Conditionals and Logic flow control

By Saddam Khan

Topics Covered:-

Conditional Interactions: alert, prompt and confirm.
If condition.
If else condition .
Nested if else and else if.
Switch case.
Ternary conditions.
Data Types-2 (Nan, Undefined, Null).

Conditional Interactions:

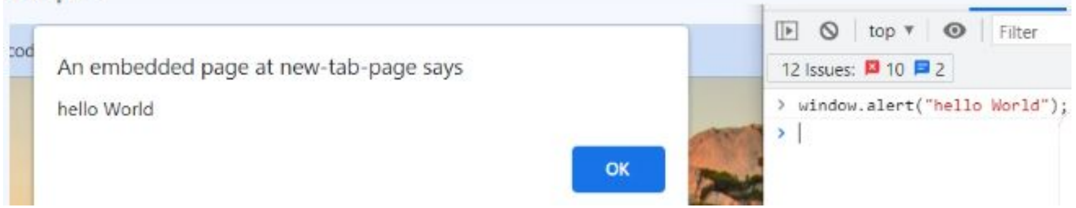
Here we look at various Javascript Functions that help the browser to interact with the user. Here we deal with an alert, prompt, Confirm javascript function.

Alert

This is displayed when we need user confirmation. i.e. a sequence of happening will not happen until the user gives ok assurance on the alert prompt displayed by the user.
We can call the alert function in majorly two ways they are:

The first way is using:-
 Code :window.alert("hello World");

Output:

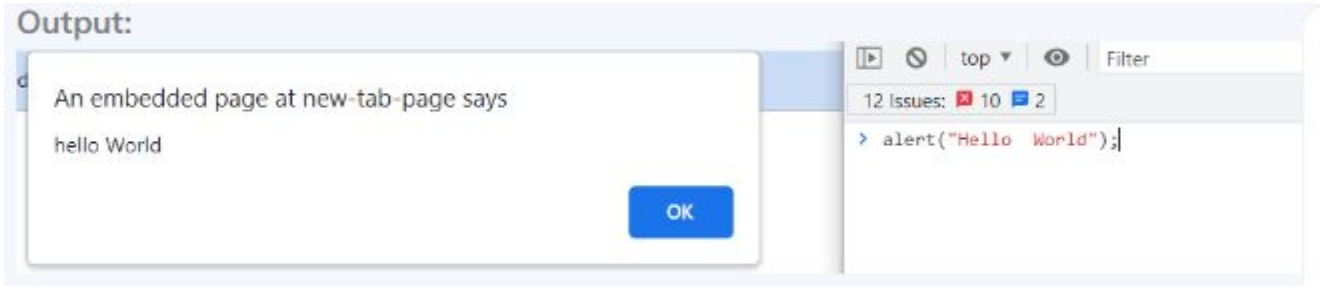

An embedded page at new-tab-page says

hello World

```
> window.alert("hello World");
```

The second way is using:-

Code: alert("hello World");


Output:

An embedded page at new-tab-page says

hello World

```
> alert("Hello  World");
```

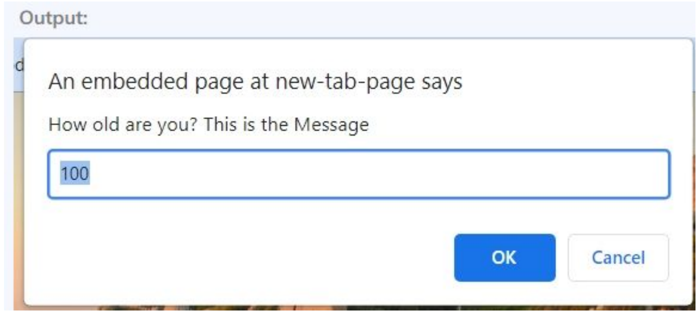
Prompt

The function prompt is used to take input from the modal window and display the message after we press ok on the modal window.

Working:
The working of the prompt function is achieved in such a way that it First displays the text message on the modal window, then takes the user's input, and then displays the message.

```
Code; let getage = prompt('How old are you? This is the Message', 100);
                //100 is the default value
    alert(`You are ${getage} years old!`);
```

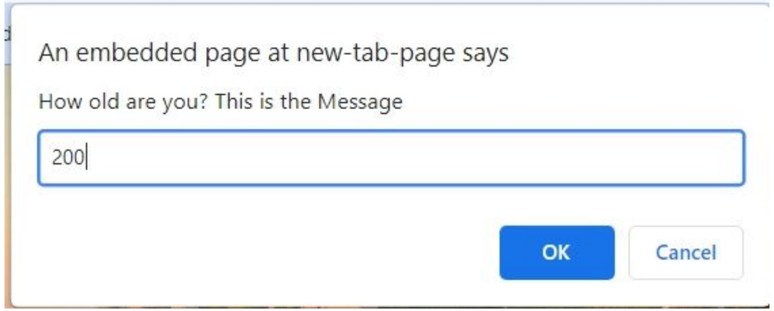
Output:



