



# S.P.M College, Udaipur

Bachelor Of Computer Application (BCA)

Part -3

– Hira Kumar

## 5. Bitwise operators (6)

- Bitwise operators perform on the binary representations of integers. OR Bitwise operators work on individual bits of integers. These are basically used for testing or shifting bits.

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) will give 12, which is 0000 1100
	Binary OR Operator copies a bit if it exists in either operand.	(A   B) will give 61, which is 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) will give 49, which is 0011 0001
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) will give -61, which is 1100 0011 in 2's complement form. Result in decimal : ~A = -(A+1)
<<	Binary Left Shift Operator. The left operand's value is moved left by the number of bits specified by the right operand.	A << 2 will give 240 which is 1111 0000
>>	Binary Right Shift Operator. The left operand's value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15 which is 0000 1111

```
1 #include<iostream>
2 #include<conio.h>
3 using namespace std;
4 int main()
5 {
6     int a=5,b=3;
7     cout << (a & b) << endl; //and
8     cout << (a | b) << endl; // or
9     cout << "xor = " << (a ^ b) << endl; //xor
10    cout << (~a) << endl; // bitwise negative OR Complement operator
11    cout << (~b) << endl; // bitwise negative
12    cout << (a << 1) << endl; // left shift operator
13    cout << (a >> 2) << endl; // right shift operator
14
15    getch();
16    return 0;
17 }
```

The screenshot shows the Code::Blocks IDE interface. The code editor window displays the C++ code above. To the right, the terminal window shows the execution results:

```
C:\Users\hira\Desktop\hira.exe
1
7
xor = 6
-6
-4
10
1
```

The terminal also shows build information at the bottom:

```
Line Message
==== Build file: "r
==== Build finished
```

### Not Confuse

Bitwise	Logical
<code>a &amp; b</code>	<code>a &amp;&amp; b</code>
<code>a   b</code>	<code>a    b</code>
<code>a ^ b</code>	<code>a != b</code>
<code>~a</code>	<code>!a</code>

## 6. Assignment Operators (11)

- Assignment operators are used to assign a value (or) an expression (or) a value of a variable to another variable.
- Syntax : variable name=expression (or) value (or) variable
  - ✓ Ex :    `x=10;`  
`y=a+b;`  
`z=p`
- `i += 2` ( The operator `+=` is called an assignment operator.)

Operator	Example	Meaning
<code>+=</code>	<code>x += y</code>	<code>x=x+y</code>
<code>-=</code>	<code>x -= y</code>	<code>x=x-y</code>
<code>*=</code>	<code>x *= y</code>	<code>x=x*y</code>
<code>/=</code>	<code>x /= y</code>	<code>x=x/y</code>
<code>%=</code>	<code>x %= y</code>	<code>X=x%y</code>

<code>=</code>	Assignment operator
<code>+= -=</code>	Addition/subtraction assignment
<code>*= /=</code>	Multiplication/division assignment
<code>%= &amp;=</code>	Modulus and bitwise assignment
<code>^=  =</code>	Bitwise exclusive/inclusive OR assignment
<code>&lt;&lt;= &gt;&gt;=</code>	Bitwise left/right shift assignment

If `x *= y + 1` means  $\rightarrow x = x * (y + 1)$  rather than  $x = x * y + 1$

```

1 #include<iostream>
2 #include<conio.h>
3 using namespace std;
4 int main()
5 {
6     int a=10;
7     int b=a;
8     cout << "a = " << a << endl;
9     cout << "b = " << b << endl;
10    b *= a+b;
11    cout << "b = " << b;
12    getch();
13    return 0;
14 }
```

```
C:\Users\hira\Desktop\hira.exe
a = 10
b = 10
b = 200
```

## 7. Ternary or Conditional Operator (1)

- In c++, the ternary operator is a shorthand way to write simple if-else statement, It's the only operator in c++ that takes three operands.
- It is also called Conditional Operator.
- Syntax : Condition ? expression\_if\_true : expression\_if\_false;
- Example: int a=10,b=20;
  - Int max = (a>b) ? a : b ;

