

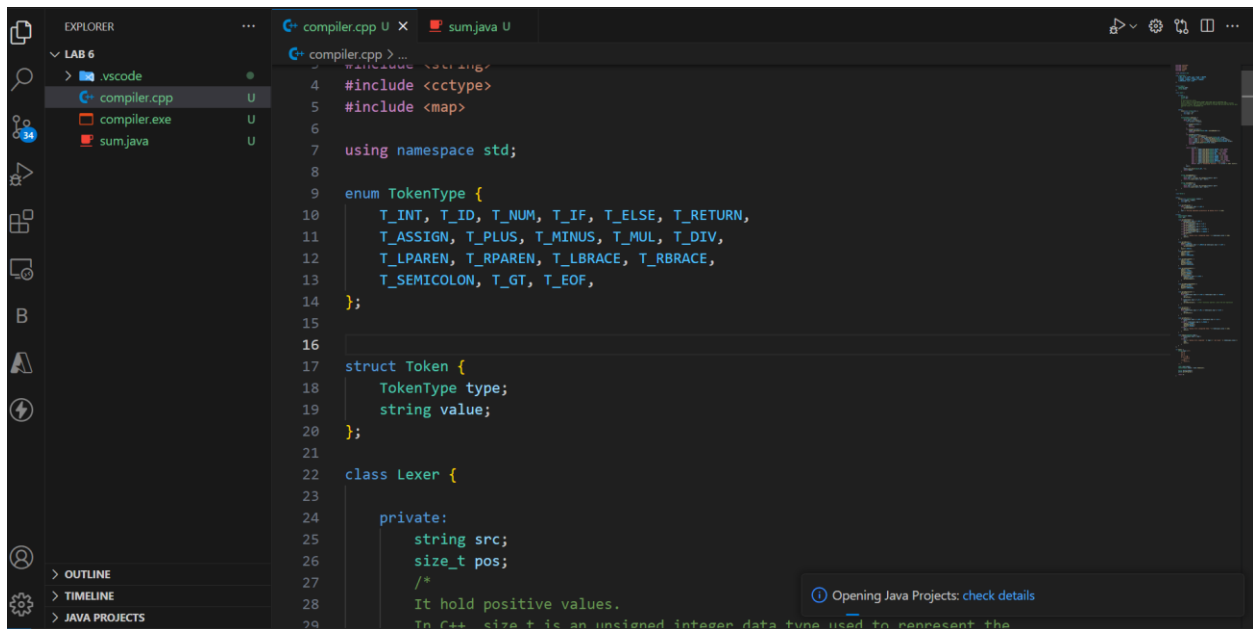
Compiler Construction Cs-471

Student Name: Fatiq Hussnain

Instructor: Mr. Laeeq khan Niazi

Lab 6

Code Snippets



```
1 // compiler.cpp
2 #include <iostream>
3 #include <string>
4 #include <cctype>
5 #include <map>
6
7 using namespace std;
8
9 enum TokenType {
10     T_INT, T_ID, T_NUM, T_IF, T_ELSE, T_RETURN,
11     T_ASSIGN, T_PLUS, T_MINUS, T_MUL, T_DIV,
12     T_LPAREN, T_RPAREN, T_LBRACE, T_RBRACE,
13     T_SEMICOLON, T_GT, T_EOF,
14 };
15
16
17 struct Token {
18     TokenType type;
19     string value;
20 };
21
22 class Lexer {
23
24     private:
25         string src;
26         size_t pos;
27         /*
28          * It hold positive values.
29          * In C++ size_t is an unsigned integer data type used to represent the
30          * size of an object in memory.
31          */
32     };
33
34     public:
35         Lexer(string s) {
36             src = s;
37             pos = 0;
38         }
39
40         Token getNextToken() {
41             Token t;
42             if (pos == src.length()) {
43                 t.type = T_EOF;
44                 return t;
45             }
46             char c = src[pos];
47             if (c == ' ' || c == '\n' || c == '\t') {
48                 pos++;
49                 return t;
50             }
51             if (c == '(' || c == ')' || c == '{' || c == '}') {
52                 t.type = T_LPAREN;
53                 t.value = c;
54                 pos++;
55                 return t;
56             }
57             if (c == ';' || c == ',' || c == '=' || c == '+' || c == '-' || c == '*' || c == '/') {
58                 t.type = T_ASSIGN;
59                 t.value = c;
60                 pos++;
61                 return t;
62             }
59         }
60     };
61
62     int main() {
63         string src = "int a = 10; if (a > 0) return 1; else return 0;";
64         Lexer l(src);
65         Token t;
66         while ((t = l.getNextToken()) != T_EOF) {
67             cout << t.type << " " << t.value << endl;
68         }
69         return 0;
70     }
```

```
22 class Lexer {
23
24 public:
25     Lexer(const string &src) {
26         this->src = src;
27         this->pos = 0;
28     }
29
30     vector<Token> tokenize() {
31         vector<Token> tokens;
32         while (pos < src.size()) {
33             char current = src[pos];
34
35             if (isspace(current)) {
36                 pos++;
37                 continue;
38             }
39             if (isdigit(current)) {
40                 tokens.push_back(Token{T_NUM, consumeNumber()});
