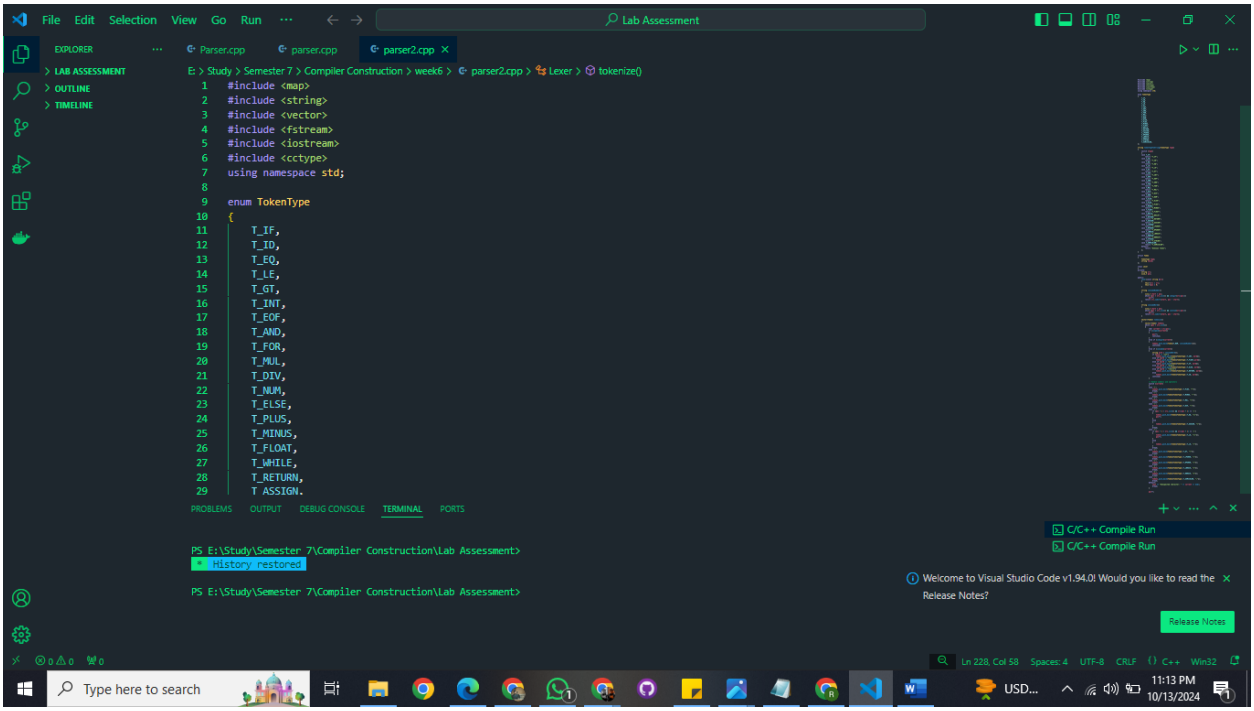
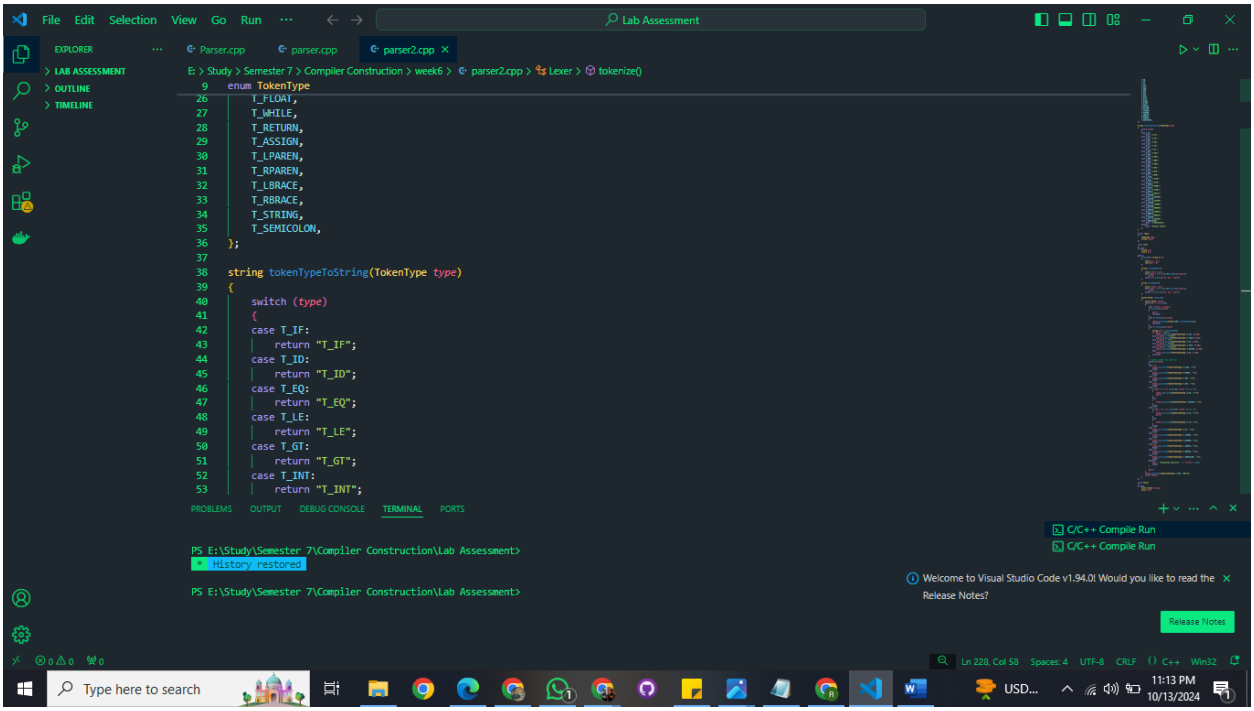
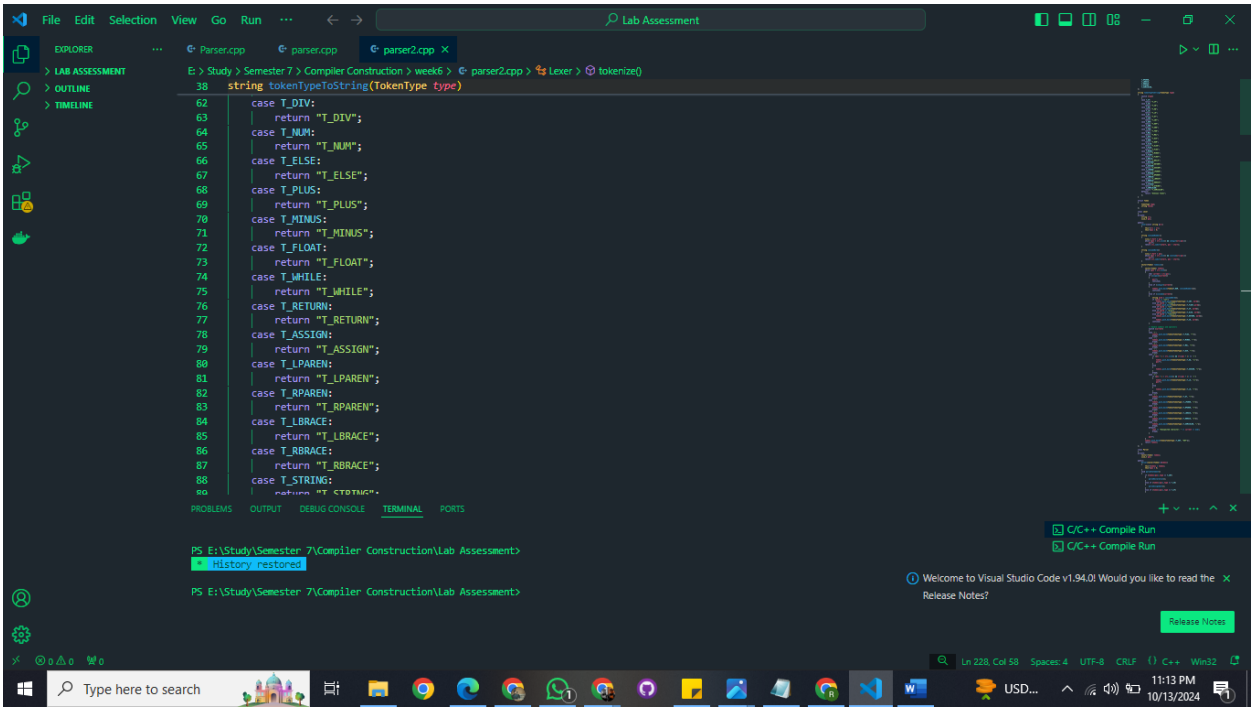
