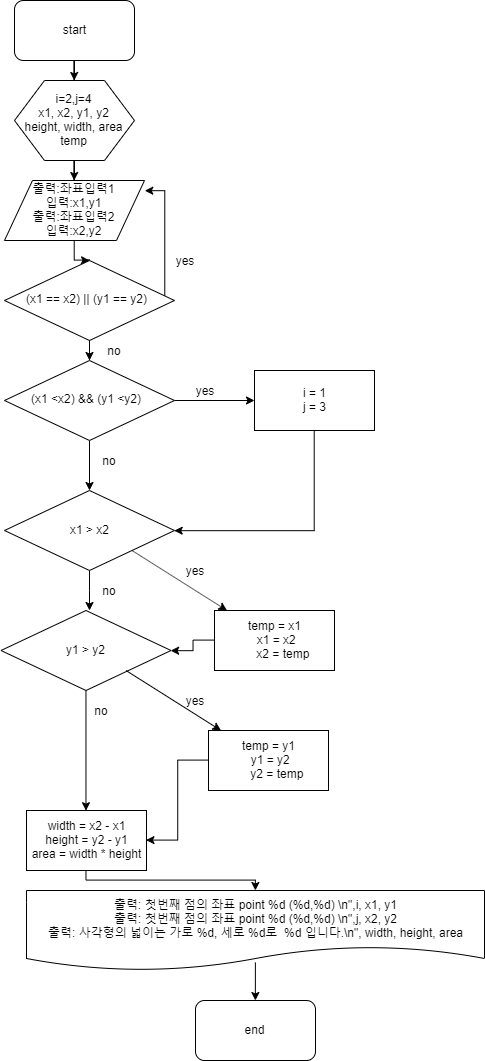
1차 프로젝트 과제/고급C프로그래밍/박철수교수님/엄희상조교님/컴퓨터정보공학부/2019202052/김호성/10.10

TEST 01

1. 과제소개

서로 다른 좌표를 입력하면 직사각형의 넓이를 출력할 수 있는 프로그램이다.(단, 직사각형의 4개의 점 중 (1,3)인지 (2,4)인지 구별할 수 있어야 하며, x좌표나 y좌표 중 하나라도 같은 값을 입력했을 시 입력을 다시 받는다.)

2. 설계내용(flowchart)



3. 코드 및 주석

#include<stdio.h> //표준입출력헤더파일 포함

int main() //main함수 생성

{

int i = 2, j = 4; //좌표 포인트 정하는 변수 생성

int x1, x2, y1, y2; //좌표 입력 받을 변수

int height, width, area; //가로 세로 넓이변수

int temp; //값 이동 변수

back: //입력받은 값에 오류가 있을경우 돌아옴.

printf("첫 번째 좌표를 입력해 주세요 : ");

scanf("%d %d", &x1, &y1); //첫 번째 좌표를 입력받음

printf("두 번째 좌표를 입력해 주세요 : ");

scanf("%d %d", &x2, &y2); //두 번째 좌표를 입력받음

if((x1 == x2) || (y1 == y2)) //x나 y값 중 같은 값이 있으면 직선이 되어버림.

{

printf("잘못된 좌표를 입력하셨습니다. \n\n");

goto back; //좌표 값 다시 받음

}

if((x1 < x2)&&(y1 < y2)) //좌표 포인트 정하는 조건

{

i = 1;

j = 3;

}

if(x1 > x2) //후에 가로에 음수 값이 안나오게 함.

{

temp = x1;

x1 = x2;

x2 = temp;

}

if(y1 > y2) //후에 세로에 음수 값이 안나오게 함.

{

temp = y1;

y1 = y2;

y2 = temp;

}

width = x2 - x1; //가로 계산

height = y2 - y1; //세로 계산.

area = width \* height; //넓이 계산

printf("첫번째 점의 좌표 point %d (%d,%d) \n",i, x1, y1); //첫 번째 좌표위치 출력

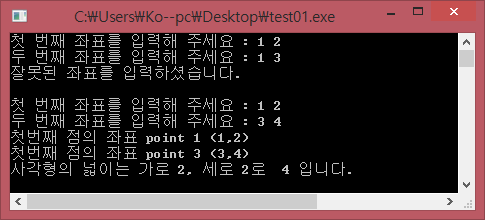
printf("첫번째 점의 좌표 point %d (%d,%d) \n",j, x2, y2); //두 번째 좌표위치 출력

printf("사각형의 넓이는 가로 %d, 세로 %d로 %d 입니다.\n", width, height, area); //넓이 출력

return 0; //main함수 종료

}

4. 실행결과



5. 고찰

-두 점의 좌표를 받을 경우 x나 y좌표 중 두 점의 좌표 값이 하나라도 같을 경우 직선으로 만들어진다.

