数値処理

電子情報工学科
加藤友彦

[第１部]　Ｃ言語 & vi エディタ

[1] Ｃ言語の復習

すでに，皆さんは１年次でＣを習ってきていますので，ここでは簡単なプログラムを例にして，文法のおさらいをしましょう．

簡単な例題

１から 100 までの１００個の数の総和と平均値を計算するプログラム

```
/* total and average */
#include <stdio.h>
main()
{
 FILE *fp1;                                  /*  fairu no teigi */
 int i;                                      /*  hennsuu no katasengen */
 int sum=0;
 int nn=100;
 float ave;                                  /*  hennsuu no katasengen no owari */
 fp1=fopen("data","w");                      /*  file no open */
  for(i=1;i<=nn;i=i+1)                       /*  kurikaesi bun */
   {
    sum=sum+i;
   }                                         /*  kurikaesi bun no owari */
 ave=(float)sum/(float)nn;                   /*  heikinti no keisan */
 printf("total=%d  average= %f \n",sum,ave);      /*  console eno print */
 fprintf(fp1,"total=%d  average= %f \n",sum,ave);   /*  file eno kakikomi */
 fclose(fp1);                                /*  file no close */
}                                            /*  main program no end */
```

1. 定数

整定数
　　0, -1234, 99999
実定数
　　123.456, 12.3456e-12
文字定数
　　'a','b'　（１文字のみ）

2. 変数

英字で始まる　桁以内の英数字(_も可）
　　a, ab, c1, result_1, SUM, SUM2　（大文字と小文字は区別される）
すべての変数は型宣言をしなければならない．

3. 型宣言文

整数型　int, short int, long int
実数型　float,double（倍精度）
文字型　char

4. 配列

int a[3]　/* 整数型でa[0],a[1],a[2] の３変数の定義 */
float abc[4]　/* 実数型で abc[0] から abc[1] までの４変数の定義 */
int xyz[10][[10]　/* ２次元配列，整数型で xyz[0][0] から xyz[9][9] までの100変数の定義 */
３次元以上の多次元の配列も[　]を付け加えることにより定義できる．ただし，所要バイト数が大きくなりすぎないよう注意する必要がある．

5.算術演算子および代入演算子

算術演算子

加算：a＋b　　減算：a－b
乗算：a＊b　　除算：a／b
剰余：a%b　（a，bは整数型でなければならない）

代入演算子

a += b　a=a+b　　a -= b　a=a-b
a *= b　a=a*b　　a /= b　a=a/b
a %= b　a=a%b

特に, b=1 のときは，a += 1 は a++，a -= 1 は a-- と略記することができる．

6. 条件文

単純な if 文

if(a) b;
a：条件（論理式）
<, <=, ==, >=, >, != ，　（論理式での等号は ==であって =ではない！）
二つの論理式(a1,a2)の結合：（論理積）a1 && a2，（論理和） a1 || a2
a が真の真のとき b（文）．

if---else 文

・if(条件)
文１;
else
文2 ;　　条件が真のとき文１を実行，偽のとき文２を実行．
・複数の文が必要なときは { }で囲む．
if(条件)　{
文１;文２;}
else{
文３，文４;}
・条件の分岐
if(条件１)　{ 文１}
else if(条件２)　{ 文２;}
else{ 文３;}

switch---case 文

```
switch( 整数式 )
 {
  case n1 : 文１;
   break;
  case n2 : 文２;
   break;
  case n3 : 文３;
```

```
      break;
    }
```

7. 繰返し演算

for 文

```
 for(i=1; i<=100; i=i+1) {          /* 初期値，終了の条件式，増分値 */
    文１;
    文２;
    文３;
  }
```

while 文

```
  while(繰返しの条件式){
    文１;
    文２;
    文３;
  }          文が１つのときは，{   } は省略可．
```

do ----- while 文

```
  do{
    文１;
    文２;
    文３;
  }
  while(繰返しの条件式);
```

多重の繰返し

```
 for(  ;   ;  ) {文１; for(  ;   ;  ){
    文２;
    文３;
  }}
```

8. 入出力

scanf( )関数による数値の入力

例：　scanf("%d %f", &number,&value)

