# プログラミング言語
# FORTRAN 入門

東海大学総合情報センター

（第３版）

# 目次


第１章　FORTRAN 言語とは …… 1

第２章　簡単な FORTRAN プログラム …… 2
２．１　プログラムの記入方法 …… 2
２．２　プログラムの説明 …… 3
２．３　簡単な入出力 …… 5

第３章　入出力 …… 6
３．１　入力 …… 6
３．２　出力 …… 6
３．３　入出力を使った例題 …… 7
３．４　入力の繰り返し …… 8

第４章　実習方法 …… １０
４．１　パーソナルコンピュータの起動 …… １０
４．２　ログイン …… １０
４．３　ソースプログラムの作成（修正) …… １１
４．４　コンパイルとリンク …… １２
４．５　実行 …… １３
４．６　ログアウト …… １３
４．７　パーソナルコンピュータの終了 …… １４

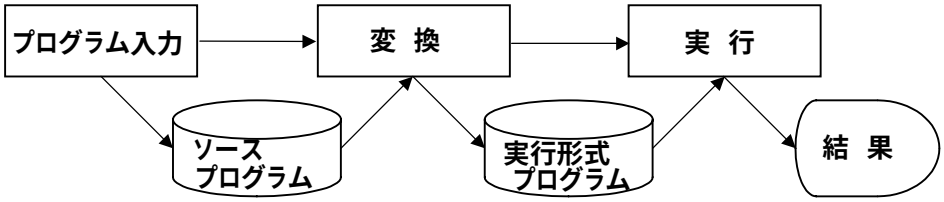
第５章　繰り返し …… １５
５．１　DO 文 …… １５
５．２　DO 文を使った例題 …… １６

第６章　組み込み関数 …… ２０
６．１　組み込み関数の例題 …… ２０

第７章　ブロック IF 文 …… ２２
７．１　ブロック IF 文の例題 …… ２３

# ■ 第1章 FORTRAN 言語とは

コンピュータにある計算をさせる場合、コンピュータの言葉（機械語）でプログラミングを行い、それをコンピュータに実行させる必要があります。しかし、機械語は2進数（0と1の組み合わせ）で表される数字の列であり、人間にとっては非常にわかりにくいものです。そこで、人間の言葉に近い言葉（プログラミング言語）でプログラミングを行い、それを機械語に変換してから実行する方法が考えられました。




FORTRAN(FORmula TRANslator)言語は、1956 年 IBM 社によって開発されたプログラミング言語であり、主に数値計算を目的として開発されました。その後、言語の文法などの規格が統一されながら FORTRAN IV や FORTRAN77 となってきています。最近では Fortran90 が制定され、利用できるようになりつつあります。

このテキストでは FORTRAN77 を対象として解説します。

FORTRAN は、数値計算や科学技術計算の分野でよく使われるプログラミング言語であり、特にスーパーコンピュータなどでは、生成される実行プログラムはより高速に計算できることを目的として最適化されるようになっています。

FORTARN 言語は高級言語と呼ばれており、人間が見て理解しやすいように IF, DO などのような予約語を使ってプログラミングを行います。その他にも高級言語と呼ばれているものとして以下のものがあります。

| | |
|---|---|
| C | オペレーティングシステム UNIX を開発するために作成された言語であり、機械語と同じようなコンピュータに近い処理から FORTRAN のような数値計算まで幅広く利用することができる。 |
| PASCAL | N. ヴィルトによって開発された言語で、教育用に適しています。<br>ブロック構造や再帰呼出しなどの機能があり、構造化プログラミングに適しています。 |
| COBOL | 事務計算や多種大量の帳票作成など大量のデータを扱うのに適した言語です。 |
| PL/I | 数値計算と事務計算のどちらでも効率よく開発できるように開発された言語ですが、コンパイラなどの処理系が大きくなっています。 |
| BASIC | 初心者向きのプログラミング言語です。インタプリタ上で開発することができます。 |
| Lisp | 関数型の言語で、リスト処理に適しています。人工知能の分野でよく使われています。 |
| PROLOG | Lisp と同様に人工知能の分野で使用されている言語で、「述語論理」を記述することによってプログラムを記述します。 |

# ■ 第2章　簡単な FORTRAN プログラム

ここでは簡単な FORTRAN プログラムを解説しながら、FORTRAN とはどのようなものかを体験してみます。

**例題１**　商品の単価と数量から金額を計算するプログラムを以下に示します。

(プログラム例)

```
1     6 7
*       EXAMPLE 1
        PROGRAM KINGAK
        INTEGER I, J, K
        I = 150
        J = 24
        K = I * J
        WRITE( *, * )K
        STOP
        END
```

(出力例)

```
 3600
```

単価が 150 円のものを 24 個購入するときの金額は 3600 円となることがわかります。

# ■ ２．１ プログラムの記入方法

FORTRAN プログラムを入力するときには、FORTRAN の文法に従って記述する必要があります。

FORTRAN プログラムは各行に記述しますが、桁位置に意味を持っています。

| | |
|---|---|
| 1桁目 | * または C を記述するとその行は注釈行です。 |
| 1桁目から5桁目 | 文番号を記述します。 |
| 6桁目 | 空白以外の文字を記述すると前の行からの継続となります。 |
| 7桁目から72桁目 | FORTRAN プログラムの文を記述します。 |

したがって例題のプログラムでは、

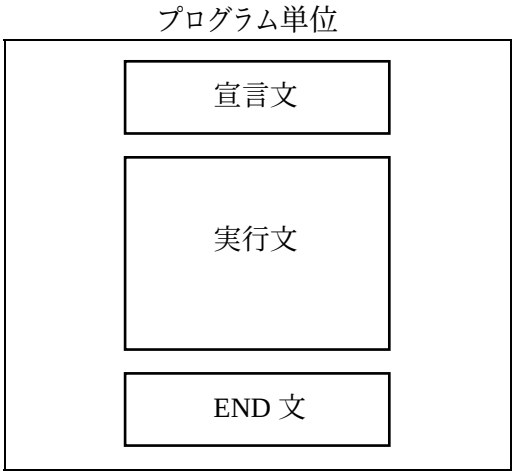
```
*       EXAMPLE 1
```

は1桁目に * が記述されており、注釈行を表しています。また、

```
        PROGRAM KINGAK
```

は7桁目から記述されており、FORTRAN プログラムの文を示しています。

使用する文字は英字を使用しますが、大文字でも小文字でもかまいません。また、FORTRAN プログラムは、プログラム単位で構成されていなければならず、プログラム単位は次のようになっていなければなりません。




## ■ ２．２ プログラムの説明

ここでは、2．1で例示したプログラムの各行について説明します。

**（１） PROGRAM KINGAK**

PROGRAM 文はプログラムに付ける名前を宣言します。主プログラムの先頭に書かなければなりません。また、KINGAK はプログラム名であり、6文字以内の英数字で指定します。

