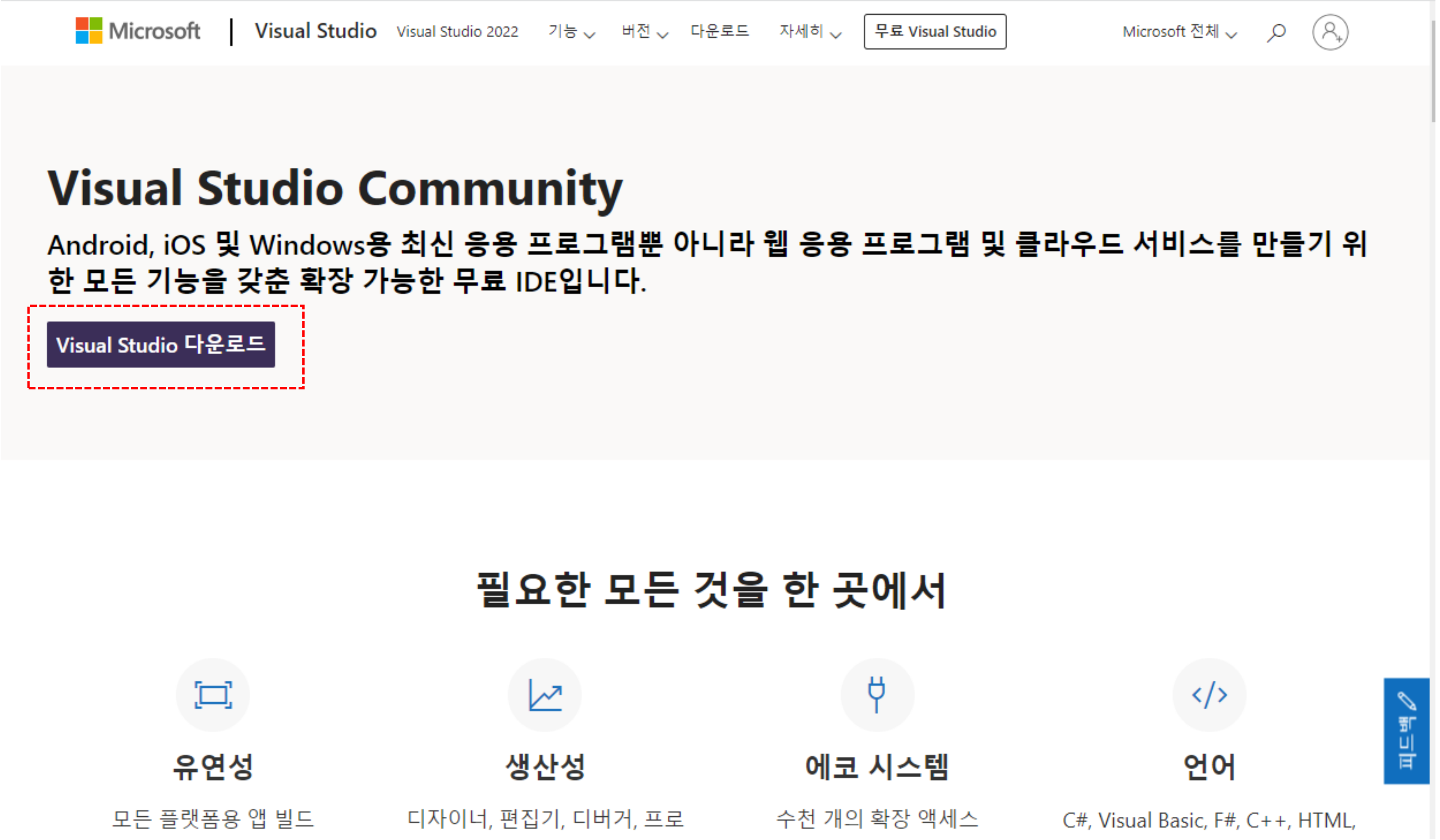
Introduction to Programming (2)

C++ Basics - 1

Laihyuk Park (lhpark@seoultech.ac.kr)

MS Visual Studio

- *Download at*
  - https://visualstudio.microsoft.com/vs/community/



Visual Studio Community

Android, iOS 및 Windows용 최신 응용 프로그램뿐 아니라 웹 응용 프로그램 및 클라우드 서비스를 만들기 위한 모든 기능을 갖춘 확장 가능한 무료 IDE입니다.

필요한 모든 것을 한 곳에서

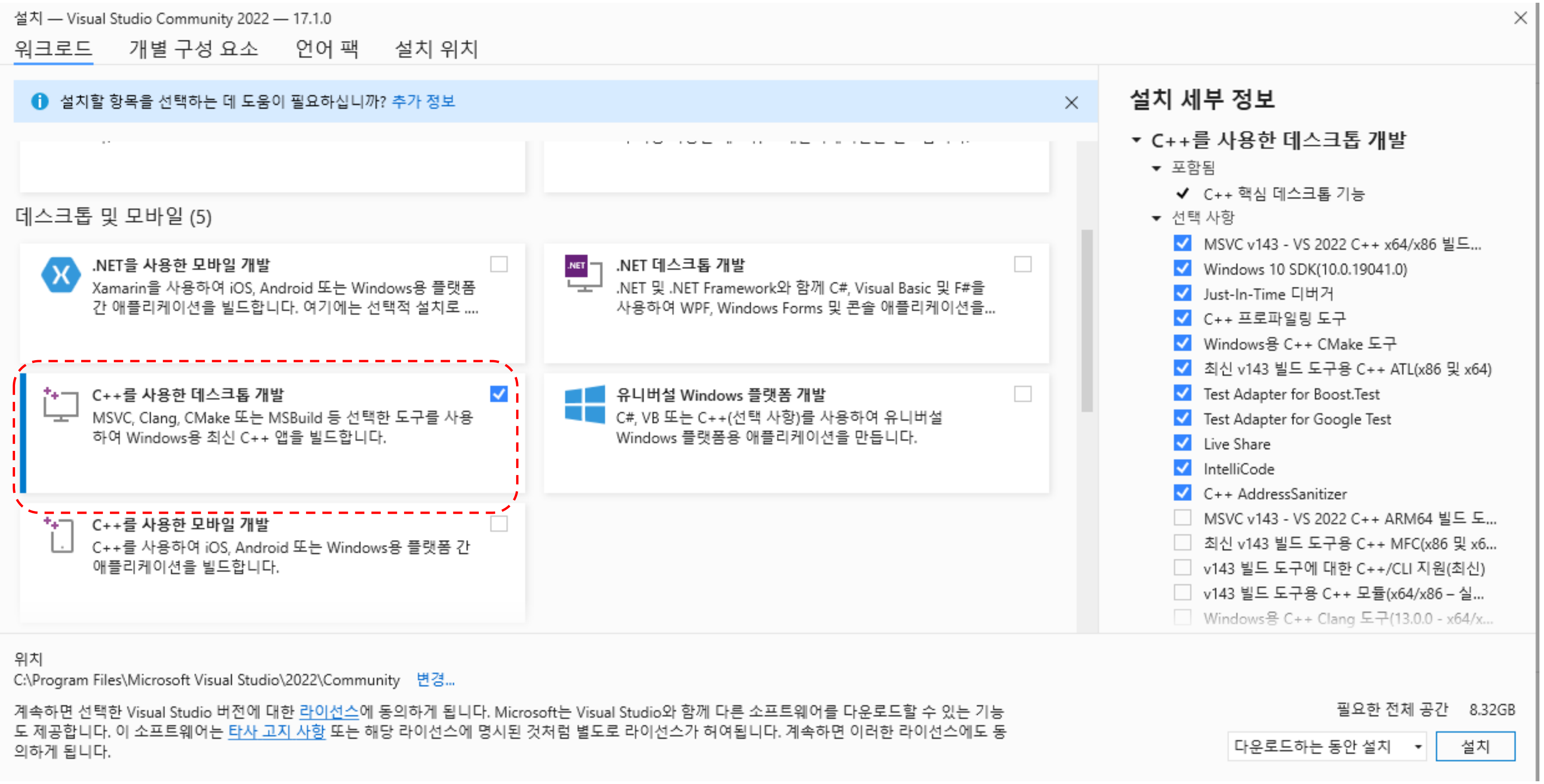
모든 플랫폼용 앱 빌드

디자이너, 편집기, 디버거, 프로

수천 개의 확장 액세스

C#, Visual Basic, F#, C++, HTML,

MS Visual Studio



설치 — Visual Studio Community 2022 — 17.1.0

설치할 항목을 선택하는 데 도움이 필요하십니까? 추가 정보

데스크톱 및 모바일 (5)

☐ **.NET을 사용한 모바일 개발**
Xamarin을 사용하여 iOS, Android 또는 Windows용 플랫폼 간 애플리케이션을 빌드합니다. 여기에는 선택적 설치로 ....

