Final Lab Report

Akash Mojumder

21-45352-2

Sec: G

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <cctype>
#include <algorithm>
using namespace std;

enum TokenType {
   KEYWORD,
   IDENTIFIER,
   INTEGER,
   FLOAT,
   STRING,
   OPERATOR,
   DELIMITER,
   COMMENT,
   WHITESPACE,
   UNKNOWN
};

struct Token {
    string value;
    TokenType type;
};
bool isKeyword(const string& token) {
    string keywords[] = {"int", "float", "if", "else", "while", "for", "return"};
    return find(begin(keywords), end(keywords), token) != end(keywords);
}

bool isIdentifier(char c) {
    return isalpha(c) || c == '_' || c == '#';
}

bool isNumber(const string& token) {
    if (token.empty()) {
        return false;
    }

    size_t start = (token[0] == '-') ? 1 : 0;
    return all_of(token.begin() + start, token.end(), [](char c) { return isdigit(c); });
}

bool isOperator(char c) {
    string operators = "+-*/=<>&|!";
    return operators.find(c) != string::npos;
}

bool isDelimiter(char c) {
    string delimiters = "{}[],;()";
    return delimiters.find(c) != string::npos;
}

bool isWhitespace(char c) {
    return isspace(c);
}

bool isComment(const string& token) {
    return token.size() >= 2 && token.substr(0, 2) == "//";
}

Token getTokenType(const string& token) {
    Token t;
    if (isKeyword(token)) {
        t.value = token;
        t.type = KEYWORD;
```

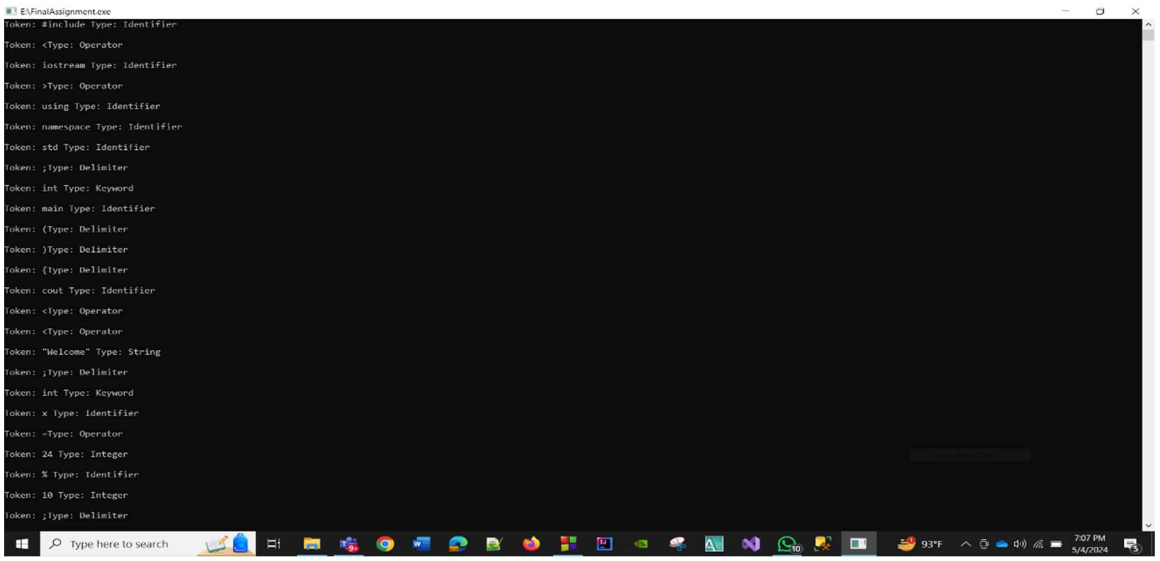
```
} else if (isNumber(token)) {
    t.value = token;
    if (token.find('.') != string::npos) {
        t.type = FLOAT;
    } else {
        t.type = INTEGER;
    }
} else if (token.size() >= 2 && token.front() == '"' && token.back() == '"') {
    t.value = token;
    t.type = STRING;
} else if (isOperator(token[0])) {
    t.value = token;
    t.type = OPERATOR;
} else if (isDelimiter(token[0])) {
    t.value = token;
    t.type = DELIMITER;
} else if (isComment(token)) {
    t.value = token;
    t.type = COMMENT;
} else if (all_of(token.begin(), token.end(), isWhitespace)) {
    t.value = token;
    t.type = WHITESPACE;
    } else {
        t.value = token;
        t.type = IDENTIFIER;
    }
    return t;
}

int main() {
    ifstream file("chck.txt");
    if (!file) {
        cerr << "Error opening file.\n";
        return 1;
    }

    char ch;
    string token;

    while (file.get(ch)) {
        if (isWhitespace(ch) || isDelimiter(ch) || isOperator(ch)) {
            if (!token.empty()) {
                Token t = getTokenType(token);
                cout << "Token: " << t.value << " Type: ";
```

