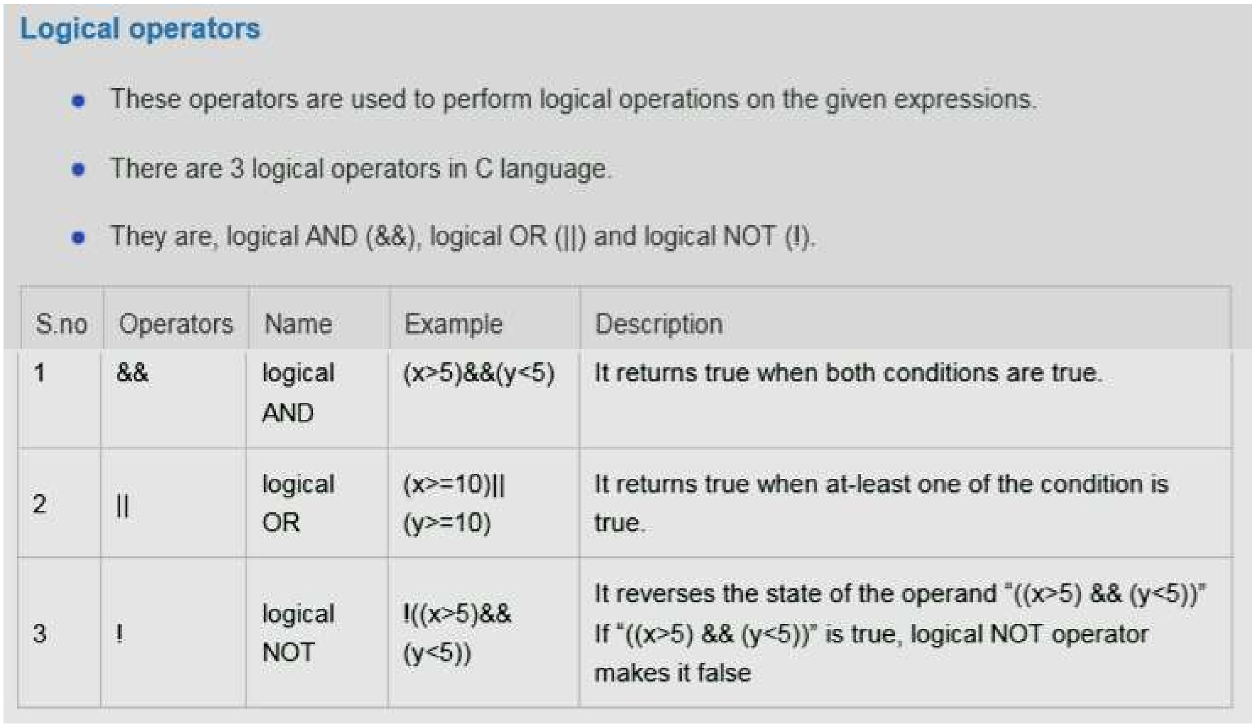
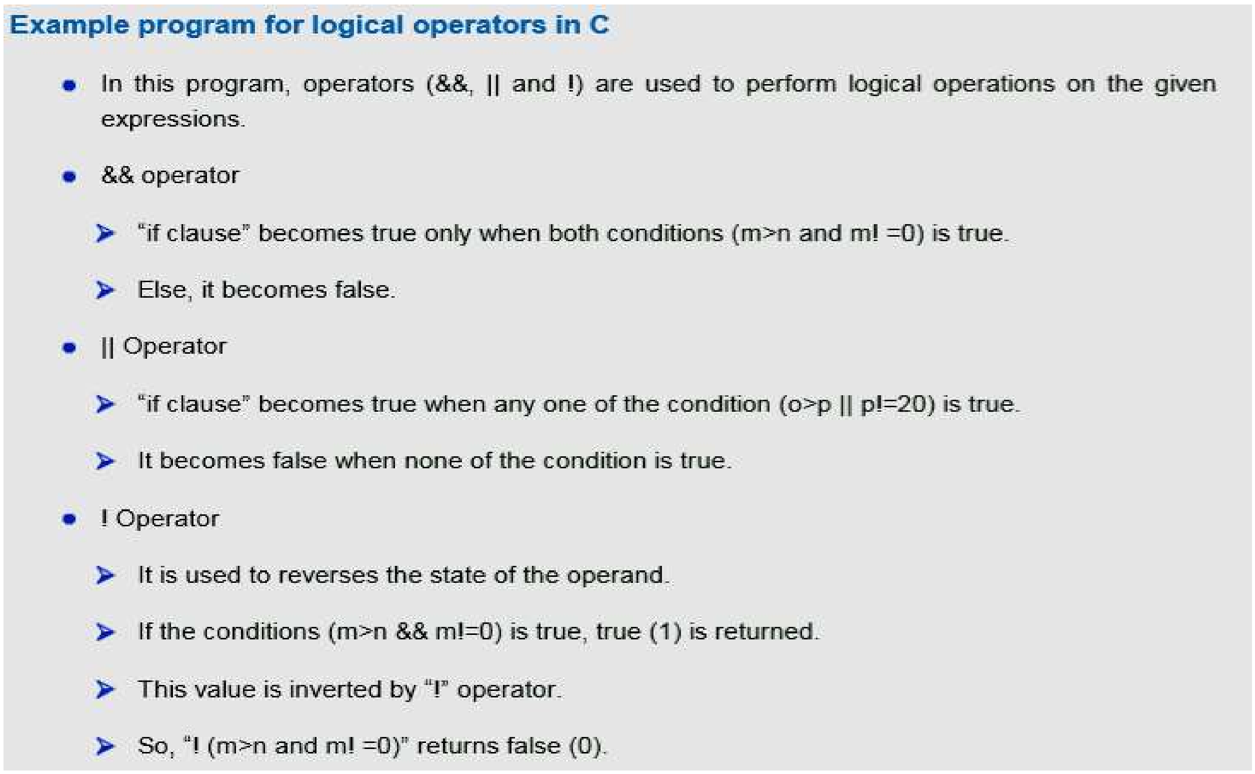
Chapter -3 : Introduction to 'C' Language



