Decision Making in C

1. if in C

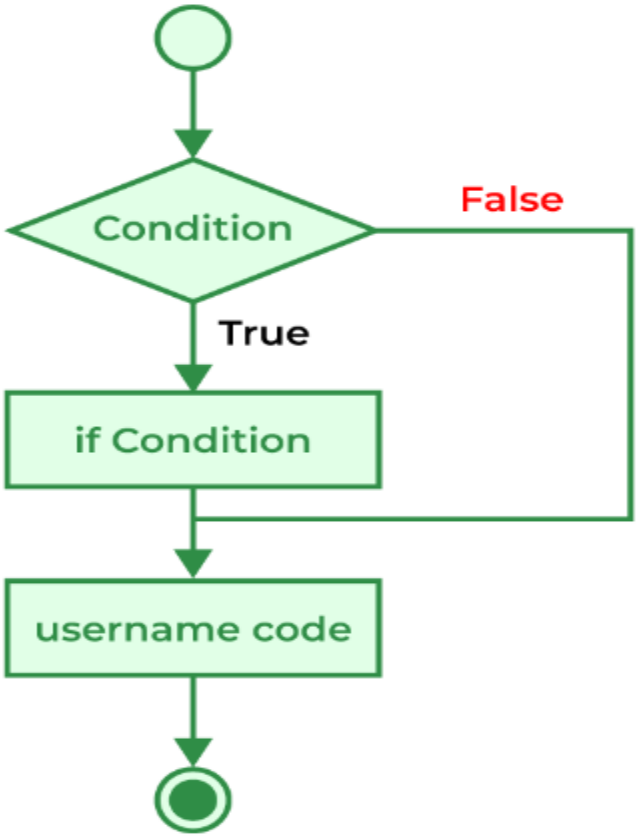
The if statement is the most simple decision-making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e if a certain condition is true then a block of statements is executed otherwise not.

Syntax of if Statement

```
if(condition)
{
  // Statements to execute if
  // condition is true
}
```

Here, the **condition** after evaluation will be either true or false. C if statement accepts boolean values – if the value is true then it will execute the block of statements below it otherwise not. If we do not provide the curly braces '{' and '}' after if(condition) then by default if statement will consider the first immediately below statement to be inside its block.

Flowchart of if Statement



```
// C program to illustrate If statement
#include <stdio.h>

int main()
{
   int i = 10;

   if (i > 15) {
      printf("10 is greater than 15");
   }

   printf("I am Not in if");
}
```

Output

I am Not in if

As the condition present in the if statement is false. So, the block below the if statement is not executed.

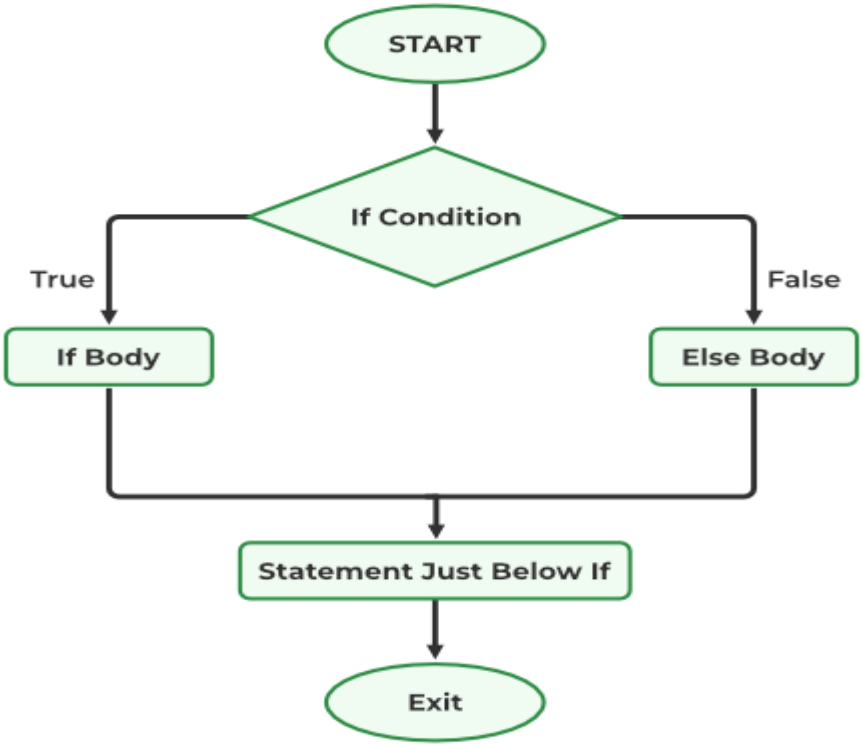
2. if-else in C/C++

The *if* statement alone tells us that if a condition is true it will execute a block of statements and if the condition is false it won't. But what if we want to do something else when the condition is false? Here comes the C *else* statement. We can use the *else* statement with the *if* statement to execute a block of code when the condition is false. The if-else statement consists of two blocks, one for false expression and one for true expression.

