Unit - 3

Decision Control statemnet

Decision Control/Conditional Statements

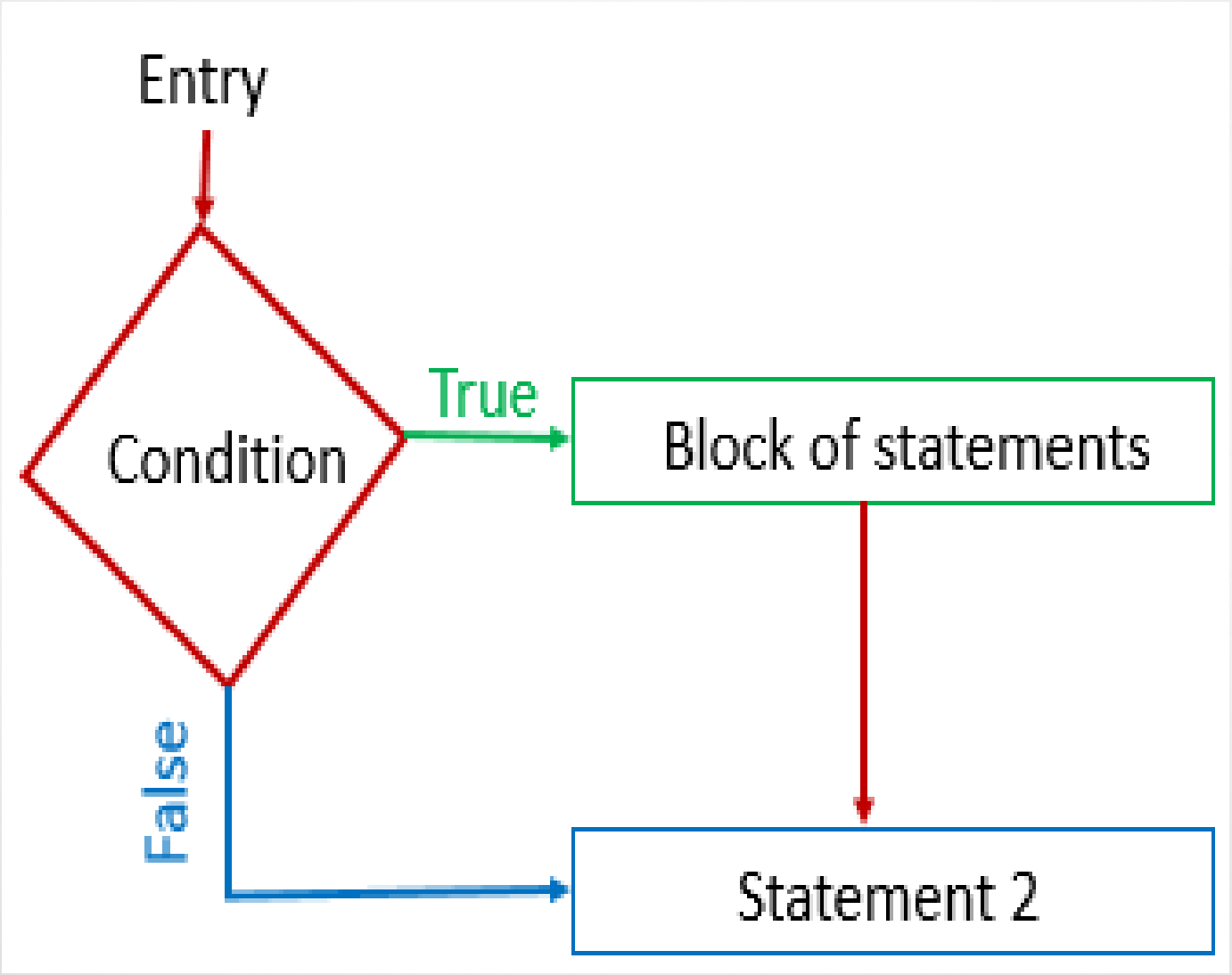
- The if-else statement in C is used to perform the operations based on some specific condition.
- In any programming language, there is a need to perform different tasks based on the condition. For example, consider an online website, when you enter wrong id or password it displays error page and when you enter correct credentials then it displays welcome page. So there must be a logic in place that checks the condition (id and password) and if the condition returns true it performs a task (displaying welcome page) else it performs a different task(displaying error page).
- There are the following types of if statement in C language.
  - If statement
  - If-else statement
  - If else-if ladder
  - Nested if