An embedded page at new-tab-page says

How old are you? This is the Message

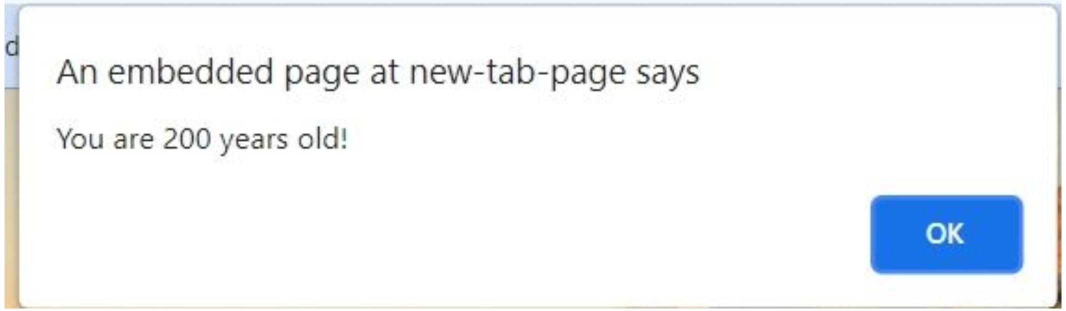
100



An embedded page at new-tab-page says

How old are you? This is the Message

200



An embedded page at new-tab-page says

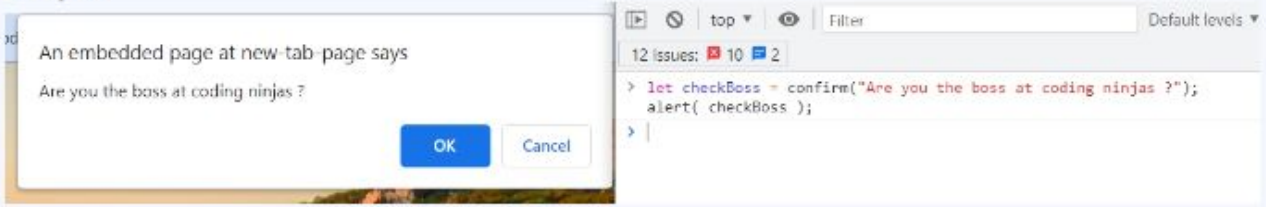
You are 200 years old!

Confirm

The confirm function is used to get the boolean input, i.e. if ok is pressed at the modal window that it returns true else, it returns false.

Code: let checkBoss = confirm("Are you the boss at coding ninjas ?");
alert(checkBoss);

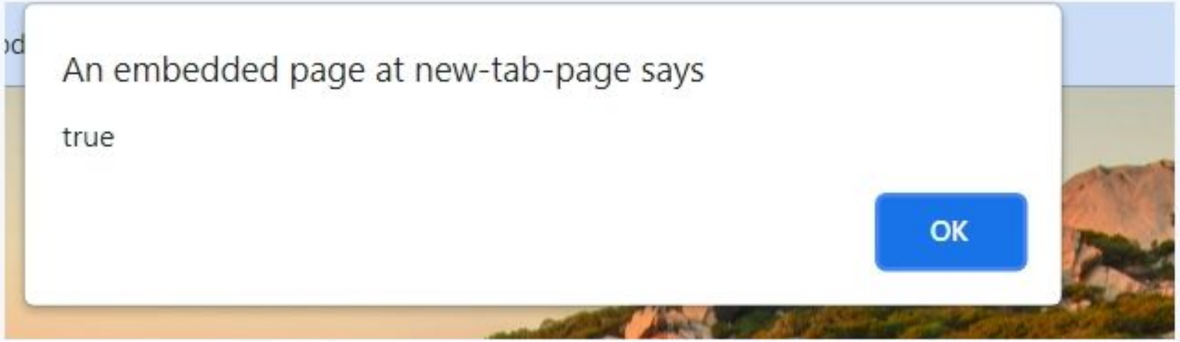
Output:



An embedded page at new-tab-page says

Are you the boss at coding ninjas ?

```
> let checkBoss = confirm("Are you the boss at coding ninjas ?");
  alert( checkBoss );
>
```



