C Programming Language

Compile time vs Runtime

- **Compile time vs Runtime**
- Compile-time and Runtime are the two programming terms used in the software development.
- Compile-time is the time at which the source code is converted into an executable code while the run time is the time at which the executable code is started running. Both the compile-time and runtime refer to different types of error.
- **Compile-time errors**
- Compile-time errors are the errors that occurred when we write the wrong syntax. If we write the wrong syntax or semantics of any programming language, then the compile-time errors will be thrown by the compiler.

- The compiler will not allow to run the program until all the errors are removed from the program.
- When all the errors are removed from the program, then the compiler will generate the executable file.
- The compile-time errors can be:
- **Syntax errors**
- **Semantic errors**

- **Syntax errors**
- When the programmer does not follow the syntax of any programming language, then the compiler will throw the syntax error.
- **For example,**
- int a, b:
- The above declaration generates the compile-time error as in C, every statement ends with the semicolon, but we put a colon (:) at the end of the statement.

- **Semantic errors**
- The semantic errors exist when the statements are not meaningful to the compiler.
- For example,
- a+b=c;
- The above statement throws a compile-time errors.
- In the above statement, we are assigning the value of 'c' to the summation of 'a' and 'b' which is not possible in C programming language as it can contain only one variable on the left of the assignment operator while right of the assignment operator can contain more than one variable.

- The above statement can be re-written as:

```
c = a+b;
```

- **Runtime errors**
- The runtime errors are the errors that occur during the execution and after compilation.
- The examples of runtime errors are division by zero, etc. These errors are not easy to detect as the compiler does not point to these errors.

- Let's look at the differences between compile-time and runtime:

Compile-time Error	Runtime Error
1. The compile-time errors are the errors which are produced at the compile-time, and they are detected by the compiler.	1. The runtime errors are the errors which are not generated by the compiler and produce an unpredictable result at the execution time.
2. In this case, the compiler prevents the code from execution if it detects an error in the program.	In this case, the compiler does not detect the error, so it cannot prevent the code from the execution.
It contains the syntax and semantic errors such as missing semicolon at the end of the statement.	It contains the errors such as division by zero, determining the square root of a negative number.

- **Example of Compile-time error**

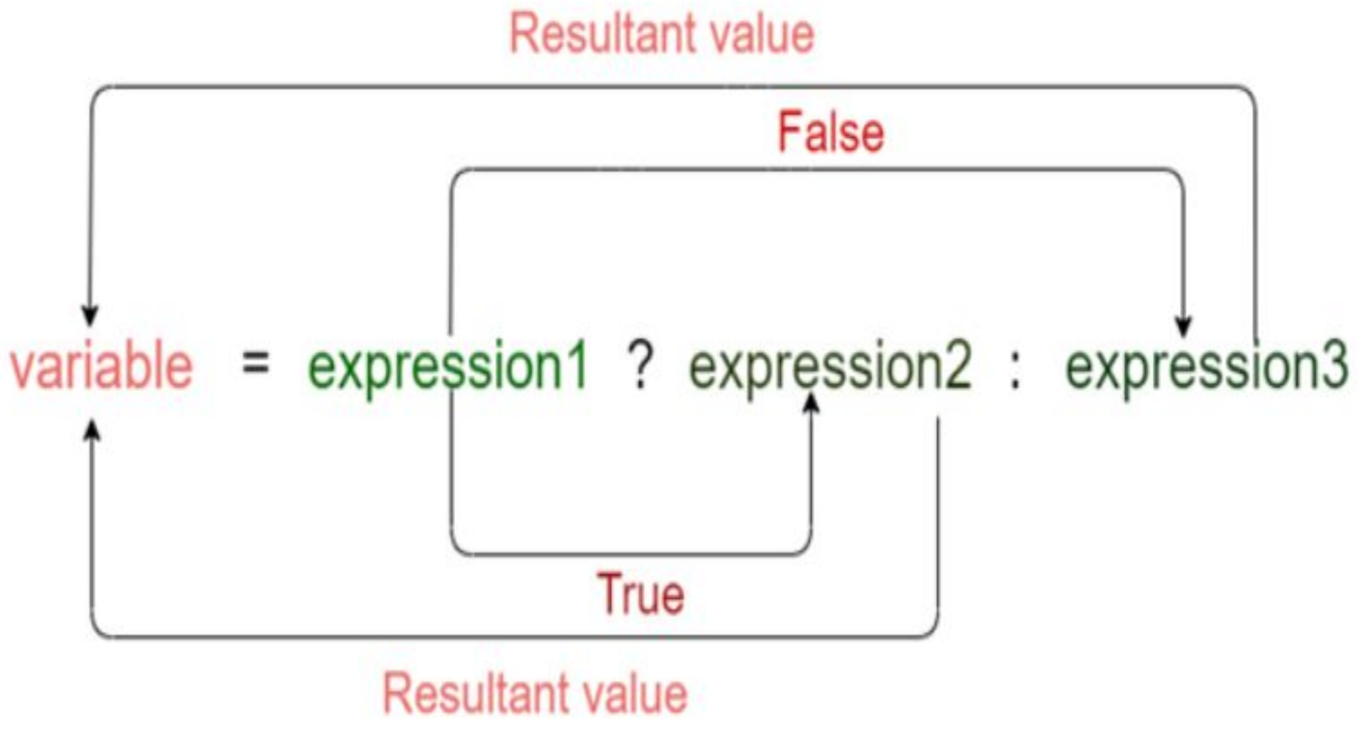
```
#include <stdio.h>
int main()
{
int a=20;
printf("The value of a is : %d",a):
 return 0;
}
```

