**Project Report Group-1**

**Project ID-16**- Develop a parser in Python language that accepts code in C++ and checks for syntax errors. (Expected: loops, integers, if-else).

|  |  |
| --- | --- |
| Uppala Vivek Narayan | 21114108 |
| Kanuru Mohith Kumar Reddy | 21114047 |
| Priyanshu Behera | 21114077 |

This is the repo of our code.

[Uviveknarayan/Cpp-parser-in-python (github.com)](https://github.com/Uviveknarayan/Cpp-parser-in-python)

## Explanation

Our Project contains two jupyter notebooks and associated test cases:

1. Lexer.ipynb
2. Parser.ipynb

### Lexer

This program parses the input C++ file and create all the tokens from it. It then store the tokens in a list cache named tokens. It identifies identifiers, keywords, separators, operators, special characters.

### Parser

This file then parses the tokens created by the lexer and also maintains a symbol table to check if any variable is undeclared in the scope.

The main loop initializes an iterator named i which starts from 0 and iterates on the tokens list.

Now a program in C++ can only contain a variable declaration/definition and function declaration/definition in global scope and some preprocessors and using statements

Similar to the below BNF grammar

Program → function def | function decl | variable decl | variable def | preprocessor | using statement

Now i combine function declaration/definition and variable declaration/definition in function named

### checkFuncDefOrVarDef returns [index, err]

* + This function checks for function or variable declaration or definition

And another function checks for preprocessor directive

### ignorePreprocessor returns [index, err]

* + This function is called when a # is found as a first character in a token
  + It parses the preprocessor directives

And one more function which just ignore the using … statements

### ignoreTillSemicolon returns [index, err]

* + This function is called when the initial token is “using”
  + It ignores all the statements until ;

Now we have other functions which are called while parsing the function declaration/ definition and variable declarations/definitions namely

### checkVarHeader returns [ index, err, dtype , ident ]

* + dtype - **datatype of the matched identifier**
  + Ident - **matched identifier**
  + This function matched the patter <dtype> <ident> for example ‘int a’ or ‘int main’ etc..

### checkScope returns [index, err]

* + This handles everything inside curly braces ‘{‘ and ‘}’ for example function declarations scope, if scope, else scope, while scope etc.
  + This also takes symbol table from the above scope to check for all the declared variables in the parent scope and it also updates the symbol table in its own scope
  + We have also took care about the missing opening brace case if there is a missing opening brace but there is one closing brace then we give error as missing open brace only and if both braces are missing we give two errors(missing {,and missing })

### checkExpr returns [index, err, dtype]

* + This parses all the expression and also checks for the datatype on which the operators work. ( for example we cannot add an int and a string ) and also checks if the operator is valid on the operand provided the lvalue or rvalue property of the operand

### checkLval return [index, err, dtype, islval]

* + This parses the lvalue in a expression and returns if the value is an actual lvalue ( that is from the specification of C++ )

The lval and expr combinely solves to parse the expression on the basis of the below defined in the grammar

expr → lval operand expr | lval

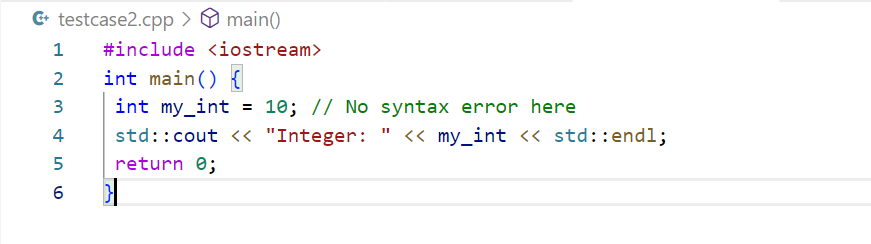
lval → unary\_op lval | identifier | identifier(param\_list) | lval post\_op | pre\_op lval