An embedded page at new-tab-page says

true

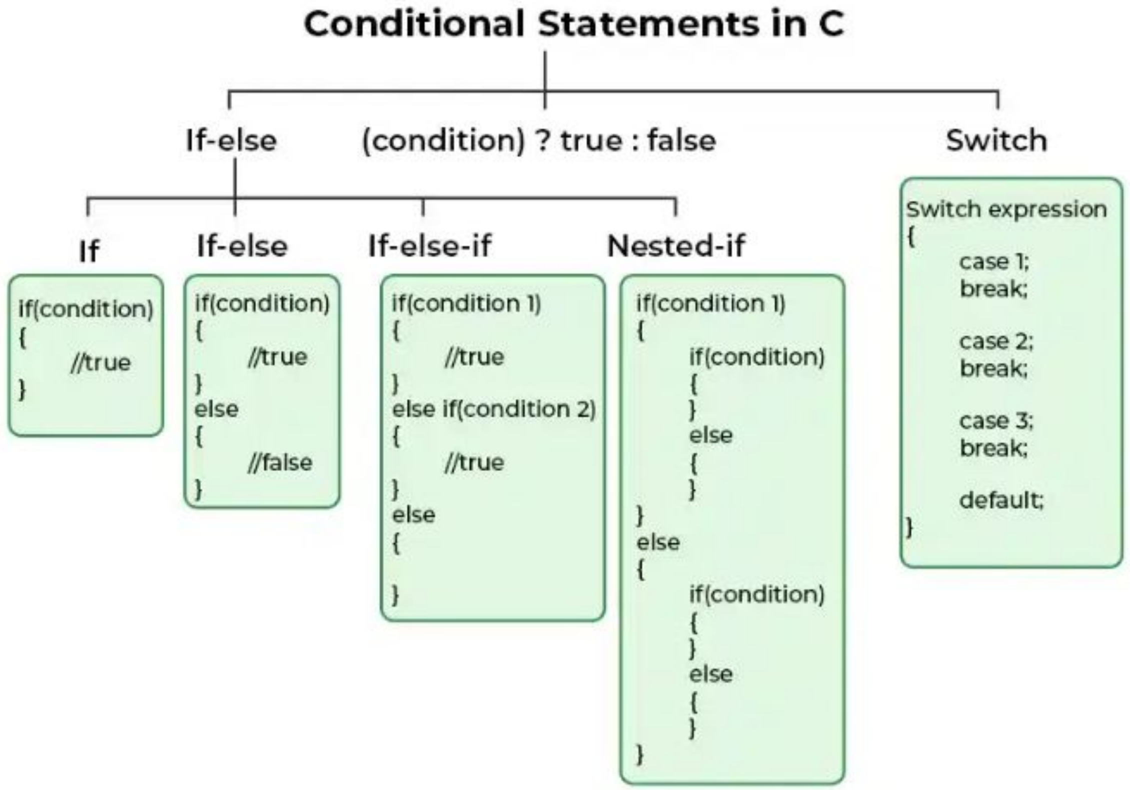
The **conditional statements** (also known as decision control structures) such as if, if else, switch, etc. are used for decision-making purposes in C programs.

They are also known as Decision-Making Statements and are used to evaluate one or more conditions and make the decision whether to execute a set of statements or not. These decision-making statements in programming languages decide the direction of the flow of program execution.

Need of Conditional Statements

There come situations in real life when we need to make some decisions and based on these decisions, we decide what should we do next. Similar situations arise in programming also where we need to make some decisions and based on these decisions we will execute the next block of code. For example, in C if x occurs then execute y else execute z. There can also be multiple conditions like in C if x occurs then execute p, else if condition y occurs execute q, else execute r. This condition of C else-if is one of the many ways of importing multiple conditions.

Types of Conditional Statements in C



```
if(condition)
{
        //true
}
```

```
if(condition)
{
        //true
}
else
{
        //false
}
```

```
if(condition 1)
{
        //true
}
else if(condition 2)
{
        //true
}
else
{

}
```

```
if(condition 1)
{
        if(condition)
        {
        }
        else
        {
        }
}
else
{
        if(condition)
        {
        }
        else
        {
        }
}
```

```
Switch expression
{
        case 1;
        break;

        case 2;
        break;

        case 3;
        break;

        default;
}
```

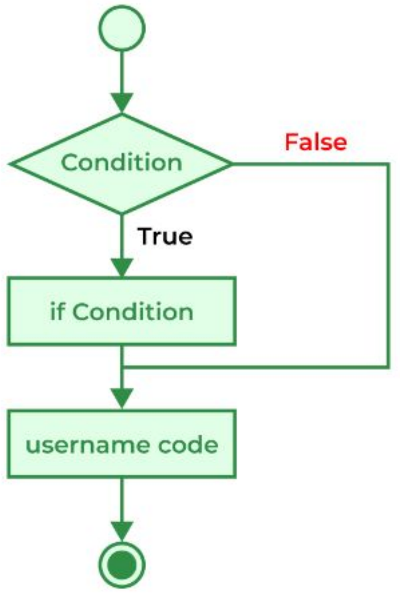
1. if in C

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
}
```

Flowchart of if Statement



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.

Example of if in C

C

