Ahsanullah University of Science and Technology

Department of Computer Science and Engineering

CSE 4130

Project Report

Submitted to

Mr. Md. Aminur Rahman

Assistant Professor

Submitted by

Name: Julfikar Ibnul Haque

ID: 17.02.04.055

Section: A2

Semester: 4.1

Contents

1 Overview

In this project, a basic compiler has been implemented. We can input a code and then it will do some operations in four steps. These four steps are: removing comments, whitespaces and newlines from the code, tokenizing lexemes, showing symbol table and showing errors.

2 Taking the input

Inputs are taken through console. After typing the desired input code, tilde '~' character has to be inputted to stop taking input. This compiler doesn't work for library functions like: scanf, printf.

Sample input 1:

```
/* 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 =      0s.01;}}
else return   z;
}
/* Beginning    of 'main' */
int    main(void)
{{{{
int n1;
double z;
n1=25;
z=f1(n1);}
```

Sample input 2:

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

3 Step One: Removing comments, whitespaces and newlines

Functions used in this steps:

- **omitComments1():** This function is used to remove any single-line or multi-line comments from the code.
- **omitSpaces1():** This function is used to remove any whitespaces, newlines and tabs from the code.

Output for sample input 1:

```
||-------------------------------------------------------------------------------------||
||-----------------------------------  Step One  -----------------------------------||
       The code after removing the comments and white space (extra spaces, tabs and newline
 characters):

float x1 = 3.125;;;double f1(float a , int int x){if(x< x1)double z ;;else z = 0s.01;}}else
return z;}int main(void){{{{int n1; double z;n1=25; z=f1(n1);}
```

Output for sample input 2:

```
||-------------------------------------------------------------------------------------||
||-----------------------------------  Step One  -----------------------------------||
       The code after removing the comments and white space (extra spaces, tabs and newline characters):

float x1 = 3.125;double f1(int x){double z;z = 0.01;return z;} /int main(void){int n1; double z;n1=25; z=f1(n1);}
```

4 Step Two: Lexeme tokenization

Functions used in this steps:

- **lexemeTokenization2():** This function is used to generate token streams.

Output for sample input 1:

```
||-------------------------------------------------------------------------------------||
||-----------------------------------  Step Two  -----------------------------------||
       Lexeme Tokenized:

[kw float] [id x1] [op =] [num 3.125] [sep ;] [sep ;] [sep ;] [kw double] [id f1] [par (] [kw float] [id a] [sep ,] [kw int]
 [kw int] [id x] [par )] [brc {] [kw if] [par (] [id x] [op <] [id x1] [par )] [kw double] [id z] [sep ;] [sep ;] [kw else]
[id z] [op =] [unkn 0s.01] [sep ;] [brc }] [brc }] [kw else] [kw return] [id z] [sep ;] [brc }] [kw int] [id main] [par (] [
kw void] [par )] [brc {] [brc {] [brc {] [brc {] [kw int] [id n1] [sep ;] [kw double] [id z] [sep ;] [id n1] [op =] [num 25]
 [sep ;] [id z] [op =] [id f1] [par (] [id n1] [par )] [sep ;] [brc }]
```

Output for sample input 2:

```
||-------------------------------------------------------------------------------------||
||----------------------------------  Step Two  ----------------------------------||
       Lexeme Tokenized:

[kw float] [id x1] [op =] [num 3.125] [sep ;] [kw double] [id f1] [par (] [kw int] [id x] [par )] [brc {] [kw double]
 [id z] [sep ;] [id z] [op =] [num 0.01] [sep ;] [kw return] [id z] [sep ;] [brc }] [op /] [kw int] [id main] [par (]
 [kw void] [par )] [brc {] [kw int] [id n1] [sep ;] [kw double] [id z] [sep ;] [id n1] [op =] [num 25] [sep ;] [id z]
 [op =] [id f1] [par (] [id n1] [par )] [sep ;] [brc }]
```

5 Step Three: Symbol table

Functions used in this steps:

- **lexemeSimplify3():** Using this function, only identifiers are kept in pairs for formation of Symbol Tables.

Output for sample input 1:

```
||-------------------------------------------------------------------------------------||
||----------------------------------  Step Three  ----------------------------------||

---------------------------------  Symbol Table  ---------------------------------
Sl.No.          Name            Id 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             f1
5               x1              var             int             f1
6               z               var             double          f1
7               z               var             double          global
8               main            var             int             global
9               n1              var             int             global          25
```

Output for sample input 2:

```
||-------------------------------------------------------------------------------------||
||----------------------------------  Step Three  ----------------------------------||

---------------------------------  Symbol Table  ---------------------------------
Sl.No.          Name            Id Type         Data Type       Scope           Value
-------------------------------------------------------------------------------------
1               x1              var             float           global          3.125
2               f1              func            double          global
3               x               var             int             f1
4               z               var             double          f1              0.01
5               main            func            int             global
6               n1              var             int             main            25
7               z               var             double          main            0.01
```

6 Step Four: Finding errors

Functions used in this steps:

- **removeComments():** This function is used to remove any single-line or multi-line comments from the code.
- **removeSpaces():** This function is used to remove any whitespaces and tabs from the code.
- **lexemeSeparator():** This function is used to separate lexemes with single space.
- **lexemeTokenization():** This function is used to generate token streams.
- **lineNumberPrint():** This function is used to generate line numbers.
- **findErrors():** This function is used to find various errors such as: detection of simple syntax errors like duplication of tokens except parentheses or braces, unbalanced braces or parentheses problem, unmatched 'else' problem, unknown lexeme error etc.

Output for sample input 1:

```
||-------------------------------------------------------------------------------------||
||-----------------------------------  Step Four  -----------------------------------||

1:
2: kw float id x1 op = num 3.125 sep ; sep ; sep ;
3:
4: kw double id f1 par ( kw float id a sep , kw int kw int id x par )
5: brc { kw if par ( id x op < id x1 par )
6: kw double id z sep ; sep ;
7: kw else id z op = unkn 0s.01 sep ; brc } brc }
8: kw else kw return id z sep ;
9: brc }
10:
11: kw int id main par ( kw void par )
12: brc { brc { brc { brc {
13: kw int id n1 sep ;
14: kw double id z sep ;
15: id n1 op = num 25 sep ;
16: id z op = id f1 par ( id n1 par ) sep ; brc }


============== Errors ==============

Unmatched '}' at line: 7,
Unmatched '}' at line: 9,
Unmatched '{' at line: 12, 12, 12,
Duplicate token int at line 4
Duplicate 'Semicolon' at line 2
Duplicate 'Semicolon' at line 2
Duplicate 'Semicolon' at line 6
'Unknown' error at line 7
Unmatched 'else' at line 8
```

Output for sample input 2:

```
||--------------------------------------------------------------------------------------||
||-----------------------------------  Step Four  ----------------------------------||

1:
2: kw float id x1 op = num 3.125 sep ;
3:
4: kw double id f1 par ( kw int id x par )
5: brc {
6: kw double id z sep ;
7: id z op = num 0.01 sep ;
8: kw return id z sep ;
9: brc } op /
10:
11: kw int id main par ( kw void par )
12: brc {
13: kw int id n1 sep ;
14: kw double id z sep ;
15: id n1 op = num 25 sep ;
16: id z op = id f1 par ( id n1 par ) sep ;
17: brc }


============== Errors ==============
```