Object-oriented scientific programming with C++

Matthias Möller, Jonas Thies, Cálin Georgescu, Jingya Li (Numerical Analysis, DIAM)
Lecture 7

Goal of this lecture

- advanced data structures (`union`, `std::any`, `std::variant`, `std::tuple`)
- structuring C++ code into multiple files (build process: preprocessor, compiler, linker)
- building code with CMake

Unions

C++ is a **strongly typed** programming language meaning that each object has a fixed type.
Don't confuse this with *casting*, e.g., `int i = 1.6f;`. Here, variable `i` is of type `int` and the value `1.6f` is downcasted.
A `union` is a special class type that can hold one of its non-static data member at a time.
Example:

In [1]:

```
union S {
    int   i;
    float f;
};
```

We can now create an object and assign a value to the first data member (int i):

In [12]:

```
#include <iostream>

S s{1234};
std::cout << s.i << std::endl;
```

```
1234
```

The content of the second data member is undefined/meaningless

In [11]:

```
std::cout << s.f << std::endl;
```

```
1.7292e-42
```

We can assign new values to the existing object `s` even for different data members

In [13]:

```
s.i = 4321; std::cout << s.i << std::endl;
s.f = 1.6f; std::cout << s.f << std::endl;
```

```
4321
1.6
```

A `union` always holds the value of the most recently assigned data member. Reading from another data member is undefined behavior.

Unions are a special class type with some **limitations** over regular classes and structures:

- a `union` can have member functions (including constructors and destructors), but not virtual functions.
- a `union` cannot have base classes and cannot be used as a base class.
- a `union` cannot have non-static data members of reference types.
- a `union` is at least as big as necessary to hold its largest data member.

std::variant

In C++17 and later, the std::variant class is a type-safe alternative for a union.
Example:

In [1]:

```
#include <iostream>
#include <variant>

std::variant<int, float> v{1234};
std::cout << std::get<int>(v) << std::endl;
std::cout << std::get<float>(v) << std::endl;
```

```
1234
```

```
Standard Exception: std::get: wrong index for variant
```

In [2]:

```
v = 1.6f;
std::cout << std::get<float>(v) << std::endl;
std::cout << std::get<int>(v) << std::endl;
```

```
1.6
```

```
Standard Exception: std::get: wrong index for variant
```

Since `std::get<TYPE>` throws an exception if the wrong `TYPE` is requested one can catch inappropriate accesses with a `try-catch` construction.
Alternatively, one can use `std::get_if` which returns a pointer to the value of a pointed-to variant or null. But don't forget to pass the argument by address and be aware that you get a pointer in return.
Example:

In [3]:

```
auto p = std::get_if<float>(&v);  std::cout << (p ? (*p) : 0) << std::endl;
auto q = std::get_if<int>(&v);    std::cout << (q ? (*q) : 0) << std::endl;
```

```
1.6
0
```

std::any

A union or std::variant can only hold values of predefined type, i.e. it is not possible to hold *any* data type. Since C++17, the container class std::any can hold single values of *any* copy constructible type.
Example:

In [3]:

```
#include <iostream>
#include <any>

std::any a = 1234;
std::cout << a.type().name() << ": " << std::any_cast<int>(a) << std::endl;

a = 1.6f;
std::cout << a.type().name() << ": " << std::any_cast<float>(a) << std::endl;
```

```
i: 1234
f: 1.6
```

