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

Übungsblatt 01

Ausgabe: 22.10. Abgabeschluss: Mittw., 29.10., 9:45 Uhr, `eClaus.informatik.uni-stuttgart.de`

Ablauf der Übungen: Ausgabe jeweils in der Übung, Abgabe jeweils 9:45 Uhr am Tag der folgenden Übung (also erstmals am 29.10., danach jeweils Mittwochs, also 5.11., 12.11., jeweils 9:45 Uhr). 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.

Noch eine Anmerkung, bevor es los geht: Programmieren ist ein „Handwerk" – man lernt es nur, indem man es selbst durchführt (ein Schreinerlehrling wird das Hobeln einer Tischplatte nie erlernen, wenn er seinem Meister beim Hobeln immer nur zuschaut, er muss es selbst tun, um ein Gefühl dafür zu bekommen). Die Anfangshürde ist dabei immer die schwerste. Jeder muss diese aber selbst überwinden. Wenn Sie nicht weiterkommen, diskutieren Sie mit Ihren Kommillitonen über Ideen zur Lösung. Programmieren Sie diese aber unbedingt selbst aus.

---

1. (3 Punkte) **Algorithmus:** Ein Algorithmus ist ein exakt formuliertes Verfahren, das von jedem, der den Text des Algorithmus erhält, auf gleiche Weise durchgeführt werden kann, ohne dass weitere Erläuterungen notwendig sind.

   Ein Computer führt in der Regel Algorithmen aus. Wichtig ist zunächst die Eindeutigkeit des Algorithmus. Hier ein umgangssprachliches Beispiel:

   Begib dich an die Nordostecke des Hauses in der Universitätsstraße 38. Warte bis Mitternacht. Peile den äußersten Deichselstern des Sternbilds des großen Wagen an und gehe dann genau 107 Schritte in diese Richtung. Grabe hier und du stößt in 2 Ellen Tiefe auf den Schatz.

   Ist diese Vorschrift eindeutig? Welche zusätzlichen Informationen fehlen?

2. (4+2 Punkte) **Algorithmus II:** Programmieranfängern bereitet häufig das abstrakte Denken in Schleifen und Fallunterscheidungen Schwierigkeiten; ebenso ist oft nicht klar, wie komplex die kleinsten Schritte sein können, die der „Ausführende" (beim Programmieren ist dies der Computer) noch versteht.

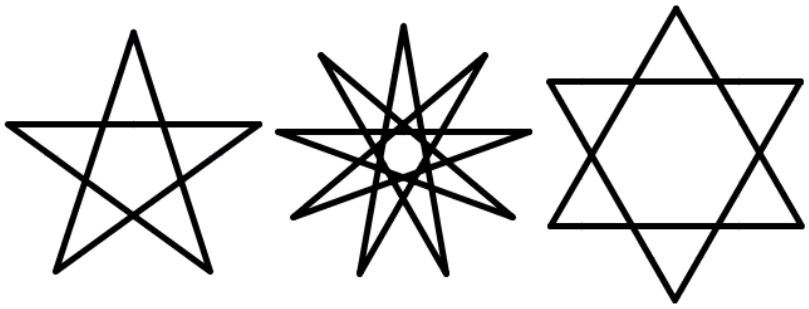
   Der Winter kommt bald ... schreiben Sie einen umgangssprachlichen Algorithmus zum *Autoreifenwechsel auf Winterreifen*. Identifizieren Sie dabei insbesondere die Schleifen und Fallunterscheidungen. Überlegen Sie sich, wie detailliert die Beschreibung sein muss, um von jedem auf gleiche Weise ohne weitere Erläuterungen durchgeführt werden zu können. Welche „Anweisungen" könnte man dabei (ggf. mit geeignet gewählten Parametern) zusammenfassen, um diese in anderen Situationen (z.B. Reifenpanne) wiederverwenden zu können?

