سلسلة تعلم البرمجة بلغة C++ الحديثة

Learn Modern C++ Programming Course

إعداد المهندس أحمد الديب

#30: C++20 Modules

Header Files

- A `.cpp` file that is compiled by itself (including the h files it `#includes`) is called a translation unit.
- Compilation time: If you `#include header.h` in 101 translation units, the text of header.h will be processed by the compiler 101 times.
- Order dependencies: declarations and macros in the first included header might affected by the following header files.
- Inconsistencies: Defining an entity, such as a type or a function, in one file and then defining it slightly differently in another file.
- Transitivity: All code that is needed to express a declaration in a header file must be present in that header file.

Source: A Tour of C++ (3th Edition), Bjarne Stroustrup

Modules Example

```
// math.cc - Module Declaration File
module;  // global module fragment

export module math;

export double add(double a, double b) { return a + b; }
export double sub(double a, double b) { return a - b; }
export double mul(double a, double b) { return a * b; }


// main.cc
import <iostream>;
import math;


int main() {  //
  std::cout << "Hello Modules" << std::endl;
  std::cout << "3 + 5 = " << add(3, 5) << std::endl;
}
```

Build Process

```
// Generate precompiled module file
clang++ -std=c++20 -c -Xclang -emit-module-interface math.cc -o math.pcm

// Compile C++ unit which uses module file.
clang++ -std=c++20 -fmodules -c -fprebuilt-module-path=. main.cc -o main.o

// Generate executable
clang++ -std=c++20 -fmodules -o main main.o math.pcm
```

Why Modules?

- Faster than include headers. Up to 10 times.
- Compiled once only (rather than in each translation unit in which it is used).
- Import order does not affect the meaning.
- Import is not transitive: users do not implicitly gain access to everything inside modules.
- Declarations do not have to be separated into header files.

Source: A Tour of C++ (3th Edition), Bjarne Stroustrup

Export

```
export { // Export Group
  double add(double a, double b) { return a + b; }
  double sub(double a, double b) { return a - b; }
  double mul(double a, double b) { return a * b; }
}

export namespace math {  // Export Namespace
double add(double a, double b) { return a + b; }
double sub(double a, double b) { return a - b; }
double mul(double a, double b) { return a * b; }
}  // namespace math
```

You can also use the export specifier, the export group, and the export namepace inside a namespace. In this case, only exported names are visible to a consumer of the module.

Source: C++20 Get the Details, Rainer Grimm

Module Interface & Implementation

```cpp
// 04-math-if.cc
module;  // global module fragment

export module math;

export {
  double add(double a, double b);
  double sub(double a, double b);
  double mul(double a, double b);
}
```

```cpp
// 04-math-imp.cc
module math;


double add(double a, double b) { return a + b; }
double sub(double a, double b) { return a - b; }
double mul(double a, double b) { return a * b; }
```

```
// Building with separate module interface and implementation units
clang++ -std=c++20 -c -Xclang -emit-module-interface 04-math-if.cc -o math.pcm
clang++ -std=c++20 -fmodules -c -fprebuilt-module-path=. 04-math-imp.cc
clang++ -std=c++20 -fmodules -c -fprebuilt-module-path=. main.cc
clang++ -std=c++20 -fmodules -o main main.o 04-math-imp.o math.pcm
```

Changes to the implementation file does not require rebuilding the module interface file and the file importing the module.

Private Module Fragment

```
module;  // global module fragment

export module math;

export {
  double add(double a, double b);
  double sub(double a, double b);
  double mul(double a, double b);
}

module : private;  // private module fragment

double add(double a, double b) { return a + b; }
double sub(double a, double b) { return a - b; }
double mul(double a, double b) { return a * b; }
```

Changes to the private module fragment does not require rebuilding the file importing the module.

Submodules

```cpp
// 06-math-p1.cc - Module Declaration File
module;  // global module fragment

export module math.p1;

export double add(double a, double b) { return a + b; }
export double sub(double a, double b) { return a - b; }
```

```cpp
// 06-math-p2.cc - Module Declaration File
module;  // global module fragment

export module math.p2;

export double mul(double a, double b) { return a * b; }
```

```cpp
// 06-math.cc - Module Declaration File
module;  // global module fragment

export module math;

export import math.p1;
export import math.p2;
```

```cpp
// main.cc
import<iostream>;

import math; // or import math.p1;

int main() {  //
  std::cout << "Hello Modules" << std::endl;
  std::cout << "3 + 5 = " << add(3, 5) << std::endl;
}
```

Building Submodules

```
// Building submodules
clang++ -std=c++20 -c -Xclang -emit-module-interface 06-math-p1.cc -o math.p1.pcm
clang++ -std=c++20 -c -Xclang -emit-module-interface 06-math-p2.cc -o math.p2.pcm
clang++ -std=c++20 -c -fprebuilt-module-path=. -Xclang -emit-module-interface 06-math.cc
-o math.pcm
clang++ -std=c++20 -fmodules -c -fprebuilt-module-path=. main.cc
clang++ -std=c++20 -fmodules -o main main.o math.p1.pcm math.p2.pcm math.pcm
```

Module Partitions

```cpp
// 07-math-p1.cc
module;  // global module fragment

export module math : p1;

export double add(double a, double b) { return a + b; }
export double sub(double a, double b) { return a - b; }
```

```cpp
// 07-math-p2.cc
module;  // global module fragment

export module math : p2;

export double mul(double a, double b) { return a * b; }
```

```cpp
// 07-math.cc
module;  // global module fragment

export module math;

export import : p1;
export import : p2;
```

```cpp
// main.cc
import<iostream>;

import math; // Error if import math:p1;

int main() {  //
  std::cout << "Hello Modules" << std::endl;
  std::cout << "3 + 5 = " << add(3, 5) << std::endl;
}
```

Building Module Partitions

```
// Building module partitions
clang++ -std=c++20 -c -Xclang -emit-module-interface 07-math-p1.cc -o math-p1.pcm
clang++ -std=c++20 -c -Xclang -emit-module-interface 07-math-p2.cc -o math-p2.pcm
clang++ -std=c++20 -c -fprebuilt-module-path=. -Xclang -emit-module-interface 07-
math.cc -o math.pcm
clang++ -std=c++20 -fmodules -c -fprebuilt-module-path=. main.cc
clang++ -std=c++20 -fmodules -o main main.o math-p1.pcm math-p2.pcm math.pcm
```

Submodules Vs Partitions

Submodules	Partitions
Not part of the top modules, can be used separately.	Part of the top module and can not be used separately.
Used to publicly split large module.	Used to privately split large module.

Header Units

```cpp
// 09-myheader.h
#include <iostream>

void print_hello();
```

```cpp
// 09-myheader.cc
import "09-myheader.h";

void print_hello() {  //
  std::cout << "Hello header units" << std::endl;
}
```

```cpp
// 09-myheader-main.cc
import "09-myheader.h";

int main() { print_hello(); }
```

Not all headers can be converted into modules, but C++ guarantees that all standard library headers are importable headers, except C headers.

```
// Building header units
clang++ -std=c++20 -fmodule-header 09-myheader.h -o myheader.pcm
clang++ -std=c++20 -fmodule-file=myheader.pcm -c 09-myheader.cc
clang++ -std=c++20 -fmodule-file=myheader.pcm -o main 09-myheader.o 09-myheader-main.cc
```

Source: C++20 Get the Details, Rainer Grimm

Thank you