

객체지향형 프로그래밍과 자료구조

Ch 1. 객체지향형 프로그래밍 개요



정보통신공학과
교수 김 영 탁

(Tel : +82-53-810-2497; Fax : +82-53-810-4742
<http://antl.yu.ac.kr/>; E-mail : ytkim@yu.ac.kr)

목 차

- ◆ 산업혁명과 핵심 기술의 발전, 제4차 산업혁명
- ◆ 대규모 소프트웨어 개발과 소프트웨어 공학
- ◆ 객체 지향 프로그래밍 (Object-Oriented Programming)
- ◆ C++ 프로그램 기본 구조
- ◆ C++ 프로그램의 표준 입출력, 파일 입출력
- ◆ C++ 프로그램에서의 조건문과 제어문
- ◆ C++ 프로그램 통합 개발 환경
 - Visual Studio Community 2019
- ◆ C++ 프로그램 디버깅



인류 문명의 진화

◆ 인류문명의 진화

- 새로운 도구와 재질, 동력장치를 사용하여 **생산성** (*productivities*)을 향상:
 - 석기, 청동기, 철기, 증기(steam) 엔진, 전기, 산업용 로봇, **컴퓨터**, 인터넷, 분산처리, 인공지능 (AI), 빅데이터, 사물인터넷 (IoT)
- 다수 인원의 효율적인 **협동/협력** (*collaborations*)
- 부가가치 창출의 **성능** (*performance*) 개선
- **에너지 효율성** (*energy efficiencies*)



(a) 석기 망치, 창



(b) 철기 농기구



(c) 증기 (steam) 엔진



(d) 전기, 산업용로봇



(e) 컴퓨터, 인터넷, 분산처리



(f) 인공지능 (AI), 빅데이터, 사물인터넷(IoT)



산업혁명과 소프트웨어

◆ 산업혁명

산업혁명	시기	주요 변화
1차	18세기	증기 기관 (steam engine)의 발명으로 사람과 동물의 힘보다 더욱 더 큰 힘을 효율적으로 발생시킬 수 있었으며, 영국에서는 이를 활용하여 섬유생산을 획기적으로 증대시킴
2차	19세기 ~ 20세기 초반	전기 에너지 생산 및 공급 기술이 개발되고, 표준화된 모듈을 컨베이어 벨트에서 조립하는 대량 생산 기술이 개발됨
3차	20세기 후반	컴퓨터와 인터넷 기술의 개발로 과학 기술 분야의 대용량 데이터 처리가 가능하게 되었으며, 서로 다른 지역에 떨어져 있는 사람과 장치간의 협동 작업이 가능하게 되어 기술 발전이 더욱 가속화 됨
4차	2015년 ~ 현재	사물인터넷, 빅데이터, 인공지능, CPS (Cyber Physical System)/메타버스 (metaverse) 기술의 개발로 사람, 사물, 공간을 초연결, 초지능화 시킴

◆ 제4차 산업혁명의 핵심 기술

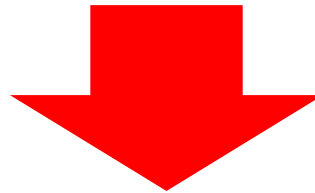
- 사물인터넷 (Internet of Things, IoT)
- 빅데이터 (Big Data)
- 인공지능 (Artificial Intelligence, AI), 기계학습 (Machine Learning)

➔ 모두 큰 규모의 소프트웨어가 핵심이 되는 기반 기술임 !



제4차 산업 혁명의 핵심 기술

Big Data
Cloud Computing
Internet of Things (IoT)
Cyber Physical System (CPS)/Metaverse
Artificial Intelligence (AI)



공통점 ?

큰 규모의 **Software**가 핵심 기반 기술



큰 규모의 소프트웨어 개발에서의
고려사항 – 객체지향형, 자료구조, 알고리즘

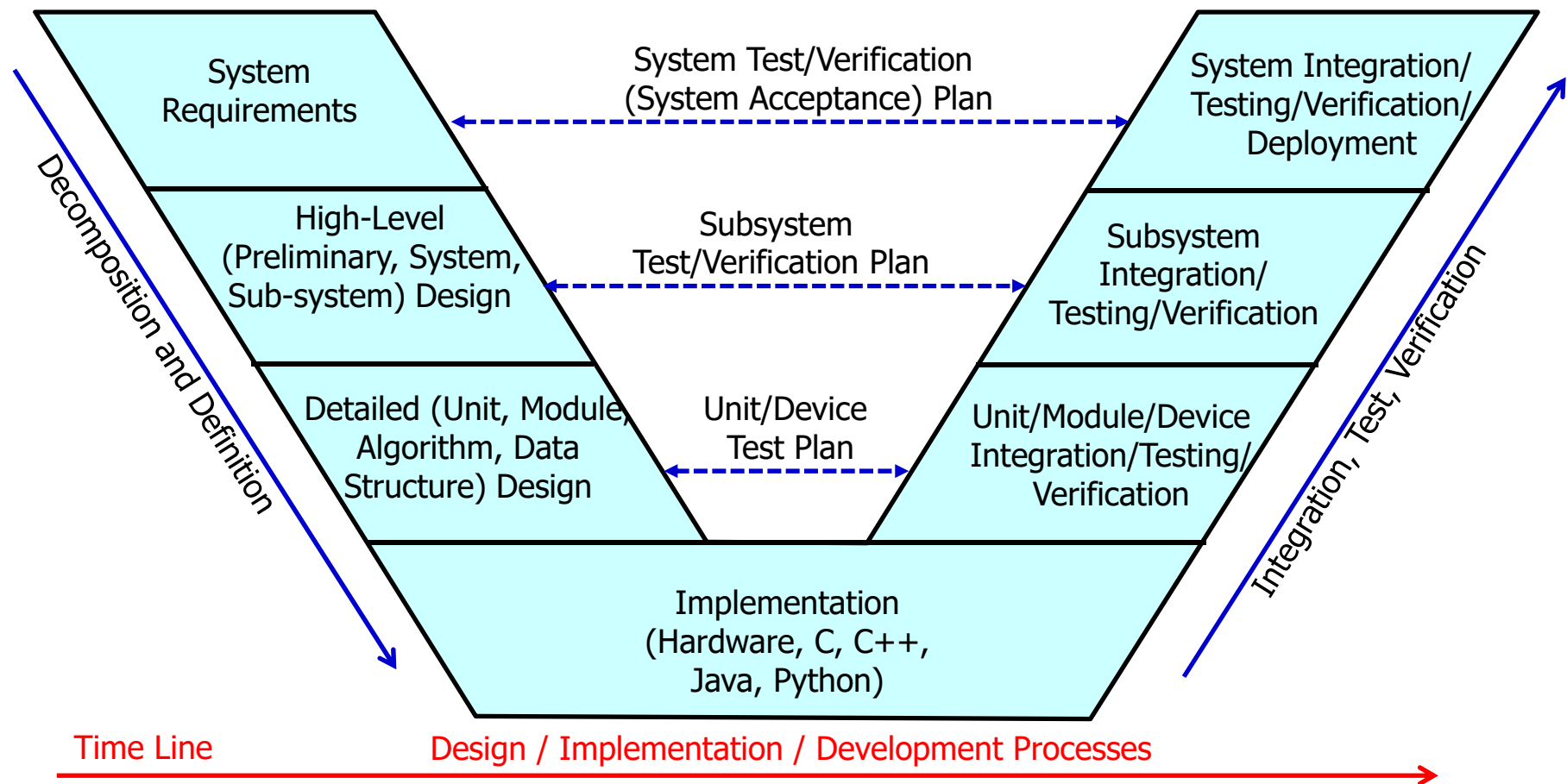
대규모 소프트웨어 개발에서의 중점 고려 사항

중점 고려사항	세부 사항
시스템 안정성 (stability)	시스템이 항상 정상적인 범위/상태에 존재하도록 유지하여야 함.
소프트웨어 재사용성 (reusability)	기존에 개발된 소프트웨어 모듈을 재사용함으로써 쉽게 새로운 시스템 개발이 가능하고, 시스템 개발 기간과 비용의 절감이 필요함.
시스템 기능 확장성과 신기술 도입 용이성 (adaptability to new technology)	새로운 최신 기술이 개발되었을 때 이를 쉽게 도입하고 수용할 수 있는 확장성 필요함.
시스템 유지 보수 및 관리 용이성 (easy maintenance and management)	시스템을 쉽게 유지 보수 및 관리할 수 있는 구조가 필요함.
사용자 친화형 인터페이스와 구현의 분리 (user friendly interface)	사용자가 쉽게 이해하고, 사용할 수 있는 사용자 인터페이스 제공하되, 내부 구현과 분리되어 있도록 하여 구현 방법이 일부 변경되어도 전체 시스템 구조에 문제가 없는 구조를 사용하여야 함.
개발 기간 단축과 개발 비용 절감 (reduced development period and cost)	새로운 기능과 기술 개발 기간을 최소화 할 수 있고, 개발 비용을 줄일 수 있는 구조를 사용하여야 함.



소프트웨어 공학 체계의 개발 절차

◆ 소프트웨어 공학 체계의 대규모 시스템 개발 절차

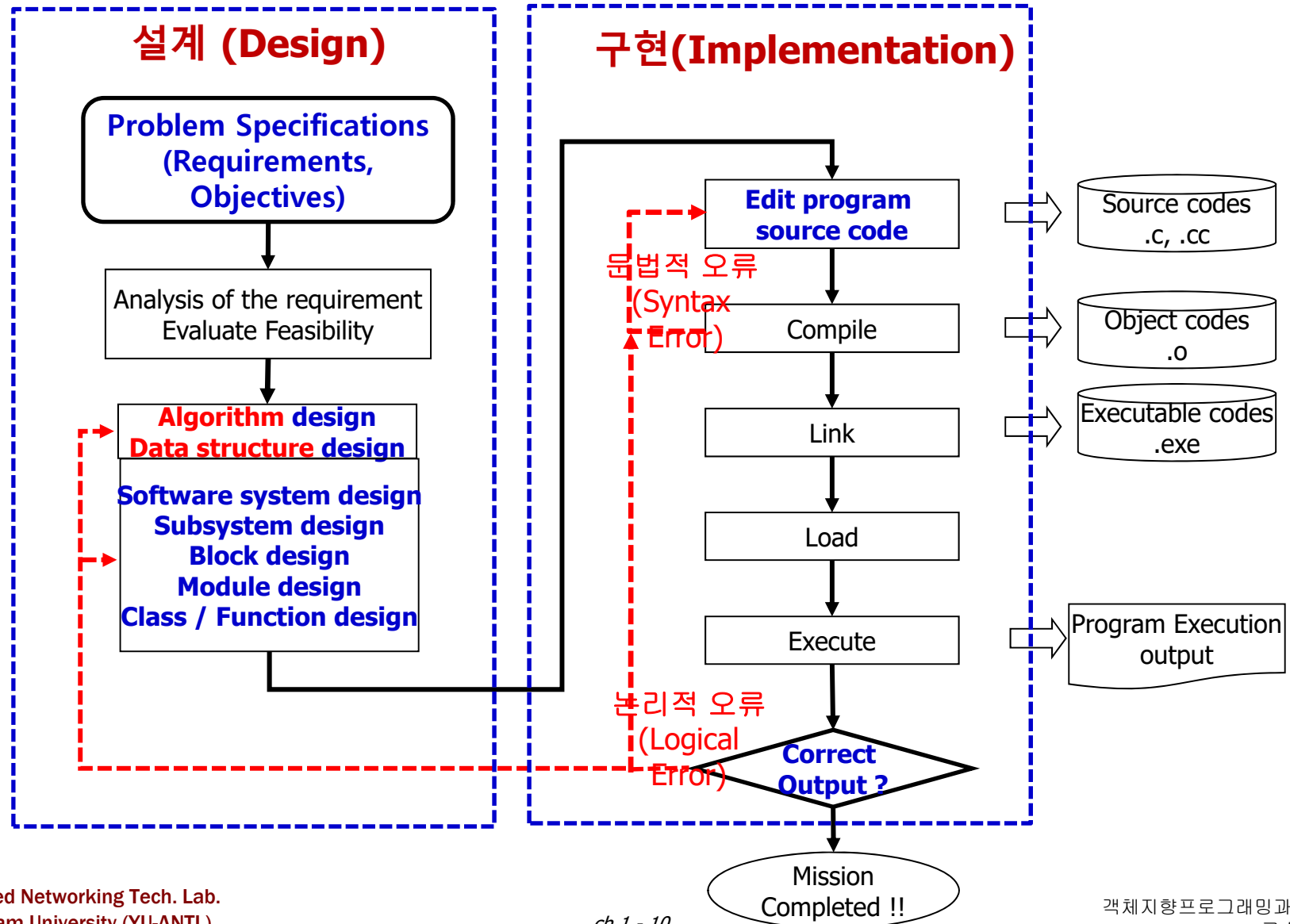


소프트웨어 개발 단계별 주요 업무와 관련 문서

소프트웨어 개발 단계	주요 업무	관련 문서
시스템 요구사항 분석	목표 시스템의 기능 및 성능 요구사항 정리 기능 및 성능의 시험 및 검증 방법 정리	시스템 기능 및 성능 요구사항 문서 시스템 시험 및 검증 계획서
시스템 구조 설계	시스템 통합 기능 구조 설계 시스템 통합 계획 수립 서브시스템 세부 기능 구조 설계	시스템 구조 설계 문서 시스템 통합 계획서 서브 시스템 시험 계획서
시스템 상세 설계	시스템 상세 기능 모듈 설계 모듈별 자료구조 및 알고리즘 설계 모듈별 기능 및 성능 시험 계획 수립	상세 설계 문서 (하드웨어 및 소프트웨어 모듈) 모듈별 시험 계획서
구현	하드웨어 단위 모듈 구현 소프트웨어 단위 모듈 구현 (C, C++, Java, Python, C# 등)	하드웨어 단위 모듈 설명서 소프트웨어 단위 모듈 설명서
단위모듈 시험, 통합 및 검증	단위 모듈 기능 및 성능 시험 실행	단위 모듈 기능 및 성능 시험 결과서
서브시스템 통합 및 시험, 검증	서브시스템 통합 및 시험 실행	서브시스템 시험 결과서
시스템 통합 및 시험, 검증	시스템 통합 및 기능/성능 시험 실행	시스템 통합 및 기능/성능 시험 결과서



프로그램 모듈의 설계 및 구현 절차



자료구조란?

◆ 소프트웨어를 사용하여 정보를 처리하기 위한 방법들

- 기본적인 데이터 유형
 - 정수 (int), 실수 (float, double), 문자 (char)
- 기본적인 데이터의 배열 (array)
 - 배열(array) : 동일한 데이터 유형의 정보 N개를 연속적인 메모리상에 배치하고, index를 사용하여 신속하게 각 항목들을 사용할 수 있게 구성
- 다수의 기본 데이터 유형이 조합되어 복잡한 정보를 표현
 - 구조체 (struct): 다양한 데이터 유형의 속성 정보들을 하나의 구조체에 포함시켜 복잡한 정보를 표현할 수 있도록 구성
 - 클래스 (class): 다양한 데이터 유형의 속성 정보들을 하나의 구조체에 포함시키며, 해당 속성 (attribute) 정보를 처리할 수 있는 함수들을 함께 포함시켜 객체지향형 방식으로 사용할 수 있도록 구성



대표적인 자료 구조

자료구조	특성
단순 배열	컴파일 단계에서 크기가 지정되어 변경되지 않는 크기의 배열
동적 확장성 배열	프로그램 실행 단계에서 크기가 지정되며, 프로그램 실행 중에 크기를 변경할 수 있는 확장성 배열
구조체 배열	구조체로 배열원소가 지정되는 배열
연결형 리스트	확장성이 있으며, 단일연결형 또는 이중연결형으로 구성 가능 다양한 컨테이너 자료구조의 내부적인 자료구조로도 활용됨
스택 (Stack)	Last In First Out (LIFO) 특성을 가짐. 배열 또는 연결형 리스트로 구현할 수 있음.
큐 (Queue)	First In First Out (FIFO) 특성을 가짐. 배열 또는 연결형 리스트로 구현할 수 있음.
우선 순위 큐 (Priority Queue)	컨테이너 내부에 가장 우선순위가 높은 데이터 항목을 추출할 수 있도록 관리하며, 배열 또는 자기 참조 구조체로 구현할 수 있음
이진 탐색 트리 (Binary Search Tree)	컨테이너 내부에 포함된 데이터 항목들을 정렬된 상태로 관리하여야 할 때 매우 효율적임. 단순 이진 탐색 트리의 경우 편중될 수 있으며, 편중된 경우 검색 성능이 저하되기 때문에, 밸런싱이 필요함.
해시 테이블 (Hash Table)	컨테이너 자료구조에 포함된 항목들을 문자열 (string) 또는 긴 숫자를 키 (key)로 사용하여 관리하여야 하는 경우, key로부터 해시 값을 구하고, 이 해시 값을 배열의 인덱스로 사용함.
맵(Map)	key와 항목 간에 1:1 관계가 유지되어야 하는 경우에 사용되며, 해시 테이블을 기반으로 구현할 수 있음
딕셔너리(Dictionary)	key와 항목 간에 1:N 관계가 유지되어야 하는 경우에 사용되며, 해시 테이블을 기반으로 구현할 수 있음
트라이(trie)	텍스트 또는 비트열의 검색을 신속하게 처리할 수 있도록 하며, 동일한 접두어 (prefix)를 사용하는 문장 또는 주소를 신속하게 검색할 수 있게 함. 예측 구문 (predictive test) 제시, longest prefix matching 등의 텍스트 처리 응용 분야에 사용됨.
그래프	정점 (vertex)/노드 (node)로 개체 (object)가 표현되고, 간선 (edge)/링크(link)들을 사용하여 개체 간의 관계를 표현하는 경우에 적합함. 그래프를 기반으로 경로 탐색, 최단거리경로 탐색, 신장트리 (spanning tree) 탐색 등에 활용됨.



알고리즘이란?

◆ 알고리즘

- 주어진 데이터 집합으로부터 요구된 목표 형태의 데이터로 처리, 가공하는 절차와 방법
- 목표 형태의 예:
 - 정렬 (sort): 배열에 포함된 항목들을 지정된 기준에 따라 순서를 정리
 - 탐색 (search): 배열에 포함된 항목 중 가장 큰 값, 가장 작은 값, 또는 지정된 key 값을 가지는 항목을 찾아 그 위치 또는 값을 반환
 - 우선 순위를 고려한 탐색: 탐색 기능에서 지정된 key 값이 우선 순위 (priority) 또는 등급 (class)으로 지정되어, 우선 순위 또는 등급이 가장 높은 항목을 찾아 상위 10명을 반환



대표적인 알고리즘과 주요 기능

알고리즘	주요 기능
순차 탐색	배열이나 연결형리스트에 포함되어 있는 항목을 차례대로 탐색함.
이진 탐색	배열에 포함되어 있는 항목의 탐색 구간을 절반씩 줄여가며 탐색함.
선택 정렬	배열에 포함되어 있는 항목의 탐색 구간을 하나씩 줄여가며, 가장 작은(또는 큰) 항목을 찾아 그 구간의 맨 앞에 두는 동작을 반복하여 전체 항목들을 정렬함.
퀵 정렬	배열에 포함되어 있는 항목의 탐색 구간을 중간 지점의 피벗 (pivot)을 중심으로 큰 항목들과 작은 항목들로 구분하여 정돈하며, (평균적으로) 절반씩 줄어든 탐색 구간에 대하여 재귀 함수 실행함.
해시 (hash)	문자열과 같이 길이가 일정하지 않는 키 (key)를 사용하여 해시 값을 생성하고, 이를 배열의 인덱스나 데이터 항목이 저장되어 있는 그룹(버킷)을 찾는데 사용함.
그래프 깊이우선 탐색 (DFS)	그래프에서 지정된 시작 정점으로부터 목적지 정점까지의 경로를 깊이 우선 방식으로 탐색. 목적지까지의 탐색 시간은 상대적으로 빠를 수 있으나, 탐색된 경로가 최단 거리를 보장하지는 않음.
그래프 넓이우선탐색 (BFS)	그래프에서 지정된 시작 정점으로부터 목적지 정점까지의 경로를 넓이 우선 방식으로 탐색. 탐색된 경로는 경유하는 간선수가 최소인 것을 보장함
최단거리 경로 탐색	간선 (edge)에 가중치 (weight)가 지정된 그래프에서 시작 정점으로부터 목적지 정점까지의 경로 중에서 전체 거리가 가장 짧은 경로를 탐색. 탐색된 경로는 최단거리인 것을 보장함.
최소비용 신장트리 탐색	주어진 그래프에서 사용되는 간선들의 총 거리 (또는 가중치)가 가장 작으면서 모든 노드 (정점)들을 연결할 수 있는 최소비용 신장트리 (minimum spanning tree)를 탐색함.