☐ **.NET 데스크톱 개발**
.NET 및 .NET Framework와 함께 C#, Visual Basic 및 F#을 사용하여 WPF, Windows Forms 및 콘솔 애플리케이션을...

☑ **C++를 사용한 데스크톱 개발**
MSVC, Clang, CMake 또는 MSBuild 등 선택한 도구를 사용하여 Windows용 최신 C++ 앱을 빌드합니다.

☐ **유니버설 Windows 플랫폼 개발**
C#, VB 또는 C++(선택 사항)를 사용하여 유니버설 Windows 플랫폼용 애플리케이션을 만듭니다.

☐ **C++를 사용한 모바일 개발**
C++를 사용하여 iOS, Android 또는 Windows용 플랫폼 간 애플리케이션을 빌드합니다.

설치 세부 정보

- C++를 사용한 데스크톱 개발
  - 포함됨
    - ✓ C++ 핵심 데스크톱 기능
  - 선택 사항
    - ☑ MSVC v143 - VS 2022 C++ x64/x86 빌드...
    - ☑ Windows 10 SDK(10.0.19041.0)
    - ☑ Just-In-Time 디버거
    - ☑ C++ 프로파일링 도구
    - ☑ Windows용 C++ CMake 도구
    - ☑ 최신 v143 빌드 도구용 C++ ATL(x86 및 x64)
    - ☑ Test Adapter for Boost.Test
    - ☑ Test Adapter for Google Test
    - ☑ Live Share
    - ☑ IntelliCode
    - ☑ C++ AddressSanitizer
    - ☐ MSVC v143 - VS 2022 C++ ARM64 빌드 도...
    - ☐ 최신 v143 빌드 도구용 C++ MFC(x86 및 x6...
    - ☐ v143 빌드 도구에 대한 C++/CLI 지원(최신)
    - ☐ v143 빌드 도구용 C++ 모듈(x64/x86 – 실...
    - ☐ Windows용 C++ Clang 도구(13.0.0 - x64/x...

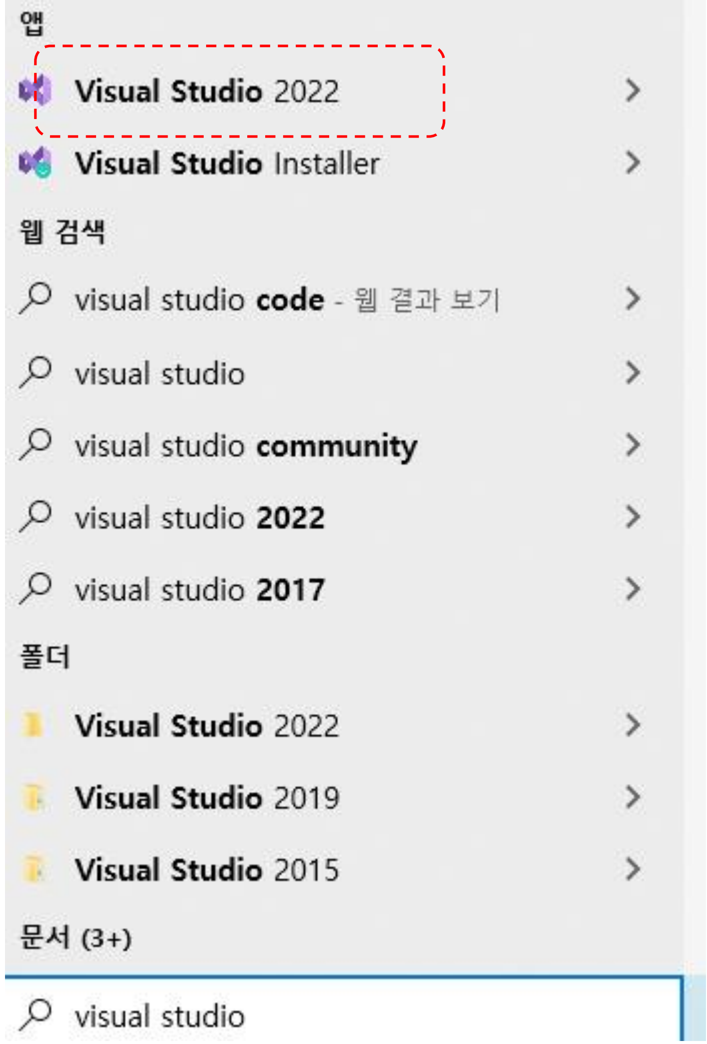
위치
C:\Program Files\Microsoft Visual Studio\2022\Community 변경...

계속하면 선택한 Visual Studio 버전에 대한 라이선스에 동의하게 됩니다. Microsoft는 Visual Studio와 함께 다른 소프트웨어를 다운로드할 수 있는 기능도 제공합니다. 이 소프트웨어는 타사 고지 사항 또는 해당 라이선스에 명시된 것처럼 별도로 라이선스가 허여됩니다. 계속하면 이러한 라이선스에도 동의하게 됩니다.

필요한 전체 공간 8.32GB

MS Visual Studio

- Run the Visual Studio 2022 on the [Start] menu



앱

Visual Studio 2022

Visual Studio Installer

웹 검색

visual studio **code** - 웹 결과 보기

visual studio

visual studio **community**

visual studio **2022**

visual studio **2017**

폴더

Visual Studio 2022

Visual Studio 2019

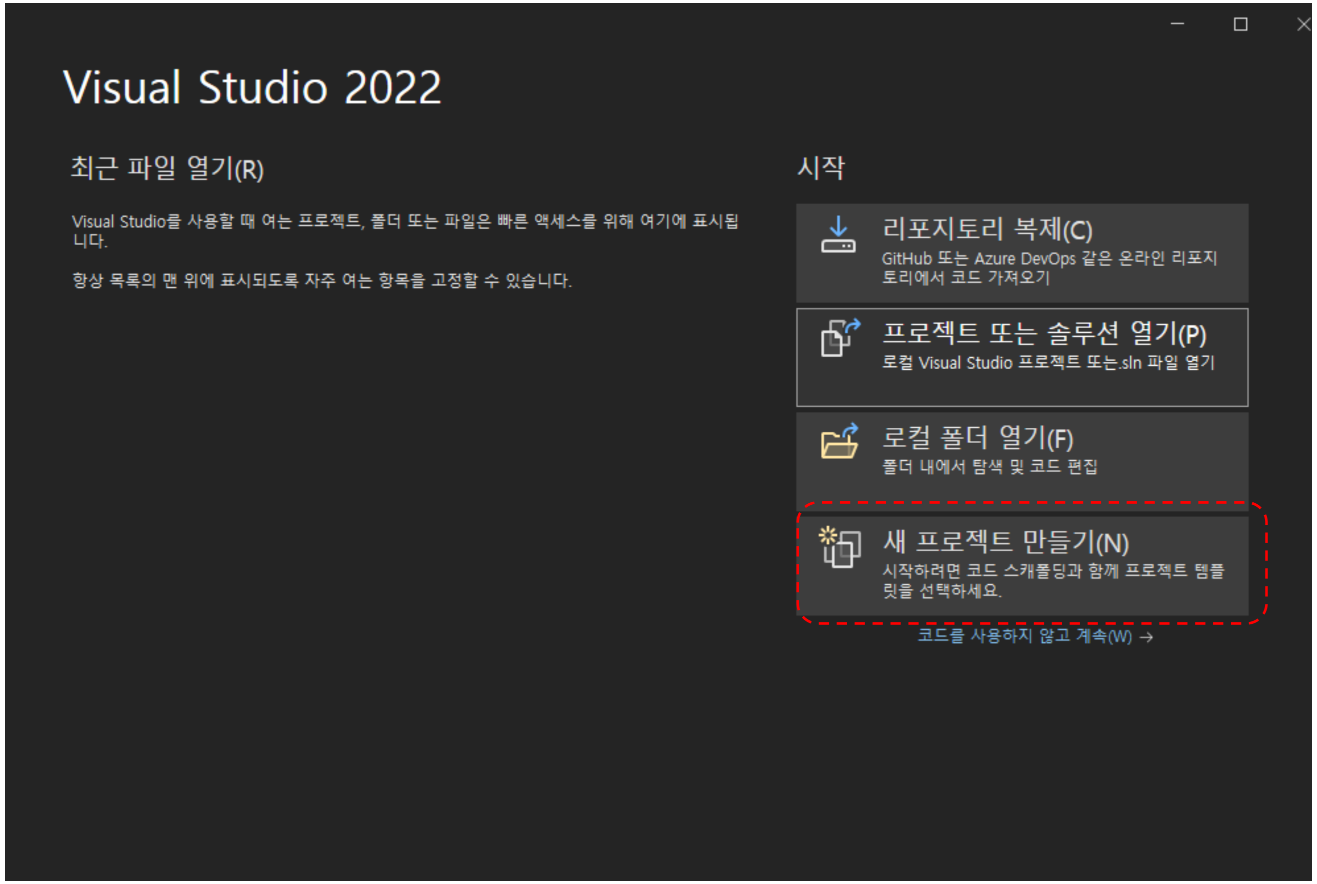
Visual Studio 2015

문서 (3+)

visual studio

MS Visual Studio

- Select the [Create a new Project] button



Visual Studio 2022

최근 파일 열기(R)

Visual Studio를 사용할 때 여는 프로젝트, 폴더 또는 파일은 빠른 액세스를 위해 여기에 표시됩니다.

