# 땅울림
# 객체 스터디

Made by 규정

# 3주차

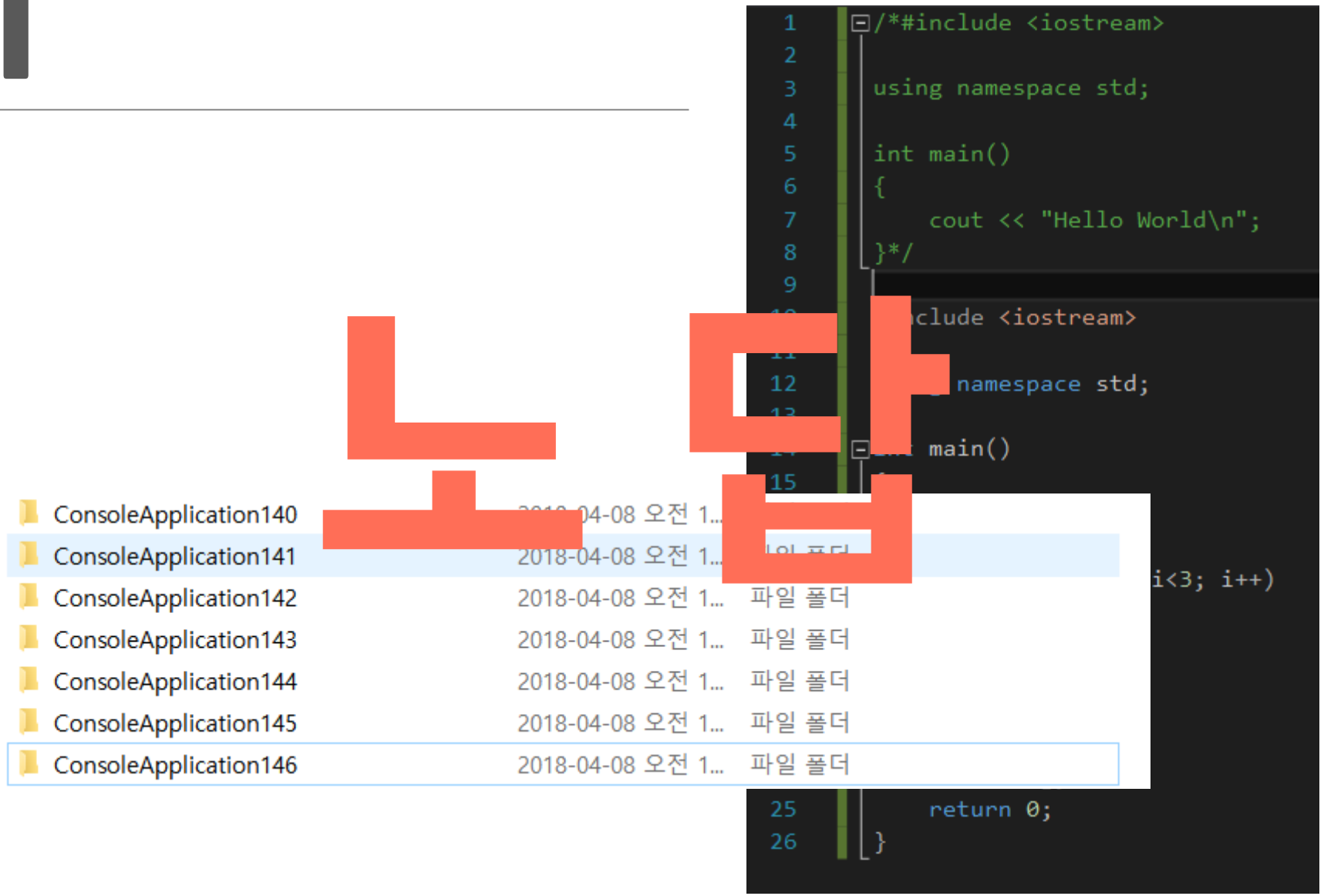
1 파일 관리

2 함수

3 배열

4 문자열

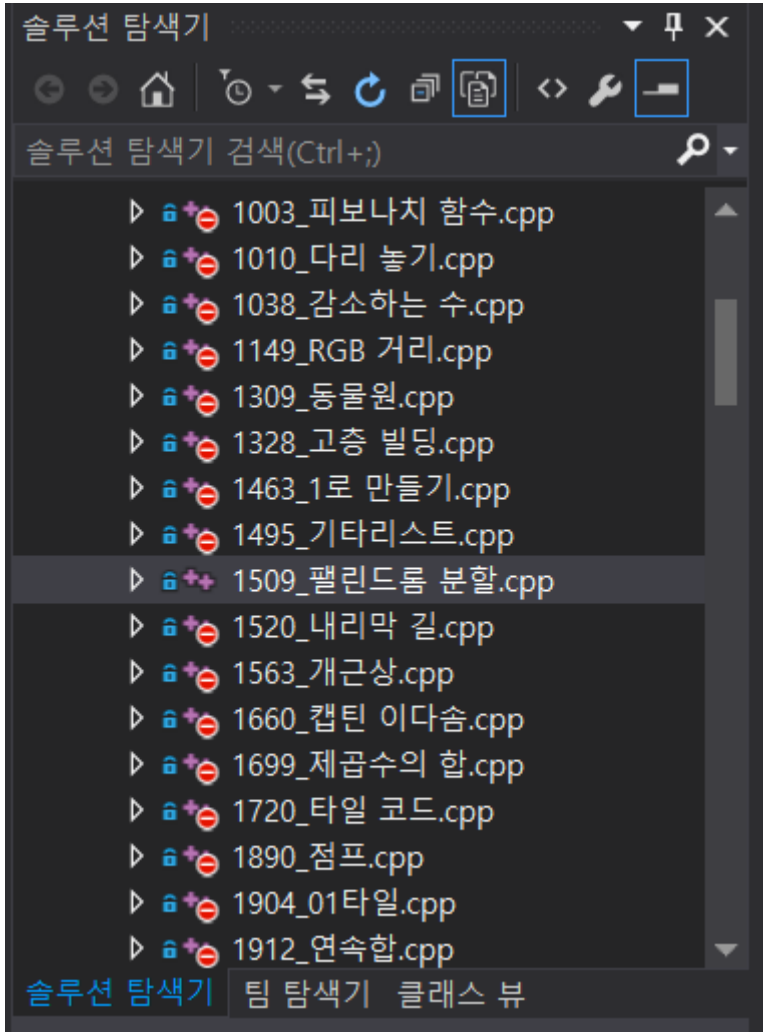
# 1 파일 관리

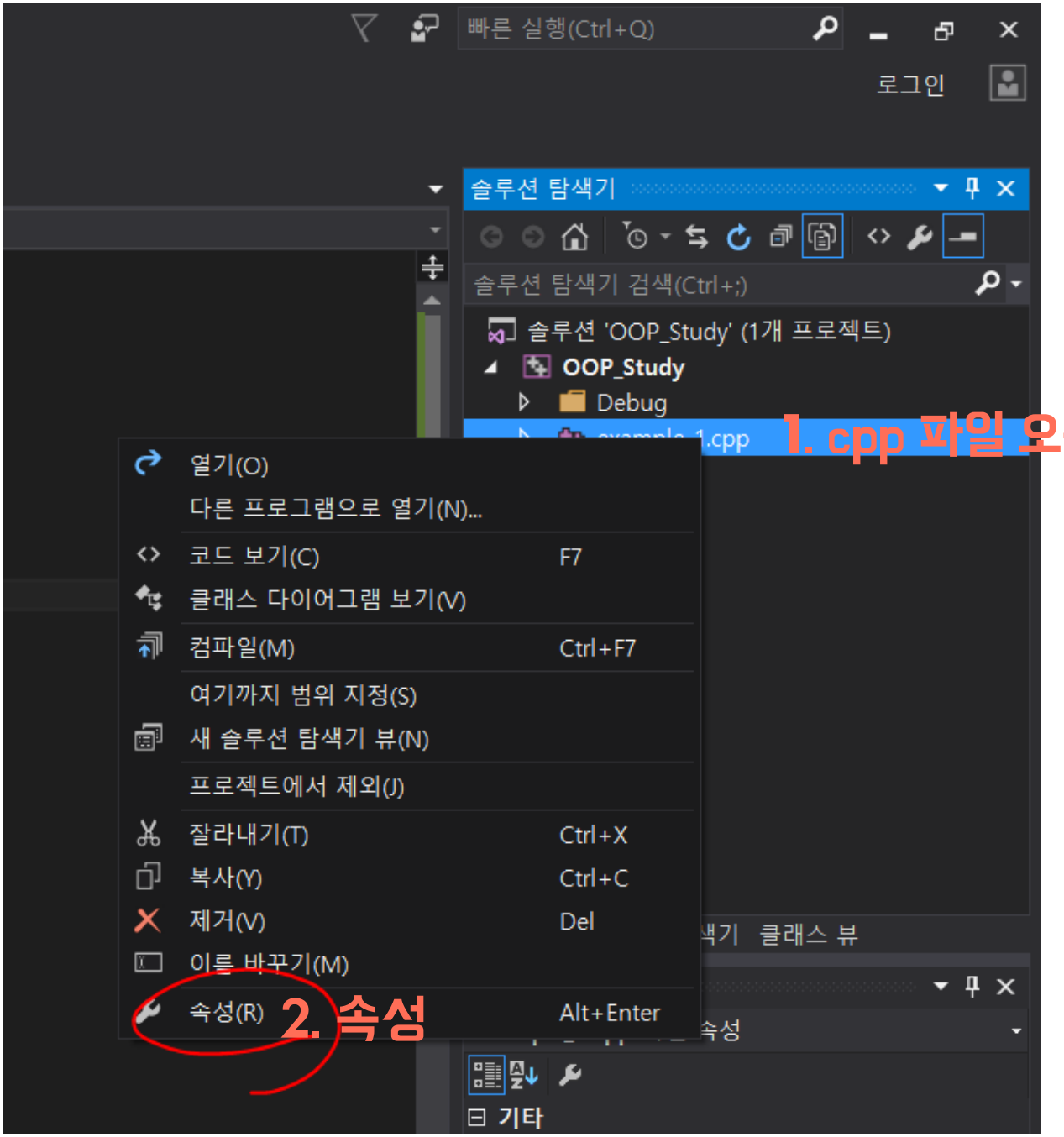

/*#include <iostream>
using namespace std;
int main()
{
cout << "Hello World\n";
}*/
clude <iostream>
namespace std;
main()
i<3; i++)
return 0;
}
ConsoleApplication140 2018-04-08 오전 1...
ConsoleApplication141 2018-04-08 오전 1...
ConsoleApplication142 2018-04-08 오전 1... 파일 폴더
ConsoleApplication143 2018-04-08 오전 1... 파일 폴더
ConsoleApplication144 2018-04-08 오전 1... 파일 폴더
ConsoleApplication145 2018-04-08 오전 1... 파일 폴더
ConsoleApplication146 2018-04-08 오전 1... 파일 폴더

