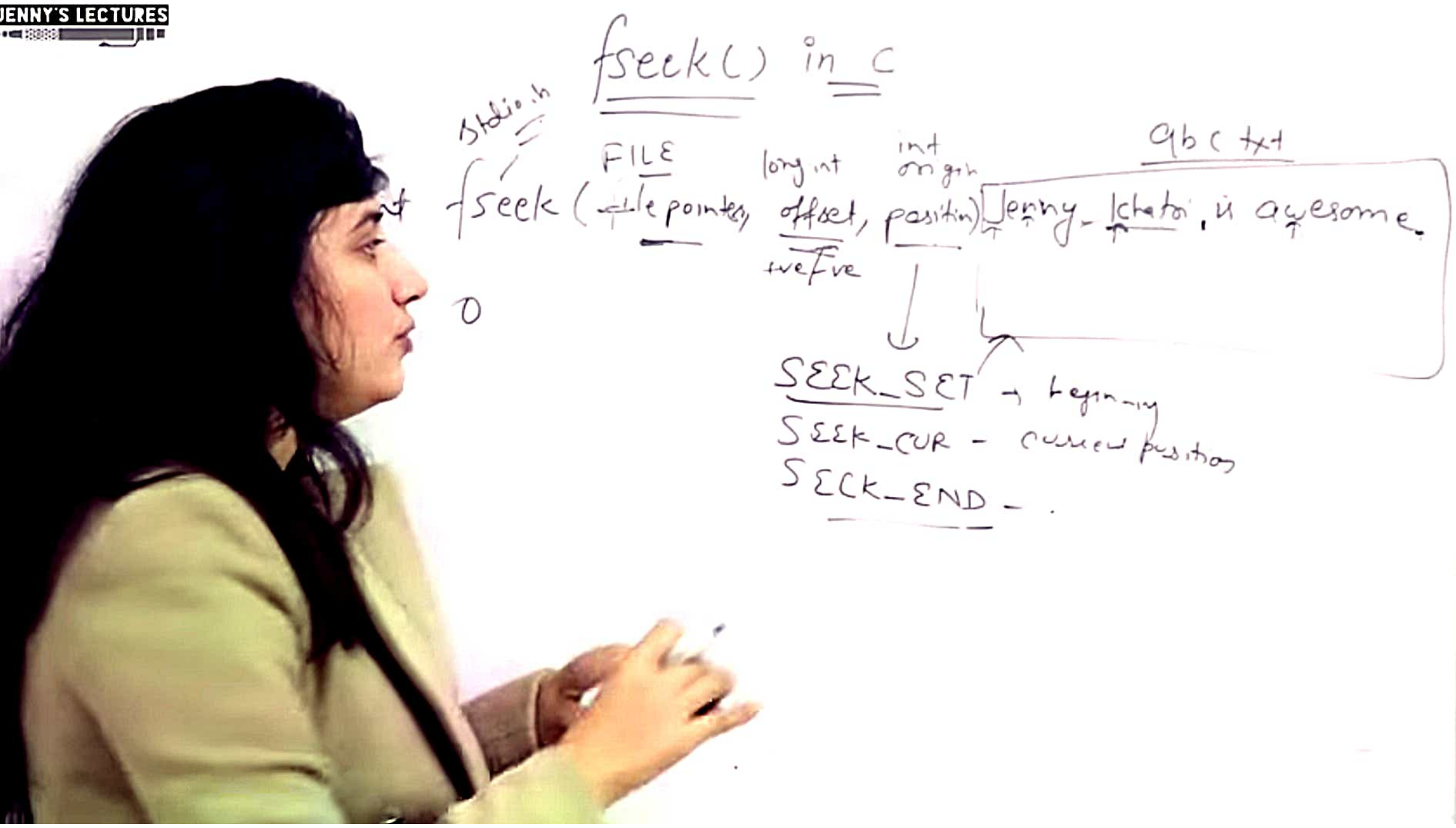


fseek() in C

stdio.h

FILE long int int origin

int fseek(file pointer, offset, position)

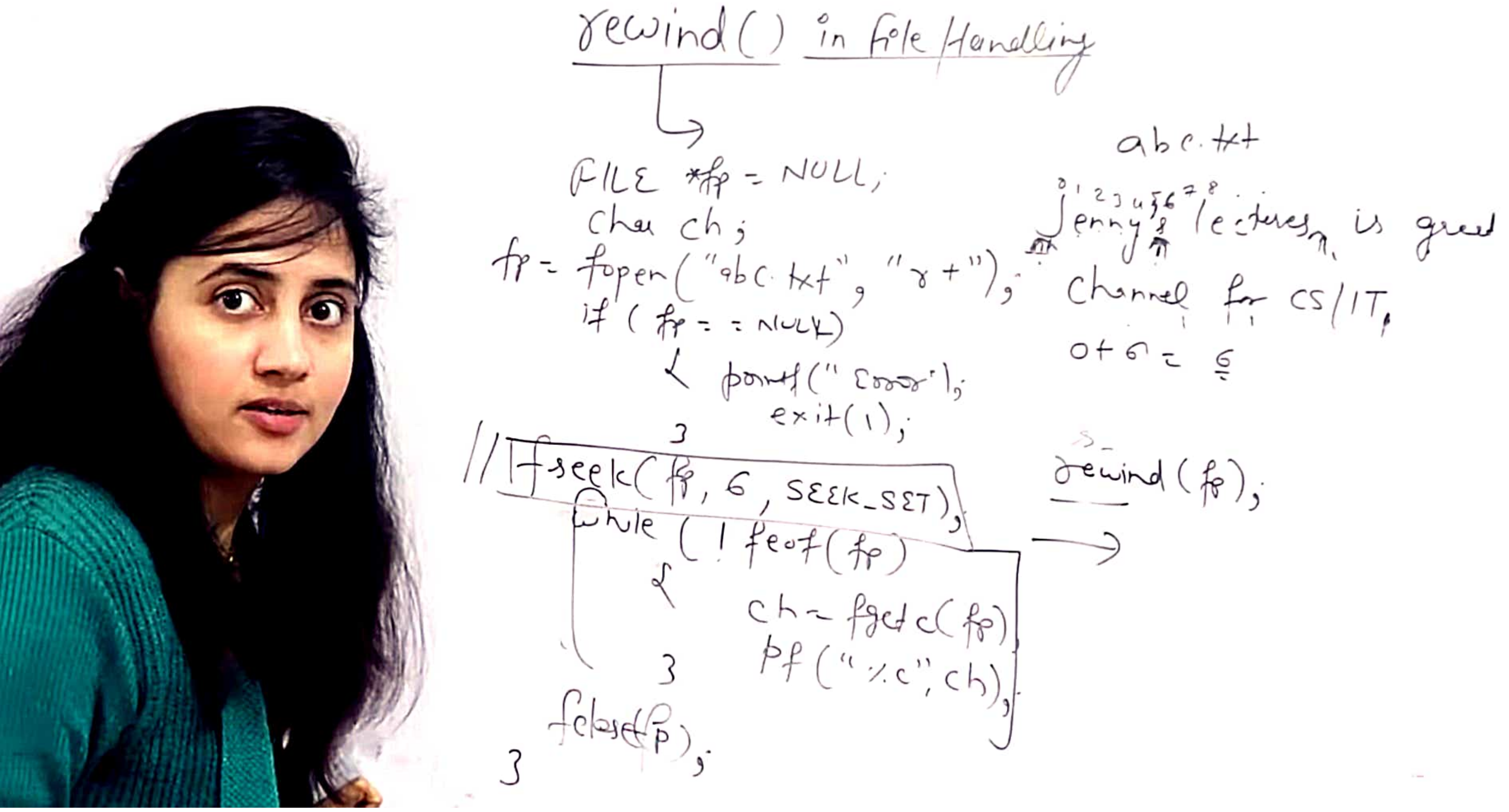
+ve/-ve

0

abc.txt

Jenny_lectures, is awesome.

SEEK_SET → beginning
SEEK_CUR - current position
SEEK_END - .