**（２） INTEGER I,J,K**

コンピュータ内では計算した値を一時的に記憶しておく入れ物が必要になりますが、この入れ物のことを「変数」と呼びます。また、コンピュータ内では値として、小数点のついた値（123.5 など）と小数点のつかない値（123 など）がありますが、それぞれ実数型定数、整数型定数と呼ばれています。変数も値の種類により入れ物の形が決められていますので、型（実数型変数、整数型変数）を決める必要があります。

変数の型を決めるには以下のように型宣言文を用います。

**INTEGER I,J,K**

また、FORTRAN プログラムでは変数を区別するために名前を付けて使用しますが、名前の命名には以下の規則があります。

・英字で始まる英数字の文字列

・文字列の長さは6文字以内

型宣言文では、プログラム中で使用する変数の型を指定しています。型宣言文 INTEGER で宣言された変数は整数型変数と呼ばれ、12 などの整数の値を記憶するための入れ物として使用されます。ちなみに、小数点の付く実数型（12.5 など）の値を記憶するための変数を宣言するには、以下のように REAL を使用します。

**REAL A,B,WA,SA**

FORTRAN プログラムではプログラム中に必要となる変数は、あらかじめ宣言しておくことによりプログラムミスを少なくできますので、プログラムの先頭には変数の型宣言文を記述するようにします。

**（３） I = 150**

I=150 は代入文と呼ばれ、＝記号の右辺の計算結果を左辺の変数に代入します。この代入

文によって変数 I のなかに値 150 が格納されます。右辺で使用されている数値 150 は定数と呼ばれ、以下のような記述方法があります。

整数型定数　　123

　　35

実数型定数　　123.5

　　1235E-1　(1235×$10^{-1}$)

**J = 24**

**K = I * J**

も同様に、代入文です。それぞれの代入文が実行されると、変数 J には 24 が、変数 K には I と J との乗算（FORTRAN では演算子＊は乗算を表します）の結果が格納されます。

右辺の式は算術式と呼ばれ、通常の計算式と同じように記述することができます。

**A + B,　A - B,　A * B,　A / B,　A ** 2**　（演算子）

*****は乗算を、**/**は除算を、******はベキ乗を表す演算子です。

**A - B * C,　(A - B) * C**　（演算子の優先順位）

### （4） WRITE(*,*) K

WRITE 文はコンピュータ内の変数の値を人間が確認できるように用紙や画面に出力するために使います。例題1のプログラムでは、計算結果の変数 K の値を表示します。

もう少し出力結果を工夫してみます。以下のように値の前に文字を表示してみます。

（出力例）

```
TANKA=150      KAZU=24      KINGAKU=3600
```

プログラムは以下のようになります。

```
WRITE(*,*)'TANKA=',I,'   KAZU=',J,'   KINGAKU=',K
```

### （5） STOP

STOP 文は実行を終了するための文であり、この文を実行するとプログラムは終了します。一つのプログラム中に一つは必要になります。

### （6） END

END 文はプログラムの記述の終わりを示す文であり、プログラム単位で必要になります。

**演習問題１**　次の式を FORTRAN で表しなさい。

1. A×(B+C)
2. $AX^3+BX^2+CX+D$
3. 2×3.14×R
4. $3.14\times R^2$
5. $\dfrac{A(B+C)^2}{D}$

## ■ ２．３ 簡単な入出力

例題1のプログラムでは、常に同じ値を計算するだけであり、他の値について計算することができません。このような場合、値をデータとして入力することができます。計算結果を表示するためにWRITE 文を用いましたが、データを入力するためにはREAD 文を用います。

**例題２**　商品の単価と数量を入力し、金額を計算するプログラムを以下に示します。

（プログラム例）

```
*       EXAMPLE 2
        PROGRAM KINGAK
        INTEGER I, J, K
        READ( *, * ) I, J
        K = I * J
        WRITE( *, * ) 'TANKA=', I, '   KAZU=',J, '   KINGAKU=',K
        STOP
        END
```

このプログラムをコンパイル、リンクした後、実行すると最初に入力を要求します。しかし、画面上には何も表示されませんが、そのまま入力します。

（入力例）

```
150  24
```

（出力例）

```
TANKA=150      KAZU=24      KINGAKU=3600
```

入力データが複数ある時には空白で区切りながら入力します。また、入力データと変数の関係は左から順に対応づけられていきます。つまり、

150 はI に

24 はJ に

読み込まれます。その後は例題1と同じ計算が行われ、結果が出力されます。

**演習問題２**　A,B2つの変数に実数を入力し、3A+2B を計算するプログラムを作成したい。空いている部分を埋めなさい。

| プログラム | 説明 |
|---|---|
| PROGRAM SHIKI | プログラム名 SHIKI |
| [1]　A,B,C | 実数変数A,B,Cを宣言 |
| READ(*,*) A,B | A,B に値を読み込む |
| [2] | 3A+2Bを計算してCに代入する |
| [3] | 変数Cの値を出力する |
| [4] | プログラムの実行の終了 |
| END | プログラムの記述の終了 |

# ■ 第3章　入出力

計算に必要なデータを外部からプログラム中に読み込むことを入力と呼びます。また、プログラム中のデータを外部に書き出すことを出力と呼びます。入力や出力は特に指定しない限り、使っているコンピュータの標準的な入力･出力装置に対して行われます。パーソナルコンピュータの場合、入力はキーボード、出力はディスプレイとなっています。

ここで紹介する入出力は、「並びによる入力」および「並びによる出力」と呼ばれ、入出力の際にデータの桁数などの指定を行わず、予め定められた形式で入出力を行うものです。

## ■ ３．１　入力

入力は READ 文で行います。

```
READ(*,*)　入力並び
```

入力並びは、読み込んだデータを入れるための変数を、必要な数だけコンマで区切って並べたものです。

プログラムの実行中に READ 文に出会うと、データが入力されるまで待ちます。キーボードから入力されたデータは、変数に順次入れられて実行を続けます。データとデータの間は、空白かコンマで区切って入力します。

（プログラム例）

```
INTEGER I,J
REAL X
READ(*,*)　I,J,X
　　　　：
　　　　：
```

（入力例）

```
10　20　12.345
```

## ■ ３．２　出力

出力は WRITE 文で行います。

```
WRITE(*,*)　出力並び
```

出力並びは、出力する変数や定数、式などをコンマで区切って並べたものです。

出力の形式は、数値型と数値型の間は2桁の空白が入ります。'I=' のような文字の前後は空白が入らずに連続して出力されます。また出力する桁数は値によって自動的に決まります。