41                 continue;
42             }
43             if (isalpha(current)) {
44                 string word = consumeWord();
45                 if (word == "int") tokens.push_back(Token{T_INT, word});
46                 else if (word == "if") tokens.push_back(Token{T_IF, word});
47                 else if (word == "else") tokens.push_back(Token{T_ELSE, word});
48                 else if (word == "return") tokens.push_back(Token{T_RETURN, word});
49             }
50         }
51     }
52
53     string consumeWord() {
54         string word;
55         while (isalpha(src[pos]) || src[pos] == '_') {
56             word += src[pos];
57             pos++;
58         }
59         return word;
60     }
61
62     string consumeNumber() {
63         string num;
64         while (isdigit(src[pos]) || src[pos] == '.') {
65             num += src[pos];
66             pos++;
67         }
68         return num;
69     }
70 }
```

Debug Java: Ready Share Code Link Generate Commit Message CRLF {} C++ Go Live AI Code Chat windows-gcc-x64 8:36 AM 10/13/2024

EXPLORER

- LAB 6
 - .vscode
 - compiler.cpp
 - compiler.exe
 - sum.java

```
22 class Lexer {
40     vector<Token> tokenize() {
62
63         switch (current) {
64             case '=': tokens.push_back(Token{T_ASSIGN, "="}); break;
65             case '+': tokens.push_back(Token{T_PLUS, "+"}); break;
66             case '-': tokens.push_back(Token{T_MINUS, "-"}); break;
67             case '*': tokens.push_back(Token{T_MUL, "*"}); break;
68             case '/': tokens.push_back(Token{T_DIV, "/"}); break;
69             case '(': tokens.push_back(Token{T_LPAREN, "("}); break;
70             case ')': tokens.push_back(Token{T_RPAREN, "}"); break;
71             case '{': tokens.push_back(Token{T_LBRACE, "{"}); break;
72             case '}': tokens.push_back(Token{T_RBRACE, "}"); break;
73             case ';': tokens.push_back(Token{T_SEMICOLON, ";"}); break;
74             case '>': tokens.push_back(Token{T_GT, ">"}); break;
75             default: cout << "Unexpected character: " << current << endl; exit(1);
76         }
77         pos++;
78     }
79     tokens.push_back(Token{T_EOF, ""});
80     return tokens;
81 }
82
83 string consumeNumber() {
84     size_t start = pos;
85     while (isdigit(src[pos]) || src[pos] == '.') {
86         pos++;
87     }
88     return src.substr(start, pos - start);
89 }
```

main*+ 3 1 Lab 6 Debug Java: Ready Share Code Link Generate Commit Message CRLF {} C++ Go Live AI Code Chat windows-gcc-x64 8:36 AM 10/13/2024