항상 목록의 맨 위에 표시되도록 자주 여는 항목을 고정할 수 있습니다.

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

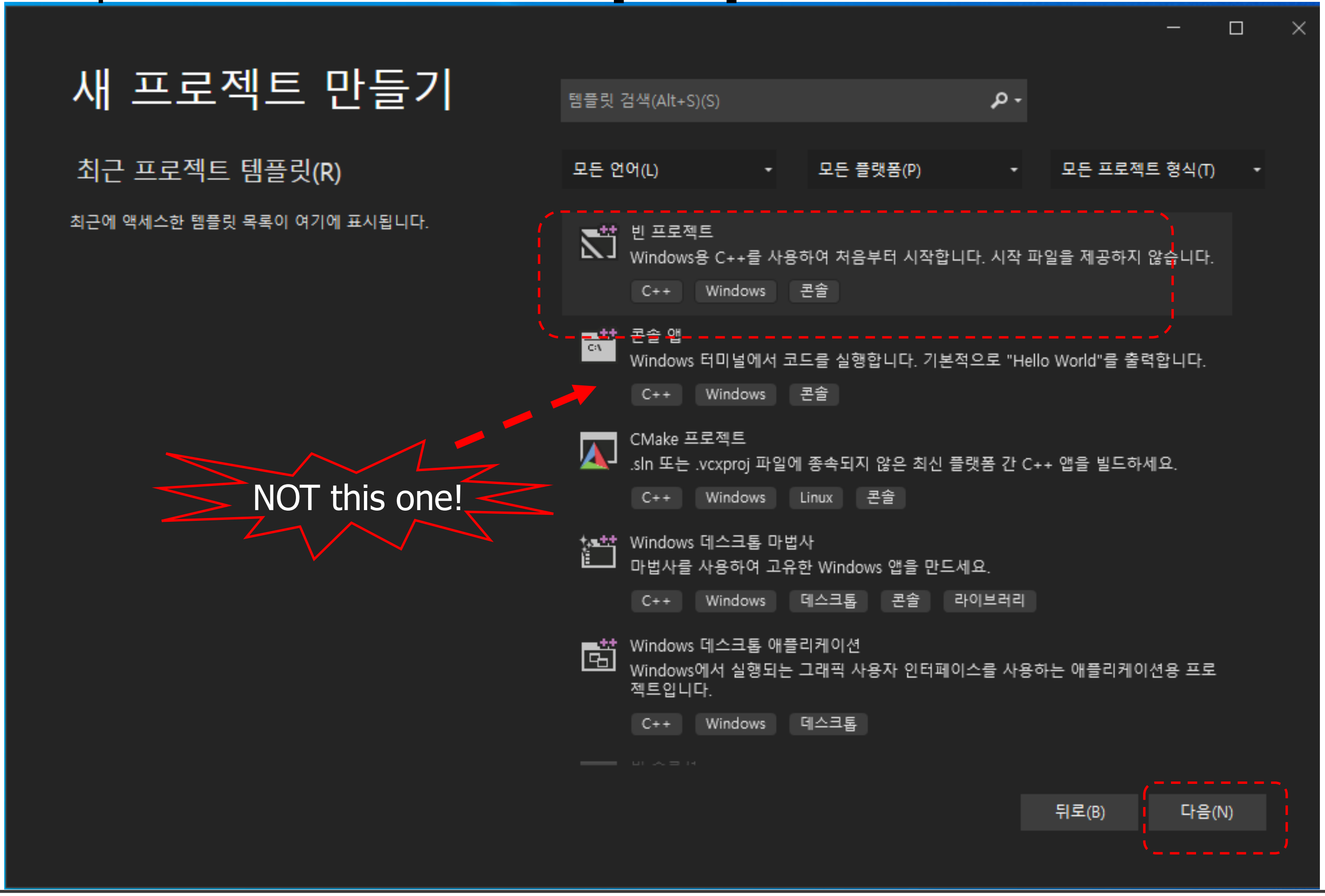
로컬 폴더 열기(F)
폴더 내에서 탐색 및 코드 편집

새 프로젝트 만들기(N)
시작하려면 코드 스캐폴딩과 함께 프로젝트 템플릿을 선택하세요.

코드를 사용하지 않고 계속(W) →

MS Visual Studio

- The [Create a new Project] dialog will be shown
- Select the [Empty Project] template, and then choose [Next]



새 프로젝트 만들기

템플릿 검색(Alt+S)(S)

모든 언어(L) | 모든 플랫폼(P) | 모든 프로젝트 형식(T)

최근 프로젝트 템플릿(R)

최근에 액세스한 템플릿 목록이 여기에 표시됩니다.

빈 프로젝트
Windows용 C++를 사용하여 처음부터 시작합니다. 시작 파일을 제공하지 않습니다.
C++ | Windows | 콘솔

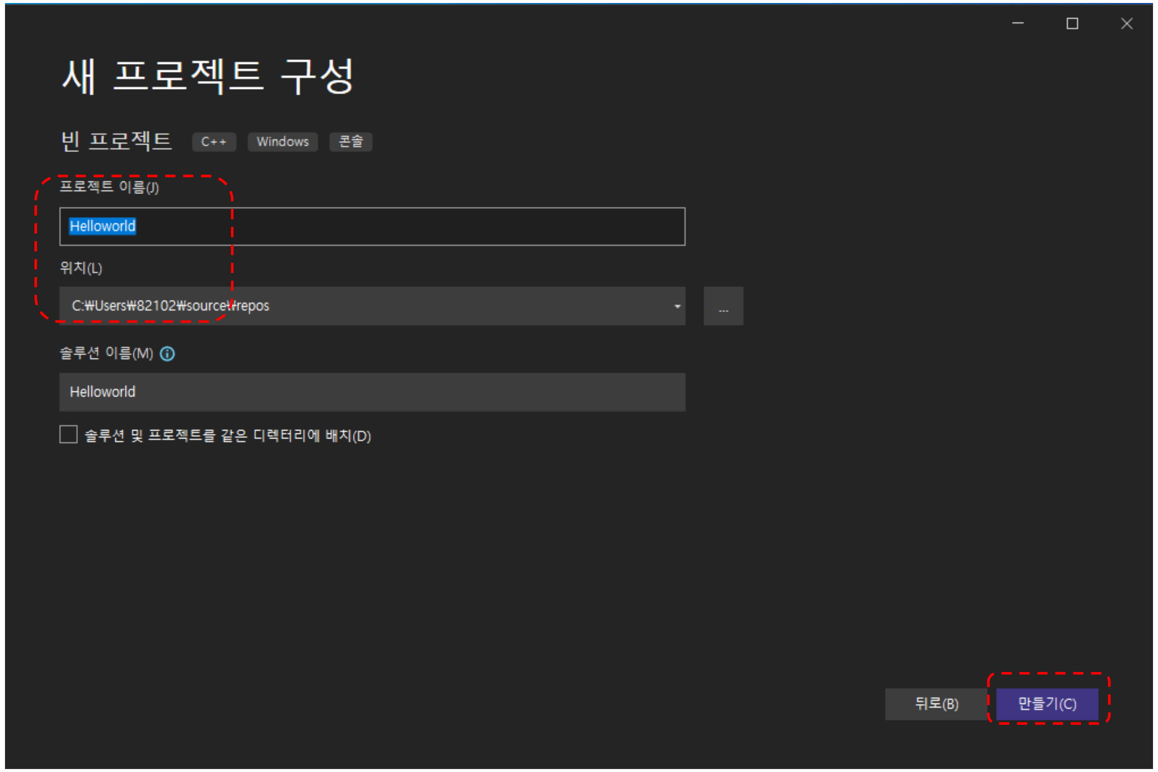
콘솔 앱
Windows 터미널에서 코드를 실행합니다. 기본적으로 "Hello World"를 출력합니다.
C++ | Windows | 콘솔

