



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 operands 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 operands 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 }
```

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

```
1
7
xor = 6
-6
-4
10
1
```

Not Confuse

Bitwise	Logical
a & b	a && b
a b	a b
a ^ b	a != b
~a	!a

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
+=	x += y	x=x+y
-=	x -= y	x=x-y
*=	x *= y	x=x*y
/=	x /= y	x=x/y
%=	x %= y	X=x%y

=	Assignment operator
+= -=	Addition/subtraction assignment
*= /=	Multiplication/division assignment
%= &=	Modulus and bitwise assignment
^= =	Bitwise exclusive/inclusive OR 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 }
15

```

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

```

a = 10
b = 10
b = 200

```

7. Ternary or Conditional Operator (1)

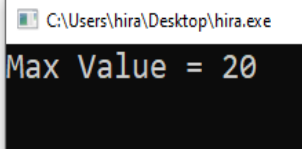
- In c++, the ternary operator is a shorthand way to write simple if-else statement, It's the only operator in c++ that takes three operands.
- It is also called Conditional Operator.
- Syntax : Condition ? expression_if_true : expression_if_false;
- Example: int a=10,b=20;
 - Int max = (a>b) ? a : b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

Explain this line

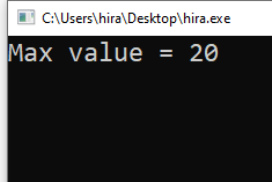
int max = (a>b) ? a : b ;

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 }
12
```



```
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 }
19
```



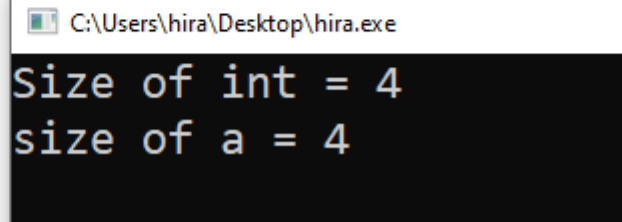
8. Special / Miscellaneous Operators

- 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 }
12
```



```
C:\Users\hira\Desktop\hira.exe
Size of int = 4
size of a = 4
```

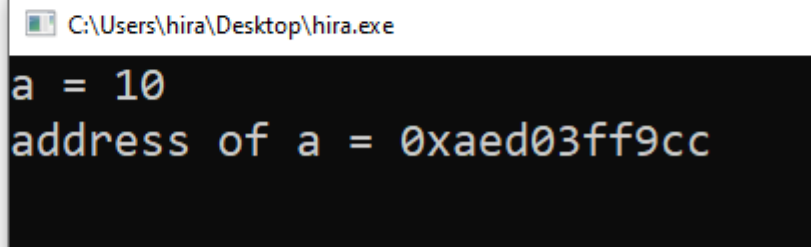
2. Comma Operator(,)

- 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 }
12
```



```
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     return 0;
12 }
13
```

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

ptr = 10

5. Scope Resolution Operator (::)

- Accesses global 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 << " Global value of a = " << ::a ;
10     getch();
11     return 0;
12 }
13
```

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

Local value of a = 10
Global value of a = 100

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 s1;
12     s1.name = "Hira"; // dot operator
13     s1.age = 90;
14     cout << "Name = " << s1.name << " & Age = " << s1.age;
15     getch();
16     return 0;
17 }
18
```

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

Name = Hira & Age = 90

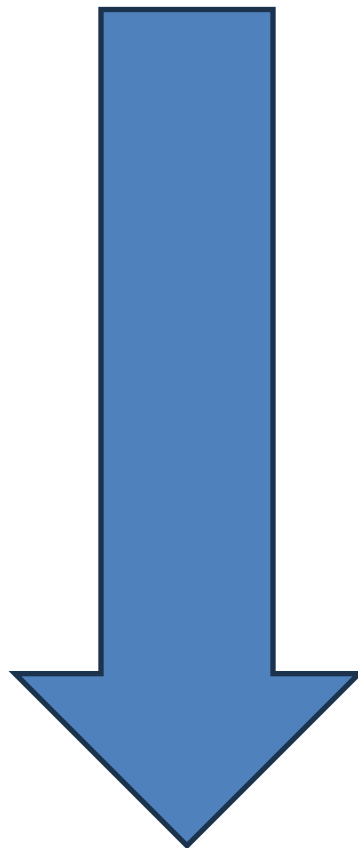
7. Arrow Operator (->)

- 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 s1;
12     student *ptr =&s1;
13     ptr -> name = "Hira";
14     ptr -> age = 100;
15     cout << "Name = " << s1.name << " & Age = " << s1.age;
16     getch();
17     return 0;
18 }
19
```



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



Operator Name	Operator	Description	Associativity	Precedence(Priority)
Partentheses Operator	() [] . -> ++ --	Parentheses or function call Brackets or array subscript Dot or Member selection operator Arrow operator Postfix increment/decrement	left to right	PUMAS REBL.TAC 1 2 3 4 5 6 7 8 9 10 11 12 solve priority number
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	
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	Just like BODMAS
Relational Operator	< <= > >= == !=	relational less than/less than equal to relational greater than/greater than or equal to Relational equal to or not 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 A-Assignment U-Unary operator
Assignment Operator	= += -= *= /= %= &= ^= = <<= >>=	Assignment operator Addition/subtraction assignment Multiplication/division assignment Modulus and bitwise assignment Bitwise exclusive/inclusive OR assignment	right to left	
Comma Operator	,	comma operator	left to right	

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