

민 코 딩 수 업 노 트

---

## 수업노트 디버깅 시작 2



# 배우는 내용

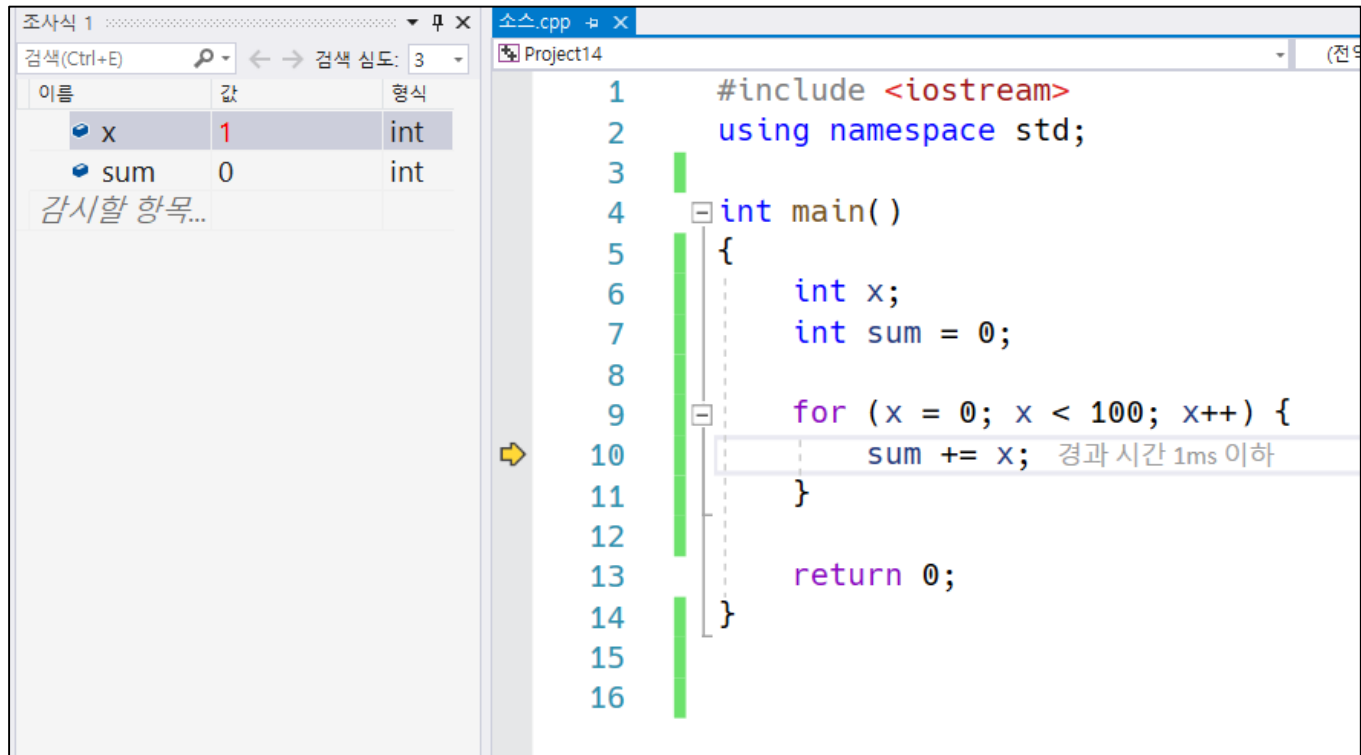
## 디버깅 시작 2

1. Step Over (F10) vs Step In (F11)
2. Runtime Error / Compile Error
3. 디버깅 모드에서 시작 (F5)
4. Breaking Pointer

# 한 Line씩 실행 단축키 : F10, F11

두 단축키 모두 소스코드를 한 줄 씩 실행 해 볼 수 있다.

아래 소스코드에서 F10 / F11을 번갈아 눌러도 똑같이 동작한다.