rewind() in file Handling

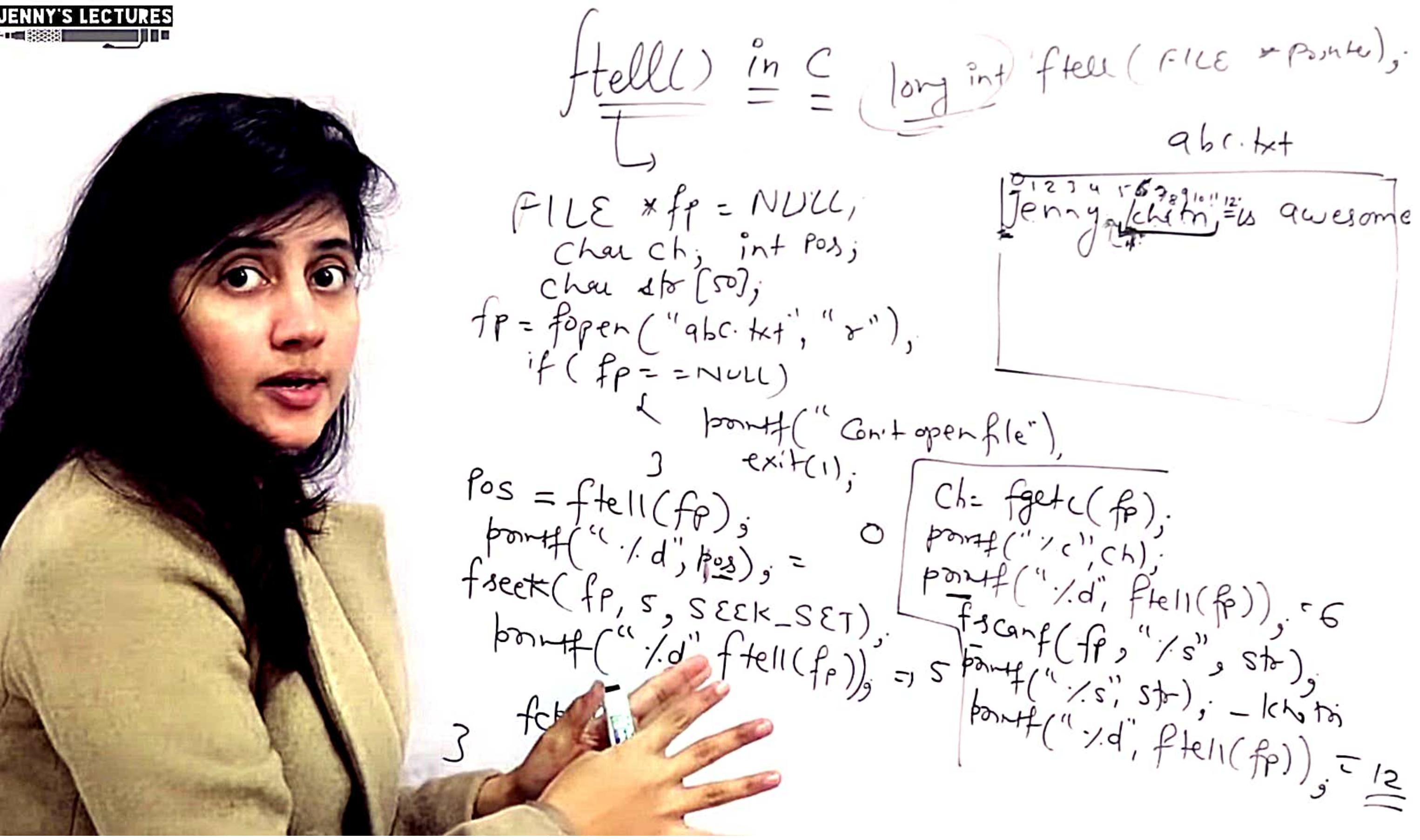
```
FILE *fp = NULL;
Char ch;
fp = fopen("abc.txt", "r+");
if (fp == NULL)
{ printf("Error");
     exit(1);
}
// fseek(fp, 6, SEEK_SET);
while (!feof(fp))
{
     ch = fgetc(fp);
     pf("%c", ch);
}
fclose(fp);
}
```

abc.txt

Jenny's lectures is great Channel for CS/IT,

0+6 = 6

rewind(fp);



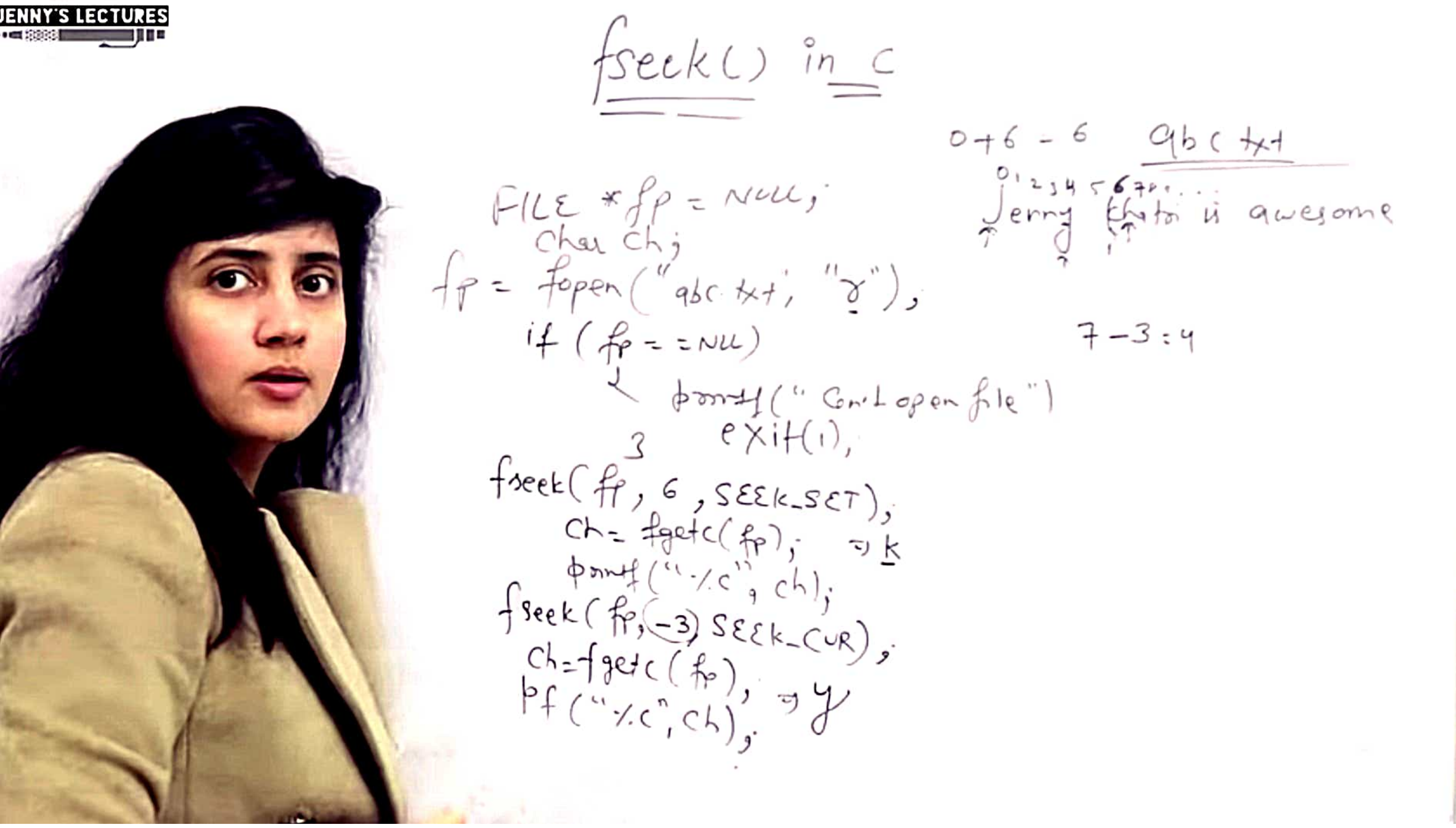
ftell() in C   long int ftell(FILE *pointer);

abc.txt

```
FILE *fp = NULL;
  char ch; int pos;
  char str[50];
fp = fopen("abc.txt", "r");
  if(fp = = NULL)
      {  printf("Can't open file");
      }     exit(1);
pos = ftell(fp);
 printf("%d", pos); = 0
fseek(fp, 5, SEEK_SET);
 printf("%d" ftell(fp)); => 5

   fcl
}
```

```
Ch= fgetc(fp);
printf("%c", ch);
printf("%d", ftell(fp)); = 6
 fscanf(fp, "%s", str);
 printf("%s", str); - khatri
 printf("%d", ftell(fp)); = 12
```



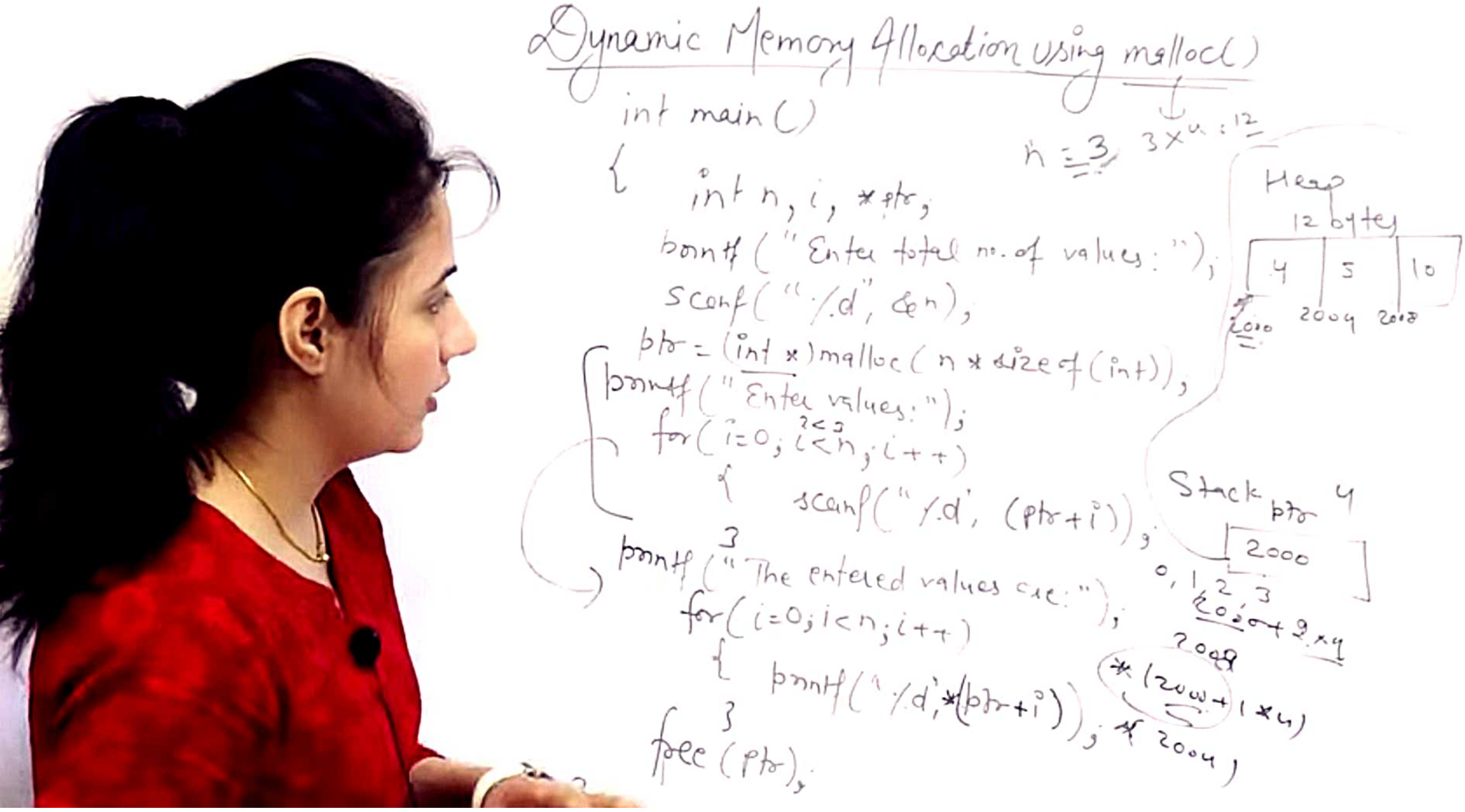
fseek() in C

```
FILE *fp = NULL;
  char ch;
fp = fopen("abc.txt", "r");
  if (fp = = NULL)
   {
     printf(" Can't open file")
       exit(1);
   }
fseek(fp, 6 , SEEK_SET);
   ch= fgetc(fp);        => k
   printf("%c", ch);
fseek(fp, -3, SEEK_CUR);
  ch=fgetc(fp);
  pf("%c", ch);   => y
```

