

**CODINGO** x **posco**

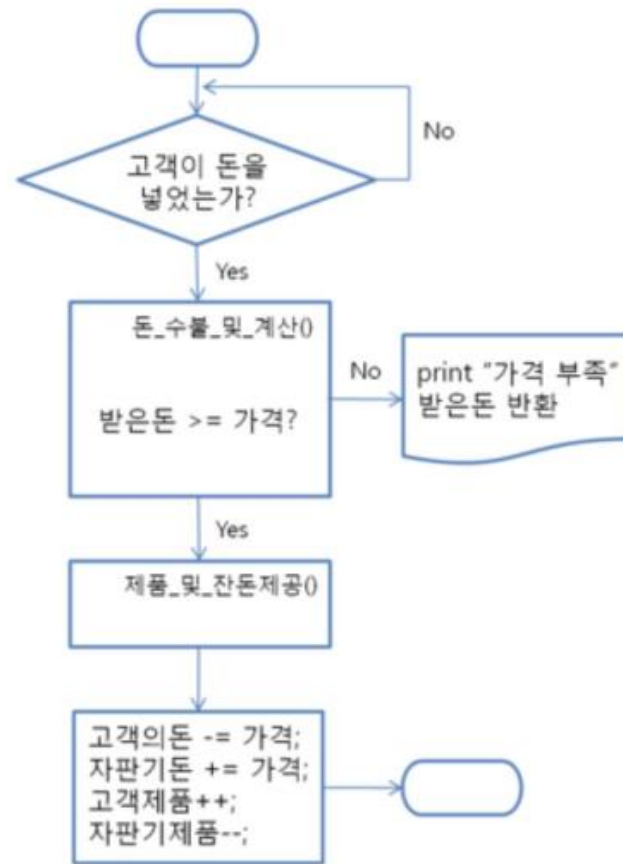
**K-Digital Training** 스마트 팩토리 3기

# C++ 이란?

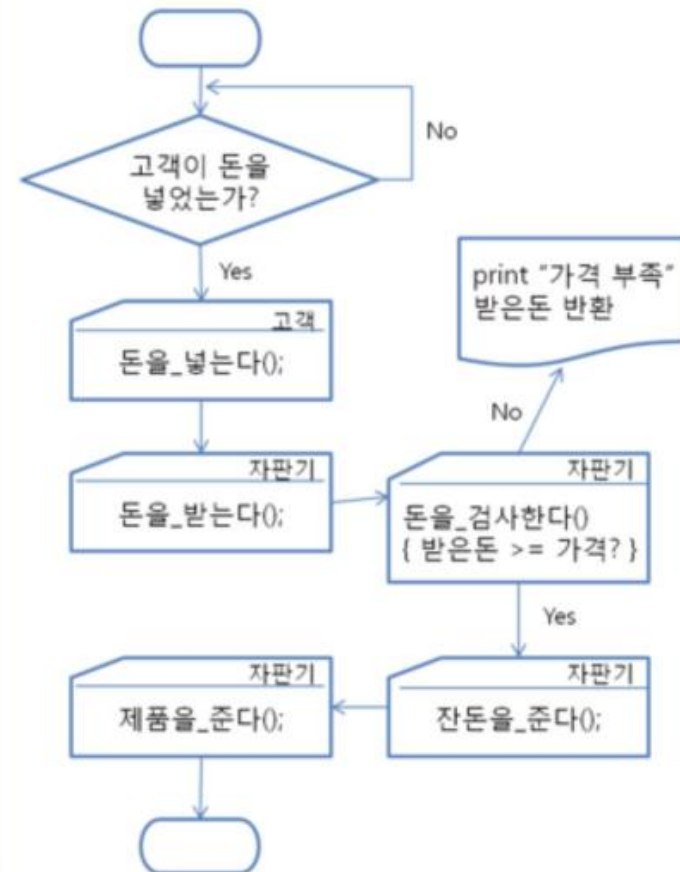
- C 언어의 확장판에서 시작된 언어
  - 절차 지향 언어인 C 와 달리 객체 지향 언어
  - 객체 지향적이기 때문에 구조화된 프로그램을 짤 수 있음
- 객체 지향 개념을 도입하여 C언어에 비해서 효율성 저하를 최소화
- 타입체크가 엄격
  - 실행 시간 오류의 가능성을 줄이고 디버깅을 돕습니다.
- 이식성이 좋습니다.
  - 다양한 운영체제에서 사용할 수 있는 언어

# 절차지향? 객체지향?

절차지향 방식



객체지향 방식



# C++ 활용 분야

- 활용 분야
  - 임베디드
  - 금융, 통신 애플리케이션
  - 서버 구축
  - 검색엔진
  - ...

# Mac 사용자 환경 설정

<https://nadocoding.tistory.com/95>

1. Homebrew 설치
2. Visual Studio Code 설치
3. Extension 설치

Visual Studio



# Visual Studio?

- 통합 개발 환경(IDE)
  - 소스 코드 편집, 디버깅(테스트), 빌드를 할 수 있는 툴
- IDE 별로 지원하는 운영체제가 다르지만 visual studio code는 MacOS와 windows 운영체제에서 지원됨
- Visual Studio, eclipse, intelliJ(JAVA), Clion(C++/C)



# Visual Studio 설치

<https://visualstudio.microsoft.com/ko/>

# Visual Studio 설치



## Visual Studio | 🖥️

Windows에서 .NET 및 C++ 개발자를 위한 최고의 포괄적인 IDE입니다. 소프트웨어 개발의 모든 단계를 향상시키고 개선할 수 있는 다양한 도구와 기능이 완벽하게 포함되어 있습니다.

[자세히 보기 >](#)