　この文は二つの変数numberとvalueに数値を読み込ませるものです．"　　"は変数の型を指定するところで，%d は整数型，%f は実数型を意味します．変数の前には &をつける決まりになっています．

printf( )関数による数値の出力

例：　printf ("number of data=%d　　result=%f　　error=%.5e\n", number,value,err)

"　　"は書式を指定する文字列で,%d は整数型，%f は実数型を指定しています．number of data= は文字として打ち出され，その後にnumberの値が出力されます．次に　　result= が書かれ，その後にvalueの値が出力されます．最後に error=が打ち出され，その後にerrの値が指数表示で少数点以下5桁で出力されます．\n は改行の記号です．なお，printf の場合は，変数の前に &をつけません．

　型指定としてこれ以外に良く使われるものとしては，　　%s　文字列　があります．

9. ファイルのオープンとファイルへの書込み

例：

```
FILE  *fp                                   /*  file no teigi ，FILEは大文字の決まり*/;
  変数の型宣言
fp=fopen("data","w");                       /*  file no open */
  実行文
fprintf(fp,"書式を指定する文字列",変数名);  /*  file eno kakikomi */
                    /* %d, %ld・・・整数型， %f, %lf・・・実数型 */
fclose(fp);                                 /*  file no close */
```

ここで "data" はファイルの名前が data であること，"w"は読込み専用であることを意味します．また，fpはこのファイルのラベルの役目を果たしています．

10. 標準数学関数

Ｃでは良く使われる指数関数や三角関数を標準数学関数として用意しています．これらを利用するには，プログラムの冒頭で
#include <stdio.h> の下に
#include <math.h> の文を入れなければなりません．
例：

```
  double a1,a2;
  a1=sqrt(a2);
```

ここで，sqrt は平方根を求める標準数学関数です．注意するべきことは，関数の引数も関数値も倍精度実数でなければならないことです．１０年程前までは，実数は特に必要でない限り単精度で定義して計算時間を節約するのが常道とされたのですが，現在の処理系では，倍精度演算が標準になっており，単精度の方が時間がかかってしまいます．従って，実数型変数は倍精度を標準として使ってください．ただし，出力時は見やすくするために必要な有効数字までに桁数を制限指定をする等の工夫が必要です．

<u>主な標準数学関数</u>

cos(x) ,　　sin(x) ,　　tan(x) ,
acos(x)　[$\cos^{-1}(x)$] ，asin(x)　[$\sin^{-1}(x)$] ，atan(x)　[$\tan^{-1}(x)$],
cosh(x) ,　　sinh(x) ,　　tanh(x) ,
exp(x)　[e^{x}] ,　pow(x,y)　[x^{y}] ,　sqrt(x) ,
log(x)　[ln(x)] ,　log10(x)　[$\log_{10}(x)$] ,
fabs(x)　[$|x|$] ,
ceil(x)　[xより小さくない最小の整数]
floor(x)　[xより大きくない最大の整数]

Ｃにはべき乗を表現する簡単な記号がないので，標準関数pow(x,y)を使わなければなりません．
しかし，x^2 やx^3 は，pow(x,y)を使うよりは x*x，x*x*x とする方が良いでしょう．

[2] ｖｉエデイターの使い方

(1) 新しいファイルを作る

例

vi problem1.c　　problem1.c はファイル名，C のプログラムは必ず .c の拡張子をつける．

(2) ３つのモード

・コマンドモード　　カーソルの移動，削除，挿入などのコマンドを入力するモード．ｖｉ起動後はまずこのモードに入る．

・テキスト入力モード　　文字入力のモード．コマンドモードで **a** とおすとこのモードに入る．**esc** を押すとコマンドモードに戻る．

・コロンモード　　コマンドモードで ： を入力すると最終行にカーソルが移動する．そこで，**wq**（セーブ後終了),または **q!**（そのまま終了）として終了する．**set nu** とすると行番号を表示します．

(3) コマンドモードにおける主なコマンド

・挿入	**i**	カーソルの前に挿入
	a	カーソルの後に挿入
・カーソルの移動	**h　j　k　l**	（矢印キーが使えることが多い）
・文字の削除等	**x**	カーソルの位置の文字の一字削除
	3x	カーソルの位置以降の３文字の削除
	r	カーソルの位置の文字を入力する文字に置換
	xp	カーソルの位置の文字と次の文字の交換
	dd	カーソルの位置の行の削除
	D	カーソルから右の部分の削除
	3dd	カーソルの位置以降の３行の削除
	yy	カーソルの位置の行のコピー
	3yy	カーソルの位置以降の３行のコピー
	p	コピーまたは削除された行を現在行の下に挿入
	P	コピーまたは削除された行を現在行の上に挿入
	o	現在行の下に１行空ける
	O	現在行の上に１行空ける