0+6 - 6   abc.txt

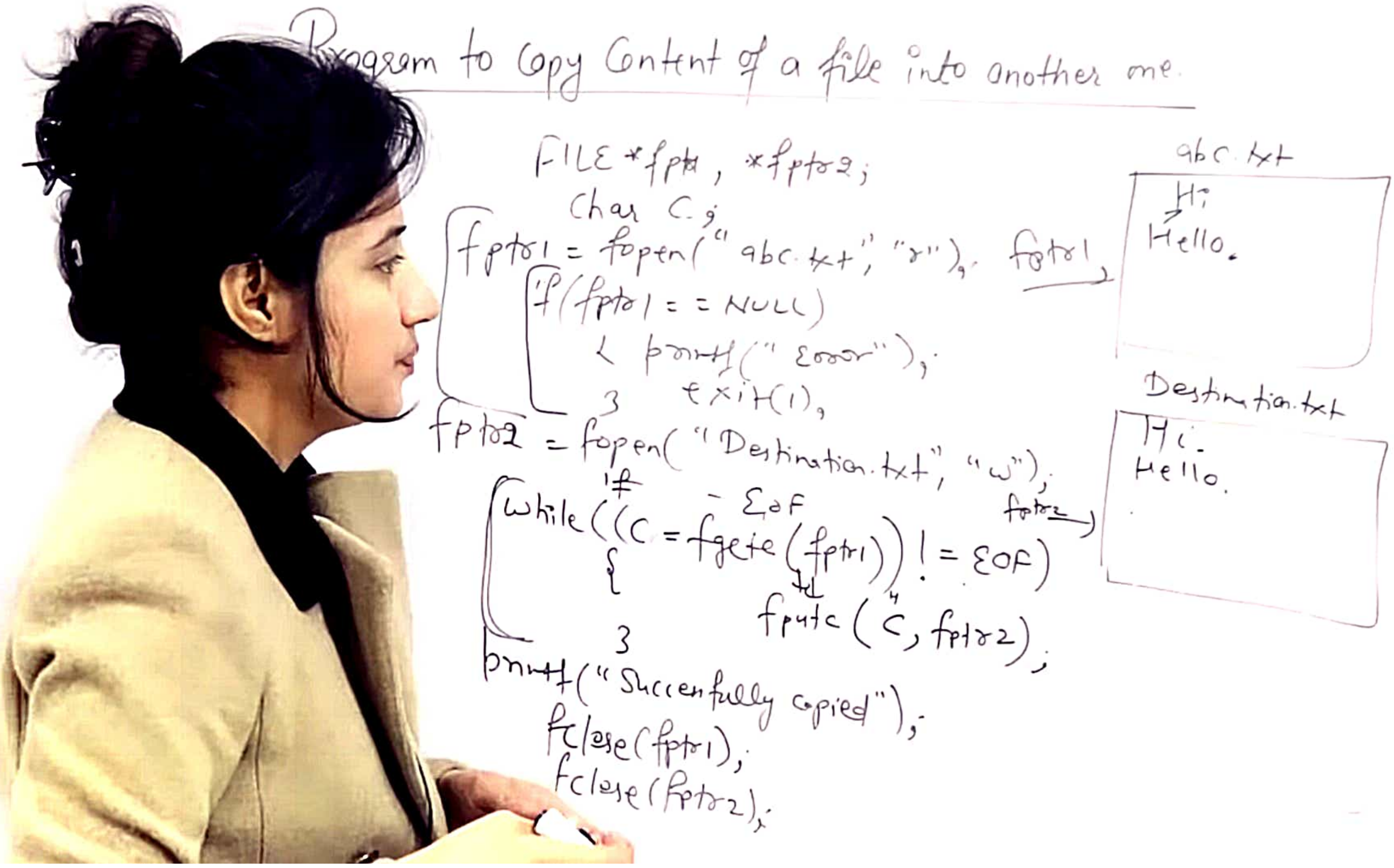
0 1 2 3 4 5 6 7 . . . . .
Jenny Khatri is awesome

7-3 = 4



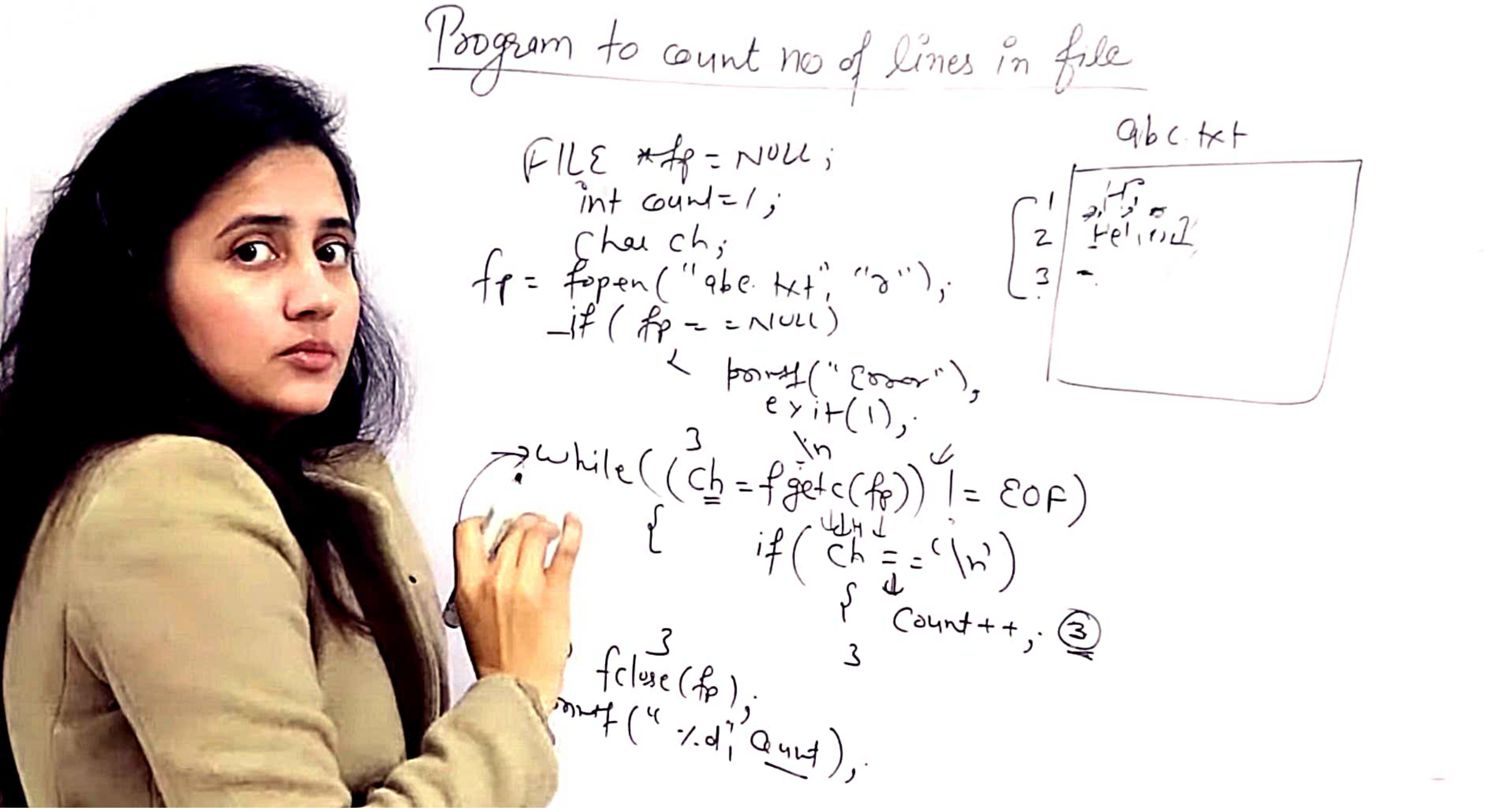
Dynamic Memory Allocation using malloc()

```
int main()
{
      int n, i, *ptr;
    printf("Enter total no. of values:");
    scanf("%d", &n);
   ptr = (int *)malloc(n * sizeof(int));
  printf("Enter values:");
    for(i=0; i<n; i++)
        {
            scanf("%d", (ptr+i));
        }
  printf("The entered values are:");
     for(i=0;i<n;i++)
        {
           printf("%d",*(ptr+i));
        }
    free(ptr);
```



Program to copy Content of a file into another one.