Visual Studio 다운로드 ▾

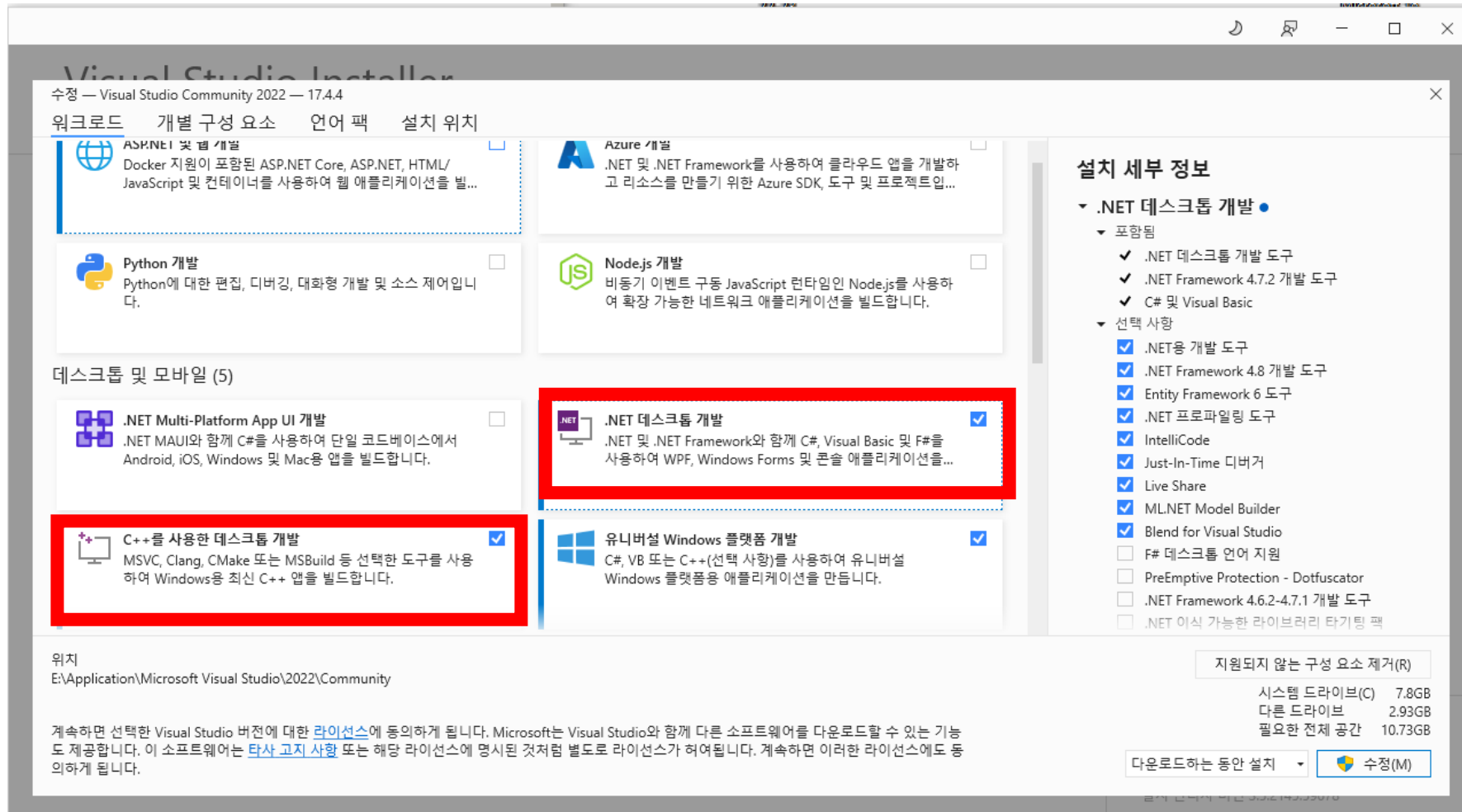
Community 2022

Professional 2022

Enterprise 2022

