配列（復習問題）（教科書Step19 p.81 演習(2)）

２行２列の行列A,Bの足し算をしなさい。

$$\begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix} + \begin{bmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{bmatrix} = \begin{bmatrix} C_{00} & C_{01} \\ C_{10} & C_{11} \end{bmatrix}$$

- 2行 2列の配列 A[2][2],B[2][2],C[2][2]を使うこと
- 各成分は整数型(int)、実数型(float, double)、どちらでも良い
- 行列A,Bの各成分について、キーボードから値を入力、もしくはあらかじめ初期値を与える(初期値を与えるやり方については教科書p.80参照)、どちらでも良い
- 結果を画面表示すること

関数（補足）　引数と返り値（戻り値）

返り値(戻り値)

```
/*sample10_1.c(教科書p.43再掲)*/
#include <stdio.h>
int sum(int a, int b)
{
        int ans;
        ans=a+b;
        return ans;
}
main()
{
        int x,y,z;
        x=1;
        y=2;
        z=sum(x,y);
        printf("%d¥n",z);
}
```

仮引数

実引数

- 関数」において、引数は複数持つことができるが、返り値(戻り値)は一つのみ
- 返り値を持たない、あるいは引数を持たない関数を定義することができ、その場合**void**と設定する

例:

```
void function(int a,int b)
int function(void)
```

- **main()**も引数と返り値を持つことができる関数の一種であり、返り値は必ず持つこととされる(教科書p.45)
- void main()という記述は、gccでは警告(warning)表示となる(警告とならないCコンパイラもある)

関数（補足）グローバル変数とローカル変数

```
/*sample10_1.c(教科書p.43一部変更)*/
#include <stdio.h>
/*グローバル変数ansの宣言*/
/*変数ansはsum(),main()両方で利用可*/
int ans;
/*関数sum()の返り値は無し(void)*/
void sum(int a, int b)
{
        ans=a+b;
}
main()
{
        int x,y;
        x=1;
        y=2;
        sum(x,y);
        printf(”%d¥n”,ans);
}
```

- あらゆる関数(main()も含む)の**外**で定義される変数のことを**グローバル変数**といい、個々の関数**内**で定義される変数のことを**ローカル変数**という
- グローバル変数はプログラム中のあらゆる関数から参照可能、値を変更することができる
- **グローバル変数と同一の変数名を、他の関数内にて、ローカル変数名としては使えない**ため、グローバル変数を多用することは好ましくないとされる

関数（補足）　プロトタイプ宣言

```
/*sample10_1.c(教科書p.43一部変更)*/
#include <stdio.h>
/*関数sum()プロトタイプ宣言*/
int sum(int a, int b);
main()
{
        int x,y,z;
        x=1;
        y=2;
        z=sum(x,y);
        printf("%d¥n",z);
}
/*プロトタイプ宣言済み関数の実体*/
int sum(int a, int b)
{
        int ans;
        ans=a+b;
        return ans;
}
```

- **プロトタイプ宣言**を利用し、あらかじめ関数の引数と返り値を設定しておくことで、関数の実体を**main()の後**に記述可
- プロトタイプ宣言では、

  int sum(int,int);

  のように、変数名を省略して記述しておくことも可

関数（補足）再帰呼び出し

```
#include<stdio.h>
/*関数func()プロトタイプ宣言*/
long func(int a);
/*自然対数の底e近似計算*/
main()
{
int aa;
double e;
e=0.0;
        for(aa=0;aa<=10;aa++)
        {
e+=1.0/(double)func(aa);
printf("aa=%d,e:%lf¥n",aa,e);
        }
}
```

```
/*関数func()実体*/
/*a!=a・(a-1)・(a-2)・・1の計算*/
long func(int a)
{
        /*0!=1とする*/
        if (a==0){
        return (1);
        }
        else{
        /*再帰呼び出し部分*/
        return(a * func(a-1));
        }
}
```

- 関数は「**それ自身**」を呼び出す事ができ、**再帰呼び出し**という
- 関数func()は階乗(a!)を計算、従って自然対数の底 **e(=2.718・・)**の近似値が求まる(=1+1/1!+1/2!+1/3!+・・+1/10!)