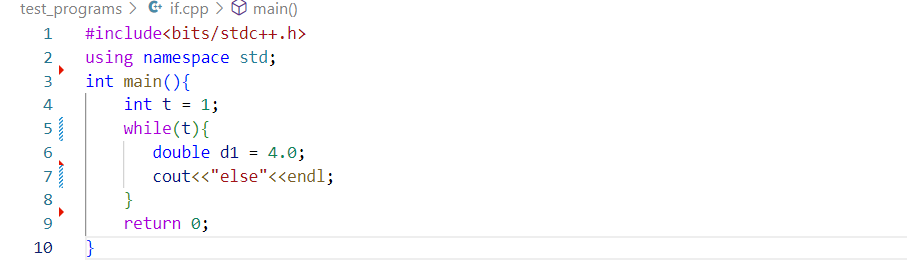
### checkFuncDecOrValDef returns [index, err, symbol\_table]

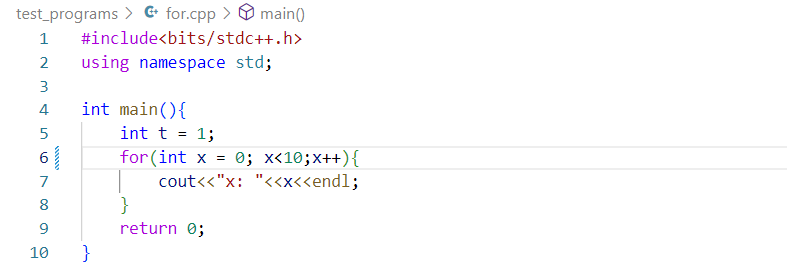
* + This parses the only function declaration and variable declaration/definition ( as function definition is not allowed inside any scope )

**Now for running the program we have to first run lexer and the parser**

# Test Case For Showing Correct Syntax







We have checked the following and implemented the parser:

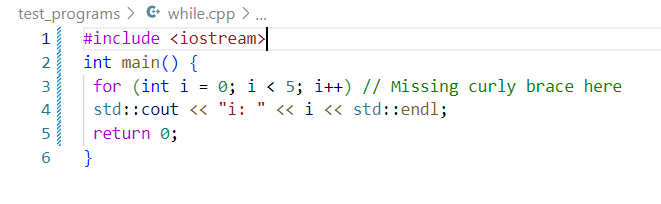
1. Declaration of Variables
2. Declaration of Functions
3. If Else (including Nesting) Statements
4. All Loops i.e. For Loop, While Loop, Do While Loop (including Nesting)
5. Function Calls (included type checking)
6. Expressions
   1. Handled with Type Checking
   2. Handled the operators on the basis of their lval or rval property
   3. Handled Assignment Operators
7. Comments
   1. Single Line Comments
   2. Multi Line Comments
8. Return
9. Cin, cout
10. Scope Checking