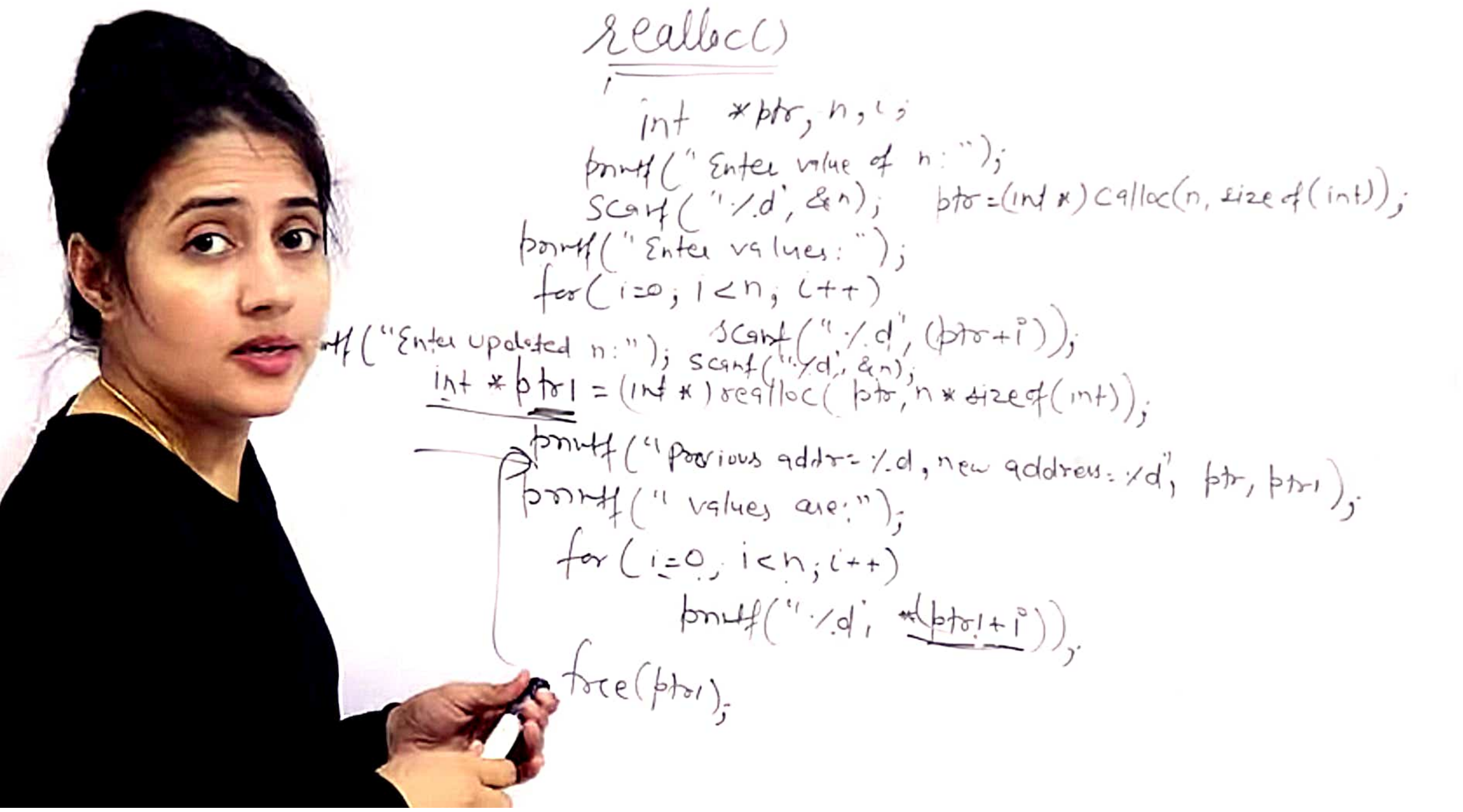
```
FILE *fptr1, *fptr2;
  char c;
fptr1 = fopen("abc.txt", "r");
  if(fptr1 == NULL)
    { printf("Error");
    }   exit(1);
fptr2 = fopen("Destination.txt", "w");
 while((c = fgetc(fptr1)) != EOF)
      {
                 fputc(c, fptr2);
      }
printf("Succenfully copied");
  fclose(fptr1);
   fclose(fptr2);
```



Program to count no of lines in file

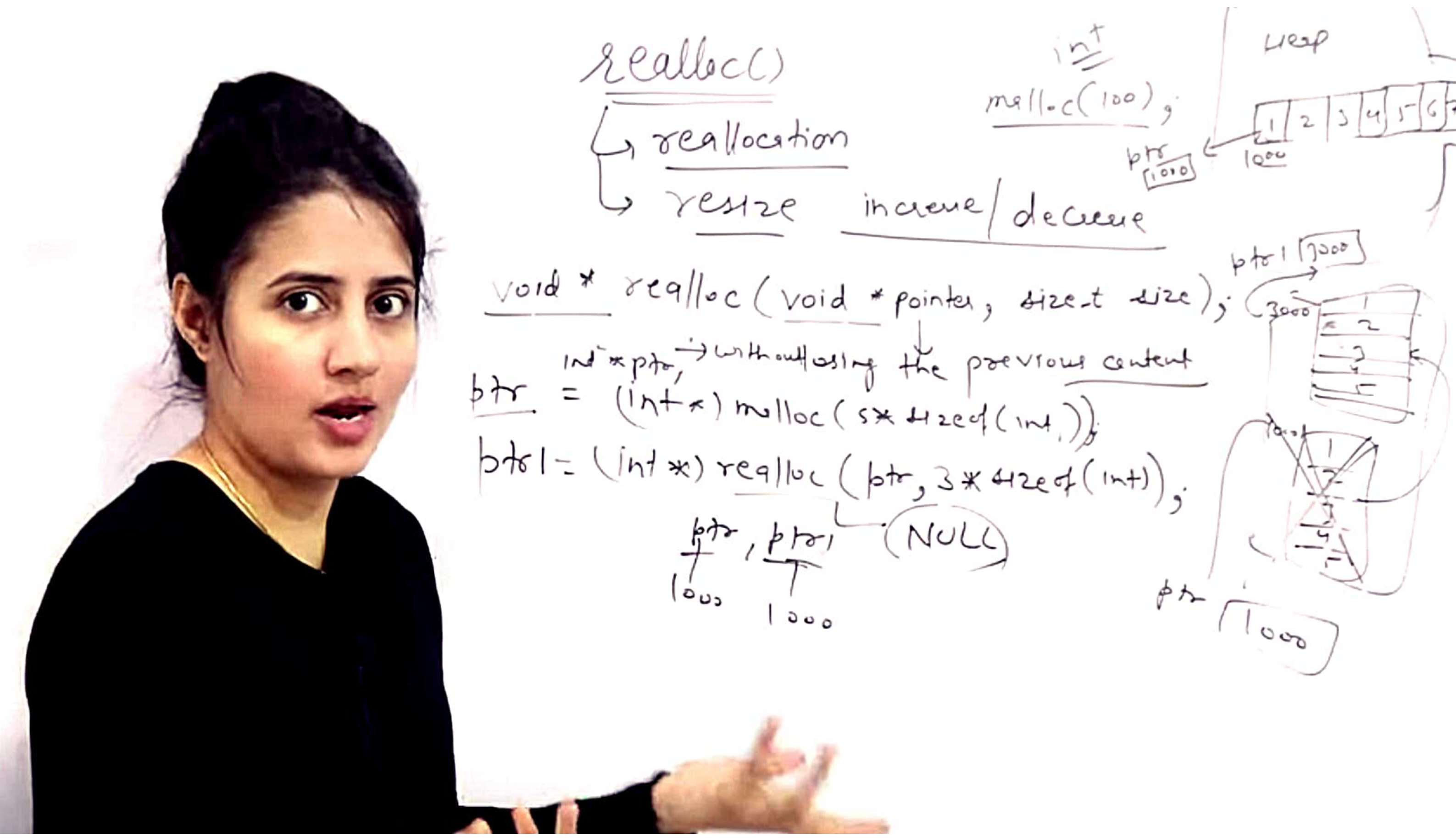
```
FILE *fp = NULL;
  int count=1;
  char ch;
fp = fopen("abc.txt", "r");
 if( fp = = NULL)
   {  printf("Error");
       exit(1);
   }
while((ch = fgetc(fp)) != EOF)
 {
       if( ch = = '\n')
          {  count++;  ③
          }
 }
 fclose(fp);
printf("%d", count);
```

abc.txt



realloc()

```
int *ptr, n, i;
printf("Enter value of n:");
scanf("%d", &n);     ptr=(int *)calloc(n, sizeof(int));
printf("Enter values:");
for(i=0; i<n; i++)
        scanf("%d", (ptr+i));
tf("Enter updated n:"); scanf("%d", &n);
int *ptr1 = (int *)realloc( ptr, n * sizeof(int));
  printf("Previous addr=%d, new address=%d", ptr, ptr1);
  printf(" values are:");
   for(i=0; i<n; i++)
          printf("%d", *(ptr1+i));
   free(ptr1);
```



