コンピュータ

北海道大学理学部数学科

1　概論

これまで Python 言語を用いて計算機プログラムを作成する演習を続けてきた。プログラム言語には膨大な種類がある。Web アプリケーションに用いられる JavaScript や PHP, OS 開発に用いられる C, C++, Java など、問題に応じて適切なプログラム言語を選択するためには、異なるプログラム言語を効率的に学習する能力を身につけなければならない。

今回以降の 3 回では C 言語を学習する。C 言語の特徴は実行形式を効率よく生成できる点にある。

2　プログラムの作成と実行

C 言語のプログラムを実行するには、テキストファイルとしてソースプログラムを作成し、ソースプログラムを実行形式に変換（コンパイル）する手順を踏む。ソースプログラムのファイル名は.c で終わる名前とする。

ソースプログラムは一般的なテキストエディタ（NotePad など）を用いて作成する。

ソースファイルを実行形式に変換する処理をコンパイルとよぶ。ソースファイルのファイル名を hello.c とする。具体的には

$ `gcc hello.c -o hello` enter

というコマンドを実行する。ソースファイルに問題がなければ hello というファイル名の実行形式ファイルが作成されるので、これをコマンドとして

$ `./hello` enter

と実行する。コンパイル時に引数`-o hello` を忘れると a.out という名前で実行形式が作成される。オプション `-o` の後に実行形式のファイル名を指定する。

ソースプログラムに文法間違いなどの問題があると、gcc コマンドはコンパイル不能というメッセージを出す。ソースプログラム中の問題のある所が行番号で表示されるので、該当する行を修正し、再度コンパイルする。

2.1　具体例

次のような hello.c というソースファイルを作成し、コンパイル、実行する。

```
/* A sample program */
#include <stdio.h>
int main( )
{
  int i;
  for( i=0; i<10; i=i+1 ){
    if( 0==(i % 2) ){
      printf("Hello.");
    }else{
      printf("World.");
    }
  }
   return(0);
}
```

これは、i という整数型変数の値を 0 から 9 まで増加させる間に偶数であれば Hello. と標準出力に印字し、奇数であれば World. と印字するプログラムです。

2.2 C 言語の文法

2.2.1 注釈

/* と */ とで囲まれた部分を注釈といい、プログラムの処理には関係ない。プログラムに関する説明は注釈を利用して書くこと望ましい。時間が経てば自分も他人なので、何を目的としたプログラムかを書いておく。

2.2.2 ブロック

{ と } とで囲まれた部分をブロックとよぶ。C 言語での処理単位である。

2.2.3 関数と文

C 言語のソースプログラムは関数と文とから構成される。ここでは main( ) と printf( ) とが関数で、int i; や for(...){ } , if(...){ } などが文になる。printf 関数は標準的に組み込まれた関数であり、main 関数はユーザの作成する関数である。
ユーザの作成する関数は

```
int main( )
{

}
```

という形をとる。ここでは main が関数名、void は関数が返す値の型を示す。

main 関数は特別な関数である。C 言語ではこの関数が最初に実行されるため、必ず必要である。中括弧で囲まれた部分に文と関数を書くと、上から順番に実行されます。
もっとも単純なソースプログラムは

```
#include <stdio.h>
void main( )
{
      printf("Hello! World.\n");
}
```

などとなる。printf 関数は、引数の内容を標準出力に印字する機能を持つ。行末にセミコロンを忘れない。

コンパイルして実行すると Hello! World. と表示される。

二重引用符¨ ¨ で文字を囲み、文字列定数とよぶ。上の printf 関数では `Hello! World.` という文字列を標準出力に印字した。ここで`\n` は改行を示す。

2.2.4 変数と算術演算子

上の hello.c には、int i; という文がある。これは、単独の i という文字を整数値をとる変数の変数名として使うという宣言である。 Python とは異なり、C 言語では使用する変数名をあらかじめ定義しなければならない。これを変数名の宣言とよぶ。

C 言語の型には int, float, double, char がある。順に整数型、単精度浮動小数点型、倍精度浮動小数点型、文字型となります。変数に値を代入するには等号記号を使い、 i=0; と書く。

