

민 코 딩 수 업 노 트

---

# 수업노트 디버깅 시작 1



# 배우는 내용

## 디버깅 시작 1

1. 디버깅과 Trace의 이해
2. Trace - Step Over (F10)
3. 조사식 (watch)

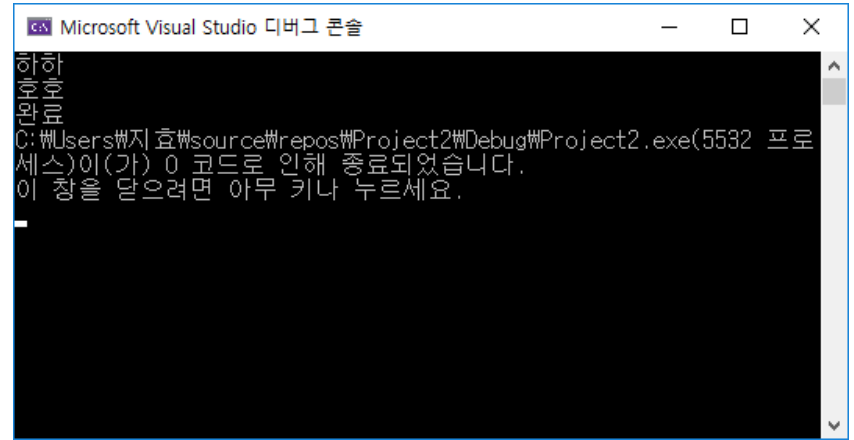
# 빌드하기

빌드 후 실행 (**Ctrl + F5**) : 소스코드 전체를 빌드 후, 실행하여 결과를 확인할 수 있다.

```
#include <iostream>
using namespace std;
int main()
{
    cout << "하하" << endl;
    cout << "호호" << endl;
    cout << "완료";

    return 0;
}
```

빌드 후 실행  
(Ctrl + F5)



The screenshot shows the 'Microsoft Visual Studio 디버그 콘솔' (Debug Console) window. It displays the output of the program: '하하' (HaHa), '호호' (HoHo), and '완료' (Complete). Below the output, it shows the file path 'C:\Users\zzz\source\repos\Project2\Debug\Project2.exe(5532 프로세스)' and a message: '이(가) 0 코드로 인해 종료되었습니다. 이 창을 닫으려면 아무 키나 누르세요.' (This was terminated by code 0. Press any key to close this window.)

# 디버깅이란?

디버깅이란 De + Bug + ing 로 버그를 잡는 행동을 뜻한다.

디버깅하는 방법

- 소스코드를 직접 눈으로 읽으며 찾아낸다.
- 소스코드 사이에 cout을 넣어 예상대로 동작하는지 확인한다.
- **Trace(트레이스) 기능을 이용하여 디버깅한다.**

# Trace 체험하기

Trace : 소스코드 한 줄씩 실행, 단축키 **F10**

Trace 종료 : **Shift + F5**

```
#include <iostream>
using namespace std;
int main()
{
    cout << "하하" << endl;
    cout << "호호" << endl;
    cout << "완료";

    return 0;
}
```

F10을 눌러 **디버깅 모드** 진입

Shift + F5 로 **디버깅 모드** 종료

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      cout << "하하" << endl;
6      cout << "호호" << endl;
7      cout << "완료";
8
9      return 0;
10 }
11
```

노란색 화살표가 등장

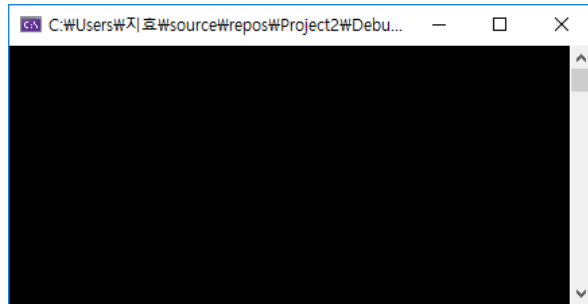
단축키를 반드시 암기할 것

# Trace 하기

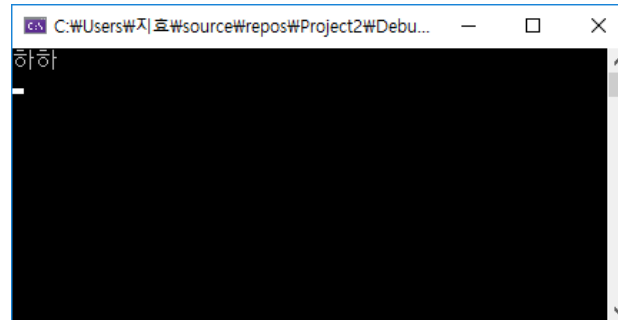
F10을 반복적으로 누르면 한 줄씩 실행된다.

종료하는 단축키는 Shift + F5

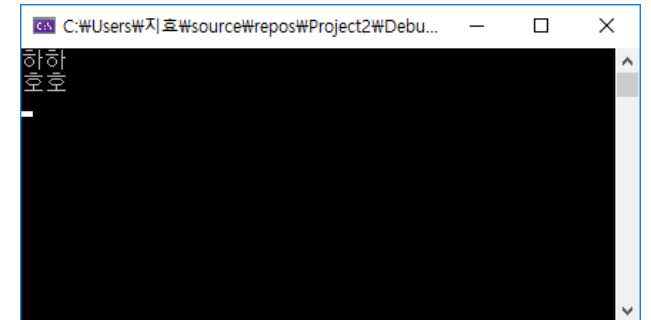
```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      cout << "하하" << endl;
6      cout << "호호" << endl;
7      cout << "완료";
8
9      return 0;
10 }
```



