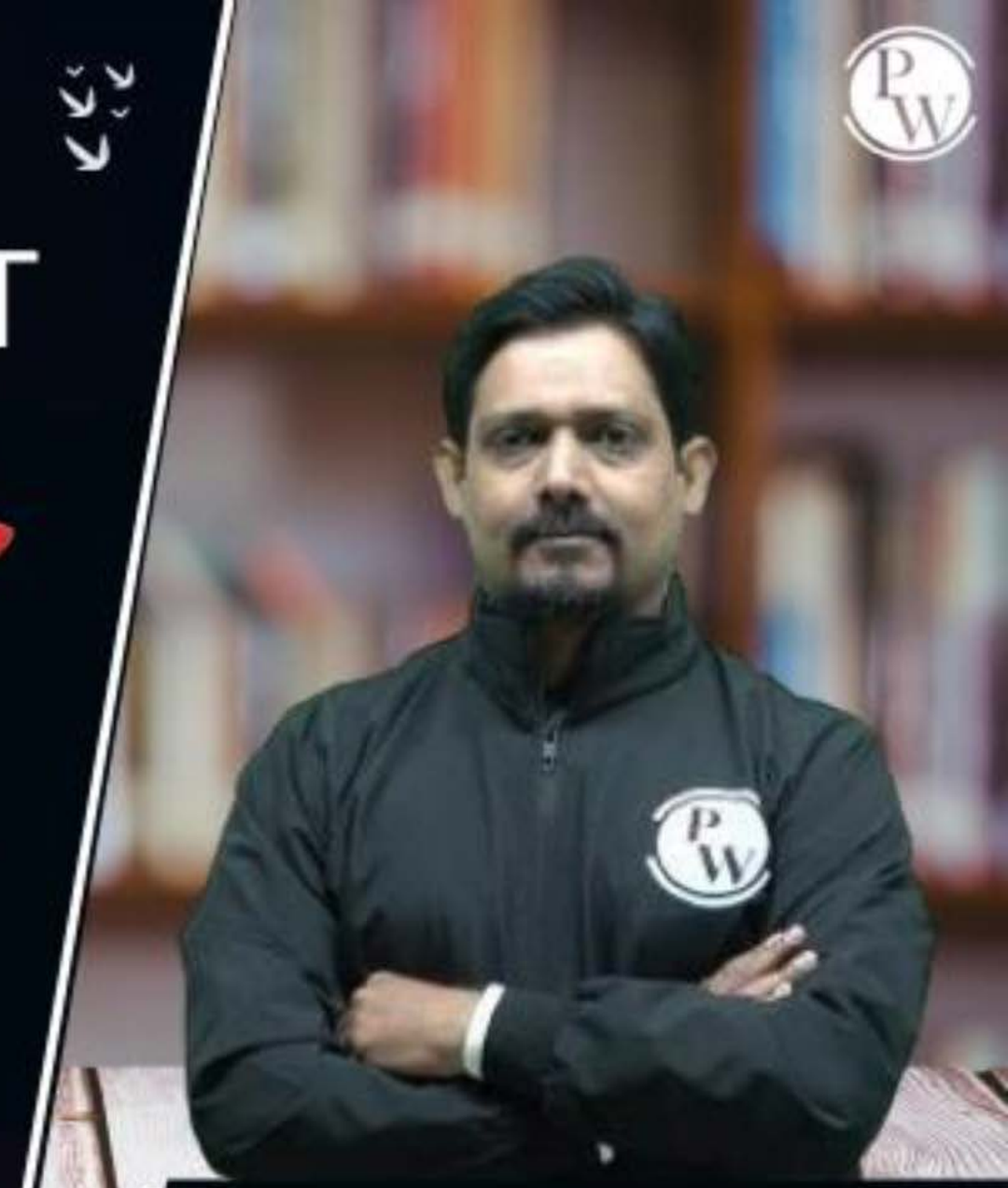
Computer Science & IT

C programming

Data Types & Operator

Lecture No. 03



By- Abhishek Sir

Recap of Previous Lecture

topic — Arithmetic expression

topic — precendence & Associativity

topic — ++ , -- post / pre

topic

topic

Slide

topics to be Covered

- topic — Relational operator
- topic — Logical operator (short circuit code)
- topic — Bit wise operator
- topic
- topic

Slide

y = ++a, first Increment
then use in any expression

y = a++, old value used
then increment.