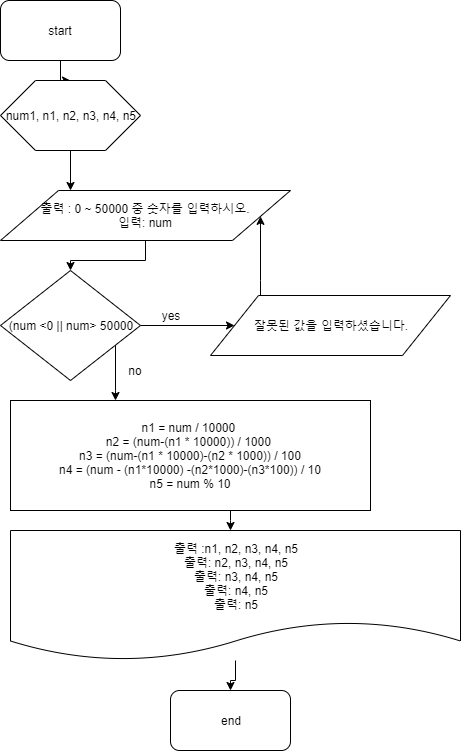
예) (1,2),(1,3)

TEST 02

1. 과제소개

0~50000의 숫자 중 하나를 입력 받아 10000의 자리부터 하나씩 없어지게 한다. (단, 음수이거나, 50000초과의 숫자를 입력 받았을 경우 다시 입력 받게 한다.)

2. 설계내용(flowchart)



3. 코드 및 주석

#include <stdio.h> //표준입출력헤더파일 포함

int main() //main함수 생성

{

int num, n1, n2, n3, n4, n5; //입력받을 num과 n1~n5 생성

back:

printf("0~50000중 숫자를 입력하시오. ");

scanf("%d", &num); //숫자 입력 받음

if(num<0||num>50000){ //예외처리

printf("잘못된 값을 입력하셨습니다.\n");

goto back;

}

n1 = num / 10000; //10000의 자리 계산

n2 = (num - (n1\*10000)) / 1000; //1000의 자리 계산

n3 = (num - (n1\*10000) -(n2\*1000)) / 100; //100의 자리 계산

n4 = (num - (n1\*10000) -(n2\*1000)-(n3\*100)) / 10; //10의 자리 계산

n5 = num % 10; //1의 자리 계산

printf("%d %d %d %d %d \n", n1, n2, n3, n4, n5); //이후 출력

printf("%d %d %d %d \n", n2, n3, n4, n5);

printf("%d %d %d \n", n3, n4, n5);

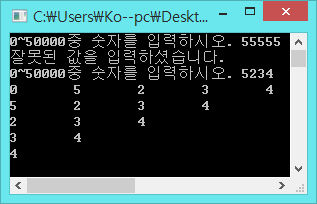
printf("%d %d \n", n4, n5);

printf("%d \n",n5);

return 0; //main함수종료

}

4. 실행결과



5. 고찰

-각 자릿수 별로 계산법

10000의 자리: 입력 받은 숫자 나누기 10000의 몫(이하 x1)

1000의 자리: (입력 받은 숫자 –(x1\*10000)) 나누기 1000의 몫(이하 x2)

100의 자리: (입력 받은 숫자 –(x1\*10000) –(x2\*1000)) 나누기 100의 몫(이하 x3)

