

민 코 딩 수 업 노 트

---

# 수업노트 디버깅 중급 1



# 배우는 내용

## 디버깅 중급

1. 커서 위치에서 부터 실행 (Ctrl + F10)
2. 외워야 하는 단축키 정리

# 빠른 디버깅을 위한 기본 안내 사항

개발자들은 코딩을 하는 시간만큼 버그를 찾는 시간을 많이 소모한다.  
실력 있는 개발자가 되기 위해서는, 빠르게 디버깅 할 줄 알아야 한다.

버그가 발생하면, 소스코드를 노려보면서 버그를 찾지 말고,  
Trace를 통해 버그를 찾는 연습을 꾸준히 해야 한다.

조사식에 내가 확인해야 할 변수들을 등록해두고,  
정확히 어느 Line부터 잘못된 값이 들어가는지 찾아내야 한다.



추리력을 발휘해야 한다.

# Trace 단축키 Ctrl + F10

Trace로 특정 라인까지 실행 결과를 확인하고 싶을 때 쓰는 방법

1. 커서 위치까지 실행 (Ctrl + F10)
2. Breaking Pointer


이 곳까지 실행하고 싶다면  
커서 클릭 후 Ctrl + F10 누르면 된다.

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

int main()
{
    int x;
    int sum = 0;

    for (x = 0; x < 100; x++) {
        sum += x;
    }

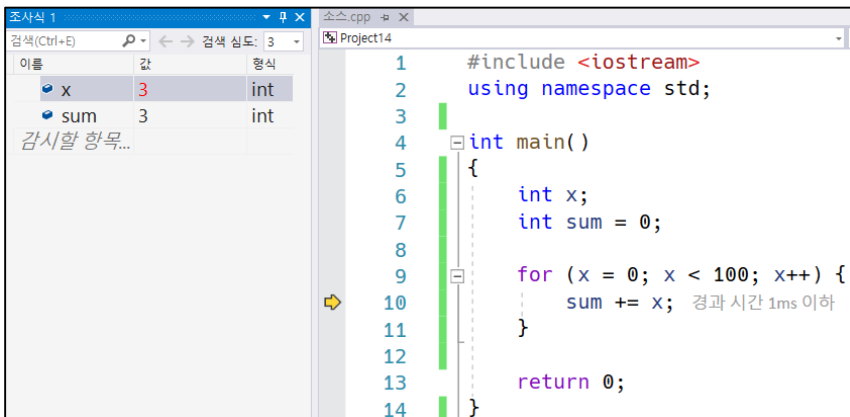
    return 0;
}
```



# Trace 단축키 Ctrl + F10

Trace 도중에도 Ctrl + F10을 쓸 수 있다.

현재 10번 Line을 Trace 중이며, x의 값은 3이다.

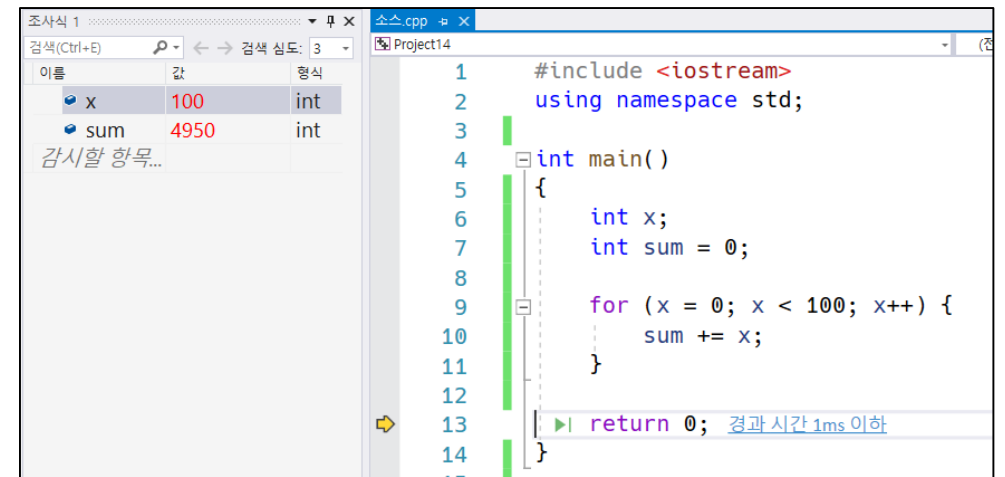


Visual Studio interface showing a C++ program being traced. The variable x is 3 and sum is 3. The trace is at line 10.

| 이름  | 값 | 형식  |
|-----|---|-----|
| x   | 3 | int |
| sum | 3 | int |

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int x;
7     int sum = 0;
8
9     for (x = 0; x < 100; x++) {
10        sum += x; 경과 시간 1ms 이하
11    }
12
13    return 0;
14 }
```

13번 라인까지 한번에 실행이 된다.