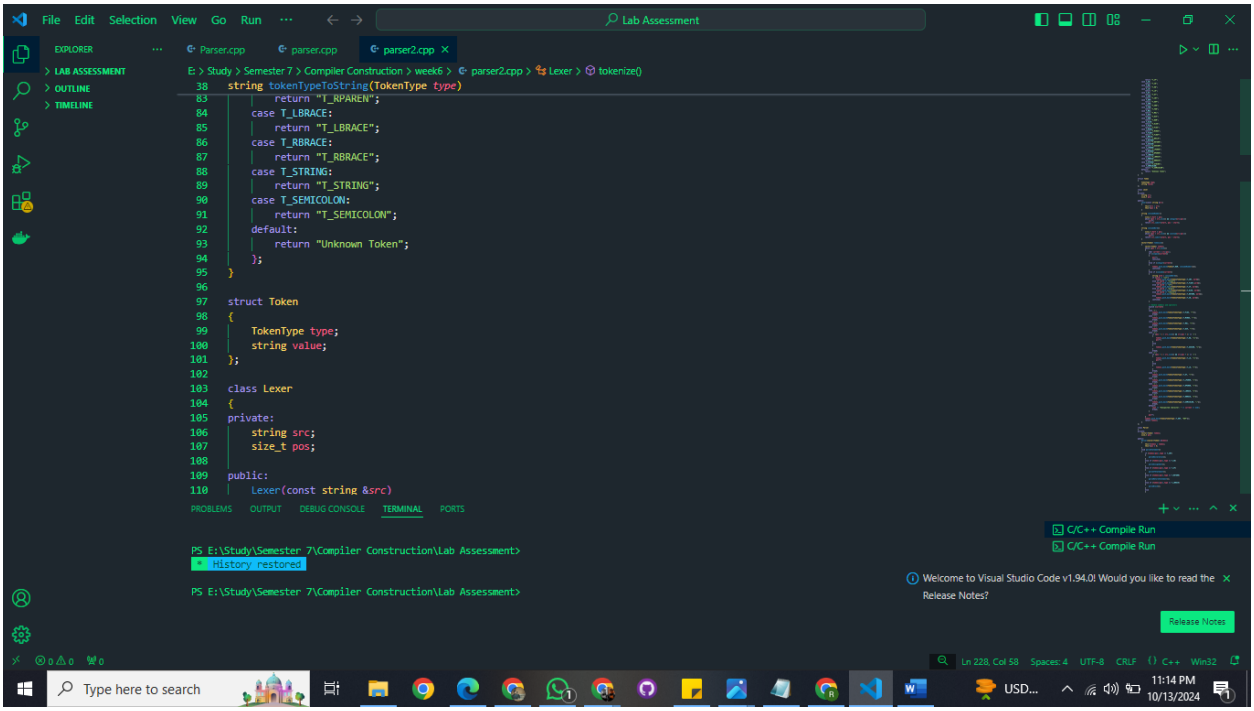
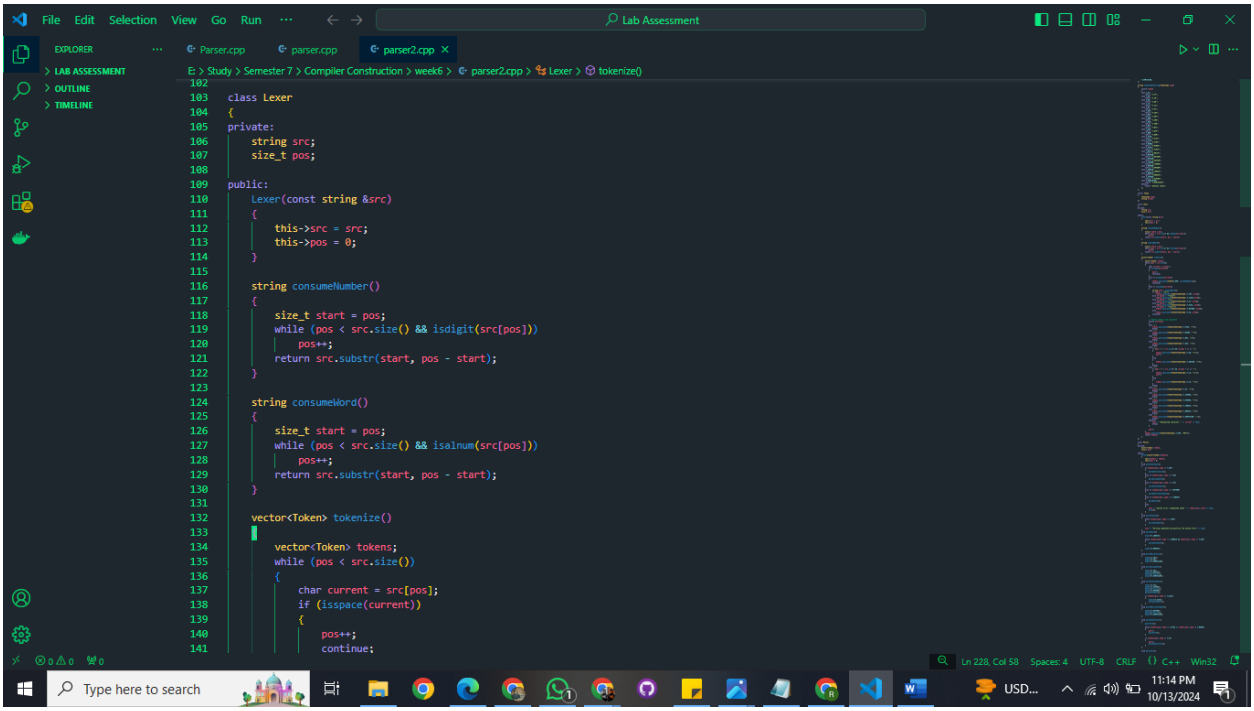
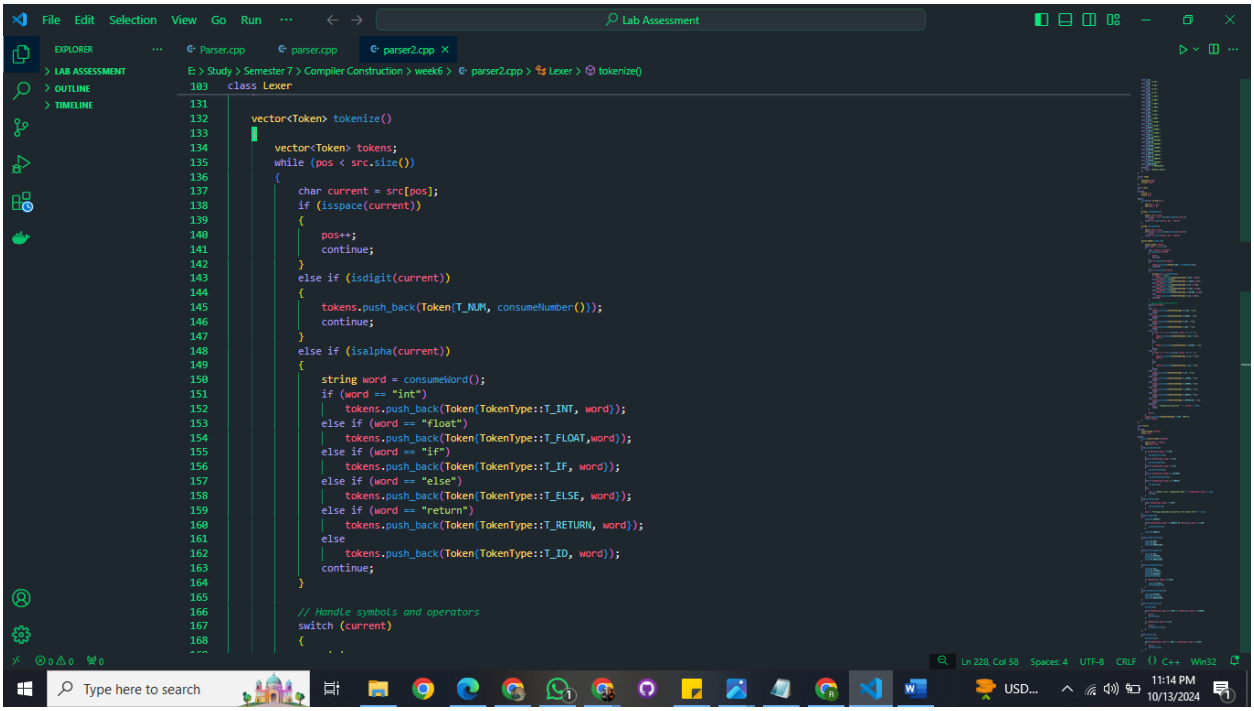