Syntax of if else in C/C++

```
if (condition)
{
   // Executes this block if
   // condition is true
}
else
{
   // Executes this block if
   // condition is false
}
```

Flowchart of if-else Statement




```
// C program to illustrate If statement
#include <stdio.h>

int main()
{
   int i = 20;

   if (i < 15) {

      printf("i is smaller than 15");
   }
   else {

      printf("i is greater than 15");
   }
   return 0;
}
```

Output

i is greater than 15

The block of code following the *else* statement is executed as the condition present in the *if* statement is false.

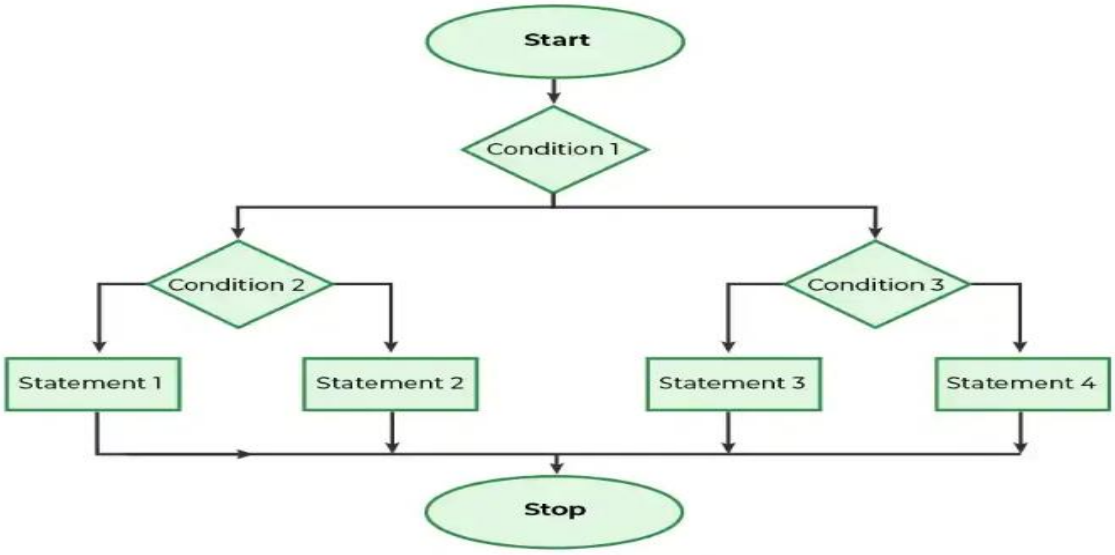
3. Nested if-else in C/C++

A nested if in C is an if statement that is the target of another if statement. Nested if statements mean an if statement inside another if statement. Yes, both C and C++ allow us to nested if statements within if statements, i.e, we can place an if statement inside another if statement.

Syntax of Nested if-else

```
if (condition1)
{
  // Executes when condition1 is true
  if (condition2)
  {
    // Executes when condition2 is true
  }
  else
  {
      // Executes when condition2 is false
}
```

Flowchart of Nested if-else



```
// C program to illustrate nested-if statement
#include <stdio.h>

int main()
{
    int i = 10;

    if (i == 10) {
        // First if statement
        if (i < 15)
            printf("i is smaller than 15\n");

        // Nested - if statement
        // Will only be executed if statement above
        // is true
        if (i < 12)
            printf("i is smaller than 12 too\n");
        else
            printf("i is greater than 15");
    }

    return 0;
}
```