Note: An `std::any` object cannot be accessed without `std::any_cast`

In [4]:

```
std::cout << a << std::endl;
```

```
input_line_13:2:12: error: invalid operands to binary expression ('std::ostream' (aka 'basic_os
tream<char>') and 'std::any')
 std::cout << a << std::endl;
 ~~~~~~~~~ ^  ~
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:245:7: note: candidate function not viable: no known conversion from 'std::any' to 'const void
*' for 1st argument; take the address of the argument with &
      operator<<(const void* __p)
      ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/cstddef
:125:5: note: candidate function template not viable: no known conversion from 'std::ostream' (
aka 'basic_ostream<char>') to 'std::byte' for 1st argument
    operator<<(byte __b, _IntegerType __shift) noexcept
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/system_
error:262:5: note: candidate function template not viable: no known conversion from 'std::any'
to 'const std::error_code' for 2nd argument
    operator<<(basic_ostream<_CharT, _Traits>& __os, const error_code& __e)
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:108:7: note: candidate function not viable: no known conversion from 'std::any' to 'std::basic
_ostream<char, std::char_traits<char> >::__ostream_type &(*)(std::basic_ostream<char, std::char
_traits<char> >::__ostream_type &)' (aka 'basic_ostream<char, std::char_traits<char> > &(*)(bas
ic_ostream<char, std::char_traits<char> > &)') for 1st argument
      operator<<(__ostream_type& (*__pf)(__ostream_type&))
      ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:117:7: note: candidate function not viable: no known conversion from 'std::any' to 'std::basic
_ostream<char, std::char_traits<char> >::__ios_type &(*)(std::basic_ostream<char, std::char_tra
its<char> >::__ios_type &)' (aka 'basic_ios<char, std::char_traits<char> > &(*)(basic_ios<char,
std::char_traits<char> > &)') for 1st argument
      operator<<(__ios_type& (*__pf)(__ios_type&))
      ^
```

```
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:127:7: note: candidate function not viable: no known conversion from 'std::any' to 'std::ios_b
ase &(*)(std::ios_base &)' for 1st argument
      operator<<(ios_base& (*__pf) (ios_base&))
      ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:166:7: note: candidate function not viable: no known conversion from 'std::any' to 'long' for
1st argument
      operator<<(long __n)
      ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:170:7: note: candidate function not viable: no known conversion from 'std::any' to 'unsigned l
ong' for 1st argument
      operator<<(unsigned long __n)
      ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:174:7: note: candidate function not viable: no known conversion from 'std::any' to 'bool' for
1st argument
      operator<<(bool __n)
      ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:178:7: note: candidate function not viable: no known conversion from 'std::any' to 'short' for
1st argument
      operator<<(short __n);
      ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:181:7: note: candidate function not viable: no known conversion from 'std::any' to 'unsigned s
hort' for 1st argument
      operator<<(unsigned short __n)
      ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:189:7: note: candidate function not viable: no known conversion from 'std::any' to 'int' for 1
st argument
      operator<<(int __n);
      ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:192:7: note: candidate function not viable: no known conversion from 'std::any' to 'unsigned i
nt' for 1st argument
      operator<<(unsigned int __n)
      ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:201:7: note: candidate function not viable: no known conversion from 'std::any' to 'long long'
for 1st argument
      operator<<(long long __n)
      ^
```

```
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:205:7: note: candidate function not viable: no known conversion from 'std::any' to 'unsigned l
ong long' for 1st argument
      operator<<(unsigned long long __n)
      ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:220:7: note: candidate function not viable: no known conversion from 'std::any' to 'double' fo
r 1st argument
      operator<<(double __f)
      ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:224:7: note: candidate function not viable: no known conversion from 'std::any' to 'float' for
1st argument
      operator<<(float __f)
      ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:232:7: note: candidate function not viable: no known conversion from 'std::any' to 'long doubl
e' for 1st argument
      operator<<(long double __f)
      ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:250:7: note: candidate function not viable: no known conversion from 'std::any' to 'std::nullp
tr_t' (aka 'nullptr_t') for 1st argument
      operator<<(nullptr_t)
      ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:276:7: note: candidate function not viable: no known conversion from 'std::any' to 'std::basic
_ostream<char, std::char_traits<char> >::__streambuf_type *' (aka 'basic_streambuf<char, std::c
har_traits<char> > *') for 1st argument
      operator<<(__streambuf_type* __sb);
      ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:511:5: note: candidate function template not viable: no known conversion from 'std::any' to 'c
har' for 2nd argument
    operator<<(basic_ostream<_CharT, _Traits>& __out, char __c)
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:517:5: note: candidate function template not viable: no known conversion from 'std::any' to 'c
har' for 2nd argument
    operator<<(basic_ostream<char, _Traits>& __out, char __c)
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:523:5: note: candidate function template not viable: no known conversion from 'std::any' to 's
igned char' for 2nd argument
    operator<<(basic_ostream<char, _Traits>& __out, signed char __c)
```

