C - Nested If Statements

It is always legal in C programming to nest **if-else** statements, which means you can use one **if** or **else-if** statement inside another **if** or **else-if** statement(s).

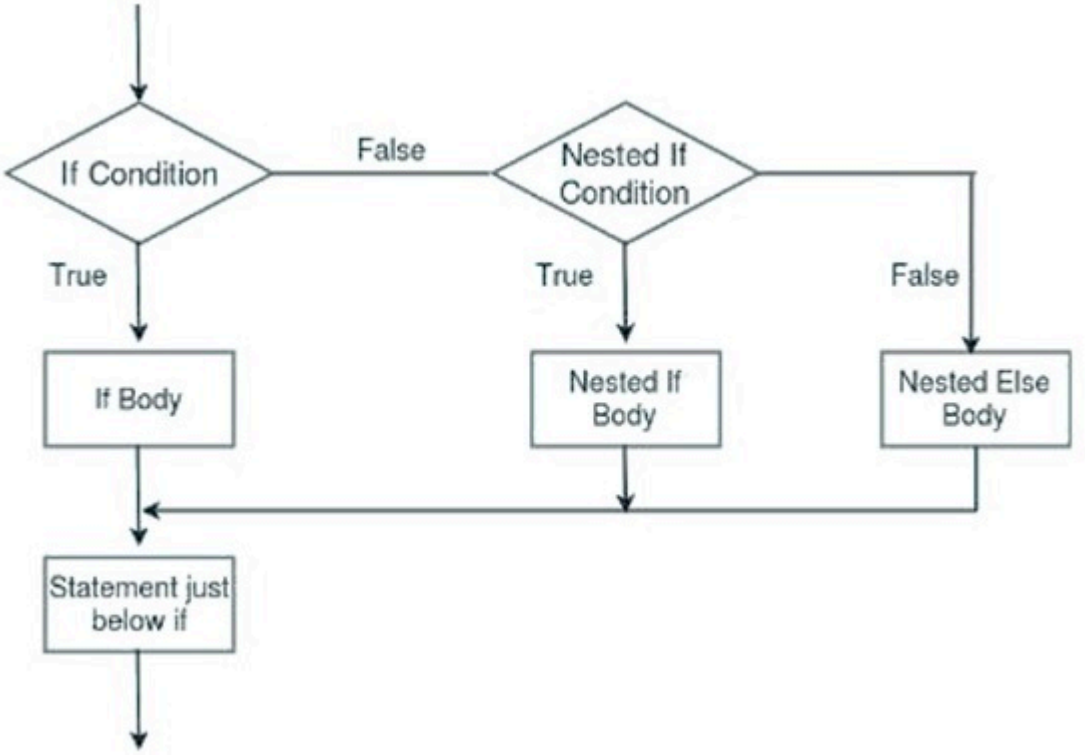
In the programming context, the term "nesting" refers to enclosing a particular programming element inside another similar element. For example, nested loops, nested structures, nested conditional statements, etc. If an **if** statement in C is employed inside another **if** statement, then we call it as a **nested if statement** in C.

Syntax

The syntax of **nested if** statements is as follows −

```
if (expr1){
   if (expr2){
      block to be executed when
      expr1 and expr2 are true
   }
   else{
      block to be executed when
      expr1 is true but expr2 is false
   }
}
```

The following flowchart represents the nesting of **if** statements −



You can compound the Boolean expressions with **&&** or **||** to get the same effect. However, for more complex algorithms, where there are different combinations of multiple Boolean expressions, it becomes difficult to form the compound conditions. Instead, it is recommended to use nested structures.

Another **if** statement can appear inside a top-level **if** block, or its **else** block, or inside both.