알고리즘과 자료구조의 관련성

- ◆ 특정 데이터 처리 기능이 가장 효율적으로 구현되기 위해서는 가장 적절한 자료구조와 알고리즘의 조합이 사용되어야 함

데이터 처리 기능 (예)	적합한 자료구조	적합한 알고리즘
선착순 방식의 은행 창구 서비스	Linked List 기반의 Queue	First-In First Out (FIFO) 방식의 enqueue(), dequeue()
우선 순위를 고려한 긴급 업무 처리	Complete Binary Tree 구조의 Heap priority queue	Complete Binary Tree 의 upheap bubbling, downheap bubbling
항상 전체 항목들의 순서를 유지하며, 데이터 탐색	AVL Balanced Binary Tree	AVL Tree 방식의 re-balancing in-order traversal
동일한 단어에 다수의 의미가 주어질 수 있는 동의어 사전 (thesaurus dictionary)	Hash table 기반의 Dictionary	Hashing을 사용한 1차 bucket 탐색과 bucket 내부에서의 선형 탐색을 사용한 Dictionary 탐색
자동차 네비게이션	그래프	Graph기반 shortest path



C++ 프로그램의 기본 구조

C와 C++ 프로그래밍의 비교

구분	C 프로그래밍언어	C++ 프로그래밍언어
기본 구조	<ul style="list-style-type: none"> - 절차적 프로그래밍 - 데이터와 함수를 분리하여 구현 - 데이터를 다수의 함수가 직접 접근하여 처리 가능 	<ul style="list-style-type: none"> - 객체 지향형 프로그래밍 - 데이터와 관련함수를 하나로 캡슐화시켜 구현 - 각 데이터는 관련함수만이 직접 접근할 수 있으며, 다른 함수들은 데이터가 속한 객체에서 공개된 함수를 거쳐 사용할 수 있음
제공 기능 비교	<ul style="list-style-type: none"> - 데이터나 함수의 외부 공개 제한 개념이 없으며, 모든 함수가 포인터를 사용하여 데이터 값을 변경할 수 있으므로 데이터 보호 기능이 없음 	<ul style="list-style-type: none"> - C/C++ 통합 개발 환경에서는 C의 모든 기능을 활용할 수 있으며, 다음과 같은 기능을 C++에서 추가적으로 제공 - 캡슐화 (encapsulation), 추상화(abstraction), 정보 은닉 (information hiding), 데이터 보호 (data protection) - 인터페이스 (interface)와 구현(implementation)을 분리 - 상속 (inheritance) 기능을 제공하여 소프트웨어 재 활용 (software reuse)을 쉽게 구현 - 연산자 오버로딩 (operator overloading) 기능을 제공하여 익숙한 연산자 (operator)를 사용자가 직접 구현한 객체에 사용할 수 있게 함 - 다형성 (polymorphism) 개념을 지원하며, 가상함수와 late binding 기능으로 구현하여 소프트웨어 확장성 제공 - 템플릿 개념을 제공하여 동일한 구성의 자료구조와 알고리즘/함수를 다양한 자료형에 적용할 수 있게 함. Standard Template Library (STL) 제공



C++ 프로그램 기본 구조

```
/** My First C++ Program (1)
 *
 * - A sample C++ program to explain the structure of C++ Program
 * - Programmed by Chul-Soo Kim
 * - Last update: 2020-09-01
 */

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

//#define TEST_WITH_INPUT_DATA
#define MAX_NUM_DATA 100
void getStatistics(int data[], int num_data);

int main(int argc, char argv[])
{
    int num_data, data;
```



```

/** My First C++ Program (2) */

#ifndef TEST_WITH_INPUT_DATA
    num_data = 10;
    int input_data[MAX_NUM_DATA] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
#else
    int input_data[MAX_NUM_DATA] = { 0 };
    cout << "Please input the number of integers (less than 100) = ";
    cin >> num_data;
    cout << "Input " << num_data << " integer data : ";
    for (int i = 0; i < num_data; i++)
    {
        cin >> data;
        input_data[i] = data;
    }
#endif
    getStatistics(input_data, num_data);
    return 0;
}

```



```
/** My First C++ Program (3) */
```

```
void getStatistics(int data_array[], int num_data)
```

```
{  
    int data, min, max;  
    double sum = 0.0, var, diff, sq_diff_sum = 0.0, avg, std_dev;  
    min = INT_MAX;  
    max = INT_MIN;  
    for (int i = 0; i < num_data; i++)  
    {  
        data = data_array[i];  
        if (data < min)  
            min = data;  
        if (data > max)  
            max = data;  
        sum = sum + (double) data;  
    }  
    avg = sum / num_data;  
    sq_diff_sum = 0.0;  
    for (int i = 0; i < num_data; i++)  
    {  
        diff = data_array[i] - avg;  
        sq_diff_sum += diff * diff;  
    }  
}
```



```
/** My First C++ Program (4) */
```

```
var = sq_diff_sum / num_data;  
std_dev = sqrt(var);  
cout << "Total " << num_data << " integer data : {";  
for (int i = 0; i < num_data; i++)  
{  
    cout << data_array[i] << ", ";  
}  
cout << "}" << endl;  
  
cout << "min (" << min << "), max (" << max << "), ";  
cout << "sum(" << sum << "), average(" << avg << ")" << endl;  
cout << "variance (" << var << "), standard deviation (" <<  
    << std_dev << ")" << endl;  
}
```



Basic C/C++ Program Style

◆ Bottom-line:

- Make programs easy to read and modify
- the other programmer (who may update your source code) must easily understand your code

◆ Comments (주석문) - two methods:

- // Two slashes indicate entire line is to be ignored
- /*Delimiters indicates everything between is ignored*/
- Both methods commonly used

◆ Identifier (식별자) naming

- ALL_CAPS for constants: e.g., MAX_NUM
- Lower case character at beginning for variables
- Word separation by camel back (e.g., intString) or under-bar (e.g., int_string)
- Most important: MEANINGFUL NAMES!



Namespaces

◆ Namespaces defined:

- Collection of name definitions
- namespace "std" has all standard library definitions we need

◆ Example (1)

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

- Includes entire standard library of name definitions

◆ Example (2)

```
#include <iostream>  
using std::cin;  
using std::cout;
```

- Can specify just the objects we want individually



표준 객체, 연산자 및 관련 라이브러리

분류	표준 객체, 연산자 및 라이브러리 함수 예	라이브러리
C++ 표준 입출력	cin (console input), cout (console output) <<, >>	<iostream>
C 표준 입출력	scanf(), printf(), getchar(), putchar(), gets(), puts()	<stdio.h>
파일 입출력	fopen(), fclose(), fscanf(), fprintf(), fget()	<stdio.h>
	open(), close(), fail(), eof()	<fstream>
시간	time(), localtime(), asctime()	<time.h>
		<ctime>
난수 생성	rand(), srand()	<cstdlib>
수학 연산	sin(), cos(), sqrt(), pow()	<math.h>
		<cmath>
문자열 (string)	strlen(), strcat(), strcpy(), strcmp()	<string>
시스템 I/O	Beep()	<Windows.h>
자료구조	vector, list, deque	<vector>, <list>, <deque>
알고리즘	sort(), search()	<algorithm>
스레드	CreateThread(), WaitForSingleObject(), TerminateThread(), CloseHandle() _beginthreadex(), _endthreadex() InitializeCriticalSection(), EnterCriticalSection(), LeaveCriticalSection(), DeleteCriticalSection()	<Windows.h> <process.h>
인터넷 통신	socket()	<sys/socket.h>



C/C++ 기본 자료형

기본 자료형		설 명	비트 수	값의 범위
char (문자형)	(signed) char	문자 및 정수	8	-128 ~ 127
	unsigned char	문자 및 부호 없는 정수	8	0 ~ 255
int (정수형)	(signed) short	short형 정수	16	-32768 ~ 32767
	unsigned short	부호 없는 short형 정수	16	0 ~ 65535
	(signed) int	정수 (integer)	32	-2147483648 ~ 2147483647
	unsigned int	부호 없는 정수	32	0 ~ 4294967295
	(signed) long	long형 정수	32	-2147483648 ~ 2147483647
	unsigned long	부호 없는 long형 정수	32	0 ~ 4294967295
floating point (부동소수점)	float	단일 정밀도 부동소수점	32	1.2E-38 ~ 3.4E38
	double	두 배 정밀도 부동소수점	64	2.2E-308 ~ 1.8E308
	long double	두 배 정밀도 부동소수점	64	2.2E-308 ~ 1.8E308



C/C++ 기본 연산자

연산자의 분류	기본 연산자	의미
대입 (assignment)	=	오른쪽을 왼쪽에 대입
산술 (arithmetic)	+ - * / %	사칙연산과 나머지 연산
부호 (sign)	+ -	
증감 (increment/ decrement)	++ --	증가, 감소 연산
관계 (relationship)	> < == != >= <=	오른쪽과 왼쪽을 비교
논리 (logical)	&& !	논리적인 AND, OR
조건 (conditional)	?	조건에 따라 선택
콤마 (comma)	,	피연산자들을 순차적으로 실행
비트 단위 연산자 (bit-wise operation)	& ^ ~ <<>>	비트별AND, OR, XOR, 이동, 반전
sizeof 연산자	sizeof	자료형이나 변수의 크기를 바이트 단위로 반환
형 변환 (type conversion)	(type)	변수나 상수의 자료형을 변환
포인터 연산자 (pointer)	* &	주소계산, 포인터가 가리키는 곳의 내용 추출
배열 인덱스 연산자	[]	배열의 개별 항목을 인덱스를 사용하여 사용
구조체 연산자 (struct)	. ->	구조체의 멤버 참조



C/C++ 전처리기 (preprocessor)

전처리기 지시어 (예)	의미
#include <stdio.h> #include "MyHeaderFile.h"	지정된 파일 (헤더파일)을 포함.
#define PI 3.141592 #define SQUARE(x) ((x) * (x)) #define MAX(x, y) ((x) >= (y)) ? (x) : (y))	기호 상수의 값을 지정. 매크로 함수 지정.
#undef	이전에 정의되었던 매크로 정의를 해제.
#if	지정된 조건이 참일 경우 #else나 #endif 까지의 구간을 실행.
#else	#if에서 지정된 조건이 참이 아닐 경우 #endif 까지의 구간을 실행.
#endif	#if, #else, #ifdef, #ifndef, #elif 등의 전처리 지시어 조건의 구역의 끝을 지정.
#ifdef DEBUG	해당 기호상수가 지정되어 있으면, #endif가 나타날 때까지의 구간을 실행.
#ifndef	해당 기호상수가 지정되어 있지 않으면, #endif가 나타날때까지의 구간을 실행.
#line	행 번호를 출력.
#elif	else-if를 의미 함.
#pragma	시스템에 따라 다른 의미를 부여. #pragma once : 이 파일을 한번만 포함시키도록 지정.



limits.h 헤더파일에서 정의하는 기호상수

기호상수	값	의미
CHAR_BIT	8	문자의 비트수 (비트단위로 나누어지지 않는 최소 크기)
SCHAR_MIN	-128	부호있는 문자형의 최소값
SCHAR_MAX	127	부호있는 문자형의 최대값
UCHAR_MAX	255 (0xFF)	부호없는 문자형의 최대값
CHAR_MIN	-128 0 if /j option used	(부호있는) 문자형의 최소값; /j 옵션이 사용되면 0
CHAR_MAX	127 255 if /j option used	(부호있는) 문자형의 최대값; /j 옵션이 사용되면 255
SHRT_MIN	-32768	부호있는 short 형의 최소값
SHRT_MAX	32767	부호있는 short 형의 최대값
USHRT_MAX	65535 (0xFFFF)	부호없는 short 형의 최대값
INT_MIN	-2147483648	부호있는 정수 (int) 형의 최소값
INT_MAX	2147483647	부호있는 정수 (int) 형의 최대값
UINT_MAX	4294967295 (0xFFFFFFFF)	부호없는 정수 (unsigned int) 형의 최대값
LONG_MIN	-2147483648	(부호있는) Long 형 정수의 최소값
LONG_MAX	2147483647	(부호있는) Long 형 정수의 최대값
ULONG_MAX	4294967295 (0xFFFFFFFF)	부호없는 Long 형 정수의 최대값
_I64_MIN	-9223372036854775808	__int64형 (64비트) 정수의 최소값
_I64_MAX	9223372036854775807	__int64형 (64비트) 정수의 최대값
_UI64_MAX	18446744073709551615 (0xFFFFFFFFFFFFFFFF)	부호없는 __int64형 (64비트) 정수의 최대값

기호상수 (Symbolic Constant) 및 함수 매크로 지정

단순 매크로 예

```
#define PI 3.14159265359
#define NUM_STUDENTS 20 // 기호 상수 정의
#define EOF -1 // end of file
#define EPSILLON 1.0e-9
```

함수 매크로 예

```
#define MAX(x, y) ((x) > (y)) ? (x) : (y)
#define MIN(x, y) ((x) < (y)) ? (x) : (y)
#define SQUARE(x) ((x) * (x))
#define CUBIC(x) ((x) * (x) * (x))
```

함수 매크로 예

```
#define GET_BIT(w, k) ((w) >> (k)) & 0x01
#define SET_BIT_ON(w, k) ((w) |= (0x01 << (k)))
#define SET_BIT_OFF(w, k) ((w) &= ~(0x01 << (k)))
```



내장 매크로

◆ 내장 매크로: 미리 정의된 매크로

내장 매크로	설명
__DATE__	이 매크로를 만나면 현재의 날짜 (월 일 년)로 치환된다.
__TIME__	이 매크로를 만나면 현재의 시간 (시:분:초)으로 치환된다.
__LINE__	이 매크로를 만나면 소스 파일에서의 현재의 줄 번호 로 치환된다.
__FILE__	이 매크로를 만나면 소스 파일 이름 으로 치환된다.

- `printf("컴파일 날짜 = %s\n", __DATE__);`
- `printf("치명적 에러 발생 파일 이름 = %s, 줄 번호 = %d\n", __FILE__, __LINE__);`



**C++ 프로그램의 변수/식별자,
기본 입출력, 파일 입출력, 조건문/제어문**

C/C++ 변수 (Variable)

◆ C++ Identifiers

- Keywords/reserved words vs. Identifiers
- Case-sensitivity and validity of identifiers
- Meaningful names!

◆ Variables

- A memory location to store data for a program
- Must declare all data before use in program



C/C++ 프로그램의 키워드

and	and_eq	asm	auto	bitand	bitor
bool	break	case	char	cin	class
const	class	compl	const_cast	continue	cout
default	delete	dynamic_cast	do	double	else
enum	extern	explicit	export	false	friend
float	for	goto	if	inline	int
long	mutable	namespace	new	not	not_eq
operator	or	or_eq	private	protected	public
reinterpret_cast	register	return	short	signed	sizeof
static	static_cast	struct	switch	template	this
throw	true	try	typeid	typename	typedef
union	using	unsigned	virtual	void	volatile
wchar_t	while	xor	xor_eq		



C/C++ 프로그램의 식별자 (identifier) 지정

식별자 예	가능 / 불가능	설 명
average, avg, avg1	사용 가능	문자와 숫자
num_data, numData,	사용 가능	문자, 밑줄
ch, ch1, ch_1	사용 가능	문자, 밑줄
i, d, f	사용 가능	
_avg, __area	사용 가능	밑줄, 문자
NUM_DATA	사용 가능	대문자, 밑줄
auto, char, double, struct, while	사용 불가능	keyword
3avg	사용 불가능	숫자로 시작
avg!, avg%, &avg, dollor\$	사용 불가능	특수문자 포함



C++ 표준 입출력

◆ C++ 표준 입출력 관련 객체

표준 객체	관련 연산자	의미	관련 라이브러리
cin	>>	console input	<iostream>
cout	<<	console output	<iostream>
cerr	<<	console error	<iostream>



C++ 표준 입출력

◆ 출력 포맷 지정

포맷 지정 함수	의미
setf()	지정된 객체로의 출력에 포맷을 지정하며, 출력 포맷 지정자는 표 1.19의 항목들을 사용함.
unsetf()	지정된 출력 포맷을 해제함.
precision()	실수의 출력에서 소숫점 이하 자리 수를 지정함.
width()	출력 칸수를 지정
setfill('0')	확보된 출력 칸수에서 빈칸을 지정된 문자로 채움
포맷 지정자	의미
ios::showpoint	소숫점을 표시
ios::showbase	정수의 출력형식 표시: 10진수에는 별도 표시 없음, 8진수는 0 16진수는 0X를 표시
ios::showpos	양수 (plus)인 경우 + 표시
ios::fixed	고정 소숫점 표시
ios::left	왼쪽 줄 맞춤
ios::right	오른쪽 줄 맞춤
std::dec	다음 숫자를 10진수로 출력
std::oct	다음 숫자를 8진수로 출력
std::hex	다음 숫자를 16진수로 출력
std::uppercase	대문자로 출력
std::lowercase	소문자로 출력
std::internal	자릿수를 확보하여 출력하는 경우, 확보한 자리의 우측에 지정된 문자 출력 숫자인 경우, 오른쪽 정렬로 출력하며, 왼쪽 빈 공간에 지정된 문자 출력; setfill() 함수와 함께 사용
bitset<n>	지정된 비트크기로 2진수 출력. <bitset> 라이브러리가 필요함. 예) cout << bitset<32>(i) << endl;은 정수형 데이터 i를 32비트 2진수로 출력



Escape Sequence

제어 문자 이름	제어 문자 표기	값	의미
NULL 문자	\0	0	문자열의 끝을 표시
alarm(bell)	\a	7	"삐"하는 경고 벨소리 발생
백스페이스(backspace)	\b	8	커서를 현재의 위치에서 한 글자 뒤로 옮긴다.
수평탭(horizontal tab)	\t	9	커서의 위치를 현재 라인에서 설정된 다음 탭 위치로 옮긴다.
줄바꿈(newline)	\n	10	커서를 다음 라인의 시작 위치로 옮긴다.
수직탭(vertical tab)	\v	11	설정되어 있는 다음 수직 탭 위치로 커서를 이동
폼피드(form feed)	\f	12	주로 프린터에서 강제로 다음 페이지로 넘길 때 사용된다.
캐리지 리턴(carriage return)	\r	13	커서를 현재 라인의 시작 위치로 옮긴다.
큰따옴표 (double quotation mark)	\"	34	원래의 큰따옴표 자체
작은따옴표 (single quotation mark)	\'	39	원래의 작은따옴표 자체
역슬래시(back slash)	\\	92	원래의 역슬래시 자체



Console Input/Output

- ◆ I/O objects `cin`, `cout`, `cerr`
- ◆ Defined in the C++ library called `<iostream>`
- ◆ Must have these lines (called pre-processor directives) near start of file:
 - `#include <iostream>`
`using namespace std;`
 - Tells C++ to use appropriate library so we can use the I/O objects `cin`, `cout`, `cerr`



Formatting Numbers

◆ "Magic Formula" to force decimal sizes:

```
cout.setf(ios::fixed);  
cout.setf(ios::showpoint);  
cout.precision(2);
```

◆ These statements force all future cout'ed values:

- To have exactly two digits after the decimal place
- Example:

```
cout << "The price is $" << price << endl;
```

- Now results in the following:
The price is \$78.50

◆ Can modify precision "as you go" as well!



cout 출력 포맷 지정 시험 프로그램

```
/** SimpleTest.cpp - Testing cout formatting (1) */
```

```
#include <iostream>
```

```
#include <iomanip>
```

```
using namespace std;
```

```
#define NUM_DATA 5
```

```
#define NUM_WIDTH 15
```

```
#define NUM_PRECISION 5
```

```
void main()
```

```
{
```

```
    double dd[NUM_DATA] = { 0.01, 123.45, 23456, 0.00123, 9876543210.12 };
```

```
    cout << endl << " Double data output without formatting:" << endl;
```

```
    for (int i = 0; i<NUM_DATA; i++)
```

```
    {
```

```
        cout << " ";
```

```
        cout << dd[i] << endl;
```

```
    }
```

```
    cout << endl;
```




```
/** SimpleTest.cpp - Testing cout formatting (2) */
```

```
cout << " Double data output with formatting (cout.width(15); cout.precision(5))" << endl;
for (int i = 0; i<NUM_DATA; i++)
{
    cout << " ";
    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.width(15);
    cout.precision(5);
    cout << dd[i] << endl;
}
cout << endl;
cout << " Double data output with formatting (cout.width(8); cout.precision(2))" << endl;
for (int i = 0; i<NUM_DATA; i++)
{
    cout << " ";
    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.width(8);
    cout.precision(2);
    cout << dd[i] << endl;
}
cout << endl;
}
```



◆ 실행 결과

Double data output without formatting:

```
0.01  
123.45  
23456  
0.00123  
9.87654e+009
```

Double data output with formatting (cout.width(15); cout.precision(5))

```
0.01000  
123.45000  
23456.00000  
0.00123  
9876543210.12000
```

Double data output with formatting (cout.width(8); cout.precision(2))

```
0.01  
123.45  
23456.00  
0.00  
9876543210.12
```

계속하려면 아무 키나 누르십시오 . . .



C++에서의 16진수 출력

```
void test_hexadecimal_formats()
{
    cout << "Testing hexadecimal output formatings ..." << endl;
    int n = 0;
    for (int i = 0; i < 16; i++)
    {
        for (int j = 0; j < 16; j++)
        {
            cout << "0x" << setw(2) << hex << internal << setfill('0')
                << uppercase << n++ << " ";
        }
        cout << endl;
    }
    cout << endl;
}
```

```
Testing hexadecimal output formatings ...
0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F
0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1A 0x1B 0x1C 0x1D 0x1E 0x1F
0x20 0x21 0x22 0x23 0x24 0x25 0x26 0x27 0x28 0x29 0x2A 0x2B 0x2C 0x2D 0x2E 0x2F
0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x3A 0x3B 0x3C 0x3D 0x3E 0x3F
0x40 0x41 0x42 0x43 0x44 0x45 0x46 0x47 0x48 0x49 0x4A 0x4B 0x4C 0x4D 0x4E 0x4F
0x50 0x51 0x52 0x53 0x54 0x55 0x56 0x57 0x58 0x59 0x5A 0x5B 0x5C 0x5D 0x5E 0x5F
0x60 0x61 0x62 0x63 0x64 0x65 0x66 0x67 0x68 0x69 0x6A 0x6B 0x6C 0x6D 0x6E 0x6F
0x70 0x71 0x72 0x73 0x74 0x75 0x76 0x77 0x78 0x79 0x7A 0x7B 0x7C 0x7D 0x7E 0x7F
0x80 0x81 0x82 0x83 0x84 0x85 0x86 0x87 0x88 0x89 0x8A 0x8B 0x8C 0x8D 0x8E 0x8F
0x90 0x91 0x92 0x93 0x94 0x95 0x96 0x97 0x98 0x99 0x9A 0x9B 0x9C 0x9D 0x9E 0x9F
0xA0 0xA1 0xA2 0xA3 0xA4 0xA5 0xA6 0xA7 0xA8 0xA9 0xAA 0xAB 0xAC 0xAD 0xAE 0xAF
0xB0 0xB1 0xB2 0xB3 0xB4 0xB5 0xB6 0xB7 0xB8 0xB9 0xBA 0xBB 0xBC 0xBD 0xBE 0xBF
0xC0 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7 0xC8 0xC9 0xCA 0xCB 0xCC 0xCD 0xCE 0xCF
0xD0 0xD1 0xD2 0xD3 0xD4 0xD5 0xD6 0xD7 0xD8 0xD9 0xDA 0xDB 0xDC 0xDD 0xDE 0xDF
0xE0 0xE1 0xE2 0xE3 0xE4 0xE5 0xE6 0xE7 0xE8 0xE9 0xEA 0xEB 0xEC 0xED 0xEE 0xEF
0xF0 0xF1 0xF2 0xF3 0xF4 0xF5 0xF6 0xF7 0xF8 0xF9 0xFA 0xFB 0xFC 0xFD 0xFE 0xFF
```



Error Output

◆ Output with cerr

- cerr works same as cout
- Provides mechanism for distinguishing between regular output and error output

◆ Re-direct output streams

- Most systems allow cout and cerr to be "redirected" to other devices
 - e.g., line printer, output file, error console, etc.