```c
// 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");
}
```

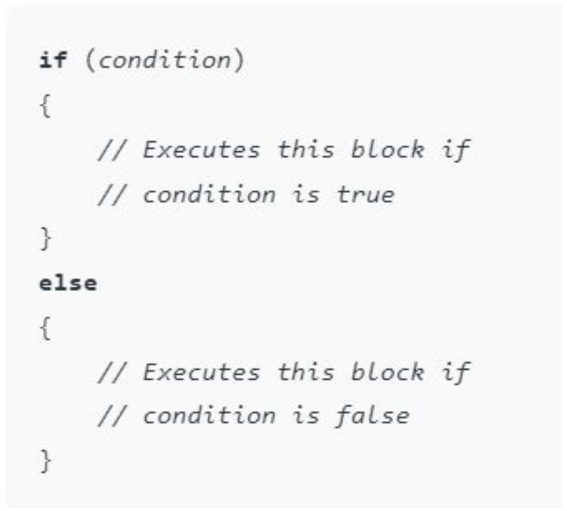
Output

```
I am Not in if
```

2. if-else in C

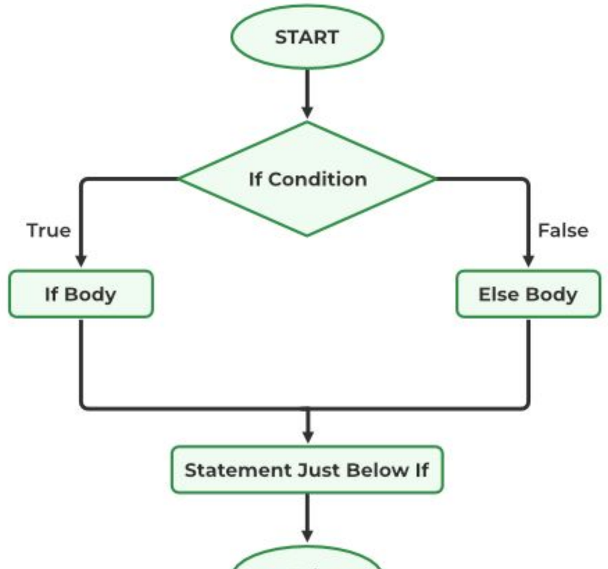
The *if* statement alone tells us that if a condition is true it will execute a block of statements and if the condition is false it won't. But what if we want to do something else when the condition is false? Here comes the C *else* statement. We can use the *else* statement with the *if* statement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

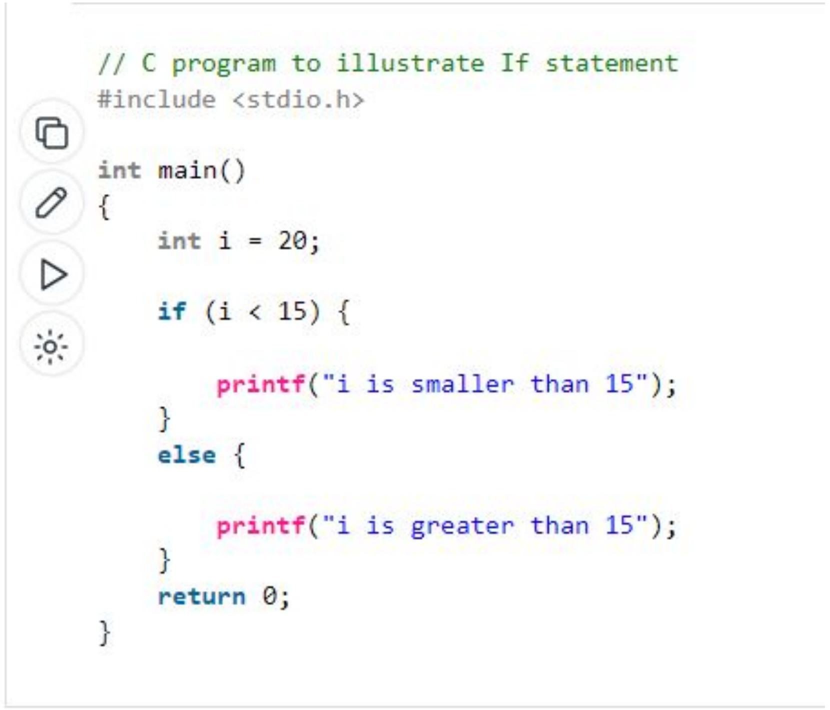
Syntax of if else in C



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

Flowchart of if-else Statement





