BERGISCHE UNIVERSITÄT
WUPPERTAL
GAUSS-STRASSE 20
42119 WUPPERTAL
TELEFAX (0202) 439-2901
TELEFON (0202) 439-0
WWW www.uni-wuppertal.de

*Fachbereich C*

MATHEMATIK UND NATURWISSENSCHAFTEN
Fachgruppe Mathematik und Informatik

Prof. Dr. Hans-Jürgen Buhl
*Praktische Informatik / Numerik*

E-Mail: Hans-Juergen.Buhl@math.uni-wuppertal.de
WWW: http://www.math.uni-wuppertal.de/∼buhl

# Generische Programmierung (Spezielle Kapitel der praktischen Informatik)

WS 2012/2013 – Übungsblatt 8

13. Dezember 2011
Abgabe: bis 20. Dezember 2012 an
*Farzin.Ghorban@studs.math.uni-wuppertal.de*

**Aufgabe 1.** *Draft Proposal: Dynamic Libraries in C++*

Wie sollten gemäß

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2003/n1496.html

shared Objekte (so's) in C++ systemunabhängig standardisiert werden?

Warum wird dieser Erweiterungsvorschlag nach

http://herbsutter.wordpress.com/2007/02/07/iso-c0x-complete-public-review-draft-in-october-2007/

nicht in C++0x realisiert werden?

**Aufgabe 2.** *Draft Proposal: Modules in C++*

Welche Vorteile wären mit Modulen in C++ verbunden:

http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/n2073.pdf

Wie soll die Syntax aussehen? Welche Vorteile brächten Module gegenüber der momentanen Praxis (welche ist das?)?

Aus welchen Gründen wurden Module noch nicht in C++11 aufgenommen (vgl.

http://herbsutter.wordpress.com/2007/02/07/iso-c0x-complete-public-review-draft-in-october-2007/)?

**Aufgabe 3.** *Compilezeit Fakultät*

Die Template-Funktion `fact` „berechnet" zur Compilezeit die Fakultät:

```
// factorial as an integral_constant
#include <iostream>
#include <type_traits>
```

```
template <unsigned n>
struct fact : std::integral_constant<int,n * fact<n-1>::value> {};

template <>
struct fact<0> : std::integral_constant<int,1> {};
```

(Früher wurde dazu etwa

```
template<int i>
class fact {
  public:
    static const long long result = i * fact<i-1>::result;
};

template<> class fact<1> {
  public:
    static const long long result = 1;
};
```

benutzt.)

Ergänzen Sie diesen Quellcode um ein Test-Hauptprogramm. Wie wird `fact` aufgerufen? Wozu kann es benutzt werden, da es eine Compilezeit-Konstante produziert? Wie Überprüft man diese Compilezeit-Evaluierung?

Modifizieren Sie das Codestück zu einer Klasse mit enum-result beziehungsweise zu einer normalen template-Funktion ohne `enum` oder `static const`. Welche Unterschiede stellen Sie fest?

**Aufgabe 4.** *Compilezeit Ganzzahlpotenz*

Schreiben Sie eine ähnliche Template-Klasse zur Berechnung von $n^m$ für ganzzahlige `n` und `m`. Beschreiben Sie eine Nutzanwendung für dieses Template, die die Compilezeit-Evaluation deutlich in den Vordergrund stellt.

Wo setzt die C++-Einschränkung an Template nontype-Parameter der Einsatzfähigkeit dieser Technik Grenzen?