```
switch (t.type) {
    case KEYWORD:
        cout << "Keyword";
        break;
    case IDENTIFIER:
        cout << "Identifier";
        break;
    case INTEGER:
        cout << "Integer";
        break;
    case FLOAT:
        cout << "Float";
        break;
    case STRING:
        cout << "String";
        break;
    case OPERATOR:
        cout << "Operator";
        break;
    case DELIMITER:
        cout << "Delimiter";
        break;
        case COMMENT:
            cout << "Comment";
            break;
        case WHITESPACE:
            cout << "Whitespace";
            break;
        case UNKNOWN:
            cout << "Unknown";
            break;
    }
    cout << endl;
    cout<<"\n";
    token.clear();
}
if (isDelimiter(ch) || isOperator(ch)) {
    token.push_back(ch);
    Token t = getTokenType(token);
    cout << "Token: " << t.value << "Type: ";
    switch (t.type) {
        case KEYWORD:
            cout << "Keyword";
            break;
```

```
case IDENTIFIER:
   cout << "Identifier";
   break;
case INTEGER:
   cout << "Integer";
   break;
case FLOAT:
   cout << "Float";
   break;
case STRING:
   cout << "String";
   break;
case OPERATOR:
   cout << "Operator";
   break;
case DELIMITER:
   cout << "Delimiter";
   break;
case COMMENT:
   cout << "Comment";
   break;
case WHITESPACE:
```

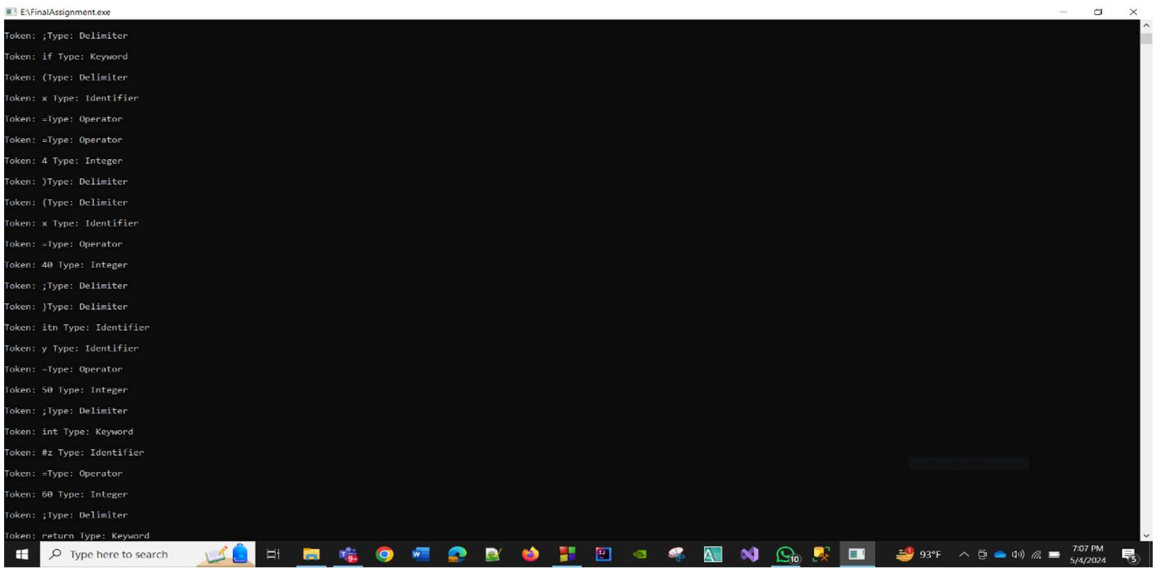
```
                    cout << "Whitespace";
                    break;
                case UNKNOWN:
                    cout << "Unknown";
                    break;
            }
            cout << endl;
            cout<<"\n";
            token.clear();
        }
    } else {
        token.push_back(ch);
    }
  }
  return 0;
}
```

P.t.o



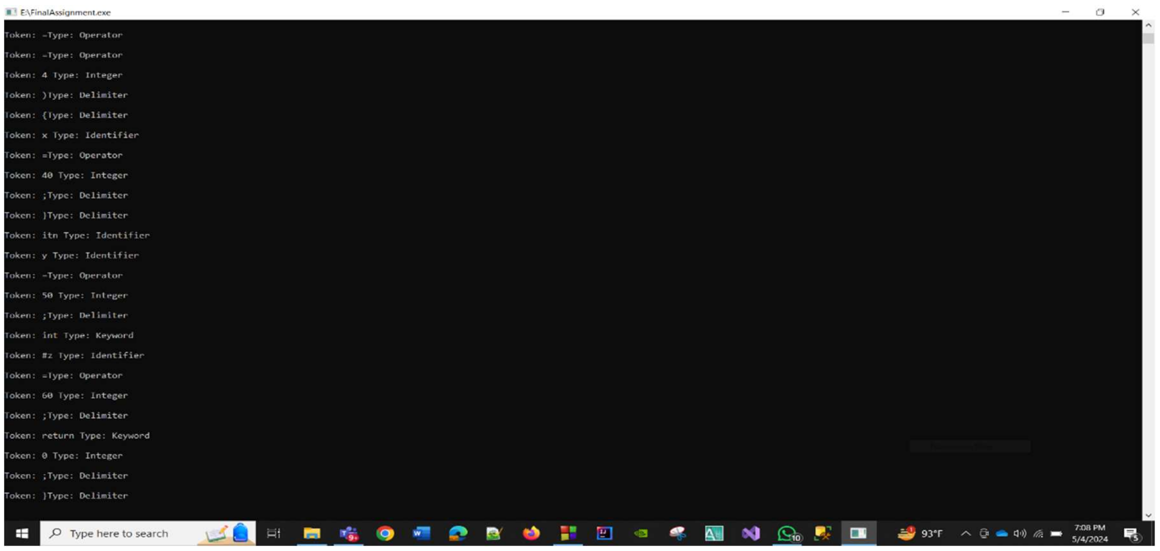
E:\FinalAssignment.exe

