COMP6771
Advanced C++ Programming

Week 10.2
Conclusion
(aka `~COMP6771()`)

COMP6771 in 60 Minutes or Less

a.k.a.: Revision

Week 01: C -> C++

- C++ is a general-purpose programming language:
- CPU-native types: `int`, `double`, `void*`, etc.
- Class-like types: `struct`, `class`, `union`
- Functions: `void foo(int, double*)`
- Opt-in immutability: `const int i` = `5`
- `auto`: `auto it = std::vector<int>{}.begin();`
- Value-semantics *and* reference semantics: `T/T&/T*`
- A rich standard library: `vector, tuple`, etc.
- Modular code-sharing: `#include<>`
- Separate compilation and linking

Week 02: STL

- **S**tandard **T**emplate **L**ibrary (STL)
- Containers, e.g.
  - `std::vector`
  - `std::list`
- Algorithms, e.g.
  - `std::copy()`
  - `std::transform`
- Iterators
  - Input, Output, RandomAccess
  - Glue between containers and algorithms

Week 03: Classes

- Scope
  - Functions, `for, if, while, {}, namespace` introduce scopes
  - Variables are accessible according to their scope
- Object Lifetime
  - Lifetime starts when brought into scope
  - Lifetime ends when the scope ends
- Classes are user-defined types that mirror primitives like `int`
  - Initialisation customisable through *constructors*
  - Clean-up customisable through *destructor*
- Internal entities of a class are *members*
  - Member functions
  - Data Members
  - Static member functions and static data members
  - API extensions through *friendship*

Week 04: Advanced Classes

- Operator-Overloading
  - Provide user-defined meanings for operators in C++
  - Chained-operations very easy to read
  - Make classes "feel" like primitives
  - e.g. `v1 + v2 == vec2d{v1.x + v2.x, v1.y + v2.y}` is more natural than `add(v1, v2)`
  - Full list of overloadable operators
- Exceptions
  - Classes that represent unexpected runtime errors
  - Dedicated syntax: `throw/try/catch`
  - Compiler-enforced *stack-unwinding*
  - Throw by value, catch by `const&` !!

Week 05: Resource Management

- C++ manages resources through RAII:
    - Acquire resources (memory, locks, etc.) in the constructor
    - Release them through the destructor
    - Every resource owned by an RAII class
    - Prevents resource leaks (by exceptions, forgetfulness, etc.)
- Ownership enforced through copy-control:
    - Able to prevent deep copies by deleting copy-constructor and copy-assign
    - Efficient transfer of ownership through move semantics
- RAII-conforming Smart Pointers replace "owning" pointers:
    - `std::unique_ptr<T>/T*` for unique ownership/observeration
    - `std::shared_ptr<T>/std::weak_ptr<T>` for shared ownership
    - Automatically free dynamically-allocated objects

Week 07: Templates

- Generic Programming through compile-time type paramerisation
- Function, Class, Alias, Variable, and Variadic templates
- Compiler synthesises function/class/typedef/variable definition from the template when required
  - Can be forced by explicit instantiation
- Primary template customisable through *specialisation*, either:
  - Fully (explicit specialisation); or
  - Partially (partial specialisation, only for class templates)
- Parameterisable by:
  - Types (e.g. `template <typename T>`
  - Non-type template parameters (e.g. `template <int N>`)
  - Template-template parameters (e.g. `template <template <typename> typename Container>`)

Week 08: TMP

- Templates are "accidentally" Turing-complete i.e. they can be used to calculate *anything*
- Type traits use templates to ask questions at compile-time:
  - Is `T` a pointer type (e.g. `int*`)?
  - What does `T` look like with `const` removed? (e.g. `const int` -> `int`)
  - Makes heavy use of struct templates and partial/explicit specialisation
  - Excessive use causes *incredibly* long compile-times and/or code bloat
- Forwarding references (`T&&`) introduced in C++11:
  - `auto` type deduction and rvalue references binds to anything
  - Can be used to "forward" arguments from one function to another whilst preserving rvalue-ness or lvalue-ness
- Modern C++ TMP moving away from abusing templates:
  - Constexpr-world: compile-time expressions e.g. `if-constexpr`
  - `decltype:` get the declared type of a variable at compile-time

