Mindluster Platform

About the course Language processing systems fundamentals

Language processing systems course in hindi, in this course we will learn about the fundamental components and techniques used in language processing systems. Starting with lexical analysis, we will explore how raw text is converted into tokens, the basic units of meaning. We will then delve into syntax analysis, where we study parsing techniques to understand the grammatical structure of sentences. Semantic analysis follows, focusing on how meaning is derived from syntactic structures. The course will also cover intermediate code generation, which serves as a bridge between high-level language and machine code. We'll discuss various code optimization techniques to enhance the efficiency of the generated code. Finally, we will explore code generation, converting intermediate representations into executable code. Additionally, the course will touch upon natural language processing (NLP) fundamentals and advanced topics, providing a comprehensive understanding of language processing systems. By the end of this course, students will have a solid foundation in designing and implementing language processing systems, ready to tackle real-world applications in compiler design, NLP, and more.

Computer Science Category's Courses

Course Lesson(61)

Lesson 1 : **Language Processing System**

Lesson 2 : **Phases of a compiler in Compiler Design Examples CD**

Lesson 3 : **Phases of a compiler example1 in Compiler Design Examples CD**

Lesson 4 : **Phases of a compiler example2 in Compiler Design Examples CD**

Lesson 5 :
Grouping of phases into passes front end back end of the compiler Analysis part Synthesis part

Lesson 6 : **Role of lexical Analyzer in Compiler Design Examples CD**

Lesson 7 : **Input Buffering**

Lesson 8 : **Specification of Tokens in Compiler Design**

Lesson 9 : **Recognition of Tokens in Compiler Design**

Lesson 10 : **LEX tool in compiler design**

Lesson 11 : **Compiler Construction Tools**

Lesson 12 : **Bootstrapping in Compiler Design**

Lesson 13 : **Role of a parser in compiler design**

Lesson 14 : **Derivation Tree Left Right Derivation Trees**

Lesson 15 : **Ambiguous Grammar**

Lesson 16 :
Elimination of left recursion Removal of Ambiguity in Grammar CFG TOC Compiler Design

Lesson 17 :
Elimination of left factoring Removal of Ambiguity in Grammar CFG TOC Compiler Design

Lesson 18 : **First and Follow in compiler design**

Lesson 19 : **LL 1 parser predictive parser non recursive predictive parsing**

Lesson 20 : **Recursive Descent parser**

Lesson 21 : **Backtracking in compiler design**

Lesson 22 : **What is Bottom Up Parsing**

Lesson 23 : **Handle and Handle Pruning**

Lesson 24 : **Shift reduce parser**

Lesson 25 : **Operator Precedence Parser in Compiler Design Example 1**

Lesson 26 : **Operator Precedence Parser in Compiler Design Example 2**

Lesson 27 : **SLR Parser in compiler design with solved example1**

Lesson 28 : **SLR Parser in compiler design with solved example2**

Lesson 29 : **SLR Parser in compiler design with solved example3**

Lesson 30 : **CLR Parser in compiler design with solved example1**

Lesson 31 : **CLR Parser in compiler design with solved example2**

Lesson 32 : **LALR parser in compiler design with solved example2**

Lesson 33 : **LALR Parser In Compiler Design with solved example1**

Lesson 34 : **YACC tool in compiler design YACC program to implement desk calculator**

Lesson 35 : **Syntax directed definition SDD in compier design Types of attributes Types of SDD**

Lesson 36 : **Syntax directed definition of simple desk calculator Annotated parse tree for 3 5 4n**

Lesson 37 : **Annotated parse tree for int a b c real a b c**

Lesson 38 : **Annotated parse tree for 3 5 Inherited attributes**

Lesson 39 : **Dependency Graph in Compiler Design**

Lesson 40 : **Syntax Directed Translation Translation scheme to convert infix to postfix expression**

Lesson 41 : **construction of syntax tree in compiler design**

Lesson 42 : **Syntax directed translation scheme for infix to postfix**

Lesson 43 : **Forms of intermediate code Forms of Three Address instructions**

Lesson 44 : **Construct Quadruples triples indirect triples for the statement a b c d a b c**

Lesson 45 :
Translation of Boolean Expressions Generating Three Address Code for Boolean Expressions

Lesson 46 :
Intermediate Code for flow of control statements flow of control statements in compiler design

Lesson 47 : **Backpatching in compiler design**

Lesson 48 : **intermediate code for switch statement translation of switch statement**

Lesson 49 : **intermediate code for procedures**

Lesson 50 :
Run time environment in compiler design Run time Storage Management in compiler design

Lesson 51 : **Activation Record in Compiler Design**

Lesson 52 :
storage allocation strategies in compiler design Storage Organization in compiler design

Lesson 53 : **code optimization techniques in compiler design principal sources of optimization**

Lesson 54 : **Peephole optimization in compiler design**

Lesson 55 : **Loop Optimization in compiler design**

Lesson 56 : **Optimization of Basic Blocks in compiler Design**

Lesson 57 : **Issues in the design of a code generator**

Lesson 58 : **basic blocks and flow graphs in compiler design**

Lesson 59 : **DAG representation of a basic block construction of dag from basic blocks**

Lesson 60 : **Object code forms A simple target machine model The target machine**

Lesson 61 : **A simple code generator algorithm in compiler design**

Related courses

Linear Algebra for Computer Scientists

Insertion Sort

Bubble Sort

GCSE Computer Science

Random Access Memory

Binary Trees

for Business Contact
business@mindluster.com