Decision Control/Conditional Statements

- Decision making structures require that the programmer specifies one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.
- **If Statement:** The statements inside if body executes only when the condition defined by if statement is true.
- **Syntax:**

```
if(boolean_expression)
{
   /* statement(s) will execute if the boolean expression is true */
}
```

Simple If

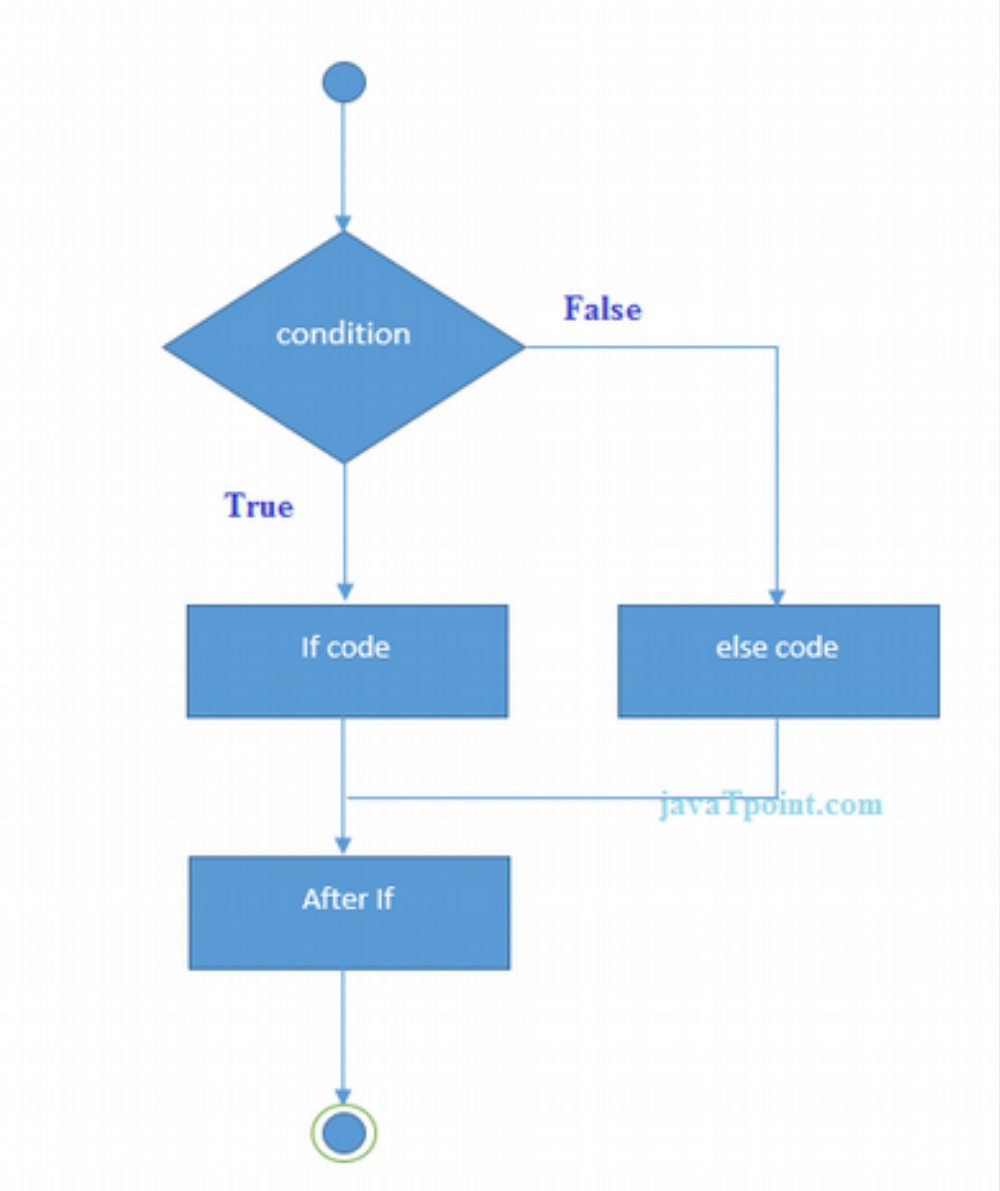

Entry
Condition
True
Block of statements
False
Statement 2

If - Else

- **If-else Statement:** The if-else statement is used to perform two operations for a single condition.
