1　C言語による数値計算の基礎（1）

1.1　実数型変数についての一般的な注意

数値計算では実数型（＝小数点数）の変数を用いて計算することになる．これまでの演習では，実数を扱う際は **float** 型の変数を宣言し扱ってきたが，通常はより精度の高い（＝より大きな桁数）少数を扱う事のできる **double** 型（倍精度実数型）を用いることが多い．よって今後実数を扱う際はすべて double 型として扱うことにする．

1.1.1　実数型変数 (float,double) の限界

コンピュータのメモリが有限である以上，実数型の変数にはいくつかの「限界」がある．これらは使っているコンピュータに依存するが，概ね次の通り．

1. 代入できる数の大きさの限界.
   float：最大の数は約 3.4×1038 程度
   double：最大の数は約 1.7×10308 程度
2. 意味のある和が計算できる精度：$1.0 + x \neq 1.0$ をみたす x の最小値:
   float：$x = 1.2 \times 10^{-7}$
   double：$x = 2.2 \times 10^{-16}$

この限界を越えた代入や計算を行っても，結果は数学の期待どおりにはならないことに注意すること．

たとえば和の限界精度については，次のサンプルプログラムを実行して確認すること．

```
#include <stdio.h>

main(){
  double x;
  double a=0.00000000000000022044604925031; /* 1.0+a ≠ 1.0 （和の計算精度の限界） */

  printf("%.30f\n",a);  /* 0.00000000000000022044604925031 と表示される */
  x=1.0+a;
  printf("%.30f\n",x);  /* 1.00000000000000022044604925031 と表示される */
  x=2.0+a;
  printf("%.30f\n",x);  /* 2.000000000000000000000000000000 と表示されてしまう！ */
}
```

1.2　無限級数の値を求める　–繰り返し計算によって精度をあげる–

たとえ収束することがわかっているような無限級数であっても，その具体的な値を正確に求めることは困難，あるいは不可能であることが多い．そこでコンピュータを利用して，完全に正確ではないが近似的な値を求める事を考える．

例題

数列 $a_n = \dfrac{1}{n^2}$ の $n = 1$ から ∞ の和は収束することが知られている（微積分の本参考）．この値の近似値を求めよ．

コンピュータでは無限に足し算を繰り返すことはできないが，とりあえずは様々な n にまでについて和を求めることにして，予想される収束値を調べてみる．以下はそのサンプルプログラムである．

```
#include <stdio.h>

double sum(int n){
  int i;
  double s;

  s=0.0;
  for(i=1 ; i<=n ; i++)
    s=s+1/double(i*i);
  return s;
}

int main(){
  int n;

  for(n=1 ; n<=10 ; n++)
    printf("n=%d までの和は %.18f\n",n,sum(n));
  return 0;
}
```

実行結果

```
% ./a.out
n=1 までの和は 1.000000000000000000
n=2 までの和は 1.250000000000000000
n=3 までの和は 1.361111111111111160
n=4 までの和は 1.423611111111111160
n=5 までの和は 1.463611111111111196
n=6 までの和は 1.491388888888888875
n=7 までの和は 1.511797052154195020
n=8 までの和は 1.527422052154195020
n=9 までの和は 1.539767731166540754
n=10 までの和は 1.549767731166540763
```

このプログラムでは sum(int n) という関数を，第 n 項までの和を計算した結果を与えるように定義している．そして，main() 関数では，n を変化させて (大きくして) sum 関数を繰り返し計算させて，和が一定の値に近づいていく様子を観察できるようにしてある．

実際 $n = 9$ と $n = 10$ までの和を比べるとそれらの値が近いことから，ある一つの値に近づいていることが示唆されるだろう．

（注）この問題については，正確な値が知られている（微積の本参照）：

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6} = 1.644934067\cdots .$$

よって，上のプログラムによる計算では，真の値にまではまだ到達していないことがわかる．十分によい結果を出すためには，n の値をかなり大きくしなければならないが，どの程度まで n を大きくするのがいいのか，考える必要がある．

