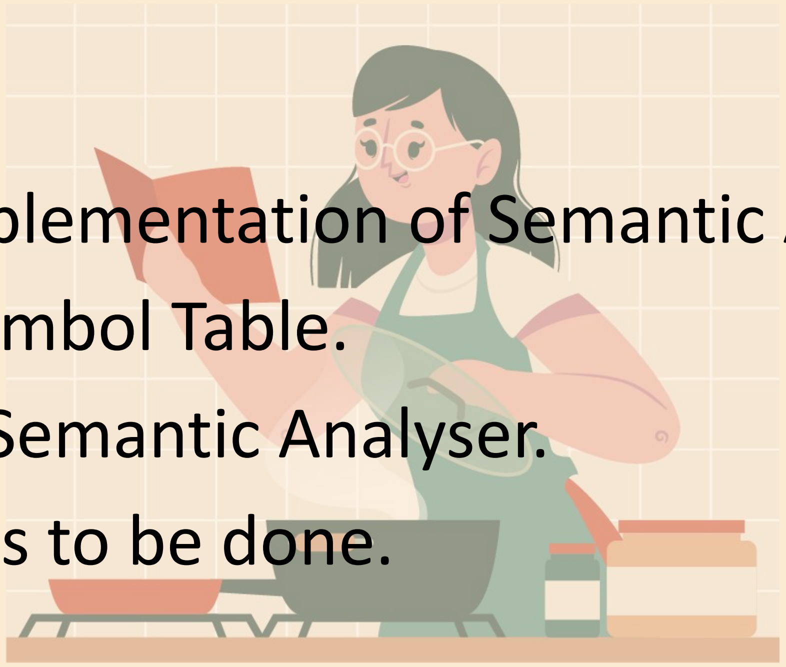
MICHELIN

A Code which Cooks

Michelin's Semantic Analyser

Contents

At this phase of the project, few important things to take a note of are:

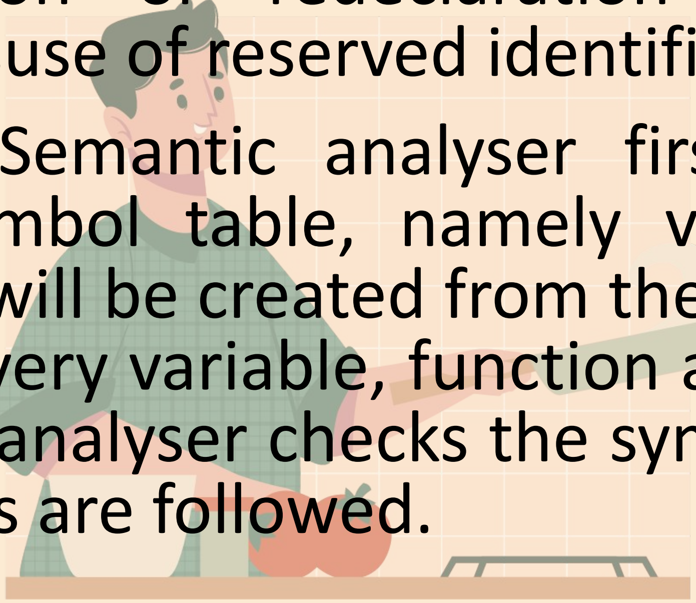


1. Analysis and implementation of Semantic Analyser.
2. Design of the Symbol Table.
3. Working of the Semantic Analyser.
4. Implementations to be done.

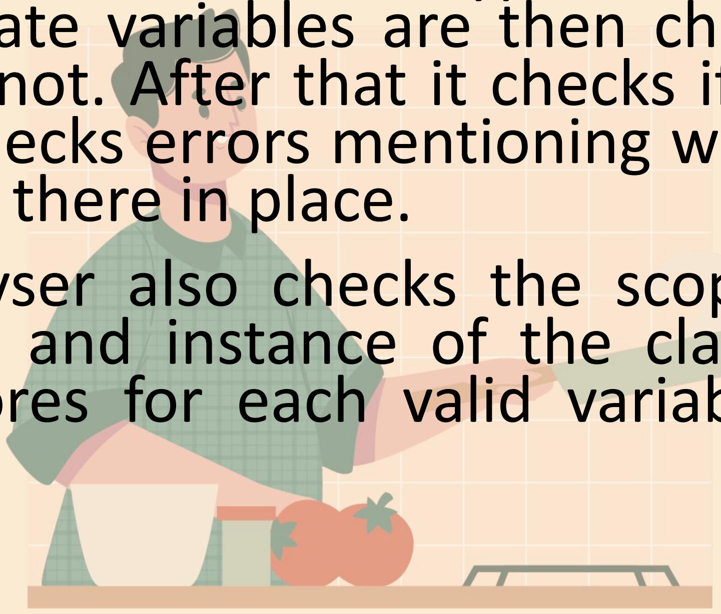
Implementation of Semantic Analyser

The job of a semantic analyser is to check the type definition errors; no declaration or redeclaration of variables/classes/ functions; and any misuse of reserved identifiers.

