Universität Karlsruhe (TH)

Lehrstuhl für Programmierparadigmen

Sprachtechnologie und Compiler WS 2009/2010 http://pp.info.uni-karlsruhe.de/
Dozent: Prof. Dr.-Ing. G. Snelting snelting@ipd.info.uni-karlsruhe.de
Übungsleiter: Sebastian Buchwald sebastian.buchwald@kit.edu

Übungsblatt 8 | Ausgabe: 10.12.2009 | Besprechung: 16.12.2009

Aufgabe 1: *AGs: Deklaration nach Benutzung*

Gegeben folgende Grammatik:

$$
\begin{aligned}
S &\rightarrow \textit{CompoundStatement} \\
\textit{CompoundStatement} &\rightarrow \{\ \textit{StatementList}\ \} \\
\textit{StatementList} &\rightarrow \textit{Statement}\ \textit{StatementList} \mid \varepsilon \\
\textit{Declaration} &\rightarrow \textit{Type}\ \textbf{id}\ ; \\
\textit{Type} &\rightarrow \textbf{id}\ ; \\
\textit{EStatement} &\rightarrow \textit{Expression}\ ; \\
\textit{Expression} &\rightarrow \textbf{id} \mid (\ \textit{Expression}\ ) \\
\textit{Typedef} &\rightarrow \textbf{newtype id}\ ; \\
\textit{Statement} &\rightarrow \textit{Declaration} \mid \textit{EStatement} \mid \textit{Typedef} \mid \textit{CompoundStatement}
\end{aligned}
$$

Erzeugen Sie ein System von Attributierungsregeln die Erlauben Typen und Variablen zu benutzen, die erst später deklariert werden.

Aufgabe 2: *Praxis: Namensanalyse*

Unter `http://pp.info.uni-karlsruhe.de/lehre/WS200910/compiler/uebung/nameana.zip` befindet sich Java Sourcecode mit einem Parser und Interpreter. Die implementierte Sprache besitzt eine Zuweisungs- und einer Ausgabeoperation. Ausserdem können mit { und } Namensbereiche geschachtelt werden. Die Verwaltung der Namenstabelle in NameTable.java wurde entfernt.

- Implementieren Sie die fehlende Funktionalität in NameTable.java

Beispiel für eine Eingabe:

```
{
    foo = "bar";
    print(foo);

    {
        foo = "bar2";
        print(foo);
    }

    print(foo);
}
```

Korrekte Ausgabe:

```
foo is bar
foo is bar2
foo is bar
```

```
class Base {
    public int X , Y;
    public int f () {
        return X1 ;
    }
}
class Derived extends Base2 {
    public class Inner extends Base {
        public void func() {
            f3 ();
            X4 = 20;
            Y5 = 10;
        }
        private int Y ;
    }
    public int f () {
        return X6 ;
    }
    public void call_f(Base b ) {
        b7 . f8 ();
        ((Derived) b). f9 ();
    }
    public int X , Y;
}
```

Abbildung 1: Beispiel 1 (Java Code)

```
struct x { int x ; };
x1 k ;
int x ;
void f(void) { x2 = 20; k3 . x4 = 42; }
```

Abbildung 2: Beispiel 2 (C++ Code)

Aufgabe 3: *Namensanalyse*

Die meisten Programmiersprachen besitzen geschachtelte Namensräume. Oft finden sich auch voneinander unabhängige Namensräume für verschiedene Programmierkonstrukte. Betrachten sie die Abbildungen 1, 2 und 3.

- Auf welche Definitionen beziehen sich die markierten (Bezeichner mit Subskript x_1) Referenzen?
- Beschreiben Sie wo in den Beispiele neue Namensraumschachteln entstehen.
- Gibt es unabhängig Namensräume?

```
extern "C" int rand(void);

int main(void) {
    int foo;

 foo :
    if (rand()) {
        float foo ;

        for (float foo = 0; foo < 42; ++ foo_1 ) {
            if (rand() < 13)
                goto foo_2 ;
        }
         foo_3 = 42;
    }
}
```

Abbildung 3: Beispiel 3 (C++ Code)