OPERATORS AND ITS TYPE

Arithmetic Operators

Computer programs are widely used for mathematical calculations. We can write a computer program which can do simple calculation like adding two numbers (2 + 3) and we can also write a program, which can solve a complex equation like $P(x) = x^4 + 7x^3 - 5x + 9$. If you have been even a poor student, you must be aware that in first expression 2 and 3 are operands and + is an operator. Similar concepts exist in Computer Programming.

Take a look at the following two examples –

2 + 3

$P(x) = x^4 + 7x^3 - 5x + 9$.

These two statements are called arithmetic expressions in a programming language and **plus**, **minus** used in these expressions are called arithmetic operators and the values used in these expressions like 2, 3 and x, etc., are called operands. In their simplest form, such expressions produce numerical results.

Similarly, a programming language provides various arithmetic operators. The following table lists down a few of the important arithmetic operators available in C programming language. Assume variable A holds 10 and variable B holds 20, then –

Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiplies both operands	A * B will give 200
/	Divides numerator by de-numerator	B / A will give 2
%	This gives remainder of an integer division	B % A will give 0

Following is a simple example of C Programming to understand the above mathematical operators –

```
#include <stdio.h>
```

```
int main() {
  int a, b, c;

  a = 10;
  b = 20;

  c = a + b;
  printf( "Value of c = %d\n", c);

  c = a - b;
  printf( "Value of c = %d\n", c);

  c = a * b;
  printf( "Value of c = %d\n", c);

  c = b / a;
  printf( "Value of c = %d\n", c);

  c = b % a;
  printf( "Value of c = %d\n", c);
}
```

When the above program is executed, it produces the following result −

```
Value of c = 30
Value of c = -10
Value of c = 200
Value of c = 2
Value of c = 0
```

Relational Operators

Consider a situation where we create two variables and assign them some values as follows −

```
A = 20
B = 10
```

Here, it is obvious that variable A is greater than B in values. So, we need the help of some symbols to write such expressions which are called relational expressions. If we use C programming language, then it will be written as follows −

```
(A > B)
```

Here, we used a symbol > and it is called a relational operator and in their simplest form, they produce Boolean results which means the result will be either true or false. Similarly, a programming language provides various relational operators. The following table lists down a few of the important relational operators available in C programming language. Assume variable **A** holds 10 and variable **B** holds 20, then −

Operator	Description	Example
==	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Here, we will show you one example of C Programming which makes use of **if conditional statement**. Though this statement will be discussed later in a separate chapter, but in short, we use **if statement** to check a condition and if the condition is true, then the body of **if statement** is executed, otherwise the body of **if statement** is skipped.

```
#include <stdio.h>

int main() {
  int a, b;

  a = 10;
  b = 20;

  /* Here we check whether a is equal to 10 or not */
  if( a == 10 ) {

    /* if a is equal to 10 then this body will be executed */
    printf( "a is equal to 10\n");
  }

  /* Here we check whether b is equal to 10 or not */
  if( b == 10 ) {

    /* if b is equal to 10 then this body will be executed */
    printf( "b is equal to 10\n");
  }

  /* Here we check if a is less b than or not */
  if( a < b ) {

    /* if a is less than b then this body will be executed */
    printf( "a is less than b\n");
  }

  /* Here we check whether a and b are not equal */
  if( a != b ) {

    /* if a is not equal to b then this body will be executed */
    printf( "a is not equal to b\n");
  }
}
```

When the above program is executed, it produces the following result −

```
a is equal to 10
a is less than b
a is not equal to b
```

Logical Operators

Logical operators are very important in any programming language and they help us take decisions based on certain conditions. Suppose we want to combine the result of two conditions, then logical AND and OR logical operators help us in producing the final result.

The following table shows all the logical operators supported by the C language. Assume variable **A** holds 1 and variable **B** holds 0, then −

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then condition becomes true.	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands is non-zero, then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is true.

Try the following example to understand all the logical operators available in C programming language −

```
#include <stdio.h>

int main() {
  int a = 1;
  int b = 0;

  if ( a && b ) {

    printf("This will never print because condition is false\n" );
  }
  if ( a || b ) {

    printf("This will be printed print because condition is true\n" );
  }
  if ( !(a && b) ) {

    printf("This will be printed print because condition is true\n" );
  }
}
```

When you compile and execute the above program, it produces the following result −

```
This will be printed print because condition is true
This will be printed print because condition is true
```

Operators in Java

Following is the equivalent program written in Java. C programming and Java provide almost identical set of operators and conditional statements. This program will create two variables **a** and **b**, very similar to C programming, then we assign 10 and 20 in these variables and finally, we will use different arithmetic and relational operators −

You can try to execute the following program to see the output, which must be identical to the result generated by the above example.

```
public class DemoJava {
  public static void main(String []args) {
    int a, b, c;

    a = 10;
    b = 20;

    c = a + b;
    System.out.println("Value of c = " + c );

    c = a - b;
    System.out.println("Value of c = " + c );

    c = a * b;
    System.out.println("Value of c = " + c );

    c = b / a;
    System.out.println("Value of c = " + c );

    c = b % a;
    System.out.println("Value of c = " + c );

    if( a == 10 ) {

      System.out.println("a is equal to 10" );
    }
  }
}
```

When the above program is executed, it produces the following result −

```
Value of c = 30
Value of c = -10
Value of c = 200
Value of c = 2
Value of c = 0
a is equal to 10
```

Operators in Python

Following is the equivalent program written in Python. This program will create two variables **a** and **b** and at the same time, assign 10 and 20 in those variables. Fortunately, C programming and Python programming languages provide almost identical set of operators. This program will create two variables **a** and **b**, very similar to C programming, then we assign 10 and 20 in these variables and finally, we will use different arithmetic and relational operators.

You can try to execute the following program to see the output, which must be identical to the result generated by the above example.

```
a = 10
b = 20

c = a + b
print "Value of c = ", c

c = a - b
print "Value of c = ", c

c = a * b
print "Value of c = ", c

c = a / b
print "Value of c = ", c

c = a % b
print "Value of c = ", c

if( a == 10 ):
  print "a is equal to 10"
```

When the above program is executed, it produces the following result −

```
Value of c = 30
Value of c = -10
Value of c = 200
Value of c = 0
Value of c = 10
a is equal to 10
```