```
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:528:5: note: candidate function template not viable: no known conversion from 'std::any' to 'u
nsigned char' for 2nd argument
    operator<<(basic_ostream<char, _Traits>& __out, unsigned char __c)
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:606:5: note: candidate function template not viable: no known conversion from 'std::any' to 'c
onst char *' for 2nd argument
    operator<<(basic_ostream<char, _Traits>& __out, const char* __s)
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:619:5: note: candidate function template not viable: no known conversion from 'std::any' to 'c
onst signed char *' for 2nd argument
    operator<<(basic_ostream<char, _Traits>& __out, const signed char* __s)
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:624:5: note: candidate function template not viable: no known conversion from 'std::any' to 'c
onst unsigned char *' for 2nd argument
    operator<<(basic_ostream<char, _Traits>& __out, const unsigned char* __s)
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/bits/os
tream.tcc:321:5: note: candidate function template not viable: no known conversion from 'std::a
ny' to 'const char *' for 2nd argument
    operator<<(basic_ostream<_CharT, _Traits>& __out, const char* __s)
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:506:5: note: candidate template ignored: deduced conflicting types for parameter '_CharT' ('ch
ar' vs. 'std::any')
    operator<<(basic_ostream<_CharT, _Traits>& __out, _CharT __c)
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/string_
view:621:5: note: candidate template ignored: could not match 'basic_string_view<type-parameter
-0-0, type-parameter-0-1>' against 'std::any'
    operator<<(basic_ostream<_CharT, _Traits>& __os,
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/bits/ba
sic_string.h:6480:5: note: candidate template ignored: could not match 'basic_string<type-param
eter-0-0, type-parameter-0-1, type-parameter-0-2>' against 'std::any'
    operator<<(basic_ostream<_CharT, _Traits>& __os,
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/bits/sh
ared_ptr.h:69:5: note: candidate template ignored: could not match '__shared_ptr<type-parameter
-0-2, _Lp>' against 'std::any'
    operator<<(std::basic_ostream<_Ch, _Tr>& __os,
```

```
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/bits/va
larray_after.h:413:5: note: candidate template ignored: could not match '_Expr' against 'basic_
ostream'
    _DEFINE_EXPR_BINARY_OPERATOR(<<, struct std::__shift_left)
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/bits/va
larray_after.h:344:5: note: expanded from macro '_DEFINE_EXPR_BINARY_OPERATOR'
    operator _Op(const _Expr<_Dom1, typename _Dom1::value_type>& __v,   \
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/bits/va
larray_after.h:413:5: note: candidate template ignored: could not match '_Expr' against 'basic_
ostream'
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/bits/va
larray_after.h:357:5: note: expanded from macro '_DEFINE_EXPR_BINARY_OPERATOR'
    operator _Op(const _Expr<_Dom, typename _Dom::value_type>& __v,     \
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/bits/va
larray_after.h:413:5: note: candidate template ignored: could not match '_Expr<type-parameter-0
-0, typename type-parameter-0-0::value_type>' against 'std::any'
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/bits/va
larray_after.h:370:5: note: expanded from macro '_DEFINE_EXPR_BINARY_OPERATOR'
    operator _Op(const typename _Dom::value_type& __t,                  \
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/bits/va
larray_after.h:413:5: note: candidate template ignored: could not match '_Expr' against 'basic_
ostream'
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/bits/va
larray_after.h:383:5: note: expanded from macro '_DEFINE_EXPR_BINARY_OPERATOR'
    operator _Op(const _Expr<_Dom,typename _Dom::value_type>& __e,      \
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/bits/va
larray_after.h:413:5: note: candidate template ignored: could not match '_Expr<type-parameter-0
-0, typename type-parameter-0-0::value_type>' against 'std::any'
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/bits/va
larray_after.h:396:5: note: expanded from macro '_DEFINE_EXPR_BINARY_OPERATOR'
    operator _Op(const valarray<typename _Dom::value_type>& __v,        \
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/valarra
y:1193:1: note: candidate template ignored: could not match 'valarray' against 'basic_ostream'
_DEFINE_BINARY_OPERATOR(<<, __shift_left)
^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/valarra
y:1155:5: note: expanded from macro '_DEFINE_BINARY_OPERATOR'
    operator _Op(const valarray<_Tp>& __v, const valarray<_Tp>& __w)    \
```

