Prof. Dr. Stefan Brass
Dipl.-Inform. Annett Thüring
Institut für Informatik
MLU Halle-Wittenberg

13. Dezember 2010

Vorlesung "Objektorientierte Programmierung" — 2. Programmiertest (Aufgabe C) —

Hinweise/Regeln:

- Bearbeitungsdauer: 75 Minuten.
- Am Ende gibt es nur "bestanden" und "nicht bestanden", keine Punkte für partiell korrekte Lösungen.
- Sie dürfen bis zu 3 Blätter "Spickzettel"/"Quick Reference" verwenden, sowie ein Buch/Hefter (nicht zu groß, es muß noch auf den Tisch passen ohne zu stören).
- Eigene Notebooks, PDAs, etc. dürfen nicht verwendet werden. Mobiltelefone bitte ausschalten (oder mit der Aufsicht besprechen).
- Legen Sie bitte ein neues Verzeichnis `oop10/praxistest2` in Ihrem Homeverzeichnis an. Sie dürfen ein eventuell existierendes `Makefile` hineinkopieren oder Shellskripte zur Programmentwicklung. Ansonsten dürfen Sie keine weiteren Dateien kopieren oder öffnen, sondern nur den Quellcode des zu entwickelnden Programms.
- Kopieren Sie sich dann bitte die Datei `~brass/oop10/test2/p2c.cpp` in dieses Verzeichnis. Diese Datei enthält ein Hauptprogramm, das als Test dient. Ihre Aufgabe ist, an der markierten Stelle die in der Aufgabenstellung beschriebene Klasse einzufügen. Am Ende muss das Hauptprogramm genau der vorgegebenen Version entsprechen, zwischenzeitlich können Sie natürlich eigene Testaufrufe einbauen.
- Sie dürfen selbstverständlich die zur Programmentwicklung üblichen Programme verwenden (auch `man`), aber nicht auf das Internet zugreifen (z.B. kein Web-Browser, kein EMail-Programm).
- Die Homeverzeichnisse werden für Zugriffe von außen gesperrt. Falls Sie spezielle Zugriffsrechte gesetzt hatten, müssen Sie diese nach dem Test selbst wieder herstellen.
- Selbstverständlich dürfen Sie auch Microsoft Visual Studio oder eine andere IDE benutzen. Das abgegebene Programm muss aber unter Linux/g++ laufen.
- Tauschen Sie keinesfalls irgendwelche Dinge mit den Nachbarn aus. Notfalls rufen Sie eine Aufsichtsperson zur Kontrolle.
- Sie müssen Mindestanforderungen an den Programmierstil erfüllen, z.B. entsprechend der Programmstruktur einrücken, sinnvolle Variablennamen wählen und zum Verständnis notwendige Kommentare schreiben.
- Fragen Sie, wenn Ihnen die Aufgabe nicht klar ist! Melden Sie sich bitte auch, wenn es technische Schwierigkeiten mit Ihrem Rechner gibt.
- Wenn Sie an einer unverständlichen Fehlermeldung länger festhängen, können Sie probieren, zu fragen. Wir wollen aber nicht zu viele Tipps geben.

Aufgabe (Variante C)

Schreiben Sie eine Klasse `wuerfelTest` zum Testen eines Würfels. Die Klasse muss drei Methoden haben (und einen Konstruktor):

- `wurf`: Diese Methode wird aufgerufen, um das Ergebnis eines Wurfes zu speichern. Sie hat einen Parameter `i` vom Typ `int`, und keinen Rückgabewert. Gültige Werte von `i` können zwischen 1 und 6 liegen, alle anderen Werte sollen einfach ignoriert werden. Diese Methode soll die Anzahl der Versuche, bei denen der Wurf des Würfels `i` ergab, um eins erhöhen.
- `anz_wuerfe`, mit einem Parameter `i` vom Typ `int` (wie in der Methode `wurf`). Diese Methode liefert die Anzahl (als `int`), wie häufig vorher `wurf` mit dem Parameterwert `i` aufgerufen wurde, d.h. wie häufig der Wert `i` gewürfelt wurde. Falls `wurf` vorher noch nicht aufgerufen wurde, muß natürlich 0 geliefert werden. Falls der Parameter `i` außerhalb des gültigen Bereichs von 1 bis 6 liegt, soll −1 geliefert werden.
- Schließlich soll die Methode `chi_quadrat` folgende Summe liefern:

  $$\sum_{i=1}^{6} \frac{(n_i - m_i)^2}{m_i},$$

  dabei ist n_i die Anzahl von Wurfe mit Ergebnis `i` (also n_i =`anz_wuerfe(i)`) und m_i das Ergebnis bei einem korrekten Würfel, also die Gesamtanzahl von Würfen durch 6 (m_i hängt beim Würfel nicht von i ab, aber die obige Formel gilt auch allgemeiner). Die Methode `chi_quadrat` hat keine Parameter und liefert einen Wert vom Typ `double`. (Falls das Ergebnis kleiner als 11.07 ist, wird man den Würfel mit Signifikanzniveau 5% als fair ansehen, das ist aber für diese Programmieraufgabe nicht relevant.) Falls die Methode aufgerufen wird, wenn noch nicht ein einziger Wurf verzeichnet ist, soll `-1` geliefert werden (falls man obige Formel anwendet, würde durch 0 dividiert werden).
- Außerdem müssen Sie selbstverständlich einen Konstruktor schreiben, der alle Anzahlen auf 0 initialisiert.

Beispiel: Nach den Aufrufen `wurf(1)`, `wurf(2)`, `wurf(1)`, `wurf(5)`, `wurf(1)`, `wurf(5)` müsste `anz_wuerfe(1)` den Wert `3` liefern, `anz_wuerfe(2)` den Wert `1`, `anz_wuerfe(3)` den Wert `0`, `anz_wuerfe(4)` auch `0`, `anz_wuerfe(5)` den Wert `2`, und `anz_wuerfe(6)` den Wert `0`. Da insgesamt `6` Mal gewürfelt wurde, ist $m_i = 1$ (für $i = 1, \ldots, 6$). Die Summe ist dann also

$$\frac{(3-1)^2}{1} + \frac{(1-1)^2}{1} + \frac{(0-1)^2}{1} + \frac{(0-1)^2}{1} + \frac{(2-1)^2}{1} + \frac{(0-1)^2}{1} = 4+0+1+1+1+1 = 8$$

Programmier-Hinweise:

Sie können das Berechnungsverfahren frei wählen. Eine Möglichkeit wäre, sich in dem Objekt ein Array von 6 Elementen anzulegen, in dem Sie die Anzahl Würfe mit dem jeweiligen Ergebnis speichern (da der Indexbereich bei 0 beginnt, wäre auch zu überlegen, ob ein Array von 7 Elementen den Programmcode nicht vereinfacht). Außerdem könnte man eine Variable (Attribut) für die Gesamtanzahl der Würfe verwenden (alternativ kann man sie als Summe über den Array-Elementen berechnen).