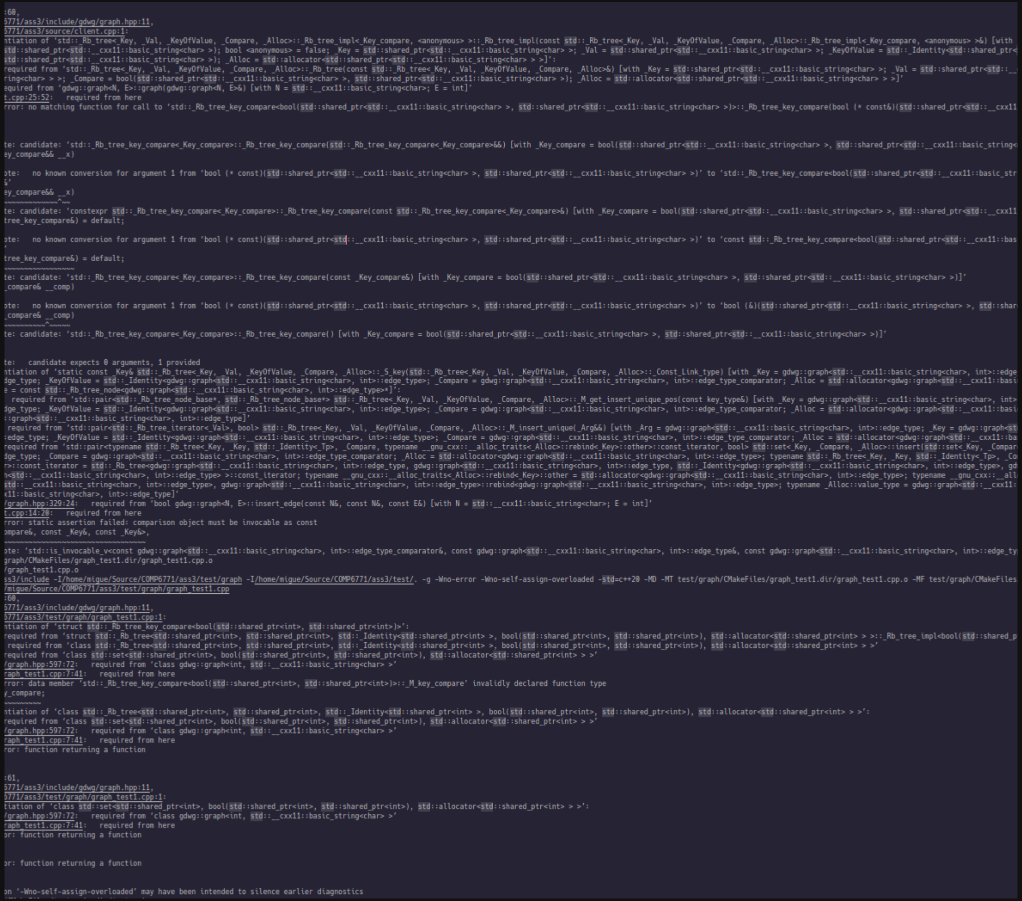
Week 09: Dynamic Polymorphism

- Classic OOP through *Dynamic Polymorphism*
  - Inheritance and derived classes
  - `virtual` methods
  - `override, final`, pure-virtual (*abstract*) methods
  - Early (at compile-time) binding vs. late (at runtime) binding
- Implemented through vtables:
  - Table of function pointers to virtual methods
  - Compiler-generated
- Can cast up and down type hierarchies with `dynamic_cast`
- Important considerations:
  - Polymorphic classes **must** have `virtual` destructors!
  - Dynamic polymorphism only happens for `T*` and `T&`!
  - Copying/moving a derived class into a base class causes *object slicing*

Week 10: Advanced C++

(from guest lecture; not assessable)