The screenshot shows a C++ IDE with two main panels. On the left is the 'Watch' window (조사식 1) which displays a table of variables being monitored:

이름	값	형식
x	1	int
sum	0	int

Below the table is a text field labeled '감시할 항목...' (Watch items...). On the right is the 'Source' window (소스.cpp) showing the following C++ code:

```
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 }
```

A yellow arrow points to line 10 in the source code, indicating the current execution point.

# F10 과 F11의 차이

함수를 호출하는 16번 Line에서 F10과 F11의 차이가 발생한다.

```
1  #include <iostream>
2  using namespace std;
3
4  int t;
5
6  void ChangeT()
7  {
8      //t 값을 100으로 교체
9      t = 100;
10 }
11
12 int main()
13 {
14     t = 0;
15
16     ChangeT(); //함수호출
17     cout << t;
18
19     return 0;
20 }
```



```
1  #include <iostream>
2  using namespace std;
3
4  int t;
5
6  void ChangeT()
7  {
8      //t 값을 100으로 교체
9      t = 100;
10 }
11
12 int main()
13 {
14     t = 0;
15
16     ChangeT(); //함수호출
17     cout << t; 경과 시간 1ms 이하
18
19     return 0;
20 }
```

```
1  #include <iostream>
2  using namespace std;
3
4  int t;
5
6  void ChangeT()
7  { 경과 시간 1ms 이하
8      //t 값을 100으로 교체
9      t = 100;
10 }
11
12 int main()
13 {
14     t = 0;
15
16     ChangeT(); //함수호출
17     cout << t;
18
19     return 0;
20 }
```

**F10** : 함수 내용을 다 실행하고 한 Line으로 내려간다. = Step Over

**F11** : 함수 안으로 진입하여 한 Line씩 실행한다. = Step In

# F10 과 F11가 필요한 경우

**F10** : 함수 내용을 다 실행하고 한 Line으로 내려간다. = Step Over

**F11** : 함수 안으로 진입하여 한 Line씩 실행한다. = Step In

## F11로 디버깅 할 때

디버깅 할 때 함수 안으로 진입하여 구체적으로 살펴봐야 하는 경우 사용 한다.

## F10으로 디버깅 할 때

함수 내부를 안 봐도 되는 경우 사용 한다.

```
1  #include <iostream>
2  using namespace std;
3
4  int t;
5
6  void ChangeT( )
7  {
8      //t 값을 100으로 교체
9      t = 100;
10 }
11
12 int main( )
13 {
14     t = 0;
15
16     ChangeT( ); //함수호출
17     cout << t;
18
19     return 0;
20 }
```

# 프로그램을 Build 하는 과정

Build는 아래와 같이 세 동작으로 이루어진다.

1. **Preprocessing** : 소스코드를 정리하고, Header 파일을 CPP 파일에 합치는 과정
2. **Compile** : 프로그래밍 언어로 작성된 1개의 CPP파일을 0과 1 명령어로 바꾼다.
3. **Linking** : 여러 개의 CPP 파일들을 하나로 합치고 (CPP파일이 여러개 있는 경우), Library를 연결시키고, 실행파일이 될 수 있도록 추가 코드를 붙여준다.

# Error의 종류

개발을 하면서 발생하는 Error 종류는 다음과 같다.

1. **Compile Error** : Compile 도중 발생하는 Error, 대부분 문법 에러이며 Build시 발생한다.
2. **Link Error** : Linking 도중 발생하는 Error, Build시 발생한다.
3. **Runtime Error** : Build에 성공했지만, 실행도중에 버그로 인해 발생한다.