File Edit Selection View Go Run
Lab Assessment
EXPLORER
LAB ASSESSMENT
OUTLINE
TIMELINE
Parser.cpp
parser.cpp
parser2.cpp
E: > Study > Semester 7 > Compiler Construction > week6 > parser2.cpp > Lexer > tokenize()
#include <map>
#include <string>
#include <vector>
#include <fstream>
#include <iostream>
#include <cctype>
using namespace std;
enum TokenType
{
T_IF,
T_ID,
T_EQ,
T_LE,
T_GT,
T_INT,
T_EOF,
T_AND,
T_FOR,
T_MUL,
T_DIV,
T_NUM,
T_ELSE,
T_PLUS,
T_MINUS,
T_FLOAT,
T_WHILE,
T_RETURN,
T ASSIGN.
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS E:\Study\Semester 7\Compiler Construction\Lab Assessment>
History restored
PS E:\Study\Semester 7\Compiler Construction\Lab Assessment>
C/C++ Compile Run
C/C++ Compile Run
Welcome to Visual Studio Code v1.94.0! Would you like to read the Release Notes?
Release Notes
Ln 228, Col 58 Spaces: 4 UTF-8 CRLF C++ Win32
Type here to search
USD...
11:13 PM
10/13/2024

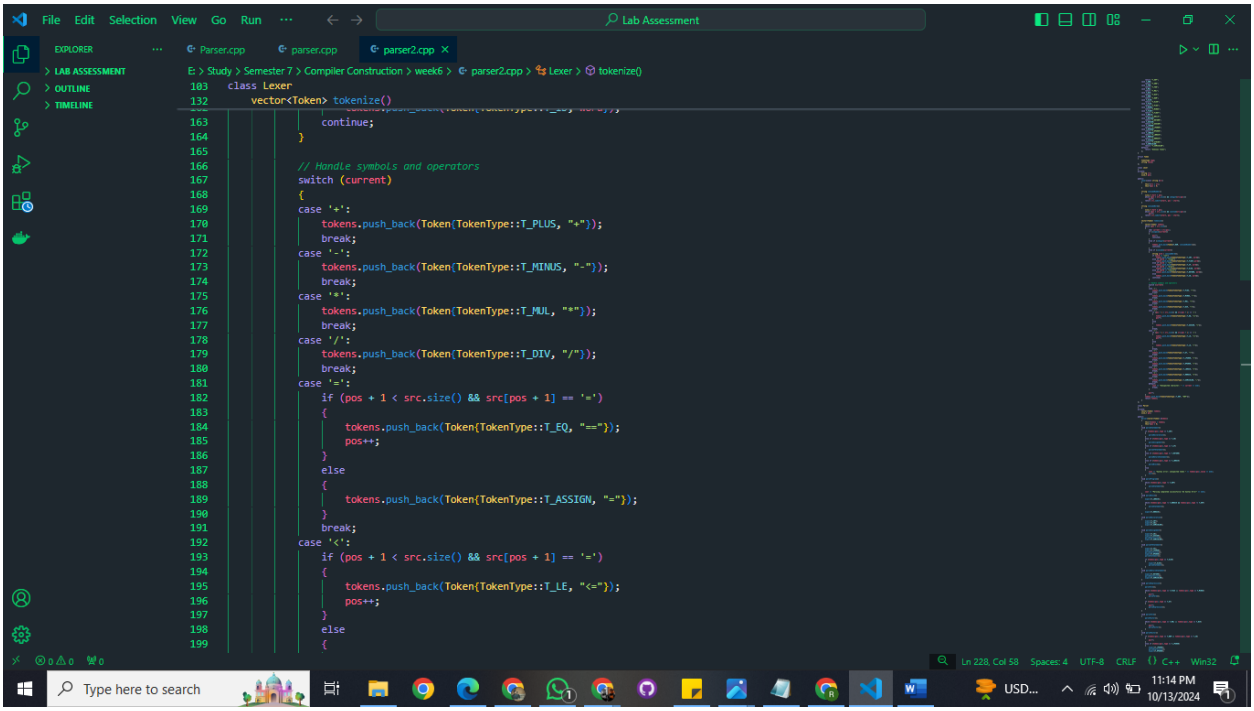

File Edit Selection View Go Run
Lab Assessment
EXPLORER
LAB ASSESSMENT
OUTLINE
TIMELINE
Parser.cpp
parser.cpp
parser2.cpp
E: > Study > Semester 7 > Compiler Construction > week6 > parser2.cpp > Lexer > tokenize()
enum TokenType
T_FLOAT,
T_WHILE,
T_RETURN,
T_ASSIGN,
T_LPAREN,
T_RPAREN,
T_LBRACE,
T_RBRACE,
T_STRING,
T_SEMICOLON,
};
string tokenTypeToString(TokenType type)
{
switch (type)
{
case T_IF:
return "T_IF";
case T_ID:
return "T_ID";
case T_EQ:
return "T_EQ";
case T_LE:
return "T_LE";
case T_GT:
return "T_GT";
case T_INT:
return "T_INT";
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS E:\Study\Semester 7\Compiler Construction\Lab Assessment>
History restored
PS E:\Study\Semester 7\Compiler Construction\Lab Assessment>
C/C++ Compile Run
C/C++ Compile Run
Welcome to Visual Studio Code v1.94.0! Would you like to read the Release Notes?
Release Notes
Ln 228, Col 58 Spaces: 4 UTF-8 CRLF C++ Win32
Type here to search
USD...
11:13 PM
10/13/2024

