Templates

- Generalizes software components
- Such components can be reused in different situations
- Operate of any kind of data
- High performance algorithms & classes
- Compile time; no runtime costs are involved
- Libraries such as ATL, WTL, Boost, POCO, ACE, etc. use templates for implementation

Function Templates

- Function that accepts template type arguments
- Always begins with template keyword
- Template type argument is called type name
- Type name is a placeholder for the actual type
- Can accept any type
- The template type can be used as return type

Template parameter list

```
template<typename T>
T Function(T arg){
    //Implementation
}
```


Template Argument Deduction

- Process of deducing the types
- Each function argument is examined
- The corresponding type argument is deduced from the argument
- The type argument deduction should lead to same type
- Type conversions are not performed
- After deduction, the template is instantiated
- Override deduction by specifying types in template argument list

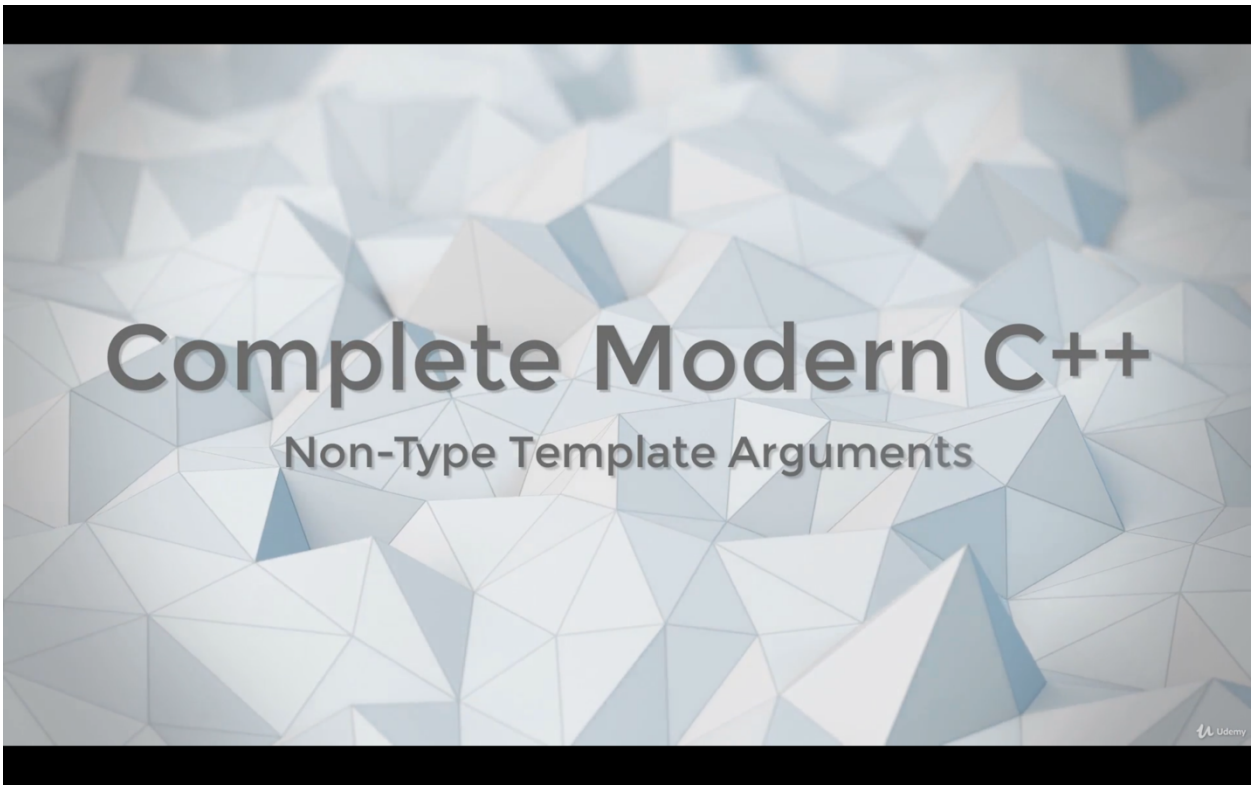
    Max<int>(3,5) ;

Template Instantiation

- A template function or class only acts as a blueprint
- The compiler generates code from the blueprint at compile time
- Known as template instantiation
- Occurs implicitly when
  - a function template is invoked
  - taking address of a function template
  - using explicit instantiation
  - creating explicit specialization
- Full definition of template should be available
- Define in header file

Explicit Specialization

- Template specialized for a particular type
- Provides correct semantics for some datatype
- Or implement an algorithm optimally for a specific type
- Explicitly specialized functions must be defined in a .cpp file
- Primary template definition should occur before specialization



Nontype Template Arguments

- Expression that is computed at compile time within a template argument list
- Must be a constant expression (addresses, references, integrals, nullptr, enums)
- Part of the template type
- Used by std::begin & std::end functions