```c
// 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;
}
```

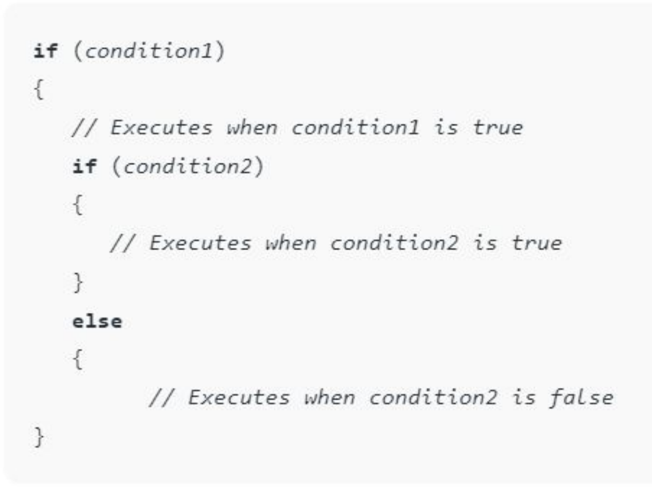
Output

```
i is greater than 15
```

3. Nested if-else in C

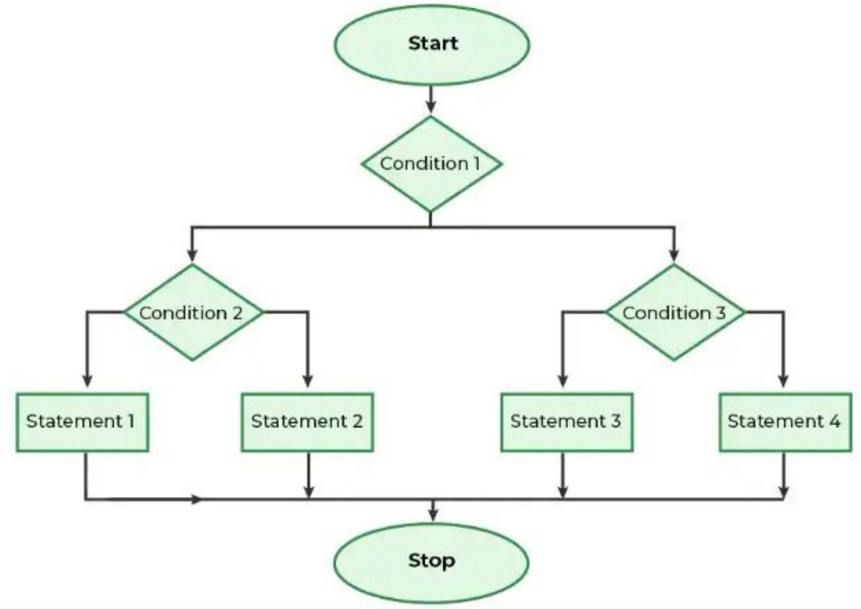
A nested if in C is an if statement that is the target of another if statement. Nested if statements mean an if statement inside another if statement. Yes, C allow us to nested if statements within if statements, i.e, we can place an if statement inside another if statement.

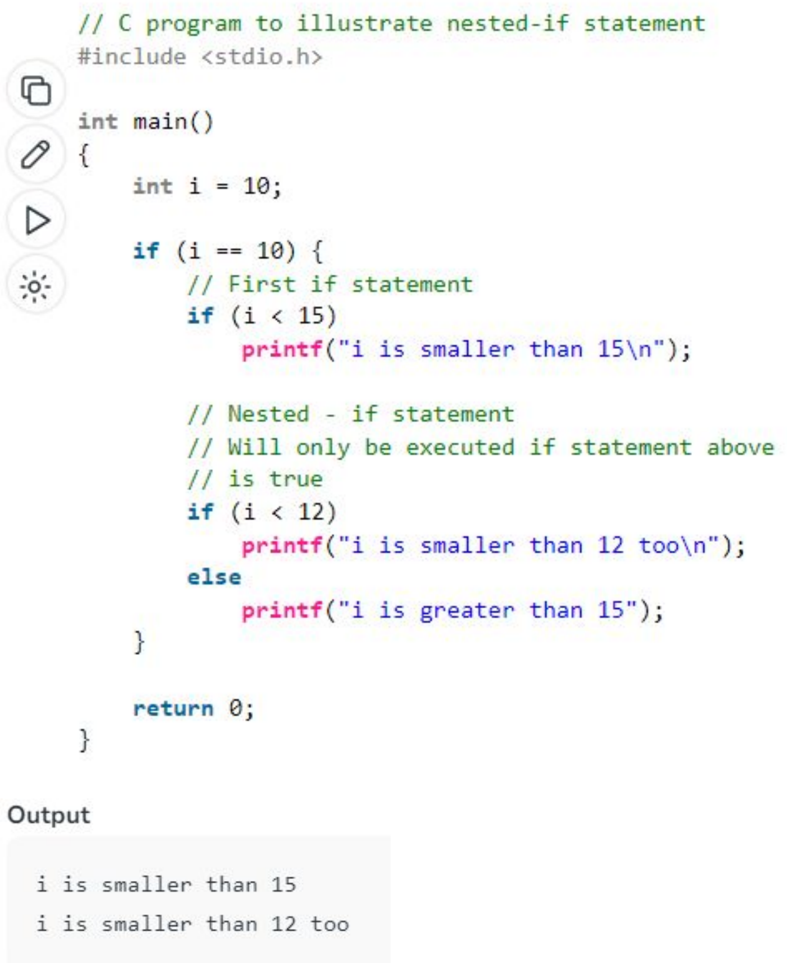
Syntax of Nested if-else