10의 자리: (입력 받은 숫자 –(x1\*10000) –(x2\*1000) –(x3\*100) 나누기 10의 몫

1의 자리: 입력 받은 숫자 나누기 10의 나머지 값

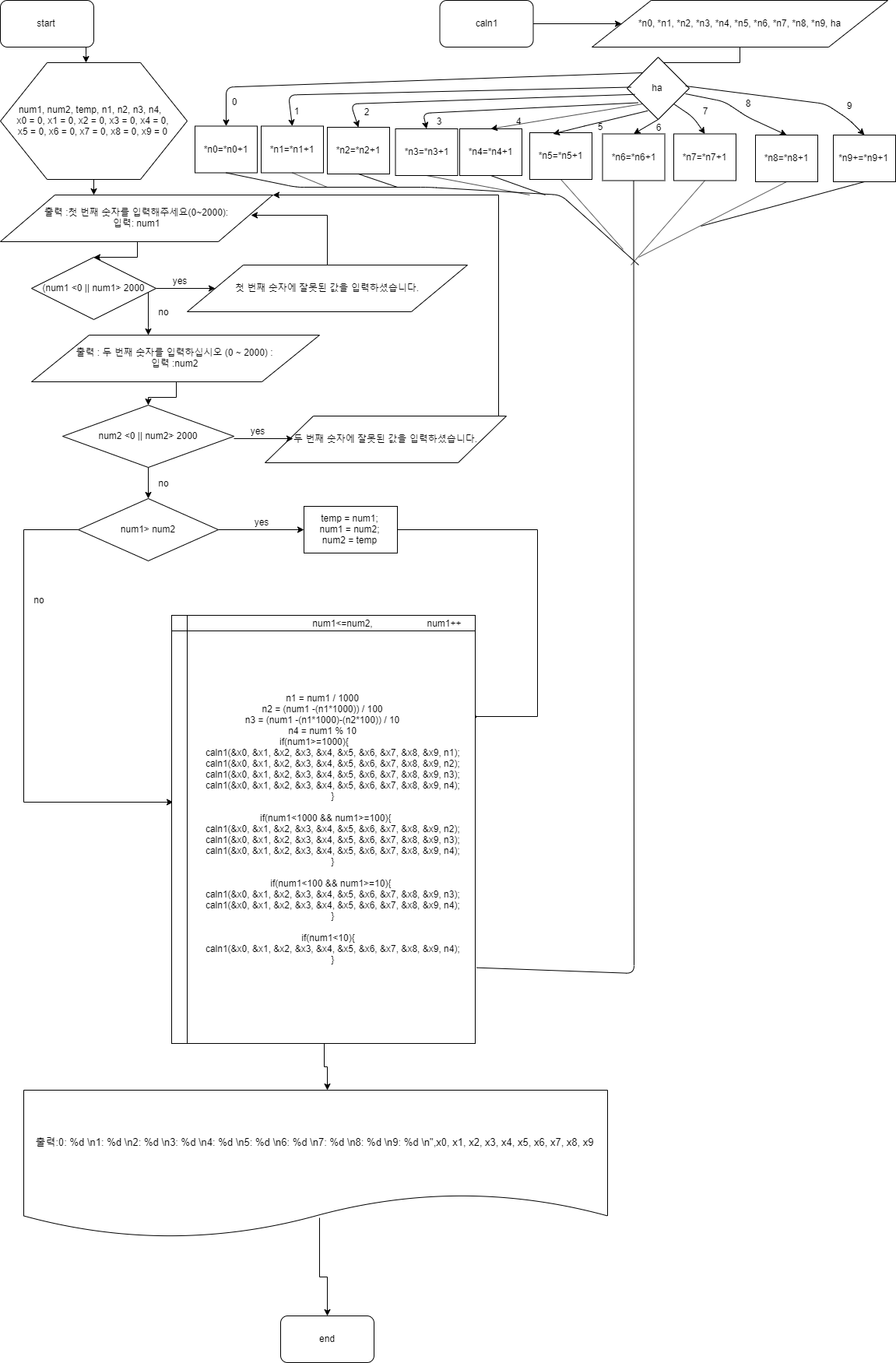
TEST 03

1. 과제소개

두 숫자를 입력 받고, 작은 숫자에서 큰 숫자까지 올라가며 나온 숫자들의 개수를 계산한다.

예) 1과 9를 입력 받음: 1~9까지 전부 하나씩 나왔습니다.

2. 설계내용(flowchart)(그림이 커서 다음 페이지에 있음)



3. 코드 및 주석

#include <stdio.h>

void caln1(int \*n0,int \*n1,int \*n2,int \*n3,int \*n4,int \*n5,int \*n6,int \*n7,int \*n8,int \*n9,int ha)

{

switch (ha) { //스위치문법을 이용해 0~9중 숫자가 나올 시 값을 증가시킴

case 0:

\*n0=\*n0+1;

break;

case 1:

\*n1=\*n1+1;

break;

case 2:

\*n2=\*n2+1;

break;

case 3:

\*n3=\*n3+1;

break;

case 4:

\*n4=\*n4+1;

break;

case 5:

\*n5=\*n5+1;

break;

case 6:

\*n6=\*n6+1;

break;

case 7:

\*n7=\*n7+1;

break;

case 8:

\*n8=\*n8+1;

break;

case 9:

\*n9+=\*n9+1;

break;

}

}

int main()

{

int num1, num2, temp, n1, n2, n3, n4; //입력 받을 두 변수, 두 숫자 중 큰 걸 뒤로 보낼 temp, 자릴 수 마다 분해할 변수생성

int x0 = 0, x1 = 0, x2 = 0, x3 = 0, x4 = 0, x5 = 0, x6 = 0, x7 = 0, x8 = 0, x9 = 0; //0~9의 숫자 증가 변수

back:

printf("첫 번째 숫자를 입력해주세요(0~2000):");

scanf("%d", &num1); //첫 번째 숫자 입력받음

if(num1<0||num1>2000){ //예외처리

printf("첫 번째 숫자에 잘못된 값을 입력하셨습니다.\n");

goto back;

}

printf("두 번째 숫자를 입력해주세요(0~2000):");

scanf("%d", &num2); //두 번째 숫자 입력 받음

if(num2<0||num2>2000){ //예외처리

printf("두 번째 숫자에 잘못된 값을 입력하셨습니다.\n");

goto back;

}

if(num1>num2) //두 숫자 중 작은 숫자를 앞으로 보냄

{

temp = num1;

num1 = num2;

num2 = temp;

