関数副プログラムとサブルーチン副プログラム

山本昌志*

2004 年 2 月 10 日

1　副プログラムとは

1.1　はじめに

ここでは、関数副プログラムとサブルーチン副プログラムについて、学習します。いずれも、副プログラムという手法が使われます。そのプログラムの文法を学習する前に、このプログラム手法がどのようなもので、なぜ便利なのか説明しておくことが重要であろう。これが分かれば、副プログラムが分かったも同然です。この手法は非常に便利なので、いかなるプログラム言語でも使われているので、ここでその内容を理解し、将来、他の言語を学ぶときに応用できるようになることが肝要です。

まず、言葉の違いについて説明が必要であろう。通常、副プログラムではなくサブルーチンと言われることが多いので、注意が必要である。また、副プログラムに対して主プログラムと言うものがあり、それはメインルーチン、あるいはメインプログラムといわれます。英語と日本語の違いですが、コンピューターはアメリカで発展したので、英語が使われることが多いので覚えておきましょう。

コンピューターのプログラムは、皆さんが練習問題で作成する数行のものから、widows の 4 千万行[1]に及ぶものまで多種多様です。サブルーチンという考え方は、長いプログラムを書く場合に非常に有効です。また、長くなくても、分かりやすいプログラムを書く場合に効果を発揮します。

1.2　自動車の設計・製作

プログラムの書き方の説明に入る前に、アナロジー (analogy 類似) として、自動車を作ることを考えます。自動車も製品ですが、プログラマーにとってプログラムは製品です。自動車は、非常に多くの部品から出来上がっています。たとえば、ピストン、シリンダー、ボルト、プラグ、タイヤ、ハンドル、ばね、電線、電球、集積回路、コンデンサー、火薬、磁石 · · · など、それこそ大変な数の部品があります。そして、これを組み合わせて自動車が出来上がります。これを、すべて一人で設計することは到底不可能なことは想像できるでしょう。それでは、どうやって設計するかとと言うと、多くのエンジニアーで寄ってたかって設計します。エンジン、トランスミッション、ブレーキ、居住空間、タイヤ、オーディオ機器、空調機器、操作パ

*国立秋田工業高等専門学校　電気工学科

[1]web で調べていたら、http://www.fluidlab.naoe.t.u-tokyo.ac.jp/∼minnie/FreeBSD/memo.html に記述がありました。windows 2000 のソースコードの行数とのこと。

ネル、サスペンション他のように、部品ごとに設計を行います。重要なことは、部品単位、はっきりと区分けをして設計をすることです。

部品ごと、設計を行うのですが、それぞれの部品の接続に関しては、予め決めておきます。たとえば、エンジンを設計する人とトランスミッションを設計する人の間で、その接続方法を決めてから、各々設計にかかります。そうしないと、最後に組み合わせるときに、それぞれの部品が接続できないということになります。

部品ごと設計が終われば、すべてを組み合わせて、自動車の設計が完了です。予めそれぞれの部品の接続方法が決められているので、ちゃんとした自動車が出来上がります。大雑把に言うと、自動車の設計は、このようにして行われます。製造もほとんどこれに似ています。部品をつくり、最後に組み立てて完成です。この方法は、大規模なものを作るときに行われる一般的な方法で、重要なことは、

- 目的の機能を果たす機械、ここでは自動車を機能ごとの部品に分ける。
- 各々の部品の接続方法は予め決めておく。

です。このように機能別に分けることにより、自動車の設計・製作は格段に分かりやすくなります。

ここでは、自動車をエンジンやトランスミッション等、1 段階に分けましたが、実際には、更に細かく分けます。エンジンであれば、ピストンやバルブ、シリンダーと分けていきます。部品によっては、更に細かく機能ごとに分けていくでしょう。機能ごとに分けると分かりやすくなることを理解してください。

プログラムでは、この特定の機能ごとの集まりをサブルーチンといいます。メインルーチンは、プログラム全体を統括しているもの、自動車のフレームみたいなものと考えてください。

1.3 分かりやすいプログラムを書くには

大量のパーツからできている自動車も機能別の部品に分けることにより、その構造が分かりやすくなります。プログラムも一緒で、大規模なプログラムも、いろいろな機能を持ったプログラムの集まりです。機能ごとに分担しないで、ひとつの大きなプログラムで、目的を果たすようなものを書くとすれば、非常に大変です。100 行くらいが限界でしょう。1000 行のプログラムを書くことはほとんど不可能でしょう。その様子を図 1 と 2 に示します。

このように、機能別に分けることにより、プログラムは分かりやすくなります。そして、機能がはっきりすれば分担してプログラムを書くこともできます。ただ、自動車の機能ごとの部品と同じように、プログラムで処理するデータの取り合いはきちんとする必要があります。