- **Example of runtime error**

```
#include <stdio.h>
int main()
{
 int a=20;
 int b=a/0; // division by zero
printf("The value of b is : %d",b):
 return 0;
}
```

- **Conditional Operator in C**
- The conditional operator is also known as a ternary operator.
- The conditional statements are the decision-making statements which depends upon the output of the expression. It is represented by two symbols, i.e., '?' and ':'.
- As conditional operator works on three operands, so it is also known as the ternary operator.
- The behavior of the conditional operator is similar to the 'if-else
- ' statement as 'if-else' statement is also a decision-making statement.
- **Syntax of a conditional operator**
- Expression1? expression2: expression3;

- The pictorial representation of the above syntax is shown below:



- **Meaning of the above syntax.**
- In the above syntax, the expression1 is a Boolean condition that can be either true or false value.
- If the expression1 results into a true value, then the expression2 will execute.
- The expression2 is said to be true only when it returns a non-zero value.
- If the expression1 returns false value then the expression3 will execute.
- The expression3 is said to be false only when it returns zero value.

- Let's understand the ternary or conditional operator through an example.

```
#include <stdio.h>
int main()
{
int age;  // variable declaration
printf("Enter your age");
scanf("%d",&age);   // taking user input for age variable
(age>=18)? (printf("eligible for voting")) : (printf("not eligible for voting"));   //
conditional operator
return 0;  }
```

- As we know that the behavior of conditional operator and 'if-else' is similar but they have some differences. Let's look at their differences.
- A conditional operator is a single programming statement, while the 'if-else' statement is a programming block in which statements come under the parenthesis.
- A conditional operator can also be used for assigning a value to the variable, whereas the 'if-else' statement cannot be used for the assignment purpose.
- It is not useful for executing the statements when the statements are multiple, whereas the 'if-else' statement proves more suitable when executing multiple statements.
- The nested ternary operator is more complex and cannot be easily debugged, while the nested 'if-else' statement is easy to read and maintain.

- **Bitwise Operator in C**
- The bitwise operators are the operators used to perform the operations on the data at the bit-level.
- When we perform the bitwise operations, then it is also known as bit-level programming.
- It consists of two digits, either 0 or 1. It is mainly used in numerical computations to make the calculations faster.
- We have different types of bitwise operators in the C programming language. The following is the list of the bitwise operators:

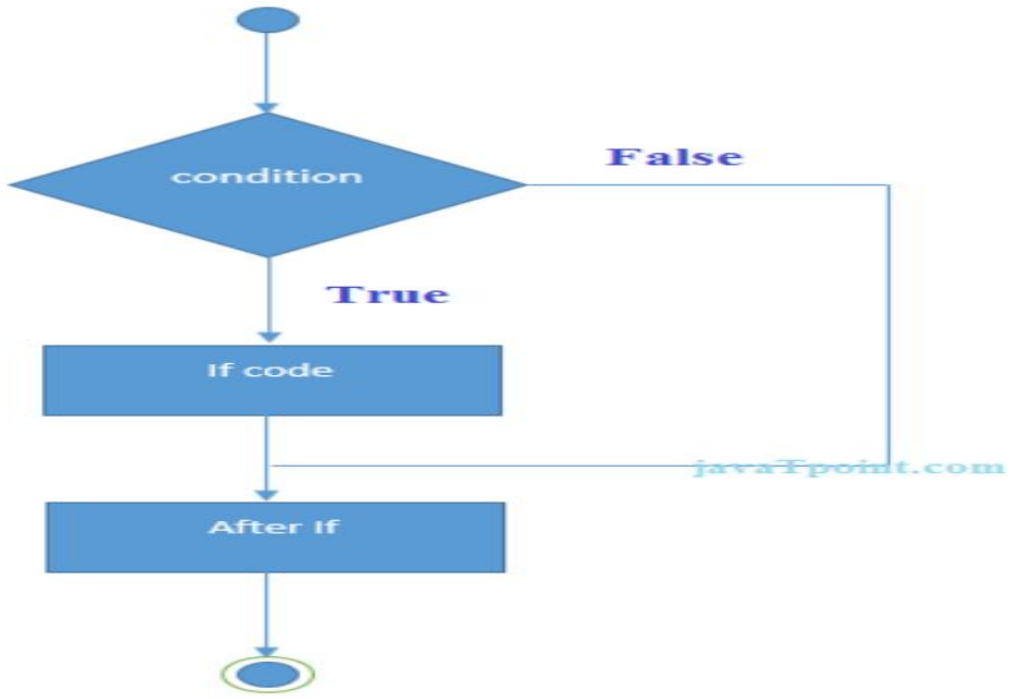
Operator	Meaning of operator
&	Bitwise AND operator
|	Bitwise OR operator
^	Bitwise exclusive OR operator
~	One's complement operator (unary operator)
<<	Left shift operator
>>	Right shift operator

