Dr. Andreas Borchert
Norbert Heidenbluth

Abteilung Angewandte Informationsverarbeitung

Allgemeine Informatik II für Mathematiker/Wirtschaftsmathematiker (SS 2005)

Abgabetermin: 02. Juni 2005

Aufgabe 7: PlotRPN - Ein Funktionsplotter (20 Punkte)

Neulich hat Sie Ihre kleine Schwester mal wieder um ein wenig Hilfe bei den Mathematik-Hausaufgaben gebeten (8. Klasse, Thema: "Funktionen und ihre Graphen"). Als hochmotivierte(r) Student(in) hatten Sie zum Helfen aber leider keine Zeit. Dummerweise war Ihr grafischer Taschenrechner auch gerade kaputt, und seit der jüngsten Viren-Attacke läuft Ihr Windows-PC (der mit Maple drauf) auch nicht mehr. Au weia, wie kann Ihre Schwester nun überprüfen, ob die Hausaufgaben richtig gelöst (gezeichnet) sind?

Wie immer im Moment der größten Verzweiflung hilft Ihnen das neueste Informatik-Übungsblatt aus der Klemme. Denn – um gleich mal mit der Tür ins Haus zu fallen – in dieser Woche lautet Ihre Aufgabe (in der Kurzfassung):

> "Erweitern Sie den Taschenrechner aus der Aufgabe 5 so, daß er in der Lage ist, Funktionen mit einer Unbekannten (x) zu plotten!"

Dazu sollen Sie als Argument die zu plottende Funktion in der umgekehrten polnischen Notation angeben. Optional stehen Flags zur Verfügung, mit denen die Intervallgrenzen für den zu plottenden Bereich (Flag `-i`) bzw. die Schrittweite (Flag `-d`) angegeben werden können. Mithin sollte die Usage-Meldung wie folgt aussehen:

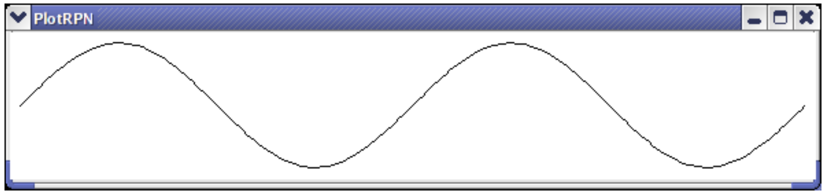
```
Usage: ./PlotRPN [-d delta] [-i a b] RPN expression
```

Beispiel:

Die Eingabe

```
theseus$ PlotRPN -i '0 pi 2 * -' 'pi 2 *' 'x sin'
```

würde dann zu der folgenden grafischen Darstellung führen:



(d.h. zur Darstellung des Graphen von $\sin(x)$ auf dem Intervall $[-2\pi, 2\pi]$).

Konkrete Aufgabenstellung:

Okay, zur Beruhigung: Das war nur die verkürzte Fassung der Aufgabenstellung! Tatsächlich ist für dieses Übungsblatt folgendes angedacht:

> "Gegegeben seien die Schnittstellen *Functions.od* und *FunctionPlotter.od* mit den weiter unten beschriebenen Spezifikationen.
>
> Suchen Sie sich für dieses Übungsblatt nun **genau eine** der beiden Schnittstellen heraus und implementieren Sie diese!"

Das heißt, Sie bekommen für dieses Übungsblatt die volle Punktzahl (20 Punkte), wenn Sie **entweder** `Functions` **oder** `FunctionPlotter` implementieren (15 Punkte) sowie das notwendige Hauptprogramm schreiben (5 Punkte), das zur Ausführung verwendet werden muß!

Da Ihr Programm mit nur einer implementierten Schnittstelle nicht lauffähig sein wird, ist für dieses Übungsblatt explizit der Austausch mit anderen Gruppen erwünscht: suchen Sie sich eine Gruppe, welche die jeweils andere Schnittstelle implementiert, und tauschen Sie zum Testen Ihre Lösungen untereinander aus. Bei der Suche nach einer entsprechenden "Partner-Gruppe" ist Ihnen einerseits Ihr Tutor behilflich. Ggf. werden wir auch eine Webseite anbieten, auf der wir eine Übersicht geben, welche Gruppe welches Modul fertiggestellt hat (siehe hierzu auch die Hinweise am Ende des Blattes).