四則演算は算術演算子を使います。加減乗除と剰余とがそれぞれ`+,-,*,/,%` である。

2.2.5 定数

定数には数値定数と文字列定数とがある。

2.2.6 配列

int a[10]; と宣言すると、a[0] から a[9] までの 10 個の変数（配列要素）を同時に宣言したことになる。これを配列とよぶ。配列の添字は必ず 0 から開始する。配列は要素ごとに代入・演算する。

2.2.7 制御構造

制御構造には if 文と for 文、while 文がある。

関係演算子と and or とは次のように書く。a == b は a が b と等しい場合に真となり、異なる場合に偽となる。<, >, <=, >= は見ての通りの関係演算子である。! = は等しくない場合に真となる。and は (a == b) && (c == d) と書く。or は (a == b) || (c == d) である。

if, while, for は次のように使う。

```
if( 条件式 ){
  条件式が真の場合の処理
}else{
  条件式が偽の場合処理
}
```

```
while( 条件式 ){
  条件式が真の場合の間の処理
}

do {
  条件式が真の場合の間の処理
}while( 条件式 )

for( 式 1 ; 条件式 ; 式 2 ){
  条件式が真の場合の間の処理
}
```

for 文は、まず式 1 を実行してから条件式の真偽を評価する。真であれば中括弧の中を実行して式 2 を実行する。条件式が真であれば再び中括弧の中を実行し, 式 2 を実行する。これが繰り返される。

2.2.8 演算の優先順位

演算には優先順位がある。条件式よりも算術演算の方が優先順が高くなり、括弧を使うとわかりやすい。不安な時はとにかく括弧でくくる。

2.2.9 標準入出力

printf 関数は標準出力へ出力する。変数の値を出力するには次のようにプレースホルダーを用いる。

```
printf( "%d %e %c", i, t, c);
```

ここで、最初の引数に文字列型で印字形式を指定します。これをフォーマット文字列あるいはプレースホルダーとよぶ。%d は整数型の印字、%e は実数型の印字、%c は文字型の印字である。出力する変数はフォーマット文字列の中に出現する順に引数へ記述する。

標準入力（キーボード）から変数へ値を読み込むには scanf 関数を使う。scanf のプレースホルダーも printf と同様である。

```
scanf( "%d", &i );
scanf( "%lf", &t );
scanf( "%c", &c );
```

これで、それぞれ i, t, c にキーボードから値を入力する。i,t,c の型は printf と同様である。プログラム中に scanf 関数が現れるとそこでキーボード (標準入力) からの入力を待つ。

複数の値を同時に入力するには

```
scanf( "%d %d", &i1, &i2 );
```

などと記述する。入力時にはスペースで区切る。

2.2.10 インクルードファイル

組み込み関数の型宣言はインクルードファイルに収められており、読み込むには

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

などとする。カレントディレクトリのインクルードファイルを読み込むには

```
#include "file.h"
```

とする。

2.3 数学関数

C 言語で使える数学関数は、math.h というインクルードファイルに記述されている。数学関数を使うにはコンパイルの際に -lm を指定する必要がある。これは、libm.a というライブラリをリンクするという意味である。

次の例では、指数関数 exp( ) を使って e^x の値を $x = 0.0$ から $x = 1.0$ まで 0.1 きざみで計算している。

```
/*
   A sample C program, part 2
   Filename: hello2.c
   gcc hello2.c -o hello2 -lm
*/
#include <stdio.h>
#include <math.h>
void main( )
{
  int i;
  double x,y;

  x = 0.0;
  for( i=0; i<10; i=i+1 ){
    x = x + 0.1;
    y = exp(x);
    printf( "%e %e\n", x, y );
  }
}
```