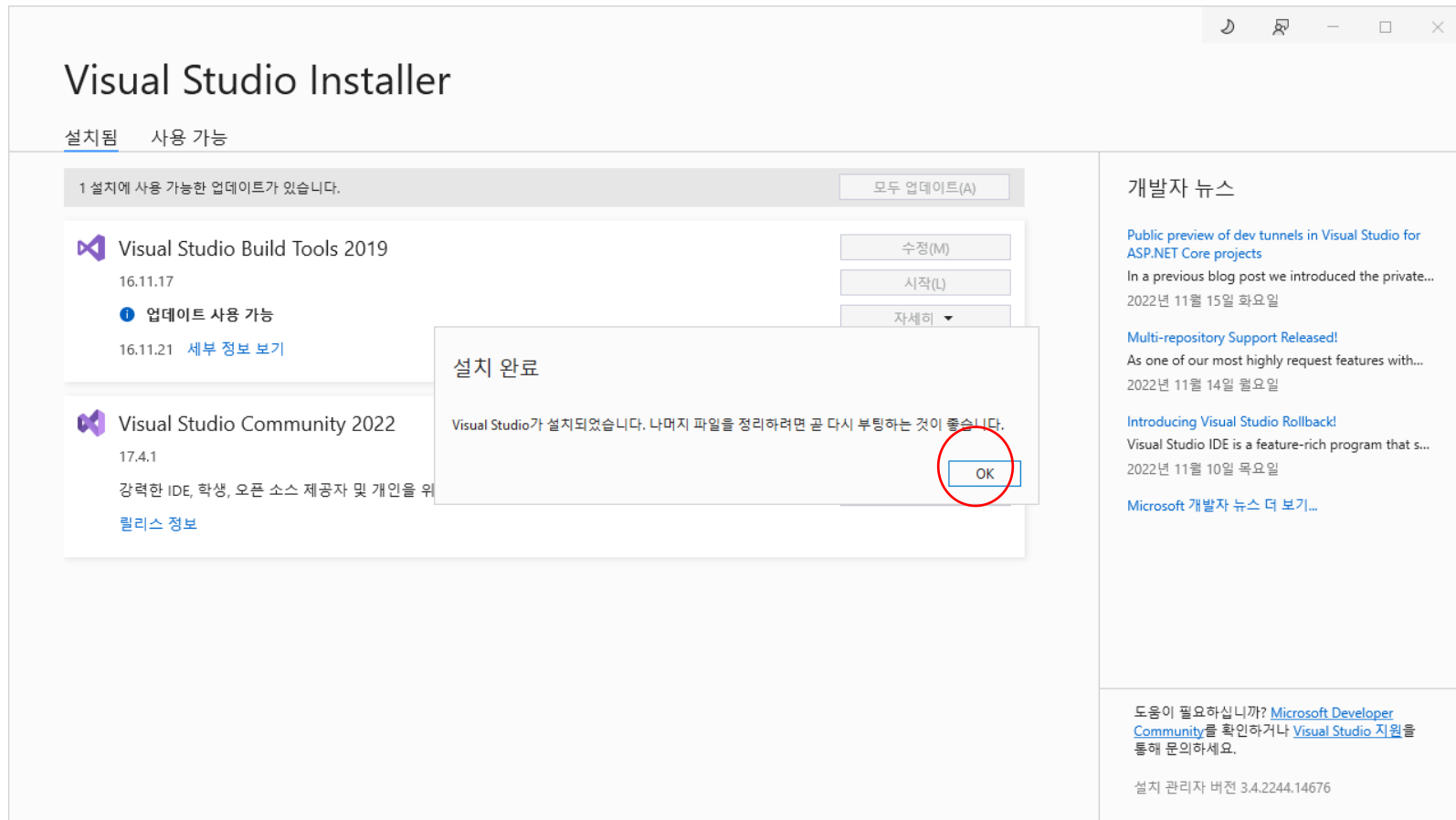
어떤 도구가 가장 적합한지 잘 모르시겠

# Visual Studio 설치

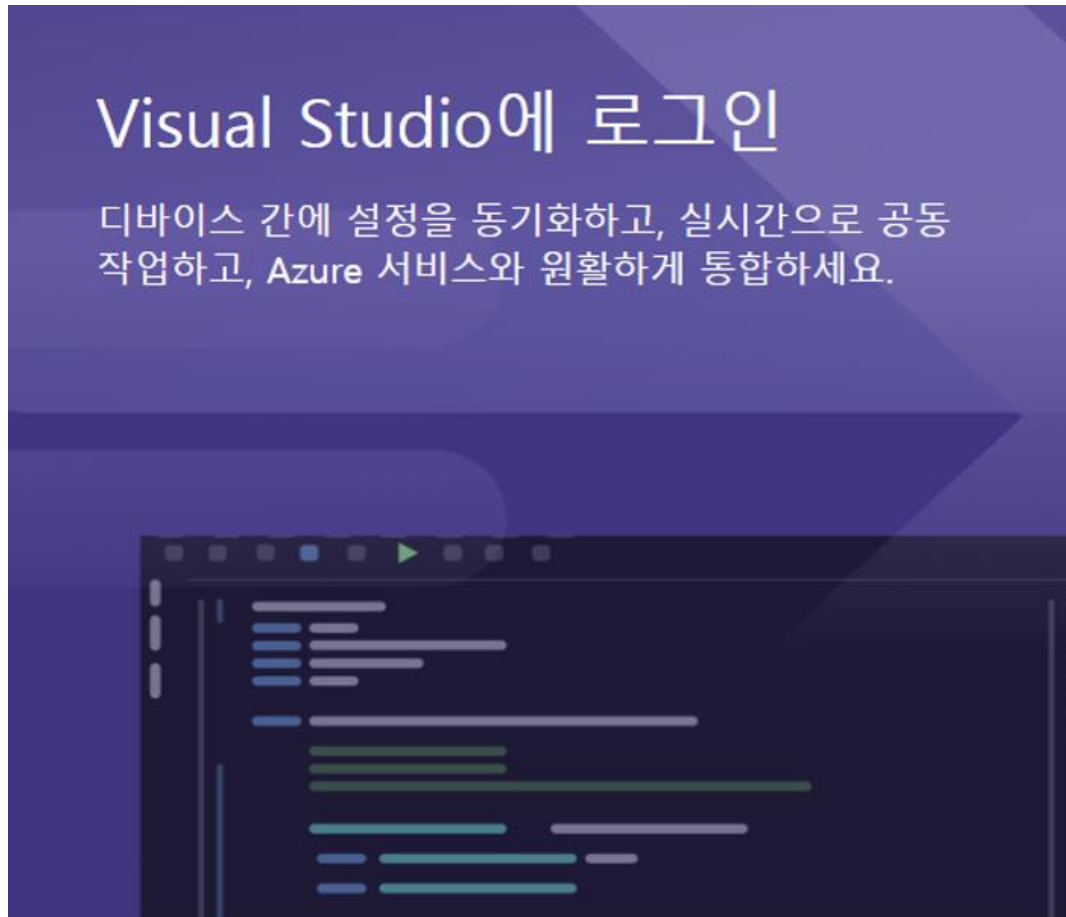


