Dokumentation der Laborarbeit:

Systemnahes Programmieren

Teil 2

Von:

Boris Bopp (Matr. Nr: 53690)

Maksim (Matr. Nr: )

Henrik Erhart (Matr. Nr: 54323)

Inhalt

[Vorwort 3](#_Toc485222837)

[Architektur 3](#_Toc485222838)

[Parser 3](#_Toc485222839)

[ParseTree 3](#_Toc485222840)

[Visitor 3](#_Toc485222841)

[TypeCheck 3](#_Toc485222842)

[MakeCode 3](#_Toc485222843)

[Neue Verzeichnisse: 3](#_Toc485222844)

# Vorwort

Im ersten Teil dieses Projekts wurde bereits ein Scanner erstellt, der eine Sequenz von Tokens erzeugt. In diesem Teil soll nun aus dieser Sequenz von Tokens ein ParseTree erstellt werden und anschließend damit ein TypeCheck gemacht werden. Läuft dieser ohne erkannte Fehler durch soll danach der ausführbare Maschinencode erzeugt werden.

# Architektur

## Parser

Der Parser verarbeitet die von Scanner gelieferte Sequenz von Token und erstellt daraus den ParseTree. Syntaktische Fehler werden während diesem Prozess erkannt und auf der Konsole ausgegeben. Dazu wird die Methode des Rekursiven Abstiegs benutzt. Bei dieser wird der ParseTree rekursiv nach den Regeln der gegebenen Grammatik aufgebaut. Dabei wird auf jeder Rekursionsebene die möglichen Folgeübergänge anhand des nächsten gelieferten Tokens überprüft werden. Da die Grammatik der Programmiersprache Linkseindeutig ist kann stehst anhand des nächsten Tokens der richtige Übergang gewählt werden. Ist der richtige Übergang gefunden, werden dem aktuellen Knoten die entsprechenden Kinderknoten rekursiv berechnet und hinzugefügt. Falls laut Grammatik ein bestimmtes Token gefordert wird, dieses aber nicht das aktuelle ist, so wird ein Fehler ausgegeben und der Parse-Vorgang wird abgebrochen. Epsilon-Übergänge werden dadurch erkannt, dass keine anderen Übergänge ausgeführt werden können. Bei Epsilon-Übergängen wird der aktuelle Knoten mit einem „\_isEps“ – Flag versehen.

## ParseTree

Der ParseTree setzt sich aus Knoten und Blättern zusammen. Die Blätter bilden dabei immer die tiefste Ebene des Baumes und können keine weiteren Kinder haben. Dafür bekommen Sie jeweils ein Token zugeordnet. Knoten haben in der Regel weitere Kinder. Die einzige Ausnahme sind Knoten, beim Parsen zu einem Epsilon-Übergang führen.

## Visitor

### TypeCheck

### MakeCode

# Neue Verzeichnisse:

|-- Parser

|-- debug

|-- includes

|-- objs

|-- src

|-- makefile

Wie auch in Teil eins wird für die Parser-Komponente ein weiteres Verzeichnis erstellt. Die makefile-Struktur wie im ersten Teil des Projekts weitergeführt.