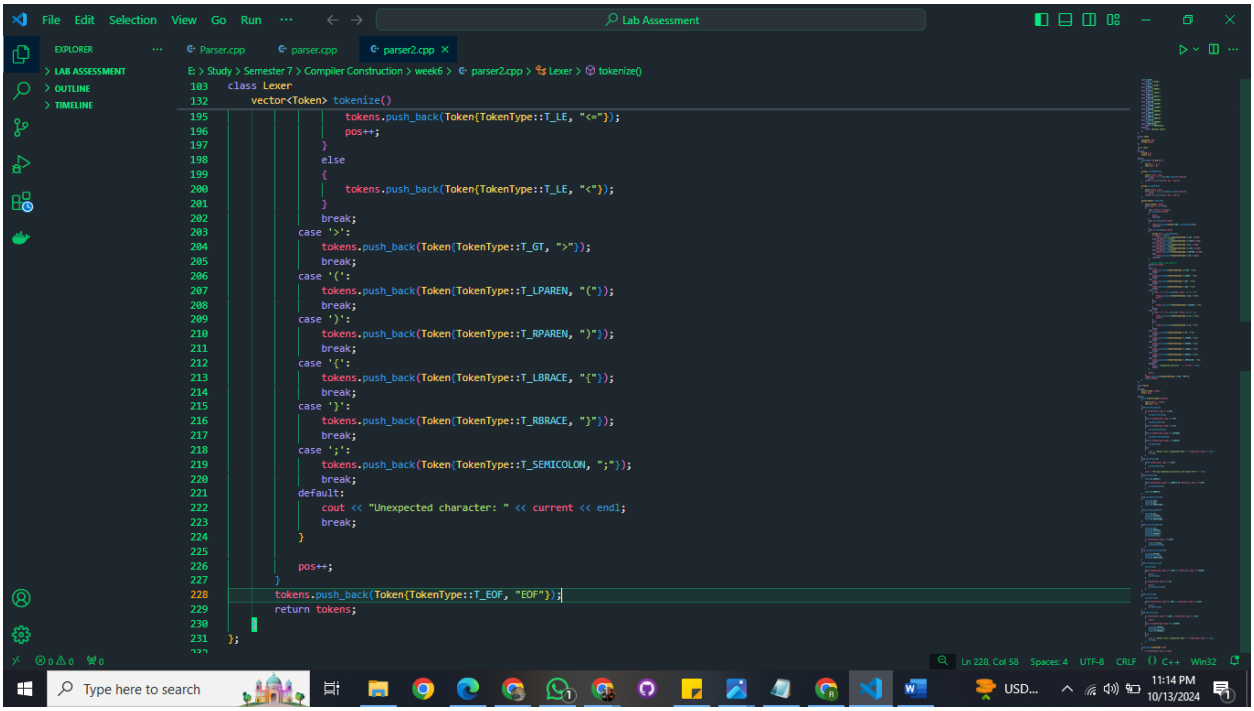

File Edit Selection View Go Run
Lab Assessment
EXPLORER
LAB ASSESSMENT
OUTLINE
TIMELINE
Parser.cpp
parser.cpp
parser2.cpp
E: > Study > Semester 7 > Compiler Construction > week6 > parser2.cpp > Lexer > tokenize()
string tokenTypeToString(TokenType type)
case T_DIV:
return "T_DIV";
case T_NUM:
return "T_NUM";
case T_ELSE:
return "T_ELSE";
case T_PLUS:
return "T_PLUS";
case T_MINUS:
return "T_MINUS";
case T_FLOAT:
return "T_FLOAT";
case T_WHILE:
return "T_WHILE";
case T_RETURN:
return "T_RETURN";
case T_ASSIGN:
return "T_ASSIGN";
case T_LPAREN:
return "T_LPAREN";
case T_RPAREN:
return "T_RPAREN";
case T_LBRACE:
return "T_LBRACE";
case T_RBRACE:
return "T_RBRACE";
case T_STRING:
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS E:\Study\Semester 7\Compiler Construction\Lab Assessment>
History restored
PS E:\Study\Semester 7\Compiler Construction\Lab Assessment>
C/C++ Compile Run
C/C++ Compile Run
Welcome to Visual Studio Code v1.94.0! Would you like to read the Release Notes?
Release Notes
Ln 228, Col 58 Spaces: 4 UTF-8 CRLF C++ Win32
Type here to search
USD...
11:13 PM
10/13/2024

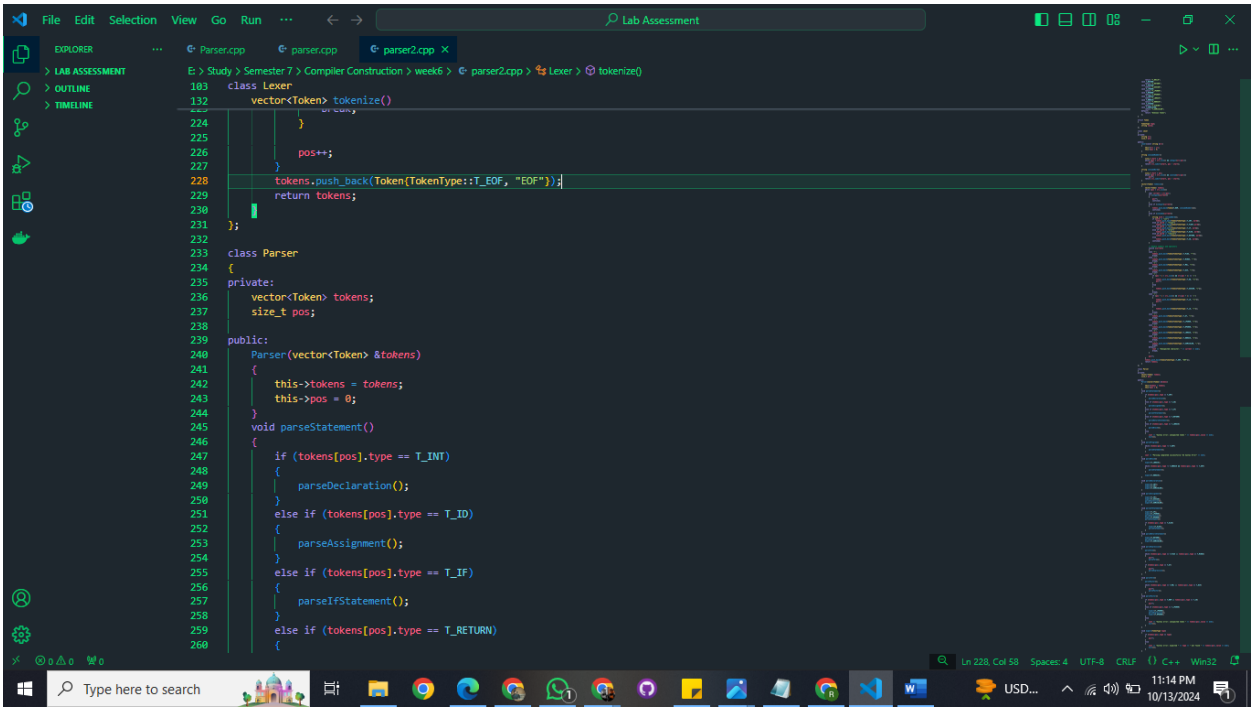

File Edit Selection View Go Run
Lab Assessment
EXPLORER
LAB ASSESSMENT
OUTLINE
TIMELINE
Parser.cpp
parser.cpp
parser2.cpp
E: > Study > Semester 7 > Compiler Construction > week6 > parser2.cpp > Lexer > tokenize()
string tokenTypeToString(TokenType type)
return "T_RPAREN";
case T_LBRACE:
return "T_LBRACE";
case T_RBRACE:
return "T_RBRACE";
case T_STRING:
return "T_STRING";
case T_SEMICOLON:
return "T_SEMICOLON";
default:
return "Unknown Token";
};
}
struct Token
{
TokenType type;
string value;
};
class Lexer
{
private:
string src;
size_t pos;
public:
Lexer(const string &src)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS E:\Study\Semester 7\Compiler Construction\Lab Assessment>
History restored
PS E:\Study\Semester 7\Compiler Construction\Lab Assessment>
C/C++ Compile Run
C/C++ Compile Run
Welcome to Visual Studio Code v1.94.0! Would you like to read the Release Notes?
Release Notes
Ln 228, Col 58 Spaces: 4 UTF-8 CRLF C++ Win32
Type here to search
USD...
11:14 PM
10/13/2024

