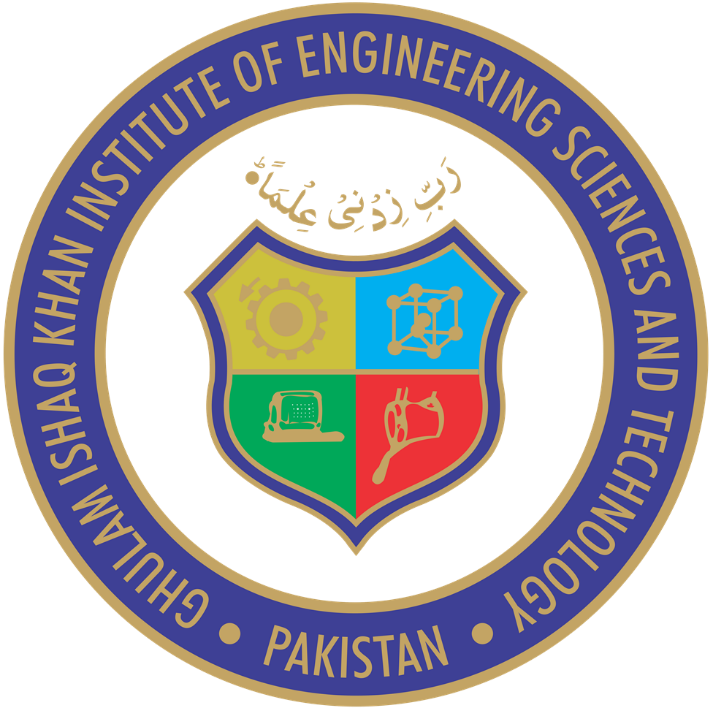
**CS-224 Automata Theory**

**Semester Project Report**



**لغوی تجزیہ کار**

**Members:**

* Mohammad Omar Khan **(2021305)**
* Muhammad Umer Tayyab **(2021306)**
* Muhammad Abdullah **(2021317)**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **1.** | **Lexical Analyzer** | **Page 3** |
| **2.** | **Samples and Results** |  |
|  | **2.1. Sample 1** | **Page 7** |
|  | **2.2. Result 1** | **Page 8** |
|  | **2.3. Sample 2** | **Page 10** |
|  | **2.4. Result 2** | **Page 11** |
|  | **2.3. Sample 3** | **Page 13** |
|  | **2.4. Result 3** | **Page 13** |
|  |  |  |
|  |  |  |
|  |  |  |

**Lexical Analyzer**

**Explanation:**

* The name of the LA is **تجزیہ کار**  **لغوی**
* The LA contains 26 different tokens, for e.g. comment, variable, function etc.
* The LA will read a sample file, and identify each token.
* Each token is the stored in a separate file, along with the token ID and the line number.

**Code:**

%{

    int lineNum = 0;

    void printResultsToFile(char \*TextType, char \*Text)

    {

        FILE \*fptr;

        fptr = fopen("results.txt", "a+");

        fprintf(fptr, "\t%sLine Number: %i\t\t%s\n", TextType, lineNum, Text);

        fclose(fptr);

    }

%}

Delimitor [ \t;]\*

Header "#include "[<"].\*[>"]

Namespace "using namespace std"

ReservedWord if|else|for|while|do|delete|new|main|return|private|public|case|switch|break|cout|cin|endl|virtual

Digit [0-9]

UC\_Alphabet [A-Z]

LC\_Alphabet [a-z]

Alphabet {UC\_Alphabet}|{LC\_Alphabet}

Symbol [\_.\-/$]

Dot \.

DataType int|string|void|float|char|double|bool|"unsigned int"

Bool true|false|0|1

Int [+-]?{Digit}+

Float [+-]?{Digit}\*{Dot}{Digit}\*

Char {Alphabet}|{Digit}|{Symbol}

String \"[^\n"]+\"

Bracket [()\[\]{}]

Operator [+\-\\*/\^%=]|"<<"|">>"

SingleLineComment "//".\*

MultiLineComment "/\*"(.|\n)\*?"\*/"

Comment {SingleLineComment}|{MultiLineComment}

Var {Alphabet}\*|{Digit}\*|{Symbol}\*

Variable {Alphabet}{Var}\*

Array {Variable}\[.\*\]

ConditionUtility [><]|">="|"<="|"=="|"!="

Number {Digit}+

Condition \({Variable}" "{ConditionUtility}" "{Variable}\)|\({Variable}" "{ConditionUtility}" "{Number}\)|

DeclareFunction {DataType}" "{Variable}\(.\*\)

FunctionCall {Variable}\(.\*\)

Struct "struct"" "{Variable}

Class "class"" "{Variable}

Template "template <typename "{Variable}>

FriendFunction "friend "{DataType}" "{Variable}\(.\*\)

