



Ahsanullah University of Science and Technology

Department of Computer Science & Engineering

Project Report

Course No.	CSE 4130
Course Name	Formal Languages and Compilers Lab

Submitted To:

Md. Aminur Rahman	Tawhidul Bhuiyan
Assistant Professor	Lecturer
Department of CSE, AUST	Department of CSE, AUST

Submitted By:

Name	Tahiya Ahmed Chowdhury
ID No.	17.02.04.048
Session	Fall – 2020
Section	A (A2)
Date of Submission:	October 10, 2021

Introduction:

In our Formal Languages and Compilers Lab course we had learnt elaborately about letters, tokens, symbols and identifiers in programming. By generating and implementing lexical analyzer, symbol tables, context-free languages, parsing, finite automatas we got a general idea of how a compiler works.

Objective:

The main objective of our project was to build a compiler by using and implementing the contents learnt in this course.

IDE used:

CodeBlocks version 20.03

Language used:

C

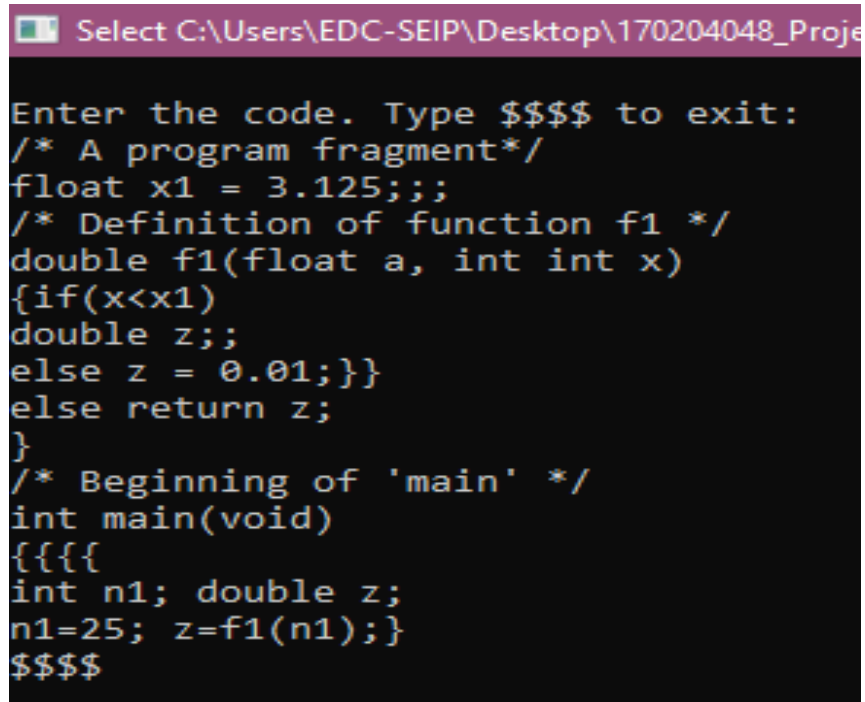
Functions used:

Function name	Usage
isDataType	Returns true if passed string is a c data type.
isNumericConstantDFA	Returns true if passed string is a numeric constant.
isDelimiter	Returns true if passed string is a delimiter.
isKeyword	Returns true if passed string is a keyword.
isOperator	Returns true if passed string is an operator.
isSeperator	Returns true if passed string is a seperator.
isParenthesis	Returns true if passed string is a parenthesis.

isBracket	Returns true if passed string is a second bracket.
isIdentifierDFA	Returns true if passed string is an identifier.
substring	Returns substring of a string passed.
detectTokens	Detects tokens of a string using above functions and stores it in a file.
generateSymbolTable	Generates symbol table with the identifiers identified from the substring.
removeDuplicate1	Symbol table is modified.
removeDuplicate2	Duplicate values from the symbol table is removed.
storeTable	Symbol table is stored.
insert	New value is inserted to the symbol table.
update	Symbol table is updated.
deleteRow	Symbol table row is deleted.
searchData	Data is searched in symbol table.
display	Symbol table is displayed.
main()	The main function where all the steps are coded accordingly and from where all other functions are called orderly.

Workflow:

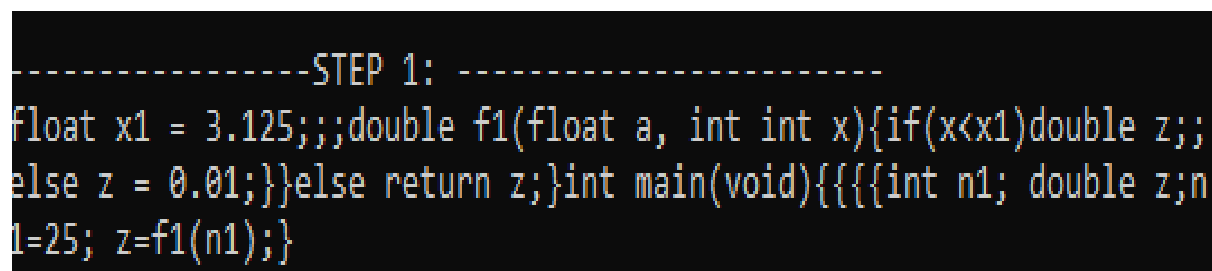
Initially the input code is typed to console:

