補講 関数、サブルーチンの練習

國仲寛人 (TA)

平成 10 年 11 月 11 日

1　サブルーチンの文法

数値計算で問題を処理するとき、そのプログラムはいくつかの内部処理を含んでいる場合が普通である。プログラミングの際にそれらをメインのプログラムの中に書くよりも、内部処理は内部処理で副プログラムとして別に書いてその結果をメインに渡すようにしておき、メインはメインで全体の幹となる流れを書いて、必要な時に内部処理の結果を呼び込むようにしておくとプログラムが見やすくなり、また別の問題を処理する機会に副プログラムをそのまま使うこともできる。この考え方が Fortran でのサブルーチンの考え方であり、C 言語では関数に相当する。

以下、Fortran と C 言語のそれぞれについて副プログラムの使い方について見ていこう。

1.1　Fortran の場合

サブルーチン副プログラムは、

$$\text{SUBROUTINE サブルーチン名 } (A, B, C, ...)\ (A, B, C, ... \text{ :仮引数})$$

で始まり、END で終る。これを呼び出すには、メイン内、もしくは他の副プログラム内で、

$$\text{CALL サブルーチン名 } (A', B', C', ...)\ (A', B', C', ... \text{ :実引数})$$

と書けばよい。

このとき大切なのは、仮引数と実引数の間に、

$$\begin{aligned} A &\longleftrightarrow A' \\ B &\longleftrightarrow B' \\ C &\longleftrightarrow C' \\ &..... \\ &..... \end{aligned}$$

という、並べられた順番で対応があるという事である。つまり、仮引数でサブルーチンを書いておいて、実際はそれらに対応する実引数を代入した結果をメイン内で使う事になる。よって対応する引数同士は変数の型 (real,double,integer 等) が一致していないといけない。

また、Fortran では関数副プログラムというものも存在する。これは FUNCTION 文で副プログラムを書いてメイン内で使うもの (レポート (7 月 15 日出題分) の解答の問 1 参照) であり、サブルーチンとは使い方が異なる。以下にそれらの相違点をまとめておく。

(関数副プログラム)

- 関数名は変数名と同じ扱いであり型がある。代入すべき数値の型に従う。
- 算術式の中に書いて引用する。

(サブルーチン副プログラム)

- サブルーチン名は、そのプログラムの呼び名であり値を持たない。
- CALL 文で呼び出し、その後で結果を算術式の中に使う。

1.2 C 言語の場合

C 言語の関数の使い方は例えば以下のようになる。

```
#include<stdio.h>
    .....

double 関数名(double,double,...);
    .....

main()
{
    .....

    c=関数名(a,b,...);

    .....
}

double 関数名(double x,double y,...)
{
    .....
   return z;
}
```

まず使う関数はわかりやすい名前で (rungekutta とか)main の前で定義する。そのとき、仮引数名は書く必要はないが、関数内で使う仮引数の型を全て書いておく。

関数を定義するのはメインの前でも後でもいいのだが、見やすくするためには後に書いた方が良い。そのとき,

型 関数名 (型 引数 1、型 引数 2、...){}

のように書いて {} 内に関数の本体を書く。このとき定義した引数は関数内で有効である。

最後に関数内で計算した結果を main に返すという意味で、

return 引数;

と書いておく。こうして main 内で関数名 (a,b,...) と書く事によって、引数 1、引数 2,... に a,b,... を代入した結果を呼び込む事ができるようになる。

さて、このように関数からメインに値を返す以外に、C 言語の関数にはその機能のみが使われ、値を返さない使い方もある。つまり、本来ならメイン内に存在するまとまった手続きを関数として外で定義して、メイン内で必要な時に外で処理してもらう事ができるのである。その場合関数は、

void 関数名 (型 引数 1、型 引数 2、...){}

として記述する。定義する時も void 型で定義すればよい。

2 プログラム例

例として、調和振動子の相図を書くプログラムを載せた。これはバネ定数 K のバネに質量 M のおもりがついており、おもりの平衡位置からのずれを x、速度を v とした時に初期条件 $(x, v) = (1.0, 0.0)$ で、

$$\frac{dv}{dt} = -\frac{K}{M}x, \frac{dx}{dt} = v$$

という連立微分方程式をルンゲクッタ法で解くというものである。

アルゴリズムは、

1. 時間刻み Δt の設定
2. 初期条件 (x, y) の設定
3. 初期値とルンゲクッタ法で、時間 Δt 後の値、(x_{new}, v_{new}) を計算する。
4. 得られた (x_{new}, v_{new}) を初期値として、3 に戻る。

となる。

Fortran と C 言語のプログラムはそれぞれサブルーチン (関数) を使う場合と使わない場合を載せてある。

プログラムを作って後で見直したり、他の人に見せたりする場合、プログラムの大まかな手順を表す Main プログラムが簡潔だと非常にわかりやすくなる。そのためにサブルーチンや関数は必要不可欠な概念であることを理解してもらいたい。

