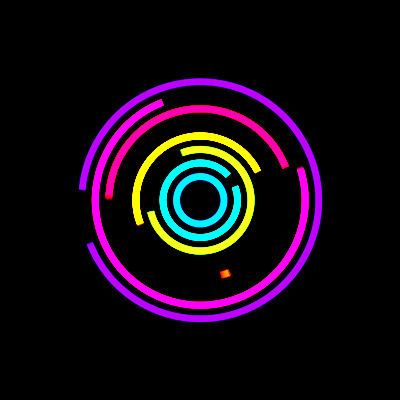
L E X I C A L

A N A L Y Z E R



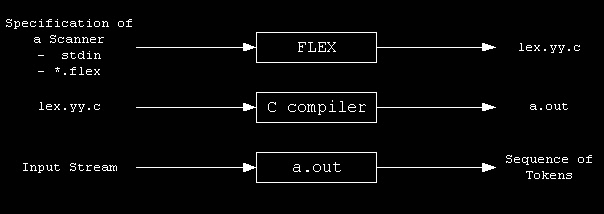
Layla Mohsen 900202391

Amr Khaled 900204834

Reem Said 900201275

We used the tool Flex (Fast Lexical Analyzer Generator) to generate our own lexical analyzer (scanner). This tool allows us to write the vocabulary of the language we want to scan instead of writing the scanner from scratch.

In order to specify our vocabulary for our language, we used regular expressions (RegEx) to write specifications of patterns accepted by our language. The process of Flex is as follows:



We wrote the following specifications in our “ass.l” file:

| // \*\*\* Please note this is not the full code, the full code will be in the ass.l file \*\*\*  //Definitions  KEYWORD (else|if|int|return|void|while)  SPECIAL\_SYMBOL ("\*"|"/"|"<"|"<="|">"|">="|"=="|"!="|"="|";"|","|"("|")"|"["|"]"|"{"|"}"|"/\*"|"\*/")  NON\_ALPHA ("`"|"!"|"@"|"#"|"%"|"$"|"?"|":"|"\*"|"^"|"~"|"/")  DIGIT [0-9]  LETTER [a-zA-Z]  DOT "."  HASH "#"  DOLLAR\_SIGN "$"  UNDER\_SCORE "\_"  EXP "E"  exp "e"  PLUS\_OP "+"  HYPHEN "-"  COMMENT "/\*"([^\*]|"\*"+[^\*/])\*"\*"+"/"  ECOMMENT "/"([^\*]|("\*"+[^\*/]))\*  %%  //Handling comments (Including Error 1 Handling)  {ECOMMENT} {printf("ERROR (1) Unclosed comment in line %d\n", line\_count);}  {COMMENT} {printf("COMMENT: %s\n", yytext);}  //Identifying Keywords  {KEYWORD} { printf("KEYWORD: %s\n", yytext); }  //Identifying Special Symbols  {SPECIAL\_SYMBOL} { printf("SPECIAL SYMBOL: %s\n", yytext); }  //Regular Expression for Accepted Identifiers  {LETTER}({LETTER}|{DIGIT})\*(({DOT}|{HASH}|{DOLLAR\_SIGN}|{UNDER\_SCORE})?({LETTER}|{DIGIT}+))? { printf("ID: %s\n", yytext); }  //Regular Expression for Accepted Numbers  ({DIGIT}+|{DIGIT}+{DOT}{DIGIT}\*)(({EXP}|{exp})({PLUS\_OP}|{HYPHEN})?{DIGIT}+)? { printf("NUM: %s\n", yytext); }  //Regular Expression for Rejected Identifiers (Error 3 Handling)  {LETTER}({LETTER}|{DIGIT})\*(({DOT}|{HASH}|{DOLLAR\_SIGN}|{UNDER\_SCORE}|{NON\_ALPHA})[^a-zA-Z0-9])\* {printf("ERROR (3) Wrong identfier in line %d : %s\n", line\_count , yytext); }  //Regular Expression for Rejected Numbers (Error 4 Handling)  ({DIGIT}+|{DIGIT}+{DOT}{DIGIT}\*)(({EXP}|{exp})({PLUS\_OP}|{HYPHEN})[^0-9])\* {printf("ERROR (4) Wrong number in line %d : %s\n", line\_count , yytext) ;line\_count++ ;}  //Regular Expression for Rejected Charcters not in the Alphabet (Error 2 Handling)  {NON\_ALPHA} { printf("ERROR (2) Charcter is not in the alphabet in line %d : %s\n", line\_count ,yytext);}  //Line counter to detect where error occurred  \n { line\_count++; } // Increment line count and character count on encountering a newline |
| --- |

The code above abides by the specifications mentioned in the assignment.

