Hochschule Aalen

Projektbericht
Studiengang Medieninformatik

---

expressionviewer

von

Laurin Agostini

60526

Betreuender Professor:	Prof. Dr. Winfried Bantel
Einreichungsdatum:	25. November 2019

Eidesstattliche Erklärung

Hiermit erkläre ich, **Laurin Agostini**, dass ich die vorliegenden Angaben in dieser Arbeit wahrheitsgetreu und selbständig verfasst habe.

Weiterhin versichere ich, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben, dass alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, kenntlich gemacht sind und dass die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Ort, Datum Unterschrift (Student)

Kurzfassung

Ziel der Kurzfassung ist es, einen (eiligen) Leser zu informieren, so dass dieser entscheiden kann, ob der Bericht für ihn hilfreich ist oder nicht (neudeutsch: Management Summary). Die Kurzfassung gibt daher eine kurze Darstellung

- des in der Arbeit angegangenen Problems
- der verwendeten Methode(n)
- des in der Arbeit erzielten Fortschritts.

Dabei sollte nicht auf die Struktur der Arbeit eingegangen werden, also Kapitel 2 etc. denn die Kurzfassung soll ja gerade das Wichtigste der Arbeit vermitteln, ohne dass diese gelesen werden muss. Eine Kapitelbezogene Darstellung sollte sich in Kapitel 1 unter Vorgehen befinden.

Länge: Maximal 1 Seite.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

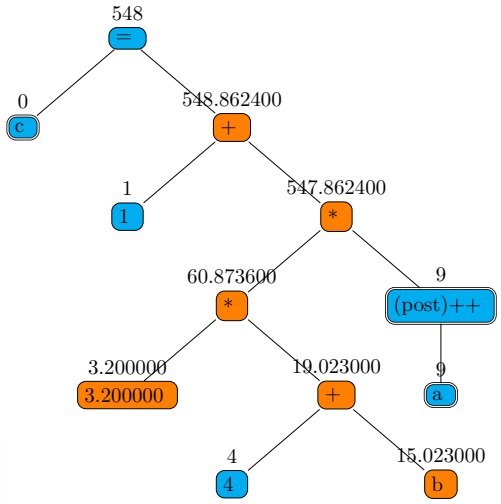
RUP Ratified Unified Process 5

1. Einleitung

Die Einleitung dient dazu, beim Leser Interesse für die Inhalte Praxissemesterberichts zu wecken, die behandelten Probleme aufzuzeigen und die zu ihrer Lösung entwickelten Konzepte zu beschreiben.

1.1. Motivation

In der Motivation wird dargestellt, welche Bedeutung die im Praxissemester zu entwickelnden Lösungen für das betreuende Unternehmen haben. Es wird beispielsweise aufzeigt, in welches Produkt sie eingehen, welcher Ablauf verbessert werden soll etc.



Name	Value	Type	const?
a	9	int	0
b	15.023000	double	1
c	0	int	0

Name	Value	Type	const?
a	10	int	0
b	15.023000	double	1
c	548	int	0

Abbildung 1.1.: Beispielausgabe in Tikz

1.2. Problemstellung und -abgrenzung

Die Problemstellung dient dazu, das zu lösende Problem klar zu definieren und abzugrenzen. Der Praktikant soll ein klares Verständnis des zu lösenden Problems haben. Insbesondere soll auch verhindert werden, dass zu viele Probleme gleichzeitig angegangen werden. Eine Negativabgrenzung verhindert, dass beim Leser später nicht erfüllte Erwartungen geweckt werden.

1.3. Ziel der Arbeit

Es soll ein Werkzeug für Programmieranfänger erstellt werden, womit diese einen C89 / ANSI C Ausdruck analysieren können. Das Werkzeug soll den Ausdruck in einen Syntaxbaum mit Typinformationen umwandeln, d. h. für jeden Knoten im Syntaxbaum soll der Datentyp und der aktuelle Wert angezeigt werden. Außerdem sollen Fehler und ihre Folgen im Syntaxbaum ausdrücklich markiert und beschrieben werden.

1.4. Vorgehen

Nachdem mit Problemstellung und Ziel gewissermaßen Anfangs- und Endpunkt des Praktikums beschrieben sind, wird hier der zur Erreichung des Ziels eingeschlagene Weg vorgestellt. Dazu werden typischerweise die folgenden Kapitel und ihr Beitrag zur Erreichung des Ziels der Arbeit kurz beschrieben. Die folgenden Kapitel sind ein – möglicher – Aufbau, Abweichungen können durchaus notwendig sein. Zur Darstellung des Vorgehens ist eine grafische Darstellung sinnvoll, bei der die einzelnen Lösungsschritte und ihr Zusammenhang dargestellt werden. Ein Beispiel hierfür findet sich in Abbildung **??**.