}

while(num1<=num2){ //작은 숫자가 큰 숫자가 되면 반복문 종료

n1 = num1 / 1000; //1000의 자리 계산

n2 = (num1 -(n1\*1000)) / 100; //100의 자리 계산

n3 = (num1 -(n1\*1000)-(n2\*100)) / 10; //10의 자리 계산

n4 = num1 % 10; //1의 자리 계산

if(num1>=1000){ //1000의 자리일 경우 n1~n4숫자 전부 유효

caln1(&x0, &x1, &x2, &x3, &x4, &x5, &x6, &x7, &x8, &x9, n1);

caln1(&x0, &x1, &x2, &x3, &x4, &x5, &x6, &x7, &x8, &x9, n2);

caln1(&x0, &x1, &x2, &x3, &x4, &x5, &x6, &x7, &x8, &x9, n3);

caln1(&x0, &x1, &x2, &x3, &x4, &x5, &x6, &x7, &x8, &x9, n4);

}

if(num1<1000 && num1>=100){ //100의 자리일 경우 n2~n4숫자만 유효

caln1(&x0, &x1, &x2, &x3, &x4, &x5, &x6, &x7, &x8, &x9, n2);

caln1(&x0, &x1, &x2, &x3, &x4, &x5, &x6, &x7, &x8, &x9, n3);

caln1(&x0, &x1, &x2, &x3, &x4, &x5, &x6, &x7, &x8, &x9, n4);

}

if(num1<100 && num1>=10){ //10의 자리일 경우 n3~n4숫자만 유효

caln1(&x0, &x1, &x2, &x3, &x4, &x5, &x6, &x7, &x8, &x9, n3);

caln1(&x0, &x1, &x2, &x3, &x4, &x5, &x6, &x7, &x8, &x9, n4);

}

if(num1<10){ //1의 자리일 경우 n4숫자만 유효

caln1(&x0, &x1, &x2, &x3, &x4, &x5, &x6, &x7, &x8, &x9, n4);

}

num1++; //반복문 탈출 조건

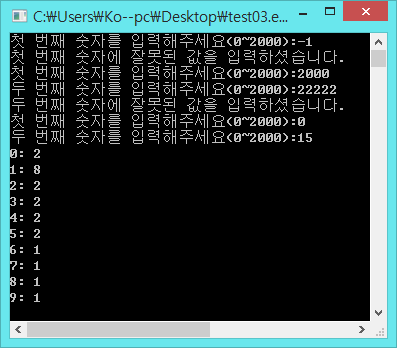
}

printf("0: %d \n1: %d \n2: %d \n3: %d \n4: %d \n5: %d \n6: %d \n7: %d \n8: %d \n9: %d \n",x0, x1, x2, x3, x4, x5, x6, x7, x8, x9); //결과 출력

return 0;

}

4. 실행결과



5. 고찰

-TEST 02번의 자릿수마다 변수를 따로 생성한 것을 응용했고, 계산하는 함수를 새로 정의해 숫자마다 값을 증가 시켰다.

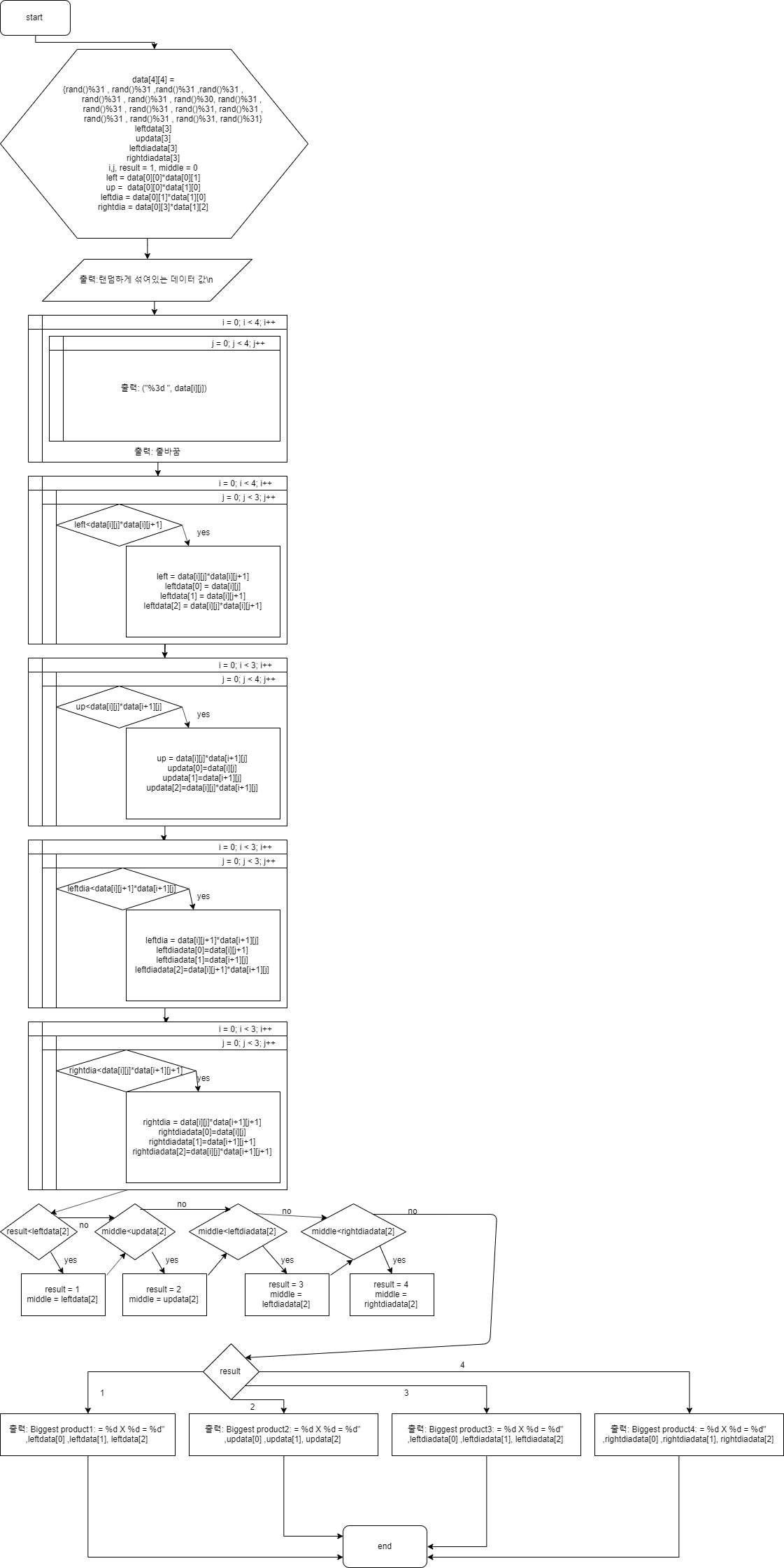
-0의 경우 입력 받은 숫자보다 큰 자릿수에 있는 0과 작은 자릿수에 있는 0으로 구분이 되는데, 이때 작은 자릿수에 있는 0만 유효하다 따라서 입력 받은 숫자가 몇의 자리인지 먼저 파악하고 계산을 진행했다.