- **C if else Statement**
- The if-else statement in C is used to perform the operations based on some specific condition. The operations specified in if block are executed if and only if the given condition is true.
- There are the following variants of if statement in C language.
- If statement
- If-else statement
- If else-if ladder
- Nested if

- If Statement
- The if statement is used to check some given condition and perform some operations depending upon the correctness of that condition.
- It is mostly used in the scenario where we need to perform the different operations for the different conditions. The syntax of the if statement is given below.

```
if(expression){
//code to be executed
}
```

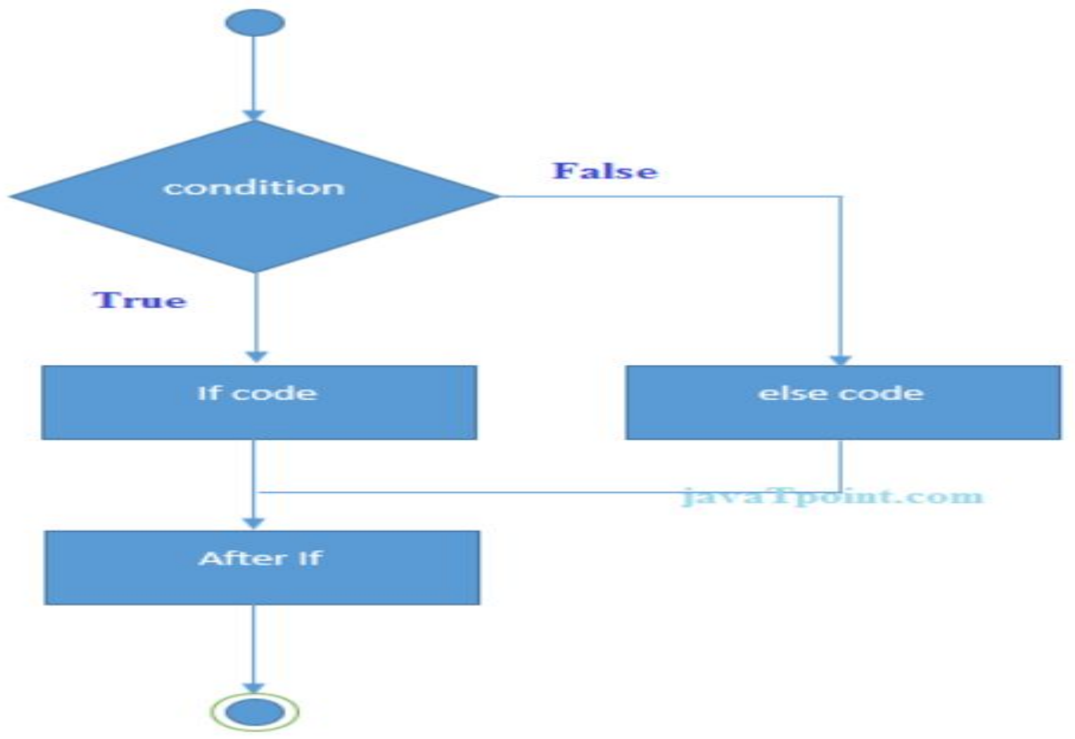
- **Flowchart of if statement in C**



- **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 simultaneously. Using if-else statement is always preferable since it always invokes an otherwise case with every if condition.
- The syntax of the if-else statement is given below.

```
if(expression){
//code to be executed if condition is true
} else{
//code to be executed if condition is false
}
```

