Khulna University of Engineering & Technology, Khulna

Assignment Report

Department of **Computer Science and Engineering**

Course No.: **CSE 3212**

Course Title: **Compiler Design Laboratory**

Report No.: 02

Report on: **report on lexical analysis of my language**

Submission Date: 03-10-2023

Submitted To	Submitted By
Nazia Jahan Khan Chowdhury Assistant Professor Computer Science and Engineering KUET, Khulna **Dipannita Biswas** Lecturer Computer Science and Engineering KUET, Khulna	**Doniel Tripura** Roll: 1907121 Year: 3rd Term: 2nd Department of Computer Science and Engineering KUET, Khulna

Objectives:

1. To design and implement a lexical analyzer using Flex.
2. To create a new programming language based on c or c++.
3. To create regular expressions that can recognize our programming language.
4. To verify our programming language.

Introduction:

The process of creating a new programming language is a fascinating and challenging endeavor that requires careful planning and implementation. One of the fundamental components in designing a programming language is the lexical analysis, which serves as the first step in the compilation process. Lexical analysis, often referred to as scanning or tokenization, is the process of breaking down the source code into a stream of meaningful units known as tokens. In this context, Flex, short for "Fast Lexical Analyzer Generator," plays a pivotal role. Flex is a powerful tool used to generate lexical analyzers, which are responsible for recognizing and categorizing the individual elements within a programming language, such as keywords, identifiers, operators, and literals. By using Flex, programmers can efficiently define the rules that govern how the source code is tokenized, making it an indispensable tool for language designers and compiler developers.

Tool Required:

1. VS code or any Editor
2. Flex

Command:

->flex 1907121.l (It tokenize our source file and create a lex.yy.c file)

-> gcc lex.yy.c -o output (for c program, It compile the program)

-> g++ lex.yy.c -o output (for cpp program, It compile the program)

-> output (To run the program in cmd)

-> ./output (To run the program in VS terminal)

Header:	TW sample.h
Data Type Keywords:	
Int	REM
Float	SUK
String	KOK
Char	KOKTHAI
Bool	ONGMANO
Void	MUNGSAYA
Control Flow	Keywords:
If	IMO
Else if	IMOYA
Else	IMOYAKHAI
Switch	LAMA
Case	CASE
Default	ONGTHAI
For	BAGWI
While	BAIFU
Do	KHAI
Break	SABAI
Continue	KHAITONG
Return	FIROK
Goto	UROTHANG
Scanf	LA
Printf	FUNU

Storage Class Keywords:

AUTO	AUTO
STATIC	STATIC
REGISTER	REGISTER
EXTERN	EXTERN

OTHER Keywords:

CLASS	CLASS
STRUCT	STRUCT
ENUM	ENUM
NAMESPACE	NAMESPACE
TEMPLATE	TEMPLATE
NEW	NEW
DELETE	DELETE
THIS	THIS
CONST	CONST
SIZEOF	SIZEOF
OPERATOR	OPERATOR
FRIEND	FRIEND
VIRTUAL	VIRTUAL
EXPLICIT	EXPLICIT
TRY	TRY
CATCH	CATCH
THROW	THROW
TYPEID	TYPEID

Arithmetic Operators:

[+]	+
[-]	-
[*]	*
[/]	/
[%]	%

Comparison Operators:

[=][=]	EQUALS
[!][=]	NOT EQUALS
[<]	LESS_THAN
[>]	GREATER_THAN
[<][=]	LESS_THAN_OR_EQUAL
[>][=]	GREATER_THAN_OR_EQUAL

Logical Operators:

[&][&]	LOGICAL AND
[|][|]	LOGICAL OR
[!]	LOGICAL NOT

Assignment Operators:

[=]	ASSIGNMENT
[+][=]	ADDITION ASSIGNMENT
[-][=]	SUBTRACTION ASSIGNMENT
[*][=]	MULTIPLICATION ASSIGNMENT
[/][=]	DIVISION ASSIGNMENT
[%][=]	MODULUS ASSIGNMENT

Increment and Decrement Operators:

[+][+]	INCREMENT
[-][-]	DECREMENT

Bitwise Operators:

[&]	BITWISE AND
[|]	BITWISE OR
"^"	BITWISE XOR
[~]	BITWISE NOT
[<][<]	LEFTSHIFT
[>][>]	RIGHTSHIFT

Comment:

// Single Line Comment

/*

This is

Multiple Line Comment

*/

Discussion:

The process of lexical analysis using Flex is a foundational step in the development of a new programming language. In this discussion, we went deeper into some of the key concepts and considerations related to lexical analysis and the use of Flex in language design. Lexical analysis is the initial phase of compilation and serves as the bridge between the human-readable source code and the subsequent stages of compilation. It breaks down the source code into smaller units called tokens, making it easier for the compiler to understand and process.

Conclusion:

By the end of this exploration, We have a foundational understanding of how Flex facilitates the creation of a lexical analyzer for your programming language, setting the stage for subsequent phases in language design, such as syntax analysis and

code generation. Lexical analysis is the crucial first step in transforming human-readable code into a form that a computer can understand and execute, making it an exciting and pivotal aspect of programming language development.