ONLINE SUPPROT SERVICES

CERTIFICATE IN INFORMATION TECHNOLOGY

IGNOU SC-2281

Jakhepal-Ghasiwala Road, Sunam

For more information visit us at: nirmancampus.co.in

Call us at: 9815098210, 9256278000

OPERATORS AND THEIR TYPES

Operators are the symbols used to perform specific operations. For example, + operator is used to perform addition operation, > is used to compare values etc. These operators perform their operation on the operands. **Operands** are the identifier on which operation is performed. Based on the operand, operators can be classified as:

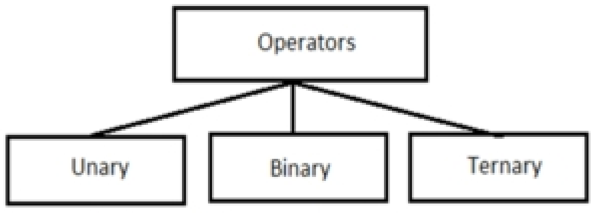

Fig: Types of Operators based on Operands

Unary Operators:

Those operators which require **one operand** to perform their operation are called unary operators. Examples of these operators are: ++, --, !etc.

For example: a++, --b

Binary Operators:

Those operators which require **two operands** to perform their operation are called binary operators. Examples of these operators are: +, -, *, %, &&, > etc. Most of the operators in C are Binary operators.

For example: a+b, a>b

Ternary Operators:

Those operators which require three operands to perform their operation are called Ternary operators. This operator is also called Conditional Operator. Example of this operator is: __?___:___
This is the only ternary operator in C language.

For Example: big = a>b **?**a **:** b ;

Operators can also be categorized according to their operations. They can be categorized as below.

1.) Arithmetic Operators.
2.) Comparison Operators
3.) Logical/Boolean Operators
4.) Assignment Operators
5.) Increment/Decrement Operators
6.) Bitwise Operators
7.) Conditional Operators
8.) Additional Operator

ARITHMETIC OPERATORS:

These operators are used for arithmetic operations. These are five operators. These are +, -, *, / and %. All these are binary operators. Following table shows the operations of these operators:

Name	Operator	Description	Example	
Addition	+	Used to perform Addition of numbers.	2+4 => 6	2.0+4.0 =>6.0
Subtract	-	Used to perform subtraction or used as any unary minus.	6-2 => 4	6.0-4.0=>2.0
Multiply	*	Used to perform multiplication of numbers.	7*2 => 14	7.0*2.0 =>14.0
Division	/	Used to perform division of numbers.	5/2 => 2 Integer division	5.0/2.0 =>2.5 Real Division
Modulus	%	Used to get remainder value after division of numbers.	7%4=>3	5.0%2.0 Not allowed

RELATIONAL OPERATORS:

These are also called **comparison operators**. These operators are used for comparing values. These are 6 operators. These are >, <, >=, <=, == and !=. All these are binary operators. Following table shows the operations of these operators:

Name	Operator	Description	Example	Result
Equal to	= =	Used to check whether two values are equal	4= =5 5= = 5	False True
Not Equal to	!=	Used to check whether two values are not equal	4! =5 4! =4	True False
Greater then	>	Used to check whether first value is greater than second	4>5 5>4	False True
Less then	<	Used to check whether first value is lass then second	4<5 5<4	True False
Greater than or equal to	>=	Used to check whether first value is greater than or equal to second value	5>=5 6>=8	True False
Less than or equal to	<=	Used to check whether first value is lesser than or equal to second value	4<=5 4<=2	True False

LOGICAL OPERATORS

These are also called **Boolean Operators**. These operators are used to form compound relational expressions. These operators are also used to compare values. These are 3 operators. These are && (AND), || (OR) and ! (NOT). AND and OR are binary operators and NOT is unary. Following table shows the operations of these operators:

Name	Operators	Description	Associatively	Example	Result
AND	&&	Return true if both operands are true otherwise returns false	Left to Right	3>5 && 4>5 3>5 && 4<5 3<5 && 4>5 3<5 && 4<5	False False False True
OR	||	Return true if at least one operand s are true	Left to Right	3>5 || 4>5 3>5 || 4<5 3<5 || 4>5 3<5 || 4<5	False True True True
NOT	!	Return true if operand is false & vice – versa	Right to Left	!(3<5) !(3>5)	False True

