++

C++ Modules 실무활용 언리쉬드

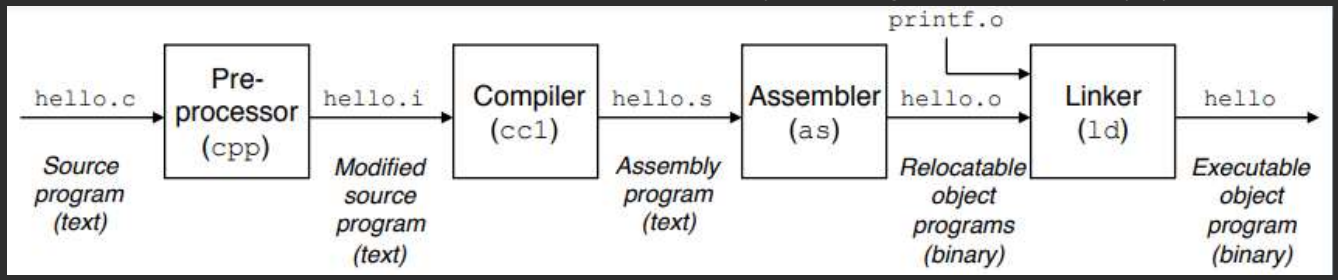
Microsoft MVP Most Valuable Professional
drvoss@gmail.com

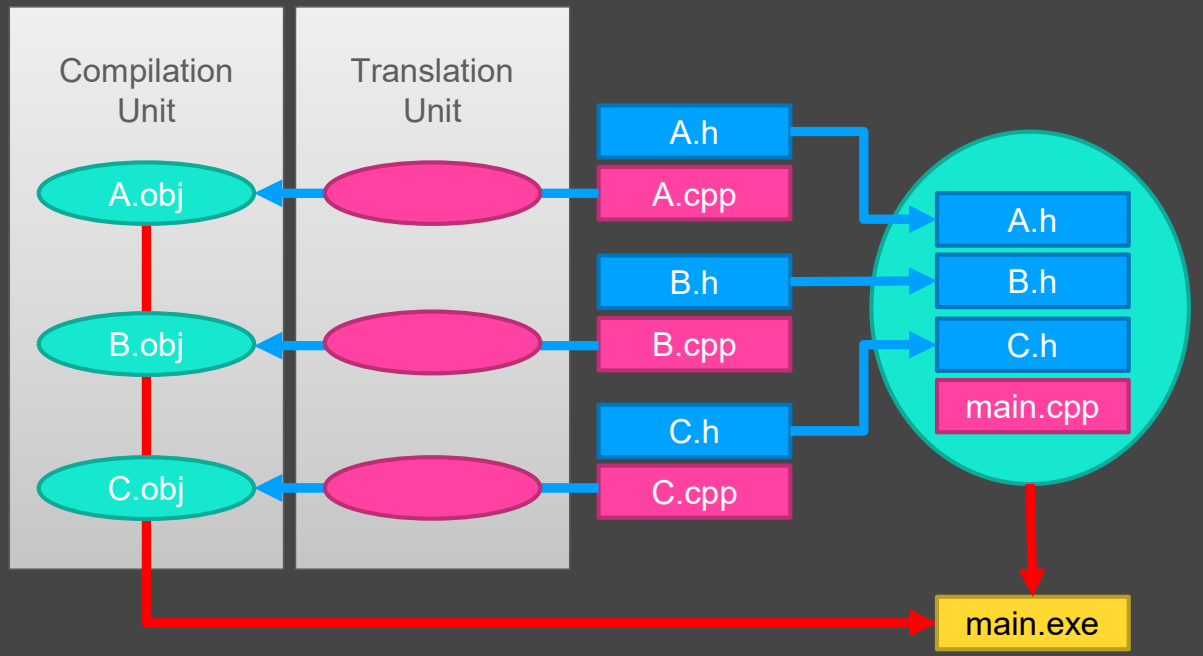
Motivation – C++20 Modules

```
package main

import (
  “fmt”
  “os”
)

func main() {
  var name string
  fmt.Print(“이름을 입력하세요: “)
  fmt.Scanf(”%s”, &name)
  fmt.Fprintf(os.Stdout, “Hello %s\n”, name)
}
```

