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.

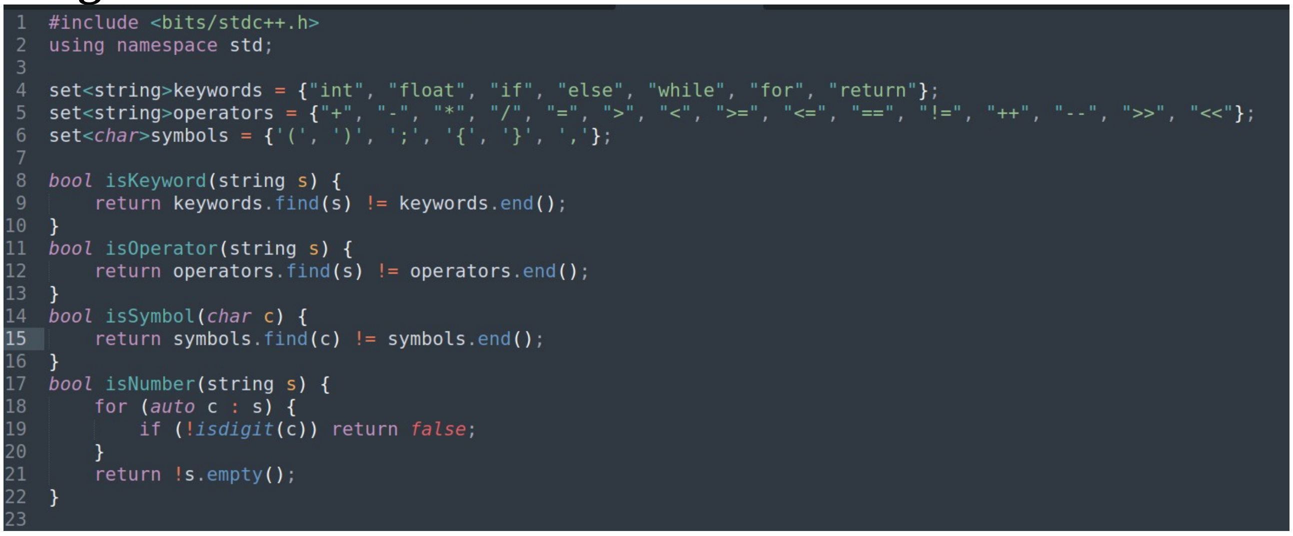
Each token is categorized into types such as:
- Keyword: Reserved words (e.g., int, if, while)
- Identifier: User-defined names (e.g., x, sum, count)
- Operator: Symbols performing operations (e.g., +, -, *, ==)
- Symbol: Punctuation and special characters (e.g., ;, {, })
- Number: Constants or numeric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

