기초컴퓨터프로그래밍 10분반 25\_이규하 (2021039057)

9주차 Report

문제 1) 다음 프로그램의 출력결과를 쓰고 왜 그렇게 출력되었는지 설명하시오.

(a)

/\*

실행결과

2

3

2

\*/

#include <stdio.h>

void f(void); // f가 매개변수를 사용하지 않음

int x = 1; // 정수형 전역변수 x를 1로 초기화

int main(void)

{

    int x = 2; // main 함수 정수형 지역변수 x를 2로 초기화

    printf("%d\n", x); // **main 함수 정수형 지역변수 x값 2 출력**

    {

        int x = 3;

// 블록 내 정수형 지역변수 x를 3으로 초기화, 이 블럭 내에서만 적용되는 변수

        printf("%d\n", x); // **블럭 내 정수형 지역변수 x값 3 출력**

    }

    printf("%d\n", x);

// **블럭 바깥에 있으므로 main 함수 정수형 지역변수 x값 2 출력**

    return 0;

}

(b)

/\*

실행결과

0

1

\*/

#include <stdio.h>

void f(void); // main 함수 아래에 함수 f가 있다

int main(void)

{

    f(); // 함수 f 불러옴, count = 0이므로 0 출력

    f(); // 함수 f 불러옴, count = 1이므로 1 출력

    return 0; // main 함수 0으로 리턴

}

void f(void) // 리턴 형식 없는 함수 f, 매개변수도 딱히 없음

{

    static int count = 0; // 정수형 정적변수 count를 0으로 초기화

    // 정적변수는 함수를 벗어나더라도 변수가 사라지지 않고 계속 유지되므로

    // ++ 연산자가 적용되어 값이 계속 증가하게 된다.

    printf("%d\n", count++); // count가 1씩 증가한다.

}

// **f 함수 흐름 설명**

void f(void)

{ //            ↓ 0

    static int count = 0;    // 프로그램이 시작될 때 정적 변수가 생성되고 초기화됨

    printf("%d\n", count++);    // 0 출력후, 0에서 1로 증가

}

//              ↓

void f(void)

{ //            ↓ 1

    static int count = 0;    // 다시 함수가 호출될 때는 값 초기화 무시

    printf("%d\n", count++);    // 1 출력후, 1에서 2로 증가

}

문제 2) 다음 함수를 주석과 같이 호출하는 경우에 화면에 출력되는 내용과 함수의 반환값을 구하라.

(a)

/\*

실행 결과

5

4

3

2

1

0

반환값 : 16

\*/

#include <stdio.h>

int sum(int n)

{

    printf("%d\n", n);

    if (n < 1)

        return 1;

    else

        return(n + sum(n - 1));

}

int main(void)

{

    sum(5);

    return 0;

}

// 실행 흐름

int sum(int n) // n = 5일때

{

    printf("%d\n", n); // 바로 5를 출력해주고

    if (n < 1) // n의 조건에 따라

        return 1; // 이건 무시해주고

    else

        return(n + sum(n - 1));

// **5 + (sum(4) = 11) 로 리턴**, sum(4)를 구해야 할 것임

}

//              ↓

int sum(int n) // n = 4일때

{

    printf("%d\n", n); // 바로 4를 출력해주고

    if (n < 1) // n의 조건에 따라

        return 1; // 이건 무시해주고

    else

        return(n + sum(n - 1));

// 4 + (sum(3) = 7) 로 리턴, sum(3)을 구해야 할 것임

}

//              ↓

int sum(int n) // n = 3일때

{

    printf("%d\n", n); // 바로 3를 출력해주고

    if (n < 1) // n의 조건에 따라

        return 1; // 이건 무시해주고

    else

        return(n + sum(n - 1));

// 3 + (sum(2) = 4) 로 리턴, sum(3)을 구해야 할 것임

}

//              ↓

int sum(int n) // n = 2일때

{

    printf("%d\n", n); // 바로 2를 출력해주고

    if (n < 1) // n의 조건에 따라

        return 1; // 이건 무시해주고

    else

        return(n + sum(n - 1));