TEST 04

1. 과제소개

4X4 2차원 배열에 무작위 숫자를 집어 넣는 난수를 사용해 배열에 숫자를 입력함. 배열 중 가로, 세로, 대각선으로 인접해 있는 값을 곱해 최댓값을 구하기.

2. 설계내용(flowchart)



3. 코드 및 주석

#include <stdio.h> //표준입출력헤더파일 포함

#include <stdlib.h> //표준라이브러리헤더파일 포함

int main() //main함수

{

// 1~30의 숫자를 랜덤하게 가지는 2차원 배열

int data[4][4]

= { rand()%31 , rand()%31 ,rand()%31 ,rand()%31 ,

rand()%31 , rand()%31 , rand()%30, rand()%31 ,

rand()%31 , rand()%31 , rand()%31, rand()%31 ,

rand()%31 , rand()%31 , rand()%31, rand()%31 };

int leftdata[3]; //left데이터 값 저장 배열

int updata[3]; //up데이터 값 저장 배열

int leftdiadata[3]; //leftdia데이터 값 저장 배열

int rightdiadata[3]; //rightdia 데이터 값 저장 배열

int i,j, result = 1, middle = 0; //반복문과 조건문들을 위한 변수 생성

int left = data[0][0]\*data[0][1]; //처음 left값

int up = data[0][0]\*data[1][0]; //처음 right값

int leftdia = data[0][1]\*data[1][0]; //처음 leftdia값

int rightdia = data[0][3]\*data[1][2]; //처음 rightdia값

printf("랜덤하게 섞여있는 데이터 값\n"); //기존의 섞여있는 값 출력

for (int i = 0; i < 4; i++)

{

for(int j = 0; j < 4; j++)

{

printf("%3d ", data[i][j]);

}

printf("\n");

}

for(i = 0; i < 4; i++) //양옆의 곱셈 중 최대 값 구함.

{

for(j = 0; j < 3; j++)

{

if(left<data[i][j]\*data[i][j+1])

{

left = data[i][j]\*data[i][j+1];

leftdata[0] = data[i][j]; //값 저장

leftdata[1] = data[i][j+1]; //값 저장

leftdata[2] = data[i][j]\*data[i][j+1]; //값 저장(곱한 값)

}

}

}

for(i = 0; i < 3; i++) //위 아래의 곱셈 중 최대 값 구함.

{

for(j = 0; j < 4; j++)

{

if(up<data[i][j]\*data[i+1][j])

{

up = data[i][j]\*data[i+1][j]; //후에 비교를 위해 저장

updata[0]=data[i][j]; //값 저장

updata[1]=data[i+1][j]; //값 저장

updata[2]=data[i][j]\*data[i+1][j]; //값 저장 (결과 값)

}

}

}

for(i = 0; i < 3; i++) //왼쪽 대각선 최대 값 구함.

