Abhishek Sharma Notes Lecture 5 and 6

Operator

Operators is used to perform arithmetic & logical operations

Operator category is

Unary : Only 1 Operand

Example a = -5

Binary : 2 Operand required

Example : a = 3+5

Ternary: 3 Operand required

?

Assignment operator : =

```
#include <stdio.h>
int main() {
    int a = -129;
    a = 120
    printf("%d",a);
    return 0;
}
```

Assignment will the value at location associated as a

L-value and R-value

L-value of a variable is Location in memory associated with variable.

R-value value of a variable is value stored in location associated with the variable.

Assignment operator changes the changes L- value of variable.

The constant (integer, string , character, float), expression cannot have L-value so you can not write them in LHS of assignment operator

10 = 20, **not allowed**

10+20 = 30, **not allowed**

'a' = 30, not allowed

Only variable have L value associated with them

RHS can be expression, literal constant , variable

char a = 'a'

int a1 = 2+3*4;

int a2 = 100;

Unary Operand

+1, - 3, &a

Binary Arithmetic Operator

+ , - * / , %

Order of evaluation of the operator

Division and % Operand

Dividend,

Divisor ,

Quotient ,

Remainder

Dividend = Quotient * Divisor+ Remainder

7 / 3, -7/3, 7/-3, -7/-3

	7/3	7/-3	-7/3	-7/-3
Dividend	7	-7	7	-7
Divisor	3	3	-3	-5
Quotient	2	-2	-2	2
Remainder	1	1	-1	-1

Sign of Remainder : Dividend positive Remainder positive , Dividend Negative Remainder Negative

```
#include <stdio.h>
int main() {
printf("%d\n",7/3);
printf("%d\n",7/-3);
printf("%d\n",-7/3);
printf("%d\n",-7/-3);
printf("%d\n",7%3);
printf("%d\n",7%-3);
printf("%d\n",-7%3);
printf("%d\n",-7%-3);
return 0;
}
```

Expression Rule:

- Operation between integer and integer is integer
- Operation between integer and float is float
- Operation between float and float is float

Precedence rule:

Operator precedence determines the order in which operators are evaluated. Operators with higher precedence are evaluated first.

Associativity Rule

Associativity: In programming languages, the associativity (or fixity) of an operator is a property that determines how operators of the same precedence are grouped in the absence of parentheses.

bracket	( )		Highest
Unary minus	-		
Multiplicative	*, / , %	Left to Right	
Additive	+ -	Left to Right	
Equal	=	Right to Left	Lowest

Problem Solving

Question 1

```
#include <stdio.h>
int main(){
        float x;
        x = 7*2.0/2+10/3;
        printf("%f", x);
        return 0;
}
```

The value of x is ___

(A) 10 (b) 10.0 (c) 10.33 (d) 11.0

Question 2

```
#include<stdio.h>
int main(){
        int x;
        x= -2 + 11 - 7 * 9 % 6 / 12;
        printf("%d",x);
        return 0 ;
}
```

The value of x is ___

(A) 6 (b) 7 (c) 8 (d) 9

Question 3

```
#include<stdio.h>
int main(){
        int x;
        x=  3/2*4+3/8+3;
        printf("%d",x);
        return 0 ;
}
```

The value of x is ___

(A) 6 (b) 7 (c) 8 (d) 9

Question 4

```
#include<stdio.h>
int main(){
        int x;
        x=  4+2%-8;
        printf("%d",x);
        return 0 ;
}
```

The value of x is ___

(A) 6 (b) 7 (c) 8 (d) 9

Question 5

```
#include<stdio.h>
int main(){
        int x;
        x=  2 * 3/4+4/4 +8-2+5/8;
        printf("%d",x);
        return 0 ;
}
```

The value of x is ___

(A) 6 (b) 7 (c) 8 (d) 9

Scoping Rule:

The scope of a variable in C is the block or the region in the program where a variable is declared, defined, and used. Outside this region, we cannot access the variable and it is treated as an undeclared identifier.