C++ 파일 입출력

```
/* main() for file input/output basics (1) */  
#include <fstream>  
#include <iostream>  
#include <iomanip>  
#include "Planet.h"  
using namespace std;  
#define MAX_NUM_DATA 100  
#define INPUT_FILE_NAME "Solar_Planets.txt"  
#define OUTPUT_FILE_NAME "output_result.txt"  
  
typedef struct Planet {  
    char name[16];  
    double relativeMass;  
    double distance;  
} Planet;
```



```
/* main() for file input/output basics (2) */
```

```
void fileInputOutput(ifstream& fin, ofstream& fout, int precision, int fieldWidth);
```

```
void main()
```

```
{  
    ifstream fin;  
    ofstream fout;  
    int precision, width;  
    fin.open(INPUT_FILE_NAME);  
    if (fin.fail())  
    {  
        cout << "Fail to open an input file (" << INPUT_FILE_NAME << ") \n";  
        exit(1);  
    }  
    fout.open(OUTPUT_FILE_NAME);  
    if (fout.fail())  
    {  
        cout << "Fail to open an output file (" << OUTPUT_FILE_NAME << ") \n";  
        exit(1);  
    }  
}
```



```

precision = 2;
width = 10;
fileInputOutput(fin, fout, precision, width);
fin.close();
fout.close();
}
void fileInputOutput(istream& fin, ostream& fout, int prec, int fieldWidth)
{
    fout.setf(ios::fixed);
    fout.setf(ios::showpoint);
    fout.setf(ios::showpos); // print '+' sign before a positive number
    fout.precision(prec);
    Planet *pPlnt;
    pPlnt = new Planet;
    while (!fin.eof())
    {
        fin >> pPlnt->name >> pPlnt->relativeMass >> pPlnt->distance;
        cout << pPlnt->name << " " << pPlnt->relativeMass << " ";
        cout << pPlnt->distance << " " << endl;
    }
}

```



```
fout.unsetf(ios::right);  
fout << setw(fieldWidth);  
fout.setf(ios::left);  
fout << pPlnt->name;  
fout << setw(fieldWidth);  
fout.setf(ios::right);  
fout << pPlnt->relativeMass;  
fout << setw(fieldWidth) << pPlnt->distance;  
fout << endl;  
} // end of while-loop  
delete pPlnt;  
}
```



◆ 입력 데이터 파일 (Solar_Planets.txt)

```
Mercury 0.0558 57.9
Venus 0.815 108
Earth 1.0 150
Mars 0.107 228
Jupiter 318 778
Saturn 95.1 1430
Uranus 14.5 2870
Neptune 17.2 4500
Pluto 0.11 5900
```

◆ 출력 결과

```
Mercury 0.0558 57.9
Venus 0.815 108
Earth 1 150
Mars 0.107 228
Jupiter 318 778
Saturn 95.1 1430
Uranus 14.5 2870
Neptune 17.2 4500
Pluto 0.11 5900
계속하려면 아무 키나 누르십시오 . . .
```

(a) 화면 출력

Mercury	+0.06	+57.90
Venus	+0.81	+108.00
Earth	+1.00	+150.00
Mars	+0.11	+228.00
Jupiter	+318.00	+778.00
Saturn	+95.10	+1430.00
Uranus	+14.50	+2870.00
Neptune	+17.20	+4500.00
Pluto	+0.11	+5900.00

(b) 파일 출력



C++ 프로그램에서의 조건문과 제어문

◆ 조건문

- if, if-else
- 중첩된 if-else
- switch-case

◆ 반복문

- while
- do-while
- for-loop
- break, continue

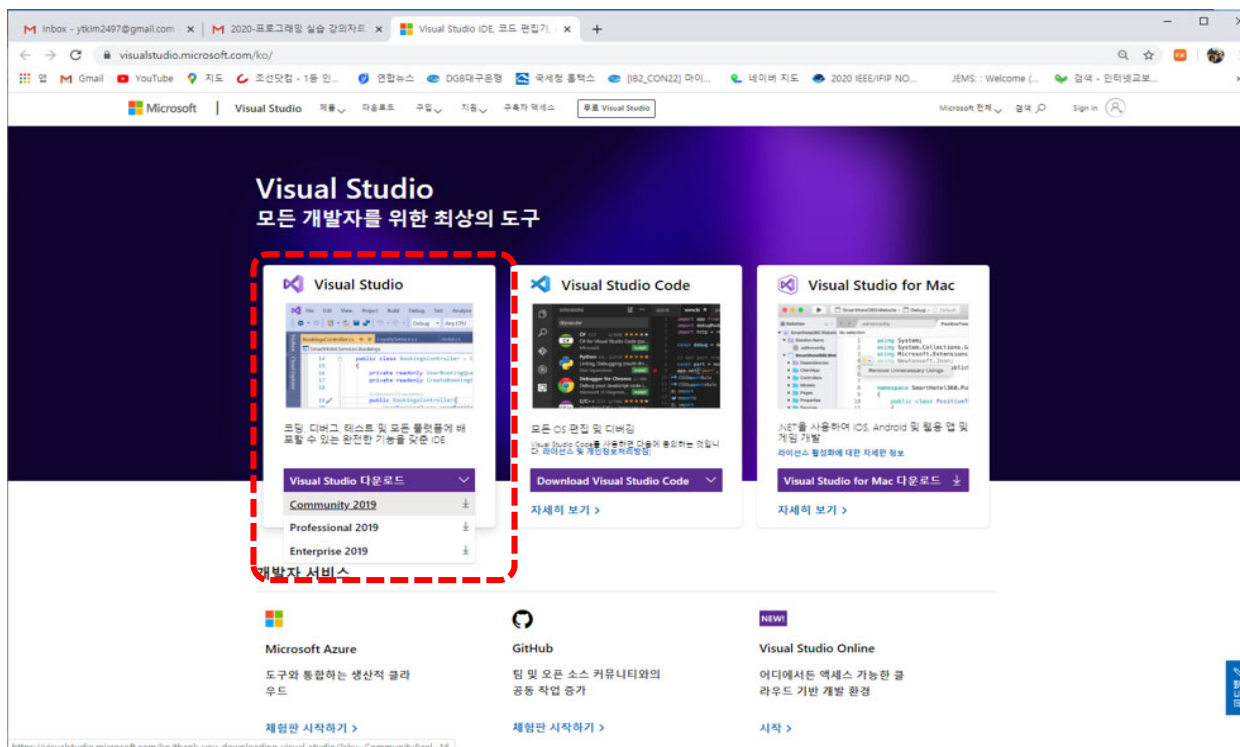


Visual Studio Community 2019

Visual Studio 소개

◆ Visual Studio 란?

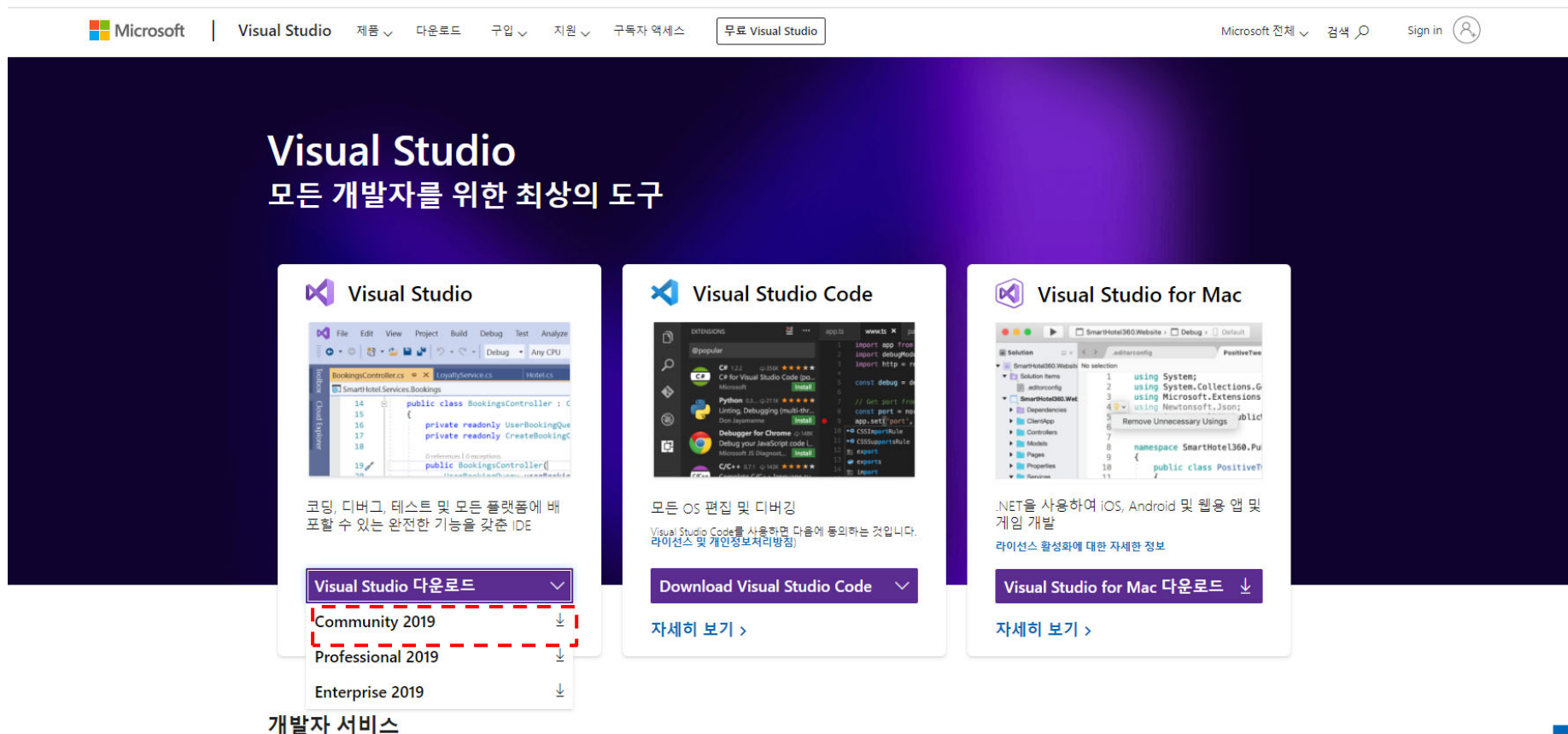
- Microsoft window 환경에서 다양한 언어를 프로그래밍할 수 있는 통합 개발환경
- C, C++, MFC GUI 등 다양한 프로그래밍 언어 지원
- <https://visualstudio.microsoft.com/ko/>



Visual Studio 설치하기(1)

◆ Visual Studio Community 2019 다운로드

- <https://visualstudio.microsoft.com/ko/>
- Visual Studio 다운로드 -> Community 2019 선택



Microsoft | Visual Studio 제품 다운로드 구입 지원 구독자 액세스 무료 Visual Studio Microsoft 전체 검색 Sign in

Visual Studio

모든 개발자를 위한 최상의 도구

Visual Studio

코딩, 디버그, 테스트 및 모든 플랫폼에 배포할 수 있는 완전한 기능을 갖춘 IDE

Visual Studio 다운로드

- Community 2019
- Professional 2019
- Enterprise 2019

Visual Studio Code

모든 OS 편집 및 디버깅

Visual Studio Code를 사용하면 다음에 동의하는 것입니다. (라이선스 및 개인정보처리방침)

Download Visual Studio Code

자세히 보기 >

Visual Studio for Mac

.NET을 사용하여 iOS, Android 및 웹용 앱 및 게임 개발

라이선스 활성화에 대한 자세한 정보

Visual Studio for Mac 다운로드

자세히 보기 >

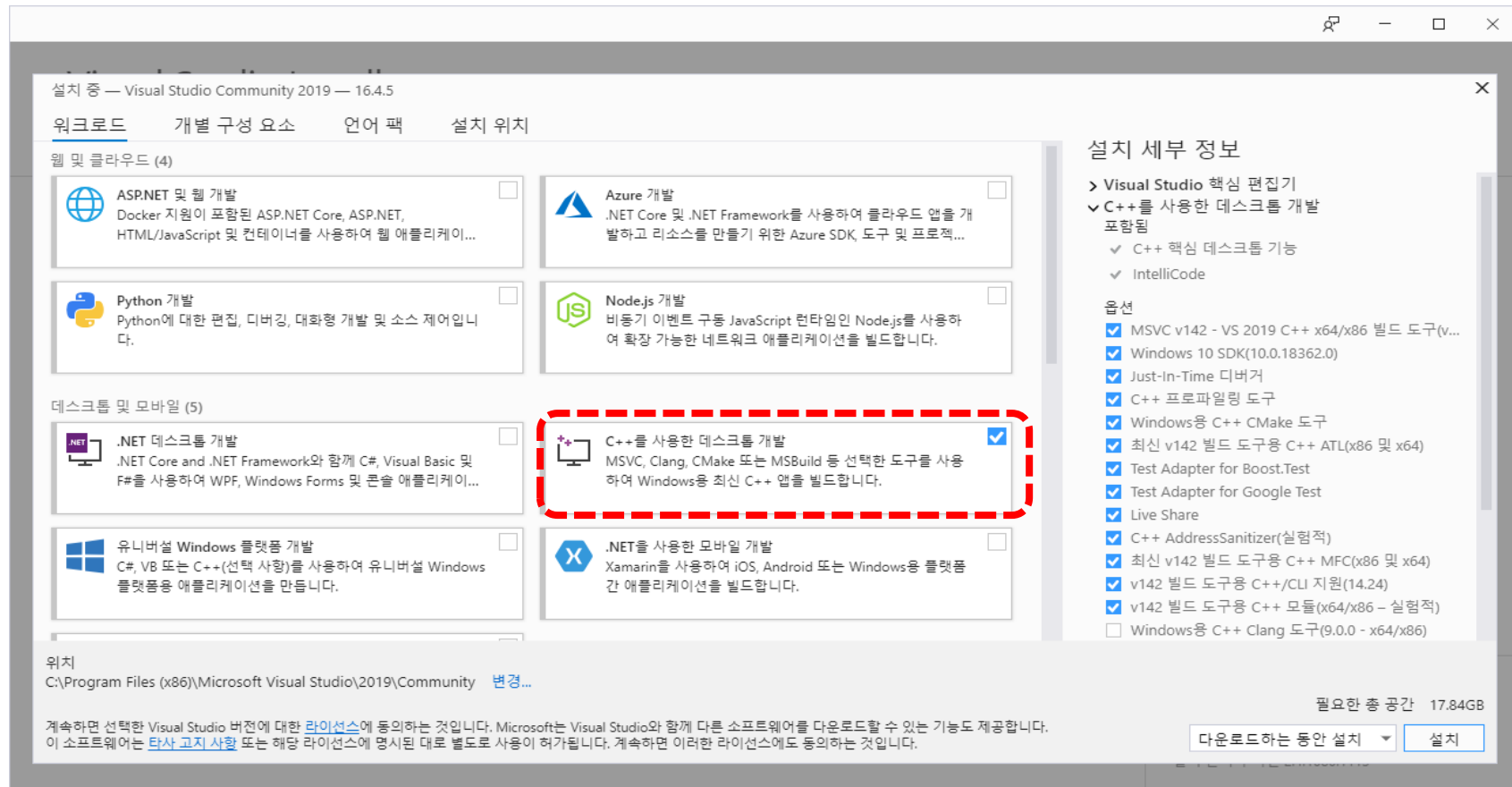
개발자 서비스



Visual Studio 설치하기(2)

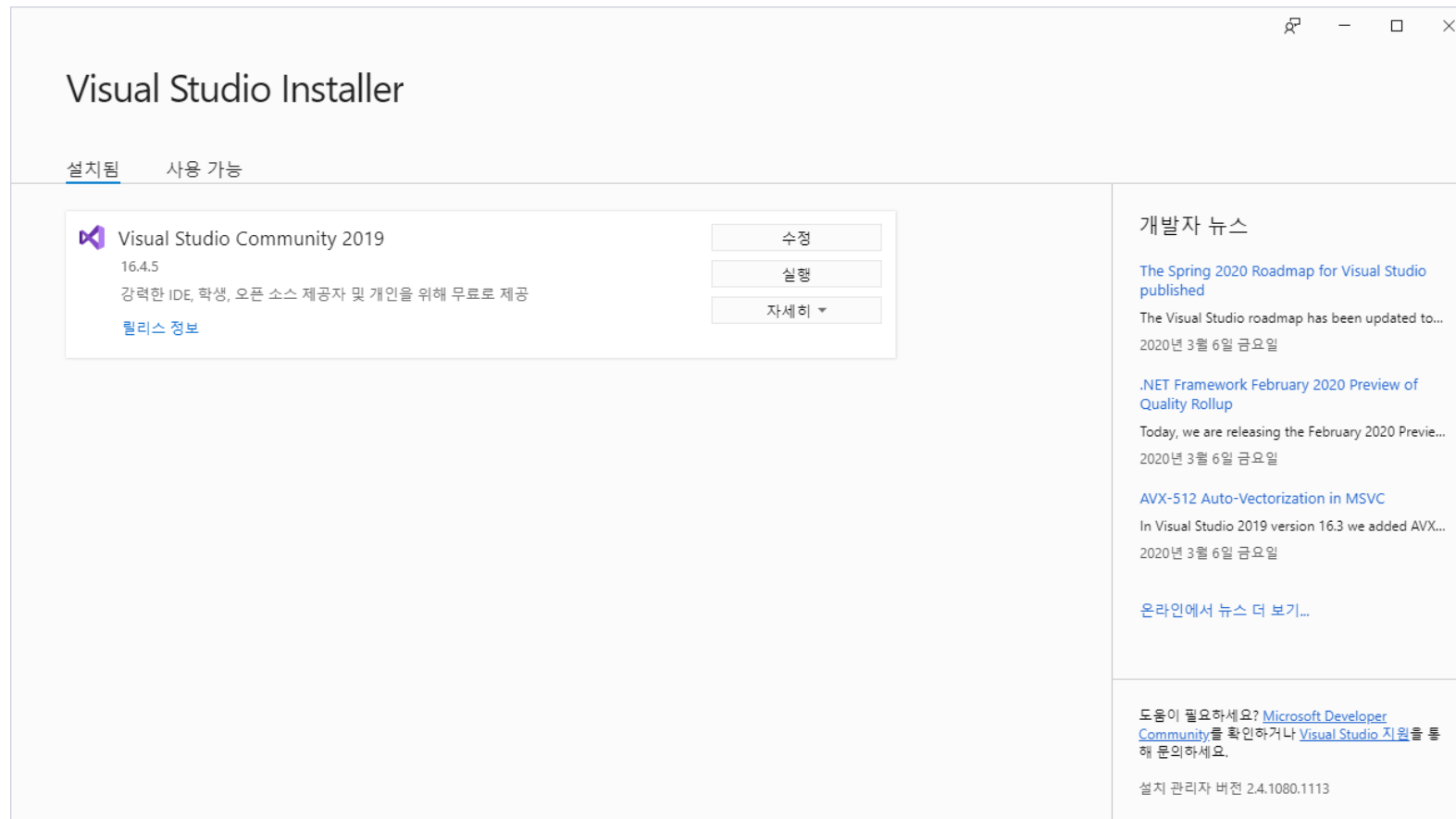
◆ Visual Studio Installer 설치 및 실행

- C++를 사용한 데스크톱 개발 워크로드 체크
 - Hard Disk 용량 20GB 확보 후 설치



Visual Studio 설치하기(3)

◆ 설치완료



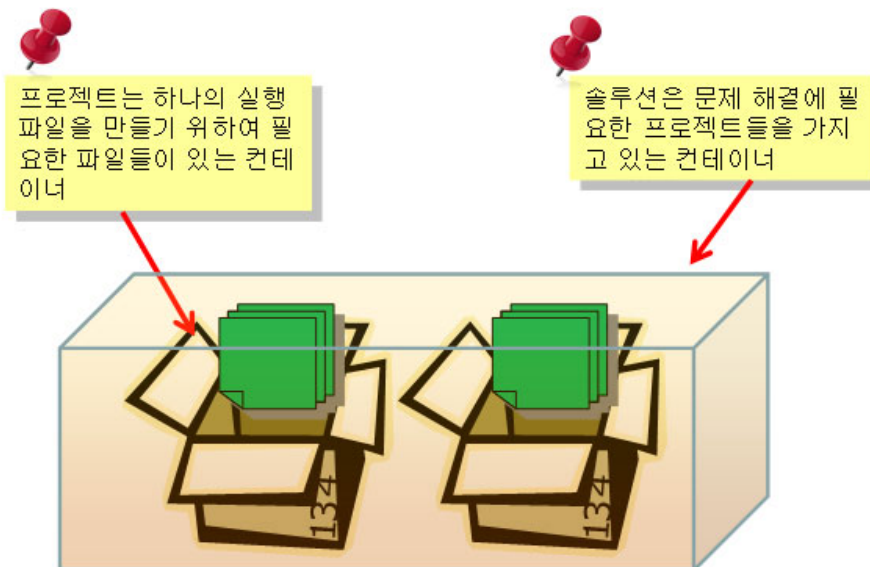
솔루션과 프로젝트

◆ 솔루션 (Solution)

