SEMANTIC ANALYZER FOR C LANGAUGE

Assignment 3

Date – 15th April 2020

Submitted to– Mrs. Khushboo Jain

Group members -

1. Arnav Doifode (BT17CSE009)
2. Pranav Rabade (BT17CSE014)
3. Amber Bhanarkar (BT17CSE022)
4. Shreyash Turkar (BT17CSE026)

Abstract:

This report contains the details of the tasks of Assignment 3 of Compilers course (mainly output for each input test cases). We have developed a Parser for C language which makes use of the C lexer to parse the given C input file. We used lexer to convert the input code into a stream of tokens which was provided to the parser. Parser matches the stream with the defined productions of the language. We used look-ahead for checking errors in comments and some other lexical errors. But lexical analyzer cannot detect errors in the structure of a language (syntax), unbalanced parenthesis etc. These errors are handled by a parser. But in syntax analysis phase, we don't check if the input is semantically correct. After parser checks if the code is structured correctly, semantic analysis phase checks if that syntax structure constructed in the source program derives any meaning or not. The output of the syntax analysis phase is parse tree whereas that of semantic phase is annotated parse tree.

We have mentioned some of the semantics errors that the semantic analyzer is expected to recognize:

1. Declaration of keyword/ identifier as variable name.
2. New declarations don't conflict with earlier defined declarations
3. Break/continue statements only appear within loops
4. Actual and formal parameter mismatch.
5. Arithmetic operations require integers/ same data types.

The source code is also available on the GitHub repository of Amber Bhanarkar (BT17CSE022)

https://github.com/amberbhanarkar/Semantic-Analyzer

Explanation:

The lex code is detecting the tokens from the source code and returning the corresponding token to the parser. We are using the symbol table and constant table. We have used functions like insertSTnest(), insertSTparamscount(), checkscope(), deletedata(), duplicate() etc., in order to check the semantics . In the production rules of the grammar semantic actions are written and these are performed by the functions listed above.

Declaration Section

In this section we have included all the necessary header files, function declaration and flag that was needed in the code. Between declaration and rules section we have listed all the tokens which are returned by the lexer according to the precedence order. We also declared the operators here according to their associativity and precedence. This ensures the grammar we are giving to the parser is unambiguous as LALR(1) parser cannot work with ambiguous grammar.

Rules Section

In this section production rules for entire C language is written. The grammar productions does the syntax analysis of the source code. When a complete statement with proper syntax is matched by the parser. Along with rules semantic actions associated with the rules are also written and corresponding functions are called to do the necessary actions.

C-Program Section

In this section the parser links the extern functions, variables declared in the lexer, external files generated by the lexer etc. The main function takes the input source code file and prints the final symbol table.

To run the program, open the terminal in the folder and type

./run.sh or bash run.sh

The 5 cases which were given in assignment on which the semantic analysis was to be performed are tested on

- **Test Case 6 –** Cannot use reserved keyword/ identifier
- **Test Case 17–** Arithmetic operations requires integers
- **Test Case 14–** New declarations does not conflict with earlier ones
- **Test Case 10–** Break and continue statements appear only inside loops
- **Test Case 12–** Actual and formal parameters types need to be compatible.

The input for each test case is given with its corresponding output.

We have printed a symbol table and a constant table in case a program does not have a semantic error.

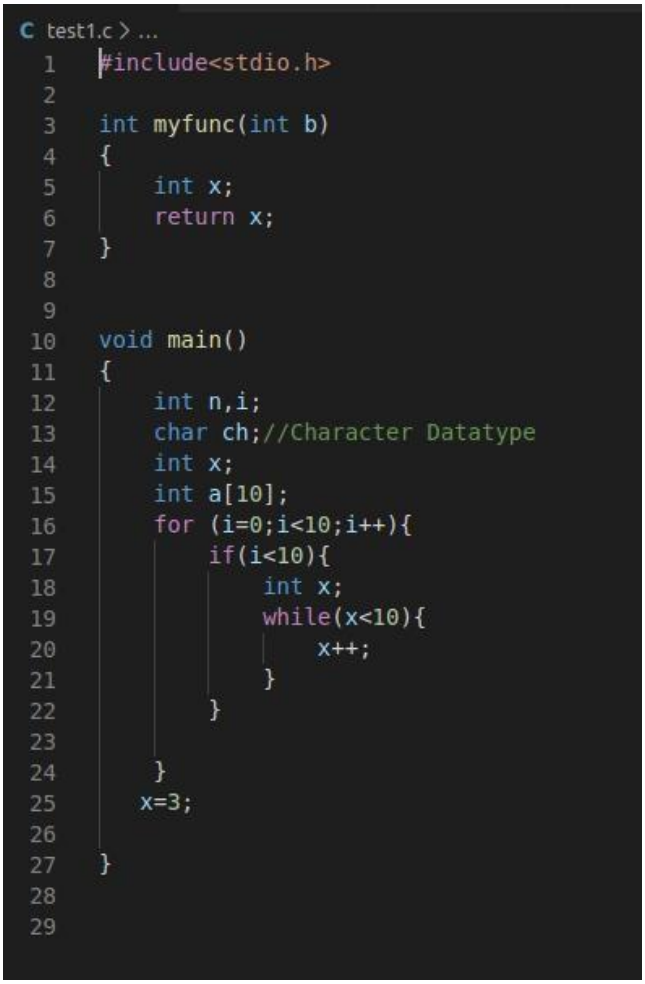
For test cases having semantic error, appropriate error message is displayed along with error line number.

