C - Decision Making

Every programming language including C has **decision-making statements** to support conditional logic. C has a number of alternatives to add decision-making in the code.

Any process is a combination of three types of logic –

- Sequential logic
- Decision or branching
- Repetition or iteration

A computer program is sequential in nature and runs from top to bottom by default. The decision-making statements in C provide an alternative line of execution. You can ask a group of statements to be repeatedly executed till a condition is satisfied.

The decision-making structures control the program flow based on conditions. They are important tools for designing complex algorithms.

We use the following **keywords and operators** in the decision-making statements of a C program – **if**, **else**, **switch**, **case**, **default**, **goto**, **the ?: operator**, **break**, and **continue** statements.

In programming, we come across situations when we need to make some decisions. Based on these decisions, we decide what should we do next. Similar situations arise in algorithms too where we need to make some decisions and based on these decisions, we will execute the next block of code.

The next instruction depends on a Boolean expression, whether the condition is determined to be True or False. C programming language assumes any non-zero and non-null values as True, and if it is either zero or null, then it is assumed as a False value.

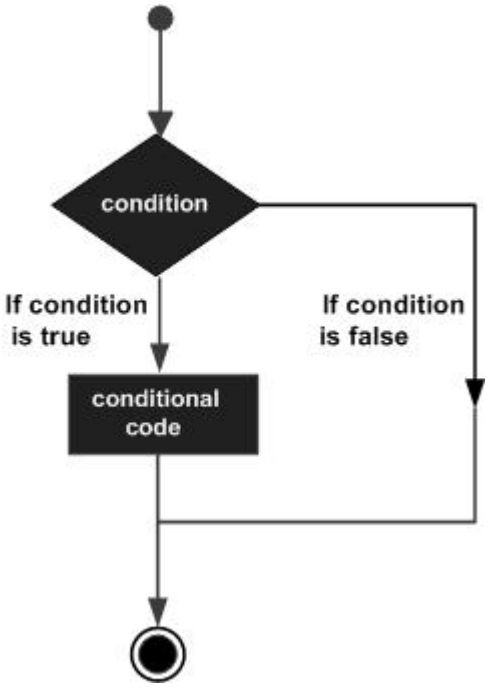
C programming language provides the following types of decision making statements.

Sr.No.	Statement & Description
1	**if statement** An **if** statement consists of a boolean expression followed by one or more statements.
2	**if...else statement** An **if** statement can be followed by an optional **else** statement, which executes when the Boolean expression is false.

3	**nested if statements** You can use one **if** or **else-if** statement inside another **if** or **else-if** statement(s).
4	**switch statement** A **switch** statement allows a variable to be tested for equality against a list of values.
5	**nested switch statements** You can use one **switch** statement inside another **switch** statement(s).

If Statement in C Programming

The **if** statement is used for deciding between two paths based on a True or False outcome. It is represented by the following flowchart –



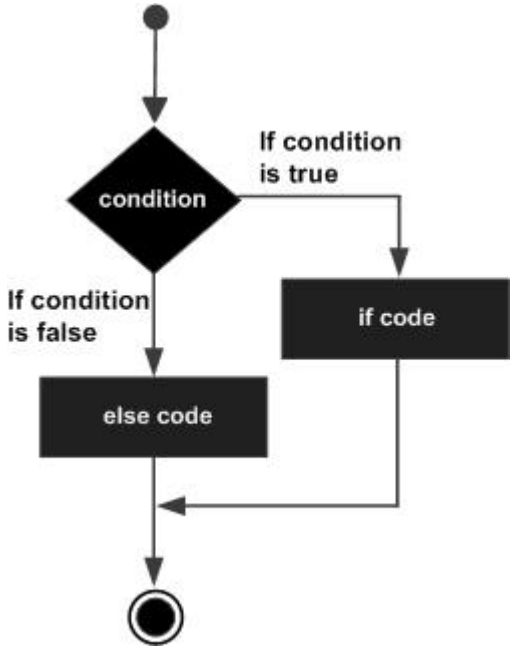
Syntax

```
if (Boolean expr){
   expression;
   . . .
}
```

An **if** statement consists of a boolean expression followed by one or more statements.

If...else Statement in C Programming

The if–else statement offers an alternative path when the condition isn't met.