(プログラム例)

```
PROGRAM REI1
INTEGER I,J
REAL A
I=10
J=20
A=123.45
WRITE(*,*) 'I= ',I,' J= ',J,' A= ',A
STOP
END
```

(出力例)

```
 I= 10 J= 20 A= 123.4500
```

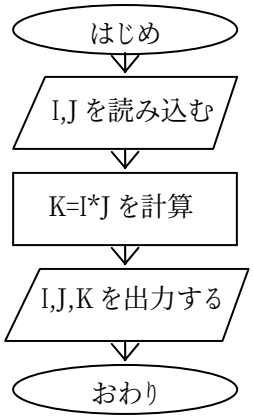
## ■ ３．３　入出力を使った例題

**例題３**　商品の単価と数量を入力し、金額を出力するプログラムを作成しなさい。

(プログラム例)

```
PROGRAM KINGAK
INTEGER I,J,K
READ(*,*) I,J
K=I*J
WRITE(*,*) 'TANKA= ',I,' KAZU= ',J
WRITE(*,*) 'KINGAKU= ',K
STOP
END
```

(流れ図)




(入力例)

```
 100  5
```

(出力例)

```
 TANKA= 100 KAZU= 5
 KINGAKU= 500
```

## ■ ３．４　入力の繰り返し

例題3では、単価と個数を読み込み、金額を計算して出力します。しかし、このままでは読み込むデータ（単価と個数）が2つ、3つと複数あった場合はその都度プログラムを実行させなければなりません。1回の実行でデータが無くなるまで計算を繰り返すためにはどのようにしたら良いでしょうか？

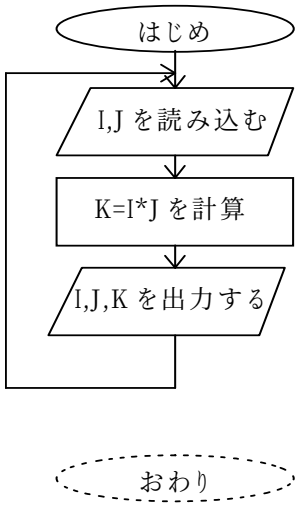
例題3のプログラムでは、3行目のREAD文でデータを読み込み、5、6行目のWRITE文で結果を出力しています。しかし、次にSTOP文があるためここでプログラムは終了してしまいます。複数のデータを扱うためには、もう一度 READ 文に戻り、データの入力･計算･結果の出力を繰り返すことが必要です。つまり、6行目の WRITE 文の後に3行目の READ 文に実行を移すための文を追加すれば良いことになります。

このような働きをする文として、GO TO 文があります。

GO TO **文番号**

文番号は、この GO TO 文によって実行を移す先を指定するためのものです。実行を移す先の文にはここで指定した文番号をつけます。文番号は、1桁目から5桁目の間に記述します。例題3のプログラムに GO TO 文を追加すると次のようになります。

```
      PROGRAM KINGAK
      INTEGER I,J,K
10    READ(*,*) I,J
      K=I*J
      WRITE(*,*) 'TANKA=   ',I,' KAZU= ',J
      WRITE(*,*) 'KINGAKU= ',K
      GO TO 10
      STOP
      END
```




これによって、WRITE 文によって結果を印刷した後は必ず3行目の READ 文に戻り、再度データを読み込むようになります。

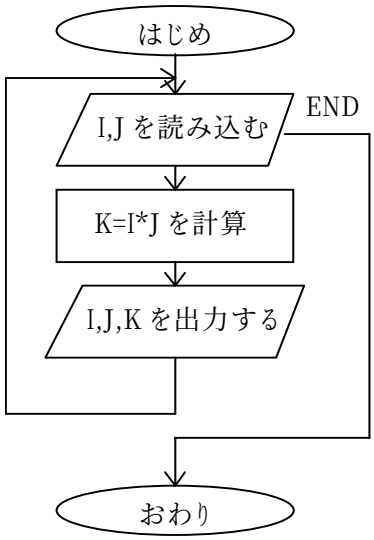
しかし、このプログラムをよく見ると、プログラムを終了する STOP 文にはたどり着かないようになっています。読み込むデータが無くなっても、プログラムはREAD文を実行してデータを読み込もうとします。その結果、プログラムはデータが足りない事になりエラーを出してしまいます。

そこで、読み込むべきデータが無くなった時に入力を止めて、指定された文へ実行を移す指定を READ 文に対して行います。これは次のように記述します。

READ(*,*,END=**文番号**)　**入力並び**

これによって、入力するデータが無くなった時、END= に指定した文番号を持つ文に実行が移ります。例題では、データが無くなったらプログラムを終了すればよいので、STOP 文に実行が移るようにします。

```
      PROGRAM KINGAK
      INTEGER I,J,K
10    READ(*,*,END=20) I,J
      K=I*J
      WRITE(*,*) 'TANKA=   ',I,' KAZU= ',J
      WRITE(*,*) 'KINGAKU= ',K
      GO TO 10
20    STOP
      END
```



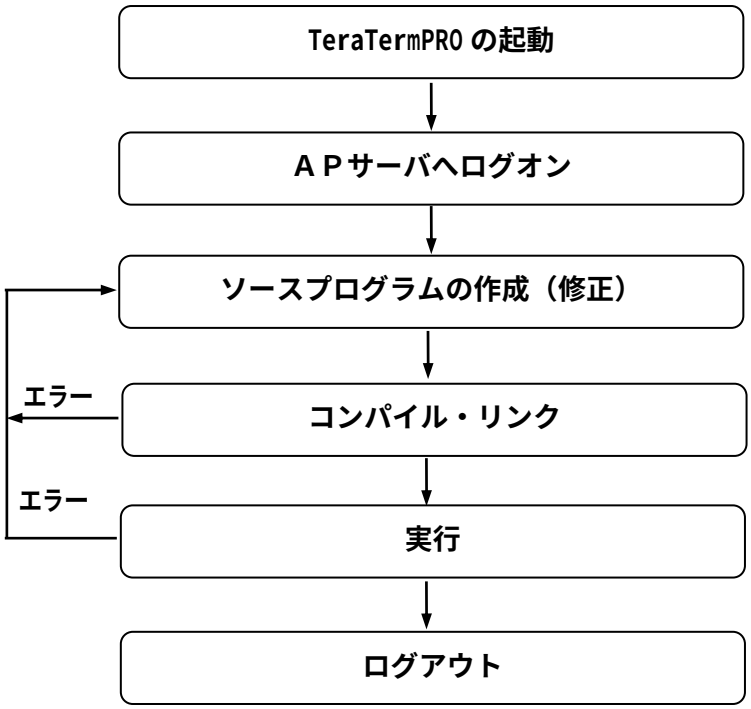

これでデータが無くなるまで繰り返すプログラムとなりました。なお、キーボードからデータを入力している場合に、データの終わりをプログラムに知らせるためには、つぎのように入力します。

