Dr. S. Lewandowski **Einführung in die Informatik I (autip)** WS 2005/06

Übungsblatt 08

Ausgabe: 14.12. Abgabeschluss: Mittw., 21.12., 9:45 Uhr, `eClaus.informatik.uni-stuttgart.de`

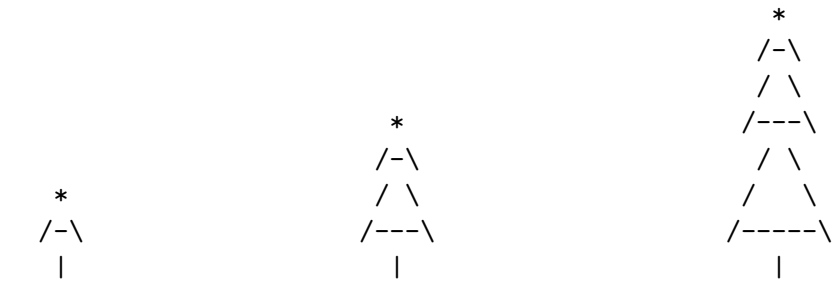
Abgabe erfolgt ausschließlich elektronisch über `eClaus.informatik.uni-stuttgart.de` – versuchen Sie nach Möglichkeit die Abgabe nicht in der letzten Minute zu machen!

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

---

1. (5(+2) Punkte, leicht–mittel) Zur Wiederholung nochmal eine relativ leichte Programmieraufgabe . . . **Der Weihnachtsbaum**: Weihnachten naht, das Jahr 2005 ist fast zu Ende und überall sieht man Weihnachtsbäume, sogar auf dem Bildschirm!?!

   Schreiben Sie ein Ada-95-Programm, das in Abhängigkeit von einem Parameter `n` Christbäume auf dem Bildschirm ausgibt. Die Christbäume sollen nach folgendem Muster gebaut sein:



   `Schwabenversion (n=1), Studentenversion (n=2), Professorenversion (n=3) usw.`

   (a) (5 Punkte) Schreiben Sie ein Ada-95-Programm, das zunächst eine Zahl `n` einliest und dann den entsprechenden Baum (mit Hilfe von Schleifen) zentriert auf dem Bildschirm ausgibt. Die Bildschirme, die Ihnen zur Verfügung stehen, können 25 Zeilen mit je 79 Zeichen darstellen (definieren Sie entsprechende Konstanten in Ihrem Programm). Es gibt eine maximale Grenze für `n`, so dass der Baum noch auf dem Bildschirm angezeigt werden kann. Wird ein größeres `n` eingegeben, so soll eine Erklärung auf dem Bildschirm ausgegeben werden, die den Benutzer zu mehr Bescheidenheit aufruft.

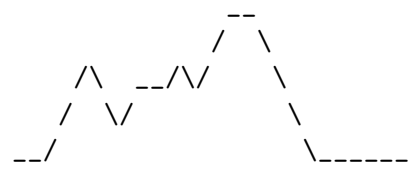
   (b) **Zusatzaufgabe (mittel–schwer, 2 Punkte):** Lesen Sie das `n` nun als Übergabeparameter aus der Kommandozeile ein. Der Aufruf `weihnachstbaum 3` sollte also dann die Professorenversion ausgeben, der Aufruf `weihnachstbaum 42` würde sich nicht mehr auf 79 Zeichen Breite darstellen lassen und würde zu dem entsprechenden Hinweis auf dem Bildschirm führen. Programmaufrufe, die nicht die Form `weihnachtsbaum <Zahl>` haben, sollen nur einen Bedienungshinweis ausgeben. Hinweis: Um Argumente von der Kommandozeile zu lesen benötigen Sie das Paket `Command_Line`.

2. (3(+2) Punkte, leicht–mittel) **Binärbrüche:** Wie bei Dezimalzahlen können wir auch Binärzahlen in Vor- und Nachkommastellen (getrennt durch einen '.') aufteilen. Ein Binärbruch sei also definiert als eine Sequenz von 0 und 1 mit genau einem Punkt, so dass vor und nach diesem jeweils mindestens eine Ziffer steht. Vor der ersten Ziffer steht optional noch ein Vorzeichen (Minus oder Plus) und eine beliebige Anzahl von Leerzeichen.

