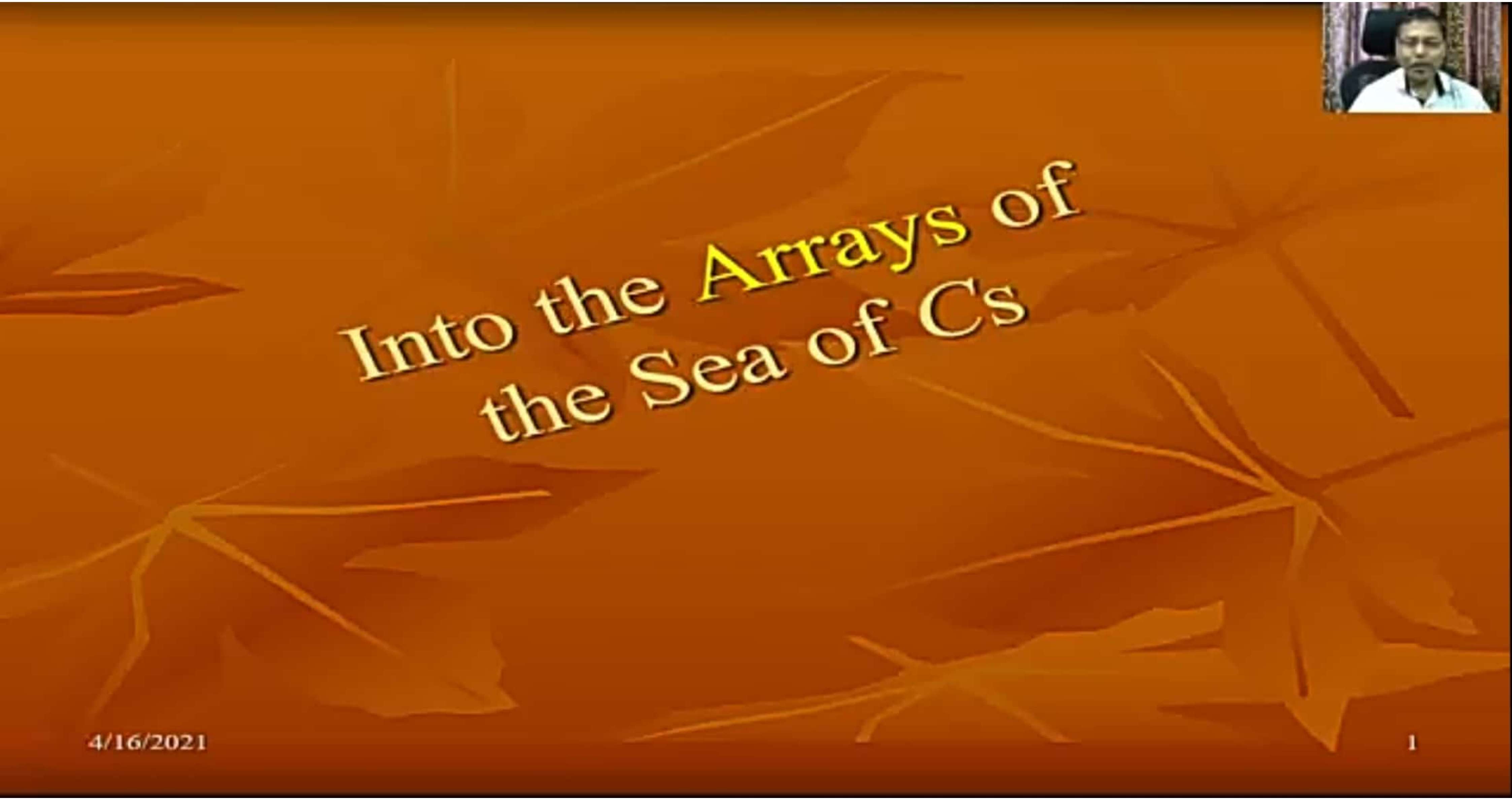