The programs are written and tested on Ubuntu 18.04

Test Cases

Test Case 1

I/P program



Output – **PASS**

```
amber@amber-HP-Pavilion-Notebook: ~/Compilers/Lab3/Semantic Analyzer
File  Edit  View  Search  Terminal  Help
(base) amber@amber-HP-Pavilion-Notebook:~/Compilers/Lab3/Semantic Analyzer$ ./run.sh
Running: 18


================================  Running TestCase 1  ================================
Status: Parsing Complete - Valid
                          SYMBOL TABLE
                           ------------
    SYMBOL |           CLASS |       TYPE |      VALUE |    LINE NO |         NESTING | PARAMS COUNT |
-----------------------------------------------------------------------------------------------------
         a | Array Identifier |        int |            |         15 |           99999 |          -1 |
         b |      Identifier |        int |            |          3 |           99999 |          -1 |
         i |      Identifier |        int |         10 |         12 |           99999 |          -1 |
         n |      Identifier |        int |            |         12 |           99999 |          -1 |
         x |      Identifier |        int |            |          5 |           99999 |          -1 |
         x |      Identifier |        int |          3 |         14 |           99999 |          -1 |
         x |      Identifier |        int |         10 |         18 |           99999 |          -1 |
       for |         Keyword |            |            |         16 |            9999 |          -1 |
      char |         Keyword |            |            |         13 |            9999 |          -1 |
        ch |      Identifier |       char |            |         13 |           99999 |          -1 |
    return |         Keyword |            |            |          6 |            9999 |          -1 |
        if |         Keyword |            |            |         17 |            9999 |          -1 |
       int |         Keyword |            |            |          3 |            9999 |          -1 |
      main |        Function |       void |            |         10 |            9999 |          -1 |
    myfunc |        Function |        int |            |          3 |            9999 |           1 |
     while |         Keyword |            |            |         19 |            9999 |          -1 |
      void |         Keyword |            |            |         10 |            9999 |          -1 |


                          CONSTANT TABLE
                           -------------
      NAME |            TYPE
-------------------------------------------------------------------------------
        10 | Number Constant
         0 | Number Constant
         3 | Number Constant
```

Test Case 2

Input -

```c
#include<stdio.h>



void main(){
    int a,b,c,d,e,f,g,h;

    c=a+b;
    d=a*b;
    e=a/b;
    f=a%b;
    g=a&&b;
    h=a||b;
    h=a*(a+b);
    h=a*a+b*b;


}

```

Output – Pass

```
================================  Running TestCase 2  ================================
Status: Parsing Complete - Valid
                        SYMBOL TABLE
                         ------------
    SYMBOL |           CLASS |       TYPE |      VALUE |    LINE NO |         NESTING | PARAMS COUNT |
-------------------------------------------------------------------------------------------------------
         a |      Identifier |        int |            |          6 |           99999 |          -1 |
         b |      Identifier |        int |            |          6 |           99999 |          -1 |
         c |      Identifier |        int |            |          6 |           99999 |          -1 |
         d |      Identifier |        int |            |          6 |           99999 |          -1 |
         e |      Identifier |        int |            |          6 |           99999 |          -1 |
         f |      Identifier |        int |            |          6 |           99999 |          -1 |
         g |      Identifier |        int |            |          6 |           99999 |          -1 |
         h |      Identifier |        int |            |          6 |           99999 |          -1 |
       int |         Keyword |            |            |          6 |            9999 |          -1 |
      main |        Function |       void |            |          5 |            9999 |          -1 |
      void |         Keyword |            |            |          5 |            9999 |          -1 |


                        CONSTANT TABLE
                         --------------
      NAME |            TYPE
-------------------------------------------------------------------------------
```