```
Token: #include Type: Identifier

Token: <Type: Operator

Token: iostream Type: Identifier

Token: >Type: Operator

Token: using Type: Identifier

Token: namespace Type: Identifier

Token: std Type: Identifier

Token: ;Type: Delimiter

Token: int Type: Keyword

Token: main Type: Identifier

Token: (Type: Delimiter

Token: )Type: Delimiter

Token: {Type: Delimiter

Token: cout Type: Identifier

Token: <Type: Operator

Token: <Type: Operator

Token: "Welcome" Type: String

Token: ;Type: Delimiter

Token: int Type: Keyword

Token: x Type: Identifier

Token: =Type: Operator

Token: 24 Type: Integer

Token: % Type: Identifier

Token: 10 Type: Integer

Token: ;Type: Delimiter
```



E:\FinalAssignment.exe

```
Token: ;Type: Delimiter

Token: if Type: Keyword

Token: (Type: Delimiter

Token: x Type: Identifier

Token: =Type: Operator

Token: =Type: Operator

Token: 4 Type: Integer

Token: )Type: Delimiter

Token: {Type: Delimiter

Token: x Type: Identifier

Token: =Type: Operator

Token: 40 Type: Integer

Token: ;Type: Delimiter

Token: }Type: Delimiter

Token: itn Type: Identifier

Token: y Type: Identifier

Token: =Type: Operator

Token: 50 Type: Integer

Token: ;Type: Delimiter

Token: int Type: Keyword

Token: #z Type: Identifier

Token: =Type: Operator

Token: 60 Type: Integer

Token: ;Type: Delimiter

Token: return Type: Keyword
```



E:\FinalAssignment.exe

```
Token: =Type: Operator

Token: =Type: Operator

Token: 4 Type: Integer

Token: )Type: Delimiter

Token: {Type: Delimiter

Token: x Type: Identifier

Token: =Type: Operator

Token: 40 Type: Integer

Token: ;Type: Delimiter

Token: }Type: Delimiter

Token: itn Type: Identifier

Token: y Type: Identifier

Token: =Type: Operator

Token: 50 Type: Integer

Token: ;Type: Delimiter

Token: int Type: Keyword

Token: #z Type: Identifier

Token: =Type: Operator

Token: 60 Type: Integer

Token: ;Type: Delimiter

Token: return Type: Keyword

Token: 0 Type: Integer

Token: ;Type: Delimiter

Token: }Type: Delimiter
```