Syntax

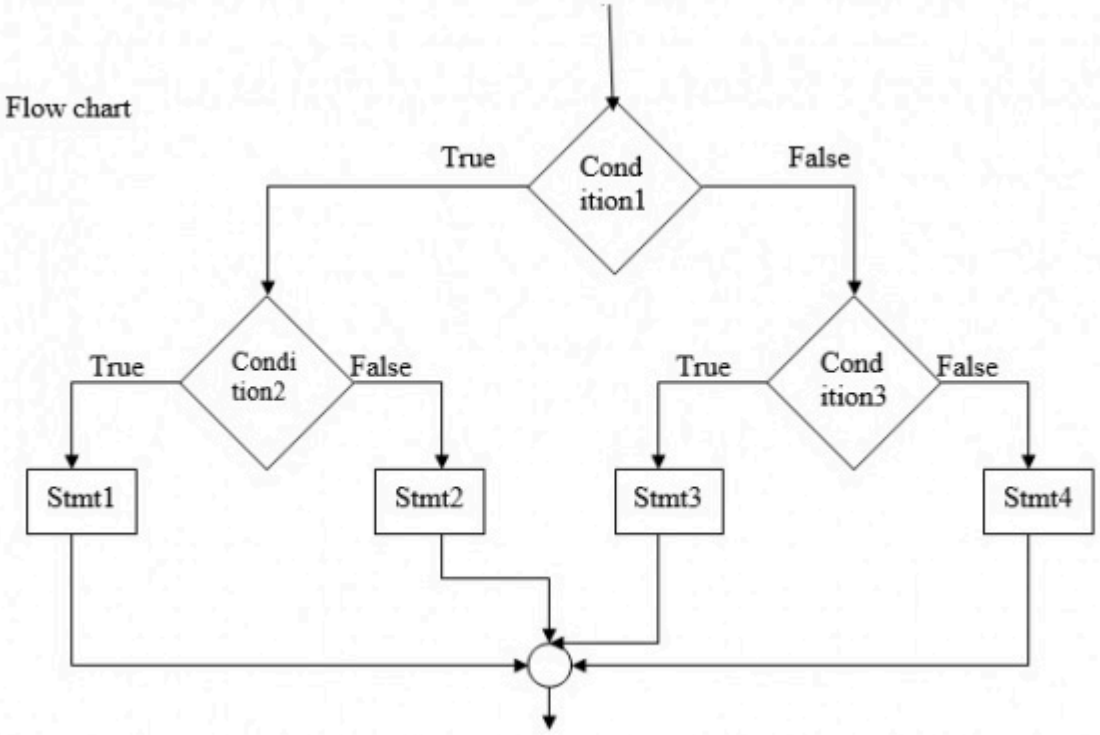
```
if (Boolean expr){
   expression;
   . . .
}
else{
   expression;
   . . .
}
```

An **if** statement can be followed by an optional **else** statement, which executes when the Boolean expression is false.

Explore our **latest online courses** and learn new skills at your own pace. Enroll and become a certified expert to boost your career.

Nested If Statements in C Programming

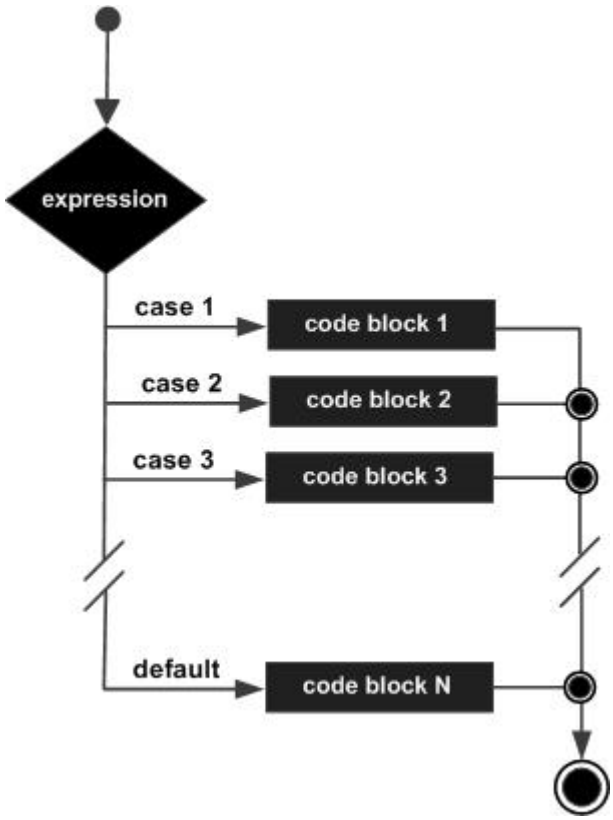
Nested if statements are required to build intricate decision trees, evaluating multiple nested conditions for nuanced program flow.



You can use one **if** or **else-if** statement inside another **if** or **else-if** statement(s).

Switch Statement in C Programming

A switch statement simplifies multi-way choices by evaluating a single variable against multiple values, executing specific code based on the match. It allows a variable to be tested for equality against a list of values.