Example 1

```
#include<stdio.h>
int  main {
      {
            int a =10 ;
            printf("%d"; a); // will print the 10
      }
}
```

Output : it will print 10

Example 2

```
#include<stdio.h>
int  main() {
     {
          int a =10 ;
     }
     printf("%d", a); // will not print 10
     return 0 ;
}
```

Output: It will not print 10, variable is not defined

Lexical Scope Rule:

If variable is not defined within the block then definition of the variable is searched in next upper block. This process is repeated until the definition is found.

```
#include<stdio.h>
int x = 40;
int  main() {
      int x = 30;
      {
          int x = 20;
          {
               int x = 10;
              printf("%d", x);
          }
      }
    return 0 ;
}
```


Relational Operator

The relational operators are

>	<	<=	>=	==	!=
Less than	Greater than	Greater than equal to	Less than equal to	Exactly equal	Not equal to

Output of the relational operator is 0 or 1

(a) 30 > 40

(b) 30 >= 40

(c) 30 == 40

(d) 40!=30

(e) 40==40

(f) 40!=40

(g) 50 <50

```
#include<stdio.h>
int x = 40;
int  main() {
    printf("%d\n", 30>40);
    printf("%d\n", 30>=40);
    printf("%d\n", 30==40);
    printf("%d\n", 30!=40);
    printf("%d\n", 40!=30);
    printf("%d\n", 40==40);
    printf("%d\n", 50>50);
    printf("%d\n", 50<=50);
    return 0 ;
}
```


Precedence of relational Operator

3	* / %	Multiplication, division, and modulus	left to right
4	+ -	Addition and subtraction	left to right
6	< <= > >=	Relational less than and less than or equal to Relational greater than and greater than or equal to	left to right
7	== !=	Relational equal to and not equal to	left to right

Logical Operator:

1. Logical AND (&&)
2. Logical OR (||)
3. Logical NOT (!)

Logical AND (&&)

FIRST OPERAND	SECOND OPERAND	RESULT
true	true	true
true	false	false
false	true	false
false	false	false

```
#include <stdio.h>
int main() {
  int a = 20;
  int b = 30;
   printf("%d", a > 10 && b > 10);
}

// output

// Both numbers are greater than 10
```

Logical OR (||)

FIRST OPERAND	SECOND OPERAND	RESULT
true	true	true
true	false	true
false	true	true
false	false	false

Example

```
#include <stdio.h>
int main() {
 int a = 20;
 int b = 5;
  printf("%d", a > 10 || b > 10);

}
```

Logical NOT (!)

OPERAND	RESULT
true	false
false	true

```
#include <stdio.h>
int main( ) {
 int a = 20;
 int b = 5;
  printf("%d", !(a > 10 ));
}
```

Question practice

Question 1

What will be the output of the following C code?

#include <stdio.h>

```
  #include<stdio.h>
int main()
  {
     int x = 1, y = 0, z = 0;
     int a = x && y || z+1;
     printf("%d", a);
  }
```

A. 6
B. 5
C. 0
D. 1

Question 2

What will be the output of the following C code?

```
#include <stdio.h>
   int main () {
     int x = 1, y = 0, z = 5;
     int a = x && y && z+1;
     printf("%d", a);
  }
```

A. 6
B. 5
C. 0
D. 1

Question 3

```
#include <stdio.h>
   int main () {
     int a =(( 5!=10 )==8)&&((6>10)==(10>6));
     printf("%d", z);
  }
```

A. 0
B. 1

Precedence of Logical operators

3	* / %	Multiplication, division, and modulus	left to right
4	+ -	Addition and subtraction	left to right
6	< <= > >=	Relational less than and less than or equal to Relational greater than and greater than or equal to	left to right
7	== !=	Relational equal to and not equal to	left to right
11	&&	Logical AND	left to right
12	||	Logical OR	left to right

Example 1

```
#include <stdio.h>
    int main ()  {
   int a =(( 5+6!=10 )==8)&&((6*2/4>10)==(10>6));
    printf("%d", a);
   }
```

Answer :0

The output of the program is _____

Example 2

```
#include <stdio.h>
    int main ()  {
   int a =(( 5+6!=10 )==8)||((6*2/4>10)==(10>6));
    printf("%d", a);
   }
```

The output of the program is _____

Answer :0

Example 3

```
#include <stdio.h>
    int main ()  {
      int a =5+5!=10|| 6+4;
```

```
    printf("%d", a);
  }
```