- We can perform two different operations, i.e., one is for the correctness(true) of that condition, and the other is for the incorrectness(false) of the condition.
- **Imp Note: if and else block cannot be executed simiulteneously.**
- **Syntax:**

```
if(boolean_expression) {
   /* statement(s) will execute if the boolean expression is true */
}
else {   /* statement(s) will execute if the boolean expression is false */
}
```

If - Else


condition
False
True
If code
else code
javaTpoint.com
After If

If else-if ladder Statement

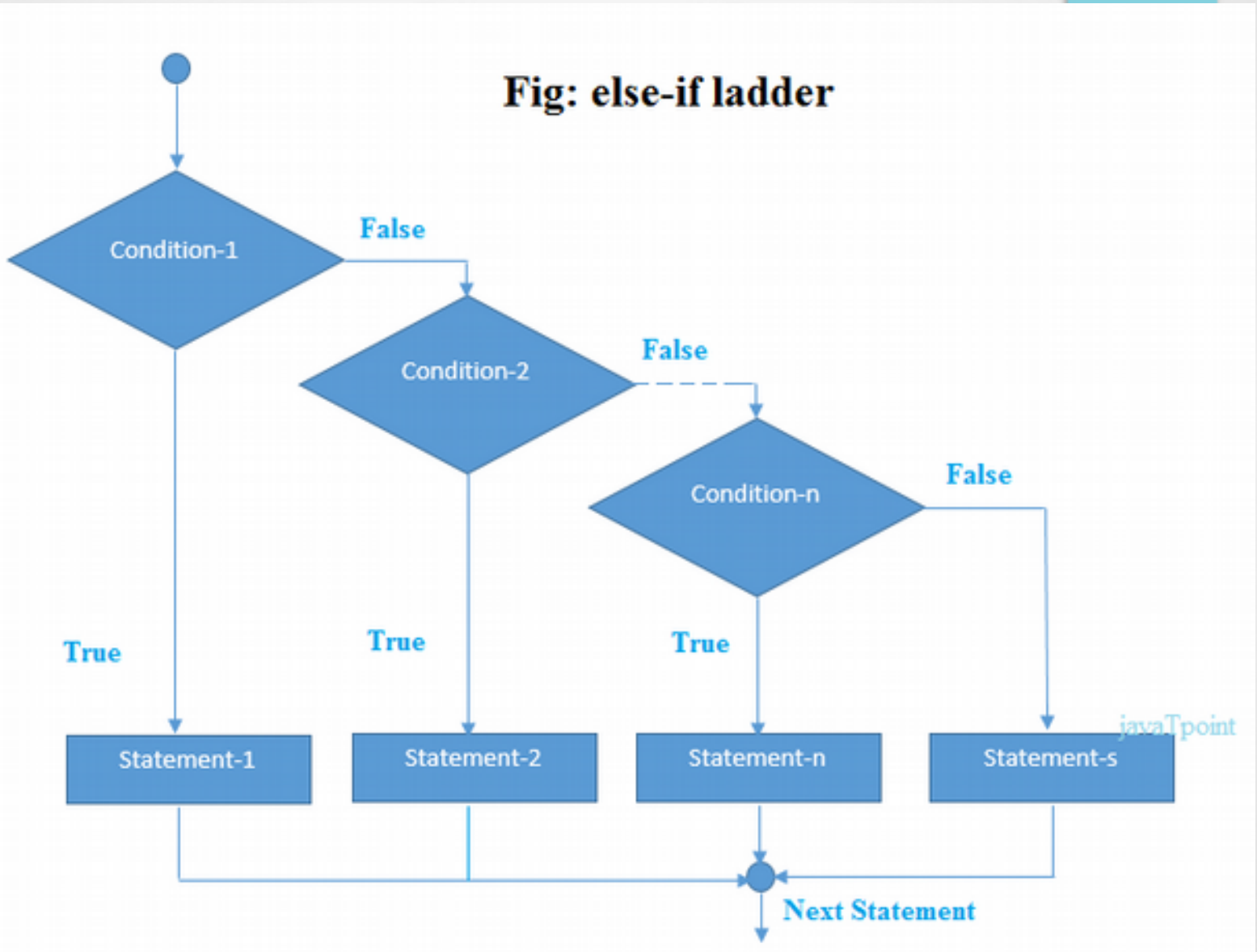
- **If-else-if ladder Statement:** The if-else-if ladder statement is an extension to the if-else statement.
- It is used in the scenario where there are multiple cases to be performed for different conditions.
- In if-else-if ladder statement, if a condition is true then the statements defined in the if block will be executed, otherwise if some other condition is true then the statements defined in the else-if block will be executed, at the last if none of the condition is true then the statements defined in the else block will be executed.
- There are multiple else-if blocks possible.
- It is similar to the switch case statement where the default is executed instead of else block if none of the cases is matched.