Logical operators

- These operators are used to perform logical operations on the given expressions.
- There are 3 logical operators in C language.
- They are, logical AND (&&), logical OR (||) and logical NOT (!).

S.no	Operators	Name	Example	Description
1	&&	logical AND	(x>5)&&(y<5)	It returns true when both conditions are true.
2	||	logical OR	(x>=10)||(y>=10)	It returns true when at-least one of the condition is true.
3	!	logical NOT	!((x>5)&&(y<5))	It reverses the state of the operand "((x>5) && (y<5))" If "((x>5) && (y<5))" is true, logical NOT operator makes it false



Example program for logical operators in C

- In this program, operators (&&, || and !) are used to perform logical operations on the given expressions.
- && operator
  - "if clause" becomes true only when both conditions (m>n and m! =0) is true.
  - Else, it becomes false.
- || Operator
  - "if clause" becomes true when any one of the condition (o>p || p!=20) is true.
  - It becomes false when none of the condition is true.
- ! Operator
  - It is used to reverses the state of the operand.
  - If the conditions (m>n && m!=0) is true, true (1) is returned.
  - This value is inverted by "!" operator.
  - So, "! (m>n and m! =0)" returns false (0).

```
#include <stdio.h>
int main()
{
    int m=40, n=20;
    int o=20, p=30;
    if(m>n && m!=0)
    {
        printf("&& Operator:  Both conditions are true\n");
    }
    if(o>p || p!=20)
    {
        printf("|| Operator:  Only one condition is true\n");
    }
    if(!(m>n && m!=0))
    {
       printf("! Operator:  Both conditions are true\n");
    }
    else
    {
       printf("! Operator:  Both conditions are true. " \
              "But, status is inverted as   false\n");
    }
}
```

Output

```
&& Operator:  Both conditions are true
|| Operator:  Only one condition is true
! Operator:  Both conditions are true. But, status is inverted as false
```

Bit wise operators

- These operators are used to perform bit operations.
- Decimal values are converted into binary values which are the sequence of bits and bit wise operators work on these bits.
- Bit wise operators in C language are & (bitwise AND), | (bitwise OR), ~ (bitwise OR), ^ (XOR), << (left shift) and >> (right shift).

