Assignment V

Aayush Nirwan 1601001 Praveen Kumar 1601041 Nilesh Agarwal 1601037

I. Introduction

The Symbol Table implemented in Lab 11 is integrated with the code that we had developed upto Assignment II as well as different types of error handling has been done.

II. Implementation

A. Symbol Table

- Created 2 structures namely `table` (Global Symbol Table) and `ref_table` (Local Symbol Table)
- Used a stack to keep track of scope of each function. It helped in determining the level of a parameter/variable.
- Prototype declaration is also taken into consideration as well as their respective definitions.

B. Error Handling

i. Type Checking

Suppose we have LHS and RHS (delimited by '='), then LHS.type == RHS.type. To achieve this we copied the type of LHS and RHS from it's corresponding Symbol table entry. If they match then type is matched. Else a proper error message is thrown.
Error Name: `TYPE MISMATCH: at line #`

ii. Declarations

If a function prototype is declared, then we check for the function definition. If Return type mismatches, Number of parameteres mismatches or Type of parameters mismatches, then for all the mentioned cases a proper error is thrown.

During a function call, Type checking and the above mentioned criteria are carried out. If failed then error messages are thrown.
Error Name: `Conflicting types for function name(): EARLIER DECALRED AS type_name`

iii. Function Calls

If a function is called from main(), there may be chances that the compiler hasn't encountered the definition yet (only prototype). So it wouldn't have the required information assocaited with the function. In this case we have to compile the input multiple times(more than 1), to collect the information about all the functions first, then checking for the errors.
Error Name: `ERROR IN FUNCTION CALL at line #: Conflicting types for the arguments` Error Name: `ERROR IN FUNCTION CALL at line #: MISMATCH in #no of Arguments`

iv. Type Casting

If type "int" is casted into "float" then appropriate error message is thrown as "int" won't be able to accomodate a "float" type.
Error Name: `TYPE CASTE ERROR: in line #`

v. Return Types

Return types of every function is type checked. If any mismatch is found then appropriate error message is thrown.
Error Name: `In function f_name():  MISMATCH IN RETURN TYPE`

NOTE: ALL ERRORS ARE PRINTED IN RED COLOUR FOR BETTER VISUALS.

III. Code Analysis

A. Robustness

The code is robust in nature, changing the input following the format will not produce unexpected results. It will be able to handle them.

B. Correctness

Almost all corner cases have been covered properly, hence it's expected that the code will generate correct output everytime.

C. Time Complexity

$T_{words} = t_1$
$T_{lines} = t_2$
$T_{char} = t_3$
Asymptotic Bound Of the entire code

$$t_2 \leq t_1 \leq t_3$$

D. Space Complexity

Structures used to carry out various tasks.

- Stack<string> s (To keep track of level information of var/par)
- Array of Strings (To keep track of incoming lexemes)
- Structures (table and ref_table)

Overall Space complexity $\leq$ $O(no\_of\_words)$

E. Optimisation

- The Symbol tables (Global and Local) are sorted for fast accessing and searching ($std :: sort()$).
- Instead of Linear, Binary Search would be preferrable $O(log(n))$
- For quick access to symbol table, hashing is also used at some parts.

`NOTE: GENERATED OUTPUT ON THE CONSOLE IS COLOUR DIFFERENTIATED. ERROR MESSAGES IN RED AND SUCCESS MESSAGES IN GREEN(SIMILAR TO INBUILT COMPILER)`

`PS: Run the complete code by executing the command ./make`