

Functions break large computing tasks into smaller ones, and enable people to build on what others have done instead of starting over from scratch. Appropriate functions hide details of operation from parts of the program that don't need to know about them, thus clarifying the whole, and easing the pain of making changes. C has been designed to make functions efficient and easy to use; C programs generally consist of many small functions rather than a few big ones. A program may reside in one or more source files. Source files may be compiled separately and loaded together, along with previously compiled functions from libraries. We will not go into that process here, however, since the details vary from system to system. Function declaration and definition is the area where the ANSI standard has made the most changes to C. As we saw first in Chapter 1, it is now possible to declare the type of arguments when a function is declared. The syntax of function declaration also changes, so that declarations and definitions match. This makes it possible for a compiler to detect many more errors than it could before. Furthermore, when arguments are properly declared, appropriate type coercions are performed automatically. The standard clarifies the rules on the scope of names; in particular, it requires that there be only one definition of each external object. Initialization is more general: automatic arrays and structures may now be initialized. The C preprocessor has also been enhanced. New preprocessor facilities include a more complete set of conditional compilation directives, a way to create quoted strings from macro arguments, and better control over the macro expansion process.