FriendClass "friend class "{Variable}

AccessSpecifier public|private|protected

InheritanceUtility ", "{AccessSpecifier}" "{Variable}

Inheritance "class"" "{Variable}" : "{AccessSpecifier}" "{Variable}{InheritanceUtility}\*

PointerDeclaration [\*]{Variable}

PointerAddress [&]{Variable}

PointerArrow "->"

Pointer {Variable}{PointerArrow}{Variable}

**%%**

{Delimitor} {;}

{Comment} {printResultsToFile("Comment\t\t\t\t",yytext);}

{Header} {printResultsToFile("Header\t\t\t\t",yytext);}

{Namespace} {printResultsToFile("Namespace\t\t\t",yytext);}

{ReservedWord} {printResultsToFile("Reserved Word\t\t",yytext);}

{DataType} {printResultsToFile("Datatype\t\t\t",yytext);}

{Bracket} {printResultsToFile("Brackets\t\t\t",yytext);}

{Bool} {printResultsToFile("Boolean\t\t\t\t",yytext);}

{Float} {printResultsToFile("Float\t\t\t",yytext);}

{Int} {printResultsToFile("Integer\t\t\t\t",yytext);}

{String} {printResultsToFile("String\t\t\t\t",yytext);}

{Char} {printResultsToFile("Character\t\t\t",yytext);}

{Operator} {printResultsToFile("Operator\t\t\t",yytext);}

{Array} {printResultsToFile("Array\t\t\t\t",yytext);}

{PointerDeclaration} {printResultsToFile("Declare Pointer\t\t",yytext);}

{PointerAddress} {printResultsToFile("Pointer Address\t\t",yytext);}

{Pointer} {printResultsToFile("Pointer\t\t\t\t",yytext);}

{DeclareFunction} {printResultsToFile("Declare Function\t",yytext);}

{FunctionCall} {printResultsToFile("Function Call\t\t",yytext);}

{Struct} {printResultsToFile("Structure\t\t\t",yytext);}

{Class} {printResultsToFile("Class\t\t\t\t",yytext);}

{Template} {printResultsToFile("Template\t\t\t",yytext);}

{FriendClass} {printResultsToFile("Friend Class\t\t",yytext);}

{FriendFunction} {printResultsToFile("Friend Function\t\t",yytext);}

{Inheritance} {printResultsToFile("Inheritance\t\t\t",yytext);}

{Condition} {printResultsToFile("Condition\t\t\t",yytext);}

{Variable} {printResultsToFile("Variable\t\t\t",yytext);}

\n {lineNum++;}

**%%**

int yywrap(){}

int main()

{

    FILE \*fptr;

    fptr = fopen("results.txt", "w");

    fclose(fptr);

    FILE \*fout;

    fout = fopen("Sample1.cpp","r");

    lineNum=0;

    printResultsToFile("FILE 1\t\t\t\t","Sample1.cpp\n");

    lineNum++;

    yyin = fout;

    yylex();

    fout = fopen("Sample2.cpp","r");

    lineNum=0;

    printResultsToFile("\n\n\tFILE 2\t\t\t\t","Sample2.cpp\n");

    lineNum++;

    yyin=fout;

    yylex();

    fout = fopen("Sample3.cpp","r");

    lineNum=0;

    printResultsToFile("\n\n\tFILE 3\t\t\t\t","Sample3.cpp\n");

    lineNum++;

    yyin=fout;

    yylex();

    return 0;

}

**Samples and Results**

**Sample 1:**

// This code is used to test the lexical analyzer

#include <iostream>

using namespace std;

class Calculator

{

public:

    void Add(int a, int b) { cout << a + b; }

    void Subtract(int a, int b) { cout << a - b; }

    void Multiply(int a, int b) { cout << a \* b; }

    void Divide(int a, int b)

    {

        if (b != 0)

            cout << a / b;

        else

            cout << "Denominator cannot be 0!";

    }

};

int main()

{

    Calculator Obj;

    Obj.Add(3, 4);

    return 0;

}

**Result 1:**

FILE 1 Line Number: 0 Sample1.cpp

Comment Line Number: 1 // This code is used to test the lexical analyzer

Header Line Number: 3 #include <iostream>

Namespace Line Number: 4 using namespace std

Class Line Number: 6 class Calculator

Brackets Line Number: 7 {

Reserved Word Line Number: 8 public

Declare Function Line Number: 10 void Add(int a, int b)

Brackets Line Number: 10 {

Reserved Word Line Number: 10 cout

Operator Line Number: 10 <<

Character Line Number: 10 a

Operator Line Number: 10 +

Character Line Number: 10 b

Brackets Line Number: 10 }

Declare Function Line Number: 12 void Subtract(int a, int b)

