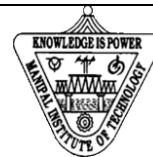




MANIPAL INSTITUTE OF TECHNOLOGY
(Constituent Institute of MANIPAL University)
MANIPAL-576104



CSE 312 LANGUAGE PROCESSORS LAB MANUAL

VI Sem , BE (CS&E)

2015

Prepared By

1. **Dr. Ashalatha Nayak**
2. **Ms. Deepthi. S.**
3. **Mr. Shyam Karanth**
4. **Ms. Ancilla Pinto**

Approved By

(H.O.D)

DEPT OF COMPUTER SCIENCE & ENGG.

M. I. T., MANIPAL

Week-wise Schedule

Week 1	:	Preliminary Scanning Applications
Week 2	:	Identification of Tokens in a given Program
Week 3, 4, 5	:	Design of Lexical Analyzer
Week 6, 7, 8, 9, 10	:	Design of Parser
Week 11	:	Design of Code Generator
Week 12	:	Usage of Lex
Week 13 & 14	:	Final Test

PROCEDURE OF EVALUATION

Student will be evaluated based on following criteria:

Scanning	:	5 Marks
Tokenizing	:	5 Marks
Lexical analyzer	:	15 Marks
Parser	:	20 Marks
Code generation	:	5 Marks
LEX	:	10 Marks
Test	:	40 Marks (15 write up+ 25 Execution.)
Total		<u>100 Marks</u>

WEEK 1: Preliminary Scanning applications

- 1) Programs on file operations.
- 2) Write a program, which will read a program written in C, recognize all of the keywords in that program and print them in upper case letters.

WEEK 2: Identification of Tokens in a given Program

- 1) Write a program which will take as input a C program consisting of single and multi-line comments and multiple blank spaces and produces as output the C program without comments and single blank space.

```
// This is a single line comment
```

```
/* *****This is a  
*****      Multiline Comment  
*****/
```

- 2) Write a program, which will read a program written in C, recognize all of the lexemes (int, float, char, for, while etc., ids, +, -, /, *, (,), numbers, <, >, <=, >=, !=, ==), tokens (keywords, identifiers, operators, special symbols, relational operators) and display them in a separate file.
- 3) Repeat Qn. (2) by storing tokens in each of the following data structures:
 - a. Arrays
 - b. Singly/doubly Linked lists
 - c. Hash tables

WEEK 3 – 10: Design of Mini Compiler for C Language for the given subset

Data Types	:	int, char
Arrays	:	1-dimensional
Expressions	:	Arithmetic and Relational
Looping statements	:	for, while
Decision statements	:	if, if – else

Note: The following grammar for C language is to be adopted with necessary corrections

Program - main () { declarations statement-list }

declarations → data-type identifier-list; declarations | ∈

data-type → int | char

identifier-list → id | id, identifier-list | id[number] , identifier-list | id[number]

statement_list → statement ; statement_list | ∈

statement → assign-stat | decision_stat | looping-stat

assign_stat \rightarrow id = expn

expn \rightarrow simple-expn eprime

epime \rightarrow relop simple-expn | \in

simple-exp \rightarrow term seprime

seprime \rightarrow addop term seprime | \in

term \rightarrow factor tprime

tprime \rightarrow mulop factor tprime | \in

factor \rightarrow id | num

decision-stat \rightarrow if (expn) stat dprime

dprime \rightarrow else stat | \in

looping-stat \rightarrow while (expn) stat | for (assign_stat ; expn ; assign_stat) stat

relop \rightarrow = | != | <= | >= | > | <

addop \rightarrow + | -

mulop \rightarrow * | / | %

WEEK 3, 4 & 5: Design of Lexical analyzer

To construct an Lexical Analyzer.

- Identifying different classes of tokens like: keywords, identifiers and special symbols.
- Selecting a suitable data structure for symbol table (the alternates are linked list, hashing, array of structures, binary search tree)
- Having selected a data structure, identifying the appropriate fields.

To test the Lexical Analyzer:

Input: C Program

Output: Tokens and their Class

Interface:

The Lexical Analyzer should tokenize a given source program and return the next token and it's class whenever the parser requests.

WEEK 6-10: Design of a Recursive Descent Parser

To code and test parser:

- Remove left recursion from each of the productions so that the underlying grammar can be parsed with a parser.
- The parser obtains a string of tokens from the lexical analyzer and verifies that the string can be generated by the grammar for the C language.
- The parser should report syntax errors if any (for eg.: Misspelling an identifier or keyword, Undeclared or Multiply declared identifier, Arithmetic or Relational Expressions with unbalanced parentheses and Expression syntax error etc.) with appropriate line-no.

Simple grammars

1. $E \rightarrow \text{num } T$
 $T \rightarrow * \text{num } T \mid \epsilon$
2. $S \rightarrow a \mid \rightarrow \mid (T)$
 $T \rightarrow T, S \mid S$
3. $E \rightarrow E + T \mid T$
 $T \rightarrow T * F \mid F$
 $F \rightarrow (E) \mid \text{id}$
4. $S \rightarrow aAcBe$
 $A \rightarrow Ab \mid b$
 $B \rightarrow d$
5. $\text{lexp} \rightarrow \text{aterm} \mid \text{list}$
 $\text{aterm} \rightarrow \text{number} \mid \text{identifier}$
 $\text{list} \rightarrow (\text{lexp_seq})$
 $\text{lexp_seq} \rightarrow \text{lexp_seq lexp} \mid \text{lexp}$
6. Build a RD parser for the above C grammar.

WEEK 11: Design of Code generator

WEEK 12: Usage of LEX

1. Write a LEX program to count the number of lines, words, blank spaces and characters in a given input.
2. Write a LEX program to find the number of vowels, consonants in the given input.
3. Write a LEX program to check if a given number is positive or negative.
4. Write a LEX program to check if a given statement is simple or compound.
5. Write a LEX program to count the type of numbers.
6. Write a LEX program to check the validity of a given arithmetic statement.
7. Write a lex program to count the number of “printf” and “scanf” statements in a valid c program and replace them with write and read statements respectively.

8. Write a lex program to count the number of characters, words, blanks and lines in a given string.

WEEK 13 and 14: Test

REFERENCES

1. Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman, "Compilers Principles, Techniques and Tools", Pearson Education, 2nd edition. 2010
2. D M Dhamdhere, "Systems Programming and Operating Systems", Tata McGraw Hill, 2nd Revised Edition, 2001.
3. Kenneth C. Loudon, "Compiler Construction - Principles and Practice", Thomson, India Edition, 2007.
