Lab 8 – Lex

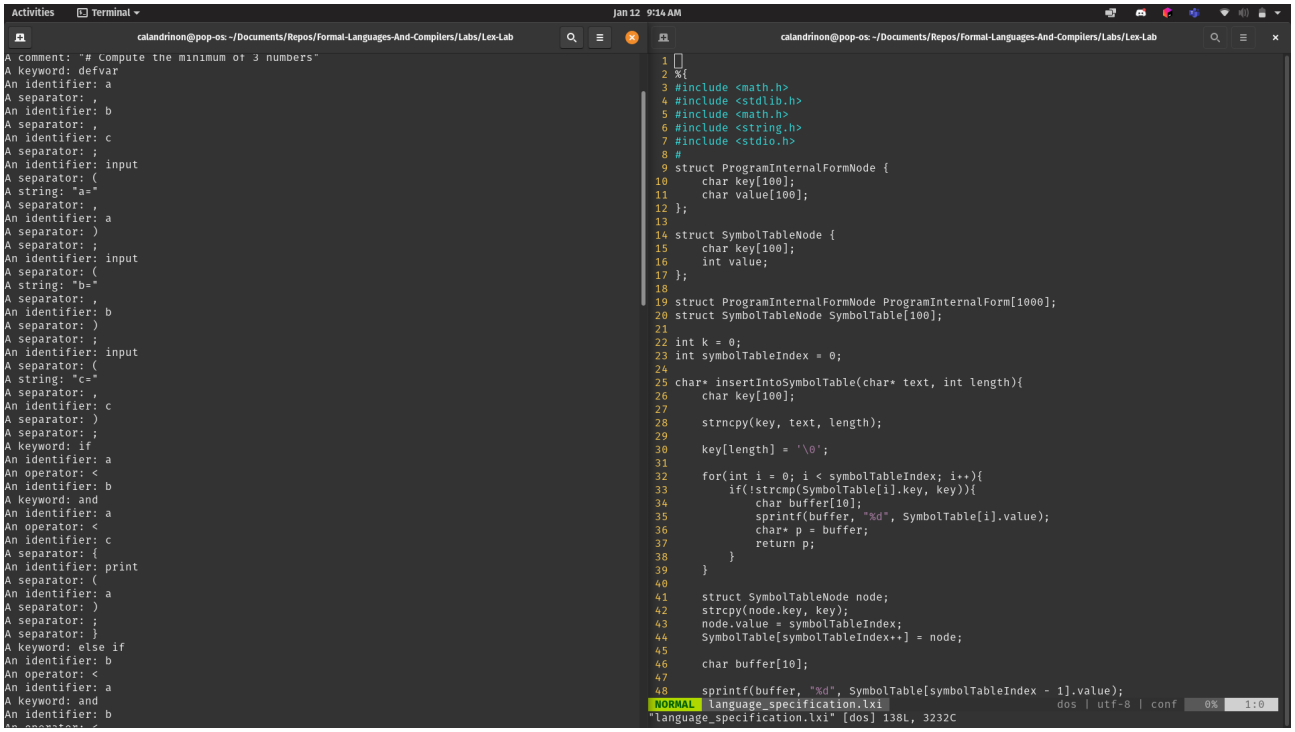
Specification:

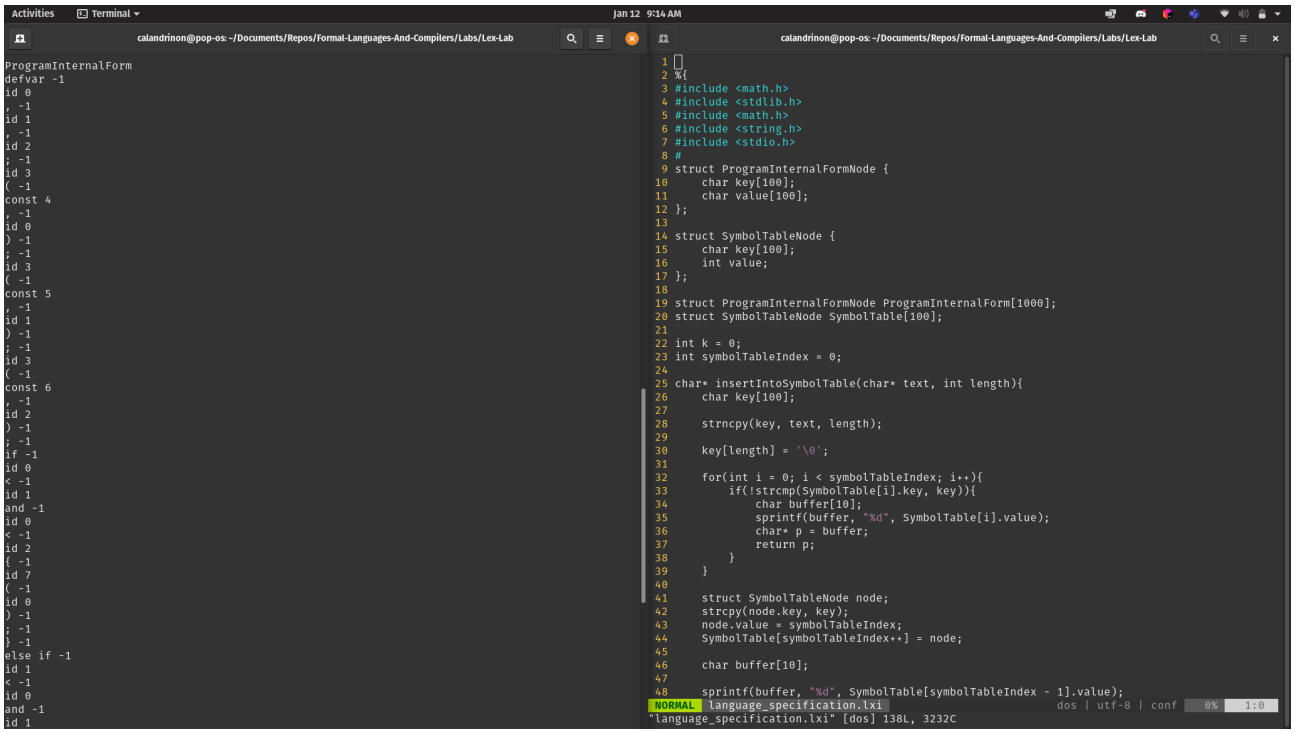
```
%{
#include <math.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <stdio.h>
#
struct ProgramInternalFormNode {
	char key[100];
	char value[100];
};

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);
```