If else-if ladder Statement

- **Syntax:**

```
if(condition1){
//code to be executed if condition1 is true
}else if(condition2){
//code to be executed if condition2 is true
}
else if(condition3){
//code to be executed if condition3 is true
}
...
else{
//code to be executed if all the conditions are false
}
```

If else-if ladder Statement

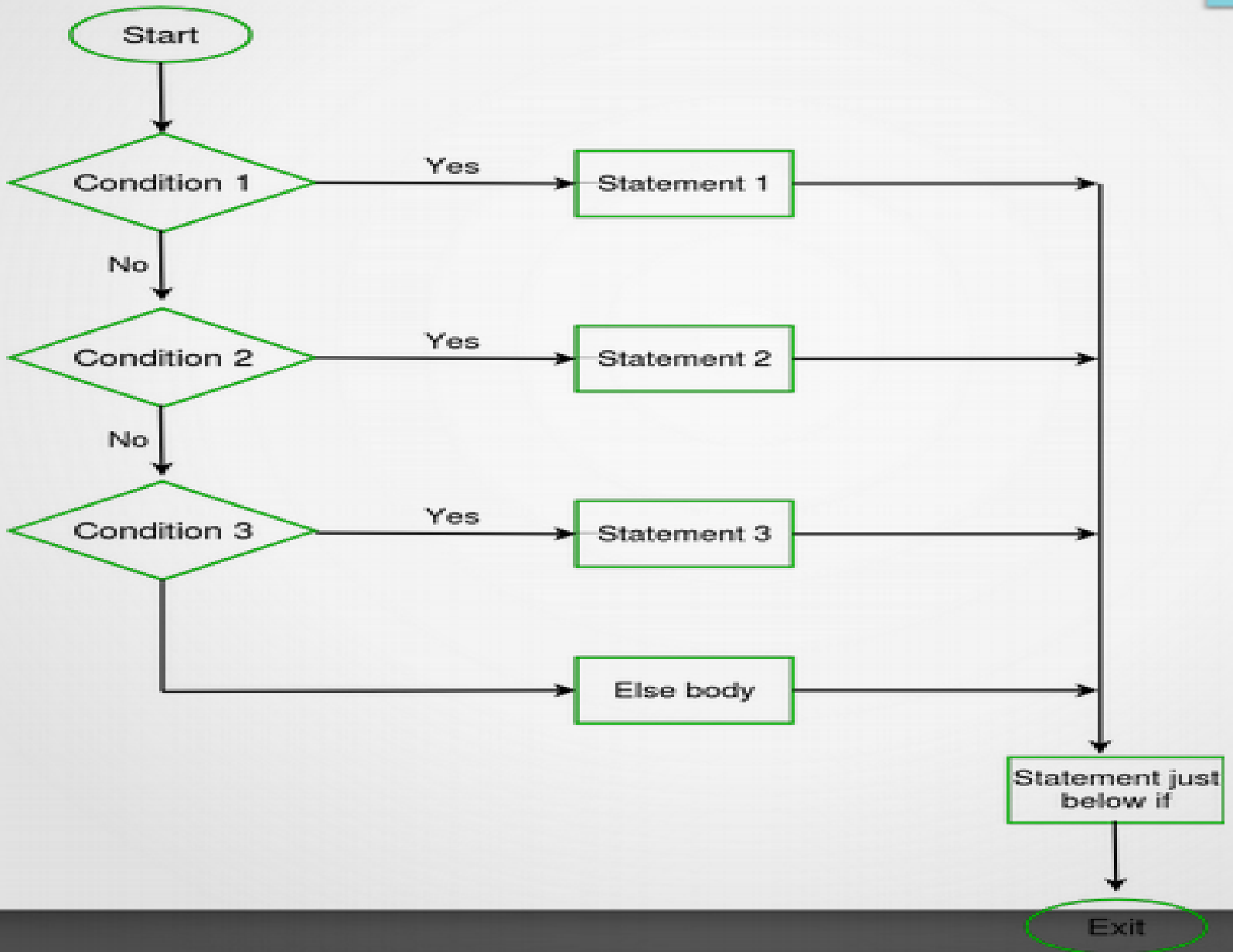


Nested If

- **Nested If: Placing If Statement inside another IF Statement**
- When an if else statement is present inside the body of another “if” or “else” then this is called nested if else.
- **Syntax:**

```
if( boolean_expression 1)
{
/* Executes when the boolean expression 1 is true */
   if(boolean_expression 2)
{
      /* Executes when the boolean expression 2 is true */
   }
}
```