// 2 + (sum(1) = 2) 로 리턴, sum(1)을 구해야 할 것임

}

//              ↓

int sum(int n) // n = 1일때

{

    printf("%d\n", n); // 바로 1를 출력해주고

    if (n < 1) // n의 조건에 따라

        return 1; // 이건 무시해주고

    else

        return(n + sum(n - 1));

// 1 + (sum(0) = 1)  로 리턴, sum(0)을 구해야 할 것임

}

//              ↓

int sum(int n) // n = 0일때

{

    printf("%d\n", n); // 바로 0를 출력해주고

    if (n < 1) // n의 조건에 따라

        return 1; // 1로 리턴함

    else

        return(n + sum(n - 1)); //이건 무시해준다

}

// sum(5)의 리턴값 = **5 + (sum(4) = 11) =** 16

(b)

/\*

실행 결과

5

4

3

2

1

0

반환값 : 95

\*/

#include <stdio.h>

int recursive(int n)

{

    printf("%d\n", n);

    if (n < 1)

        return 2;

    else

        return(2 \* recursive(n - 1) + 1);

}

int main(void)

{

    recursive(5);

    return 0;

}

// 실행 흐름

int recursive(int n) // n = 5일때

{

    printf("%d\n", n); // 바로 5를 출력해주고

    if (n < 1) // n의 조건에 따라

        return 2; // 이건 무시해주고

    else

        return(2 \* recursive(n - 1) + 1);

        // 2 \* (recursive(4) = 47) + 1로 리턴, recursive(4)를 구해야 할 것임

}

//              ↓

int recursive(int n) // n = 4일때

{

    printf("%d\n", n); // 바로 4를 출력해주고

    if (n < 1) // n의 조건에 따라

        return 2; // 이건 무시해주고

    else

        return(2 \* recursive(n - 1) + 1);

        // 2 \* (recursive(3) = 23) + 1로 리턴, recursive(3)를 구해야 할 것임

}

//              ↓

int recursive(int n) // n = 3일때

{

    printf("%d\n", n); // 바로 3를 출력해주고

    if (n < 1) // n의 조건에 따라

        return 2; // 이건 무시해주고

    else

        return(2 \* recursive(n - 1) + 1);

        // 2 \* (recursive(2) = 11) + 1로 리턴, recursive(2)를 구해야 할 것임

}

//              ↓

int recursive(int n) // n = 2일때

{

    printf("%d\n", n); // 바로 2를 출력해주고

    if (n < 1) // n의 조건에 따라

        return 2; // 이건 무시해주고

    else

        return(2 \* recursive(n - 1) + 1);

        // 2 \* (recursive(1) = 5) + 1로 리턴, recursive(1)를 구해야 할 것임

}

//              ↓

int recursive(int n) // n = 1일때

{

    printf("%d\n", n); // 바로 1를 출력해주고

    if (n < 1) // n의 조건에 따라

        return 2; // 이건 무시해주고

    else

        return(2 \* recursive(n - 1) + 1);

        // 2 \* (recursive(0) = 2) + 1로 리턴, recursive(0)를 구해야 할 것임

}

//              ↓

int recursive(int n) // n = 0일때

{

    printf("%d\n", n); // 바로 0를 출력해주고

    if (n < 1) // n의 조건에 따라

        return 2; // 2로 리턴함

    else

        return(2 \* recursive(n - 1) + 1);

        // 이건 무시해준다

}

// recursive(5)의 리턴값 =  2 \* (recursive(4) = 47) + 1 = 95

문제3) 다음의 순환적인 프로그램을 반복 구조를 사용한 비순환적 프로그램으로 바꾸어 보자.

before : 순환구조

int recursive(int n)

{

    if(n == 0)

        return 1;

    else

        return (n + recursive(n-1));

}

after : 반복문을 이용한 비순환구조

int recursive(int n)

{

     int i, hap=0;

     for (i=n; i>=1; i--)

        hap += i;

        return hap;

}

문제 4) 덧셈, 뺄셈, 곱셈, 나눗셈을 지원하는 계산기 프로그램을 작성하여 보자. 이번에는 각 연산들이 몇 번씩 계산되었는지를 기억하게 하자. 각 연산을 지원하는 함수들은 자신이 호출된 횟수를 화면에 출력한다.

