Institut für Numerische und Angewandte Mathematik
FB Mathematik und Informatik der Universität Münster
Prof. Dr. M. Ohlberger, Dr. B. Haasdonk

29.11.2007

Aufgaben zum Praktikum

Wissenschaftliches Rechnen

WS 2007/2008 — Blatt 3

Aufgabe 1 (Template Methoden)
Gegeben ist die Datei `integrator.cc` auf der Kursseite. Dort sind zwei Klassen definiert, die verschiedene Funktionen repräsentieren. Gemeinsam ist allen eine Auswerte-Methode `evaluate(x, result)`. Weiter sind eine Menge von Quadraturen (auf dem Referenzdreieck) gegeben, die alle Methoden `npoints()`, `point(pnumber)` und `weight(pnumber)` besitzen.

(a) Schreiben Sie eine Klasse `ReferenceElementIntegrator`, welche eine template Methode `integrate(function, quadrature, result)` besitzt. Diese Methode soll die Integration der Funktion `function` mit Hilfe der Quadratur `quadrature` berechnen und in `result` zurückgeben.

(b) Bestimmen Sie durch eine geeignete `main`-Routine für alle Kombinationen der gegebenen Funktionen und Quadraturen die approximierten Integrale.

Aufgabe 2 (Template Meta Programming)
Implementieren Sie basierend auf der Tupel-Klasse in `tupeltest.cc` und `tupelutils.hh` folgende Komponenten, so dass das Programm lauffähig wird:

(a) Eine Template-Struktur `Length<class TupelType>`- zur Längenbestimmung.

(b) Eine Template-Struktur `Element<int N>`, die den Zugriff auf das N-te Element eines Tupels erlaubt.

Aufgabe 3 (Statischer Polymorphismus mit Barton-Nackman Trick)
Verwenden Sie für diese Aufgabe entweder ihre eigene Lösung der Aufgabe 2 (*) von Blatt 2 oder die Lösung `finite_volume_oop.cc`. Dort ist dynamischer Polymorphismus in Form von Objektorientierter Implementation und eine Trennung von Schnittstelle, Defaultimplementierung und Spezial-Implementierung anhand der Klasse **NumericalFluxInterface** und abgeleiteter Klassen realisiert. Implementieren Sie den Barton-Nackman Trick für diese Klassenhierarchie, um statischen Polymorphismus mit Templates zu erreichen. Führen Sie einen Laufzeitvergleich der Template- und der Objektorientierten Version für den Enquist-Osher- und den Lax-Friedrichs-Fluss durch (Compileroptionen -O2 zur Optimierung).