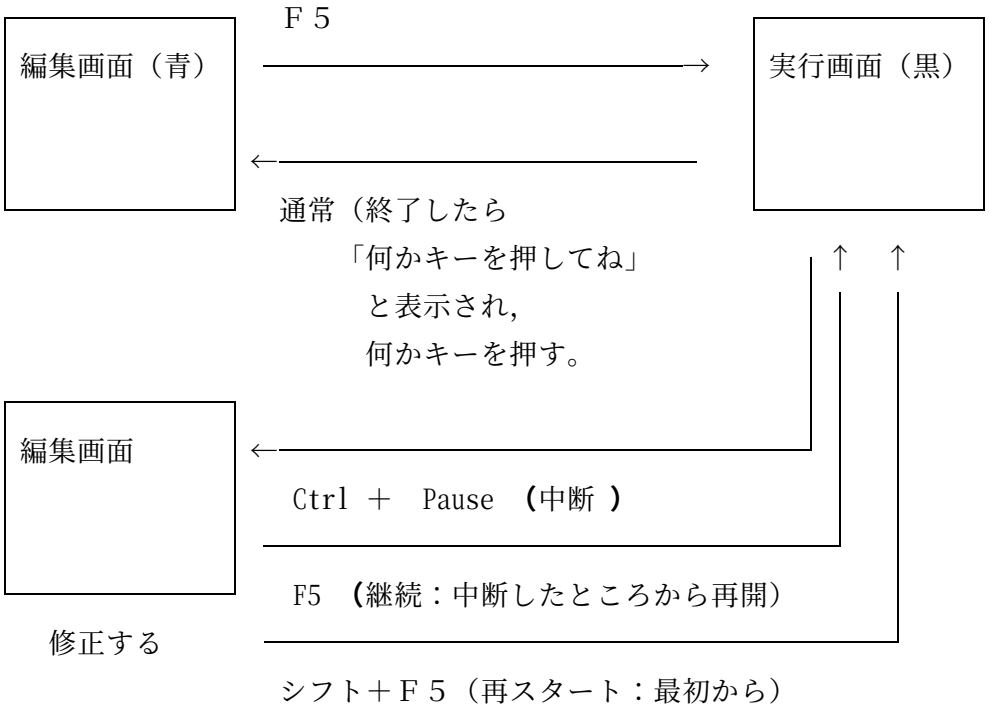
修正等をして，最初からスタートしなおしたいときには，「シフト＋Ｆ５」です。

また，このようなキー操作を覚えるのはちょっとね，という人は，メニューの中で操作する，つまり，

「Alt キー」→「RUN 」→「Start 」，「Contiune」などを選択，

という手もあります。

図示すると，



という具合になります。

３－３　条件の判定としての式　－　IF THEN ～　と IF THEN　～ ELSE ～　－
（再録）

「式」のもう一つの役割は，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，

例(1)
```
DO
    INPUT n
    IF n > 10 THEN PRINT n; ”is big number.”
LOOP until n = 0
```

例(2)
```
DO
    INPUT n
    IF n > 10 THEN PRINT n; ”is big number.”ELSE PRINT n; ”is small.”
LOOP until n = 0
```

３－４　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                                    意味
    IF n > 10 THEN                             もしｎ＞０だったら
        PRINT n; ”is big number.”                  ＊＊＊をしろ
    ELSE                                       そうでないときは
        PRINT n; ”is small.”                       ○○○をしろ
    END IF                                     以上。
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。

３－５　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。（入力と結果から予想しましょう）

例(1)
```
INPUT n
FOR i = 2 to n-1
    IF (n MOD i) = 0 THEN                      n を　i で割った余り
        check = -1
        EXIT FOR                               for nextから抜け出る
```

```
    END IF
NEXT
IF check = -1 THEN
    PRINT n ; ”is not prime number.”
ELSE
    PRINT n ; ”is a prime number.”
END IF
```

(2) (1)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

３－６　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。

３－７　使いやすいプログラムへの修正

「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」という問題を先週したときに，授業の中では，

```
INPUT N
FOR i = 1 to N
     sum = sum + i
NEXT
PRINT sum
```

というようなプログラムを使いました。

「毎回 F5 を押すのは面倒くさい。何とかしてよ。」

というような声がありましたが，そういう気持ちになることは，使いやすいプログラムを作る上で，とても大切です。たとえば，

「いくつも続けて入力できるようにする」

には，たとえば，

```
DO
  INPUT N
  FOR i = 1 to N
       sum = sum + i
  NEXT
  PRINT sum
LOOP
```

とする手があります。しかし，実は，このプログラムには，二つの問題点があります。

(1) ２回目以降，和が正しくない。

これを修正するためには，次の部分にある一行を追加する必要があります。

```
DO
  INPUT N
  *******                              ←　自分で考えてね
  FOR i = 1 to N
       sum = sum + i
  NEXT
  PRINT sum
LOOP
```

(2) 終わらない。

終わりにするための条件を作っておかないと，こういうことになります。たとえば「０を入力したらおしまい」とするためには，

```
DO
  INPUT N
  IF N = 0 THEN EXIT DO
  *******
  FOR i = 1 to N
       sum = sum + i
  NEXT
  PRINT sum
LOOP
```

