

XLang

Max Heidinger, Pascal Riesinger

April 6, 2019

1 Einleitung

XLang ist eine Programmiersprache entwickelt im Rahmen der Vorlesung *Compilerbau* an der Dualen Hochschule Baden-Württemberg Karlsruhe.

2 Syntax

2.1 Grammatik

Die folgende Grammatik in der Backus-Naur-Form beschreibt die Syntax von XLang. Die Regeln folgen diesen Konventionen:

- Der Aufbau ist `Symbol ::= Ersetzungen.`, jede Regel wird also mit einem Punkt beendet.
- Terminalsymbole werden zwischen Hochkommata gesetzt, alle anderen Symbole sind Nichtterminale.
- Ausnahmen zu obiger Regel sind das “Pipe”-Symbol, runde Klammern und der Punkt. Wie vorher beschrieben beendet der Punkt eine Regel. Das Pipe-Symbol bezeichnet alternative Ersetzungen. Runde Klammern werden zur Gruppierung verwendet, um die Lesbarkeit zu verbessern.
- Leerzeichen zwischen Nichtterminalen, sowie Terminalen und Nichtterminalen sind zu ignorieren. Falls Leerzeichen explizit notwendig sind, ist dies durch ein Terminal der Form `' '` gekennzeichnet.

```
Programm ::= NamenTeil '\n'
           TeilSeparator '\n'
           VariablenTeil '\n'
           TeilSeparator '\n'
           CodeTeil.
```

```
TeilSeparator ::= '=' TeilSeparatorZeichen.
```

```
TeilSeparatorZeichen ::= '=' | ('=' TeilSeparatorZeichen).
```

```
NamenTeil ::= Zeichen | (Zeichen NamenTeil).
```

```
VariablenTeil ::= Variable | (Variable '\n' VariablenTeil).
```

```
Variable ::= (ImportExportFlag ' ' VariablenName ' ' VariablenTyp)
           | (VariablenName ' ' VariablenTyp).
```

```
ImportExportFlag ::= ImportFlag | ExportFlag.
```

```
ImportFlag ::= '->'.
```

```
ExportFlag ::= '<-'.
```

```
VariablenName ::= Buchstabe | Buchstabe Bezeichner.
```

```
VariablenTyp ::= 'int' | 'float' | 'string'.
```

```
CodeTeil ::= Instruktion | Instruktion CodeTeil.
```

```
Instruktion ::= (Zuweisung '\n') | Verzweigung | (Kommentar '\n').
```

```
Kommentar ::= '//' | '//' Zeichenfolge.
```

```

Zuweisung ::= VariablenName '=' EvaluationsWert.
EvaluationsWert ::= Wert
                    | Addition
                    | Subtraktion
                    | Multiplikation
                    | Division.
Wert ::= VariablenName | Literal.
Literal ::= String | Int | Float.
Addition ::= EvaluationsWert '+' EvaluationsWert.
Subtraktion ::= EvaluationsWert '-' EvaluationsWert.
Multiplikation ::= EvaluationsWert '*' EvaluationsWert.
Division ::= EvaluationsWert '/' EvaluationsWert.

Verzweigung ::= IfVerzweigung | WhileVerzweigung | ForVerzweigung.
InstruktionsBlock ::= '{' InstruktionsListe '}'.
InstruktionsListe ::= Instruktion | Instruktion InstruktionsListe.

IfVerzweigung ::= ('if' Vergleich InstruktionsBlock)
                  | ('if' ' ' Vergleich InstruktionsBlock ElseVerzweigung)
                  | ('if' ' ' Vergleich InstruktionsBlock ElseIfVerzweigungen)
                  | ('if' ' ' Vergleich InstruktionsBlock
                     ElseIfVerzweigungen ElseVerzweigung).
Vergleich ::= GleichheitsVergleich
              | UngleichheitsVergleich
              | KleinerVergleich
              | GroesserVergleich.
GleichheitsVergleich ::= EvaluationsWert '==' EvaluationsWert.
UngleichheitsVergleich ::= EvaluationsWert '!=' EvaluationsWert.
KleinerVergleich ::= EvaluationsWert '<' EvaluationsWert.
GroesserVergleich ::= EvaluationsWert '>' EvaluationsWert.
ElseIfVerzweigung ::= 'else if' Vergleich InstruktionsBlock.
ElseIfVerzweigungen ::= ElseIfVerzweigung
                       | ElseIfVerzweigung ElseIfVerzweigungen.
ElseVerzweigung ::= 'else' InstruktionsBlock.

WhileVerzweigung ::= 'while' Vergleich InstruktionsBlock.
ForVerzweigung ::= 'for' Zuweisung ';' Vergleich ';' Zuweisung InstruktionsBlock.

Zeichen ::= Buchstabe | Ziffer
Bezeichner ::= Zeichen | (Zeichen Bezeichner).
AlleZeichen ::= Zeichen | Sonderzeichen.
Zeichenfolge ::= AlleZeichen | (AlleZeichen Zeichenfolge).
Sonderzeichen ::= ' ' | '!' | ... | '@'.
Buchstabe ::= 'A'|...| 'Z'| 'a'|...| 'z'.
Ziffer ::= '0'|...| '9'.
String ::= '''
           | ''' Zeichenfolge '''
           | """ Zeichenfolge """
Int ::= Ziffer | Ziffer Int.
Float ::= Int '.' Int.