{

for(j = 0; j < 3; j++)

{

if(leftdia<data[i][j+1]\*data[i+1][j])

{

leftdia = data[i][j+1]\*data[i+1][j]; //후에 비교를 위해 저장

leftdiadata[0]=data[i][j+1]; //값 저장

leftdiadata[1]=data[i+1][j]; //값 저장

leftdiadata[2]=data[i][j+1]\*data[i+1][j]; //값 저장 (결과 값)

}

}

}

for(i = 0; i < 3; i++) //오른쪽 대각선 최대 값 구함

{

for(j = 0; j < 3; j++)

{

if(rightdia<data[i][j]\*data[i+1][j+1])

{

rightdia = data[i][j]\*data[i+1][j+1]; //후에 비교를 위해 저장

rightdiadata[0]=data[i][j]; //값 저장

rightdiadata[1]=data[i+1][j+1]; //값 저장

rightdiadata[2]=data[i][j]\*data[i+1][j+1]; //값 저장 (결과 값)

}

}

}

if(result<leftdata[2]) //결과 값 정하기 (switch문 사용을 위해)

{

result = 1;

middle = leftdata[2];

}

if(middle<updata[2])

{

result = 2;

middle = updata[2];

}

if(middle<leftdiadata[2])

{

result = 3;

middle = leftdiadata[2];

}

if(middle<rightdiadata[2])

{

result = 4;

middle = rightdiadata[2];

}

switch (result){ //switch문을 이용해 결과 출력

case 1:

printf("Biggest product1: = %d X %d = %d" ,leftdata[0] ,leftdata[1], leftdata[2]);

break;

case 2:

printf("Biggest product2: = %d X %d = %d" ,updata[0] ,updata[1], updata[2]);

break;

case 3:

printf("Biggest product3: = %d X %d = %d" ,leftdiadata[0] ,leftdiadata[1], leftdiadata[2]);

break;

case 4:

printf("Biggest product4: = %d X %d = %d" ,rightdiadata[0] ,rightdiadata[1], rightdiadata[2]);

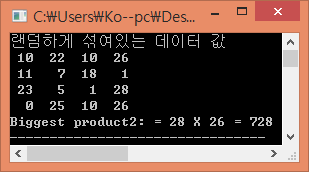
break;

}

return 0; //main함수 종료

}

4. 실행결과



5. 고찰

가로 세로 대각선마다 반복문의 중첩을 통해 비교함. 그 때 마다 배열의 크기가 정해져 있고, 비교 방법이 다르므로 조건식이 달라짐.

