A C++ Standard Template Library használatának validációs módszerei

Norbert Pataki

Programozási Nyelvek és Fordítóprogramok Tanszék

Bolyai Kollégium, 2021

Miről lesz szó?

A C++ Standard Template Library(STL) komponensei

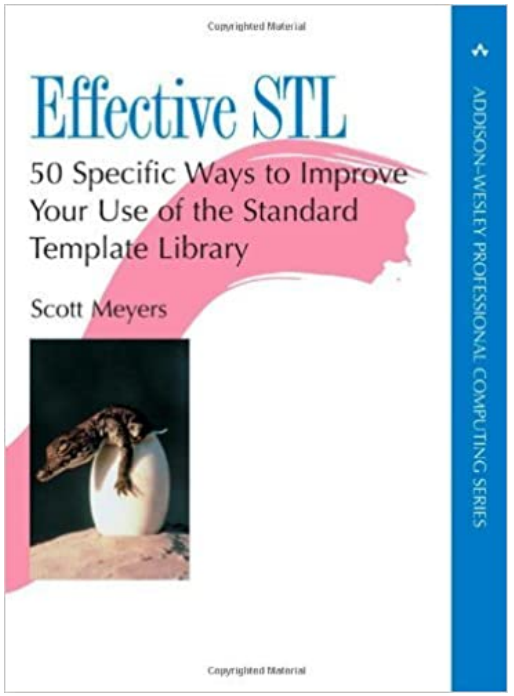
- Konténerek, pl. `std::vector`, `std::map`
- Algoritmusok, pl. `std::find_if`, `std::sort`
- Iterátorok
- Funktorok (és lambdák)
- Adaptorok
- Allokátorok

Tulajdonságok

- Generikus programozás, OOP limitációk feloldása
- Implementáció vs. Szabvány
- Aszimptotikus műveletigény garanciák
- Hagyományos hibák elkerülése
- Hatékonyság
- C++ sablonok lehetőségeinek kiaknakázása

Új típusú hibalehetőségek

- Futási idejű hibák
- Nemdefiniált viselkedés
- Bugok, furcsa viselkedés
- Hatékonyság-veszteség
- Hordozhatósági problémák
- Inkonzisztens konténerek
- Stb.



Dr. Gődény György nehézségi szint

Dr. Gődény György nehézségi szint

```
std::list<int> c;
// ...
std::sort( c.begin(), c.end() );
```

Dr. Gődény György nehézségi szint

```
In file included from /usr/include/c++/5/algorithm:62:0,
                 from listsort.cpp:2:
/usr/include/c++/5/bits/stl_algo.h: In instantiation of 'void std::__sort(_RandomAccessIterator, _RandomAccessIterator, _Compare) [with _RandomAccessIterator = std::_Li
st_iterator<int>; _Compare = __gnu_cxx::__ops::_Iter_less_iter]':
/usr/include/c++/5/bits/stl_algo.h:4698:18:   required from 'void std::sort(_RAIter, _RAIter) [with _RAIter = std::_List_iterator<int>]'
listsort.cpp:7:33:   required from here
/usr/include/c++/5/bits/stl_algo.h:1964:22: error: no match for 'operator-' (operand types are 'std::_List_iterator<int>' and 'std::_List_iterator<int>')
     std::__lg(__last - __first) * 2,
                ~~~~~~^~~~~~~~
In file included from /usr/include/c++/5/bits/stl_algobase.h:67:0,
                 from /usr/include/c++/5/list:60,
                 from listsort.cpp:1:
/usr/include/c++/5/bits/stl_iterator.h:328:5: note: candidate: template<class _Iterator> typename std::reverse_iterator<_Iterator>::difference_type std::operator-(const
 std::reverse_iterator<_Iterator>&, const std::reverse_iterator<_Iterator>&)
     operator-(const reverse_iterator<_Iterator>& __x,
     ^
/usr/include/c++/5/bits/stl_iterator.h:328:5: note:   template argument deduction/substitution failed:
In file included from /usr/include/c++/5/algorithm:62:0,
                 from listsort.cpp:2:
/usr/include/c++/5/bits/stl_algo.h:1964:22: note:   'std::_List_iterator<int>' is not derived from 'const std::reverse_iterator<_Iterator>'
     std::__lg(__last - __first) * 2,
                ~~~~~~^~~~~~~~
In file included from /usr/include/c++/5/bits/stl_algobase.h:67:0,
                 from /usr/include/c++/5/list:60,
                 from listsort.cpp:1:
/usr/include/c++/5/bits/stl_iterator.h:385:5: note: candidate: template<class _IteratorL, class _IteratorR> typename std::reverse_iterator<_Iterator>::difference_type s
td::operator-(const std::reverse_iterator<_Iterator>&, const std::reverse_iterator<_IteratorR>&)
     operator-(const reverse_iterator<_IteratorL>& __x,
     ^
/usr/include/c++/5/bits/stl_iterator.h:385:5: note:   template argument deduction/substitution failed:
In file included from /usr/include/c++/5/algorithm:62:0,
                 from listsort.cpp:2:
/usr/include/c++/5/bits/stl_algo.h:1964:22: note:   'std::_List_iterator<int>' is not derived from 'const std::reverse_iterator<_Iterator>'
     std::__lg(__last - __first) * 2,
                ~~~~~~^~~~~~~~
```