(パーソナルコンピュータの場合)
・CTRL キーを押しながら Z キーを押して、リターンキーを押す。

(UNIX コンピュータの場合)
・CTRL キーを押しながら D キーを押す。

# ■ 第4章　実習方法

ここでは、実際にコンピュータを使ってプログラムを実行するまでの、実習の方法について説明します。利用するコンピュータは、総合情報センターのアプリケーションサーバ（bosei）です。実習の流れは次のようになります。

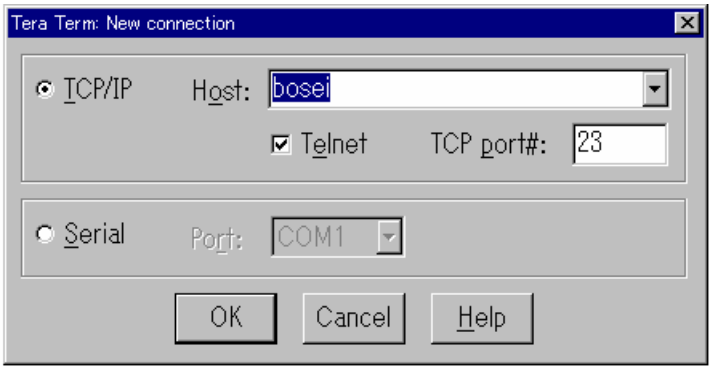



## ■ ４．１　パーソナルコンピュータの起動

パーソナルコンピュータの電源を入れ、Windows の初期画面を表示させます。なお、この手順についての詳細は、端末室備え付けの『パーソナルコンピュータ操作ガイド』を参照して下さい。

## ■ ４．２　ログイン

スタートボタンをクリックし、「プログラム(P)」から「ネットワーク」へマウスポインタを移動し、「TeraTerm PRO」の項目をクリックします。すると以下の画面が表示しますので Host:の欄に **ｂｏｓｅｉ** を選択し**ＯＫ**を押します。



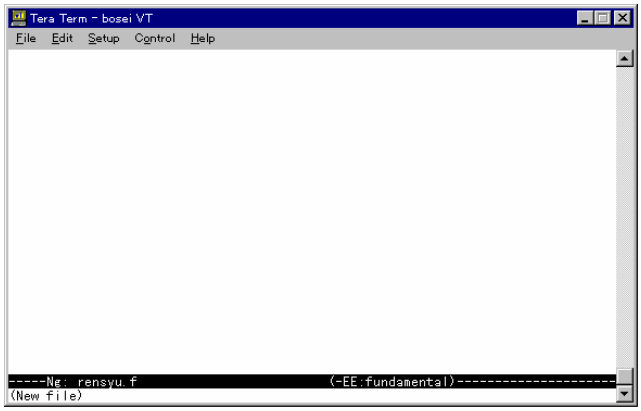
次のようなメッセージが現れますので、「login:」に対して自分のログイン名、「Password:」に対してパスワードを入力します。パスワードは入力しても表示されないので注意して下さい。

```
UX/4800 (bosei) (pts/29)
login : 9jzz1234        ←  各自のログイン名を入力します。
Password:               ←  パスワードを入力しても画面は何も変化しません。
:
101 9jzz1234@bosei      ←  セッションが開設。これをプロンプトといいます。
                           この状態でコマンドを入力することができます。
```

## ■ ４．３　ソースプログラムの作成（修正）

プログラムの作成または修正は ng エディタを使って行います。ここでは rensyu.f という名前のプログラムを作成します。また、 FORTRAN のソースプログラムの拡張子には「.f」を必ずつけるようにして下さい。プログラムの入力や修正がおこなえるようになります。

```
9jzz1234@bosei% ng  rensyu.f
```




以下のプログラム名を入力してください。

キーボードから入力した文字は、画面上のカーソル（点滅している四角）の位置に表示されます。例題のプログラムを文の開始位置などに注意して入力してみましょう。

```
PROGRAM KINGAKU
INTEGER  I,J,K
WRITE(*,*) 'INPUT TANKA AND KAZU'
READ(*,*) I,J
K=I*J
WRITE(*,*) 'TANKA =',I,'KAZU =',J
WRITE(*,*) 'KINGAKU= ',K
STOP
 END
```

文字の修正は、次のようなキーを使って行います。

(BS キー)
カーソルの左側の文字を消去します。
(DEL キー)
カーソル上の文字を消去します。
(カーソル移動キー:↑、↓、← 、→ )
カーソルを上・下・左・右に移動します。
(TAB キー)
次の入力位置を右に8桁移動します。文の書き出しに6個の空白を入れる代わりに、TAB キーを使うと便利です。

プログラム入力後、エディタを終了するためには CTRL キーを押しながら x、CTRL キーを押しながら c を押します。すると次のメッセージが画面下に表示されます。

```
■ Save file /uhome/a/9jzz1234/rensyu.f ? (y or n)
```

保存終了の場合はこのメッセージに対して **y** と答えます。

他にも ng エディタは以下のコマンドを使ってファイルを編集することができます。

| | |
|---|---|
| CTRL - k | カーソル位置から行末まで削除 |
| BS | カーソルの前の文字を1文字削除 |
| CTRL - d | カーソル上の文字を1文字削除 |
| CTRL - SPACE | カーソルの位置に始点を設定 |
| CTRL - w | CTRL - SPACE で設定した始点からカーソル位置までを削除してバッファにコピー |
| ESC w | CTRL - SPACE で設定された始点からカーソルの位置までをバッファにコピー |
| CTRL - y | 削除やコピーでバッファに保存された内容をカーソルの位置に挿入 |
| CTRL - x CTRL - c | ng エディタの終了 |

※ CTRL - k CTRL キーを押したまま k キーを押します

ESC w ESC キーを押した後、w キーを押します

※他にも利用できるコマンドは数多くあります。詳細は総合情報センターホームページ (http://www.cc.u-tokai.ac.jp/) から「設定方法・利用方法・Q&A」->「CPU サーバ、AP サーバの利用について」->「エディタの使い方」->「ng エディタ」を参照してください。

## ■ ４．４　コンパイルとリンク

実際にプログラムを動かすためには、コンパイル・リンクという作業をおこないます。コンパイルとはプログラムをコンピュータが理解できる形式(機械語)に翻訳し、文法などに誤りがあるかを見つける作業です。リンクとは入出力などのシステムに関連するプログラムをなどを集めて実行可能な形式にする作業です。コンパイルとリンクは以下のコマンドで同時に実行できます。実行形式のファイル名を指定しないと a.out という名前の実行形式のファイルができあがります。また、-o というオプションを使って実行形式のファイル名を指定することもできます。

```
■ 9jzz1234@bosei%  f77  rensyu.f                ←実行形式 a.out を作成
■ 9jzz1234@bosei%  f77  rensyu.f  -o  rensyu   ←実行形式 rensyu を作
```