次のプログラムは実係数の二次方程式の解を出力するプログラムである。

```
/*
  二次方程式 ax^2+bx+c=0 の解を求める
  File:    quad.c
  Compile: gcc quad.c -o quad -lm
*/

#include <stdio.h>
#include <math.h>
void main( )
{
  double a,b,c;                 /* 各項の係数 */
  double D,s1,s2;               /* 判別式と二つの解 */
```

```
  a = 1.0;
  b = 3.0;
  c = 2.0;

  D = b*b-4*a*c;

  s1 = (-b+sqrt(D))/(2*a);
  s2 = (-b-sqrt(D))/(2*a);

  printf( "%e %e\n", s1,s2 );
}
```

3 Newton 法

Newton 法は解析的に求まらない方程式の解を数値的に求める際の基本的な解法である。これを C 言語でプログラムする。

3.1 プログラム

Newton 法に従って $x^2 - 2 = 0$ の正値解を求めるプログラムを組むと次のようになる。fabs 関数は引数の絶対値を返す。

```
/*
  newton.c  Newton's method
  gcc newton.c -o newton -lm
*/
#include <stdio.h>
#include <math.h>

void main( )
{
  double x,y,eps,err;  /* eps は許容誤差、err は各段階の誤差 */

  x=10.0;              /* 初期値の設定 */
  eps = 0.00001;       /* 許容誤差の設定 */

  do{
    y = x - (x*x-2.0)/(2.0*x);  /* Newton 法の式 */
    err = fabs(x-y);                  /* 現在の誤差 */
    printf( "%e %e error=%e\n", x, y, err );
    x = y;
  }while( err > eps );         /* err > eps の間は do-while の中を繰り返し */
}
```

3.2 プログラムのブロック化

関数を新たに定義して書き直す。次のようなプログラムになる。

```
/*
  newton.c  Newton's method
```

```
  gcc newton.c -o newton -lm
*/
#include <stdio.h>
#include <math.h>

double f( double x)   /* 関数 f の宣言 */
{
  double y;           /* 局所変数 y の宣言 */

  y=x*x-2.0;
  return(y);          /* y=x^2-2 を関数値として返す */
}

double Df( double x )
{
  return( 2.0*x );    /* y=2x を関数値として返す */
}

void main( )
{
  double x,y,eps;  /* eps は許容誤差、err は各段階の誤差 */

  x=10.0;              /* 初期値の設定 */
  eps = 0.00001;       /* 許容誤差の設定 */

  /* f(x) > eps の間は while の中を繰り返し */
  while( fabs(f(x)) > eps ){
  y = x - f(x)/Df(x);  /* Newton 法の式 */
  printf( "%e %e error=%e\n", x, y, x-y);
  x = y;
  }
}
```

$|f(x)| < eps$ を計算終了の条件とした。

Newton 法の漸化式を明確にするため、$f(x)$ とその導関数を C 言語の関数として定義した。関数の内部で宣言した変数は局所変数と呼び、他で同じ変数名を使っても独立に使える。

4 常微分方程式の差分法による数値解法

プログラムでは f と初期値を決めなければならないので、ここでは $f(x) = 2x$ とする。

```
/*
  Euler 法
  euler.c
  gcc euler.c -o euler -lm
*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define Max 100  /* Max という定数を 100 と定義 */
double x[Max];   /* x[0] から x[99] までの配列 x を定義 */
double dt;       /* 刻み幅 */

double f( double );
```

```
void main( int argc, char *argv[] )
{
  int i;

  x[0]=1;

  for( i=0; i<Max-1; ++i ){
    x[i+1]=x[i]+f(x[i])*dt;
  }

  for( i=0; i<Max-1; ++i ){
    printf( "%d %e\n", i, x[i] );
  }
}

double f( double x )
{
  return( 2.0*x );
}
```

4.1 define 文

`#define Max 100` という部分は、Max という文字列を以後 100 で置き換えるという意味である。従って、Max を定数と思って使える。

参考文献

[1] B.W. Kernighan, D.M. Ritchie, 石田 晴久 訳 「プログラミング言語 C 第 2 版 ANSI 規格準拠」(共立出版)

[2] William H. Press 他「ニューメリカルレシピ・イン・シー 日本語版 – C 言語による数値計算のレシピ」(技術評論社)

[3] 柴田 望洋 「明解 C++」(ソフトバンククリエイティブ)