Test case 3

Input -

```c
#include<stdio.h>

int main()
{
    int a = 5;
    while(a>0)
    {
        printf("Hello world");
        a--;
    }

    a=4;
    while(a>0)
    {
        printf("%d",a);
        a--;
        int b;
        b= 4;
        while(b>0)
        {
            printf("%d", a*b);
            b--;
        }
    }
}
```

Output – Pass

```
================================  Running TestCase 3  ================================
Status: Parsing Complete - Valid
                            SYMBOL TABLE
                             ------------
    SYMBOL |           CLASS |       TYPE |      VALUE |    LINE NO |         NESTING | PARAMS COUNT |
-------------------------------------------------------------------------------------------------------
         a |      Identifier |        int |          0 |          5 |           99999 |           -1 |
         b |      Identifier |        int |          0 |         17 |           99999 |           -1 |
       int |         Keyword |            |            |          3 |            9999 |           -1 |
      main |        Function |        int |            |          3 |            9999 |           -1 |
    printf |        Function |            |            |          8 |            9999 |           -1 |
     while |         Keyword |            |            |          6 |            9999 |           -1 |


                            CONSTANT TABLE
                             --------------
      NAME |            TYPE
-----------------------------------------------------------------------------
"Hello world" | String Constant
       "%d" | String Constant
          0 | Number Constant
          4 | Number Constant
          5 | Number Constant
```

Test Case 4 –

Input

test4.c > ...

```c
#include<stdio.h>

int main()
{
    int a = 2;
    printf("%d",a);
    a++;
    int b = 4;
    int c = 3;

    //int b = 8;
    //int c = 3;
    a--;
}

```

Output – Pass

```
================================  Running TestCase 4  ================================
Status: Parsing Complete - Valid
                            SYMBOL TABLE
                             ------------
    SYMBOL |           CLASS |       TYPE |      VALUE |    LINE NO |         NESTING | PARAMS COUNT |
-------------------------------------------------------------------------------------------------------
         a |      Identifier |        int |          2 |          5 |           99999 |          -1 |
         b |      Identifier |        int |          4 |          8 |           99999 |          -1 |
         c |      Identifier |        int |          3 |          9 |           99999 |          -1 |
       int |         Keyword |            |            |          3 |            9999 |          -1 |
      main |        Function |        int |            |          3 |            9999 |          -1 |
    printf |        Function |            |            |          6 |            9999 |          -1 |


                            CONSTANT TABLE
                             -------------
      NAME |           TYPE
-----------------------------------------------------------------------------
      "%d" | String Constant
         2 | Number Constant
         3 | Number Constant
         4 | Number Constant
```

Test Case 5

Input –

```c
#include<stdio.h>
#define NUM 5

int main()
{
char A[] = "#define MAX 10";
char B[ ] = "Hello";
char ch  = 'B';
unsigned int a = 1;
printf("String = %s Value of Pi = %f", 3.14);

    return 0;
}

```

Output – Pass

```
================================  Running TestCase 5  ================================
Status: Parsing Complete - Valid
                            SYMBOL TABLE
                             ------------
    SYMBOL |           CLASS |       TYPE |      VALUE |    LINE NO |         NESTING | PARAMS COUNT |
-------------------------------------------------------------------------------------------------------
         A | Array Identifier |       char | "#define MAX 10" |          6 |           99999 |         -1 |
         B | Array Identifier |       char |    "Hello" |          7 |           99999 |         -1 |
  unsigned |         Keyword |            |            |          9 |            9999 |         -1 |
         a |      Identifier |        int |          1 |          9 |           99999 |         -1 |
      char |         Keyword |            |            |          6 |            9999 |         -1 |
        ch |      Identifier |       char |        'B' |          8 |           99999 |         -1 |
    return |         Keyword |            |            |         12 |            9999 |         -1 |
       int |         Keyword |            |            |          4 |            9999 |         -1 |
      main |        Function |        int |            |          4 |            9999 |         -1 |
    printf |        Function |            |            |         10 |            9999 |         -1 |


                            CONSTANT TABLE
                             --------------
      NAME |            TYPE
-------------------------------------------------------------------------------
       'B' | Character Constant
"#define MAX 10" | String Constant
   "Hello" | String Constant
"String = %s Value of Pi = %f" | String Constant
      3.14 | Floating Constant
         0 | Number Constant
         1 | Number Constant
```