関数（補足）　乱数１

```
/*乱数を10個表示する*/
#include<stdio.h>
#include<stdlib.h>
main()
{
int i, rn;
unsigned int seed;
printf(”乱数の種(0以上の整数)入力¥n");
scanf("%d",&seed);
srand(seed); /*乱数表(種)選択*/
printf("乱数10個表示¥n");
        for(i=0; i< 10 ; i++)
        {
        rn=rand();/*乱数発生*/
        printf("%d¥",rn);
        }
}
```

- Cで乱数を利用するには
  乱数の種(表)作成関数srand()
  乱数発生関数rand()
  を使用
- 関数srand()とrand()を
  利用するために必要なヘッダー
  stdlib.h
- **rand()により発生する乱数値は0～RAND_MAXの範囲**
- **RAND_MAXの値はコンパイラにより異なる**

例:
gcc:16進数0x7fffffff=10進数2147483647
VC++:16進数0x7fff=10進数32767

問題点:
同じ値を入力すると常に同じ10個の値→発生する乱数値を予測できてしまう

関数（補足） 乱数２(実行ごとに異なる値を発生)

```
/* seedの値を現在時刻で代用*/
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
main()
{
int i, rn;
unsigned int seed;
int nowtime;
time(&nowtime);
seed=(unsigned int)nowtime;
srand(seed);
printf("乱数10個表示￥n");
        for(i=0; i< 10 ; i++)
        {
        rn=rand();
        printf("%d¥n",rn);
        }
}
```

- 現在時刻を取得する関数time()を使う
- ヘッダーファイルtime.hが必要
- 起動ごとにtime()で異なった値を取得し、関数srand()へ引き渡す
- srand()により、起動ごとに異なる乱数表が選択され、乱数10個発生

発生する乱数の予測はできない

↓

ゲーム等、「偶然」を利用するプログラムに適用可

課題：モンテカルロ法(教科書p.57sample13_2.c)

教科書p.57sample13_2.cを、以下の条件を付加して書き換え、実行しなさい

•試行回数について、long int型変数Nをlong int型配列N[3]={100,10000,100000}とし、それぞれ3種類について計算、結果を配列ans[3]の各成分に入力、画面に表示する

復習課題 解答例1(A,B成分はキーボードから入力)

```
#include<stdio.h>
/*配列計算(A,B成分キーボード入力)*/
main(){
        int a[2][2],b[2][2],c[2][2];
        int i,j;
        for(i=0; i<2;i++){
                for(j=0;j<2;j++){
                        printf("A_%d%d成分入力¥n",i,j);
                        scanf("%d",&a[i][j]);
                        printf("B_%d%d成分入力¥n",i,j);
                        scanf("%d",&b[i][j]);
                }
        }
        for(i=0; i<2;i++){
                for(j=0;j<2;j++){
                        c[i][j]=a[i][j]+b[i][j];
                }
        }
        for(i=0;i<2;i++){
                for(j=0;j<2;j++){
                        printf("i=%d,j=%d¥n",i,j);
        printf("a_ij:%d+b_ij:%d=c_ij:%d¥n",a[i][j],b[i][j],c
[i][j]);
                }
        }
}
```

復習課題 解答例2 (A,B成分の初期値を与えておく)

```
#include<stdio.h>
/*配列計算(A,B成分の初期値を与えておく)*/
main(){
        int a[2][2]={{1,0},{0,1}};
        int b[2][2]={{0,1},{1,0}};
        int c[2][2];
        int i,j;
        for(i=0; i<2;i++){
                for(j=0;j<2;j++){
                        c[i][j]=a[i][j]+b[i][j];
                }
        }
        for(i=0;i<2;i++){
                for(j=0;j<2;j++){
                        printf("i=%d,j=%d¥n",i,j);
printf("a_ij:%d+b_ij:%d=c_ij:%d¥n",a[i][j],b[i][j],c[i][j]);
                }
        }
}
```

課題 解答例(モンテカルロ法)

P.57sample13_2.cの変更部分のみ赤字表示(変更部分はmain()の中のみ)

```
main(){
        int i,j,sum;
        long N[3]={100,10000,100000};
        float ans[3],x,y,L,P;
        for(j=0;j<3;j++){
        srand((unsigned)time(NULL));
        sum=0;
        /*N=1000;は削除*/
        for(i=0;i<N[j];i++){
                x=myrand();
                y=myrand();
                L=x*x+y*y;
                if(L<=1){
                        sum++;
                }
        }
        ans[j]=(float)(sum)/N[j];
        P=ans[j]*4;
        printf("N[%d]=%d,ans[%d]=%f pi=%f¥n",j,N[j],j,ans[j],P);
        }

}
```