

C-Compiler

Team 9



*Team Members: -*

*Ahmed Mohamed Salah 1152114*

*Ahmed Shams 1124313*

*Mohamed Bassel 1152253*

*Mustafa Mufeed 1162249*

**Project Description**

Simple C language Compiler that is designed to compile main functions of C language components.  
the compiler is design is divided into two phases

**Phase-1**

1. Lex file that contains main tokens used in the project.
2. YACC file that contains the Production Rules used in the language with compilation indicators either the compiling was successful or not.
3. Simple C# windows application GUI to write and compile code. It also shows quadruples and symbol table

**Phase-2**

1. Symbol table
2. Semantic Analyzer
3. Enhanced GUI
4. Arrays Production Rules

**Technologies Used**

1. Flex: Compile LEX files
2. Bison: Compile YACC file
3. DEV-C: Generate Executable Files

**List of Tokens**

|  |  |
| --- | --- |
| OCBRACKET | { |
| CCBRACKET | } |
| ORBRACKET | ( |
| CRBRACKET | ) |
| SEMICOLON | ; |
| COLON | : |
| COMMA | , |
| PLUSEQUAL | += |
| MINUSEQUAL | -= |
| MULTIPLYEQUAL | \*= |
| DIVIDEEQUAL | /= |
| GREATERTHAN | > |
| LESSTHAN | < |
| GREATERTHANOREQUAL | >= |
| LESSTHANOREQUAL | =< |
| EQUALEQUAL | == |
| NOTEQUAL | != |
| PLUS | + |
| MINUS | - |
| MULTIPLY | \* |
| DIVIDE | / |
| EXPONENT | ^ |
| ASSIGN | = |
| REM | % |
| AND | && |
| OR | || |
| NOT | ! |
| WHILE | WHILE |
| FOR | FOR |
| IF | IF |
| ELSE | ELSE |
| PRINT | PRINT |
| BOOL | Data Type |
| INT | Data Type |
| CHAR | Data Type |
| STRING | Data Type |
| CONST | Data Type |
| DO | DO |
| BREAK | BREAK |
| SWITCH | SWITCH |
| CASE | CASE |
| FALSE | Boolean Value |
| TRUE | Boolean Value |
| RET | Return |

**Associativity List**

|  |  |
| --- | --- |
| ASSIGN | Left |
| GREATERTHAN | Left |
| LESSTHAN | Left |
| GREATERTHANOREQUAL | Left |
| LESSTHANOREQUAL | Left |
| EQUALEQUAL | Left |
| NOTEQUAL | Left |
| AND | Left |
| OR | Left |
| NOT | Left |
| PLUS | Left |
| MINUS | Left |
| DIVIDE | Left |
| MULTIPLY | Left |
| REM | Left |

**Production Rules**

Variables and Constants declaration.

Mathematical and logical expressions.

Assignment statement.

If-then-else statement, while loops, repeat-until loops, for loops, switch statement.

Block structure (nested scopes where variables may be declared at the beginning of blocks).

Functions.

**Quadruples**

|  |  |
| --- | --- |
| Quadruple | Description |
| MOV R0, 3 | Move value 3 to register R0 |
| MOV x, R0 | Move value in register R0 to variable x |
| ADD R2, R1, R0 | Add R1 to R0 and put sum in R2 |
| SUB R2, R1, R0 | Sub R1 from R0 and put difference in R2 |
| IMUL R2, R1, R0 | Multiply R1 by R0 and put product in R2 |
| DIV R2 | Divide R0 by R2 (R0/R2) |
| REM R2, R1, R0 | Find remainder of R1 and R0 and put it in R2 |
| INC R0 | Add 1 to R0 |
| DEC R0 | Sub 1 from R0 |
| JF Label | F false jump to ‘Label’ |
| CMPGEQ R2, R1, R0 | Compares R1 and R0 and the greater one or equal is put in R2 and changes flag |
| CMPEQ R2, R1, R0 |  |
| CMPNEQ R2, R1, R0 |  |
| AND R2, R1, R0 |  |
| OR R2, R1, R0 |  |
| PRINT R2 | Prints the value of register R2 |
|  |  |
|  |  |
|  |  |