Programmieren für Physiker

Interfakultatives Institut für Anwendungen der Informatik
Institut für Theoretische Teilchenphysik

Prof. Dr. M. Steinhauser, Dr. A. Mildenberger
http://comp.physik.kit.edu

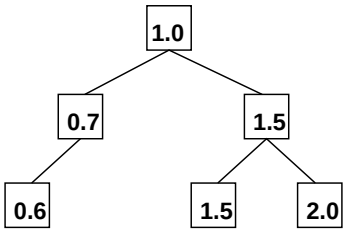
SS 2017 – Blatt 12 (letztes Blatt)
Bearbeitungszeitraum: bis 19. Juli 2017

Klausurtermin: Dienstag, 25. Juli 2017, 17.30 Uhr.
Es ist erforderlich, sich bis zum 18.07.2017 im Studierendenportal zur Klausur anzumelden.
Erster Buchstabe Ihres Nachnamens: A-K $\widehat{=}$ Gottlieb-Daimler-HS (HMO) (10.21),
L-Z $\widehat{=}$ Carl-Benz-HS (HMU) (10.21).
Dauer: 90 Minuten. Es sind keine Hilfsmittel erlaubt. Bitte Studierendenausweis mitbringen.

Aufgabe 31: Binärer Baum — Pflichtaufgabe

Sogenannte binäre Suchbäume sind Datenstrukturen, in denen Daten schnell sortiert hinzugefügt und gesucht werden können. Ein Knoten des Baums hat jeweils einen linken und einen rechten Nachfolger und einen Wert, hier eine `double`-Zahl. Die Bäume, die wir betrachten wollen, sollen stets sortiert sein: Alle Knoten im linken Ast haben einen Wert kleiner gleich dem aktuellen Knotenwert, alle Knoten rechts haben einen größeren Wert.
Folgende Klassenstruktur soll zum Einsatz kommen:

```
class Node {
private:
    double value ;  // Wert des Knoten
    Node* left ;    // linker Ast
    Node* right ;   // rechter Ast
public:
    // zu ergaenzen...
} ;
```



Beispiel eines sortierten Baums.

Schreiben Sie folgende Methoden und Funktionen:

- Einen Konstruktor mit einem `double`-Wert als Argument. Der Wert soll im aktuellen Knoten abspeichert und die beiden Zeiger des Knotens sollen auf `0` (=NULL) gesetzt werden. (Konvention: Ein Knotenzeiger, der den Wert `0` hat, steht für das Ende das Astes.)
- Eine Methode `insert(double x)`, die den Wert x in den Baum einfügt. Dabei muss ein neuer Knoten erzeugt und dieser so am bestehenden Baum angehängt werden, dass die Sortierung erhalten bleibt. Bereits existierende Knoten und Werte werden also nicht verschoben, sondern es wird lediglich der neue Knoten mit seinem Wert an geeigneter Stelle an einen noch nicht besetzten Ast angehängt.
  Beispiel: Im obigen Baum müsste der Wert 1.7 als neuer Knoten an den linken Ast des Knotens mit dem Wert 2.0 angefügt werden.
- Eine Methode `min()`, die als Rückgabewert den minimalen Wert im Baum ausgibt.
- Eine befreundete Funktion `operator<<`, mit der alle Zahlen im Baum sortiert ausgegeben werden. (Tipp: Sie können `<<` rekursiv mit Ästen des Baums aufrufen, solange die Äste nicht leer sind.)
- Zusatz (freiwillig): Schreiben Sie einen Destruktor, der alle im Baum dynamisch erzeugten Knoten und Blätter entfernt.

Verwenden Sie folgendes Hauptprogramm, welches auch von der Webseite erhältlich ist:

```
int main()
{
    Node tree(1.0) ;         // Basiselement des Baums
    for(int i=0; i<10; i++)  // 10 Zufallszahlen (0..2) in den Baum einhaengen
        tree.insert(2*double(rand())/RAND_MAX) ;
    cout << "Die minimale Zahl im Baum ist: " << tree.min() << endl ;
    cout << "Alle Eintraege sortiert: " << endl << "  " << tree << endl ;
}
```

Aufgabe 32: Pointer-Knobelei **freiwillig**

Was macht das folgende Programm und, wichtiger noch, wie funktioniert es? Versuchen Sie, das Programm zu analysieren ohne es ablaufen zu lassen.

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

int main()
{
    char zeile[] = "Ist das ein Stuss!" ;
    char *p = zeile ;
    while (*p++ != 'S') ;
    --------*p ;
    *++p = *(zeile+5) ;
    cout << zeile << endl ;
}
```

Aufgabe 33: Dreier-Vektoren → Vierer-Vektoren **freiwilliges Zusatztestat**[1]

Programmieren Sie zwei Klassen, `Vec3D` und `Vec4D`, zur Beschreibung von dreidimensionalen kartesischen Raumpunkten (x, y, z) und von Lorentzvektoren (ct, x, y, z). Das Einheitensystem sei im Folgenden so gewählt, dass $c = 1$ ist.

Die Vierervektoren sind dabei durch Vererbung von den dreidimensionalen Vektoren abzuleiten. In beiden Klassen sollen die eigentlichen Datenkomponenten nicht direkt von außerhalb zugreifbar sein, gegebenenfalls über geeignete Methoden. `Vec4D` kann der direkte Zugriff auf `Vec3D`-Daten erlaubt werden.

Implementieren Sie bitte mindestens folgende Methoden: (i) Setzen der Vektoren, z.B. per Konstruktor, (ii) Längenquadrat `len_sq` in beiden Klassen (3D: nach Euklid-Norm $x^2 + y^2 + z^2$, 4D: nach Minkowski-Metrik $t^2 - x^2 - y^2 - z^2$) und (iii) *nur* in der Basisklasse eine Methode `angle`, die den Winkel zwischen zwei dreidimensionalen Vektoren berechnet. Verwenden Sie bitte beim Längenquadrat der Vierervektoren dasjenige der Dreiervektoren.

Schreiben Sie ein Hauptprogramm, in dem zwei Vierervektoren mit Werten gesetzt werden. Dann ist die Länge der Vektoren und der Winkel zwischen den Raumkomponenten zu berechnen und auszugeben.

---

[1] Sie können bei dieser freiwilligen Aufgabe ein Testat erhalten, welches gewertet wird; die Aufgabe zählt aber nicht als Pflichtaufgabe.