- 문제 해결에 필요한 프로젝트가 들어있는 컨테이너

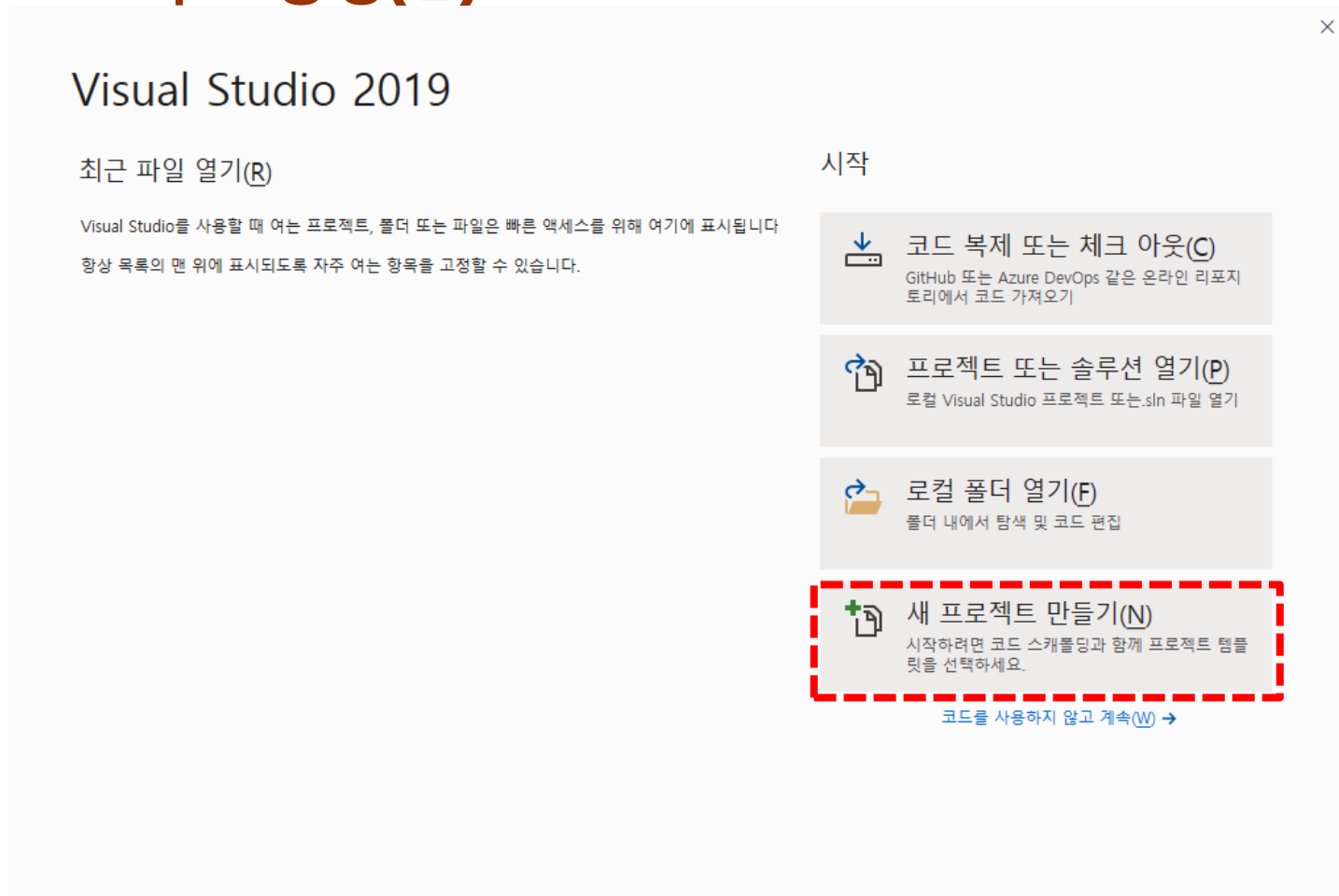
◆ 프로젝트 (Project)

- 하나의 실행 파일을 만드는데 필요한 여러 가지 항목들이 들어 있는 컨테이너
- 프로젝트에 필요한 소스파일, 헤더 파일 및 리소스 파일들을 포함



프로젝트 생성(1)

◆ 프로젝트 생성(1)



프로젝트 생성(2)

◆ 프로젝트 생성(2)

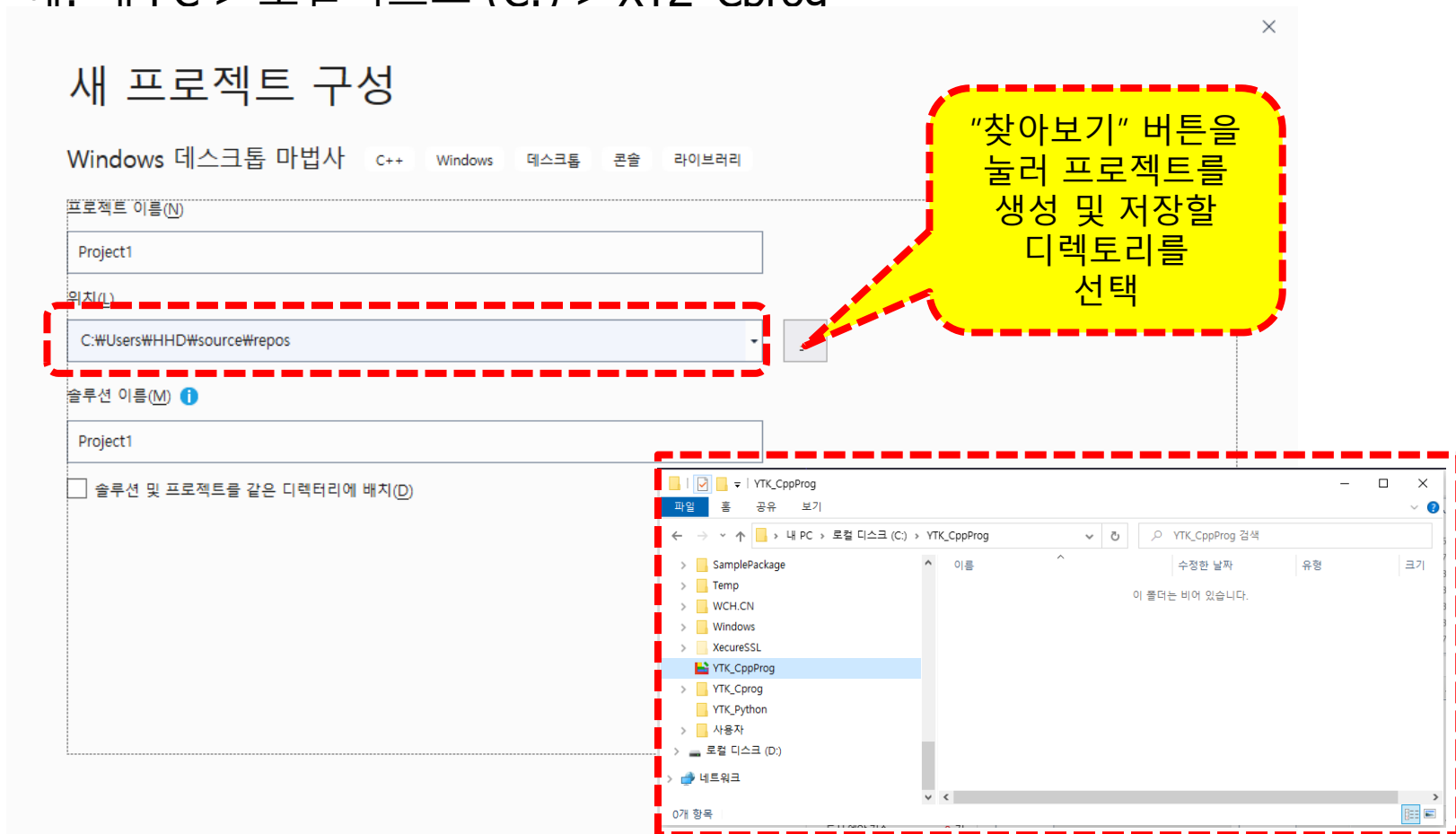
- 빈 프로젝트 선택 후 다음



프로젝트 생성(3)

◆ 프로젝트 생성(3)

- 저장위치(디렉토리) 생성/지정
- 예: 내 PC > 로컬디스크 (C:) > XYZ Cproa



프로젝트 생성(4)

◆ 프로젝트 생성(4)

- 프로젝트 이름 설정
- 예: Lab01 – My First Cpp Programming

새 프로젝트 구성

빈 프로젝트 콘솔 C++ Windows

프로젝트 이름(N)

Lab01 - My First Cpp Programming

위치(L)

C:\YTK_CppProg\

솔루션 이름(M) ⓘ

Lab01 - My First Cpp Programming

☐ 솔루션 및 프로젝트를 같은 디렉터리에 배치(D)

뒤로(B) 만들기(C)

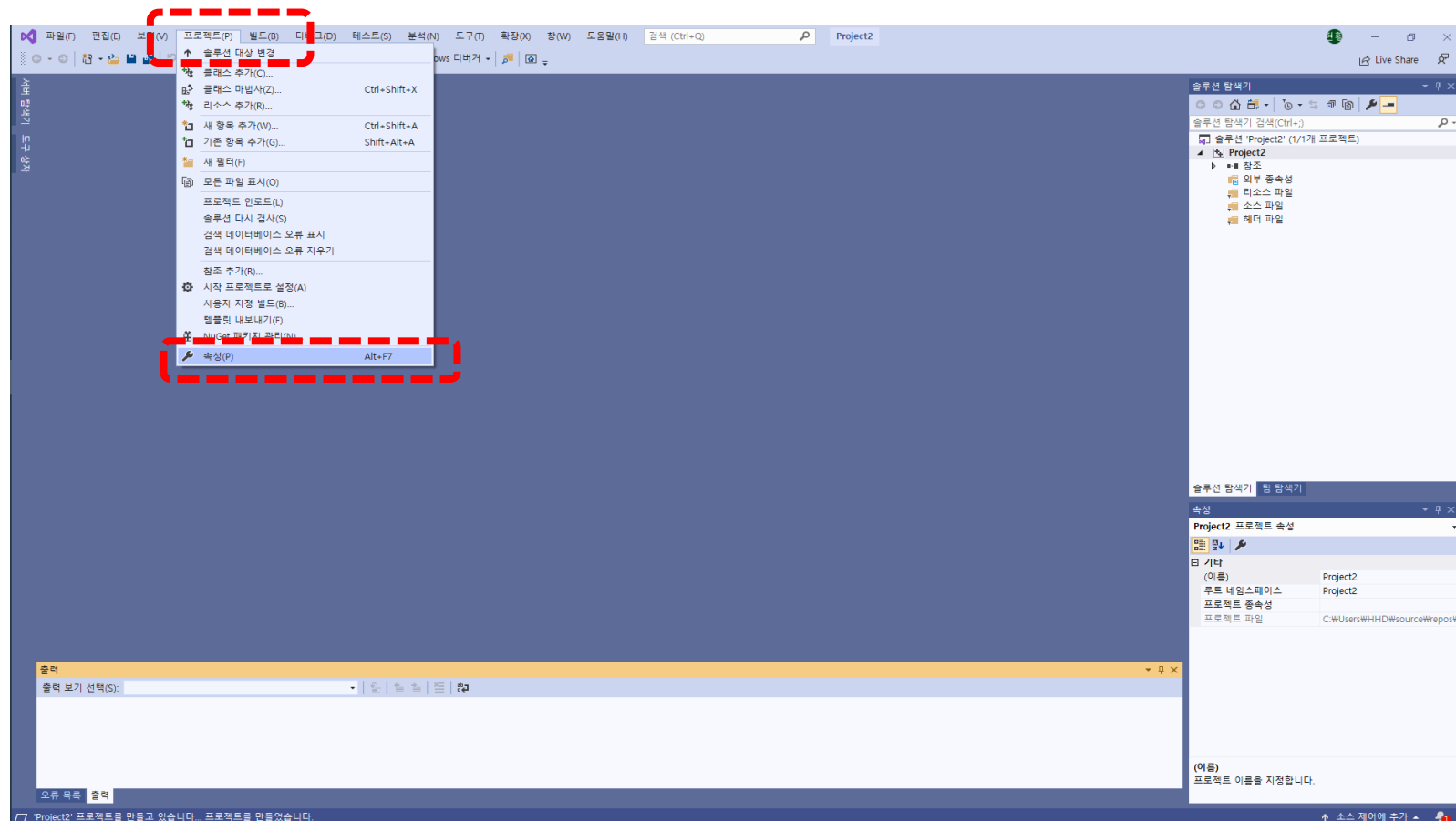
프로젝트 이름에
"Lab01 – My First Cpp Programming"
입력



프로젝트 속성 설정(1)

◆ 프로젝트 속성 설정(1)

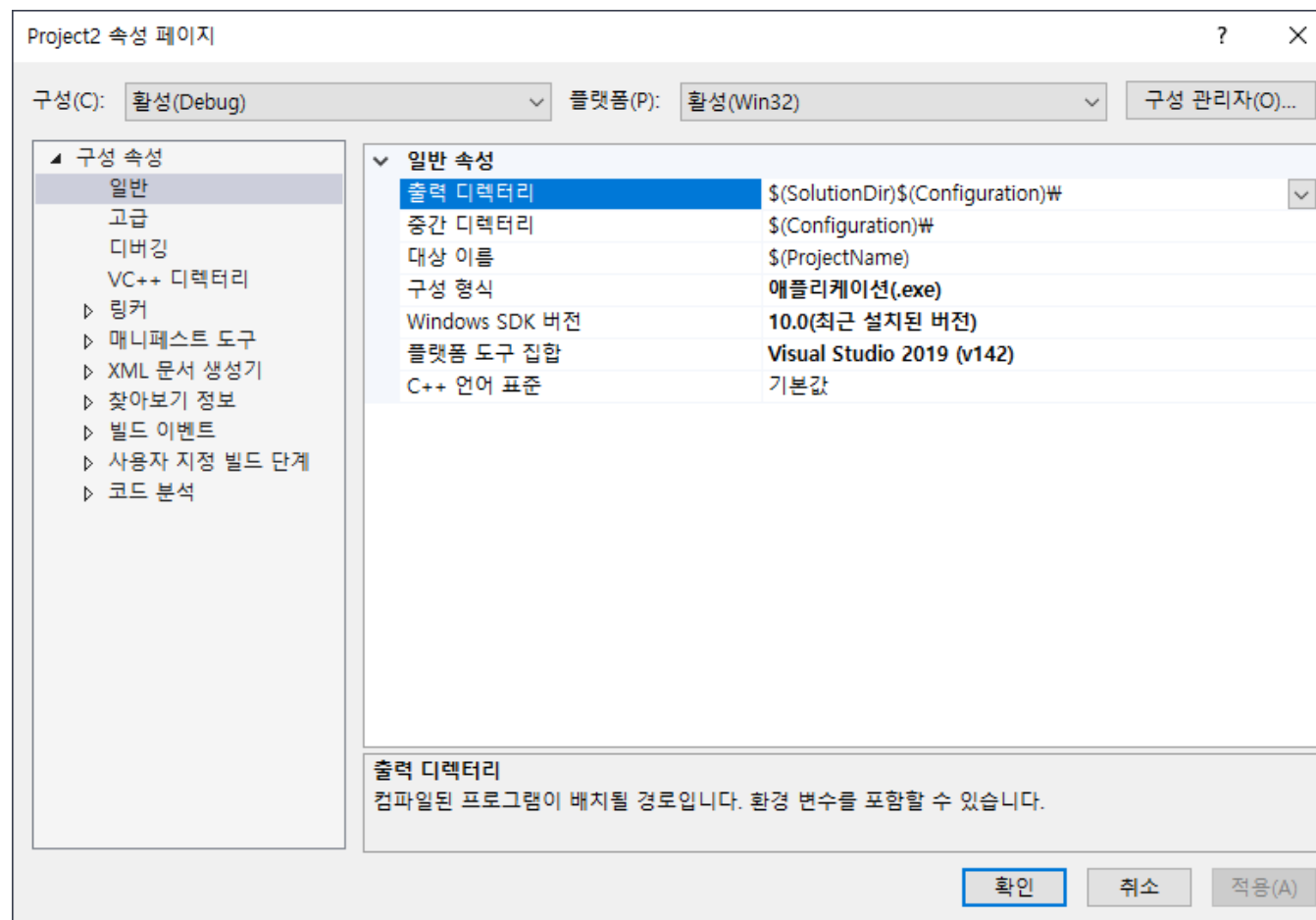
- 프로젝트 tab -> 속성 클릭(Alt+F7)



프로젝트 속성 설정(2)

◆ 프로젝트 속성 설정(2)

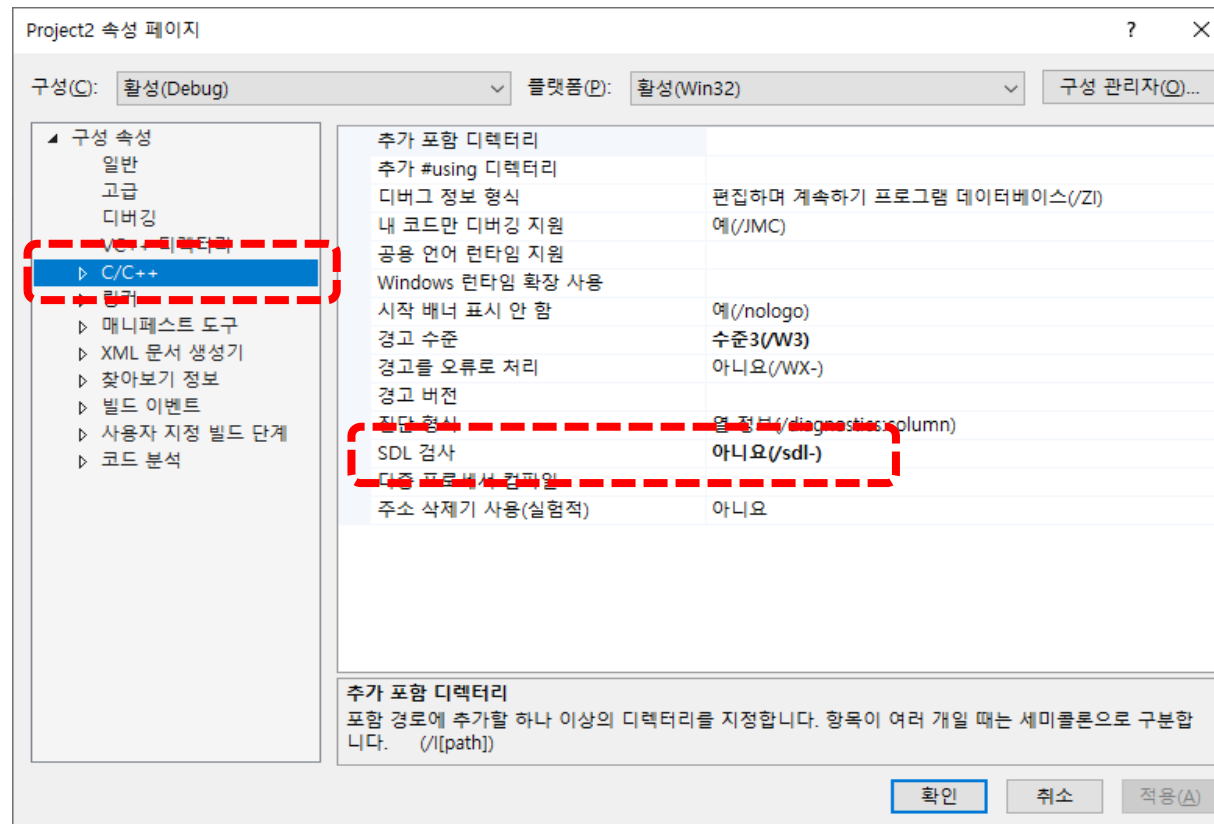
● 프로젝트 속성 페이지



프로젝트 속성 설정(3)

◆ 프로젝트 속성 설정(3)

- C/C++ 항목 클릭 후 SDL 검사를 아니요(/sdl-)로 변경 후 확인
 - scanf() 함수 사용시 Visual Studio 2010 이상 컴파일러에서는 scanf_s() 함수 사용을 권장함에 따라 scanf() 함수를 컴파일시 에러 발생.

