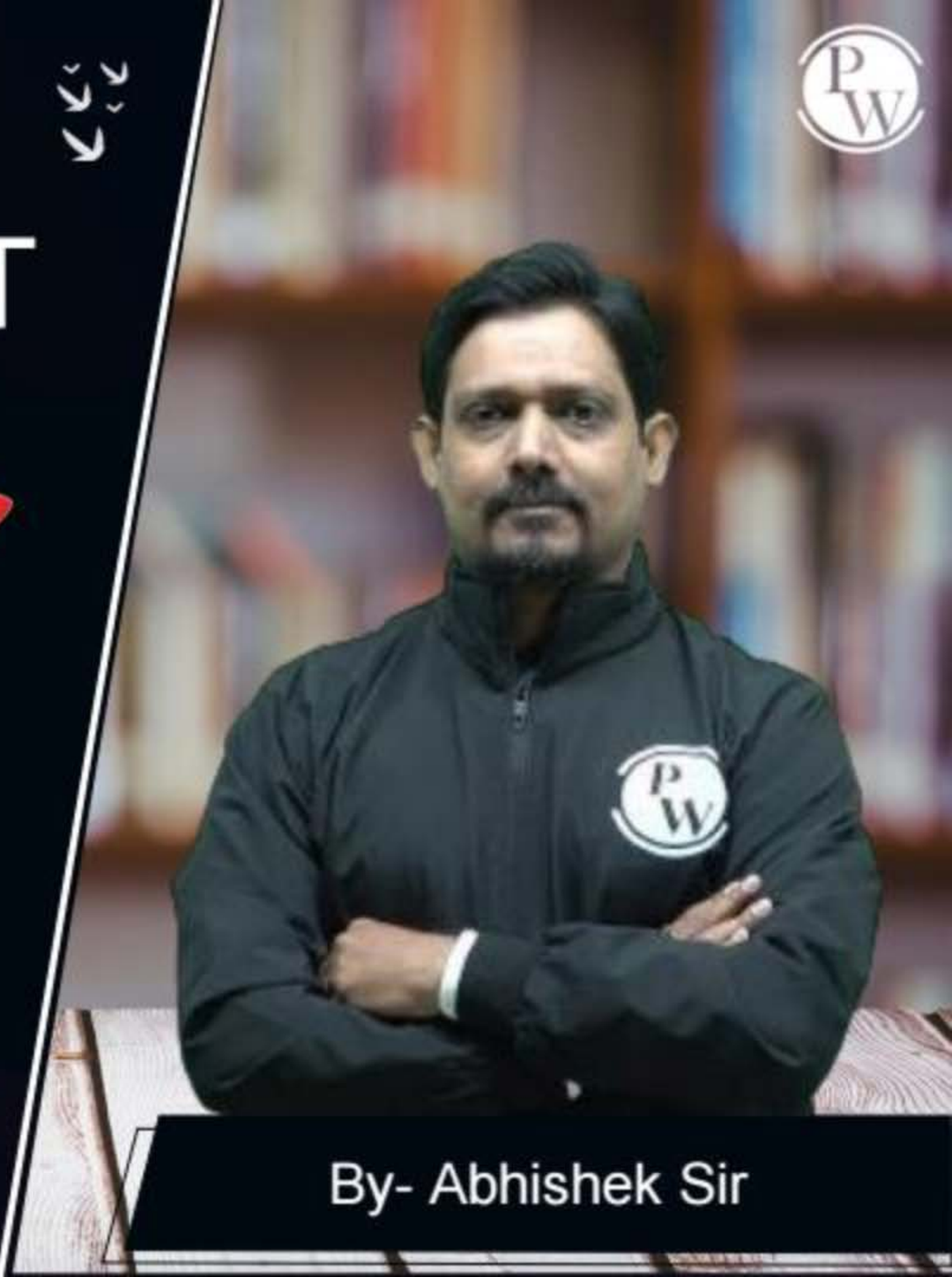
Computer Science & IT

C programming

Array & Pointers

Lecture No. 01



By- Abhishek Sir

Recap of Previous Lecture

Topic — Tower of Hanoi

Topic — Register, extern

Topic — pointer.