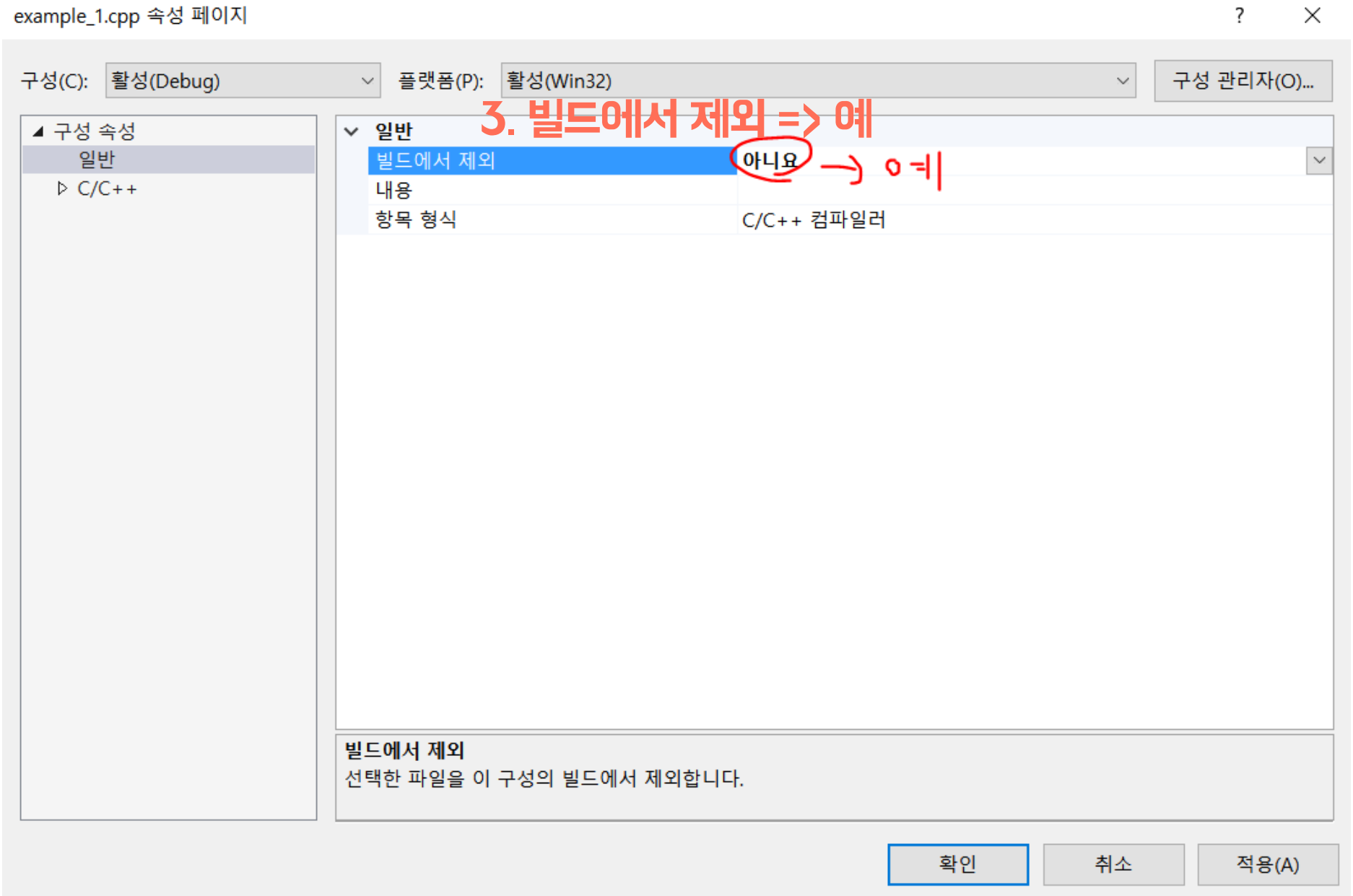
# 1 파일 관리

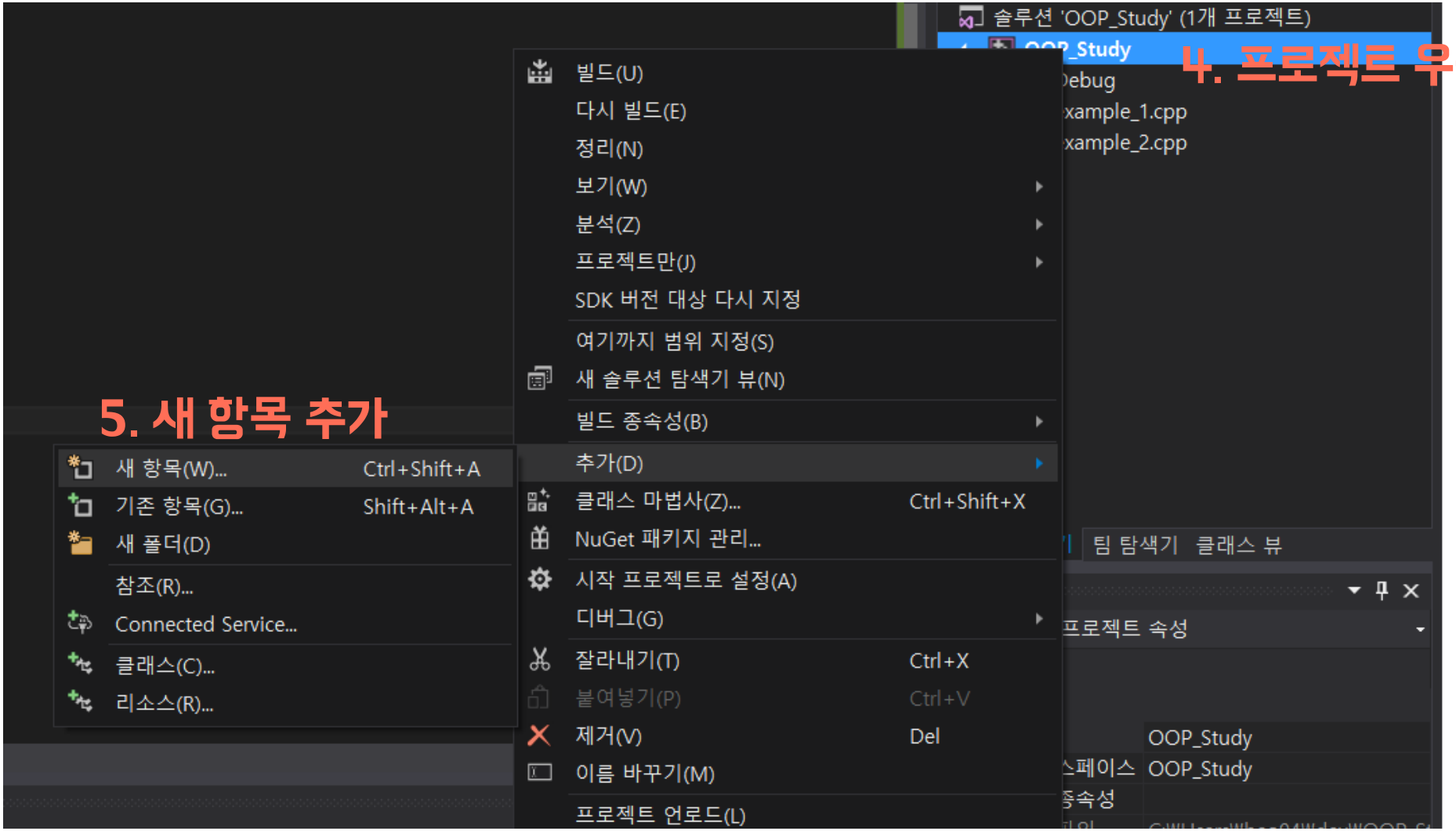
# Like This

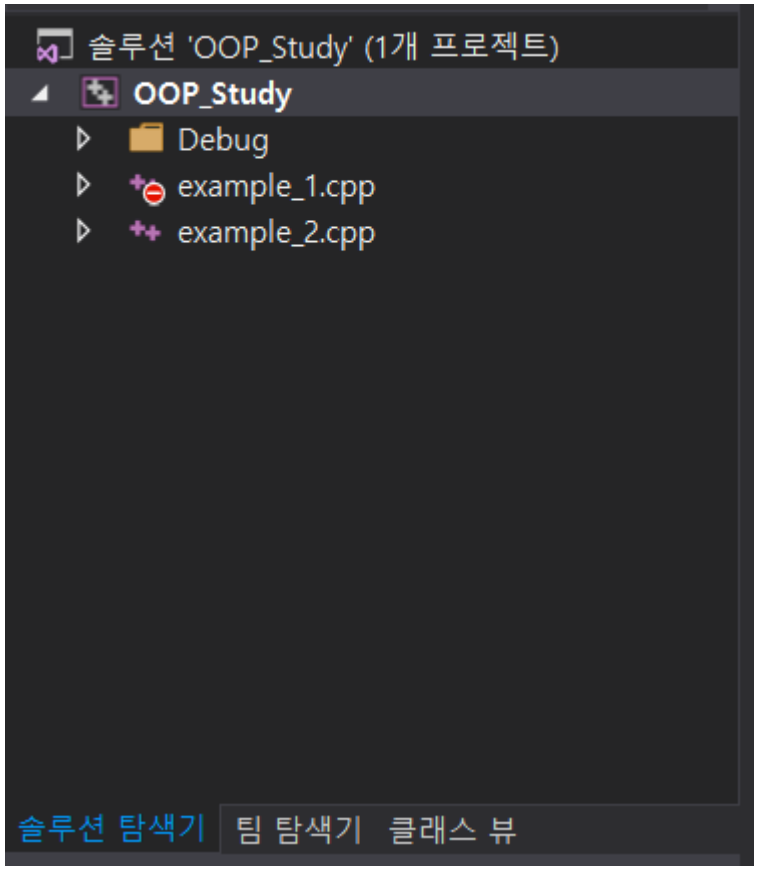
| | | |
|---|---|---|
| AlgoStudy | 2017-04-14 오후 1... | 파일 폴더 |
| Problem | 2017-06-16 오전 1... | 파일 폴더 |
| network | 2017-09-12 오후 3... | 파일 폴더 |
| embe | 2017-09-12 오후 1... | 파일 폴더 |
| codeFestival | 2017-10-01 오후 5... | 파일 폴더 |
| SecondTouch | 2017-10-12 오후 1... | 파일 폴더 |
| security | 2017-11-17 오전 2... | 파일 폴더 |
| AES | 2017-11-30 오전 2... | 파일 폴더 |
| Algorithm | 2017-12-11 오후 6... | 파일 폴더 |
| sj | 2017-12-17 오전 1... | 파일 폴더 |