CMake 프로젝트
.sln 또는 .vcxproj 파일에 종속되지 않은 최신 플랫폼 간 C++ 앱을 빌드하세요.
C++ | Windows | Linux | 콘솔

Windows 데스크톱 마법사
마법사를 사용하여 고유한 Windows 앱을 만드세요.
C++ | Windows | 데스크톱 | 콘솔 | 라이브러리

Windows 데스크톱 애플리케이션
Windows에서 실행되는 그래픽 사용자 인터페이스를 사용하는 애플리케이션용 프로젝트입니다.
C++ | Windows | 데스크톱

MS Visual Studio



새 프로젝트 구성

빈 프로젝트 C++ Windows 콘솔

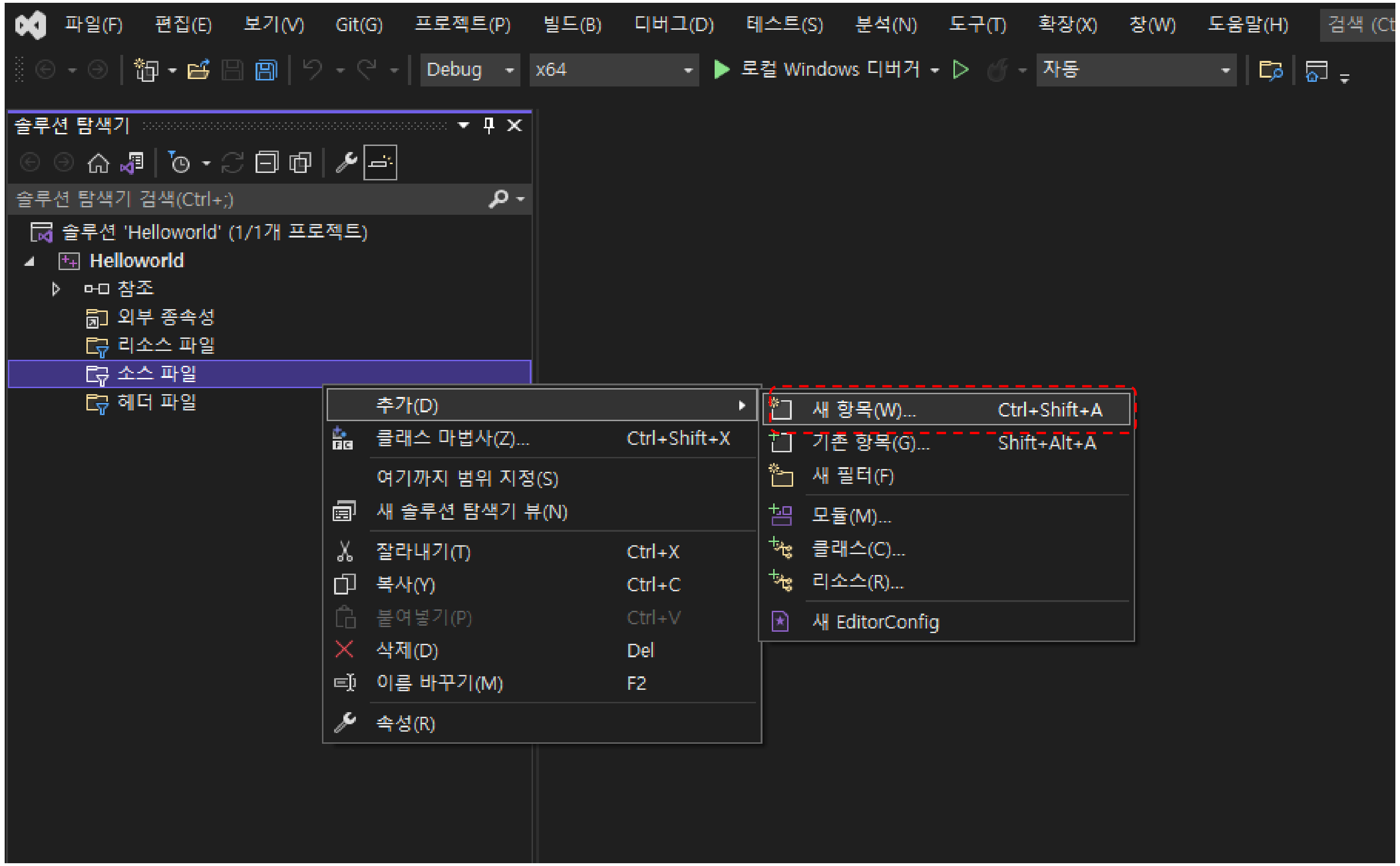
프로젝트 이름(J): Helloworld

위치(L): C:₩Users₩82102₩source₩repos

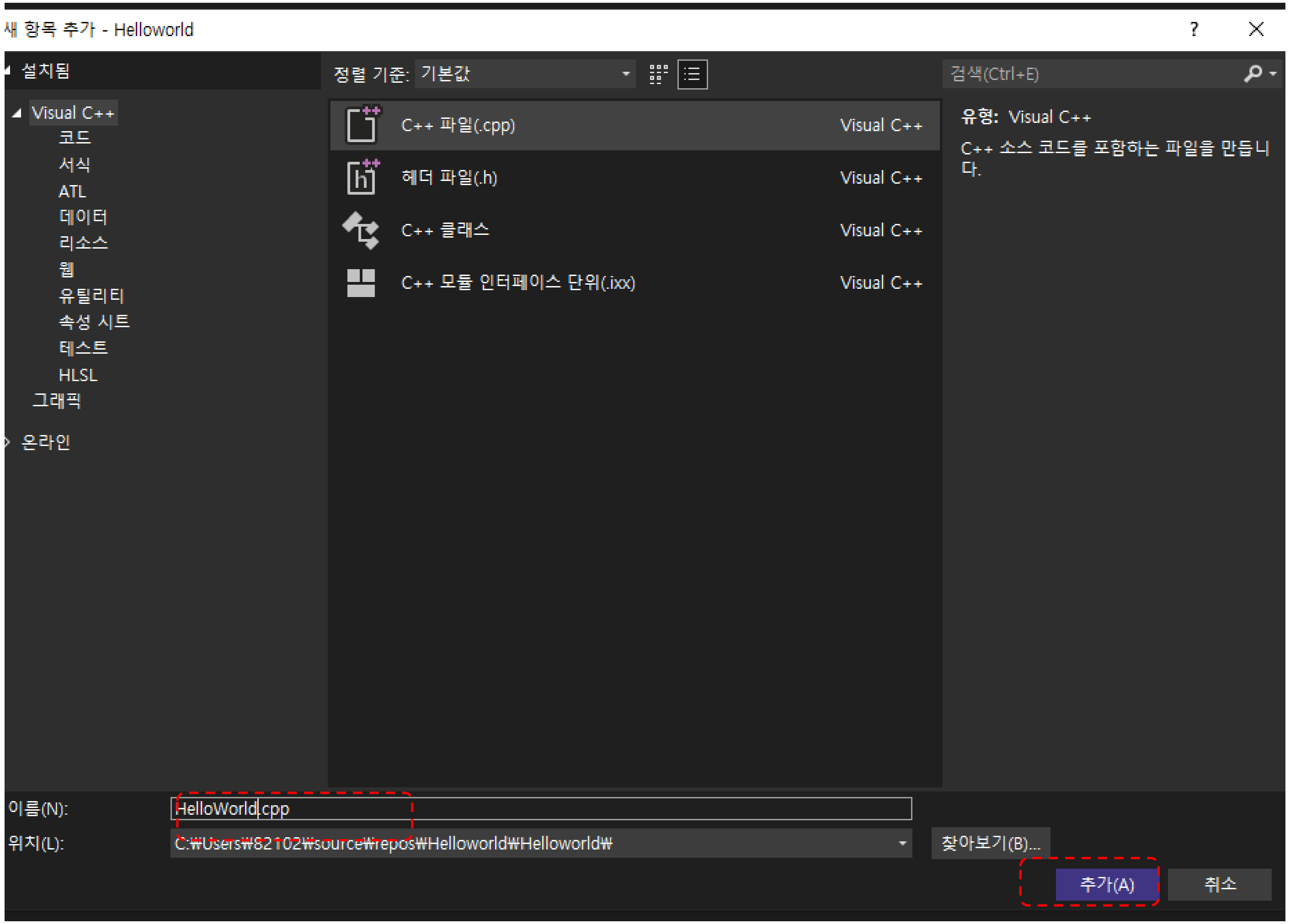
솔루션 이름(M): Helloworld

□ 솔루션 및 프로젝트를 같은 디렉터리에 배치(D)

