Übungsblatt 17 zu "Programmiersprachen"

Berthold Hoffmann, Studiengang Informatik (hof@informatik.uni-bremen.de)

Besprechung am 21. Juni 2010

Funktions- und Typ-Parameter in Ada

In einer funktionalen Sprache wie Haskell kann eine allgemeiene Sortierfunktion so hingeschrieben werden:

```
sort :: (a → a → Bool) → [a] → [a]
sort rel l = ...
```

Diese Funktion sortiert Listen beliebiger Elementtypen mit einer ebenfalls angegebenen Vergleichsfunktion precedes. Aufrufe dieser Funktion sehen dann so aus:

```
sort (≤) [3,1,2,5,4,6]
```

In Ada ist das alles nicht so einfach: Weder kann man polymorphe Typen verwenden, noch kann man Funktionen einfach als Parameter an Funktionen oder Prozeduren übergeben. Eine allgemeine Prozedur zum Sortieren muss als *generische Proizedur* vereinbart werden:

generic
  type Item **is private**;
with
  function precedes (x, y: Item) returns Boolean;
procedure sort (x, y: **in out array range** <> **of** Item);

Der Rumpf der Prozedur wird dann so vereinbart:

procedure sort (x, y: **in out array range** <> **of** Item) **is**
...

Die Prozedur muss erst instanziiert werden, bevor sie benutzt werden kann:

procedure ascendingSort **is new** sort(Integer,"<=");

Erst danach kann die Instanz wie eine normale Prozedur benutzt werden:

a: **array** (. . .) **of** Integer;
...
ascendingSort(a);

Fragen

1. Finden Sie es angemessen, Funktionsparameter so umständlich zu realisieren?
2. Weshalb (nicht) ?
3. Können Sie sich vorstellen, weshalb dies so definiert wurde?