| test | 2018-02-26 오후 8... | 파일 폴더 |
| codeplus | 2018-03-02 오후 9... | 파일 폴더 |
| FirstTouch | 2018-03-03 오후 7... | 파일 폴더 |




빠른 실행(Ctrl+Q)
로그인
솔루션 탐색기
솔루션 탐색기 검색(Ctrl+;)
솔루션 'OOP_Study' (1개 프로젝트)
OOP_Study
Debug
1. cpp 파일 오른쪽 클릭
열기(O)
다른 프로그램으로 열기(N)...
코드 보기(C) F7
클래스 다이어그램 보기(V)
컴파일(M) Ctrl+F7
여기까지 범위 지정(S)
새 솔루션 탐색기 뷰(N)
프로젝트에서 제외(J)
잘라내기(T) Ctrl+X
복사(Y) Ctrl+C
제거(V) Del
이름 바꾸기(M)
속성(R) Alt+Enter
2. 속성
클래스 뷰
기타


example_1.cpp 속성 페이지
구성(C): 활성(Debug)
플랫폼(P): 활성(Win32)
구성 관리자(O)...
3. 빌드에서 제외 => 예
구성 속성
일반
C/C++
일반
빌드에서 제외
아니요
예
내용
항목 형식
C/C++ 컴파일러
빌드에서 제외
선택한 파일을 이 구성의 빌드에서 제외합니다.
확인
취소
적용(A)