조건 1 : 정적 지역 변수를 사용하여 프로그램을 작성하라.

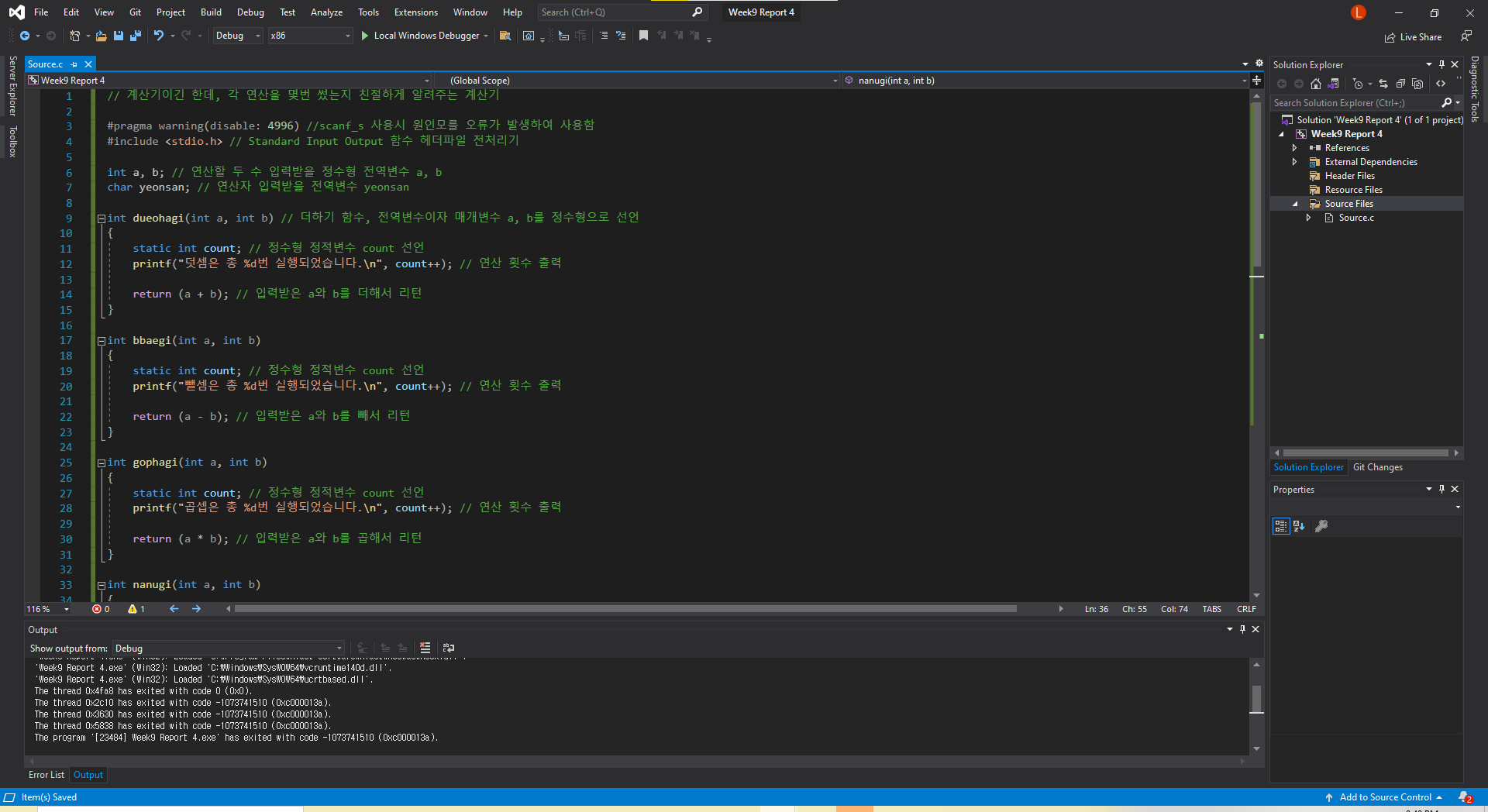
조건 2 : 전역 변수를 사용하여 프로그램을 작성하라.

