**2. Operators and Expressions**

**What is an Operator and what are the types of Operators**

An operator is a symbol that tells the compiler to perform a specific mathematical or logical function. In other words an operator is a symbol which operates on a variable or value. C is rich in built-in operator it provides various types of operators. They are as follows:

1. Arithmetic Operator
2. Relational Operator
3. Logical Operator
4. Assignment Operator
5. Increment and Decrement Operator
6. Conditional Operator
7. Bitwise Operators
8. Special Operators

**Arithmetic Operators**

Arithmetic Operators as the name suggests perform arithmetic operations on the operands. In other words we can say that these operators perform mathematical operations on the operations. The arithmetic Operators are as follows:

|  |  |  |
| --- | --- | --- |
| **Arithmetic Operator** | **Meaning** | **Example** |
| + | Unary Plus or Addition | X+Y, 1+2, A+B |
| - | Unary Minus or Subtraction | X-Y, A-B, 5-5 |
| \* | Multiplication | X\*Y, 3\*4 |
| / | Division | X/Y ,1/2 |
| % | Modulo Division | 10%2, 5%3 |

**Note:**

Normal Division gives the ***quotient*** where as Modulo Division gives the ***remainder***.

**Relational Operators**

Relational Operators as the name suggests is used to find the relation between two or more variables. We often take decisions based on relations. For example if age ***is greater than or equal*** to 18 the candidate is eligible to vote. Some of the relational operators are as follows:

|  |  |
| --- | --- |
| **Relational Operator** | **Meaning** |
| < | Is less than |
| > | Is greater than |
| <= | Is less than or equal to |
| >= | Is greater than or equal to |
| == | Is equal to |
| != | Is not equal to |

**Logical Operators**

Logical operators as the name suggests are operators which connect 2 or more expressions in a logical manner. For example if the marks of the student is above 80 ***and*** less than equal to 100 then the student has got distinction. Some of the logical operators are as follows:

|  |  |
| --- | --- |
| **Logical Operator** | **Meaning** |
| && | AND Operator |
| || | OR Operator |
| ! | NOT Operator |

**&& Operator (AND)**

The AND Operator is an operator which takes both statements into consideration that means if both the statement is TRUE then the output is True. Else the output is false. The truth table is as follows:

|  |  |  |
| --- | --- | --- |
| **Statement 1** | **Statement 2** | **Output** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**|| Operator (OR)**

The OR Operator is an operator which considers either the first statement or the second statement that means only if both statements are FALSE the output will be FALSE. The truth table is as follows:

|  |  |  |
| --- | --- | --- |
| **Statement 1** | **Statement 2** | **Output** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**! Operator (NOT)**

The NOT operator is an operator which gives the compliment of a statement or variable that means if the value of the variable is TRUE then output is FALSE. The truth table of the NOT operator is as follows:

|  |  |
| --- | --- |
| **Variable or Statement** | **Output** |
| 0 | 1 |
| 1 | 0 |

**Assignment Operator**

The Assignment Operator as the name suggests is used to assign constants to variables. In other words it is used to assign the result of an expression to a variable. The table below shows all the assignment operators which as present in C.

|  |  |
| --- | --- |
| **Assignment Operator** | **Meaning** |
| = | Assigning Value |
| += | Incrementing and assigning the same value |
| -= | Decrementing and assigning the same value |
| \*= | Multiplying and assigning the same value |
| /= | Dividing and assigning the same value |
| %= | Modulo Dividing and assigning the same value |

**Increment and Decrement Operators**

The increment and decrement operator as the name suggests is used to increment or decrement the value of a variable. The operators are given below:

|  |  |
| --- | --- |
| **Increment and Decrement Operator** | **Meaning** |
| ++A | Incrementing the value by 1 before it is used |
| A++ | Incrementing the value by 1 after it is used |
| --A | Decrementing the value by 1 before it is used |
| A-- | Decrementing the value by 1 after it is used |

**Conditional Operators**

The Conditional Operators as the name suggests are used to check the condition. It is also called ternary operator. The following are the ternary operators. In C gives two ternary operators they are ‘?’ and ‘:’. The syntax for conditional operator is as follows

***<Expression 1>? <Expression 2>: < Expression 3>***

Here, first expression 1 is checked if it is a non-zero value i.e. true then **Expression 2** is evaluated and it becomes the value of the expression. Else Expression 3 becomes the value of the expression. **?** is True and: is False.

**Bitwise Operator**

Bitwise Operator as the name suggests is used to perform calculations on data at the bit level. C provides various Bitwise Operators some of them are as follows:

|  |  |
| --- | --- |
| **Bitwise Operator** | **Meaning** |
| & | Bitwise AND Operator |
| | | Bitwise OR Operator |
| ^ | Bitwise XOR Operator |
| << | Shift Left |
| >> | Shift Right |

**Special Operators**

Special Operators as the name suggests are used for a special purpose such as to get the address of the variable or to use it as a pointer variable. Some of the special operators that C provides is as follows:

|  |  |
| --- | --- |
| **Special Operator** | **Meaning** |
| & | This is used to get the address of the variable.  Example : &a will give address of a |
| \* | This is used as a pointer to a variable.  Example: \*a is a pointer to the variable a |
| Sizeof() | This gives the size of the variable  Example : Sizeof(char) will return 1 |

**What is an Expression?**

We can define expression as a sequence of operands and operators. There are three types of expressions.

1. Infix Expression
2. Postfix Expression
3. Prefix Expression

**Infix Expressions:**

Infix expressions as the name suggests the operator is in between the operands. The syntax of infix expression is as follows.

***<Operand><Operator><Operand>***

Examples of Infix Expressions are a + b, 5 + 15, 0.5 + 2.5 etc.

**Postfix Expressions:**

Postfix expressions as the name suggests the operator comes after the operands. The syntax of postfix expression is as follows.

***<Operand><Operand><Operator>***

Examples of Postfix expressions are ab+, de+ etc.

**Prefix Expression:**

Prefix expressions as the name suggests the operator comes before the operands. The syntax for writing a prefix expression is as follows.

***<Operator><Operand><Operand>***