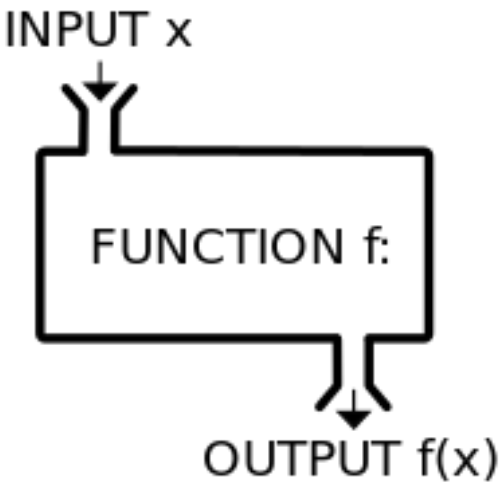

솔루션 'OOP_Study' (1개 프로젝트)
4. 프로젝트 우클
빌드(U)
다시 빌드(E)
정리(N)
보기(W)
분석(Z)
프로젝트만(J)
SDK 버전 대상 다시 지정
여기까지 범위 지정(S)
새 솔루션 탐색기 뷰(N)
빌드 종속성(B)
추가(D)
클래스 마법사(Z)... Ctrl+Shift+X
NuGet 패키지 관리...
시작 프로젝트로 설정(A)
디버그(G)
잘라내기(T) Ctrl+X
붙여넣기(P) Ctrl+V
제거(V) Del
이름 바꾸기(M)
프로젝트 언로드(L)
5. 새 항목 추가
새 항목(W)... Ctrl+Shift+A
기존 항목(G)... Shift+Alt+A
새 폴더(D)
참조(R)...
Connected Service...
클래스(C)...
리소스(R)...
팀 탐색기
클래스 뷰
프로젝트 속성
OOP_Study
OOP_Study


솔루션 'OOP_Study' (1개 프로젝트)
OOP_Study
Debug
example_1.cpp
example_2.cpp
솔루션 탐색기
팀 탐색기
클래스 뷰

# 2 함수

## 수학에서의 함수

**=> x를 넣으면 계산해서 f(x)를 반환**




$$f(x) = 3x + 7$$

# 2 함수

## 프로그래밍에서의 함수

특정 작업을 수행하기 위한 코드의 집합

Input을 받아서 작업을 수행후 Output을 반환

프로그램의 기본 단위

```
int Function1(int x)
{   // x를 입력받고 3x + 7을 리턴
    return 3 * x + 7;
}
```

# 2 함수

## 함수를 사용하는 이유

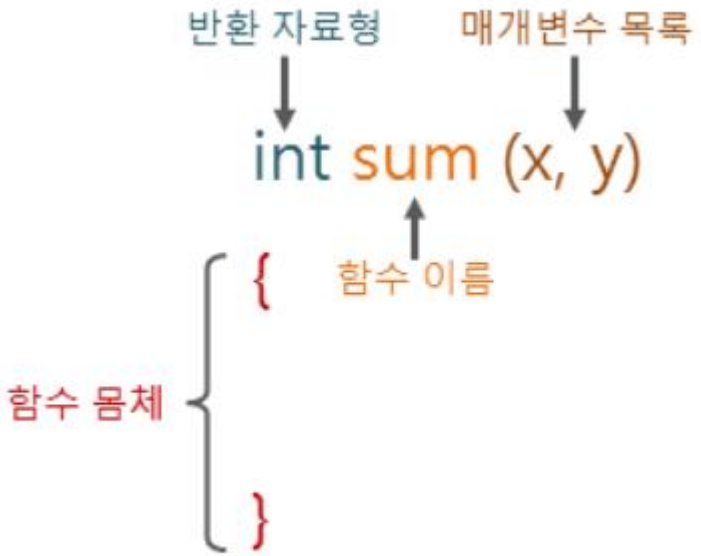
1. 프로그램에서 같은 작업을 여러 번 반복해야 할 때 불필요한 **코드의 중복**을 피할 수 있다!

2. 전체적인 코드의 가독성이 좋아진다.

3. 프로그램을 유지보수하기 쉬워진다.

