윈도우프로그래밍
Lab 01

- Visual Studio 2015에서의 디버깅 -

김한슬

uo3359@sookmyung.ac.kr

소개

- TA: 김한슬

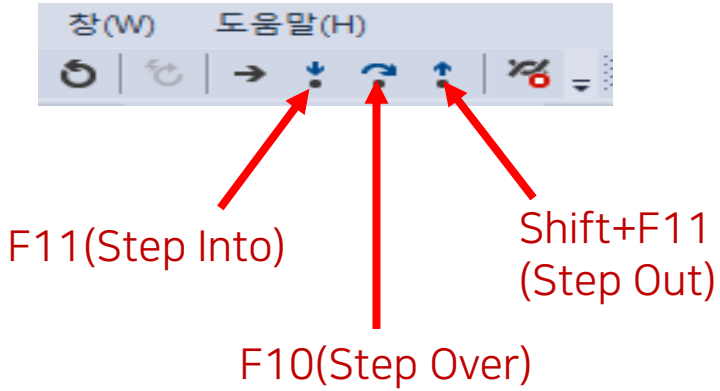
- E-mail: uo3359@sookmyung.ac.kr
  - 실습 시간 외에 질문이 있으시면 메일 보내주세요

- Lab: 명신관 405A 데이터인공지능 연구실
  - 직접 만나야 할 경우 미리 메일로 연락주세요

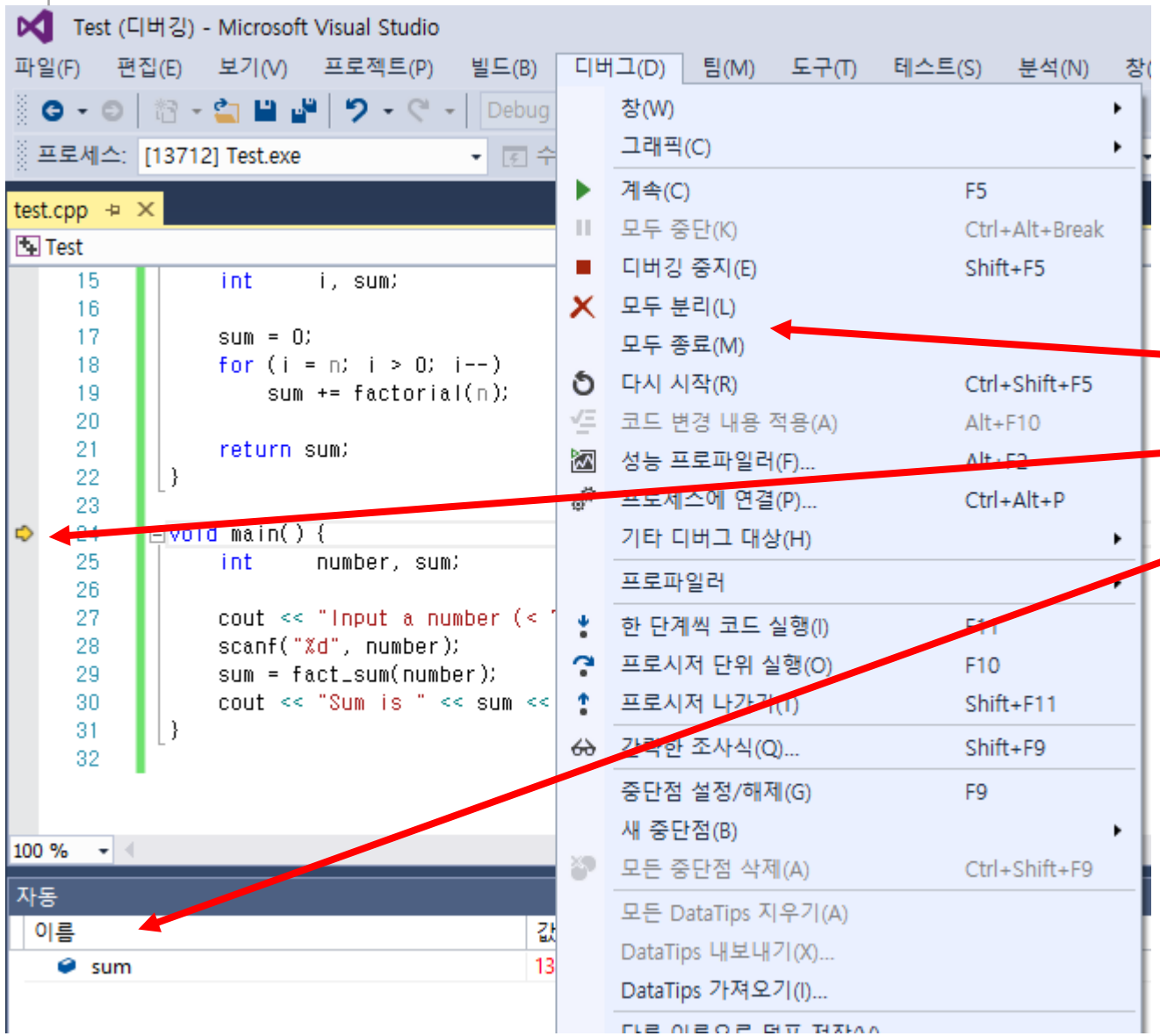
Visual Studio에서 디버깅 모드 실행

- 한 행씩 실행시키기
  - F11 (Step Into; 한 단계씩 코드 실행): 한 행씩 실행하되, 함수 호출을 만나면 해당 함수로 이동한다. (scanf() 함수, cout 표현 등의 라이브러리 함수 안으로 이동하지 않도록 주의한다.)
  - F10 (Step Over; 프로시저 단위 실행): 한 행씩 실행하되, 함수 호출을 만나면 해당 함수 전체를 한 스텝으로 간주하여 실행시킨다.
  - Shift+F11 (Step Out; 프로시저 나가기): 현재 실행되고 있는 함수의 끝까지 실행해버린다.

- 정해진 위치까지 실행하기
  - Ctrl+F10 (Run to Cursor): 현재 커서가 위치한 곳에 와서 실행을 멈춘다.
  - F9에 의한 중단점(breakpoint) 설정 + F5에 의한 계속



