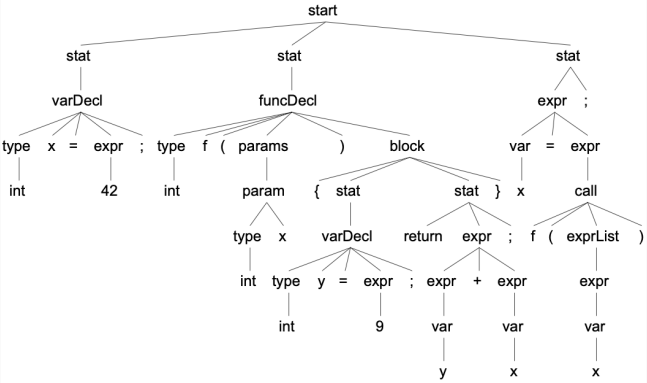
SymbTab0: Überblick Symboltabellen

Carsten Gips (HSBI)

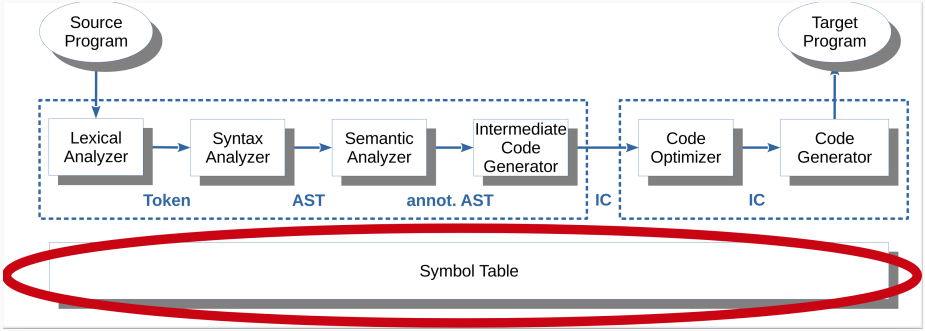
Was passiert nach der Syntaxanalyse?

```
int x = 42;
int f(int x) {
    int y = 9;
    return y+x;
}

x = f(x);
```



Semantische Analyse und Symboltabellen



Syntax und Semantik

- **Syntaxregeln**: Formaler Aufbau eines Programms
- **Semantik**: Bedeutung eines (syntaktisch korrekten) Programms

=> Keine Codegenerierung für syntaktisch/semantisch inkorrekte Programme!

Aufgaben der semantischen Analyse

- Identifikation und Sammlung der Bezeichner
- Zuordnung zur richtigen Ebene (Scopes)

- Typ-Inferenz
- Typkonsistenz (Ausdrücke, Funktionsaufrufe, ...)

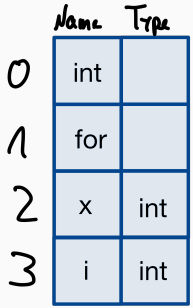
- Validieren der Nutzung von Symbolen
  - Vermeidung von Mehrfachdefinition
  - Zugriff auf nicht definierte Bezeichner
  - (Lesender) Zugriff auf nicht initialisierte Bezeichner
  - Funktionen werden nicht als Variablen genutzt
  - ...

=> Hilfsmittel dazu sind **Symboltabellen**

Einfache Verwaltung von Variablen primitiven Typs

```
int x = 0;
int i = 0;

for (i=0; i<10; i++) {
    x++;
}
```



Was kann jetzt weiter passieren?

```
int x = 0;
int i = 0;

for (i=0; i<10; i++) {
    x++;
}

a = 42;
```

Wo werden Verweise in Symboltabellen gebraucht?

=> Parse Tree und AST enthalten Verweise auf Symboltabelleneinträge

Grenzen der semantischen Analyse

Welche semantischen Eigenschaften kann die semantische Analyse nicht überprüfen?

- Wer ist dann dafür verantwortlich?
- Wie äußert sich das im Fehlerfall?

Wrap-Up

- Semantische Analyse:
  - Identifikation und Sammlung der Bezeichner
  - Zuordnung zur richtigen Ebene (Scopes)
  - Validieren der Nutzung von Symbolen
  - Typ-Inferenz
  - Typkonsistenz (Ausdrücke, Funktionsaufrufe, ...)
- Symboltabellen: Verwaltung von Symbolen und Typen (Informationen über Bezeichner)
- Symboltabelleneinträge werden an verschiedenen Stellen des Compilers generiert und benutzt

LICENSE