作成したソースプログラム中に文法の誤りがあると、下のようにコンパイル時に error メッセージが表示されます。

```
9jzz1234@bosei%  f77 rensyu.f
 62 Error on line 2 of rensyu.f: Execution error unclassifiable statement
9jzz1234@bosei%
```

この例では2行目(line 2 of rensyu.f)にエラーがあることを知らせていますので、「(2)プログラムの作成」からやり直し、誤りを訂正後ファイルを保存し、「(3)コンパイルとリンク」に進んでください。

## ■ ４．５　実行

コンパイル･リンクによって作成された実行可能形式のファイルをコマンドとして入力します。

```
■ 9jzz1234@bosei.cc%  a.out       実行形式 a.out を実行
■ 9jzz1234@bosei.cc%  rensyu      実行形式 rensyu を実行
```

ここで、入力データが必要な場合は、プログラム中に記述されているフォーマットに従って、データを入力します。

## ■ ４．６　ログアウト

セッションを終了します。bosei にログインをしたら最後には必ずログアウトが必要です。

```
9jzz234@bosei.cc%  logout
```

ログアウトを実行するとアプリケーションサーバとの接続が解除され「TeraTerm PRO」のウィンドウは閉じます。

## ■ ４．７　パーソナルコンピュータの終了

実習を終了し、パーソナルコンピュータの電源を切る場合は、まず「FORTRAN77 メニュー」から「終了」を選択します。Windows の初期画面に戻りますので、スタートボタンをクリックし、「**シャットダウン**(U)」をクリックします。「Windows の**シャットダウン**」というウインドウが開きますので、「コンピュータを**シャットダウンする(S)**」を選択して、「はい」をクリックします。しばらくすると、「コンピュータの電源を切る準備ができました」と表示され自動的に電源が切れます。

この手順を行わずに、いきなり電源を切ることは絶対にしないで下さい。

注意

一部のコンピュータでは自動的に電源が切れません。「電源を切断しても安全です。」と表示されたあと電源スイッチを押して電源を切ってください。

**演習問題３**　あるクラスでテストを行った。クラス全員分の点数を読み込み、クラス全体の平均点を計算するプログラムを作成しなさい。（クラスの人数は読み込んだデータの件数とする）

# ■ 第5章 繰り返し

## ■ ５．１ DO 文

プログラムでは、同じ処理を値だけを変えて何度も実行したい場合があります。例えば、九九の表を印刷するプログラムを考えてみましょう。まず1つの段を印刷することを考えると、1の段なら 1×1、1×2、・・・、1×9、2の段ならば 2×1、2×2、・・・、2×9 と言うように、1から9まで値(J)を変えながら1つの掛け算(段×J)を実行することになります。

このような繰り返しの処理はループ文といい、DO 文を使って表現します。

```
      DO  文番号  変数=式 1，式 2，式 3
                    •
                    •
                    •
文番号  CONTINUE
```

DO 文から、DO 文で指定した文番号の文までが繰り返し実行されます。文番号で指定された繰り返しの最後の文を DO ループの端末文と呼びます。DO ループの端末文には実行文を使うことができますが、使ってはいけない文もあるため通常は、CONTINUE 文を使います。CONTINUE 文は実行されても、なんの働きもしない文です。繰り返しの最後を明示するのに利用すると便利です。

ここで次のような用語を使って説明することにします。DO 文の次の文から DO ループの端末文までを DO ループの範囲と呼びます。この部分が繰り返し実行される部分です。また、DO 文に指定された変数を DO 変数、式1を初期値、式2を終値、式3を増分と呼びます。

DO 文による繰り返し処理は次のように実行されます。DO 変数の値が初期値から終値を越えるまで増分の値ずつ増加しながら DO ループの範囲が繰り返し実行されます。

例えば、

```
DO 100 N=1,10,1
```

という DO 文で始まる DO ループは、DO 変数の値が 1,2,3・・・,10 と変化しながら DO ループの範囲を合計10回実行します。

もう少し詳しく説明すると以下のようになります。

1. まず DO 変数(N)に初期値 1 が代入されます。そして DO ループの範囲が1回実行されます。
2. 次に DO 変数(N)に増分の値 1 が加えられ 2 となります。そして DO ループの範囲がさらに1回実行されます。
3. この処理を10回繰り返し、DO ループを抜けて端末文(行番号 10 の行)の次の文の実行へ移る。(DO 文の実行の終了)

また、増分の値が１の場合には増分の記述を省略することができます。したがって上の DO 文は以下のように書くことができます。

```
      DO 100 N=1,10
```

さらに増分に負の値を指定することができます。この場合には、DO 文の繰り返し処理は DO 変数の値が終値を下回るまで行われます。

```
      DO 200 N=10,1,-1
```

この DO 文では DO 変数の値が 10,9,8,･･･,1 と変化しながら DO ループの範囲を合計１０回実行することになります。

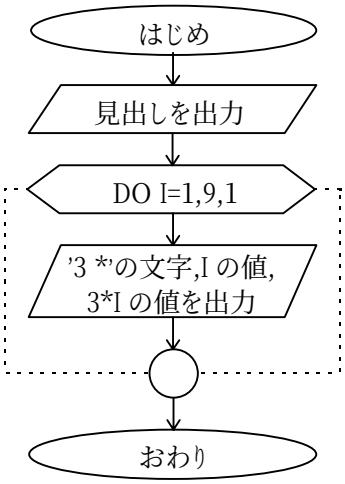
## ■ ５．２ DO 文を使った例題

**例題４** 九九の三の段を計算するプログラムを作成しなさい。

九九の１つの段を計算するには、九九の１つの値を計算する掛け算（段×J）を変数 J の値を変化させながら繰り返し実行することで計算できます。

（プログラム例）

```
      PROGRAM KUKU
      INTEGER I
      WRITE(*,*) '3 NO DAN'
      DO 10 I=1,9,1
        WRITE(*,*) '3 *',I,' = ',3*I
   10 CONTINUE
      STOP
      END
```

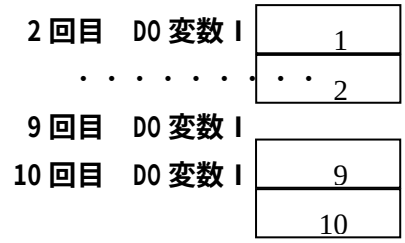


**[出力結果]**

```
3 NO DAN
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
  ・・・
3 * 9 = 27
```

**[DO ループの実行のされ方]**

DO ループの範囲が実行される際の DO 変数の値を示します。



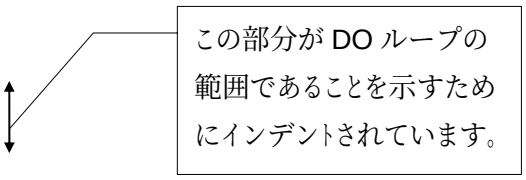

10 回目の繰り返しをする前に DO 変数に増分の値を加えてその値が 10 になります。この値は終値を超えているので 10 回目の繰り返しは行われません。DO ループの範囲の次の文へ実行が移ります。

