**Report 1 – 설명**

**20182931 신석경**

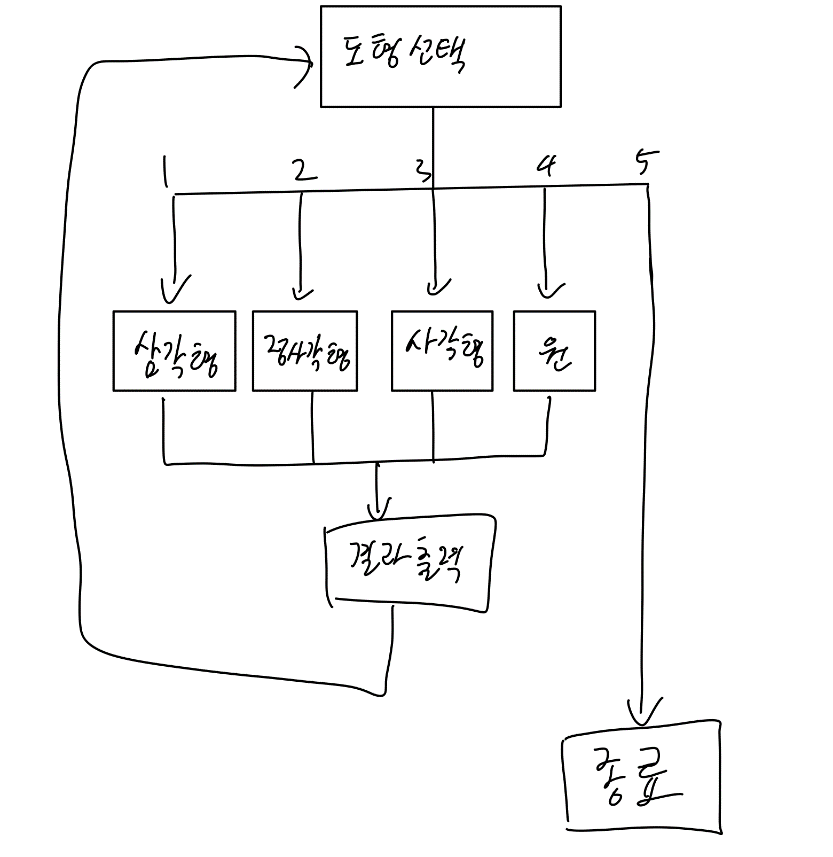
Report 1-1. 도형의 면적을 계산하는 프로그램을 쓰시오. 도형의 종류는 삼각형, 정사각형, 사각형, 원의 4가지로 한다.

이 프로그램은 반복적으로 도형의 종류를 입력 받고, 그 면적을 구하기 위한 정보를 입력 받으면 그에 해당하는 면적을 출력하는 프로그램이다. 이 프로그램의 메인 흐름은

1번. 도형의 종류를 입력 받는다.

2번. 각 도형에 맞게 필요한 정보를 입력 받는다.

3번. 결과를 출력하고 다시 1번 루프로 돌아온다.

프로그램의 동작 흐름을 살펴보면 옆의 그림과 같다. 1 ~ 4번에 해당되는 도형이 선택되면 각각에 해당되는 함수를 실행하여 결과를 출력하고 다시 도형을 선택하는 곳으로 돌아간다. 5번이 선택된 경우 루프를 빠져나와 프로그램이 종료된다.

이 프로그램은 반복적 루프이기 때문에 while문을 사용하였다. while문 내부에서 도형이 선택되면 if – else if- else 구문을 통해 각각의 도형에 맞춰 정보를 입력 받고 결과를 출력한다. 이 과정을 코드를 보면 살펴보자.

#define PI 3.141592 // 원주율 정의

float triangle(float width, float height); // 삼각형 넓이 계산 함수

float square(float length); // 정사각형 넓이 계산 함수

float rectangle(float width, float height); // 사각형 넓이 계산 함수

float circle(float radius); // 원 계산 함수

먼저, 프로그램에 필요한 매크로와 함수를 정의하였다. 원의 넓이를 계산하기 위한 PI를 정의하였고, 각각의 도형의 넓이를 계산하는 함수를 정의하였다.

float triangle(float width, float height) { // 삼각형 넓이 계산 함수

// 가로 세로를 입력 받아 삼각형 넓이 계산

return 0.5 \* width \* height; // 삼각형 넓이 계산

}

float square(float length) { // 정사각형 넓이 계산 함수

// 한 변의 길이를 입력 받아 정사각형 넓이 계산

return length \* length; // 정사각형 넓이 계산

}

float rectangle(float width, float height) { // 사각형 넓이 계산 함수

// 가로 세로를 입력 받아 사각형 넓이 계산

return width \* height; // 사각형 넓이 계산

}

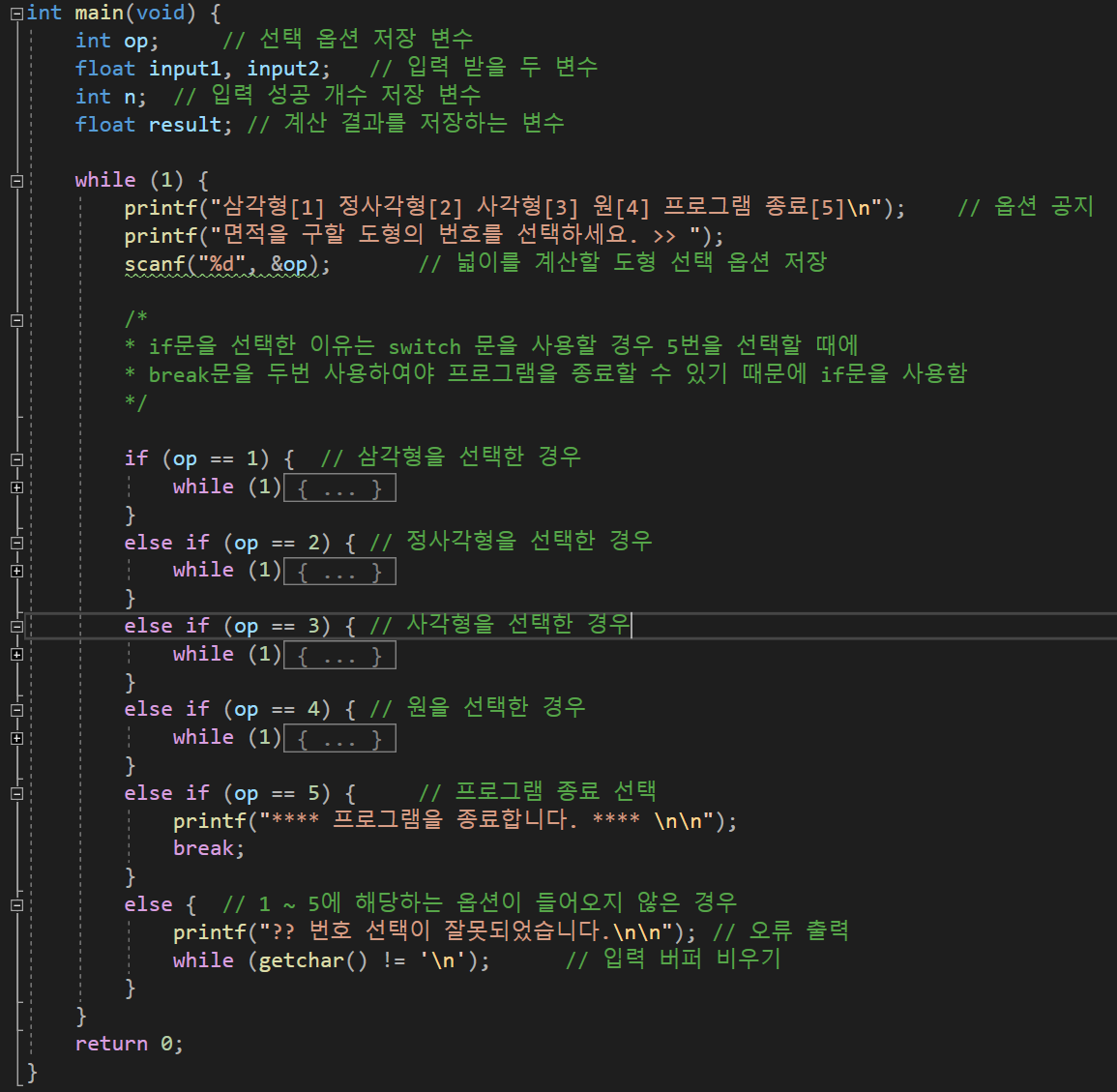
float circle(float radius) { // 원 계산 함수

// 반지름을 입력 받아 원 넓이 계산

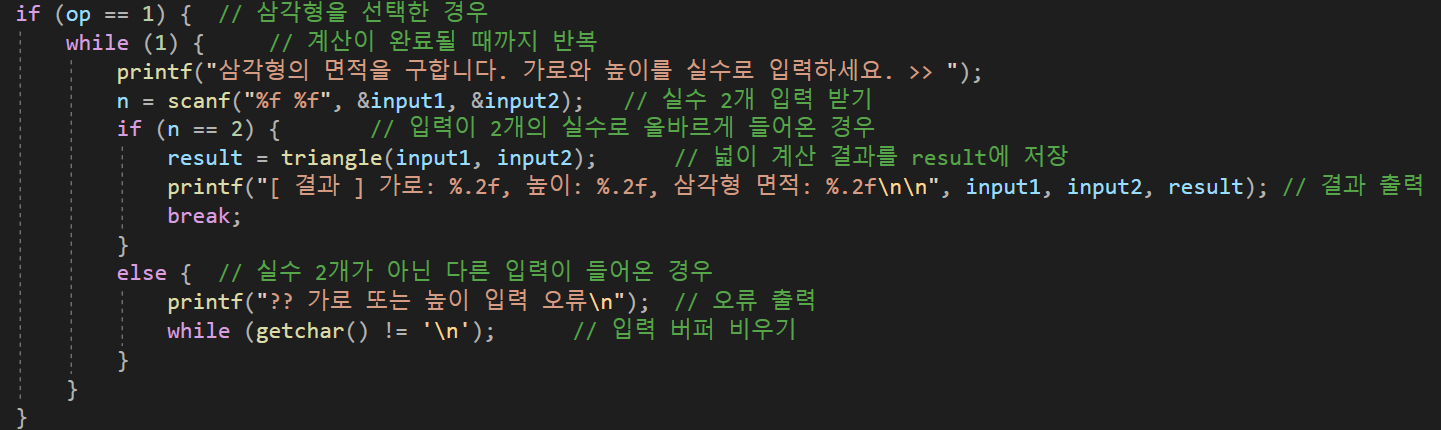
return PI \* radius \* radius; // 원의 넓이 계산, PI는 header에 정의

