

## Lab 8 – Lex

Specification:

```
%{
#include <math.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <stdio.h>
#
struct ProgramInternalFormNode {
    char key[100];
    char value[100];
};

struct SymbolTableNode {
    char key[100];
    int value;
};

struct ProgramInternalFormNode ProgramInternalForm[1000];
struct SymbolTableNode SymbolTable[100];

int k = 0;
int symbolTableIndex = 0;

char* insertIntoSymbolTable(char* text, int length){
    char key[100];

    strncpy(key, text, length);

    key[length] = '\0';

    for(int i = 0; i < symbolTableIndex; i++){
        if(!strcmp(SymbolTable[i].key, key)){
            char buffer[10];
            sprintf(buffer, "%d", SymbolTable[i].value);
            char* p = buffer;
            return p;
        }
    }

    struct SymbolTableNode node;
    strcpy(node.key, key);
    node.value = symbolTableIndex;
    SymbolTable[symbolTableIndex++] = node;

    char buffer[10];

    sprintf(buffer, "%d", SymbolTable[symbolTableIndex - 1].value);
```

```

    char* p = buffer;
    return p;
}
void printSymbolTable(){
    printf("\nSymbolTable\n");
    for(int i = 0; i < symbolTableIndex; i++){
        printf("%s %d\n", SymbolTable[i].key, SymbolTable[i].value);
    }
}

void insertIntoProgramInternalForm(char* text, char* value, int length){
    struct ProgramInternalFormNode node;
    strncpy(node.key, text, length);
    strcpy(node.value, value);
    node.key[length] = '\0';
    ProgramInternalForm[k++] = node;
}

void printProgramInternalForm() {
    printf("\nProgramInternalForm\n");
    for(int i = 0; i < k; i++)
        printf("%s %s\n", ProgramInternalForm[i].key, ProgramInternalForm[i].value);
}

%}

DIGIT          [0-9]
ID             [a-z][_a-z0-9]*
STRING        \'.*\'

%%

{DIGIT}+      {
                printf( "An integer: %s (%d)\n", yytext, atoi( yytext ));
                insertIntoProgramInternalForm("const",
insertIntoSymbolTable(yytext, yyleng ), 5);
            }

{DIGIT}+"."{DIGIT}* {
                                printf( "A float: %s (%g)\n", yytext, atof( yytext ) );
                                insertIntoProgramInternalForm("const",
insertIntoSymbolTable(yytext, yyleng), 5);
            }

"#" .* printf("A comment: \"%s\"\n", yytext);

"defvar"|"deflist"|"if"|"else"|"else if"|"and"|"or"|"not"|"loop" {
    printf( "A keyword: %s\n", yytext);
    insertIntoProgramInternalForm(yytext, "-1", yyleng);
}

```

```

{ID} {
    printf( "An identifier: %s\n", yytext );
    insertIntoProgramInternalForm("id", insertIntoSymbolTable( yytext, yyleng ), 2);
}

{STRING} {
    printf("A string: %s\n", yytext);
    insertIntoProgramInternalForm("const", insertIntoSymbolTable(yytext, yyleng), 5);
}

"+"|"-"|"*"|" "/"|"%"|"="|" "<"|" ">"|" "<="|" ">="|" "+"|" "-"|" ++|" --|" "**" {
    printf( "An operator: %s\n", yytext );
    insertIntoProgramInternalForm(yytext, "-1", yyleng);
}

";"|"["|"]|"{"|"}|"("|")|"|",|":|" {
    printf("A separator: %s\n", yytext);
    insertIntoProgramInternalForm(yytext, "-1", yyleng);
}

"{" "[^\\n]*" }          /* eat up one-line comments */

[ \\t\\n]+              /* eat up whitespace */

. printf("Error: %s\n", yytext);

%%
main( argc, argv )
int argc;
char **argv;
{
    ++argv, --argc; /* skip over program name */
    if ( argc > 0 )
        yyin = fopen( argv[0], "r" );
    else
        yyin = stdin;
    yylex();
    printProgramInternalForm();
    printSymbolTable();
}

```

Demo:

The image shows two terminal windows side-by-side, displaying C code for a lexical analyzer. The left window shows the main loop of the lexer, and the right window shows the internal data structures and functions.

**Left Terminal Window:**

```
calandrino@pop-os: ~/Documents/Repos/Formal-Languages-And-Compilers/Labs/Lex-Lab
A comment: "# Compute the minimum of 3 numbers"
A keyword: defvar
An identifier: a
A separator: ,
An identifier: b
A separator: ;
An identifier: c
A separator: ;
An identifier: input
A separator: (
A string: "a="
A separator: ,
An identifier: a
A separator: )
A separator: ;
An identifier: input
A separator: (
A string: "b="
A separator: ,
An identifier: b
A separator: )
A separator: ;
An identifier: input
A separator: (
A string: "c="
A separator: ,
An identifier: c
A separator: )
A separator: ;
A keyword: if
A separator: {
An operator: <
An identifier: b
A keyword: and
An identifier: a
An operator: <
An identifier: c
A separator: {
An identifier: print
A separator: (
An identifier: a
A separator: )
A separator: ;
A separator: ;
A keyword: else if
An identifier: b
An operator: <
An identifier: a
A keyword: and
An identifier: b
```