The output of the program ___

Answer :1

Example 4

What is the output of the program?

```
#include <stdio.h>
   int main ()  {
     int a =3 < 6!= 3<6 && 9>11== 9>11;
   printf("%d", a);
  }
```

The output of the program ___

Answer 0

Short Circuit Code:

int a;

a = 0&& [ ] ;

if the first operand is zero then second operand is not evaluated.

int a;

a = 12&& [ ] ;

if the first operand is one then second operand is evaluated.

int a;

a = 1|| [ ]

if the first operand is one then second operand is not evaluated.

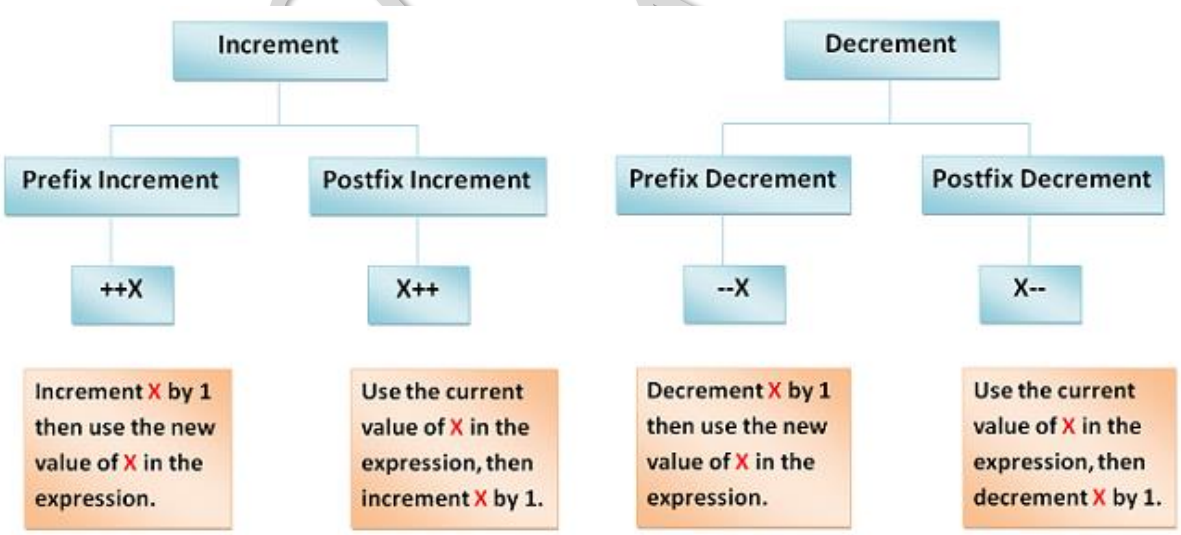
int a;

a = 0|| [ ]

if the first operand is zero then second operand is evaluated.

Increment & decrement Operator

These Operator modify the value of the variable



Example 1

What is the output of the program?

```
#include <stdio.h>
     int main ()  {
      int x=5;
      int y;
      y = ++x;
      printf("%d %d", x, y);
 }
```

The output of the program

(A) 5,5 (B) 5,6 (C) 6,5 (D) 6,6

Answer

D

Example2

```
#include <stdio.h>
      int main ()  {
       int x=5;
       int y;
       y = x++;
       printf("%d %d", x, y);
  }
```

(A) 5,5 (B) 5,6 (C) 6,5 (D) 6,6

Answer

C

Compound Assignment Operator

x += y assign (x+y) to x

x -= y assign (x-y) to x

x *= y assign (x*y) to x

x /= y assign (x/y) to x

x %= y assign (x%y) to x

Example 1

```
#include <stdio.h>
      int main ()  {
       int x=5;
       int y=5,z;
       z = ++x ||++y ;
       printf("%d %d %d", x, y,z);
  }
```

The output of the program

(A) 6,5 ,1 (B) 6,6,1 (C) 5,5,1 (D) 1,1,1

Example 2

```
#include <stdio.h>
     int main ()  {
      int x=5;
      int y=5,z;
      z = ++x && ++y ;
      printf("%d %d %d", x, y,z);
 }
```