# 2 함수

## 함수의 선언



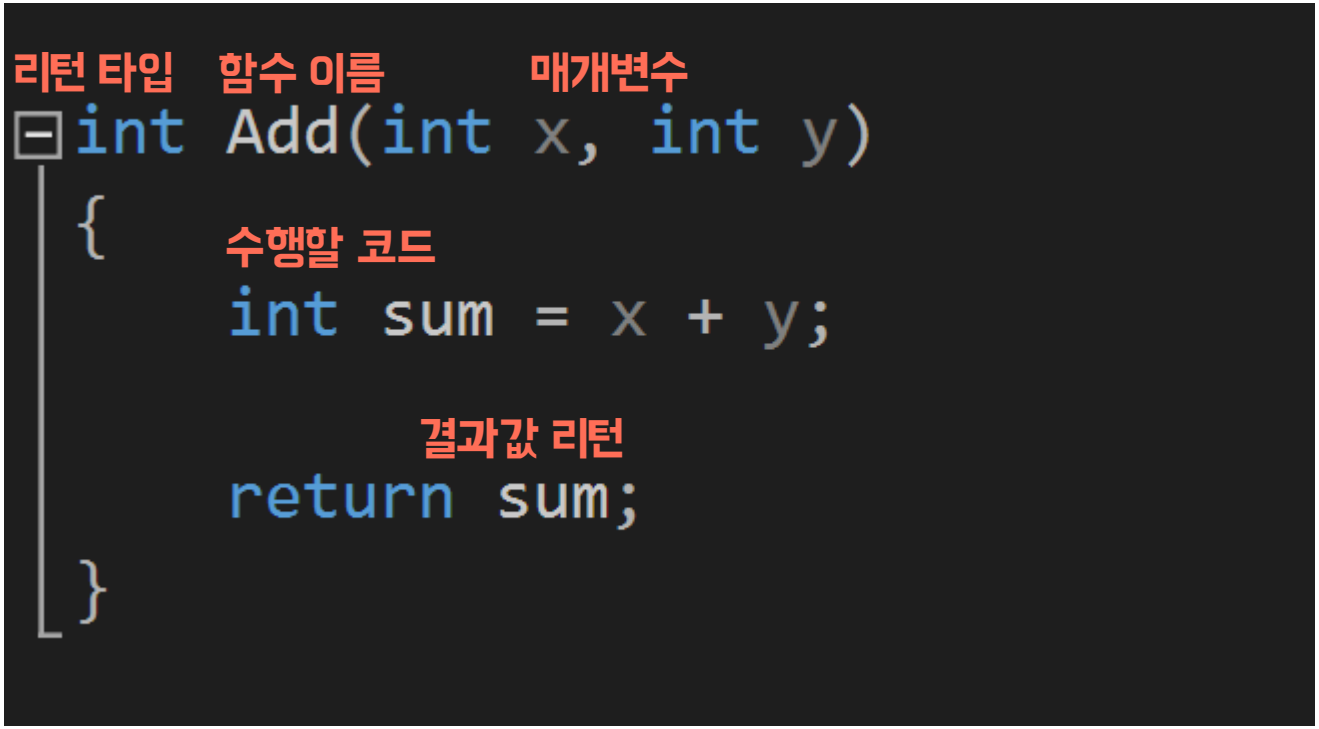

1. 반환 type : 함수가 작업을 마치고 반환하는 데이터의 자료형을 명시

2. 함수 이름 : 함수를 호출하기 위한 이름

3. 매개변수 : 함수를 호출할 때 전달하는 인자의 수와 자료형

4. 함수 몸체 : 함수의 기능을 수행하는 코드

# 2 함수


리턴 타입
함수 이름
매개변수
int Add(int x, int y)
{
수행할 코드
int sum = x + y;
결과값 리턴
return sum;
}

# 2 함수

## void 함수 = return 값이 없는 함수

# 2 함수

매개변수(Parameter)

```cpp
void PrintScore(int x, int y)
{
    int sum = x + y;

    cout << "math = " << x << "\n";
    cout << "english = " << y << "\n";
    cout << "sum = " << sum << "\n";
}

int main()
{
    cin.tie(NULL);
    std::ios::sync_with_stdio(false);

    int math = 50;
    int english = 70;
                인자(Argument)
    PrintScore(math, english);

    return 0;
}
```

(int x = math;
int y = engilish;)

math,english를
파라미터로 전달

# 2 함수

```cpp
int Add(int x, int y)
{
    int sum = x + y;

    return sum;
}

int main()
{
    int math = 50;
    int korean = 70;

    int sum = Add(math, korean);

    cout << sum << "\n";

    return 0;
}
```

# 2 함수

## 함수의 프로토타입

함수는 사용되기 전에 반드시 선언이 되어있어야 한다

main 함수 위에 계속 새로운 함수를 작성하면 나중에 main을 찾기가 힘들다

“함수 정의가 뒤에 나올 거니까 에러 없이 넘어가줘”라고 알려주는 것

# 2 함수

```
int Add(int x, int y); 프로토타입

int main()
{
    int math = 50;
    int korean = 70;

    int sum = Add(math, korean);

    cout << sum << "\n";

    return 0;
}

int Add(int x, int y)
{
    int sum = x + y;

    return sum;
}
```

# 3 배열

## 배열이란?

C++의 가장 기본적인 자료구조

같은 타입으로 이루어진 변수들을 한꺼번에 다룰 수 있음

모든 배열은 메모리에 연속적으로 할당된다.

| 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|

# 3 배열

## 숫자 100개를 입력받고 싶을 때

```
int number0, number1, number2, ... number99;
cin >> number0 >> number1 >> ... number 99
```

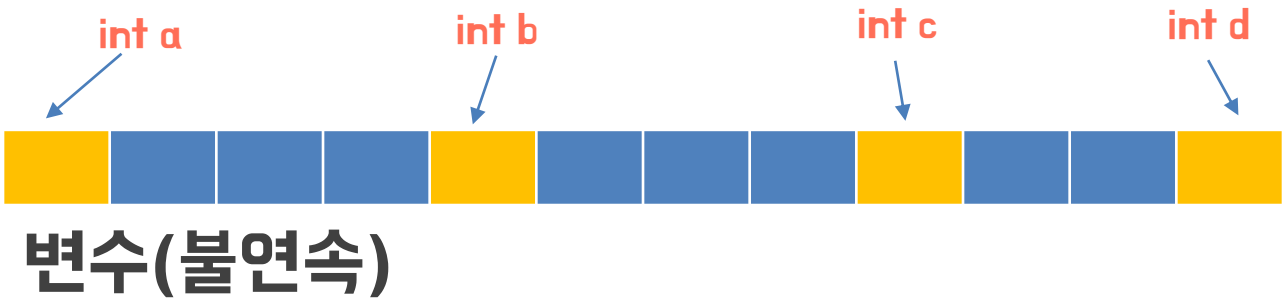
=> 노답

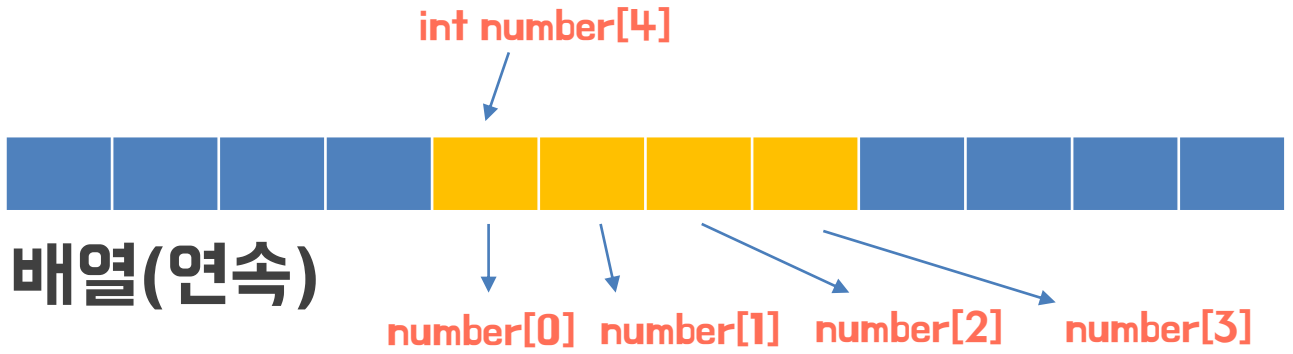
```
int number[100];
for(int i = 0; i < 99; i++)
{
    cin >> number[i];
}
```