```
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/valarra
y:1193:1: note: candidate template ignored: could not match 'valarray' against 'basic_ostream'
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/valarra
y:1166:5: note: expanded from macro '_DEFINE_BINARY_OPERATOR'
    operator _Op(const valarray<_Tp>& __v,                              \
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/valarra
y:1193:1: note: candidate template ignored: could not match 'valarray<type-parameter-0-0>' agai
nst 'std::any'
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/valarra
y:1177:5: note: expanded from macro '_DEFINE_BINARY_OPERATOR'
    operator _Op(const typename valarray<_Tp>::value_type& __t,         \
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:589:5: note: candidate template ignored: could not match 'const _CharT *' against 'std::any'
    operator<<(basic_ostream<_CharT, _Traits>& __out, const _CharT* __s)
    ^
/srv/conda/envs/notebook/bin/../lib/gcc/../../x86_64-conda-linux-gnu/include/c++/10.4.0/ostream
:773:5: note: candidate template ignored: requirement '__and_<std::__not_<std::is_lvalue_refere
nce<std::basic_ostream<char> &> >, std::__is_convertible_to_basic_ostream<std::basic_ostream<ch
ar> &>, std::__is_insertable<std::basic_ostream<char> &, const std::any &, void> >::value' was
not satisfied [with _Ostream = std::basic_ostream<char> &, _Tp = std::any]
    operator<<(_Ostream&& __os, const _Tp& __x)
    ^