Motivation – 컴파일 과정

'Computer Systems A Programmer's Perspective' by Bryant, O'Hallaron



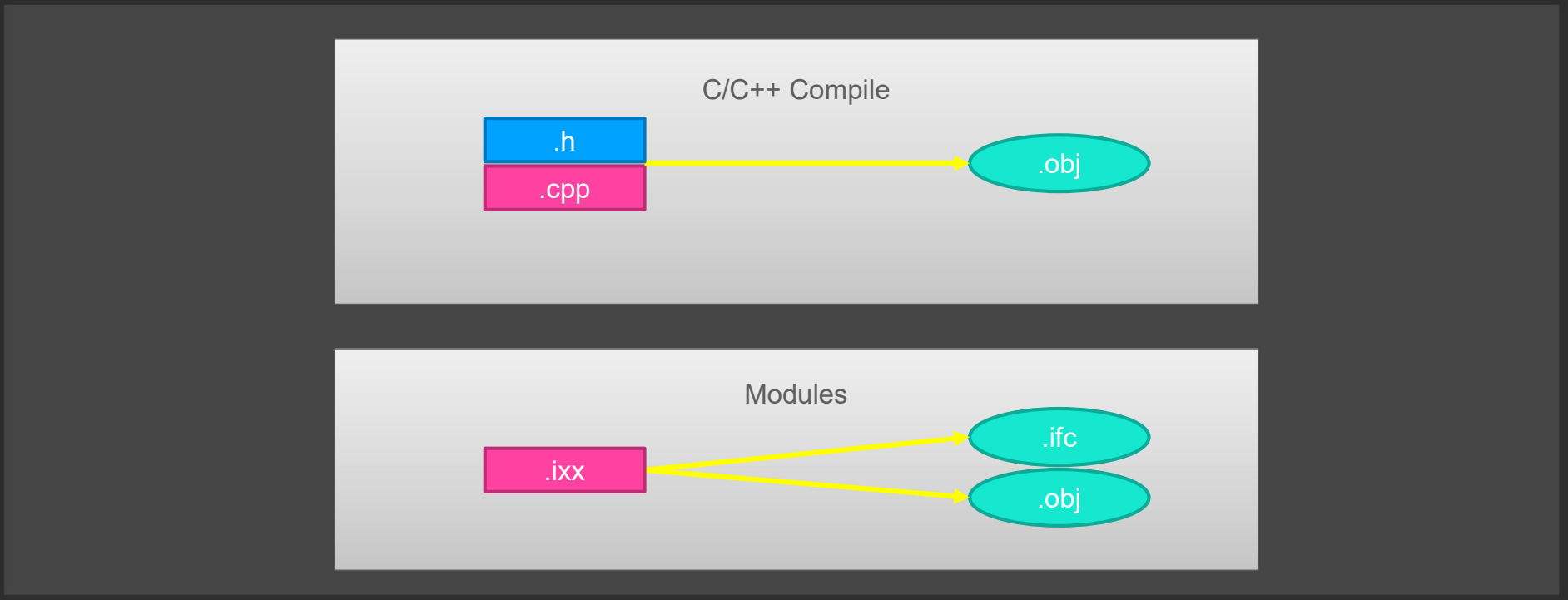


Motivation – Translation Unit

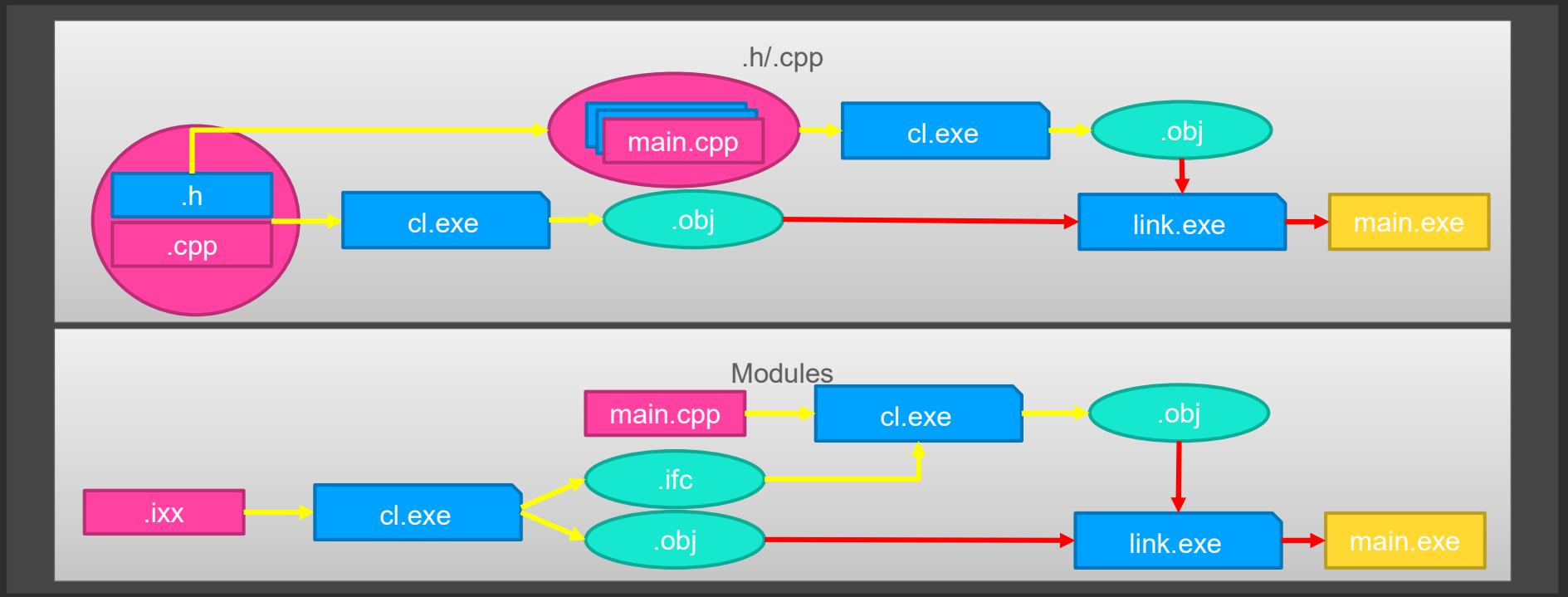
```cpp
#include <iostream>
int main() {
  std::cout << "Hello World" << std::endl;
}
```

```
PS > type .\Source.cpp | Measure-Object -Character -Line
Lines Words Characters Property
----- ----- ---------- --------
    4                73
PS > cl /E .\Source.cpp | Measure-Object -Character -Line
Lines Words Characters Property
----- ----- ---------- --------
31084          1363652
```

개요 - .h/.cpp와 차이점

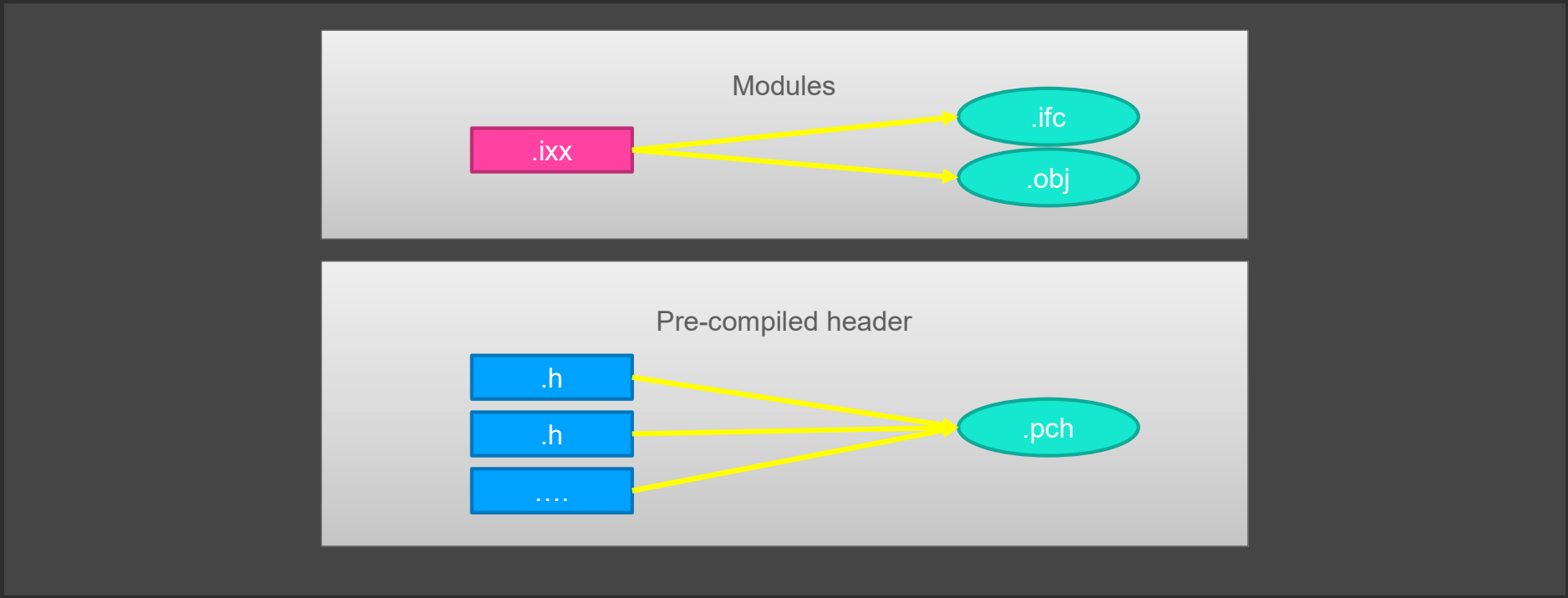


개요 - .h/.cpp와 비교



개요 - PCH와 비교

- 모듈이 확산될 경우 PCH를 점차 대체하게 될 예정



개요 – 장점

- 모듈은 한번만 처리, **컴파일 시간** 단축 효과
- 헤더파일 **과잉** 방지
- 인클루드 가드 불필요
- 매크로 격리(이름 충돌, 의존성 문제)
- 소스코드<-> 헤더**파일간 영향 격리**
- 모듈에서는 기본이 비노출, **노출할 항목** 명시적 **선택**
- 구현부 패키징 후 **논리적인 네이밍**(파일경로x)
- 어떤 모듈이 import되었는지 중요하지 X
- 가능한 모듈 사용을 권장