# Compile Error 예시와 해결책

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

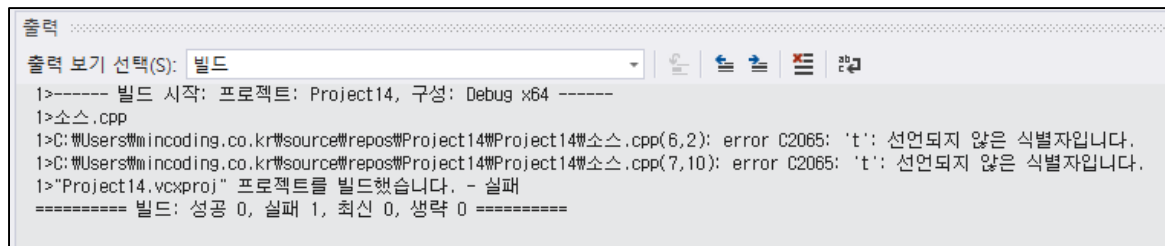
int main()
{
    t = 0;
    cout << t;

    return 0;
}
```

변수를 선언하지 않고, 사용했기 때문에 Compile Error 발생

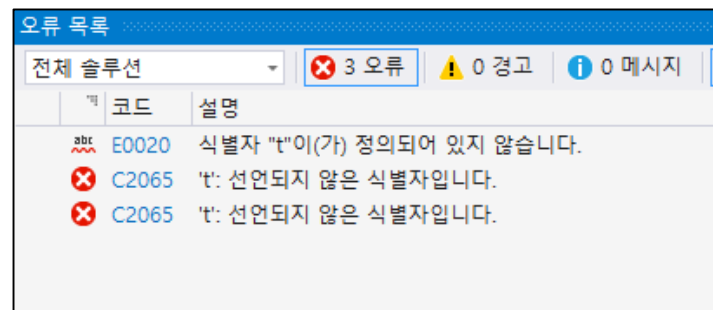
## 디버깅 방법

1. 빨간색 밑줄이 그어진다. (정확하진 않음)
2. 오류 창 or 출력 창을 확인



출력 :  
출력 보기 선택(S): 빌드

```
1>----- 빌드 시작: 프로젝트: Project14, 구성: Debug x64 -----
1>소스.cpp
1>C:\Users\mincoding.co.kr\source\repos\Project14\Project14\소스.cpp(6,2): error C2065: 't': 선언되지 않은 식별자입니다.
1>C:\Users\mincoding.co.kr\source\repos\Project14\Project14\소스.cpp(7,10): error C2065: 't': 선언되지 않은 식별자입니다.
1>"Project14.vcxproj" 프로젝트를 빌드했습니다. - 실패
===== 빌드: 성공 0, 실패 1, 최신 0, 생략 0 =====
```



오류 목록		
전체 솔루션		
3 오류 0 경고 0 메시지		
이름	코드	설명
	E0020	식별자 "t"이(가) 정의되어 있지 않습니다.
	C2065	't': 선언되지 않은 식별자입니다.
	C2065	't': 선언되지 않은 식별자입니다.



# Link Error 예시와 해결책

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

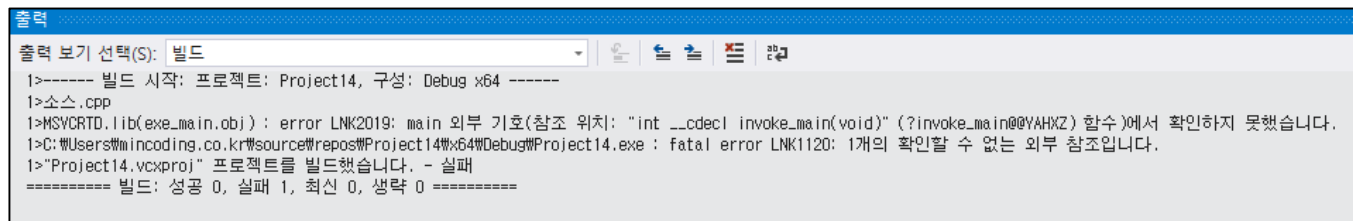
int abs()
{
    int t = 0;
    cout << t;

    return 0;
}
```