プログラムのコンパイル

・ｖｉでプログラムを作った後（**esc　：　wq** で抜けて）

ipc09% **gcc** problem1.c と入力する．ここで problem1.c のところは各自　　のファイル名であ

る．#include <math.h> を宣言しているときは

ipc09% **gcc** problem1.c **-lm** とすること．

コンパイルがエラーなく完了すると，ipc09% が出る．エラーがあると，　エラーメッセージが出るので，ｖｉに戻ってプログラムを修正し，再度，コンパイルする．

プログラムの実行

・ipc09% **a.out**

と入力する．正常終了すると，結果が打ち出される．

実行結果を入れるファイルの作成

```
FILE *fp                          /* file no teigi */;
fp=fopen("result1","w");           /* file no open （宣言文の下に書く）*/
・・・・・
fclose(fp);                        /* file no close （最後に書く）*/
```

ファイルのコピーと結合

ipc09% **cp** problem1.c report1・・・problem1.c を複製し，report1と名前をつける．

ipc09% **cat** result1 **>>** report1・・・result1を report1の後ろに結合

ファイルの送信

ipc09% **mail katou@fit.ac.jp <** report1

演習問題 １　（Ｃの最も簡単な問題）

ディスプレイに Hello World と表示する．

演習問題 ２

「C言語の復習」の「簡単な例題」を vi editor で打ち込み，コンパイル，実行，プログラムと結果の結合を行い，そのファイルを加藤まで送信せよ．

[第２部]　基本的数値計算法

[1] 平均値と標準偏差

対象とするある性質を表す変数 x に対して，Ｎ個のデータ：$x_1, x_2, \cdots x_N$の集合があったとする．この集合の平均値 <x>，分散 S^2，標準偏差 S は次の式で与えられる．

平均値

$$< x > = \frac{1}{N}\sum_{i=1}^{N} x_i$$

分散

$$S^2 = \frac{1}{N}\sum_{i=1}^{N} (x_i - < x >)^2$$

標準偏差

$$S = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (x_i - < x >)^2}$$

また，個々のデータが与えられていなくて，度数分布のみが与えられているとき（ｐ個の区間に分ける），平均値と標準偏差は下記の式で計算される．

$$< x > = \frac{1}{N}\sum_{i=1}^{p} m_i f_i$$

$$S = \sqrt{\frac{1}{N}\sum_{i=1}^{p} (m_i - < x >)^2 f_i}$$

ここで，m_i は ｉ 番目の区間の階級値（各区間の中央値）である．

証明問題

$S = \sqrt{< x^2 > - < x >^2}$　と表されることを証明せよ．

例題 1　自然数 2 から100 までの５０個の偶数の集合がある．このデータの平均値と標準偏差を求めよ．

解答例

```
/* toukei syori */
#include <stdio.h>
#include <math.h>
main()
{
 FILE *fp;
 int sum1=0,sum2=0;
 int i,ndata=50;
 double ave1,ave2;
 double s1,s2;
 fp=fopen("result2","w");
 for(i=2;i<=100;i=i+2)
{
      sum1=sum1+i;
      sum2=sum2+i*i;
  }
 ave1=(double)sum1/(double)ndata; ave2=(double)sum2/(double)ndata;
 s2=ave2-ave1*ave1; s1=sqrt(s2);
```

```
 printf("heikin= %.3f   hensa= %f \n",ave1,s1);
 fprintf(fp,"heikin=%.3f   hensa= %f \n",ave1,s1);
 fclose(fp);
}
```

```
  heikin=51.000000   hensa= 28.861739
```

演習問題 3

Aグループ，Bグループ　各１０人の所持金を調べたところ下記の様であった.

A　200, 600, 1000, 3500, 3800, 5000, 5200, 7300, 9000,13600
B　2500, 3600, 3800, 4100, 5100, 5300, 5500, 5900, 6600, 6800

それぞれの平均値，標準偏差を求めよ.

演習問題　4

ある小学校の４０人のクラスの身長の度数分布は以下の様であった.