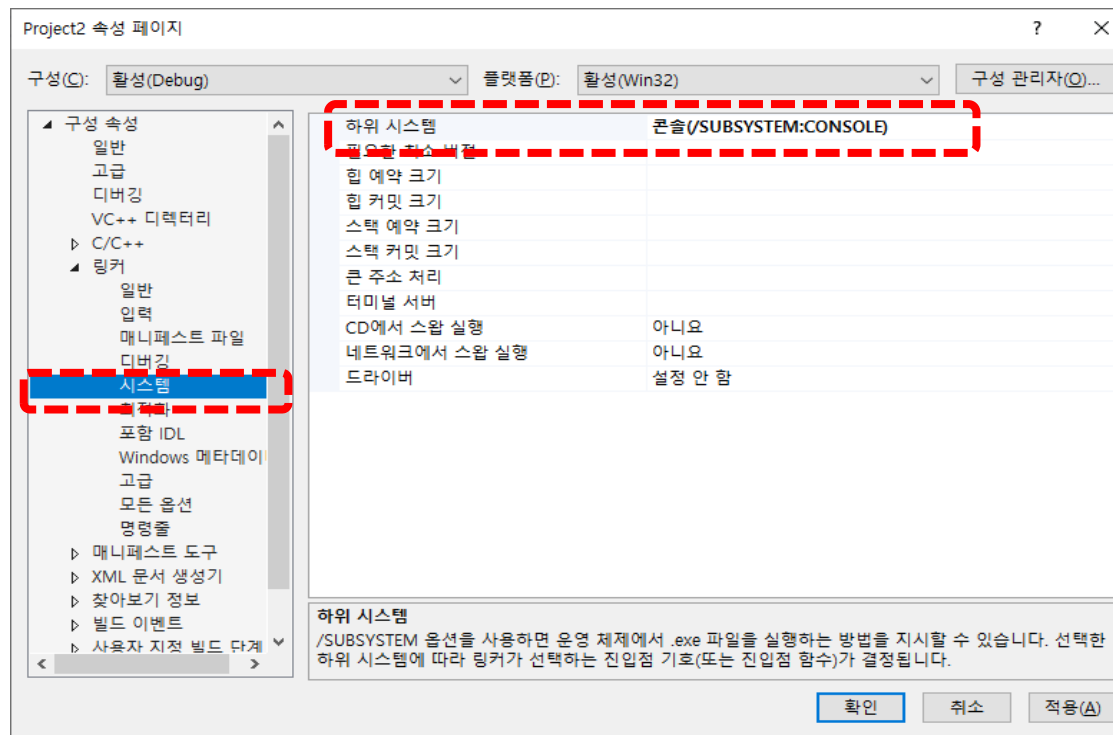


프로젝트 속성 설정(4)

◆ 프로젝트 속성 설정(4)

● 링커->시스템

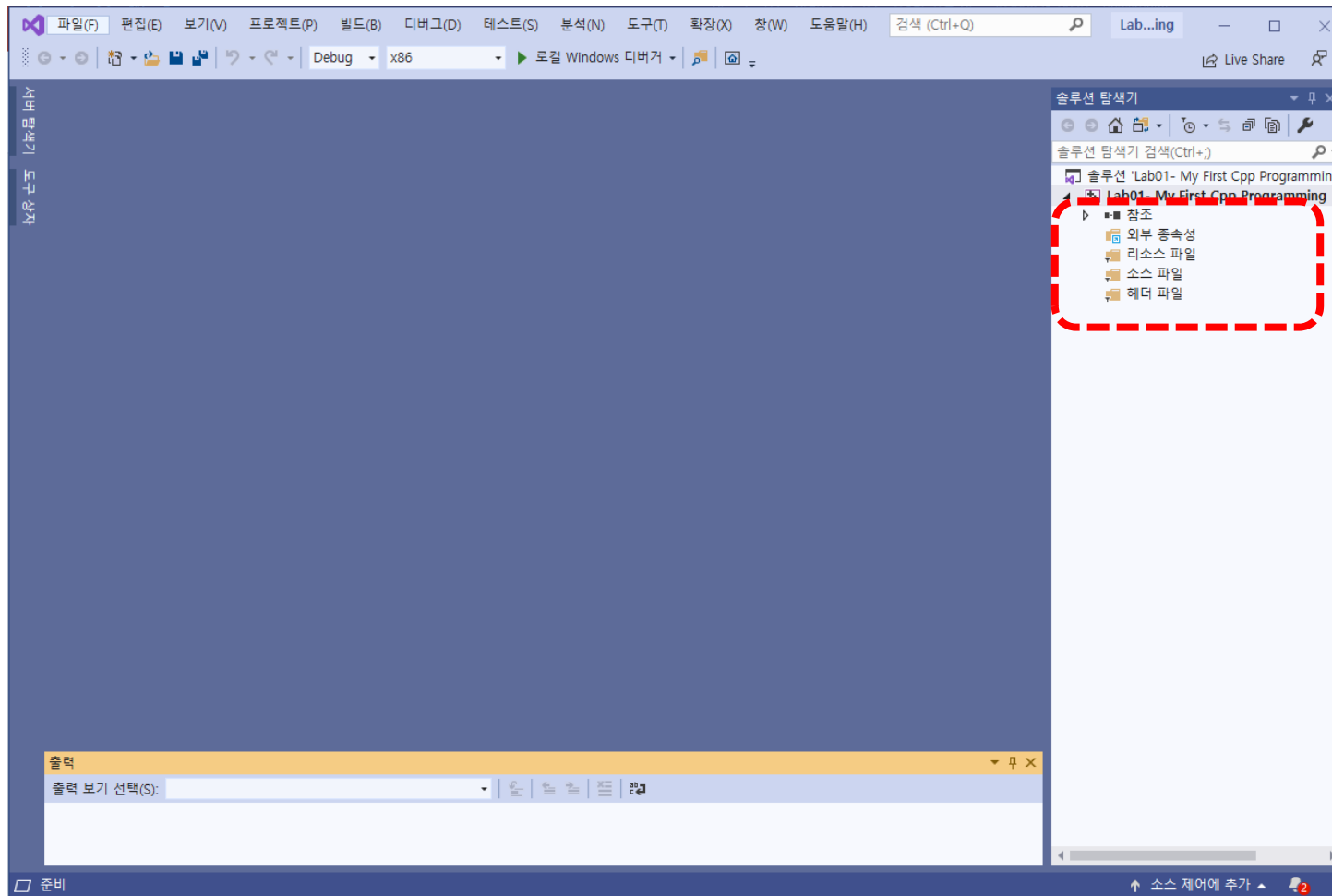
- 하위 시스템의 목록이 콘솔(/SUBSYSTEM:CONSOLE)로 체크 되어 있는지 확인
- 컴파일후 실행 창이 자동으로 종료되는 현상 발생시 다음 속성을 확인 후 수정한다.



Visual Studio Community 2019
C++ 프로그램 프로젝트 생성 및
C++ 프로그램 소스 코드 작성

프로젝트 시작하기(1)

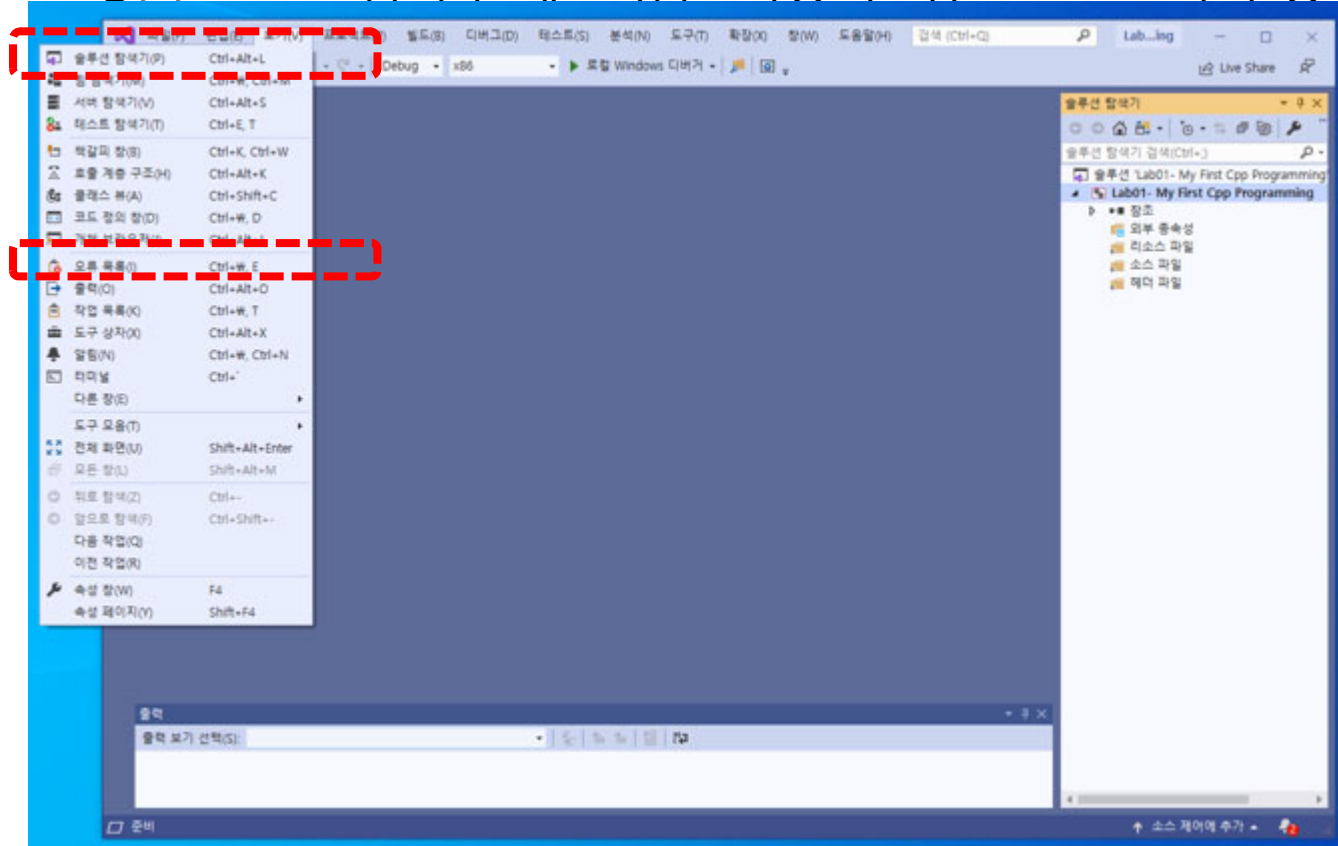
◆ 기본 구성 화면



프로젝트 시작하기(2)

◆ Visual Studio 창(window)의 종류

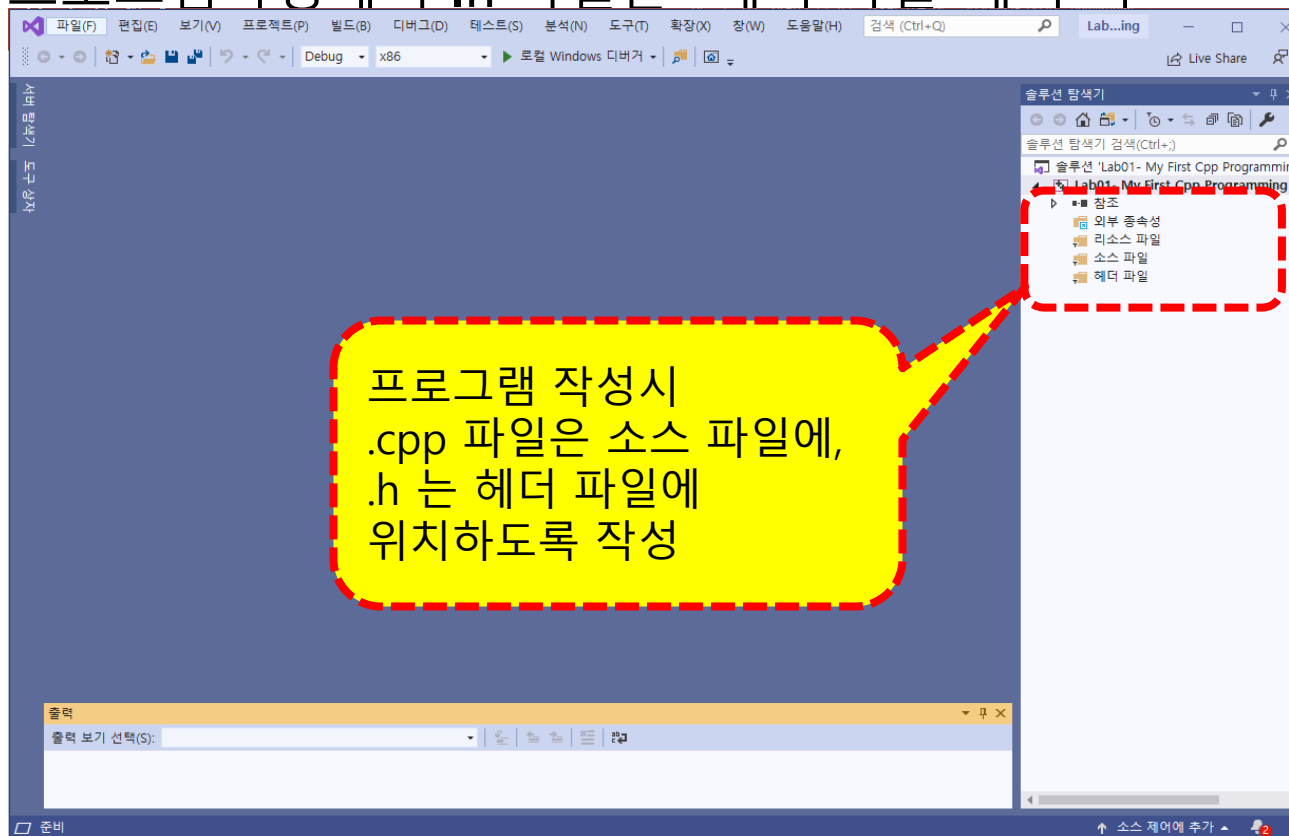
- 메뉴->보기 클릭 후 원하는 탐색기 on/off 가능
 - C 언어 프로그램시 필요 창: 솔루션 탐색기, 출력 창, 오류목록
 - C++ 프로그램시 클래스 뷰를 이용시 빠른 소스 트리 구성 가능



프로젝트 시작하기(3)

◆ 솔루션 탐색기

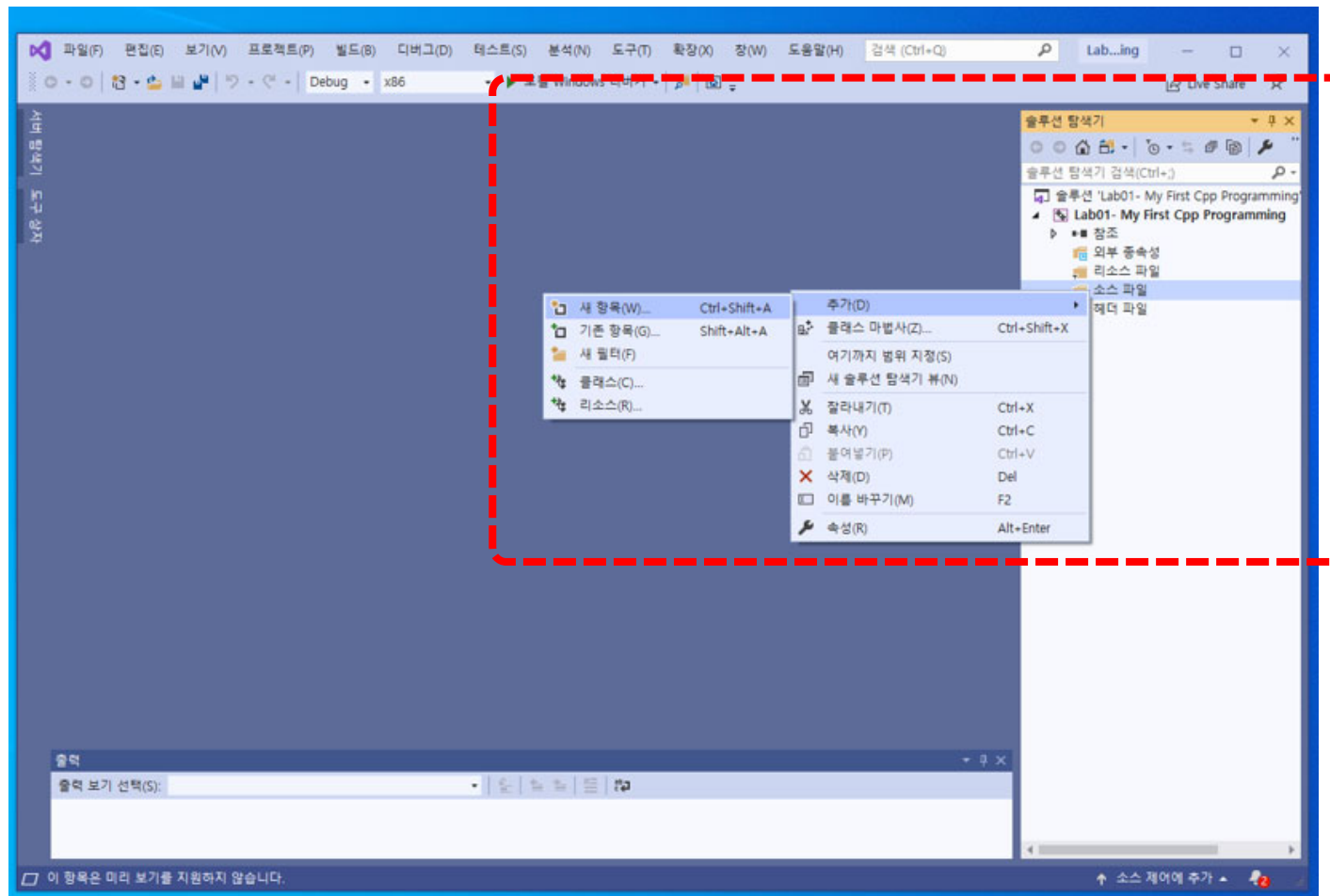
- 프로그램작성에서 .cpp 파일은 "소스파일 " 에 추가
- 프로그램작성에서 .h 파일은 "헤더 파일"에 추가



프로젝트 시작하기(4)

◆ 소스 파일 생성하기(1)

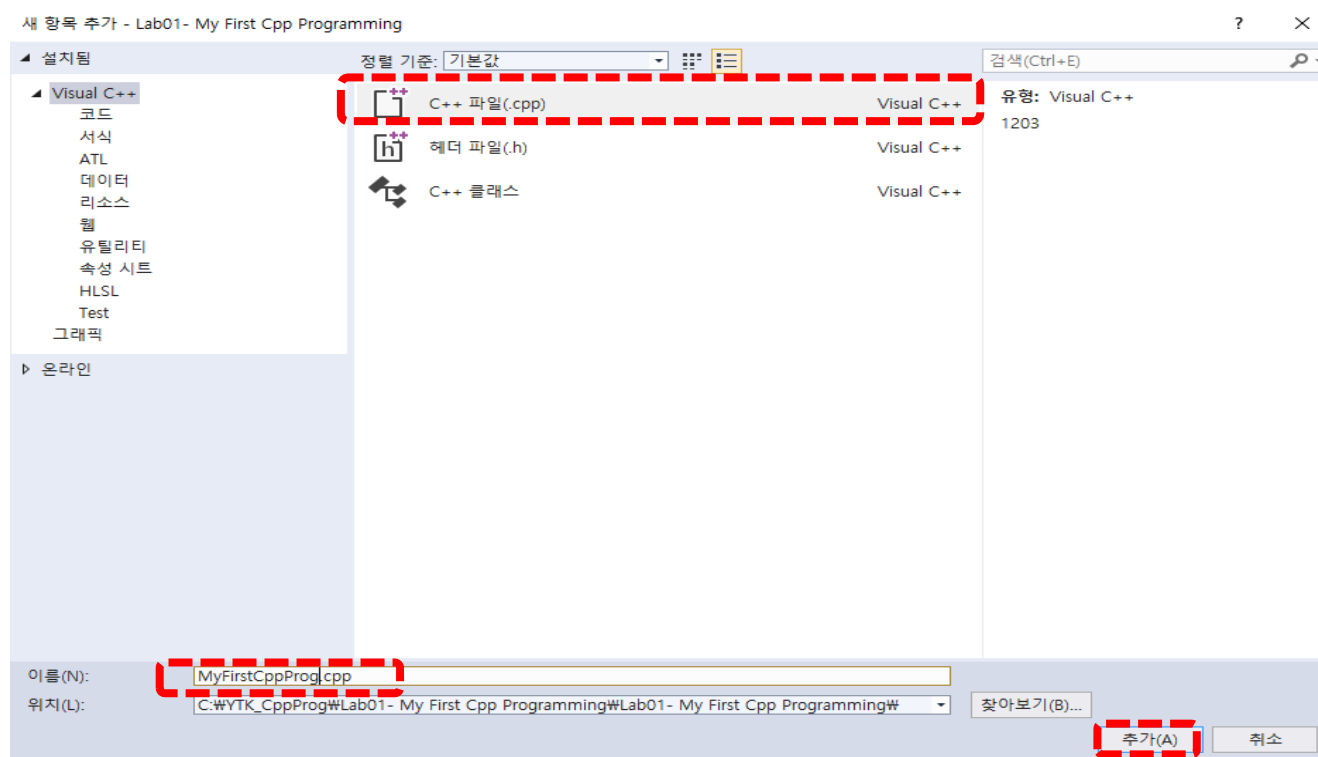
- 솔루션 탐색기 --> 소스파일 --> 추가 --> 새항목



프로젝트 시작하기(5)

◆ 소스 파일 생성하기(2)

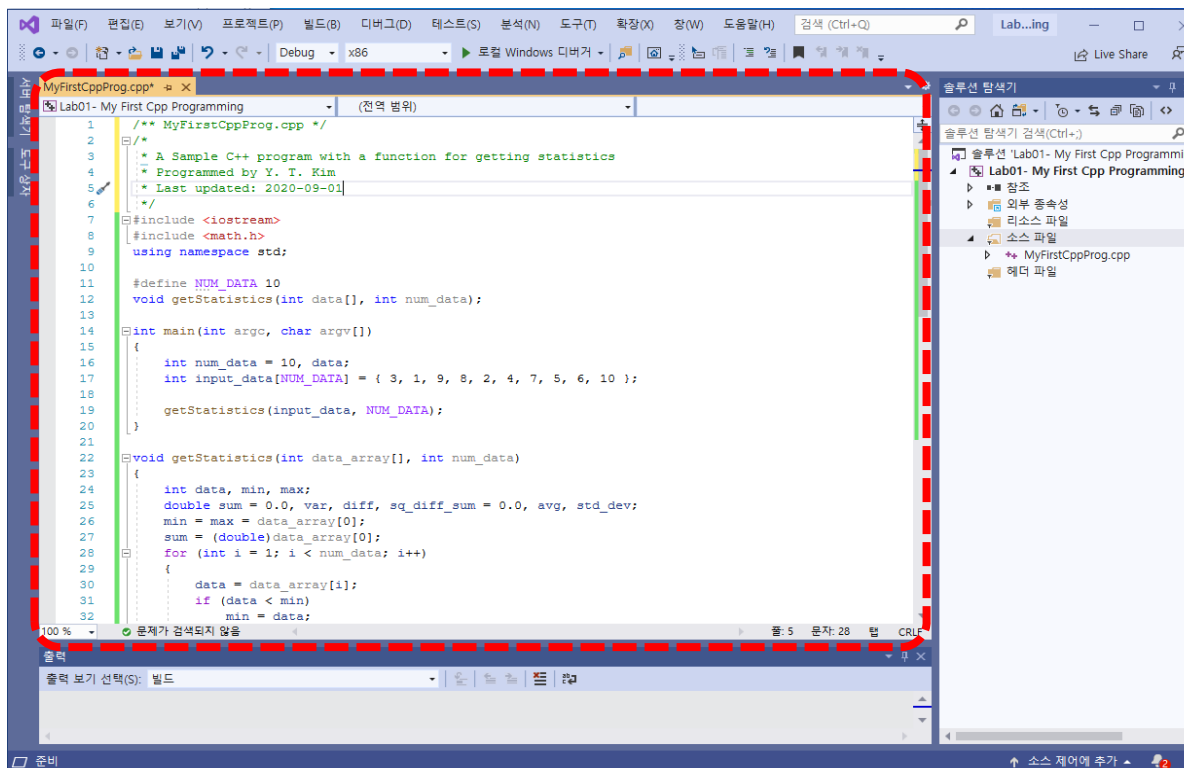
- C++파일(.cpp) 유형 선택 및 파일 이름 작성
- 파일 이름 (예) : MyFirstCppProg



프로젝트 시작하기(6)