=> 정답

# 3 배열

## 메모리 구조


int a
int b
int c
int d
변수(불연속)


int number[4]
배열(연속)
number[0]
number[1]
number[2]
number[3]

# 3 배열

## 배열의 선언

자료형 + 배열이름 + [크기]

```
int score[30];
double array[100];
```

# 3 배열

## 배열의 선언

[ ] < 대괄호 안에는 무조건 상수만 들어가야 한다.

```
int N = 4;

int arr[N];
```

불가능

# 3 배열

한 번 선언하면 크기 변경 불가능!

모자란 거보다 남는 게 나음

애매하면 차라리 크게 선언

# 3 배열

## 배열의 초기화

자료형 배열이름 [크기] = {값, 값, 값, 값, …};

int score[30] = {20, 37, 88, 73, … 91};

or

자료형 배열이름 [] = {값, 값, 값, 값, …};

int score[] = {20, 37, 88, 73, … 91};

# 배열의 초기화(default 값)

```
int score[30] = { 0 };
char word[20] = { '/0' };
```

# 3 배열

초기화해주지 않으면 쓰레기값이 들어있다

반드시 초기화해주는 습관 들일 것!

# 배열 접근

```
int array_score[5] = { 10, 20, 30, 40, 50 };
```

| array_score[0] | array_score[1] | array_score[2] | array_score[3] | array_score[4] |
|---|---|---|---|---|
| 10 | 20 | 30 | 40 | 50 |

# 3 배열

## 배열을 다루는 가장 쉬운 방법
## => for문을 사용하자

```
int array_score[100] = { 0 };
for (int i = 0; i < 100; i++)
{
    array_score[i] = i;
}
```

# 잘못된 코드 찾기

1

```
int score[30] = { 1 };
for (int i = 0; i < 30; i++)
{
    cout << score[i] << "\n";
}
```

2

```
int arr[100];
for (int i = 0; i <= 100; i++)
{
    arr[i] = 0;
}
```

3

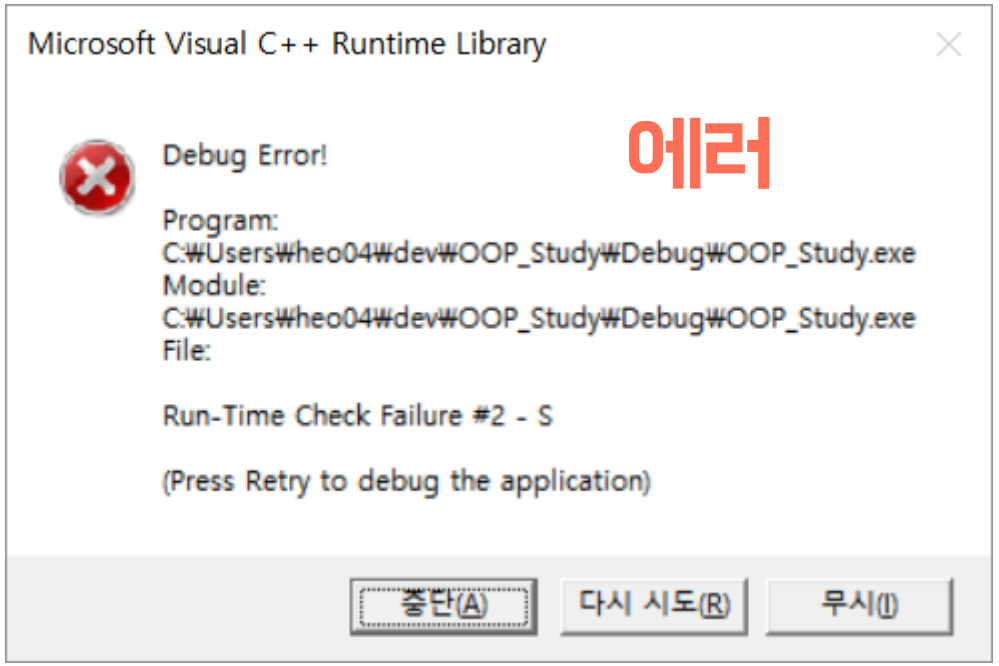
```
int number[5] = { 10,20,30,40,50 };
int sum;
for (int i = 0; i < 5; i++)
{
    sum += number[i];
}
```