Truth table for bit wise operation

X	Y	X | Y	X &Y	X ^ Y
0	**0**	**0**	**0**	**0**
0	**1**	**1**	**0**	**1**
1	**0**	**1**	**0**	**1**
1	**1**	**1**	**1**	**0**

Bit wise operators

Operator symbol	Operator name
&	Bitwise AND
|	Bitwise OR
~	Bitwise NOT
^	XOR
<<	Left Shift
>>	Right Shift

- Consider x=40 and y=80.
- Binary form of these values are given below.
  - x = 00101000.
  - y= 01010000.
- All bit wise operations for x and y are given below.
  - x&y = 00000000 (binary) = 0 (decimal).
  - x|y = 01111000 (binary) = 120 (decimal).
  - ~x = 1111111111111111111111111111111111111111111111111111111111010111 = -41 (decimal).
  - x^y = 01111000 (binary) = 120 (decimal).
  - x << 1 = 01010000 (binary) = 80 (decimal).
  - x >> 1 = 00010100 (binary) = 20 (decimal).
- Note:
  - Bit wise NOT : Value of 40 in binary is 0000000000000000000000000000000000000000000000000000000000101000.
  - So, all 0's are converted into 1's in bit wise NOT operation.
  - Bit wise left shift and right shift : In left shift operation "x << 1 ", 1 means that the bits will be left shifted by one place.

    If we use it as "x << 2 ", then, it means that the bits will be left shifted by 2 places.

```
#include <stdio.h>
int main()
{
   int m=40, n=80, AND_opr, OR_opr, XOR_opr, NOT_opr;
   AND_opr = (m&n);
   OR_opr = (m|n);
   NOT_opr = (~m);
   XOR_opr = (m^n);
   printf("AND_opr value = %d\n", AND_opr);
   printf("OR_opr value = %d\n", OR_opr);
   printf("NOT_opr value = %d\n", NOT_opr);
   printf("XOR_opr value = %d\n", XOR_opr);
   printf("left_shift value = %d\n", m << 1);
   printf("right_shift value = %d\n", m >> 1);
}
```

Output:

```
AND_opr value = 0
OR_opr value = 120
NOT_opr value = -41
XOR_opr value = 120
left_shift value = 80
right_shift value = 20
```

Conditional or ternary operators

- Conditional operators return one value if condition is true and returns another value is condition is false.
- This operator is also called as ternary operator.
  - Syntax : (Condition? true_value: false_value);
  - Example : (A > 100 ? 0 : 1);
- In above example, if A is greater than 100, 0 is returned else 1 is returned.
- This is equal to if else conditional statements.

```
#include <stdio.h>
int main()
{
       int x=1, y;
       y = (x ==1 ? 2 : 0);
       printf("x value is %d\n", x);
       printf("y value is %d", y);
}
```

Output:

```
x value is 1
y value is 2
```

Increment/decrement Operators

- Increment operators are used to increase the value of the variable by one and decrement operators are used to decrease the value of the variable by one in C programs.
- Syntax:
  - Increment operator : ++var_name; (or) var_name++;
  - Decrement operator : – - var_name; (or) var_name – -;
- Example:
  - Increment operator : ++ i ; i ++ ;
  - Decrement operator: – - i ; i – - ;

Example program for increment operators in C

- In this program, value of "i" is incremented one by one from 1 up to 9 using "i++" operator and output is displayed as "1 2 3 4 5 6 7 8 9".

```
#include <stdio.h>
int main()
{
     int i=1;
     while(i<10)
     {
        printf("%d ",i);
        i++;
     }
}
```

Output:

```
1 2 3 4 5 6 7 8 9
```

Special Operators in C

- Below are some of special operators that C language offers.

S.no	Operators	Description
1	&	This is used to get the address of the variable. Example : &a will give address of a.
2	*	This is used as pointer to a variable. Example : * a where, * is pointer to the variable a.
3	Sizeof ()	This gives the sizeof the variable. Example : sizeof(char) will give us 1.

Example program for & and * operators in C

- In this program, “&” symbol is used to get the address of the variable and “*” symbol is used to get the value of the variable that the pointer is pointing to.
- Please refer C – pointer topic to know more about pointers.

```
#include <stdio.h>

int main()
{
        int *ptr, q;
        q = 50;
        /* address of q is assigned to ptr       */
        ptr = &q;
        /* display q's value using ptr variable */
        printf("%d", *ptr);
        return 0;
}
```

Output:

```
50
```

Example program for sizeof() operator in C

- sizeof() operator is used to find the memory space allocated for each C data types.

```
#include <stdio.h>
#include <limits.h>
int main()
{
  int a;
  char b;
  float c;
  double d;
  printf("Storage size for int data type:%d \n", sizeof(a));
  printf("Storage size for char data type:%d \n", sizeof(b));
  printf("Storage size for float data type:%d \n", sizeof(c));
  printf("Storage size for double data type:%d\n", sizeof(d));
  return 0;
}
```

Output:

```
Storage size for int data type: 4
Storage size for char data type: 1
Storage size for float data type: 4
Storage size for double data type: 8
```