自動車の部品にはどれも優劣がありませんが、プログラムの場合、1 つだけ特別な機能のルーチンがあります。メインルーチンと呼ばれるもので、必ず、最初に実行されるプログラムです。プログラムの場合、1 行毎順番に実行されるため、どれから実行するか決める必要があります。自動車の部品の場合、それぞれが同時に動き機能するため、メインルーチンに対応するものがありません。

それでは、実際のプログラムでサブルーチンが便利で、プログラムを分かりやすくする様子を見ていきましょう。

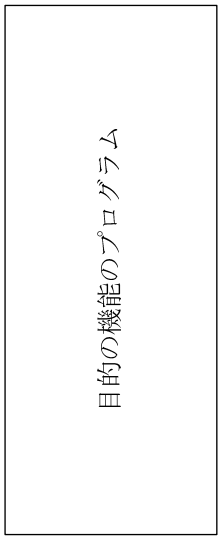

図 1: メインルーチンだけのプログラム

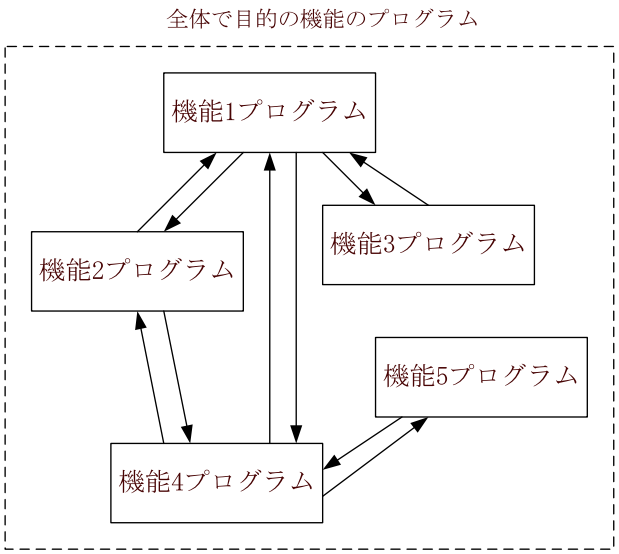

図 2: 機能毎 (サブルーチン) に分割されたプログラム。矢印は、データの流れをあらわす。

2 関数副プログラム

2.1 どのような時につかうか

教科書の例にならい、次のような関数を計算する場合、どうするか考えよう。

$$f(x) = \begin{cases} \sqrt{1-x^2} & -1 \leq x \leq 1 \text{ のとき} \\ 0 & \text{それ以外のとき} \end{cases} \tag{1}$$

この関数を 1 回のみ計算する場合は、IF 文を使うことで対処できますが、何回も計算する場合は厄介です。通常のプログラムでは似たような計算を大量に行うので、できるだけ同じ処理はまとめたほうが効率的です。先の関数を何億回も計算することは、決して珍しくありません。

そのために、この関数を定義して、いつでも呼び出せるようにすると便利です。あたかも、$\sin(x)$ のように x を入れれば、計算してくれるようにすれば非常に便利です。一つの解決方法が文関数ですが、これは 1 行で関数を定義する必要があるので、式 (1) のように IF 文を使って、場合分けが必要の時には使えません。文関数で定義できない複雑な関数を定義するときに、関数副プログラムを使います。

もう一度、文関数と関数副プログラムの違いをまとめておくと次のようになります。

文関数 1 行で定義できる簡単な関数のときに使います。

関数副プログラム 複数の行で定義する必要がある複雑な関数のときに使います。

2.2 実際のプログラム

教科書の例を用いて説明しよう。教科書のプログラムは分かりにくいので、実際、通常のプログラマーが記述するように書くと以下のようになります。

メインルーチンを見るだけで、このプログラムの動作が分かると思います。関数 F(X) の値を X=0.5 から 0.1 ステップで 1.5 まで計算して、書き出していると想像できます。実際の関数は、関数副プログラムを見ればすぐに分かるようになっています。

関数 F(X) はどこからでも使うこともでき便利です。同じ処理を何回も書かないですむということです。これは、関数を変えたい場合、プログラムを書き直すのは 1 箇所で済みます。これにより、プログラムは分かりやすくなり、作成時間も非常に短くできます。

```
*=========================================================
* MAIN ROUTINE
*=========================================================
      PROGRAM MAIN

      DO 10 X=0.5, 1.5, 0.1
        Y=F(X)
        WRITE(6,601) X,Y
 601    FORMAT(2E20.8)
  10  CONTINUE

      STOP
      END

*=========================================================
*     USEER DEFINEDED FUNCTION
*=========================================================
      FUNCTION F(X)

      IF(ABS(X).LE.1.0)THEN
        F=SQRT(1.0-X*X)
      ELSE
        F=0.0
      ENDIF

      RETURN
      END
```

