

|  |  |
| --- | --- |
| Credit Hours System  Communication & Computer Engineering (CCE-C Track) | Cairo University Faculty of Engineering |

Language & Compilers

Project Report

|  |  |
| --- | --- |
|  | Submitted by: |
| Yasmine hatem | 11520509 |
| Farah mostafa | 1158066 |
| Nermine safwat ahmed | 1154228 |
| Ahmed khaled | 1152085 |

# LIST oF TOKENS

|  |  |  |
| --- | --- | --- |
| ***Pattern*** | ***Token*** | ***Explanation*** |
| **int** | INT | Int type |
| **float** | FLOAT | Float type |
| **char** | CHAR | Char type |
| **;** | SEMICOLON | Semicolon: end of statement |
| **:** | COLON | Colon |
| **+** | PLUS | Addition |
| **-** | MINUS | Subtraction |
| **\*** | MULTIPLY | Multiplication |
| **/** | DIVIDE | Division |
| **==** | COMPARE\_ EQUAL | Check if equal |
| **!=** | COMPARE\_NOT\_EQUAL | Not equal |
| **>=** | COMPARE\_GREATER\_EQUAL | Greater than or equal |

|  |  |  |
| --- | --- | --- |
| **<=** | COMPARE\_LESS\_EQUAL | Smaller than or equal |
| **>** | COMPARE\_GREATER | Greater than |
| **<** | COMPARE\_LESS | Smaller than |
| **if** | IF | Defining an If statement |
| **else** | ELSE | Defining “Else” |
| **for** | FOR | Defining a “for” loop |
| **while** | WHILE | Defining a “while” loop |
| **do** | DO | Do one statement regardless of any conditions that is mentioned later |
| **switch** | SWITCH | Switch on different cases |
| **case** | CASE | Different cases to switch on |
| **repeat** | REPEAT | Repeat a certain program |
| **until** | UNTIL | Repeat until a condition |

|  |  |  |
| --- | --- | --- |
| **=** | EQUALS | assign |
| **[0-9]+** | INTEGER | Values of integer |
| **((([0-9]+\.[0-9]\*)|(\.[0-9]+))([eE]?[+-]?[0-9]+)?)** | FLOAT | Values of float |
| **[a-zA-Z][\_a-zA-Z0-9]\*** | IDENTIFIER | Defining variable names |
| **[\'][a-zA-Z][\']** | CHARACTER | Values of char |
| **\"[\x23-\x7E \t !]\*\"** | STRING\_VALUE | Values of string |
| **!** | LOGICAL\_NOT | Not operator |
| **&&** | LOGICAL\_AND | And operator |
| **||** | LOGICAL\_OR | Or operator |
| **default** | DEFAULT | DEFAULT VALUE |
| **( )** | BRACKET\_OPEN BRACKET\_CLOSE | - |
| **{ }** | CURLY\_BRACE\_OPEN  CURLY\_BRACE\_CLOSE | - |
| **%** | REMAINDER |  |

DEFINING GRAMMER

PROGRAM: PROGRAM STATEMENT ;

STATEMENT : DECLARATION |

INTTYPE IDENTIFIER EQUALS MATH\_OR\_LOGICAL\_EXPRESSION SEMICOLON

|FLOATTYPE IDENTIFIER EQUALS MATH\_OR\_LOGICAL\_EXPRESSION SEMICOLON

|STRINGTYPE IDENTIFIER EQUALS CHARACTER SEMICOLON

|IDENTIFIER EQUALS MATH\_OR\_LOGICAL\_EXPRESSION SEMICOLON

| IF\_ELSE\_STAT

WHILE\_STAT |

FOR\_LOOP\_STAT |

REPEAT\_STAT |

SWITCH\_CASE |

JUMP\_STAT | IF\_STAT {

;

DECLARATION :EXP SEMICOLON

;

EXP:INTTYPE IDENTIFIER

| FLOATTYPE IDENTIFIER

| STRINGTYPE IDENTIFIER

| CONST INTTYPE IDENTIFIER EQUALS MATH\_OR\_LOGICAL\_EXPRESSION

CONST FLOATTYPE IDENTIFIER EQUALS MATH\_OR\_LOGICAL\_EXPRESSION {

| CONST STRINGTYPE IDENTIFIER EQUALS CHARACTER {

;

TYP: INTTYPE |FLOATTYPE|STRINGTYPE

;

IF\_STAT : IF BRACKET\_OPEN MATH\_OR\_LOGICAL\_EXPRESSION BRACKET\_CLOSE CURLY\_BRACE\_OPEN PROGRAM CURLY\_BRACE\_CLOSE

IF\_ELSE\_STAT : IF\_STAT ELSE CURLY\_BRACE\_OPEN PROGRAM CURLY\_BRACE\_CLOSE

FOR\_LOOP\_STAT : FOR BRACKET\_OPEN STATEMENT MATH\_OR\_LOGICAL\_EXPRESSION SEMICOLON IDENTIFIER EQUALS MATH\_OR\_LOGICAL\_EXPRESSION BRACKET\_CLOSE CURLY\_BRACE\_OPEN PROGRAM CURLY\_BRACE\_CLOSE