2. Grundlagen

In diesem Kapitel das für das Praktikum relevante Grundlagenwissen dargestellt. Der Praktikant soll hierzu das ihm durch Vorlesungen bekannte, bzw. durch Recherchen vertiefte theoretische Wissen darstellen, das für die Lösung der im Praktikum gestellten Probleme notwendig ist.

Dabei ist darauf zu achten, nur solche Inhalte in das Grundlagenkapitel aufzunehmen, die später auch verwendet werden (Problembezogenheit). Ebenso ist auf eine ausreichend tiefe und vollständige Darstellung der Grundlagen zu achten.

Für die Erstellung des Literaturverzeichnisses wird das Werkzeug JabRef[**JabRef:JabRef**] verwendet.

Sie können aber auch das Werkzeug Citavi[**SAS:Citavi**] benutzen und dort nach BIBTEX exportieren.

2.1. Programmiersprachen

2.1.1. C89 / ANSI C

Erster offizieller Standard der C-Programmiersprache, der 1989 veröffentlicht wurde. Wird als Grundlage für die zu verarbeitenden Ausdrücke genommen.

2.1.2. C99

Zweiter offizieller Standard der C-Programmiersprache, 1999 veröffentlicht. Mit diesem Standard wurde das Kommandozeilenprogramm der Projektarbeit geschrieben.

2.2. Compilerbau

2.2.1. Ausdruck

Eine Kombination von Konstanten, Variablen, Funktionsaufrufen und Operationen die einen Wert liefert. Ausdrücke an sich haben keinen direkten Einfluss auf den Programmfluss, anders wie Verzweigungen(if) oder Schleifen(while, for). «TODO»

2.2.2. Syntaxbaum

Hierarchische Darstellung der Zergliederung des zu verarbeitenden Quellcodes.

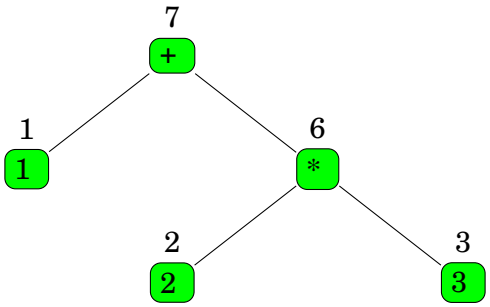


Abbildung 2.1.: Syntaxbaum für den Ausdruck *1 + 2 * 3*

3. Problemanalyse

Die Analyse des zu lösenden Problems ist Grundlage für jedes ingenieurmäßige Vorgehen. Daher soll in diesem Kapitel das zu lösenden Problem auf Basis des im Grundlagenkapitel aufbereiteten Wissens analysiert werden. Hierzu ist insbesondere notwendig zu klären, wie sich das Gesamtproblem in Teilprobleme zerlegen lässt und welche Abhängigkeiten zwischen diesen bestehen.

Bei Software-Projekten befindet sich an dieser Stelle typischerweise die Anforderungsanalyse des Ratified Unified Process (RUP).

4. Implementierung

Die Grundarchitektur des Lösungskonzeptes besteht aus einer klassischen Unterteilung in Front- und Backend. Das Frontend ist für die Datenabfrage und die Darstellung zuständig. Im Backend wiederum steckt die eigentliche Logik des Programms. Da das eigentliche Backend auch als eigenständiges Kommandozeilenprogramm funktionieren soll, wurde die Kommunikation mit dem Frontend auf ein Backend-Interface ausgelagert, die das Kommandozeilenprogramm auf dem Server aufruft und die Resultate zurück an das Frontend schickt.

4.1. Frontend

Das gesamte Frontend besteht aus einer HTML-, einer CSS- und zwei JavaScript-Dateien (eine davon ist die zur Darstellung des Syntaxbaum verwendete D3.js Bibliothek). Die HTML und CSS Dateien sind je unter 50 Quellcodezeilen lang und somit sehr trivial gehalten. In der eigenen JavaScript-Datei werden die Eingaben des Benutzers ausgelesen, formatiert und dann an das Backend geschickt. Die Antwort wird wiederum verwendet um den resultierenden Syntaxbaum zu zeichnen.

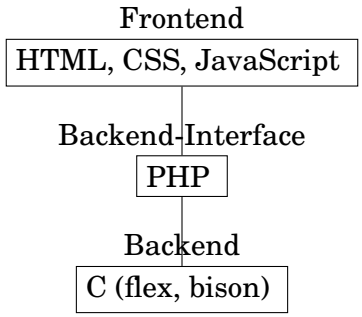


Abbildung 4.1.: Grundarchitektur

4.2. Backend-Interface

Das Backend-Interface besteht aus einer einzigen PHP-Datei, die im Grunde nichts anderes macht, als das eigentliche Backend mit den Parametern aus der Anfrage des Frontends aufzurufen und die Ergebnisse formatiert wieder ans Frontend zurückzuschicken.

