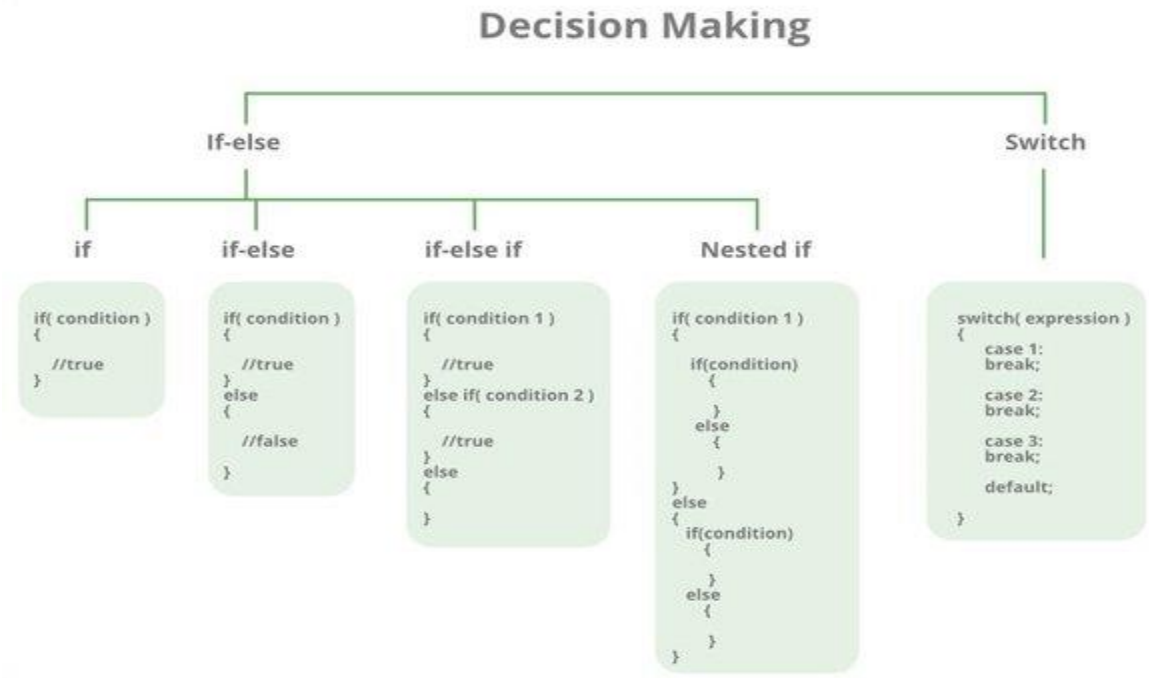
Conditional Statements/Selection Statements.

Conditional statement in C programming are used to base choices on the conditions. When there is no condition surrounding the statements, conditional statement in C are executed sequentially.

The execution flow may alter if a condition is added to a block of statements depending on the outcome of the condition's evaluation. In "C", this procedure is known as decision-making.

Conditional statements in C are programming constructs that allow a program to execute different blocks of code based on whether a certain condition is true or false.
The most common types of conditional statements in C are



if Statement
if-else Statement
Nested if Statement
if-else-if Ladder
switch Statement

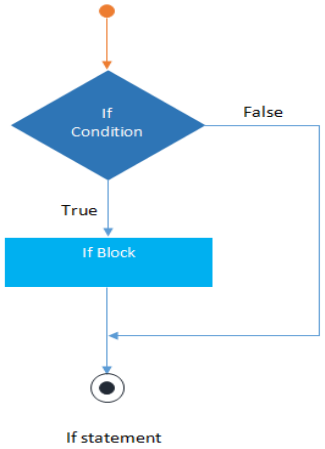
if Statement

- The **if in C** is the most simple decision-making statement.
- It consists of the test condition and if block or body.
- If the given condition is true only then the if block will be executed.
- 'if' keyword is used to execute a set of statements when the logical condition is true.

