Prof. G. Zachmann
Daniel Mohr

TU Clausthal
Institut für Informatik
21. April 2011

Sommersemester 2011

Übungen zu Informatik II - Blatt 2

Abgabe am 26.04

Organisatorisches

- Die theoretischen Aufgaben geben Sie **ausnahmsweise** am Dienstag bei Daniel Mohr in Raum 218, Julius-Albert-Str. 4 ab.
- Die Programmieraufgaben müssen Sie dienstags bis spätestens 15:00 Uhr an Ihren Tutor per Email (**christian.schnarr@tu-clausthal.de**) schicken.
- Die Programmieraufgaben müssen von Ihnen in der Übung vorgeführt und erklärt werden.

Aufgabe 1 (Typsysteme und Ausdruck-Bäume, *2 Punkte*)

Zeichnen Sie einen Ausdruck-Baum (Folie 27 im Kapitel Typsysteme) inklusive der notwendigen Typ-Konvertierungen für den Ausdruck `(1.0+1)*(1/2+1.0)`.

Hinweise:
Die Knoten bei einem Audruck-Baum können Operanden wie `1` und `2` oder Operatoren wie `+` und `*` sein. Operanden sind Blattknoten. Operatoren enthalten Verweise zu ihren Operanden. (Alle diese Operatoren sind binär, haben also genau zwei Operanden.) Es ist darauf zu achten, dass die Operatoren nur Operanden gleichen Types bearbeiten können.

Die Promotion-Hierarchie ist nach dem C++-Standard wie folgt definiert:
`bool` → `char` → `int` → `unsigned int` → `long int` → `float` → `double` → `long double`

In einem Ausdruck werden die Blätter mit den Typen der Operanden annotiert. Danach werden die Typen im Baum nach oben propagiert, wobei eine geeignete Promotion eingefügt werden muß, falls die beiden Operanden eines Operators nicht vom selben Typ sind.

Aufgabe 2 (Pi-Berechnung, *6 Punkte*)

a) Ein Zufallszahlengenerator (Gleichverteilung) gebe Paare von Zufallszahlen $x \in [0,1) \times [0,1)$ aus. Sei

$$A := \left\{(x,y) \in [0,1) \times [0,1) \mid x^2 + y^2 \leq 1\right\}$$

Ein Ereignis (d.h. Generieren einer Zufallszahl x) wird als Erfolg gewertet gdw $x \in A$. Geben Sie an, inwiefern sich durch relative Häufigkeiten des "Erfolgs" ein Vefahren zur näherungsweisen Berechnung von π definieren läßt. Implementieren Sie das Verfahren in Python. Solch ein Verfahren nennt man Monte-Carlo-Verfahren.

b) Der Mathematiker Leibniz fand heraus, dass sich die Kreiszahl π durch folgende unendliche Reihe berechnen lässt:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} - ...$$

Schreiben Sie ein Programm `leibniz.py`, welches nach diesem Verfahren π berechnet. Das Ergebnis nähert sich π mit zunehmender Summenlänge immer weiter an. Beim Start soll dem Programm die Anzahl der Summanden n per Kommandozeile übergeben werden, damit das Programm nicht endlos läuft.

Tip: Ein ähnliches Problem wurde in den Übungen behandelt (siehe Python Basics Folie 67).

c) Der Mathematiker John Wallis fand eine andere Methode zur Berechnung von π:

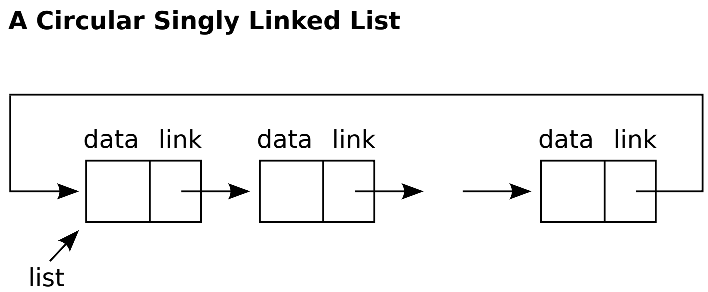
$$\frac{\pi}{2} = \frac{2}{1} * \frac{2}{3} * \frac{4}{3} * \frac{4}{5} * \frac{6}{5} * \frac{6}{7} * \frac{8}{7} * \frac{8}{9} * ...$$

Schreiben Sie ein Programm `wallis.py`, welches π nach diesem Verfahren berechnet. Das Ergebnis nähert sich π wiederum mit zunehmender Produktlänge weiter an. Berechnen Sie, wieviele Iterationen m benötigt werden, um die gleiche Genauigkeit (bis auf eine Abweichung von maximal 10^{-7}) wie das Verfahren von Leibniz aus Aufgabenteil b) bei n Iterationen zu erreichen. Ihre Kommandozeileneingabe für diesen Algorithmus ist also die Zahl n an Iterationen für den Leibnizalgorithmus. Die Ausgabe ist die Anzahl m an Iterationen, die der Algorithmus von Wallis für die gleiche Genauigkeit benötigt.

Aufgabe 3 (Zyklische Listen, *3+4 Punkte*)

a) In der Vorlesung wurden bereits verkettete lineare Listen vorgestellt. Statt linearer Listen kann man auch zyklische Listen betrachten, die folgende Eigenschaften haben:

Wie lineare Listen besitzt jedes Listenelement eine Vorwärtsreferenz. Bei einer zyklischen Liste erreicht man allerdings von jedem Listenelement aus nach endlich vielen Schritten wieder das Listenelement selbst.



Implementieren Sie eine zyklische Listen-Klasse, die Operationen zum Einfügen, Löschen und Ausgeben eines Elements besitzen soll.

b) Um der Sklaverei zu entgehen, vereinbarten die 40 Belagerten von Masada, sich im Kreis aufzustellen. Dann sollte jeder Siebte getötet werden, bis nur noch einer übrig bleibt, der dann Selbstmord begehen sollte.

Diese Begebenheit wurde von dem Geschichtsschreiber Flavius Josephus berichtet. Daher hat das sogenannte Josephus-Problem seinen Namen, bei dem es darum geht, sich so aufzustellen, daß man als letzter übrig bleibt. Implementieren Sie mit Hilfe der zyklischen Listen aus Aufgabenteil a) ein Programm zur Simulation des Josephus-Problems. Dem Programm soll die Anzahl der beteiligten Personen und die Schrittweite per Kommandozeile übergeben werden.

Beispiel: Bei 9 Personen und Schrittweite 5 ist die Hinrichtungsreihenfolge:

5 1 7 4 3 6 9 2 8.