ulm university universität
uulm

Labor Softwareentwurf mit Multiparadigmen-Programmiersprachen
H. Falk, A. Luppold

Versuch 5
vom 16. Dezember 2013

Bearbeitung bis: 12. Januar 2013

Generische Programmierung I

Versuch 5-1: Theorie (6 Punkte)

a) Was versteht man unter dem Begriff „Metaprogrammierung“?

b) Was versteht man in C++ unter einem Template? Was ist ein variadisches Template?

c) Wodurch unterscheiden sich Templates fundamental von Klassen?

Versuch 5-2: Einführung (15 Punkte)

a) Erweitern Sie Ihre Listen-Klasse aus dem vorherigen Versuch mit Hilfe von Templates so, dass beliebige Datentypen abgespeichert werden können.

b) Schreiben Sie ein kleines Programm, das Ihre neue Liste mit unterschiedlichen Datentypen testet.

c) Beschreiben Sie die grundlegenden Unterschiede zwischen der Implementierung als Klasse und der Implementierung über ein Template.

Versuch 5-3: Ein einfacher Graph (35 Punkte)

a) Erstellen Sie eine Template-Klasse, die es ermöglicht gerichtete Graphen abzubilden. Die Knoten des Graphen sollen alle vom gleichen, aber beliebig wählbaren, Datentypen sein. Nutzen Sie für Ihre Implementierung das separat herunterladbare Gerüst der Template-Klasse MY_GRAPH und vervollständigen Sie dieses gegebenenfalls.

b) Testen Sie Ihre Template-Klasse ausgiebig mit einem kleinen selbsterstellten Testprogramm.

c) Ist es sinnvoll, in einem Knoten node nur einen Zeiger auf ein Objekt, anstatt das Objekt selbst zu speichern? Welche Arten von Zeigern eignen sich hierfür? Wo liegen

jeweils die Vor- und Nachteile? Passen Sie Ihr Testprogramm aus der vorherigen Teilaufgabe gegebenenfalls entsprechend an.

Können anstatt Zeigern auch Referenzen auf Objekte in dem Graphen abgespeichert werden? Begründen Sie Ihre Antwort.

Versuch 5-4: Bäume (15 Punkte)

a) Erweitern Sie Ihre Klasse um eine Funktion, die testet ob Ihr Graph ein gerichteter Baum ist.

b) Erweitern Sie Ihre Klasse um die Funktion MY_GRAPH::getPrevRec(). Diese soll alle Vorgängerknoten (nicht nur die direkten Vorgänger) eines als Parameter zu übergebenden Knotens zurückgeben. Hinweis: Achten Sie darauf, dass Ihre Funktion keine Endlosschleife erzeugt, wenn sie auf einen nicht baumförmigen Graphen angewandt wird.

c) Testen Sie ihre neuen Funktionen mit einem kleinen Testprogramm.

Versuch 5-5: Ausgabe (10 Punkte)

a) Erstellen Sie eine Klassen-Funktion, die den Graphen im `.dot`-Format (vgl. `http://graphviz.org/`) an einen als Parameter zu übergebenden std::ofstream ausgibt. Sie können davon ausgehen, dass der Operator << für die Objekte in Ihrem Graphen definiert ist.

b) Testen Sie die Ausgabe und erstellen Sie mithilfe des Kommandozeilentools `dot` png Graphiken für einen Graph mit Schleifen, sowie für einen Baum.

c) Wäre es sinnvoller, statt einer eigenen Memberfunktion den << operator zu überladen? Welche Vor- und Nachteile können Sie erkennen?