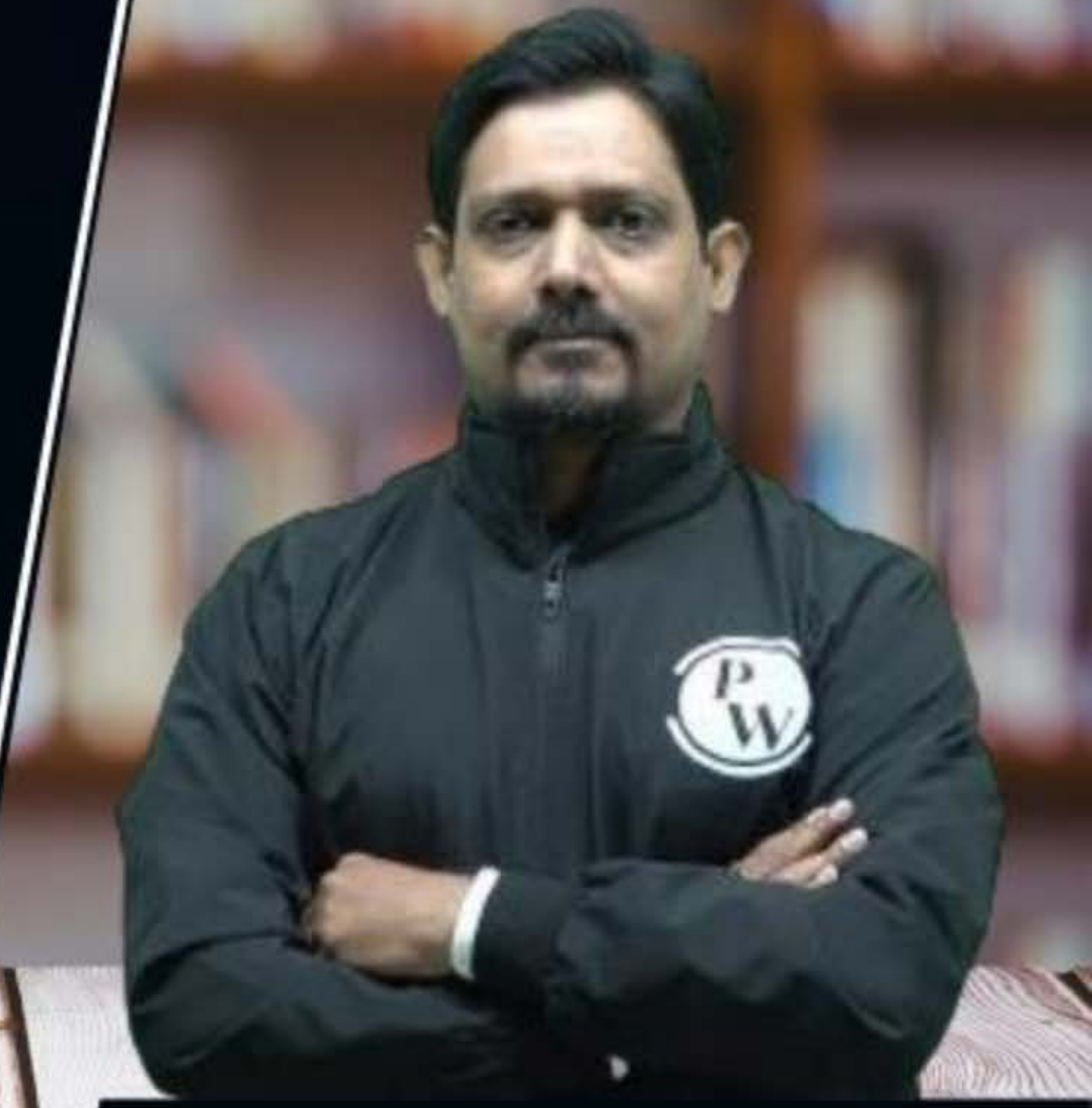
Computer Science & IT

Programming in C

Control Flow Statement

Lecture No. 01



By- Abhishek Sir

Recap of Previous Lecture

Topic Logical operator

Topic Bitwise Operator

Topic

Topic

Topic

Slide

Topics to be Covered

Topic

Topic

Topic

Topic

Topic

Contool flow statement

Slide

Bit-wise Operator

What is the output the program

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

(A) 1

(B) 21

(C) -11

(D) -6

$\boxed{|x| - 1}$

$|-10| - 1 = 10 - 1$
$= 9$

X is a Negative No

$-(x+1)$

X is a Negative

$\boxed{\text{Negative}}$

$X = -10 = -(-10+1)$
$\sim x = 9$

$X = -102 - (-102+1)$
$\sim x = 101$

$X = -510 = -(-510+1)$
$\sim x = 509 \quad = 509$

Slide

Bit-wise Operator

What is the output the program

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

