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 2011/2012 – Übungsblatt 10

19. Dezember 2011
Abgabe: bis 9. Januar 2012 12 Uhr an
sbieleck@studs.math.uni-wuppertal.de

Aufgabe 1. *Metaprogrammierung*

Lesen Sie

http://divyekapoor.blogspot.com/2008/07/walking-through-your-first-template.html

und testen Sie das Fibonacci-Beispiel.

Welche Einsatzgebiete sieht der Autor für Metaprogrammierung?

Welche Einsatzgebiete sehen Sie?

Aufgabe 2. *CONCEPT_CHECK_REQUIRES()*

Ergänzen Sie die generischen Funktionen `my_min(-)`, `arith_average(-,-)` und `geomMittel2(-,-)` der letzten Übungsblätter um die Überprüfung geeignet gewählter Konzepte.

Provozieren Sie Konzeptverletzungen bei der Template-Instantiierung: Welche Fehlermeldungen werden erzeugt?

Aufgabe 3. *EqualTypes* Testen Sie mit Hilfe des Templates

```
template< typename T1, typename T2 >
struct EqualTypes {
enum { result = false };
};
template< typename T >
struct EqualTypes<T,T> {
enum { result = true };
};
```

in wieweit durch typedefs erklärte Typnamen von C++ als identisch zu ihren Ursprungstypen aufgefasst werden (bei selbstdefinierten Klassen, enum's, ...).

Aufgabe 4. *Inheritance check*

Welche Ausgabe produziert das Programm:

```
template <class Derived, class Base>
class Check
{
class Nope {};
class Yep {char Dummy[3];};
static Yep Test(Base*);
static Nope Test(...);
public:
enum {
IsDerived = sizeof(Test(static_cast<Derived*>(0)))
== sizeof(Yep)
};
};

class X {};
class Y : public X {};

int main()
{
cout << Check<Y, X>::IsDerived << endl;
cout << Check<int, string>::IsDerived << endl;
return 0;
}
```

Warum?

Schreiben Sie eine kurze Anwendungsdokumentation für die Metafunktion `Check<.,.>::IsDerived`