\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Report: hw4

Author: F74055047張財實 <changchaishi@gmail.com>

Class: 乙班

Description:

Learned how to use recursion, and knowing recursion’s characteristics can have more ability to design our program.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Code:

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

//header for random number generation

int N;//global variable

//prototype of calculation function

void add(float [][\*],float [][\*],float [][\*]);

void multiplication(float [][\*],float [][\*],float [][\*]);

float det(int ,float [][\*]);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

main function

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main(int argc,char \*argv[]){

int fun,type,i,j;

//read N,functions,type of input

N=atoi(argv[1]);

fun=atoi(argv[2]);

type=atoi(argv[3]);

//declaration of matrix by NxN

float Result[N][N],M1[N][N],M2[N][N],R[N][N];

//for the selection of the input type

switch(fun){

case 0: case 1:

switch(type){

case 0:

//random number generation

srand((unsigned) time(NULL));

for(j=0;j<N;j++){

for(i=0;i<N;i++){

M1[j][i]=rand()%100;

}

}

for(j=0;j<N;j++){

for(i=0;i<N;i++){

M2[j][i]=rand()%100;

}

}

break;

case 1:

//user input

for(j=0;j<N;j++){

for(i=0;i<N;i++){

scanf("%f",&M1[j][i]);

}

}

for(j=0;j<N;j++){

for(i=0;i<N;i++){

scanf("%f",&M2[j][i]);

}

}

break;

}

break;

case 2:

switch(type){

case 0:

srand((unsigned) time(NULL));

for(j=0;j<N;j++){

for(i=0;i<N;i++){

R[j][i]=rand()%100;

}

}

break;

case 1:

for(j=0;j<N;j++){

for(i=0;i<N;i++){

scanf("%f",&R[j][i]);

}

}

break;

}

break;

}

//for the selection of function,and call of function

switch(fun){

case 0:

add(Result,M1,M2);

break;

case 1:

multiplication(Result,M1,M2);

break;

case 2:

printf("%.2f\n",det(N,R));

break;

}

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

add function:

adding corresponding array number together from M1 and M2,

which will produce Result

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void add(float Result[][N],float M1[][N],float M2[][N]){

int i,j;

for(j=0;j<N;j++){

for(i=0;i<N;i++){

Result[j][i]=M1[j][i]+M2[j][i];

printf("%.2f ",Result[j][i]);

printf("\n");

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

multiplication function:

To calculate each Result array,it has to be N layar for the times

of M1,M2 array ,then sum up to be one array of Result array.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void multiplication(float Result[][N],float M1[][N],float M2[][N]){

int i,j,k;

for(j=0;j<N;j++){

for(i=0;i<N;i++){

Result[j][i]=0;

for(k=0;k<N;k++){

//k is the pattern of algorithm,the layer

Result[j][i]+=M1[j][k]\*M2[k][i];

}

printf("%.2f\n",Result[j][i]);

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

determinant function:

using recursion to decrease the order of the matrix.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

float det(int N,float R[][N]){

int i,j;

int sign,x,y;

//sign bit stands for the "sign" +/- sign for the element

int p,q;

float r[N-1][N-1],v[N];

//r is selection for the new decreased order array

float value=0;

if(N==2){

//lowest degree matrix

value=0;

value=(R[0][0]\*R[1][1])-(R[0][1]\*R[1][0]);

return value;

}

else{

//first ROW has N element

for(x=0;x<N;x++){

//----------------------------

//r calculation

q=0;

for(j=1;j<N;j++){

p=0;

for(i=0;i<N;i++){

if(i==x){

r[q][p]=R[j][++i];

}

else{

r[q][p]=R[j][i];

}

++p;

}

++q;

}

//----------------------------

//each subarray's value and recursion

sign=1;

for(y=1;y<=x;y++){

sign=(-1)\*sign;

}

v[x]=sign\*det(N-1,r);

}

value=0.0f;

for(x=0;x<N;x++){

//sum up and done

value+=(R[0][x]\*v[x]);

}

return value;

}

}

Compilation:

gcc -o hw4 hw4.c

Execution:

./hw4 3 2 1

Input:

2 4 6 8 10 1 3 5 7

Output:

-22.00