# Visual Studio 설치

## 재부팅 하기!



# Visual Studio 실행



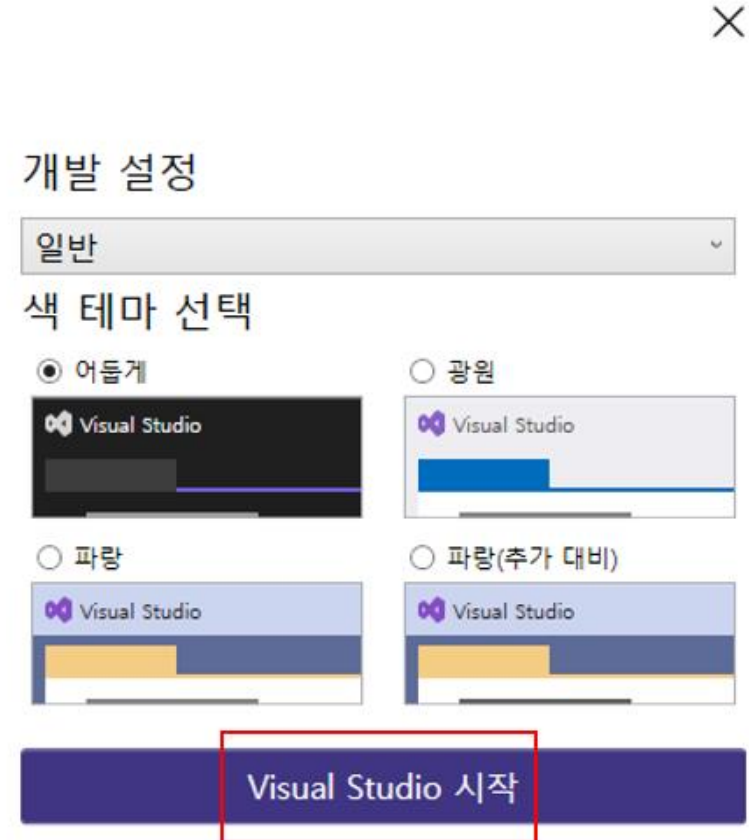
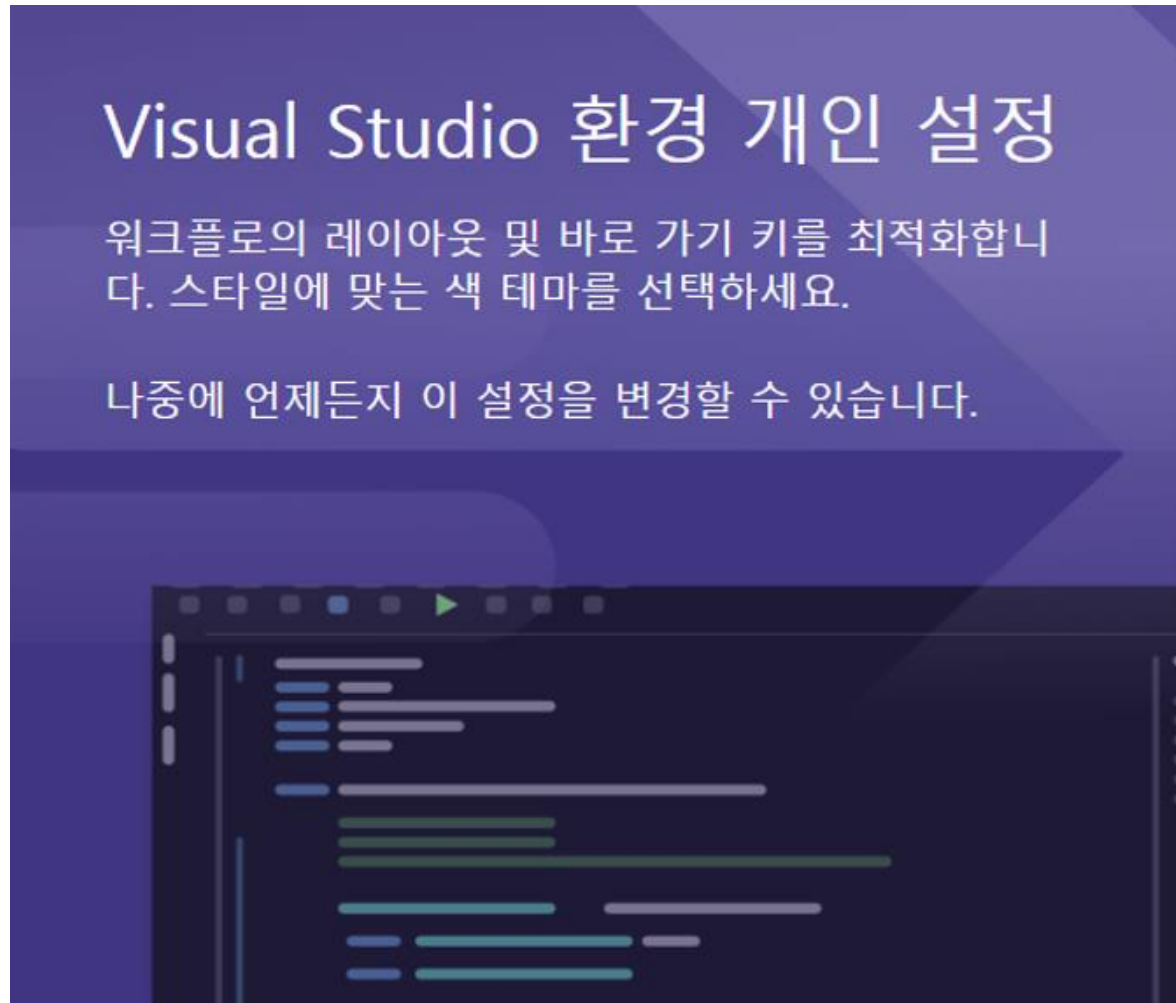
×