The output of the program

(A) 6,5 ,1 (B) 6,6,1 (C) 5,5,1 (D) 1,1,1

Answer

B

Example 3

```
#include <stdio.h>
     int main ()  {
      int x=0;
      int y=0,z;
      z = ++x || ++y ;
      printf("%d %d %d", x, y,z);
 }
```

(A) 1,1 ,0 (B) 1,0,1 (C) 1,1,1 (D) 0,1,1

Answer

B

Example 4

```
#include <stdio.h>
     int main ()  {
      int x=0;
      int y=0,z;
      z = x++ || ++y ;
      printf("%d %d %d", x, y,z);
```

```
}
```

(A) 1,1 ,0 (B) 1,0,1 (C) 1,1,1 (D) 0,1,1

Answer

C

Example 5

```
#include <stdio.h>
     int main ()  {
      int x=0;
      int y=0,z;
      z = x++ || y++ ;
      printf("%d %d %d", x, y,z);
}
```

(A) 1,1 ,0 (B) 1,0,1 (C) 1,1,1 (D) 0,1,1

Answer

A

GATE 2017

Consider the following C program.

```
# include <stdio.h>
int main () {
   int m = 10;
   int n, n1;
   n = ++m;
   n1 = m++;
   n--;
   --n1;
   n - = n1;
   printf (“%d”, n) ;
   return 0;
```

}

The output of the program is _____.

Key: (0)

Bit Wise Operator

Number System

Decimal number system (base 10)

0,1,2,3….9

Binary Number system (Base 2)

0,1

Octal Number System (base 8)

01,2,3…7

Hexadecimal number System (base 16)

0,1,2,3…10 A,B,C,D,E,F

Number base system:

1022 Expand it

$1\times10^3+0\times10^2+2\times10+2$

Same goes with Binary Octal and hexadecimal

Octal constant starts with 0

int x = 012

hexadecimal Number start with 0x

intx = 0x12A

Example 1

What is the output the program

```
#include <stdio.h>
  int main ()  {
        int x = 018;
       printf("%d", x);
  }
```

(A) 16 (B) Compiler Error

(C) warning (D) 18

Example 2

```
#include <stdio.h>
  int main ()  {
        int x = 017;
       printf("%d", x);
  }
```

(A) 17 (B) Compiler Error

(C) warning (D) 15

Answer

D

Hexadecimal

Example 4

What is the output the program

```
#include <stdio.h>
  int main ()  {
       int x = 0x18;
       printf("%d", x);
  }
```

(A) 16 (B) 24

(C) warning (D) 18

Answer

A

Example 5

What is the output the program

```
#include <stdio.h>
  int main ()  {
        int x = 0x1A;
       printf("%d", x);
  }
```

