Types and Declarations

The applications of types can be grouped under checking and translation:

Type checking uses logical rules to reason about the behavior of a program at run time. Specifically, it ensures that the types of the operands match the type expected by an operator. For example, the && operator in Java expects its two operand to be booleans; the result is also of type boolean.

Translation Applications. From the type of a name, a compiler can determine the storage that will be needed for that name at run time. Type information is also needed to calculate the address denoted by an array reference, to insert explicit type conversions, and to choose the right version of an arithmetic operator, among others.

Type Expressions

Types hace structure, which we shall represent using type expressions : a type expressions is either a basic type or is formed by applying an operator called a type constructor to a type expression.The sets of basic types and constructors depend on the language to be checked.

Definitions:
-A basic type is a type expression. Typical basic types for a language include boolean, char ,integer, float and void; the latter denotes “the absence of a value”
-A type name is a type expression
-A type expression can be formed by applying the array type constructor to a number and a type expression
-A record is a data structure with name fields(symbol table or stack). A type expression can be formed by applying the record type constructor to the field names and their types.

Type Equivalence

Two types are structurally equivalent if and only if one of the following conditions is true:

-They are the same basic type
-They are formed by applying the same constructor to structurally equivalent types
-One is a type name that denotes the other