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

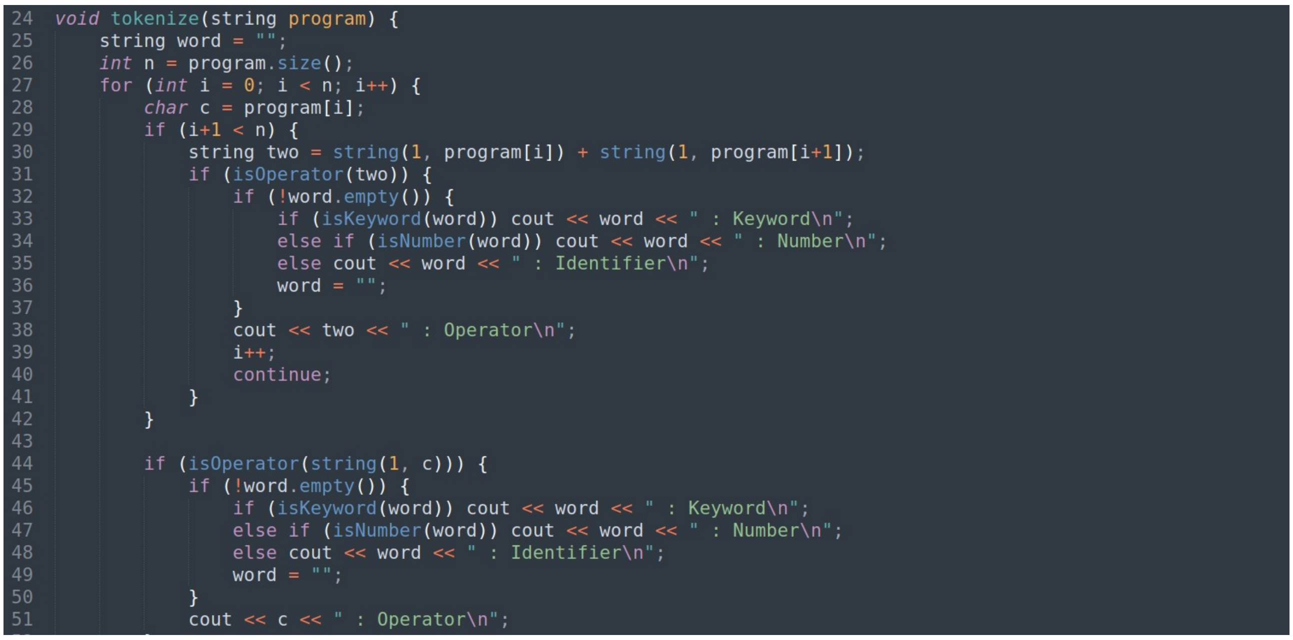
Algorithm:

- 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.

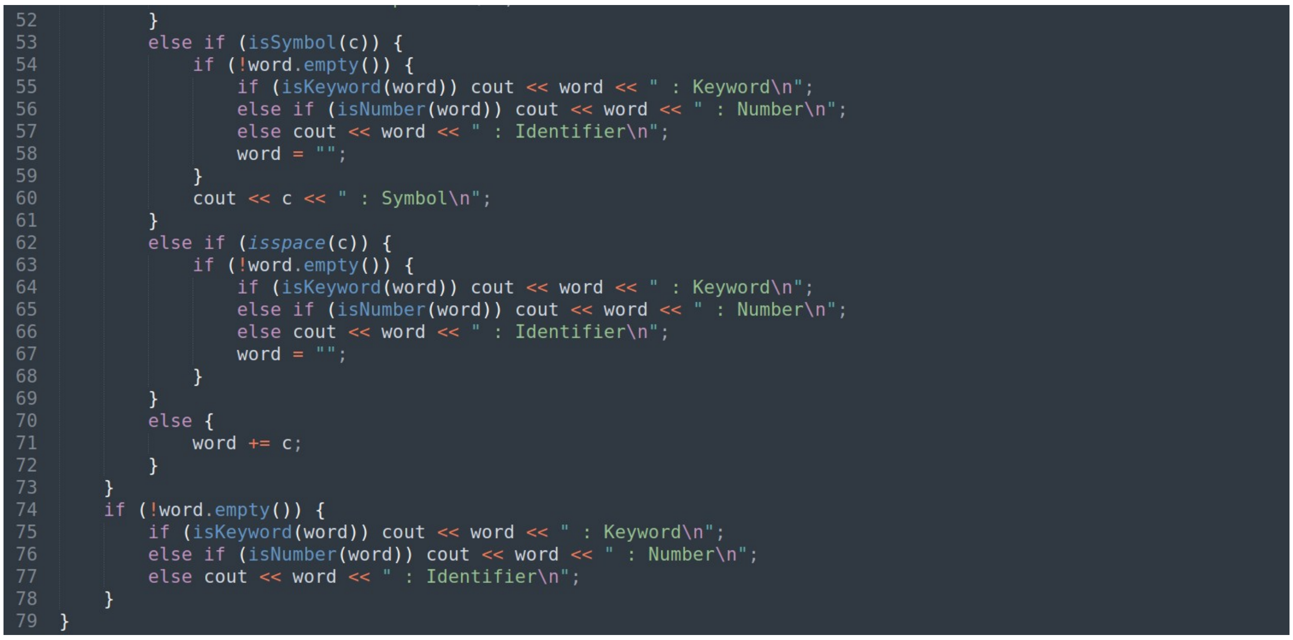
Program Code:



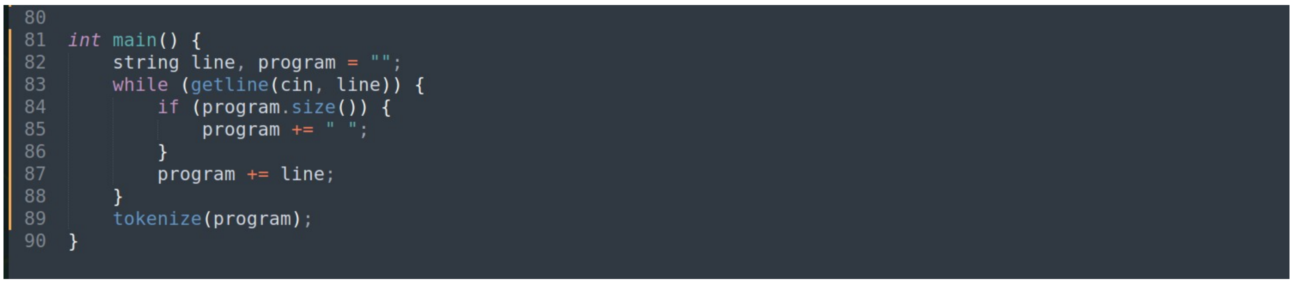
```cpp
#include <bits/stdc++.h>
using namespace std;

set<string>keywords = {"int", "float", "if", "else", "while", "for", "return"};
set<string>operators = {"+", "-", "*", "/", "=", ">", "<", ">=", "<=", "==", "!=", "++", "--", ">>", "<<"};
set<char>symbols = {'(', ')', ';', '{', '}', ','};

bool isKeyword(string s) {
    return keywords.find(s) != keywords.end();
}
bool isOperator(string s) {
    return operators.find(s) != operators.end();
}
bool isSymbol(char c) {
    return symbols.find(c) != symbols.end();
}
bool isNumber(string s) {
    for (auto c : s) {
        if (!isdigit(c)) return false;
    }
    return !s.empty();
}

```



```cpp
void tokenize(string program) {
    string word = "";
    int n = program.size();
    for (int i = 0; i < n; i++) {
        char c = program[i];
        if (i+1 < n) {
            string two = string(1, program[i]) + string(1, program[i+1]);
            if (isOperator(two)) {
                if (!word.empty()) {
                    if (isKeyword(word)) cout << word << " : Keyword\n";
                    else if (isNumber(word)) cout << word << " : Number\n";
                    else cout << word << " : Identifier\n";
                    word = "";
                }
                cout << two << " : Operator\n";
                i++;
                continue;
            }
        }

        if (isOperator(string(1, c))) {
            if (!word.empty()) {
                if (isKeyword(word)) cout << word << " : Keyword\n";
                else if (isNumber(word)) cout << word << " : Number\n";
                else cout << word << " : Identifier\n";
                word = "";
            }
            cout << c << " : Operator\n";
```