Dr. Gődény György nehézségi szint

```
std::list<int> c;
// ...
c.sort();
```

Gál Kristóf nehézségi szint

Gál Kristóf nehézségi szint

```
const int max = 1000;
std::vector<int> v;
for( int i = 0; i < max; ++i )
{
 v.push_back( i );
}
```

Gál Kristóf nehézségi szint

```
const int max = 1000;
std::vector<int> v;
v.reserve( max );
for( int i = 0; i < max; ++i )
{
 v.push_back( i );
}
```

Győrfi Pál nehézségi szint

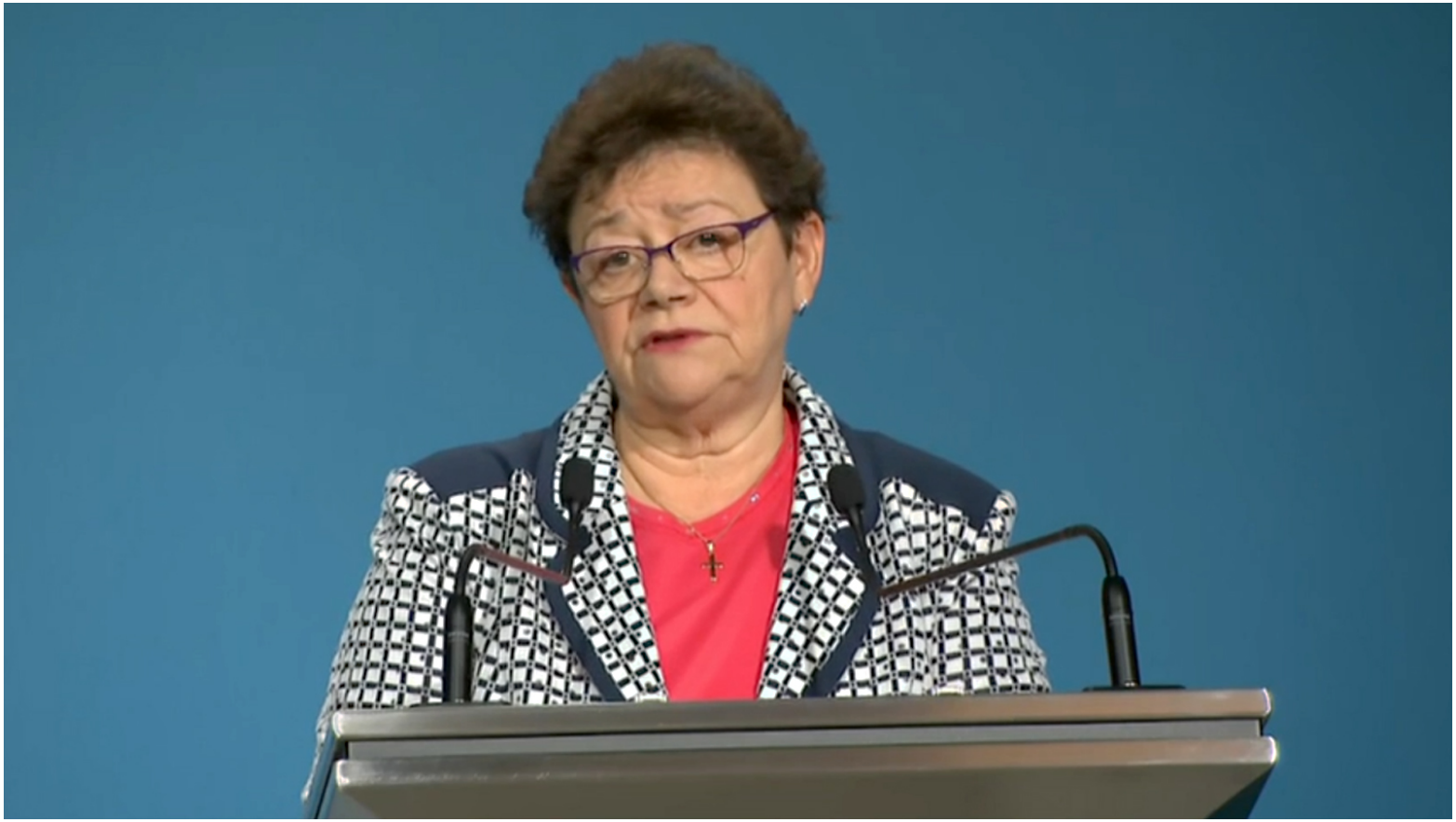
Győrfi Pál nehézségi szint

```
#include <vector>

void f()
{
 std::vector<int> v;
 // ...
 std::reverse( v.begin(), v.end() );
}
```