ASSIGNMENT OPERATORS

These Operators are used to assign/store values in a variable. The symbol of assignment operator is '='. Consider the following examples which show how to use assignment operator in C programs:

```
a = - 2;                // assigns –ve value (-2) to the variable.
b = 5;                  // assigns value (5) to the variable.
c = a + b;              // assigns the result of expression to the variable.
a = a + 10;             // self-assignment of a variable.
```

Assignment operators can also be used as shorthand operators. Shorthand assignment operators are useful in self-assignment statements. Following table shows the examples of shorthand operators used in C:

Shorthand Operator	Example for Shorthand Assignment	Equivalent Self-Assignment
+=	a+ =2	a = a + 2
- =	a- =2	a = a – 2
*=	a* =2	a = a * 2
/=	a / =2	a = a / 2
%=	a%=2	a = a % 2

Table – List of Shorthand Assignment Operands

INCREMENT AND DECREMENT OPERATOR

These are unary operators. They require only one operand. In C, '++' is the *increment* operator and '--' is the *decrement* operator. Increment operator adds one to the current value while the decrement operator decreases one to the current value.

Consider the following example:

```
int a = 5, b = 6;
a ++ ;                //a becomes 6
b – –;                //becomes 5
```

These operators can be classified into two categories. These categories are named as:

- Prefix Increment/Decrement operator. (++a/--a)
- Postfix Increment/Decrement operator. (a++/a--)

CONDITIONAL OPERATOR

It is the only ternary operator used in c language. It requires three operands to perform its operation. This operator is used to carry out conditional operations. It can be used in place of if – else statement. The syntax for conditional operator is:

Condition?Expression1 for True Condition:Expression2 for False Condition

Example of Ternary Operator:

```
int x, y, big;
x = 100;
y = 15;
big = x>y ?x : y;
```

EXPRESSION

An expression is the valid combination of operators and operands. Here, operands may be a variable or it can be a constant. Consider the following example:

```
c = a+b;
a = c + 10;
```

Here c, a,b, and 10 are the operands and + and = are the operators. Expressions can be categorized according to the operators used in the expressions:

1. Arithmetic Expressions
2. Relational Expressions
3. Logical Expressions
4. Mixed mode Expressions

1. **Arithmetic Expressions**

When Arithmetic operators are used in an expression, itis called as *Arithmetic Expression*. This expressions return numeric value. For Example:

```
c=a+b          c=a*b          etc.
```

2. **Relational Expressions**

When relational operators are used in an expression, it is called as *Relational Expressions*. Relational Expression returns either True (non zero value) or False (zero value). For example:

```
a > b        a = = b        5 > = 2        etc.
```

3. **Logical expressions**

When logical operators along with relational operators are used in the expression, then the expression is termed as Logical Expression. Logical Expression also returns either True or False value. For example:

```
(a > b)&&(a > c)        (a > b) || (b > c)        ! (a > b)        etc.
```

4. **Mixed mode expressions**

When an expression contains more than one type of operator, it is called mixed mode expression. For example:

```
(a+b) >= (c*d)
```

HIERARCHY OR PRECEDENCE OF OPERATORS IN EXPRESSIONS

An expression may contain more than one type of operators. In such situation, which operator is evaluated first?, is decided by the hierarchy of operators. The sequence of operators in which they are applied on the operands in an expression is called the *Precedence of Operators*.

Consider the following examples which illustrates how expressions are evaluated using operators precedence–

```
a = 5 * 4 / 4 + 8 – 9 / 3;          (* is evaluated)
a = 20 / 4 + 8 – 9 / 3;             (/ is evaluated)
a = 5 + 8 – 9 / 3;                  (/ is evaluated)
a = 5 + 8 - 3;                      (+ is evaluated)
a = 13 - 3;                         (- is evaluated)
a = 10;
```

TYPE CONVERSION

When value of one type is converted into other type, it is called *Type Conversion*. There are two types of type conversions.

(1.) Automatic Conversion or Implicit Conversion
(2.) Casting Value or Explicit Conversion.

<u>Automatic/Implicit Conversion:</u>

This type of conversion is automatic. For this type of conversion, we use assignment (=) operator. It is also called implicit conversion. This type of conversion is used when lower data type operand is converted into higher data type. There is not loss of information in this type of conversion.

Consider the following example for automatic conversion:

```
int m = 15;
float n = m;
```

<u>Casting a value or Explicit Conversion</u>

This is forceful conversion. For this type of conversion, we use caste operator. It is also called explicit conversion. There may or may not be any loss of information in this type of conversion. This type of conversion is used when higher data type operand is converted into lower data type.

The syntax for this type of casting is:

(Desired data type) Expression

For example:

```
float m;
int n = 7;
m = (float)n/2;
```