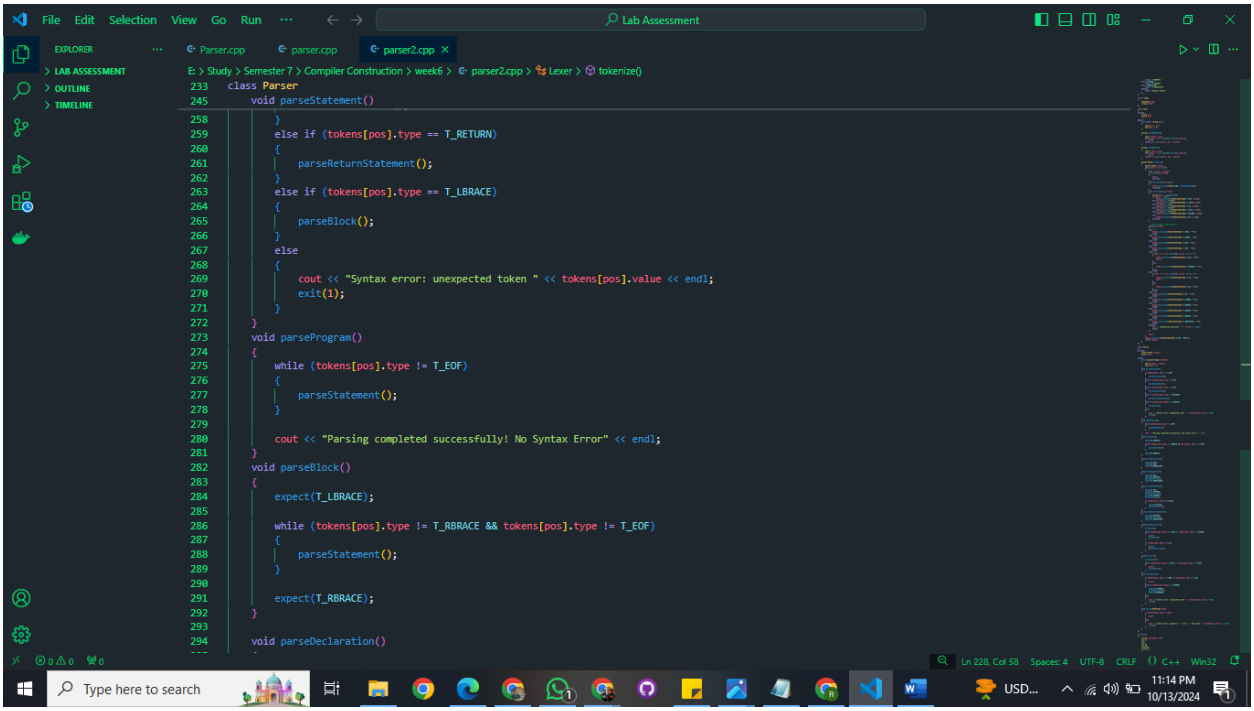

File Edit Selection View Go Run
Lab Assessment
EXPLORER
LAB ASSESSMENT
OUTLINE
TIMELINE
Parser.cpp
parser.cpp
parser2.cpp
E: > Study > Semester 7 > Compiler Construction > week6 > parser2.cpp > Lexer > tokenize()
class Lexer
{
private:
string src;
size_t pos;
public:
Lexer(const string &src)
{
this->src = src;
this->pos = 0;
}
string consumeNumber()
{
size_t start = pos;
while (pos < src.size() && isdigit(src[pos]))
pos++;
return src.substr(start, pos - start);
}
string consumeWord()
{
size_t start = pos;
while (pos < src.size() && isalnum(src[pos]))
pos++;
return src.substr(start, pos - start);
}
vector<Token> tokenize()
{
vector<Token> tokens;
while (pos < src.size())
{
char current = src[pos];
if (isspace(current))
{
pos++;
continue;
Ln 228, Col 58 Spaces: 4 UTF-8 CRLF C++ Win32
Type here to search
USD...
11:14 PM
10/13/2024

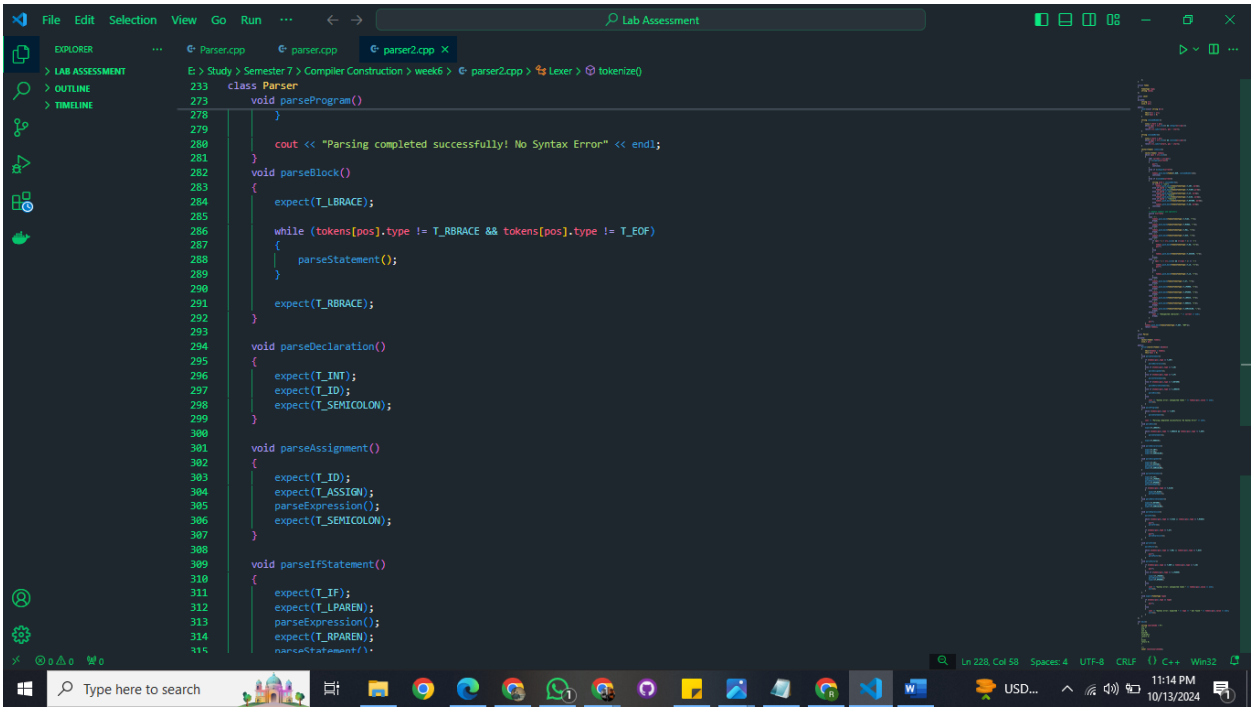

File Edit Selection View Go Run
Lab Assessment
EXPLORER
Parser.cpp
parser.cpp
parser2.cpp
LAB ASSESSMENT
OUTLINE
TIMELINE
E: > Study > Semester 7 > Compiler Construction > week6 > parser2.cpp > Lexer > tokenize()
103 class Lexer
131
132 vector<Token> tokenize()
133 {
134 vector<Token> tokens;
135 while (pos < src.size())
136 {
137 char current = src[pos];
138 if (isspace(current))
139 {
140 pos++;
141 continue;
142 }
143 else if (isdigit(current))
144 {
145 tokens.push_back(Token{T_NUM, consumeNumber()});
146 continue;
147 }
148 else if (isalpha(current))
149 {
150 string word = consumeWord();
151 if (word == "int")
152 tokens.push_back(Token{TokenType::T_INT, word});
153 else if (word == "float")
154 tokens.push_back(Token{TokenType::T_FLOAT,word});
155 else if (word == "if")
156 tokens.push_back(Token{TokenType::T_IF, word});
157 else if (word == "else")
158 tokens.push_back(Token{TokenType::T_ELSE, word});
159 else if (word == "return")
160 tokens.push_back(Token{TokenType::T_RETURN, word});
161 else
162 tokens.push_back(Token{TokenType::T_ID, word});
163 continue;
164 }
165
166 // Handle symbols and operators
167 switch (current)
168 {
Ln 228, Col 58 Spaces: 4 UTF-8 CRLF C++ Win32
Type here to search
USD...
11:14 PM
10/13/2024