Agenda

1. Motivation
2. 모듈(Modules) 브리핑
3. 모듈(Modules)-훑어보기, 만들기
4. 세부사항
5. 서브모듈(Submodules)
6. 모듈 파티션(Module Partitions)
7. 모듈 헤더 유닛
8. 템플릿
9. 모듈 연결(Module Linkage)
10. Visual Studio IDE 지원
11. 모듈 적용시 고려할 점

모듈(Modules)-훑어보기(import)

- Modules은 #include 대신 #import를 사용

```
// typical hello
#include <iostream>
int main() {
  std::cout << "HelloWorld!\n";
}
```

```
// module
import std.core;
int main() {
  std::cout << "HelloWorld!\n";
}
```

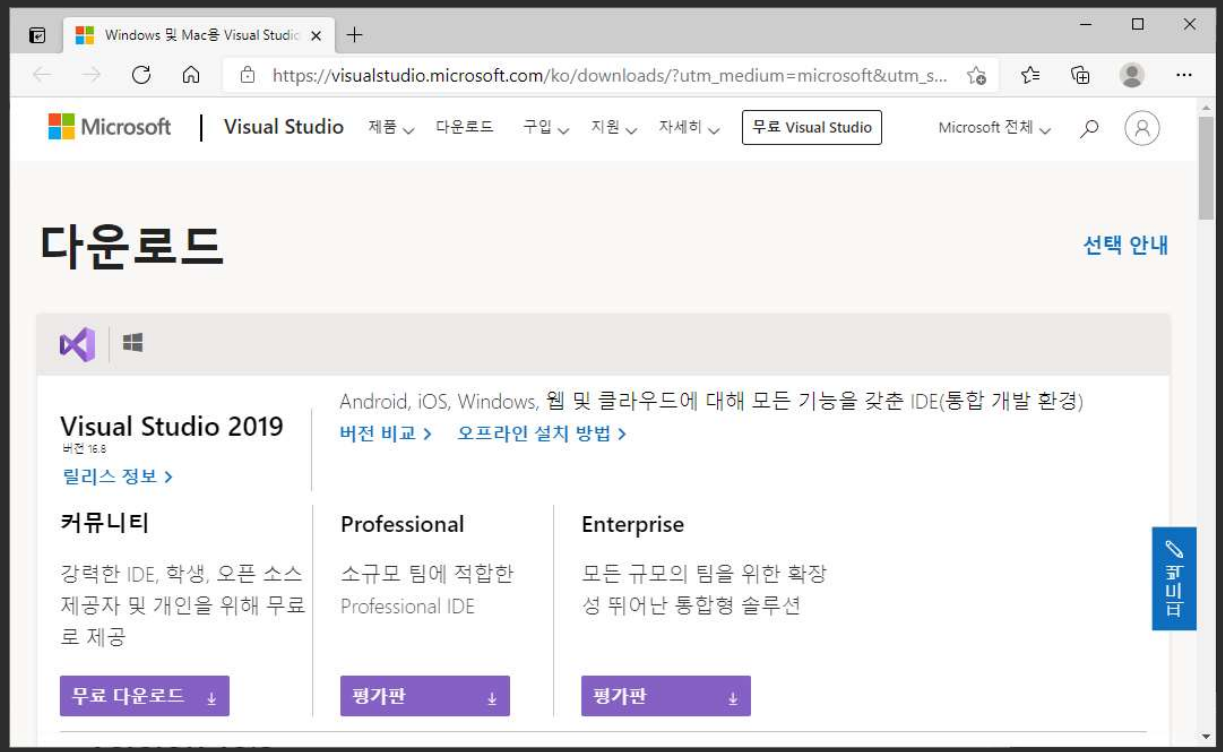
모듈(Modules)-훑어보기(C++SL)

- C++20을 지원하는 **모든 std:: 헤더**는 **모듈**로 제공 (<c*> 제외)
- 커스텀 헤더파일 import는 컴파일러의 지원여부 확인
- 같은 라이브러리는 #include와 #import 동시 사용 불가

C++ 표준 라이브러리		모듈화된 표준 라이브러리
<regex>	→	std.regex
<filesystem>	→	std.filesystem
<memory>	→	std.memory
<atomic>, <future>, <condition_variable>, <mutex>, <shared_mutex>	→	std.threading
그 외 C++ 표준 헤더	→	std.core

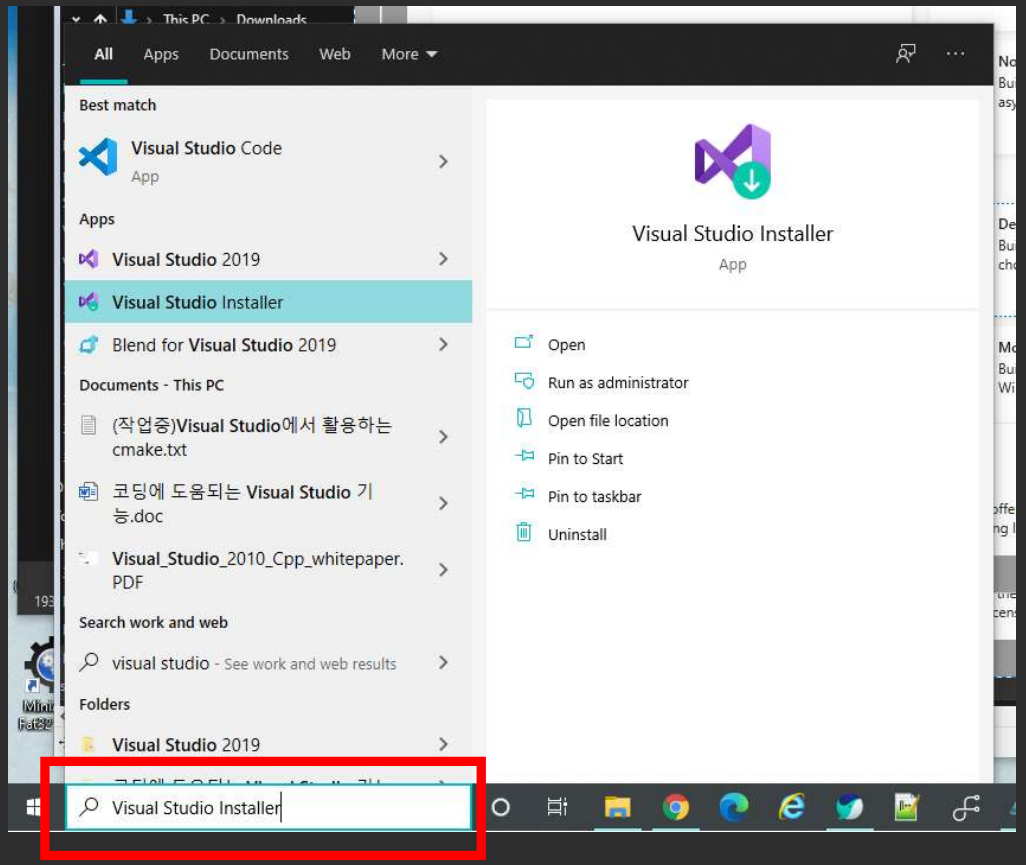
모듈(Modules)-훑어보기(실습준비1)