MS Visual Studio



MS Visual Studio



새 항목 추가 - Helloworld

정렬 기준: 기본값

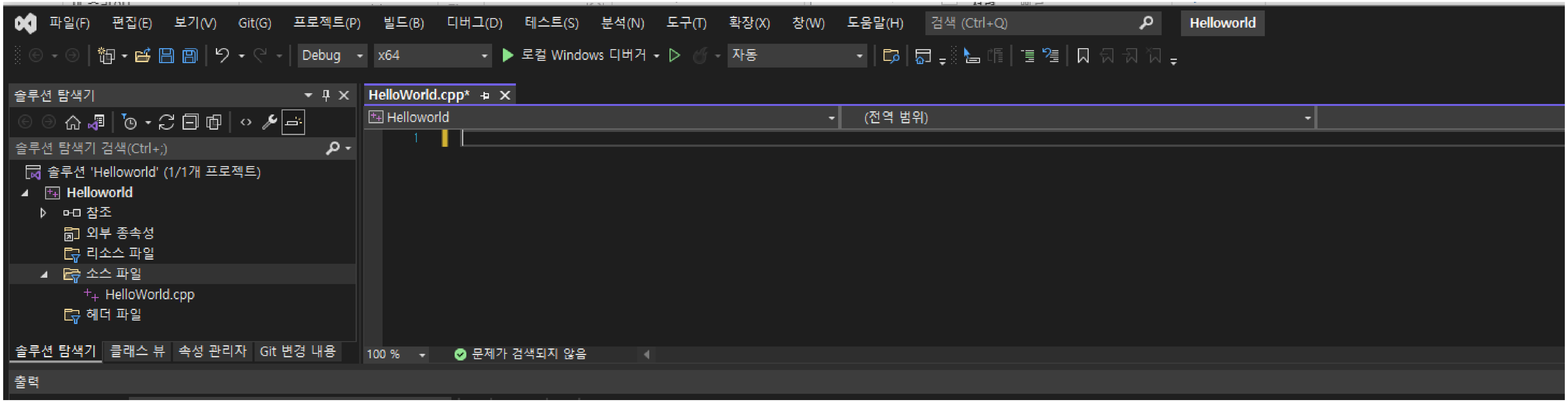
유형: Visual C++

C++ 소스 코드를 포함하는 파일을 만듭니다.

이름(N): HelloWorld.cpp

위치(L): C:₩Users₩82102₩source₩repos₩Helloworld₩Helloworld₩

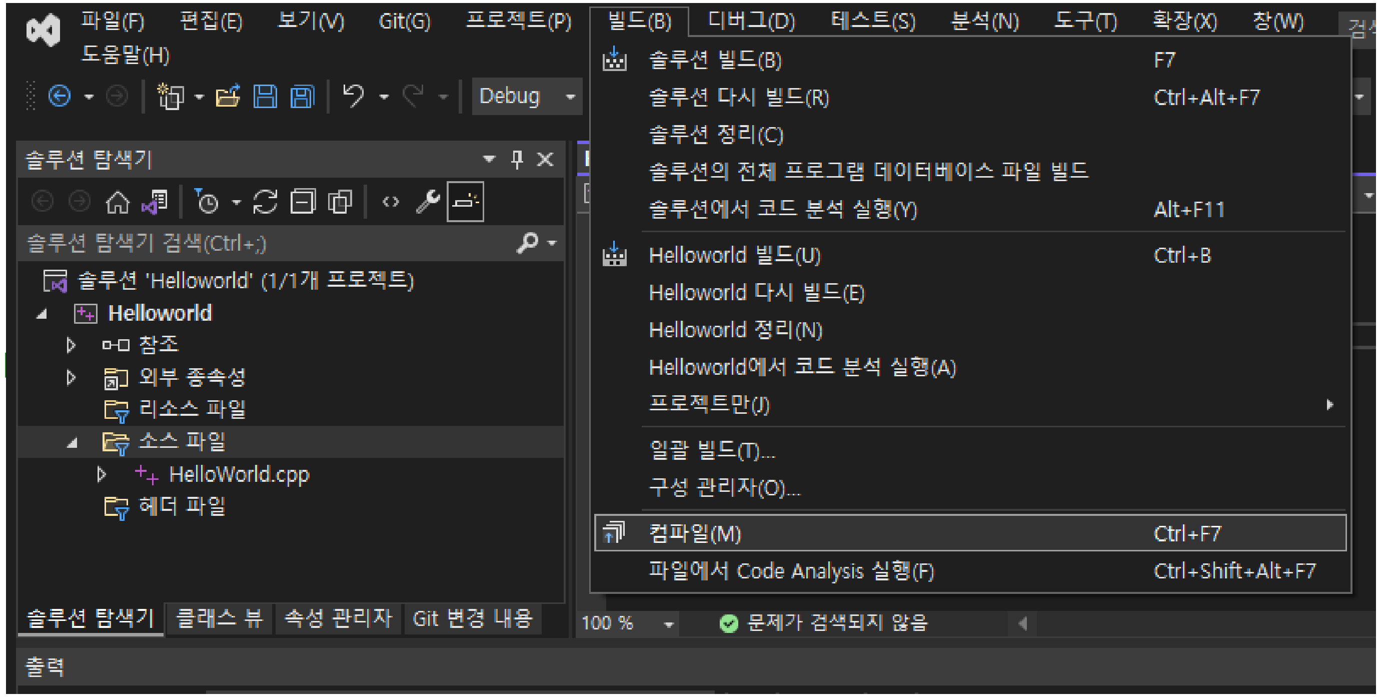
MS Visual Studio



문제가 검색되지 않음

C++ Program

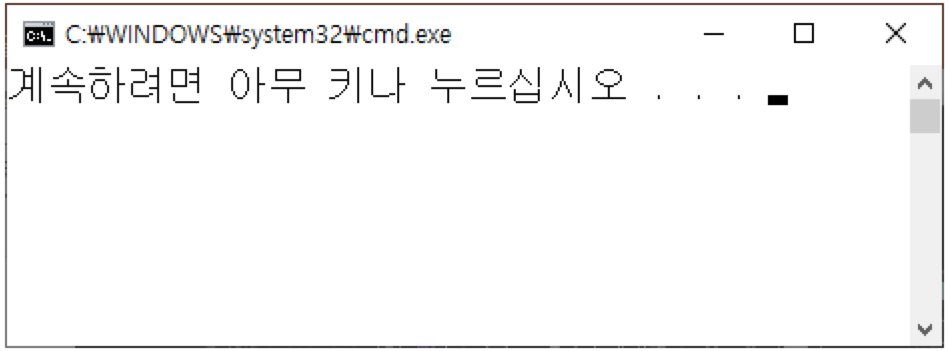
```
#include <iostream>

int main() {

    //variable declaration
    //read values input from user
    //computation  and print output to use
    return 0;
}
```



- After you write a C++ program you compile it; that is, you run a program called compiler that checks whether the program follows the C++ syntax
- if it finds errors, it lists them