In Michelin, the Semantic analyser first creates two tables analogous to the symbol table, namely variable table and the function table, which will be created from the given token stream by the Parser. Then for every variable, function and class usage and it's declaration, semantic analyser checks the symbol tables and verifies if the appropriate rules are followed.



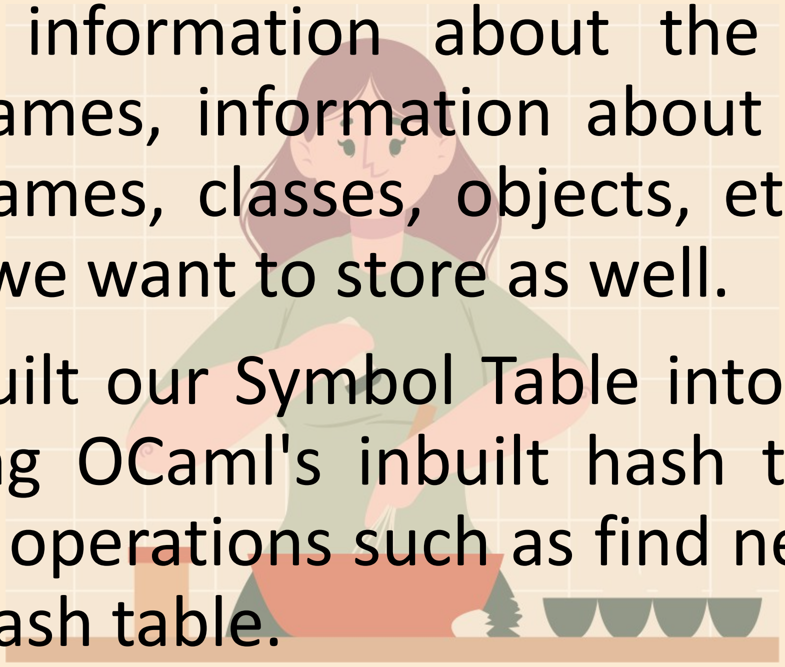
Implementation of Semantic Analyser



Then the Semantic analyser checks if all the arithmetic operations performed are of the likewise variable types and is allowed. The assigned values to the appropriate variables are then checked to see if the data types are matching or not. After that it checks if the variables which are giving as input, then checks errors mentioning what was the original data type which needs to be there in place.

The semantic analyser also checks the scope, declaration, function number, class number and instance of the classes and functions. The variable table also stores for each valid variable, which function it is declared in.

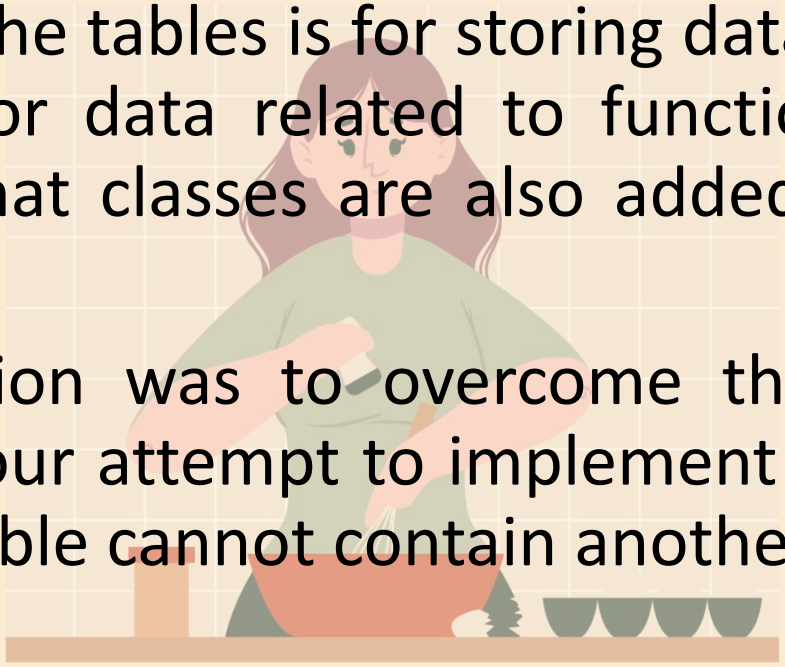
Symbol Table



Symbol Table is an essential data structure in a compiler which keeps track of the information about the scope and binding information about names, information about instances of various variables, function names, classes, objects, etc. It helps store any additional attributes we want to store as well.