}

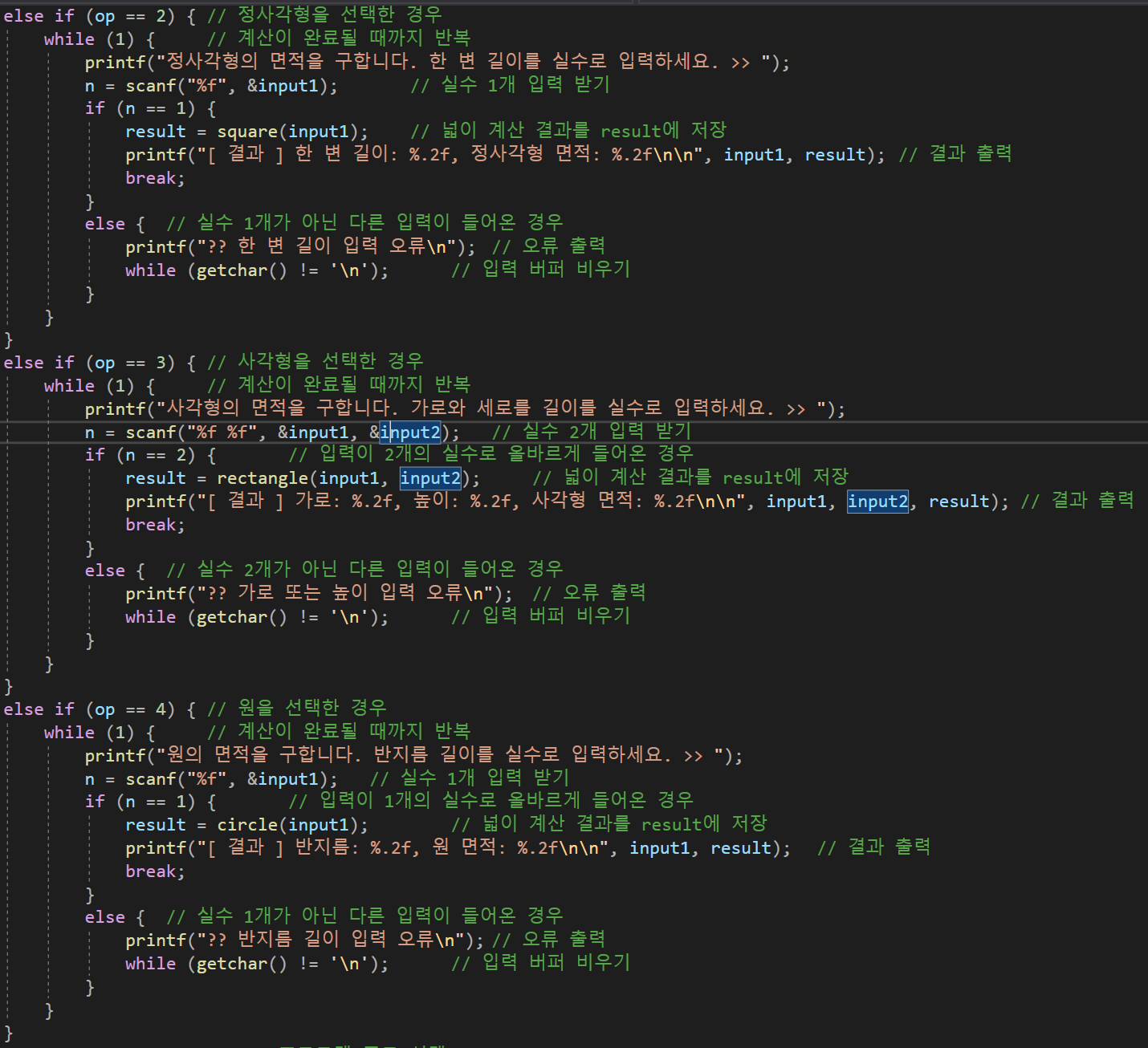
각각의 함수 구현이다. 입력된 파라미터를 바탕으로 각각의 넓이를 계산하여 반환한다.



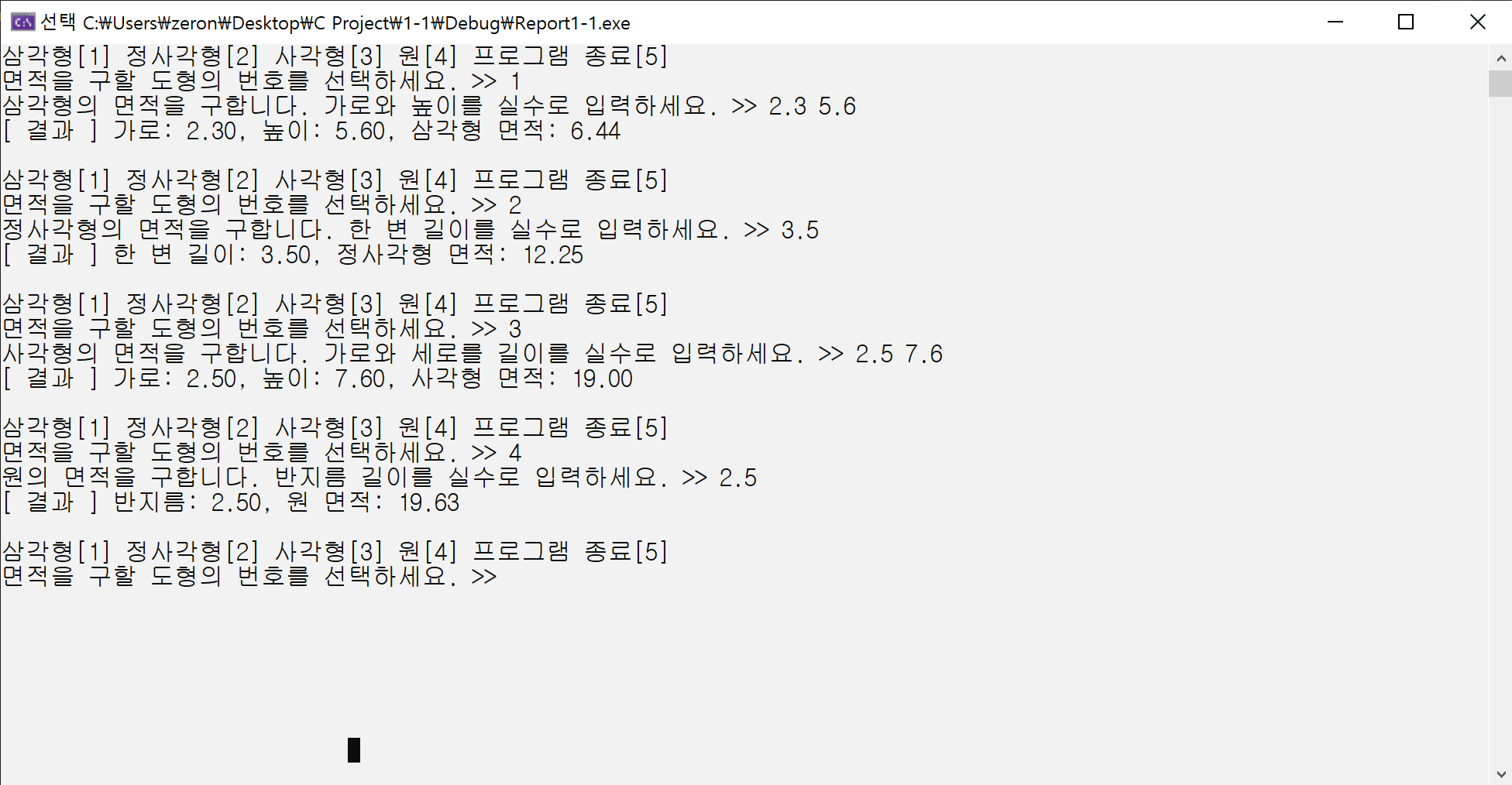
메인 함수이다. 변수는 총 5개이다. 도형을 선택하는 op변수와 각각에 대한 필요한 정보를 입력 받는 input1, input2가 존재한다. n의 경우, 도형 정보의 입력의 유효성을 판단하기 위해 사용되어졌다. 자세한 내용은 뒤에서 알아보겠다. 그리고 결과를 저장하는 result 변수가 있다. 메인 while문에서 도형의 입력을 받았을 때, 1 ~ 4까지는 도형에 관련된 선택이니 각각의 경우에 맞춰 if문을 실행하고, 5번의 경우에는 메인 while문을 빠져나오면서 프로그램이 종료, 예외의 것들이 입력되면 오류를 출력하고 입력 버퍼를 비우고 다시 메인 while문을 실행한다. 주석에도 적혀 있지만 보통 여러가지 옵션에 대해 선택할 때 switch 문을 많이 사용한다. 하지만 이 구현의 경우 switch문을 선택하지 않았는데 그 이유는 switch문은 프로그램을 종료할 때 break를 두 번 사용해야 하기 때문이다. 메인 while문에서 5번을 선택한 경우 switch문으로 구현하면 swtich문을 빠져나오기 위한 break를 한 이후, 다시 메인 while문을 빠져나오는 break문이 필요하다. 이 과정이 불필요한 작업인 것 같아 여기서는 if문으로 구현하여 하나의 break 문으로 메인 while문을 탈출하게 만들었다.



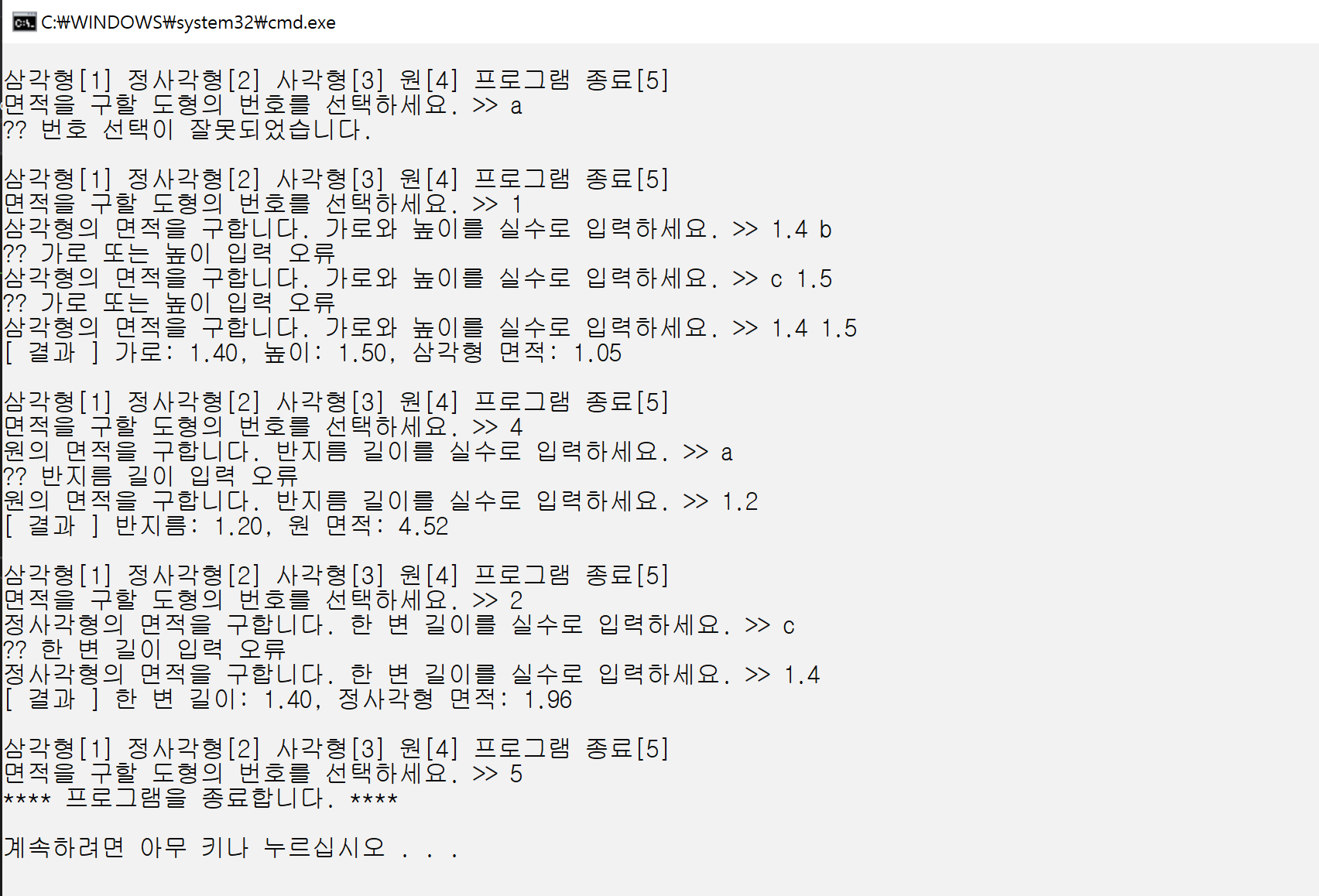
이제 각각의 도형 선택 이후 어떻게 동작하는지 알아보자. 삼각형이 선택되었다고 가정하자. 여기서는 서브 while문을 사용한다. 그 이유는 여기서도 올바른 정보의 입력을 판단하여 오류가 있을 경우 재입력을 받아야 하기 때문이다. 여기서 n 변수가 사용된다. scanf는 입력에 성공한 개수를 반환한다. scanf(“%f %f”, &input1, &input2)의 경우, float형 입력이 들어온 경우에만 성공적인 입력이고 이때 2의 결과를 반환한다. 즉, 위의 코드에서 n == 2인 경우는 float형 2개가 성공적으로 입력된 경우이다. 나머지 경우에는 오류를 출력하며 버퍼를 비우고 다시 입력을 받도록 하였다. n == 2인 경우, 즉 성공적인 입력이 되면 triangle(input1, input2)를 실행하여 면적을 계산해 result로 받는다. 그리고 결과를 출력하고 break를 사용해 서브 while문을 탈출하도록 한다. 그러면 if 문이 끝나고 다시 메인 while문으로 돌아가 도형 종류 선택을 받는다.