これを関数副プログラムを使わないで書くと、以下のようになります。これだとプログラムの意図した内容がわかりにくく、保守が難しくなります。また、長いプログラムを書く場合も大変です。

```
      PROGRAM KEISAN

      DO 10 X=0.5, 1.5, 0.1

        IF(ABS(X).LE.1.0)THEN
```

```
        Y=SQRT(1.0-X*X)
      ELSE
        Y=0.0
      ENDIF

      WRITE(6,601) X,Y
601   FORMAT(2E20.8)

 10 CONTINUE

    STOP
    END
```

2.3 関数副プログラムの文法

2.3.1 関数の呼び出し

関数は、関数名で呼び出して、算術代入文で変数に入れるのが普通です。関数名は、組み込み関数と異なれば、任意の名前をつけることができます。実引数並びは、関数の独立変数で、複数個が許されます。多変数関数に対応しているということです。また、配列も許されます。ただし、実引数と仮引数の数と型は一致させる必要があります。

同様に、関数の戻り値の型と代入される変数の型も一致させる必要があります。

代入される変数=関数名（実引数並び）

ただし、関数の名前と型が暗黙の型宣言に従わない場合、呼び出し側でも型宣言が必要である。たとえば、次のように場合である。

```
*=========================================================
* MAIN ROUTINE
*=========================================================
     INTEGER WA

     I=2
     J=3
     K=WA(2,3)

     STOP
     END

*=========================================================
* FUNCTION
*=========================================================
     INTEGER FUNCTION WA(M,N)
```

```
WA=M+N

RETURN
END
```

2.3.2 関数の定義

関数の定義は以下のように行います。関数の型は、REAL や INTEGER 等、変数の場合と同じように書きます。これは、戻り値の型を示します。ただし、これは省略可能で、その場合は関数名に従い暗黙の型宣言[2]になります。

仮引数並びと実引数並びは、同じ数で型でなくてはなりません。ただし、引数の変数名は異なっても、何ら問題はありません。関数の処理の部分は、今まで学習してきた FORTRAN のプログラムを書けばよいのです。計算のみならず、WRITE 文や READ 文も書けます。ただし、最後に関数名に値を代入する必要があります。これが戻り値になります。

関数副プログラム内で使われた変数は、引数以外は全く、メインルーチンに影響を及ぼしません。メインルーチンと同じ名前の変数が使われていても、その結果が反映されることはありません。

```
関数の型 FUNCTION 関数名 (仮引数並び)
宣言文

実行文 (関数の値を求めるための処理)

関数名=処理の結果の関数の値

RETURN
END
```

3 サブルーチン副プログラム

3.1 どのような時につかうか

関数副プログラムの場合、入力の変数は複数個でも良いが、出力 (戻り値) は 1 個に限ります。それは文法から明らかでしょう。それに対して、サブルーチン副プログラムは、複数の値を返すことができます。

3.2 実際のプログラム

教科書の例を用いて説明しよう。教科書のプログラムは分かりにくいので、実際、通常のプログラマーが記述するように書くと以下のようになります。このプログラムでは、パイプの外径と内径、長さ、密度を与えて、体積と重さを計算しています。それを、SUBROUTINE PIPE で計算しています。

[2]関数名の始まりが、I, J, K, L, M, N の時は INTEGER とみなされる

最初の CALL 文では、キーボードから入力された、外径 (A) と内径 (B)、長さ (L1)、密度 (G1) を使って、体積 (V) と質量 (W) を計算しています。

2 番目の CALL 文では、直接サブルーチンに値を与えています。すなわち、外径を 8.0、内径を 6.0、長さを 350.0、密度を 7.8 として、体積 (V) と質量 (W) を計算しています。

```
*=========================================================
* MAIN ROUTINE
*=========================================================
      PROGRAM MAIN
      REAL A, B, L1, G1, V, W

      READ(5,*)A, B, L1, G1
      CALL PIPE(A, B, L1, G1, V, W)
      WRITE(6,600)V,W

      CALL PIPE(8.0, 6.0, 350.0, 7.8, V, W)
      WRITE(6, 600)V,W

 600  FORMAT('V=',E15.8, ' W=', E15.8)

      STOP
      END

*=========================================================
*     SUBROUTINE OF PIPE CALCULATION
*=========================================================
      SUBROUTINE PIPE(X, Y, L, G, V, W)
      REAL X, Y, L, G, V, W

      V=3.14159/4.0*(X*X-Y*Y)*L
      W=V*G

      RETURN
      END
```