Nun aber zu den

Spezifikationen

Das Modul `Functions.od`:

Dieses Modul ist zur Verwaltung und Verwendung eines Ausdrucks mit einer Veränderlichen in der umgekehrten polnischen Notation (*Reversed Polish Notation (RPN)*) gedacht. Entsprechend kommt im Vergleich zur Notation aus Aufgabe 5 noch "x" als zusätzlicher Operand hinzu. Beispiel: "x sin" entspricht der Sinus-Funktion $sin(x)$.

- Die Funktion `Create`
  Diese Funktion bekommt zwei Parameter übergeben, von denen einer vom Typ `Streams.Stream` ist und einen Ausdruck in der umgekehrten polnischen Notation enthält. Dieser Ausdruck wird von `Create` in die als weiterer Parameter (vom selbst zu implementierenden Typ `FunctionRec`) übergebene Referenz-Variable `f` überführt. Abhängig davon, ob der übergebene Ausdruck gültig war oder nicht wird `TRUE` bzw. `FALSE` zurückgeliefert.

- Die Funktion `Evaluate`
  Auch diese Funktion bekommt zwei Parameter übergeben: einerseits die Funktion in Form des Datentyps `Function`, andererseits eine REAL-Zahl x. Der Rückgabewert ist die Wert der übergebenen Funktion an der Stelle x.

Bei der Implementierung dieses Moduls ist zu beachten, daß `Create` den Ausdruck in eine lineare Liste von Tokens konvertiert, die dann später von `Evaluate` verwendet wird. Entsprechend vereinfacht sich der Aufwand erheblich, da jeder Ausdruck nur ein einziges Mal parsiert werden muß – selbst, wenn sehr viele Funktionswerte anschließend bestimmt werden. Die lineare Liste mit den Tokens darf dabei keinesfalls in globalen Variablen untergebracht werden. Stattdessen können Sie innerhalb von `Functions.om` weitere Felder im Record-Typ `FunctionRec` deklarieren, die auf die lineare Liste verweisen. Record-Felder, die nur innerhalb des `MODULE` genannt werden, jedoch nicht im zugehörigen `DEFINITION` sind sogenannte $p$rivate Felder.

Das Modul `FunctionPlotter.od`:

Dieses Modul hat nur eine einzige Methode, nämlich `Plot`. Diese bekommt die folgenden Parameter übergeben:

- f: enthält die zu plottende Funktion
- plotter: Objekt der Schnittstelle `Plotters`
- a: Beginn des Intervalls, auf dem der Funktiongraph gezeichnet werden soll
- b: Ende des Intervalls, auf dem der Funktiongraph gezeichnet werden soll
- delta: Schrittweite für die Berechnung der Werte von x

Da `Plot` keine Angaben über den zu plottenden Wertebereich $[y_{\min}, y_{\max}]$ auf der y-Achse erhält, muß dieser aus den berechneten Funktionswerten selbst ermittelt werden. Bei der Abbildung des zu plottenden Rechtecks $[a, b] \times [y_{\min}, y_{\max}]$ auf das Koordinatensystem des Plotters ist zu beachten, daß der gleiche Maßstab auf der x- und der y-Achse verwendet und gleichzeitig möglichst wenig Raum verschenkt wird. Falls es vom Plotter unterstützt wird, ist es zulässig, dessen Plot-Bereich unter Verwendung von `Plotters.SetSpace` passend zu verändern.

Bemerkungen und Hinweise

- Sie brauchen lediglich den eigentlichen Funktionsgraphen plotten zu lassen. Es ist ausdrücklich *nicht* gefordert, Koordinatenachsen, Beschriftungen oder ähnliche Dinge zu plotten.

- Das Erzeugen eines Objekts vom Typ `Plotters.Plotter` und die Ausgabe des zu plottenden Inhalts erfolgt nahezu analog zu den Beispiel im Zusammenhang mit den Turtle-Graphics (siehe dortige Bespiele).

- Zum Plotten der Funktionsgraphen empfiehlt sich insbesondere die Verwendung der folgenden Prozeduren:

  - `Plotters.GetSpace`
  - `Plotters.SetSpace` (optional)
  - `Plotters.Move`
  - `Plotters.Cont`

  Zur Erklärung dieser Prozeduren: siehe *man Plotters* sowie das Beispiel aus den Übungen!

- Solange Ihnen die Implementierung der "anderen" Schnittstelle nicht zur Verfügung steht, können und sollten Sie eine sogenannte "Dummy-Implementierung" verwenden. Wenn Sie zum Beispiel das Modul `FunctionPlot` implementieren möchten, so können Sie das Modul `Functions` ganz minimal gestalten, so daß es Ihnen zu Testzwecken stets die Sinus-Funktion in der gewünschten Form zurückliefert ("hart verdrahten").

Viel Erfolg!