로그인

계정 만들기

지금은 이 항목을 건너뛰니다.

# Visual Studio 실행



# Visual Studio 2022

## 최근 파일 열기(R)

최근 항목 검색(Alt+S)(S)



▶ 이번 달

## 시작



### 리포지토리 복제(C)

GitHub 또는 Azure DevOps 같은 온라인 리포지토리에서 코드 가져오기



### 프로젝트 또는 솔루션 열기(P)

로컬 Visual Studio 프로젝트 또는.sln 파일 열기



### 로컬 폴더 열기(F)

폴더 내에서 탐색 및 코드 편집



### 새 프로젝트 만들기(N)

시작하려면 코드 스캐폴딩과 함께 프로젝트 템플릿을 선택하세요.

코드를 사용하지 않고 계속(W) →

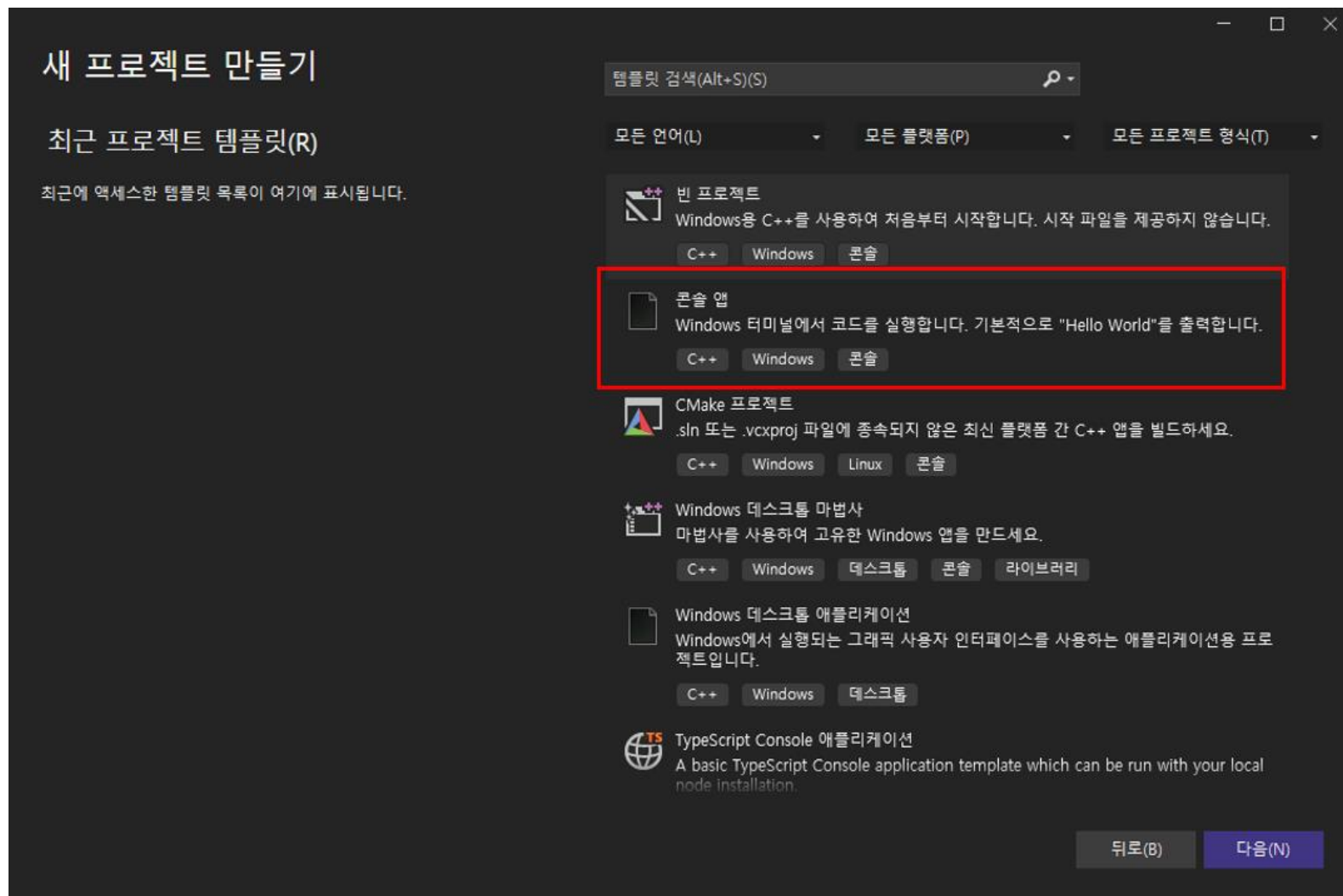
C++ 프로젝트



# C++ 프로젝트

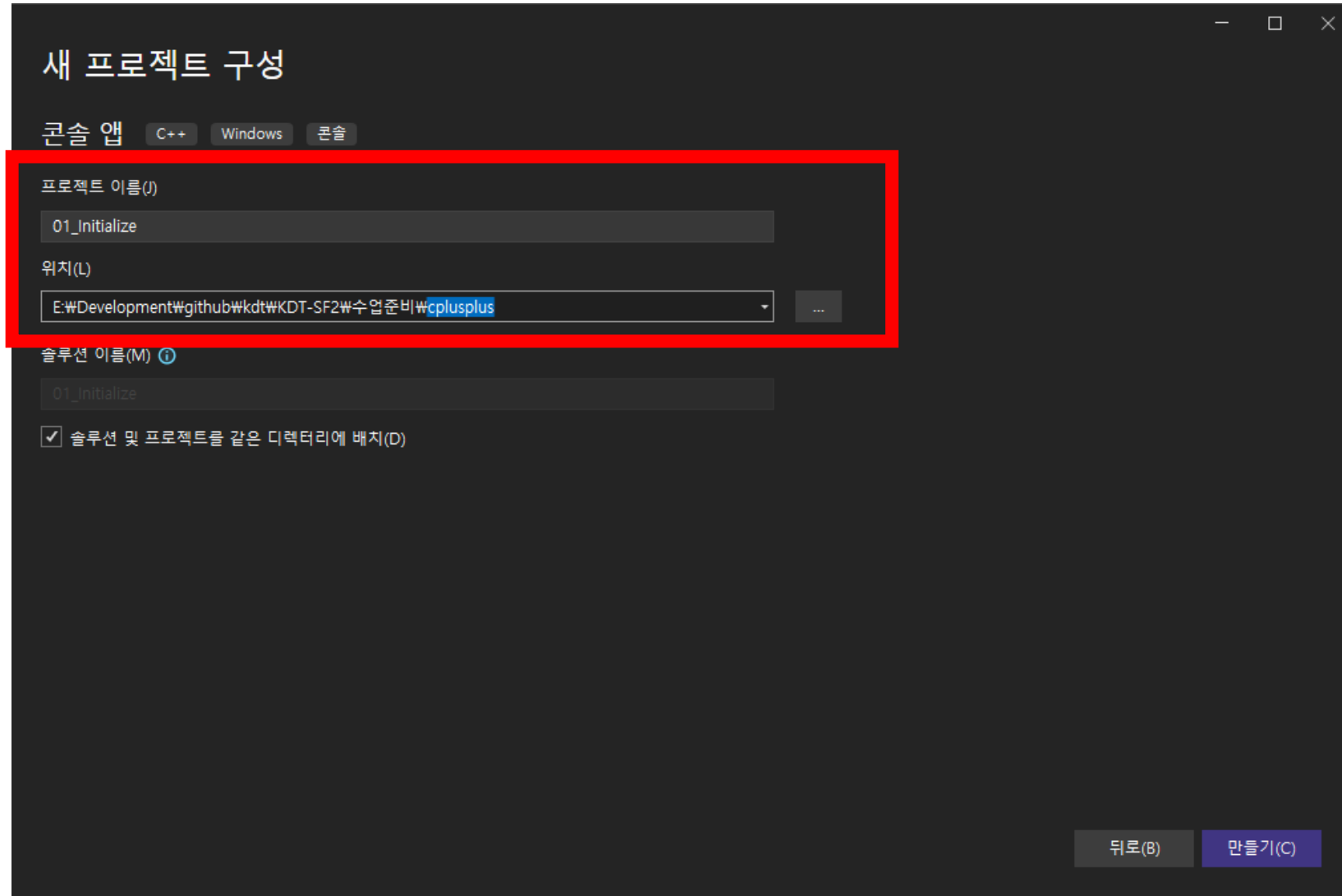


# C++ 프로젝트

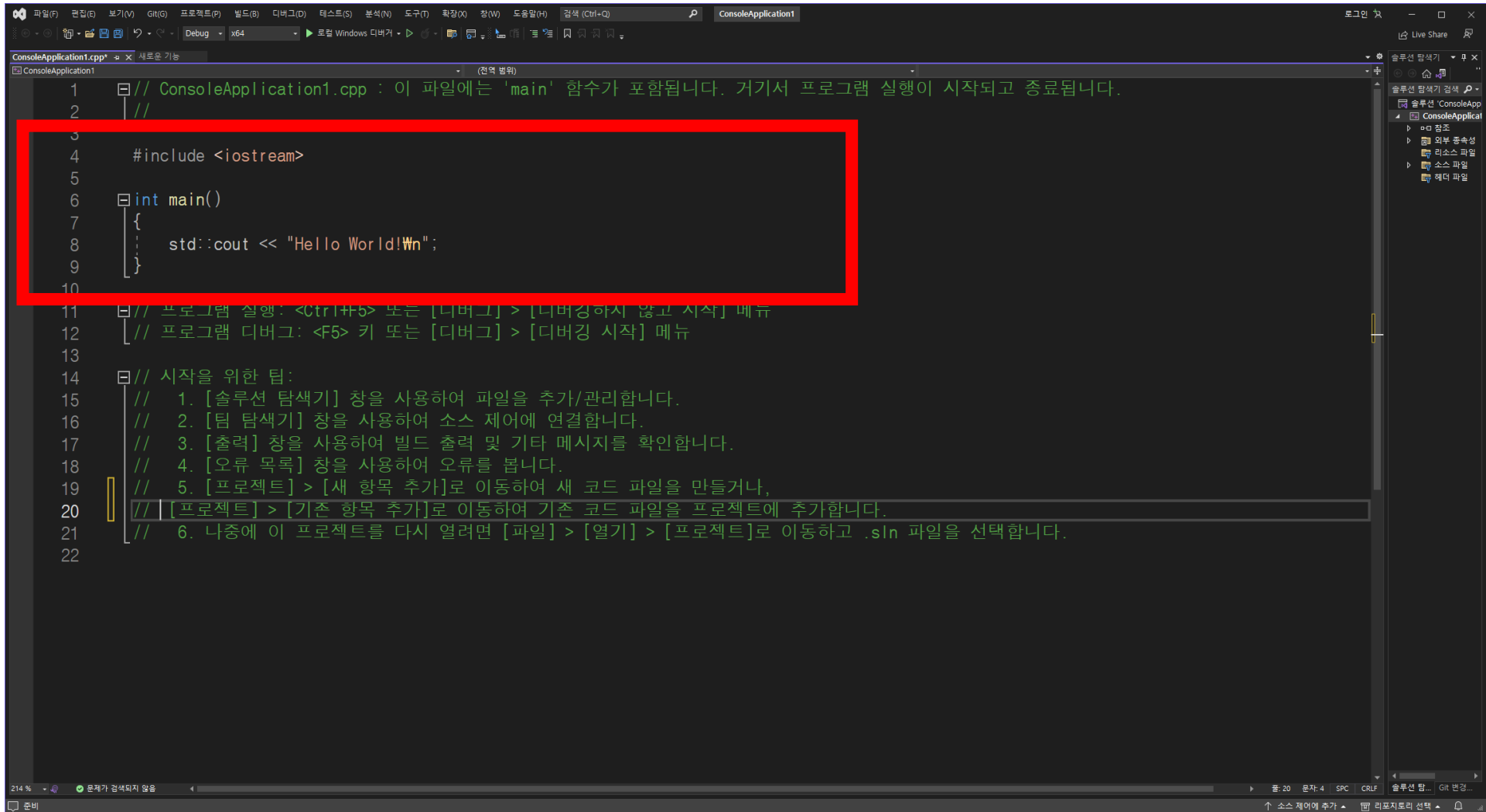


# C++ 프로젝트

- 프로젝트 이름은  
숫자와 영어로만!
- 경로는 바탕화면의  
Github 폴더!



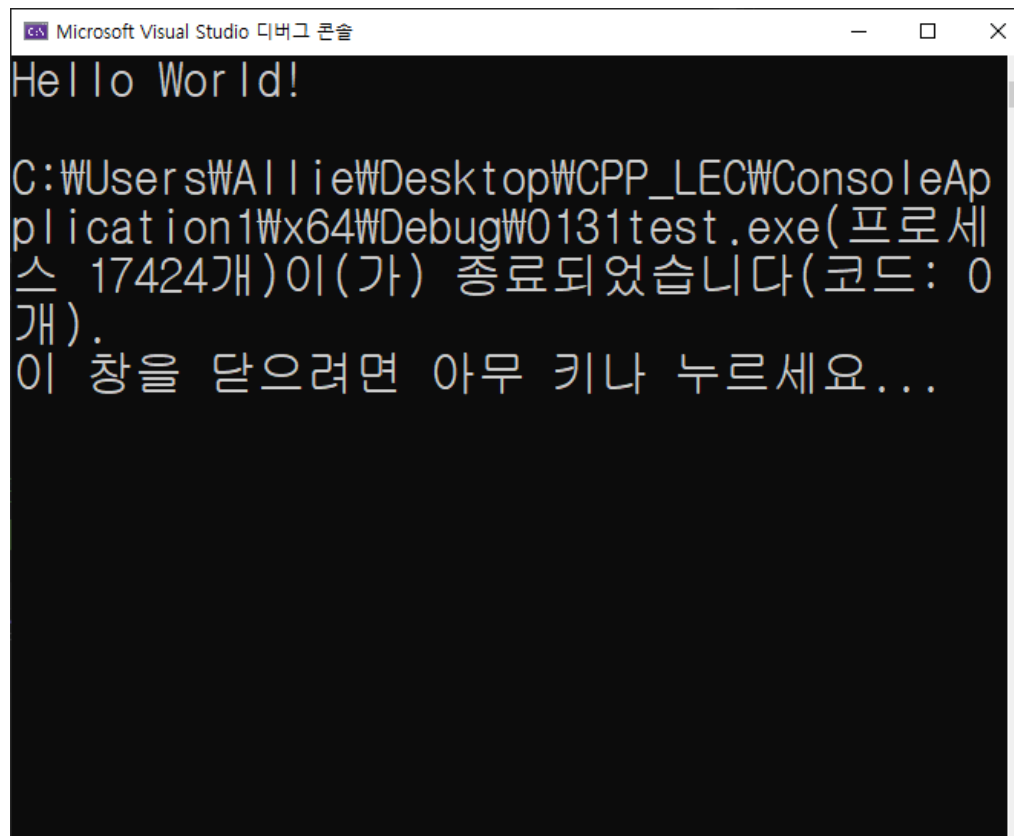