## インデント

DO ループの範囲の文を DO 文よりも少し右に下げて書いておくと、DO ループの範囲が見やすくなって便利です。このような書き方をインデント(段付け)と呼びます。

(プログラム例)

```
      PROGRAM KUKU
      INTEGER I
      WRITE(*,*) '3 NO DAN'
      DO 10 I=1,9,1
        WRITE(*,*) '3 *',I,' = ',3*I
   10 CONTINUE
      STOP
      END
```




DO ループだけでなく、IF 文などでも同じようにインデントによって範囲を見やすくすることができます。

前の例題では、九九の三の段を計算しましたが、九九の表全体を計算するプログラムを考えてみましょう。前の例題の1つの段を計算する処理を、段の値を1から9まで変化させながら繰り返せば九九の表全体が印刷できることになります。

**例題５**　九九の計算をするプログラムを作成しなさい。

（プログラム例）

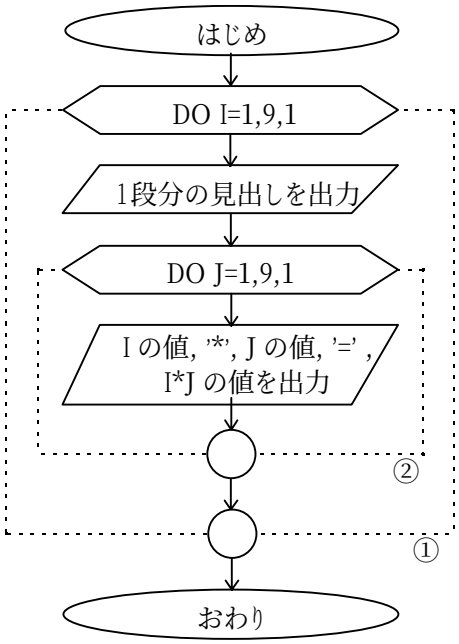
```
      PROGRAM KUKU
      INTEGER I,J
      DO 10 I=1,9,1
        WRITE(*,*) '        ---------------------'
        WRITE(*,*) I,' NO DAN'
        DO 20 J=1,9,1
          WRITE(*,*) I,' * ',J,' = ',I*J
20      CONTINUE
10    CONTINUE
      WRITE(*,*) '        ---------------------'
      STOP
      END
```

①

②

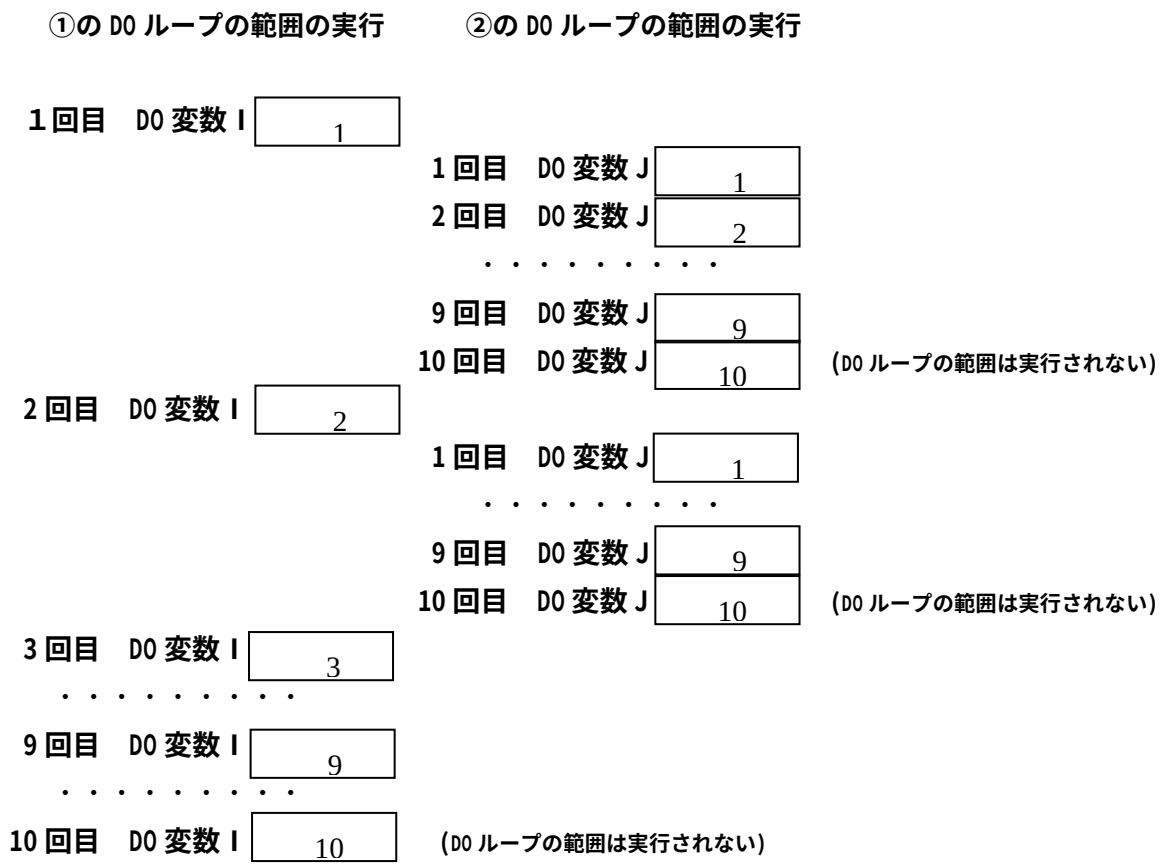
（出力例）

```
        --------------------
1 NO DAN
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
  • • •
1 * 9 = 9
        --------------------
2 NO DAN
2 * 1 = 2
  • • •
9 * 9 = 81
        --------------------
```



［DO ループの実行のされ方］

DO 文が実行される際の DO 変数の値の変化を以下に示します。




**演習４**　データ N (N は奇数)の値を読みとり、1+3+5+･･･+N の値を求めるプログラムを作成しなさい。

# ■ 第6章 組み込み関数

式の中で関数を利用した計算を行うことができます。例えば平方根の値を計算したいとすると、関数を利用しないで計算するには、平方根の値を求めるための公式を使って複雑な式を書かなければなりません。しかし、平方根を求める関数 SQRT が FORTRAN プログラムの中で利用できるようになっているため、次のような代入文で 2.0 の平方根の値を求めることができます。

```
X=SQRT(2.0)
```