◆ 소스 파일 편집하기

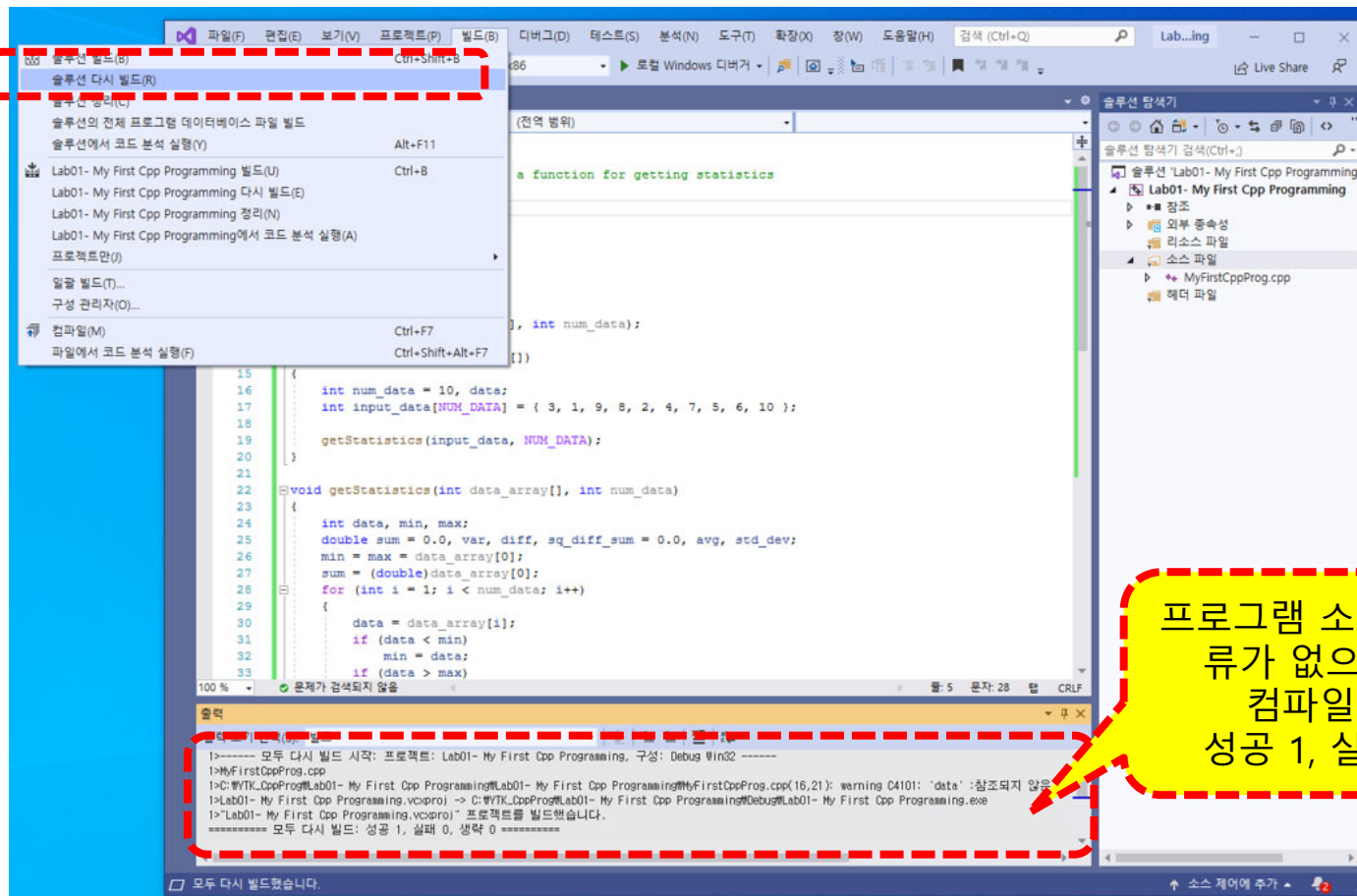
- 편집창에 소스코드 입력 및 편집
- 파일->파일이름_저장 또는 Ctrl+S 키를 이용해 작성/편집한 코드 파일로 저장



프로젝트 시작하기(7)

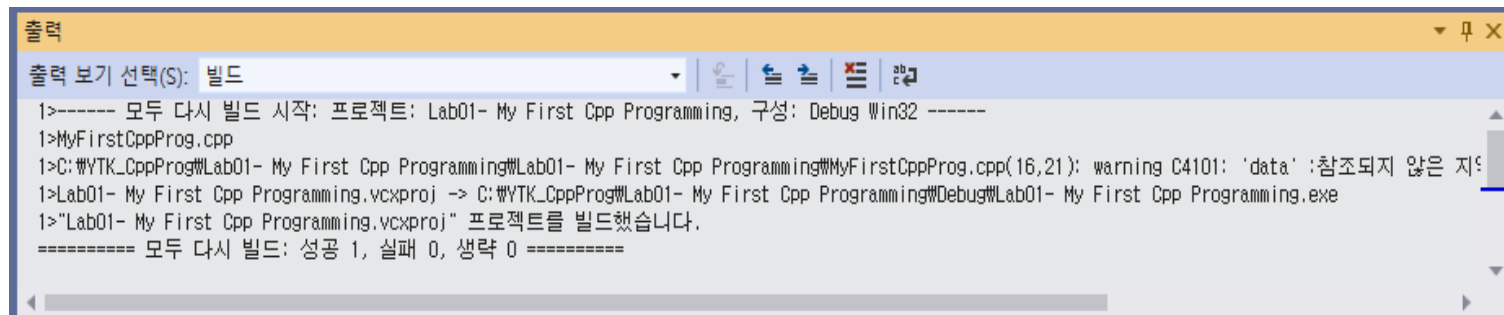
◆ 컴파일 하기

- 빌드 탭-->솔루션 다시 빌드(R) 명령 수행(단축키 : Ctrl+Alt+F7)



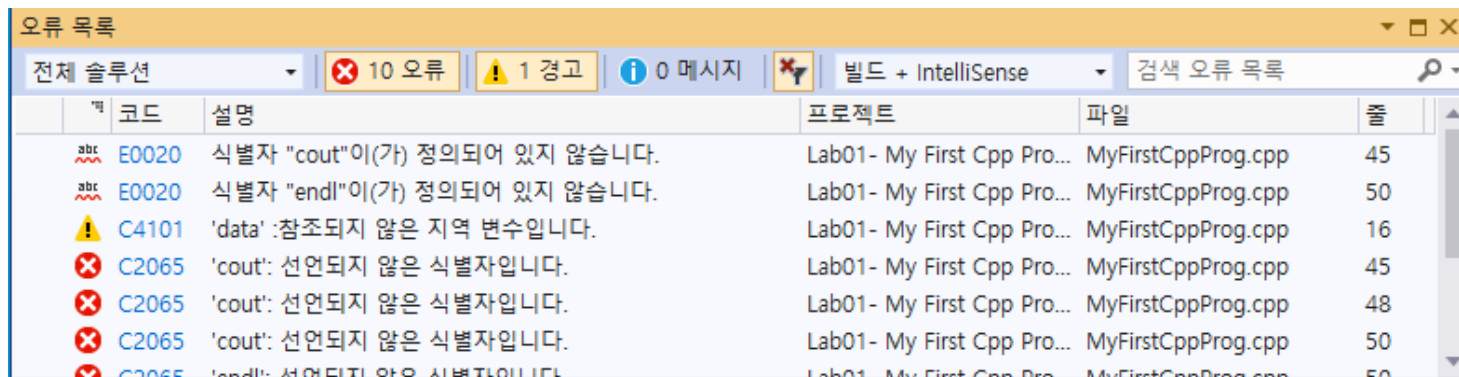
프로젝트 시작하기(8)

◆ 출력 창 - 컴파일시 발생하는 각종 상태들을 출력해주는 창



The screenshot shows the 'Output' window in Visual Studio. The 'Output View Selection(S):' dropdown is set to 'Build'. The output text shows the start of a build for 'Lab01- My First Cpp Programming' in 'Debug Win32' configuration. It lists the files being compiled, including 'MyFirstCppProg.cpp', and shows a warning C4101: 'data': 참조되지 않은 지역 변수입니다. The build process completes successfully, showing '성공 1, 실패 0, 생략 0'.

◆ 오류 목록 - 오류 발생시 오류 코드와 설명, 파일 및 줄번호 제공



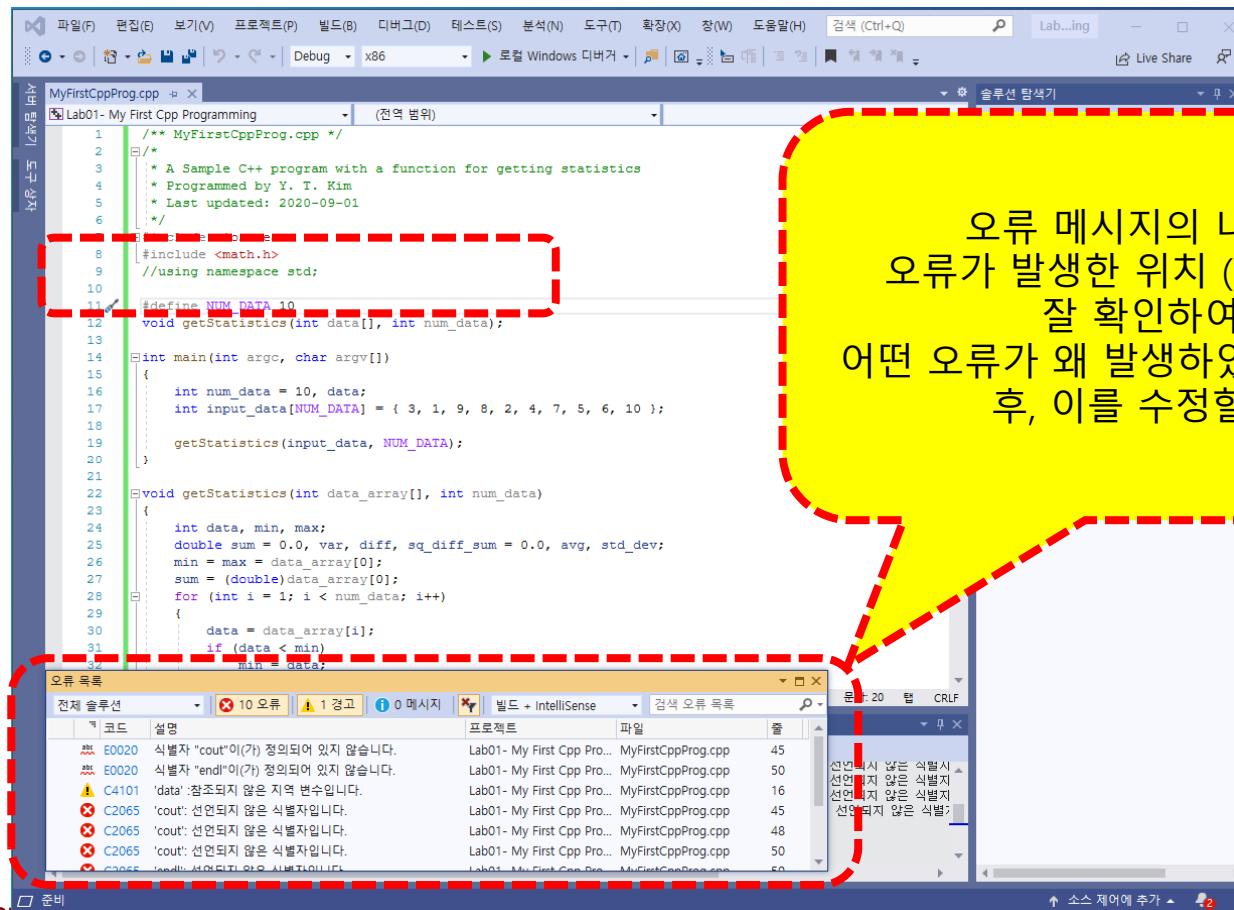
The screenshot shows the 'Error List' window in Visual Studio. It displays a table of errors and warnings. The table has columns for '코드' (Code), '설명' (Description), '프로젝트' (Project), '파일' (File), and '줄' (Line). The errors include 'E0020' (identifier not defined) and 'C4101' (unreferenced local variable). The warnings include 'C2065' (variable not declared).

코드	설명	프로젝트	파일	줄
E0020	식별자 "cout"이(가) 정의되어 있지 않습니다.	Lab01- My First Cpp Pro...	MyFirstCppProg.cpp	45
E0020	식별자 "endl"이(가) 정의되어 있지 않습니다.	Lab01- My First Cpp Pro...	MyFirstCppProg.cpp	50
C4101	'data': 참조되지 않은 지역 변수입니다.	Lab01- My First Cpp Pro...	MyFirstCppProg.cpp	16
C2065	'cout': 선언되지 않은 식별자입니다.	Lab01- My First Cpp Pro...	MyFirstCppProg.cpp	45
C2065	'cout': 선언되지 않은 식별자입니다.	Lab01- My First Cpp Pro...	MyFirstCppProg.cpp	48
C2065	'cout': 선언되지 않은 식별자입니다.	Lab01- My First Cpp Pro...	MyFirstCppProg.cpp	50
C2065	'endl': 선언되지 않은 식별자입니다.	Lab01- My First Cpp Pro...	MyFirstCppProg.cpp	50

프로젝트 시작하기(5)

◆ 오류 목록

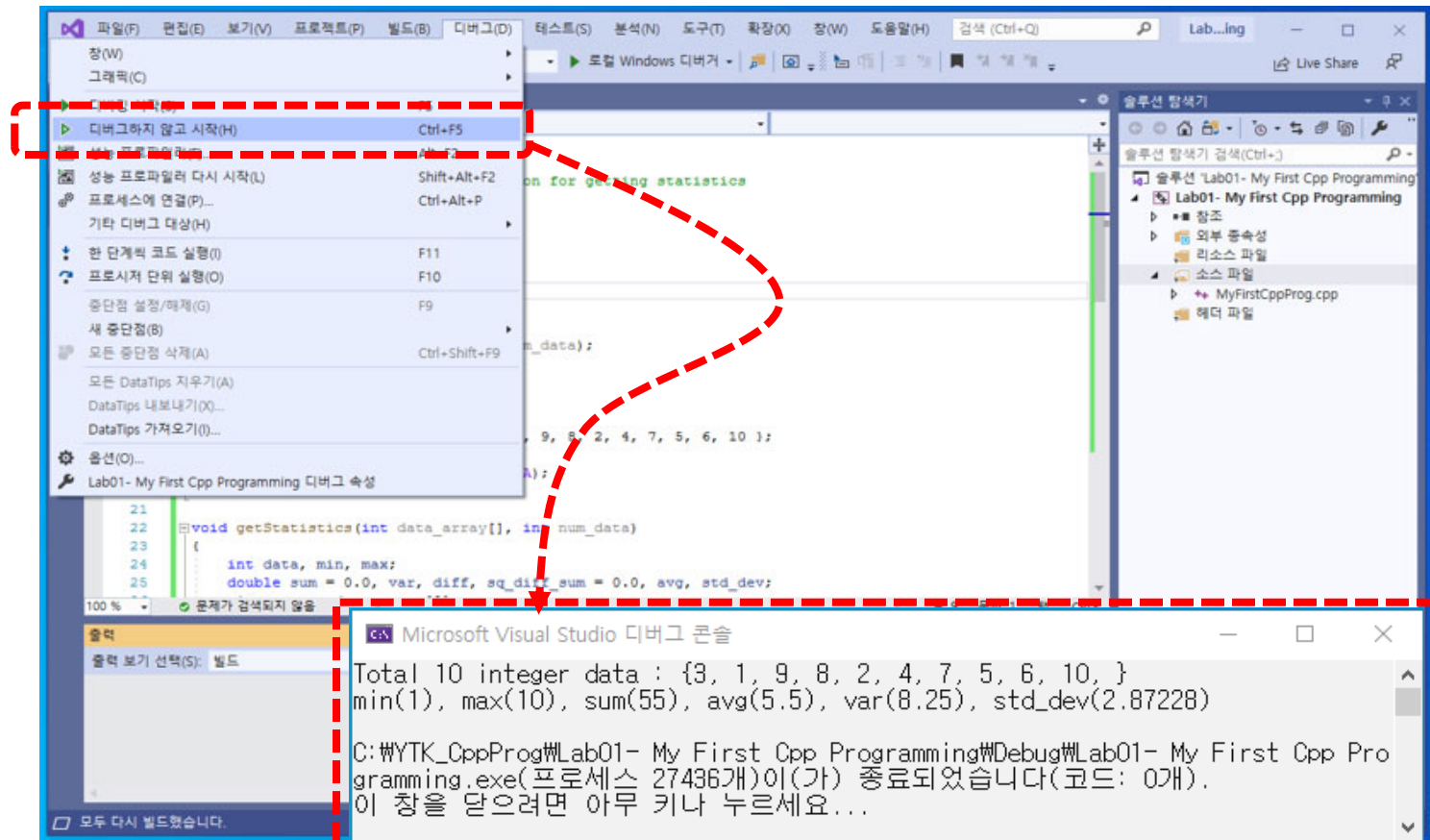
- 컴파일 오류 발생시 내용 표시창.
 - 오류 코드와 설명 및 오류가 발생한 파일과 줄을 표시한다.



프로젝트 시작하기(10)

◆ 프로그램 실행 하기

- 디버그 탭 -> “디버깅 하지 않고 시작(H)” 메뉴선택(단축키 : Ctrl+F5)



기본적인 C++ 코드 구조

```
/** SampleCppProg.cpp */  
/* A Sample C++ program  
 * Programmed by Y. T. Kim  
 * Last updated: 2020-09-01 */
```

주석문 (comment)

```
#include <iostream>
```

include system library for
standard input/output and mathematics functions

```
using namespace std;
```

define name space for standard library modules

```
int add(int a, int b); // 함수원형 (function prototype) 선언
```

```
int main()
```

```
{  
    int x, y, z;  
    cout << "input two integers : ";  
    cin >> x >> y;  
    z = add(x, y);  
    cout << x << " + " << y << " = " << z << endl;  
}
```

```
int add(int a, int b)
```

```
{  
    int c;  
    c = a + b;  
    return c;  
}
```



주석 (Comment)

◆ 프로그램 소스 코드 작성에서의 기본 주석 표기내용

```
/**
 * 파일명: "MyFirstCppProg.cpp" or "xxx.h", or "yyy.cpp"
 * 프로그램의 목적 및 기본 기능:
 *   - 이 프로그램은 사물인터넷의 온도 센서를 읽고, 사전에 설정된 온도가 유지될 수 있도록 히터/에어컨을
 *     동작시키는 .....
 *
 * 프로그램 작성자: 홍길0 (2020년 9월 1일),
 * 최종 수정 및 보완 : Version 2.0, 2020년 9월 4일 (박영0).
 *
 * =====
 * 프로그램 수정/보완 이력
 * =====
 * 프로그램 수정/보완작업자   일자   수정/보완 내용
 * 홍길0       2020/03/01   v1.0   온도센서 읽기, 히터 동작, 에어컨 동작 모듈 완성
 * 정동0       2020/09/03   v1.1   GUI 기능 보완
 * 박영0       2020/09/04   v2.0   온도 센서 읽기의 데이터 오류 발생 여부 확인 기능 추가
 * =====
 */
```



오류 수정 및 디버깅

오류 수정 및 디버깅

◆ 컴파일이나 실행 시에 오류가 발생할 수 있다.