   Überlegen Sie sich eine weitere Aufgabenstellung aus dem Alltag und bearbeiten Sie diese auf die gleiche Art und Weise (z.B. Kochrezept, Fotobearbeitung, Festplatte im Rechner austauschen, ... seien Sie kreativ).

   Zur Darstellung können Sie die Algorithmentexte entweder einfach durch Einrückungen strukturieren oder – falls aus der Schule bekannt – Darstellungsformen wie Nassi-Shneiderman-Diagramme (Struktogramme) oder Programmablaufpläne verwenden.

3. (3 Punkte) **Zeichnen von regelmäßigen n-Ecken:** In der letzten Woche wurde in der Übung eine `procedure dreieck(len:integer)` zum Zeichnen von Dreiecken mit Kantenlänge `len` erstellt. Erweitern Sie die Prozedur zu einer `procedure vieleck(anz:integer;len:integer)`, die als Parameter neben der Kantenlänge auch die Anzahl der Ecken übergeben bekommt. Fügen Sie die Idee als Kommentarzeilen im Programm hinzu.

4. (3 Punkte) **Zeichnen von Sternen mit n Zacken:** Schreiben Sie eine Prozedur zum Zeichnen von Sternen, die Anzahl der Zacken und die Kantenlänge sollen dabei als Parameter übergeben werden. Zunächst soll die Prozedur Sterne mit 4 Zacken (ein Quadrat) oder einer ungeraden Anzahl von Zacken zeichnen können. Wird als Parameter eine Zackenanzahl übergeben, die von der Prozedur nicht fehlerfrei gezeichnet werden kann, so soll eine entsprechende Meldung ausgegeben werden.



Zusatzaufgabe (schwer, 6 Punkte): Erweitern Sie Ihr Programm, so dass auch gerade Zackenanzahlen gezeichnet werden können. (Hinweis: Da AdaLogo derzeit nur ganze Zahlen verarbeitet, müssen Sie etwas tüfteln ... `len/cos(`α`)` kann für kleine Winkel α durch `len+len*`α^2`/2` angenähert werden, wobei α dann im Bogenmaß angeben werden muss, $\pi \approx$`22/7`, beachten Sie ggf. noch, dass der Operator „/“ ganzzahlig dividiert)

5. (2+3+1 Punkte) **Zeichnen von Kreisen:** Es gibt in AdaLogo keinen Befehl, um einen Kreis zu zeichnen. Die letzte Woche vorgestellte Variante `for i in 1..360 loop forward(1); turn(1); end loop;` ist unbefriedigend, da wir dabei keinen Radius und keinen Mittelpunkt angeben können.

   (a) (2 Punkte) Kreise lassen sich durch Vielecke annähern. Erweitern Sie Ihre Prozedur aus Aufgabe 3 zu einer `procedure kreis(x:integer;y:integer;r:integer)`, die einen Kreis mit Mittelpunkt `(x,y)` und Radius `r` zeichnet. (Hinweis: Nutzen Sie die Beziehung zwischen Umfang und Radius eines Kreises!)

   (b) (3 Punkte) Die mathematische Definition eines Kreises besagt, dass genau die Punkte (p_x, p_y) zu einem Kreis mit Mittelpunkt (m_x, m_y) und Radius r gehören, für die $(p_x - m_x)^2 + (p_y - m_y)^2 = r^2$ gilt. Schreiben Sie mit Hilfe dieser Eigenschaft eine `procedure schoener_kreis(x:integer;y:integer;r:integer)`, die ebenfalls einen Kreis mit Mittelpunkt `(x,y)` und Radius `r` zeichnet.

   (c) (1 Punkt) Verwenden Sie Ihre Prozeduren aus den Teilaufgaben (a) oder (b), um die Olympischen Ringe zu zeichnen.

Fragen können im Forum `http://www.autip.de/forum/viewforum.php?f=236` diskutiert werden. Weitere Informationen zur Vorlesung und Übung unter `http://www.fmi.uni-stuttgart.de/fk/lehre/ws08-09/autip1/`