Christoph Lüth <cxl>
Lutz Schröder <lschrode>

1. Übungsblatt

Ausgabe: 16.04.08
Abgabe: 14.05.08

Dieses Übungsblatt beschäftigt sich mit dem Parsieren von Elementen der Sprache C. Lernziele sind hierbei:

- Kenntnis der Syntax von Deklarationen;
- Kenntnis der Präzedenzen der Operatoren;
- Das Arbeiten mit Parsergeneratoren (`lex`, `yacc` und Freunde).

1 *Zur Auflockerung*

Um das Arbeiten mit `yacc` zu trainieren, entwickeln wir eine Grammatik und einen Parser, der einfache arithmetische Ausdrücke der folgenden Art parsiert:

```
Expr   ::= Expr '+' Term
         | Term
Term   ::= Term '*' Factor
         | Factor
Factor ::= '(' Expr ')'
         | Number
Number ::= ('0' ... '9')+
```

In den folgenden Teilen des Übungsblattes werden wir mit dieser Technik Deklarationen und Ausdrücke in C parsieren.

2 *Parsieren von Deklarationen*

Entwickeln Sie einen Parser für die Deklarationen (C-Standard, §6.7):

1. Definieren Sie einen Datentyp zur Repräsentation von Typen, beispielsweise wie folgt in Haskell:

```
data Type = BasicType BasicType
          | PointerType Type
          | StructType (Maybe Identifer) [Structfield]
          | ArrayType Type (Maybe Int)
          | ConstType Type

data Structfield = Structfield Identifier Type

data BasicType = IChar  CharQualifier
               | IInt   IntQualifier
               | INat   IntQualifier
               | IFloat FloatQualifier
               | IVoid

data CharQualifier = CQUnsigned
```

```
                    | CQPlain
                    | CQSigned

data IntQualifier = IQShort
                  | IQPlain
                  | IQLong
                  | IQLongLong

data FloatQualifier = FQFloat
                    | FQDouble
                    | FQLongDouble
```

2. Erweitern Sie diesen Datentyp, um Deklarationen zu repräsentieren.
3. Unter Benutzung der Grammatik aus dem C-Standard (Anhang A, Nichtterminalsymbol *declaration*) entwickeln Sie einen Parser, welcher Deklarationen liest, und in der oben gewählten Repräsentation zurückgibt.
4. Zum Testen ihres Parsers entwickeln Sie eine Funktion, welche die Repräsentation von Deklarationen wieder ausgibt; testen Sie den Parser an ausgewählten Beispielen.

Hinweise: Sie können annehmen, dass alle Präprozessor-Anweisunge aufgelöst werden. Für die Größe von Felder können Sie vereinfachend konstante ganze Zahlen annehmen. Typdefininitionen (`typedef`) brauchen nicht behandelt zu werden.

3 *Parsieren von Ausdrücken*

Hier entwickeln wir einen Parser, der Ausdrücke in C parsieren kann (Nichtterminalsymbol *expression* in Anhang A).

1. Entwickeln Sie einen Datentyp, der Ausdrücke repräsentiert. Der Datentyp braucht die verschiedenen Präzedenzen der Operatoren nicht zu repräsentieren, sollte aber eine Möglichkeit haben, für jeden Teilausdruck den Typ einzutragen (im Vorgriff auf die nächste Aufgabe).
2. Unter Benutzung der Grammatik aus dem C-Standard (Anhang A) entwickeln Sie einen Parser, welcher Ausdrücke liest, und in der oben gewählten Repräsentation zurückgibt.
3. Zum Testen ihres Parsers entwickeln Sie eine Funktion, welche diese Repräsenation wieder ausgibt, und testen Sie den Parser an ausgewählten Beispielen.

Hinweise: Auch hier können Sie annehmen, dass alle Präprozessor-Anweisunge aufgelöst worden sind, und brauchen Typdefininitionen (`typedef`) nicht zu behandeln.