

Registration No : 2021-CS-140

Instructor : Mr. Laeeq Khan Niazi

CS-471 Lab 4

Compiler Construction

By using best optimized data structures write the tokenizer for your own language.

The screenshot shows a Visual Studio Code editor with a C++ file named `assignment.cpp` open. The code defines a `TokenType` enum and a `Token` struct. The `TokenType` enum includes `KEYWORD`, `IDENTIFIER`, `NUMBER`, `OPERATOR`, `DELIMITER`, `STRING`, and `EOF_TOKEN`. The `Token` struct has a `TokenType` `type` and a `string` `value`. A `Token` constructor is also defined. The output window shows the results of running the program, displaying the tokens for the input string `if == 10 + x return x + 2 ; ;`.

```
1 #include <iostream>
2 #include <string>
3 #include <vector>
4 #include <unordered_set>
5 #include <regex>
6
7 using namespace std;
8
9 // Enum for token types
10 enum class TokenType {
11     KEYWORD,
12     IDENTIFIER,
13     NUMBER,
14     OPERATOR,
15     DELIMITER,
16     STRING,
17     EOF_TOKEN
18 };
19
20 // Token structure
21 struct Token {
22     TokenType type;
23     string value;
24
25     Token(TokenType t, string v) : type(t), value(v) {}
26 };
27
28 int main() {
29     string input = "if == 10 + x return x + 2 ; ;";
30     vector<Token> tokens;
31     // Tokenization logic would go here
32     for (const auto& token : tokens) {
33         cout << token.type << " " << token.value << " ";
34     }
35     return 0;
36 }
```

Output:

```
Microsoft Windows [Version 10.0.19045.4894]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Fatig\Desktop\Compiler-Construction-CS-471\Lab 4>g++ assignment.cpp
C:\Users\Fatig\Desktop\Compiler-Construction-CS-471\Lab 4>assignment.exe
Keyword: if
Operator: ==
Operator: ==
Number: 10
Operator: +
Operator: +
Keyword: return
Identifier: x
Operator: +
Number: 2
Delimiter: ;
Delimiter: ;
End of File

C:\Users\Fatig\Desktop\Compiler-Construction-CS-471\Lab 4>
```

```
29 class Tokenizer {
61     vector<Token> tokenize(const string& code) {
107
108
109         // Match delimiters
110         if (isDelimiter(code[pos])) {
111             tokens.emplace_back(TokenType::DELIMITER, string(1, code[pos]));
112             pos++;
113             continue;
114         }
115
116         // Match string literals
117         if (code[pos] == '"') {
118             smatch match;
119             string remainingCode = code.substr(pos);
120             if (regex_search(remainingCode, match, stringRegex)) {
121                 tokens.emplace_back(TokenType::STRING, match.str());
122                 pos += match.length();
123                 continue;
124             }
125         }
126
127         // Unexpected character
128         cout << "Unexpected character: " << code[pos] << endl;
129         pos++;
130     }
}
```

```
138 void printTokens() {
155     break;
156     case TokenType::STRING:
157         cout << "String: " << token.value << endl;
158         break;
159     case TokenType::EOF_TOKEN:
160         cout << "End of File" << endl;
161         break;
162 }
163
164 }
165 };
166
167 // Main function to demonstrate the tokenizer
168 int main() {
169     string code = R"(if (x == 10) { return x + 2; })";
170
171     Tokenizer tokenizer;
172     vector<Token> tokens = tokenizer.tokenize(code);
173
174     // Print the tokens
175     tokenizer.printTokens();
176
177     return 0;
178 }
```