Interpreter Error:
```

Separate compilation

C++ programs can vary greatly in size and complexity, as well as the number of people involved in the development.

Single-file programs

Problem

- Many people work on the same project.
- A lot of source code.
- Need long time to compile.
- Want to share codes among different projects.

Solution: Multi-file programs

- Divide source code into seperate files.
- Compile files separately.
- Only recompile files when modified.

Single-file programs vs. multi-file programs

	Single-file programs	Multi-file programs
Suitability	Ideal for **small, simple projects** (Most assignments in this course)	**Large, complex projects**
Compilation Time	Very long for large programs, taking hours or days	Much **faster** as files are compiled separately. Only changed files need recompilation.
Impact on Compiler	Large single files might strain or even break the compiler due to resource demands.	Less strain on the compiler as each file is smaller and compiled individually.
Team Collaboration	Difficult for team collaboration. Team members might interfere with each other's work.	Facilitates **parallel work**. Team members can work on separate files without interference, enhancing productivity and minimizing conflicts.
Modularity and Maintenance	Low.	High.

Linking vs. Compilation Time	Not applicable as there is only one compilation step.	Linking is required to combine the separately compiled files into a single executable. **Linking is generally much faster than compilation.**

Seperate compilation

Every ***source code*** file is transformed through compilation into an ***object code*** file. These object code files are then ***linked together*** to create the final ***executable program***.

Source Code:

- Program in human-readable form (C++ language).

Object Code:

- Binary code: low-level representation of the source code, it's not yet a standalone program.
- Exact addresses of variables and functions not known, represented by symbols.

Seperate compilation

Linking:

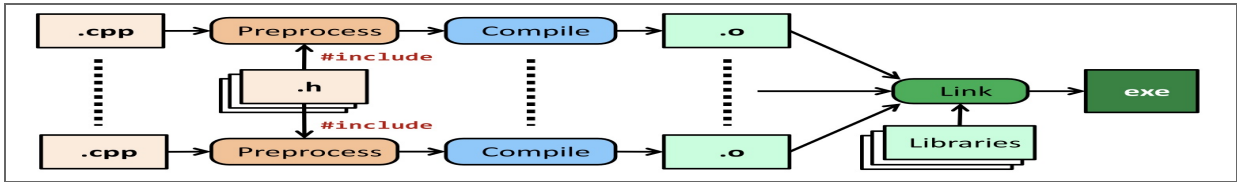
- The linker takes all the object code files and combines them to create a single executable file.
- It resolves references to symbols that are defined in different object files or libraries. For example, if one object file has a function call to a function defined in another object file, the linker connects these two.

Executable:

- Output of the linking process is the executable file, which can be run on a computer.
- This file contains all the code and resources needed to execute the program.

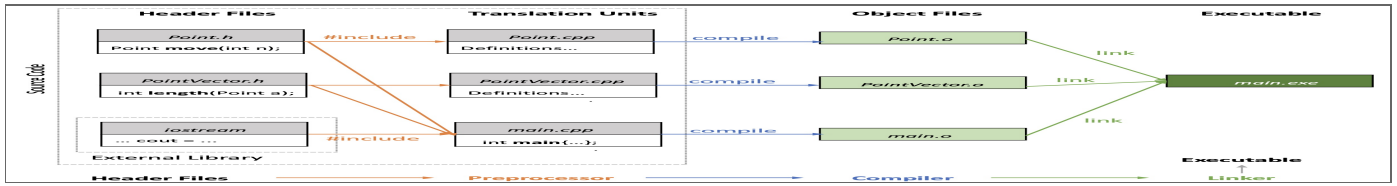
C++ build process

- **Headers** (*.h)* + ***Translation Units*** *(*.cpp) contain source code.
- **Preprocessor** performs text substitutions.
- **Compiler** translates translation units into object files.
- **Linker** links object files and *external libraries* into an executable.



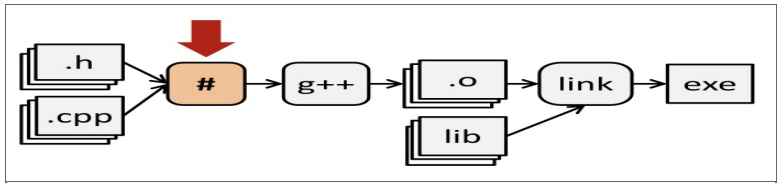
C++ build process

Based on Weblab assignment



Preprocessor

In the C++ build process, the preprocessor is the first step that runs before the actual compilation.



The primary functions of the preprocessor include:

- modifying the source code
- processing preprocessor instructions (handling lines beginning with `#`)
- stripping out comments

Preprocessor

Usage in C++ (and you should limit it to that)

- combining source code (`#include`)
- conditional compilation
- obtaining platform information during compilation

Preprocessor MACROs

Macro: a powerful feature provided by the C++ preprocessor, a tool that processes your code before it is compiled.

1. **Constant definitions**: Macros are often used to define constants. For example:

```
#define PI 3.14159
```

2. **Function-like macro:** Macros can also mimic functions. These are useful for small, repetitive code that doesn't need the overhead of a function call. For instance:

```
#define SQUARE(x) ((x)*(x))
```

NOTE: it is important to use parentheses around parameters and the entire definition to ensure correct order of operations when the macro is used.

Preprocessor MACROs

3. **Conditional compliation:** Macros can be used in conjunction with `#if`, `#ifdef`, `#ifndef`, and other preprocessor directives for conditional compilation.

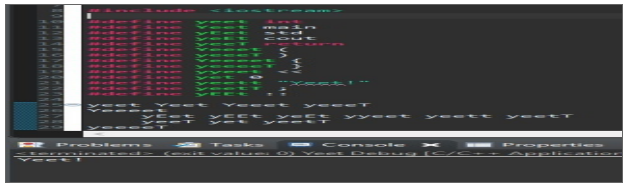
```
#define DEBUG

#ifdef DEBUG
// Your own debug code goes there
#endif
```

4. **Platform-specific code:** They can be used to compile code conditionally for different platforms or compilers.

```
#ifdef _WIN32
// Windows-specific code
#endif
```

Preprocessor MACROs

5. And more... (but don't do it!)



Special MACROs

- `__FILE__` (current file name), `__LINE__` (current line number), `__DATE__`(MMM DD YYYY) and `__TIME__` (hh:mm:ss)
- `__cplusplus`: This macro is defined when a source code file is being compiled by the C++ compiler. Its value reflects the version of the C++ standard used by the compiler.
    - C++98: 199711L
    - C++11: 201103L
    - C++14: 201402L
    - C++17: 201703L
    - C++20: 202002L

Example usage:

```
#if __cplusplus >= 201703L
    // C++17 (and later) code goes here
#endif
```

#include directive

#include: used for including other files into the source file, typically header files.
There are two common usages:

- Inserts a system header file from a location defined when the compiler was installed `#include <headerName>`. Example:

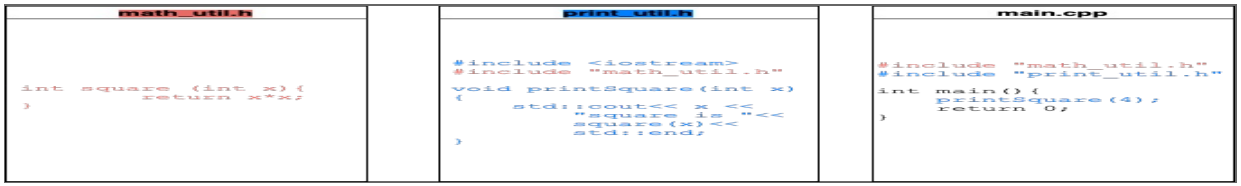
```
#include <iostream> // As we mentioned during lecture 1
```

- Inserts a file from the current directory. Example:

```
#include "filename"
```

NOTE:
- Since header files are included in one or more `*.cpp` files using the `#include` directive, the content of these header files can be compiled multiple times -- once for each `*.cpp` that includes them.
- Each `*.cpp` file is compiled only once to produce its corresponding object file (`*.o`).

Problems with the #include directive



math_util.h

```
int square (int x){
        return x*x;
}
```

print_util.h

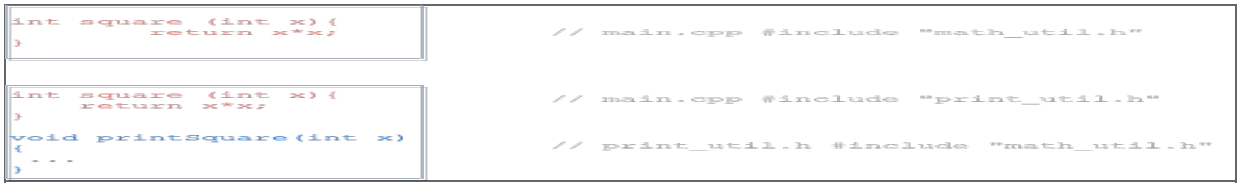
```
#include <iostream>
#include "math_util.h"
void print$quare(int x)
{
    std::cout<< x <<
        "square is "<<
        square(x)<<
        std::endl;
}
```

main.cpp