main함수를 만들지 않았기 때문에 Link Error 발생

## 디버깅 방법

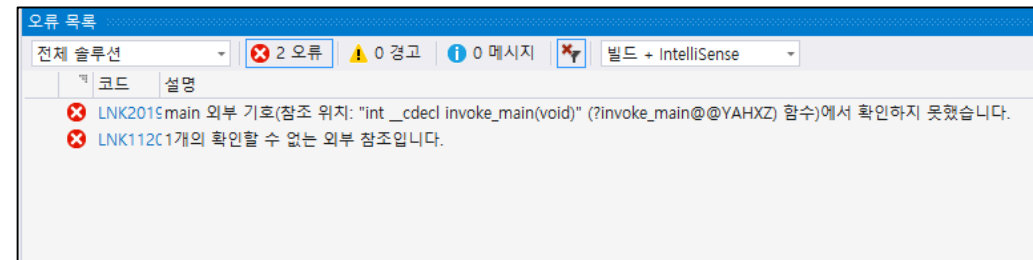
1. 오류 창 or 출력 창을 확인



출력

출력 보기 선택(S): 빌드

```
1>----- 빌드 시작: 프로젝트: Project14, 구성: Debug x64 -----
1>소스.cpp
1>MSVCRTD.lib(exe.main.obj): error LNK2019: main 외부 기호(참조 위치: "int __cdecl invoke_main(void)" (?invoke_main@@YAHXZ) 함수)에서 확인하지 못했습니다.
1>C:\Users\mincoding.co.kr\source\repos\Project14\Debug\Project14.exe: fatal error LNK1120: 1개의 확인할 수 없는 외부 참조입니다.
1>"Project14.vcxproj" 프로젝트를 빌드했습니다. - 실패
===== 빌드: 성공 0, 실패 1, 최신 0, 생략 0 =====
```



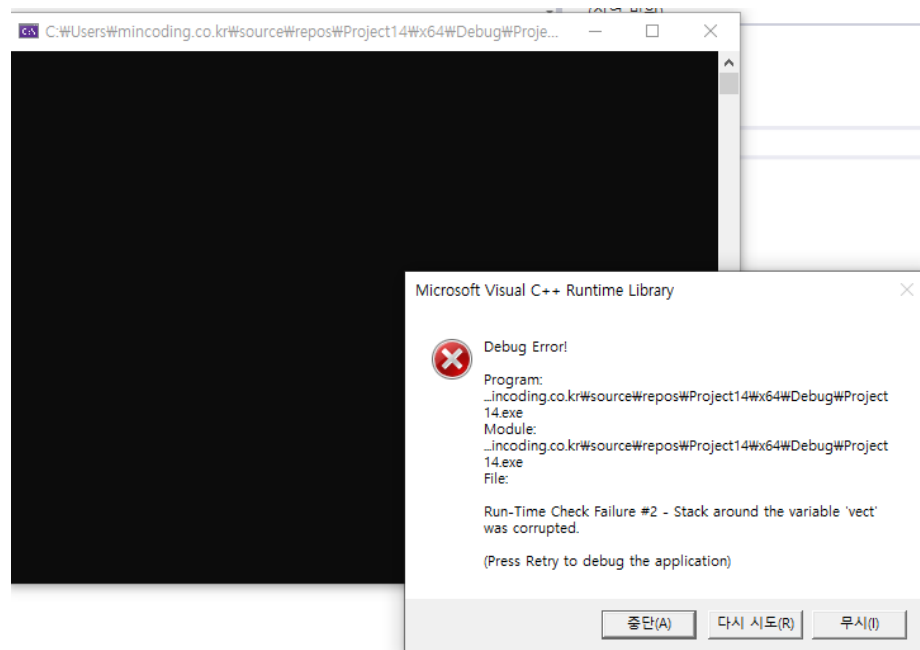
# Runtime Error 예시

## Ctrl + F5 누르면 다음과 같은 Runtime Error 발생

문법적으로 문제는 없지만 실행도중 에러메세지 발생

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

int main()
{
    int vect[10];
    vect[-1] = 123;
    return 0;
}
```



# Runtime Error 해결방법 : Trace

어느 Line부터 조사식에 예상하지 못한 값들이 들어가는지 알아내야 한다.

**미리 예측부터 한 뒤**, Trace로 결과를 확인하는 것이 중요하다.

(특정 Line까지 진행했을 때 미리 예측을 먼저하고 난 뒤, 결과를 확인하는 것)

## Runtime Error 디버깅 방법