```

2.2 Beschreibung

2.2.1 Struktur

Ein XLang Programm ist in 3 Teile aufgeteilt. Diese werden durch eine Folge von Gleichheitszeichen = voneinander abgetrennt, welche mindestens 3 Zeichen lang sein muss. Zunächst wird der Name des Programmes vermerkt. Anschließend folgt der Variablendeklarationsteil, in welchem alle im Programmteil verwendeten Variablen deklariert werden müssen. Der dritte Teil des Programmes ist der sogenannte Programmteil, welcher alle Instruktionen beinhaltet.

Leerzeichen und Kommentare können überall im Programm eingefügt werden. Zeilenenden werden durch ein einzelnes Linefeed-Zeichen (ASCII 0x0A) markiert. Diese Zeilenenden werden auch unter UNIX-ähnlichen Systemen verwendet.

2.2.2 Variablen

Es gibt in XLang drei Datentypen:

- Ganzzahlen, deklariert durch den Typ `int`.
- Gleitkommazahlen, deklariert durch den Typ `float`.
- Zeichenfolgen, deklariert durch den Typ `string`.

Wie bereits angemerkt müssen alle im Programm verwendeten Variablen im Variablenteil deklariert werden. Da XLang ohne Funktionen und Unterprogramme auskommt, müssen Variablen, welche als Eingabeparameter fungieren sollen mit einem sogenannten `ImportFlag` gekennzeichnet werden. Eine Deklaration einer Eingabevariable sieht dann beispielsweise wie folgt aus: `-> name string`.

Die Eingabevariablen werden beim Programmstart aus der Kommandozeile ausgelesen. Dabei ist die Reihenfolge der Deklaration gleich der Reihenfolge der Übergabe.

Variablen, deren Wert am Ende des Programmes ausgegeben werden soll, müssen mit einem `ExportFlag` gekennzeichnet werden. Eine Deklaration einer Ausgabevariable sieht dann beispielsweise wie folgt aus: `<- ergebnis float`. Die Ausgabevariablen werden in der Reihenfolge ausgegeben, in welcher sie deklariert wurden.

Es gibt keine Möglichkeit, Variablen während der Ausführung auszugeben oder einzulesen.

Variablen können Literale, also konstante Werte oder Ausdrücke zugewiesen werden. Ein Ausdruck ist bei numerischen Werten entweder eine andere Variable oder eine mathematische Formel. Bei Rechnungen können die Grundrechenarten (Addition, Subtraktion, Multiplikation und Division) verwendet werden. Hierbei gilt die gleiche Priorisierung wie im C-Standard.

Bei Zeichenketten ist ein Ausdruck entweder eine andere Variable oder eine Verkettung von mehreren Literalen und Variablen über den Additionsoperator.

2.2.3 Konditionen

Die Konditionalausdrücke in XLang folgen dem Beispiel anderer Programmiersprachen. Wenn der Vergleich nach dem `if` zu einem logisch "wahren" Wert evaluiert, werden alle Anweisungen im folgenden Anweisungsblock ausgeführt. Der `if`-Anweisung können optional beliebig viele `else if`-Anweisungen folgen. Diese werden nur ausgewertet, wenn der Vergleich der voranstehenden Anweisung zu einem unwahren Wert evaluiert wurde. Abgeschlossen werden kann ein Konditionalausdruck durch ein `else`. Die Anweisungen im `else`-Block werden ausgeführt, wenn kein vorheriger Vergleich im Konditionalausdruck zu einem wahren Wert evaluiert werden konnte.

2.2.4 Kommentare

Es gibt in XLang nur Zeilenkommentare. Diese werden mit doppelten Schrägstrichen eingeleitet und können jegliche Zeichen enthalten. Kommentare können an jeder Stelle beginnen und enden mit dem Ende der Zeile. Kommentare werden vom Compiler ignoriert und haben keine Bedeutung für das Programm.

2.2.5 Schleifen

Es gibt in XLang zwei Arten von Schleifen. Eine **while**-Schleife führt dein Codeblock so lange aus, bis die Bedingung im Kopf der Schleife zu einem unwahren Wert evaluiert wird. While-Schleifen sind kopfgesteuert.

Die zweite-Art der Schleifen in XLang sind **for**-Schleifen. Diese erlauben es, eine Zuweisung im Fuß der Schleife durchzuführen. Zunächst wird eine Variable zugewiesen, welche dann in der Kondition in der Schleife verwendet wird. Der dritte Teil der Schleife weist der Variable einen neuen Wert zu. Vor jedem Durchlauf wird der Vergleich durchgeführt, dann der Programmblock, falls der Vergleich zu einem wahren Wert evaluiert, nach dem Ausführen des Programmblockes wird dann die Zuweisung ausgeführt.