4.3. Backend

Der Großteil der Arbeit ist in das, auch eigenständig als Kommandozeilenprogramm benutzbare, Backend geflossen. Dieses nimmt grundlegend einen Ausdruck und eine Reihe von Symboldefinitionen(die auch leer sein kann) entgegen und gibt den resultierenden Syntaxbaum in einem der Formate (JSON oder tex) auf der Standardausgabe aus. Alternativ kann die Ausgabe auch direkt in eine Datei geschrieben werden.

Bei Software-Projekten besteht dieses Kapitel typischerweise aus den Phasen Implementierung & Test im RUP.

5. Evaluierung

Aufgabe des Kapitels Evaluierung ist es, in wie weit die Ziele der Arbeit erreicht wurden. Es sollen also die erreichten Arbeitsergebnisse mit den Zielen verglichen werden. Ergebnis der Evaluierung kann auch sein, das bestimmte Ziele nicht erreicht werden konnten, wobei die Ursachen hierfür auch außerhalb des Verantwortungsbereichs des Praktikanten liegen können.

6. Zusammenfassung und Ausblick

6.1. Erreichte Ergebnisse

Die Zusammenfassung dient dazu, die wesentlichen Ergebnisse des Praktikums und vor allem die entwickelte Problemlösung und den erreichten Fortschritt darzustellen. (Sie haben Ihr Ziel erreicht und dies nachgewiesen).

6.2. Ausblick

6.2.1. Erweiterbarkeit der Ergebnisse

Hinzufügen weiterer primitiver Datentypen (char, long, bool, float, etc.)

Prinzipiell dürfte das Hinzufügen weiterer primitiver Datentypen keine großen Probleme bereiten, jedoch müsste so gut wie jeder *Node* angepasst werden, um mit dem neuen Datentyp umgehen zu können.

- Schwierigkeit: Niedrig
- Aufwand: Mittel

Felder als Datentyp

Für die Implementierung von Feldern müsste auf jeden Fall die Symbol Tabelle überarbeitet werden. Ein einfacher Weg wäre hier zum Beispiel die Definition von 10 *int*-Symbolen mit automatisch generierten Namen für ein *int*-Feld der Größe 10. Wenn wir dann nur von einfachen Lese- und Schreiboperationen mithilfe des Index-Operators *[]* auf die einzelnen Datenfelder ausgehen, würde sich der Aufwand bei den *Nodes* hier wohl auf eine Anpassung des Lese-*Nodes* von Variablen und dem Zuweisungs-*Nodes* beschränken.

- Schwierigkeit: Niedrig

- Aufwand: Niedrig

Zeiger als Datentyp

Da man Variablen bisher nur konstante Werte und keine Ausdrücke oder andere Variablen zuweisen kann ist der Nutzen für Zeiger noch sehr begrenzt. Zur Lösung des Problems müssten die Werte der Variablendefinitionen auch als Ausdrücke ausgewertet werden. Um den Zugriff über die Adresse einer Variable zu erlauben, könnte die Symbol Tabelle wahrscheinlich relativ einfach angepasst werden.

- Schwierigkeit: Mittel
- Aufwand: Hoch

Zeigerarithmetik

Zeigerarithmetik (Berechnung der Zeigeradresse inklusive Inkrement und Dekrement) benötigt eine Emulation des Speichers und somit grundlegende Änderungen an mehreren Basissystemen (zum Beispiel an der Symbol Tabelle).

- Schwierigkeit: Hoch
- Aufwand: Hoch

Hinzufügen von Anweisungen (Verzweigungen, Schleifen)

?

- Schwierigkeit: ?
- Aufwand: ?

Bearbeiten des Syntaxbaums...

Das Bearbeiten des Syntaxbaums im Frontend müsste mit der D3.js-Bibliothek möglich sein, jedoch habe ich mich im Rahmen dieser Arbeit nicht weiter damit befasst.

- Schwierigkeit: Nicht einschätzbar
- Aufwand: Nicht einschätzbar

...und Generieren des entstandenen *C*-Ausdrucks

Das Generieren des *C*-Ausdrucks aus einem Syntaxbaum besteht größtenteils nur aus Textkonkatenation der Namen der einzelnen Knoten. Bei einzelnen Knoten (zum Beispiel der ternäre Operator oder Funktionen mit Parametern) ist etwas mehr Aufwand von Nöten.

- Schwierigkeit: Niedrig
- Aufwand: Niedrig

6.2.2. Übertragbarkeit der Ergebnisse

A. Anhang A

B. Anhang B