- Visual Studio 2019 설치
  - https://visualstudio.microsoft.com/ko/downloads



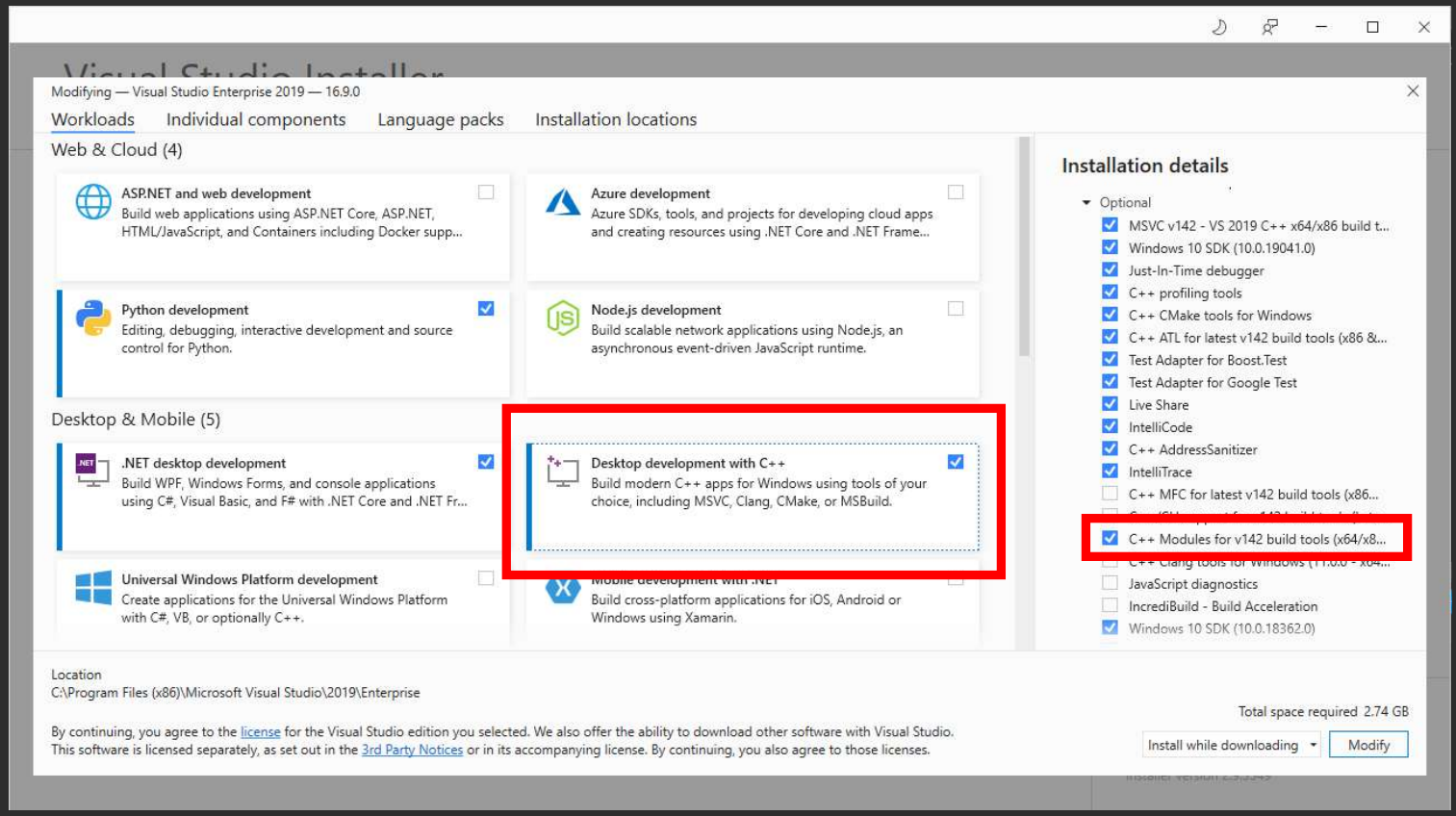
모듈(Modules)-훑어보기(실습준비2)

- Visual Studio Installer 검색, 실행



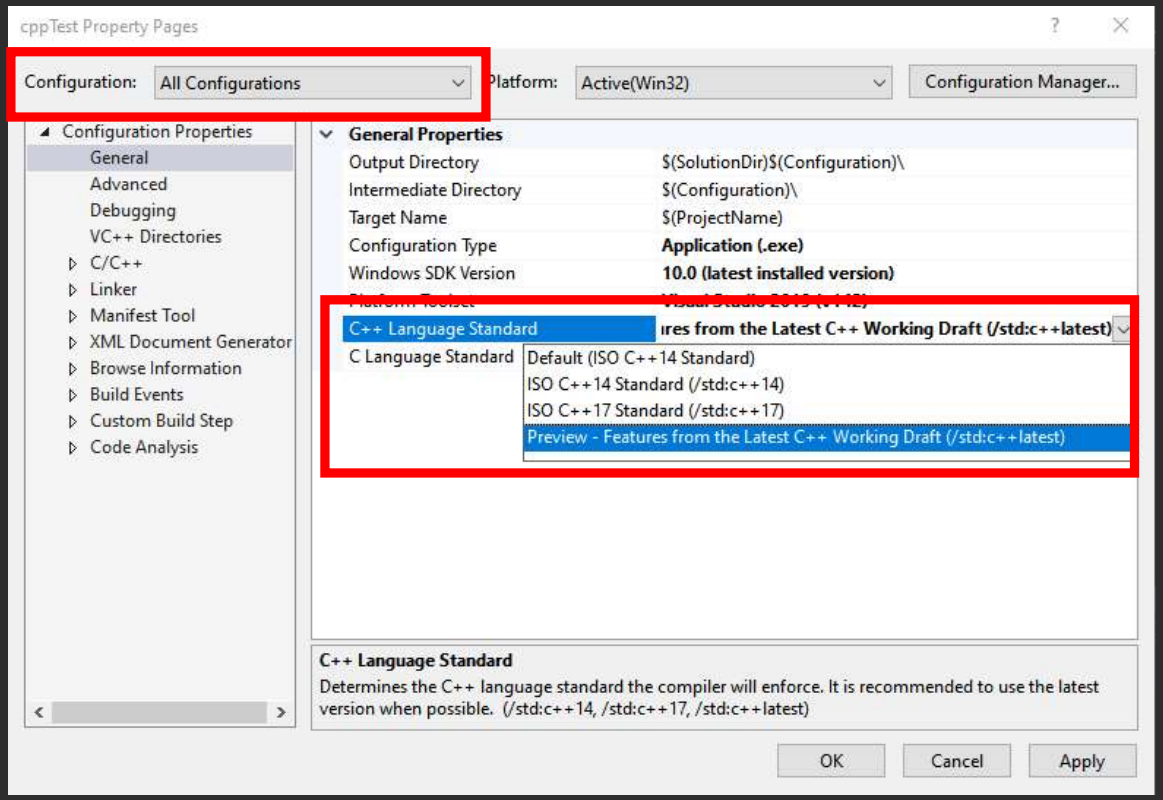
모듈(Modules)-훑어보기(실습준비3)