◆ 에러와 경고

- 에러(error): 심각한 오류
- 경고(warning): 경미한 오류

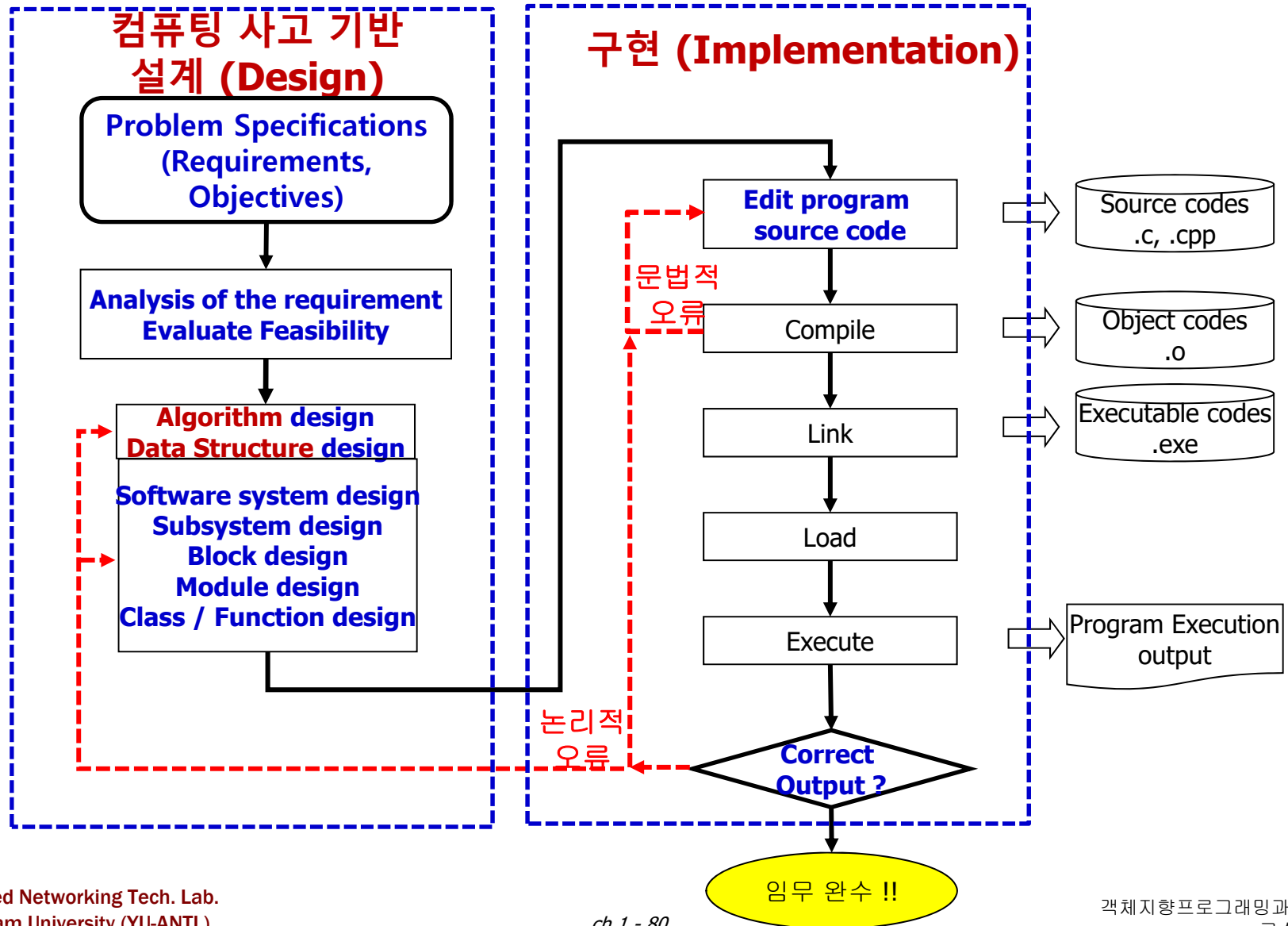


◆ 오류의 종류

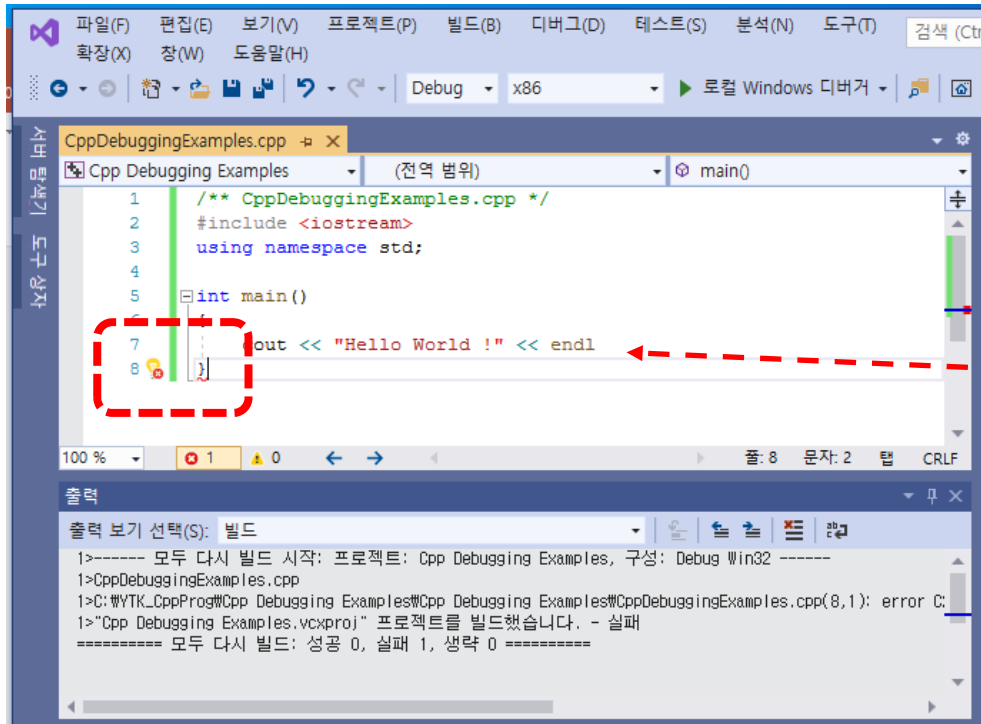
- 컴파일 시간 오류: 대부분 문법적인 오류
- 실행 시간 오류: 실행 중에 0으로 나누는 연산 같은 오류
- 논리 오류: 논리적으로 잘못되어서 결과가 의도했던 대로 나오지 않는 오류



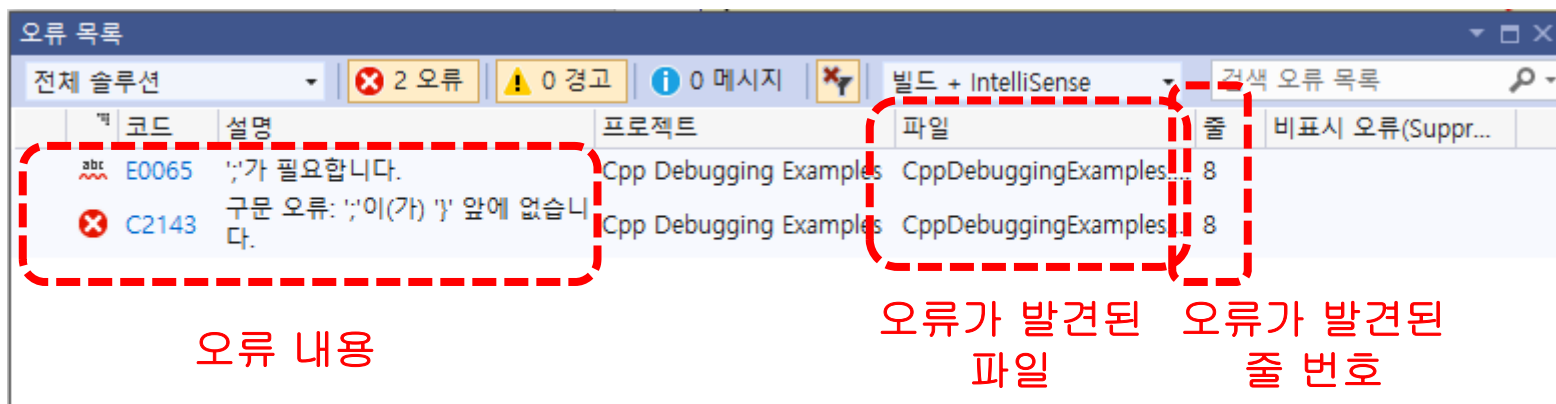
프로그램 개발 및 오류 수정 과정



오류 #1



문장의 끝에 ;
(세미콜론)이
없음!!



오류 #2

```
1  /** CppDebuggingExamples.cpp */
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello World !" << endl
8  }
```

*과 /이 떨어져
있음 ->
나머지 프로그램
전체가
주석처리 됨

오류 목록					
전체 솔루션		2 오류	0 경고	0 메시지	빌드 + IntelliSense
아이콘	코드	설명	프로젝트	파일	줄
	E0006	주석이 파일 끝에서 닫히지 않았습니다.	Cpp Debugging Examples	CppDebuggingExamples....	1
	C1071	주석에서 예기치 않은 파일의 끝이 나타났습니다.	Cpp Debugging Examples	CppDebuggingExamples....	9

주석은 프로그램에 대한 설명글로서 /* */ 안에 표시한다.



오류 #3

```
1  /** CppDebuggingExamples.cpp */
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello World !" << end;
8  }
```

키워드 이름을
잘못 입력
end : X
endl : O

abc E0349 이러한 피연산자와 일치하는 "<<" 연산자가 없습니다. Cpp Debugging Examples CppDebuggingExamples.... 7

✖ C2679 이항 '<<': 오른쪽 피연산자로 'overloaded-function' 형식을 사용하는 연산자가 없거나 허용되는 변환이 없습니다. Cpp Debugging Examples CppDebuggingExamples.... 7



많이 발생하는 문법적 오류 (1)

◆ #include <iostream>을 하지 않았을 경우

● 잘못된 코드 작성 내용

```
1  /** CppDebuggingExamples.cpp */
2  // #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello World !" << endl;
8  }
```

올바른 코드작성 내용

```
1  /** CppDebuggingExamples.cpp */
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello World !" << endl;
8  }
```

● 오류 목록 내용

abc E0020	식별자 "cout"이(가) 정의되어 있지 않습니다.	Cpp Debugging Examples	CppDebuggingExamples....	7
abc E0020	식별자 "endl"이(가) 정의되어 있지 않습니다.	Cpp Debugging Examples	CppDebuggingExamples....	7
✗ C2065	'cout': 선언되지 않은 식별자입니다.	Cpp Debugging Examples	CppDebuggingExamples....	7
✗ C2065	'endl': 선언되지 않은 식별자입니다.	Cpp Debugging Examples	CppDebuggingExamples....	7



많이 발생하는 문법적 오류 (2)

◆ using namespace std;를 설정 하지 않았을 경우





● 잘못된 코드 작성 내용

```
1  /** CppDebuggingExamples.cpp */
2  #include <iostream>
3  //using namespace std;
4
5  int main()
6  {
7      cout << "Hello World !" << endl;
8  }
```



```
1  /** CppDebuggingExamples.cpp */
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello World !" << endl;
8  }
```

● 오류 목록 내용

 E0020	식별자 "cout"이(가) 정의되어 있지 않습니다.	Cpp Debugging Examples	CppDebuggingExamples....	7
 E0020	식별자 "endl"이(가) 정의되어 있지 않습니다.	Cpp Debugging Examples	CppDebuggingExamples....	7
 C2065	'cout': 선언되지 않은 식별자입니다.	Cpp Debugging Examples	CppDebuggingExamples....	7
 C2065	'endl': 선언되지 않은 식별자입니다.	Cpp Debugging Examples	CppDebuggingExamples....	7

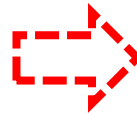


많이 발생하는 문법적 오류 (3)

◆ ; (semicolon)을 붙이지 않은 경우

- 잘못된 코드 작성 내용

```
1  /** CppDebuggingExamples.cpp */
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello World !" << endl
8  }
```



```
1  /** CppDebuggingExamples.cpp */
2  #include <iostream>
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello World !" << endl;
8  }
```

- 에러 목록 내용

abc	E0065	';'가 필요합니다.	Cpp Debugging Examples	CppDebuggingExamples....	8
✖	C2143	구문 오류: ';'이(가) ';' 앞에 없습니다.	Cpp Debugging Examples	CppDebuggingExamples....	8



많이 발생하는 문법적 오류 (4)

◆ 함수 prototype 선언 시 발생할 수 있는 에러 (함수 원형 마지막에는 반드시 세미콜론 포함)

● 잘못된 코드 작성 내용

```
1  /** CppDebuggingExamples.cpp */
2  #include <iostream>
3  using namespace std;
4
5  void functionA()
6  int main()
7  {
8      cout << "Hello World !" << endl;
9      functionA();
10 }
11
12 void functionA()
13 {
14     cout << "functionA() ..." << endl;
15 }
```

올바른 코드 작성 내용

```
1  /** CppDebuggingExamples.cpp */
2  #include <iostream>
3  using namespace std;
4
5  void functionA();
6  int main()
7  {
8      cout << "Hello World !" << endl;
9      functionA();
10 }
11
12 void functionA()
13 {
14     cout << "functionA() ..." << endl;
15 }
```

● 에러 목록 내용

abc E0130	'<'가 필요합니다.	Cpp Debugging Examples	CppDebuggingExamples....	6
✖ C2144	구문 오류: int'은(는) ';' 다음에 와야 합니다.	Cpp Debugging Examples	CppDebuggingExamples....	6



많이 발생하는 문법적 오류 (5)

◆ 함수 proto type 선언 시 발생할 수 있는 외부참조 에러

- 잘못된 코드 작성 내용

```
1  /** CppDebuggingExamples.cpp */
2  #include <iostream>
3  using namespace std;
4
5  void functionA();
6  int main()
7  {
8      cout << "Hello World !" << endl;
9      functionA();
10 }
11
12 void function()
13 {
14     cout << "functionA() ..." << endl;
15 }
```

- 올바른 코드 작성 내용

```
1  /** CppDebuggingExamples.cpp */
2  #include <iostream>
3  using namespace std;
4
5  void functionA();
6  int main()
7  {
8      cout << "Hello World !" << endl;
9      functionA();
10 }
11
12 void functionA()
13 {
14     cout << "functionA() ..." << endl;
15 }
```

❌ LNK2015 "void __cdecl functionA(void)" (? functionA@@YAXXZ)_main 함수에서 참조되는 확인할 수 없는 외부 기호 Cpp Debugging Examples CppDebuggingExamples.... 1

❌ LNK1120 1개의 확인할 수 없는 외부 참조입니다. Cpp Debugging Examples Cpp Debugging Example... 1



많이 발생하는 문법적 오류와 오류 메시지

Cpp프로그램 문법적 오류	오류 메시지	올바른 입력
#incude <iostream>	E0011 인식할 수 없는 전처리기 지시문입니다	#include<iostream>
#include<iostrem>	E1696 파일 소스을(를) 열수 없습니다. "iostrem"	#include<iostream>
#include iostream	C2006 '#include': "FILENAME" 또는 <FILENAME>이 필요합니다. C1083 포함 파일을 열 수 없습니다. ":No such file or directory	#include<iostream>
int x double y;	C2144: 구문오류: double은 ';'다음에 와야 합니다.	int x; double y;
main() { ... }	C4430 형식 지정자가 없습니다. int로 가정합니다. 참고: C++에 서는 기본 int를 지원하지 않습니다.	int main() { ... }
void main() int x;	E0130 '{'가 필요합니다. C2144: int는 ';'다음에 와야 합니다.	void main() { int x; }



Visual Studio 단축키

◆ 단축키

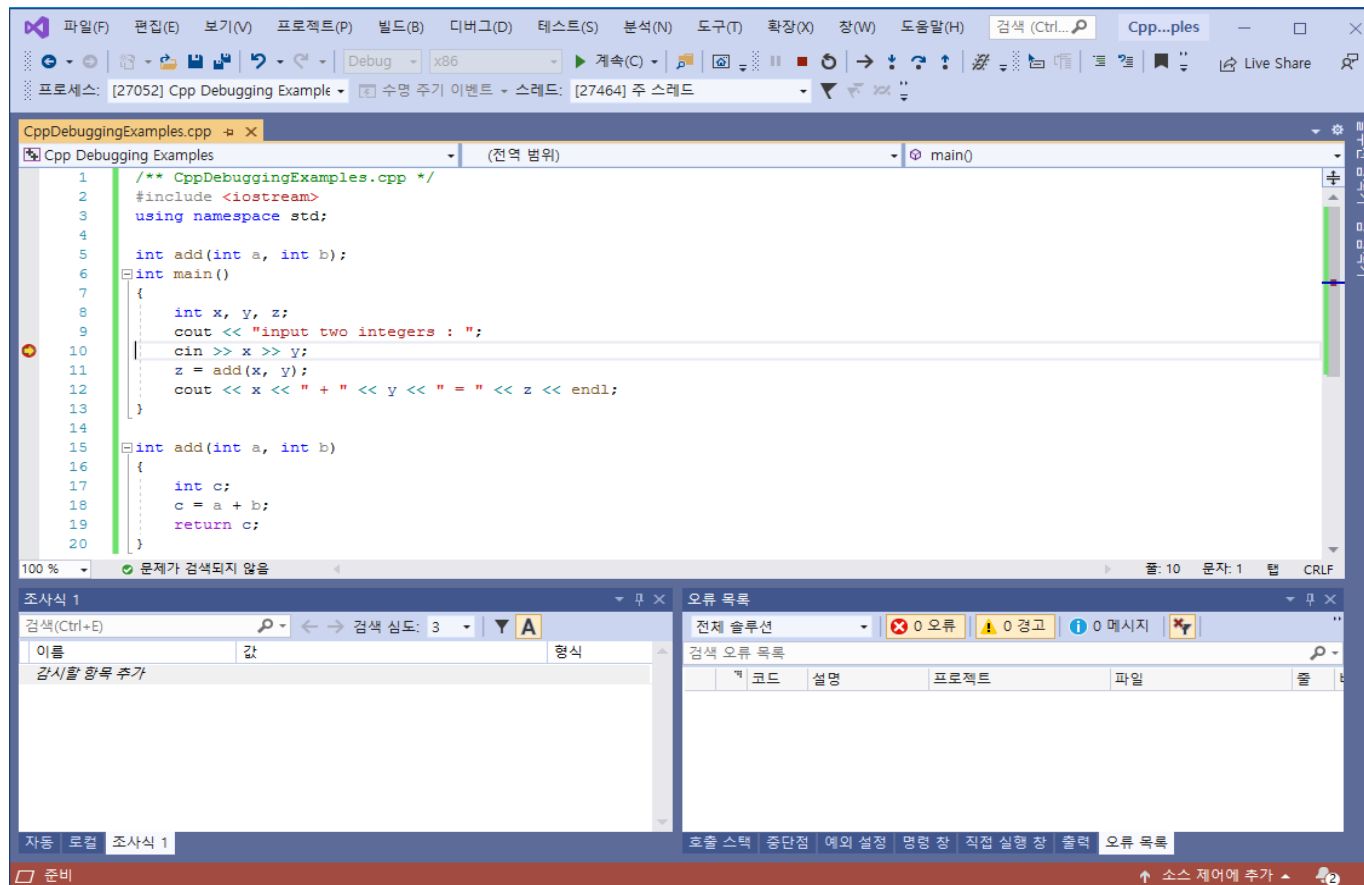
단축키	기능
Ctrl+F5	프로젝트 빌드 및 실행
Ctrl+F7	프로젝트 빌드
F5	디버깅 모드 진입
F9	커서가 있는 부분에 중단점 생성 (왼쪽 편 빨간색 점)
F10	디버깅 모드 진입 후 코드 라인 바이 라인 진행 (함수 내 진입 x)
F11	디버깅 모드 진입 후 코드 라인 바이 라인 진행 (함수 내 진입 o)



Visual Studio Debugging tool 사용법(1)

◆ Debugging 모드 진입

- **F5(Go)** 를 눌러 Debugging 모드로 진입.



Visual Studio Debugging tool 사용법(2)

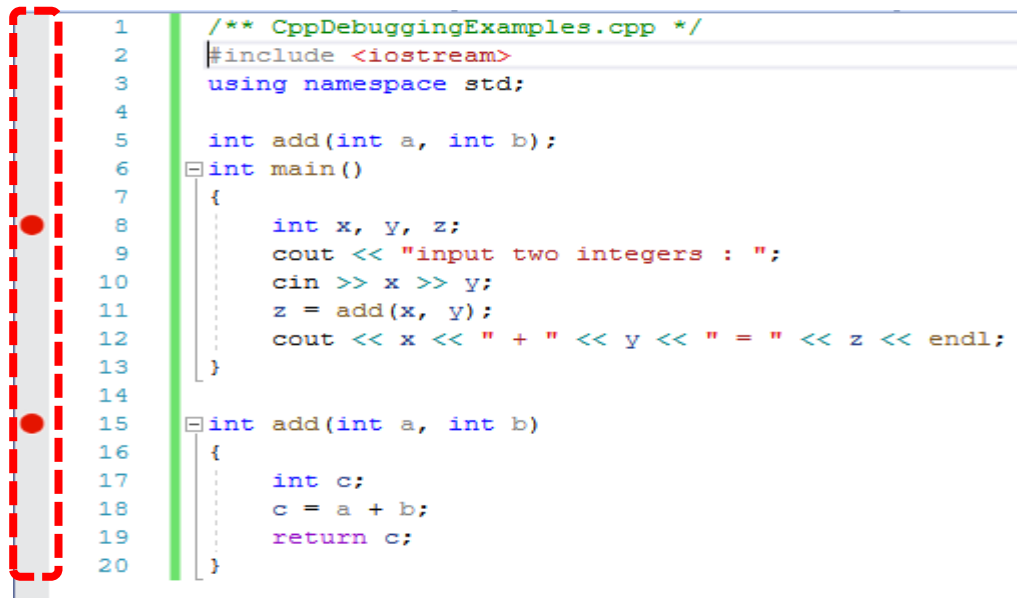
◆ Break Point 설정 - F9 키

● Break Point란?