Test Case 6 –

Input –

```c
#include<stdio.h>
void main()
{
    int if;
    if = 2;
}
```

Output – Failed

Since, we have declared a reserved keyword ‘if’ as a variable, this gives an error.

```
================================  Running TestCase 6  ================================
4 syntax error if
Status: Parsing Failed - Invalid
```

Test Case 7

Input –

```
C test7.c > ...
//for loop
//continue
//while loop
//do while loop

#include<stdio.h>

int main()
{
    int a=0;
    for (a = 0; a < 10; a++)
        {
            printf("H1");
        }

    while(a>0) {
        a--;
    }

    do {
        a++;
    }while(a<10);
}
```

Output – Pass

```
====================================  Running TestCase 7  ====================================
Status: Parsing Complete - Valid
                              SYMBOL TABLE
                               ------------
    SYMBOL |           CLASS |        TYPE |      VALUE |     LINE NO |          NESTING | PARAMS COUNT |
---------------------------------------------------------------------------------------------------------
         a |      Identifier |         int |         10 |          10 |            99999 |           -1 |
       for |         Keyword |             |            |          11 |             9999 |           -1 |
        do |         Keyword |             |            |          20 |             9999 |           -1 |
       int |         Keyword |             |            |           8 |             9999 |           -1 |
      main |        Function |         int |            |           8 |             9999 |           -1 |
    printf |        Function |             |            |          13 |             9999 |           -1 |
     while |         Keyword |             |            |          16 |             9999 |           -1 |


                              CONSTANT TABLE
                               --------------
      NAME |            TYPE
-----------------------------------------------------------------------------------
      "H1" | String Constant
        10 | Number Constant
         0 | Number Constant
```

Test Case 8 –

Input –

test8.c > ...

```c
// Mixture of tests, comments

#include<stdio.h>
int main()
{
    int a, b;
    char c;
    for(a = 0; a < 29; a++)
    {
        if(a < 15) {
            printf("Hello World");
        }
    }
    int x = a + b;
    // Single Line Comment
    /* This is a
       multi-line comment */

    int var1;
    char var2;
    printf("%d",x);
}

```

Output – Pass

```
================================  Running TestCase 8  ================================
Status: Parsing Complete - Valid
                          SYMBOL TABLE
                           ------------
    SYMBOL |           CLASS |       TYPE |      VALUE |    LINE NO |         NESTING | PARAMS COUNT |
-------------------------------------------------------------------------------------------------------
         a |      Identifier |        int |         29 |          6 |           99999 |         -1 |
         b |      Identifier |        int |            |          6 |           99999 |         -1 |
         c |      Identifier |       char |            |          7 |           99999 |         -1 |
         x |      Identifier |        int |            |         14 |           99999 |         -1 |
       for |         Keyword |            |            |          8 |            9999 |         -1 |
      char |         Keyword |            |            |          7 |            9999 |         -1 |
        if |         Keyword |            |            |         10 |            9999 |         -1 |
       int |         Keyword |            |            |          4 |            9999 |         -1 |
      main |        Function |        int |            |          4 |            9999 |         -1 |
      var1 |      Identifier |        int |            |         18 |           99999 |         -1 |
      var2 |      Identifier |       char |            |         19 |           99999 |         -1 |
    printf |        Function |            |            |         11 |            9999 |         -1 |


                          CONSTANT TABLE
                           --------------
      NAME |            TYPE
-----------------------------------------------------------------------------
"Hello World" | String Constant
      "%d" | String Constant
        15 | Number Constant
        29 | Number Constant
         0 | Number Constant
```

Test Case 9 –

Input –

test9.c > ...

```c
// Implicit Error that our Language doesn't support

#include<stdio.h>

int main() {
    char @hello;
    @hello = 'c';
}

```

Output – Failed

```
==================================  Running TestCase 9  ==================================
ERROR at line no. 6
@
6 syntax error @
Status: Parsing Failed - Invalid
```

Test Case 10 –

Input –

```c
#include <stdio.h>
int main()
{
    int i;
    for(i=0;i<5;i++)
    {
        if(i==3){
        }
    }
    continue;
}
```

Output – Failed

Continue outside for loop

```
==================================  Running TestCase 10  ==================================
continue
Undeclared
```