나머지 경우에도 비슷한 흐름과 코드로 구현되었다. 이제 정사각형과 원의 경우 1개의 정보만 필요하기 때문에 input1으로만 입력을 받았고 n도 1인 경우에 실행하도록 하였다. 이제 각각의 상황에 대한 결과를 보겠다.



위의 출력은 정상적인 출력이다. 1 ~ 4에 해당하는 프로그램의 흐름을 보여준다.



위의 첫번째 출력은 1 ~ 5에 해당하지 않는 경우가 입력 되었을 때, 오류를 출력하는 것이다. 2 ~ 4번째 출력은 도형 선택은 되었지만 그에 해당하는 정보의 입력이 제대로 되지 않을 경우 올바르게 입력될 때까지 반복해서 물어보고 올바른 결과를 출력하는 것이다. 마지막 출력은 5를 입력할 경우 프로그램이 종료되는 것을 보여준다.

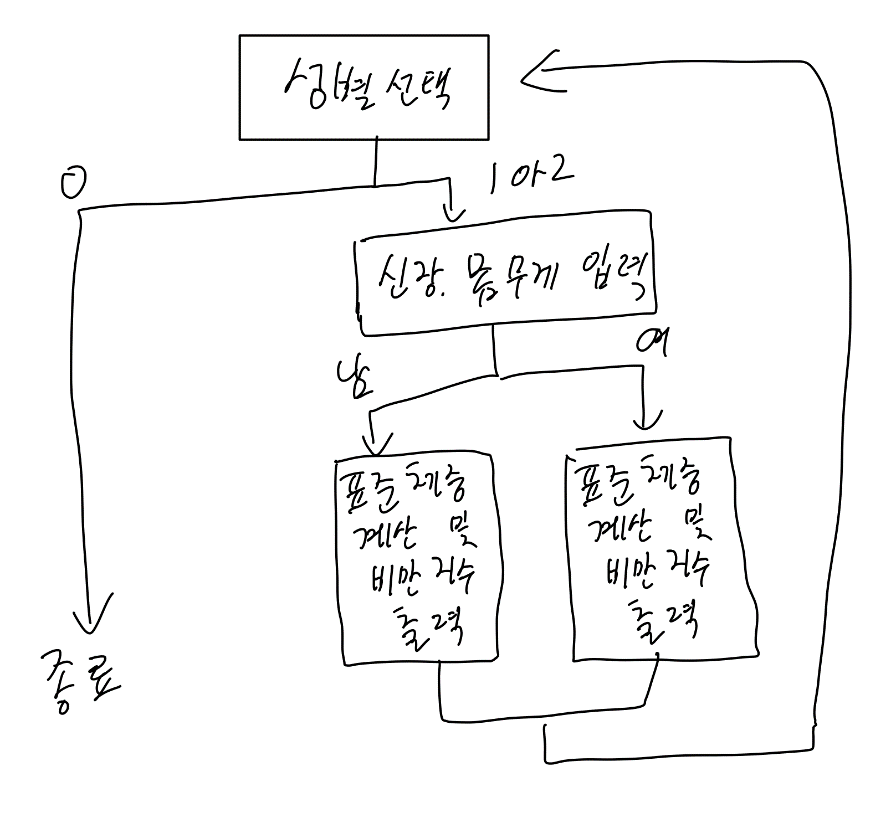
Report 1-2. 표준 입력으로 받은 성별, 신장(키), 몸무게를 이용하여 비만도를 출력하는 프로그램

이 프로그램은 반복적으로 성별을 입력 받고, 그에 해당하는 신장, 몸무게를 입력 받은 다음, 계산한 표준 체중을 이용하여 비만 지수를 결정해 그 결과를 출력하는 프로그램이다. 이 프로그램의 메인 흐름은

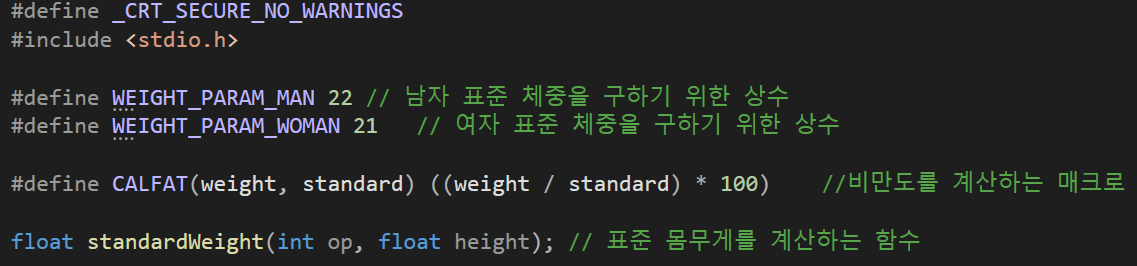
1번. 성별을 입력 받는다.

2번. 성별에 관련 없이 신장, 몸무게를 입력 받는다.

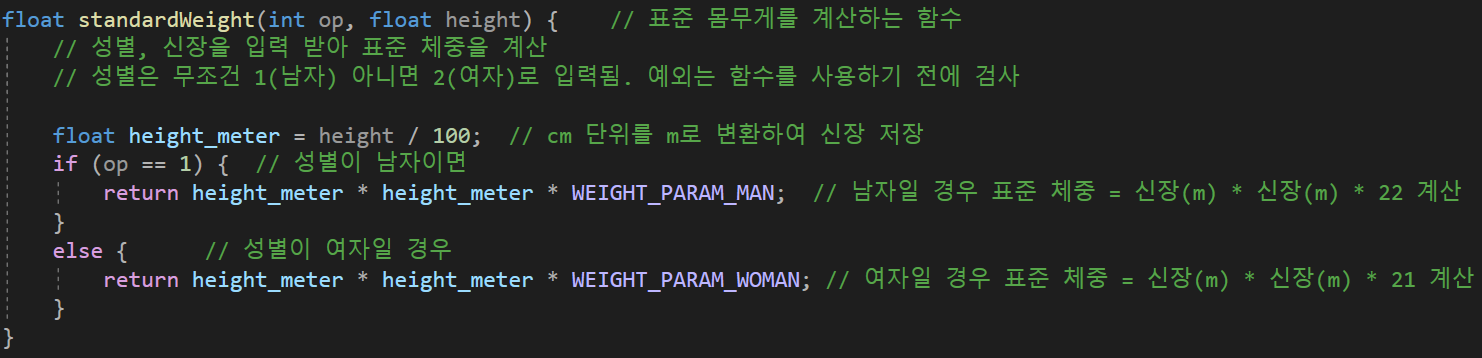
3번. 올바른 입력인 경우 성별에 따라 표준 체중을 구하고 비만 지수를 구해 그 결과를 출력하고 다시 1번으로 돌아온다.

프로그램의 흐름 그림이다. 성별을 선택하는 부분에서 0이 선택되면 프로그램을 종료하고, 1 혹은 2가 선택되면 신장, 몸무게를 입력 받는다. 이후, 입력된 정보를 가지고 각각 성별에 맞춰 결과를 계산하고 다시 성별을 선택한다.

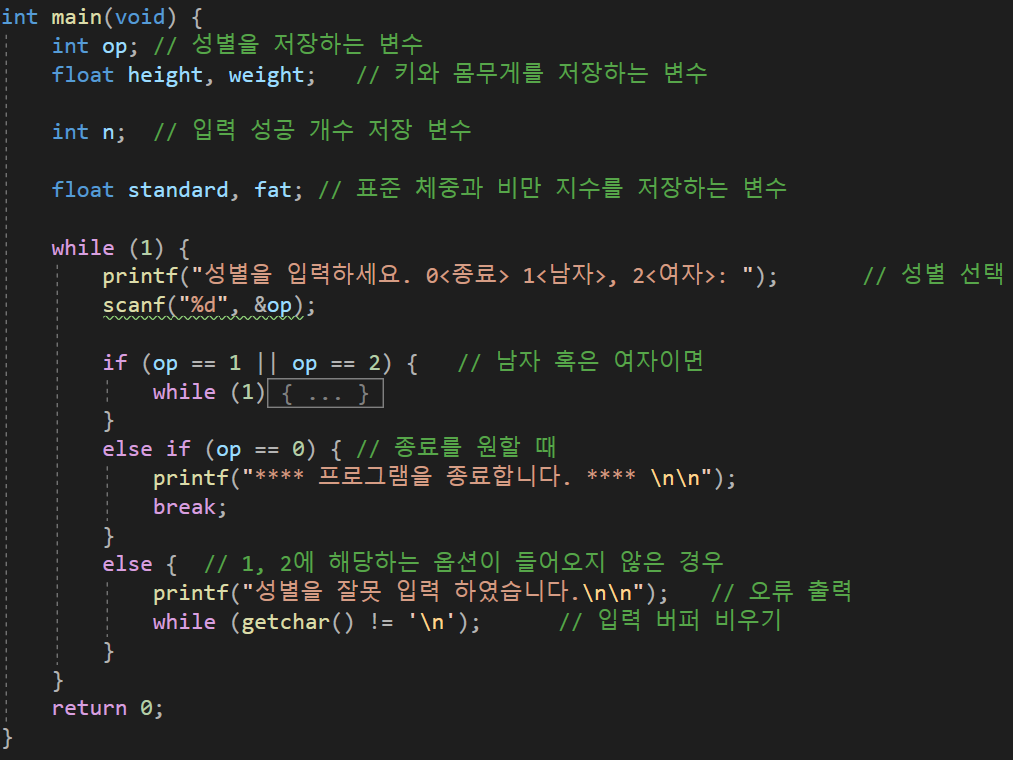
이 프로그램은 일정 패턴의 반복 실행이기 때문에 while문을 사용한다. 성별 선택과 성별에 따라 계산하고 비만 지수에 따라 결과를 출력해야 하기 때문에 3개의 if문이 사용된다. 자세한 것은 코드를 보며 알아보자.