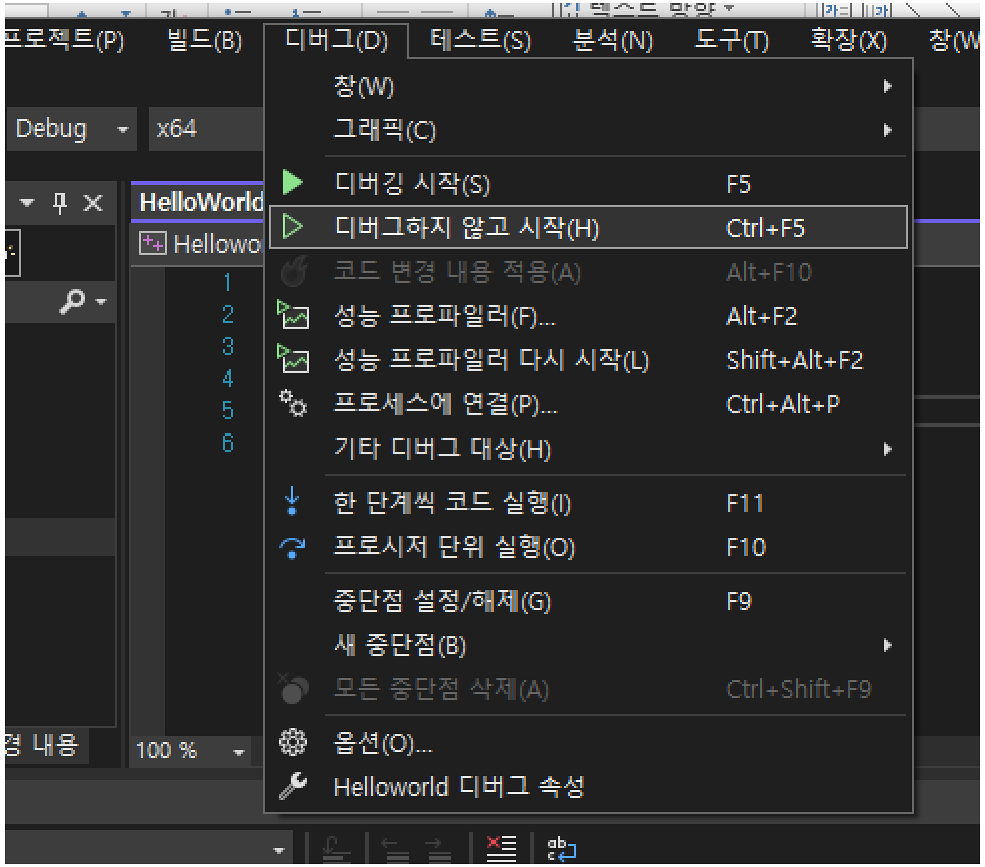
C++ Program

- Execute



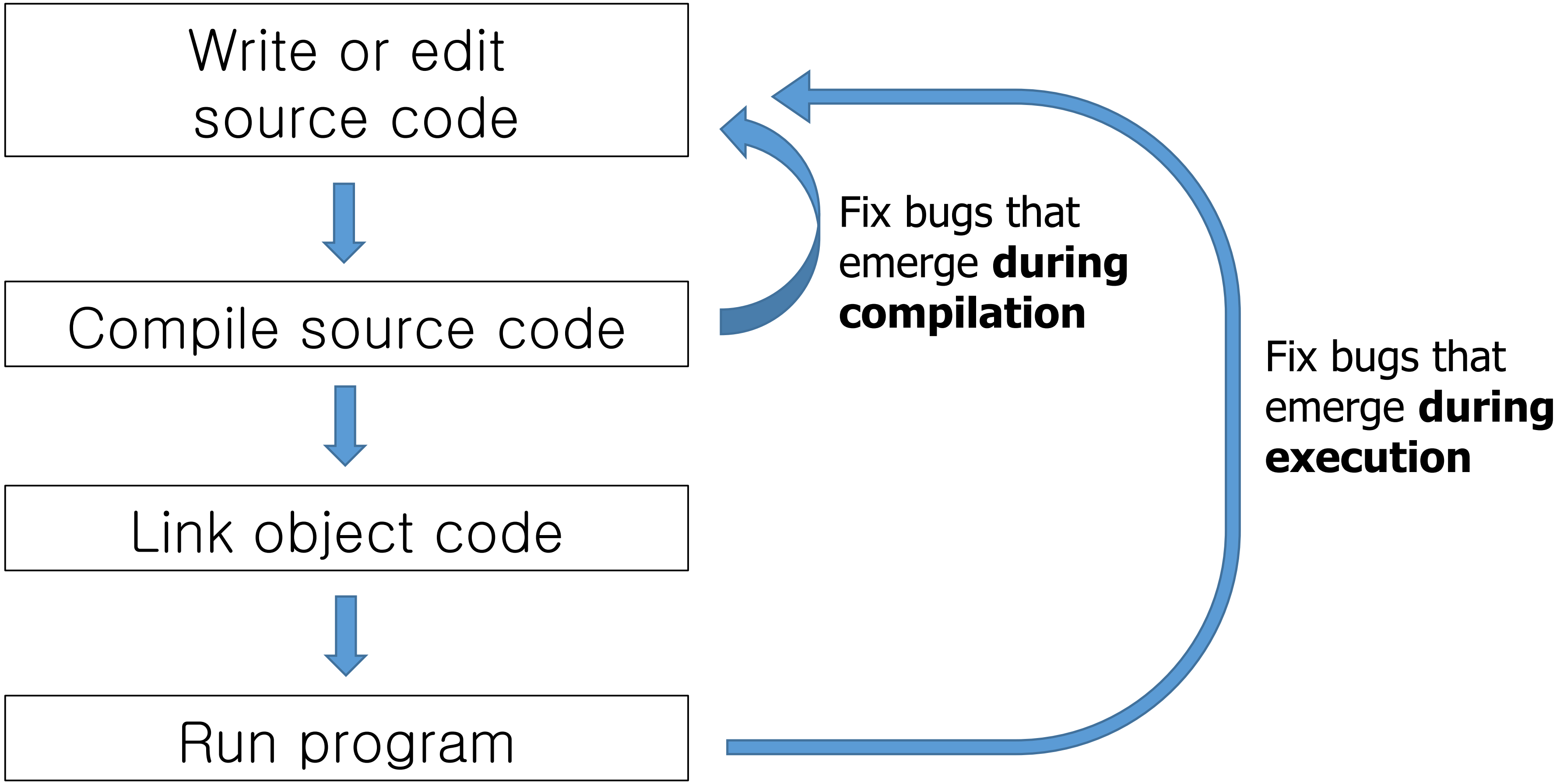
C:₩WINDOWS₩system32₩cmd.exe

```
계속하려면 아무 키나 누르십시오 . . .
```



프로젝트(P)
빌드(B)
디버그(D)
테스트(S)
분석(N)
도구(T)
확장(X)
창(W
Debug
x64
HelloWorld
Hellowo
창(W)
그래픽(C)
디버깅 시작(S) F5
디버그하지 않고 시작(H) Ctrl+F5
코드 변경 내용 적용(A) Alt+F10
성능 프로파일러(F)... Alt+F2
성능 프로파일러 다시 시작(L) Shift+Alt+F2
프로세스에 연결(P)... Ctrl+Alt+P
기타 디버그 대상(H)
한 단계씩 코드 실행(I) F11
프로시저 단위 실행(O) F10
중단점 설정/해제(G) F9
새 중단점(B)
모든 중단점 삭제(A) Ctrl+Shift+F9
옵션(O)...
Helloworld 디버그 속성
100 %

Compile and Execute



Fix bugs that emerge **during compilation**

Fix bugs that emerge **during execution**

C++ Program

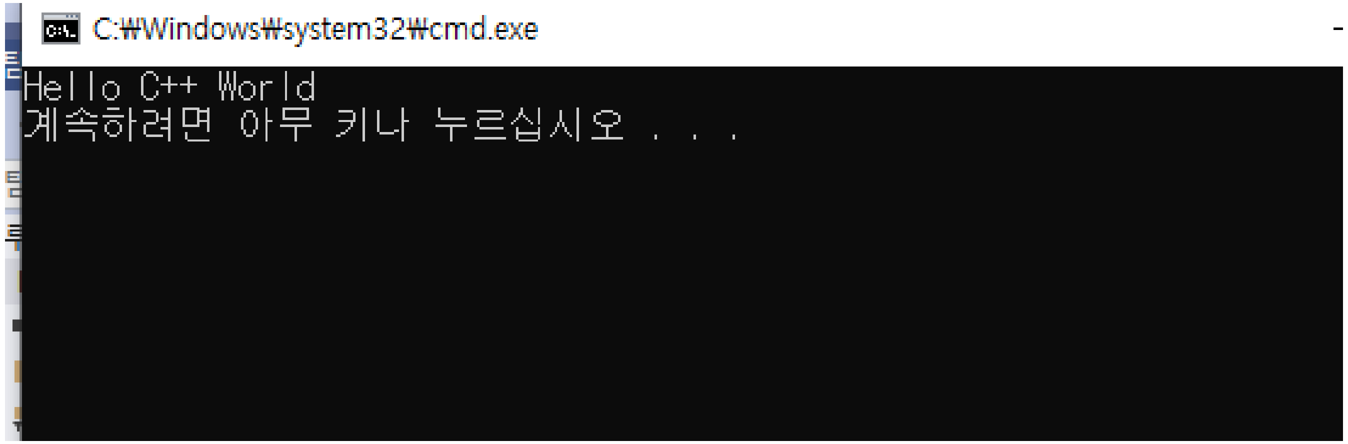
- Every C++ Program contains one or more **functions**, one of which must be named **main**
- The operating system runs a C++ program by calling main

```
#include <iostream>

int main() {

    //variable declaration
    //read values input from user
    //computation  and print output to user
    return 0;
}
```

- A function definition has four elements; a return type, a function name, a (possibly empty) parameter list enclosed in parentheses, and a function body

Hello C++ World

```cpp
#include <iostream>

int main()
{
    std::cout << "Hello C++ World" << std::endl;
    return 0;
}
```