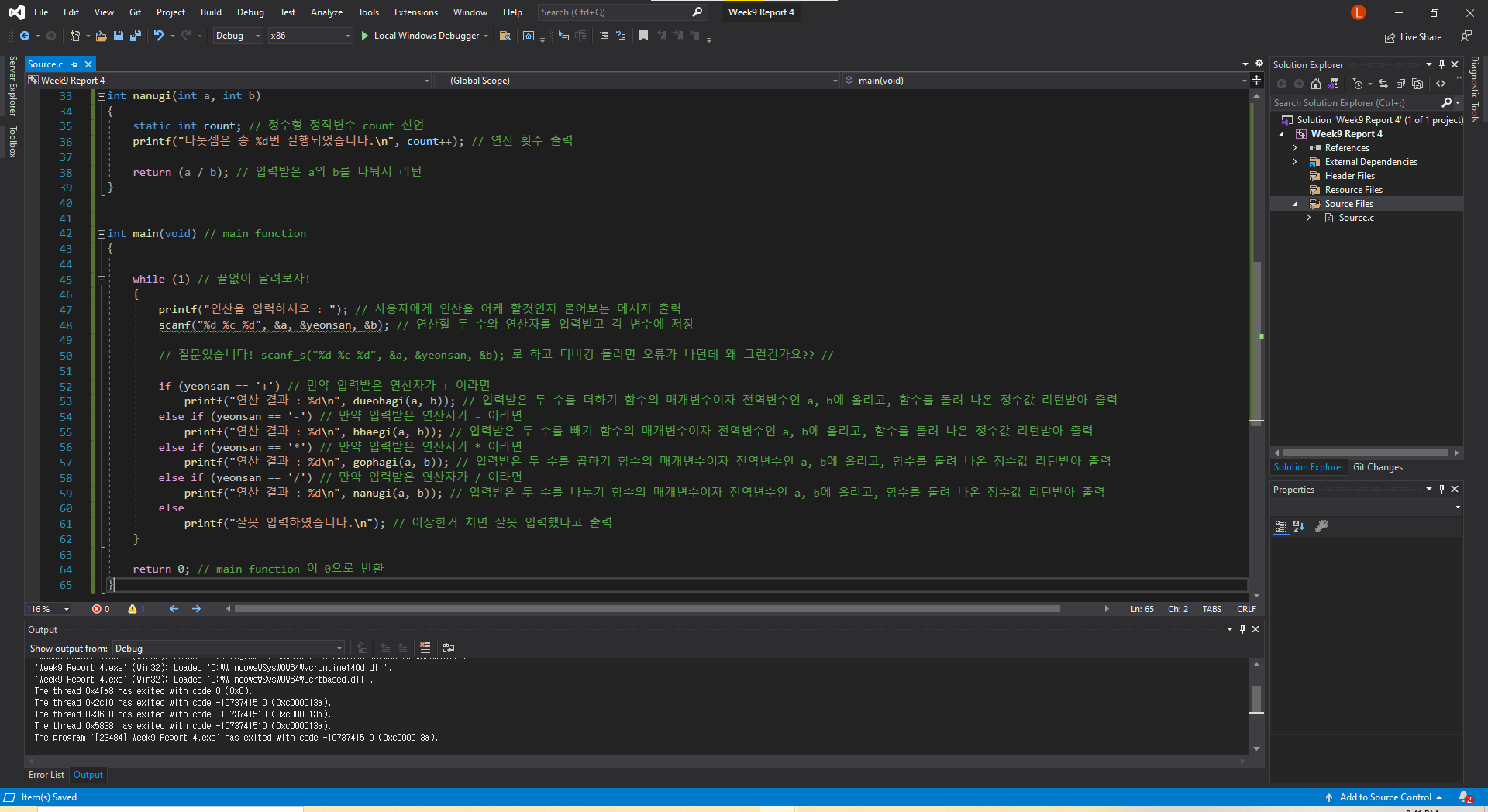
Hint : 정적 지역 변수는 static int count; 와 같이 선언한다.