Example 1

Let us take an example, where the program needs to determine if a given number is less than 100, between 100 to 200, or above 200. We can express this logic with the following compound Boolean expression –

</> Open Compiler

```
#include <stdio.h>

int main (){

   // local variable definition
   int a = 274;
   printf("Value of a is : %d\n", a);

   if (a < 100){
      printf("Value of a is less than 100\n");
```

```
   }

   if (a >= 100 && a < 200){
      printf("Value of a is between 100 and 200\n" );
   }

   if (a >= 200){
      printf("Value of a is more than 200\n" );
   }
}
```

Output

Run the code and check its output. Here, we have intialized the value of "a" as 274. Change this value and run the code again. If the supplied value is less than 100, then you will get a different output. Similarly, the output will change again if the supplied number is in between 100 and 200.

```
Value of a is : 274
Value of a is more than 200
```


Example 2

Now let's use nested conditions for the same problem. It will make the solution more understandable when we use nested conditions.

First, check if "a >= 100". Inside the true part of the **if** statement, check if it is <200 to decide if the number lies between 100-200, or it is >200. If the first condition (a >= 100) is false, it indicates that the number is less than 100.

</> Open Compiler

```
#include <stdio.h>

int main (){

   // local variable definition
   // check with different values 120, 250 and 74
   int a = 120;
```

```
   printf("value of a is : %d\n", a );

   // check the boolean condition
   if(a >= 100){

      // this will check if a is between 100-200
      if(a < 200){
         // if the condition is true, then print the following
         printf("Value of a is between 100 and 200\n" );
      }
      else{
          printf("Value of a is more than 200\n");
      }
   }
   else{
       // executed if a < 100
       printf("Value of a is less than 100\n");
   }

   return 0;
}
```

Output

Run the code and check its output. You will get different outputs for different input values of "a" −

```
Value of a is : 120
Value of a is between 100 and 200
```

Example 3

The following program uses nested **if** statements to determine if a number is divisible by 2 and 3, divisible by 2 but not 3, divisible by 3 but not 2, and not divisible by both 2 and 3.

</> Open Compiler

```
#include <stdio.h>

int main(){
   int a = 15;
```

```
   printf("a: %d\n", a);

   if (a % 2 == 0) {
      if (a % 3 == 0){
         printf("Divisible by 2 and 3");
      }
      else {
         printf("Divisible by 2 but not 3");
      }
   }
   else {
      if (a % 3 == 0){
         printf("Divisible by 3 but not 2");
      }
      else{
         printf("Not divisible by 2, not divisible by 3");
      }
   }

   return 0;
}
```

Output

Run the code and check its output –

```
a: 15
Divisible by 3 but not 2
```

For different values of "a", you will get different outputs.

Example 4

Given below is a C program to check if a given year is a leap year or not. Whether the year is a leap year or not is determined by the following rules –

- Is the year divisible by 4?
- If yes, is it a century year (divisible by 100)?
- If yes, is it divisible by 400? If yes, it is a leap year, otherwise not.
- If it is divisible by 4 and not a century year, it is a leap year.
- If it is not divisible by 4, it is not a leap year.

Here is the C code –

</> Open Compiler

```
#include <stdio.h>

int main(){

   // Test the program with the values 1900, 2023, 2000, 2012
   int year = 1900;
   printf("year: %d\n", year);

   // is divisible by 4?
   if (year % 4 == 0){

      // is divisible by 100?
      if (year % 100 == 0){

         // is divisible by 400?
         if(year % 400 == 0){
            printf("%d is a Leap Year\n", year);
         }
         else{
            printf("%d is not a Leap Year\n", year);
         }
      }
      else{
         printf("%d is not a Leap Year\n", year);
      }
   }
   else{
      printf("%d is a Leap Year\n", year);
   }
}
```

Output

Run the code and check its output –

```
year: 1900
1900 is not a Leap Year
```

Test the program with different values for the variable "year" such as 1900, 2023, 2000, 2012.

The same result can be achieved by using the compound Boolean expressions instead of nested if statements, as shown below –

```
If (year % 4 == 0 && (year % 400 == 0 || year % 100 != 0)){
   printf("%d is a leap year", year);
}
else{
   printf("%d is not a leap year", year);
}
```

With **nested if** statements in C, we can write structured and multi-level decision-making algorithms. They simplify coding the complex discriminatory logical situations. Nesting too makes the program more readable, and easy to understand.