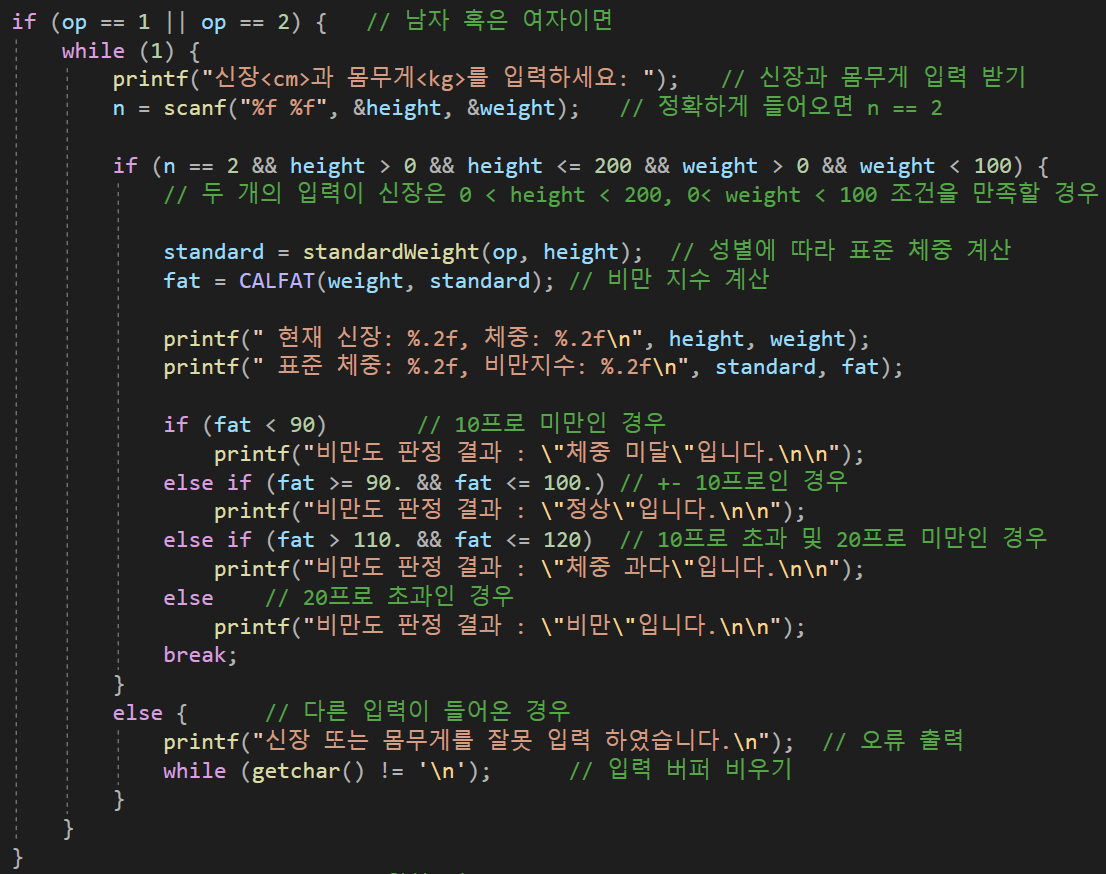
먼저, 정의된 매크로와 함수이다. 표준 체중을 계산할 때, 성별에 따라 상수를 다르게 하므로 그 상수를 정의하였다. 또한, 비만도를 계산하는 매크로 CALFAT을 정의하였다. 표준체중을 계산할 때에는 성별에 따라 상수가 달라지므로 if문을 사용하기 위해 함수로 정의하였다.



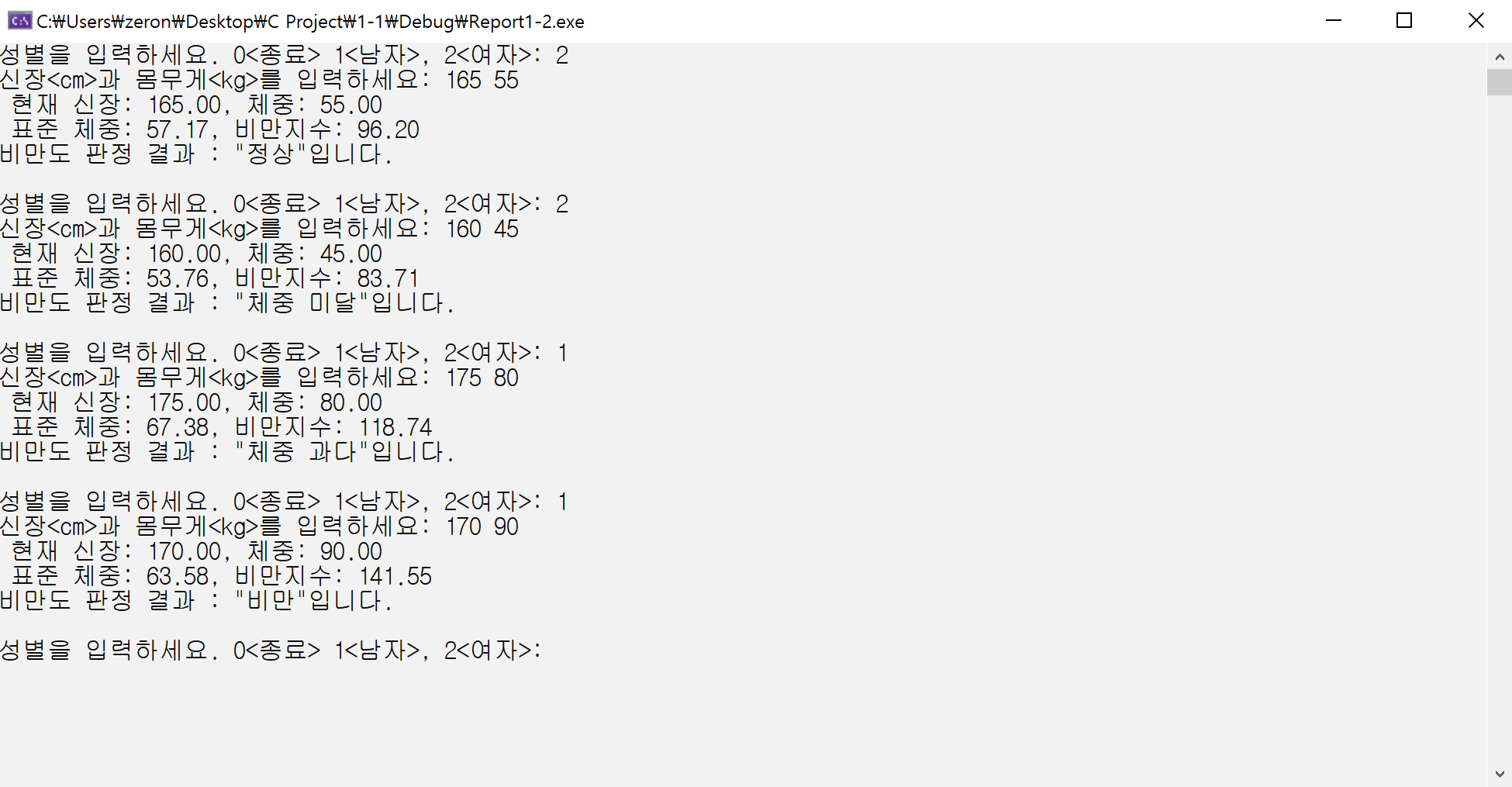
표준 체중을 계산하는 standardWeight 함수이다. 이 함수는 성별(op), 키(height)를 입력 받아 계산한 표준 체중을 반환하는 함수이다. 먼저 입력의 height는 cm단위이지만, 표준 체중 계산에 사용되는 단위는 m단위이므로 height\_meter 변수를 선언해 m로 변환한다.이후, 입력된 성별에 의하여 표준 체중을 계산한다. 이때, 남자는 if(op ==1)으로 검사했지만 여자의 경우, 1이 아니면 2이기 때문에 else로 처리한다. 함수를 실행하기전 오류 검사를 진행하기 때문에 op에 1과 2 이외의 값이 들어올 수 없다.



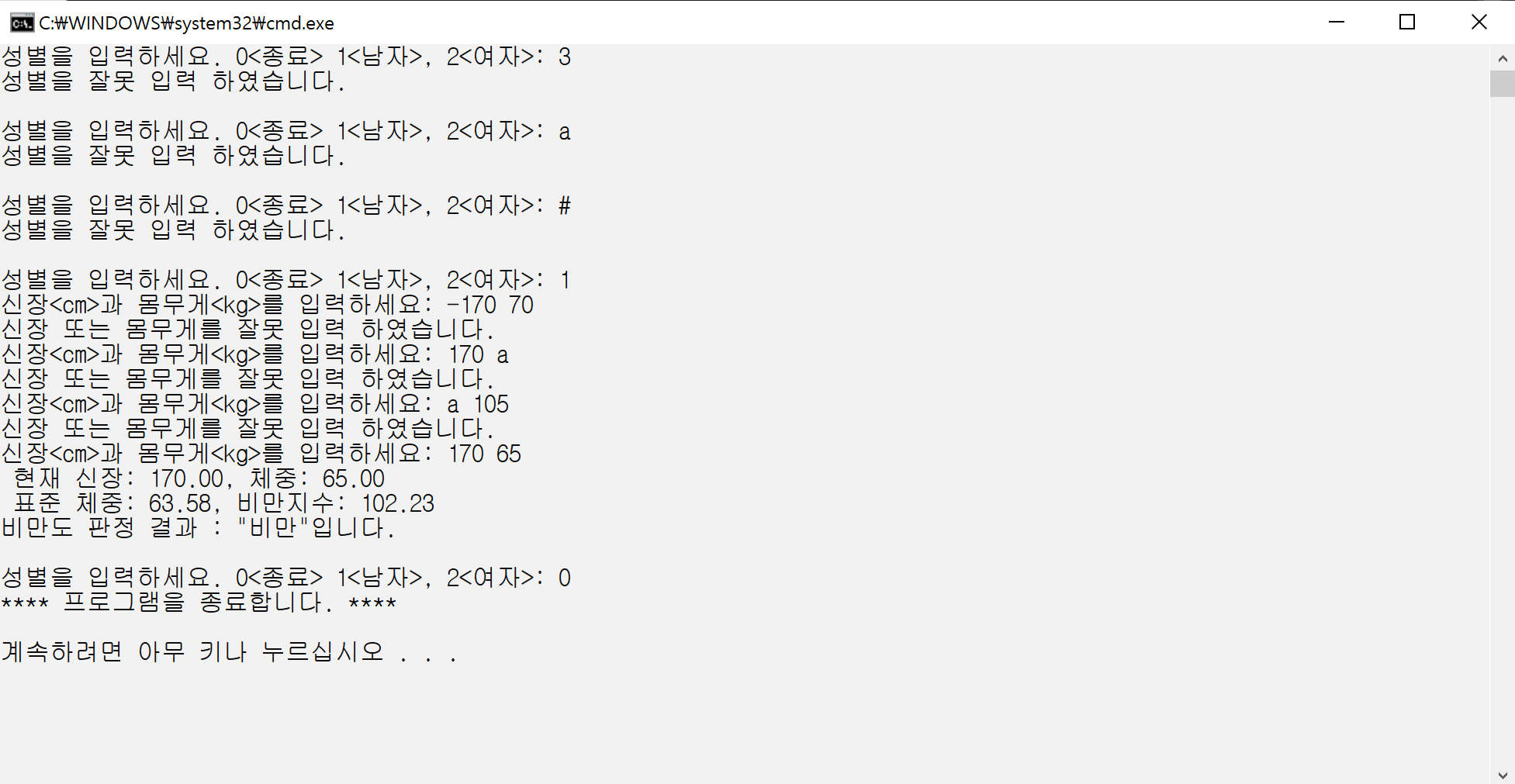
메인 함수이다. 총 6개의 변수가 있다. op는 성별을 저장하는 변수이고 height와 weight는 각각 키와 몸무게를 저장한다. standard와 fat는 표준 체중과 비만도를 저장한다. n의 경우 1-1에서 설명한 것처럼 입력이 성공적인지를 판단하는 변수이다. 프로그램이 시작하면 성별을 입력 받는다. 이때, 유효한 입력은 0, 1, 2이다. 이 입력된 op가 1 혹은 2이면 표준 체중과 비만도를 계산해 출력하는 if 문을 실행한다. 0일 경우에는 프로그램을 종료한다. 이외에 나머지가 입력되면 잘못 입력되었기 때문에 버퍼를 비우고 다시 입력을 받는다. 여기서도 마찬가지로 switch 구문을 사용하지 않고 if - else if – else로 구현하였다.



if문에서 남자와 여자를 따로 구별하지 않았다. 그 이유는 남자와 여자를 선택했을 때 달라지는 것은 계산 밖에 없기 때문에 그것은 standardWeight 함수에서 처리하도록 하였다. 그 이외에 정보를 받는 과정과 비만도의 결정은 동일하기 때문에 같은 if문에서 처리하도록 하였다. 이 if문 속에서도 잘못된 입력에 대한 처리를 하고 재입력을 받아야 하기 때문에 서브 while문을 만들었다. 신장과 몸무게를 입력 받고 그에 해당하는 조건을 검사한다. 신장과 몸무게가 둘 다 실수로 들어오면 n == 2가 된다. 만약 ‘a’ 혹은 ‘#’과 같은 실수가 아닌 다른 것이 입력 되면 n != 2가 되기 때문에 else 구문을 실행한다. 그러면 오류를 출력하고 버퍼를 비워 다시 입력을 받게 된다. 두 번째 경우는, 실수는 입력되었지만 그 범위가 다른 경우이다. 문제에서 신장은 2m(200cm)이하, 몸무게는 100kg 이하로 제한을 두고 있다. n == 2일지라도, 해당 조건에 만족하지 않으면 이때에도 else 구문을 실행하여 다시 입력 받는다. 입력이 제대로 되면 그에 따라 표준 체중을 계산한 결과를 standard에 저장하고, 그 결과를 가지고 비만 지수를 계산하여 fat에 저장한다. 이후, fat의 범위에 따라 비만도 판정 결과를 출력하고 break를 통해 서브 while문을 탈출하므로 if(op == 1 || op == 2) 구문을 끝내고 다시 메인 while문을 실행한다. 각각의 상황에 대한 출력을 보겠다.



