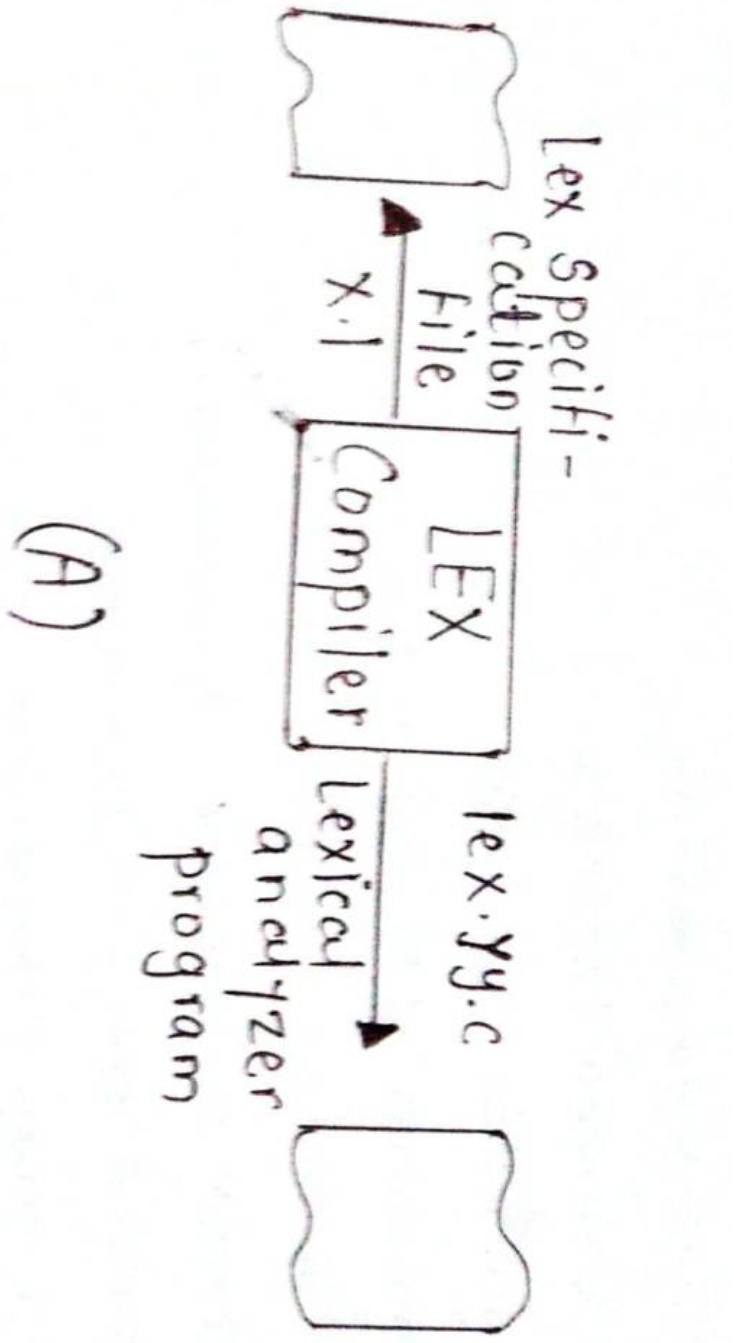
Explain the automatic generation of lexical analyzer.

1. Automatic generation of lexical analyzer is done using LEX programming language.
2. The LEX specification file can be denoted using the extension .l (often pronounced as dot L).
3. For example, let us consider specification file as x.l.
4. This x.l file is then given to LEX compiler to produce lex.yy.c as shown in image (A)This lex.yy.c is a C program which is actually a lexical analyzer program.