Topic

Topic

Slide

Topics to be Covered

- Topic — pointer
- Topic — Double pointer
- Topic — Call by reference
- Topic — Swap function
- Topic — practice problem

Slide

2015

1M

```
int fun( int n)
{
  int x=1, k;
  if (n==1) return x;
  for (k=1; k<n, ++k)
      x = x + fun(k) * fun(n-k)
  return x;
}
fun(5)
```

value calulale

Smaller to Large

fun(1) = 1

fun(2) $\overset{n=2}{=}$ x = 1 + fun(1) * fun(1) (n-1)

x = 1+1 = 2

fun(3) = $\overset{n=3}{}$ k=1, x = 1 + fun(1) * fun(2)

= 1 + 1*2 = 3

k=2, x = 3 + fun(2) * fun(1)

3 + 2*1 = 5

2015

1M

```
int fun( int n)
{
int x=1, k;
if (n==1) return x;
for (k=1; k<n, ++k)
    x = x + fun(k) * fun(n-k)
return x;
}
fun(5)
```

value calulale

Smaller to Large

$\hat{}$ n=4

fun(4) = k=1, 1 + fun(1) * fun(3)

1 + 1 * 5 = 6

k=2, 6 + fun(2) * fun(2)

6 + 2 * 2 = 10

k=3, 10 + fun(3) * fun(1)

10 + 5 * 1 = 15

2015

1M

```
int fun( int n)
{
int x=1, k;
if (n==1) return x;
for (k=1; k<n, ++k)
    x = x + fun(k) * fun(n-k)
return x;
}
fun(5)
```