Syntax

```
if (condition)
{
  (TRUE) Statement (s);
}
```

FlowChart of Simple if



If statement

Working of 'simple if' statement

- It is basically a "Two-way" decision statement (one for TRUE and other for FALSE)
- It has only one option.
- The statement as executed only when the condition is true.
- In case the condition is false the compiler skips the lines within the "if Block".
- The condition is always enclosed within a pair of parenthesis i.e. ( ) ,
- The conditional statement should not terminated with Semi-colons (ie ;)
- The Statements following the "if"-statement are normally enclosed in Curly Braces {}.
- The Curly Braces indicates the scope of "if" statement. The default scope is one statement. But it is good practice to use curly braces even with a single statement.
- The statement block may be a single statement or a group of statements.
- If the Test Expression / Conditions are TRUE, the Statement Block will be executed and executes rest of the program.
- If the Test Expression / Condition are FALSE, the Statement Block will be skipped and rest of the program executes next.

Example 1

```
#include<stdio.h>
#include<conio.h>
void main()
{
int num=0;
printf("enter the number");
scanf("%d",&num);
if(n%2==0)
{
printf("%d number in even",num);
}
getch();
}
```

Example 3

multiple if statements

```
#include <stdio.h>
int main()
{
   int x, y;
   printf("enter the value of x:");
   scanf("%d", &x);
   printf("enter the value of y:");
   scanf("%d", &y);
   if (x>y)
   {
        printf("x is greater than y\n");
   }
   if (x<y)
   {
        printf("x is less than y\n");
   }
   if (x==y)
   {
        printf("x is equal to y\n");
   }
   printf("End of Program");
   return 0;
}
```

Example 2

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

int main()
{
        int i = 10;

        if (i > 15) {
                printf("10 is greater than
15");
        }
Return 0;
        }
```

Example 4

```
#include<stdio.h>
void main()
 {
int num;
printf("Hello user, Enter a number");
 scanf("%d",&num);
 // Collects the number from user
if(num==1)
{
        printf("UNITED STATES");
}
if(num==2)
{
        printf("SPAIN");
 }
if(num==3)
 {
        printf("INDIA");
 }
return 0;
}
```

“if-else” Statement :

The if-else statement is used to perform two operations for a single condition. The if-else statement is an extension to the if statement using which, we can perform two different operations, i.e., one is for the correctness of that condition, and the other is for the incorrectness of the condition. Here, we must notice that if and else block cannot be executed simiulteneously. Using if-else statement is always preferable since it always invokes an otherwise case with every if condition.

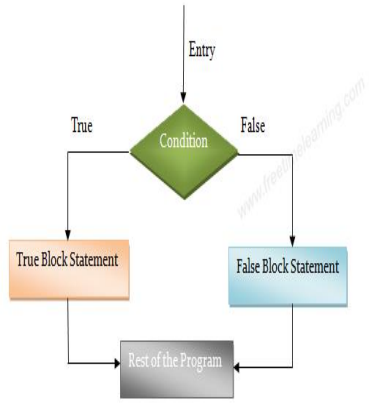
It is observed that the “if” statement executes only when the condition following if is true. It does nothing when the condition is false.

In if-else either True-Block or False – Block will be executed and not both. The “else” Statement cannot be used without “if”.

Syntax :

```
if ( Test Expression or Condition )
     {
      Statements;  /*true block (or) if block */
      }
     else
      {
      Statements;   /* false block (or) else
block */
      }
```

FlowChart



If the test expression is evaluated to true,

- statements inside the body of if are executed.
- statements inside the body of else are skipped from execution.

If the test expression is evaluated to false,

- statements inside the body of else are executed
- statements inside the body of if are skipped from execution.

Examples:

1.Program: Write a program to print the given number is even or odd.

```
#include <stdio.h>
int main() {
   int number;
```

```
    printf("Enter an integer: ");
    scanf("%d", &number);
    if  (number%2 == 0) {
        printf("%d is an even integer.",number);
    }
    else {
        printf("%d is an odd integer.",number);
    }
    return 0;
}
```

Program to check whether a person is eligible to vote or not