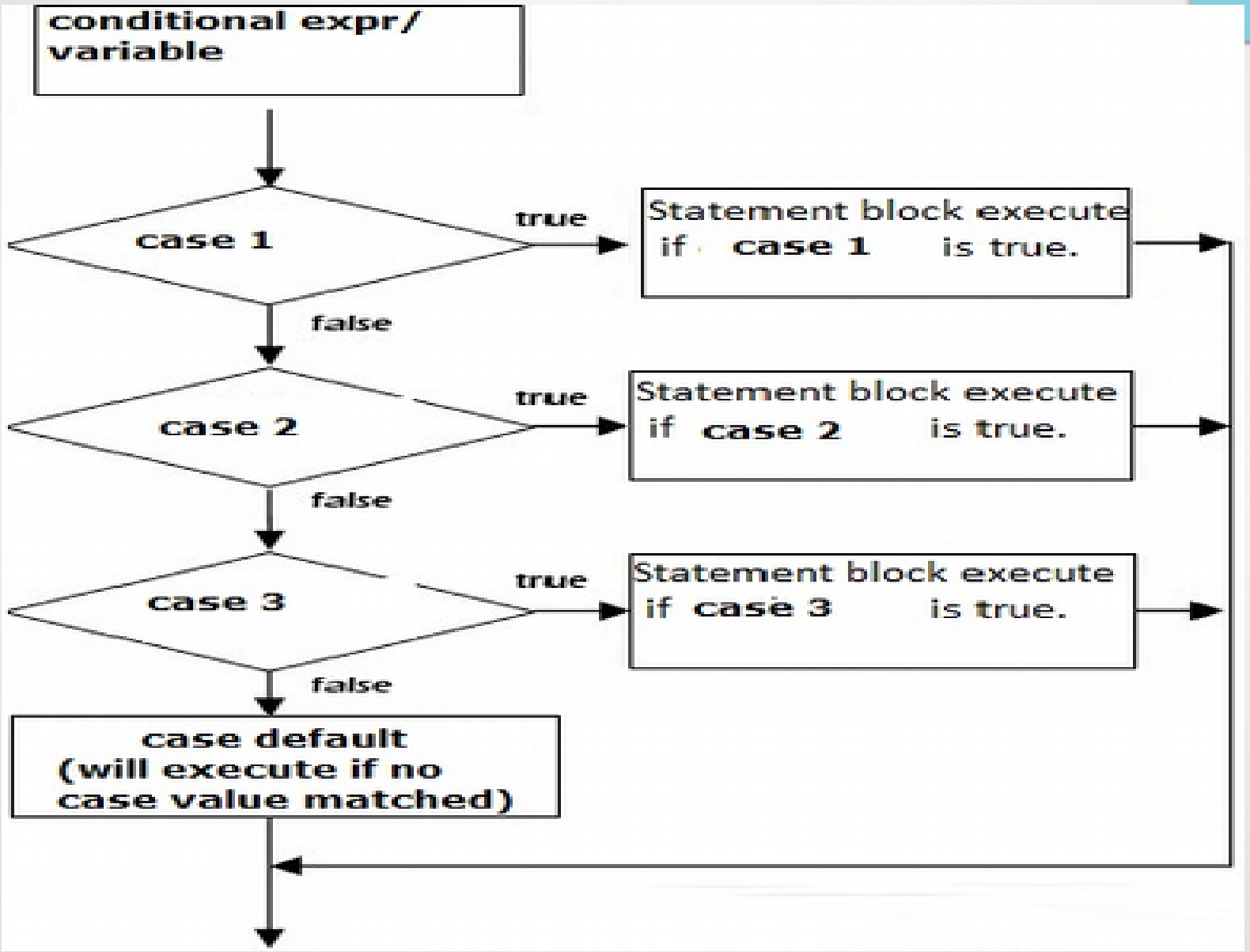
Nested If


Start
Condition 1
Yes
Statement 1
No
Condition 2
Yes
Statement 2
No
Condition 3
Yes
Statement 3
Else body
Statement just below if
Exit

Switch Case

- Switch case is a branching statement used to perform action based on available choices, instead of making decisions based on conditions. Using switch case you can write more clean and optimal code than if else statement.
- **Switch case only works with integer, character and enumeration constants.**
- A switch statement allows a variable to be tested for equality against a list of values. Each value is called a **case**, and the variable being switched on is checked for each switch case.

Switch Case


conditional expr/
variable
case 1
true
Statement block execute
if case 1 is true.
false
case 2
true
Statement block execute
if case 2 is true.
false
case 3
true
Statement block execute
if case 3 is true.
false
case default
(will execute if no
case value matched)

Switch Case Syntax

```
switch(expression) {
   case constant-expression  :
      statement(s);
      break;

   case constant-expression  :
      statement(s);
      break;

   /* you can have any number of case statements */
   default :
   statement(s);
}
```

Terms And condition For Switch Case

- The variable used in switch must be of int or char datatype.If one is using char then the value should be in single quate ('a').
- You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.
- You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.
- When a break statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement

Terms And condition For Switch Case

- Not every case needs to contain a break. If no break appears, the flow of control will fall through to subsequent cases until a break is reached.
- A switch statement can have an optional default case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No break is needed in the default case.