# Test Cases For Showing Incorrect Syntaxes

## Test cases

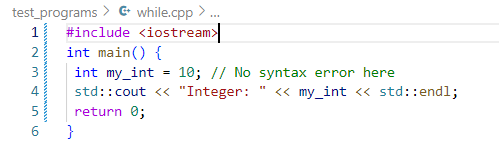
1. Missing curly braces

Here we can see that there is a missing curly brace after the for loop



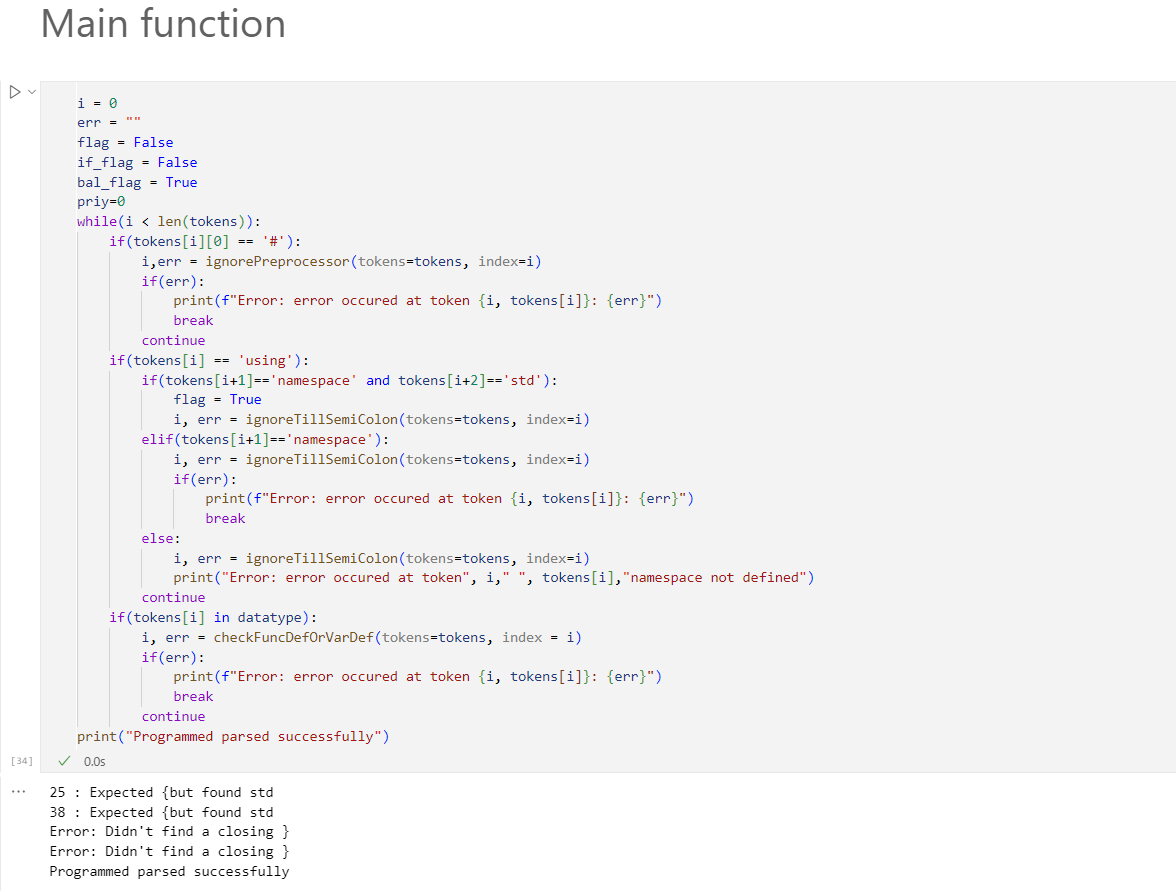
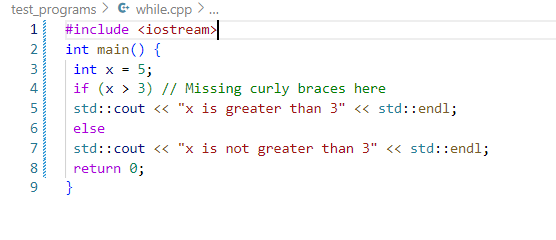


2)C++ code with no errors

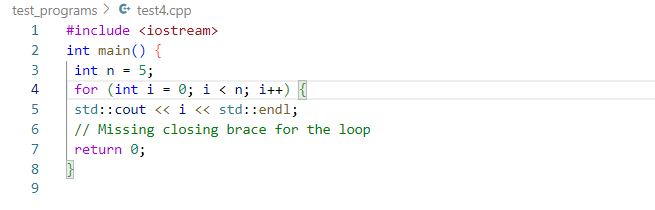




3) C++ code with a syntax error in an if-else statement

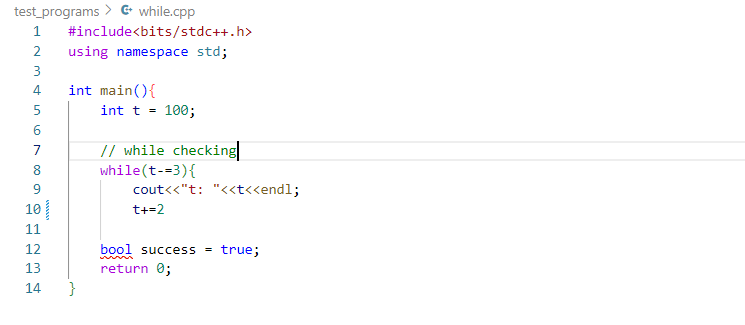


4) C++ code without closing bracket





5)C++ code with two errors a missing semicolon and and a missing closing brace





6)C++ code with three errors 1) missing ) brace at while declaration 2)missing semicolon and

3) missing closing brace for the while loop