Visual Studio에서 디버깅 모드 실행

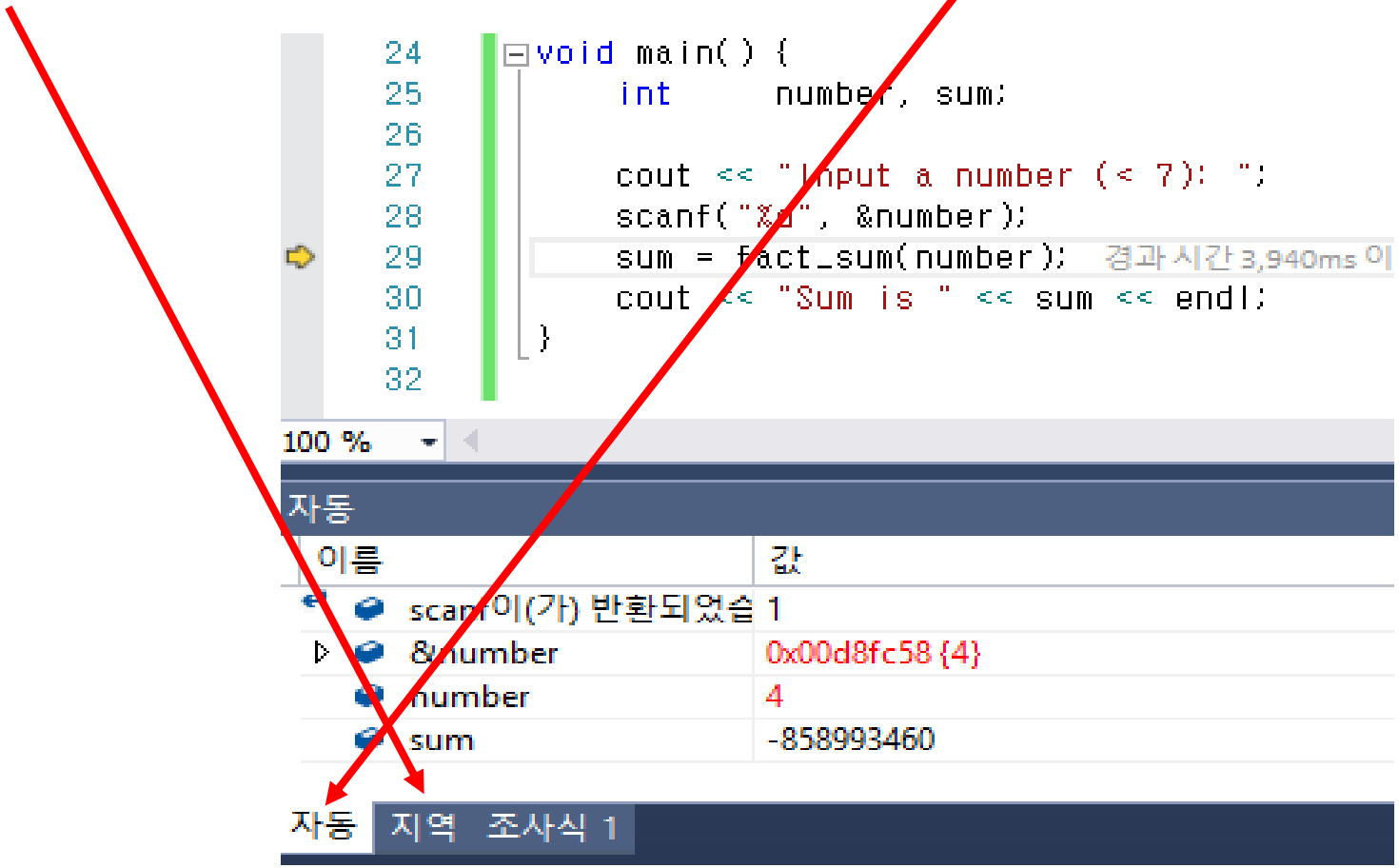


F10(Step Over)을 누른 후의 모습

- Debug 메뉴가 변경되어 있다
- 현재 실행 위치를 가리키는 화살표
- 변수 창 (Variables Window)

변수 값의 관찰 방법

- 변수 위에 마우스 커서 올려 놓기: 해당 변수 값이 툴팁으로 나타난다.
- 변수 창: 현재 실행되는 행 및 직전 행에 의해 영향 받는 변수들([자동] 탭) 또는 현재 함수의 지역 변수들([지역] 탭)의 값을 보여준다.



실습 문제 #1

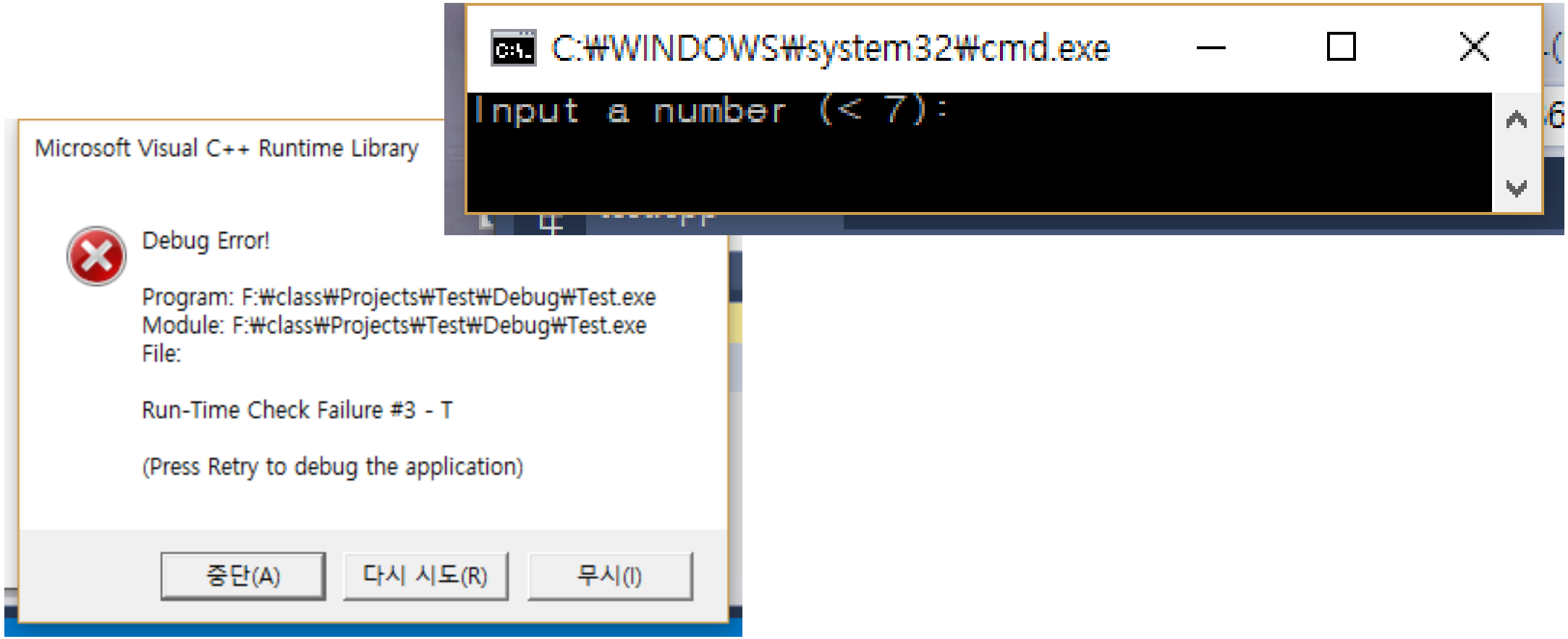
Visual Studio 2015의 디버깅 기능을 사용하여 오류 수정하기

```
/* 이 프로그램은 정수 n을 매개변수로 받아
    (1! + 2! + ... + n!)를 계산하여 출력하도록  되어 있다.

    factorial(n)은 n!을 리턴한다.

    fact_sum(n)은 1! + 2! + ... + n!을 리턴한다. */

#include  <iostream>
using namespace std;
#include  <stdio.h>

int factorial(int n) {
   int      i, prod;

   for (i = 2; i <= n; i++)
                prod *= i;
   return prod;
}
```

```
int fact_sum (int n) {
       int     i, sum;

       sum = 0;
       for (i = n; i > 0; i--)
              sum += factorial(n);

       return sum;
}

void main() {
       int     number, sum;

       cout << "Input a number (< 7): ";
       scanf("%d", number);
       sum = fact_sum(number);
       cout << "Sum is " << sum << endl;
}
```

