Lab 4

Session: 2021 – 2024

Submitted by:

Uswa Arif

2021-CS-77

Submitted To:

Laeeq Khan Niazi

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Task 1:

Code:

tokenizer.cpp | first.lex | lexical.cpp

D: > 7 semester > CC Lab > Lab4 > first.lex

```
%{
#include <stdio.h>
%}

%%
if|else|printf|for|while|return|int|float|char|void|double      {printf("%s is a keyword\n", yytext);}
[0-9]+                                                          {printf("%s is a number\n", yytext);}
[a-zA-Z]+                                                       {printf("%s is a word\n", yytext);}
"+"|"-"|"*"|"/"|"="|">"|"<"|"&"|"|"                             {printf("%s is an operator\n", yytext);}
"{"|"}"|"("|")"|";"|","                                         {printf("%s is a punctuation\n", yytext);}
[ \t\n]                                                         ; // skip whitespace (spaces, tabs, newlines)
.                                                               {printf("Unknown character: %s\n", yytext);}
%%

int main(){
    printf("\nEnter the string here:\n");
    yylex(); //Call the lexical analyzer to start scanning input
    return 0;
}

int yywrap(){
    return 1;
}
```

Output:

```
D:\7 semester\CC Lab\Lab4>flex first.lex

D:\7 semester\CC Lab\Lab4>g++ lex.yy.c -o lexer.exe

D:\7 semester\CC Lab\Lab4>lexer.exe

Enter the string here:
int main()
int is a keyword
main is a word
( is a punctuation
) is a punctuation
```

Task 2:

Code:

```cpp
#include<iostream>
#include<vector>
#include<cctype>
#include<string>
#include <algorithm>
using namespace std;

enum TokenType
{
    KEYWORD,
    IDENTIFIER,
    NUMBER,
    OPERATOR,
    PUNCTUATION,
    UNKNOWN
};

struct Token
{
    string token;
    TokenType type;
};

bool isOperator(char c)
{
    return c == '+' || c == '-' || c == '*' || c == '/' || c == '=' || c == '!' || c == '<' || c == '>';
}

bool isPunctuation(char c)
{
    return c == ';' || c == ',' || c == '.' || c == '(' || c == ')' || c == '{' || c == '}' || c == '[' || c == ']';
}

bool isKeyword(string str)
{
    std::vector<string> keywords = {"int", "return", "if", "else", "while", "for", "main"};
```

```cpp
bool isKeyword(string str)
    return find(keywords.begin(), keywords.end(), str) != keywords.end();
}

vector<Token> tokenize(string input)
{
    vector<Token> tokens;

    for(int i=0; i<input.length(); i++)
    {
        char current = input[i];
        if (isspace(current))
        {
            i++;
            continue;
        }

        if(isalpha(current))
        {
            string lexeme;
            while (i < input.length() && (isalnum(input[i]) || input[i] == '_'))
            {
                lexeme += input[i];
                i++;
            }
            if (isKeyword(lexeme))
            {
                tokens.push_back({lexeme, KEYWORD});
            } else
            {
                tokens.push_back({lexeme, IDENTIFIER});
            }
            continue;
        }

        if (isdigit(current))
```

```
73          {
74              string number;
75              while (i < input.length() && isdigit(input[i]))
76              {
77                  number += input[i];
78                  i++;
79              }
80              tokens.push_back({number, NUMBER});
81              continue;
82          }
83
84          if(isOperator(current))
85          {
86              string operatorToken(1, current);
87              tokens.push_back({operatorToken, OPERATOR});
88              i++;
89              continue;
90          }
91
92          if (isPunctuation(current))
93          {
94              string punctuationToken(1, current);
95              tokens.push_back({punctuationToken, PUNCTUATION});
96              i++;
97              continue;
98          }
99
100
101         string unknownToken(1, current);
102         tokens.push_back({unknownToken, UNKNOWN});
103         i++;
104     }
105
106     return tokens;
107 }
```

```cpp
int main()
{
    string code = "int main() { int a = 10; return a; }";

    vector<Token> tokens = tokenize(code);

    for (const auto& token : tokens)
    {
        cout << "Token: " << token.token << ", Type: ";
        switch (token.type)
        {
            case KEYWORD: cout << "Keyword"; break;
            case IDENTIFIER: cout << "Identifier"; break;
            case NUMBER: cout << "Number"; break;
            case OPERATOR: cout << "Operator"; break;
            case PUNCTUATION: cout << "Punctuation"; break;
            case UNKNOWN: cout << "Unknown"; break;
        }
        cout << endl;
    }

    return 0;
}
```

Output:

```
D:\7 semester\CC Lab\Lab4>tokenizer.exe
Token: int, Type: Keyword
Token: main, Type: Keyword
Token: ), Type: Punctuation
Token: {, Type: Punctuation
Token: int, Type: Keyword
Token: a, Type: Identifier
Token: =, Type: Operator
Token: 10, Type: Number
Token: eturn, Type: Identifier
Token: a, Type: Identifier
```