```
#include "math_util.h"
#include "print_util.h"
int main(){
    print$quare(4);
    return 0;
}
```

Problems with the #include directive continued

The definition of square(int x) is processed two times during the compilation of main.cpp.



Dealing with repeated `#includes`

The repeated inclusion of the same header file(s) can lead to **multiple definition errors** (conflicts during the linking stage) if not managed correctly.
To manage these issues, two common techniques can be used:

- Include guards
- `#pragma once`

Include guards

math_util.h

```
#ifndef MATH_UTIL.H
#define MATH_UTIL.H
int square (int x){
        return x*x;
}
#endif
```

print_util.h

```
#ifndef PRINT_UTIL.H
#define PRINT_UTIL.H
#include <iostream>
#include "math_util.h"
void printSquare(int x)
{
     std::cout<< x <<
          "square is "<<
          square(x)<<
          std::end;
}
#endif
```

main.cpp

```
#include "math_util.h"
#include "print_util.h"
int main(){
     printSquare(4);
     return 0;
}
```

#pragma directive

#pragma: a special preprocessor directive used to provide additional instructions to the compiler.
It is easier to use and less error-prone compared to traditional include guards.

```
#pragma once // tells the compiler to include a file only once in a single compilation
```

Other usage:

```
#pragma warning(disable: 4507) // disables a specific warning with the number 4507
#pragma warning(default: 4507) // re-enables the warning
```

Review: Declarations and definitions

A *declaration* in C++ serves to introduce or reiterate the name of an entity in the program.

```cpp
extern int globalVar; // Declaration of a variable
void printMessage();  // Declaration of a function
class MyClass;        // Forward declaration of a class
```

A *definition* in C++ goes beyond the declaration by not only **introducing or reiterating the name and specifying the type** but also by **providing the actual implementation or value and allocating storage**.

```cpp
int globalVar = 10;        // Definition of a variable
void printMessage() {      // Definition of a function
    std::cout << "Hello";
}
class MyClass {            // Definition of a class
public:
    void doSomething() {}
};
```

Organizing declarations and definitions into multiple files

Simple example:

math_util.h

```
class math_util{
public:
    int square(int x);
    int add(int a, int b);
private:
    //member variables
}
```

math_util.cpp

```
#include "math_util.h"
int math_util::square(int x){
    return (x*x);
}
int math_util::add(int a, int b){
    return (a + b);
}
```

What's in a header file ?

Header files are intended to contain:

- **Function declarations**: declare the functions you intend to use or implement. This tells the compiler about their existence and how they should be called.
- **Class declarations**: declare the classes you intend to use of implement. This tells the compiler about their existence and how they should be instantiated.
- **Templates**: since templates need to be known at compile time in every file where they are used, they are usually defined in header files.
- **Global constants and macros**: define constants and macros that are to be shared across multiple source files.
- **Inline functions**: small functions that benefit from being inline (like getters and setters) can be defined in header files.

Best practices

- **Include guards:** always use include guards (`#ifndef`, `#define`, `#endif`) or `#pragma once` in header files to prevent multiple inclusion issues.
- **Minimize dependencies:** include only what is **necessary** in header files to reduce compilation times and dependencies.
- **Consistent file naming:** keep a consistent naming scheme for your files. Typically, class names are used for the names of the corresponding header and source files.
- **Avoid using directives in headers**

Namespace pollution

Avoid in headers

- using namespaces

```
using namespace std;
```

- using symbols of a namespace

```
using std::cout;
```

WHY?

> *It forces all symbols from the specified namespace into the global namespace for all source files that include that header. This can lead to unexpected name conflicts and* ***namespace pollution****.*

Linking

