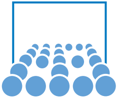
Technische Universität München

Template Metaprogramming and STL

Tutorial for Advanced Programming

Friedrich Menhorn

January 19, 2016



Contents

1. What is template metaprogramming?

2. How do we use templates?

3. Standard Template Library
   - 3.1 C++ std library
   - 3.2 Boost

4. Conclusion

What is template metaprogramming?

From Wikipedia:

- Metaprogramming is the writing of computer programs that can treat programs (itself or others) as their data
- Template metaprogramming (TMP) is a metaprogramming technique that uses templates to generate temporary code that will be merged and compiled at compile-time

Contents

1. What is template metaprogramming?

2. How do we use templates?

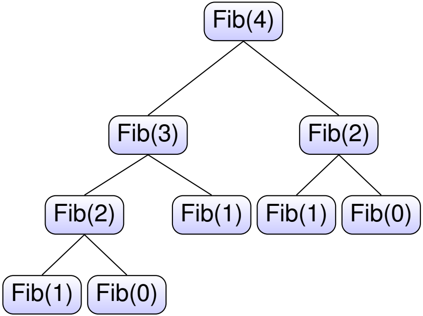
3. Standard Template Library
   - 3.1 C++ std library
   - 3.2 Boost

4. Conclusion

How do we use templates?

- Fibonacci with classical recursion:

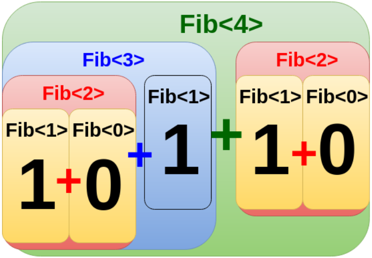
```
size_t Fib(size_t n){
if(n==1) return 1;
if(n==0) return 0;

return Fib(n-1)+Fib(n-2);
}

int main(){
    size_t result = Fib(4);
}
```



How do we use templates?

- Fibonacci with template metaprogramming:

```
template<int n> class Fib{
public:
      static const size_t value =
      Fib<n-1>::value + Fib<n-2>::value;
};
template<> class Fib<1>{
     public:
          static const size_t value = 1;
};
template<> class Fib<0>{
     public:
          static const size_t value = 0;
};
int main(){
     size_t result = Fib<4>::value;
}
```



Template tricks

- Variadic template

```
template<typename... Arguments> class tuple;
tuple<> tupleInstance1;
tuple<int> tupleInstance2;
tuple<int, std::map<bool, std::vector<float>>> tupleInstance3;
template<typename... Params>
void printf(const std::string &str_format, Params... parameters);
```

- Curiously Recurring Template Pattern:

```
    template<class T>
    class Base{
    // methods within Base can use template to access members of Derived
        void interface(){
            static_cast<T*>(this)->implementation();
        }
    };
    class Derived : public Base<Derived>{
        void implementation();
    };
```

Contents

1. What is template metaprogramming?

2. How do we use templates?

3. Standard Template Library
   - 3.1 C++ std library
   - 3.2 Boost

4. Conclusion

STL

- The Standard Template Library is a software library for the C++ programming language
- Influenced many parts of the C++ standard library
- Provides four components:
  - Algorithms
  - Containers
  - Functional
  - Iterators

Containers and Iterators

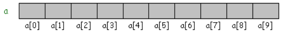
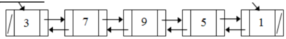
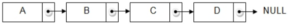
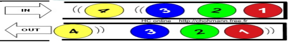
- Iterator:

  Pointer-like object (object, which supports pointer operations) that is able to point to a specific element in a container

- Container:

  Object that represents a group of elements of a certain type, stored in a way that depends on the type of the container

- Most used STL container:

vector	Array	 a a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9]
list	Doubly-linked list	
slist	Singly-linked list	
queue	FIFO structure	
stack	LIFO structure	 IN 4 3 2 1 HC online http://chohmann.free.fr OUT 4 3 2 1
pair	2-tuple	
set	Set of unique elements	