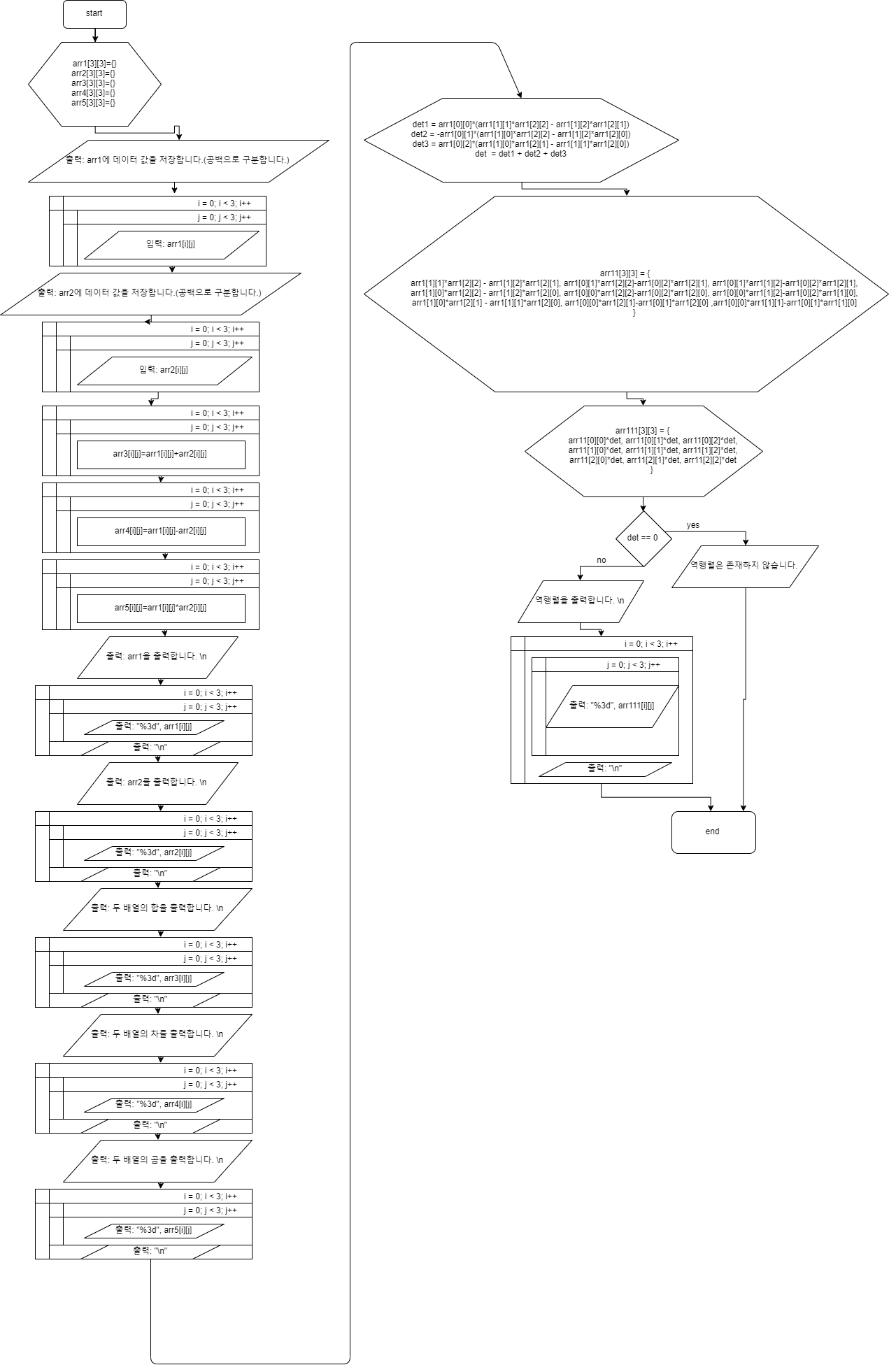
TEST 05

1. 과제소개

3X3 2차원 배열을 두 개 만들어 각각에 값을 입력 받고, 같은 좌표상에 있는 값들의 합, 차, 곱을 구함.

단, 먼저 입력 받은 2차원 배열에 역행렬이 존재한다면 역행렬을 출력함.

2. 설계내용(flowchart)



3. 코드 및 주석

#include <stdio.h>

int main()

{

double arr1[3][3]; //첫 번째 배열의 값 들어감

/\*

arr1[0][0] arr1[0][1] arr1[0][2]

arr1[1][0] arr1[1][1] arr1[1][2]

arr1[2][0] arr1[2][1] arr1[2][2]

행렬식 arr1[0][0] = arr1[1][1]\*arr1[2][2] - arr1[1][2]\*arr1[2][1]...

역행렬 조건 = arr1[0][0]\*(arr1[1][1]\*arr1[2][2] - arr1[1][2]\*arr1[2][1])

- arr1[0][1]\*(arr1[1][0]\*arr1[2][2] - arr1[1][2]\*arr1[2][0])

+ arr1[0][2]\*(arr1[1][0]\*arr1[2][1] - arr1[1][1]\*arr1[2][0])

\*/

double arr2[3][3]; //두 번째 배열의 값 들어감

double arr3[3][3]; //두 배열의 합에 이용

double arr4[3][3]; //두 배열의 차에 이용

double arr5[3][3]; //두 배열의 곱에 이용

printf("arr1에 데이터 값을 저장합니다.(공백으로 구분합니다.)");

for(int i = 0; i < 3; i++) //첫 번째 배열에 값 입력

{

for(int j = 0; j < 3; j++)

{

scanf\_s("%lf", &arr1[i][j]);

}

}

printf("arr2에 데이터 값을 저장합니다.(공백으로 구분합니다.)");

for(int i = 0; i < 3; i++) //두 번째 배열에 값 입력

{

for(int j = 0; j < 3; j++)

{

scanf\_s("%lf", &arr2[i][j]);

}

}

for(int i = 0; i < 3; i++) //배열간 덧셈 반복

{

for(int j = 0; j < 3; j++)

{

arr3[i][j]=arr1[i][j]+arr2[i][j];

}

}

for(int i = 0; i < 3; i++) //배열간 뺄셈 반복

{

for(int j = 0; j < 3; j++)

{

arr4[i][j]=arr1[i][j]-arr2[i][j];

}

}

for(int i = 0; i < 3; i++) //배열간 곱셈 반복

{

for(int j = 0; j < 3; j++)

{

arr5[i][j]=arr1[i][j]\*arr2[i][j];

}

}

printf("arr1을 출력합니다. \n"); //첫 번째 배열 출력

for(int i = 0; i < 3; i++)

{

for(int j = 0; j < 3; j++)

{

printf("%13g", arr1[i][j]);

}

printf("\n");

}

printf("arr2를 출력합니다. \n"); //두 번째 배열 출력

for(int i = 0; i < 3; i++)

{

for(int j = 0; j < 3; j++)

{

printf("%13g", arr2[i][j]);

}

printf("\n");

}

printf("두 배열의 합을 출력합니다. \n"); //배열의 합 출력

for(int i = 0; i < 3; i++)

{

for(int j = 0; j < 3; j++)

{

printf("%13g", arr3[i][j]);

}

printf("\n");

}

printf("두 배열의 차를 출력합니다. \n"); //배열의 차 출력

for(int i = 0; i < 3; i++)

{

for(int j = 0; j < 3; j++)

{

printf("%13g", arr4[i][j]);

}

printf("\n");

}

printf("두 배열의 곱을 출력합니다. \n"); //배열의 곱 출력

for(int i = 0; i < 3; i++)

{

for(int j = 0; j < 3; j++)