100-105	1名
105-110	3
110-115	3
115-120	4
120-125	6
125-130	9
130-135	5
135-140	4
140-145	4
145-150	1

平均値と標準偏差を求めよ.

［２］最小二乗法

いくつかの種類の量を測定し，この間の関係式を求める有力な方法である．あらかじめ各変数間の関数関係が分かっている場合に使われることが多い．簡単な場合について説明するが，これらを理解すれば，より複雑な場合への拡張も容易である．

A　１種類の量の最確値

ある量Ｘをｎ回測定し，データ $x_1,x_2,\cdots x_n$ を得た．これから，最確値を求めるにはどうすれば良いか．Ｘの最確値を X_0（真値は分からない）とし，各データとの差 $v_i=x_i-X_0$（これを残差と呼ぶ）を考える．v_i の発生は全く偶然であると仮定すると，残差が v_i と v_i+dv_i の間の値を取る確率は，誤差論により正規分布によって表されることが分かっている.すなわち　　$f(v_i)dv_i = \left(2\pi\sigma^2\right)^{-1/2}\exp\left(-v_i^2/2\sigma^2\right)dv_i$

ここで，σは標準偏差でこの分布の幅を表わす．ｎ回の測定で一連の残差 $(v_1,v_2,\cdots v_1)$ が発生する確率密度は毎回の測定が独立であると仮定して次式で与えられる．

$$f(v_1,v_2,\cdots v_n) = \left(2\pi\sigma^2\right)^{-n/2}\exp\{-(v_1^2+v_2^2+\cdots v_n^2)/2\sigma^2\}$$

われわれが，実際の観測において，この一連の残差を測定するのは，この確率が最も大きな場合に当たっていると考えられる．このことは，上式の指数部Ｓが最小になっていることを意味する． $S=(v_1^2+v_2^2+\cdots v_n^2)$

と定義すると

$$\frac{dS}{dX_0} = 2\sum_{i=1}^{n}(x_i-X_0) = 0$$

$$X_0 = \frac{1}{n}\sum_{i=1}^{n} x_i$$

となる．これは，算術平均にほかならない．この説明は，算術平均が最確値を与えることの根拠を与えるものである．

B　２種類の量の関係

測定される量ＸとＹの間に，１次式の関係があることが分かっているとしよう．

$Y=aX+b$　　{X,Y} の例として，{温度,金属棒の長さ}，{バネに吊す重りの重量,バネの伸び}，{温度,電気抵抗}などがあげられる．

ｎ組の測定データ，$(x_1,y_1),(x_2,y_2),\cdots(x_n,y_n)$　が得られたとき，最も確からしいａとｂの値を最小二乗法で求めることができる．ｉ番目のデータについて，Ｙの測定値と最確値　ax_i+b　との差を残差と定義し，その二乗の和をＳとする．すなわち，

$$S = \sum_{i=1}^{n}\{y_i-(ax_i+b)\}^2$$

この量が，ａとｂについて最小であることを要請すると，

$$\frac{dS}{da} = 0 \quad \rightarrow \quad \sum_{i=1}^{n}\{y_i-(ax_i+b)\}x_i = 0$$

$$\frac{dS}{db} = 0 \quad \rightarrow \quad \sum_{i=1}^{n}\{y_i-(ax_i+b)\} = 0$$

この２式より，

$[x^2]a+[x]b=[xy]$

$[x]a+nb=[y]$　を得る．ここに $[x^2]=\Sigma x_i^2$, $[x]=\Sigma x_i$, $[xy]=\Sigma x_i y_i$ である．

この連立方程式を解いて，

$$a = \frac{n[xy]-[x][y]}{n[x^2]-[x]^2}, \quad b = \frac{[x^2][y]-[x][xy]}{n[x^2]-[x]^2}$$

となる．

測定される量ＸとＹの間に２次式 Y=aX^2+bX+c が成立する場合についても，上記と同様に考えればａ，ｂ．ｃの最確値を与える表式を得ることが出来る．すなわち，

$$S = \sum_{i=1}^{n} \{y_i-(ax_i^2+bx_i+c)\}^2$$

について，dS/da=0, dS/db=0, dS/dc=0 の３式から

[x^4]a+[x^3]b+[x^2]c=[x^{2}y]

[x^3]a+[x^2]b+[x]c =[xy]