C++ stdlib: Usage examples

- Getting the maximum value of an array:

```cpp
vector<int> values(3,5); //Three ints with value 5
values.push_back(2); // values: 5 5 5 2
values[1] = 1; // values: 5 1 5 2
vector<int>::const_iterator current = values.begin();
vector<int>::const_iterator pos, end = values.end();
int max = *current; current++;
while(current!=values.end()){
    if(*current>max){
        max = *current;
    }
    current++;
}
```

- Finding an element satisfying a constraint:

```cpp
class EqualPred{
 int match;
 EqualPred(int n): match(n){};
 bool operator()(int x){return x==match};
}
...const_iterator pos = std::find_if(start, end, EqualPred(3));
```

Boost C++ Libraries (`boost.org`)

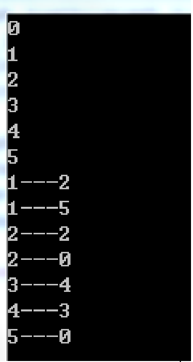
- ”Boost is the most powerful and complicated 3rd part library. However, Boost is so heavy that people and companies may refuse to use it.” (jdxyw.com)
- About 50 major sub-components based on STL:
  - ”Better” smart pointers
  - Maths and Matrices
  - Threads
  - Boost Graph Library
  - many many more... (`http://www.codeproject.com/Articles/4496/An-Introduction-to-Boost`)

Boost Graph Example

```
typedef boost::adjacency_list<boost::vecS, boost::vecS, boost::directedS>
    graph_t;
graph_t g(6);
boost::add_edge(1,2,g);
boost::add_edge(1,5,g);
boost::add_edge(2,2,g);
boost::add_edge(2,0,g);
boost::add_edge(3,4,g);
boost::add_edge(4,3,g);
boost::add_edge(5,0,g);

typedef graph_traits<graph_t>::vertex_iterator
    vertex_iter;
pair<vertex_iter, vertex_iter> vrange=vertices(g);
for(vertex_iter it =vrange.first; it!=vrange.second; ++it)
cout << *it << endl;

typedef graph_traits<graph_t>::edge_iterator
    edge_iter;
pair<edge_iter, edge_iter> erange=edges(g);
for(edge_iter it =erange.first; it!=erange.second; ++it)
cout << source(*it,g) <<"——"<<target(*it,g) << endl;
```

Boost Graph Example

```
typedef boost::adjacency_list<boost::vecS, boost::vecS, boost::directedS>
    graph_t;
graph_t g(6);
boost::add_edge(1,2,g);
boost::add_edge(1,5,g);
boost::add_edge(2,2,g);
boost::add_edge(2,0,g);
boost::add_edge(3,4,g);
boost::add_edge(4,3,g);
boost::add_edge(5,0,g);

typedef graph_traits<graph_t>::vertex_iterator
    vertex_iter;
pair<vertex_iter, vertex_iter> vrange=vertices(g);
for(vertex_iter it =vrange.first; it!=vrange.second; ++it)
cout << *it << endl;

typedef graph_traits<graph_t>::edge_iterator
    edge_iter;
pair<edge_iter, edge_iter> erange=edges(g);
for(edge_iter it =erange.first; it!=erange.second; ++it)
cout << source(*it,g) <<"---"<<target(*it,g) << endl;
```



```
0
1
2
3
4
5
1---2
1---5
2---2
2---0
3---4
4---3
5---0
```

Contents

1. What is template metaprogramming?

2. How do we use templates?

3. Standard Template Library

   3.1 C++ std library

   3.2 Boost

4. Conclusion

Conclusion

- Template metaprogramming is a powerful C++ tool to generate code at compile-time
- STL is a very useful collection of generic classes and functions (great example of TMP)
- C++ Standard Library $\approx$ STL
- Very powerful Boost libraries ($\Rightarrow$ CSE)
- References:
  - `https://monoinfinito.wordpress.com/series/introduction-to-c-template-metaprogramming/`
  - Scott Meyers, Effective C++
  - Davide di Gennaro, Advanced C++ Template Metaprogramming
  - Dave Abrahams, Aleksey Gurtovoy, C++ Template Metaprogramming