Overloading

In C the plus sign (+) is used to name several different functions, including signed and unsigned integer and floating-point addition.

Within the symbol table of a compiler, overloading must be handled (*resolved*) by arranging for the `lookup` routine to return a *list* of possible meanings for the requested name. The semantic analyzer must then choose from among the elements of the list based on context. When the context is not sufficient to decided, then the semantic analyzer must announce an error.

In C, one cannot overload enumeration constants at all; every constant visible in a given scope must be distinct. In C++, Java and C# A given name may refer to an arbitrary number of subroutines in the same scope, so long as the subroutines differ in the number or types of their arguments.

Redefining Built-in Operators

Many languages also allow the built-in operators $(+, -, *,$ etc.) to be overloaded with user-defined fuctions. Ada, C+, and C# do this by defining alternatice *prefix* forms of each operator, and defining the usual *infix* forms to be abbreviations for the prfix forms.

Related Concepts

When considering function and subroutine calls, it is important to distinguish overloading from the related concepts of *coercion* and *polymorphism*. All three can be used, in certain circumstances, to pass arguments of multiple types to (or return values of multiple types from) what appears to be a single named routine. The syntactic similarity, hides significant differences in semantic and pragmatics.

Corcion, is the process by which a compiler automatically converts a value of one type into a value of another type when that second type is required by the surrounding context. Polymorphism, allows a single subroutine to accept arguments of multiple types.