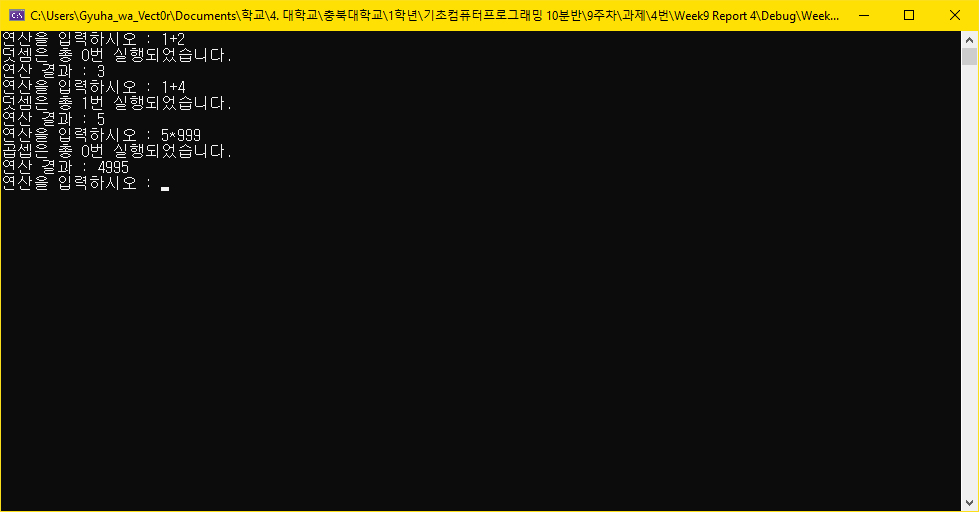
* 문제 분석 & 동작 설명

지난주처럼 사칙연산 계산기를 만들면 되는데, 이번 문제는 각 연산이 몇 번씩 계산되었는지 기록을 남기라 했다. 이는 문제의 조건과 힌트에 제시되어 있는데로 각 연산별 함수 내에 정적 지연변수 static int count; 를 선언하여 횟수를 구현하도록 하자. 지난주처럼 입력받은 연산자를 구별하는 것을 if절로 구별하도록 하고, 각 연산에 해당되는 함수를 만들어 그 함수 내에서 연산을 한 후 리턴해서 출력하도록 하자.

* 프로그램 소스 (주석 포함)

(Line 1 ~ Line 32)

 (Line 33 ~ Line 65)