위의 출력은 정상적으로 돌아가는 경우의 출력이다.



1 ~3번째 출력은 성별 입력에서 0, 1, 2에 해당하지 않는 경우 오류를 출력하는 모습이다. 4번째 출력은 신장과 몸무게가 제대로 입력이 되지 않는 경우 다시 입력을 받는 모습이다. 마지막 출력은 성별을 고를 때 0을 입력하면 프로그램이 종료되는 모습이다.

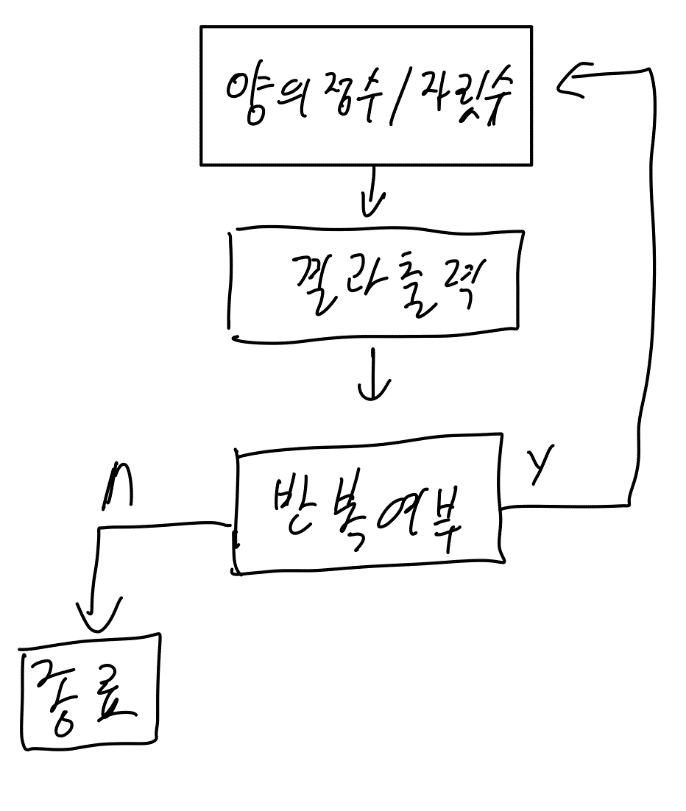
Report 1-3. 임의의 양의 정수와 자릿수 정수(n)을 입력 받아 n의 자릿수에서 반올림하는 프로그램

이 프로그램은 반복 여부에 따라 양의 정수와 자릿수 정수를 입력 받아 해당되는 자릿수에서 반올림한 결과를 출력하는 프로그램이다. 이 프로그램의 흐름은

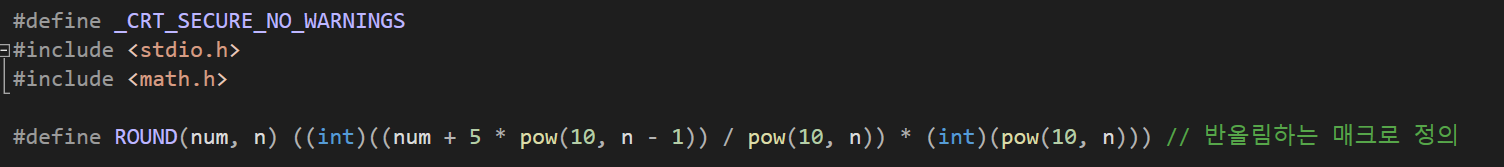
1번. 임의의 양의 정수와 자릿수 정수를 입력 받는다.

2번. 결과를 출력하고 반복 여부를 입력 받는다.

3번. 반복 여부에 따라 프로그램을 종료 혹은 1번으로 돌아간다.

프로그램의 흐름을 그림으로 나타낸 것이다. 임의의 양의 정수와 자릿수를 입력 받으면, 그에 해당하는 결과를 출력하고 반복 여부를 묻는다. 이때, ‘n’을 입력하면 프로그램을 종료하고 ‘y’를 입력하면 다시 양의 정수와 자릿수를 입력 받는다.

일정 패턴의 반복을 구현하기 위해 while문을 사용하였다. 또한, 여러 조건들을 검사해야 하고, 입력에 따라 처리하기 때문에 if구문을 사용하였다. 자세하게 코드를 보며 알아보자.

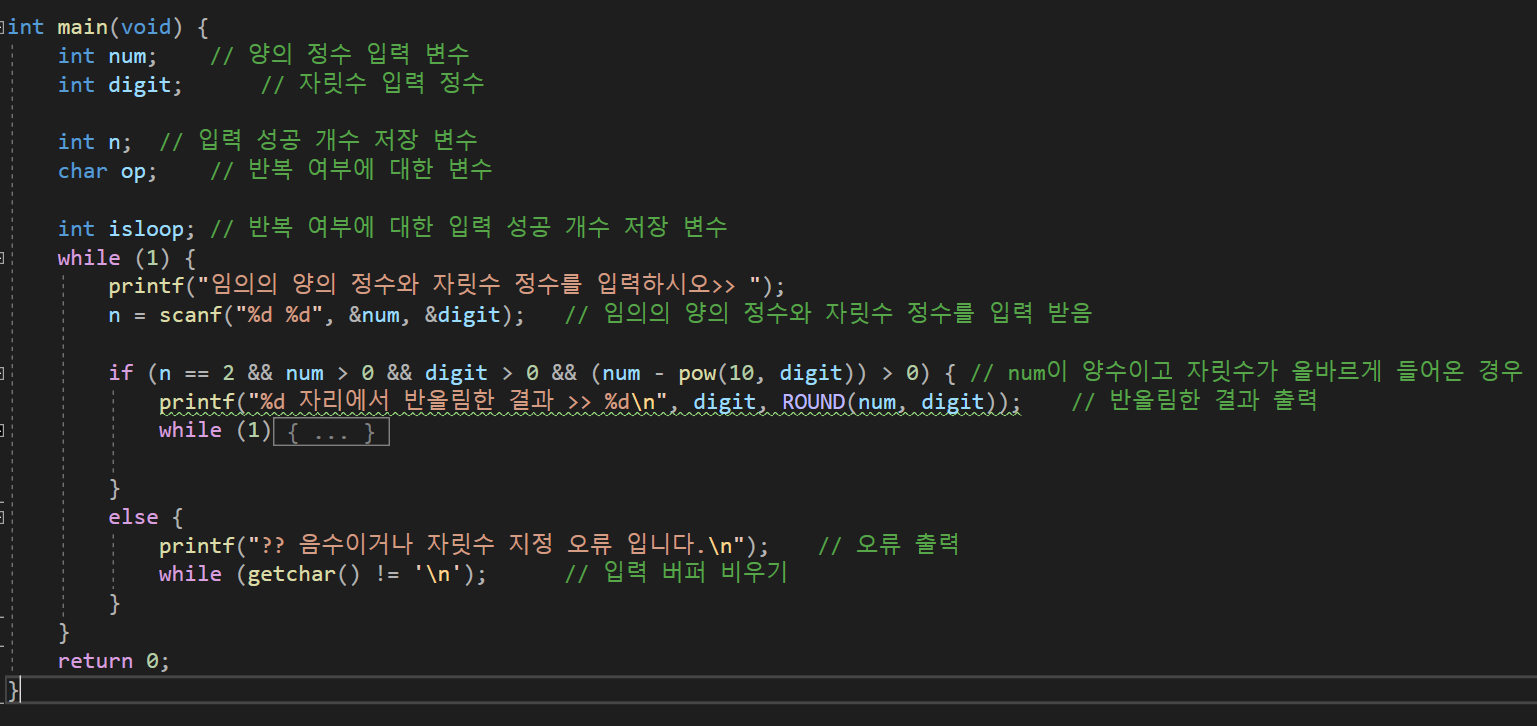


거듭제곱을 표현하기 위한 함수로 pow를 사용하기 위해 라이브러리에서 math.h를 include 하여 사용했다. 그리고 반올림하는 매크로를 정의하였다. 이는 수학적 이해가 필요하다. 수학적 개념으로 특정 자릿수에서 반올림은 그 전 자릿수에 해당하는 10의 거듭제곱 수에 5를 곱하여 더한 이후, 원래 자릿수에 해당하는 10의 거듭제곱 수로 몫 나누기를 하면 반올림 결과의 숫자부분이 나온다. 이 수에 자릿수만큼의 10의 거듭제곱을 곱하면 최종 반올림 결과가 나온다. 수학적으로 더 살펴보자.

먼저 13을 1번 자릿수에서 반올림해보자. 지금 1번 자릿수에서 반올림 하기에 그 전 자릿수는 0번째이다. 10의 0승은 1이므로 13 + 5 \* 1을 하면 18이 나온다. 이를 10의 1승으로 몫 나누기를 하면 1이 나오고, 다시 10의 1승을 곱하면 10이 나오므로 반올림이 완료되었다.

또 다른 예시를 보면 1462가 있고 2번 자릿수에서 반올림 한다고 가정하자. 현재 2번 자릿수의 전 자릿수는 1번째 이므로, 10의 1승 \* 5 = 50을 더한다. 그러면 1512가 나오고 이를, 10의 2승으로 나눈다. 그러면 15가 나오게 된다. 여기에 다시 10의 2승을 곱하면 1500이 나오므로 반올림이 완료 되었다.

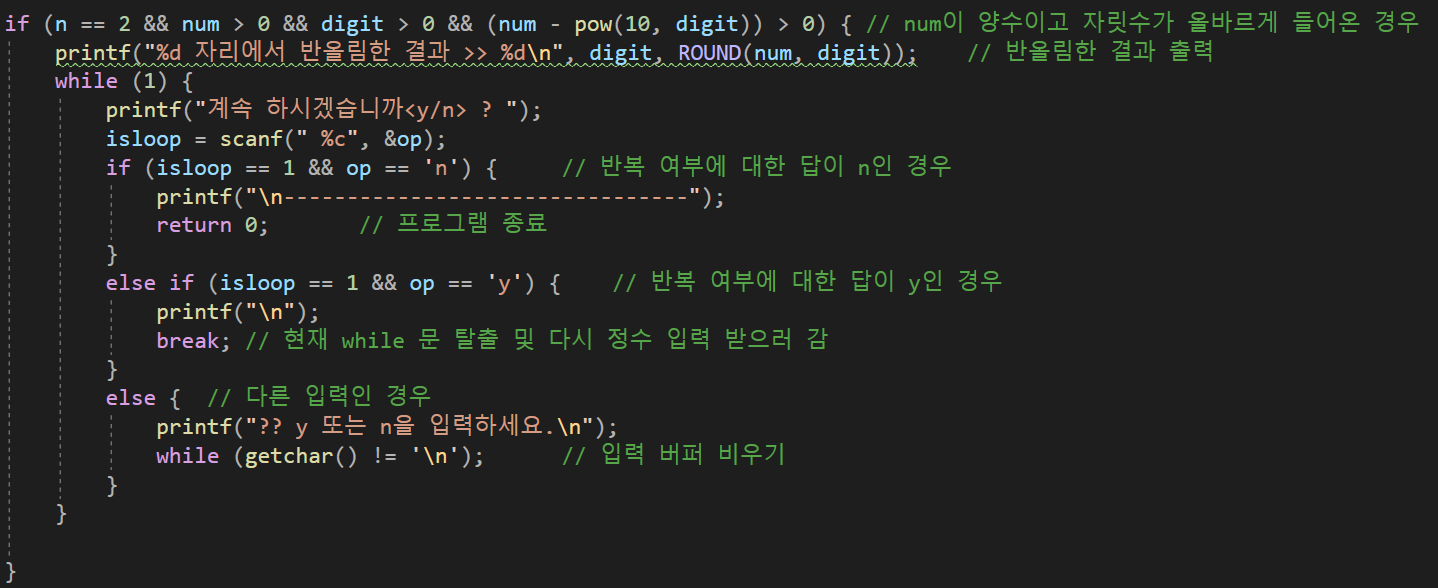
이 과정을 코드로 구현하였다. num + 5 \* pow(10, n-1)을 통해 전 자릿수에 해당하는 10의 거듭제곱 수에 5를 곱하여 num에 더한다. 이를 pow(10, n)으로 나누어 자릿수에 해당하는 10의 거듭제곱 수로 나눈다. 이때, pow는 double 형을 반환하기 때문에 몫 계산이 되지 않고 소수점까지 계산이 된다. 이를 해결하기 위해 int로 형변환을 하여 소수점 부분을 없앤다. 이를 다시 해당되는 10의 거듭제곱 수, int형으로 바꾼 pow(10, n)을 곱하여 반올림을 완료한다.