**Explain this line**

**int max = (a>b) ? a : b ;**



```
int max;
if(a>b)
{
    max =a;
}
else
{
    max =b;
}
```

```
1 #include<iostream>
2 #include<conio.h>
3 using namespace std;
4 int main()
5 {
6     int a=10, b=20;
7     int max_value = (a>b) ? a : b ;
8     cout << "Max Value = " << max_value;
9     getch();
10    return 0;
11 }
```

Max Value = 20

```
1 #include<iostream>
2 #include<conio.h>
3 using namespace std;
4 int main()
5 {
6     int a=10, b=20;
7     int max_value;
8     if (a>b)
9     {
10         cout << "Max value = " << a;
11     }
12     else
13     {
14         cout << "Max value = " << b;
15     }
16     getch();
17     return 0;
18 }
```

C:\Users\hira\Desktop\hira.exe

Max value = 20

## 8. Special / Miscellaneous Operators

- A part from these operators, there are a few operators that do not fit in any of the above categories. These are:-

## 1. sizeof Operator

- Return the size (in byte) of a data type or variable.
- Syntax : sizeof(data\_type / Variable\_name)
- Example:-

```
1  #include<iostream>
2  #include<conio.h>
3  using namespace std;
4  int main()
5  {
6      int a=10;
7      cout << "Size of int = " << sizeof(int) << endl;
8      cout << "size of a = " << sizeof(a);
9      getch();
10     return 0;
11 }
```

C:\Users\hira\Desktop\hira.exe

```
Size of int = 4
size of a = 4
```

## 2. Comma Operator(,)

- Evaluates multiple expressions and returns the value of the last expression.
- Example : int a=5,b=10;  
Int c = (a++,b++,a+b);  
Cout << c; // c = 17

## 3. Address-of-operator (&)

- Return the memory address of a variable.
- Example :  
Int a=10;  
Cout << &a; //ask address of operator

```
1  #include<iostream>
2  #include<conio.h>
3  using namespace std;
4  int main()
5  {
6      int a=10;
7      cout << "a = " << a << endl;
8      cout << "address of a = " << &a;
9      getch();
10     return 0;
11 }
```

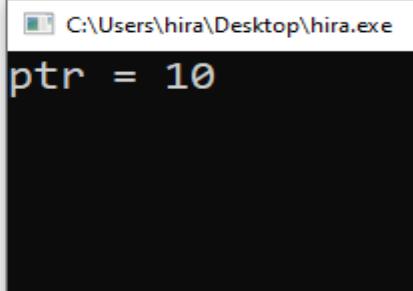
C:\Users\hira\Desktop\hira.exe

```
a = 10
address of a = 0xaed03ff9cc
```

## 4. Dereference Operator (\*)

- Accesses the value at a memory address (opposite of &)

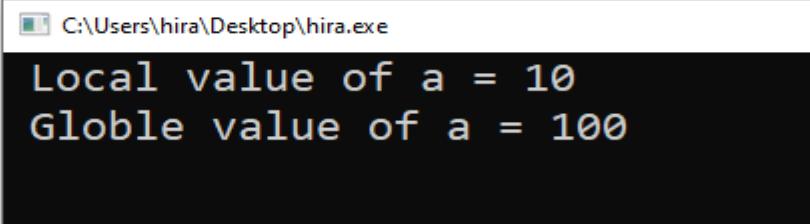
```
1 #include<iostream>
2 #include<conio.h>
3 using namespace std;
4 int a=100;
5 int main()
6 {
7     int a=10;
8     int *ptr = &a;
9     cout << "ptr = " << *ptr;
10    getch();
11 }
12
13
```



## 5. Scope Resolution Operator (::)

- Accesses globle variable, namespace members, or class static members.

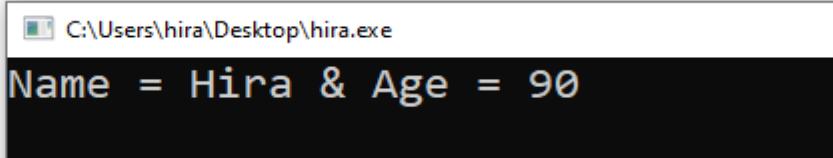
```
1 #include<iostream>
2 #include<conio.h>
3 using namespace std;
4 int a=100;
5 int main()
6 {
7     int a=10;
8     cout << " Local value of a = " << a << endl;
9     cout << " Globle value of a = " << ::a ;
10    getch();
11 }
12
13
```



## 6. Dot operator (.)

- Accesses members of an object/struct.

