Dr. S. Lewandowski **Einführung in die Informatik I (autip)** WS 2008/09

Übungsblatt 03

Ausgabe: 05.11. Abgabeschluss: Mittw., 12.11., 9:45 Uhr, `eClaus.informatik.uni-stuttgart.de`

Abgabe erfolgt ausschließlich elektronisch über `eClaus.informatik.uni-stuttgart.de` – versuchen Sie nach Möglichkeit die Abgabe nicht in der letzten Minute zu machen!

Von jedem Aufgabenblatt werden maximal 20 Punkte auf den Schein angerechnet.

Aus gegebenen Anlass noch einige Anmerkungen: Kommentieren Sie Ihr Programm! Dies heißt insbesondere, dass Sie neben Autor und Datum in den Kommentarzeilen die Idee für Ihr Vorgehen beschreiben sollten und die wesentlichen Bestandteile des Programms beschreiben. Dies ist in der Regel mehr als der Text „Aufgabenblatt 2: Aufgabe 5“.

Einrückungen machen Programme deutlich lesbarer! Nutzen Sie diese Möglichkeit.

Halten Sie sich an die Aufgabenstellung: wenn eine Prozedur mit Parametern gefordert ist, reicht es nicht ein Hauptprogramm mit gleicher Funktionalität zu schreiben.

Und noch ein Wort zum Kopieren von Lösungen anderer: Versuchen Sie unbedingt, zu Beginn des Semesters die Programme selbst zu schreiben, es wird definitiv nicht leichter bis zum Ende des Semesters. Wenn Sie selbst nicht weiter kommen, setzen Sie sich mit Ihren Kommillitonen zusammen, lassen Sie sich die Programmierschritte nochmals erklären oder schauen Sie Ihren Kommillitonen beim Programmieren zu, aber programmieren Sie die Aufgaben unbedingt selbst aus!

---

1. (3+2+2+2 Punkte) **Summe der natürlichen Zahlen:** In der Vorlesung wurde eine Funktion zur Berechnung der Fakultät geschrieben. Dabei wurden über das Attribut `integer'last` Fehler durch Überschreiten des erlaubten Zahlenbereichs abgefangen. Wir wollen dieses Prinzip nun auf eine andere Problemstellung übertragen.

   (a) (3 Punkte) Schreiben Sie ein Programm, das die Summe der natürlichen Zahlen 1+2+3+...+n für eine einzugebene Zahl n berechnet. Dabei soll ein Überschreiten des Zahlenbereichs im Programm abgefangen werden.

   (b) (2 Punkte) Eine Anekdote über den Mathematiker Gauß erzählt, dass er in der Grundschule in Mathematik die Aufgaben immer so schnell gerechnet hat, dass der Lehrer ihm die Aufgabe stellte, die Zahlen von 1 bis 50 zu addieren, in der Hoffnung, dass so der junge Gauß einige Zeit beschäftigt sei. Dieser lieferte die korrekte Antwort 1275 jedoch nach ungewöhnlich kurzer Zeit: Er hatte sich die Zahlen 1 bis 50 einmal aufsteigend und einmal absteigend aufgeschrieben und gesehen, dass die übereinanderstehenden Zahlen jeweils 51 ergaben.

   ```
    1  2  3  ...  48 49 50
   50 49 48  ...   3  2  1
   ```

   Die Summe der Zahlen 1 bis 50 war also die Hälfte von 50 mal 51, also 1275.
   Geben Sie eine allgemeine Formel für die Summe der natürlichen Zahlen

   $$\sum_{i=1}^{n} i$$

   an (1 Punkt) und beweisen Sie deren Korrektheit mit vollständiger Induktion (1 Punkt).

(c) (2 Punkte) Verwenden Sie die Formel aus Teil (b) und schreiben Sie damit Ihr Programm aus Teil (a) neu.

(d) (2 Punkte) Geben Sie an, mit welchen Zahlen Sie Ihr Programm getestet haben, bis Sie sich sicher waren, dass es korrekt arbeitet. Begründen Sie Ihre Antwort.

2. (2+5+1+2+3(+3) Punkte) **Freitag der 13. – Teil 2:** Wenn Programme umfangreicher werden, bieten Prozeduren und Funktionen eine Möglichkeit Programme strukturiert aufzubauen und den Ablauf in überschaubare Teile zu gliedern.

   Wir wollen hier nun ein Programm schreiben, das ein Datum einliest, den Wochentag an diesem Datum berechnet und diesen dann ausgibt. Der heute gültige Gregorianische Kalender beginnt am Freitag, 15. Oktober 1582. Informationen zum Gregorianischen Kalender finden Sie z.B. unter `http://de.wikipedia.org/wiki/Gregorianischer_Kalender`

   (a) (2 Punkte) Überlegen Sie sich zunächst, welche Bausteine Sie benötigen. Das Hauptprogramm sollte nur folgende Befehle enthalten:

   ```
   begin
     Eingabe_Datum; -- liest Werte für die Variablen Tag, Monat und Jahr ein
     Wochentag := Berechne_Wochentag(Tag,Monat,Jahr);
     Ausgabe_Wochentag(Wochentag);
   end;
   ```

   Zumindest die Funktion `Berechne_Wochentag` wird sich sicherlich in kleinere Bausteine aufteilen. Solch ein Baustein sollte nie so groß werden, dass er nicht mehr auf eine (Bildschirm-)Seite passt.

   (b) (5 Punkte) Implementieren Sie Ihre Bausteine aus Teil (a). Beachten Sie dabei, dass das eingegebene Datum gültig sein muss, insbesondere soll es nach dem 15. Oktober 1582 liegen.

   (c) (1 Punkt) Wir werden später im Semester noch lernen, wie man eigene Datentypen, z.B. für ein Datum, definiert. Überlegen Sie sich, welche Konstrukte Ihnen bei der Programmierung dieser Aufgabe gefehlt haben, die aus Ihrer Sicht das Programm übersichtlicher, besser lesbar oder einfacher zu schreiben gemacht hätten.

   (d) (2 Punkte) Vergleicht man die Wochentage vom 1.1.2000 und 1.1.2400, so sieht man (unter Berücksichtigung der Schaltjahrregeln), dass sich der Kalender alle 400 Jahre bzgl. der Wochentage exakt wiederholt.

   Untersuchen Sie, wie oft der 13. eines Monats auf einen Montag, Dienstag, ... oder Sonntag fällt.

   (e) (3 Punkte) Bestimmen Sie die längste Zeitspanne (in Tagen gerechnet) ohne einen Freitag, den 13., und geben Sie diese und alle Zeiträume dieser Länge innerhalb eines 400-Jahres-Zyklus an. (Ein einfacher Algorithmus ist hier ausreichend.) Spielt es eine Rolle, an welchem Tag dieser 400-Jahres-Zyklus beginnt? Begründen Sie Ihre Antwort und fügen Sie diese als Kommentarzeilen mit ein.

   Zusatzaufgabe (+ 3 Punkte): Erweitern Sie Ihr Programm so, dass der Benutzer einen Zeitraum angeben kann, für den obiges Problem gelöst werden soll. Welche Zusätzlichen „Probleme“ können hier auftreten? Versuchen Sie einen effizienten Algorithmus zu entwickeln, um die längste Spanne zu berechnen.