실습 문제 #1 프로그램 디버깅

- 예제 프로그램을 입력한 후 [디버깅하지 않고 시작] 메뉴항목을 사용하여 실행시키면, 옆의 화면과 같은 실행화면 상태에서 아래와 같은 오류 메시지 창이 나타난다.



실습 문제 #1 프로그램 디버깅

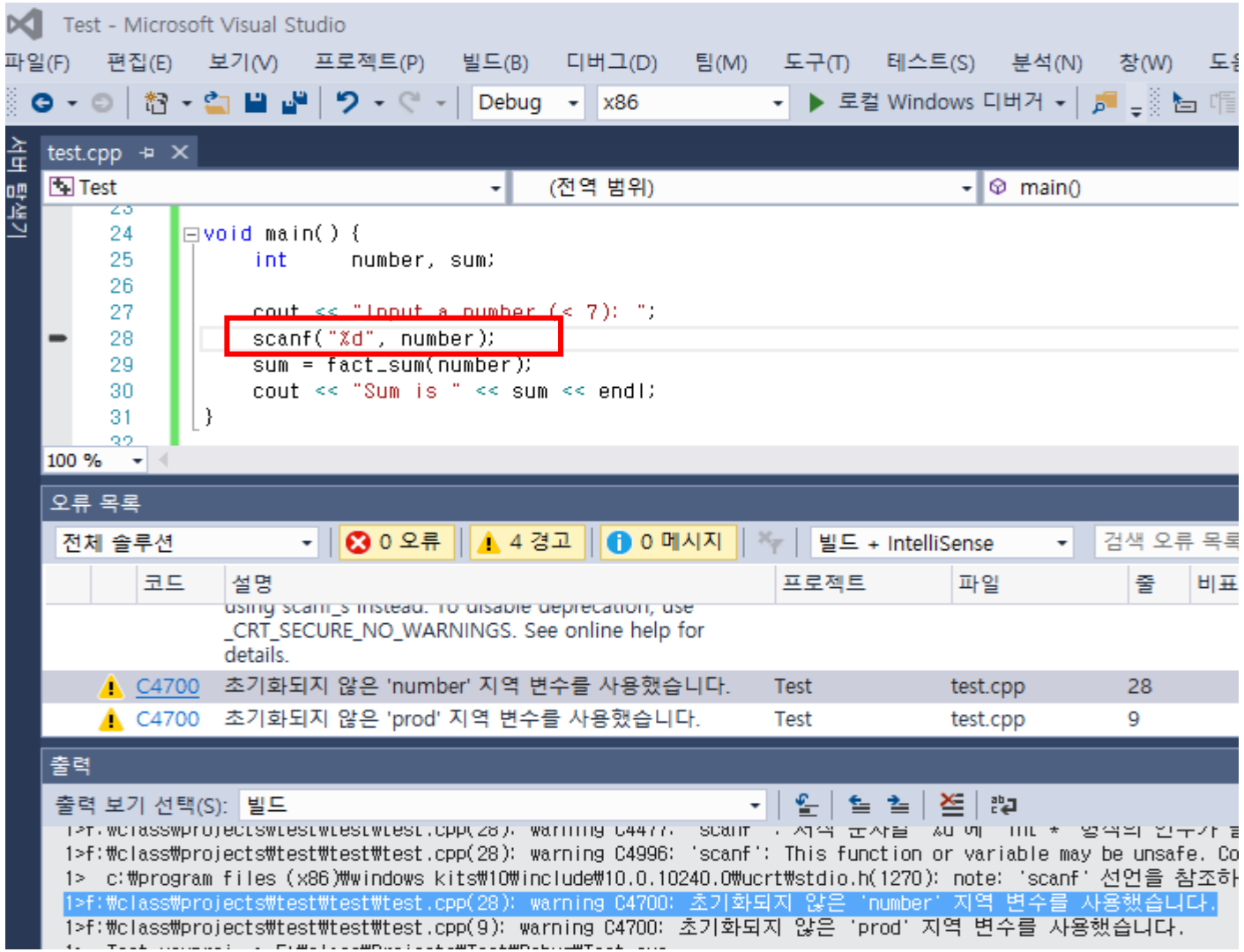
```
scanf("%d", number);
```

- 앞의 오류가 발생하기 전 마지막으로 실행된 문장은 scanf() 함수 호출이다.
  - 물론 오류의 실제 원인이 꼭 마지막으로 실행된 문장에 있지 않을 수도 있다.
  - 그러나 이 예제의 경우 scanf() 함수 호출에 문제가 있다.
  - 이 문장을 자세히 들여다 봄으로써 문제를 찾을 수 있을 것이다.
  - 여기에서 일단 실행을 종료시킨다. 그런 다음 [빌드(Build)] 메뉴의 [솔루션 다시 빌드] 기능을 사용하여 다시 컴파일 해본다.

    - 컴파일 과정에서 warning 오류 메시지들을 볼 수 있을 텐데, error 오류를 만나면 컴파일러가 실행 코드를 생성하지 않는 반면, warning 오류는 무시하고 실행 코드가 생성된다. 그러나 때로 warning 오류가 문제가 될 수도 있다.

    - 오류 메시지를 클릭하면 프로그램 소스 상의 위치를 알려준다.

실습 문제 #1 프로그램 디버깅



실습 문제 #1 프로그램 디버깅

- 오류 메시지의 의미

warning C4700: 초기화되지 않은 'number' 지역 변수를 사용했습니다.

- scanf() 함수에 전달될 인수는 원래 number의 주소라야 하며, &number로 표시되어야 한다.