Visual Studio interface showing the same C++ program after pressing Ctrl+F10. The variable x is 100 and sum is 4950. The trace has moved to line 13.

| 이름  | 값    | 형식  |
|-----|------|-----|
| x   | 100  | int |
| sum | 4950 | int |

```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int x;
7     int sum = 0;
8
9     for (x = 0; x < 100; x++) {
10        sum += x;
11    }
12
13    return 0; 경과 시간 1ms 이하
14 }
```

# 소스코드 버그 발생 범위 찾기

긴 소스코드에서 버그가 발생한 경우

어디까지 정상이고, 어디서부터 비정상인지

Ctrl + F10 또는 Breaking Pointer 로 **버그의 범위를 찾아내자.**

이 3개의 코드 덩어리 중

어느 범위에서 버그가 났는지

Ctrl + F10으로 확인해봐야 함

```
int main()
{
    int vect[10];

    int x;
    int count = 0;

    for (x = 0; x < 10; x++)
    {
        cin >> vect[x];
    }

    for (x = 0; x < 10; x++)
    {
        if (vect[x] % 2 == 0)
        {
            vect[x]++;
        }
    }

    for (x = 9; x >= 0; x--)
    {
        cout << vect[x];
    }

    return 0;
}
```

# 소스코드 버그 발생 범위 찾기

1

```
5 int main()
6 {
7     int vect[10];
8
9     int x;
10    int count = 0;
11
12    for (x = 0; x < 10; x++)
13    {
14        cin >> vect[x];
15    }
16
17    for (x = 0; x < 10; x++)
18    {
19        if (vect[x] % 2 == 0)
20        {
21            vect[x]++;
22        }
23    }
24
25    for (x = 9; x >= 0; x--)
26    {
27        cout << vect[x];
28    }
29
30    return 0;
31 }
```

1. 16번 Line에서 Ctrl + F10 후,  
vect배열에 정상적으로 값을 입력 받  
아지는 확인.

조사식에 vect 배열을 등록해 두고,  
확인해야한다.

2

```
5 int main()
6 {
7     int vect[10];
8
9     int x;
10    int count = 0;
11
12    for (x = 0; x < 10; x++)
13    {
14        cin >> vect[x];
15    }
16
17    for (x = 0; x < 10; x++)
18    {
19        if (vect[x] % 2 == 0)
20        {
21            vect[x]++;
22        }
23    }
24
25    for (x = 9; x >= 0; x--)
26    {
27        cout << vect[x];
28    }
29
30    return 0;
31 }
```

2. 24번 Line에서 Ctrl + F10 후,  
vect 배열에 예상대로 값이  
채워져있는지 확인.

3

```
5 int main()
6 {
7     int vect[10];
8
9     int x;
10    int count = 0;
11
12    for (x = 0; x < 10; x++)
13    {
14        cin >> vect[x];
15    }
16
17    for (x = 0; x < 10; x++)
18    {
19        if (vect[x] % 2 == 0)
20        {
21            vect[x]++;
22        }
23    }
24
25    for (x = 9; x >= 0; x--)
26    {
27        cout << vect[x];
28    }
29
30    return 0;
31 }
```

3. 30번 Line에서 Ctrl + F10 후,  
콘솔창에 예상대로 출력되었는지 확인

# Ctrl + F10 / Breaking Pointer를 이용한 디버깅

오른쪽 소스코드 같이, 긴 소스코드를  
눈으로 잘못되는 부분을 찾기에는 시간이 너무 오래 걸린다.

Ctrl + F10을 활용해서  
어느 범위에서 버그가 발생하는지  
버그의 범위를 점점 좁혀가야 한다.

버그가 발생하는 범위를 찾아 냈으면  
천천히 F10 / F11을 통해 Trace 필요

```
int main()
{
    int vect[10];
    int x;
    int count = 0;
    for (x = 0; x < 10; x++)
    {
        cin >> vect[x];
    }
    for (x = 0; x < 10; x++)
    {
        if (vect[x] % 2 == 0)
        {
            vect[x]++;
        }
    }
    for (x = 9; x >= 0; x--)
    {
        cout << vect[x];
    }
    for (x = 0; x < 10; x++)
    {
        if (vect[x] % 2 == 0)
        {
            vect[x]++;
        }
    }
    for (x = 9; x >= 0; x--)
    {
        cout << vect[x];
    }
    for (x = 0; x < 10; x++)
    {
        if (vect[x] % 2 == 0)
        {
            vect[x]++;
        }
    }
    for (x = 9; x >= 0; x--)
    {
        cout << vect[x];
    }
    return 0;
}
```



# 암기해야 할 디버깅 단축키

F5 : 디버깅 모드에서 실행

F9 : Breaking Pointer 걸기 / 제거

Ctrl + F5 : 빌드 후 실행

**Ctrl + F10 : 커서 위치부터 실행**

F10 : 한 Line씩 실행 (함수 안으로 진입 안함)

F11 : 한 Line씩 실행 (함수 안으로 진입)

Shift + F5 : 디버깅 모드 종료

Ctrl + Alt + W, 1 : 조사식으로 커서 포커스 맞추기 (ESC : 원상복구)