&a

!*&a

1. postfix
2. prefix Right to left
3. * /, %
4. + -
5. >> <<
6. > >=
   < <=
7. == !=
8. &
9. ^
10. |
11. &&
12. ||
13. ?: Right to Left
14. = *= Right to left
15. ,

post

poe

Arithmetic

→ >> <<

Relational

Bitwise

logical

T

Ass

C

PPARBLTEC

Topic: Relational Operator

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

Toipc: Relational Operator Example

```
#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 ;
}
```

0
0
0
1
1
1
0
1

Trichotomy

Slide

Toipc: Relational Operator Precedence

$> \; >=$ greater, Left to Ri

$< \; <=$ less . . . .

$== \; !=$ lesser precedence Left to Right

Slide

Topic: Relational Operator Precedence

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

Topic: Relational Operator Example

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

Which of the following assignment value printed is 0?

A. x= 50 , y=20 , z=20 → 1

B. x= 50 , y=100 , z=0; → 0

C. x= 10 , y=5 , z=10 → 1

D. x= 100 , y=5 , z=100; → 1

50>20<20

1<20

(B) 50>100<0

0<0 –

(C) 10>5<10

1<10 := 1

(D) 100>5<100

1<100 – (1)

Slide

Topic: Relational Operator Example

```
#include<stdio.h>
int  main() {
    int x = 1, y = 50 , z = 50;
      printf("%d\n", (x==y<=z)+(x+y>=z) + (x!=y-z ));
}
```

1 + 1

1 == 50 <= 50 ↓ + 1 != 0

1 == 1 51 >= 50

The value printed by the program is 3

Slide

Toipc:Question

```
#include <stdio.h>
    int main ()  {
    int a =5+5!=(6<4)+10;
    int b =5+5==(6<4)+10;
    int c =5!=5>3!= (6<4)+10;
     printf("%d", a+b+c);
    }
```

10 != 0+10

10 != 10

10 == 10

5 != 1 != 10

1 != 10 : 1

0 + 1 + 1

The output of the program 2

Slide

Toipc: Logical Operator

Logical operator

a	b	a&&b
0	0	0
0	1	0
1	0	0
1	1	1

a	b	a||b
0	0	0
0	1	1
1	0	1
1	1	1

a	!a
0	1
1	0

Slide

Toipc: Logical Operator (AND) &&

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

Logical AND

20>10 && 30>10

1 && 1 = 1

Slide

Toipc: Logical Operator (OR) ||

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

20>5 || 5>10

1 || [ ]

^ - XOR Bitwise

& - Bitwise AND

| Bitwise OR

Slide

Zero && Anything

The expression will evaluate to Zero Regardless of second expression. Hence second expression Not evaluated.

1 || Anything

Do we know the result of expression??

if first expression is 1

we dont required to evaluate second expression.

short circuit code

Toipc: Logical Operator (NOT) !

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

!(20<10)

!0 = 1

!10

0

!-1

0

! Nonzero is zero

! zero is 1

Slide

Topic: Question

```
#include<stdio.h>
int main(){
        int x = 1, a;

        a = x || ++x
        printf("%d", x);


        return 0;
}
```

The value printed is _____1_____

1 || ++x

↑ this expression will short

Slide

Topic: Question

```
#include<stdio.h>
int main(){
        int x = 0, y = 0, a;

        a = x && ++y;
        printf("%d %d", x, y);

        return 0;
}
```

a = 0 && ++y
↑ short

A. 1 1
B. 1 0
C. 0 1
D. 0 0

0 0

Slide

Topic: Question

```
#include<stdio.h>
int main(){
        int x = 0, y = 0, a;

        a = x && ++y;
        printf("%d %d", x, y);


        return 0;
}
```