- C++ Modules for build tools 설치



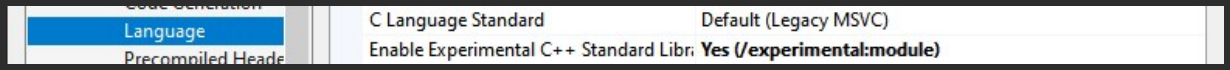
모듈(Modules)-훑어보기(컴파일)

- 프로젝트 속성 > 일반(General) > C++ 언어 표준 > /std:c++latest

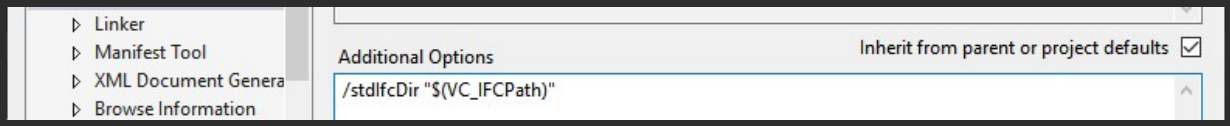


Error(C1011) Troubleshoot

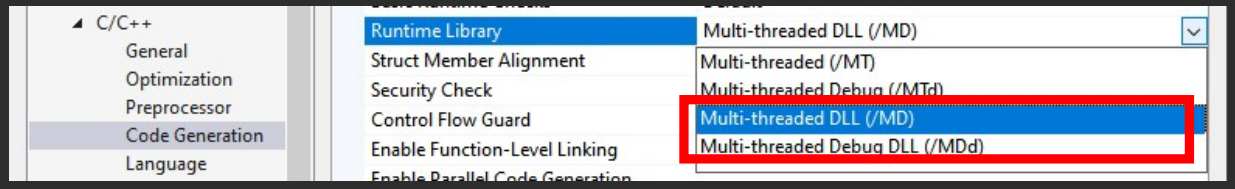
프로젝트/파일 속성 > C/C++ > Language > Experimental Features



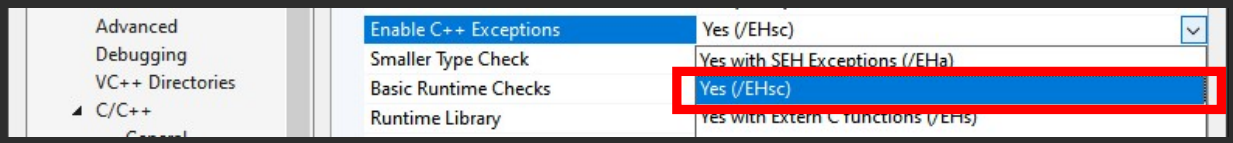
프로젝트/파일 속성 > C/C++ > Command Line > /stdIfcDir "$(VC_IFCPath)"



프로젝트/파일 속성 > C/C++ > Code Generation > Runtime Library



프로젝트/파일 속성 > C/C++ > Code Generation > Enable C++ Exceptions



CppKorea Facebook Community Group

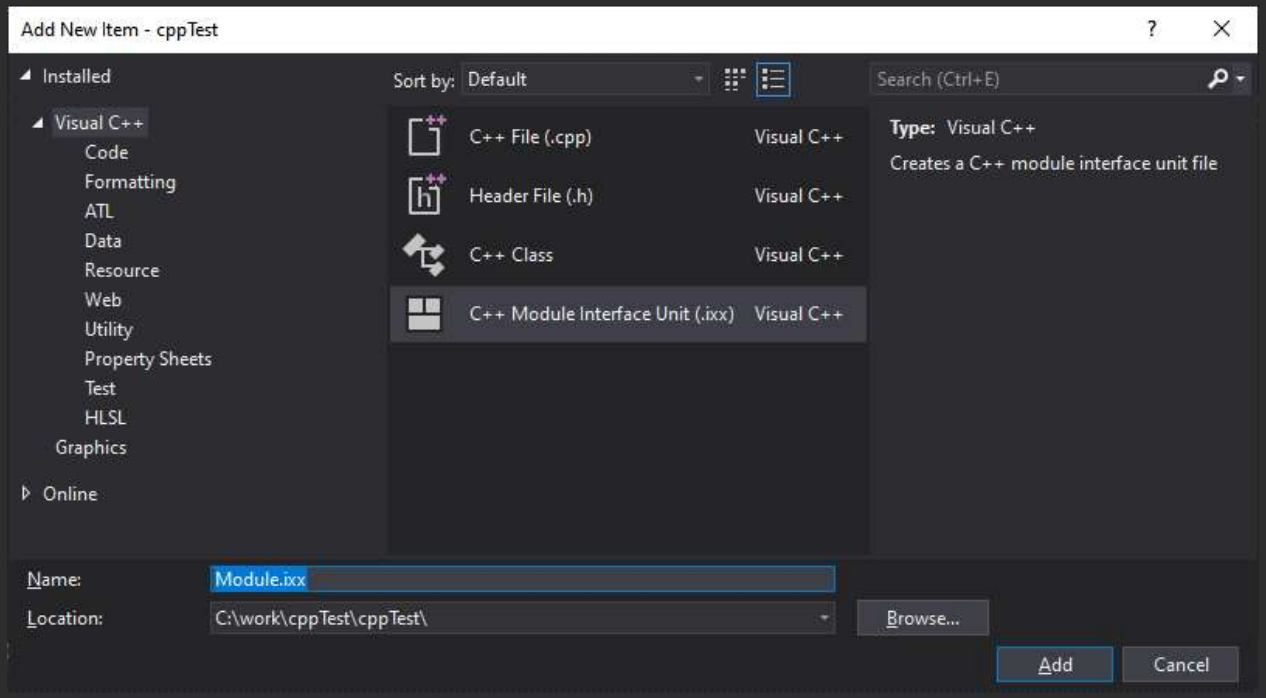
CppKorea8thSeminar.cpp

DEMO

모듈(Modules)-훑어보기

모듈(Modules)-만들기(확장자1)

