**Introduction to Computer and Lab**

**Homework #2**

**Due date: Apr 4, 2016**

**학번: 201404051**

**이름: 정 용 석**

**2. 화씨를 섭씨로 변환**

**1.1 Solution**

사용자에게 입력 받은 화씨 온도를 섭씨로 그리고 섭씨 온도를 화씨로 바꿔주는 것이 목적이다. 화씨와 섭씨 같은 경우는 대부분 정수로 표현이 되지만, 사실상 소수점과 더 관련이 있기에 두 변수를 float형으로 선언해주었다. 섭씨에서 화씨로, 화씨에서 섭씨로 계산하는 공식을 이용하여 각각의 변환 값을 계산하고 알맞은 변수에 할당하여 출력시켜 주었다.

**1.2. Source code**

**void** **computeTemperature**()

{

**float** F, C;

**printf**("화씨 온도를 입력하시오: ");

**fflush**(stdout);

**scanf**("%f" , &F);

C = (F-32)/(1.8); //섭씨를 구한다

**printf**("화씨 온도 %.0f는 섭씨 온도 %.2f 입니다.\n", F, C);

**printf**("섭씨 온도를 입력하시오: ");

**fflush**(stdout);

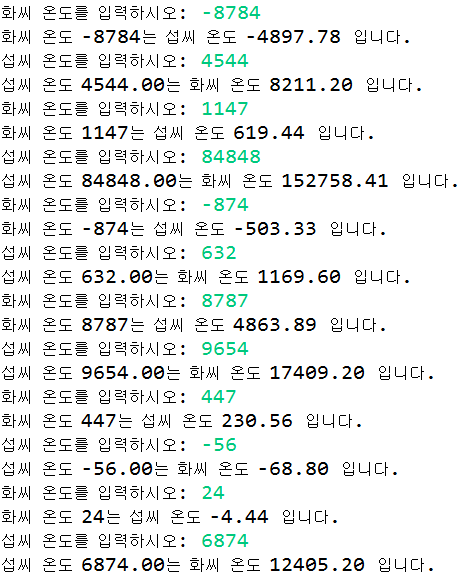
**scanf**("%f" , &C);

F = (C \* 1.8) + 32; //화씨를 구한다

**printf**("섭씨 온도 %.2f는 화씨 온도 %.2f 입니다.\n", C, F);

}

**1.3. Result (snapshot)**

****

**2 화성까지의 시간 계산**

**2.1. Solution**

화성까지의 거리와 각 이동수단의 속도를 상수 형 변수로 선언하여 값이 변환되는 경우를 배제하였다. 여기서 역 추진 로켓의 속도 같은 경우는 km/h대신 km/s으로 나와있어서 이를 km/h 변환하여 값을 초기화했다. 걸리는 시간을 저장할 수 있는 변수를 float형으로 통일 선언하고, 각 이동수단을 통하여 걸리는 시간을 계산과 동시에 출력하는 방식을 선택했다.

**2.2. Source code**

**void** **computeHoursforMars**()

{

**const** **int** earthToMars = 75000000; //화성까지의 거리 75000000 km

//역추진 로켓 속도(11.9km/s - > (11.9 \* 3600 km/h)

**const** **float** rocketSpeed = (11.9 \* 3600);

**const** **float** plane = 850; //민간 항공기 속도 850km/h

**const** **float** F1 = 350; //경주용 자동차(F1) 속도 350km/h

**const** **float** KTX = 300; //KTX 열차 속도 300km/h

**float** time; //걸리는 시간

time = earthToMars / rocketSpeed;

**printf**("역추진 로켓을 이용했을때 화성까지 걸리는 시간: %.2f\n" , time);

time = earthToMars / plane;

**printf**("민간 항공기를 이용했을때 화성까지 걸리는 시간: %.2f\n" , time);

time = earthToMars / F1;

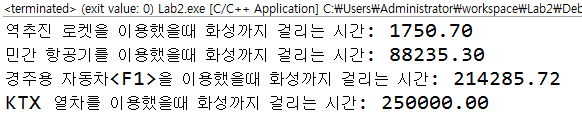
**printf**("경주용 자동차<F1>을 이용했을때 화성까지 걸리는 시간: %.2f\n" , time);

time = earthToMars / KTX;