# C++ 프로젝트



```
1 // ConsoleApplication1.cpp : 이 파일에는 'main' 함수가 포함됩니다. 거기서 프로그램 실행이 시작되고 종료됩니다.
2 //
3
4 #include <iostream>
5
6 int main()
7 {
8     std::cout << "Hello World!\\n";
9 }
10
11 // 프로그램 실행: <Ctrl+F5> 또는 [디버그] > [디버깅하지 않고 시작] 메뉴
12 // 프로그램 디버그: <F5> 키 또는 [디버그] > [디버깅 시작] 메뉴
13
14 // 시작을 위한 팁:
15 // 1. [솔루션 탐색기] 창을 사용하여 파일을 추가/관리합니다.
16 // 2. [탐색기] 창을 사용하여 소스 제어에 연결합니다.
17 // 3. [출력] 창을 사용하여 빌드 출력 및 기타 메시지를 확인합니다.
18 // 4. [오류 목록] 창을 사용하여 오류를 봅니다.
19 // 5. [프로젝트] > [새 항목 추가]로 이동하여 새 코드 파일을 만들거나,
20 // [프로젝트] > [기존 항목 추가]로 이동하여 기존 코드 파일을 프로젝트에 추가합니다.
21 // 6. 나중에 이 프로젝트를 다시 열려면 [파일] > [열기] > [프로젝트]로 이동하고 .sln 파일을 선택합니다.
22
```