In Michelin, we built our Symbol Table into two parts of similar implementation, using OCaml's inbuilt hash table feature as it is mutable and also the operations such as find need average constant time complexity for hash table.

Symbol Table



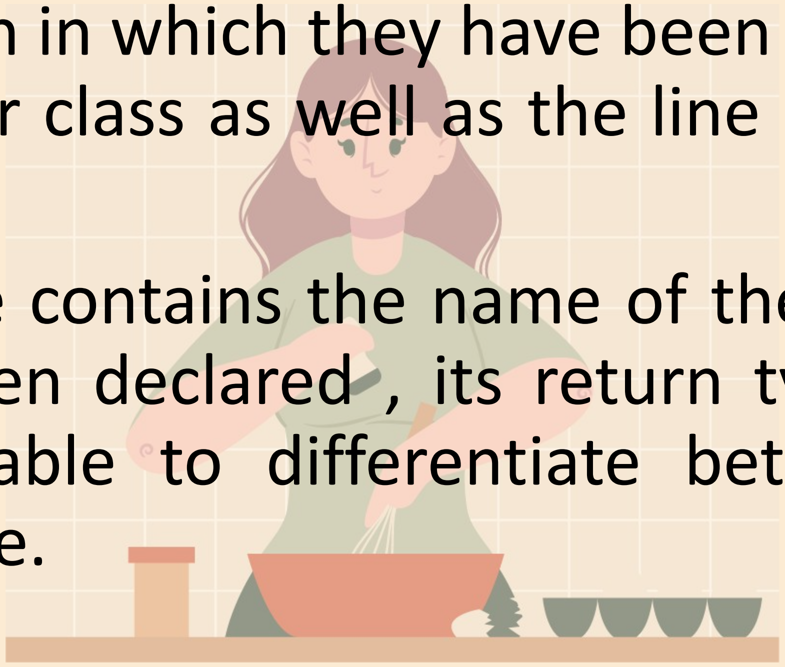
Currently we have implemented the symbol table using two hash tables where one of the tables is for storing data related to variables while the other is for data related to functions ,which we have implemented such that classes are also added under the function hash table.

This implementation was to overcome the the limitation we encountered during our attempt to implement it as a single symbol table where a hash table cannot contain another hash table.

Symbol Table

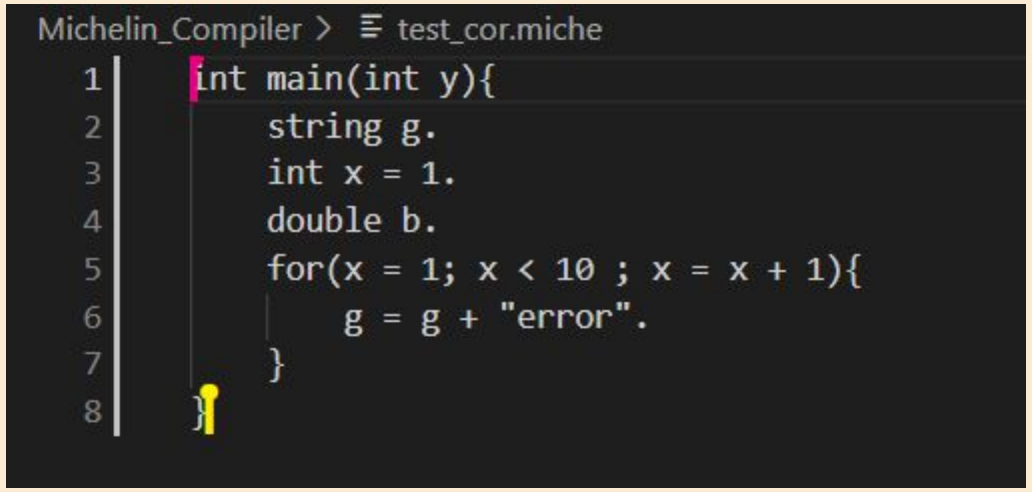
The variable table contains the names of the variables, their datatype, the function in which they have been declared, their scope within the function or class as well as the line number where it has been declared.