```
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
// 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
```

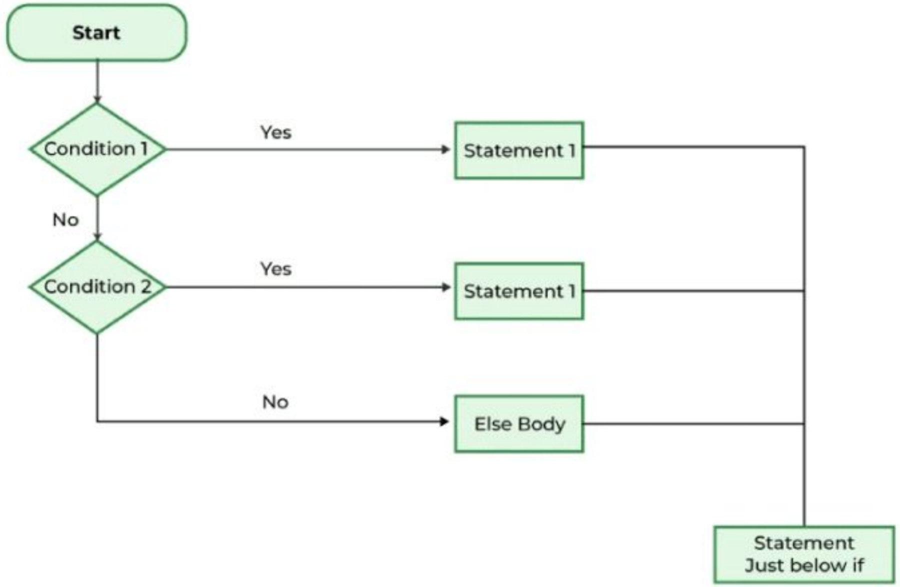
4. if-else-if Ladder in C

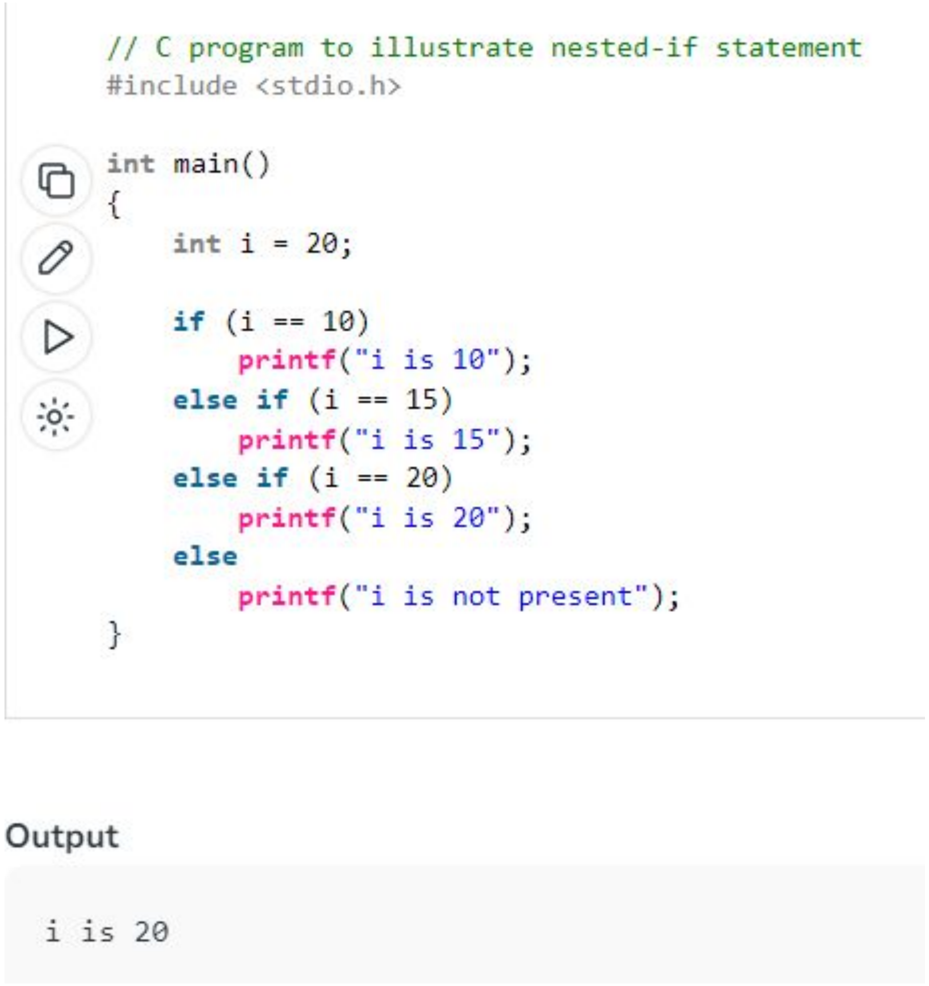
The if else if statements are used when the user has to decide among multiple options. The C if statements are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the C else-if ladder is bypassed. 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





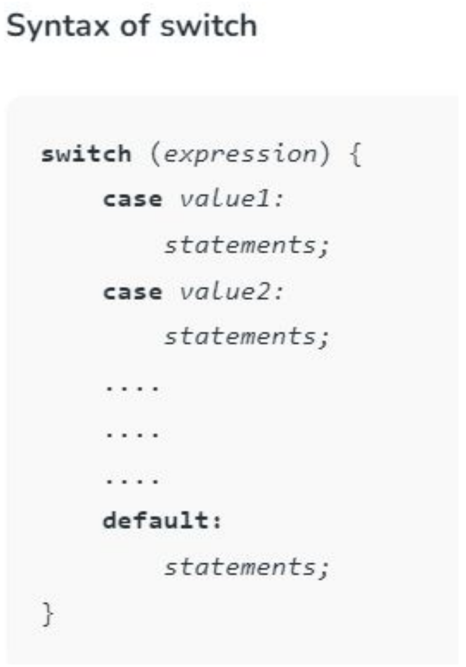
```c
// C program to illustrate nested-if statement
#include <stdio.h>