```
1.  #include <stdio.h>
2.  int main()
3.  {
4.      int age;
5.      printf("Enter your age?");
6.      scanf("%d",&age);
7.      if(age>=18)
8.      {
9.          printf("You are eligible to vote...");
10.     }
11.     else
12.     {
13.         printf("Sorry ... you can't vote");
14.     }
15.}
```

Output

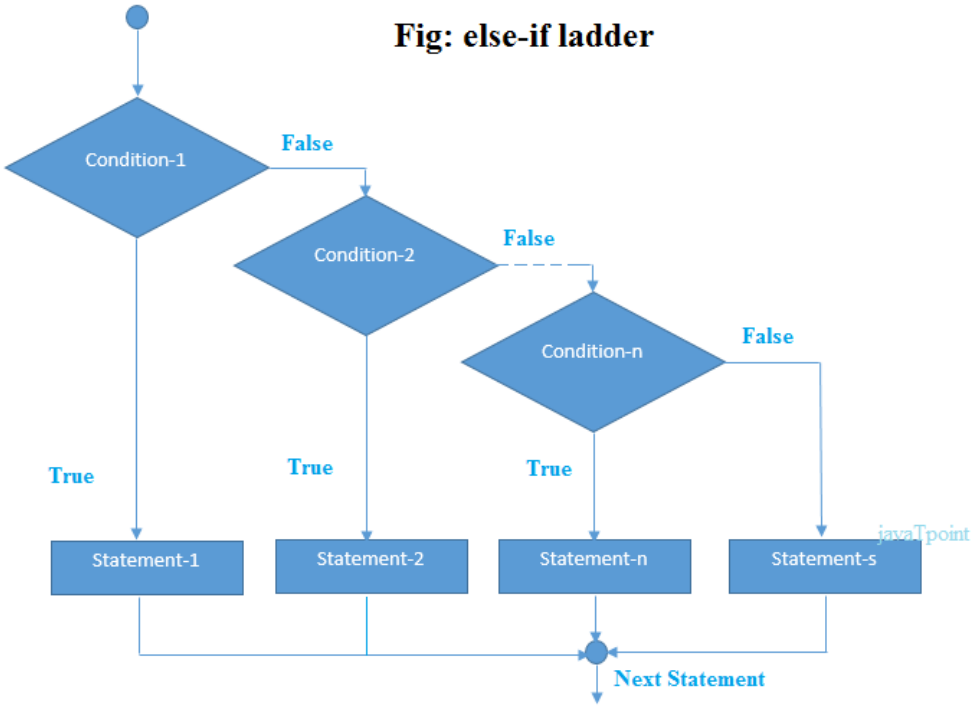
```
Enter your age?18
You are eligible to vote...
Enter your age?13
Sorry ... you can't vote
```

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(condition1){
1.  //code to be executed if condition1 is true
2.  }else if(condition2){
3.  //code to be executed if condition2 is true
4.  }
5.  else if(condition3){
6.  //code to be executed if condition3 is true
7.  }
8.  ...
9.  else{
10.//code to be executed if all the conditions are false
11.}
```

Flowchart of else-if ladder statement in C



The example of an if-else-if statement in C language is given below.

```
1. #include<stdio.h>
2. int main(){
3. int number=0;
4. printf("enter a number:");
5. scanf("%d",&number);
6. if(number==10){
7. printf("number is equals to 10");
8. }
9. else if(number==50){
10.printf("number is equal to 50");
11.}
12.else if(number==100){
13.printf("number is equal to 100");
14.}
15.else{
16.printf("number is not equal to 10, 50 or 100");
17.}
18.return 0;
19.}
```

Output

```
enter a number:4
number is not equal to 10, 50 or 100
enter a number:50
number is equal to 50
```

Program to calculate the grade of the student according to the specified marks.

```
1. #include <stdio.h>
2. int main()
3. {
4.     int marks;
5.     printf("Enter your marks?");
6.     scanf("%d",&marks);
7.     if(marks > 85 && marks <= 100)
8.     {
9.         printf("Congrats ! you scored grade A ...");
10.    }
11.    else if (marks > 60 && marks <= 85)
12.    {
13.        printf("You scored grade B + ...");
14.    }
15.    else if (marks > 40 && marks <= 60)
16.    {
17.        printf("You scored grade B ...");
18.    }
19.    else if (marks > 30 && marks <= 40)
20.    {
21.        printf("You scored grade C ...");
22.    }
23.    else
24.    {
25.        printf("Sorry you are fail ...");
26.    }
27.}
```

Output

```
Enter your marks?10
Sorry you are fail ...
Enter your marks?40
You scored grade C ...
```

```
Enter your marks?90
Congrats ! you scored grade A ...
```

Nested if else statement in C

One of the fundamental constructs in programming is ***conditional statements***. They allow a program to take different paths based on the values of certain conditions. In C, conditional statements are implemented using ***if-else statements***. In more complex scenarios, ***nested if-else statements*** can be used to make more sophisticated decisions. This blog post will provide an in-depth explanation of nested if-else statements in C, including syntax, example, and output.