value calulale

Smaller to Large

fun(5): n=5 $x = 1 + fun(1) * fun(4)$

$1 + 1 * 15 = 16$

$k=2$, $x = 16 + fun(2) * fun(3)$

$16 + 2 * 5 = 26$

$k=3$, $x = 26 + fun(3) * fun(2)$

$= 26 + 5 * 2 = 36$

$k=4$ $x = 36 + fun(4) * fun(1)$

$= 36 + 15 * 1 = 51$

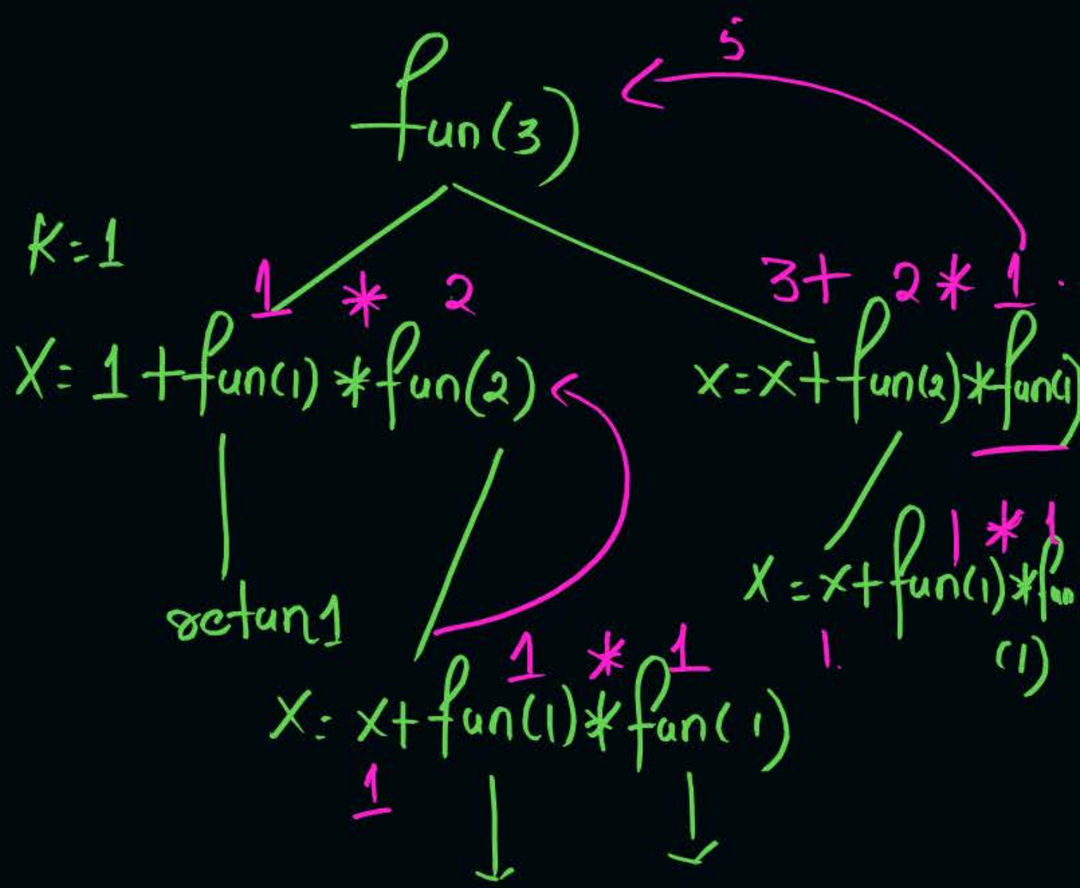
2015

1 M

```
int fun( int n)
{
  int x=1, k;
  if (n==1) return x;
  for (k=1; k<n, ++k)
      x = x + fun(k) * fun(n-k)
  return x;
}
fun(5)
```

value calulale

K=1



Pointers

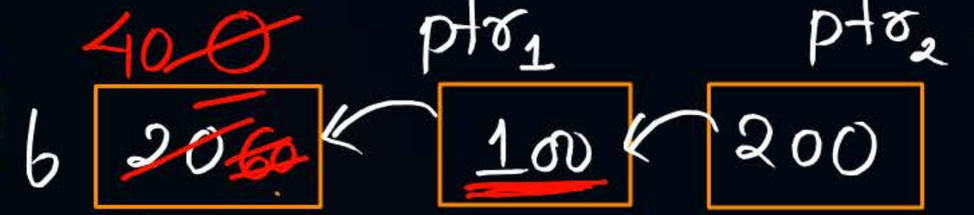
pointer is a variable that can hold Address of another variable of similar data type.

```
int a = 10;
int * ptr = &a;
```

assign. `ptr = &a;`

Pointers

```
#include<stdio.h>
int a = 10;
int main() {
    int *ptr1, **ptr2;
    int b = 20;
    ptr1 = &b;
    ptr2 = &ptr1;
    a = a+10;
    b = b+40;
    *ptr1 = *ptr1 - b;
    **ptr2 = a+20;
    printf("%d", b);
    return 0;
}
```



