CAIRO UNIVERSITY
FACULTY OF ENGINEERING
(CHS)

CMPN403: Compilers

PREPARED BY	Mahmoud Samy	ID	1200493
PREPARED BY	Islam Amr	ID	1200487
PREPARED BY	Malek Essam	ID	1200479
PREPARED BY	Moaaz Tarek	ID	1200871
SUBMITTED TO	Dr. Ayman AboElhassan		

Project Overview:

Our project aims to design and implement a programming language reminiscent of C that we called IM++, leveraging Lex and Yacc compiler-generating packages.

The primary objective of this endeavor is to create a language that encompasses fundamental programming paradigms and constructs commonly found in imperative programming languages. These include variable and constant declaration, arithmetic and logical expressions, control flow structures such as conditional statements (if-then-else), loops (while, repeat-until, for), switch statements, and the ability to define and call functions.

To ensure the language's robustness and usability, we are designing it with a block structure that supports nested scopes, allowing for the declaration of variables at the beginning of blocks. This feature enhances code modularity and encapsulation, promoting better code organization and management.

Furthermore, our language incorporates error handling mechanisms to detect and report syntax and semantic errors effectively. Additionally, our semantic analyzer checks for various issues, including variable declaration conflicts, improper variable usage concerning types, uninitialized and unused variables, thereby promoting code correctness and reliability.

The project will culminate in the creation of a functional programming environment where users can write code in our custom language, compile it, and receive feedback on syntax and semantic errors, as well as access to the symbol table and generated quadruples.

Tools & technologies:

- g++11
- Bison
- Flex
- Nodejs
- Reactjs

Tokens:

INTEGER	any number without decimal point
FLOATING	any number with decimal point
CHAR	Any one char in a single quotes
CHARARRAY	Any number of chars in double quotes
BOOLEAN	True or false
-+/*(){}=;&|~><:,	Each symbol is sent directly
WHILE	“while”
REPEAT	"repeat"
UNTIL	"until"
FOR	"for"
SWITCH	"switch"
CASE	"case"
IF	"if"
THEN	"then"
ELSE	"else"
FUNCTION	"function"
RETURN	"ret"
INT	"int"
FLOAT	"float"
BOOL	"bool"
CHARACTER	"char"
STRING	"string"
CONST	"const"
VOID	"void"
GE	">="

LE	"<="
EQ	"=="
NE	"!="
VARIABLE	Sequence of characters that start with any letter or underscore then accepts any letter, number or underscore

Quadruples:

PUSH X	Push value of X into stack
POP X	Assign value on top of the stack to X
JMP label	Continue execution from label;
ADD	Add 2 values on top of stack and push result
SUB	Subtract top from the value below it and push result
MUL	Multiply 2 values and push result
DIV	Divide second top value by the top value and push result
AND	Logical AND on top 2 values and push result
OR	Logical OR on top 2 values and push result
NOT	Logical NOT on values and push result
LT	Evaluate second top < first top and push result (1 or 0)
GT	Evaluate second top > first top and push result (1 or 0)
LE	Evaluate second top <= first top and push result (1 or 0)
GE	Evaluate second top >= first top and push result (1 or 0)
EQ	Evaluate second top == first top and push result (1 or 0)
NEQ	Evaluate second top != first top and push result (1 or 0)
NEG	Negate value on top of stack and push result
JZ label	If the result on top of stack is 0, continue execution from label and pop result.
CALL label	Push current address to stack and jump to label

RET	Jump to address in stack and pop it
RET $retvalue	Jump to address in stack and pop it, push return value
CAST int/float/bool	Convert value on top of stack to a specified type

WorkLoad division:

Mahmoud Samy	Quadruples, semantics, grammar, lexer, symbol table, GUI
Islam Amr	Quadruples, semantics, grammar, lexer, symbol table
Malek Essam	Quadruples, semantics, grammar, lexer, symbol table, GUI
Moaaz Tarek	Quadruples, semantics, grammar, lexer, symbol table, Casting