Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim Geisenheim

Fachbereich Design Informatik Medien
Studiengang Bachelor Allgemeine Informatik

Programmieren 3 AI – Praktikum

Übungsblatt 7

1 Punkt

Sebastian Flothow

2011-11-28

Abnahme: 2011-12-05
Zur Abnahme ist die persönliche Anwesenheit erforderlich. Bearbeiten Sie die Aufgaben so, dass Sie nicht nur funktionierenden Code vorzuweisen haben, sondern diesen auch erläutern und Fragen dazu beantworten können. Zusätzlich sind die Lösungen am Tag der Abnahme per Email an `sebastian@flothow.de` einzusenden; dies dient u.a. der Prüfung auf eventuelle Plagiate.

Es muss für jede Aufgabe eine getrennte Lösung in einem separaten Verzeichnis vorliegen. Statten Sie jedes Abgabeverzeichnis mit einem Makefile aus. Dieses muss ein Default-Target haben, dass Ihr Programm mit `g++ -Wall -Wextra -pedantic` kompiliert und linkt, sowie das Target „`clean`“, dass die erzeugten Dateien wieder entfernt. Rufen Sie vor dem Packen `make clean` auf (d.h. die eingesandten Archive sollen nur Quelltexte enthalten, keine Kompilate).

Senden Sie Ihre Abgabe als tar.gz-Archiv(e). Diese müssen so gepackt sein, dass jedes Archiv genau ein Verzeichnis enthält, dessen Name (bis auf das Suffix) mit dem des Archivs übereinstimmt; ein solches Archiv kann durch einen Aufruf der Form `tar -czf` *`verzeichnis`*`.tar.gz` *`verzeichnis`* erzeugt werden.

Beim Kompilieren der Programme mit `g++ -Wall -Wextra -pedantic` dürfen keine Fehler oder Warnungen auftreten. Stellen Sie sicher dass Ihr Programm keine Speicherlecks aufweist und nicht auf nichtallozierten oder uninitialisierten Speicher zugreift; dies ist bei der Abnahme mit Valgrind unter Beweis zu stellen.

Aufgabe 1

Übernehmen Sie die Klasse `TreeNode` von Blatt 5 Aufgabe 1.

(a) Die Klasse soll nicht mehr auf die Speicherung von Ganzzahlen festgelegt sein, sondern beliebige Datentypen speichern können. Implementieren Sie die Klasse jetzt also als Template mit variablem Typ.

Probieren Sie dann aus, mit welchen der folgenden Datentypen der Baum verwendet werden kann:

- `int`
- `double`
- `std::complex<double>`
- `std::string`
- `Rational` (von Blatt 6 Aufgabe 2)

(b) Ergänzen Sie `TreeNode` um einen Kopierkonstruktor und überladen Sie den Zuweisungsoperator (`=`). Beide sollen dafür sorgen, dass der zu kopierende bzw. zuzuweisende Baum vollständig kopiert wird (*deep copy*).

Testen Sie Ihre Implementation mit dem Testprogramm `treetest.cpp` (Listing 1, auch als Datei von der Veranstaltungsseite herunterladbar). Lassen Sie dazu das Programm im DDD laufen, halten Sie es umittelbar vor der `return`-Anweisung an, und lassen Sie sich die Baumstrukturen grafisch anzeigen: Es müssen drei Bäume entstanden sein, die die gleichen Werte in der gleichen Anordnung enthalten, jedoch keine Knoten miteinander teilen.

Listing 1: treetest.cpp

```
#include "treenode.hpp"

int main() {
    TreeNode<short> a(7);
    a.insert(new TreeNode<short>(666));
    a.insert(new TreeNode<short>(-5));

    TreeNode<short> b(42);
    b.insert(new TreeNode<short>(23));
    b.insert(new TreeNode<short>(17));
    b.insert(new TreeNode<short>(105));
    b.insert(new TreeNode<short>(69));

    a = b;

    TreeNode<short> c(a);

    // an diesem Punkt müssen a, b, c unabhängige Bäume
       gleichen Inhalts sein

    return 0;
}
```