ISSUES IN LANGUAGE TRANSLATION

A programmer code a program using subprograms, statements, conditions, declaration etc. But a translator looks programming languages as a collection of symbols made up of thousands of characters. During translation a program is analyzed character by character.

During language translation issues occurs in the following way:

1. Lexical analysis
2. Syntax analysis
3. Semantic analysis
4. Symbol table
5. Error handler

1) Lexical analysis: In this phase isuue is with grouping of charcters in to meaningfull words known as lexems. Here meaningfull words means identifiers, numbers, keywords, comments etc. The basic units means characters on which this analysis is performed known as tokens. Lexical analyzer must analyze the type of each lexems and with a type tag that lexems is sends for higher analysis i.e, syntax analysis. This phase requires much translation time as compared to other phases.

2) Syntax analysis: Here syntax of lexems are checked using parse/syntax tree. Structures defined by the programmer are validated here. If getting any error during language translation in this phase, compiler throws a syntax error.

3) Semantic analysis: Semantic analysis judges whether the syntax structure constructed in the source program derives any meaning or not.

4) Symbol table: Symbol Table is an important data structure created and maintained by the compiler. All the phases uses symbol table as follows:

1. Lexical Analysis: Creates new table entries in the symbol table, like tokens.
2. Syntax Analysis: Adds information regarding attribute type, scope etc in the table.
3. Semantic Analysis: Uses available information in the table to check for semantics i.e. to verify that expressions and assignments are semantically correct(type checking) and update it accordingly.
4. Intermediate Code generation: Refers symbol table for knowing how much and what type of run-time is allocated and table helps in adding temporary variable information.
5. Code Optimization: Uses information present in symbol table removing unnecessary codes from Intermediate codes.
6. Target Code generation: Generates code by using address information of identifier present in the table.

5) Error handler: The tasks of the Error Handler is to detect each error, report it to the user, and then recover and implement them to handle error.

During this process processing time of program should not be slow. An Error is the blank entries in the symbol table.

References:

1. Sebesta,"Concept of programming Language", Pearson Edu
2. Louden, "Programming Languages: Principles & Practices" , Cengage Learning
3. Tucker, "Programming Languages: Principles and paradigms ", Tata McGraw –Hill.
4. E Horowitz, "Programming Languages", 2nd Edition, Addison Wesley

Related Posts:

1. Sequence Control & Expression | PPL
2. PPL:Named Constants
3. Parse Tree | PPL | Prof. Jayesh Umre
4. Basic elements of Prolog
5. Loops | PPL | Prof. Jayesh Umre
6. Subprograms Parameter passing methods | PPL | Prof. Jayesh Umre
7. Programming Paradigms | PPL | Prof. Jayesh Umre
8. Subprograms Introduction | PPL | Prof. Jayesh Umre
9. Phases of Compiler | PPL | Prof. Jayesh Umre
10. Parse Tree | PPL
11. Influences on Language design | PPL | Prof. Jayesh Umre
12. Fundamentals of Subprograms | PPL | Prof. Jayesh Umre
13. Programming Paradigm
14. Influences on Language Design
15. Language Evaluation Criteria
16. OOP in C++ | PPL
17. OOP in C# | PPL
18. OOP in Java | PPL
19. PPL: Abstraction & Encapsulation
20. PPL: Semaphores
21. PPL: Introduction to 4GL
22. PPL: Variable Initialization
23. PPL: Conditional Statements
24. PPL: Array
25. PPL: Strong Typing
26. PPL: Coroutines

27. PPL: Exception Handler in C++

28. PPL: OOP in PHP

29. PPL: Character Data Type

30. PPL: Exceptions

31. PPL: Heap based storage management

32. PPL: Primitive Data Type

33. PPL: Data types

34. Programming Environments | PPL

35. Virtual Machine | PPL

36. PPL: Local referencing environments

37. Generic Subprograms

38. Local referencing environments | PPL | Prof. Jayesh Umre

39. Generic Subprograms | PPL | Prof. Jayesh Umre

40. PPL: Java Threads

41. PPL: Loops

42. PPL: Exception Handling

43. PPL: C# Threads

44. Pointer & Reference Type | PPL

45. Scope and lifetime of variable

46. Design issues for functions

47. Parameter passing methods

48. Fundamentals of sub-programs

49. Subprograms

50. Design issues of subprogram

51. Garbage Collection

52. PPL Previous years solved papers

53. Type Checking | PPL | Prof. Jayesh Umre