Test Case 11 –

Input –

```c
#include<stdio.h>
int square(int a, int b)
{
    int b = 2;
    return b;
}

int main()
{
    int num = 2;
    int num2;
    square(num,num);

    //printf("Square of %d is %d", num, square2(5));

    return 0;
}
```

Output – Failed

```
==================================  Running TestCase 11  ==================================
Duplicate
```

Test Case 12 –

Input –

```c
#include <stdio.h>
int func(char a)
{
    return a;
}

int main()
{
    int z = 5, ll;
    //int m = 6;
    ll = func(z);
    printf("%d", ll);
}
```

Output – Failed

Since we have passed an int from main but used char in function.

```
==================================  Running TestCase 12  ==================================
Type mismatch
```

Test Case 13 –

Input –

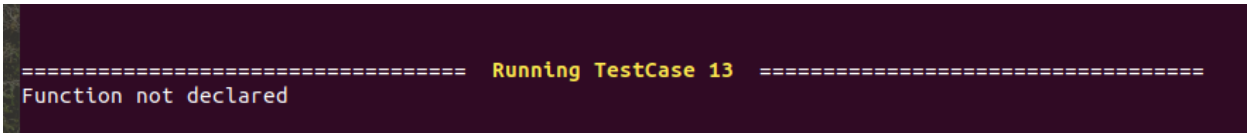
```c
#include<stdio.h>

void main()
{
    int i,n;

    myfunc(i);

}
```

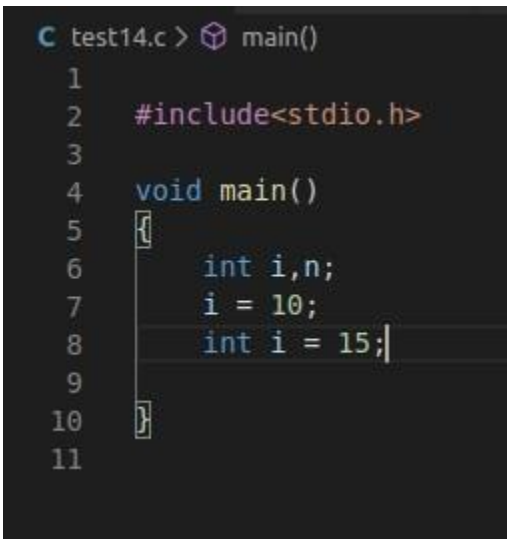
Output – Failed



```
==================================  Running TestCase 13  ==================================
Function not declared
```

Test Case 14 –

Input –



```c
#include<stdio.h>

void main()
{
    int i,n;
    i = 10;
    int i = 15;

}
```

Output – Failed

Since we have declared same variable twice. This gives an error. New declaration conflicted with earlier ones.

```
==================================  Running TestCase 14  ==================================
Duplicate
```

Test Case 15 –

Input –

test15.c

```c
#include<stdio.h>

int myfunc(int a)
{
    return a;
}

void main()
{
    int i,n;

    myfunc(i,n);

}
```

Output – Failed

Number of parameters does not match

```
==================================  Running TestCase 15  ==================================
12 Number of arguments in function call doesn't match number of parameters )
Status: Parsing Failed - Invalid
```

Test Case 16-

Input –

```c
#include<stdio.h>

void main()
{
    int i=3,n=6;
    float a=0.0;
    a = i+n;
}

```

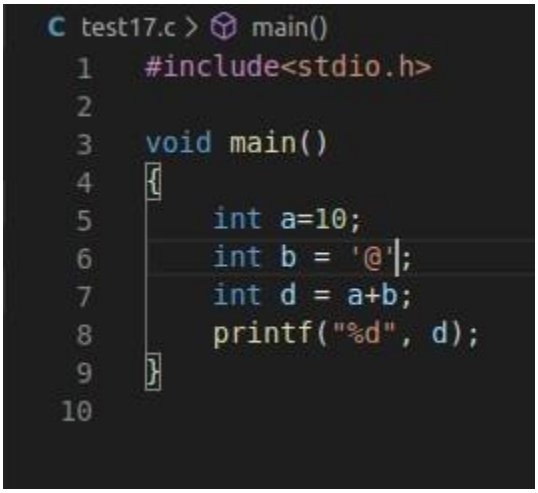
Output –

Since we are performing arithmetic operations on two different data types. Data type mismatch.

```
==================================  Running TestCase 16  ==================================
Type mismatch
```

