The

CKcc

Programming Language

Compiler Construction Course

Summer 2006

Team Members:

Clemens Krainer 9020112

Salzburg, 6 June 2006

Content

1. Introduction

The CKcc programming language is a subset of the C programming language, as standardised in [1].

2. EBNF of the Compiler

2.1 Formal Definition

CharSet = '!', '#' .. '0xFF'.

Identifier = [Letter | '_']{Letter | Digit | '_'}.

String = '"' { CharSet } '"'.

Constant = Number | '\'' Letter '\'' | String.

Number = Digit { Digit }.

Letter = 'A' .. 'Z' | 'a' .. 'z'.

Digit = '0' .. '9'.

2.2 Expressions

Program = { Declaration }.

Declaration = DataDeclaration | FunctionDeclaration.

DataDeclaration = [("extern" | "static")] (StructDeclaration | SimpleDeclaration).

FunctionDeclaration = [("extern" | "static")] TypeName [Pointer] Identifier "(" NameList ")" (";" | Block).

SimpleDeclaration = MemberDeclaration [Init] ";".

StructDeclaration = "struct" Identifier "{" MemberDeclaration ";" { MemberDeclaration ";" } "}" ";".

MemberDeclaration = TypeName [Pointer] Identifier ["[" Number "]"].

TypeName = SimpleType | StructName.

Pointer = "*" { Pointer }.

SimpeType = "void" | "char" | "int".

StructName = "struct" Identifier.

NameList = TypeName [Pointer] Identifier { "," TypeName [Pointer] Identifier }.

Block = "{" { DataDeclaration } { Statement } "}".

Statement = Block
| AssignmentExpression ";"
| "if" "(" AssignmentExpression ")" Statement
["else" Statement]
| "while" "(" AssignmentExpression ")" Statement
| "break" ";"
| "continue" ";"
| "return" [AssignmentExpression] ";" .

```
AssignmentExpression  = { [ Pointer ] [ ("++" | "--") ] ListValue { ( "." | "->" ) ListValue }[ ("++" | "--") ] "=" }
                        [ TypeCastExpression ] LogicalOrExpression.

TypeCastExpression    = "(" TypeName [ Pointer ] ")".

LogicalOrExpression   = LogicalAndExpression { "||" LogicalAndExpression }.

LogicalAndExpression  = ConditionalExpression { "&&" ConditionalExpression }.

ConditionalExpression = SimpleExpression [ ("==" | "!=" | "<" | "<=" | ">=" | ">") SimpleExpression ] .

SimpleExpression      = UnaryOperator Term { ( "+" | "-" | "|" ) Term }.

UnaryOperator         = "+" | "-" | "!" | "&" | "*" | "~".

Term                  = Factor { ( "*" | "/" | "%" | "&" | "<<" | ">>" ) Factor }.

Factor                = "sizeof" "(" TypeName ")"
                        | Constant
                        | "(" AssignmentExpression ")"
                        | Identifier "(" [ Parameter ] ")"
                        | [ ("++" | "--") ] ListValue { ( "." | "->" ) ListValue } [ ("++" | "--") ].

ListValue             = Identifier [ "[" LogicalOrExpression "]" ].

Parameter             = LogicalOrExpression { "," LogicalOrExpression }.

Constant              = Number | Character | String.
SignedConstant        = "-" Number | Constant

Init                  = "=" SignedConstant
                        | "{" SignedConstant { "," SignedConstant } "}"
                        | AssignmentExpression.
```

3. Compiler Keywords

Used compiler keywords:
`if`, `elsif`, `else`, `while`, `struct`, `sizeof`, `return`, `break`, `continue`, `char`, `int`, `void`, `static`, `extern`, `enum`.

Not used compiler keywords:
`for`, `short`, `long`, `float`, `double`, `signed`, `unsigned`, `auto`, `register`, `typedef`, `union`, `const`, `volatile`, `goto`, `switch`, `case`, `default`, `do`.

4. References

[1] American National Standards Institute (1989) *Programming Language C.* ANSI X3.159-1989