Dirk Praetorius,
Marcus Page

Wintersemester 2011/12
13.01.2012

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 11

Aufgabe 11.1*. (Pointer auf Objekte)
Wie Sie in der Vorlesung gelernt haben, ist es aus Effizienzgründen besser, Objekte nicht direkt, sondern per Pointer zu übergeben. Erklären Sie diesen Umstand. Ändern Sie Ihr Multiplikationsroutine `operator*` aus Aufgabe 10.5 so ab, dass sowohl Ein- als auch Ausgabeparameter als Zeiger übergeben werden.

Aufgabe 11.2*. (Referenzen auf Objekte)
Überladen Sie die Multiplikationsroutine erneut. Nun sollen Ein- und Ausgabeparameter als Referenz übergeben werden. Was sind die Unterschiede zur vorherigen Aufgabe. Was sind Vor- und Nachteile?

Aufgabe 11.3*. (Polymorphie und abstrakte Klassen)
Den `Minsort`-Algorithmus kennen Sie bereits aus der C-Vorlesung (vgl. Folie 185). Schreiben Sie eine Klasse `MinSort` die Sie von `Sortierverfahren` (vgl. Folie 106) aus der Vorlesung ableiten. Rufen Sie die `sort`-Methode von `MinSort` anschließend mittels `sortMe` aus der Vorlesung polymorph auf. Was ist hierbei zu beachten? Was sind die Vor- und Nachteile abstrakter Datentypen? Wann würden Sie abstrakte Datentypen verwenden?

Aufgabe 11.4*. (Polymorphie)
Gegeben seien die Klassen CRectangle und CTriangle die wie folgt von CPolygon abgeleitet sind:

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

class CPolygon {
  protected:
    int width, height;
  public:
    void setValues (int a, int b)
      { width=a; height=b; }
  };

class CRectangle: public CPolygon {
  public:
    int getArea ()
      { return (width * height); }
  };

class CTriangle: public CPolygon {
  public:
    int getArea ()
      { return (width * height / 2); }
  };
```

Schreiben Sie ein aufrufendes Hauptprogramm in dem Sie ein Array von 5 Polygonen anlegen. Drei der Polygone sollen hierbei Dreiecke, und zwei Rechtecke sein. Befüllen Sie Ihre Polygone mittels `setValues` mit Daten.

Aufgabe 11.5. (Polymorphie)
Ihr Array aus Aufgabe 11.4 soll nun in einer Schleife durchlaufen werden. Hierbei soll die Fläche jedes einzelnen Polygons ausgegeben werden. Wie können Sie das bewerkstelligen?

Aufgabe 11.6. (Polymorphie und abstrakte Klassen)
Von `CPolygon` werden überhaupt keine echten Objekte instanziiert. Das ist auch nicht beabsichtigt. Diese Klasse wird nur zur Vererbung und Ausnutzung der Polymorphie verwendet. Was sollten Sie als C++ Programmierer also tun, damit sich dieser Umstand auch in Ihrem Code widerspiegelt?
Die Ausgabe aus der vorangegangenen Aufgabe ist noch immer etwas armselig. Zusätzlich zur Gesamtfläche des Polygons, soll jedes Polygon nun auch seinen Typ mitteilen. Durchlaufen Sie ihr Array wieder in einer Schleife und stellen Sie sicher, dass jede einzelne Ausgabe etwa wie folgt lautet:

```
Ich bin ein Rechteck und mein Flächeninhalt ist 24.
```

Erweitern Sie alle nötigen Klassen um diese Funktionalität sicherzustellen.

Aufgabe 11.7. (Pointer und Referenzen)
Schreiben Sie eine Funktion `minMax` der Sie ein `int`-Array übergeben und die den größten und den kleinsten Eintrag zurückliefert. Welche Möglichkeiten kennen Sie um das zu realisieren? Implementieren Sie beide und vergleichen Sie den Code.

Aufgabe 11.8. (Mehrfachvererbung)
Die Klassen `Auto` und `Fortbewegungsmittel` kennen Sie bereits aus der Vorlesung (vgl. Folie 72). Schreiben Sie eine zusätzliche Klasse `Wertgegenstand` mit einem privaten Datenfeld `wert`. Schreiben Sie außerdem entsprechende Zugriffsmethoden. Entgegen der Vorlesung soll `Auto` nun von `Wertgegenstand` und `Fortbewegungsmittel` erben. Was denken Sie über das Konzept der Mehrfachvererbung?

Aufgabe 11.9. (Diamantvererbung)
Die beiden Klassen `Wertgegenstand` und `Fortbewegungsmittel` aus der vorherigen Aufgabe sollen nun eine weitere gemeinsame Basisklasse `Gegenstand` erhalten die lediglich den Namen des Besitzers speichert. Bei der Klasse `Auto` liegt also das Diamant-Problem vor. Erweitern Sie die Klasse `Auto` um eine Methode `getBesitzer`, die den Besitzer am Bildschirm ausgibt. Welche Möglichkeiten kennen Sie um die gegebenen Unklarheiten aufzulösen. Welche würden Sie bevorzugen und warum?

Aufgabe 11.10. Implementieren Sie ein einfaches *Tic Tac Toe* Spiel. Falls Sie Ihnen nicht bekannt sind, können Sie die Regeln unter `http://de.wikipedia.org/wiki/Tic_Tac_Toe` nachlesen. Das Spiel soll mindestens die folgenden Kriterien erfüllen:

1. Es sollen zwei Spieler gegeneinander antreten können die jeweils abwechselnd am Zug sind.
2. Es soll automatisch erkannt werden, wenn ein Spieler gewonnen hat.
3. Nach jedem Zug soll das aktuelle Spielfeld grafisch in der Konsole ausgegeben werden.
4. Kann sich am Ende keiner der Spieler durchsetzen, so soll `'unentschieden'` ausgegeben werden.

Tip: Planen Sie Ihre Datenstrukturen und den Programmverlauf bevor sie mit der tatsächlichen Implementierung beginnen.