This screenshot shows the initial implementation of a Lexer and Parser in C++. The Explorer sidebar on the left shows a project named 'LAB 6' containing files 'compiler.cpp', 'compiler.exe', and 'sum.java'. The main editor window displays the code for 'compiler.cpp'.

```
22 class Lexer {
90     string consumeWord() {
91         int start = pos;
92         while (pos < src.size() && isalnum(src[pos])) pos++;
93         return src.substr(start, pos - start);
94     }
95 };
96
97
98 class Parser {
99
100
101 public:
102     Parser(const vector<Token> &tokens) {
103         this->tokens = tokens;
104         this->pos = 0;
105     }
106
107     void parseProgram() {
108         while (tokens[pos].type != T_EOF) {
109             parseStatement();
110         }
111         cout << "Parsing completed successfully! No Syntax Error" << endl;
112     }
113
114 private:
115     vector<Token> tokens;
```

The status bar at the bottom indicates the current file is 'main.cpp' and the active workspace is 'Lab 6'.

This screenshot shows the implementation of the 'parseStatement' and 'parseBlock' methods in the 'Parser' class. The Explorer sidebar on the left shows the same project structure as the first screenshot.

```
98 class Parser {
114 private:
115     vector<Token> tokens;
116     size_t pos;
117
118     void parseStatement() {
119         if (tokens[pos].type == T_INT) {
120             parseDeclaration();
121         } else if (tokens[pos].type == T_ID) {
122             parseAssignment();
123         } else if (tokens[pos].type == T_IF) {
124             parseIfStatement();
125         } else if (tokens[pos].type == T_RETURN) {
126             parseReturnStatement();
127         } else if (tokens[pos].type == T_LBRACE) {
128             parseBlock();
129         } else {
130             cout << "Syntax error: unexpected token " << tokens[pos].value << endl;
131             exit(1);
132         }
133     }
134
135     void parseBlock() {
136         expect(T_LBRACE);
137         while (tokens[pos].type != T_RBRACE && tokens[pos].type != T_EOF) {
138             parseStatement();
```

The status bar at the bottom indicates the current file is 'main.cpp' and the active workspace is 'Lab 6'.

```
U 134
U 135     void parseBlock() {
U 136         expect(T_LBRACE);
137         while (tokens[pos].type != T_RBRACE && tokens[pos].type != T_EOF) {
138             parseStatement();
139         }
140         expect(T_RBRACE);
141     }
142     void parseDeclaration() {
143         expect(T_INT);
144         expect(T_ID);
145         expect(T_SEMICOLON);
146     }
147
148     void parseAssignment() {
149         expect(T_ID);
150         expect(T_ASSIGN);
151         parseExpression();
152         expect(T_SEMICOLON);
153     }
154
155     void parseIfStatement() {
156         expect(T_IF);
157         expect(T_LPAREN);
158         parseExpression();
```

Debug 0 Java: Ready Share Code Link Generate Commit Message CRLF {} C++ Go Live AI Code Chat windows-gcc-x64 8:37 AM 10/13/2024

```
compiler.cpp > ...
98     class Parser {
U 155         void parseIfStatement() {
U 157             expect(T_LPAREN);
4, U 158             parseExpression();
159             expect(T_RPAREN);
160             parseStatement();
161             if (tokens[pos].type == T_ELSE) {
162                 expect(T_ELSE);
163                 parseStatement();
164             }
165         }
166
167         void parseReturnStatement() {
168             expect(T_RETURN);
169             parseExpression();
170             expect(T_SEMICOLON);
171         }
172
173         void parseExpression() {
174             parseTerm();
175             while (tokens[pos].type == T_PLUS || tokens[pos].type == T_MINUS) {
176                 pos++;
177                 parseTerm();
178             }
179             if (tokens[pos].type == T_GT) {
180                 pos++;
```

5 Debug 0 Java: Ready Share Code Link Generate Commit Message CRLF {} C++ Go Live AI Code Chat windows-gcc-x64 8:37 AM

```

98     class Parser {
206         void expect(TokenType type) {
213     }
214 };
215
216 int main() {
217     string input = R"(
218         int a;
219         a = 5;
220         int b;
221         b = a + 10;
222         if (b > 10) {
223             return b;
224         } else {
225             return 0;
226         }
227     )";
228
229     Lexer lexer(input);
230     vector<Token> tokens = lexer.tokenize();
231
232     Parser parser(tokens);
233     parser.parseProgram();
234
235     return 0;
236 }

```