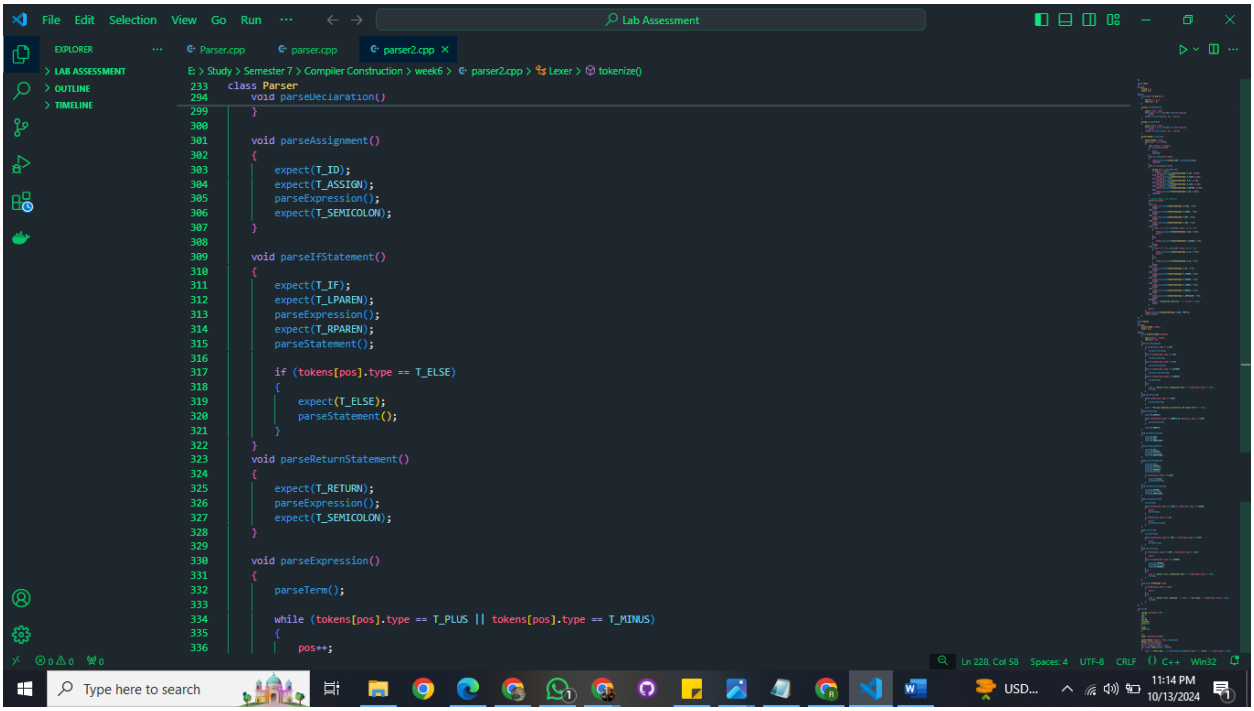

File Edit Selection View Go Run
Lab Assessment
EXPLORER
Parser.cpp
parser.cpp
parser2.cpp
LAB ASSESSMENT
OUTLINE
TIMELINE
E: > Study > Semester 7 > Compiler Construction > week6 > parser2.cpp > Lexer > tokenize()
103 class Lexer
132 vector<Token> tokenize()
163 continue;
164 }
165
166 // Handle symbols and operators
167 switch (current)
168 {
169 case '+':
170 tokens.push_back(Token{TokenType::T_PLUS, "+"});
171 break;
172 case '-':
173 tokens.push_back(Token{TokenType::T_MINUS, "-"});
174 break;
175 case '*':
176 tokens.push_back(Token{TokenType::T_MUL, "*"});
177 break;
178 case '/':
179 tokens.push_back(Token{TokenType::T_DIV, "/"});
180 break;
181 case '=':
182 if (pos + 1 < src.size() && src[pos + 1] == '=')
183 {
184 tokens.push_back(Token{TokenType::T_EQ, "=="});
185 pos++;
186 }
187 else
188 {
189 tokens.push_back(Token{TokenType::T_ASSIGN, "="});
190 }
191 break;
192 case '<':
193 if (pos + 1 < src.size() && src[pos + 1] == '=')
194 {
195 tokens.push_back(Token{TokenType::T_LE, "<="});
196 pos++;
197 }
198 else
199 {
Ln 228, Col 58 Spaces: 4 UTF-8 CRLF C++ Win32
Type here to search
USD...
11:14 PM
10/13/2024

