Top-down programming

Top-down programming is the opposite of bottom-up programming. It refers to a style of programming where an application is constructed starting with a high-level description of what it is supposed to do, and breaking the specification down into simpler and simpler pieces, until a level has been reached that corresponds to the primitives of the programming language to be used.

Disadvantages of top-down programming

Top-down programming complicates testing. Noting executable exists until the very late in the development, so in order to test what has been done so far, one must write stubs .

Furthermore, top-down programming tends to generate modules that are very specific to the application that is being written, thus not very reusable.

But the main disadvantage of top-down programming is that all decisions made from the start of the project depend directly or indirectly on the high-level specification of the application. It is a well-known fact that this specification tends to change over time. When that happens, there is a great risk that large parts of the application need to be rewritten.

How does top-down programming work?

Top-down programming tends to generate modules that are based on functionality, usually in the form of functions or procedures. Typically, the high-level specification of the system states functionality. This high-level description is then refined to be a sequence or a loop of simpler functions or procedures, that are then themselves refined, etc.

In this style of programming, there is a great risk that implementation details of many data structures have to be shared between modules, and thus globally exposed. This in turn makes it tempting for other modules to use these implementation details, thereby creating unwanted dependencies between different parts of the application.