

9 주차 Report

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

(a)

```
/*
실행결과

2
3
2
*/

#include <stdio.h>
void f(void); // f 가 매개변수를 사용하지 않음
int x = 1; // 정수형 전역변수 x 를 1 로 초기화

int main(void)
{
    int x = 2; // main 함수 정수형 지역변수 x 를 2 로 초기화
    printf("%d\n", x); // main 함수 정수형 지역변수 x 값 2 출력
    {
        int x = 3;
        // 블록 내 정수형 지역변수 x 를 3 으로 초기화, 이 블록 내에서만 적용되는 변수
        printf("%d\n", x); // 블록 내 정수형 지역변수 x 값 3 출력
    }
    printf("%d\n", x);
    // 블록 바깥에 있으므로 main 함수 정수형 지역변수 x 값 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 = 0 이므로 0 출력
    f(); // 함수 f 불러옴, count = 1 이므로 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 count 를 0으로 초기화
    // 정적변수는 함수를 벗어나더라도 변수가 사라지지 않고 계속 유지되므로
    // ++ 연산자가 적용되어 값이 계속 증가하게 된다.
    printf("%d\n", count++); // count 가 1 씩 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 = 5 일때
{
    printf("%d\n", n); // 바로 5 를 출력해주고

    if (n < 1) // n 의 조건에 따라
        return 1; // 이걸 무시해주고
```

```

    else
        return(n + sum(n - 1));
        // 5 + (sum(4) = 11) 로 리턴, sum(4)를 구해야 할 것임
}
//          ↓
int sum(int n) // n = 4 일때
{
    printf("%d\n", n); // 바로 4 를 출력해주고

    if (n < 1) // n 의 조건에 따라
        return 1; // 이걸 무시해주고
    else
        return(n + sum(n - 1));
        // 4 + (sum(3) = 7) 로 리턴, sum(3)을 구해야 할 것임
}
//          ↓
int sum(int n) // n = 3 일때
{
    printf("%d\n", n); // 바로 3 를 출력해주고

    if (n < 1) // n 의 조건에 따라
        return 1; // 이걸 무시해주고
    else
        return(n + sum(n - 1));
        // 3 + (sum(2) = 4) 로 리턴, sum(3)을 구해야 할 것임
}
//          ↓
int sum(int n) // n = 2 일때
{
    printf("%d\n", n); // 바로 2 를 출력해주고

    if (n < 1) // n 의 조건에 따라
        return 1; // 이걸 무시해주고
    else
        return(n + sum(n - 1));
        // 2 + (sum(1) = 2) 로 리턴, sum(1)을 구해야 할 것임
}
//          ↓
int sum(int n) // n = 1 일때

```

```

{
    printf("%d\n", n); // 바로 1 를 출력해주고

    if (n < 1) // n 의 조건에 따라
        return 1; // 이걸 무시해주고
    else
        return(n + sum(n - 1));
        // 1 + (sum(0) = 1) 로 리턴, sum(0)을 구해야 할 것임
}
//          ↓
int sum(int n) // n = 0 일때
{
    printf("%d\n", n); // 바로 0 를 출력해주고

    if (n < 1) // n 의 조건에 따라
        return 1; // 1 로 리턴함
    else
        return(n + sum(n - 1)); //이걸 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 = 5 일때
{
    printf("%d\n", n); // 바로 5 를 출력해주고

    if (n < 1) // n 의 조건에 따라
        return 2; // 이걸 무시해주고
    else
        return(2 * recursive(n - 1) + 1);
        // 2 * (recursive(4) = 47) + 1 로 리턴, recursive(4)를 구해야 할 것임
}
//          ↓
int recursive(int n) // n = 4 일때
{
    printf("%d\n", n); // 바로 4 를 출력해주고

    if (n < 1) // n 의 조건에 따라
        return 2; // 이걸 무시해주고
    else
        return(2 * recursive(n - 1) + 1);
        // 2 * (recursive(3) = 23) + 1 로 리턴, recursive(3)를 구해야 할 것임
}
//          ↓
int recursive(int n) // n = 3 일때
{
    printf("%d\n", n); // 바로 3 를 출력해주고

```

```

    if (n < 1) // n 의 조건에 따라
        return 2; // 이걸 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 = 2 일때
{
    printf("%d\n", n); // 바로 2를 출력해주고

    if (n < 1) // n 의 조건에 따라
        return 2; // 이걸 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 = 1 일때
{
    printf("%d\n", n); // 바로 1를 출력해주고

    if (n < 1) // n 의 조건에 따라
        return 2; // 이걸 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 = 0 일때
{
    printf("%d\n", n); // 바로 0를 출력해주고

    if (n < 1) // n 의 조건에 따라
        return 2; // 2로 리턴함
    else
        return(2 * recursive(n - 1) + 1);
        // 이걸 무시해준다
}

```

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

문제 3) 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.

