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

14. Dezember 2010

Vorlesung „Objektorientierte Programmierung“ — 2. Programmiertest (Aufgabe B) —

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/p2b.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 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 B)

Schreiben Sie eine Klasse `tower`, welche einen Turm bestehend aus maximal 100 Etagen verwaltet. In jeder Etage steht genau eine nichtnegative ganze Zahl. Die Zahlen können mehrfach in einem Turm vorhanden sein. Die Klasse besitzt die drei Methoden:

- `heighten`, die einen Parameter `n` vom Typ `int` hat, und einen Rückgabewert vom Typ `bool`. Diese Methode legt die im Parameter übergebene Zahl oben auf den Turm. Dabei muss vor dem Speichern geprüft werden, ob der Turm schon die maximal zulässige Höhe erreicht hat. Ist dies der Fall, so gibt die Methode `false` zurück. Negative Eingabewerte sind unzulässig. Somit muss auch in diesem Fall der Rückgabewert `false` sein. Darf der Turm noch erhöht werden, so wird der Zahlenwert in der neu hinzugekommenen Etage gespeichert und die Methode gibt `true` zurück.
- `shorten`, ohne Parameter aber mit einem Rückgabewert vom Typ `int`. Diese Methode soll die oberste Etage des Turmes abbauen und die in der Etage stehende Zahl als Funktionswert zurückgeben. Es muss darauf geachtet werden, dass, falls der Turm Null Etagen besitzt, keine Etage entfernt werden kann. In diesem Fall soll die Methode `-1` zurückgeben. Wird die Methode mehrfach aufgerufen, so wird die jeweils zuletzt auf den Turm gelegte Zahl (ausgenommen der bereits gelöschten Zahlen) entfernt und zurückgegeben.
- `get_max`, ebenfalls ohne Parameter aber mit einem Rückgabewert vom Typ `int`. Diese Methode soll die Nummer der Etage, in welcher die gröte Zahl steht, als Funktionswert zurückgeben. Falls der Turm Null Etagen besitzt, soll die Methode `-1` zurückgeben.
- Außerdem müssen Sie selbstverständlich einen Konstruktor schreiben, der den Turm mit einer Höhe von Null Etagen initialisiert.

Beispiel: Nach den Aufrufen `heighten(10)`, `heighten(8)`, `heighten(12)`, `heighten(5)` müsste `shorten()` den Wert `5` liefern. Ruft man danach `heighten(6)` auf, so sollen die nächsten beiden Aufrufe von `shorten()` die Werte `6` und `12` liefern. Die größte Zahl (10) befindet sich in der 1. Etage, somit sollte der Aufruf von `get_max` eine `1` liefern.

Beispiel-Algorithmus:

Sie können das Berechnungsverfahren frei wählen. Eine Möglichkeit wäre, sich ein Array mit 100 Elementen zu erstellen, und mit `-1` zu initialisieren. Speichert man nun die Zahlen nacheinander im Array ab, so enthält die Position, hinter der das letzte eingefügte Element gespeichert wurde, immer den Wert `-1`. In der Methode `shorten` muss beim Entfernen eines Elementes weiterhin darauf geachtet werden, dass dieser Zustand auch nach dem Löschen eines Elementes wieder hergestellt wird.