메인 함수이다. 총 5개의 변수가 사용된다. 양의 정수와 자릿수를 저장하기 위한 num과 digit 변수, 반복 여부 저장을 위한 op변수가 있다. n변수의 경우 임의의 양의 정수와 자릿수 정수를 받을 때, 성공적으로 받아졌는지 판단하기 위한 변수이고, isloop변수는 프로그램의 반복 여부를 물을 때, 올바른 입력이 들어왔는지를 판단하는 변수이다.

프로그램이 시작하면 먼저 양의 정수와 자릿수 정수를 입력 받는다. 이때, 올바르게 입력 되면 n == 2이다. 그리고 문제에서 ‘양’의 정수를 제한했기 때문에 num > 0 조건도 동시에 검사한다. 또한, 올바른 자릿수를 검사하기 위해 digit > 0도 검사하고, num – pow(10, digit)이 양수인지 검사한다. 만약 입력된 양의 정수를 초과하는 자릿수가 입력되면 num – pow(10, digit)이 음수가 나오기 때문이다. 위에 해당하지 않는 입력이 들어오면 오류를 출력, 버퍼 비우기를 실행하고 다시 양의 정수와 자릿수 정수를 입력 받는다.

올바른 조건에 해당하면 반올림한 결과를 출력하고 서브 while문에서 반복 여부를 묻고 여부에 따라 프로그램을 종료한다.

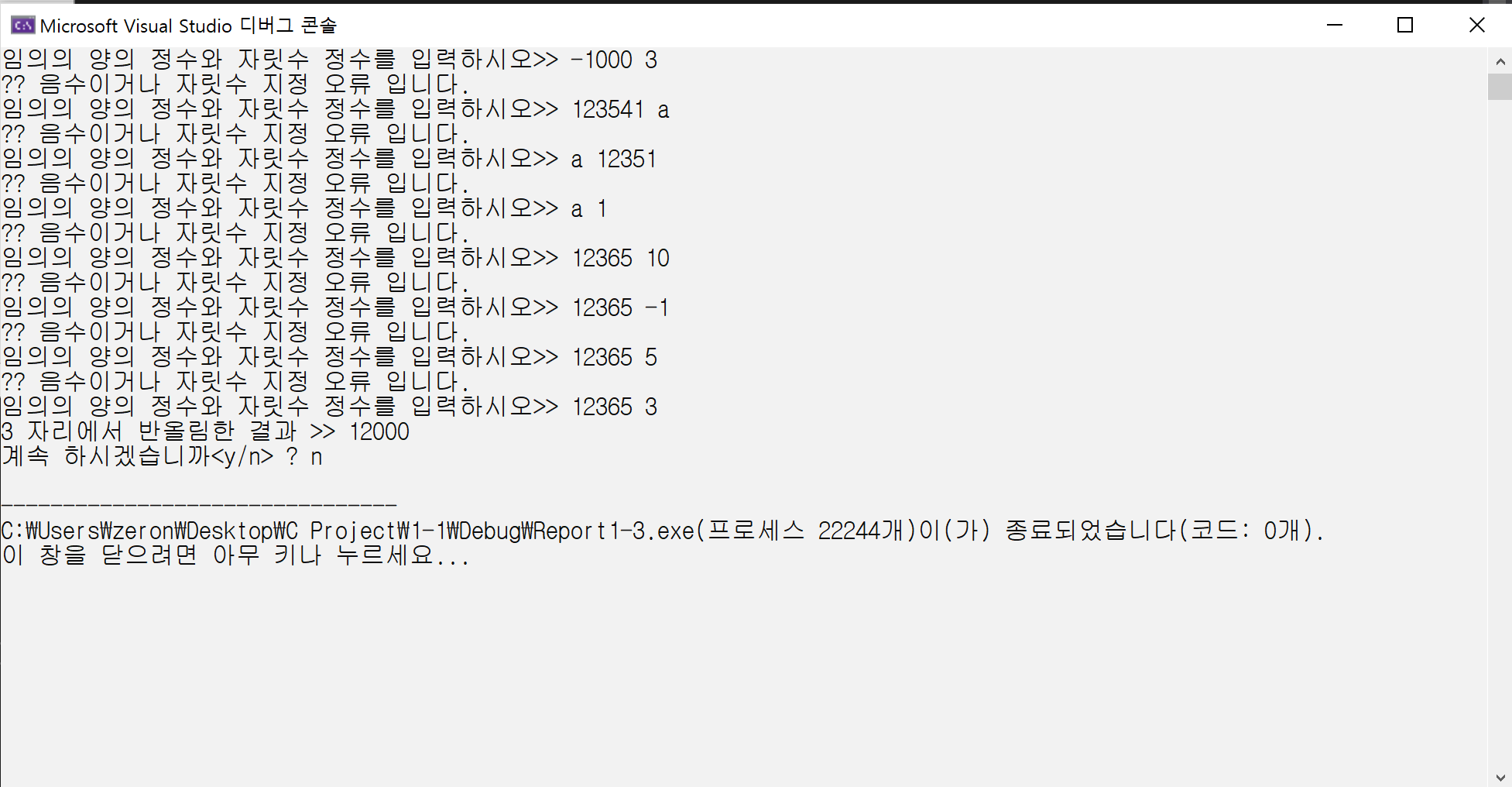


서브 while문이다. 올바른 조건이면 먼저 반올림한 결과를 출력하고 반복 여부를 확정하기 위한 서브 while문으로 들어간다. 이때, 올바르게 문자가 입력되고 ‘n’이면 프로그램을 종료, ‘y’이면 서브 while문을 종료하고 if문이 종료되므로 메인 while문으로 돌아가고, 다른 입력인 경우에 오류를 출력하고 다시 입력을 받는다. 이때, ‘n’이 입력된 경우 곧바로 return 0를 통해 프로그램을 종료하는데, 기존의 1-1, 1-2에서는 break를 통해 메인 while문을 나갈 수 있었기 때문에 break를 사용했다. 그러나 이 경우 ‘n’이 입력되었을 때, break를 사용한다면 서브 while문만 탈출하므로 메인 while문에서 다시 한번 검사를 하는 수고로움이 생긴다. 그래서 그냥 ‘n’이 입력되면 프로그램이 종료되도록 하였다.

결과창을 보도록 하자.



위의 결과는 정상적인 출력이다.



1번 출력은 양의 정수가 아닌 경우 오류를 출력하는 모습이고, 2번 출력은 자릿수의 입력이 올바르지 않은 경우 오류이다. 3 ~ 4번 출력은 양의 정수가 아닌 다른 것이 들어올 때의 오류이고, 5 ~ 7번 출력은 올바르지 않은 자릿수가 들어왔을 때의 오류이다. 마지막 출력은 n을 입력 했을 때 프로그램이 종료되는 것을 보여준다.

Report 1-4. 표준 입력으로 받은 두 실수의 사칙 연산을 수행하는 프로그램

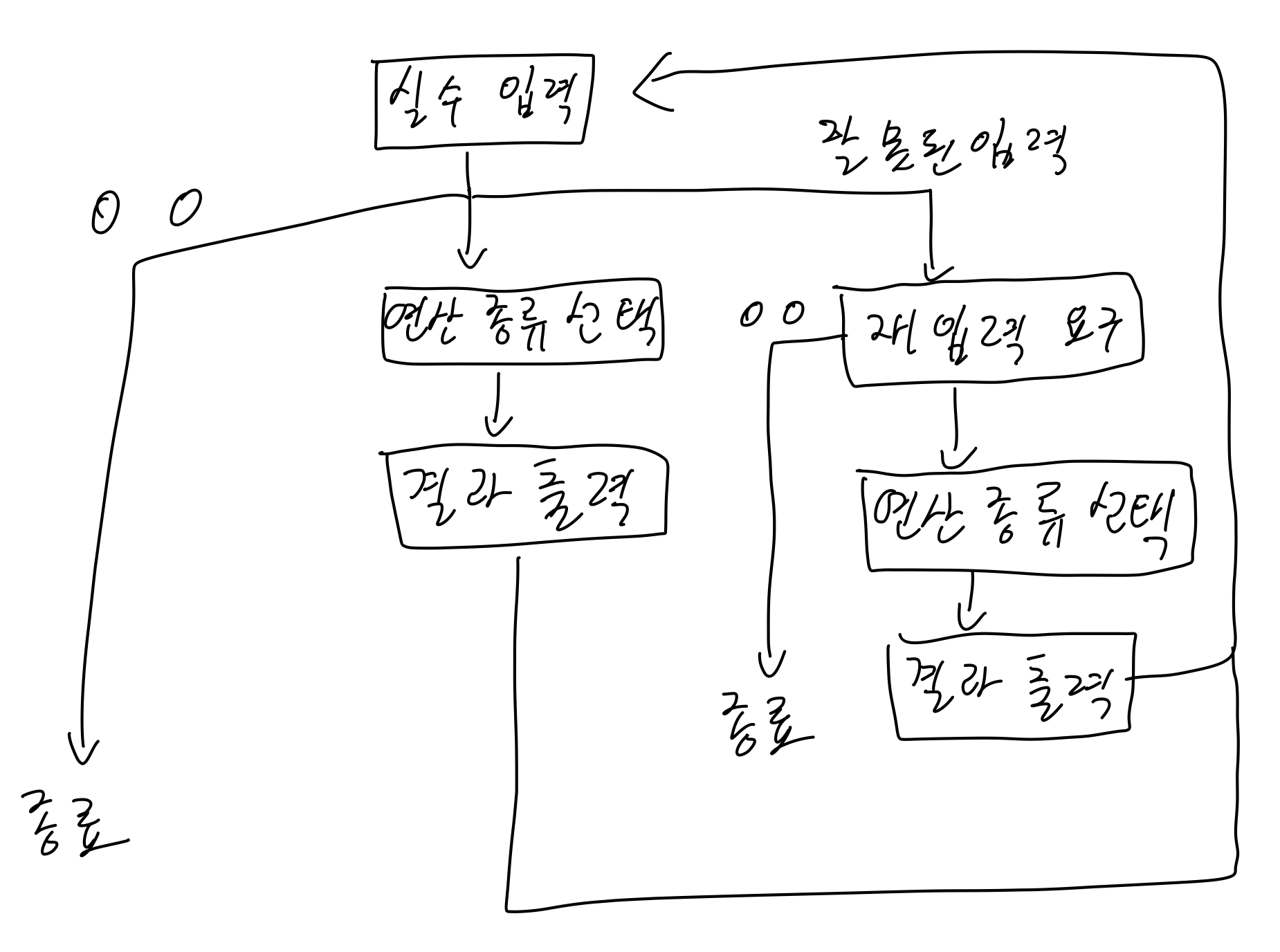
이 프로그램은 반복적으로 두 실수를 입력 받고, 연산 종류를 입력 받아 해당하는 연산을 수행하는 프로그램이다. 이 프로그램의 흐름은

1번. 두 실수를 입력 받는다.