Győrfi Pál nehézségi szint

```
#include <vector>
#include <algorithm>

void f()
{
 std::vector<int> v;
 // ...
 std::reverse( v.begin(), v.end() );
}
```

Müller Cecília nehézségi szint



Müller Cecília nehézségi szint

```
struct Comp
{
 bool operator()( int lhs, int rhs ) const
 {
   return !( lhs < rhs );
 }
};
```

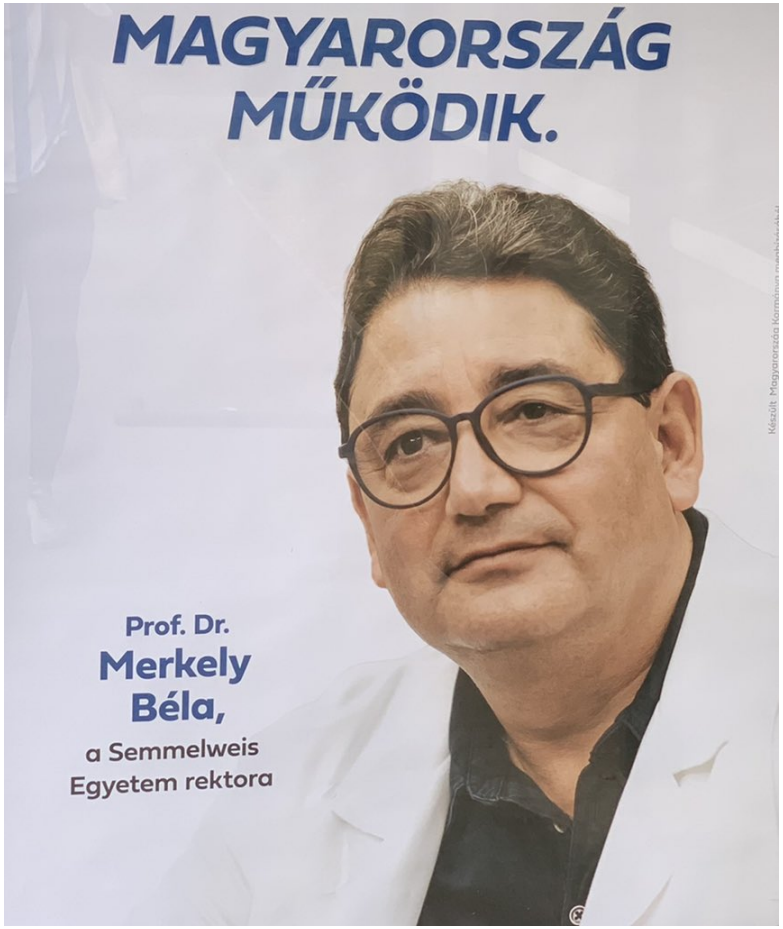
Müller Cecília nehézségi szint

```
std::set<int, Comp> s;
s.insert( 7 );
s.insert( 7 );
std::cout « s.size();

std::multiset<int, Comp> m;
m.insert( 7 );
m.insert( 7 );
std::cout « m.count( 7 );
```