このような処理系にあらかじめ用意されている関数を組み込み関数と呼びます（以下単に、関数と呼ぶ）。関数は式の中に以下の形式で書いて使用します。

```
関数名（引数）
```

使用する関数の名前と引数を指定することで、引数の値に対する関数値を求めることができます。引数の部分には式を書くことができます。複数の引数を使う関数では、コンマで区切って複数の引数を書きます。

代表的な組み込み関数を以下に示します。

| 関数名 | 使用方法 | 機能 |
|---|---|---|
| MOD | MOD(A1,A2) | A1 を A2 で割った余りを求める |
| MAX | MAX(A1,A2,…) | A1,A2, …の中の最大値を求める |
| MIN | MIN(A1,A2,…) | A1,A2, …の中の最小値を求める |
| ABS | ABS(A) | A の絶対値を求める |
| SIN | SIN(A) | A（単位ラジアン）の正弦の値を求める |
| COS | COS(A) | A（単位ラジアン）の余弦の値を求める |
| SQRT | SQRT(A) | A の平方根の値を求める |
| LOG | LOG(A) | A の自然対数の値を求める |

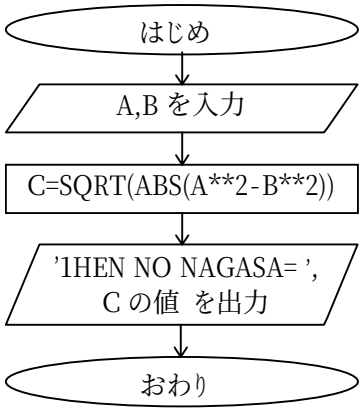
この他にも多数の組み込み関数が利用できます。詳しくは、使用する処理系のマニュアルを参照してください。

## ■ 6.1 組み込み関数の例題

**例題6** 直角三角形の斜辺ともう一辺の長さが分かっている時、残りの一辺の長さを求めるプログラムを作成しなさい。

(プログラム例)

```
      PROGRAM NAGASA
      REAL A,B,C
      READ(*,*) A,B
      C=SQRT(ABS(A**2-B**2))
      WRITE(*,*) '1HEN NO NAGASA = ',C
      STOP
      END
```



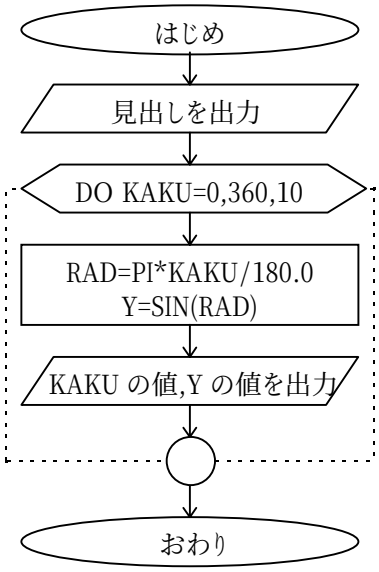

(実行例)

```
5
3
 1HEN NO NAGASA =    4.000000
```

**例題７**　角度 を 0 度から 360 度まで 10 度刻みで変化させたときの、角度の値と sin の値を表にするプログラムを作成しなさい。ただし、sin 関数は角度をラジアンとして計算するので、度からラジアンへの変換を行うこと。

(プログラム例)

```
      PROGRAM SINX
      INTEGER KAKU
      REAL RAD,Y
      WRITE(*,*) '      DEGREE      SIN(X)'
      DO 10 KAKU=0,360,10
        RAD=3.1415926*KAKU/180.0
        Y=SIN(RAD)
        WRITE(*,*) KAKU,Y
   10 CONTINUE
      STOP
      END
```




(出力例)

```
    DEGREE      SIN(X)
         0   0.0000000E+00
        10   0.1736482
             ......
       360  -3.0199161E-07
```

**演習５**　商品を箱詰めにする時、商品の数を入力し、箱詰めのあまりの商品の数を表にするプログラムを作成しなさい。箱の大きさは6個入り、8個入り、10個入り、12個入り、14個入りの5種類であり、それぞれの箱を使った場合のあまりの商品の数を表にするものとします。

# ■ 第7章 条件判定（ブロック IF 文）

IF 文を使うと様々な条件による判定をおこないその条件を満たした場合には、指定した命令を実行させることができます。

```
IF(条件 A)THEN
    命令 A
ELSE
    命令 B
ENDIF
```

```
IF(条件 A)THEN
    命令 A
ENDIF
```

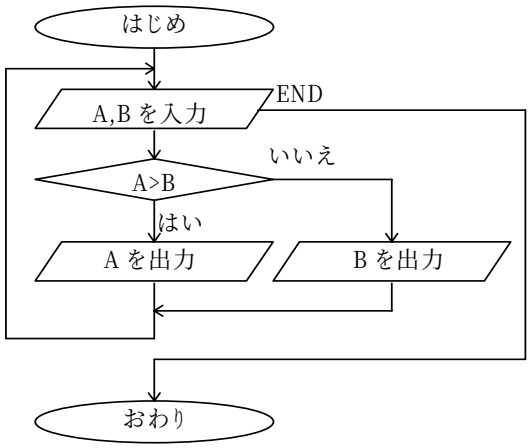
考え方として上記のように「もし(IF)」（条件 A）の条件を満たした「そのときは(THEN)」“命令 A”を「それ以外(ELSE)」のときは“命令 B”を実行し ENDIF 文で条件判定を終了しています。

（条件 A）を満たすかどうかの判定だけを行う場合は ELSE 文と命令 B を省略することができます。

**例題８** 2つの数 A,B を読み込み大小関係を判定して数値の大きな方を出力するプログラムを作成しなさい。

（プログラム例）

```
      PROGRAM HANTEI
      REAL A,B
10    READ(*,*,END=20)A,B
      IF(A.GT.B)THEN
        WRITE(*,*)A
      ELSE
        WRITE(*,*)B
      ENDIF
      GOTO 10
20    STOP
      END
```




プログラム中3行目の(A.GT.B）を関係式といいます。この関係式の代表的なものに下記のものがあります。

| 数　式 | FORTRAN | 関係演算子 |
|---|---|---|
| A ＝ B | **A.EQ.B** | **(Equal to)** |
| A ≠ B | **A.NE.B** | **(Not Equal to)** |
| A ＞ B | **A.GT.B** | **(Greater Than)** |
| A ＜ B | **A.LT.B** | **(Less Than)** |
| A ≧ B | **A.GE.B** | **(Greater than or Equal to )** |
| A ≦ B | **A.LE.B** | **(Less than or Equal to)** |