2번. 연산 종류를 선택한다.

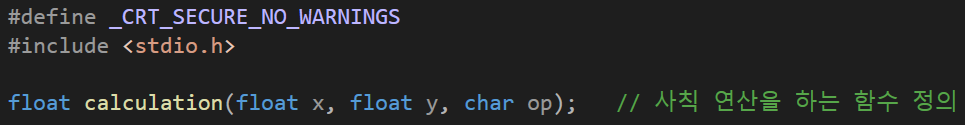
3번. 결과를 출력하고 1번으로 돌아온다.

4번. 입력된 두 수가 0 0 이면 프로그램을 종료한다.

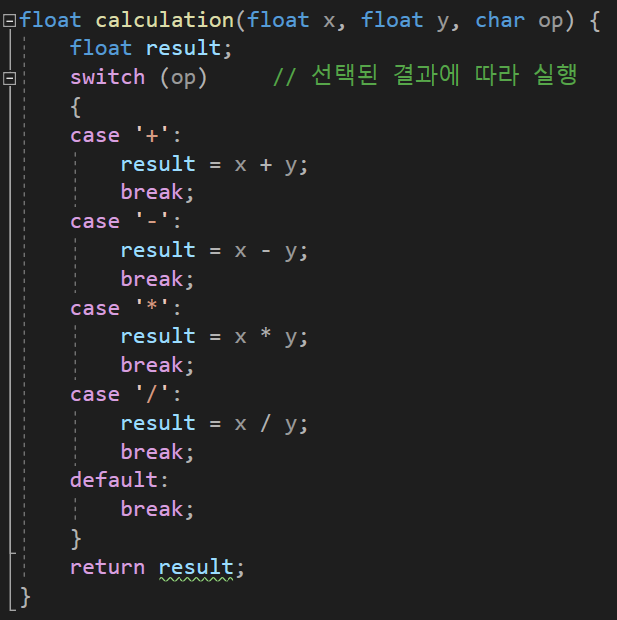


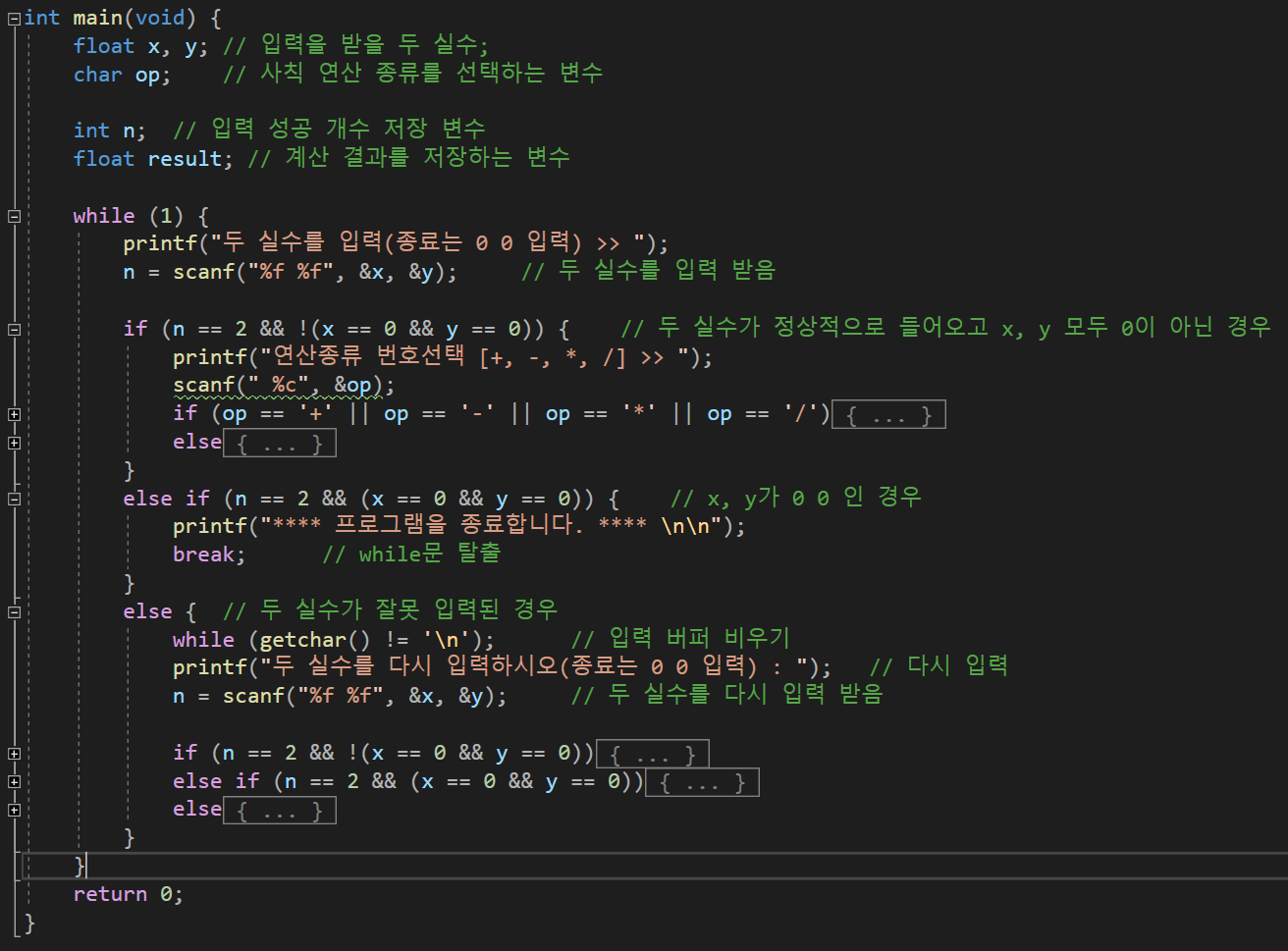
프로그램의 흐름을 그림으로 나타낸 것이다. 먼저 실수를 입력 받는다. 입력이 0 0 이라면 프로그램을 종료한다. 입력이 올바르게 되었으면 그 다음 연산 종류를 선택한다. 선택된 연산의 결과를 출력하고 다시 처음으로 돌아가 실수를 입력 받는다. 만약 실수를 입력 받을 때 잘못된 입력이 되면 재입력을 요구한다. 이때도 0 0을 입력 받으면 프로그램을 종료한다. 이번에도 잘못된 입력을 받으면 다시 처음으로 돌아간다. 만약 올바르게 입력이 된다면 연산 종류를 선택하고 그에 따른 결과를 출력한 이후 다시 처음으로 돌아간다.

이 흐름을 구현하기 위해 while문과 if문이 사용되었다. 그리고 연산 부호에 따라 연산을 수행하기 위해 switch문이 사용되었다. 자세한 것은 코드를 보며 알아보자.



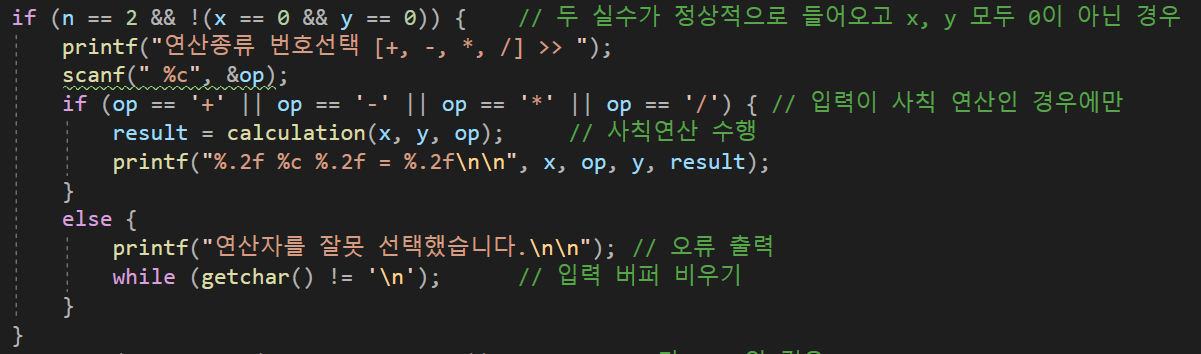
사칙 연산을 수행하기 위한 함수이다. 입력된 두 수 x, y를 가지고 op에 해당하는 연산을 실행한다.

해당하는 함수의 구현 부분이다. 결과를 저장하기 위해 result라는 float 변수를 선언하였고 switch문을 사용하여 입력된 op에 따라 각각에 해당하는 연산을 실행한다. 계산된 결과 result를 반환하므로 함수 내부에서 사칙 연산이 완료된다.



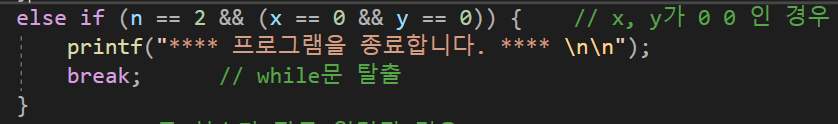
메인 함수이다. 총 5개의 변수가 사용된다. x, y는 입력된 두 실수를 저장하기 위한 변수이고, op는 연산 종류를 저장하기 위한 변수, result는 계산 결과를 저장하는 변수이다. n의 경우 입력의 성공 여부를 위해 사용된다.

프로그램이 시작되면 두 실수를 입력 받는다. 올바르게 입력을 받으면 n == 2가 된다. 이때, x == 0이 아니고 y == 0도 0 이 아니면 올바른 실수 입력이므로 연산을 실행하고 결과를 출력한다. 만약 올바른 실수이지만 x == 0 y == 0이면 프로그램을 종료하므로 메인 while문을 탈출한다. 만약 올바르지 않은 입력이면 버퍼를 비우고 두 실수를 다시 입력하라는 메세지와 함께 다시 실수를 입력 받는다. 이때도 위와 마찬가지로 올바르게 입력되고 !(x==0 && y==0)인 경우에는 연산 종류를 선택하고, x == 0 && y == 0이면 break를 통해 메인 while문을 빠져나온다. 만약 이때에도 실수가 잘못 입력되면 오류 출력과 함께 메인 while문의 처음으로 돌아간다. 자세한 것은 코드를 보며 알아가겠다.

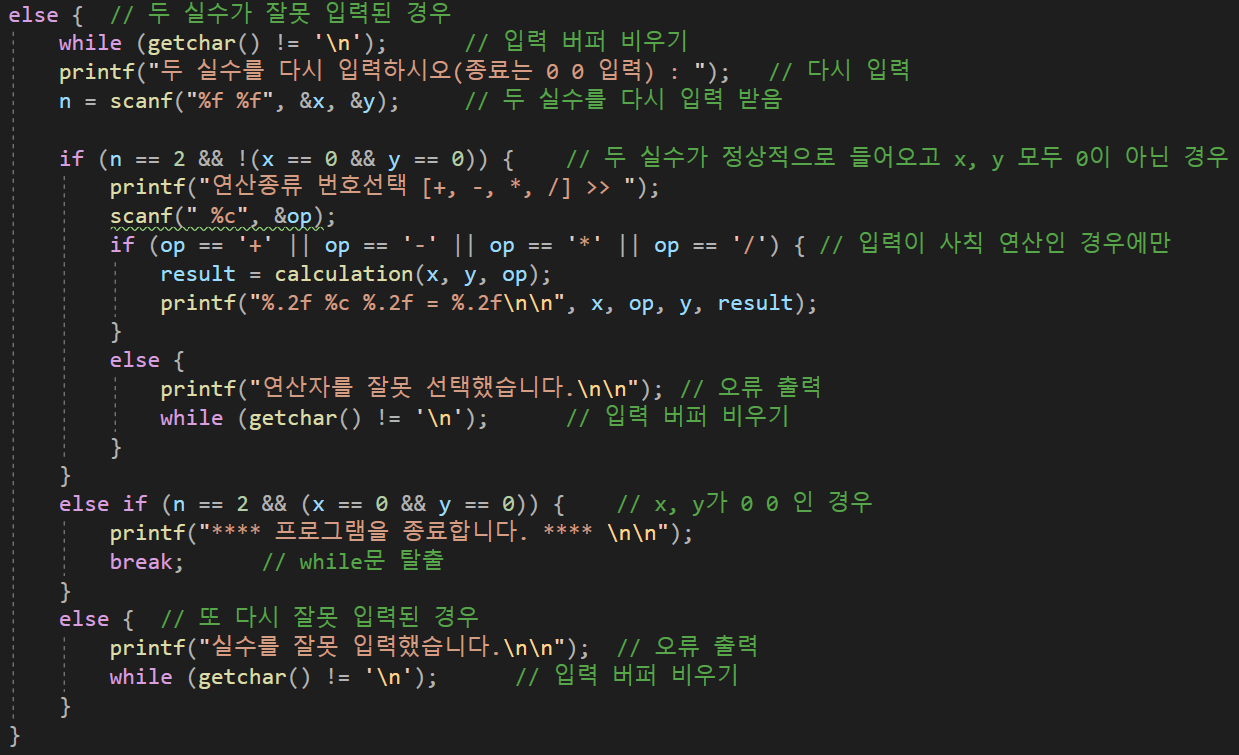


먼저 올바른 두 실수가 들어오고 x 와 y 모두 0이 아닌 경우의 상황이다. 연산 종류를 입력 받아 op에 저장하고, op가 +. -. \*. / 중 하나일 때, calculation(x, y, op)를 통해 입력된 두 수와 op에 해당하는 연산을 수행하여 result에 저장한다. 그리고 결과를 출력하면 if (n==2 && !(x == 0 && y == 0)) 문이 끝나고 다시 메인 while문으로 돌아간다.