WHILE\_STAT : WHILE BRACKET\_OPEN MATH\_OR\_LOGICAL\_EXPRESSION BRACKET\_CLOSE CURLY\_BRACE\_OPEN PROGRAM CURLY\_BRACE\_CLOSE

| DO CURLY\_BRACE\_OPEN PROGRAM CURLY\_BRACE\_CLOSE WHILE BRACKET\_OPEN MATH\_OR\_LOGICAL\_EXPRESSION BRACKET\_CLOSE SEMICOLON

REPEAT\_STAT : REPEAT PROGRAM UNTIL MATH\_OR\_LOGICAL\_EXPRESSION SEMICOLON

MATH\_OR\_LOGICAL\_EXPRESSION : MATH\_OR\_LOGICAL\_EXPRESSION PLUS MATH\_OR\_LOGICAL\_EXPRESSION

MATH\_OR\_LOGICAL\_EXPRESSION : MATH\_OR\_LOGICAL\_EXPRESSION MINUS MATH\_OR\_LOGICAL\_EXPRESSION

MATH\_OR\_LOGICAL\_EXPRESSION : MATH\_OR\_LOGICAL\_EXPRESSION MULTIPLY MATH\_OR\_LOGICAL\_EXPRESSION

MATH\_OR\_LOGICAL\_EXPRESSION : MATH\_OR\_LOGICAL\_EXPRESSION DIVIDE MATH\_OR\_LOGICAL\_EXPRESSION

MATH\_OR\_LOGICAL\_EXPRESSION : MATH\_OR\_LOGICAL\_EXPRESSION REMAINDER MATH\_OR\_LOGICAL\_EXPRESSION | MATH\_OR\_LOGICAL\_EXPRESSION POWER MATH\_OR\_LOGICAL\_EXPRESSION

MATH\_OR\_LOGICAL\_EXPRESSION : MATH\_OR\_LOGICAL\_EXPRESSION LOGICAL\_AND MATH\_OR\_LOGICAL\_EXPRESSION

MATH\_OR\_LOGICAL\_EXPRESSION : MATH\_OR\_LOGICAL\_EXPRESSION LOGICAL\_OR MATH\_OR\_LOGICAL\_EXPRESSION

MATH\_OR\_LOGICAL\_EXPRESSION : LOGICAL\_NOT MATH\_OR\_LOGICAL\_EXPRESSION

MATH\_OR\_LOGICAL\_EXPRESSION : MATH\_OR\_LOGICAL\_EXPRESSION COMPARE\_EQUAL MATH\_OR\_LOGICAL\_EXPRESSION

MATH\_OR\_LOGICAL\_EXPRESSION : MATH\_OR\_LOGICAL\_EXPRESSION COMPARE\_NOT\_EQUAL MATH\_OR\_LOGICAL\_EXPRESSION

MATH\_OR\_LOGICAL\_EXPRESSION : MATH\_OR\_LOGICAL\_EXPRESSION COMPARE\_GREATER MATH\_OR\_LOGICAL\_EXPRESSION

MATH\_OR\_LOGICAL\_EXPRESSION : MATH\_OR\_LOGICAL\_EXPRESSION COMPARE\_LESS MATH\_OR\_LOGICAL\_EXPRESSION

MATH\_OR\_LOGICAL\_EXPRESSION : MATH\_OR\_LOGICAL\_EXPRESSION COMPARE\_GREATER\_EQUAL MATH\_OR\_LOGICAL\_EXPRESSION

MATH\_OR\_LOGICAL\_EXPRESSION : MATH\_OR\_LOGICAL\_EXPRESSION COMPARE\_LESS\_EQUAL MATH\_OR\_LOGICAL\_EXPRESSION

MATH\_OR\_LOGICAL\_EXPRESSION : BRACKET\_OPEN MATH\_OR\_LOGICAL\_EXPRESSION BRACKET\_CLOSE

: |NTEGER|FLOAT |IDENTIFIER

NUMBER :INTEGER

FLOAT

;

SWITCH\_CASE : SWITCH BRACKET\_OPEN IDENTIFIER BRACKET\_CLOSE CURLY\_BRACE\_OPEN CASES DEFAULT COLON PROGRAM CURLY\_BRACE\_CLOSE