- Concepts
  - aka avoiding ->
- Modules
- Ranges
- Coroutines



```
:60,
5771/ass3/include/gdwg/graph.hpp:11,
5771/ass3/source/client.cpp:1:
ntiation of ‘std::_Rb_tree<_Key, _Val, _KeyOfValue, _Compare, _Alloc>::_Rb_tree_impl<_Key_compare, <anonymous> >::_Rb_tree_impl(const std::_Rb_tree<_Key, _Val, _KeyOfValue, _Compare, _Alloc>::_Rb_tree_impl<_Key_compare, <anonymous> >&) [with
std::shared_ptr<std::__cxx11::basic_string<char> >); bool <anonymous> = false; _Key = std::shared_ptr<std::__cxx11::basic_string<char> >; _Val = std::shared_ptr<std::__cxx11::basic_string<char> >; _KeyOfValue = std::_Identity<std::shared_ptr<
std::shared_ptr<std::__cxx11::basic_string<char> >); _Alloc = std::allocator<std::shared_ptr<std::__cxx11::basic_string<char> > >]’:
required from ‘std::_Rb_tree<_Key, _Val, _KeyOfValue, _Compare, _Alloc>::_Rb_tree(const std::_Rb_tree<_Key, _Val, _KeyOfValue, _Compare, _Alloc>&) [with _Key = std::shared_ptr<std::__cxx11::basic_string<char> >; _Val = std::shared_ptr<std::__
ring<char> > >; _Compare = bool(std::shared_ptr<std::__cxx11::basic_string<char> >, std::shared_ptr<std::__cxx11::basic_string<char> >); _Alloc = std::allocator<std::shared_ptr<std::__cxx11::basic_string<char> > >]’
equired from ‘gdwg::graph<N, E>::graph(gdwg::graph<N, E>&) [with N = std::__cxx11::basic_string<char>; E = int]’
t.cpp:25:52:   required from here
rror: no matching function for call to ‘std::_Rb_tree_key_compare<bool(std::shared_ptr<std::__cxx11::basic_string<char> >, std::shared_ptr<std::__cxx11::basic_string<char> >)>::_Rb_tree_key_compare(bool (* const&)(std::shared_ptr<std::__cxx11



te: candidate: ‘std::_Rb_tree_key_compare<_Key_compare>::_Rb_tree_key_compare(std::_Rb_tree_key_compare<_Key_compare>&&) [with _Key_compare = bool(std::shared_ptr<std::__cxx11::basic_string<char> >, std::shared_ptr<std::__cxx11::basic_string<
ey_compare&& __x)

ote:   no known conversion for argument 1 from ‘bool (* const)(std::shared_ptr<std::__cxx11::basic_string<char> >, std::shared_ptr<std::__cxx11::basic_string<char> >)’ to ‘std::_Rb_tree_key_compare<bool(std::shared_ptr<std::__cxx11::basic_str
&’
ey_compare&& __x)
~~~~~~~~~~~~~~~^~~
te: candidate: ‘constexpr std::_Rb_tree_key_compare<_Key_compare>::_Rb_tree_key_compare(const std::_Rb_tree_key_compare<_Key_compare>&) [with _Key_compare = bool(std::shared_ptr<std::__cxx11::basic_string<char> >, std::shared_ptr<std::__cxx11
tree_key_compare&) = default;

ote:   no known conversion for argument 1 from ‘bool (* const)(std::shared_ptr<std::__cxx11::basic_string<char> >, std::shared_ptr<std::__cxx11::basic_string<char> >)’ to ‘const std::_Rb_tree_key_compare<bool(std::shared_ptr<std::__cxx11::bas
’
tree_key_compare&) = default;
~~~~~~~~~~~~~~~~~~~~
te: candidate: ‘std::_Rb_tree_key_compare<_Key_compare>::_Rb_tree_key_compare(const _Key_compare&) [with _Key_compare = bool(std::shared_ptr<std::__cxx11::basic_string<char> >, std::shared_ptr<std::__cxx11::basic_string<char> >)]’
_compare& __comp)

ote:   no known conversion for argument 1 from ‘bool (* const)(std::shared_ptr<std::__cxx11::basic_string<char> >, std::shared_ptr<std::__cxx11::basic_string<char> >)’ to ‘bool (&)(std::shared_ptr<std::__cxx11::basic_string<char> >, std::shar
_compare& __comp)
~~~~~~~~~~~^~~~~~
te: candidate: ‘std::_Rb_tree_key_compare<_Key_compare>::_Rb_tree_key_compare() [with _Key_compare = bool(std::shared_ptr<std::__cxx11::basic_string<char> >, std::shared_ptr<std::__cxx11::basic_string<char> >)]’