# 3 배열

## 0부터 시작한다는 점 항상 주의!

```
int arr3[5];
arr3[5] = 5;
```

=>

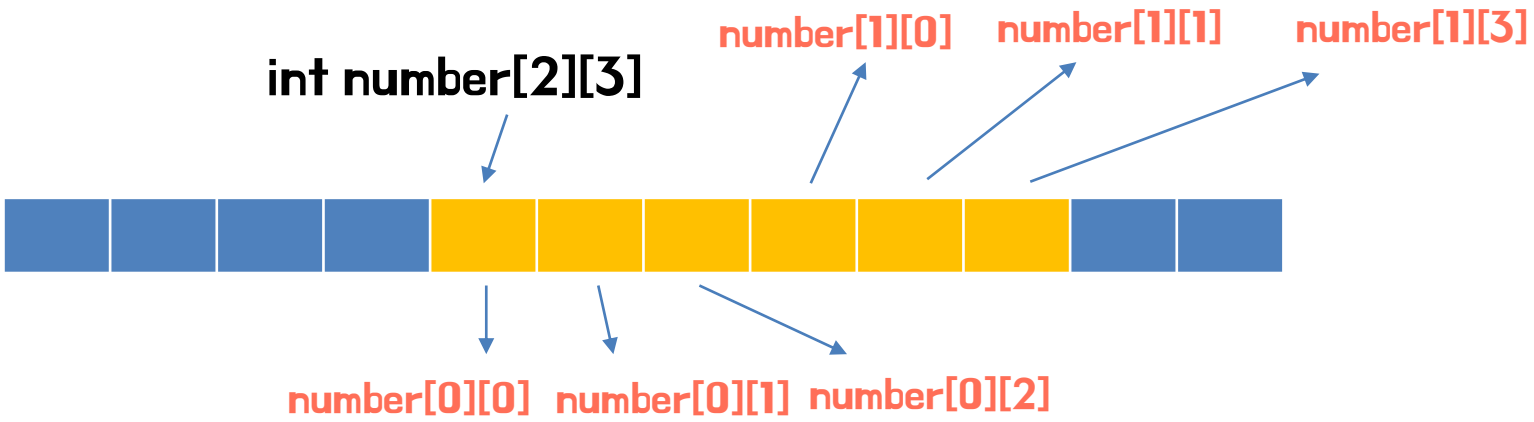


# 2차원 배열

## int number[5][5]

| | | | | |
|---|---|---|---|---|
| number[0][0] | number[0][1] | number[0][2] | number[0][3] | number[0][4] |
| number[1][0] | number[1][1] | number[1][2] | number[1][3] | number[1][4] |
| number[2][0] | number[2][1] | number[2][2] | number[2][3] | number[2][4] |
| number[3][0] | number[3][1] | number[3][2] | number[3][3] | number[3][4] |
| number[4][0] | number[4][1] | number[4][2] | number[4][3] | number[4][4] |

# 3 배열




**연속된 메모리 구조(== int number[6])**

# 2차원 배열의 초기화

```
int score[2][3] = {1, 2, 3, 4, 5, 6};

int score[2][3] = {
        {1, 2, 3},
        {4,5, 6}
};

int score[2][3] = { 0 }; (default)
```

# 4 문자열

## 문자열이란?

char 타입의 배열은 문자열(string)이다.

정확히 말하면,
'/0'로 끝나는 char 타입의 배열은 문자열이다.
(NULL)

# 4 문자열

## 문자열의 선언 및 초기화

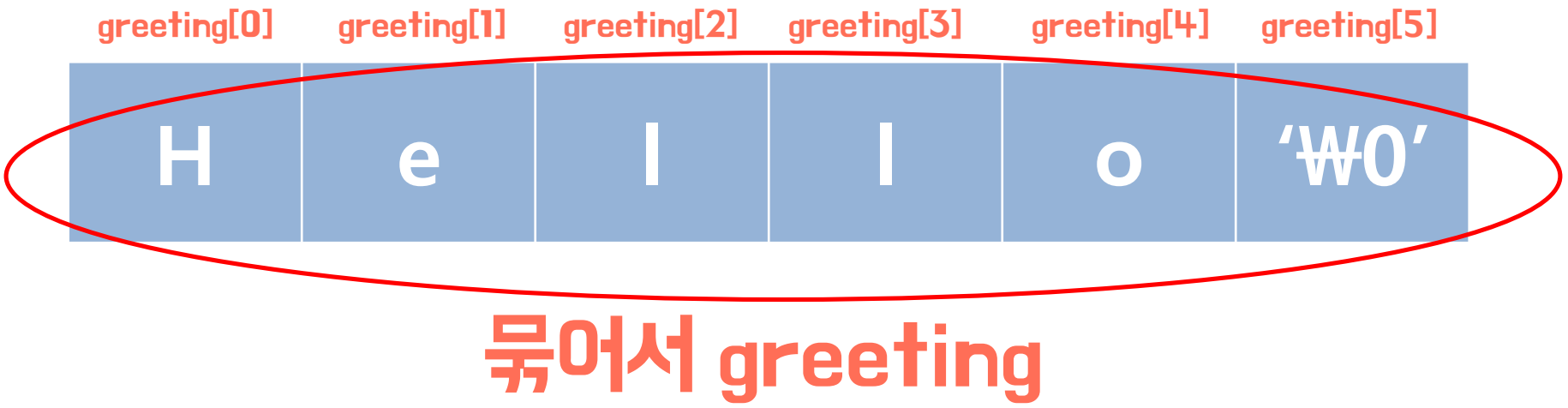
char + 배열이름 + [크기]

```
char greeting[6] = {'H', 'e', 'l', 'l', 'o'};

char greeting[] = "Hello";
```

# 문자열의 접근

```
char greeting[] = “Hello”;
```



# 4 문자열

```
int arr[5] = { 1,2,3,4,5 };
char ch[6] = { 'H','e','l','l','o' };

cout << "arr = " << arr << "\n";
cout << "ch = " << ch << "\n";
```

```
arr = 008FFC50
ch = Hello
계속하려면 아무 키나 누르십시오 . . .
```

**배열의 이름을 출력했을 때,**
**일반적인 배열은 주소값을 출력하지만**
**char 타입의 배열은 원소(문자열)을 출력한다**

# 4 문자열

## 문자열 관련 함수들

strcpy(string copy) ex) strcpy(s1, s2);

=> 문자열 s2를 s1에 복사한다.

strcat(string concatenate) ex) strcat(s1, s2);

=> 문자열 s1의 끝에 s2를 이어붙인다.

strlen(string length) ex) strlen(s1);

=> 문자열 s1의 길이를 알려준다.

# 4 문자열

Q. 3개의 문자열을 입력받고 각각의 문자열 사이에
공백(' ')을 넣고 합친 뒤에 출력하시오.

입력 : Landvibe
C++
Study

출력 예시 : Landvibe C++ Study

strcat => strcat_s

# 감사합니다.

Made by 규정