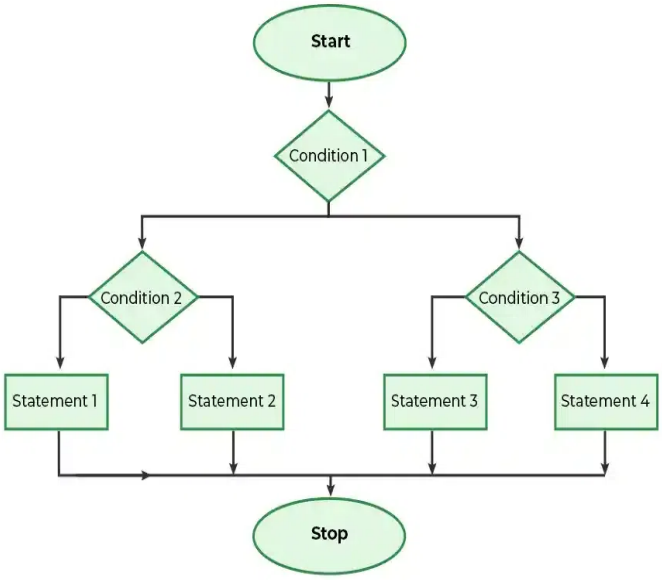
Syntax:

A ***nested if-else statement*** is an ***if statement*** inside another ***if statement***. The general syntax of nested if-else statement in C is as follows:

```
1. if (condition1) {
2.     /* code to be executed if condition1 is true */
3.     if (condition2) {
4.         /* code to be executed if condition2 is true */
5.     } else {
6.         /* code to be executed if condition2 is false */
7.     }
8. } else {
9.     /* code to be executed if condition1 is false */
10.}
```

As you can see, the outer ***if statement*** has two possible paths: one for when the condition is ***true***, and another for when the condition is ***false***. If the condition is true, the program will execute the code inside the block associated with the outer if statement. However, if the condition is false, the program will skip over that block and move to the else block. Within the outer if block, there is another if statement, which can also have two possible paths depending on whether the condition is true or false.

FlowChart for Nested if-else



Example: program that takes in a number and checks whether it is ***positive, negative***, or ***zero***.

```
1.  #include <stdio.h>
2.
3.  int main() {
4.      int num;
5.
6.  printf("Enter a number: ");
7.  scanf("%d", &num);
8.
9.      if (num > 0) {
10.printf("%d is positive.\n", num);
11.   } else {
12.       if (num < 0) {
13.printf("%d is negative.\n", num);
14.       } else {
15.printf("%d is zero.\n", num);
16.       }
17.   }
18.
19.   return 0;
20.}
```

Output:

Let's run the above program with some sample inputs and see the output.

```
Enter a number: 10
10 is positive.

Enter a number: -5
-5 is negative.

Enter a number: 0
0 is zero.
```