{

printf("%13g", arr5[i][j]);

}

printf("\n");

}

// deteriment 구하기( det가 0이면 역행렬이 존재하지 않음.)

double det1 = arr1[0][0]\*(arr1[1][1]\*arr1[2][2] - arr1[1][2]\*arr1[2][1]);

double det2 = -arr1[0][1]\*(arr1[1][0]\*arr1[2][2] - arr1[1][2]\*arr1[2][0]);

double det3 = arr1[0][2]\*(arr1[1][0]\*arr1[2][1] - arr1[1][1]\*arr1[2][0]);

double det = det1 + det2 + det3;

//전치행렬 구하기

double arr11[3][3] = {

arr1[1][1]\*arr1[2][2] - arr1[1][2]\*arr1[2][1], arr1[0][1]\*arr1[2][2]-arr1[0][2]\*arr1[2][1], arr1[0][1]\*arr1[1][2]-arr1[0][2]\*arr1[2][1],

arr1[1][0]\*arr1[2][2] - arr1[1][2]\*arr1[2][0], arr1[0][0]\*arr1[2][2]-arr1[0][2]\*arr1[2][0], arr1[0][0]\*arr1[1][2]-arr1[0][2]\*arr1[1][0],

arr1[1][0]\*arr1[2][1] - arr1[1][1]\*arr1[2][0], arr1[0][0]\*arr1[2][1]-arr1[0][1]\*arr1[2][0] ,arr1[0][0]\*arr1[1][1]-arr1[0][1]\*arr1[1][0]

};

//전치행렬 \* det분 의 1 곱한 배열 = 역행렬

double arr111[3][3] = {

arr11[0][0]\*det, arr11[0][1]\*det, arr11[0][2]\*det,

arr11[1][0]\*det, arr11[1][1]\*det, arr11[1][2]\*det,

arr11[2][0]\*det, arr11[2][1]\*det, arr11[2][2]\*det

};

//역행렬 구하기

if(det == 0)

{

printf("역행렬은 존재하지 않습니다.");

}

else{

printf("역행렬을 출력합니다. \n"); //역행렬 출력

for(int i = 0; i < 3; i++)

{

for(int j = 0; j < 3; j++)

{

printf("%13g", arr111[i][j]);

}

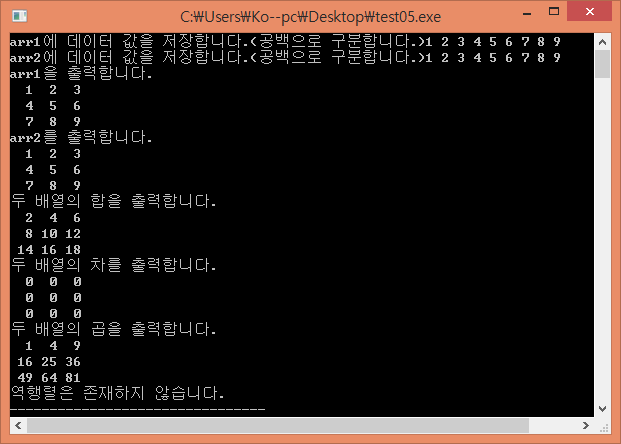
printf("\n");

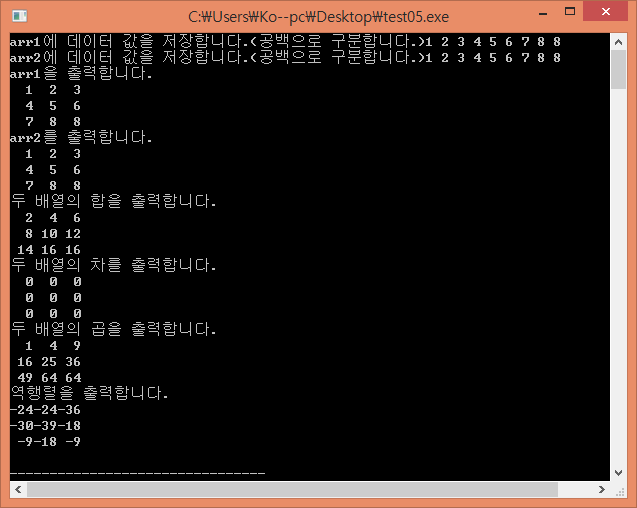
}

}

return 0;

}4. 실행결과





5. 고찰

-역행렬이 존재할 수 있는 조건: (det 값)

arr1[0][0]\*(arr1[1][1]\*arr1[2][2] - arr1[1][2]\*arr1[2][1]) -arr1[0][1]\*(arr1[1][0]\*arr1[2][2] - arr1[1][2]\*arr1[2][0])

+ arr1[0][2]\*(arr1[1][0]\*arr1[2][1] - arr1[1][1]\*arr1[2][0]) =! 0

-역행렬 구하는 방법

1. det값 구하기

2. 여인수행렬 구하기

3. 전치행렬 구하기

4. 전치행렬 \* det분 의 1 곱한 배열 = 역행렬