te:   candidate expects 0 arguments, 1 provided
ntiation of ‘static const _Key& std::_Rb_tree<_Key, _Val, _KeyOfValue, _Compare, _Alloc>::_S_key(std::_Rb_tree<_Key, _Val, _KeyOfValue, _Compare, _Alloc>::_Const_Link_type) [with _Key = gdwg::graph<std::__cxx11::basic_string<char>, int>::edge
dge_type; _KeyOfValue = std::_Identity<gdwg::graph<std::__cxx11::basic_string<char>, int>::edge_type>; _Compare = gdwg::graph<std::__cxx11::basic_string<char>, int>::edge_type_comparator; _Alloc = std::allocator<gdwg::graph<std::__cxx11::basi
e = const std::_Rb_tree_node<gdwg::graph<std::__cxx11::basic_string<char>, int>::edge_type>*]’:
 required from ‘std::pair<std::_Rb_tree_node_base*, std::_Rb_tree_node_base*> std::_Rb_tree<_Key, _Val, _KeyOfValue, _Compare, _Alloc>::_M_get_insert_unique_pos(const key_type&) [with _Key = gdwg::graph<std::__cxx11::basic_string<char>, int>
dge_type; _KeyOfValue = std::_Identity<gdwg::graph<std::__cxx11::basic_string<char>, int>::edge_type>; _Compare = gdwg::graph<std::__cxx11::basic_string<char>, int>::edge_type_comparator; _Alloc = std::allocator<gdwg::graph<std::__cxx11::basi
::graph<std::__cxx11::basic_string<char>, int>::edge_type]’
 required from ‘std::pair<std::_Rb_tree_iterator<_Val>, bool> std::_Rb_tree<_Key, _Val, _KeyOfValue, _Compare, _Alloc>::_M_insert_unique(_Arg&&) [with _Arg = gdwg::graph<std::__cxx11::basic_string<char>, int>::edge_type; _Key = gdwg::graph<st
:edge_type; _KeyOfValue = std::_Identity<gdwg::graph<std::__cxx11::basic_string<char>, int>::edge_type>; _Compare = gdwg::graph<std::__cxx11::basic_string<char>, int>::edge_type_comparator; _Alloc = std::allocator<gdwg::graph<std::__cxx11::ba
required from ‘std::pair<typename std::_Rb_tree<_Key, _Key, std::_Identity<_Tp>, _Compare, typename __gnu_cxx::__alloc_traits<_Alloc>::rebind<_Key>::other>::const_iterator, bool> std::set<_Key, _Compare, _Alloc>::insert(std::set<_Key, _Compar
dge_type; _Compare = gdwg::graph<std::__cxx11::basic_string<char>, int>::edge_type_comparator; _Alloc = std::allocator<gdwg::graph<std::__cxx11::basic_string<char>, int>::edge_type>; typename std::_Rb_tree<_Key, _Key, std::_Identity<_Tp>, _Co
r>::const_iterator = std::_Rb_tree<gdwg::graph<std::__cxx11::basic_string<char>, int>::edge_type, gdwg::graph<std::__cxx11::basic_string<char>, int>::edge_type, std::_Identity<gdwg::graph<std::__cxx11::basic_string<char>, int>::edge_type>, gd
h<std::__cxx11::basic_string<char>, int>::edge_type> >::const_iterator; typename __gnu_cxx::__alloc_traits<_Alloc>::rebind<_Key>::other = std::allocator<gdwg::graph<std::__cxx11::basic_string<char>, int>::edge_type>; typename __gnu_cxx::__all
std::__cxx11::basic_string<char>, int>::edge_type>, gdwg::graph<std::__cxx11::basic_string<char>, int>::edge_type>::rebind<gdwg::graph<std::__cxx11::basic_string<char>, int>::edge_type>; typename _Alloc::value_type = gdwg::graph<std::__cxx11:
x11::basic_string<char>, int>::edge_type]’
/graph.hpp:329:24:   required from ‘bool gdwg::graph<N, E>::insert_edge(const N&, const N&, const E&) [with N = std::__cxx11::basic_string<char>; E = int]’
t.cpp:14:20:   required from here
rror: static assertion failed: comparison object must be invocable as const
ompare&, const _Key&, const _Key&>,
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
ote: ‘std::is_invocable_v<const gdwg::graph<std::__cxx11::basic_string<char>, int>::edge_type_comparator&, const gdwg::graph<std::__cxx11::basic_string<char>, int>::edge_type&, const gdwg::graph<std::__cxx11::basic_string<char>, int>::edge_ty
graph/CMakeFiles/graph_test1.dir/graph_test1.cpp.o
