

	Universität Karlsruhe (TH) Lehrstuhl für Programmierparadigmen Sprachtechnologie und Compiler WS 2008/2009 http://pp.info.uni-karlsruhe.de/ Dozent: Prof. Dr.-Ing. G. Snelting snelting@ipd.info.uni-karlsruhe.de Übungsleiter: Matthias Braun braun@ipd.info.uni-karlsruhe.de
	Übungsblatt 5 Ausgabe: 28.11.2008 Besprechung: 3.12.2008

Aufgabe 1: Praxis: Namensanalyse

Unter <http://pp.info.uni-karlsruhe.de/lehre/WS200809/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
```

Aufgabe 2: Earley Parser

Gegeben folgende Grammatik:

$$\begin{array}{lcl}
 T & \rightarrow & T < - T \\
 & | & T - > T \\
 & | & \& T \\
 & | & \text{id}
 \end{array}$$

Konstruieren Sie mit Hilfe des Earley Algorithmus alle Möglichen Parsebäume für folgende Sätze:

- `id -> id <- id`
- `& id <- id`

```

class Base {
    public int X, Y;
    public void f(void) {
        return X1;
    }
}
class Derived extends Base2 {
    public class Inner extends Base {
        public void func(void) {
            f3 ();
            X4 = 20;
            Y5 = 10;
        }
        private int Y;
    }
    public void f(void) {
        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 Beispielen neue Namensraumschachteln entstehen.
- Gibt es unabhängig Namensräume?

Aufgabe 4: Strukturäquivalenz

Gegeben seien die Typdefinitionen in Abbildung 4

4.1 NEA bauen

Konstruieren Sie für jeden der beiden Typen t und t' einen endlichen Automaten, so dass aus der Gleichheit der Automaten die Strukturäquivalenz der Typen folgt!

4.2 NEA vergleichen

Sind beide Typen strukturäquivalent?

```

extern "C" int rand(void);

int main(void) {
    int foo;

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

        for (float foo = 0; foo < 42; ++foo1) {
            if (rand() < 13)
                goto foo2;
        }
        foo3 = 42;
    }
}

```

Abbildung 3: Beispiel 3 (C++ Code)

```

type t = record
a: INT;
b: ^t;
c: ARRAY[1..7] of tt
end;

type t' = record
a: INT;
b: ^record a: INT; b: ^t' end;
c: ARRAY[1..7] of tt'
end};

type tt' = record
a: INT;
b: ^record a: INT; b: ^tt end
end;

type tt = record
a: INT;
b: ^tt'
end};

```

Abbildung 4: Typdefinitionen