[x^2]a+ [x]b + nc= [y]

と，ａ，ｂ．ｃについての３元連立方程式を得る．これを解けばよい．

また，変数ＸとＹの関係が１次式でなくても，適当な変数変換によって１次式に帰着させることが出来る場合がある．以下に示す関係式は，適用例の多いものである．

y=ax^3+bx → y/x=ax^2+b y'=y/x, x'=x^2

y=bx^a → log(y)=alog(x)+log(b) y'=log(y), x'=log(x), b'=log(b)

y=be^{ax} → log(y)=ax +log(b) y'=log(y), x'=x, b'=log(b)

以上の方法は３変数以上の場合についても容易に拡張できる．

例題２（最小二乗法）

金属の低温における定積比熱 C_V は絶対温度 T を変数として $C_V = \gamma T + \beta T^3$ と表されることが知られている．以下に示すのは，金属カリウムについての測定結果である．最小自乗法を用いてγとβを求めよ．

T	C_V
0.2604	0.5852
0.3478	0.8362
0.4515	1.177
0.5435	1.551
0.6414	2.027
0.7236	2.511

解答例

```
/*---------specific heat----------*/
#include <stdio.h>
main()
{
FILE *fp;
double c[6]={0.5852,0.8362,1.177,1.551,2.027,2.511};
```

```
double t[6]={0.2604,0.3487,0.4515,0.5435,0.6414,0.7236};
double x[6],y[6];
double sumx=0.0, sumy=0.0, sumx2=0.0, sumxy=0.0;
double dd,bb,gg;
int i, ndata=6;
/* sengen owari */
fp=fopen("data2","w");
for (i=0; i<=5;i++)
 {
     y[i]=c[i]/t[i];
     x[i]=t[i]*t[i];
 }
for (i=0; i<=5;i++)
{
 sumx=sumx+x[i];
 sumy=sumy+y[i];
 sumx2=sumx2+x[i]*x[i];
 sumxy=sumxy+x[i]*y[i];
 }
    dd=ndata*sumx2-sumx*sumx;
    bb=(ndata*sumxy-sumx*sumy)/dd;
    gg=(sumx2*sumy-sumx*sumxy)/dd;
  printf("behta= %f    ganma= %f\n",bb,gg);
  fprintf(fp,"behta= %f    ganma= %f\n",bb,gg);
  fclose(fp);
}
```

演習問題 5

ある金属棒の電気抵抗Ｒの温度変化を測定して次の結果を得た．R(t) を表す実験式を最小二乗法によって求めよ．なお，常温では，金属の電気抵抗は温度にほぼ比例して増加することが知られている．

t(C)	14.0	21.0	25.1	31.0	35.0	41.1	45.0
R(Ω)	61.04	62.24	65.40	66.26	65.88	67.12	69.78

演習問題 6

ある物質が時間とともに減衰する現象を測定した．物質の量をｍ(g)，時間をｔ(sec)とすると，m=b exp(-t/T) の関係があることが理論的に分かっている．測定データは下記のとおりである．b およびT（時定数）を最小二乗法によって求めよ．

t(sec)	1.0	2.0	3.0	4.0	5.0
m(g)	90.1	36.4	14.8	6.0	2.3

［３］補間法と数値積分法

補間法

関数関係 y=f(x) において，いくつかのｘについてのみｙの値が与えられているとき，言い換えれば，ｘとｙの関係が数表の形で与えられているときに，この関数形を推定し，任意のｘに対するｙの値を求めることを補間と呼ぶ．関数のタイプについては，あらかじめ分かっていない場合が多く，通常は多項式を用いる．代表的な補間法として以下のようなものがある。

(1) 補間多項式
(2) ニュートンの補間公式
(3) ラグランジュの補間公式
(4) エルミートの補間公式
(5) スプライン補間

(1) 補間多項式

ｎ個データの点 $(x_1,y_1), (x_2,y_2), \cdots (x_n,y_n)$ を通る（ｎ－１）次式 $P_{n-1}(x)$ を求める．

$$P_{n-1}(x) = a_1+a_2x+a_3x^2+\cdots+a_nx^{n-1}$$

この曲線上にｎ個の点が乗っていることから，

$$y_1=a_1+a_2x_1+a_3x_1^2+\cdots+a_nx_1^{n-1}$$

$$y_2=a_1+a_2x_2+a_3x_2^2+\cdots+a_nx_2^{n-1}$$

$$\cdots\cdots\cdots\cdots\cdots\cdots$$

$$y_n=a_1+a_2x_n+a_3x_n^2+\cdots+a_nx_n^{n-1}$$

のように $a_0,a_1.,a_2+\cdots,a_{n-1}$ についてのｎ元の連立１次方程式が得られる．この方程式を解いて，各係数 a_i が求められる．については、講義で説明する。

