IML Spracherweiterung: Datentyp String

Pascal Büttiker
Samuel Stachelski

Abstract
Unsere Erweiterung führt den neuen Datentyp "string" ein. Strings sollen ASCII-Zeichen speichern können. Ein String muss zwar intern in einer From von einem Array gespeichert werden, aber da ist nicht relevant für die Anwendung der Spracherweiterung: Ein String ist ein homogener Datentyp, der nicht auf anderen Datentypen von IML basieren soll. Dementsprechend braucht er z. B. auch eigene Operatoren. Ausserdem soll es möglich sein, eine Variable eines beliebigen Typs in einen String einzubauen, mittels Inline-Syntax.

Grundsätzliche Idee
Mit Texten, bzw. Zeichenketten zu arbeiten ist oft notwendig oder hilfreich wenn man Programmiert. In IML gibt es bisher keinen Datentyp für Zeichenketten. Wir möchten deshalb einen Datentyp „String“ einführen. Einen String soll man mit anderen Variablen beliebigens Typs zusammenhängen können, wobei daraus wieder ein String entsteht.

Lexikalischer Syntax
Da der String-Typ genau wie ein anderer Grunddatentyp behandelt wird, braucht es keine lexikalischen Ergänzungen.

Grammatikalischer Syntax
Deklaration:

```
var myString1 : string;
const myString2 : string;
var myString3 : string;
var number1: int32;
```

Initialisierung:
Statischer String

```
myString1 init := "schnurzel ";
```

Mit anderen Variablen

```
myString2 init := myString1 + number1;
```

Im Funktionsaufruf

```
call foo(myString3 init);
```

Ein-/Ausgabe:

```
? myString1;
! myString1;
```

Konkatenation:
myString1 := "dum" + "di";
myString1 := "foo" + number1;
myString1 := myString1 + myString1;

Inline-Syntax:
Das Zeichen % wird als Markierung für den Inline-Syntax verwendet. Zwischen zwei %-Zeichen muss sich ein Identifier befinden. Um das Zeichen "%" selber auszugeben, schreibt man zwei mal hintereinander %
myString1 := "Hello %myString1%!";
myString1 := "Hello %number1%!";
myString1 := "Onehundred%%";

Da der String-Typ genau wie ein anderer Grunddatentyp behandelt wird, braucht es keine grammatikalischen Ergänzungen in der Sprachdefinition.

Kontext- und Typ-Einschränkung
Es gibt für alle Datentypen die Möglichkeit, eine Variable zum Datentyp String zu konvertieren. Anwendung findet diese Konvertierung nur in Expressions und nur implizit. Ansonsten würden die Freiheiten z. B. bei den Prozedur- und Funktionsaufrufen eingeschränkt mit dem Typ String (z. B. ist dann ein Aufruf mir Referenzparametern nicht mehr klar definiert). Wenn in einer Expression irgendwo der Datentyp String vorkommt, wird die ganze Expression am Schluss ein String sein.

Zeichensatz
Es wird der ASCII-Zeichensatz unterstützt, wobei nicht-druckbare Zeichen ignoriert werden.

Maximale Länge
Die maximale Länge wird provisorisch auf 256 Zeichen festgelegt. Zu lange Strings werden (wie bei einem int-Overflow) einfach abgeschnitten.

Speicherverbrauch
Pro String-Variable wird ein von Anfang die maximale Länge eines Strings reserviert. Dies verbraucht zwar Platz, erleichtert die Handhabung der Speicherverwaltung dafür extrem.

Vergleich mit anderen Programmiersprachen

Kriterium	Java	C
Realisierung des Datentyps String	Klasse, welche auf dem Grunddatentyp char-Array aufbaut	Typ String existiert nicht, in den Standard-Bibliotheken wird String als ein char-Array betrachtet, wobei der Wert 0 das Stringende kennzeichnet

Operationen auf Strings	Syntax ähnlich wie bei uns: Konkatenation mittels +-Operator, dabei implizite Konvertierung von anderen Datentypen Sehr viele Funktionen in der Klasse String	Standard-Bibliothek mit diversen Funktionen für char-Arrays
Inline-Syntax	Nicht unterstützt	Nicht unterstützt
Maximale Länge	Länge des verfügbaren Arbeitsspeicher dividiert durch 2 (da UTF-16) oder 2^31 Zeichen	Länge des verfügbaren Speichers
Speicherverbrauch	2 Bytes pro Zeichen + Overhead der Klasse	1 Byte pro Zeichen

6. Begründung des Entwurfs

Datenstruktur:

Für uns war es essentiell, dass String ein elementarer Datentyp ist, also nicht auf anderen Konstrukten der Sprache basiert. In den meisten modernen Programmiersprachen verwendet man eine Kombination von Characters und Arrays, um Strings zu repräsentieren. Wir finden es deshalb viel interessanter, einen homogenen Datentyp String einzuführen der vollkommen in die Sprache integriert ist und (je nach Komplexität des Typs) keine Änderungen an der Sprachdefinition erfordert, bzw. nicht mehr als beispielsweise die Einführung eine numerischen Datentyps.

Funktionalität

Unser Ziel ist es, dem Programmierer zu ermöglichen Nachrichten zu erstellen und Auszugeben. Ausserdem soll er beliebige Variablen in Strings um wandeln und aneinander hängen können. Modifikationen von Strings sind kein Ziel, deshalb beschränken wir uns auf das Implizite Casting, den schon vorhandenen +Operator und die Inline-Syntax.