1.3 数値積分（長方形公式）–無限級数の応用–

任意の関数 $f(x)$ を区間 $[a,b]$ 上で積分（定積分）した値を正確に求める事も一般には不可能である．そこで関数 $f(x)$ と x 軸で囲まれた図形を多数の長方形で近似し，それらの長方形の面積の和を計算する事で積分値の近似計算を行うことを考える．区間 $[a,b]$ を n 等分し，i 番目の小区間の左端の点 $a+\frac{b-a}{n}i$ での関数値 $f(a+\frac{b-a}{n}i)$ を高さとするような長方形を各小区間毎につくり，それらの和をとることでことで面積を近似できる．正確な定積分は，この分割の個数 n を ∞ にしなければならないが，必要な精度が得られるぐらい大きな n までの計算を行えば，実用的には十分だと考える．

以下は，$\int_0^\pi \sin(x)dx$ の近似値を求めるプログラムの例.

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

double numinteg(double a, double b, int n){ /* sin(x) 関数の数値積分 */
  int i;
  double integ;         /* 積分値を格納する変数 */
  double dx;            /* [a,b] を n 等分したときの，各小区間の長さ */

  integ=0.0;
  dx=(b-a)/n;
  for(i=0;i<n;i++)
    integ=integ+sin(a+dx*i)*dx;  /* 各小区間上の長方形の面積を加えていく，
                                    sin(a+dx*i)*dx は各小区間上の長方形の面積 */
  return integ;
}

main(){
  int n;

  for(n=1;n<=10;n++)
    printf("分割数=%d 数値積分の結果=%.16f\n",n,numinteg(0.0,3.141592,n));
}
```

このプログラムでは，数値積分を行う関数 numinteg(double a, double b, int n) を定義してある．変数 dx には各小区間の長さをあらかじめ計算したものを代入しておく．こうすることで何度も同じ計算を繰り返さないようにし，結果として計算速度の向上などにつながる．

1.4 収束の判定

これまでの例では，和の個数や分割の個数 n を大きくして，確かにある値に計算結果が近づいているのを観察できるものだったが，実際必要なことは，実用的に求める精度の近似値が得られるまで計算するには，どのぐらい大きな n まで計算しなければならないかを自動的に判定することである．一つの方法として，n を大きくしていっても計算結果があまり変わらなければ，それは収束値に十分近づいたと判断することができると考えて（しかしこれは必要条件に過ぎないことに注意），繰り返しの計算で一つ前の結果と，現在の結果の差をとって，それがある精度（誤差）以内なら計算を終了するという規則をプログラミングすればよいだろう．

数列 $x_1=1,\ x_{n+1}=\frac{x_n^2+2}{2x_n}$ の収束値を求めるプログラム．（この数列は $\sqrt{2}$ に収束することが知られている）

```
#include <stdio.h>


double dif(double x, double y){/* |x-y|を計算する関数 */
```

```
  if(x>y)
    return x-y;    /* 差が正なら，そのまま差を返す */
  else
    return y-x;    /* 差が負なら，正の方の差を返す */
}

main(){
  double a,b;

  a=2.0;                    /* 初項の値を 2.0 にする */
  b=(a*a+2)/(2*a);          /* 現在の項 b には a の次の項を代入する */
  while(dif(a,b)>0.00001){ /* 隣接二項の差が大きい間繰り返す */
    a=b;                    /* 現在の項を a に代入する（一つ前の項とする）*/
    b=(a*a+2)/(2*a);        /* b に次の項を代入する */
  }
  printf("%f\n",b);
}
```

ここでは数列 x_n の真の収束値に十分近い値を得られる程の n を求め，収束値の近似値を求めています．変数 a は数列の一つ前の項，b は数列の現在の項が代入され，これらの差が $0,00001$ 以下になるまで計算を繰り返します．計算が終わるときには，真の収束値との誤差はかなり小さくなることが期待できます．

問：このプログラムのアイデアを利用して，積分 $\int_0^{\pi} \sin(x)dx$ を長方形で n 分割して近似した値 x_n が $|x_{n+1} - x_n| < 0,00001$ となるような n と x_n を求めるプログラムを作成せよ．（ヒント：数値積分のプログラム例中にある numinteg 関数を用いよ.）

問：numinteg 関数は長方形による近似であるので，一般に収束が遅い．長方形の代わりに，小区間の両端点での関数値を利用し，面積を台形で近似することで定積分を計算する関数 numinteg2 実現するプログラムを作成せよ．