# 코드를 실행해볼까요?

- 콘솔 창에 Visual Studio 에 있던 **Hello World** 가 출력되게 됩니다.



```
Microsoft Visual Studio 디버그 콘솔
Hello World!
C:\Users\WAllie\Desktop\WCPP_LEC\WConsoleApplication1\Wx64\Debug\W0131test.exe(프로세스 17424개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

# C++ 파헤치기

# # include <iostream>

- #
  - 컴파일 시작 전 미리 처리하기 위한 **전처리기** 를 의미
- #include
  - 외부에 선언되어 있는 함수 or 상수 등을 사용하기 위해 선언
- #include <iostream>
  - `iostream` : 스트림을 이용한 입출력을 제공하는 객체 지향 라이브러리

# int main()

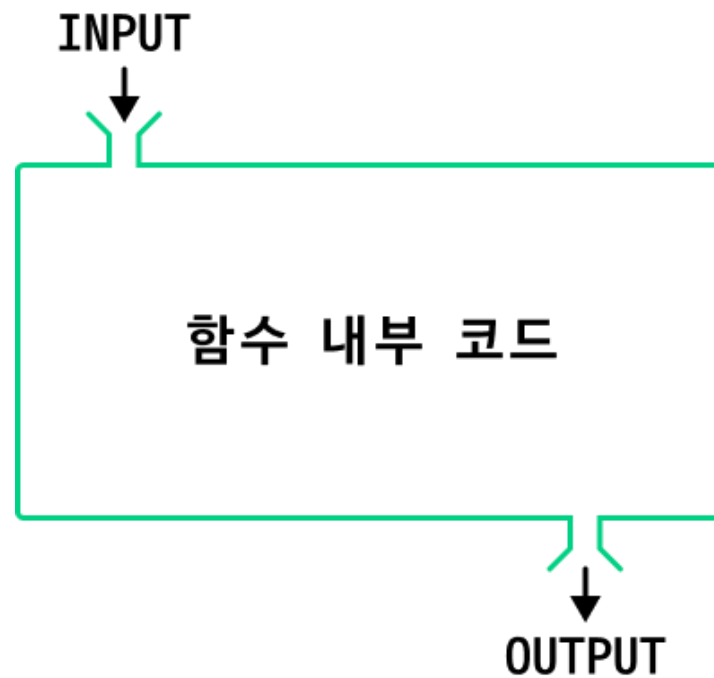
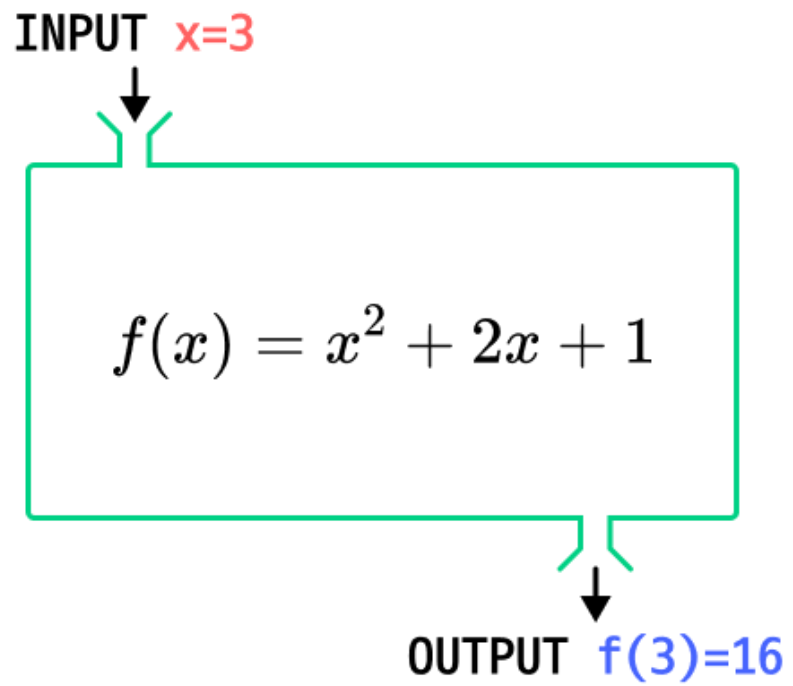
- 프로그램이 시작될 때 가장 먼저 호출되는 함수
- main 함수는 int ( 정수 ) 를 Return 하는 함수이다.
- C++ 프로젝트에는 main() 함수가 필수로 존재해야 하며, return 이 없으면 기본적으로 0 을 return 해준다.

```
int main()  
{  
    std::cout << "Hello World!\n";  
}
```

```
int main()  
{  
    std::cout << "Hello World!\n";  
    return 0;  
}
```



# 함수



# C++ 함수

```
return타입 함수명()  
{  
  
}
```

- return타입은 변수의 타입 중 하나!
- 함수 더 자세한 건 다다음 시간에.....

# 입/출력

- std::cout <<

- console out 이라는 의미로 콘솔에 값을 출력해준다.

```
std::cout << "Hello World!\n";
```

- std::cin >>

- console in 이라는 의미로 콘솔이 열렸을 때 원하는 값을 입력할 수 있게 해준다.
  - 값을 입력받으면 변수에 넣어줘야 한다.

# 변수와 자료형

# 변수

- 데이터를 저장하기 위해 할당받은 공간

Identifier	Memory	
	Address	Value
myNumber	0012CCGWH80	23

자료형 변수명;

자료형 변수명 = 값;

# 변수 네이밍 규칙

1. 변수의 이름은 영문자(대소문자), 숫자, \_, \$ 로만 구성된다.
2. 변수의 이름은 숫자로 시작할 수 없다.
3. 변수의 이름 사이에는 공백이 존재하면 안 된다.
4. 변수의 이름으로 예약어를 사용할 수 없다.
5. 변수 이름은 길이의 제한이 없다.

# 예약어란?

- C++ 에 미리 정의되어 있는 특별한 의미가 있는 단어

alignas **	char	do	goto	operator	static	typeid *
alignof **	char16_t **	double	if	or	static_assert **	typename *
and	char32_t **	dynamic_cast *	inline	or_eq	static_cast *	union
and_eq	class	else	int	private	struct	unsigned
asm	compl	enum	long	protected	switch	using *
auto	const	explicit *	mutable *	public	template	virtual
bitand	constexpr **	export *	namespace *	register	this	void
bitor	const_cast *	extern	new	reinterpret_cast *	thread_local **	volatile
bool *	continue	false *	noexcept **	return	throw	wchar_t *
break	decltype **	float	not	short	true *	while
case	default	for	not_eq	signed	try	xor
catch	delete	friend	nullptr **	sizeof	typedef	xor_eq

# 언어 타입

## 강한 타입 언어

타입 검사를 통과하지 못한다면 실행 자체가 안 된다.

String, int, double 등처럼 타입을 1종류로 명확히 지정

## 약한 타입 언어

런타임에서 타입 오류를 만나더라도 실행을 막지 않는다.

타입이 여러 종류인 값들이 상관없이 지정된다.



# 언어 타입

강한 타입 언어 ( Strong )	약한 타입 언어 ( Weak )
Java, C, C++, C# .....	Javascript, Python .....

# C++ 는 강한 언어!

- C++ 는 데이터 종류를 명확하게 지정해줘야 한다.
- `int a = "안녕" ( X )`
- `int a = 1 (O)`

# 기본 자료형 ( Data Type )

자료형	용량	분류
bool	1바이트	불형
char	1바이트	문자형
wchar_t	2바이트	
short	2바이트	정수형
int	4바이트	
long	4바이트	
float	4바이트	실수형
double	8바이트	
long double	16바이트	

# 기본 자료형 – int, float