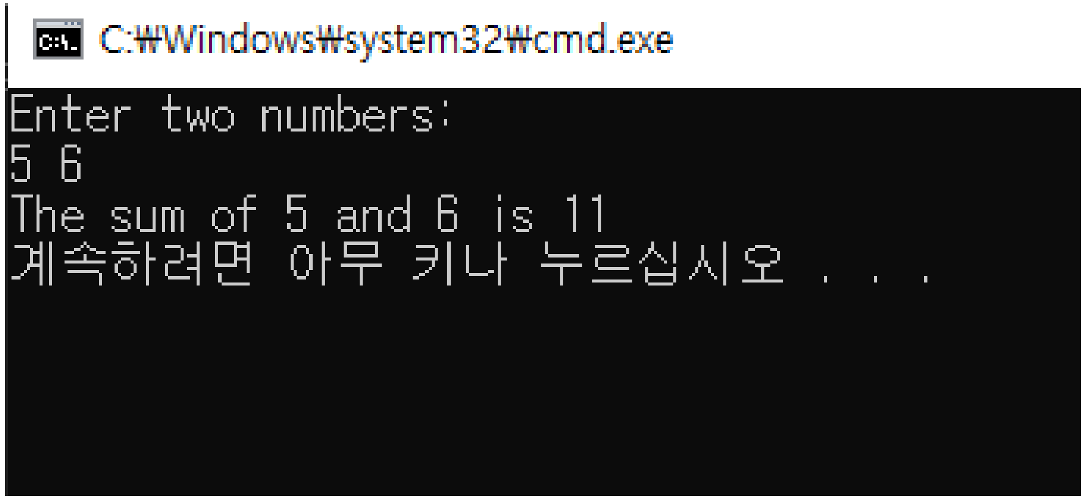


```
Hello C++ World
계속하려면 아무 키나 누르십시오 . . .
```

Input/Output

- C++ includes an extensive standard library that provides IO.
- iostream library
  - To handle input, we use an object named **cin**
  - To handle output, we use an object named **cout**
- Simple IO stream example

```
#include <iostream>

int main()
{
std::cout << "Enter two numbers:" << std::endl;
int v1 = 0, v2 = 0;
std::cin >> v1 >> v2;
std::cout << "The sum of " << v1 << " and " << v2
<< " is " << v1 + v2 << std::endl;
return 0;
}
```



```
Enter two numbers:
5 6
The sum of 5 and 6 is 11
계속하려면 아무 키나 누르십시오 . . .
```

Namespace std

- Careful readers will note that this program uses std::cout and std::endl rather than just cout and endl
- The prefix std:: indicates that the names cout and endl are defined inside the **namespace** named **std**
- Namespaces allow us to avoid inadvertent collisions between the names we define and uses of those same name sinside a library
- All the names defined by the standard library are in the std namespace.
- One side effect of the library’s use of a namespace is that when we use a name from the library, we must say explicitly that we want to use the name from the std namespace
- Writing std::cout uses the scope operator (the **:: operator**) to say that we want to use the name cout that is defined in the namespace std.

Comments

- **Comments** are ignored by the compiler.
- Comments can help readers understand the code.
  - /* … */
  - //

```
#include <iostream>

void main() {
    int u = 1, v = 2;
    std::cout << u + v;    // sum of u and v
                           /* sum of u and v */
}
```

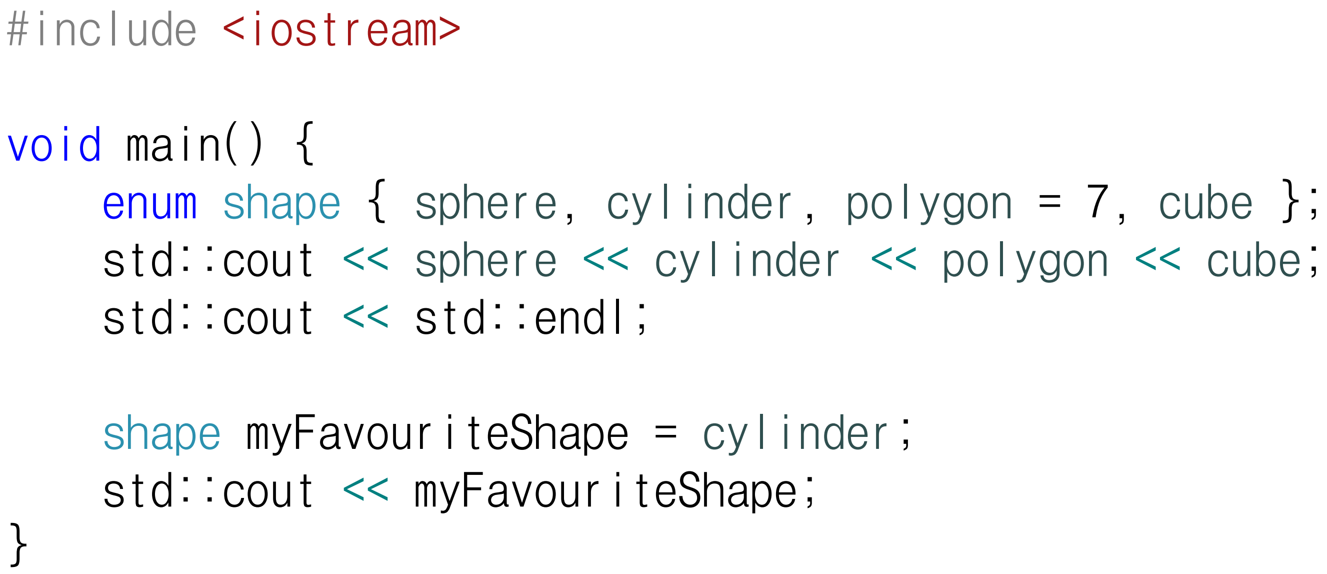
Types, Variables

- Primitive built-in types
  - bool, char, wchar_t, short, int, long, float, double
  - unsigned char, unsigned int, …
  - void

```
#include <iostream>

void main() {
    int height = 11, width = 9, length = 40;
    int result = height * width * length;

    std::cout << "The volume of the box car is ";
    std::cout << result << std::endl;
}
```

Types, Variables

- Enumerations
  - **Enumerations** provide an alternative method for defining/grouping sets of integer type constants.



```cpp
#include <iostream>

void main() {
    enum shape { sphere, cylinder, polygon = 7, cube };
    std::cout << sphere << cylinder << polygon << cube;
    std::cout << std::endl;

    shape myFavouriteShape = cylinder;
    std::cout << myFavouriteShape;
}
```

Types, Variables

- typedef
  - **typedef** allows us to define a synonym for a type

```
#include <iostream>

typedef double wages;// wages is double
typedef int exam_score;// exam_score is int
typedef wages salary;// salary is wages (double)

void main() {
    wages   wage0 = 200, wage1 = 300;
    exam_score  score0 = 90, score1 = 100;

    std::cout << wage0 << std::endl << score0 << std::endl;
    std::cout << wage1 << std::endl << score1 << std::endl;
}
```

Types, Variables

- sizeof
  - The **sizeof** operator returns the size (in bytes) of a type or an object

```
#include <iostream>

typedef double wages;// wages is double
typedef int exam_score;// exam_score is int

void main() {
    wages w;
    std::cout << sizeof(int) << ":" << sizeof(exam_score) << std::endl;
    std::cout << sizeof(double) << ":" << sizeof(wages) << std::endl;
    std::cout << sizeof(w) << std::endl;
}
```

Scope of Variables

- Local and global variables

```
#include <iostream>

int a = 3;

void main() {
    int b = 5;
    {
        int c = 7;
        std::cout << a << b << c;
    }
    std::cout << a << b;
    std::cout << c;// Compilation Error !
}
```

Global variable

Scope of variable c

Scope of Variables

- extern
  - We can declare a global variable without defining it by using the **extern** keyword.

main.cpp

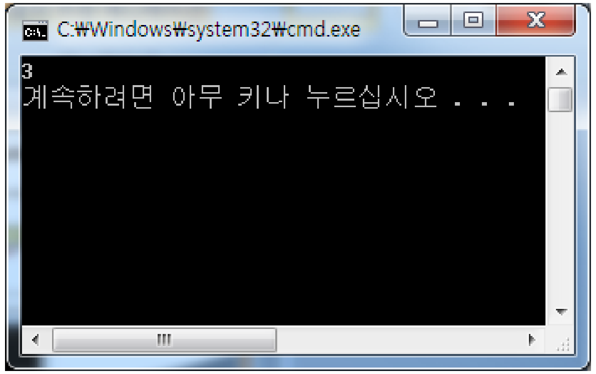
```
#include <iostream>

extern int a;

void main() {
    std::cout << a << std::endl;
}
```

declaration of a

tmp.cpp

```
int a = 3;
```

definition of a



```
3
계속하려면 아무 키나 누르십시오 . . .
```

Scope of Variables

- static global variable
  - We can define a global variable as **static** to make its scope local to a file.

