Semantic Analyzer

What is Semantic Analysis?

- Semantic Analysis makes sure that declarations and statements of program are semantically correct.
- It uses syntax tree and symbol table to check whether the given program is semantically consistent with language definition.
- Type checking is an important part of semantic analysis which makes sure that each operator has matching operands.

What is Symbol Table?

- Symbol Table is an important data structure created and maintained by the compiler in order to keep track of semantics of variables i.e. it stores information about the scope and binding information about names, information about instances of various entities such as variable and function names, classes, objects, etc.
- It is built-in lexical and syntax analysis phases.
- The information is collected by the analysis phases of the compiler and is used by the synthesis phases of the compiler to generate code.
- It is used by the compiler to achieve compile-time efficiency.

How is our Symbol Table implemented?

- We have used STL Map Data Structure.
- There are 4 types of labels:
  - IDENT(for identifiers),
  - FUNCTION (for functions),
  - STRUCT_IDENT (for structs),
  - UNDEF (Other than the above 3)

How to run Semantic Analyzer?

- **To run the semantic analyzer using Makefile on test cases, use command:**
  make
- **To run the semantic analyzer manually,**
  - flex lexer.l
  - bison -d -t parser.y
  - g++ -o testing_parser.out lex.yy.c parser.tab.c -lfl
  - ./testing_parser.out testFile.cos

What does our Semantic Analyzer do?

- Checks for any undeclared variables.
- Checks for any undeclared functions in use.
- Checks for variables which are declared but unused.
- Checks if any operator has invalid operands.
- Type checking.

Image of working Semantic Analyzer

```
1   # Undeclared variable m2
2
3   struct astrobject
4   {
5       mass m;
6       dist r;
7   };
8
9   proc mass speaaa(mass a, acc b)
10  {
11      return b*a*6.66e-31;
12  }
13
14  proc int main()
15  {
16      mass m1 = 1.12e23;
17      m1= 1.1e2*m2;
18
19      return 0;
20  }
21
```

```
1   ERROR at line no. 17 : Undeclared
    variable used name- m2
2
3   Compilation terminated! 1 errors and 0
    warnings generated.
```