(A) 26 (B) 100

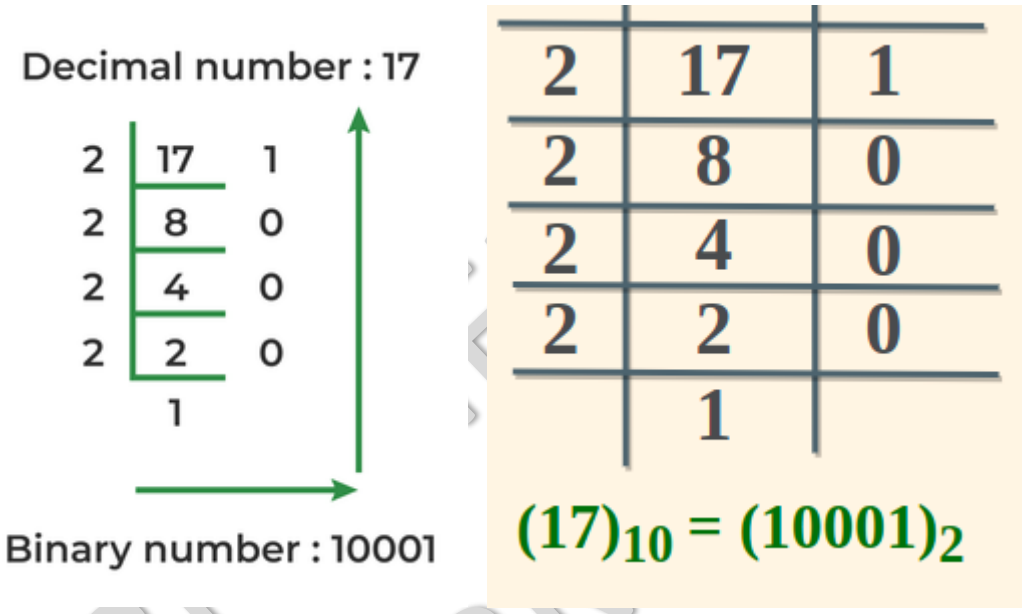
(C) warning (D) 18

Answer

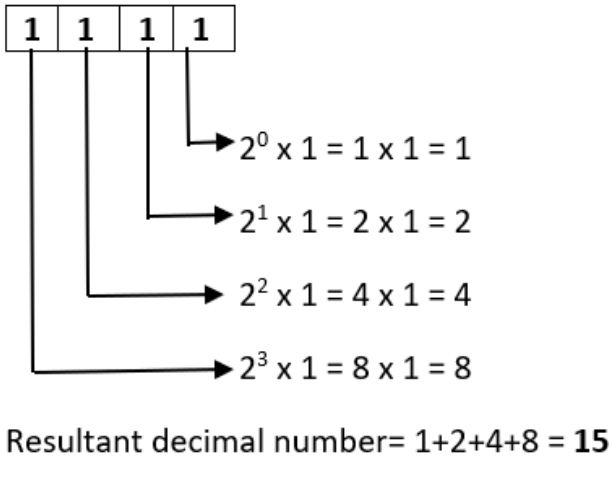
A

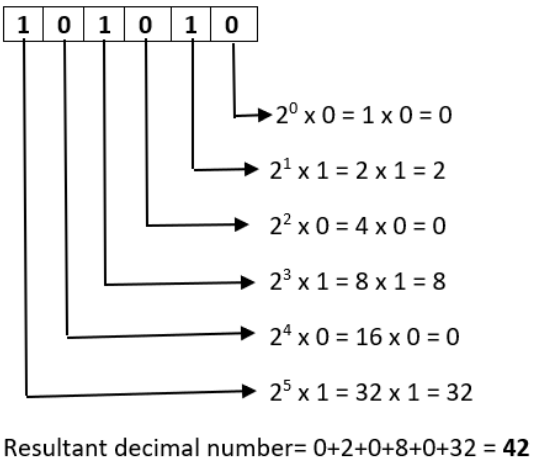
Bit Wise Operator

Binary Conversion process



Binary to Decimal





Bitwise Operators

Operators	Meaning of operators

&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
~	Bitwise complement
<<	Shift left
>>	Shift right

Example 1

What is the output the program

```
#include <stdio.h>
  int main () {
      int x = 5, y=17, z
      z = x&y;
      printf("%d", z);
  }
```

(A) 1 (B) 21

(C) 2 (D) -6

Answer

A

Example 2

What is the output the program

```
#include <stdio.h>
  int main () {
      int x = 5, y=17, z
      z = x|y;
      printf("%d", z);
  }
```

(A) 1 (B) 21

(C) 2 (D) -6

Answer

B

Example 3

What is the output the program

```
#include <stdio.h>
  int main ()  {
        int x = 5, y=17, z
        z = x ^ y;
        printf("%d", z);
  }
```

(A) 1 (B) 21

(C) 20 (D) -6

Answer

B

Example 4

What is the output the program

```
#include <stdio.h>
  int main ()  {
        int x = 5, z
        z = ~x;
        printf("%d", z);
  }
```

(A) 1 (B) 21

(C) 20 (D) -6

Answer

D

Example 5

```
#include<stdio.h>
      int main(){
```

```
        char a = 8;
        int k;
        k =a<<3;
        printf("%d", k);
        return 0;
    }
```

(A) 1 (B) 64

(C) 20 (D) -6

Example 5

```
#include<stdio.h>
    int main(){
       char a = 64;
       int k;
       k =a>>3;
       printf("%d", k);
       return 0;
    }
```

(A) 1 (B) 21

(C) 8 (D) -6