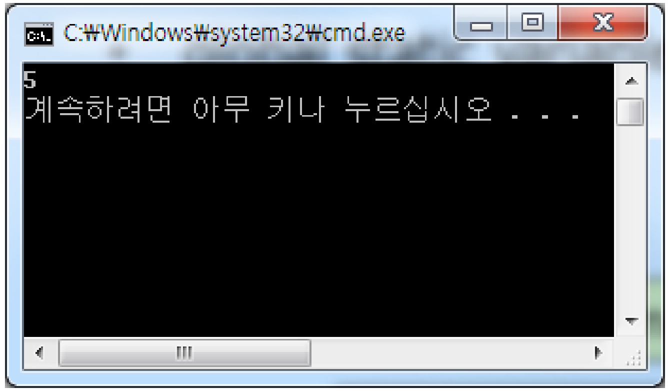
main.cpp

```
#include <iostream>

static int a = 5;

static int f() {}

void main() {
    std::cout << a << std::endl;
}
```

tmp.cpp

```
static int a = 3;
static int f() { }
```

different definitions of variable a



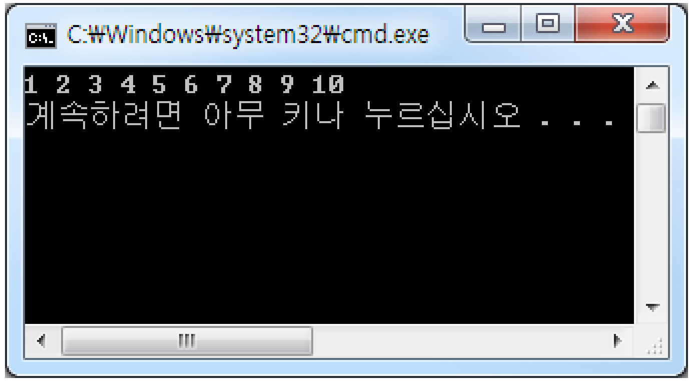
```
5
계속하려면 아무 키나 누르십시오 . . .
```

Scope of Variables

- local static variable
    - Local static variables of a function are kept intact when the function returns.

```
#include <iostream>

void func() {
    static int a = 0;
    a++;
    std::cout << a << " ";
}

void main() {
    for(int i=0;i<10;++i)
        func();
    std::cout << std::endl;
}
```

Local Static Variable



```
1 2 3 4 5 6 7 8 9 10
계속하려면 아무 키나 누르십시오 . . .
```

Scope of Variables

- const
  - A **constant** is a special kind of variable whose value cannot be altered in the program.

```
#include <iostream>

void main() {
    int a = 3;
    const int b = 5;          // b is const variable

    a = 7;
    b = 7;                    // Compilation Error !
}
```

Basic Expressions

- Arithmetic expressions

  +, -, *, /, %

```
#include <iostream>

void main() {
    std::cout << 6 + 3 << std::endl;
    std::cout << 6 - 3 << std::endl;
    std::cout << 6 * 3 << std::endl;
    std::cout << 6 / 3 << std::endl << std::endl;
    std::cout << 5 / 3 << std::endl;
    std::cout << 5 % 3 << std::endl;
    std::cout << 5.0 / 3.0 << std::endl;
}
```

Basic Expressions

- Numerical predicates
  - ==, !=, >, <, >=, <=

```
#include <iostream>

void main() {
    int i = 50;
    double d = 50.0;
    std::cout << (i == (int)d) << std::endl;
    std::cout << ((double)i != d) << std::endl;
}
```

Basic Expressions

- Conditional operator
  - cond ? expr1 : expr2;

```cpp
#include <iostream>

void main() {
    int score;
    std::cin >> score;
    std::cout << "The score is " << score <<
    (score == 1 ? " point" : " points") << "." << std::endl;
}
```

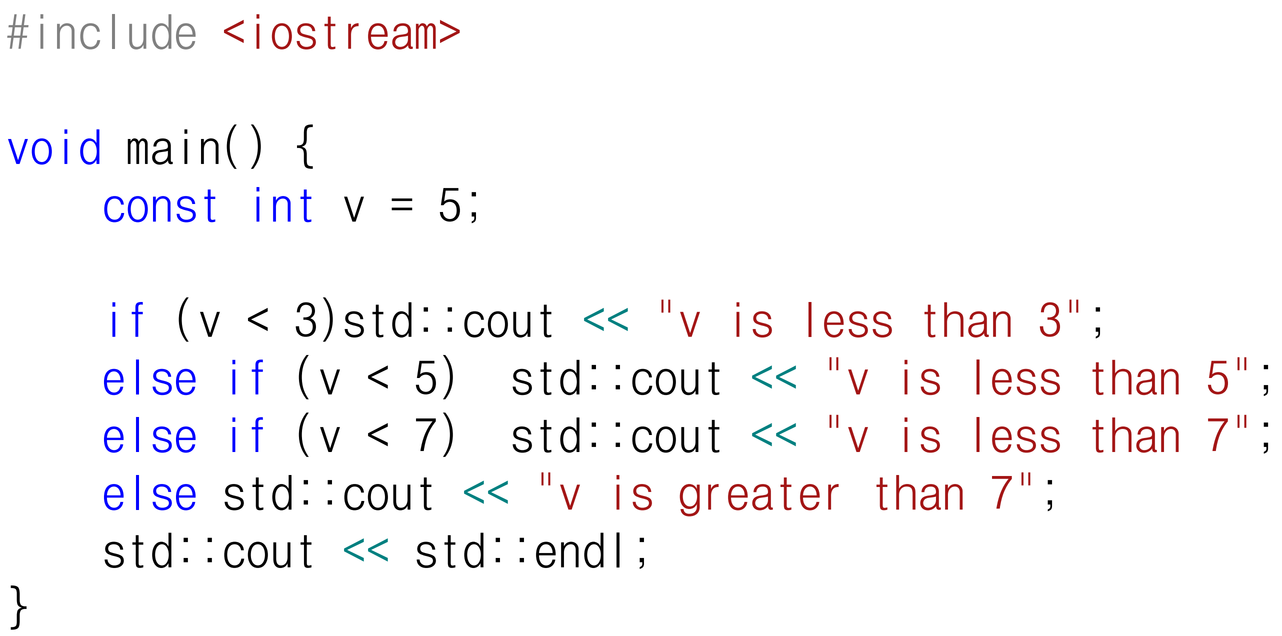
Basic Expressions

- Memory management
  - new, delete

```
#include <iostream>

void main() {
   int * v = new int[10];       // allocate ten consecutive
                                // integer variables

   delete v;                    // de-allocate
}
```

Basic Statements

- Conditional statement
  - **if … else**, switch



```
#include <iostream>

void main() {
    const int v = 5;

    if (v < 3)std::cout << "v is less than 3";
    else if (v < 5)  std::cout << "v is less than 5";
    else if (v < 7)  std::cout << "v is less than 7";
    else std::cout << "v is greater than 7";
    std::cout << std::endl;
}
```

Basic Statements

- Conditional statement
  - if … else, **switch**

```
#include <iostream>

void main() {
    const int v = 5;

    switch (v) {
    case 3: std::cout << "v is 3"; break;
    case 5: std::cout << "v is 5"; break;
    case 7: std::cout << "v is 7"; break;
    default: std::cout << "v is not 3 or 5 or 7";
    }

    std::cout << std::endl;
}
```

Basic Statements

- Loops
  - **for**, while, do_while

- Problem
  - Do summation from 1 to 10

```
#include <iostream>

void main() {
    int sum = 0;
    for (int i = 1; i <= 10; ++i)
        sum += i;
    std::cout << sum << std::endl;
}
```

Basic Statements

- Loops
  - for, **while**, do_while

- Problem
  - Do summation from 1 to 10

```
#include <iostream>

void main() {
    int sum = 0, i = 1;
    while (i <= 10) {
        sum += i;
        i++;
    }
    std::cout << sum << std::endl;
}
```

Basic Statements

- Loops
  - for, while, **do_while**

- Problem
  - Do summation from 1 to 10

```
#include <iostream>

void main() {
    int sum = 0, i = 1;
    do {
        sum += i;
        i++;
    } while (i <= 10);
    std::cout << sum << std::endl;
}
```

Class

- A **class** consists of the datafields and interface.

```cpp
#include <iostream>

class Box {
public:
    void print() {
        std::cout << height << " " << width << " " << length << std::endl;
    }
    double height, width, length;
};

void main() {
    Box box;
    box.height = 3; box.width = 5; box.length = 7;
    box.print();
}
```