- MSVC(마이크로소프트) : .ixx
- GCC/Clang : .cpp (초창기 Clang은 .cppm 사용)



모듈(Modules)-만들기(확장자2)

- Visual Studio IDE는 .ixx와 .cppm을 자동인식(인텔리센스 등)
- 프로젝트/파일 속성 > C/C++ > Advanced > Compile As에서 변경

종류	확장자
C 소스코드	.c
C++ 소스코드	.cpp, .c++, .cc, .cxx
헤더	.h, .h++, .hxx, .hpp
인라인	.inl, (GCC).ii, .ixx, .ipp
템플릿	.txx, .tpp, .tpl
모듈	(VS).ixx, (Clang).c++m, .cppm, .cxxm

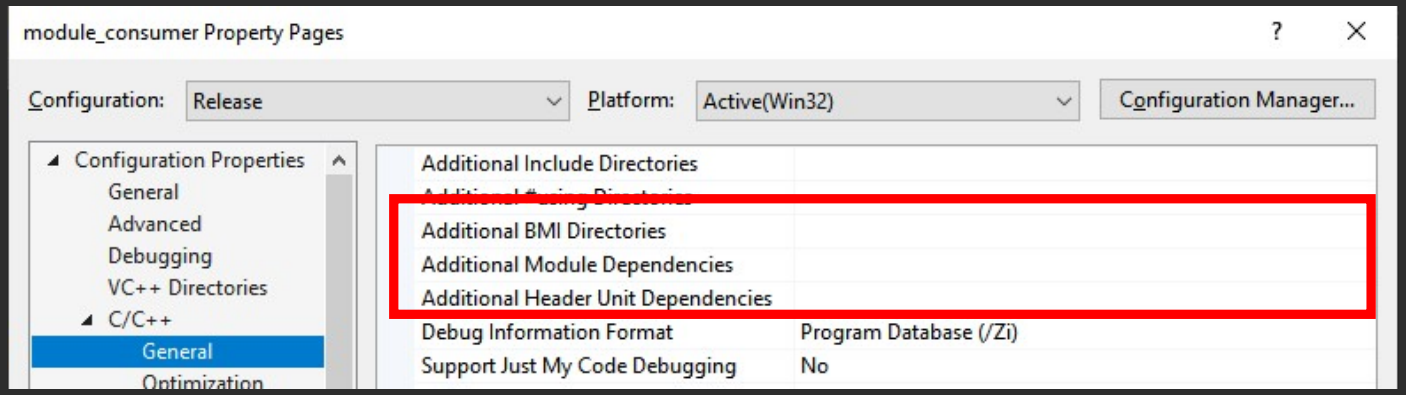
모듈(Modules)-ixx 파일 구조

```
module;
#include <iostream>

export module messages;

import std.filesystem;

/*non-export*/ struct Impl;
/*non-export*/ int helloWorld_internal();

export int helloWorld(); // helloworld() 함수 노출
export {                 // hello()와 world()를 한꺼번에 노출
  int hello();
  int world();
}
export namespace messages { // 네임스페이스 노출
  int hello();
  int world();
}

module :private;
struct Impl {
  void do_stuff() {/* ... */}
}
```

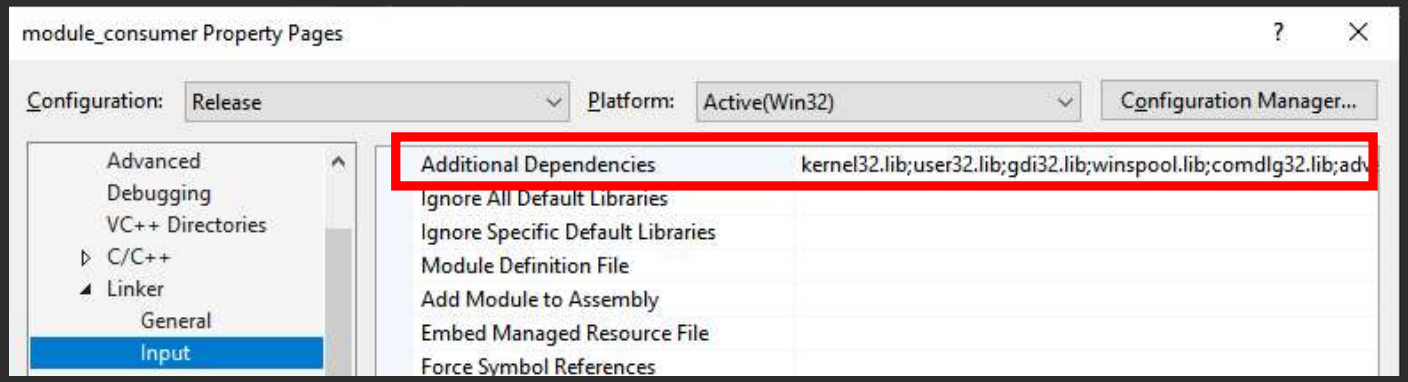
```
module; // optional
 // 전역모듈영역(Global module fragment)
 // 전처리 지시자 전용
export module 모듈이름;

// 임포트 선언 전용

// 모듈범위(Module purview)

module :private; // optional
   // 프라이빗모듈영역(Private module fragment)
```

모듈(Modules)-프로젝트 설정

- 프로젝트/파일 속성 > C/C++ > General > Additional Dependencies



- 프로젝트/파일 속성 > Linker > Input > Additional Dependencies



CppKorea Facebook Community Group

CppKorea8thSeminar.cpp

DEMO

모듈(Modules)-만들기

모듈(Modules)-컴파일

- 모듈 컴파일 시 .ifc, obj 파일 생성
- 두 파일을 공유, 외부 프로젝트에서 기능 사용 가능

```
# MSVC
cl /std:c++latest /EHsc /experimental:module /MD /c Module.ixx
cl /std:c++latest /EHsc /experimental:module /reference messages=Module.ifc /MD Source.cpp messages.obj
```

```
# Clang
clang++ -std=c++2a -stdlib=libc++ -c Module.cpp -Xclang -emit-module-interface -o Module.pcm
clang++ -std=c++2a -stdlib=libc++ -fprebuilt-module-path=. client.cpp Module.pcm -o client
```

모듈 인터페이스/구현 유닛

```
// module_interface_unit.ixx
module;
#include <vector>
export module messages
export int generate_message(std::vector<int> const& vec);

// module_implementation_unit.cpp
module messages
#include <string>
int generate_message(std::vector<int> const& vec) {
  // ...
}
```

```
cl.exe /std:c++latest /EHsc /MD /experimental:module /c module_interface_unit.ixx
cl.exe /std:c++latest /EHsc /MD /experimental:module /c module_implementation_unit.cpp
cl.exe /std:c++latest /EHsc /MD /experimental:module /reference messages=module_interface_unit.ifc \
  Source.cpp module_interface_unit.obj module_implementation_unit.obj
```

