

Übungen zu Grafikprogrammierung in C++

Aufgabe 12 (H) Pointer (Zeiger) und Referenzen

Für einen Datentyp T ist T^* ein Zeiger. Mit anderen Worten T^* ist eine Adresse, die auf ein Objekt vom Typ T zeigt. Eine Referenz $T\&$ ist nur ein anderer Namen für ein bereits existierendes Objekt und wird häufig bei den Übergabeparametern von Funktionen verwendet.

In dieser Aufgabe gilt es eine Datenstruktur auf der Basis von `int` zu implementieren, die es erlaubt Integer hinzuzufügen oder herauszulöschen. Es soll maximal 25 Elemente beinhalten können. Es solle ein Interface `vector.h` und die Implementierung `vector.cpp` erstellt werden. Es sollen die Funktionen `addElement()` (füge neues Element hinzu) und `removeElement(int i)` (lösche Element an der Stelle `i`) implementiert werden. Eine Variable `m_last` soll jederzeit auf das letzte eingefügte Element zeigen. Ein counter soll die aktuelle Feldgröße festlegen.

Es soll eine Testapplikation implementiert werden, die diverse Elemente in das Feld einfügt und wieder löscht.

Die implementierten Dateien sollen nun in den namespace `REINVENTION` eingefügt werden.

Aufgabe 13 (H) Die Standard Template Library

Die Standard Template Library, kurz STL, wurde designed um oft verwendete Funktionen, Datenstrukturen, und Muster effizient und generisch zur Verfügung zu stellen. Ein Element der STL ist die Datenstruktur `vector`. Diese Datenstruktur soll nun verwendet werden um eine Telefonliste zu verwalten.

Implementiere ein Objekt Adresse (`Adresse.h` und `Adresse.cpp` mit den Variablen für Vornamen, Nachnamen und Telefonnummer, sowie den jeweiligen getter und setter Methoden). Die Telefonnummer soll vom Typ der in Aufgabe 12 implementierten Klasse `vector` sein. In einer `main` Funktion sollen nun eine beliebige Anzahl von Kontaktdaten eingegeben werden. Am Ende der `main` Methode soll mit einem `iterator` (Element der STL) der komplette Inhalt der Adressliste an die Standardausgabe ausgegeben werden.

Hinweis: Für Strings gibt es ebenfalls eine STL Struktur (`std::string`). Eine kurzes Tutorial findet man unter

<http://www.hlrs.de/people/mueller/tutorials/script/scriptse29.html#x53-1220005.1>. Für eine komplette Übersicht über die STL und speziell für die Funktionen von `std::vector` und `std::string` kann <http://www.sgi.com/tech/stl/> (englisch) oder <http://www.cppreference.com/> (englisch) verwendet werden.

Aufgabe 14 (H) Objektorientierung in C++

Das Konzept der Objektorientierung sollte Ihnen von früheren Semestern bekannt sein. Sie sind auch mit der Java-Syntax für OO-Programmierung vertraut. Diese Aufgabe soll dazu dienen, Sie in die C++-Syntax für die OO-Programmierung einzuführen und die Unterschiede zu Java aufzuzeigen.

- a) Erzeugen Sie folgende Klassen in C++: Polygon, eine Klasse, die Polygone modelliert, Triangle, eine Klasse für Dreiecke, Rectangle, eine Klasse für Rechtecke, Transform, eine Klasse, die Transformationen auf Graphikobjekten ausführen kann und Point2D, eine Klasse, um Punkte im zweidimensionalen Raum, die von den anderen Klassen verwendet werden können, zu modellieren. Erzeugen Sie die Klassen anhand des in Abbildung 1 aufgezeigten UML-Diagramms und erzeugen Sie daraus die polylib (ähnlich der imagelib). Alle Klassen müssen dem namespace POLYLIB zugeordnet werden.

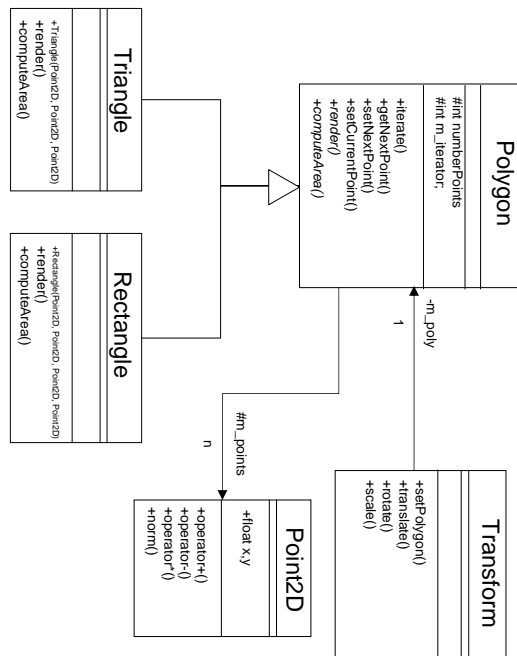


Abbildung 1: UML-Klassendiagramm

- b) Erzeugen sie wie bei allen Aufgaben die zugehörige `sln` und `vcproj` Datei
- c) Implementieren Sie alle Klassen und deren Eigenschaften bis auf die `render()`-Methode. Im SVN ist unter `Aufgaben/Aufgabe14` die Datei `mainA14.cpp` eingchecked. Diese testet die Eigenschaften der Klasse und das Interface. *Kopieren Sie diese Datei in den Ordner Ihres Projektes Aufgabe14.* Fügen Sie `mainA14.cpp` dem Projekt hinzu und testen Sie Ihre Implementierung.

Beschreibung der zu implementierenden Funktionen:

iterate erhöht den Iterator, der auf den aktuellen Punkt zeigt um eins. Am Ende der Liste fängt der Iterator wieder beim ersten Element an

getNextPoint erhöht den Iterator und gibt den aktuellen Punkt zurück

setNextPoint fügt einen Punkt hinter dem aktuellen ein (insertPoint wäre passender)

setCurrentPoint ändert aktuellen Punkt zu dem übergebenen

computeArea berechnet die Fläche des Polygons

norm berechnet die Länge des Vektors

operator+ gibt das Resultat der Vektoraddition zurück

operator- gibt den Differenzenvektor zurück

operator* gibt den skalierten Vektor zurück

translate verschiebt das Polygon um den Vektor

rotate dreht das Polygon um x Grad

scale skaliert das Polygon um den Faktor x

Hinweis:

Unter C++ kann man auch die Operatoren +,-,*,/,=,new,delete,+=, usw. überladen. Damit kann man die Operatoren auch auf komplexen Objekten verwenden.

MyObject MyObject::operator+(const MyObject &a) wäre eine sinnvolle Signatur für die Überladung des Operators +. Damit kann man auch folgenden Code verwenden
MyObject m1,m2;
MyObject& m3 = m1+m2;