The function table contains the name of the function/class , the line where it has been declared , its return type in the case of a function and a variable to differentiate between the functions having the same name.



Working of the Semantic Analyser

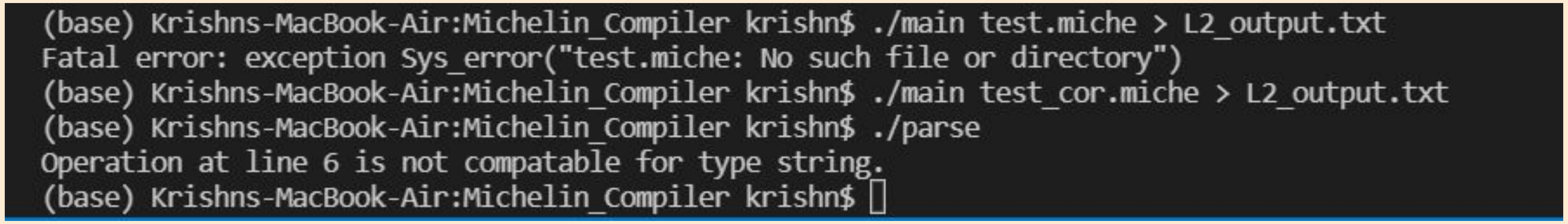
Input code:



Michelin_Compiler > test_cor.miche

```
int main(int y){
    string g.
    int x = 1.
    double b.
    for(x = 1; x < 10 ; x = x + 1){
        g = g + "error".
    }
}
```

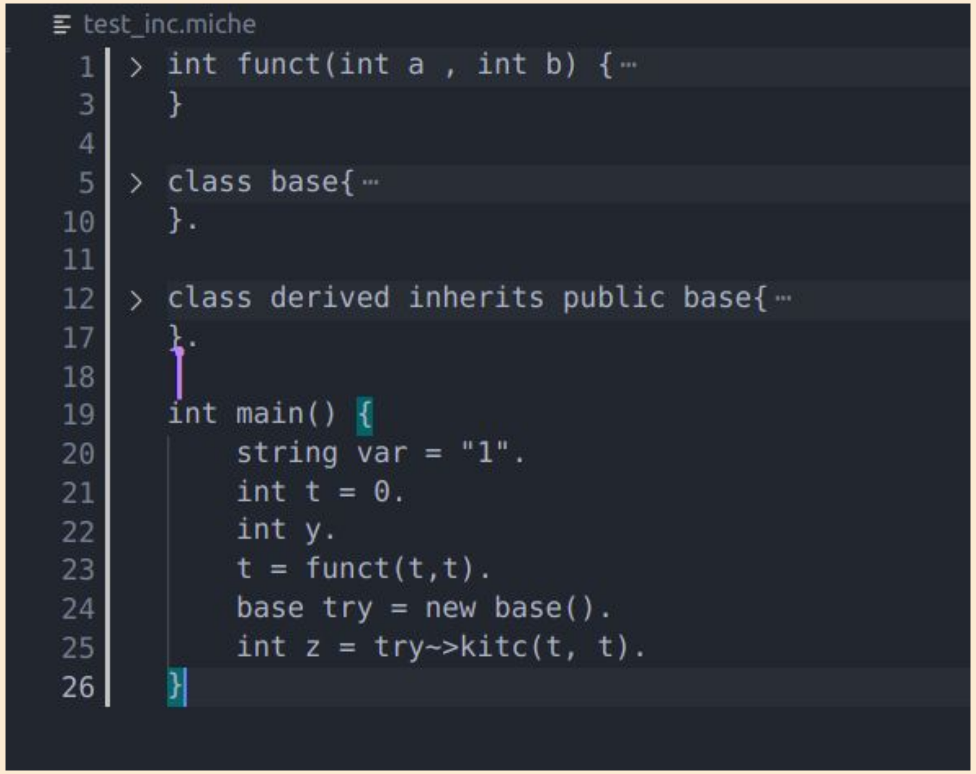
Semantic error:



```
(base) Krishns-MacBook-Air:Michelin_Compiler krishn$ ./main test.miche > L2_output.txt
Fatal error: exception Sys_error("test.miche: No such file or directory")
(base) Krishns-MacBook-Air:Michelin_Compiler krishn$ ./main test_cor.miche > L2_output.txt
(base) Krishns-MacBook-Air:Michelin_Compiler krishn$ ./parse
Operation at line 6 is not compatable for type string.
(base) Krishns-MacBook-Air:Michelin_Compiler krishn$ 
```