서브모듈(Submodules)

```
// MainModule.ixx
export module main_module;
export import sub_module1;
export import sub_module2;
export char const* endmark = "!!";

// SubModule1.ixx
export module sub_module1;
export char const* hello = "hello";

// SubModule2.ixx
export module sub_module2;
export char const* world = "world";

// Source.cpp
#include <iostream>
import main_module;
int main() {
  std::cout << hello; std::cout << world; std::cout << endmark;
  return 0;
}
```

서브모듈(Submodules)-중첩

```
// 서브 모듈도 하나의 완전한 모듈
#include <iostream>
import sub_module1;
int main() {
  std::cout << hello;
  return 0;
}

// 중첩 사용시 문제없음
#include <iostream>
import main_module;
import sub_module1;
int main() {
  std::cout << hello; std::cout << world; std::cout << endmark;
  return 0;
}
```

모듈 파티션(Module Partitions)

```
// module_partition1.ixx
export module module_partition:module_partition1;
export char const* hello = "hello";

// module_partition2.ixx
export module module_partition:module_partition2;
export char const* world = "world";

// module_partition.ixx
export module module_partition;
export import :module_partition1;
export import :module_partition2;
export char const* endmark = "!!";

// Source.cpp
#include <iostream>
import module_partition;
int main() {
  std::cout << hello; std::cout << world; std::cout << endmark;
  return 0;
}
```

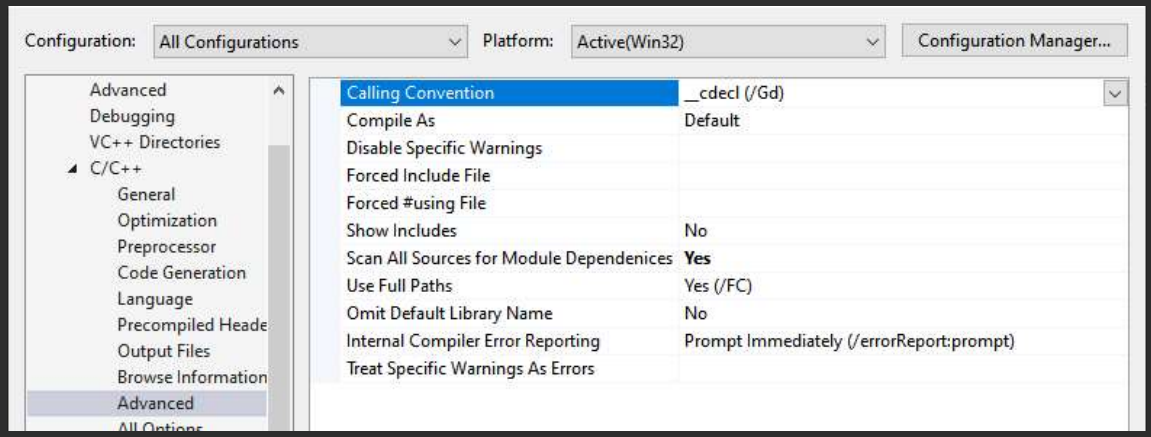
모듈안의 템플릿

```cpp
// 테스트코드
#include <iostream>
template <typename T, typename T2>
auto sum(T fir, T2 sec) {
  return fir + sec;
}
int main() {
  std::cout << sum(1, 1.5) << std::endl;
}

// 해석되는 템플릿 코드의 예
// https://cppinsights.io/
#include <iostream>
template <typename T, typename T2>
auto sum(T fir, T2 sec) {
  return fir + sec;
}
#ifdef INSIGHTS_USE_TEMPLATE
template<>
double sum<int, double>(int fir, double sec) {
  return static_cast<double>(fir) + sec;
}
#endif
int main() {
  std::cout.operator<<(sum(1, 1.5)).operator<<(std::endl);
}
```

모듈 헤더 유닛

- #include <iostream> -> import <iostream>
- #include “foo.h” -> import “foo.h”

- 프로젝트/파일 속성 > C/C++ > Language \
  > Enable Experimental C++ Standard Library(/experimental:module)
- 프로젝트/파일 속성 > C/C++ > Advanced \
  > Scan All Sources for Module Dependencies

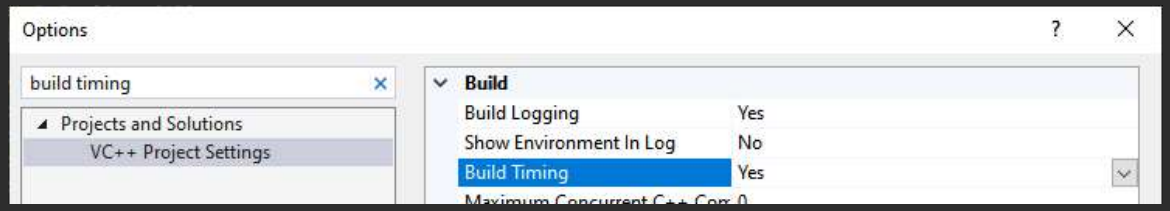


모듈 헤더 유닛-세부사항

- 헤더 파일을 모듈화 하여 컴파일 시간 단축이 목표
- 헤더 유닛은 헤더 안의 매크로 및 심볼 정의와 상태가 노출
- 기존 헤더파일들을 모듈로 전환하는 과도기 기술
- 가능하다면 모듈로 새롭게 만들 것을 권장

빌드시간 측정

- MSBuild
  Tools > Options > Project and Solutions > VC++ Project Settings
  > Build Timing을 Yes로 변경



- 컴파일(cl) 옵션 : **/Bt**, 링커 옵션 : /time
  time(...\c1xx.dll)=200s
  time(...\c2.dll)=100s

- 컴파일(cl) 옵션 : /Bt+, 링커 옵션 : /time+
  time(...\c1xx.dll)=90s < 100 - 10 > BB [main.cpp]
  time(...\c2.dll)=10s < 20 - 10 > BB [main.cpp]

CppKorea Facebook Community Group

CppKorea8thSeminar.cpp

DEMO

모듈 헤더 유닛

모듈 연결(Module Linkage)