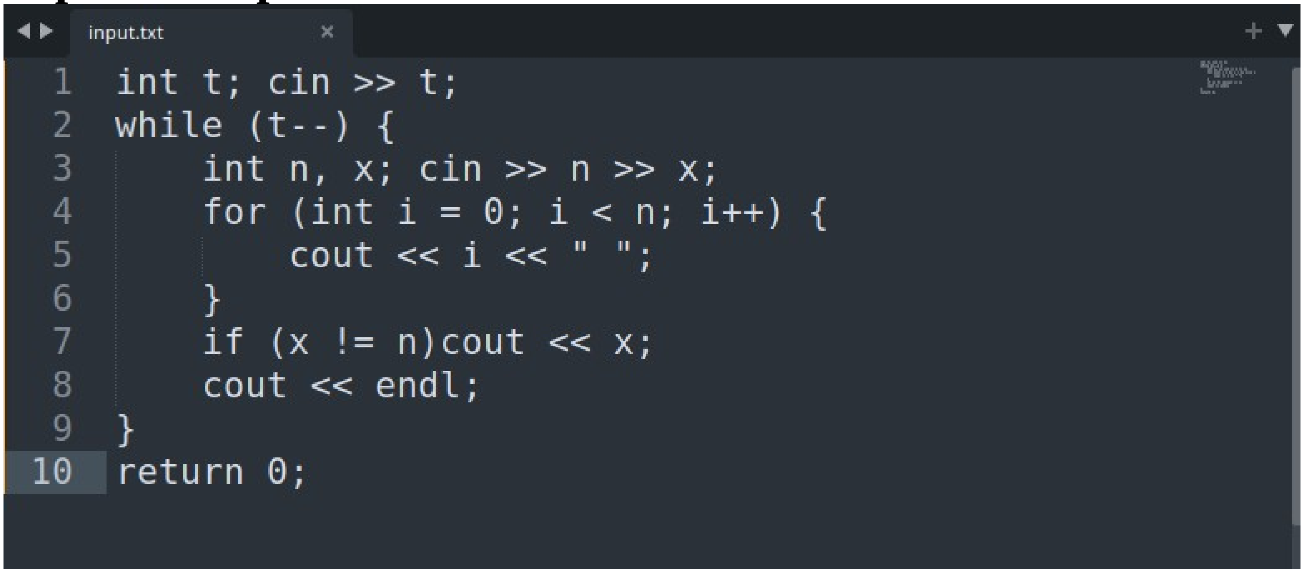


```cpp
        }
        else if (isSymbol(c)) {
            if (!word.empty()) {
                if (isKeyword(word)) cout << word << " : Keyword\n";
                else if (isNumber(word)) cout << word << " : Number\n";
                else cout << word << " : Identifier\n";
                word = "";
            }
            cout << c << " : Symbol\n";
        }
        else if (isspace(c)) {
            if (!word.empty()) {
                if (isKeyword(word)) cout << word << " : Keyword\n";
                else if (isNumber(word)) cout << word << " : Number\n";
                else cout << word << " : Identifier\n";
                word = "";
            }
        }
        else {
            word += c;
        }
    }
    if (!word.empty()) {
        if (isKeyword(word)) cout << word << " : Keyword\n";
        else if (isNumber(word)) cout << word << " : Number\n";
        else cout << word << " : Identifier\n";
    }
}
```