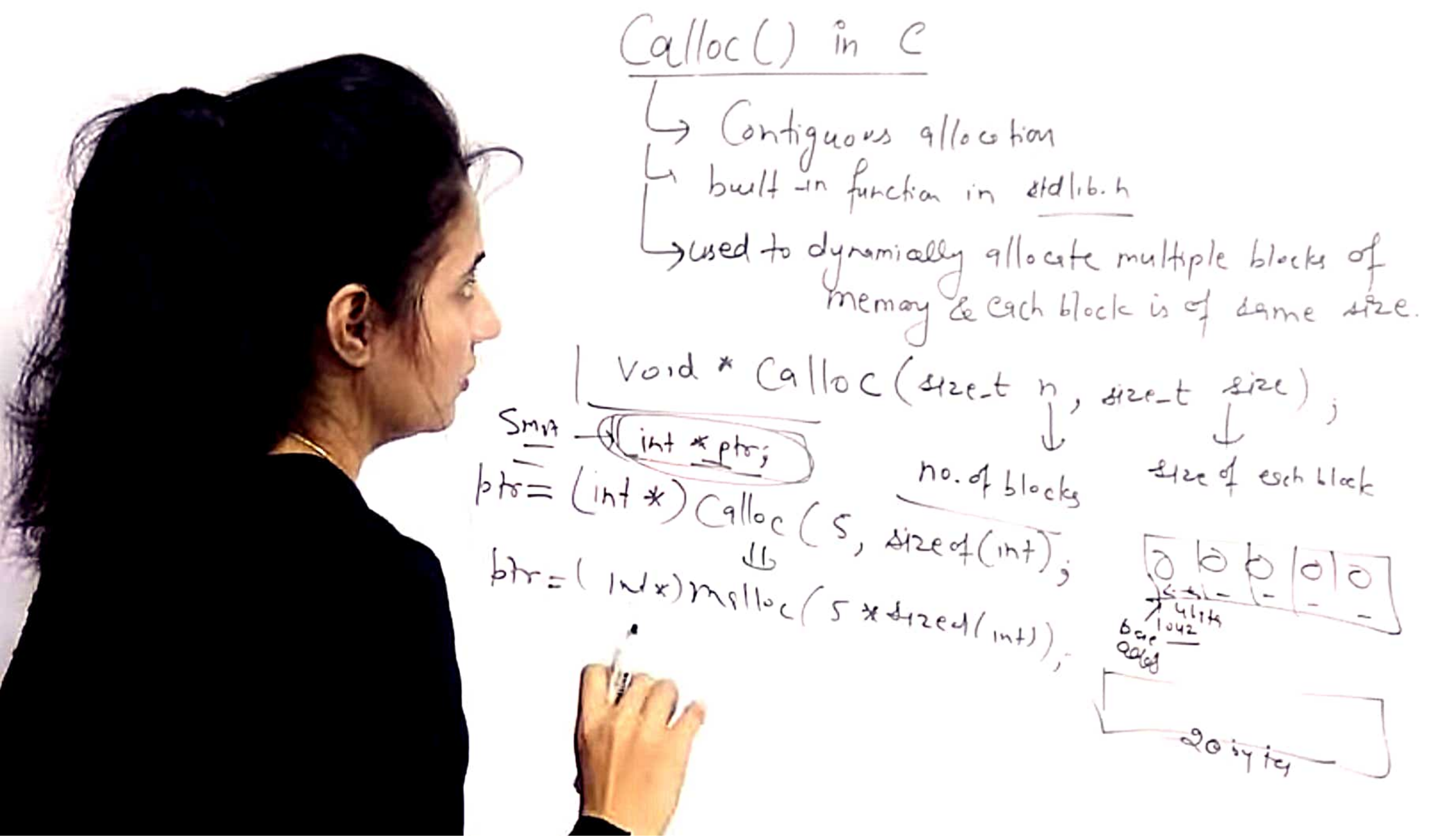
realloc()

↳ reallocation

↳ resize increase/decrease

```
int
malloc(100);

void * realloc(void *pointer, size_t size);
int *ptr, → without losing the previous content
ptr = (int*) malloc(5* sizeof(int));
ptr1 = (int*) realloc(ptr, 3* sizeof(int));
                                  NULL
ptr, ptr1
1000  1000
```



Calloc() in C

↳ Contiguous allocation

↳ built-in function in stdlib.h

↳ used to dynamically allocate multiple blocks of memory & each block is of same size.

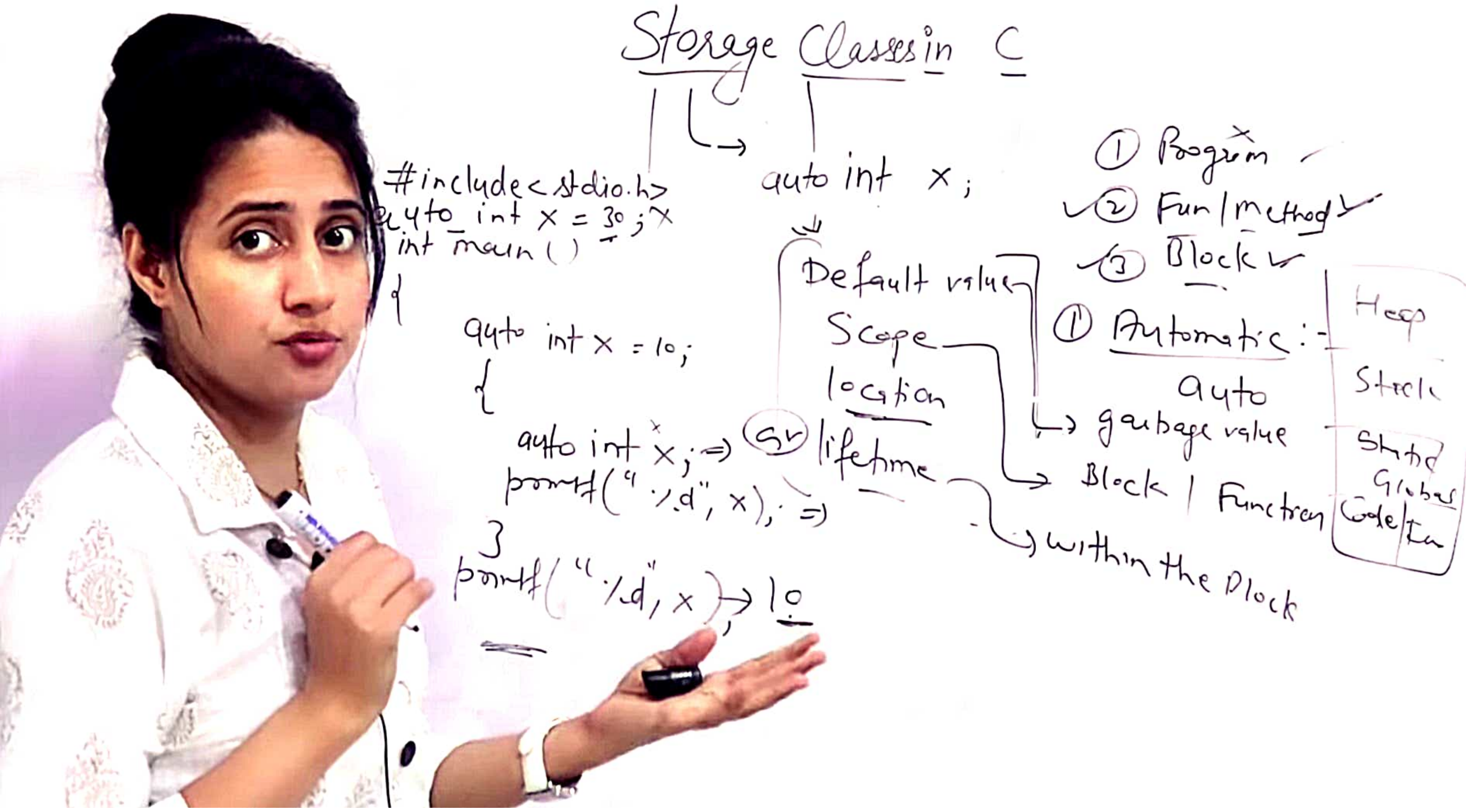
```
void * Calloc(size_t n, size_t size);
```

n ↓ no. of blocks

size ↓ size of each block

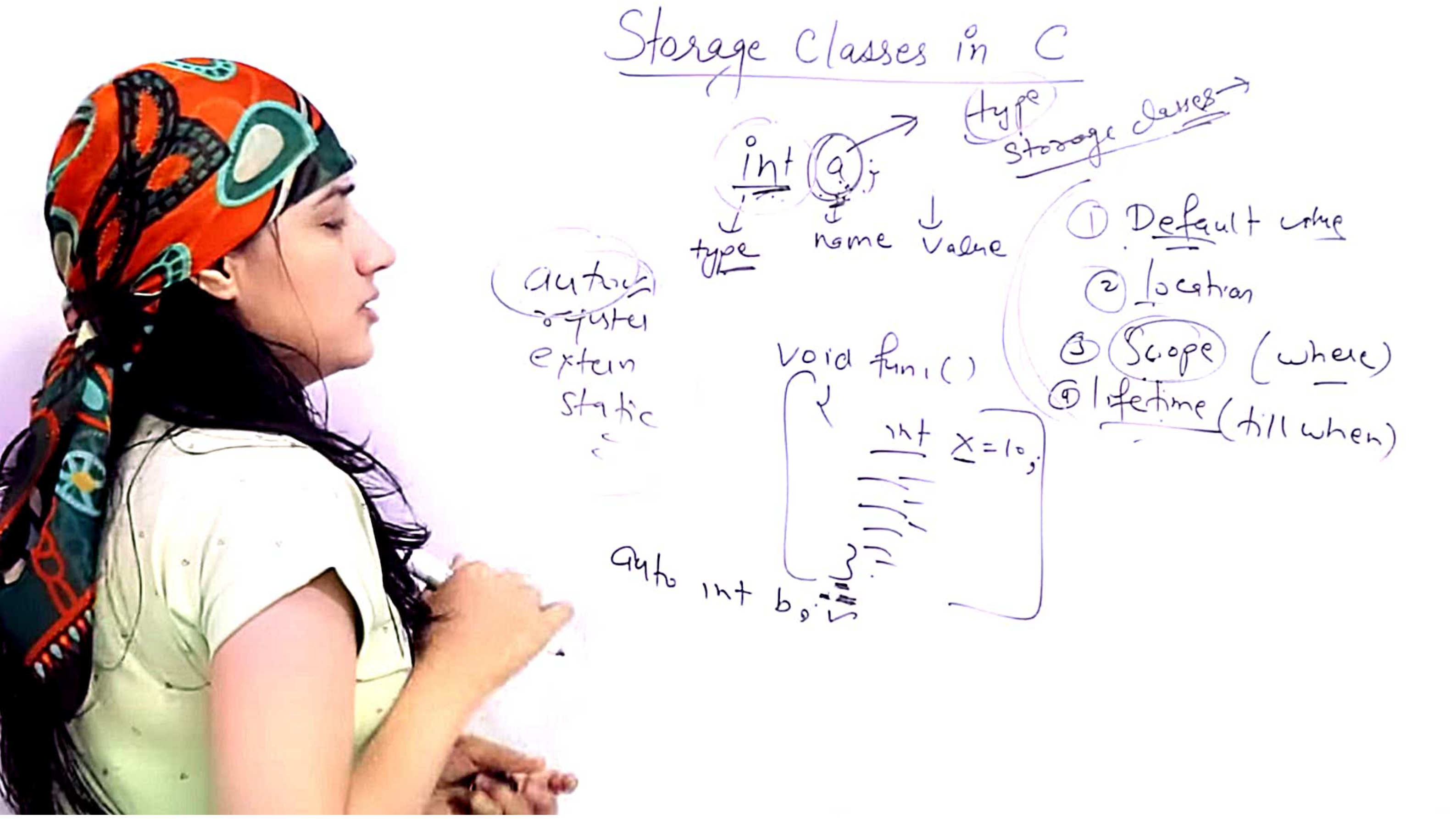
Smnt → int *ptr;

```
ptr = (int *) Calloc(5, sizeof(int));
        ⇓
ptr = (int*)malloc(5 * sizeof(int));
```