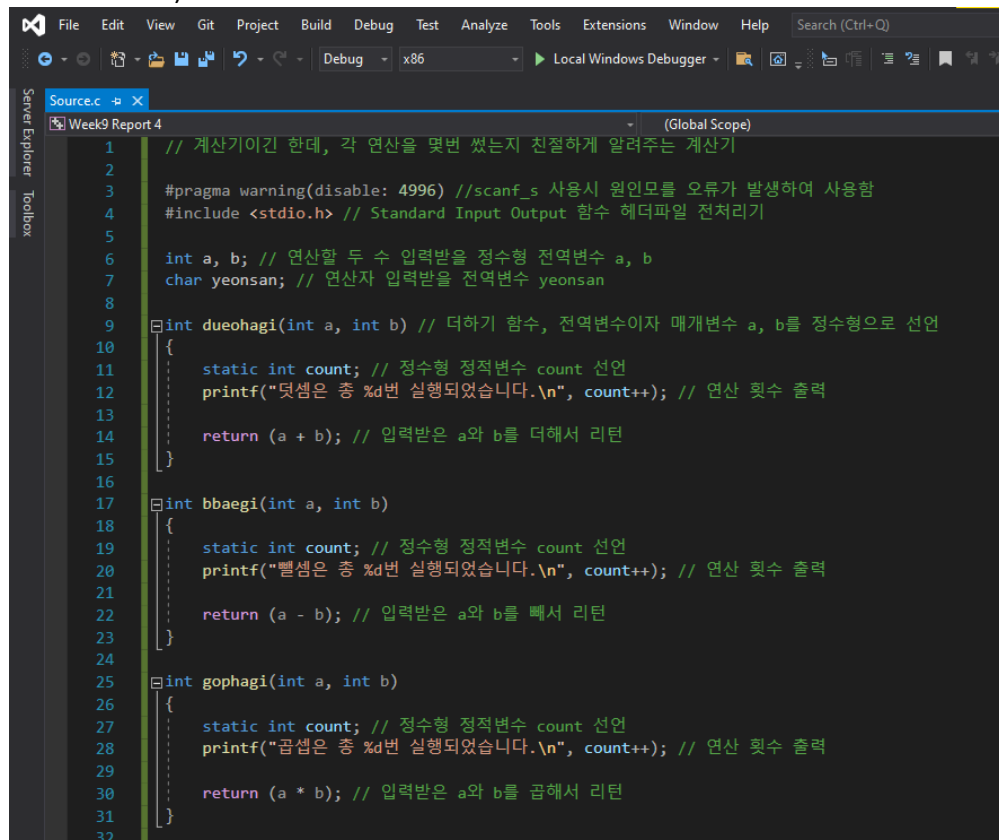
Hint : 정적 지역 변수는 `static int count;` 와 같이 선언한다.

- 문제 분석 & 동작 설명

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

- 프로그램 소스 (주석 포함)

(Line 1 ~ Line 32)



```
1 // 계산기이긴 한데, 각 연산을 몇번 썼는지 친절하게 알려주는 계산기
2
3 #pragma warning(disable: 4996) //scanf_s 사용시 원인모를 오류가 발생하여 사용함
4 #include <stdio.h> // Standard Input Output 함수 헤더파일 전처리기
5
6 int a, b; // 연산할 두 수 입력받을 정수형 전역변수 a, b
7 char yeonsan; // 연산자 입력받을 전역변수 yeonsan
8
9 int dueohagi(int a, int b) // 더하기 함수, 전역변수이자 매개변수 a, b를 정수형으로 선언
10 {
11     static int count; // 정수형 정적변수 count 선언
12     printf("덧셈은 총 %d번 실행되었습니다.\n", count++); // 연산 횟수 출력
13
14     return (a + b); // 입력받은 a와 b를 더해서 리턴
15 }
16
17 int bbaegi(int a, int b)
18 {
19     static int count; // 정수형 정적변수 count 선언
20     printf("뺄셈은 총 %d번 실행되었습니다.\n", count++); // 연산 횟수 출력
21
22     return (a - b); // 입력받은 a와 b를 빼서 리턴
23 }
24
25 int gophagi(int a, int b)
26 {
27     static int count; // 정수형 정적변수 count 선언
28     printf("곱셈은 총 %d번 실행되었습니다.\n", count++); // 연산 횟수 출력
29
30     return (a * b); // 입력받은 a와 b를 곱해서 리턴
31 }
32
```