(A) 1

(B) 21

(C) -11

(D) -6

$\boxed{|x|-1}$

$\boxed{\text{Negative}}$

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

$|x|-1$

$$\begin{array}{ccccc} 1 & 0 & 1 & 1 & 0 \\ 4 & 3 & 2 & 1 & 0 \end{array}$$

$-2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1$

$-16+4+2 = -10$

$$\begin{array}{ccccc} 1 & 0 & 1 & 1 & 0 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 0 & 1 & 0 & 0 & 1 \end{array}$$

$(+9)$

Slide

Bit-wise Operator

```
#include<stdio.h>
   int main(){
         char a = 8;
         int k;
         k =a<<3;
         printf("%d", k);
         return 0;
   }
```

Left shift

(A) 1 (B) 21

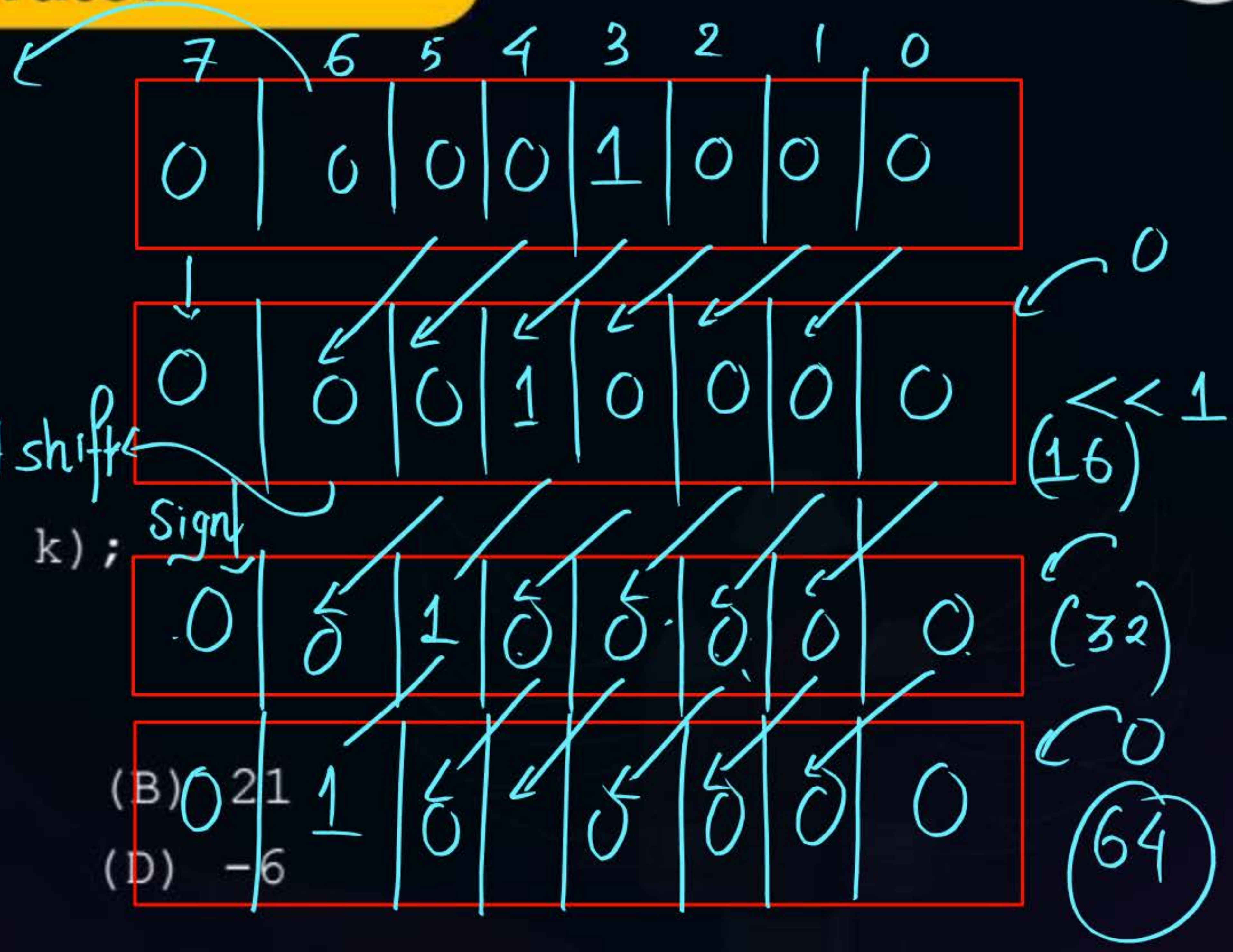
(C) 8 (D) -6

$$\boxed{a<<3}$$

$$= a\times 2^3$$

$$8\times 2^3$$

$$= 8\times 8 = 64$$



* One Left shift equivalent to multiply by 2

overflow may occur.