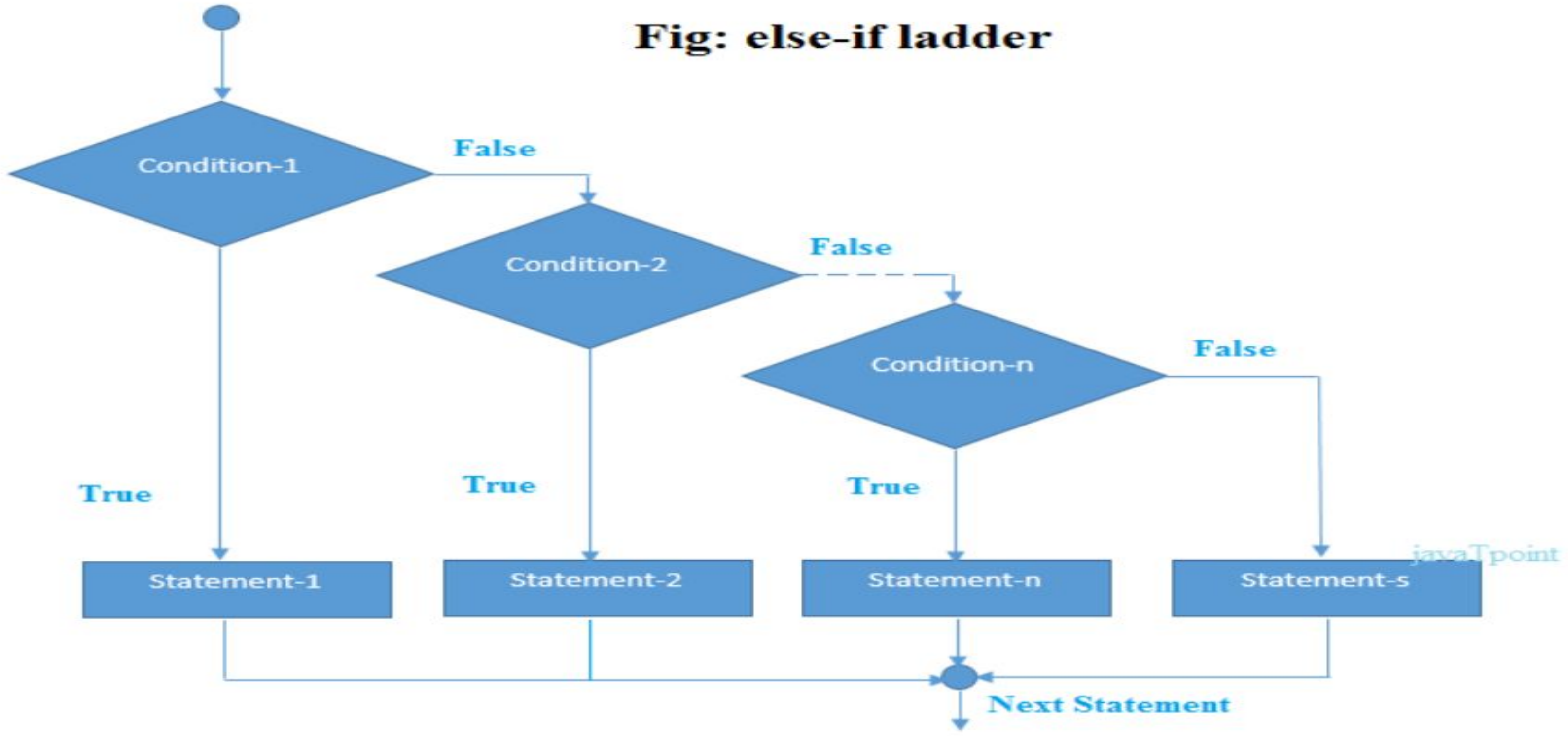
- **Flowchart of the if-else statement in C**



- **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){
//code to be executed if condition1 is true
}else if(condition2){
//code to be executed if condition2 is true
}
else if(condition3){
//code to be executed if condition3 is true
}
...
else{
//code to be executed if all the conditions are false
}
```

- Flowchart of else-if ladder statement in C



- **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:
//code to be executed;
 break;  //optional
case value2:
 //code to be executed;
 break;  //optional
......

default:
code to be executed if all cases are not matched;
}
```

- **Rules for switch statement in C language**

1) 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.

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.

Let's see a simple example of c language switch statement.

```
#include<stdio.h>
int main(){
int number=0;
printf("enter a number:");
scanf("%d",&number);
switch(number){
case 10:
printf("number is equals to 10");
break;
case 50:
printf("number is equal to 50");
break;
case 100:
printf("number is equal to 100");
break;
default:
printf("number is not equal to 10, 50 or 100");
}
return 0;
}
```