Müller Cecília nehézségi szint

```
std::set<int, Comp> s;
s.insert( 7 );
s.insert( 7 );
std::cout « s.size(); // 2

std::multiset<int, Comp> m;
m.insert( 7 );
m.insert( 7 );
std::cout « m.count( 7 ); // 0
```

Merkely Béla nehézségi szint



Merkely Béla nehézségi szint

```
class Pred
{
 int cnt;
public:
 Pred():  cnt( 0 ) { }
 bool operator()( int )
 {
   return 3 == ++cnt;
 }
};
```

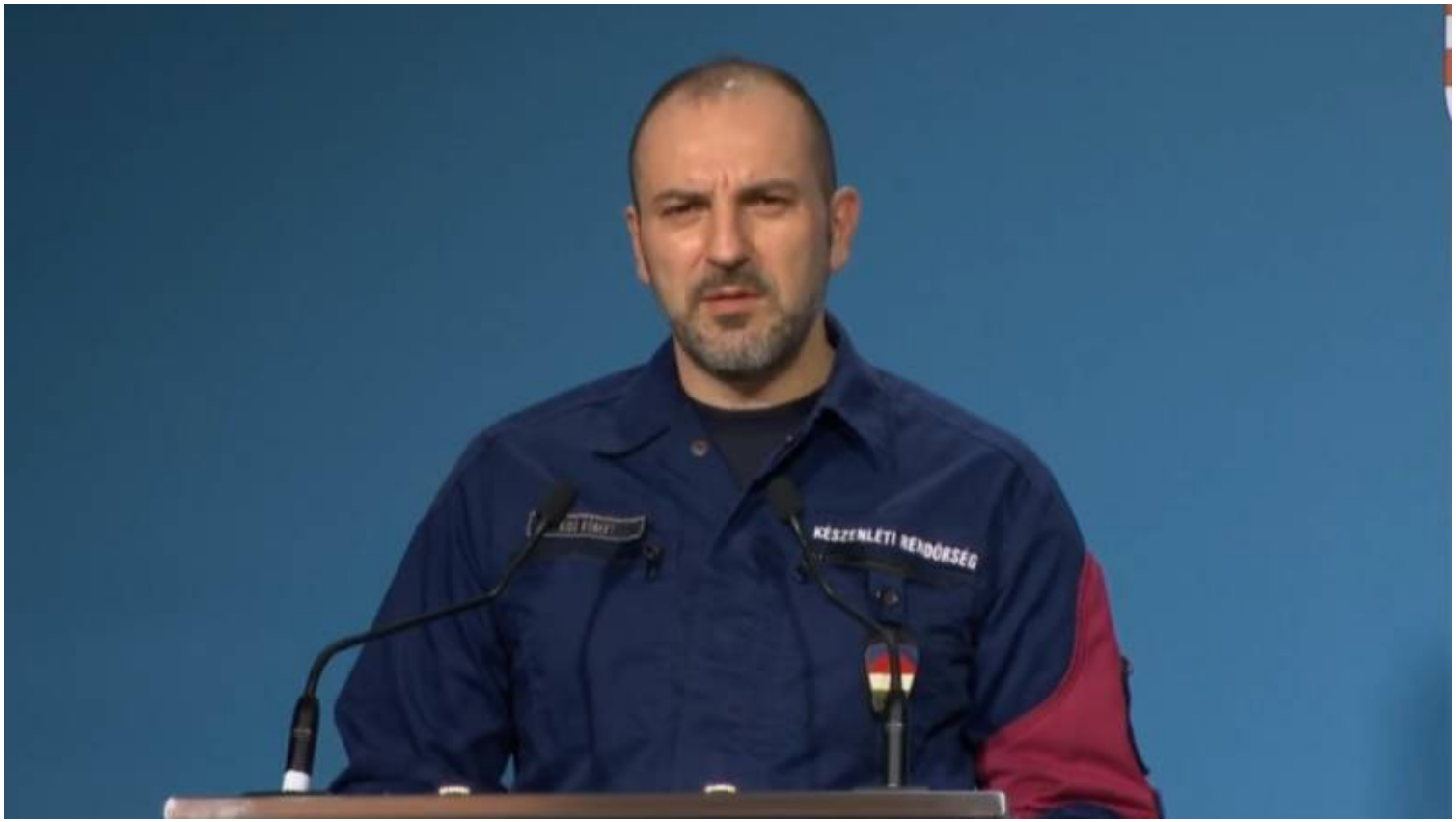
Merkely Béla nehézségi szint

```
std::list<int> c;
//...
c.erase( std::remove_if( c.begin(),
                         c.end(),
                         Pred() ),
         c.end() );
```