Syntax

```
switch(expression) {

   case constant-expression  :
      statement(s);
      break; /* optional */

   case constant-expression  :
      statement(s);
      break; /* optional */

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

As in if statements, you can use one switch statement inside another switch statement(s).

The ?: Operator in C Programming

We have covered the **conditional operator (?:)** in the previous chapter which can be used to replace **if-else** statements. It condenses an if-else statement into a single expression, offering compact and readable code.

It has the following general form –

```
Exp1 ? Exp2 : Exp3;
```

Where Exp1, Exp2, and Exp3 are expressions. Notice the use and placement of the colon (:). The value of a "?" expression is determined like this –

Exp1 is evaluated. If it is true, then Exp2 is evaluated and becomes the value of the entire **? expression**.

If Exp1 is false, then Exp3 is evaluated and its value becomes the value of the **: expression**.

You can simulate nested if statements with the **?** operator. You can use another ternary operator in true and/or false operand of an existing **?** operator.

An algorithm can also have an iteration logic. In C, the **while**, **do–while** and **for** statements are provided to form loops.

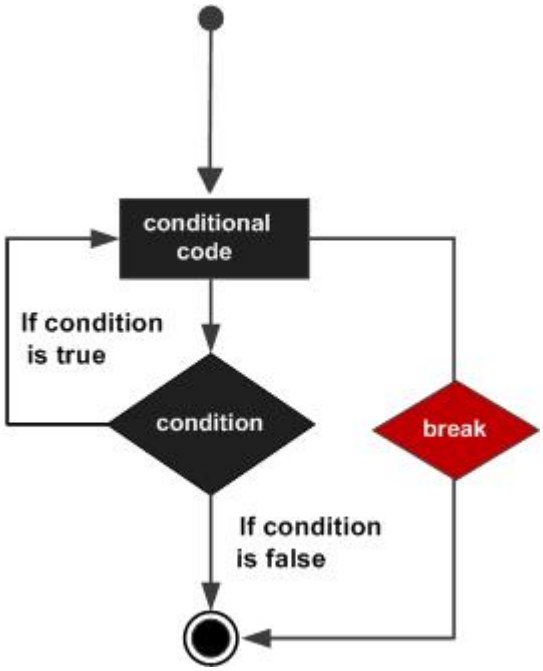
The loop formed by **while** and **do–while** are conditional loops, whereas the **for** statement forms a counted loop.

The loops are also controlled by the Boolean expressions. The C compiler decides whether the looping block is to be repeated again, based on a condition.

The program flow in a loop is also controlled by different **jumping statements**. The **break** and **continue** keywords cause the loop to terminate or perform the next iteration.

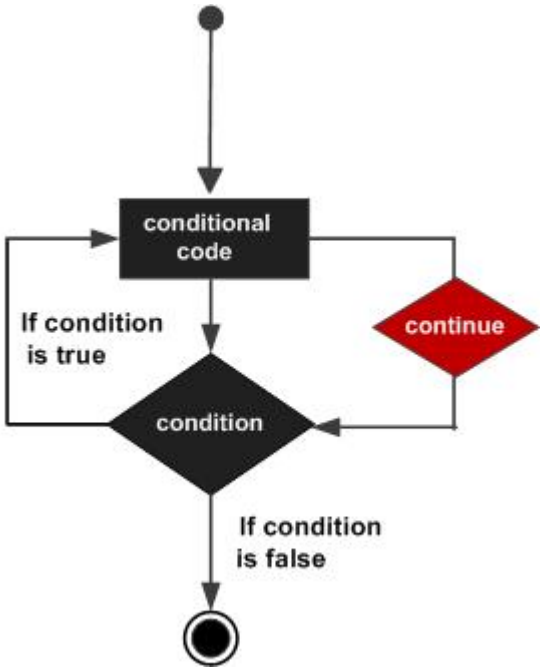
The Break Statement in C Programming

In C, the break statement is used in switch–case constructs as well as in loops. When used inside a loop, it causes the repetition to be abandoned.



The Continue Statement in C Programming

In C, the continue statement causes the conditional test and increment portions of the loop to execute.

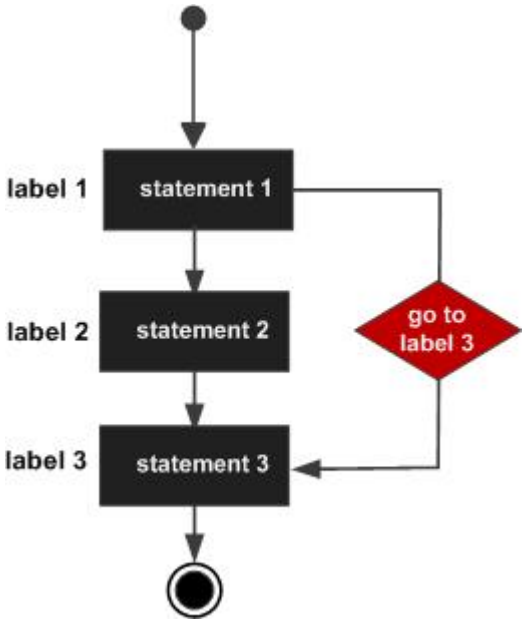


The goto Statement in C Programming

C also has a goto keyword. You can redirect the program flow to any labelled instruction in the program.

Here is the **syntax** of the goto statement in C –

```
goto label;
..
.
label: statement;
```



With the goto statement, the flow can be directed to any previous step or any subsequent step.

In this chapter, we had a brief overview of the decision-making statements in C. In the subsequent chapters, we will have an elaborate explanation on each of these decision-making statements, with suitable examples.