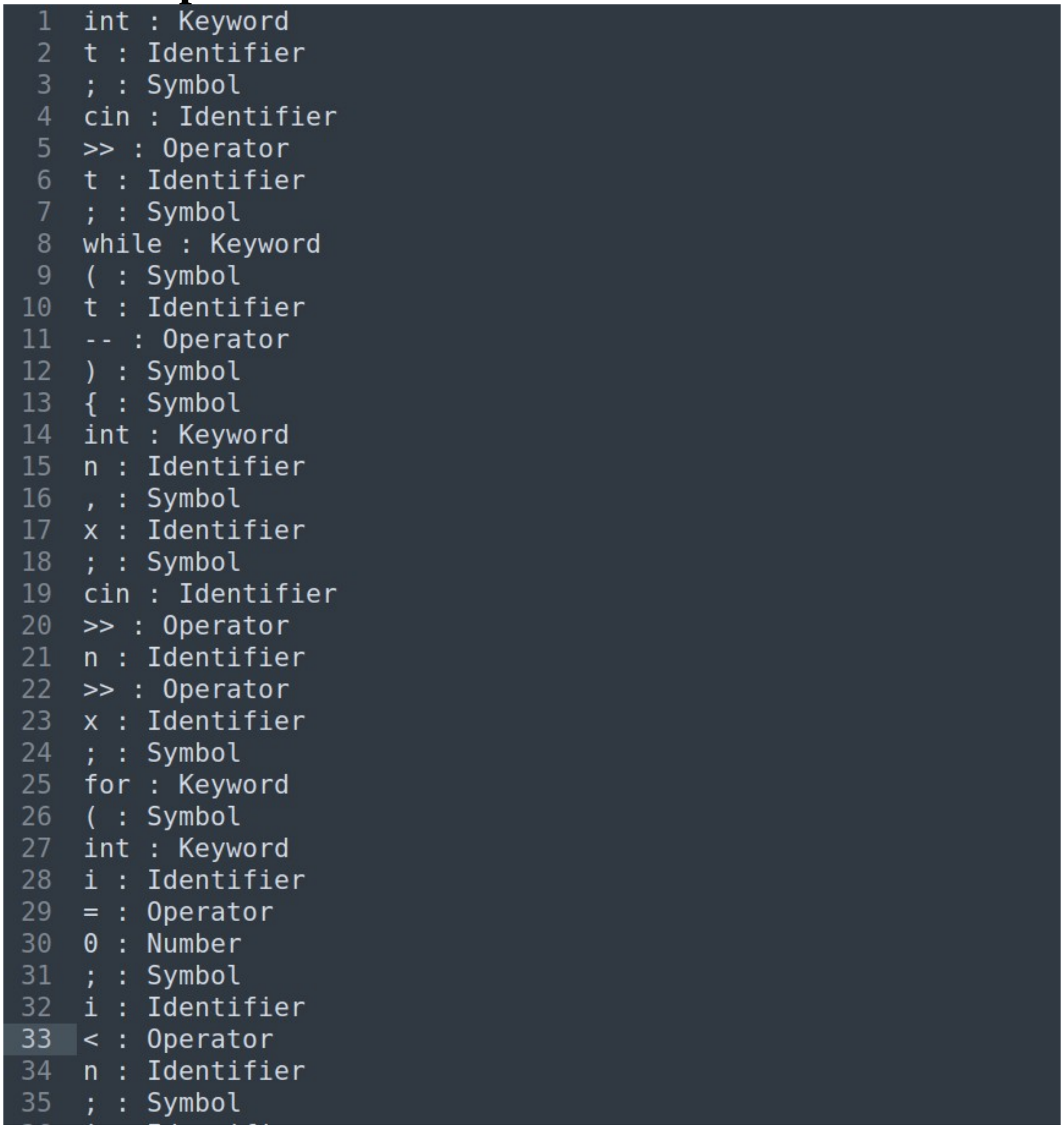
```cpp

int main() {
    string line, program = "";
    while (getline(cin, line)) {
        if (program.size()) {
            program += " ";
        }
        program += line;
    }
    tokenize(program);
}
```

Input Example:

