Praktische Informatik 3 WS 08/09

Bonus-Übungsblatt

Ausgabe: 12.01.09
Abgabe: 02.02.09

Christoph Lüth
Dominik Dietrich
Marcus Ermler
Klaus Hartke
Christian Maeder
Dominik Luecke
Ewaryst Schulz

14 *Excel zum zweiten* *15 Punkte*

Diese Aufgabe ist die lang erwartete Fortsetzung der Aufgabe 8 aus dem 3. Übungsblatt; hier werden wir eine Tabellenkalkulation implementieren.

Abstrakt besteht eine Tabelle aus einer zweidimensional indizierten Menge von *Zellen*. Jede Zelle enthält entweder eine Formel oder eine Zeichenkette:

```
data Cell  = CellString String
           | CellFormula Expr

type Sheet = Array (Int, Int) Cell
```

Der Typ `Array` stammt aus dem Modul `Array` der Haskell98-Standardbücherei, die auch verschiedene Funktionen zur Manipulation von Feldern implementiert, insbesondere die folgenden:

```
array  :: (Ix a) => (a,a) -> [(a,b)] -> Array a b
(!)    :: (Ix a) => Array a b -> a -> b
(//)   :: (Ix a) => Array a b -> [(a,b)] -> Array a b
bounds :: (Ix a) => Array a b -> (a,a)
```

Die Typklasse `Ix` bezeichnet mögliche Indizierungen, insbesondere sind Tupel von `Int` eine Instanz von `Ix`; `bounds` gibt die für dieses Feld gültigen Feldgrenzen zurück (welche genau den bei der Konstruktion mit `array` im ersten Argument angegebenen entsprechen).

Implementieren Sie zuerst eine Funktion

```
data Result a = Error String | Ok a
eval :: Sheet -> Expr-> Result Double
```

welche einen Ausdruck (also eine Formel) in dem Kontext einer Tabelle auswertet. Hierbei wertet z.B. die Variable `c10` zu dem Inhalt der Zelle (2, 9)[1] aus. Die Auswertung ist rekursiv; falls das zu Nichttermination führen würde (weil die Auswertung einer Zelle direkt oder indirekt von ihrem eigenen Wert abhängt), soll ein Fehler zurückgegeben werden, d.h. die Auswertung soll immer terminieren. Die Auswertung einer Zeichenkette als Zahl führt auch zu einem Fehler.

Die Tabellenverwaltung besteht in jedem Fall aus zwei Komponenten:

- Dem Kernmodul, (*7 Punkte*)
- und der Benutzerschnittstelle. (*8 Punkte*)

[1] Der Index im Feld startet mit 0, der in der Tabelle mit 1.

Das Kernmodul soll folgende Funktionen implementieren:

```
new       :: (Int, Int)-> Sheet
updNumber :: (Int, Int)-> Double-> Sheet-> Sheet
updString :: (Int, Int)-> String-> Sheet-> Sheet
updExpr   :: (Int, Int)-> Expr-> Sheet-> Sheet
cellCont  :: Sheet-> (Int, Int)-> Cell
cellShow  :: Sheet-> (Int, Int)-> String
```

Die Funktion `cellShow` soll den Wert einer Zelle als String anzeigen, während `cellCont` den Inhalt zurückgibt.

Die Benutzerschnittstelle wird beim Aufruf des Programmes aus der Kommandozeile gestartet, und gibt dem Benutzer Zugriff auf die Funktionalität des Kernmoduls. Sie arbeitet in einer Schleife wie folgt: es wird eine Eingabeaufforderung ("prompt") ausgegeben, eine Kommandozeile von der Konsole eingelesen, verarbeitet, und das Ergebnis auf der Konsole ausgegeben; danach erfolgt eine neue Eingabeaufforderung. Es sollen folgende Eingaben in folgender Syntax akzeptiert werden:

- Mit `n` oder `new` gefolgt von einem Paar von ganzen Zahlen (z.B. `(3, 10)`), das mit `readIO` eingelesen werden kann, wird eine leere Tabelle der angegebenen Größe erzeugt.
- Eine Eingabe der Form `<Variable> = <Expr>` oder `<Variable> = <String>` setzt den Inhalt einer Zelle (Zeichenketten stehen in Anführungszeichen). Nach einer solchen Eingabe wird die gesamte Tabelle formatiert ausgegeben.
- Mit `d` oder `display` wird die gesamte Tabelle formatiert ausgegeben.
- Die Eingabe von einer einzelnen Variable (z.B. `a13`) zeigt nur den Inhalt der entsprechenden Zelle an.
- Die Eingabe von `s` oder `save` gefolgt von einem Dateinamen (ohne Anführungszeichen) schreibt den Inhalt der Tabelle in die angegebene Datei mit einer Sicherheitsabfrage, falls die Datei schon existiert.
- Mit `o` oder `open` gefolgt von von einem Dateinamen (ohne Anführungszeichen) kann man eine Tabelle aus einer Datei einlesen. Falls die momentan bearbeitete Datei ungesicherte Änderungen enthält, muss der Benutzer die Aktion gesondert bestätigen.
- Ein `q` oder `quit` beenden das Programm. Falls die momentan bearbeitete Datei ungesicherte Änderungen enthält, muss der Benutzer die Aktion gesondert bestätigen.
- Ein `?`, `h`, `help` oder ein Leerzeile zeigen einen Hilfetext an.

Die Argumente von `n`, `s` und `o` sind von dem Kommando durch eine oder mehrere Leerzeichen getrennt.

Hinweis: Zur bequemen Eingabe können Sie die Funktion `readline` aus `System.Console.Readline` verwenden (nicht unter allen Betriebssystemen verfügbar).

15 *Excel zum Besseren* *5 Punkte*

Ein Problem der Implementation der vorherigen Aufgabe ist die Effizienz, insbesondere dass bei einer Änderung an der Tabelle alle Werte neu ausgerechnet werden.

Um das zu verhindern, müssen wir explizit die Abhängigkeiten zwischen einzelnen Zellen berücksichtigen. Wir sagen, dass eine Zelle (i_1, i_2) von einer anderen Zelle (j_1, j_2) *abhängig* ist, wenn eine Änderung des Inhaltes der Zelle (j_1, j_2) eine Neuberechnung des Wertes der Zelle (i_1, i_2) nach sich zieht. Wenn also beispielsweise die Zelle `b3` die Formel `sum(a1:a3)` enthält, dann ist die Zelle `b3` genau von den Zellen `a1`, `a2` und `a3` abhängig. Daraus folgt, dass nur Zellen, die eine Formel enthalten, von anderen abhängig sind.

Implementieren Sie zuerst eine Funktion

```
dependencies :: Expr-> Set (Int, Int)
```

die für eine Formel diejenigen Zellen berechnet, von dem der Wert dieser Formel abhängt.

Erweitern Sie die Datentstruktur `Cell` um einen Eintrag, der für jede Zelle die von diesen Zellen abhängigen Zellen enthält; in unserem Beispiel oben müsste also in den Zellen `a1`, `a2`, `a3` jeweils die Zelle `b3` vermerkt sein. Für Zellen, die eine Formel enthalten, muss zudem der zuletzt berechnete Wert vermerkt werden.

Hiermit läßt sich jetzt die Änderung der Tabelle optimieren, weil bei einer Änderung nicht mehr die gesamte Tabelle neu berechnet werden muss. Wird eine Zelle geändert, muss folgendes neu berechnet werden:

- der Wert aller von dieser Zelle abhängigen Zellen (und das gegebenfalls rekursiv),
- wenn die Änderung eine Formel neu erzeugt, ändert, oder löscht (durch einen Wert/Zeichenkette ersetzt), die Abhängigkeiten: beim löschen und ändern müssen zuerst die alten Abhängigkeiten ausgetragen werden, beim neu eintragen und ändern die neuen Abhängigkeiten eingetragen werden.

Implementieren Sie diese optimierte Änderung in Ihrer Tabellenkalkulation.