**printf**("KTX 열차를 이용했을때 화성까지 걸리는 시간: %.2f\n" , time);

}

**2.3. Result (snapshot)**

****

**3 자릿수의 합이 7의 배수?**

**3.1. Solution**

사용자에게 입력 받은 수와 각 자리의 합을 저장할 변수를 선언한 뒤, 사용자에게 값을 입력 받는다. 여기서 의아했던 것은 만약 10000이상의 자리를 받았을 시 나타나는 오류 처리를 어떻게 해야 될 지 모르겠다. 일단 사용자가 그런 실수를 범하지 않는 다는 가정하에 1000의 자리부터 1의 자리까지 값을 한번에 구해서 합한 값을 구했다. 그리고 조건 문을 이용해서 7의 배수일 시와 아니 일 시의 출력 문을 작성했다.

**3.2. Source code**

**void** **computeDigitSum**()

{

**int** num, sum;

**printf**("숫자 <0 <= num <= 10000> 를 입력하시오: ");

**fflush**(stdout);

**scanf**("%d" , &num);

//각 자리수의 합 = 1000의 자리 + 100의 자리 + 10의 자리 + 1의 자리

sum = (num/1000)+((num%1000)/100)+(num%1000%100/10)+(num%1000%100%10);

**printf**("각 자리의 합은: %d\n" , sum);

//7의 배수 조건문

((sum%7)==0) ? **printf**("%d는 7의 배수입니다", sum):**printf**("%d는 7의 배수가 아닙니다", sum);

}

**3.3. Result (snapshot)**

****

**4 사분면 출력**

**4.1. Solution**

두 수 x, y를 저장하는 변수를 선언하고, 그 값을 사용자에게 입력 받는다. X와 y의 값이 양이나 음이냐 에 따라 변하는 조건 문을 각각 작성했다.

**4.2. Source code**

**void** **checkQuadrant**()

{

**int** x, y;

**printf**("2차원 공간읭 좌표 <x,y>를 입력하세요: ");

**fflush**(stdout);

**scanf**("%d %d", &x, &y);

//<x, y>조건 문

((x>0) && (y>0)) ? **printf**("<%d, %d>는 1 사분면에 속합니다\n", x,y):**printf**("<%d, %d>는 1 사분면에 속하지 않습니다\n", x,y);

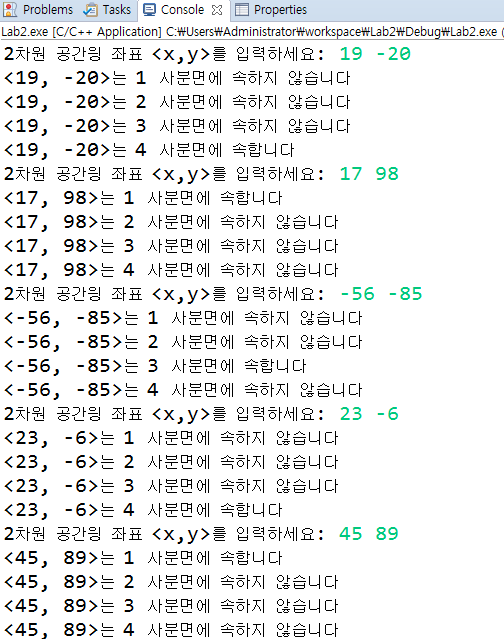
((x<0) && (y>0)) ? **printf**("<%d, %d>는 2 사분면에 속합니다\n", x,y):**printf**("<%d, %d>는 2 사분면에 속하지 않습니다\n", x,y);

((x<0) && (y<0)) ? **printf**("<%d, %d>는 3 사분면에 속합니다\n", x,y):**printf**("<%d, %d>는 3 사분면에 속하지 않습니다\n", x,y);

