



Generische Programmierung (Spezielle Kapitel der praktischen Informatik)

WS 2012/2013 – Übungsblatt 7

6. Dezember 2012

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

Aufgabe 1. *promote_trait*

Benutzen Sie das Template `promote_trait` in einer Templatefunktion `my_min` und testen Sie es für mindestens 10 unterschiedliche Typpaare.

Schreiben Sie ein ähnliches Template `arithAverage_trait` zur Nutzung in einer Templatefunktion `arithAverage(T x, T y)` zweier numerischer skalarer Parameter. Der Ergebnistyp für `T = int` soll dabei jedoch `double` sein (warum?). Welche Konzepte sollte `T` modellieren?

Aufgabe 2. *non-type template-Parameter*

Warum funktioniert:

```
#include <iostream>
using namespace std;

template <bool k> void print ()
{
    if(k==true) // oder auch: if(k)
        cout << "true" << endl;
    else
        cout << "false" << endl;
    return;
}

int main()
{
    print <7>();
    print <0>();
}
```

```

    return 0;
}

```

Wie viele und welche Inkarnationen der Templatefunktion `print()` werden automatisch erzeugt?

Welche Regeln gelten für die Typ-Transformation von non-type Funktionstemplate-Parametern über diejenigen der type Funktionstemplate-Parameter hinaus?

Ergänzen Sie in `main()` Aufrufe von

```
print<8>(); print<9>(); ...
```

Was erwarten Sie? Überprüfen Sie Ihre Erwartungen durch Benutzung von `nm`.

Aufgabe 3. *Template-Funktion mit array-Parametern*

Vergleichen Sie

```

#include <iostream>
#include <numeric>

template <typename Type>
Type sum(Type *tp, size_t n)
{
    return std::accumulate(tp, tp+n, Type());
}

int main()
{
    int x[10];
    for(int l = 0; l < 10; l++)
        x[l] = l;

    std::cout << sum(x, 10) << std::endl;

    return(0);
}

```

mit

```

#include <iostream>
#include <numeric>

template <typename Type, size_t n>
Type sum(const Type (&tp)[n])
{
    return std::accumulate(tp, tp+n, Type());
}

```

```

int main()
{
    int x[10];
    for(int l = 0; l < 10; l++)
        x[l] = l;

    std::cout << sum(x) << std::endl;

    return(0);
}

```

Welche Vor- und Nachteile bietet die erste Variante gegenüber der zweiten? Schreiben Sie eine Template-Funktion `unsigned int get_size(.)`, die die Anzahl der Komponenten eines übergebenen Arrays als Funktionsergebnis liefert.

Aufgabe 4. *InputIterator*

Beschreiben Sie umgangssprachlich in eigenen Worten die einzelnen Requirements des Konzepts `InputIterator` in **Iterator-Concepts**.

Welche Konzepte müssen die dem Iterator assoziierten Typen modellieren? Skizzieren Sie die Verfeinerungshierarchie aller Iteratoren.