```
scanf("%d", number);   →   scanf("%d", &number);
```

- 주어진 예제 코드에서는 number라고 표시되어 있으므로 number의 주소 대신 number가 현재 가지고 있는 값이 전달될 것이다.
- number의 값이 정해진 적이 없으므로 Visual Studio에서는 실행 과정에서 이에 대한 오류 메시지를 보여준다.
- 만일 number = 0 등에 의해 미리 이 변수가 초기화되거나 배정문에 의해 값이 정해진다면, 컴파일러에 의한 warning 오류는 발생하지 않겠지만, 여전히 scanf() 함수 호출은 잘못된 것이다. 이로 인해 다른 종류의 오류가 발생할 수도 있고 그냥 지나갈 수도 있다. 프로그램 상의 오류는 실행할 때마다 다른 행동을 보일 수 있음을 유의해야 한다.

실습 문제 #1 프로그램 디버깅

- 오류 메시지의 의미

warning C4700: 초기화되지 않은 'prod' 지역 변수를 사용했습니다.

  - 이는 prod가 적절히 초기화되지 않았기 때문에 발생한 일이다.
    for문 앞에 다음 문장을 끼워 넣음으로써 문제가 해결된다.

```
prod = 1;
```

```
prod *= i;
```

- 또 다른 오류

warning C4996: 'scanf': This function or variable may be unsafe. Consider using scanf_s instead.

  - C++에서는 scanf() 함수 대신 다른 입력 방식의 사용을 권고한다. 이 프로그램에서 이 오류는 중요하지 않다.
    따라서 무시한다.