((x>0) && (y<0)) ? **printf**("<%d, %d>는 4 사분면에 속합니다\n", x,y):**printf**("<%d, %d>는 4 사분면에 속하지 않습니다\n", x,y);

}

**4.3. Result (snapshot)**

****

**5 평행 유무 확인**

**4.1. Solution**

**이 문제의 핵심은 기울기를 계산하는 데에 있다고 판단했다. 일단 4개의 점을 표현하는 좌표들을 변수로 선언하고, 2개의 점들이 이루는 선의 기울기를 계산하는 방식을 사용했다. 각각의 좌표는 모두 정수 형으로 받았지만, 기울기는 실수 형으로 선언함으로서, 평행 유무의 정확도를 높였다. 유저에게 모든 좌표에 대한 정보를 받은 뒤, 기울기를 구하고 평행 유무를 판단하는 조건 문을 작성함으로 마무리했다.**

**4.2. Source code**

**void** **checkParallelLine**()

{

**int** x1, x2, x3, x4, y1, y2, y3, y4;

**float** slope1, slope2;

**printf**("첫 번째 2차원 공간의 두 점 <x1, y1>, <x2, y2>를 입력하세요: ");

**fflush**(stdout);

**scanf**("%d %d %d %d", &x1, &y1, &x2, &y2);

**printf**("두 번째 2차원 공간의 두 점 <x3, y3>, <x4, y4>를 입력하세요: ");

**fflush**(stdout);

**scanf**("%d %d %d %d", &x3, &y3, &x4, &y4);

**printf**("<x1, y1>, <x2, y2>의 값은 <%d, %d>, <%d, %d>\n", x1, y1, x2, y2);

**printf**("<x3, y3>, <x4, y4>의 값은 <%d, %d>, <%d, %d>\n", x3, y3, x4, y4);

//각 직선의 기울기 계산

slope1 = (y2-y1)/(x2-x1);

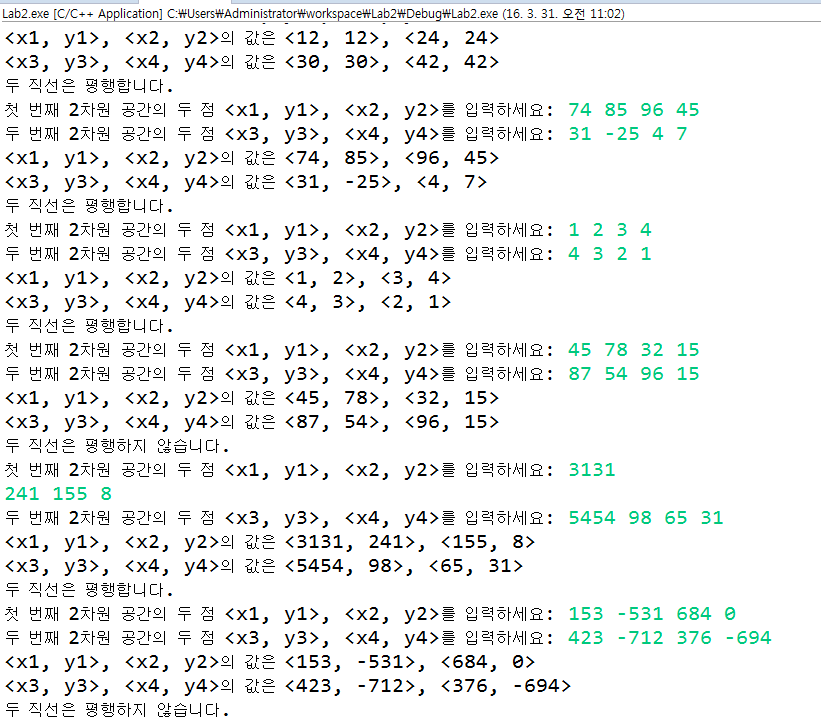
slope2 = (y4-y3)/(x4-x3);

//평행 조건문

(slope1 == slope2) ? **printf**("두 직선은 평행합니다.\n"):**printf**("두 직선은 평행하지 않습니다.\n");

}

**4.3. Result (snapshot)**

****