* Recursive methods count /execution count
* Iteration count
* Time to build/execute
* OOP concepts (Method inheritance, Attribute inheritance, coupling factor, Clustering factor, Polymorphism factor, Method hiding, Attribute hiding, Reusability factor, Child count, descendant count, Parent count, Ancestor count)
* AOP concepts
* Complexity of called libraries, apis, webservices.
* Entity/class count
* LOC
* Effort to code
* Number of defects
* Program structure
* Nesting levels
* Variable counts
* Methods counts
* Number of function points
* Number of unused imports

**Complexity of a normal method with nesting**

Method A(){ //line 1

int i=0; //line 2

if(a<b){ //line 3

for(int j=0;j<5;j++){ //line 4

for(int k=0;k<5;k++){ //line 5

if(c>d){ //line 6

i++; //line 7

} //line 8

} //line 9

} //line 10

} //line 11

} //line 12

* Normally we can define a formula like N(S+C) to calculate a complexity of a simple code.
* There N is nesting level, S is size which calculates the count line wise basically program statement. We can ignore operators like (), {}, ; and etc as they are dependent upon the programming language even. And C is the control structures like for, if and etc where weights can be given 3,2 and etc respectively because in “**If”** there is only to check one or more conditions but in a loop like “**for”** there may have condition plus the initializations and increments. And also we should multiply the complexity inside the loop by number of iterations.