また「 ELSE　IF(それ以外でもし) 」を利用して複数の条件判定をおこなうことも可能です。

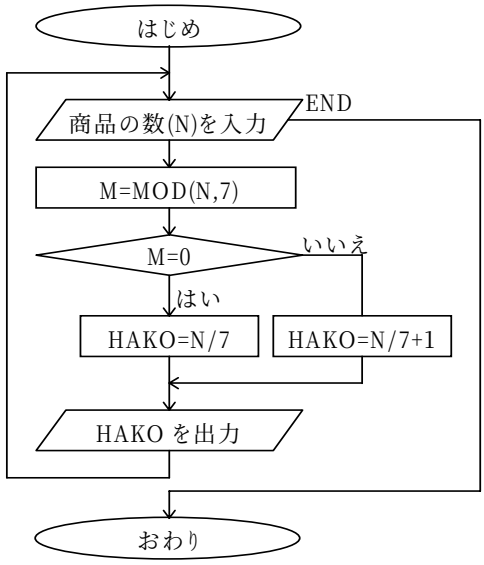
```
IF(条件 A)THEN
                    命令 A
ELSE  IF(条件 B)
                    命令 B
ELSE  IF(条件 C)
                    命令 C
ELSE
                    命令 D
ENDIF
```

## ■ ７．１　ブロック IF 文の例題

**例題９**　商品を7個ずつ箱詰めにする時、商品の数を入力すると必要な箱の数を計算し出力するプログラムを作成しなさい。

（プログラム例）

```
      PROGRAM SET
      INTEGER M,N,HAKO
10    WRITE(*,*)'SHOHIN NO KAZU?'
      READ(*,*,END=20)N
      WRITE(*,*)'SHOHIN NOKAZU ',N
      M=MOD(N,7)
            IF(M.EQ.0)THEN
            HAKO=N/7
            ELSE
                  HAKO=N/7+1
            ENDIF
      WRITE(*,*)'HAKO NO KAZU=',HAKO
      GOTO 10
20    STOP
      END
```

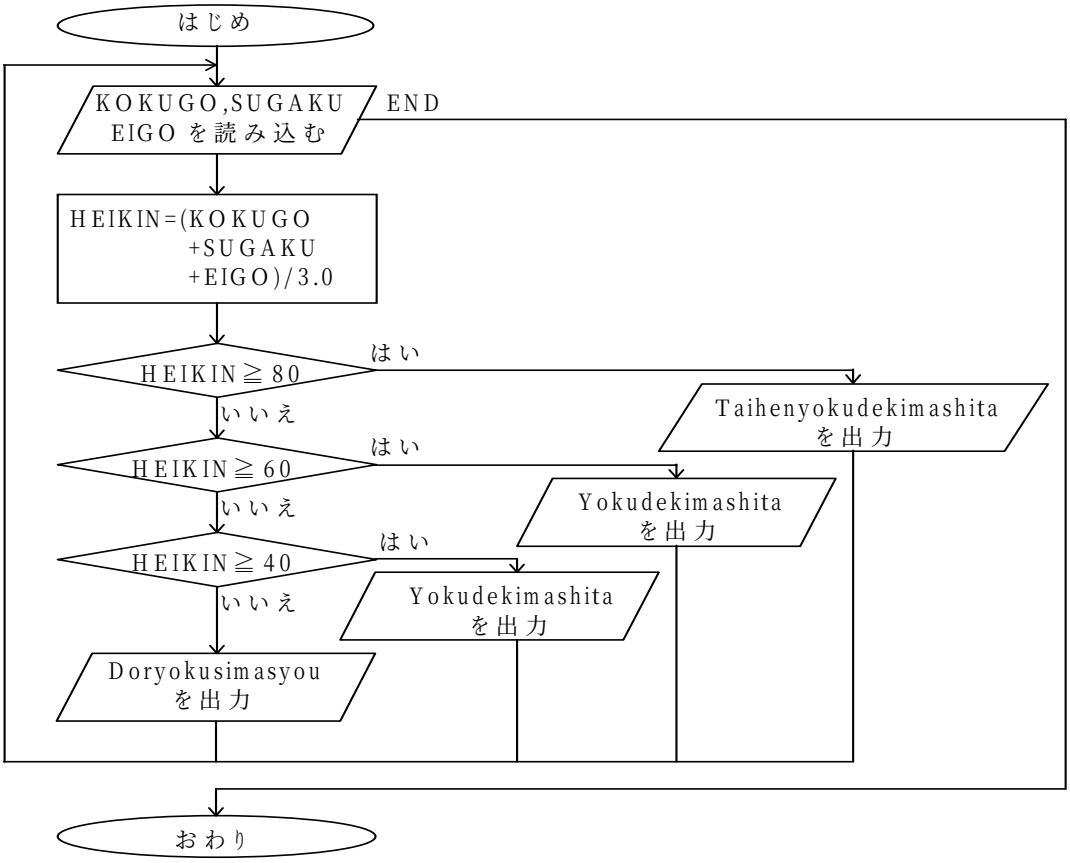


**例題１０**　国語、数学、英語の点数を入力するとそれらの平均点と成績評価を出力するプログラムを作成しなさい。評価の仕方は次のようにします。

| 判　定 | 成績評価 |
|---|---|
| 平均点 ≧ 80 | Taihen　Yokudekimashita |
| 80 ＞ 平均点 ≧ 60 | Yokudekimashita |
| 60 ＞ 平均点 ≧ 40 | Ganbarimasyou |
| 40 ＞ 平均点 | Doryokushimasyou |

（プログラム例）

```
      PROGRAM   HYOUKA
      INTEGER   KOKUGO,SUGAKU,EIGO
      REAL   HEIKIN
   10 READ(*,*,END=20)KOKUGO,SUGAKU,EIGO
      HEIKIN = (KOKUGO+SUGAKU+EIGO)/3.0
        IF(HEIKIN.GE.80.0)THEN
           WRITE(*,*)HEIKIN,'Taihenyokudekimashita'
        ELSE IF(HEIKIN.GE.60)THEN
           WRITE(*,*)HEIKIN,'Yokudekimashita'
        ELSE IF(HEIKIN.GE.40)THEN
           WRITE(*,*)HEIKIN,'Ganbarimasyou'
        ELSE
           WRITE(*,*)HEIKIN,'Doryokushimayou'
        ENDIF
        GOTO 10
   20 STOP
      END
```


はじめ
KOKUGO,SUGAKU
EIGO を読み込む
END
HEIKIN=(KOKUGO
+SUGAKU
+EIGO)/3.0
HEIKIN≧80
はい
いいえ
Taihenyokudekimashita
を出力
HEIKIN≧60
はい
いいえ
Yokudekimashita
を出力
HEIKIN≧40
はい
いいえ
Yokudekimashita
を出力
Doryokusimasyou
を出力
おわり