Our code also allows users to enter their input program to see if it is accepted by the language.

The following assumptions were made:

* Any character considered not in the alphabet of the language (NONALPHA) are restricted to: "`"|"!"|"@"|"#"|"%"|"$"|"?"|":"|"\*"|"^"|"~"|"/"
* Error handling was performed exactly like the asked examples in the assignment specifications and the erroneous token and its line are identified.

The following test input and corresponding Flex output are below:

INPUT:

| int main()  {  int a1, b1 ;  ?  int sum = 243e3 + 241 ;  /\* comment \*/  while ( x < 10 )  {  if ( x == 10 )  {  b1 = b1 + 1 ;  }  else  {  x = x + 1 ;  }  }  for ( int i = 0 ; i < 10 ; i = i + 1 )  {  sum = i + sum ;  }  12ab ;  12 ab ;  x = 242e+A ;  int a#?  /\* unclosed comment  } |
| --- |

Output:

| KEYWORD: int  ID: main  SPECIAL SYMBOL: (  SPECIAL SYMBOL: )  SPECIAL SYMBOL: {  KEYWORD: int  ID: a1  SPECIAL SYMBOL: ,  ID: b1  SPECIAL SYMBOL: ;  ERROR (2) Charcter is not in the alphabet in line 4 : ?  KEYWORD: int  ID: sum  SPECIAL SYMBOL: =  NUM: 243e3  + NUM: 241  SPECIAL SYMBOL: ;  COMMENT: /\* comment \*/  KEYWORD: while  SPECIAL SYMBOL: (  ID: x  SPECIAL SYMBOL: <  NUM: 10  SPECIAL SYMBOL: )  SPECIAL SYMBOL: {  KEYWORD: if  SPECIAL SYMBOL: (  ID: x  SPECIAL SYMBOL: ==  NUM: 10  SPECIAL SYMBOL: )  SPECIAL SYMBOL: {  ID: b1  SPECIAL SYMBOL: =  ID: b1  + NUM: 1  SPECIAL SYMBOL: ;  SPECIAL SYMBOL: }  KEYWORD: else  SPECIAL SYMBOL: {  ID: x  SPECIAL SYMBOL: =  ID: x  + NUM: 1  SPECIAL SYMBOL: ;  SPECIAL SYMBOL: }  SPECIAL SYMBOL: }  ID: for  SPECIAL SYMBOL: (  KEYWORD: int  ID: i  SPECIAL SYMBOL: =  NUM: 0  SPECIAL SYMBOL: ;  ID: i  SPECIAL SYMBOL: <  NUM: 10  SPECIAL SYMBOL: ;  ID: i  SPECIAL SYMBOL: =  ID: i  + NUM: 1  SPECIAL SYMBOL: )  SPECIAL SYMBOL: {  ID: sum  SPECIAL SYMBOL: =  ID: i  + ID: sum  SPECIAL SYMBOL: ;  SPECIAL SYMBOL: }  NUM: 12  ID: ab  SPECIAL SYMBOL: ;  NUM: 12  ID: ab  SPECIAL SYMBOL: ;  ID: x  SPECIAL SYMBOL: =  ERROR (4) Wrong number in line 24 : 242e+A  SPECIAL SYMBOL: ;  KEYWORD: int  ERROR (3) Wrong identfier in line 26 : a#?  ERROR (1) Unclosed comment in line 27  Total lines: 27 |
| --- |

Terminal commands:

| (base) layla@laylas-MacBook-Air ~ % flex /Users/layla/Desktop/ass.l  (base) layla@laylas-MacBook-Air ~ % gcc lex.yy.c  (base) layla@laylas-MacBook-Air ~ % ./a.out |
| --- |