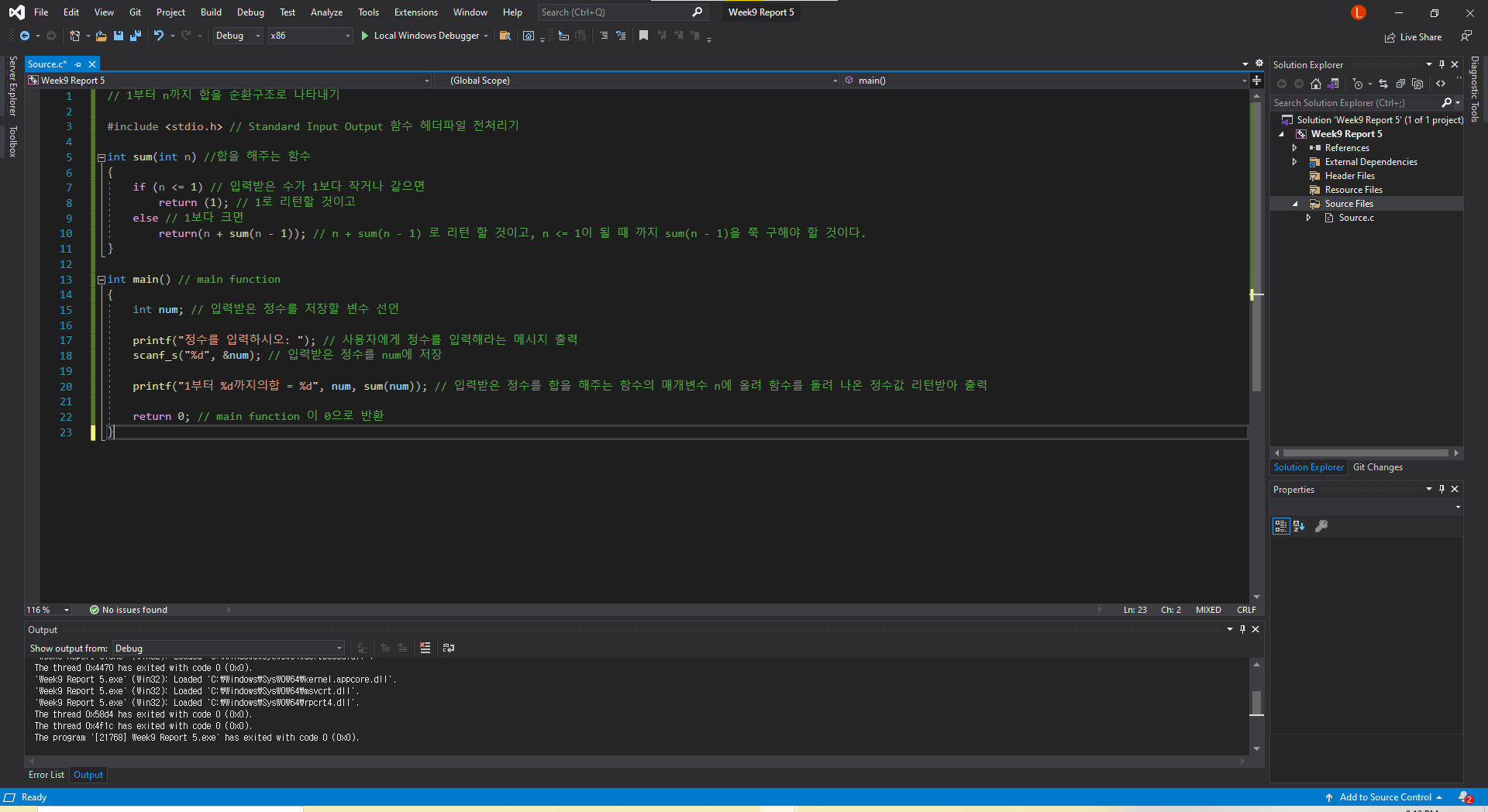


문제 5) 1부터 n까지의 합 (1 + 2 + 3 + ………….. + n) 을 계산하는 문제를 순환기법을 이용하여 작성해보자.

Hint : sum(3)은 3 + sum(2)로 바꿀 수 있다.

* 문제 분석 & 동작 설명

반복문이 아니라 함수의 순환기법을 이용하여 수의 합을 나타내라 하였다. 어디까지 더할지 수를 입력 받아 합을 해주는 함수로 들어가는데, 그 함수는 입력 받은 수의 조건이 1보다 작아질 때 까지 그 함수에 들어가는 수가 하나씩 작아지면서 계속 함수를 돌리는 방식으로 리턴 된 값들을 다 더하면 될 것이다. 그러고나서 그 것을 출력하면 될 것이다.

* 프로그램 소스 (주석 포함)