/graph_test1.cpp.o
ss3/include -I/home/migue/Source/COMP6771/ass3/test/graph -I/home/migue/Source/COMP6771/ass3/test/. -g -Wno-error -Wno-self-assign-overloaded -std=c++20 -MD -MT test/graph/CMakeFiles/graph_test1.dir/graph_test1.cpp.o -MF test/graph/CMakeFiles
/migue/Source/COMP6771/ass3/test/graph/graph_test1.cpp
:60,
5771/ass3/include/gdwg/graph.hpp:11,
5771/ass3/test/graph/graph_test1.cpp:1:
ntiation of ‘struct std::_Rb_tree_key_compare<bool(std::shared_ptr<int>, std::shared_ptr<int>)>’:
required from ‘struct std::_Rb_tree<std::shared_ptr<int>, std::shared_ptr<int>, std::_Identity<std::shared_ptr<int> >, bool(std::shared_ptr<int>, std::shared_ptr<int>), std::allocator<std::shared_ptr<int> > >::_Rb_tree_impl<bool(std::shared_p
 required from ‘class std::_Rb_tree<std::shared_ptr<int>, std::shared_ptr<int>, std::_Identity<std::shared_ptr<int> >, bool(std::shared_ptr<int>, std::shared_ptr<int>), std::allocator<std::shared_ptr<int> > >’
required from ‘class std::set<std::shared_ptr<int>, bool(std::shared_ptr<int>, std::shared_ptr<int>), std::allocator<std::shared_ptr<int> > >’
/graph.hpp:597:72:   required from ‘class gdwg::graph<int, std::__cxx11::basic_string<char> >’
raph_test1.cpp:7:41:   required from here
rror: data member ‘std::_Rb_tree_key_compare<bool(std::shared_ptr<int>, std::shared_ptr<int>)>::_M_key_compare’ invalidly declared function type
y_compare;
~~~~~~~~~
ntiation of ‘class std::_Rb_tree<std::shared_ptr<int>, std::shared_ptr<int>, std::_Identity<std::shared_ptr<int> >, bool(std::shared_ptr<int>, std::shared_ptr<int>), std::allocator<std::shared_ptr<int> > >’:
required from ‘class std::set<std::shared_ptr<int>, bool(std::shared_ptr<int>, std::shared_ptr<int>), std::allocator<std::shared_ptr<int> > >’
/graph.hpp:597:72:   required from ‘class gdwg::graph<int, std::__cxx11::basic_string<char> >’
raph_test1.cpp:7:41:   required from here
ror: function returning a function


:61,
5771/ass3/include/gdwg/graph.hpp:11,
5771/ass3/test/graph/graph_test1.cpp:1:
tiation of ‘class std::set<std::shared_ptr<int>, bool(std::shared_ptr<int>, std::shared_ptr<int>), std::allocator<std::shared_ptr<int> > >’:
/graph.hpp:597:72:   required from ‘class gdwg::graph<int, std::__cxx11::basic_string<char> >’
raph_test1.cpp:7:41:   required from here
or: function returning a function


or: function returning a function


on ‘-Wno-self-assign-overloaded’ may have been intended to silence earlier diagnostics
```

~~Week 11:~~ Goodbye*

https://www.youtube.com/watch?v=qROu_TyeolU&t=77s&ab_channel=BoyzIIMen-Topic

Although we've
come to the end
of the road
Still I can't let go

End Of The Road Lyrics

* Not yet (click right)

Final Exam

- See the Week 10 Notice for in-depth information
- Practical exam with two questions:
  - Q1 - STL, algorithms, dynamic polymorphism
  - Q2 - classes, templates, compile-time programming
- Q1 targets:
  - Students aiming for a PS or a CR
  - Easier than Q2
- Q2 targets:
  - Students aiming for a D or HD
  - Quite difficult but completable with everything taught in this course
- Partial marks available for Q1 and Q2
- Sample Exam released NOW!
  - No solutions will be released
  - Can ask questions about it on the forum

Goodbye 👋

- Further awesome C++ resources
- Books:
  - The Design & Evolution of C++ by Bjarne Stroustrup (creator of C++!)
  - Anything by Herb Sutter (ISO Chair for C++)
- Videos:
  - Cppcon (free conference talks, held annually)
  - C++ Weekly with Jason Turner
- I Tried This ONE Trick to INCREASE Exam Time and My Life Changed FOREVER...

Feedback