(2) ニュートンの補間公式

この方法は，(1)の補間多項式と本質的には同じであるが，多項式の形を変えて，係数を求め易く工夫したものである．補間多項式を次の形で表す．

$$P_{n-1}(x) = b_1+b_2(x-x_1)+b_3(x-x_1)(x-x_2)+\cdots+b_n(x-x_1)(x-x_2)\cdot\cdot(x-x_{n-1})$$

このとき，展開係数 b_i を決める連立方程式は，

$$y_1=b_1$$

$$y_2=b_1+b_2(x_2-x_1)$$

$$\cdots\cdots\cdots$$

$$y_n=b_1+b_2(x_n-x_1)+b_3(x_n-x_1)(x_n-x_2)+\cdots+b_n(x_n-x_1)(x_n-x_2)\cdot\cdot(x_n-x_{n-1})$$

となり，上から順に $b_1,b_2,b_3,\cdots b_n$ を求めることができる．

(3), (4), (5) については講義で説明する．

数値積分法

解析的に計算できない積分、あるいは被積分関数が有限個の点しか与えられていない場合は、数値的に積分

を計算することになる。その場合、積分可能な関数（通常は多項式）によって、数点を補間し、その区間の積分値を計算し、これを繰り返して足し合わせ全区間の積分値を求める。このことから、数値積分と補間法は密接な関係があることが分かるであろう。

$$S = \int_a^b f(x)dx$$

について以下説明する．

(1) 台形公式

２点を１次式で補間して、各区間を台形と近似し、面積を足し合わせるものである。

区間 [a,b] をｎ等分すると，分割の幅ｄは (b-a)/n である． ｉ番目のｘ座標をxn とすると

$x_i = a + i*d$ と表される．台形公式は

$$S = \{(f(a)+f(b))/2+(f(x_1)+(f(x_2)+\cdots+(f(x_{n-1}))\}*d$$

で与えられる．証明は講義で行う。（簡単なので各人で確かめると良い）

(2) シンプソン公式

区間を偶数個に細分し、各３点を２次式で補間して、その積分を足し合わせるものである。

区間 [a,b] をｎ等分（ｎは偶数）すると，分割の幅ｄは (b-a)/n である． ｉ番目のｘ座標をxn とするとxn = a + i*d と表される．シンプソンの近似式は n=2*m とおいて，

$$S = \{(f(a)+f(b))+2*(f(x_2)+(f(x_4)+\cdots+(f(x_{2m-2})) +4*(f(x_1)+(f(x_3)+\cdots+(f(x_{2m-1}))\}*d/3$$