Double pointer

Address of int pointer

60 − 60

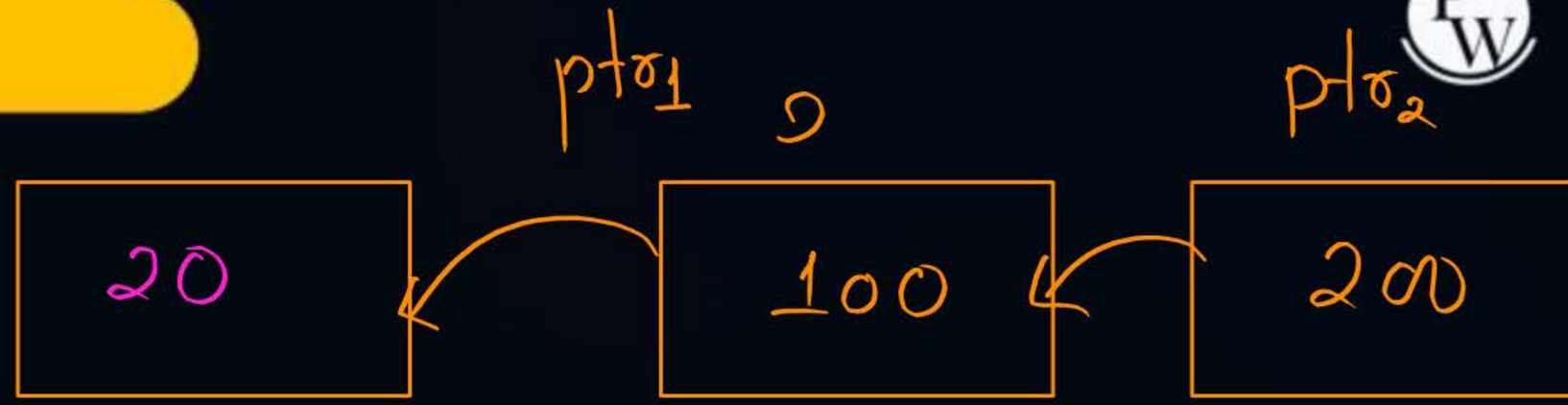
20+20

a 20 ✓

40

Slide

Pointers



$**ptr_2$

$*ptr_1$

int b = 20

variable b

$**ptr_2$

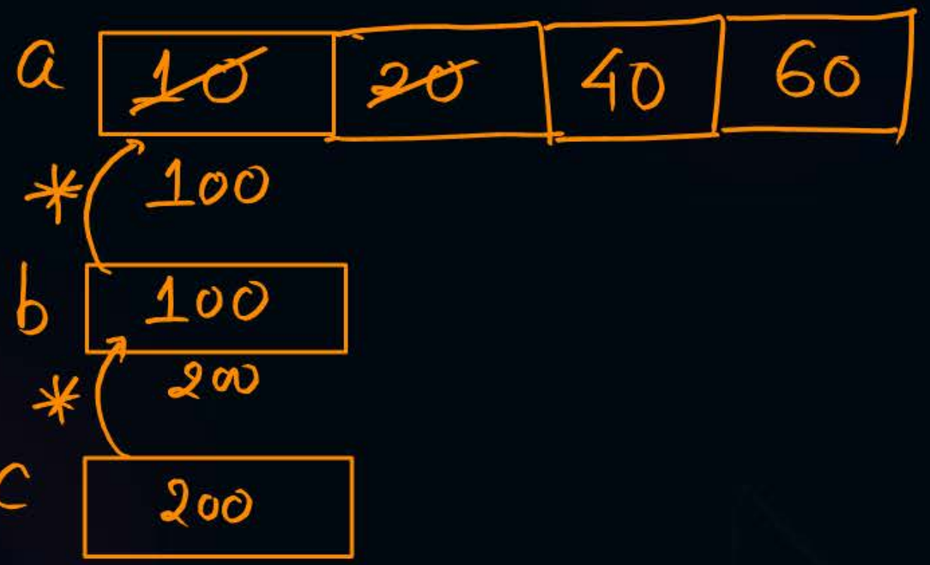
$ptr_2 = \&ptr_1$

Pointers

```
#include<stdio.h>
int main (){
        int a, *b;
        a = 10;
        b = &a;
        a =a+10;
        *b = *b+20;
        printf("%d", a);
        return 0;
}
```

Pointers

```
#include<stdio.h>
int main (){
        int a, *b, **c;
        a = 10;
        b = &a;
        c = &b;
        a =a+10;
        *b = *b+20;
        **c = **c+20;
        printf("%d", a);
        return 0;
}
```



Slide

Size of Pointer

```
include <stdio.h>

int main(){
    int *p;
    char *p1;
    float *p2;

    printf("%lu", sizeof(p));
    printf("%lu", sizeof(p1));
    printf("%lu", sizeof(p2));

    return 0;
}
```

GATE 2017

#Q Consider the following function implemented in C:

```
void printxy (int x, int y) {
{ int *ptr ;
  x = 0;
  ptr = &x;
  y = * ptr;
  * ptr = 1;
  printf ("%d, %d," x, y);
}
```