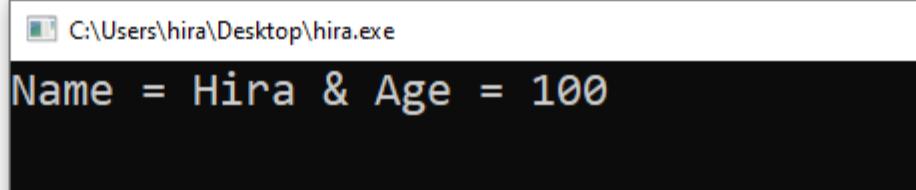
```
1 #include<iostream>
2 #include<conio.h>
3 using namespace std;
4 struct student
5 {
6     string name;
7     int age;
8 };
9 int main()
10 {
11     student sl;
12     sl.name ="Hira"; // dot operator
13     sl.age = 90;
14     cout << "Name = " << sl.name << " & Age = " << sl.age;
15     getch();
16 }
17
18
```



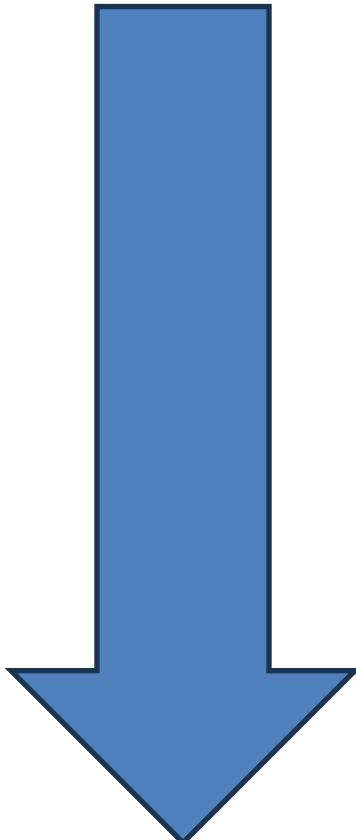
## 7. Arrow Operator (->)

- Accesses members through a pointer.

```
1 #include<iostream>
2 #include<conio.h>
3 using namespace std;
4 struct student
5 {
6     string name;
7     int age;
8 };
9 int main()
10 {
11     student sl;
12     student *ptr = &sl;
13     ptr -> name = "Hira";
14     ptr -> age = 100;
15     cout << "Name = " << sl.name << " & Age = " << sl.age;
16     getch();
17 }
18
19
```



```
C:\Users\hira\Desktop\hira.exe
Name = Hira & Age = 100
```



Operator Name	Operator	Description	Associativity	Precedence(Priority)
Part parentheses Operator	() [] . -> ++--	Parentheses or function call Brackets or array subscript Dot or Member selection operator Arrow operator Postfix increment/decrement	left to right	
Unary Operator	++-- + - ! ~ (type)* & sizeof	Prefix increment/decrement Unary plus and minus not operator and bitwise complement type cast Indirection or dereference operator Address of operator Determine size in bytes	right to left	PUMAS REBL TAC 1 2 3 4 5 6 7 8 9 10 11 12
Multiplication Operator	* / %	Multiplication, division and modulus	left to right	
Addition Operator	+ -	Addition and subtraction	left to right	
Shift operators	<< >>	Bitwise left shift and right shift	left to right	
Relational Operator	< <= > >= == !=	relational less than/less than equal to relational greater than/greater than or equal to	left to right	
Bitwise Operator	&& ^	Bitwise AND Bitwise exclusive OR Bitwise inclusive OR	left to right	
Logical Operator	&&	Logical AND Logical OR	left to right	
Conditional / Ternary Operator	? :	Ternary operator	right to left	TAU (Right to left) T- Ternary
Assignment Operator	= += -= *= /= %= &= ^=  = <<= >>=	Assignment operator Addition/subtraction assignment Multiplication/division assignment Modulus and bitwise assignment Bitwise exclusive/inclusive OR assignment	right to left	A-Assignment U-Unary operator
Comma Operator	,	comma operator	left to right	

solve priority number

Just like  
BODMAS

TAU (Right to left)  
T- Ternary  
A-Assignment  
U-Unary operator

=====HIRA KUMAR=====

HIRA KUMAR