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

08. November 2010

Vorlesung "Objektorientierte Programmierung" — 1. Programmiertest (Aufgabe B) —

Hinweise/Regeln:

Vermeiden Sie bitte jedes Verhalten, was als Täuschungsversuch (miss-)verstanden werden könnte. Wir müssten Sie sonst disqualifizieren, d.h. Sie hätten den Programmiertest nicht bestanden. Sie werden sowohl durch das Aufsichtspersonal im Rechnerpool überwacht, als auch auf den Rechnern per Programm/Aufzeichnung, sowie über das Netz. Wenn wir nachträglich in der Aufzeichnung einen Täuschungsversuch erkennen sollten, müssen wir Ihnen auch einen zunächst als bestanden gewerteten Programmiertest wieder aberkennen.

- Bearbeitungsdauer: 60 Minuten (plus eventuell leichte Verlängerung).
- Am Ende gibt es nur "bestanden" und "nicht bestanden", keine Punkte für partiell korrekte Lösungen. Möglicherweise werden am Ende noch einige Minuten Bearbeitungszeit angehängt, wenn mehrere Teilnehmer ganz knapp vor einer fertigen Lösung sind — aber nicht funktionierende Lösungen werden nicht akzeptiert, selbst wenn nur eine Kleinigkeit fehlt.
- Sie dürfen bis zu 3 Blätter "Spickzettel"/"Quick Reference" verwenden, sowie ein Buch (nicht zu groß, es muß noch auf den Tisch passen ohne den Nachbarn zu stören — ein Aktenordner wäre nicht akzeptabel).
- Sie dürfen außerdem ein Blatt leeres Papier und einen Stift verwenden, um sich Notizen zu machen.
- Eigene Notebooks, PDAs, etc. dürfen nicht verwendet werden. Mobiltelefone bitte ausschalten (oder mit der Aufsicht besprechen).
- Sie dürfen nur Terminalfenster für den Compileraufruf und Editor-Fenster offen haben. Selbstverständlich können Sie auch einen Debugger benutzen, sowie `make` und `man`. Dagegen ist z.B. ein Web-Browser nicht erlaubt, und auch kein EMail-Programm (allgemein keine Zugriffe auf das Internet). Ebensowenig dürfen Sie versuchen, Dateien außerhalb Ihres Homeverzeichnisses abzulegen, oder auf Dateien außerhalb Ihres Homeverzeichnisses zuzugreifen (außer Dateien, die der Compiler verwendet).
- Das Programm müssen Sie vollständig neu eintippen, und nicht durch Modifikation von eventuell in Ihrem Homeverzeichnis vorhandenen Dateien erstellen. Für `make` dürfen Sie ein vorhandenes `Makefile` benutzen und bei Bedarf modifizieren.
- 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. Sie dürfen auch Programme zum Remote Login auf `anubis` verwenden, sowie zum Kopieren von Dateien zwischen Windows und Linux.

- 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.
- Fragen Sie, wenn Ihnen die Aufgabe nicht klar ist!
- Wenn Sie an einer unverständlichen Fehlermeldung länger festhängen, können Sie probieren, zu fragen. Wir wollen aber nicht zu viele Tipps geben.

Aufgabe (Variante B)

Sie sollen die Wurzel einer Zahl berechnen, nicht mit einer Bibliotheksfunktion, sondern mit dem unten angegebenen Iterationsverfahren. Es soll zunächst der Text

```
Bitte Zahl eingeben:
```

ausgegeben werden (am Ende mit einem Leerzeichen, aber ohne Zeilenumbruch). Dann soll eine Zahl x eingelesen werden in eine Variable vom Typ `double` (Fliesskomma-Zahl). Wenn der eingegebene Zahlwert < 0 ist, soll folgende Fehlermeldung ausgegeben werden:

```
Negative Zahlen haben keine Wurzel!
```

(mit einem Zeilenumbruch am Ende). Das Programm soll sich dann beenden. Sie können davon ausgehen, dass eine syntaktisch korrekte Fließkomma-Zahl eingegeben wurde.

Zur Berechnung der Wurzel führen Sie eine `double`-Variable für den aktuellen Näherungswert ein, die Sie mit 1 initialisieren. Wenn y der bisherige Näherungswert ist, und x die Zahl, von der Sie die Wurzel berechnen wollen, ist der nächste Näherungswert: $(y + x/y)/2$. Wiederholen Sie die Berechnung von Näherungswerten so lange, bis das Quadrat des aktuellen Näherungswertes im Bereich $x - 0.01$ bis $x + 0.01$ liegt (einschließlich der Grenzen), also auf zwei Stellen hinter dem Komma genau ist. Weil die Schleife solange laufen soll, bis das Quadrat des Näherungswertes weder zu gross noch zu klein ist, brauchen Sie eventuell einen der booleschen Operatoren aus der letzten Vorlesung: `&&` (und), `||` (oder), `!` nicht.

Geben Sie berechneten Näherungswert in folgender Form aus:

```
Wurzel: 1.41667
```

(mit einem Zeilenumbruch am Ende, das Format der Fliesskomma-Zahl muss nicht unbedingt übereinstimmen).

Beispiele für Tests:

- Bei einer negativen Eingabe muss die vorgegebene Fehlermeldung ausgegeben werden.
- Ein Beispiel-Lauf ist:

  ```
  Bitte Zahl eingeben: 2
  Wurzel: 1.41667
  ```

  Das Quadrat dieser Zahl ist 2.01.