실습 문제 #1 프로그램 디버깅

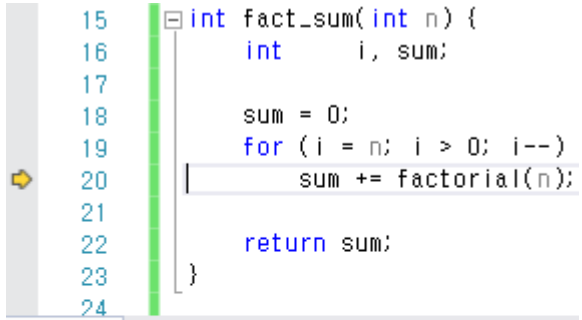
- 오류 수정이 후 다시 실행
  - 입력 값이 4일 경우 최종 결과는 1! + 2! +3! + 4!, 즉, 33이 되어야 하는데, 아주 다른 결과가 출력되는 것을 볼 수 있을 것이다.

  - 이제 F11을 사용하여 함수 fact_sum()으로 들어가보자.
  - fact_sum() 안에서는 다시 F10을 사용한다.

  - 지정된 위치에 도달했을 때 변수들의 값을 관찰해보자. factorial() 함수의 실행 후 변수 sum의 값이 부적절하게 변한 것을 관찰할 수 있을 것이다.

  - fact_sum() 함수 안에서는 factorial() 함수를 호출하여 n!, (n-1)!, (n-2)!, …, 1! 등을 계산하고 이들에 대한 합을 계산하여 리턴해야 하는데, sum의 변화하는 모습은 이러한 계산을 제대로 수행하지 못하고 있음을 알 수 있다. 무엇이 문제인지 발견하고 수정하라.

  - 참고: 여기에서는 지정된 위치에 여러 번 도달하는 일이 필요한데, 이를 달성하는 방법에는 여러 가지가 있다.
    - F10과 F11의 적절한 조합
    - F9에 의한 중단점 설정과 F5
    - Ctrl+F10



자동

이름	값
i	4
n	4
sum	0

자동

이름	값
i	3
n	4
sum	24

자동

이름	값
i	2
n	4
sum	48

실습 문제 #2

Visual Studio 2015의 디버깅 기능을 사용하여 오류 수정하기

```
#include <iostream>
using namespace std;

struct BigInt {
        int digits[100];
        int ndigits;
};

// Converts digit strings into BigInt structures.
// e.g.) "123" ==> (digits = [3, 2, 1], ndigits = 3)

void convert(BigInt *p, char *s) {
        int n = 0;
        while (*s++ != '₩0') // Find the end of
            n++;                    //the string s

        p->ndigits = n;
        while (n >= 0)
                p->digits[n--] = *s--;
}
```

```
// Prints BigInt structures.
// e.g.) For (digits = [3, 2, 1], ndigits = 3),
//          it displays "123"

void print(BigInt x) {
        int n = x.ndigits;

        while (--n >= 0)
                cout << x.digits[n];
}

// Prints "a = 123456789123456789"
void main() {
        BigInt a;

        convert(&a, "123456789123456789");
        cout << "a = ";
        print(a);
        cout << endl;
}
```

실습 문제 #2

Visual Studio 2015의 디버깅 기능을 사용하여 오류 수정하기

- 디버깅 전략
  - 오류를 포함하는 함수 찾기
    convert()인가? print()인가? 또는 둘 모두 인가?

    convert() 실행 후 a 값 검사

실습 문제 #3

Visual Studio 2015의 디버깅 기능을 사용하여 오류 수정하기

```
#include <iostream>
using namespace std;
#include <stdlib.h>
#include <time.h>

// 배열 s[]에 1에서 9까지의 숫자를 랜덤하게 배열한 문자열을 넣기 위해 s[i]에 랜덤한
// 숫자를 넣은 후, s[] 상의 다른 칸에 있는 숫자와 겹칠 경우, i 값을 후퇴시킨 다음
// 새로 s[i]를 정하는 방법을 사용한다.
void main() {
        char s[10];

        s[9] = '₩0';
        srand((unsigned) time(NULL));
        for (int i = 0; i < 9; i++) {
                s[i] = rand() % 9 + '1';
                for (int j = 0; j < 9; j++)
                        if (s[i] == s[j])
                                i--;
        }
        cout << "s = " << s << endl;
}
```

과제 제출: 실습 문제 #2, 3 보고서

- 내용
  - 디버깅 과정 설명(오류 원인 분석)
  - 수정한 결과 소스
  - 실행 화면 캡처

- 제출 형식
  - 스노우보드에서 Lab01_이름_학번.doc 다운로드
  - 스노우보드에 제출: Lab01_이름_학번.doc

- 제출 기한
  - 3월 16일(화) PM 11:59 까지