A. 1 1
B. 1 0
C. 0 1
D. 0 0

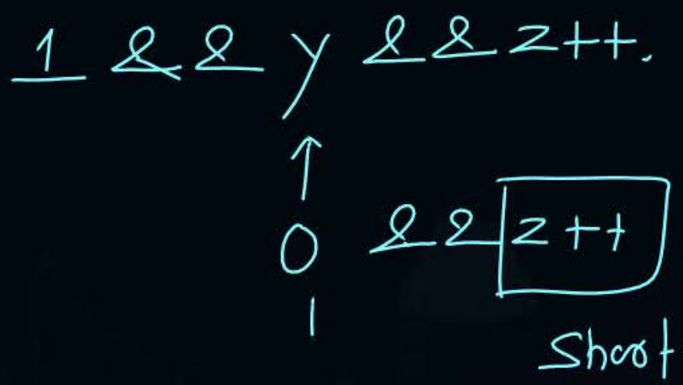
Slide

Topic: Question

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

The value printed is ?

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



Slide

Topic: Question

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

The value printed is ?

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

Slide

Topic: Question

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

x && (++y || z++)

1

++y || z++

1 || Anything

↑ Shoot

1 1 0

A. 1 0 0
B. 1 1 0
C. 1 0 1
D. 1 1 1

Slide

Topic: Question

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

A. 1 0 0
B. 1 1 0
C. 1 0 1
D. 1 1 1

Slide

Topic: Question

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

A. -1 1 0
B. -1 1 1
C. -1 0 1
D. 1 1 1

X = -1

Y = 0

Z = 1

X = -1

partial short circuit

x-- && y++ || z++

1

0 && y++ || z++

↑ short

partial short

|| 0

Slide

Topic: Question

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

partial Short circuit

A. -1 1 0
B. -1 1 1
C. **-1 0 1** ✓
D. 1 1 1

Slide

Bit-wise Operator

Operators	Meaning of operators
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR
~	Bitwise complement
<<	Shift left
>>	Shift right

poefix ~

*/ %

+ -

>> <<

> >=

< <=

== !=

&

^

|

Slide

Bit-wise Operator

What is the output the program

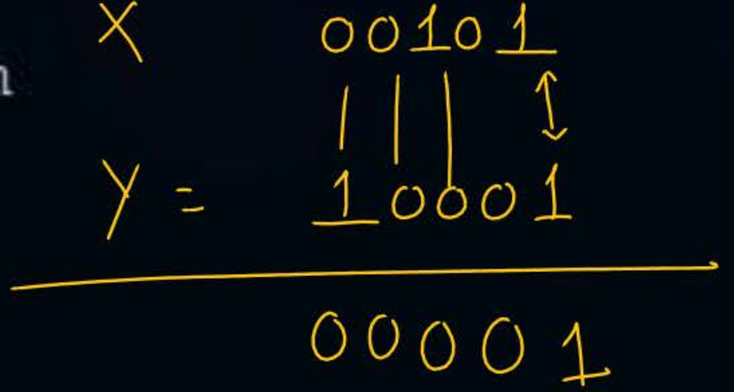
```
#include <stdio.h>
    int main ()  {
      int x = 5, y=17, z
      z = x&y;
      printf("%d", z);
 }
```

(A) 1 ✓

(B) 21

(C) 2

(D) -6



Slide

Bit-wise Operator

What is the output the program

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

$11\underline{1}011$ ← (−5)

−5 → 111011 (2's complement)

17 → 010001

010001 → Sign bit

$$\begin{array}{r} 111011 \\ 0\,1000\underline{1} \\ \hline 0\,10001 \end{array} \quad (17)$$