で与えられる．（ここで，２ｍはｎと置き換えても良いことに注意せよ）証明は講義で行う

例題３（数値積分）

$S = \int_0^1 4\sqrt{(1-x^2)}\,dx$　を数値積分によって求めよ．

(1) 台形公式を用いる．分割数は 20, 100, 500 の３通りで計算せよ．

(2) シンプソン公式を用いる．分割数は 20, 100, 500 の３通りで計算せよ．

解答例

(1) 台形公式

```
/*--- reidai3 suuti-sekibun -----*/
#include <stdio.h>
#include <math.h>
double f(double x)
  {
  double y;
  y=4.0*sqrt(1-x*x);
  return(y);
  }
main()
{
FILE *fp;
double a=0.0,b=1.0;
int n[3]={20,100,500};
double sum,sum1,sum2;
double d,x,area;
int i,j;
```

```
fp=fopen("data3","w");
/* daikei kousiki */
for(i=0;i<=2;i++)
  {
    d=(b-a)/n[i];
    sum=(f(a)+f(b))/2.0;
    for(j=1;j<=n[i]-1;j++)
      {
        x=a+j*d;
        sum=sum+f(x);
      }
    area=d*sum;
    printf("daikei  n=%d  menseki=%f\n",n[i],area);
    fprintf(fp,"daikei  n=%d  menseki=%f\n",n[i],area);
  }
/* Sympson kousiki */
for(i=0;i<=2;i++)
  {
    d=(b-a)/n[i];
    sum1=0.0;
    sum2=0.0;
    for(j=2;j<=n[i]-2;j=j+2)
      {
        x=a+j*d;
        sum1=sum1+f(x);
      }
    for(j=1;j<=n[i]-1;j=j+2)
      {
        x=a+j*d;
        sum2=sum2+f(x);
      }
    area=(f(a)+f(b)+sum1*2.0+sum2*4.0)*d/3.0;
    printf("Sympson  n=%d  menseki=%f\n",n[i],area);
    fprintf(fp,"Sympson  n=%d  menseki=%f\n",n[i],area);
  }
fclose(fp);
}
```

```
daikei  n=20  menseki=3.128465
daikei  n=100  menseki=3.140418
daikei  n=500  menseki=3.141486
Sympson  n=20  menseki=3.136447
Sympson  n=100  menseki=3.141133
Sympson  n=500  menseki=3.141552
```

参考：上記の定積分の値は π (=3.141592654)

演習問題 7

$$\sqrt{\frac{2}{\pi}}\int_0^a e^{-x^2/2}\,dx$$ この積分の値を台形公式を用いて，a=1, 2, 3, 5 の場合について計算せよ．（分割数は 100ａとせよ．）

演習問題 8

同じことをシンプソン公式を用いて行え.

演習問題 9

定積分

$$\int_0^{\pi} \exp(-2x)\sin(x)\ln(1+x^2)\,dx$$

をシンプソン公式で求めよ．分割数を 10, 50, 100, 500 の 4 通りの場合について計算し，結果を比較せよ.

[4] 方程式の解法

２次方程式には，解の公式がある．複雑な式であるが，３次，４次までは解の公式がある．しかし，５次以上の代数方程式には解の公式を作ることができないことが証明されている．また，三角関数，対数関数，指数関数などの超越関数を含む方程式は，簡単なものを除いて，解析的に解を求めることはできない．このことは解が存在しないことを意味しない．このような場合に解を求めるには，数値的な解法によるしかない．以下に， f(x)=0 の解を近似的に求める方法のうち，代表的な３つの方法を記す．

１ はさみ打ち法

x=a と x=b　の間に解があることが分かっているとき（f(a)・f(b)<0），近似解として (a,f(a)) と (b,f(b)) を通る直線がｘ軸と交わる点　c=(b f(a) - a f(b))/(f(a) - f(b)) をとる．これを要請される精度まで繰り返して解を求める．

2 単純反復法

f(x)=0 を適当に変形して，x=g(x) の形にする．適当な初期値 x_1を選び，$x_2=f(x_1)$, $x_3=f(x_2)$, ・・・と収束するまで繰返す．初期値を含む解の近傍で $|f'(x)|<1$ であれば収束する．最初の方程式の形をこの条件を満たすように選ぶことが必要である．

3 ニュートン法

はじめに解の近くの点をとり，その点における接線がｘ軸と交わる点を次の近似解として，
これを要請される精度まで繰り返して解を求める．すなわち，

$$x_{n+1} = x_n - f(x_n)/f'(x_n)$$

初期値の選び方が大切である．

例題 4

exp(x) = 4 x の解を有効数字４桁までニュートン法を用いて求めよ．誤差および収束までの回数も記録せよ．初期値を x=0.5の場合と x=2.0 の場合の二通りについて実行せよ．

解答例

```
/*--- reidai4 Newton method -----*/
#include <stdio.h>
#include <math.h>
double f(double x)
  {
  double y;
  y=exp(x)-4.0*x;
  return(y);
  }
double g(double x)
  {
  return(exp(x)-4.0);
  }
main()
{
FILE *fp;
double x1,x2;
double f1,g1,er;
double x0[2]={0.5,2.0};
int n,i;
/*--------------*/
```

```
fp=fopen("data4","w");
for(n=0;n<=1;n++){
 x1=x0[n];
 i=0;
 er=1.0;
while(er>1.0e-4){
   i=i+1;
   f1=f(x1);g1=g(x1);
   x2=x1-f1/g1;
   er=fabs((x2-x1)/x1);
   x1=x2;
      }
 printf("shokichi=%f  syuusokuti=%f  gosa=%f  kaisuu=%d\n",x0[n],x2,er,i);
 fprintf(fp,"shokichi=%f  syuusokuti=%f  gosa=%f  kaisuu=%d\n",x0[n],x2,er,i);
 }
 }
```