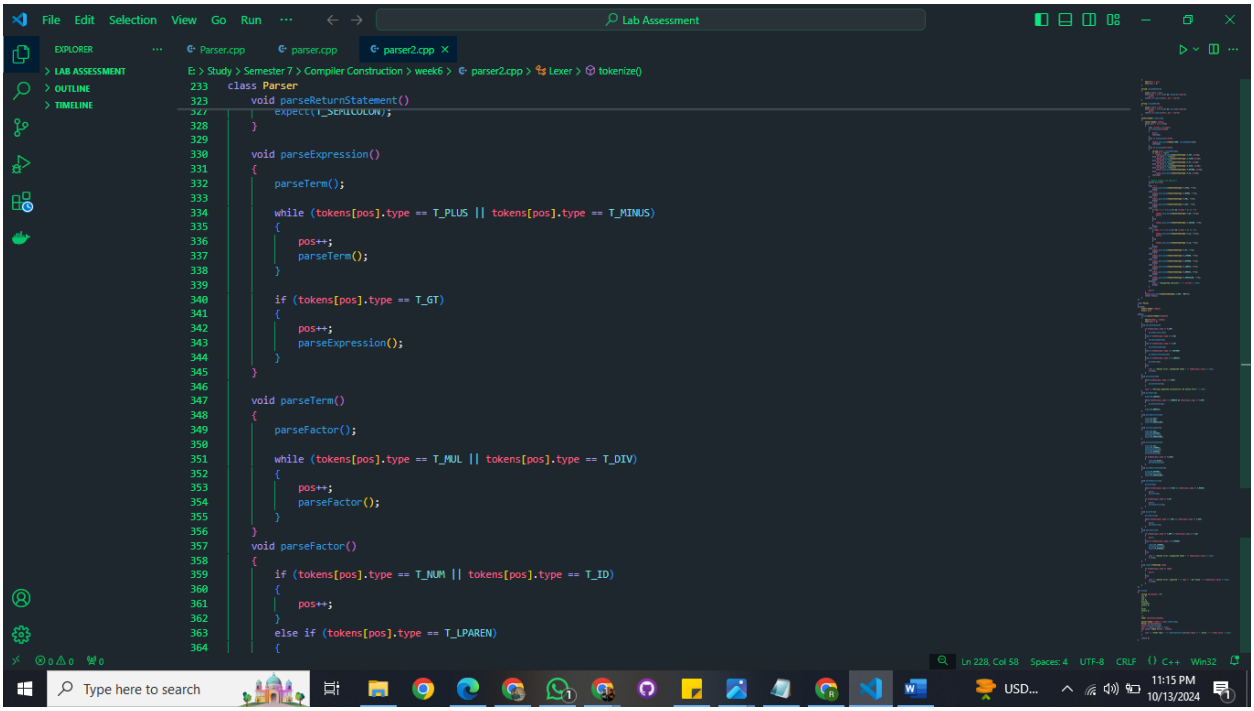

File Edit Selection View Go Run
Lab Assessment
EXPLORER
LAB ASSESSMENT
OUTLINE
TIMELINE
Parser.cpp
parser.cpp
parser2.cpp
E: > Study > Semester 7 > Compiler Construction > week6 > parser2.cpp > Lexer > tokenize()
class Lexer
vector<Token> tokenize()
tokens.push_back(Token{TokenType::T_LE, "<="});
pos++;
}
else
{
tokens.push_back(Token{TokenType::T_LE, "<"});
}
break;
case '>':
tokens.push_back(Token{TokenType::T_GT, ">"});
break;
case '(':
tokens.push_back(Token{TokenType::T_LPAREN, "("});
break;
case ')':
tokens.push_back(Token{TokenType::T_RPAREN, ")"});
break;
case '{':
tokens.push_back(Token{TokenType::T_LBRACE, "{"});
break;
case '}':
tokens.push_back(Token{TokenType::T_RBRACE, "}"});
break;
case ';':
tokens.push_back(Token{TokenType::T_SEMICOLON, ";"});
break;
default:
cout << "Unexpected character: " << current << endl;
break;
}
pos++;
}
tokens.push_back(Token{TokenType::T_EOF, "EOF"});
return tokens;
};
Ln 228, Col 58 Spaces: 4 UTF-8 CRLF C++ Win32
Type here to search
USD...
11:14 PM
10/13/2024

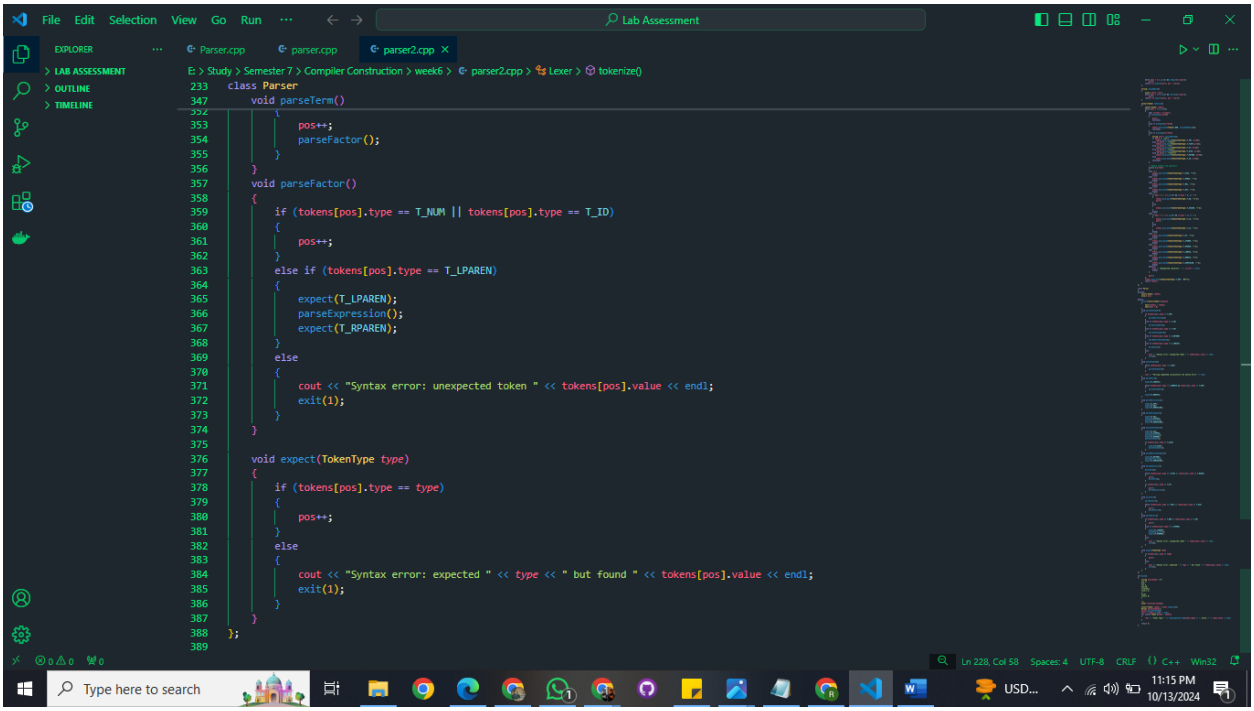

File Edit Selection View Go Run
Lab Assessment
EXPLORER
LAB ASSESSMENT
OUTLINE
TIMELINE
Parser.cpp
parser.cpp
parser2.cpp
E: > Study > Semester 7 > Compiler Construction > week6 > parser2.cpp > Lexer > tokenize()
class Lexer
vector<Token> tokenize()
}
pos++;
}
tokens.push_back(Token{TokenType::T_EOF, "EOF"});
return tokens;
};
class Parser
{
private:
vector<Token> tokens;
size_t pos;
public:
Parser(vector<Token> &tokens)
{
this->tokens = tokens;
this->pos = 0;
}
void parseStatement()
{
if (tokens[pos].type == T_INT)
{
parseDeclaration();
}
else if (tokens[pos].type == T_ID)
{
parseAssignment();
}
else if (tokens[pos].type == T_IF)
{
parseIfStatement();
}
else if (tokens[pos].type == T_RETURN)
{
Ln 228, Col 58 Spaces: 4 UTF-8 CRLF C++ Win32
Type here to search
USD...
11:14 PM
10/13/2024