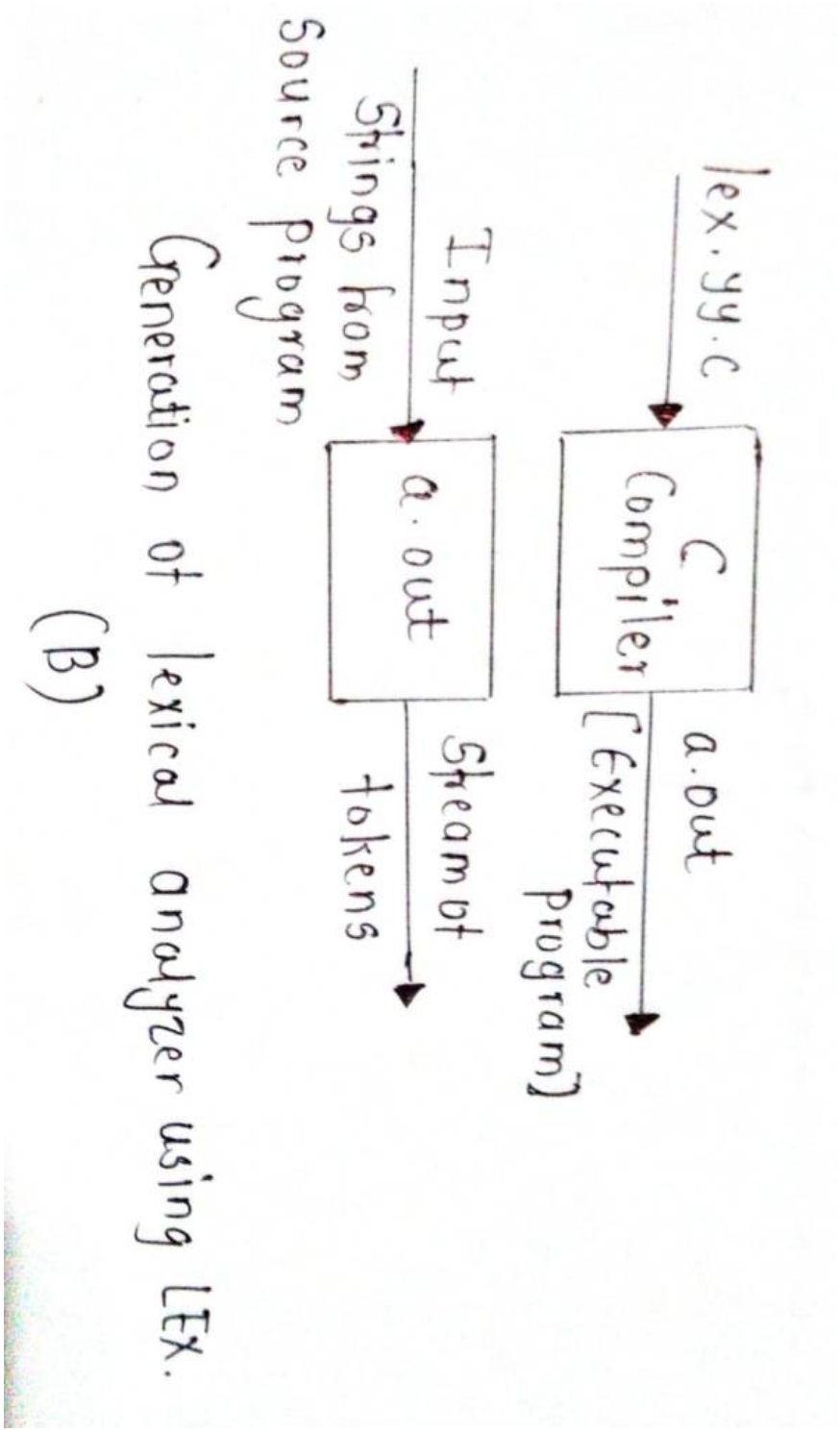
Explain the automatic generation of lexical analyzer.



Explain the automatic generation of lexical analyzer.

1. The LEX specification file stores the regular expressions for the token and the lex.yy.c file consists of the tabular representation of the transition diagrams constructed for the regular expression.
2. In specification file, LEX actions are associated with every regular expression.
3. These actions are simply the pieces of C code that are directly carried over to the lex.yy.c.
4. Finally, the C compiler compiles this generated lex.yy.c and produces an object program a. out as shown in image.
5. When some input stream is given to a. out then sequence of tokens gets generated. The described scenario is shown in image(B).

Explain the automatic generation of lexical analyzer.



(B) Generation of lexical analyzer using LEX.

Explain the automatic generation of lexical analyzer.

Related Posts:

1. What are the types of passes in compiler ?
2. Discuss the role of compiler writing tools. Describe various compiler writing tools.
3. What do you mean by regular expression ? Write the formal recursive definition of a regular expression.
4. How does finite automata useful for lexical analysis ?
5. Explain the implementation of lexical analyzer.
6. Write short notes on lexical analyzer generator.
7. Explain the term token, lexeme and pattern.
8. What are the various LEX actions that are used in LEX programming ?
9. Describe grammar.
10. Explain formal grammar and its application to syntax analyzer.
11. Define parse tree. What are the conditions for constructing a parse tree from a CFG ?
12. Describe the capabilities of CFG.
13. What is parser ? Write the role of parser. What are the most popular parsing techniques ? OR Explain about basic parsing techniques. What is top-down parsing ? Explain in detail.
14. What are the common conflicts that can be encountered in shift-reduce parser ?
15. Differentiate between top-down and bottom-up parser.Under which conditions predictive parsing can be constructed for a grammar ?
16. Differentiate between recursive descent parsing and predictive parsing.
17. What is the difference between S-attributed and L-attributed definitions ?
18. What is intermediate code generation and discuss benefits of intermediate code ?
19. Define parse tree. Why parse tree construction is only possible for CFG ?
20. Discuss symbol table with its capabilities ?
21. What are the symbol table requirements ? What are the demerits in the uniform structure of symbol table ?