Storage Classes in C

```
#include<stdio.h>
auto int x = 30; X
int main()
{
    auto int x = 10;
    {
        auto int x; =>
        printf("%d", x); =>
    }
    printf("%d", x); → 10
```



Storage Classes in C

int a;

type name value

type

storage classes

① Default value

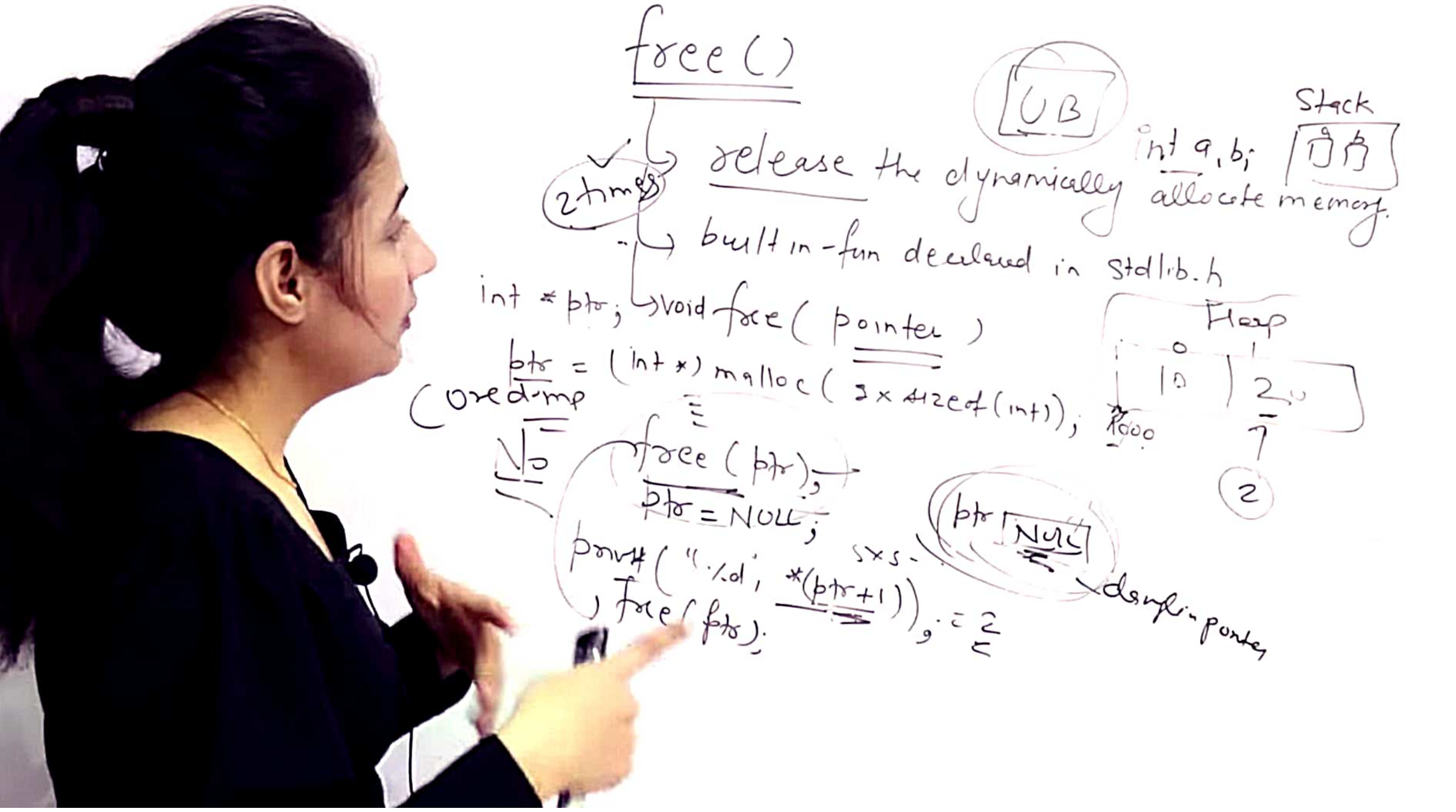
② location

③ Scope (where)

④ lifetime (till when)

auto
register
extern
static

```
void fun1()
{
    int x=10;
}
auto int b;
```



free()

→ release the dynamically allocate memory.

→ built in-fun declared in stdlib.h

→ void free(pointer)

UB

int a, b;