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
```

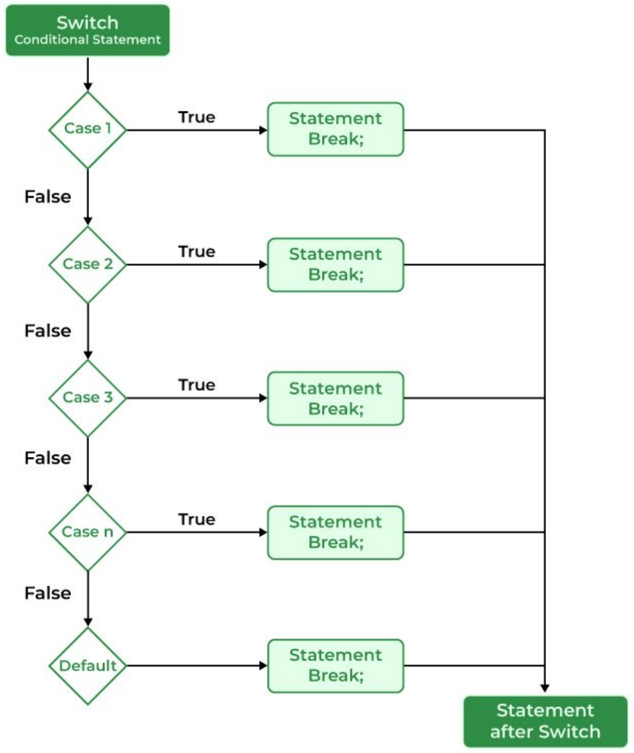
5. switch Statement in C

The switch case statement is an alternative to the if else if ladder that can be used to execute the conditional code based on the value of the variable specified in the switch statement. The switch block consists of cases to be executed based on the value of the switch variable.

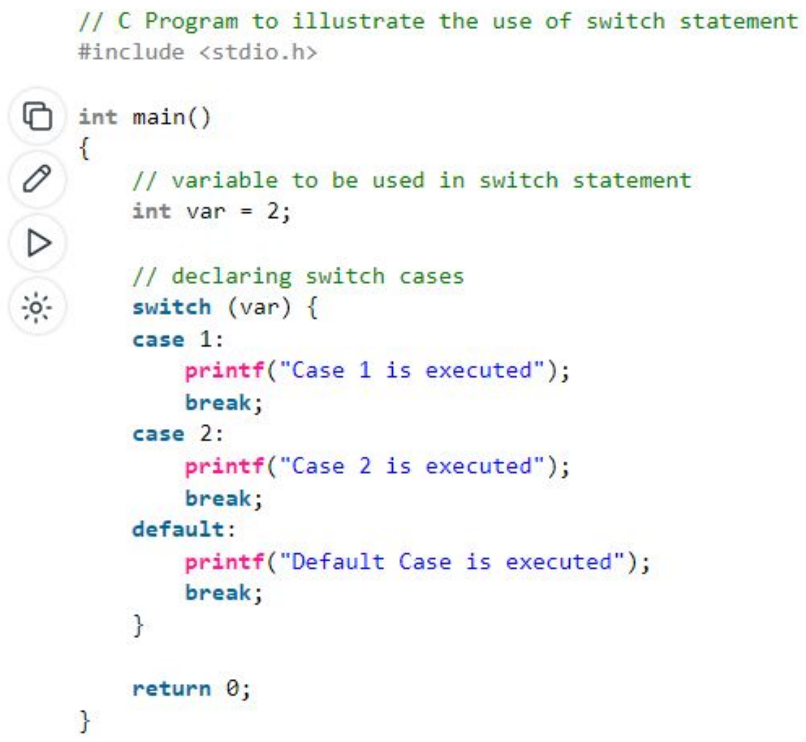


Syntax of switch