(Line 33 ~ Line 65)

```
File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) Week9 Report 4
Debug x86 Local Windows Debugger
Source.c
Week9 Report 4 (Global Scope) main(void)
33 int nanugi(int a, int b)
34 {
35     static int count; // 정수형 정적변수 count 선언
36     printf("나눗셈은 총 %d번 실행되었습니다.\n", count++); // 연산 횟수 출력
37
38     return (a / b); // 입력받은 a와 b를 나눠서 리턴
39 }
40
41
42 int main(void) // main function
43 {
44
45     while (1) // 끝없이 돌려보자!
46     {
47         printf("연산을 입력하시오 : "); // 사용자에게 연산을 여케 할것인지 물어보는 메시지 출력
48         scanf("%d %c %d", &a, &yeonsan, &b); // 연산할 두 수와 연산자를 입력받고 각 변수에 저장
49
50         // 질문있습니다! scanf_s("%d %c %d", &a, &yeonsan, &b); 로 하고 디버깅 돌리면 오류가 나는데 왜 그런건가요?? //
51
52         if (yeonsan == '+') // 만약 입력받은 연산자가 + 이라면
53             printf("연산 결과 : %d\n", dueohagi(a, b)); // 입력받은 두 수를 더하기 함수의 매개변수이자 전역변수인 a, b에 올리고, 함수를 돌려 나온 정수값 리턴받아 출력
54         else if (yeonsan == '-') // 만약 입력받은 연산자가 - 이라면
55             printf("연산 결과 : %d\n", bbaegi(a, b)); // 입력받은 두 수를 빼기 함수의 매개변수이자 전역변수인 a, b에 올리고, 함수를 돌려 나온 정수값 리턴받아 출력
56         else if (yeonsan == '*') // 만약 입력받은 연산자가 * 이라면
57             printf("연산 결과 : %d\n", gophagi(a, b)); // 입력받은 두 수를 곱하기 함수의 매개변수이자 전역변수인 a, b에 올리고, 함수를 돌려 나온 정수값 리턴받아 출력
58         else if (yeonsan == '/') // 만약 입력받은 연산자가 / 이라면
59             printf("연산 결과 : %d\n", nanugi(a, b)); // 입력받은 두 수를 나누기 함수의 매개변수이자 전역변수인 a, b에 올리고, 함수를 돌려 나온 정수값 리턴받아 출력
60         else
61             printf("잘못 입력하였습니다.\n"); // 이상한거 치면 잘못 입력했다고 출력
62     }
63
64     return 0; // main function 이 0으로 반환
65 }
```

```
C:\Users\Gyuha_wa_Vect0r\Documents\학교\4.
연산을 입력하시오 : 1+2
나눗셈은 총 0번 실행되었습니다.
연산 결과 : 3
연산을 입력하시오 : 1+4
나눗셈은 총 1번 실행되었습니다.
연산 결과 : 5
연산을 입력하시오 : 5*999
나눗셈은 총 0번 실행되었습니다.
연산 결과 : 4995
연산을 입력하시오 : _
```

