**Orthogonality** in a programming language means that a relatively small set

of primitive constructs can be combined in a relatively small number of ways

to build the control and data structures of the language. Furthermore, every

possible combination of primitives is legal and meaningful. For example, consider data types. Suppose a language has four primitive data types (integer, float,

double, and character) and two type operators (array and pointer). If the two

type operators can be applied to themselves and the four primitive data types,

a large number of data structures can be defined.

The meaning of an orthogonal language feature is independent of the

context of its appearance in a program. (The word *orthogonal* comes from the

mathematical concept of orthogonal vectors, which are independent of each

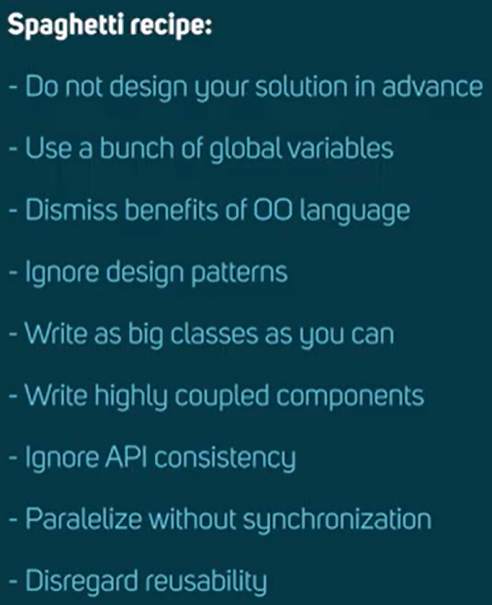
other.) Orthogonality follows from a symmetry of relationships among primitives. A lack of orthogonality leads to exceptions to the rules of the language.

For example, in a programming language that supports pointers, it should be

possible to define a pointer to point to any specific type defined in the language.

However, if pointers are not allowed to point to arrays, many potentially useful

user-defined data structures cannot be defined.



A Control Flow Graph (CFG) is the graphical representation of control flow or computation during the execution of programs or applications. Control flow graphs are mostly used in static analysis as well as compiler applications, as they can accurately represent the flow inside of a program unit.

The control flow graph was originally developed by Frances E. Allen.

Characteristics of Control Flow Graph: 

Control flow graph is process oriented. 

Control flow graph shows all the paths that can be traversed during a program execution. 

Control flow graph is a directed graph. 

Edges in CFG portray control flow paths and the nodes in CFG portray basic blocks.

There exist 2 designated blocks in Control Flow Graph:

1. Entry Block: Entry block allows the control to enter into the control flow graph.
2. Exit Block: Control flow leaves through the exit block. Hence, the control flow graph is comprised of all the building blocks involved in a flow diagram such as the start node, end node and flows between the nodes.

**General Control Flow Graphs:**  
Control Flow Graph is represented differently for all statements and loops. Following images describe it:

if A = 10 then

if B > C

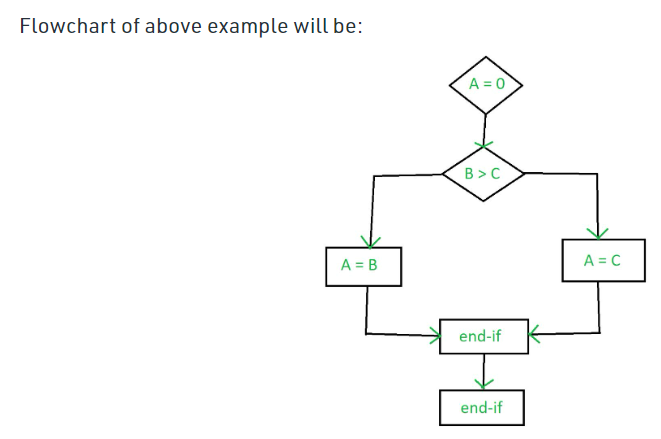
A = B

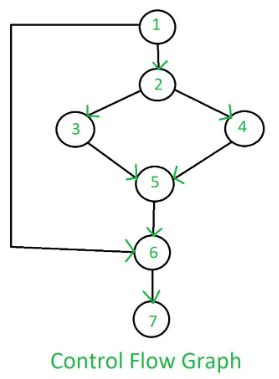
else A = C

endif

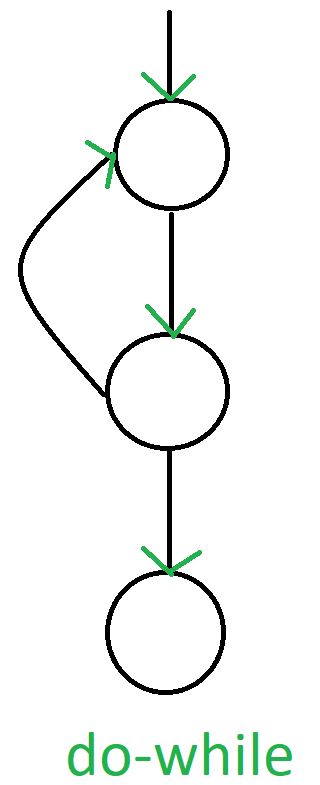
endif

print A, B, C

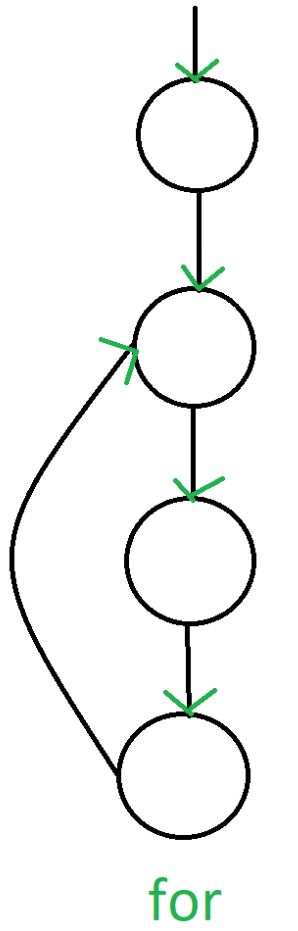




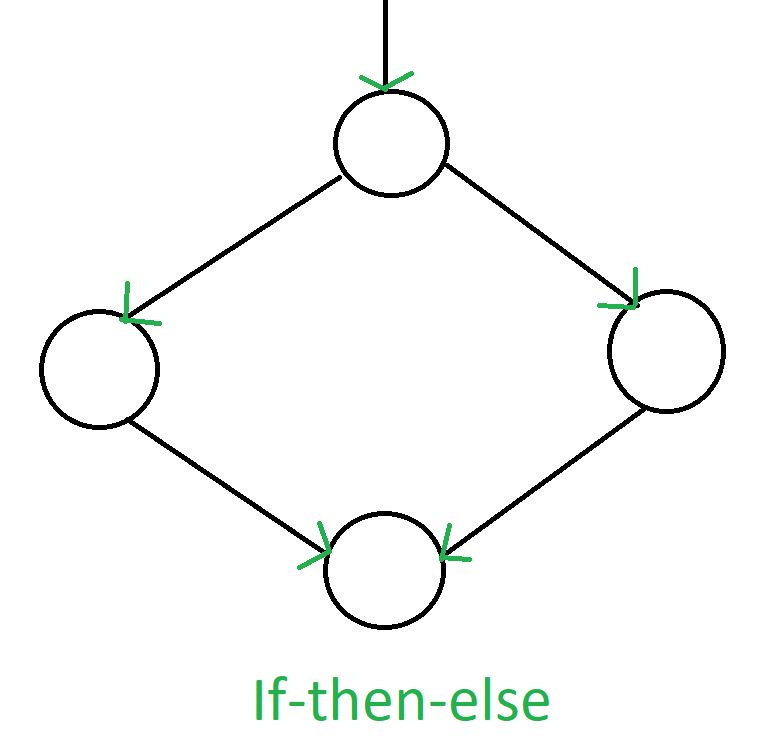
3. **do-while:**



4. **for:**



1. **If-else:**



2. **while:**