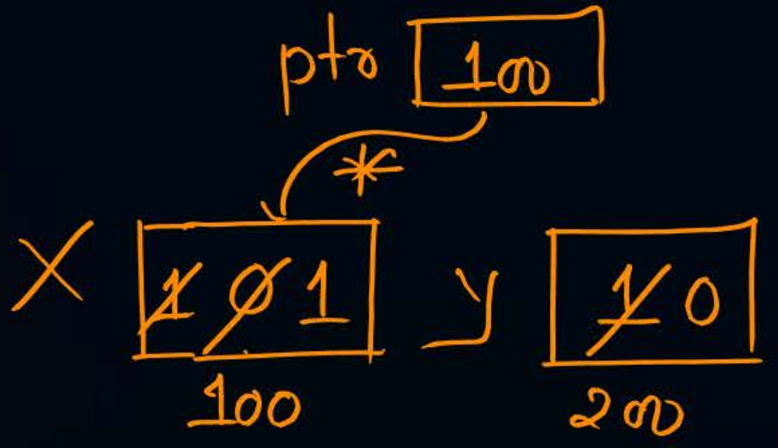
The output of invoking printxy (1, 1) is

(A) 0,0

(B) 0,1

(C) 1,0

(D) 1,1



int *ptr

ptr = &x;

y = *ptr

1 0

Slide

GATE 2017

#Q Consider the following function implemented in C:

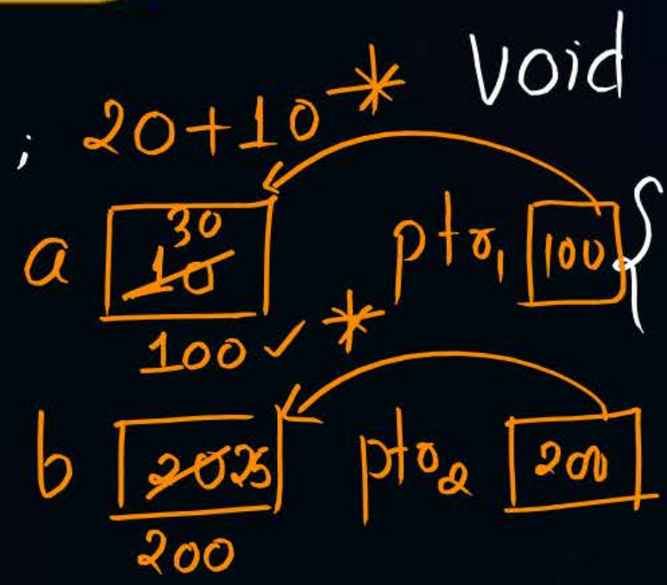
```
void printxy (int x, int y) {
        int *ptr ;
        x = 0;
        ptr = &x;
        y = * ptr;
        * ptr = 1;
        printf ("%d, %d," x, y);
}
```

Slide

Call by reference

```
#include<stdio.h>
void fun(int *, int *);
int main(){
    int a=10, b=20;
    fun(&a, &b);
    printf("%d %d", a, b);
    return 0;
}
```

30, 25



ptr1 = 100 ptr2 = 200

```
void fun(int *ptr1, int *ptr2)
{
    *ptr1 = *ptr2 + *ptr1;
    *ptr2 = *ptr2 - 5;
}
```

20+5 = 25

%p - Hexdecimal

%a - unsigned

Slide

Call by reference

```
void fun(int*, int*);
```

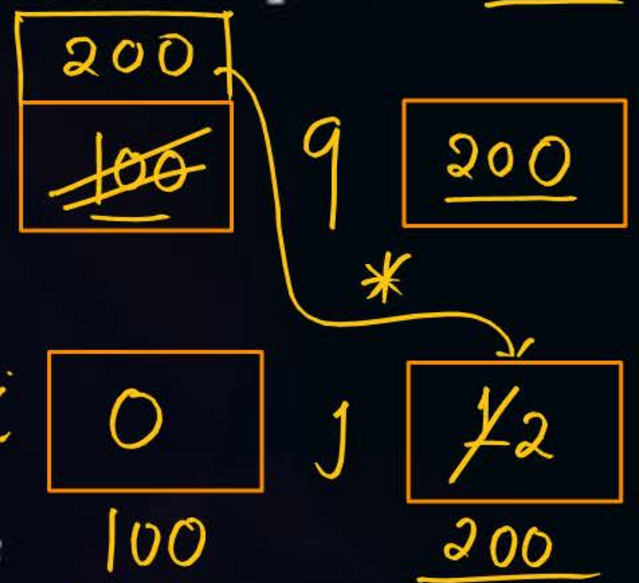
Slide

Call by reference

#Q What does the following program print?

```
#include<stdio.h>
void f(int *p, int *q) {
    p=q;
    *p=2;
}
int i=0, j=1;
int main() {
    f(&i, &j);
    printf("%d %d\n", i, j);
    return 0;
}
```