Brackets Line Number: 12 {

Reserved Word Line Number: 12 cout

Operator Line Number: 12 <<

Character Line Number: 12 a

Character Line Number: 12 -

Character Line Number: 12 b

Brackets Line Number: 12 }

Declare Function Line Number: 14 void Multiply(int a, int b)

Brackets Line Number: 14 {

Reserved Word Line Number: 14 cout

Operator Line Number: 14 <<

Character Line Number: 14 a

Operator Line Number: 14 \*

Character Line Number: 14 b

Brackets Line Number: 14 }

Declare Function Line Number: 16 void Divide(int a, int b)

Brackets Line Number: 17 {

Reserved Word Line Number: 18 if

Condition Line Number: 18 (b != 0)

Reserved Word Line Number: 19 cout

Operator Line Number: 19 <<

Character Line Number: 19 a

Character Line Number: 19 /

Character Line Number: 19 b

Reserved Word Line Number: 21 else

Reserved Word Line Number: 22 cout

Operator Line Number: 22 <<

String Line Number: 22 "Denominator cannot be 0!"

Brackets Line Number: 23 }

Brackets Line Number: 24 }

Declare Function Line Number: 26 int main()

Brackets Line Number: 27 {

Variable Line Number: 28 Calculator

Variable Line Number: 28 Obj

Function Call Line Number: 29 Obj.Add(3, 4)

Reserved Word Line Number: 30 return

Boolean Line Number: 30 0

Brackets Line Number: 31 }

**Sample 2:**

// This code is used to test lexical analyzer

#include <iostream>

using namespace std;

struct Node

{

    int Data;

    Node\* Next;

};

void Insert(Node\* Head, int NewData)

{

    Node\* NewNode = new Node;

    NewNode->Data = NewData;

    NewNode->Next = NULL;

    Node\* Temp = Head;

    while (Temp != NULL)

        Temp = Temp->Next;

    Temp = NewNode;

}

int main()

{

    Node\* Head;

    Insert(Head, 3);

    Insert(Head, 7);

    return 0;

}

**Result 2:**

FILE 2 Line Number: 0 Sample2.cpp

Comment Line Number: 1 // This code is used to test lexical analyzer

Header Line Number: 3 #include <iostream>

Namespace Line Number: 4 using namespace std

Structure Line Number: 6 struct Node

Brackets Line Number: 7 {

Datatype Line Number: 8 int

Variable Line Number: 8 Data

Variable Line Number: 9 Node

Operator Line Number: 9 \*

Variable Line Number: 9 Next

Brackets Line Number: 10 }

Declare Function Line Number: 12 void Insert(Node\* Head, int NewData)

Brackets Line Number: 13 {

Variable Line Number: 14 Node

Operator Line Number: 14 \*

Variable Line Number: 14 NewNode

Operator Line Number: 14 =

Reserved Word Line Number: 14 new

Variable Line Number: 14 Node

Pointer Line Number: 15 NewNode->Data

Operator Line Number: 15 =

Variable Line Number: 15 NewData

Pointer Line Number: 16 NewNode->Next

Operator Line Number: 16 =

Variable Line Number: 16 NULL

Variable Line Number: 18 Node

Operator Line Number: 18 \*

Variable Line Number: 18 Temp

Operator Line Number: 18 =

Variable Line Number: 18 Head

Reserved Word Line Number: 20 while

Condition Line Number: 20 (Temp != NULL)

Variable Line Number: 21 Temp

Operator Line Number: 21 =

Pointer Line Number: 21 Temp->Next

Variable Line Number: 23 Temp

Operator Line Number: 23 =

Variable Line Number: 23 NewNode

Brackets Line Number: 24 }

Declare Function Line Number: 26 int main()

Brackets Line Number: 27 {

Variable Line Number: 28 Node

Operator Line Number: 28 \*

Variable Line Number: 28 Head

Function Call Line Number: 29 Insert(Head, 3)

Function Call Line Number: 30 Insert(Head, 7)

Reserved Word Line Number: 31 return

Boolean Line Number: 31 0

Brackets Line Number: 32 }

**Sample 3:**

/\* This code is used to test the lexical analyzer \*/

template <typename t>;

class Derived : public Base

{

};

friend void Func(int x, int y);

friend class Derived;

**Result 3:**

FILE 3 Line Number: 0 Sample3.cpp

Comment Line Number: 1 /\* This code is used to test the lexical analyzer \*/

Template Line Number: 3 template <typename t>

Inheritance Line Number: 5 class Derived : public Base

Brackets Line Number: 6 {

Brackets Line Number: 8 }

Friend Function Line Number: 10 friend void Func(int x, int y)

Friend Class Line Number: 12 friend class Derived