$a << k$

$= a \times 2^k$

overflow

* one Right shift equivalent to divide by 2

+ ve No. will become '0'

$a >> K$

$a \times 1/2^K$

value may become 0

Bit-wise Operator

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

int a = -15;

Negative No

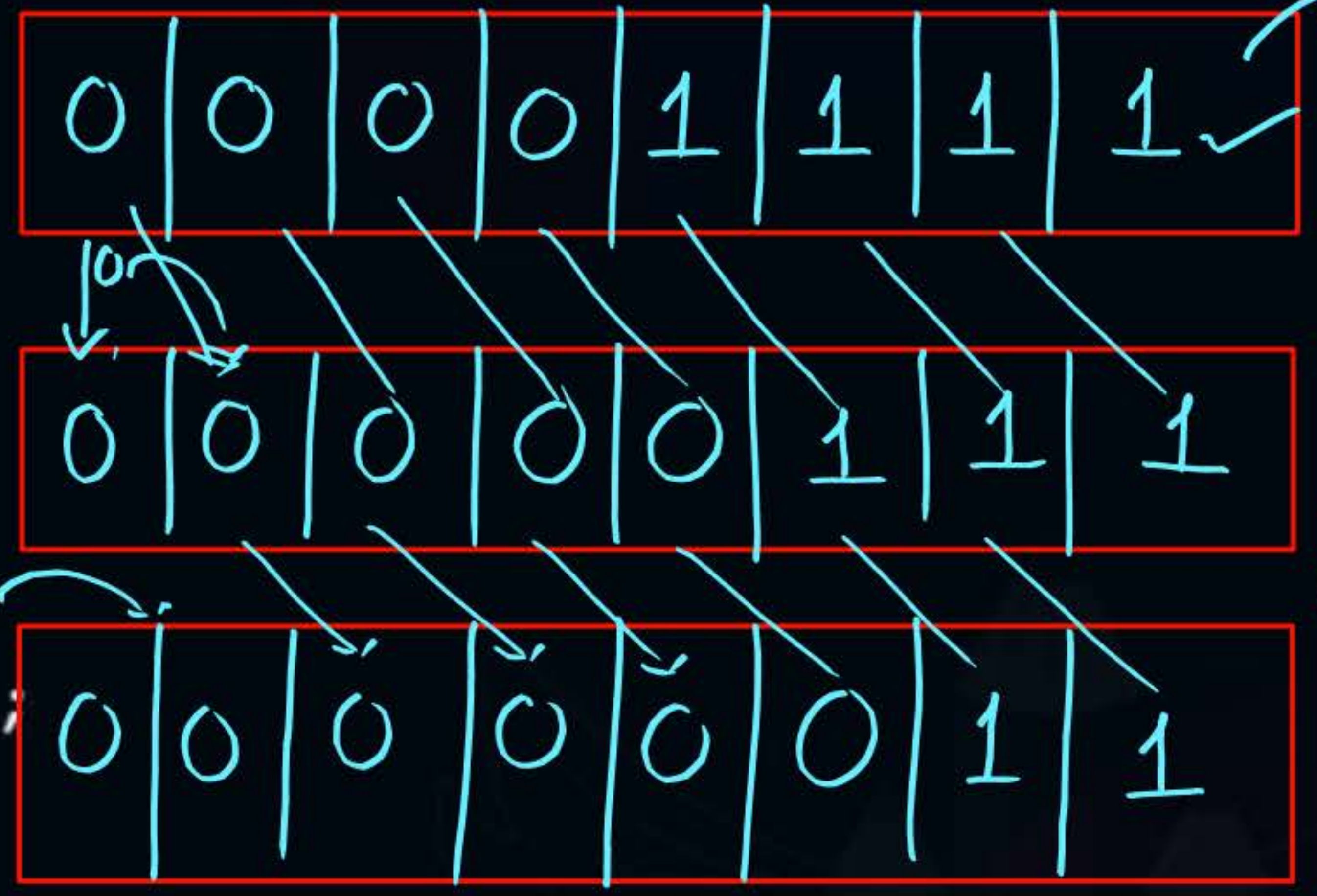
$a << 3$

$a >> 4$

(A) 1 (B) 21

(C) 8 (D) -6

$\underline{a = 15}$ Binary



$\underline{15}$

$\underline{a>>1}$

$\boxed{7}$

$a>>2$

$\boxed{3}$

$15 \times \frac{1}{2} = 7$

$7 \times 1/2 = 3$

Slide

Print() Return value

```
#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

Slide

Toipc: Scope of a variable

Scope : Scope defines visibility and alive ness of a variable.

$\uparrow$ usable