(A) 2,2
(B) 2,1
(C) 0,1
(D) 0,2



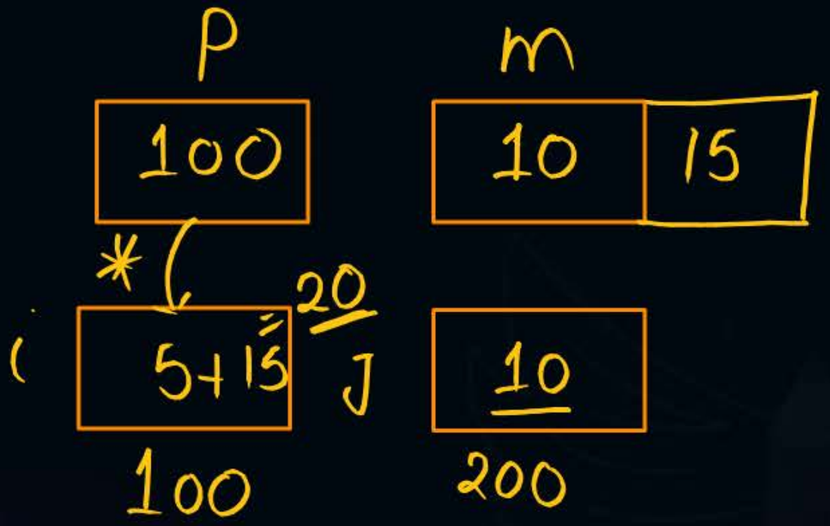
Slide

GATE 2016

The value printed by the following program is _____.

```
void f (int * p, int m) {
     m = m + 5;
    *p = *p + m;
     return;
}
void main () {
 int i=5, j=10;

 f (&i, j);
 printf ("%d", i+j);
}
```



30

Slide

GATE 2008

What is printed by the following C program?

```
int f(int x, int *py, int **ppz){
      int y, z;
     **ppz += 1; z = **ppz;
     *py += 2; y = *py;
     x += 3;
     return x+y+z;
}
void main(){
     int c, *b, **a;
     c = 4; b = &c; a = &b;
     printf("%d", f(c, b, a));

}
```

HW question

Slide

3 Types of Swap

Local a,b, swap

```
void Swap(int a, int b)
{
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

Address Swap

```
void Swap(int *a, int *b){
    int *temp;
    temp = a;
    a = b;
    b = temp;
}
```

a 100 b 200

a 200 b 100

temp = 100

a : 200

b = 100

Actual Swap

```
void Swap(int *a, int *b){
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
```

Slide

3 Types of Swap

a | 10 | 100 b | 20 | 200

int a = 10, b = 20

Swap(&a, &b);

ptr1 | 100 ptr2 | 200

void Swap(int *ptr1, *ptr2) {

ptr1 | 200 ptr2 | 100

GATE 2016

Consider the following C program.

```
#include<stdio.h>
void mystery(int* ptra, int* ptrb){
    int *temp;
    temp= ptrb;
    ptrb= ptra;
    ptra= temp;
}
```

Address Swap

```
int main(){
    int a=2016, b=0, c=4, d=42;
    mystery(&a, &b);
    if (a<c)
        mystery(&c, &a);
    mystery(&a, &d);
    printf("%d\n", a);
    }
```

The output of the program is 2016 .

Slide

2 mins Summary

Topic — Double pointer

Topic — Call by reference

Topic — Swap function

Topic — practice

Topic — 2015 question

Slide

THANK - YOU