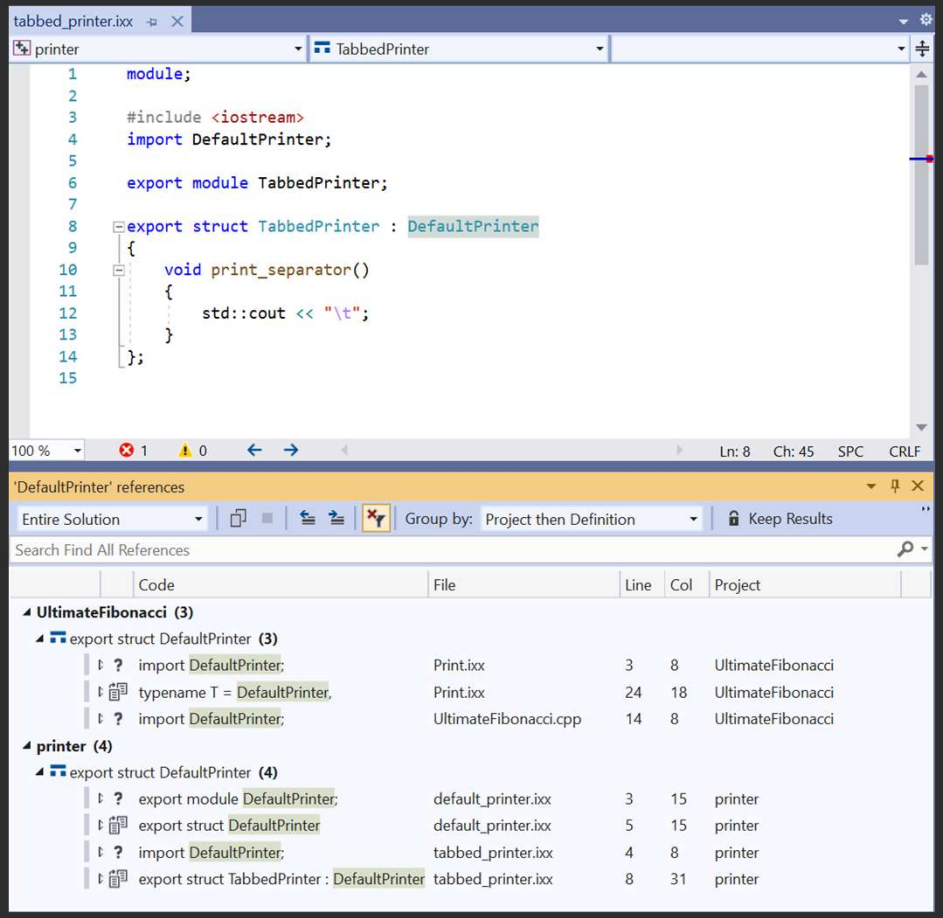
- 내부연결(Internal Linkage) : 단일 소스파일내에서만 사용 가능
  - static 키워드
  - translation unit안에서만 사용 가능

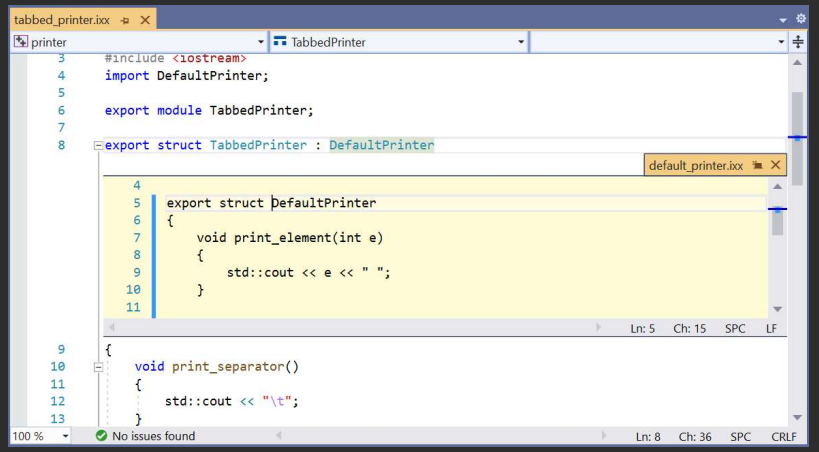
- 외부연결(External Linkage) : 외부 소스에서도 사용가능
  - extern을 명시하거나 static을 명시하지 않은것
  - link된 모든 translation unit에서 사용가능

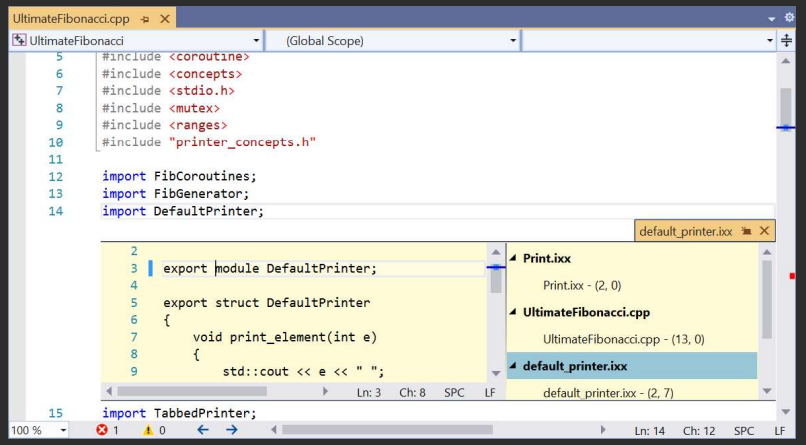
- 모듈연결(Module Linkage) : 모듈안에서 사용 가능

Visual Studio IDE 지원

- Find All References, Peek Definition, Go To Definition







모듈은 현재 완성도를 높이는 중

```
// PublicModule.ixx
export module PublicModule;
export struct PublicType { int x; };
struct PrivateType { int x; };
```

```
// main.cpp
import PublicModule;
int main() {
  PublicType publicType;
  PrivateType privateType; // error C2065: 'PrivateType': undeclared identifier
  return 0;
}
```

```
// main.cpp
import PublicModule;
struct PrivateType { int x; };
int main() {
  PublicType publicType;
  PrivateType privateType; // error C1117: unrecoverable error importing module
                           // 'PublicModule': symbol 'PrivateType' has already been defined
  return 0;
}
```

모듈적용시 고려할 점

- 컴파일러 기능은 개발되었지만 버그 수정 등 개선 필요
- 아직 모든 컴파일러가 Modules를 지원하지는 않음
- C++ 20을 지원하는 코드만 사용 가능
- 단일 프로젝트의 rebuild all 시간이 줄어든 것은 아님
- 파일 경로('\')를 이름으로 사용할 수 없음
- 순환 의존성(Circular Dependencies)을 가질 수 없음

Reference

- A Tour of C++ Modules in Visual Studio(2020.10.29)
https://devblogs.microsoft.com/cppblog/a-tour-of-cpp-modules-in-visual-studio

- Standard C++20 Modules support with MSVC in Visual Studio 2019 version 16.8(2020.9.14)
https://devblogs.microsoft.com/cppblog/standard-c20-modules-support-with-msvc-in-visual- \
studio-2019-version-16-8

- C++ Modules conformance improvements with MSVC in Visual Studio 2019 16.5(2020.1.22)
https://devblogs.microsoft.com/cppblog/c-modules-conformance-improvements-with-msvc- \
in-visual-studio-2019-16-5

- Better template support and error detection in C++ Modules with MSVC 2017 \
version 15.9(2018.11.27)
https://devblogs.microsoft.com/cppblog/better-template-support-and-error-detection-in- \
c-modules-with-msvc-2017-version-15-9/

- Using C++ Modules in Visual Studio 2017(2017.5.5)
https://devblogs.microsoft.com/cppblog/cpp-modules-in-visual-studio-2017

- C++ Modules in VS 2015 Update 1(2015.12.3)
https://devblogs.microsoft.com/cppblog/c-modules-in-vs-2015-update-1

CppKorea Facebook Community Group

CppKorea8thSeminar.cpp - Session End

C++ Modules 실무활용 언리쉬드

drvoss@gmail.com