CoPro:
Semantic Analysis

T. Rohan - CS20BTECH11064
I. Rajasekhar - CS20BTECH11020
P. Vikas - CS20BTECH11037
S. Satvik - CS20BTECH11048
B. Revanth - CS20BTECH11007
J. Vamsi Preetham - CS20BTECH11058
G. Yagnavalkya - CS20BTECH11019

Semantic Analysis

- **Third phase of the compiler project.**
- Checks the semantic consistency of the input code using syntax tree and symbol table.
- Produces errors for various cases such as redeclaration of variables, type mismatches etc.

Where we were after Parser

- In the parser stage, we've taken tokens as input from lexer and then analyzed them against the production rules mentioned.
- The output after the parser stage was the parser tree.
- In this phase of our compiler, we take the parser tree as input and check if it makes sense semantically.
- We are also expected to show various errors in this phase.

Semantic Analysis

- Semantic analysis is the process of extracting the meaning of the text.
- It is the third phase of building our Compiler.
- This phase of our project makes sure that all the statements and declarations of program are semantically correct.
- It occurs during compile time (static semantics) and run time (dynamic semantics)
- It helps us in maintaining semantic correctness of the program.

Implementation

- One important step in semantic analysis is building a symbol table.
- The symbol table is a data structure created to store information such as names, types etc. of identifiers, functions etc.
- Using this symbol table, we were able to write functions to point out errors along with their line no's
- Our implementation of the symbol table consists of four attributes:
  - Symbol
  - Datatype
  - Type
  - Line number

Implementation

There are four major static checks done by the semantic analyzer:

- **Undeclared variables:** It makes sure that the variables used should've been declared before, and shows an error message otherwise, indicating that the variable hasn't been declared.

- **Multiple declaration of variables:** It makes sure that a variable cannot be redeclared. If the variable is already present in the symbol table, an error message would be printed informing the user regarding multiple declarations of that variable.

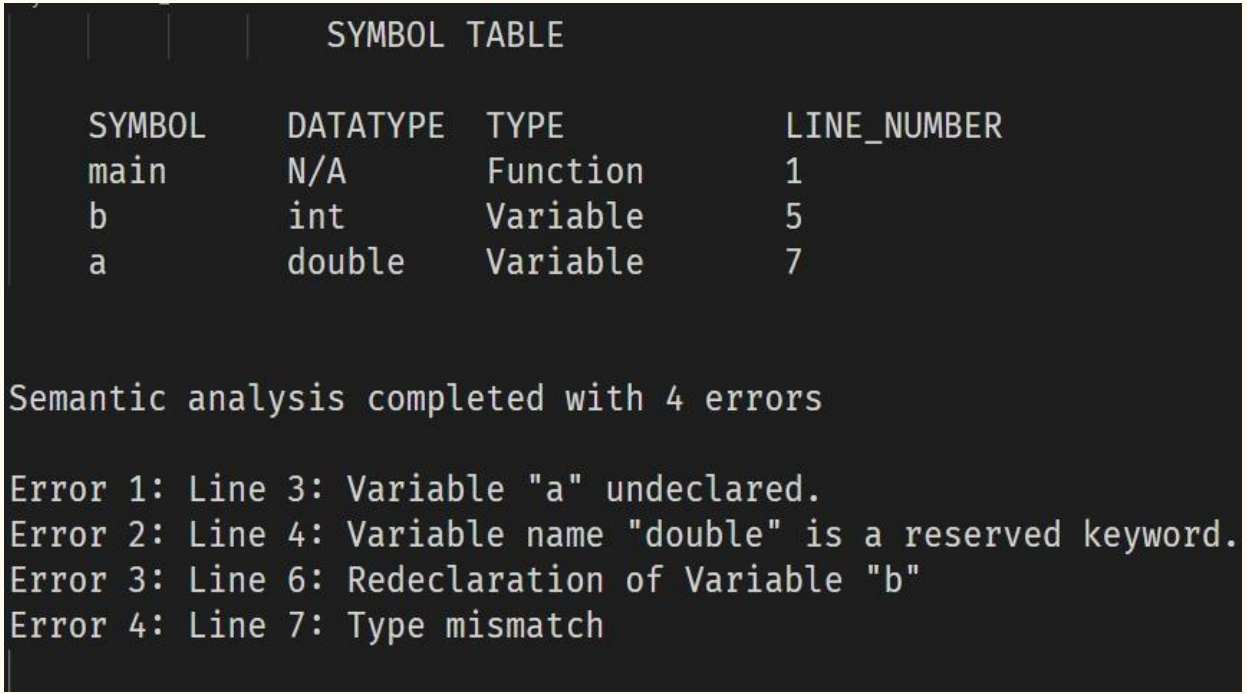
Implementation

- **Variables cannot be reserved words:** It makes sure that the variable name does not belong to the list of reserved words. This is performed before it is added to the symbol table during declaration. An error would be shown if a variable was declared with the same name as a reserved word.

- **Type Checking :** It makes sure that two given variables have the same data type. The “type” field is added to the struct representing value and expression tokens to keep track of the type of the compound expressions. An error is thrown if two variables of different data types are compared.

Error case Example

```
main->int
<<
    int a=10
    int double
    int b=10
    int b=100
    double a=10+”hi”
>>
```



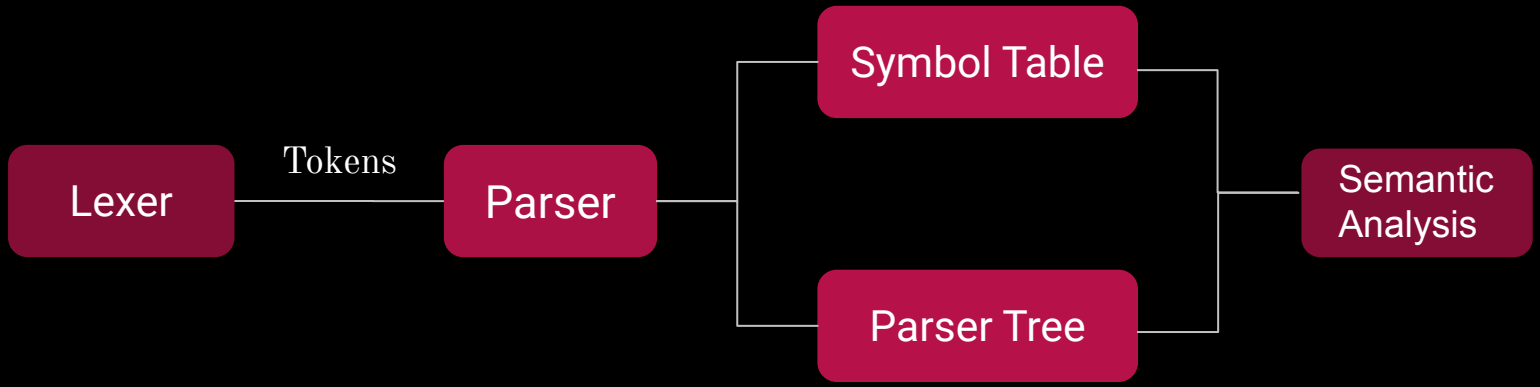
SYMBOL TABLE

SYMBOL	DATATYPE	TYPE	LINE_NUMBER
main	N/A	Function	1
b	int	Variable	5
a	double	Variable	7

Semantic analysis completed with 4 errors

Error 1: Line 3: Variable "a" undeclared.
Error 2: Line 4: Variable name "double" is a reserved keyword.
Error 3: Line 6: Redeclaration of Variable "b"
Error 4: Line 7: Type mismatch

Control Flow



Thank You