Ez nem biztos, hogy csak a harmadik elemet törli.

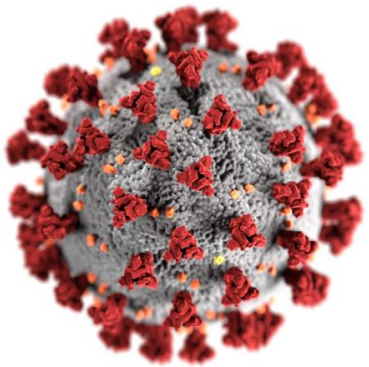
- Mitől függ?
- Mi a probléma általánosan?
- Adattagok, globális változók, lokális statikusok, stb.

Mit tehetünk?



Módszerek

- Fordítási idejű megoldások
  - C++ template metaprogramozás
  - Statikus elemzés
- Futási idejű megoldások
  - Debugger-alapú validáció (gdb)
  - Aspektus-orientált validáció



Template metaprogramozás

- Fordítás közbeni template példányosítások
- Turing-teljes résznyelv
- Fordítási hibák/figyelmeztetések generálása
- Template metaprogramok beleírása az STL forrásába
- Például: copy algoritmusok példányosításakor annak kiértékelése, hogy inserter iterátorral dolgoznak-e

Clang példa

```
memberCallExpr(
  on( hasType( container ) ),
  callee( methodDecl( hasName( "size" ) ) ),
  unless( anyOf(
      hasAncestor( binaryOperator(
        unless( has( integerLiteral(
          equals( 0 ) ) ) ),
        unless( anyOf(
          hasOperatorName( "&&" ),
          hasOperatorName( "||" ) ) ) ) ),
    hasAncestor( varDecl() ) ) ) ).bind( "id" );
```

Debugger-based approach

- Átfogó keretrendszer
- A *gdb* debugger meghajtása
- Breakpointokat helyezünk el az STL forrásában
- gdb scriptek futtatása, állapotok karbantartása
- Végrehajtás folytatása
- Kiértékelés, logolás

Aspect-orientált példa

```
pointcut usageUnique() =
  "% std::unique<...>(...)";

advice call(usageUnique()) : after()
{
  if ( !isUnique( *tjp->arg<0>(),
                  *tjp->result() ) )
  {
    // report the problem
  }
}
```

Kiértékelés

- Előnyök
- Hátrányok

Összefoglalás

- C++ STL – kényelmes, hatékony
- Klasszikus hibák elkerülése
- Új jellegű hibák merülnek fel
- Megoldási módok
  - C++ template metaprogramozás
  - Statikus elemzés
  - Debugger-alapú validáció
  - Aspektus-orientált validáció

Köszönöm a figyelmet!