計算結果

```
shokichi= 0.500000  syuusokuti=0.357403  gosa=0.000036  kaisuu=3
shokichi= 2.000000  syuusokuti=2.153292  gosa=0.000000  kaisuu=4
```

演習問題 10

$\exp(x) + x^2 = 3$ の正の解を有効数字６桁まで ニュートン法によって求めよ．（初期値は適当に選ぶこと）誤差および収束までの回数も記録せよ

[5] 微分方程式の解法

１階の常微分方程式

$$\frac{dy}{dx}=f(x,y)$$

を初期条件 $y(x_0)=y_0$ の解を数値的に求める方法を説明する．上式と等価な式

$$y = y_0 + \int_{x_0}^{x} f(\xi,y(\xi))d\xi$$

に変形して考える．右辺は解かるべきy を含んでいるが，x を少しずつ増やしていき，各段階でy に近似的な値を代入して積分を評価すれば，y(x) が求められる．

普通の方法は一段階法と呼ばれるもので

$y_{n+1} = y_n + \Phi(x_n,y_n)\cdot h$　　の形で 積分の中のyを近似する．ここで，はの区間を等間隔に区分したときの区分の幅であり，$\Phi(x_n,y_n)$は近似の段階に応じて以下のように与えられる．

オイラー法

$\Phi(x_n,y_n) = f(x_n,y_n)$

誤差はh^2 のオーダー

ルンゲクッター（２次）法

$\Phi(x_n,y_n) = 1/2\{f(x_n,y_n) + f(x_n+h\ ,\ y_n + h\ f(x_n,y_n))\}$

誤差はh^3 のオーダー

変数が２つ以上の連立微分方程式および２階以上の常微分方程式は，上記の方法を拡張して近似解法を得ることができる．詳細は講義で説明する．

例題５

dy/dx = -2y + 5 e^{3x} + 4　y(0)=6をルンゲクッター法（２次公式）で x=[0,0.1] の区間の解を求めよ．　ただしh=0.01 とせよ．

解

```
/*--- reidai5 Runge-Kutta method -----*/
#include <stdio.h>
#include <math.h>
double f(double x,double y)
  {
  return(-2.0*y+5.0*exp(3*x)+4.0);
  }
main()
{
FILE *fp;
double x0=0.0,y0=6.0,h=0.01;
double x1,y1,f1,f2;
int n;
/*--------------*/
fp=fopen("data5","w");
for(n=0;n<=9;n++){
```

```
 f1=f(x0,y0);
 x1=x0+h;
 f2=f(x1,y0+h*f1);
 y1=y0+0.5*(f1+f2)*h;
  printf("x=%f     y=%f\n",x1,y1);
 fprintf(fp,"x=%f     y=%f\n",x1,y1);
 x0=x1;
 y0=y1;
}}
```

```
x=0.010000     y=5.971061
x=0.020000     y=5.944226
x=0.030000     y=5.919500
x=0.040000     y=5.896888
x=0.050000     y=5.876400
x=0.060000     y=5.858042
x=0.070000     y=5.841827
x=0.080000     y=5.827765
x=0.090000     y=5.815870
x=0.100000     y=5.806156
```

（参考）　例題５の微分方程式の解は解析的に得られて $y(x) = 3e^{-2x} + e^{3x} + 2$
この解を用いると　$y(0.01) = 5.971051$, $y(0.05) = 5.876347$, $y(0.1) = 5.806051$　　となる．
上の結果と比較せよ．

演習問題 12

次の微分方程式のx=[1,1.2] の区間の解をルンゲクッター法（２次公式）で 求めよ．ただしh=0.01 とせよ．

$dy/dx = 2\,y/x - (y/x)^2$,　　$y(1)=2$

演習問題 13

次の微分方程式の x=[0,1.0] の区間の解をルンゲクッター法（２次公式）で 求めよ．ただしh=0.01 とせよ．

$dy/dx - xy + xy^2\exp(-x^2/2) = 0$,　　$y(0)=1.0$