とすればＯＫです。また，その条件を分かるようにするには，

```
DO
  INPUT "0 -> end :";N                    -> で，矢印「→」のつもり
  IF N = 0 THEN EXIT DO                   「0 だったらおしまい」のつもり
  *******
  FOR i = 1 to N
       sum = sum + i
  NEXT
  PRINT sum
LOOP
```

のように，入力する人用のメッセージを書くといいでしょう。

３－８　約数の個数を調べる

前回，素数について調べました。素数とは，約数の個数が２個の数です。そこで，内容を一般化し，約数の個数を調べるプログラムを作りましょう。

日本語で，処理の概略を書くと，次のようになります。

> 数Ｎを入力する。
> １～Ｎまでの数ｉについて，
> 　もし，ｉがＮの約数ならば，「約数の個数カウンター」を１増やす
> 「約数の個数」を表示する。

これを，QBASICで書いてみましょう。

```
INPUT N
FOR **********                              (*******) の部分は自分でね。
    IF ********** THEN Yaku = Yaku + 1
NEXT
PRINT N ; Yaku
```

３－９　使いやすくするための修正（１）

このプログラムは１つの数しか受け付けないので，いくつも続けて入力できるようにしましょう。そして，そのときに，０を入れたら，終了するようにしましょう。

```
DO

    Yaku = 0          ←　これを挿入することを忘れずに。



LOOP
```

３－１０　予想

ただ計算するだけではあまり面白くないので，予想を立てて，それを検証してみましょう。

	予想		実際	
	最大の約数の個数	最大になる数	最大の約数の個数	最大になる数
～１００				
～２００				
～５００				
～１０００				
～１００００				

３－１１　使いやすくするための修正（２）

いくつも続けて入力できるのはいいですが，１～１００までの数を毎回入力するというのは，面倒ですね。そう，そういう時は，ＤＯ～ＬＯＯＰでなく，ＦＯＲ～ＮＥＸＴ！

```
INPUT LAST
FOR N = 1 TO LAST


    Yaku = 0          ←　これはそのまま必要




PRINT N ; Yaku ;     ←　こうすると，表示はどうなる？
NEXT
```

３－１２　もっと使いやすくするための修正（３）

目的に応じて，便利にできるのが，プログラムです。必要な情報は提示し，不要な情報は提示しないというのが，その基本です。たとえば，

「約数」は，途中でチェックはしていても，表示しない情報の一つです。

「それぞれのＮに対する，約数の個数」というのも，一見必要に見えますが，上記の問題では，「１～＊＊＊の中の最大値」のみを調べようということですから，必ずしも必要ではありません。最も欲しい情報を表示するために，

```
  MaxYaku  :  Yaku  の最大値
```

という変数を作りましょう。そして，

```
  PRINT N ; Yaku
```

の代わりに，

```
  IF MaxYaku =< Yaku THEN
     PRINT N ; Yaku
     MaxYaku = Yaku
  END IF
```

というブロックを書きましょう。

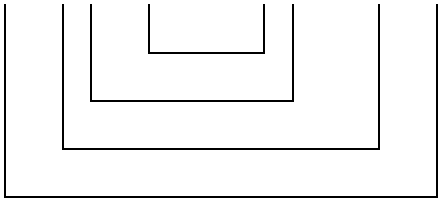
３－１３　もし計算の実行に時間がかかったら

使うパソコンによっても必要な時間は変わりますが，アルゴリズムに問題があると，そういうこと以上に大きな影響が生じてしまいます。調べたい数値に対して，あまりに時間がかかるようであれば，アルゴリズムをもう少し工夫することを検討してもいいでしょう。たとえば，Ｎ／２より大きいところにある約数はＮだけですから，

```
  FOR I = 1 TO INT (N/2)
```

に変えてもいいでしょう。ただし，その前に，

Yaku = 1
としておく必要があります。
あまり効果はないですが,
Yaku = 2
FOR I = 2 TO INT (N/2)
としてもいいですね。
　さらに，もっと工夫することを考えると，たとえば,
100 の約数 : 1，2，4，5，10，20，25，50，100
であって，これらは,
100 の約数 : 1，2，4，5，10，20，25，50，100



という関係であることを考えると，√N までで計算を終了することもできます。
ただ，この場合，少し工夫しないと，N が小さいときに，ミスが出るので，工夫が必要ですが。
　余裕があったら，その観点で修正してみてください。
　N/2 は計算の量を半分にしてくれますが，√N の場合，計算の量を「桁を半分」にしてくれ，とっても効果的です。

３－１４　もう一つの方法

そうそう，整数しか扱わない場合，計算を早くするための，もう一つの別の「おまじない」があります。それは，冒頭に,
DEFINT A-Z
と書く「おまじない」です。ただし，これを書いたら，約30000 までの整数しか使ってはいけません。もし，もっと大きな数（桁数２倍まで）を扱いたければ,
DEFLNG A-Z
としてみてください。INT ほどの速度は出ませんが。