- 사용자가 원하는 지점을 중단점으로 지정함으로 해서 상세 관찰이 필요한 지점으로 신속하게 이동할 수 있게 함

● Break Point 설정하는 법

- 붉은 box로 표시된 부분을 마우스로 클릭하여 원하는 지점에 Break Point 설정
- 커서가 깜박이고 있는 지점에서 F9를 누르면 해당 라인에 Break Point 설정
- 2개 이상의 복수개의 Break Point도 설정가능



```
1  /** CppDebuggingExamples.cpp */
2  #include <iostream>
3  using namespace std;
4
5  int add(int a, int b);
6  int main()
7  {
8      int x, y, z;
9      cout << "input two integers : ";
10     cin >> x >> y;
11     z = add(x, y);
12     cout << x << " + " << y << " = " << z << endl;
13 }
14
15 int add(int a, int b)
16 {
17     int c;
18     c = a + b;
19     return c;
20 }
```

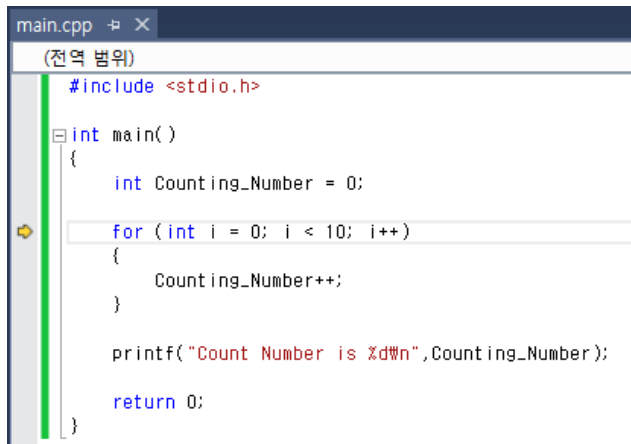


Visual Studio Debugging tool 사용법(3)

◆ Break Points 의 필요성

- 순차적 코드진행을 skip하기 때문에 더욱 빠른 Debugging 속도를 보장해줌.

ex)

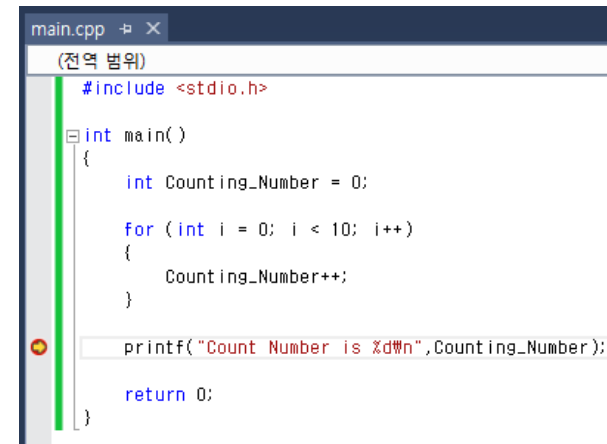


```
main.cpp [X]
(전역 범위)
#include <stdio.h>

int main()
{
    int Counting_Number = 0;
    for (int i = 0; i < 10; i++)
    {
        Counting_Number++;
    }

    printf("Count Number is %d\n", Counting_Number);

    return 0;
}
```



```
main.cpp [X]
(전역 범위)
#include <stdio.h>

int main()
{
    int Counting_Number = 0;
    for (int i = 0; i < 10; i++)
    {
        Counting_Number++;
    }

    printf("Count Number is %d\n", Counting_Number);

    return 0;
}
```

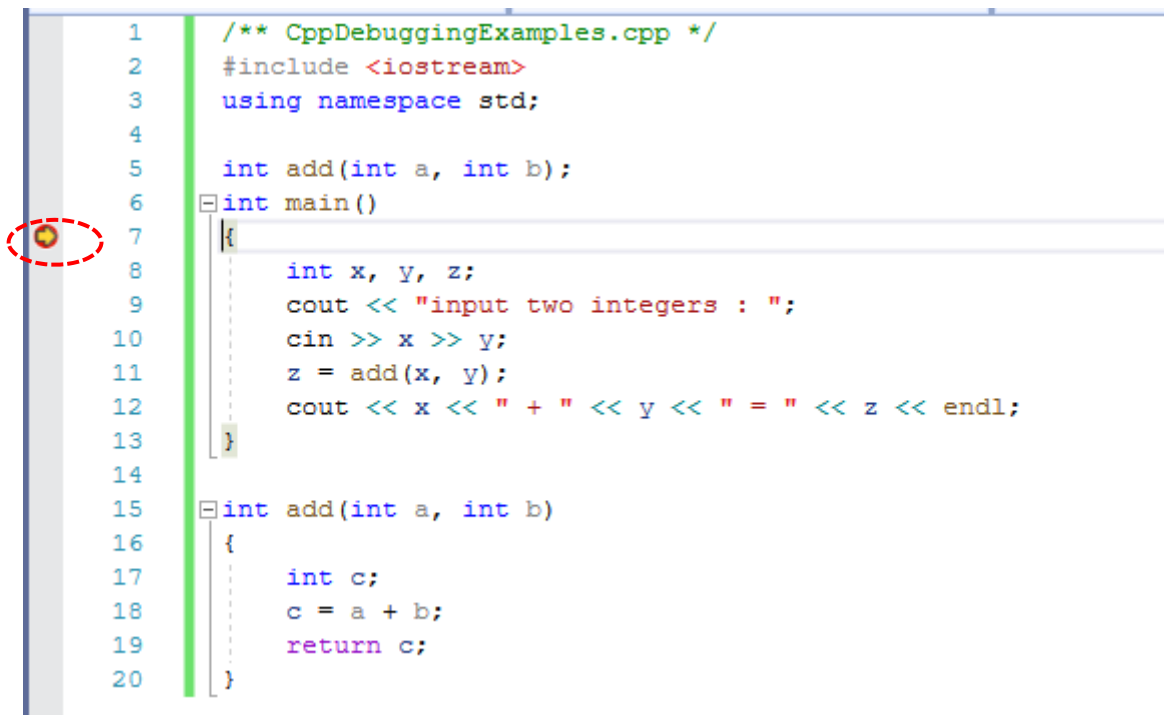
- 왼쪽 그림처럼 Break Point를 설정하지 않고 Debugging을 한다면 for-loop이 종료될 때까지 10번의 F10을 눌러 Debugging을 진행해야 함.
- 하지만 오른쪽 그림처럼 for loop 이후에 break point를 설정해두고, F5를 눌러 debugging을 실행하면 for loop을 진행하지 않고 바로 break point 위치로 이동 가능



Visual Studio Debugging tool 사용법(4)

◆ Break point 설정 후의 단계별 전진 (tracing) – F10, F11

- main()문 바로 다음에 break point를 설정 (마우스를 사용하여 줄 번호 앞에 클릭)
- **F5**를 눌러 디버깅을 시작하면 break point가 설정된 지점에서 정지/대기 함.
- **F10(Step Over)** 또는 **F11(Step into)**를 사용하여 한 단계씩 전진시킴.

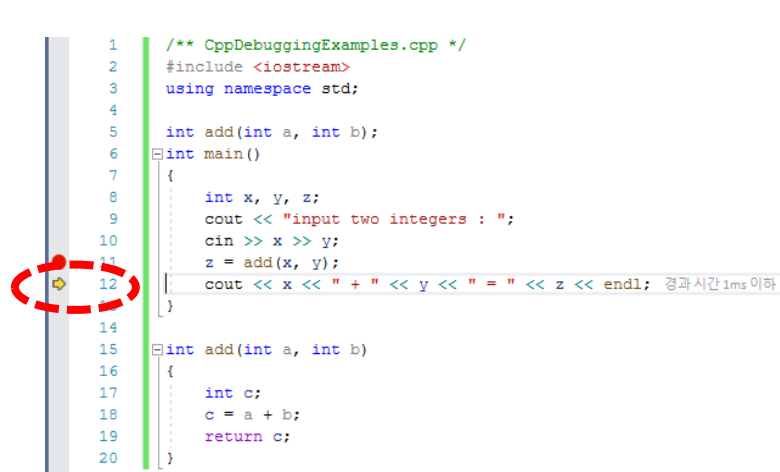


```
1  /** CppDebuggingExamples.cpp */
2  #include <iostream>
3  using namespace std;
4
5  int add(int a, int b);
6  int main()
7  {
8      int x, y, z;
9      cout << "input two integers : ";
10     cin >> x >> y;
11     z = add(x, y);
12     cout << x << " + " << y << " = " << z << endl;
13 }
14
15 int add(int a, int b)
16 {
17     int c;
18     c = a + b;
19     return c;
20 }
```

Visual Studio Debugging tool 사용법(5)

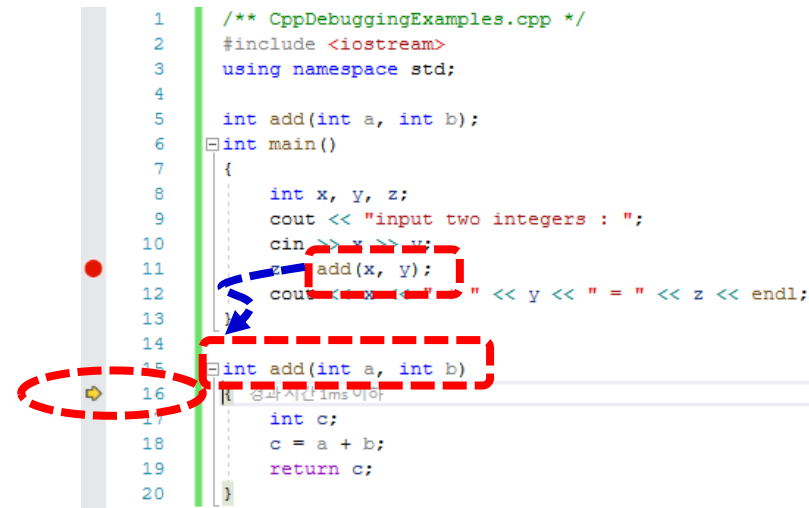
◆ step over (F10) / step into (F11)

- Debugging 중 함수호출 라인을 만나게 되면 두 가지 선택이 가능함
 - (a) step over : 호출된 함수 내부 코드를 디버깅하지 않고, 함수 호출 바로 다음 line 진행
 - (b) step into : 호출된 함수 내부 코드로 진행하여, 디버깅



```
1  /** CppDebuggingExamples.cpp */
2  #include <iostream>
3  using namespace std;
4
5  int add(int a, int b);
6  int main()
7  {
8      int x, y, z;
9      cout << "input two integers : ";
10     cin >> x >> y;
11     z = add(x, y);
12     cout << x << " + " << y << " = " << z << endl; 경과시간 1ms 이하
13 }
14
15 int add(int a, int b)
16 {
17     int c;
18     c = a + b;
19     return c;
20 }
```

(a) F10을 눌러 step over를 실행한 경우



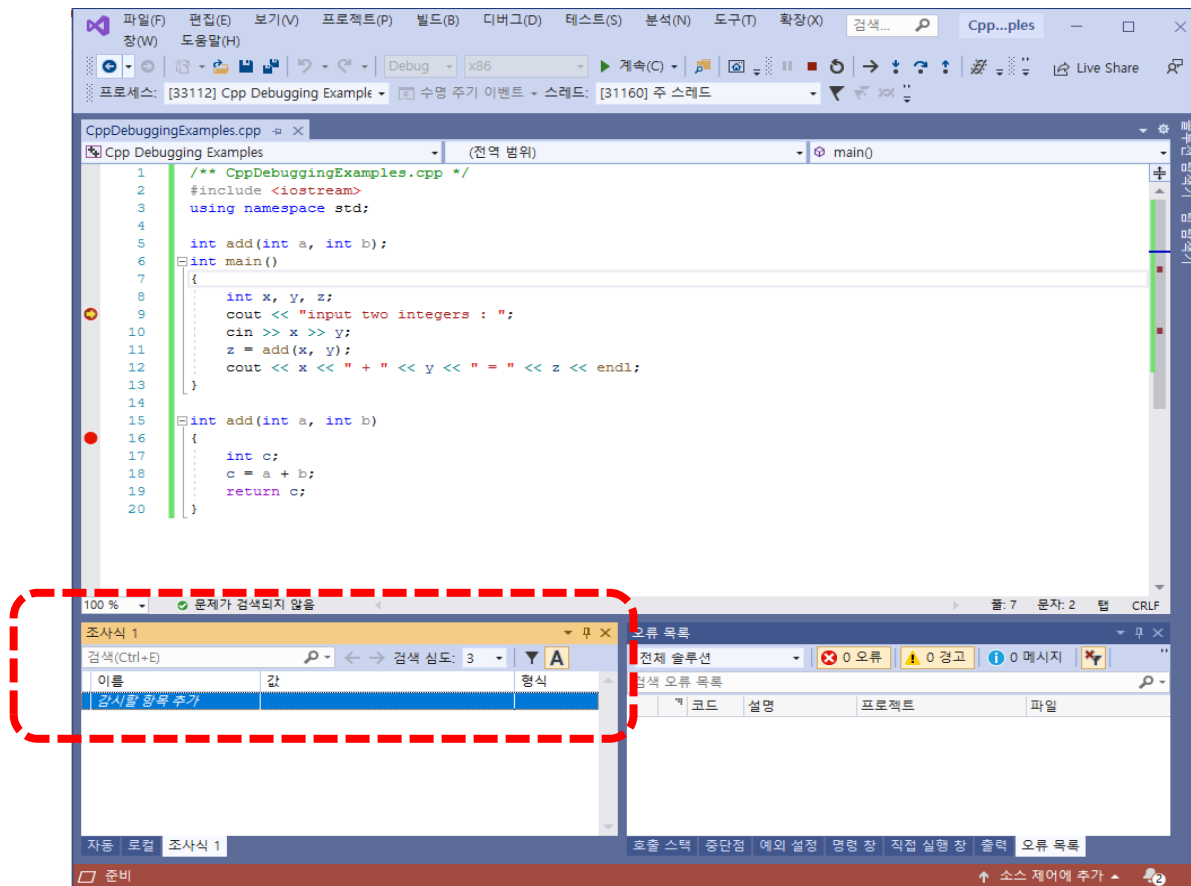
```
1  /** CppDebuggingExamples.cpp */
2  #include <iostream>
3  using namespace std;
4
5  int add(int a, int b);
6  int main()
7  {
8      int x, y, z;
9      cout << "input two integers : ";
10     cin >> x >> y;
11     z = add(x, y);
12     cout << x << " + " << y << " = " << z << endl;
13 }
14
15 int add(int a, int b)
16 {
17     int c;
18     c = a + b;
19     return c;
20 }
```

(b) F11을 눌러 step into를 실행한 경우

Visual Studio Debugging tool 사용법(6)

◆ 변수 (Variables) 값 확인하기

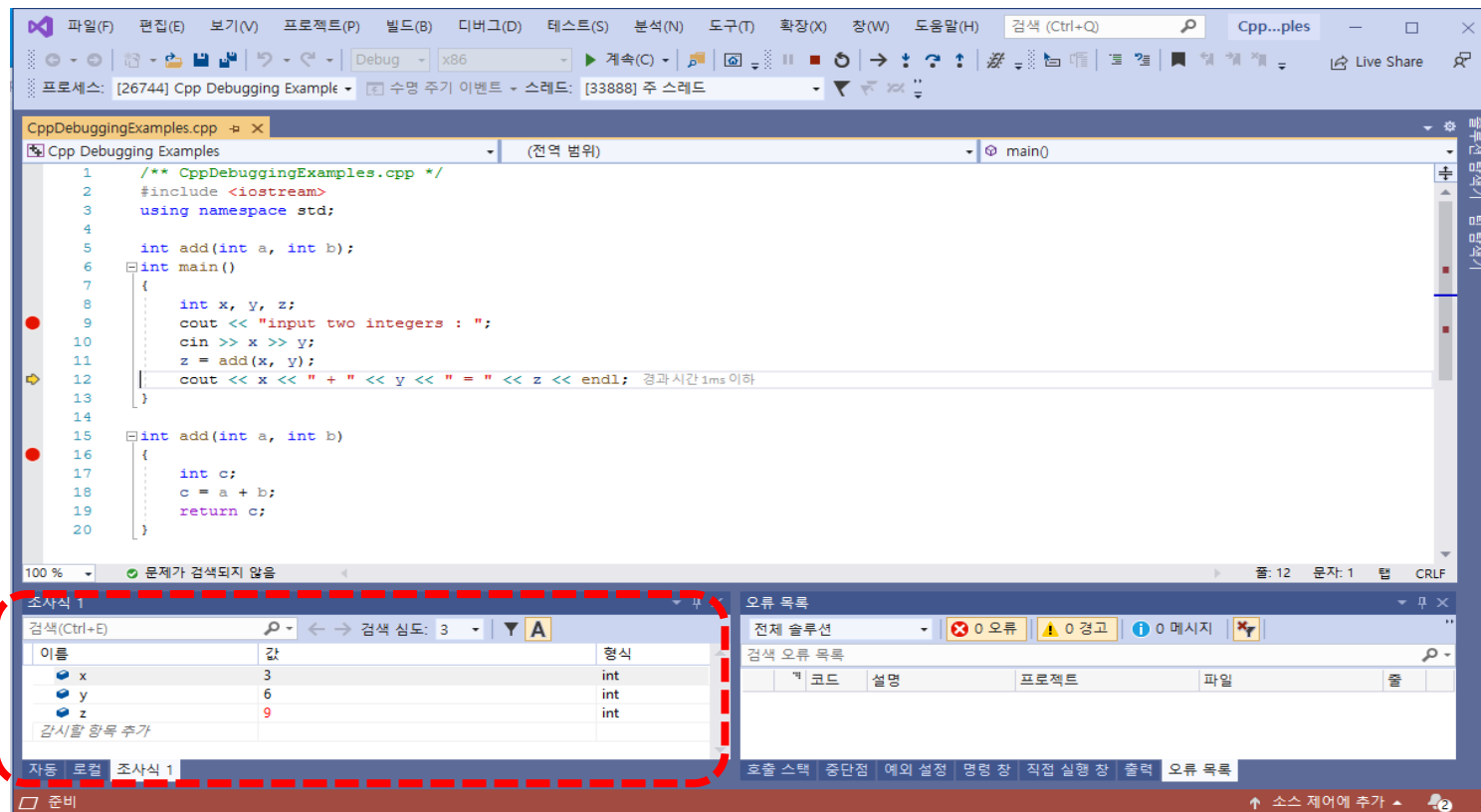
- 조사식 출력창에서 감시 (monitoring)가 필요한 변수의 이름 설정



Visual Studio Debugging tool 사용법(8)

◆ 조사식

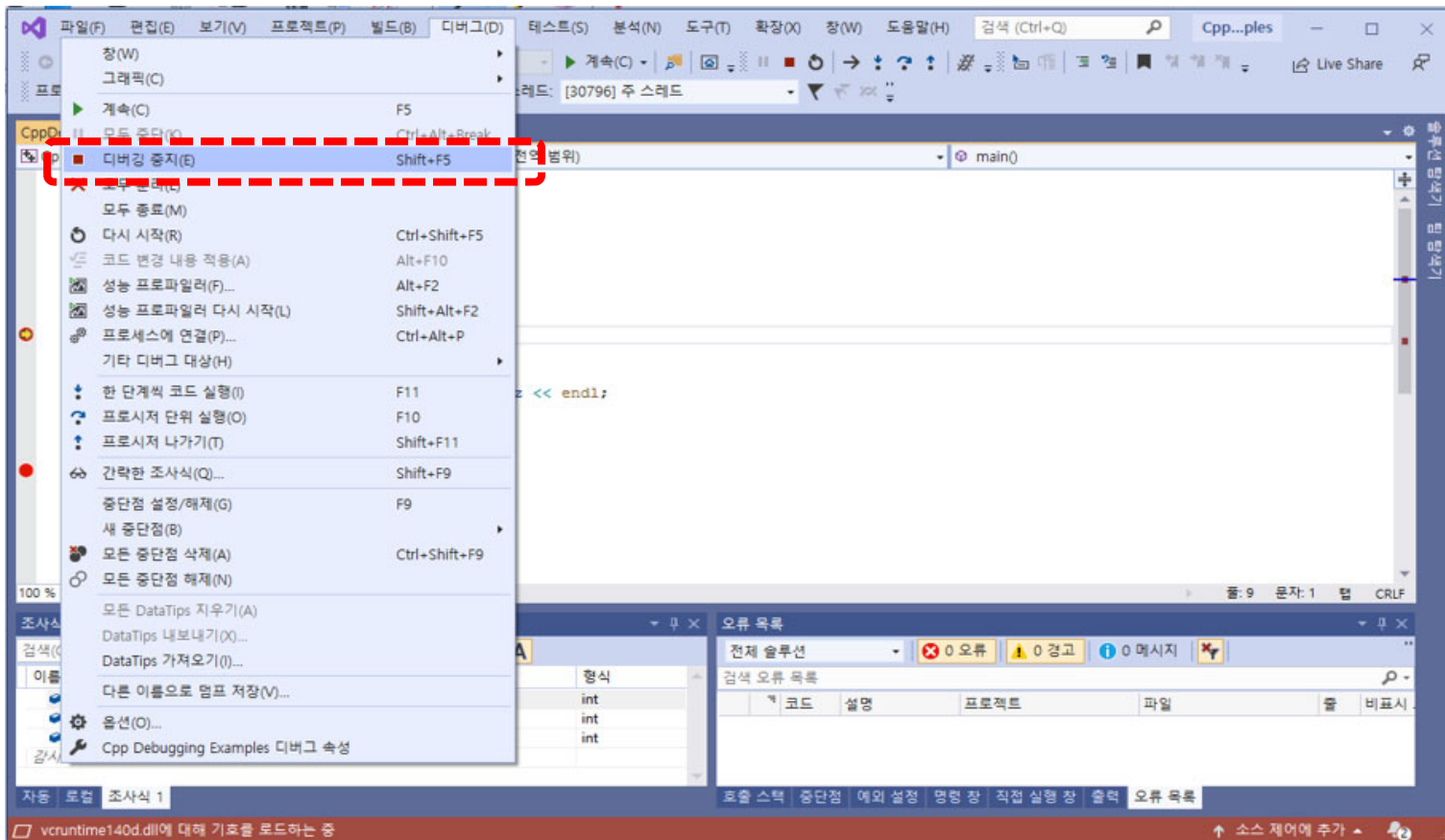
- 조사식 창에서 감시 대상 항목들을 추가하고, 지정된 변수들 값의 변화를 관찰 할 수 있음



Visual Studio Debugging tool 사용법(8)

◆ Debugging 모드 종료

- 디버그 탭 -> 디버깅 중지 (Shift+F5)을 누르면 디버깅 모드 종료



Homework 1

Homework 1

1.1 파일 입력 및 데이터 통계 분석

- 입력 데이터 파일 (input.dat)로부터 정수 데이터를 입력 받아 int 형 배열에 차례대로 저장하고, 이들 정수 데이터의 최댓값, 최솟값, 평균값, 분산 및 표준편차를 계산하는 C++ 프로그램을 작성하라.
- 10개의 정수 데이터를 입력 받는 기능은 `int fileInputData(ifstream& fin, int *data_array, int max_num_data)` 함수로 구현하라. `fileInputData()` 함수는 입력된 데이터 개수를 반환한다.
- 입력된 데이터의 통계 분석 기능은 `getStatistics_fileOut(int *data_array, int num_data, ofstream& fout)` 함수로 구현할 것.
- 입력된 데이터와 계산된 통계 데이터는 출력파일 (output.txt)에 출력할 것.
- 입력 데이터 파일 예시:

```
output.txt  input_data.txt  FileIO_and_Statistics.cpp
1 3 1 9 8 2 4 7 5 6 10
```

- 출력 예시:

```
output.txt  input_data.txt  FileIO_and_Statistics.cpp
1 Total 10 input from input data file.
2 Total 10 integer data : {3, 1, 9, 8, 2, 4, 7, 5, 6, 10, }
3 min(1), max(10), sum(55), avg(5.5), var(8.25), std_dev(2.87228)
4
```



Homework 1

1.2 정수 (integer) 1 ~ 32 출력 포맷 지정

- 정수 (integer) 1 ~ 32를 십진수, 8진수, 16진수, 2진수로 출력하라. 각 진수의 접두어 (8진수는 0, 16진수는 0X)를 표시 하도록 할 것.
- 출력 예는 다음 그림과 같이 10진수, 8진수, 16진수, 2진수 순서로 설정할 것.

0	0	0	00000000
1	01	0X1	00000001
2	02	0X2	00000010
3	03	0X3	00000011
4	04	0X4	00000100
5	05	0X5	00000101
6	06	0X6	00000110
7	07	0X7	00000111
8	010	0X8	00001000
9	011	0X9	00001001
10	012	0XA	00001010
11	013	0XB	00001011
12	014	0XC	00001100
13	015	0XD	00001101
14	016	0XE	00001110
15	017	0XF	00001111
16	020	0X10	00010000
17	021	0X11	00010001
18	022	0X12	00010010
19	023	0X13	00010011
20	024	0X14	00010100
21	025	0X15	00010101
22	026	0X16	00010110
23	027	0X17	00010111
24	030	0X18	00011000
25	031	0X19	00011001
26	032	0X1A	00011010
27	033	0X1B	00011011
28	034	0X1C	00011100
29	035	0X1D	00011101
30	036	0X1E	00011110
31	037	0X1F	00011111
32	040	0X20	00100000

계속하려면 아무 키나 누르십시오 . . .