Slide

Bit-wise Operator

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

x 00101
y 10001

10101

16 + 4 + 1 = 21

Slide

Bit-wise Operator

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

^ - exclusive-OR

a	b	a^b
0	0	0
0	1	1
1	0	1
1	1	0

X = 00101
Y = 10001
10100

16 + 4 + 0 = 20

Slide

Bit-wise Operator

What is the output the program

```
#include <stdio.h>
    int main ()  {
        int x = 15, z
       z = ~x;
       printf("%d", z);
    }
```

(A) 1 (B) 21

(C) -16 (D) -6

-16

$= -(-16+1)$

$= -(-15) = 15$

16

$-(16+1)$

$= -17$

X	$\sim X$
5	-6
10	-11
20	-21
50	-51
100	-101

X	$\sim X$
-5	4
-10	9
-20	19
-50	49
-100	99

~ 0101

1010

$-2^3 + 2^1$

$-8+2 = -6$

$$\sim X = -(X+1)$$

$X = 5 = -(5+1) = -6$

$X = -10 = -(-10+1) = -(-9) = 9$

Bit-wise Operator

What is the output the program

```
#include <stdio.h>
    int main ()  {
        int x = 16, z
       z = ~x;
       printf("%d", z);
    }
```

(A) -17 (B) 21

(C) 20 (D) -6

Bit-wise Operator

What is the output the program

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

(A) 4 (B) 21

(C) 20 (D) 6

Bit-wise Operator

What is the output the program

```
#include <stdio.h>
    int main ()  {
        int x = -10, z
       z = ~x;
       printf("%d", z);
    }
```

(A) 1

(B) 21

(C) 9

(D) -6

Bit-wise Operator

```
#include <stdio.h>
    int main ()  {
      int x = 5,y=24, z=10;
          z = x & y | z ;
       printf("%d", z);
    }
```

(A) 10

(B) -5

(C) 24

(D) 11

x 00101

y 11000

00000

x&y = 0

00000
01010
01010

Slide

Bit-wise Operator

```
#include <stdio.h>
    int main ()  {
      int x = 5,y=24, z=10;
          z = x & y | z ;
       printf("%d", z);
    }
```

(A) **10**

(B) **-5**

(C) 24

(D) 11

Bit-wise Operator

```
#include <stdio.h>
    int main ()  {
      int x = 5,y=24, z=10;
          z = x | y ^ z ;
       printf("%d", z);
    }
```

Output of the above program is

(A) 10

(B) -5

(C) 23

(D) 11

x : 00101

y : 11000

z : 01010

y^z 10010

x 00101

1011 1 (23)

Slide

Question

Consider the following ANSI C program.

```
#include <stdio.h>
int main()  {
   int i, j, count;
   count=0;
   i=0;
   for (j=-3; j<=3; j++)   {
     if (( j >= 0) && (i++))
          count = count + j;
  }
  count = count +i;
  printf("%d", count);
  return 0;
}
```

Which one of the following options is correct?

(A)The program will not compile successfully

(B)The program will compile successfully and output 13when executed

(C)The program will compile successfully and output 8 when executed

(D)The program will compile successfully and output 10 when executed

Slide

Question

Consider the following ANSI C program.

```
#include <stdio.h>
int main()  {
   int i, j, count;
   count=0;
   i=0;
   for (j=-3; j<=3; j++)   {
      if (( j >= 0) && (i++))
            count = count + j;
   }
   count = count +i;
   printf("%d", count);
   return 0;
}
```

	i=0	i+1	count
j = -3	—	x	
j = -2	—	x	
j = -1	—	x	
j = 0	—	1	0
j = 1	—	2	1
j = 2	→	3	3
j = 3		4	6

0 >= 0

1

6 + 4 : 10

Slide

2 mins Summary

$\sim x = -(x+1)$

Topic — Relational operator

Topic — Logical operator, short circuit code

Topic — Bitwise operator

Topic

Topic

$0\boxed{101}\ (5 \quad \boxed{2^n - 1 - 5}$

$010\ \underline{(2)} \quad 2^3\ \underline{8-1}-5=2$

$0111 = (7) \quad 7-7=0$

$000 \quad 0$

$0100 \quad - \quad 7-4=3$

011

Slide

THANK - YOU