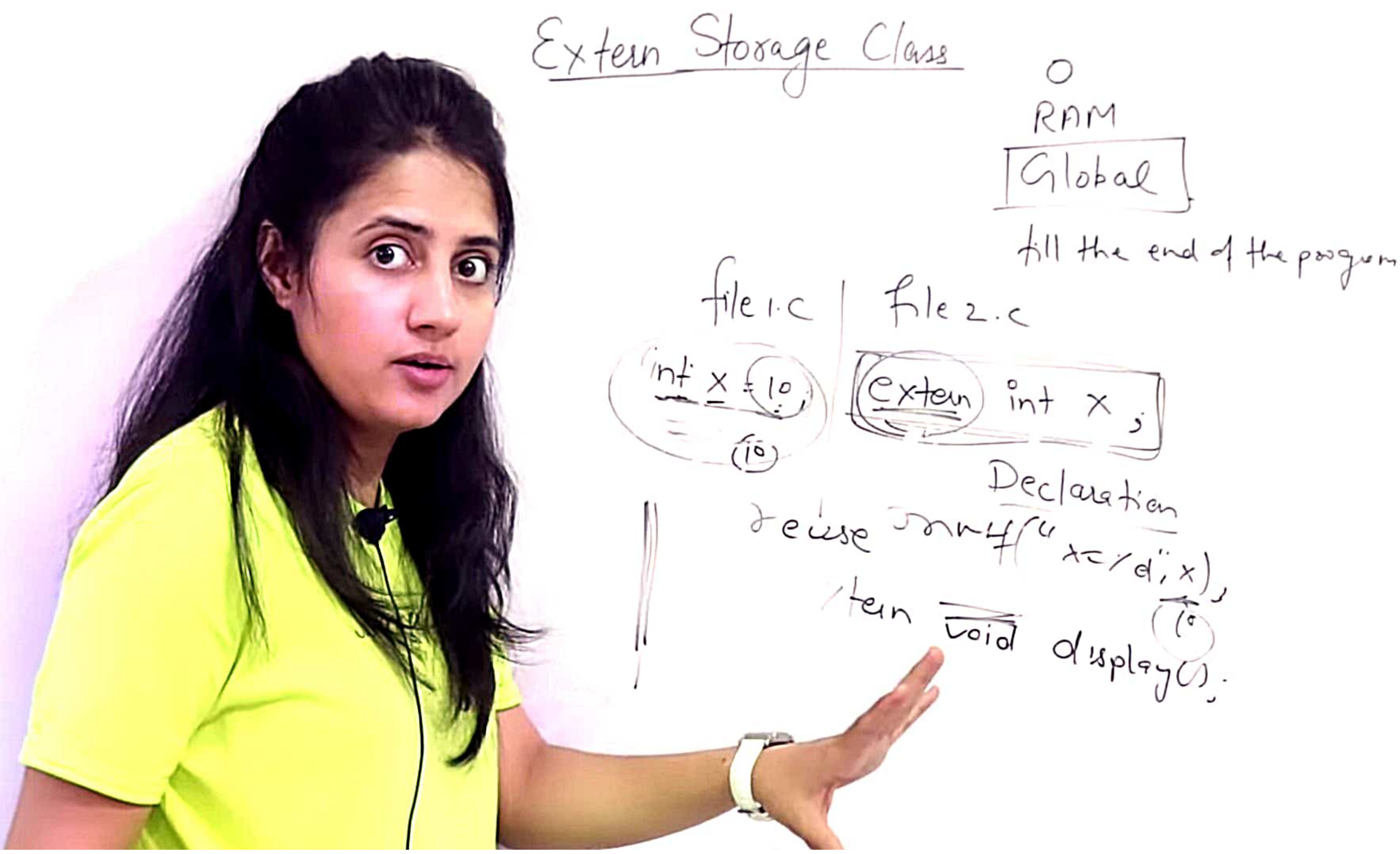
Stack

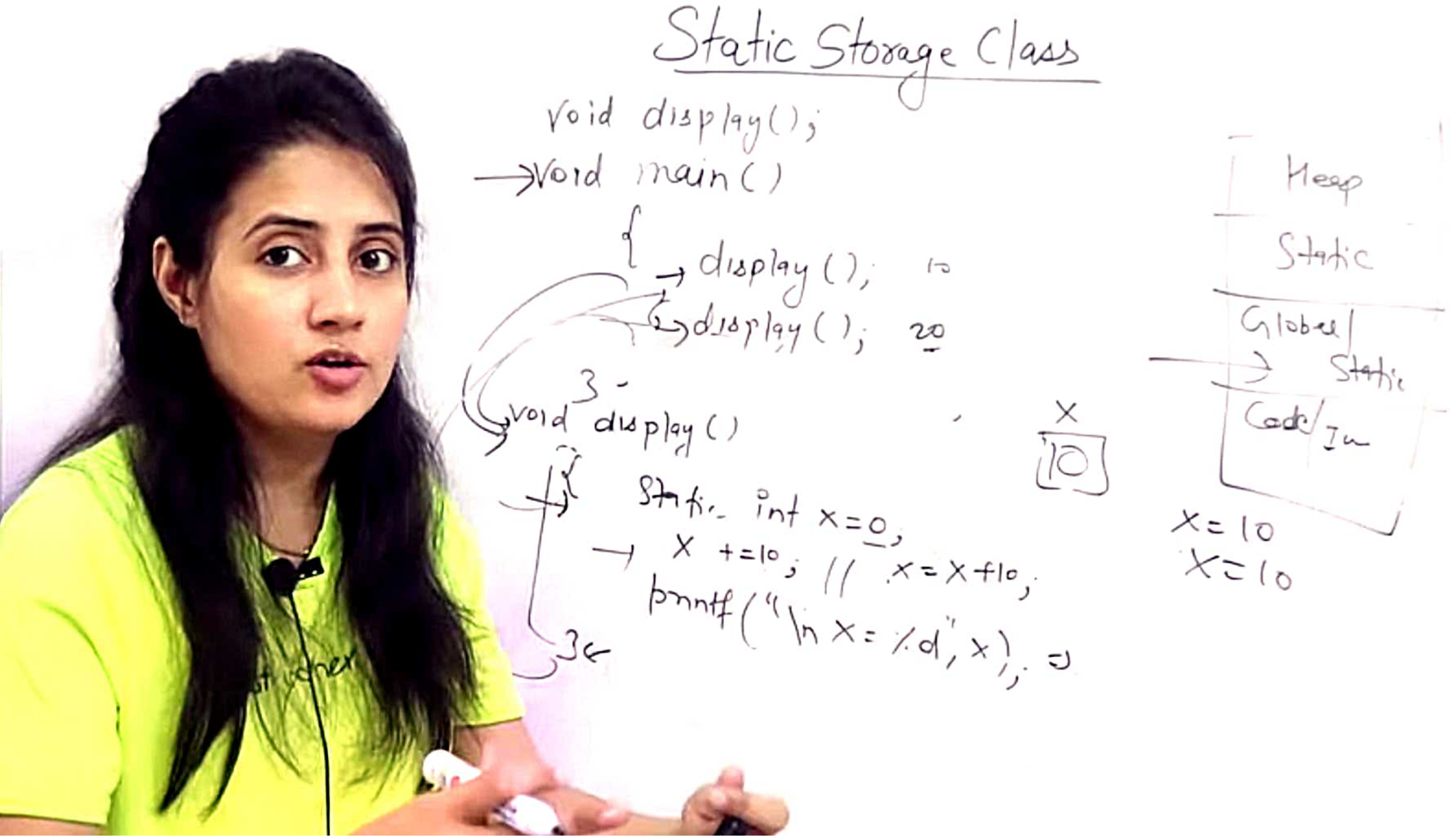
2times

```
int *ptr;
ptr = (int *) malloc( 3 x sizeof(int));
free(ptr);
ptr = NULL;
printf("%d", *(ptr+1));
free(ptr);
```

Core dump

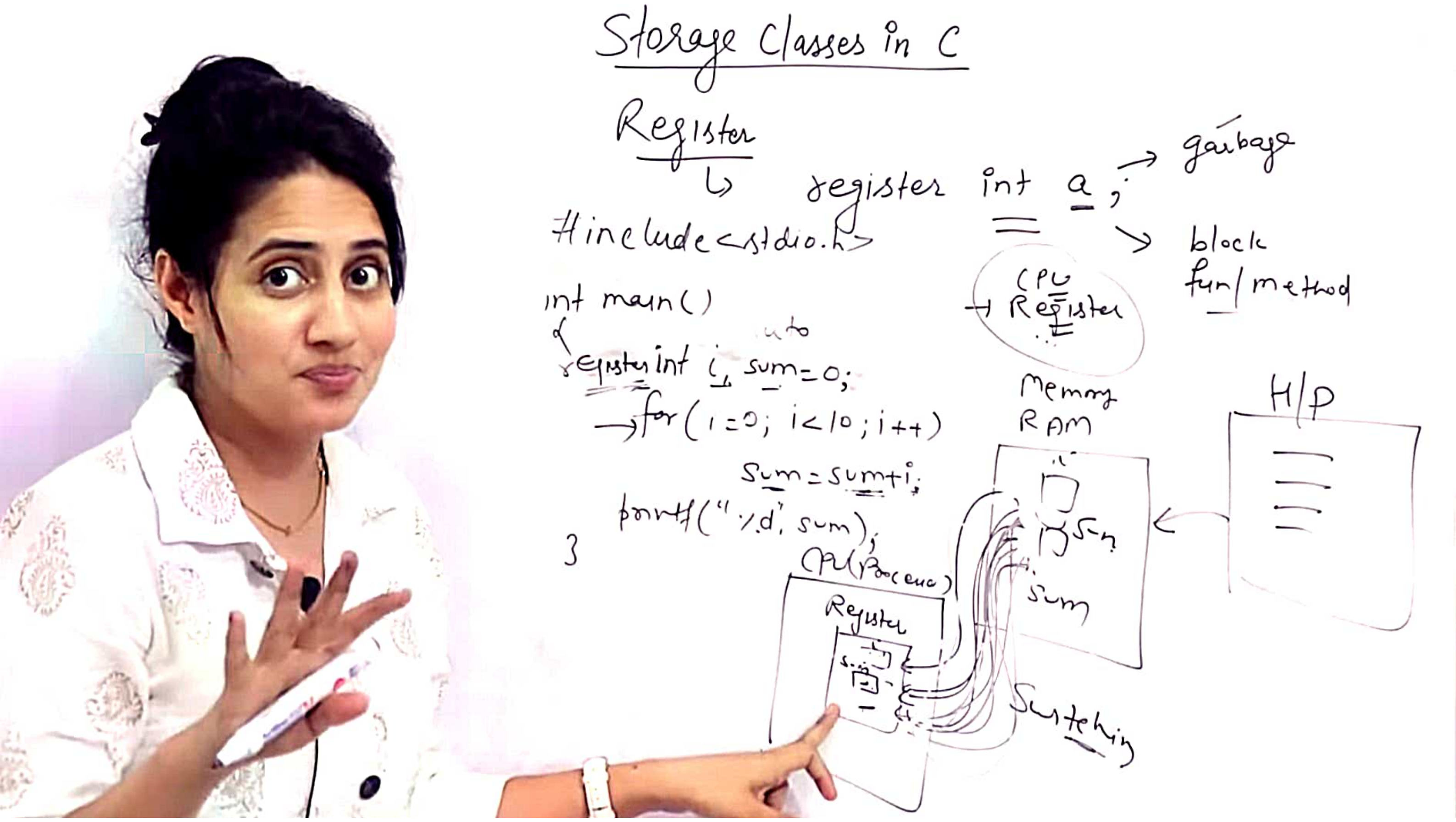
No

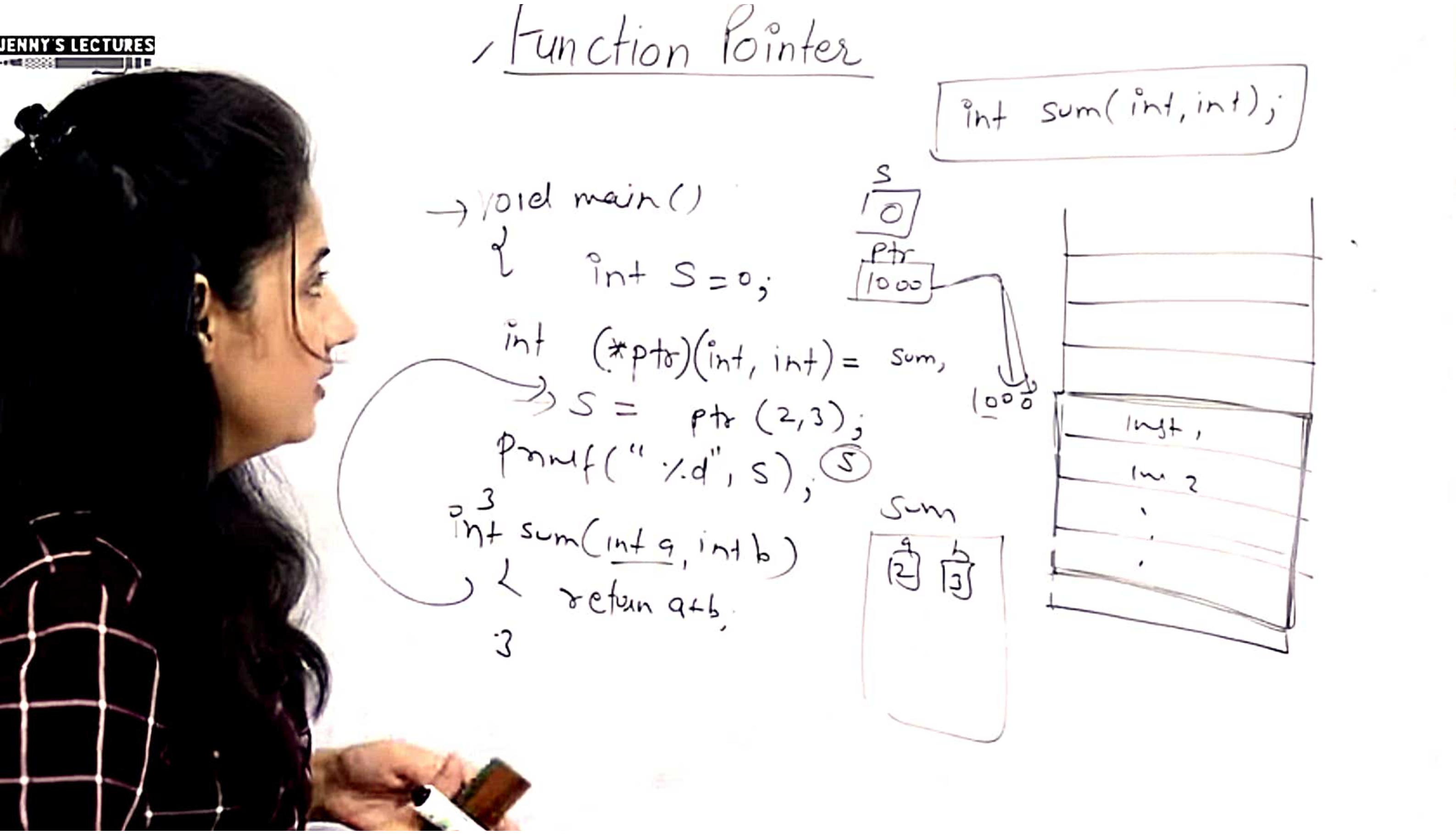

Extern Storage Class
0
RAM
Global
till the end of the poogum
file1.c
file2.c
int x = 10;
10
extern int x ;
Declaration
deuse printf("x=%d", x);
10
tern void display();