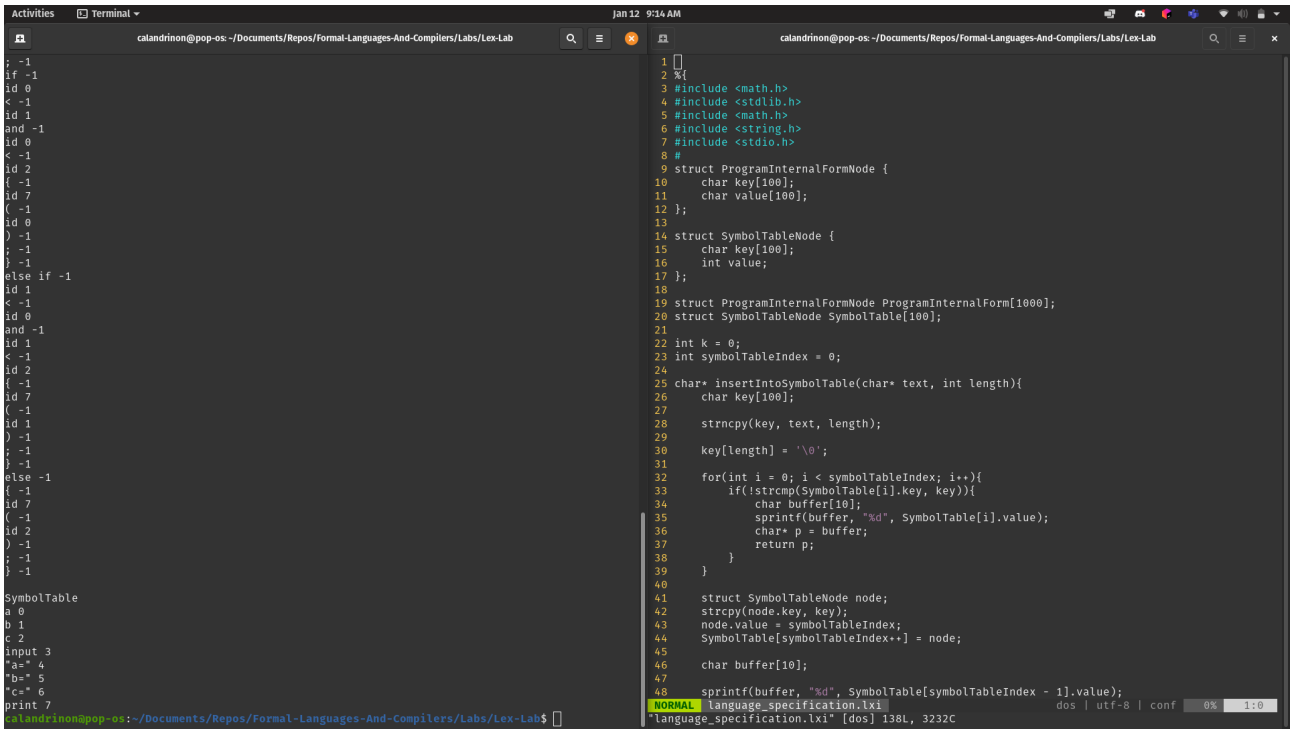
```
	char* p = buffer;
	return p;
	}
	void printSymbolTable(){
	printf("\nSymbolTable\n");
	for(int i = 0; i < symbolTableIndex; i++){
		printf("%s %d\n", SymbolTable[i].key, SymbolTable[i].value);
	}
}

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:


Activities
Terminal
Jan 12 9:14 AM
calandrinon@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
An identifier: a
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
%{
#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);
NORMAL language_specification.lxi
dos | utf-8 | conf 0% 1:0
"language_specification.lxi" [dos] 138L, 3232C


Activities
Terminal
Jan 12 9:14 AM
calandrinon@pop-os: ~/Documents/Repos/Formal-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
; -1
id 3
(-1
const 5
, -1
id 1
) -1
; -1
id 3
(-1
const 6
, -1
id 2
) -1
; -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
%{
#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);
NORMAL language_specification.lxi
dos | utf-8 | conf 0% 1:0
"language_specification.lxi" [dos] 138L, 3232C


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
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$
calandrinon@pop-os: ~/Documents/Repos/Formal-Languages-And-Compilers/Labs/Lex-Lab
%{
#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);
NORMAL language_specification.lxi
dos | utf-8 | conf 0% 1:0
"language_specification.lxi" [dos] 138L, 3232C