このプログラムをサブルーチン副プログラムを使わないで書くと、以下のようになります。分かりにくいし、パイプの断面が、円から、正方形に変更になった場合、同じような部分を 2 箇所書き直す必要があります。それに比べて、サブルーチン副プログラムだと、サブルーチン内を書き直すことで済みます。これは、プログラマーの労力を格段に減らすことができます。

```
      PROGRAM PIPECAL
      REAL A, B, L1, G1, V, W

      READ(5,*)A, B, L1, G1
      V=3.14159/4.0*(X*X-Y*Y)
      W=V*G
      WRITE(6,600)V,W
```

```
      CALL PIPE(8.0, 6.0, 350.0, 7.8, V, W)
      V=3.14159/4.0*(8.0*8.0-6.0*6.0)*350.0
      W=V*7.8
      WRITE(6, 600)V,W

  600 FORMAT('V=',E15.8, ' W=', E15.8)

      STOP
      END
```

3.3 サブルーチン副プログラムの文法

3.3.1 サブルーチンの呼び出し

サブルーチンは、CALL サブルーチン名で呼び出します。データの受け渡しは、引数で行います[3]。関数副プログラム同様、実引数並びは、複数個が許されます。また、配列も許されます。ただし、実引数と仮引数の数と型は一致させる必要があります。

```
CALL サブルーチン名 (実引数並び)
```

3.3.2 サブルーチンの定義

サブルーチンでの処理の定義は以下のように行います。すなわち、SUBROUTINE と最初に書いて、その後、仮引数が並びます。それから、通常のプログラムのように書いて、RETURN と END 文で終わります。

仮引数並びと実引数並びは、同じ数で型でなくてはなりません。ただし、引数の変数名は異なっても、何ら問題はありません。関数の処理の部分は、今まで学習してきた FORTRAN のプログラムを書けばよいのです。計算のみならず、WRITE 文や READ 文も書けます。

サブルーチン副プログラム内で使われた変数は、引数以外は全く、メインルーチンに影響を及ぼしません。メインルーチンと同じ名前の変数が使われていても、その結果が反映されることはありません。

```
SUBROUTINE サブルーチン名 (仮引数並び)
宣言文

実行文 (関数の値を求めるための処理)

RETURN
END
```

[3]引数以外にデータの受け渡しは可能ですが、ここでは述べません。DATA 文等の方法があります。

4　練習問題

以下のプログラムを作成して、3 月 16 日 (月) 中にレポートとして提出すること。3 月 17 日 (火) の授業では、それを元に情報処理センターで演習を行う。

4.1　関数副プログラム

［問題 1］　階乗の計算

キーボードから、N の値を読み込んで、その階乗 (N!) を計算し、出力するプログラムを作成せよ。階乗とは、

$$N! = N \times (N-1) \times (N-2) \times (N-3) \times \cdots \times 3 \times 2 \times 1 \tag{2}$$

の計算のことを言う。

プログラムを作成する場合、以下の条件に従うこと。

- N はキーボードから入力するが、そのプログラムはメインルーチンに書くこと。
- 階乗の計算は、関数副プログラム中で計算すること。
- 計算結果の出力のプログラムも、メインルーチンに書くこと。

［問題 2］　三角形の面積 (ヘロンの公式)

三角形の面積 S は、ヘロンの公式を用いると

$$S = \sqrt{s(s-a)(s-b)(s-c)} \tag{3}$$

と求めることができる。ただし、$s = (a+b+c)/2$ で、a,b,c は三角形の各辺の長さである。

プログラムを作成する場合、以下の条件に従うこと。

- 各辺の長さはキーボードから入力するが、そのプログラムはメインルーチンに書くこと。
- 面積の計算は、関数副プログラム中で計算すること。
- 計算結果の出力のプログラムも、メインルーチンに書くこと。

4.2　サブルーチン副プログラム

［問題 1］　並び替え

教科書 P.156(10) の問題である。与えられた数字を小さい順に並べるサブルーチンを作成し、以下のデータを小さい順に並べよ。

123,　35,　99,　41,　2,　999,　567,　15,　68,　362,

ただし、データの入力は以下のように、メインルーチンで配列に直接代入すること。

```
A(1)=123
A(2)=35
A(3)=99
A(4)=41
A(5)=2
A(6)=999
A(7)=567
A(8)=15
A(9)=68
A(10)=362
```

［問題 2］　三角関数の計算

2 つの角度を度単位で入力し、それをラジアン単位に変換して、

$$z1 = \sin(\theta_1) + \sin(\theta_2) \tag{4}$$

$$z2 = \sin(\theta_1) - \sin(\theta_2) \tag{5}$$

$$z3 = \sin(\theta_1) * \sin(\theta_2) \tag{6}$$

$$z4 = \sin(\theta_1) \div \sin(\theta_2) \tag{7}$$

を計算せよ。ただし、これら 4 つの計算については、サブルーチン副プログラムを用いたプログラムであること。