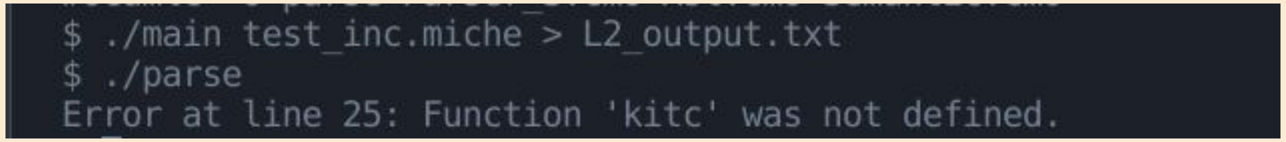
Working of the Semantic Analyser

Input code:



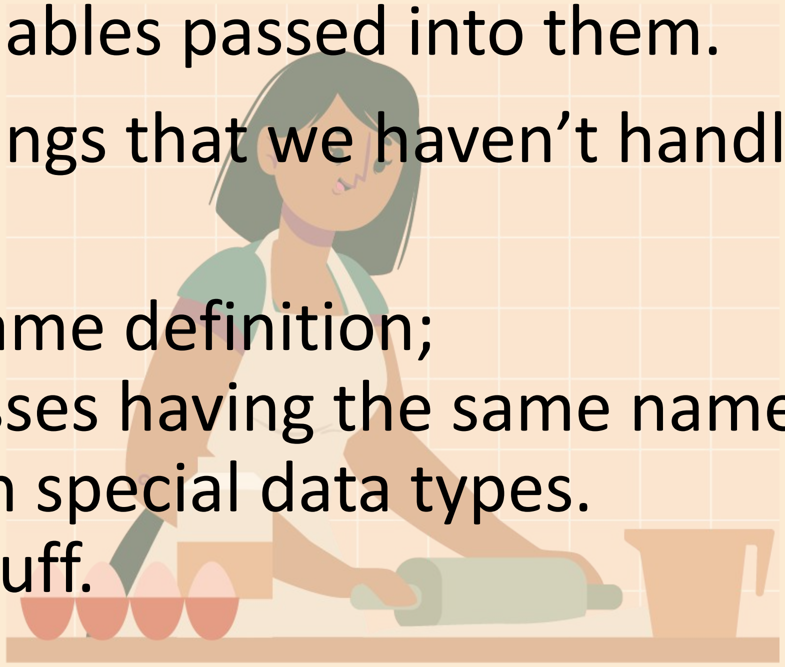
```
test_inc.miche
 1 > int funct(int a , int b) {⋯
 3   }
 4
 5 > class base{⋯
10   }.
11
12 > class derived inherits public base{⋯
17   }.
18
19   int main() {
20       string var = "1".
21       int t = 0.
22       int y.
23       t = funct(t,t).
24       base try = new base().
25       int z = try~>kitc(t, t).
26   }
```

Semantic error:



```
$ ./main test_inc.miche > L2_output.txt
$ ./parse
Error at line 25: Function 'kitc' was not defined.
```

Further Additions Yet to be Done



We are yet to check the semantic errors faced by classes and functions and the variables passed into them.

In particular, some things that we haven't handled as of now include:

1. Classes with the same definition;
2. Functions and Classes having the same name;
3. Insertion of certain special data types.
4. 1D array related stuff.

Team Details

Sai Sidhardha Grandhi
CS19BTECH11050
Project Manager
GitHub ID: G-Sidhardha

Krishn Vishwas Kher
ES19BTECH11015
Language Guru
GitHub ID: KrishnKher

Mukkavalli Bharat Chandra
ES19BTECH11016
System Architect
GitHub ID: chandra3000

Sujeeth Reddy
ES19BTECH11022
System Integrator
GitHub ID: Sujeeth13

Sree Prathyush Chinta
CS19BTECH11043
Tester
GitHub ID: Prathyush-1886

Vemulapalli Aditya
CS19BTECH11025
System Integrator
GitHub ID: VEMULAPALLI-ADITYA

Praneeth Nistala
CS19BTECH11054
Tester
Github ID: Praneeth-Nistala

Thank You