```
int main() {
    Local variable

}
```

Scope of variable

whenever we defind opening & closing bracket we define Scope of variable

after closing bracket variable deallocated (dead)

Slide

Toipc: Scope of a variable

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.

Slide

Toipc:Scope of variable

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

}
```

Slide

Toipc:Scope of variable

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

variable deallocate

Slide

Toipc:Scope of variable

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

   return 0 ;
}
```

Slide

Toipc: Question

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

outside of main

Outside main a variable declared called as global variable (Storage)

global visibility is for every function access

Life : during entire program

Slide

Toipc: Question

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

Static Scooping Rule :

If variable declaration is not found within the block then declaration of variable searched in next upper block. This process repeated until declaration of variable is found if No declaration present then it gives Error.

Slide

Toipc: Question

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

Static Scooping Rule :

If variable declaration is not found within the block then declaration of variable searched in next upper block. This process repeated until declaration of variable is found if No declaration present then it gives Error.

print 20

Slide

Toipc: Question

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

Static Scooping Rule :

If variable declaration is not found within the block then declaration of variable searched in next upper block. This process repeated until declaration of variable is found if No declaration present then it gives Error.

print 30

Slide

Toipc: Question

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

Static Scooping Rule :

If variable declaration is not found within the block then declaration of variable searched in next upper block. This process repeated until declaration of variable is found if No declaration present then it gives Error.

print 40

Slide

Toipc: Question

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

Static Scooping Rule :

If variable declaration is not found within the block then declaration of variable searched in next upper block. This process repeated until declaration of variable is found if No declaration present then it gives Error.

then error

Slide

Print() Return value

```
print(" Hello world");

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

string

Name of variable

Slide

Print() Return value

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

[✓ (A) ABCD4] (B) 4

(C) ABCD 4 (D) ABCD

4

printf Return value

No. of charactes printed

Icluding space

integer value

No

ABCD4

Slide

Print() Return value

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

ABCD
4

int a=4;
int b=3;
printf("%d %d", a++,++b);

what will be the output

a=4;
b=3;

printf("%d %d", ++a, a++)

4

$\boxed{4\ 4}$

No Rule in C Languag in what order parameter passed

Slide

```
int a=14, b=13

printf("%d", printf("%d%d", a,b))
```

$\underline{1413}4$

$\boxed{14134}$

Print() Return value

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

```
ABCD
4
```

Slide

Print() Return value

```
#include <stdio.h>
  int main(){
     int a=1000;
     printf("%d",printf("\n%d",a));
     return 0;
}
```

Output of the program is _____

(A) 1005

(B) 10005

(C) 1000

(D) 1000
    5

Slide

Ternary Operator

unary $-$, $++$, $--$, $!$, $\sim$

only one Ternary operator

Relational Expression ?

Conditional operation

$$\boxed{\text{Expr 1 ? Statement}_1 : \text{statement}_2}$$

Nonzero — zero

True — false

statement$_1$ — statement$_2$

Slide

```
int a=10

14>13 ? a++ : ++a ;
printf("%d", a);

11
```

```
int a=10, b=14

a==4 ? printf("%d",a) :
       printf("%d", b)

0

a=4

discuss

14
```

Ternary Operator

```
#include <stdio.h>
int main()
{
   int a;
   a = 10>7?10:20;
   printf("%d", a);
   return 0;
}
```

[✓ (A) 10]
(B) 20
(C) 1
(D) 0

Slide

Ternary Operator

```
#include<stdio.h>
int main(){
      int x=3, y=4, z=4;
      printf("%d", (z>=y>=x?100:200));
      return 0 ;
}
```

(a) 100 [✓ (b) 200] (c) 0 (d) 1

$(4 >= 4) >= 3$

$(1 >= 3)$

Slide

Control Flow Statement

Sequential Flow of Execution:

1. if
2. if else
3. Nested
4. switch
5. Loop: for, while, do while
6. Break, continue goto

Slide

2 mins Summary

Topic

Topic

Topic

Topic

Topic

Left shift & Right shift

Scope of variable

printf return

Ternay operator

Slide

THANK - YOU