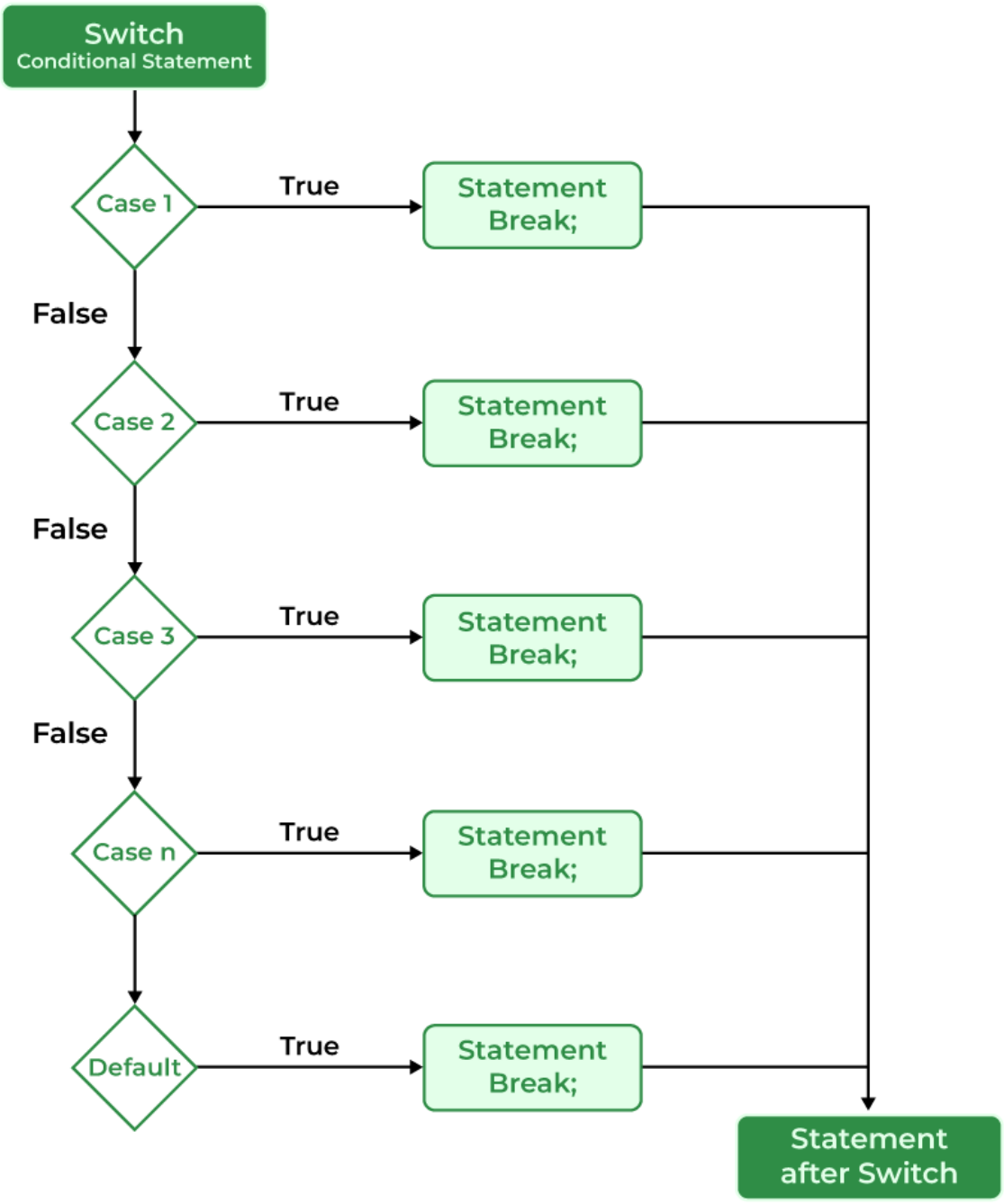
C Switch Statement

The switch statement in C is an alternate to if-else-if ladder statement which allows us to execute multiple operations for the different possibles values of a single variable called switch variable. Here, We can define various statements in the multiple cases for the different values of a single variable.

The syntax of switch statement in c language is given below:

```
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.*



Rules for switch statement in C language

The *switch expression* must be of an integer or character type.

2) The *case value* must be an integer or character constant.

3) The *case value* can be used only inside the switch statement.

The break statement breaks out of the switch block and stops the execution

The default statement is optional, and specifies some code to run if there is no case match

4) The *break statement* in switch case is not must. It is optional. If there is no break statement found in the case, all the cases will be executed present after the matched case. It is known as *fall through* the state of C switch statement.

example of c language switch statement.

```
1. #include<stdio.h>
2. int main(){
3. int number=0;
4. printf("enter a number:");
5. scanf("%d",&number);
6. switch(number){
7. case 10:
8. printf("number is equals to 10");
9. break;
10.case 50:
11.printf("number is equal to 50");
12.break;
13.case 100:
14.printf("number is equal to 100");
15.break;
16.default:
17.printf("number is not equal to 10, 50 or 100");
18.}
19.return 0;
20.}
```

Output

```
enter a number:4
number is not equal to 10, 50 or 100
enter a number:50
number is equal to 50
```

Switch case example 2

```
#include <stdio.h>
int main() {
  int digit;
  printf("Enter digit:");
  scanf("%d", &digit); //Input digit
```

```
  switch (digit) { //Writing a case for every digit in 0-9

  case 1:
    printf("One\n");
    break;
  case 2:
    printf("Two\n");
    break;
  case 3:
    printf("Three\n");
    break;
  case 4:
    printf("Four\n");
    break;
  case 5:
    printf("Five\n");
    break;
  case 6:
    printf("Six\n");
    break;
  case 7:
    printf("Seven\n");
    break;
  case 8:
    printf("Eight\n");
    break;
  case 9:
    printf("Nine\n");
    break;
  case 0:
    printf("Zero\n");
    break;
  default:
    printf("Invalid Digit\n");
  }

  return 0;
}
```