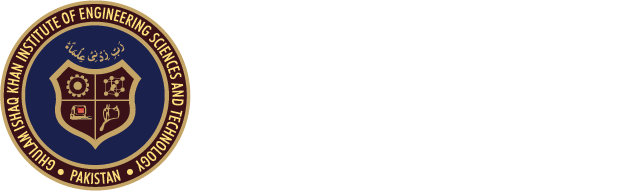
**CS-224 Formal Languages and Automata Theory**

**Semester Project Report**

Ghulam Ishaq Khan Institute of Engineering Sciences & Technology, Topi, Swabi 23460, Pakistan

****

**Beta Tokenizer**

**Members:**

* 2021468 (Muhammad Talha Imtiaz)
* 2021446(Muhammad Saad Ullah)
* 2021276 (Mahnoor Amir)

**Table of Contents:**

**1. Introduction--------------------------------------------------Page.3**

**2. Project Requirements--------------------------------------Page.3**

**3. Project Description-----------------------------------------Page.4**

**4. Source Code--------------------------------------------------Page.5**

**5. Example Input Text----------------------------------------Page.6**

**5. Output---------------------------------------------------------Page.6**

**6. References----------------------------------------------------Page.8**

**Introduction:**

This project was carried out in three main stages. First part dealt with understanding the Flex software, installing, and getting familiar with its front end. Second part was writing and running flex code examples from different sources, including websites and tutorial videos online. The third part was hands-on implementation where we wrote our own flex code using prior knowledge from practice sets to create a lexical analyzer that can analyze any C++ code to create tokens.

**Project Requirements:**

1. **Build Lexical Analyzer:**

Build Lexical Analyzer (LA) of basic compiler front-end.

1. **Use *FLEX* to create the Lexical Analyzer:**

A free and open-source software alternative to lex.

1. **Group Members:**

This Project group consists of 3 members.

1. **Your Lexical Analyzer Name:**

Beta Tokenizer

1. **Lexical Analyzer code:**

Our Lexical Analyzer covers most of the tokens supported by C++ language.

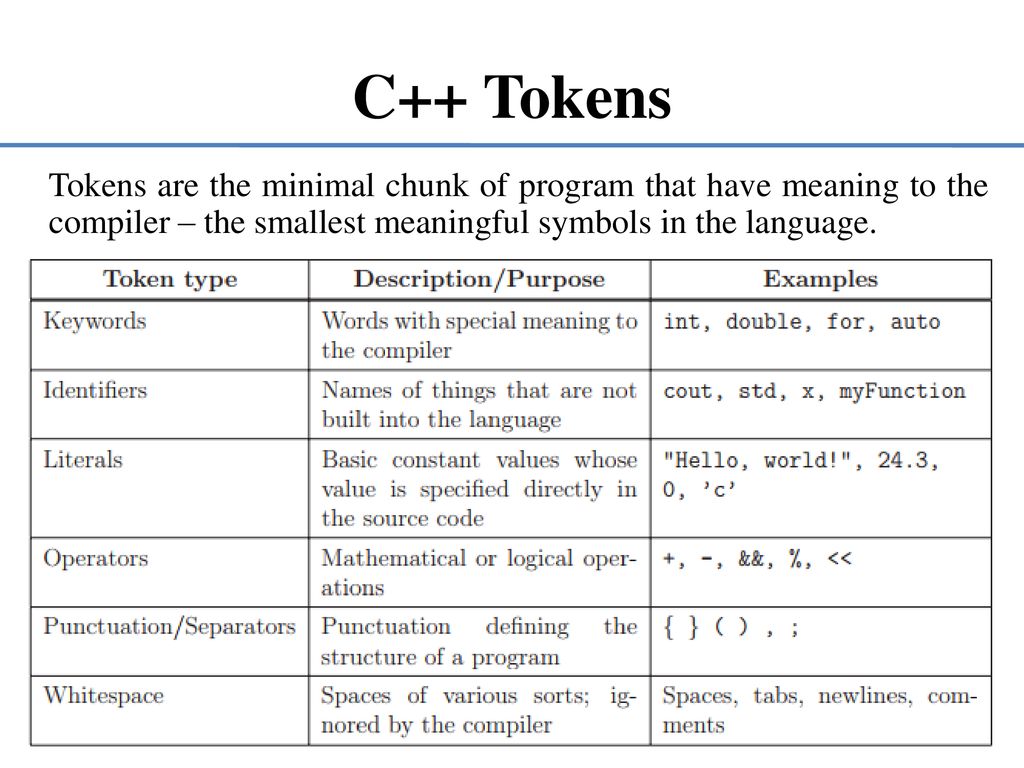
1. **Sample Examples and Outputs for Lexical Analyzer:**

A sample example and its output are included in this report.