```
switch (expression) {
    case value1:
        statements;
    case value2:
        statements;
    ....
    ....
    ....
    default:
        statements;
}
```



Note: *The switch expression should evaluate to either integer or character. It cannot evaluate any other data type.*



```
// C Program to illustrate the use of switch statement
#include <stdio.h>

int main()
{
    // variable to be used in switch statement
    int var = 2;

    // declaring switch cases
    switch (var) {
    case 1:
        printf("Case 1 is executed");
        break;
    case 2:
        printf("Case 2 is executed");
        break;
    default:
        printf("Default Case is executed");
        break;
    }

    return 0;
}
```

Output

```
Case 2 is executed
```

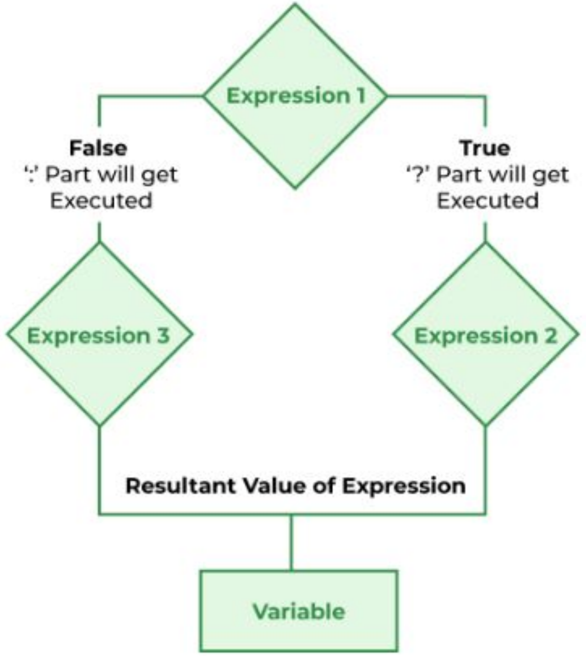
6. Conditional Operator in C

The conditional operator is used to add conditional code in our program. It is similar to the if-else statement. It is also known as the ternary operator as it works on three operands.

Syntax of Conditional Operator

```
(condition) ? [true_statements] : [false_statements];
```

Flowchart of Conditional Operator


Expression 1
False
':' Part will get Executed
True
'?' Part will get Executed
Expression 3
Expression 2
Resultant Value of Expression
Variable

```c
// C Program to illustrate the use of conditional operator
#include <stdio.h>

// driver code
int main()
{

    int var;
    int flag = 0;

    // using conditional operator to assign the value to var
    // according to the value of flag
    var = flag == 0 ? 25 : -25;
    printf("Value of var when flag is 0: %d\n", var);

    // changing the value of flag
    flag = 1;
    // again assigning the value to var using same statement
    var = flag == 0 ? 25 : -25;
    printf("Value of var when flag is NOT 0: %d", var);

    return 0;
}
```

Output

```
Value of var when flag is 0: 25
Value of var when flag is NOT 0: -25
```

Data types:

NaN:

`Not a number`: As the name implies, it is used to denote that the value of an object is not a number. There are many ways that you can generate this error, one being invalid math opertions such as 0/0 or sqrt(-1).

"not a valid number" usually after you did some operation that should produce one but couldn't.

`undefined`:

It means that the object doesn't have any value, therefore undefined.

This occurs when you create a variable and don't assign a value to it.

it *exists* but hasn't been given a value *yet*

`Null`:

It means that the object is empty and isn't pointing to any memory address.
it exists and it has deliberately been given an empty value.

Thanks!