```
$ g++ -c main.cpp -o main.o
$ g++ -c math_util.cpp -o math_util.o
$ g++ main.o math_util.o -o myexe
$ ./myexe
```

Explanation:

- Preprocessing and compiling `main.cpp` yields `main.o`
- Preprocessing and compiling `math_util.cpp` yields `math_util.o`
- Linking `math_util.o` and `main.o` yields executable `myexe`
- Run program `myexe`

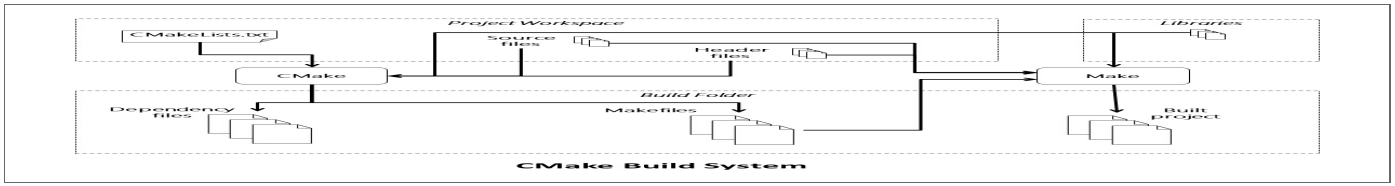
CMake: depart from the Weblab nutshell to the great universe

A solution to streamline the often tedious and complex task of managing linking in C++ projects.
CMake is **not** a build system like **Unix Make**) but a **build system generator**.
It provides a family of tools and a *domain-specific* language (DSL) to describe what the build system should achieve, you can reuse the same CMake scripts to obtain native build systems on any platform.

Generate and Build

The process of generating a `Makefile` and compiling with CMake on a Linux platform is as follows:

- Write the configuration file `CMakeLists.txt`.
- Run the command `cmake path/to/your/CMakeLists.txt` to generate a `Makefile`.
- Run the command `make` to compile your project.



CMake Build System

Hello CMake!

A minimal project:

```cpp
#include <iostream>

int main(){
    std::cout << "Hello CMake!" << std::endl;
    return 0;
}
```

Assuming the working directory only contains `main.cpp`. We need to add `CMakeLists.txt`.

```
# Set the requirement on minimum version of CMake
cmake_minimum_required(VERSION 3.9)

# Declare project and its programming language.
project(HelloCMake)
set(CMAKE_CXX_STANDARD 11)

# Create executable target and linking
add_executable(hellocmake main.cpp)
```

Hello CMake!

Now we are ready to call CMake and generate the `Makefile`:

```
cmake .
```

`.` represents the current directory.

Best practice: out-of-source build

By specifying the project source root (**-S** option) and target build location (**-B** option) on the command line:

```
cmake -S . -B build/
```

Hello CMake!

Classic Approach

The older CMake approach was to change to the build folder to explicitly run the build tool (**make**) from that folder:

```
mkdir build
cd build
cmake ..
make
```

The output should be look like this:

```
Scanning dependencies of target hellocmake
[ 50%] Building CXX object CMakeFiles/hellocmake.dir/main.cpp.o
[100%] Linking CXX executable hellocmake
[100%] Build target hellocmake
```

END …

For more information about CMake

CMake – The Dark Arts: **https://blog.feabhas.com/2021/07/cmake-part-1-the-dark-arts/**
CMake Documentation: **https://cmake.org/documentation/**
CMake hands-on workshop: **https://enccs.github.io/cmake-workshop/**
C++ Starter Project with Complete CMake Setup by Jason Turner: **https://github.com/cpp-best-practices/cmake_template**

For more information about C++ program structure

Back to Basics: Compiling and Linking - Ben Saks: **https://www.youtube.com/watch?v=cpkDQaYttR4**
Beginner's Guide to Linkers: **https://www.lurklurk.org/linkers/linkers.html**

Loading [MathJax]/extensions/Safe.js