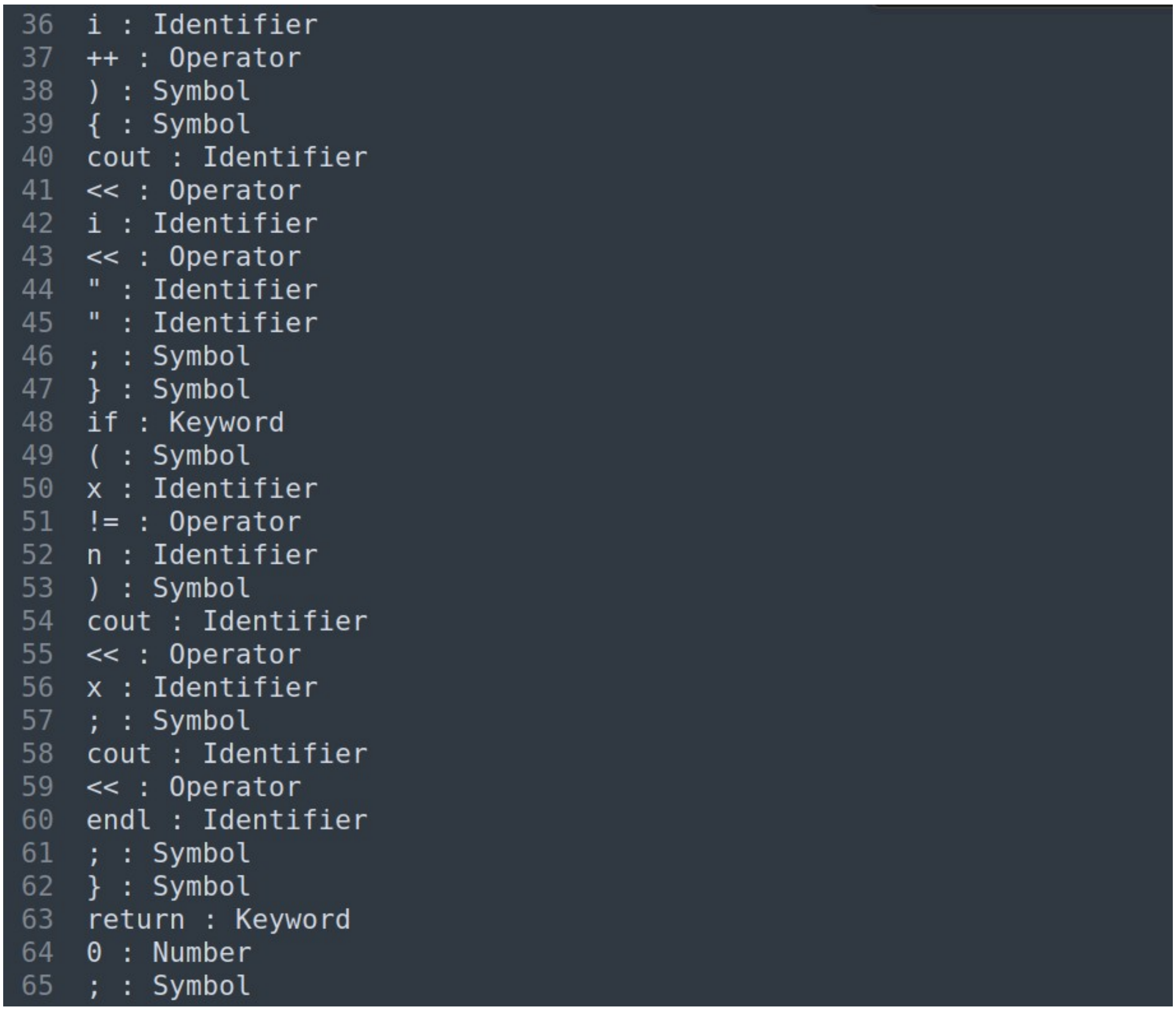


```
int t; cin >> t;
while (t--) {
    int n, x; cin >> n >> x;
    for (int i = 0; i < n; i++) {
        cout << i << " ";
    }
    if (x != n)cout << x;
    cout << endl;
}
return 0;
```

Output Example:



```
int : Keyword
t : Identifier
; : Symbol
cin : Identifier
>> : Operator
t : Identifier
; : Symbol
while : Keyword
( : Symbol
t : Identifier
-- : Operator
) : Symbol
{ : Symbol
int : Keyword
n : Identifier
, : Symbol
x : Identifier
; : Symbol
cin : Identifier
>> : Operator
n : Identifier
>> : Operator
x : Identifier
; : Symbol
for : Keyword
( : Symbol
int : Keyword
i : Identifier
= : Operator
0 : Number
; : Symbol
i : Identifier
< : Operator
n : Identifier
; : Symbol
```



```
i : Identifier
++ : Operator
) : Symbol
{ : Symbol
cout : Identifier
<< : Operator
i : Identifier
<< : Operator
" : Identifier
" : Identifier
; : Symbol
} : Symbol
if : Keyword
( : Symbol
x : Identifier
!= : Operator
n : Identifier
) : Symbol
cout : Identifier
<< : Operator
x : Identifier
; : Symbol
cout : Identifier
<< : Operator
endl : Identifier
; : Symbol
} : Symbol
return : Keyword
0 : Number
; : 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.