```
while(1)
```

```
{
```

1. 특정 Line까지 실행했을 때, 어떤 값들이 들어갈지 **미리 예측**
2. Trace를 진행
3. **예측한 값**과 조사식 값과 일치하는지 확인

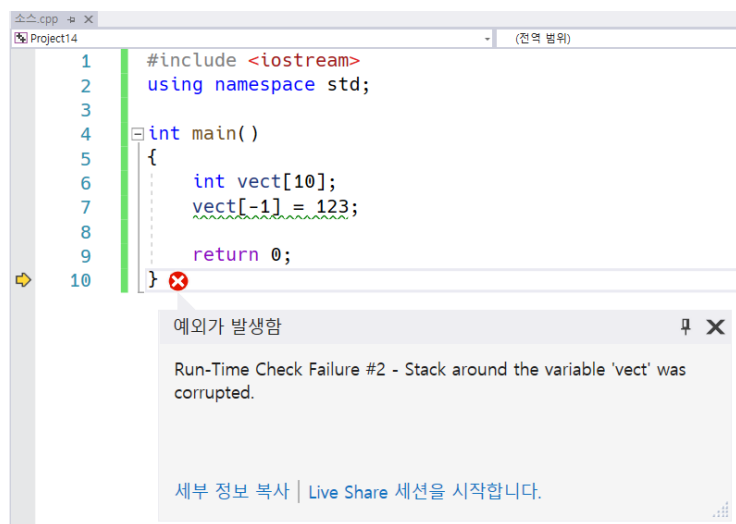
```
}
```

추리력을 발휘해서  
범인(버그)을 찾아낼 것

# F5와 Ctrl + F5의 차이

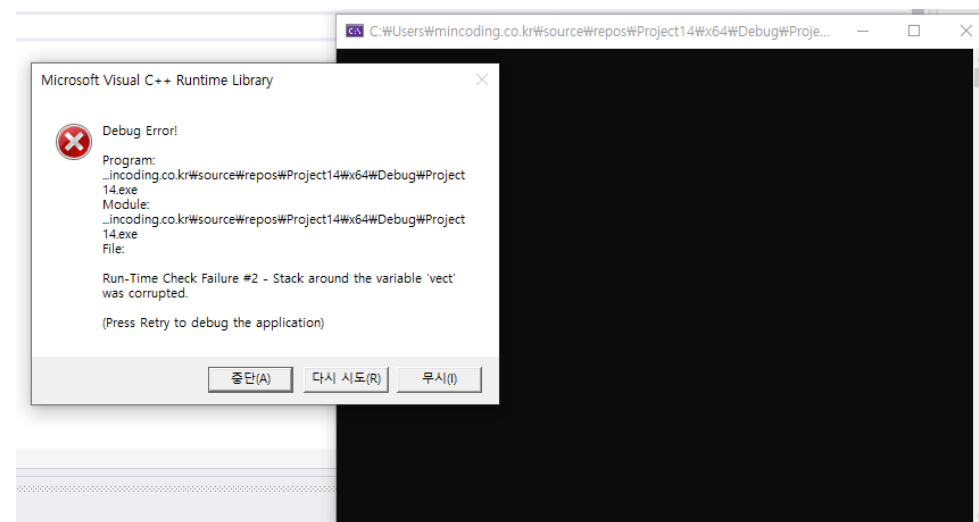
- Ctrl + F5 (빌드 후 실행) : 실행파일을 만든 후 실제로 실행 하는 것.
- F5 (디버깅 모드에서 실행) : F10을 처음부터 끝까지 누르는 것과 같다.

두 기능 모두 소스코드 실행결과를 보여준다는 점은 같지만, Runtime Error시 차이가 분명히 들어난다.



## F5 눌렀을 때 화면

디버깅모드가 되고, 에러메세지와 에러발생위치까지 알려준다.



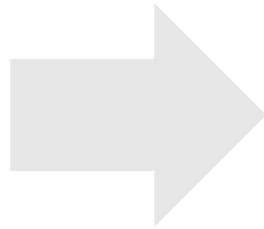
## Ctrl + F5 눌렀을 때 화면

에러메세지가 출력되고, 중단 버튼을 누르면 프로그램이 종료된다.

# Breaking Pointer

특정 Line까지 실행된 결과를 보고싶을 때 사용한다. (물론 F10을 연타해도 결과는 같다.)  
소스코드를 클릭하고 F9를 누르면 Breaking Pointer가 찍힌다.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int vect[10];
7      vect[0] = 5;
8      vect[1] = 5;
9      vect[2] = 5;
10     vect[3] = 5;
11     vect[4] = 5;
12     vect[5] = 5;
13     vect[6] = 5;
14
15     return 0;
16 }
```



