Stack–basierte High–Level Programmiersprachen

Ulrich Hoffmann <uho@xlerb.de>

Zusammenfassung

Traditionelle Programmiersprachen orientieren sich in ihrer Notation zumeist an der üblichen mathematischen Schreibweise und verwenden für arithmetische Ausdrücke eine Infix– und für Funktionsanwendungen eine Präfix–Notation. Zur Programmausführung wird häufig ein Laufzeitkeller (Runtime–Stack) eingesetzt, der Bestandteil des Laufzeit–Systems ist und der auf Programmiersprach–Ebene selbst nicht direkt in Erscheinung tritt.

Stack–basierte Programmiersprachen schlagen einen anderen Weg ein. Sie stellen einen Stack in das Zentrum ihrer Programmausführung, der ausdrücklich und unmittelbar von Programmen zur Ausdrucksauswertung und für Funktionsaufrufe verwendet wird. Ausdrücke und Funktionsaufrufe werden dabei typischer Weise in Postfix–Notation formuliert. Zur Übergabe werden aktuelle Parameter in den obersten Stack–Elementen berechnet. Die aufgerufene Funktion konsumiert ihre formalen Parameter vom Stack und hinterlässt auch dort ihr Funktionsergebnis. Dies kann zu Programmen ohne benannte Variable führen. Die Postfix–Notation führt zur interessanten Eigenschaft der Konkatenativität (concatenativity): Unter bestimmte Eigenschaften führt das syntaktische Konkatenieren von Teil–Programmen und –Ausdrücken zur semantischen sequentiellen Komposition der Teile. Auch können solche Programme häufig auf rein syntaktischer Ebene refaktorisiert werden.

High–Level Programmiersprachen zeichnen sich durch die Verwendung von nicht–trivialen Daten– und Kontrollstrukturen aus. Auch setzen sie möglicher Weise Funktionen höherer Ordnung ein.

Die vorliegende Arbeit stellt eine Reihe solcher Stack–basierter High–Level Programmiersprachen vor und erläutert ihre Eigenschaften, Gemeinsamkeiten und Unterschiede im Detail.