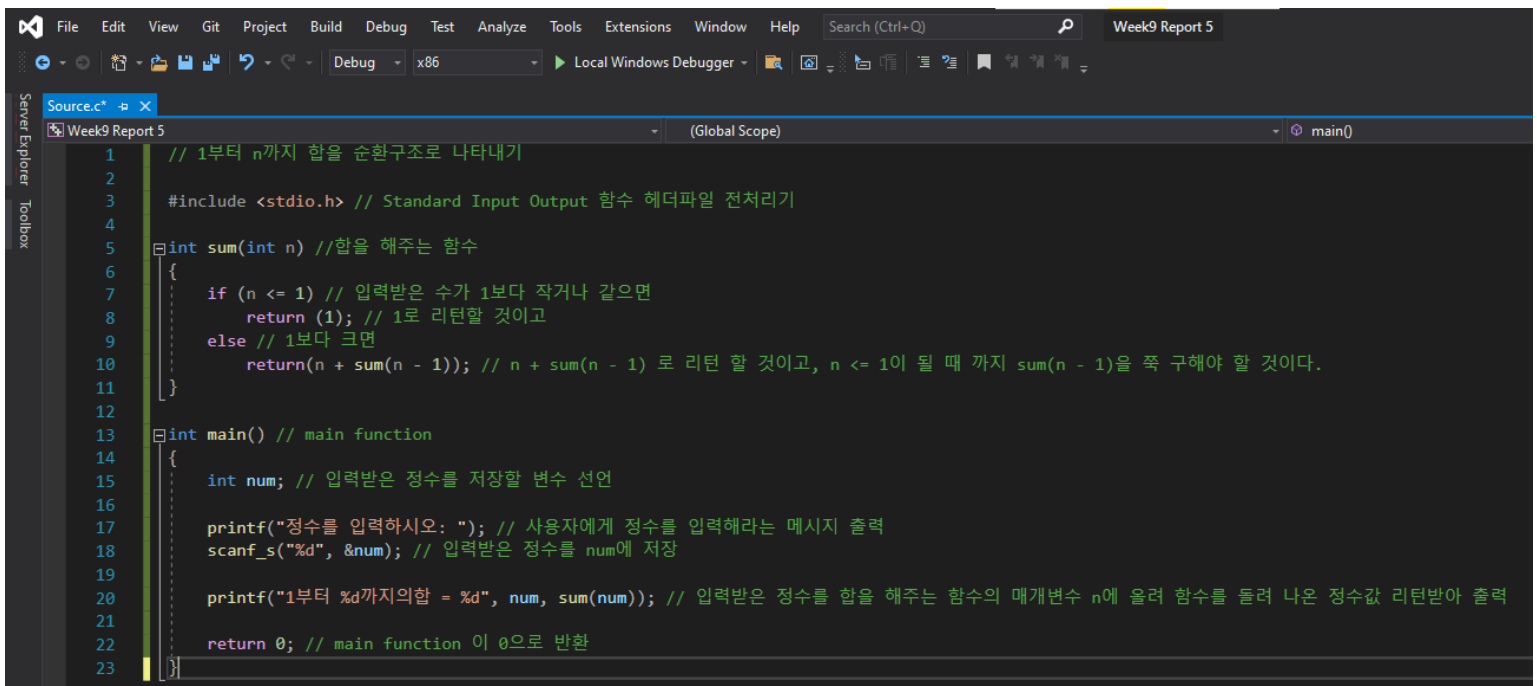
문제 5) 1 부터 n 까지의 합 ($1 + 2 + 3 + \dots + n$) 을 계산하는 문제를 순환기법을 이용하여 작성해보자.

Hint : $\text{sum}(3)$ 은 $3 + \text{sum}(2)$ 로 바꿀 수 있다.

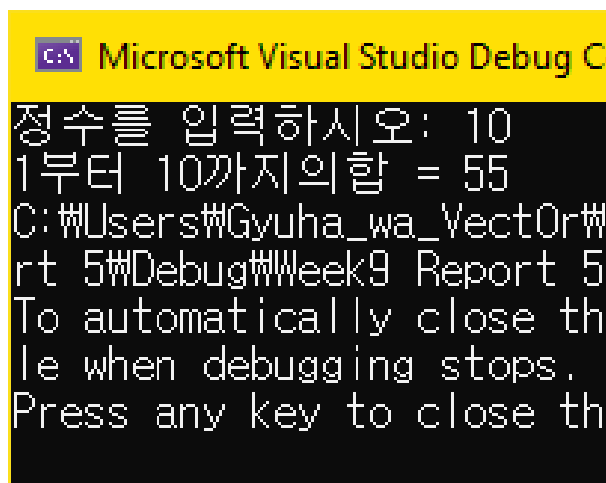
- 문제 분석 & 동작 설명

반복문이 아니라 함수의 순환기법을 이용하여 수의 합을 나타내라 하였다. 어디까지 더할지 수를 입력 받아 합을 해주는 함수로 들어가는데, 그 함수는 입력 받은 수의 조건이 1 보다 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)



```
1 // 1부터 n까지 합을 순환구조로 나타내기
2
3 #include <stdio.h> // Standard Input Output 함수 헤더파일 전처리기
4
5 int sum(int n) //합을 해주는 함수
6 {
7     if (n <= 1) // 입력받은 수가 1보다 작거나 같으면
8         return (1); // 1로 리턴할 것이고
9     else // 1보다 크면
10         return(n + sum(n - 1)); // n + sum(n - 1) 로 리턴 할 것이고, n <= 1이 될 때 까지 sum(n - 1)을 꼭 구해야 할 것이다.
11 }
12
13 int main() // main function
14 {
15     int num; // 입력받은 정수를 저장할 변수 선언
16
17     printf("정수를 입력하시오: "); // 사용자에게 정수를 입력해라는 메시지 출력
18     scanf_s("%d", &num); // 입력받은 정수를 num에 저장
19
20     printf("1부터 %d까지의합 = %d", num, sum(num)); // 입력받은 정수를 합을 해주는 함수의 매개변수 n에 올려 함수를 돌려 나온 정수값 리턴받아 출력
21
22     return 0; // main function 이 0으로 반환
23 }
```



```
Microsoft Visual Studio Debug C
정수를 입력하시오: 10
1부터 10까지의합 = 55
C:\Users\Gyuha_wa_VectOr\rt 5\Debug\Week9 Report 5
To automatically close the window when debugging stops,
Press any key to close the window.
```