Fortran でサブルーチンを使わない場合、

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C        SUBROUTINE を使わないプログラム          C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      parameter (DT=0.05,MAXMDS=150,M=1.0,K=1.0)
      double precision x,v,xnew,vnew
      double precision  s, ss, sss, ssss
      double precision  a, aa, aaa, aaaa
      integer mds

      x=1.0d0
      v=0.0d0
```

```
      do 10 mds = 1,MAXMDS,1

      a = -K/M * x;
      s = v;
      aa = -K/M * (x+s*DT/2.0d0);
      ss = v + a*DT/2.0d0;
      aaa = -K/M * (x+ss*DT/2.0d0);
      sss = v + aa*DT/2.0d0;
      aaaa = -K/M * (x+sss*DT);
      ssss = v + aaa*DT;
      vnew = v + (a+2.0d0*aa+2.0d0*aaa+aaaa)*DT/6.0d0;
      xnew = x + (s+2.0d0*ss+2.0d0*sss+ssss)*DT/6.0d0;
      v = vnew;
      x = xnew;

      write(*,*) x,v
 10   continue

      end
```

となるが、サブルーチンを使うと、

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C          SUBROUTINE を使ったプログラム          C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC


      double precision x,v

      x=1.0d0
      v=0.0d0

      call rungekutta(x,v)

      end

CCCC  SUBROUTINE  CCCC

      subroutine rungekutta(x,v)

      parameter (DT=0.05,MAXMDS=150,M=1.0,K=1.0)
      double precision xnew,vnew
      double precision  s, ss, sss, ssss
      double precision  a, aa, aaa, aaaa
```

```
      integer mds

      do 10 mds = 1,MAXMDS,1

      a = -K/M * x;
      s = v;
      aa = -K/M * (x+s*DT/2.0d0);
      ss = v + a*DT/2.0d0;
      aaa = -K/M * (x+ss*DT/2.0d0);
      sss = v + aa*DT/2.0d0;
      aaaa = -K/M * (x+sss*DT);
      ssss = v + aaa*DT;
      vnew = v + (a+2.0d0*aa+2.0d0*aaa+aaaa)*DT/6.0d0;
      xnew = x + (s+2.0d0*ss+2.0d0*sss+ssss)*DT/6.0d0;
      v = vnew;
      x = xnew;

      write(*,*) x,v
 10   continue

      end
```

となる。

また、C 言語で関数を使わない場合は、

```
#include <stdio.h>

#define  DT      0.05          /* ルンゲクッタ法の時間刻み*/
#define  MAXMDS  150          /*計算の最大値*/
#define  M       1.            /*おもりの質量*/
#define  K       1.            /*バネ定数*/

int main()
{
  double  x, v, xnew, vnew;
  double  s, ss, sss, ssss;
  double  a, aa, aaa, aaaa;
  int    mds;

  x = 1.0;                  /*初期条件の設定*/
  v = 0.0;

  for (mds = 1; mds <= MAXMDS ; mds++){
```

```
    a = -K/M * x;
    s = v;
    aa = -K/M * (x+s*DT/2.);
    ss = v + a*DT/2.;
    aaa = -K/M * (x+ss*DT/2.);
    sss = v + aa*DT/2.;
    aaaa = -K/M * (x+sss*DT);
    ssss = v + aaa*DT;
    vnew = v + (a+2.*aa+2.*aaa+aaaa)*DT/6.;
    xnew = x + (s+2.*ss+2.*sss+ssss)*DT/6.;
    v = vnew;
    x = xnew;

    printf("%f %f\n", x, v);

  }
  return 0;
}
```

となる。関数を使うと、

```
/*ルンゲクッタ法を関数にする*/

#include <stdio.h>

#define  DT      0.05          /*ルンゲクッタ法の時間刻み*/
#define  MAXMDS  150           /*計算の最大値*/
#define  M       1.0           /*おもりの質量*/
#define  K       1.0           /*バネ定数*/

void rungekutta(double,double);

int main()
{
  double  x, v;

  x = 1.0;                   /*初期条件の設定*/
  v = 0.0;

  rungekutta(x,v);  /*ルンゲクッタ法で解いて解を表示*/

  return 0;
}
```

```
void rungekutta(double x,double v)
{
  double  xnew, vnew;
  double  s, ss, sss, ssss;
  double  a, aa, aaa, aaaa;
  int    mds;

  for (mds = 1; mds <= MAXMDS ; mds++){

    a = -K/M * x;
    s = v;
    aa = -K/M * (x+s*DT/2.0);
    ss = v + a*DT/2.0;
    aaa = -K/M * (x+ss*DT/2.0);
    sss = v + aa*DT/2.0;
    aaaa = -K/M * (x+sss*DT);
    ssss = v + aaa*DT;
    vnew = v + (a+2.0*aa+2.0*aaa+aaaa)*DT/6.0;
    xnew = x + (s+2.0*ss+2.0*sss+ssss)*DT/6.0;
    v = vnew;
    x = xnew;

    printf("%f %f\n", x, v);

  }

  return;
}
```

となる。

参考文献

[1] 杉江日出澄、他著、『Fortran77 と数値計算法』(培風館、1993 年)

[2] 冨田博之、他著、『Fortran77 プログラミング』(培風館、1997 年)

[3] 小澤哲、D.W.Heermann 著、『UNIX ワークステーションによる計算機シミュレーション入門』(学術図書出版社、1995 年)

[4] 三田典玄著、『入門 C 言語』(アスキー出版局、1994 年)