A screenshot of a Windows command prompt window. The title bar shows the path "C:\Users\EDC-SEIP\Desktop\170204048_Proje". The prompt is "Enter the code. Type \$\$\$\$ to exit:". The user has entered the following C code:

```
/* A program fragment*/  
float x1 = 3.125;;;  
/* Definition of function f1 */  
double f1(float a, int int x)  
{if(x<x1)  
double z;;  
else z = 0.01;}}  
else return z;  
}  
/* Beginning of 'main' */  
int main(void)  
{{{{  
int n1; double z;  
n1=25; z=f1(n1);}  
$$$$
```

Step 1:

In the main function, the sample input code is scanned and stored inside a file which is then revisited to remove comments from the code.

A screenshot of a console window showing the output of Step 1. The text is as follows:

```
-----STEP 1: -----  
float x1 = 3.125;;;double f1(float a, int int x){if(x<x1)double z;;  
else z = 0.01;}}else return z;}int main(void){{{{int n1; double z;n  
1=25; z=f1(n1);}
```

Step 2:

After that the code is saved inside a string and passed onto detectTokens function which lists out all the words and assigns appropriate tokens to them.

```
-----STEP 2: -----  
[Keyword : float]  
[Identifier : x1]  
[Operator : =]  
[Numeric Constant : 3.125]  
[Seperator : ;]  
[Seperator : ;]  
[Seperator : ;]  
[Keyword : double]  
[Identifier : f1]  
[Parentheses : (]  
[Keyword : float]  
[Identifier : a]  
[Seperator : ,]  
[Keyword : int]  
[Keyword : int]  
[Identifier : x]  
[Parentheses : )]  
[Bracket : {]  
[Keyword : if]  
[Parentheses : (]  
[Identifier : x]  
[Operator : <]  
[Identifier : x1]  
[Parentheses : )]  
[Keyword : double]  
[Identifier : z]  
[Seperator : ;]  
[Seperator : ;]  
[Keyword : else]  
[Identifier : z]  
[Operator : =]  
[Numeric Constant : 0.01]  
[Seperator : ;]  
[Bracket : }]  
[Bracket : }]  
[Keyword : else]  
[Keyword : return]  
[Identifier : z]  
[Seperator : ;]  
[Bracket : }]
```

```
[Keyword : int]  
[Identifier : main]  
[Parentheses : (]  
[Keyword : void]  
[Parentheses : )]  
[Bracket : {]  
[Bracket : {]  
[Bracket : {]  
[Bracket : {]  
[Keyword : int]  
[Identifier : n1]  
[Seperator : ;]  
[Keyword : double]  
[Identifier : z]  
[Seperator : ;]  
[Identifier : n1]  
[Operator : =]  
[Numeric Constant : 25]  
[Seperator : ;]  
[Identifier : z]  
[Operator : =]  
[Identifier : f1]  
[Parentheses : (]  
[Identifier : n1]  
[Parentheses : )]  
[Seperator : ;]  
[Bracket : }]
```

Step 3:

Then from those tokens the identifiers are listed out and passed to the generateSymbol Table function. After scope,value,data type assignment to each identifiers, the removeDuplicate and removeDupliacte2 functions are used to reshape the symbol table for the next steps.

```
-----STEP 3: -----
[float][id x1][=][3.125][;][;][;][double][id f1][(][float][id a][,][int][int][id x][)][{][if][(][id x][<][id x1][)][double][id z][;][;][else][id z][=][0.01][;][{][}][else][return][id z][;][}][int][id main][(][void][)][{][{][{][{][int][id n1][;][double][id z][;][id n1][=][25][;][id z][=][id f1][(][id n1][)][;][}][}][}][}]
```

1.Insert
2.Update
3.Delete
4.Search
5.Display
6. exit
Select Operation: 5

SI No.	Name	Type	Data Type	Scope	Value
1	x1	var	float	global	3.125
2	f1	func	double	global	
3	a	var	float	f1	
4	x	var	int		
5	x1	var			
6	z	var	double		0.01
7	z	var		global	
8	main	func	int	global	
9	n1	var	int	main	25
10	z	var	double	main	
11	f1	func		main	

Step 4:

Now for error detection part, the next step is determining the undeclared variables, the scope column of symbolTable is used and thus the one with null scope is identified as undeclared. Finally separate loops were run for identifying unmatched brackets, if else blocks and duplicate assignment of data types. For the error detection part no separate function was used. This was coded inside the main block.

```
-----STEP 4: -----
1
2 float x1 = 3.125;;
3
4 double f1(float a, int int x)
5 {if(x<x1)
6 double z;;
7 else z = 0.01;}}
8 else return z;
9 }
10
11int main(void)
12{{{
13int n1; double z;
14n1=25; z=f1(n1);}

Undeclared variables:
x
x1
z

Duplicate token at line 2
Duplicate token at line 6
Unmatched } at line 7
Unmatched } at line 9
Unmatched } at line 15
Two or more datatypes in declaration specifier at line 4
Unmatched else at line 8
```

Conculsion:

Finally a simple compiler was made using raw logics from scratch.