Static Storage Class

```
void display();
void main()
{
    display();   10
    display();   20
}
void display()
{
    Static int x=0;
    x +=10; // x=x+10;
    printf("\n x= %d", x);
}
```

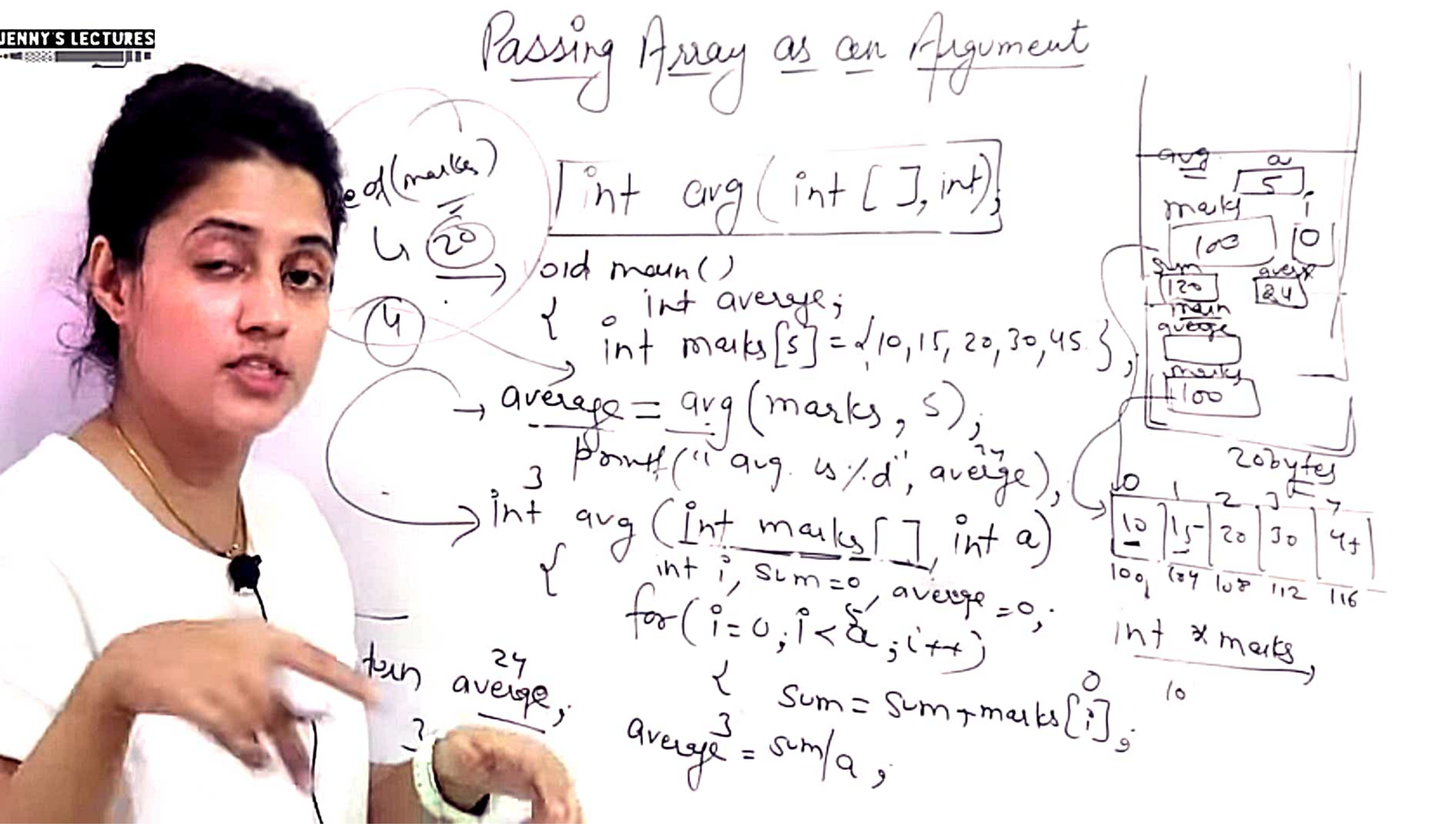




Function Pointer

```
int sum(int, int);
```

```
void main()
{       int S = 0;
 int    (*ptr)(int, int) = sum;
        S =   ptr (2,3);
 printf(" %d", S);
}
int sum(int a, int b)
 {  return a+b;
}
```

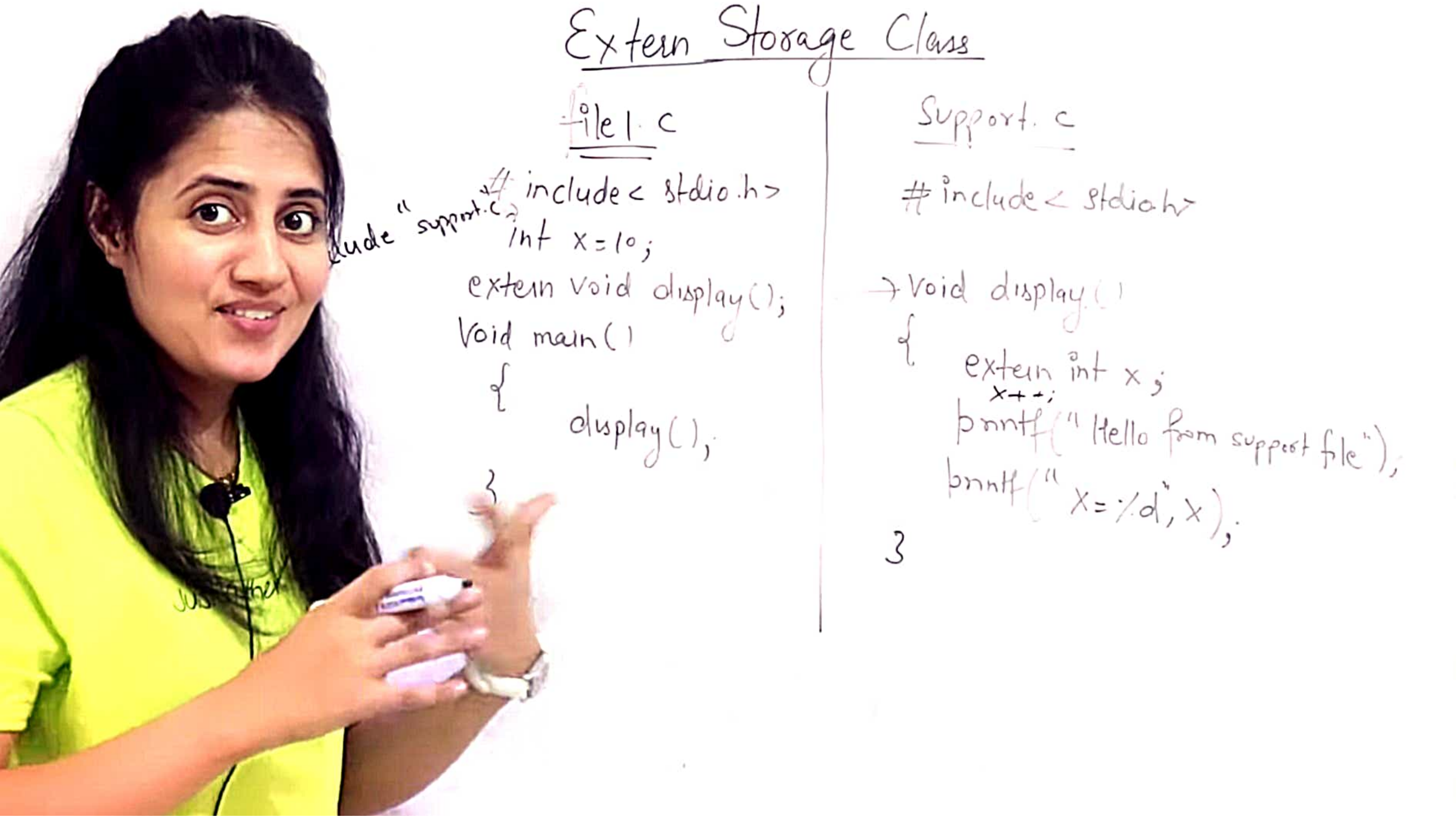


Passing Array as an Argument

```
int avg(int [], int);

eof(marks)
  ↳ 20
  4

oid main()
{
    int averge;
    int marks[5] = {10,15,20,30,45};
→ average = avg(marks, 5);
    printf("avg is %d", averge);
→ int avg(int marks[], int a)
{
    int i, sum=0, averge=0;
    for(i=0; i<a; i++)
    {
        sum = sum + marks[i];
    averge = sum/a;
turn averge;
```



Extern Storage Class

file1.c

```
lude "support.c"
# include< stdio.h>
int x=10;
extern void display();
Void main()
{
        display();
}
```

Support.c

```
# include< stdio.h>

→ void display()
{
        extern int x;
        x++;
        printf("Hello from support file");
        printf("x=%d",x);
}
```