조사식 1

이름	값	형식
vect	0x00000022ff8ff858 ...	int[10]
[0]	5	int
[1]	5	int
[2]	-858993460	int
[3]	-858993460	int
[4]	-858993460	int
[5]	-858993460	int
[6]	-858993460	int
[7]	-858993460	int
[8]	-858993460	int
[9]	-858993460	int

감시할 항목...

소스.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int vect[10];
7      vect[0] = 5;
8      vect[1] = 5;
9      vect[2] = 5;
10     vect[3] = 5;
11     vect[4] = 5;
12     vect[5] = 5;
13     vect[6] = 5;
14
15     return 0;
16 }
```

9번 Line 클릭 후 F9를 누르면 빨간 동그라미 (Breaking Pointer)가 생긴다.  
동그라미를 클릭하거나, F9를 한번 더 누르면 없어진다.

F5를 누르면 Breaking Pointer에서 멈추어서,  
9번 Line전까지 실행했을 때 결과를 조사식으로 확인 가능하다.

# 다수의 Breaking Pointer

Breaking Pointer를 여러 개 걸고 F5를 반복적으로 누르면, 해당되는 Breaking Pointer 까지 실행 된다.  
도중 F10 or F11을 눌러도 잘 동작한다.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int vect[10];
7      vect[0] = 5;
8      vect[1] = 5;
9      vect[2] = 5;
10     vect[3] = 5;
11     vect[4] = 5;
12     vect[5] = 5;
13     vect[6] = 5;
14
15     return 0;
16 }
```

다수의 Breaking Pointer 걸기

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int vect[10];
7      vect[0] = 5;
8      vect[1] = 5;
9      vect[2] = 5;
10     vect[3] = 5;
11     vect[4] = 5;
12     vect[5] = 5;
13     vect[6] = 5;
14
15     return 0;
16 }
```

F5 한번 눌렀을 때

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int vect[10];
7      vect[0] = 5;
8      vect[1] = 5;
9      vect[2] = 5;
10     vect[3] = 5;
11     vect[4] = 5;
12     vect[5] = 5;
13     vect[6] = 5;
14
15     return 0;
16 }
```

F5 한번 더 눌렀을 때

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int vect[10];
7      vect[0] = 5;
8      vect[1] = 5;
9      vect[2] = 5;
10     vect[3] = 5;
11     vect[4] = 5;
12     vect[5] = 5;
13     vect[6] = 5;
14
15     return 0;
16 }
```

F10 눌렀을 때

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int vect[10];
7      vect[0] = 5;
8      vect[1] = 5;
9      vect[2] = 5;
10     vect[3] = 5;
11     vect[4] = 5;
12     vect[5] = 5;
13     vect[6] = 5;
14
15     return 0;
16 }
```

이어서 F5 눌렀을 때