   (a) (3 Punkte) Beschreiben Sie die Menge der Binärbrüche als EBNF, Grammatik und Syntaxdiagramm (das Syntaxdiagramm braucht nicht in eClaus abgegeben zu werden).

   (b) **Zusatzaufgabe (2 Punkte, mittel):** Verändern Sie Ihre EBNF und Grammatik so, dass unnötige führende Nullen verboten sind.

3. (3+4+6+4 Punkte, leicht–schwer) **Bergpanoramen:** Mit den Zeichen `/`, `_` und `\` lassen sich Bergpanoramen zeichnen, z.B.



Dabei sollen Anfang und Ende gleich hoch und nicht höher als irgendein anderer Punkt des Panoramas liegen. Wenn wir die Zeichen eines Bergpanoramas einfach von links nach rechts lesen, ohne dabei die vertikale Anordnung auf dem Papier zu beachten, so können wir ein Bergpanorama als Wort einer Sprache auffassen, in unserem Beispiel das Wort `__///\\/__/\//__\\\\_____`.

(a) (3 Punkte, leicht–mittel) Beschreiben Sie die Sprache der Bergpanoramen in EBNF. Geben Sie auch eine kontextfreie Grammatik für die Sprache der Bergpanoramen an. (Entwerfen Sie auch ein Syntaxdiagramm, dieses brauchen Sie jedoch nicht in eClaus abzugeben.)

(b) (4 Punkte, mittel) Schreiben Sie ein Ada-95-Programm, bei dem der Benutzer eine Zeichenkette eingeben kann, die in ein Bergpanorama umgesetzt werden soll.

Mit Hilfe von `Get_Line(<string>,<integer>)`, kann man in eine String-Variable eine Zeile Text einlesen. Die Integer-Variable (ein Ausgangsparameter) gibt danach an, wieviel Zeichen eingelesen wurden.

Die eingelesenen Zeichen sollen nun als Bergpanorama ausgegeben werden: Um es uns etwas einfacher zu machen, betrachten wir dabei den linken Bildschirmrand als Grundlinie – der Benutzer muss also den Kopf um 90 Grad drehen. Um es dem Benutzer etwas einfacher zu machen, passen wir die Zeichen entsprechend an. Die Eingabe `/_\_` würde dann zu der Ausgabe

```
\
 |
/
|
```

D.h., ein `'/'` wird als `'\'` ausgegeben und umgekehrt und statt dem `'_'` geben wir ein `'|'` aus. Achten Sie auch darauf, dass die Einrückungen korrekt sind. Entspricht die Eingabe keinem Bergpanorama, so soll dies dem Benutzer mitgeteilt werden.

(c) (6 Punkte, mittel–schwer) Geben Sie das Bergpanorama nun auch richtig herum aus. Die Eingabe `/_\_` soll also die Ausgabe

```
 _
/ \_
```

erzeugen. (Hinweis: Es empfiehlt sich hier ein schrittweises Vorgehen: Definieren Sie sich zunächst eine Datenstruktur, die für jede Position im Bergpanorama die zugehörige Höhe angibt; Sie können dann jede Schicht einzeln ausgeben – beginnend mit der maximalen Höhe (definieren Sie sich zur Berechnung eine entsprechende Funktion) bis zum Tal (Höhe 0).)

(d) (4 Punkte, mittel) Schreiben Sie ein Ada-95-Programm, das zufällige Bergpanoramen erzeugt, indem der Ableitungsprozess eines Wortes der Sprache simuliert wird. (Führen Sie für jedes Nichtterminal eine Prozedur ein, die ggf. über eine Zufallszahl eine ihrer Regeln auswählt und die entsprechenden Zeichen in eine globale String-Variable (unser Bergpanorama) schreibt. Das Panorama kann dann mit Hilfe unserer Prozedur aus Teil (b) oder (c) ausgegeben werden. Hinweis: Achten Sie darauf, dass der Ableitungsprozess irgendwann enden muss, d.h., die Wahrscheinlichkeit für die Regeln ohne Nichtterminalsymbole muss ausreichend groß sein.