```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      cout << "하하" << endl;
6      cout << "호호" << endl;
7      cout << "완료";
8
9      return 0;
10 }
```



```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      cout << "하하" << endl;
6      cout << "호호" << endl;
7      cout << "완료"; 경과 시간 41
8
9      return 0;
10 }
```



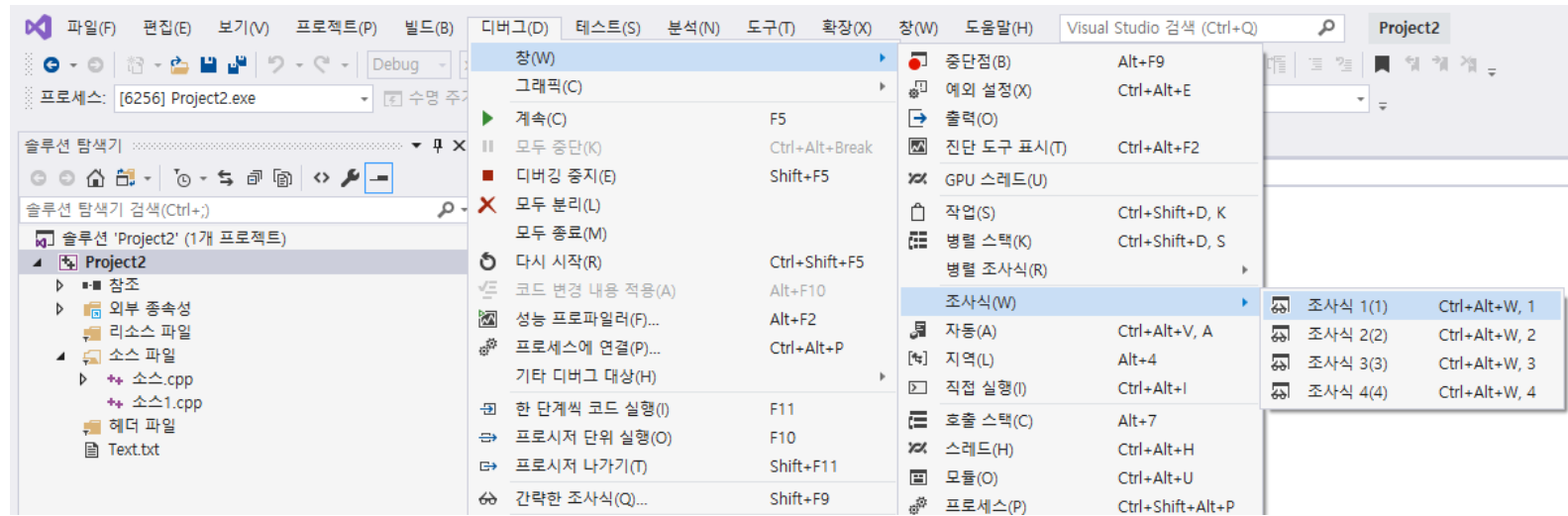
# 변수에 저장된 값 확인하기 1

```
#include <iostream>
using namespace std;
int main()
{
    int a = 10;
    int b = 20;
    int c = a + b;

    return 0;
}
```

Trace로 변수에 어떤 값이 들어있는지 확인할 수 있다.

1. 아래 소스코드를 입력하고 F10 누르기
2. [디버그] > [창] > [조사식] > [조사식 1] 을 눌러 조사식 창을 연다.



# 변수에 저장된 값 확인하기 2

변수 a, b, c 값을 확인하기 위해, 조사식 창에 a, b, c를 등록한다.  
F10을 반복적으로 눌러보면 a, b, c에 값이 들어가는 것을 확인할 수 있다.

```
#include <iostream>
using namespace std;
int main()
{
    int a = 10;
    int b = 20;
    int c = a + b;

    return 0;
}
```

조사식 1	
검색(Ctrl+E)	
이름	값

변수 a,b,c 등록하기

조사식 1	
검색(Ctrl+E)	
이름	값
a	1
b	261089186
c	261088478

이제 F10을 천천히 반복적으로 눌러봄

조사식 1	
검색(Ctrl+E)	
이름	값
a	10
b	20
c	30

변수 a, b, c 값이 들어감



# Trace를 하는 이유

1. 프로그램 안에 찾기 힘든 버그들을 찾아낼 수 있다.  
→ 빠른 디버깅을 위해 Trace를 사용한다.
2. Trace를 통한 디버깅 능력이 좋으면 야근을 줄일 수 있다.  
→ 소스코드가 복잡할 수록 버그 찾기가 어렵다. Trace를 쓰면 버그를 쉽게 찾을 수 있다.
3. 프로그래밍을 배울 때 부터 Trace습관을 들여놓으면, 디버깅 능력을 크게 키울 수 있다.  
→ 소스코드를 작성 후, Trace를 한번씩 해 보는 것을 크게 권장 함 (강력추천)