

Lab Report

Subject: CSE-3214 Compiler Design Lab

Experiment Name: Implementation of Tokenization
using C++

Tool Used: C++ Programming Language

Submitted By:

Eyasir Ahamed

Roll No:15

Registration No: 202004017

Submitted To:

Kohinur Parvin

Lecturer,

Netrokona University, Netrokona

Objective:

To implement a program that breaks an input C++ code into tokens like keywords, identifiers, operators, symbols, and numbers.

Theory:

Tokenization is the process of breaking a sequence of characters (source code) into meaningful units called tokens.

Each token is categorized into types such as:

- Keyword: Reserved words (e.g., int, if, while)
- Identifier: User-defined names (e.g., x, sum, count)
- Operator: Symbols performing operations (e.g., +, -, *, ==)
- Symbol: Punctuation and special characters (e.g., ;, {, })
- Number: Constants or numeric values (e.g., 10, 99)

Tokenization is the first phase of a compiler, called Lexical Analysis.

Algorithm:

- Define sets for keywords, operators, and symbols.
- Read the input source code from the user until \$ is entered.
- Traverse the code character by character:
- If two consecutive characters form an operator, recognize it.
- If a character is an operator or symbol, recognize and print it.
- If a word is complete (detected by space or symbol), check:
- If it is a keyword, number, or identifier.
- Print each token and its type.
- End of program.

Program Code:

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 set<string>keywords = {"int", "float", "if", "else", "while", "for", "return"};
5 set<string>operators = {"+", "-", "*", "/", "=", ">", "<", ">=", "<=", "==", "!=", "++", "--", ">>", "<<"};
6 set<char>symbols = {'(', ')', ';', '{', '}', ',', '.'};
7
8 bool isKeyword(string s) {
9     return keywords.find(s) != keywords.end();
10 }
11 bool isOperator(string s) {
12     return operators.find(s) != operators.end();
13 }
14 bool isSymbol(char c) {
15     return symbols.find(c) != symbols.end();
16 }
17 bool isNumber(string s) {
18     for (auto c : s) {
19         if (!isdigit(c)) return false;
20     }
21     return !s.empty();
22 }
23 }
```

```

24 void tokenize(string program) {
25     string word = "";
26     int n = program.size();
27     for (int i = 0; i < n; i++) {
28         char c = program[i];
29         if (i+1 < n) {
30             string two = string(1, program[i]) + string(1, program[i+1]);
31             if (isOperator(two)) {
32                 if (!word.empty()) {
33                     if (isKeyword(word)) cout << word << " : Keyword\n";
34                     else if (isNumber(word)) cout << word << " : Number\n";
35                     else cout << word << " : Identifier\n";
36                     word = "";
37                 }
38                 cout << two << " : Operator\n";
39                 i++;
40                 continue;
41             }
42         }
43
44         if (isOperator(string(1, c))) {
45             if (!word.empty()) {
46                 if (isKeyword(word)) cout << word << " : Keyword\n";
47                 else if (isNumber(word)) cout << word << " : Number\n";
48                 else cout << word << " : Identifier\n";
49                 word = "";
50             }
51             cout << c << " : Operator\n";

```

```

52     }
53     else if (isSymbol(c)) {
54         if (!word.empty()) {
55             if (isKeyword(word)) cout << word << " : Keyword\n";
56             else if (isNumber(word)) cout << word << " : Number\n";
57             else cout << word << " : Identifier\n";
58             word = "";
59         }
60         cout << c << " : Symbol\n";
61     }
62     else if (isspace(c)) {
63         if (!word.empty()) {
64             if (isKeyword(word)) cout << word << " : Keyword\n";
65             else if (isNumber(word)) cout << word << " : Number\n";
66             else cout << word << " : Identifier\n";
67             word = "";
68         }
69     }
70     else {
71         word += c;
72     }
73 }
74 if (!word.empty()) {
75     if (isKeyword(word)) cout << word << " : Keyword\n";
76     else if (isNumber(word)) cout << word << " : Number\n";
77     else cout << word << " : Identifier\n";
78 }
79 }

```

```

80
81 int main() {
82     string line, program = "";
83     while (getline(cin, line)) {
84         if (program.size()) {
85             program += " ";
86         }
87         program += line;
88     }
89     tokenize(program);
90 }

```

Input Example:

```
input.txt
1  int t; cin >> t;
2  while (t--) {
3      int n, x; cin >> n >> x;
4      for (int i = 0; i < n; i++) {
5          cout << i << " ";
6      }
7      if (x != n) cout << x;
8      cout << endl;
9  }
10 return 0;
```

Output Example:

```
1  int : Keyword
2  t : Identifier
3  ; : Symbol
4  cin : Identifier
5  >> : Operator
6  t : Identifier
7  ; : Symbol
8  while : Keyword
9  ( : Symbol
10 t : Identifier
11 -- : Operator
12 ) : Symbol
13 { : Symbol
14 int : Keyword
15 n : Identifier
16 , : Symbol
17 x : Identifier
18 ; : Symbol
19 cin : Identifier
20 >> : Operator
21 n : Identifier
22 >> : Operator
23 x : Identifier
24 ; : Symbol
25 for : Keyword
26 ( : Symbol
27 int : Keyword
28 i : Identifier
29 = : Operator
30 0 : Number
31 ; : Symbol
32 i : Identifier
33 < : Operator
34 n : Identifier
35 ; : Symbol
```

```
36 i : Identifier
37 ++ : Operator
38 ) : Symbol
39 { : Symbol
40 cout : Identifier
41 << : Operator
42 i : Identifier
43 << : Operator
44 " : Identifier
45 " : Identifier
46 ; : Symbol
47 } : Symbol
48 if : Keyword
49 ( : Symbol
50 x : Identifier
51 != : Operator
52 n : Identifier
53 ) : Symbol
54 cout : Identifier
55 << : Operator
56 x : Identifier
57 ; : Symbol
58 cout : Identifier
59 << : Operator
60 endl : Identifier
61 ; : Symbol
62 } : Symbol
63 return : Keyword
64 0 : Number
65 ; : Symbol
```

Result:

- Successfully implemented the tokenization process.
- Program correctly identified and categorized keywords, identifiers, operators, symbols, and numbers from the input C++ code.

Conclusion:

- Tokenization is a very important first step in compiler design.
- It helps in breaking the source code into logical parts to further process in parsing, syntax analysis, and code generation.