CASES : CASE NUMBER COLON PROGRAM CASES

JUMP\_STAT : BREAK SEMICOLON

Quadruples

|  |  |  |
| --- | --- | --- |
| **Production** | **Example** | **Quadruples** |
| INTTYPE / FLOATTYPE IDENTIFIER EQUALS MATH\_OR\_LOGICAL\_EXPRESSION | **INT X =5;** | **Mov Ri,0**  **Mov x, Ri** |
|  | **INT X =5+6;** | **MOV R0, 5**  **MOV R1, 6**  **ADD R0, R1**  **MOV x, R0** |
|  | **INT X =10/5;** | **MOV R0, 10**  **MOV R1, 5**  **DIV R0, R1**  **MOV x, R0** |
|  | **INT X=10\*5;** | **MOV R0, 10**  **MOV R1, 5**  **MUL R0, R1**  **MOV X, R0** |
|  | **INT X=10/5;** | **MOV R0, 10**  **MOV R1, 5**  **DIV R0, R1**  **MOV X, R0** |
|  | **INT X=10^5;** |  |
|  | **INT X=10%5;** | **MOV R0, 10**  **MOV R1, 5**  **REM R0, R1**  **MOV x, R0** |
| REPEAT\_STAT : REPEAT PROGRAM UNTIL MATH\_OR\_LOGICAL\_EXPRESSION SEMICOLON | **int x=0;**  **repeat x=x+1;**  **until x<50;** | **MOV R0, 0**  **MOV x, R0**  **LABEL\_0:**  **MOV R0, x**  **MOV R1, 1**  **ADD R0, R1**  **MOV x, R0**  **MOV R0, x**  **MOV R1, 50**  **CMPL R0, R1**  **JZ LABEL\_1**  **JMP LABEL\_0**  **LABEL\_1:** |
| WHILE\_STAT : WHILE BRACKET\_OPEN MATH\_OR\_LOGICAL\_EXPRESSION BRACKET\_CLOSE CURLY\_BRACE\_OPEN PROGRAM CURLY\_BRACE\_CLOSE | **int z=0;**  **while(z<5)**  **{z=z+11;**  **}** | **MOV R0, 0**  **MOV z, R0**  **LABEL\_0:**  **MOV R0, z**  **MOV R1, 5**  **CMPL R0, R1**  **JNZ LABEL\_1**  **MOV R0, z**  **MOV R1, 11**  **ADD R0, R1**  **MOV z, R0**  **JMP LABEL\_0**  **LABEL\_1:** |
| IF\_STAT : IF BRACKET\_OPEN MATH\_OR\_LOGICAL\_EXPRESSION BRACKET\_CLOSE CURLY\_BRACE\_OPEN PROGRAM CURLY\_BRACE\_CLOSE | **int y=5;**  **if (y<6)**  **{y=3;}** | **MOV R0, 5**  **MOV y, R0**  **MOV R0, y**  **MOV R1, 6**  **CMPL R0, R1**  **JNZ LABEL\_0**  **MOV R0, 3**  **MOV y, R0**  **JMP LABEL\_1**  **LABEL\_0:** |
| SWITCH\_CASE : SWITCH BRACKET\_OPEN IDENTIFIER BRACKET\_CLOSE CURLY\_BRACE\_OPEN CASES DEFAULT COLON PROGRAM CURLY\_BRACE\_CLOSE  CASES : CASE NUMBER COLON PROGRAM CASES | **int x=3;**  **switch (x)**  **{ case 1 : x=x+1;**  **break;**  **case 2 : x=x+3;**  **break;**  **default : x=0;**  **}** | **MOV R0, 3**  **MOV x, R0**  **MOV RS1, x**  **CASE\_0:**  **MOV RS2, 1.000000**  **CMPE RS1, RS2**  **JNZ CASE\_1**  **MOV R0, x**  **MOV R1, 1**  **ADD R0, R1**  **MOV x, R0**  **JMP CASE\_DEFAULT**  **CASE\_1:**  **MOV RS2, 2.000000**  **CMPE RS1, RS2**  **JNZ CASE\_2**  **MOV R0, x**  **MOV R1, 3**  **ADD R0, R1**  **MOV x, R0**  **JMP CASE\_DEFAULT**  **CASE\_2:**  **CASE\_DEFAULT:**  **MOV R0, 0**  **MOV x, R0** |
| IF\_ELSE\_STAT : IF\_STAT ELSE CURLY\_BRACE\_OPEN PROGRAM CURLY\_BRACE\_CLOSE | **int y=5;**  **if (y<6)**  **{y=3;}**  **else {y=4;}** | **MOV R0, 5**  **MOV y, R0**  **MOV R0, y**  **MOV R1, 6**  **CMPL R0, R1**  **JNZ LABEL\_0**  **MOV R0, 3**  **MOV y, R0**  **JMP LABEL\_1**  **LABEL\_0:**  **MOV R0, 4**  **MOV y, R0**  **LABEL\_1:** |
| FOR\_LOOP\_STAT : FOR BRACKET\_OPEN STATEMENT MATH\_OR\_LOGICAL\_EXPRESSION SEMICOLON IDENTIFIER EQUALS MATH\_OR\_LOGICAL\_EXPRESSION BRACKET\_CLOSE CURLY\_BRACE\_OPEN PROGRAM CURLY\_BRACE\_CLOSE | **FOR( int x; x<5;x=x+1;)**  **{x=x+1;}** | **Label 0:**  **Mov R0,1**  **Mov x,R0**  **Mov R0,5**  **Mov R1,5**  **Cmpe R0,R1**  **Jne Label 1**  **Progrom**  **Jmp label 0**  **Label 1:**  **progrom** |