Output

i is smaller than 15

i is smaller than 12 too

4. if-else-if Ladder in C/C++

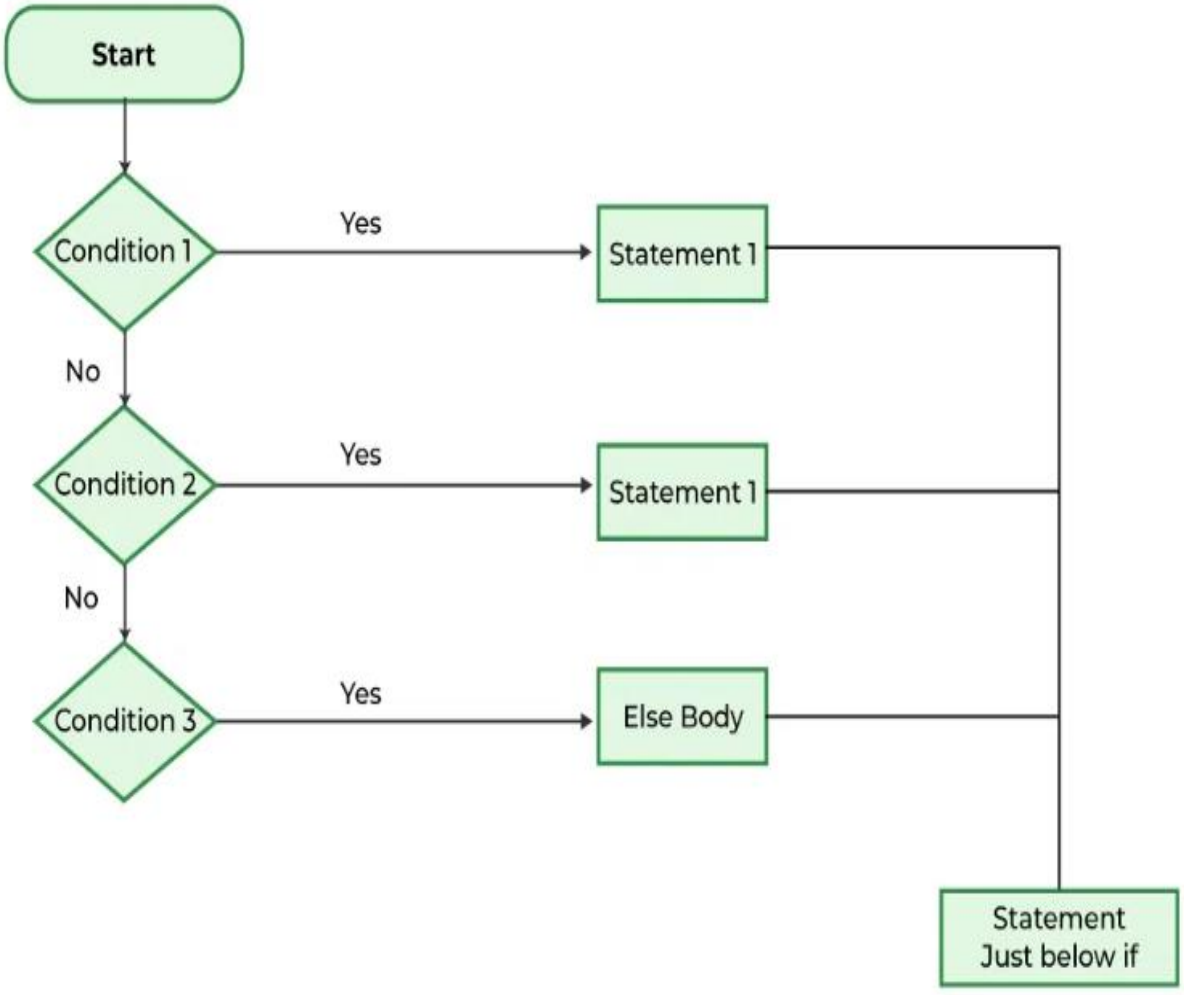
The if else if statements are used when the user has to decide among multiple options. The C if statements are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the C else-if ladder is bypassed. If none of the conditions is true, then the final else statement will be executed. if-else-if ladder is similar to the switch statement.

Syntax of if-else-if Ladder

if (*condition*)
    statement;
else if (*condition*)
    statement;
.
.

else
    statement;

Flowchart of if-else-if Ladder



```c
// C program to illustrate nested-if statement
#include <stdio.h>

int main()
{
    int i = 20;

    if (i == 10)
        printf("i is 10");
    else if (i == 15)
        printf("i is 15");
    else if (i == 20)
        printf("i is 20");
    else
        printf("i is not present");
}
```

Output

i is 20

C Program to Check Whether a Number is Prime or Not

```c
#include <stdio.h>
int main() {
  int n, i, flag = 0;
  printf("Enter a positive integer: ");
  scanf("%d", &n);

  // 0 and 1 are not prime numbers
  // change flag to 1 for non-prime number
  if (n == 0 || n == 1)
    flag = 1;
  for (i = 2; i <= n / 2; ++i) {
    // if n is divisible by i, then n is not prime
    // change flag to 1 for non-prime number
    if (n % i == 0) {
      flag = 1;
      break;
    }
  }

  // flag is 0 for prime numbers
  if (flag == 0)
    printf("%d is a prime number.", n);
  else
    printf("%d is not a prime number.", n);
  return 0;
}
```

Run Code

Output

```
Enter a positive integer: 29
29 is a prime number.
```