```
int a = 1;  
int b = 2;  
std::cout << a + b;  
  
std::cout << "Wn";
```

```
float f = 1.2;  
float g = 3.9;  
  
std::cout << f + g;
```

# 기본 자료형 - bool

```
bool c1 = true;  
std::cout << c1;  
std::cout << "\n";
```

```
bool c2 = false;  
std::cout << c2;  
std::cout << "\n";
```

# 기본 자료형 - char

```
char d = 'a';  
std::cout << d;  
std::cout << "\n";
```

```
char e = 'b';  
std::cout << e;  
std::cout << "\n";
```

# 문자열 std::string

```
std::string 변수명;  
std::string 변수명 = "내용";
```

- 문자열을 이용할 때는 std::string 타입을 이용해야 한다.
- 문자는 항상 " " 를 이용해 쌍따옴표로 감싸주기!!

```
std::string str1 = "안녕?";  
std::cout << str1;  
std::cout << "\n";  
  
std::string str2 = "반가워";  
std::cout << str2;
```

## 깜짝 질문 No.1 !

```
std::string test1 = "1";  
std::string test2 = "2";  
  
std::cout << test1 + test2;
```

위의 코드를 실행시켰을 때 나오는 결과는?

① 3

② 12

③ 오류가 나온다.



## 깜짝 질문 No.2 !

```
std::string test3 = "포스코";  
test3 = "코딩온";  
  
std::cout << test3;
```

위의 코드를 실행시켰을 때 나오는 결과는?

① 포스코

② 코딩온

③ 오류가 나온다.

# 변수 입력받기

```
int a;  
std::string b;  
  
std::cin >> a;  
std::cin >> b;
```

- std::cin 입력은 띄어쓰기를 기준으로 입력이 된다.
- 타입에 맞게 입력할 것!
  - a 1 은? 정상적으로 동작하지 않는다!
  - 1 a 는? 정상적으로 동작!

# 변수 출력하기

```
int a;  
std::string b;  
std::string c;  
  
std::cout << a << b;  
std::cout << b + c;
```

- 타입이 다를 경우?
  - << 로 연결하기
- 타입이 같을 경우?
  - + 로 연결하기

연산자

# 연산자

- 대입 연산자: =
- 비교 연산자: ==, !=, >, >=, <, <=
- 산술 연산자: +, -, \*, /
- 논리 연산자: !, &&, ||

# 산술 연산자, 증감 연산자

연산자 종류	의미	사용 예시(모두 int 일 때)
+	덧셈	5 + 5
-	뺄셈	6 - 2
*	곱셈	10 * 2
/	나눗셈	10 / 3
%	나머지 연산자	10 % 3
++	증가	a++; ++b;
--	감소	a--; --b;

# 논리 연산자, 비교 연산자

- 논리 연산자

- ! (NOT)
- &&(AND)
- || (OR)

- 비교 연산자

- < , > , <=, >=
- !=(not equal), ==(equal)

- 이외에도 비트, 비트시프트 , .. 등의 연산자가 있습니다

조건문



# 조건문

특정 조건 만족 시 ( 조건이 참인 경우 ) 실행하는 명령의 집합

특정한 조건 속에서 작업을 수행하고 싶을 때 사용

if

switch

# if 문

만약 ~~라면

```
if ( 조건1 ) {  
    // 조건1이 참이라면 실행  
} else if ( 조건2 ) {  
    // 조건1이 참이 아니고 조건2가 참이라면 실행  
} else {  
    // 조건 1과 2가 모두 참이 아닐 때 실행  
}
```

# if문

## 비교연산자

$a == b$  : a와 b가 동일하면 참

$a != b$  : a와 b가 동일하지 않으면 참

$a < b$  : a가 b보다 작으면 ( b가 a보다 크면 ) 참

$a <= b$  : a가 b보다 작거나 같으면 참

## 논리연산자

$a \&\& b$  : a AND b. a 그리고 b

$a \parallel b$  : a OR b. a 또는 b

# if 문

```
if (a > 10) {  
    실행 코드1;  
}else if(a==5){  
    실행 코드2;  
}else {  
    실행 코드3;  
}
```

```
int a = 5;  
if (a > 10) {  
    std::cout << "a가 10보다 큼니다.";  
}  
else if (a == 5) {  
    std::cout << "a는 5입니다.";  
}  
else {  
    std::cout << "a는 10보다 크지 않고 5가 아닙니다.";  
}
```

# if문 중첩

```
if ( 조건1 ) {  
    if ( 조건2 ) {  
        //실행  
    } else {  
        //실행2  
    }  
}
```

# switch 문

```
switch (변수) {  
    case 값1 :  
        실행코드1;  
        break;  
    case 값2:  
        실행코드2;  
        break;  
}
```

# switch 문

```
switch (변수) {  
    case 값1 :  
        실행코드1;  
        break;  
    case 값2 :  
        실행코드2;  
        break;  
    case 값3:  
        실행코드3;  
        break;  
    default :  
        기본실행코드; // 위의 case에 모두 부합하지 않을 때 실행할 코드를 작성  
}
```

# 삼항연산자

- 간단한 조건문

조건 ? 조건이 참일 때 실행할 코드 : 조건이 거짓일 때 실행할 코드;

ex ) `score=='F' ? std::cout<<"재수강" : std::cout<<"참 잘했어요.";`

- score의 조건에 따라 실행할 코드 작성
- if~else 문과 동작 방법이 일치합니다.



# 단축키

- ctrl + k + c : 주석  
ctrl + k + u : 주석 해제
- alt + 상하 방향키 : 코드 이동
- ctrl + f : 찾기, ctrl + h 바꾸기
- ctrl + k + d : 전체 코드 정렬, ctrl + k + f : 선택 코드 정렬
- ctrl + d : 코드 한줄 복사
- ctrl + alt + 클릭 : 다중 커서
- 솔루션 탐색기 ctrl + alt + L