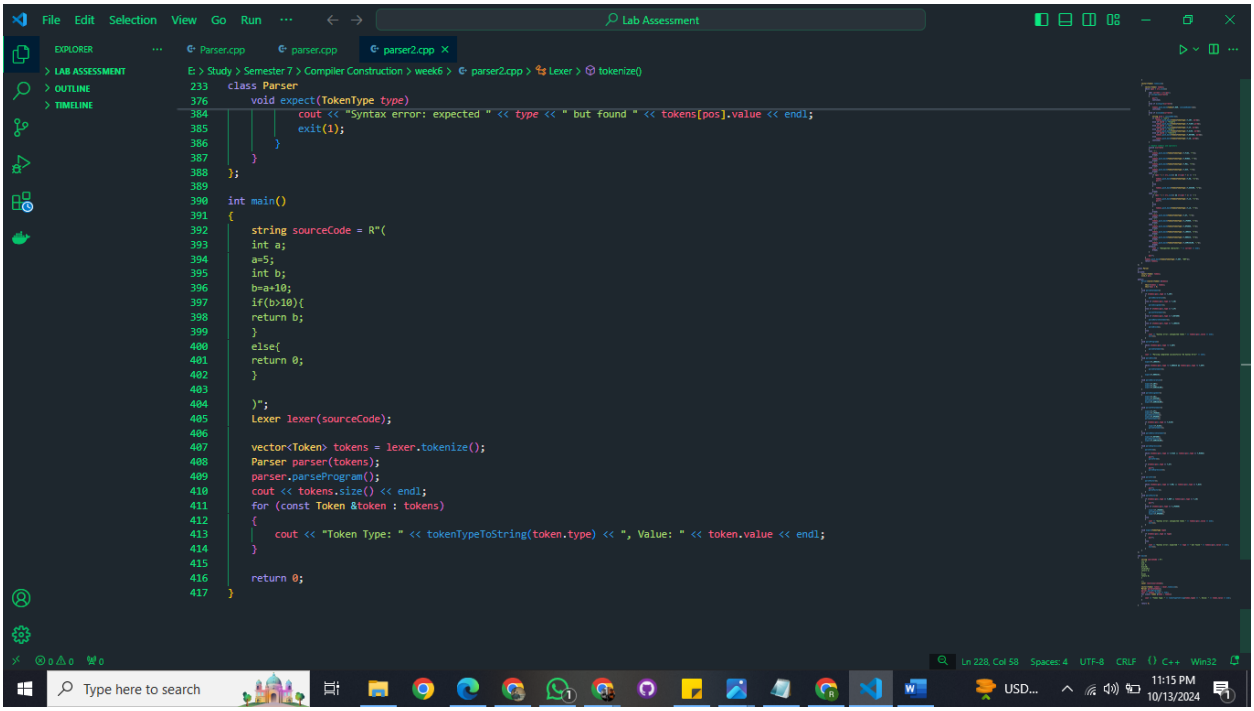

File Edit Selection View Go Run
Lab Assessment
EXPLORER
LAB ASSESSMENT
OUTLINE
TIMELINE
Parser.cpp
parser.cpp
parser2.cpp
E: > Study > Semester 7 > Compiler Construction > week6 > parser2.cpp > Lexer > tokenize()
class Parser
void parseStatement()
}
else if (tokens[pos].type == T_RETURN)
{
parseReturnStatement();
}
else if (tokens[pos].type == T_LBRACE)
{
parseBlock();
}
else
{
cout << "Syntax error: unexpected token " << tokens[pos].value << endl;
exit(1);
}
}
void parseProgram()
{
while (tokens[pos].type != T_EOF)
{
parseStatement();
}
cout << "Parsing completed successfully! No Syntax Error" << endl;
}
void parseBlock()
{
expect(T_LBRACE);
while (tokens[pos].type != T_RBRACE && tokens[pos].type != T_EOF)
{
parseStatement();
}
expect(T_RBRACE);
}
void parseDeclaration()
Ln 228, Col 58 Spaces: 4 UTF-8 CRLF C++ Win32
Type here to search
USD...
11:14 PM
10/13/2024


File Edit Selection View Go Run
Lab Assessment
EXPLORER
LAB ASSESSMENT
OUTLINE
TIMELINE
Parser.cpp
parser.cpp
parser2.cpp
E: > Study > Semester 7 > Compiler Construction > week6 > parser2.cpp > Lexer > tokenize()
class Parser
void parseProgram()
}
cout << "Parsing completed successfully! No Syntax Error" << endl;
}
void parseBlock()
{
expect(T_LBRACE);
while (tokens[pos].type != T_RBRACE && tokens[pos].type != T_EOF)
{
parseStatement();
}
expect(T_RBRACE);
}
void parseDeclaration()
{
expect(T_INT);
expect(T_ID);
expect(T_SEMICOLON);
}
void parseAssignment()
{
expect(T_ID);
expect(T_ASSIGN);
parseExpression();
expect(T_SEMICOLON);
}
void parseIfStatement()
{
expect(T_IF);
expect(T_LPAREN);
parseExpression();
expect(T_RPAREN);
Ln 228, Col 58 Spaces: 4 UTF-8 CRLF C++ Win32
Type here to search
USD...
11:14 PM
10/13/2024


File Edit Selection View Go Run
Lab Assessment
EXPLORER
Parser.cpp
parser.cpp
parser2.cpp
LAB ASSESSMENT
OUTLINE
TIMELINE
E: > Study > Semester 7 > Compiler Construction > week6 > parser2.cpp > Lexer > tokenize()
class Parser
void parseDeclaration()
}
void parseAssignment()
{
expect(T_ID);
expect(T_ASSIGN);
parseExpression();
expect(T_SEMICOLON);
}
void parseIfStatement()
{
expect(T_IF);
expect(T_LPAREN);
parseExpression();
expect(T_RPAREN);
parseStatement();
if (tokens[pos].type == T_ELSE)
{
expect(T_ELSE);
parseStatement();
}
}
void parseReturnStatement()
{
expect(T_RETURN);
parseExpression();
expect(T_SEMICOLON);
}
void parseExpression()
{
parseTerm();
while (tokens[pos].type == T_PLUS || tokens[pos].type == T_MINUS)
{
pos++;
Ln 228, Col 58
Spaces: 4
UTF-8
CRLF
C++
Win32
Type here to search
USD...
11:14 PM
10/13/2024


File Edit Selection View Go Run
Lab Assessment
EXPLORER
Parser.cpp
parser.cpp
parser2.cpp
LAB ASSESSMENT
OUTLINE
TIMELINE
E: > Study > Semester 7 > Compiler Construction > week6 > parser2.cpp > Lexer > tokenize()
class Parser
void parseReturnStatement()
}
void parseExpression()
{
parseTerm();
while (tokens[pos].type == T_PLUS || tokens[pos].type == T_MINUS)
{
pos++;
parseTerm();
}
if (tokens[pos].type == T_GT)
{
pos++;
parseExpression();
}
}
void parseTerm()
{
parseFactor();
while (tokens[pos].type == T_MUL || tokens[pos].type == T_DIV)
{
pos++;
parseFactor();
}
}
void parseFactor()
{
if (tokens[pos].type == T_NUM || tokens[pos].type == T_ID)
{
pos++;
}
else if (tokens[pos].type == T_LPAREN)
{
Ln 228, Col 58
Spaces: 4
UTF-8
CRLF
C++
Win32
Type here to search
USD...
11:15 PM
10/13/2024


File Edit Selection View Go Run
Lab Assessment
EXPLORER
LAB ASSESSMENT
OUTLINE
TIMELINE
Parser.cpp
parser.cpp
parser2.cpp
E: > Study > Semester 7 > Compiler Construction > week6 > parser2.cpp > Lexer > tokenize()
class Parser
void parseTerm()
pos++;
parseFactor();
void parseFactor()
if (tokens[pos].type == T_NUM || tokens[pos].type == T_ID)
pos++;
else if (tokens[pos].type == T_LPAREN)
expect(T_LPAREN);
parseExpression();
expect(T_RPAREN);
else
cout << "Syntax error: unexpected token " << tokens[pos].value << endl;
exit(1);
void expect(TokenType type)
if (tokens[pos].type == type)
pos++;
else
cout << "Syntax error: expected " << type << " but found " << tokens[pos].value << endl;
exit(1);
};
Ln 228, Col 58 Spaces: 4 UTF-8 CRLF C++ Win32
Type here to search
USD...
11:15 PM
10/13/2024


File Edit Selection View Go Run
Lab Assessment
EXPLORER
LAB ASSESSMENT
OUTLINE
TIMELINE
Parser.cpp
parser.cpp
parser2.cpp
E: > Study > Semester 7 > Compiler Construction > week6 > parser2.cpp > Lexer > tokenize()
class Parser
void expect(TokenType type)
cout << "Syntax error: expected " << type << " but found " << tokens[pos].value << endl;
exit(1);
};
int main()
{
string sourceCode = R"(
int a;
a=5;
int b;
b=a+10;
if(b>10){
return b;
}
else{
return 0;
}
)";
Lexer lexer(sourceCode);
vector<Token> tokens = lexer.tokenize();
Parser parser(tokens);
parser.parseProgram();
cout << tokens.size() << endl;
for (const Token &token : tokens)
{
cout << "Token Type: " << tokenTypeToString(token.type) << ", Value: " << token.value << endl;
}
return 0;
}
Ln 228, Col 58 Spaces: 4 UTF-8 CRLF C++ Win32
Type here to search
USD...
11:15 PM
10/13/2024