Credit hours system

Faculty of engineering

Compiler Project

Simple Programming Language using Lex and Yacc

Team #16:

Mostafa Ashraf Ahmed Kamal	1180406
Ahmed Mohamed Ismail Nabeel	1180501
Habiba Assem Mohamed Osman	1180450
Moaz Mohamed el Sherbiny	1180528

Project Overview:

The aim of our project is to build a simple compiler of a programming language similar to the C language using the Lex and Yacc compiler generating packages.

The important constructs considered in our language are:

- Variable and constant declarations
- Assignment statements
- Mathematical and Boolean expressions
- Enum statements
- Conditional statements (if statements, switch cases)
- Loops (for, while, do while loops)
- Function declaration and calls
- Block structures

The program takes a source code and performs all the compilation steps (lexical analysis, syntax analysis and lastly semantic analysis and code generation).

The output for each source code would be:

- Error handling file (containing all the error messages)
- Symbol table file
- Quadruples file

Tools and Technologies used:

We used the Lex and Yacc compiler generating packages combined with some C++ programs to implement the symbol table and quadruples logic.

A list of tokens and a description of each:

Token	Description	
Types		
INT	int	Integer type
FLOAT	float	Float type
CHAR	char	Character type
BOOL	bool	Boolean type
STRING	string	String type
Constant		
CONST	const	Constant declaration
Mathematical Expressions		
PLUS	+	Addition
MINUS	-	Subtraction
MULT	*	Multiplication
DIV	/	Division
PLUS_EQ	+=	Add value and store it in LHS
MINUS_EQ	-=	Subtract value and store it in LHS
MULT_EQ	*=	Multiply value and store it in LHS
DIV_EQ	/=	Divide value and store it in LHS
INC	++	Increment value
DEC	--	Decrement value

Comparison Operators		
LT	<	Lower than
GT	>	Greater than
GE	>=	Greater than or equal
LE	<=	Lower than or equal
EQ_EQ	==	Equal
NE	!=	Not equal
Logical Expressions		
AND	&&	And operator
OR	||	Or operator
NOT	!	Not operator
Assignment Operator		
EQUAL	=	assign
If then else statement		
IF	if	Defining an If statement
ELSEIF	else if	Defining “if else”
ELSE	else	Defining “else”
Loops		
WHILE	while	Defining while loop
DO	do	Defining do while loop
FOR	for	Defining for loop
BREAK	break	Break statement
CONTINUE	continue	Continue statement
Switch statement		
SWITCH	switch	Defining switch cases
CASE	case	Cases to switch on

DEFAULT	default	Default statement
Functions		
VOID	void	Void type
RETURN	return	Return from function
COMMA	,	Comma
COLON	:	colon
Enums		
ENUM	enum	Enum type
Brackets		
OPENBRACKET	(	Opening bracket
CLOSEDBRACKET	)	Closing bracket
OPENCURL	{	Opening curly bracket
CLOSEDCURL	}	Closing curly bracket
Stop Characters		
SEMICOLON	;	Semicolon: end of statement
Identifier and Numbers		
TRUE_VAL	true	Defining true Boolean value
FALSE_VAL	false	Defining false Boolean value
IDENTIFIER	[a-zA-Z_][a-zA-Z0-9_]*	Defining variable name
NUMBER	[0-9]+	Defining an integer value
FLOAT_NUM	[0-9]+\.[0-9]+	Float value
STRING_VAL	\"[^\"]*\"	String value

A list of the language production rules:

statements	
Statements:	statements statement
	statement
Statement:	expression SEMICOLON
	assignment_statement
	var_declaration
	constant_declaration
	enum_statement
	if_statement
	while_statement
	do_while_statement
	for_statement
	switch_statement
	break_statement
	continue_statement
	function
	OPENCURL statements CLOSEDCURL
	RETURN return_value SEMICOLON
	SEMICOLON

Values & Types	
Value:	expression
	STRING_VAL
	CHAR_VAL
Type:	INT
	FLOAT
	CHAR
	STRING
	BOOL
constant:	INT_VAL
	FLOAT_VAL
	STRING_VAL
	CHAR_VAL
	TRUE_VAL
	FALSE_VAL
Expressions	
expression:	boolean_expression
	arithmetic_expression
boolean_expression:	expression EQ_EQ arithmetic_expression
	expression NE arithmetic_expression
	expression GE arithmetic_expression
	expression LE arithmetic_expression
	expression GT arithmetic_expression
	expression LT arithmetic_expression
	expression AND arithmetic_expression

