TYPE CHECKING

Introduction:

The compiler must perform static checking (checking done at compiler time).This ensures that certain types of programming errors will be detected and reported.

A compiler must check that the source program follows both the syntactic and semantic conversions of the source language. This checking is called static checking example of static checks include.

Some example of static checks is:

Type checks: A compiler should report an error if an operator is applied to an incompatible operand.

Flow-of-control checks: Statements that cause flow of control to leave a construct must have some place to which to transfer flow of control. For example, branching to non-existent labels.

Uniqueness checks: Objects should be defined only once. This is true in many languages.

Name-related checks: Sometimes, the same name must appear two or more times. For example, in Ada the name of a block must appear both at the beginning of the block and at the end.

A compiler should report an error if an operator is applied to an incompatible operand. This checking is called Type checking.

Type information gathered by a type checker may be needed when code is generated. For example, arithmetic operators may be different at the machine level for different types of operands (real and integer).

TYPE SYSTEM:

The type analysis and type checking is an important activity done in the semantic analysis phase. The need for type checking is:

- To detect the errors arising in the expression due to incompatible operand.
- To generate intermediate code for expressions and statements. Typically language supports two types of data types- basic and constructed.

The basic data type are- integer, character, and real, Boolean, enumerated data type. And Arrays, record (structure),set and pointer are the constructed types. The constructed data types are build using basic data types.

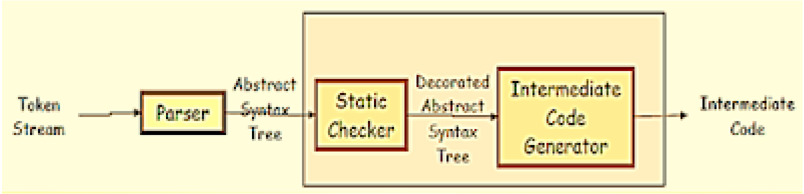

Fig 1:- Position of Type checking

Type Expression: Type of a language construct. It is either a basic type or is formed by applying an operator called a type constructor to other type expressions.

A type system is a collection of rules for assigning type expression to the various parts of a program. A type checker implements a type system. Different type system may be used by different compilers or processors of the system Language.

Checking done by a compiler is said to be static checking of types, while checking done when the target program runs is terminal dynamic checking of types.

A source type system eliminates the need for dynamic checking for type errors because it allows us to determine statically that these errors cannot occur when the target program runs.
Type checking should have a property of error recovery.

Related Posts:

1. Introduction to Compiler
2. Analysis and synthesis model of compilation
3. Bootstrapping and Porting
4. Lexical Analyzer: Input Buffering
5. Storage Allocation Strategies
6. Type Checking
7. Specification & Recognition of Tokens
8. Front end and back end of the compiler
9. LEX
10. Analysis synthesis model of compilation
11. Data structure in CD
12. Register allocation and assignment
13. Loops in flow graphs
14. Dead code elimination
15. Syntax analysis CFGs
16. L-attribute definition
17. Operator precedence parsing
18. Analysis of syntax directed definition
19. Recursive descent parser
20. Function and operator overloading
21. Storage allocation strategies
22. Equivalence of expression in type checking

23. Storage organization
24. Parameter passing
25. Run time environment
26. Code generation issue in design of code generator
27. Boolean expression
28. Declaration and assignment in intermediate code generation
29. Code optimization
30. Sources of optimization of basic blocks
31. Loop optimization
32. Global data flow analysis
33. Data flow analysis of structure flow graph (SFG)