**Project Description:**

We have built a Lexical Analyzer (LA) of basic compiler front-end for the C++ Language. This means that our compiler can read an input from source file and generate required tokens according to what our language allows and write them out in a Lex file. We have included a Lex file along with example source files to make sure examples cover most of the tokens supported by C++ language. Our LA identifies all tokens from a source input file and stores them in an output file including, token\_id, token, and line number.

Below is a summary of the tokenizing criteria we followed for creating our Lexical Analyzer:



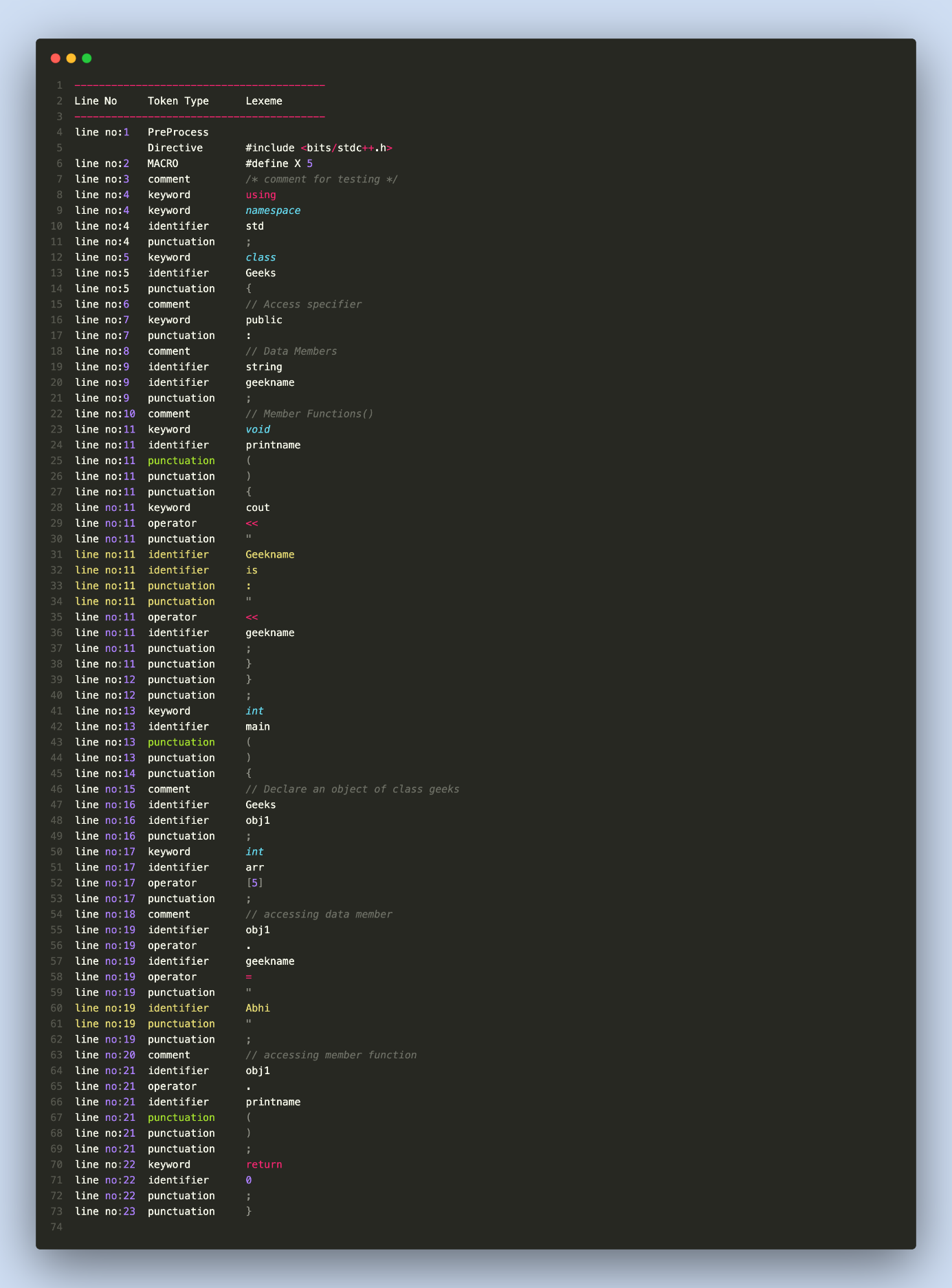
**Source Code:**



**Example Input text:**

****

**Output:**



**References:**

* <http://alumni.cs.ucr.edu/~lgao/teaching/flex.html>
* <https://web.stanford.edu/class/archive/cs/cs143/cs143.1112/materials/other/manflex.html>
* <https://www.geeksforgeeks.org/flex-fast-lexical-analyzer-generator/>
* <https://web.stanford.edu/class/archive/cs/cs143/cs143.1128/handouts/050%20Flex%20In%20A%20Nutshell.pdf>
* <https://www.cs.ucr.edu/~mafar001/compiler/webpages1/lab01_lexer.html>
* <https://www.youtube.com/watch?v=LpVufkH4gog>