Test Case 17 –

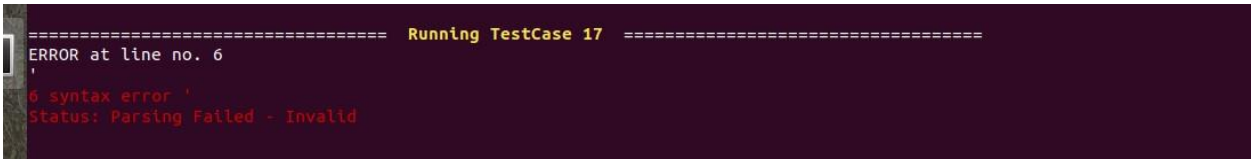
Input –



```c
#include<stdio.h>

void main()
{
    int a=10;
    int b = '@';
    int d = a+b;
    printf("%d", d);
}
```

Output – Failed



```
================================  Running TestCase 17  ================================
ERROR at line no. 6
'
6 syntax error '
Status: Parsing Failed - Invalid
```

Test Case 18 –

Input –

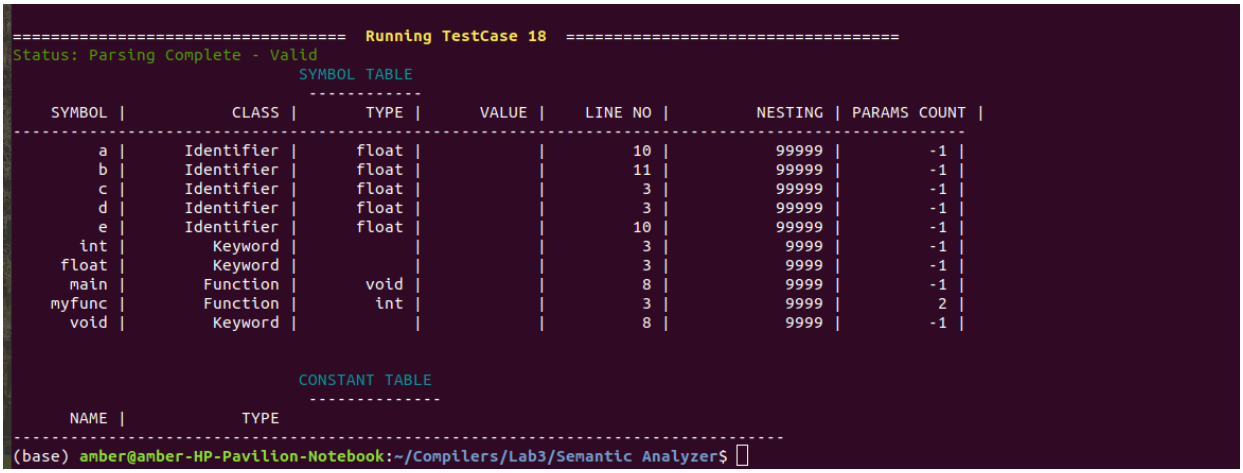
```c
#include<stdio.h>

int myfunc(float c, float d)
{

}

void main()
{
    float a,e;
    float b;
    myfunc(e, b);
}
```

Output – Passed



```
================================  Running TestCase 18  ================================
Status: Parsing Complete - Valid
                            SYMBOL TABLE
                             ------------
    SYMBOL |           CLASS |       TYPE |      VALUE |    LINE NO |         NESTING | PARAMS COUNT |
-------------------------------------------------------------------------------------------------------
         a |      Identifier |      float |            |         10 |           99999 |         -1 |
         b |      Identifier |      float |            |         11 |           99999 |         -1 |
         c |      Identifier |      float |            |          3 |           99999 |         -1 |
         d |      Identifier |      float |            |          3 |           99999 |         -1 |
         e |      Identifier |      float |            |         10 |           99999 |         -1 |
       int |         Keyword |            |            |          3 |            9999 |         -1 |
     float |         Keyword |            |            |          3 |            9999 |         -1 |
      main |        Function |       void |            |          8 |            9999 |         -1 |
    myfunc |        Function |        int |            |          3 |            9999 |          2 |
      void |         Keyword |            |            |          8 |            9999 |         -1 |


                            CONSTANT TABLE
                             -------------
      NAME |            TYPE
-----------------------------------------------------------------------------
(base) amber@amber-HP-Pavilion-Notebook:~/Compilers/Lab3/Semantic Analyzer$
```