Cairo University

Faculty Of Engineering

Computer Department

Compilers

Mini-C compiler

Team Members

Andrew Hany Gabriel

Mohamed Abbas Sharaky

Mohamed El-Wattar

Project Overview

We implemented a mini-version of a C compiler that supports the basic data types (int, bool, float) and support C operations like assignments, mathematical expressions and logical expressions. We also support switch, for, while and do while statements in our languages while trying to match the original syntax of the C language.

Tools used

- Win Flex-bison: a windows port of the GNU distribution of Lex and Yacc
- Visual C++ 2013: For compiling the *.tab.cpp, *.felx.cpp, *.tab.h files.
- Visual studio 2013: As our prefered IDE.
- Visual C# Winforms: For our GUI.
- FastColoredTextBox: For syntax highlighting in the GUI.

Tokens used

Token	Description	Token	Description
INTEGER	An Integer number	IF	If statement
FLOAT	A floating point number	ELSE	Else in an If statement
BOOL	A boolean value	AND	Logical and (&&)
VARIABLE	A variable identifier	OR	Logical OR (||)
CONST	Constant operator	GE	Greater than or equal (>=)
DEC_INT	Declare an integer	LE	Less than or equal (<=)
DEC_FLOAT	Declare a float	EQ	Equal operator (==)
DEC_BOOL	Declare a boolean	NE	Not equal operator (!=)
DO	Starts a do-while statement	- + * / % | &	Arithmetic operators
WHILE	While statement	< >	Comparison operators
FOR	For loop statement	!	Not operator
CONTINUE	Skip current iteration	( ) { } ; :	Miscellaneous tokens
SWITCH	Switch statement		
CASE	Case inside a switch		
BREAK	Breaks a loop or a case		
DEFAULT	Default case in a switch		

Production rules

- **program**: statement_list
- **scope**: '{' '}' | '{' statement_list '}'
- **statement_list**: statement | statement_list statement
- **statement**: ';' | assignment ';' | declaration | const_declaration | switch_statement | loop | condition | break | continue | error '}' | error ';'
- **assignment**: expression | VARIABLE '=' assignment
- **expression**: constant | VARIABLE | expression '+' expression | expression '-' expression | expression '*' expression | expression '/' expression | expression '%' expression | expression '&' expression | expression '|' expression | expression '<' expression | expression '>' expression | expression GE expression | expression LE expression | expression NE expression | expression EQ expression | expression AND expression | expression OR expression | '-' expression | '!' expression | '+' expression | '(' assignment ')'
- **constant**: INTEGER | FLOAT | BOOL
- **declaration**: DEC_INT VARIABLE ';' | DEC_INT VARIABLE '=' assignment ';' | DEC_FLOAT VARIABLE ';' | DEC_FLOAT VARIABLE '=' assignment ';' | DEC_BOOL VARIABLE ';' | DEC_BOOL VARIABLE '=' assignment ';'
- **const_declaration**: CONST DEC_INT VARIABLE '=' assignment ';' | CONST DEC_FLOAT VARIABLE '=' assignment ';' | CONST DEC_BOOL VARIABLE '=' assignment ';'
- **switch_statement**: SWITCH '(' assignment ')' '{' case_list '}'
- **case**: CASE constant ':' scope
- **default**: DEFAULT ':' scope
- **case_list**: ε | case_list case | case_list default
- **loop**: WHILE '(' assignment ')' scope | DO scope WHILE '(' assignment ')' | FOR '(' for_decl for_cond for_inc ')' scope
- **for_decl**: declaration | ';'
- **for_cond**: assignment ';' | ';'
- **for_inc**: assignment | ε
- **condition**: IF '(' assignment ')' scope | IF '(' assignment ')' scope ELSE scope
- **break**: BREAK ';'
- **continue**: CONTINUE ';'

Quadruples

Quadruple	Description
MOV A, B	A = B
ADD A, B, C	A = B + C
SUB A, B, C	A = B - C
MUL A, B, C	A = B * C
DIV A, B, C	A = B / C
MOD A, B, C	A = B % C
ANDB A, B, C	A = B & C
ORB A, B, C	A = B | C
GT A, B, C	A = B > C
GTE A, B, C	A = B >= C
LT A, B, C	A = B < C
LTE A, B, C	A = B <= C
NEQ A, B, C	A = B != C
EQ A, B, C	A = B == C
AND A, B, C	A = B && C
OR A, B, C	A = B || C
NEG A, B	A = -B
NOT A, B	A = !B
DEF TYPE, A	Declare A with type “Type”
DEFC TYPE, A, B	Declare A as a constant with type “Type” and A = B
JNZ A, B	If A is not zero jump to B
JZ A, B	If A is zero jump to B
JMP A	Jump to A