Vertalerbouw

Het verslag moet inzicht geven hoe de taal gedeﬁnieerd is, en hoe de problemen die zich voordeden

bij het maken van de vertaler opgelost zijn. Vermeld ook wie voor welk onderdeel verantwoordelijk is en welke delen samen gemaakt zijn. Het verslag van de practicumopdracht dient in ieder

geval de volgende onderdelen te bevatten:

# Inleiding - Kim

Korte beschrijving van de practicumopdracht.

In dit project wordt een complete vertaler voor een zelf ontworpen programmeertaal ontwikkeld. Dit wordt gedaan met behulp van ANTLR. De zelf ontworpen programmeertaal zal vertaald worden naar Java ByteCode.

Er moet een parser, een lexer en een checker gemaakt worden om de zelf ontworpen taal te kunnen inlezen en te controleren op fouten. Ook moet er een codegenerator gemaakt worden om van de ingelezen informatie Java ByteCode te maken (met behulp van Jasmin).

In dit verslag zal als eerste de zelf ontworpen programmeertaal worden gedefinieerd. Er zal beschreven worden welke problemen zich voordeden en hoe ze zijn opgelost. Daarna wordt uitgelegd wat de syntax, context-beschrijving en semantiek van de programmeertaal zijn. Als volgt worden de vertaalregels voor de taal uiteengezet. Vervolgens wordt er een beschrijving gegeven van Java-programmatuur. Het testplan en de testresultaten worden daarna getoond en als laatste zullen er conclusies neergezet worden ter afsluiting van het project.

# Beknopte beschrijving - Kim

van de programmeertaal (maximaal e´en A4-tje)

De programmeertaal heeft de naam Chocolate omdat de taal veel *begrippen/termen?* bevat die gerelateerd zijn aan chocola. De volgende (chocolade) termen hebben de volgende betekenissen:

reep : constant | bounty : +

bonbon : variable | mars : -

| nuts : !

milka : assign | milkyway : \*

droste : read | dove : /

verkade : print | maltesers : %

| snickers : <

wit : integer | kinder : <=

melk : char | bueno : >=

puur : boolean | lion : >

| kitkat : ==

cacao : comment | bros : !=

| rolo : &&

| twix : ||

Een programma geschreven in Chocolate bestaat uit declaraties en statements, die door elkaar heen gebruikt kunnen worden, er wordt alleen vereist dat een constante of variabele gedeclareerd moet zijn voordat het gebruikt kan worden. Elke declaratie en statement wordt afgesloten met een puntkomma.

Voor het declareren van een constante moet er eerst worden aangegeven dat het gaat om een constante. Dit wordt gevolgd door het type, een identifier (een identifier kan een combinatie zijn van letters en cijfers, maar begint altijd met een letter) en de ‘assignment’ waardoor een waarde wordt toegekend aan de constante. Het declareren van een variabele gaat op vrijwel dezelfde wijze, alleen is het bij een variabele niet nodig om er een ‘assignment’ te doen. De taal heeft drie basis typen: integer (wit), char (melk) en boolean (puur).

Er zijn vijf verschillende soorten statements, namelijk de assign statement, de read statement, de print statement, de if-then-else statement en de while-do statement. De keywords van de if-then-else statement en de while-do statement hebben geen speciale termen gekregen om de taal Chocolate leesbaar te houden.

cacao Dit is een stukje voorbeeld code

bonbon wit a; cacao Een variabele wordt hier gedeclareerd

reep wit b milka 5; cacao Een constante wordt hier gedeclareerd

droste(a); cacao Er wordt een integer gelezen en in a opgeslagen

a milka a milkyway b; cacao a wordt met b vermenigvuldigd en in a

cacao opgeslagen

verkade(a); cacao a wordt geprint

*Stukje voorbeeld code*

# Problemen en oplossingen - Kim

uitleg over de wijze waarop je de problemen die je bent tegengekomen bij het maken van de opdracht hebt opgelost (maximaal twee A4-tjes).

# Syntax, context-beperkingen en semantiek - Tanja

van de taal met waar nodig nadere uitleg over de

betekenis. Geef de beschrijving bij voorkeur in dezelfde terminologie als die gebruikt is bij

de beschrijving van Triangle in Watt & Brown (hoofdstuk 1 en appendix B).

# Vertaalregels - Tanja

voor de taal, d.w.z. de transformaties waaruit blijkt op welke wijze een opeenvolging van symbolen die voldoet aan een produktieregel wordt omgezet in een opeenvolging van TAM-instructies. Vertaalregels zijn de ‘code templates’ van hoofdstuk 7 van Watt

& Brown.

# Beschrijving van Java-programmatuur - Tanja

r. Beknopte bespreking van de extra Java-klassen die

u gedeﬁnieerd heeft voor uw compiler (b.v. symbol table management, type checking, code

generatie, error handling, etc.). Geef ook aan welke informatie in de AST-nodes opgeslagen

wordt.

# Testplan en –resultaten - Kim

. Bespreking van de ‘correctheids-tests’ aan de hand van de criteria

zoals deze zijn beschreven in het xA.5 van deze appendix. Aan de hand van deze criteria moet

een verzameling test-programma’s in het taal geschreven worden die de juiste werking van de

vertaler en interpreter controleren. Tot deze test-set behoren behalve correcte programma’s

die de verschillende taalconstructies testen, ook programma’s met syntactische, semantische

en run-time fouten. Alle uitgevoerde tests moeten op de CD aanwezig zijn; van e´en testprogramma moet de ´

uitvoer in de appendix opgenomen worden (zie onder).

# Conclusies - Tanja