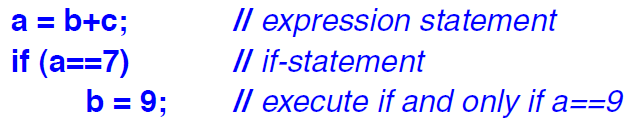
* **Introduction –**
* C++ offers a conventional and flexible set of statements.
* An expression becomes a statement when a semicolon is added at the end.
* Unlike an expression, a statement does not have a value.



* A compiler may reorder code to improve performance as long as the result is identical to that of simple order of execution.
* A semicolon is by itself a statement, an empty statement.
* A (possibly empty) sequence of statements within braces {} is called a ***block*** or ***compound*** statement.
* ***Declaration:*** A statement where there is no assignment statement or procedure-call statement.
* ***Expression:*** Assignments and functions are called expressions.
* ***for-init-statement:*** Must be either a declaration or an expression statement.
* ***for-init-declaration:*** Must be a declaration of a single uninitialised variable.
* C++ has 2 conditional statements –
* **switch:** It makes the code look for compact and doesn’t need to read the same variable again and again.
* **if-else:** Mostly used for complicated conditions and the ones involving more than one variable.
* C++ has 3 loop statements –
* **for:** Used to write compact code and where the initialisation, update and condition check are very clear and organised.
* **while:** Used when the update statement might occur in the middle of the statement. Also, more complicated condition checks are usually handled by while loop.
* **do:** It first implements the statement and then checks the condition, so it will be implemented at least once. It is best avoided if it is not extremely important.
* **goto:** These are usually used to jump to a particular part in the code. It can be used to break out of a loop.
* **Indentation:** There is no particular reason to prefer one type of indentation over another, although every programmer has their own preferred types that are reflected in their code.
* **Comments:** A good comment states what a piece of code is supposed to do, whereas the code states what it does and how it does it.
* A comment is expressed in a suitably high level of abstraction so that it is easy for a human to understand without delving into minute details.
* Comments should preferably be present in the following places –
* A comment for each source file stating what the declarations in it have in common, reference to manuals, the name of the programmer, general hints for maintenance, etc.
* A comment for each class, namespace and template.
* A comment for each non-trivial function stating its purpose, the algorithm used, and maybe something about the assumptions it makes about its environment.
* A comment for each global and namespace variable and constant.
* A few comments where the code is non-obvious or non-portable.
* Very little else.
* /\* \*/ style comments do not nest.
* **Advice –**
* Don’t declare a variable until you have a value to initialise it with.
* Prefer a switch statement to an if-statement when there is a choice.
* Prefer a range-for-statement to a for-statement when there is a choice.
* Prefer a for-statement to a while-statement when the loop variable is obvious.
* Prefer a while-statement to a for-statement when the loop variable is not obvious.
* Avoid do-statements.
* Avoid goto.
* Keep comments crisp.
* Don’t say in comments what can be clearly stated in code.
* State the intent in comments.
* Maintain a consistent indentation style.