	expression OR arithmetic_expression
	NOT expression
	TRUE_VAL
	FALSE_VAL
arithmetic_expression:	binary_expression
	unary_expression
unary_expression:	IDENTIFIER INC
	IDENTIFIER DEC
binary_expression:	binary_expression PLUS term
	binary_expression MINUS term
	term
term:	factor
	term MULT factor
	term DIV factor
factor:	INT_VAL
	FLOAT_VALA
	function_call
	IDENTIFIER
	OPENBRACKET expression CLOSEDBRACKET
Variable Declaration and assignment	
assignment_statement:	IDENTIFIER EQUAL expression SEMICOLON
	IDENTIFIER PLUS_EQ expression SEMICOLON
	IDENTIFIER MINUS_EQ expression SEMICOLON

	IDENTIFIER MULT_EQ expression SEMICOLON
	IDENTIFIER DIV_EQ expression SEMICOLON
var_declaration:	type IDENTIFIER EQUAL value SEMICOLON
	type IDENTIFIER SEMICOLON
	ENUM IDENTIFIER IDENTIFIER SEMICOLON
constant_declaration:	CONST type IDENTIFIER EQUAL value SEMICOLON
If statement	
if_statement:	IF OPENBRACKET value CLOSEDBRACKET OPENCURL statements CLOSEDCURL else_if_statement else_statement
else_statement:	ELSE OPENCURL statements CLOSEDCURL
else_if_statement:	else_if_statement ELSEIF OPENBRACKET value CLOSEDBRACKET OPENCURL statements CLOSEDCURL
While and do while statement	
while_statement:	WHILE OPENBRACKET value CLOSEDBRACKET statement

do_while_statement:	DO statement WHILE OPENBRACKET value CLOSEDBRACKET SEMICOLON
For statement	
for_statement:	FOR OPENBRACKET for_initialization value SEMICOLON for_expression CLOSEDBRACKET OPENCURL statements CLOSEDCURL
for_initialization:	assignment_statement
	var_declaration
	constant_declaration
	value SEMICOLON
	SEMICOLON
for_expression:	IDENTIFIER EQUAL value SEMICOLON
	IDENTIFIER PLUS_EQ expression
	IDENTIFIER MINUS_EQ expression
	IDENTIFIER MULT_EQ expression
	IDENTIFIER DIV_EQ expression
	value
Switch statement	
switch_statement:	SWITCH OPENBRACKET value CLOSEDBRACKET OPENCURL case_list CLOSEDCURL
case_list:	case_list case_statement
	case_statement

case_statement:	CASE value COLON statements
	DEFAULT COLON statements
Break or Continue	
break_statement:	BREAK SEMICOLON
continue_statement:	CONTINUE SEMICOLON
Enums	
enum_statement:	enum_declaration
	enum_initialization
enum_initialization:	ENUM IDENTIFIER IDENTIFIER EQUAL IDENTIFIER SEMICOLON
enum_declaration:	ENUM IDENTIFIER OPENCURL enum_list CLOSEDCURL SEMICOLON
enum_list:	enum_list COMMA IDENTIFIER
	IDENTIFIER
Functions	
function:	function_prototype OPENCURL statements CLOSEDCURL
return_value:	value
function_prototype:	type IDENTIFIER OPENBRACKET parameters CLOSEDBRACKET
	type IDENTIFIER OPENBRACKET CLOSEDBRACKET
	VOID IDENTIFIER OPENBRACKET parameters CLOSEDBRACKET
	VOID IDENTIFIER OPENBRACKET CLOSEDBRACKET

parameters:	parameters COMMA single_parameter
	single_parameter
single_parameter:	type IDENTIFIER
	type IDENTIFIER EQUAL constant
function_call:	IDENTIFIER OPENBRACKET call_parameters CLOSEDBRACKET
call_parameters:	call_parameters COMMA value
	value

A list of the quadruples:

Quadruple	Description
Boolean Expressions	
T0 := a==b	a==b
T0 := a!=b	a!=b
T0 := a>=b	a>=b
T0 := a<=b	a<=b
T0 := a>b	a>b
T0 := a<b	a<b
T0 := a AND b	a AND b
T0 := NOT a	NOT a
Mathematical Expressions	
T0 := INC a	a++

T0 := DEC a	a--
T0 := a ADD b	a+b
T0 := a SUB b	a-b
T0 := a MUL b	a*b
T0 := a DIV b	a/b
a :+=b	a+=b
a :-=b	a-=b
a :*=b	a*=b
a :/=b	a/=b
Assignment statement	
a := b	a=b
Var and const declaration	
a :=b	Type a = b
a :=b	Const Type a = b