만약 연산이 잘못 입력되면 연산자를 잘못 선택했다는 문구와 함께 버퍼를 비우고 if (n==2 && !(x == 0 && y == 0)) 문이 끝나게 된다. 그리고 다시 메인 while문으로 돌아간다.



메인 while문에서 n == 2이지만 x == 0 && y == 0인 상황이다. 이때는 프로그램을 종료하는 상황이므로 break를 통해 메인 while문을 탈출한다.



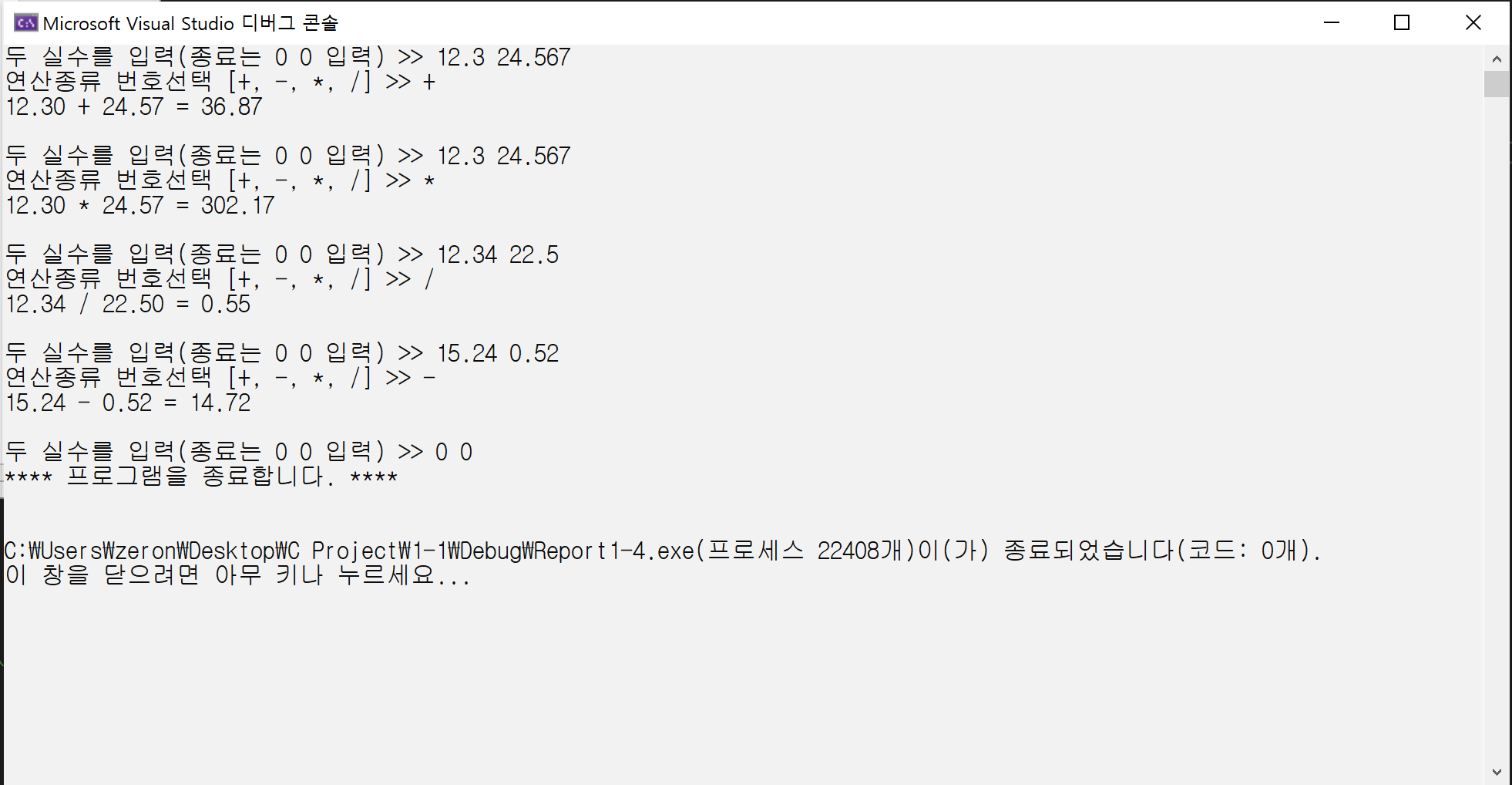
메인 while문에서 두 실수를 입력 받을 때 잘못 입력된 경우이다. 이럴 경우 버퍼를 비우고 다시 입력하라는 명령이 나오고 그 입력을 x와 y에 저장하게 된다. 그 이후부터는 메인 while문에 있는 if – else if (위에서 설명한 if와 else if 부분)과 유사하다.

먼저 두 실수가 올바르게 입력이 되었으면 연산 종류를 선택하고 결과를 출력한다. 이때, 연산이 올바르게 선택되지 않으면 오류를 출력하고 (else // 두 실수가 잘못 입력된 경우) 이 else 문이 끝나게 된다. 그러면 다시 메인 while문으로 가서 처음부터 실수를 입력 받는다.

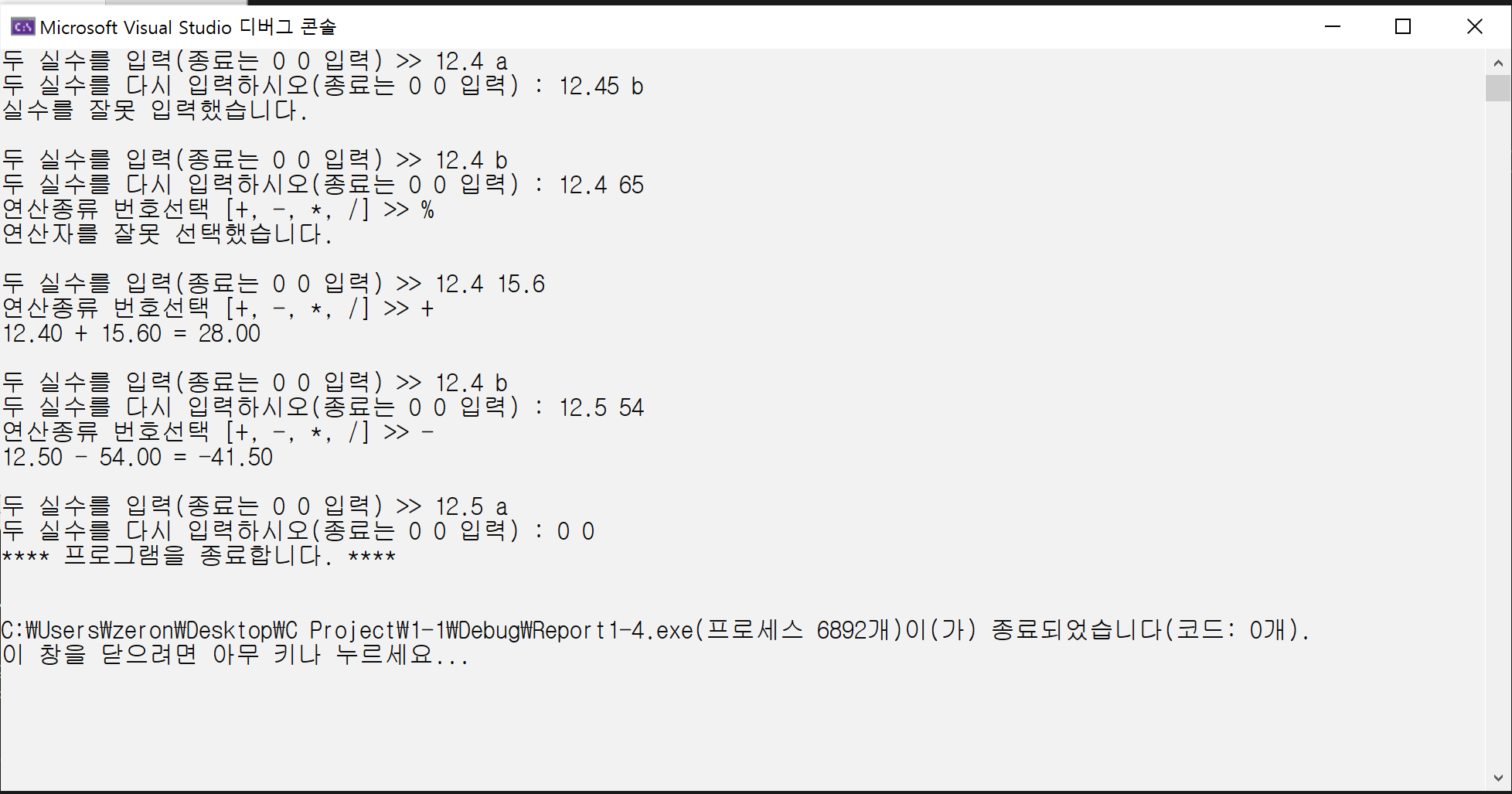
입력은 올바르게 되었지만 x, y 둘 다 0 인 경우 break를 통해 메인 while문을 탈출한다. 이번 구현에서 반복문은 메인 while문만 사용되었기 때문에 break를 한 번만 사용해도 메인 while문을 탈출하여 프로그램을 종료할 수 있다.

만약 이번에도 실수가 잘못 입력 되었으면 잘못 입력되었다는 오류와 함께 메인 while문의 처음으로 돌아가서 다시 입력을 받는다.

이 전체적인 흐름의 결과를 보도록 하자.



정상적인 흐름의 출력이다. 마지막 출력을 보면 0 0을 입력하여 프로그램을 종료한다.



1번 출력은 실수가 올바르게 입력이 되지 않은 경우, 다시 입력을 받는데 이때도 올바르지 않으면 다시 처음으로 돌아가는 모습을 보여준다. 2번 출력은 처음 두 실수가 올바르게 입력이 되지 않았을 때, 이후 올바르게 입력을 받았지만 연산자를 잘못 입력한 경우 연산자 오류를 출력하고 다시 처음으로 돌아가는 모습을 보여준다. 3번은 정상적인 출력이다. 4번은 처음 두 실수의 입력이 잘 되지 않은 경우 다시 입력 받고, 연산자가 올바르게 입력된 경우 결과값을 출력하고 처음으로 돌아가는 출력이다. 마지막 출력은 처음 두 실수가 잘못 입력된 상황에서 다시 입력할 때, 0 0을 입력하면 프로그램이 정상 종료된 것을 보여준다.