**Right Terminal Window:**

```
calandrino@pop-os: ~/Documents/Repos/Formal-Languages-And-Compilers/Labs/Lex-Lab
1
2 {
3 #include <math.h>
4 #include <stdlib.h>
5 #include <math.h>
6 #include <string.h>
7 #include <stdio.h>
8
9 struct ProgramInternalFormNode {
10     char key[100];
11     char value[100];
12 };
13
14 struct SymbolTableNode {
15     char key[100];
16     int value;
17 };
18
19 struct ProgramInternalFormNode ProgramInternalForm[1000];
20 struct SymbolTableNode SymbolTable[100];
21
22 int k = 0;
23 int symbolTableIndex = 0;
24
25 char* insertIntoSymbolTable(char* text, int length){
26     char key[100];
27
28     strncpy(key, text, length);
29
30     key[length] = '\0';
31
32     for(int i = 0; i < symbolTableIndex; i++){
33         if(!strcmp(SymbolTable[i].key, key)){
34             char buffer[10];
35             sprintf(buffer, "%d", SymbolTable[i].value);
36             char* p = buffer;
37             return p;
38         }
39     }
40
41     struct SymbolTableNode node;
42     strcpy(node.key, key);
43     node.value = symbolTableIndex;
44     SymbolTable[symbolTableIndex++] = node;
45
46     char buffer[10];
47
48     sprintf(buffer, "%d", SymbolTable[symbolTableIndex - 1].value);
49     NORMAL language_specification.lxi dos | utf-8 | conf 0% 1:0
50     "language_specification.lxi" [dos] 138L, 3232C
```

The image shows a dual-pane terminal window. The left pane displays a C program named `ProgramInternalForm` with various constants and conditional logic. The right pane displays the implementation of `ProgramInternalFormNode` and `SymbolTableNode` structures, along with functions for inserting text into a symbol table and printing the table.

```
Activities Terminal Jan 21 9:14 AM calandrinon@pop-os - /Documents/Repos/Format-Languages-And-Compilers/Labs/Lex-Lab
```

```
ProgramInternalForm
defvar -1
id 0
-1
id 1
-1
id 2
-1
id 3
-1
const 4
-1
id 0
-1
id 3
-1
const 5
-1
id 1
-1
id 3
-1
const 6
-1
id 2
-1
if -1
id 0
-1
and -1
id 0
-1
id 2
-1
id 7
-1
else if -1
id 1
-1
id 0
-1
and -1
```

```
1
2 {
3 #include <math.h>
4 #include <stdlib.h>
5 #include <math.h>
6 #include <string.h>
7 #include <stdio.h>
8 #
9 struct ProgramInternalFormNode {
10     char key[100];
11     char value[100];
12 };
13
14 struct SymbolTableNode {
15     char key[100];
16     int value;
17 };
18
19 struct ProgramInternalFormNode ProgramInternalForm[1000];
20 struct SymbolTableNode SymbolTable[100];
21
22 int k = 0;
23 int symbolTableIndex = 0;
24
25 char* insertIntoSymbolTable(char* text, int length){
26     char key[100];
27
28     strncpy(key, text, length);
29
30     key[length] = '\0';
31
32     for(int i = 0; i < symbolTableIndex; i++){
33         if(!strcmp(SymbolTable[i].key, key)){
34             char buffer[10];
35             sprintf(buffer, "%d", SymbolTable[i].value);
36             char* p = buffer;
37             return p;
38         }
39     }
40
41     struct SymbolTableNode node;
42     strcpy(node.key, key);
43     node.value = symbolTableIndex;
44     SymbolTable[symbolTableIndex++] = node;
45
46     char buffer[10];
47
48     sprintf(buffer, "%d", SymbolTable[symbolTableIndex - 1].value);
49     NORMAL language_specification.lxi dos | utf-8 | conf 0% 1/0
```

```
Activities Terminal Jan 12 9:14 AM
calandrinon@pop-os: ~/Documents/Repos/Formal-Languages-And-Compilers/Labs/Lex-Lab
; -1
if -1
id 0
< -1
id 1
and -1
id 0
< -1
id 2
{ -1
id 7
( -1
id 0
) -1
; -1
} -1
else if -1
id 1
< -1
id 0
and -1
id 1
< -1
id 2
{ -1
id 7
( -1
id 1
) -1
; -1
} -1
else -1
{ -1
id 7
( -1
id 2
) -1
; -1
} -1
} -1

SymbolTable
a 0
b 1
c 2
input 3
*a= 4
*b= 5
*c= 6
print 7

calandrinon@pop-os: ~/Documents/Repos/Formal-Languages-And-Compilers/Labs/Lex-Lab$

1
2 %{
3 #include <math.h>
4 #include <stdlib.h>
5 #include <math.h>
6 #include <string.h>
7 #include <stdio.h>
8 #
9 struct ProgramInternalFormNode {
10     char key[100];
11     char value[100];
12 };
13
14 struct SymbolTableNode {
15     char key[100];
16     int value;
17 };
18
19 struct ProgramInternalFormNode ProgramInternalForm[1000];
20 struct SymbolTableNode SymbolTable[100];
21
22 int k = 0;
23 int symbolTableIndex = 0;
24
25 char* insertIntoSymbolTable(char* text, int length){
26     char key[100];
27
28     strncpy(key, text, length);
29
30     key[length] = '\0';
31
32     for(int i = 0; i < symbolTableIndex; i++){
33         if(!strcmp(SymbolTable[i].key, key)){
34             char buffer[10];
35             sprintf(buffer, "%d", SymbolTable[i].value);
36             char* p = buffer;
37             return p;
38         }
39     }
40
41     struct SymbolTableNode node;
42     strcpy(node.key, key);
43     node.value = symbolTableIndex;
44     SymbolTable[symbolTableIndex++] = node;
45
46     char buffer[10];
47
48     sprintf(buffer, "%d", SymbolTable[symbolTableIndex - 1].value);
49     NORMAL language_specification.lxi
50     "language_specification.lxi" [dos] 138L, 3232C
dos | utf-8 | conf 8% 1:0
```