Lazy Evaluation

Source: Scott Meyers, More Effective C++

Christoph Kolb

Lazy Evaluation

- As opposed to „eager evaluation“
- Computations are only performed when the results are needed
- … possibly not at all!
- Uses:
  - Database queries
  - Large matrix operations
  - Boost Lambda Library

Lazy Evaluation Example 1

```
Class String { … };  // String class with lazy copy constructor

String s1 = „Hello“;
String s2 = s1;      // s2 gets the address of s1 in the copy constructor

Cout << s1;
Cout << s1 + s2;     // no problems! Only read operations


s2.convertToUpperCase(); // now, the promised copy operation has to be performed
                         // s2‘s value is changed but not s1
```

Source: Scott Meyers, More Effective c++, Item 17

Lazy Evaluation Example 2

```
template<class T>
class Matrix { ... };

Matrix<int> m1(1000x1000);
Matrix<int> m2(1000x1000);
…
Matrix<int> m3 = m1 + m2;          //save expression m1+m2 in m3 but don‘t
                                   compute it yet!


m3 = m1 – m2;                      //reassign m3… previous computations would
                                   have been useless.


cout << m3[0];                     //only print the first row of m3 and thus
                                   only compute those values
                                   (1000 operations instead of 1000000)
```

Source: Scott Meyers, More Effective c++, Item 17

Lazy Evaluation Example 3

```
mitk::ScalarType mitk::ImageStatisticsHolder::GetScalarValueMin(int t, unsigned
int component)
{
  ComputeImageStatistics(t, component);
  return m_ScalarMin[t];
}

mitk::ScalarType mitk::ImageStatisticsHolder::GetScalarValueMax(int t, unsigned
int component)
{
  ComputeImageStatistics(t, component);
  return m_ScalarMax[t];
}

…
```

Boost Lambda Library

- Functional Programming in C++
- Define lambda expressions with n parameters:

```
int i = 1; int j = 2;
(_1 + _2)(i, j);
```

- Lambda expressions can be used in stl functions:

```
#include <boost/lambda/lambda.hpp>
int main()
{
  std::vector<int> v{1, 3, 2};
  std::for_each(v.begin(), v.end(),
    std::cout << boost::lambda::_1 << "\n");
}
```

Source: http://www.boost.org/doc/libs/1_59_0/doc/html/lambda.html

Thank you for your attention!

Further information on www.dkfz.de

dkfz.
GERMAN CANCER RESEARCH CENTER
IN THE HELMHOLTZ ASSOCIATION

50 Years – Research for
A Life Without Cancer