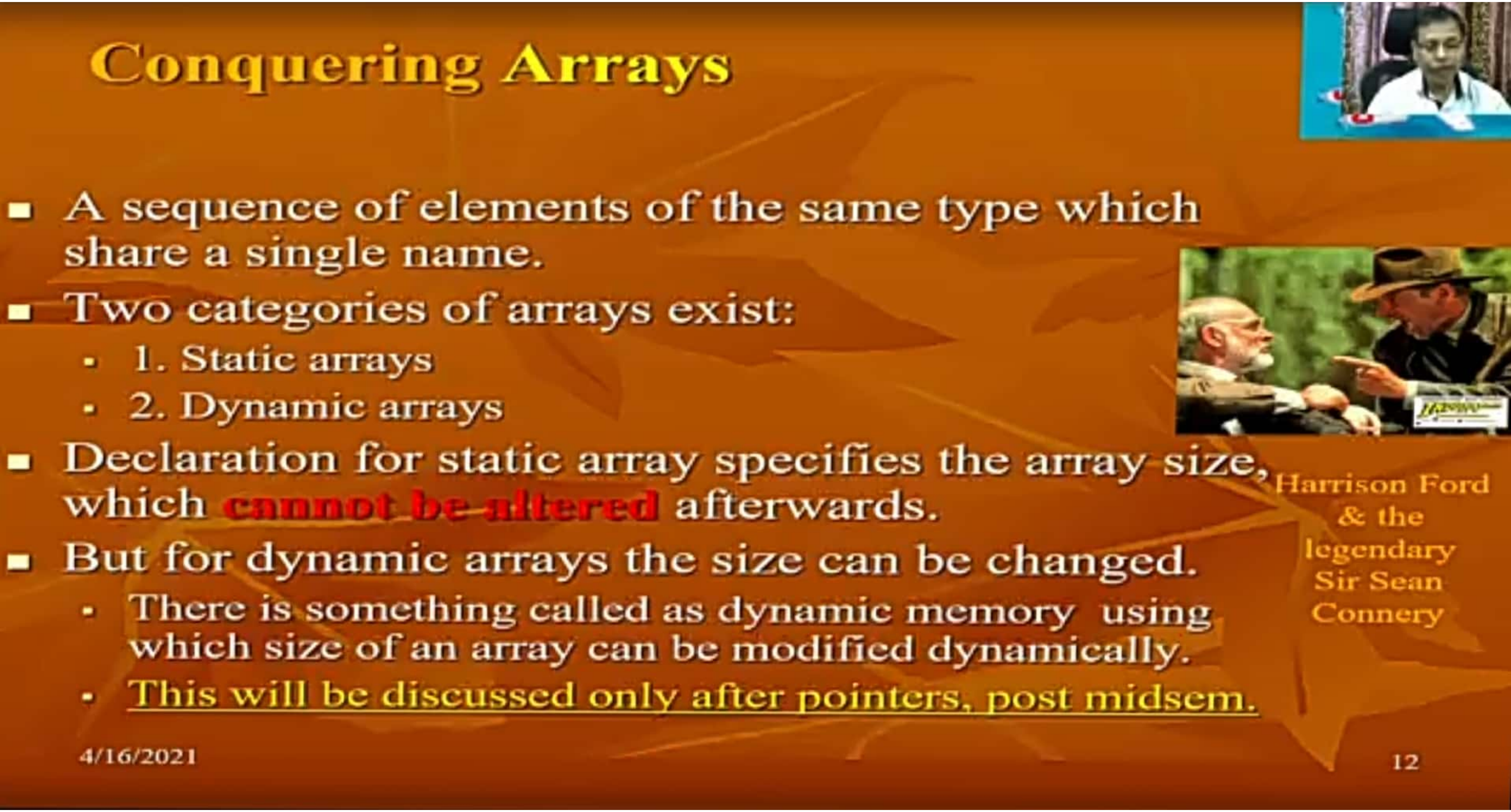


Into the Arrays of the Sea of Cs

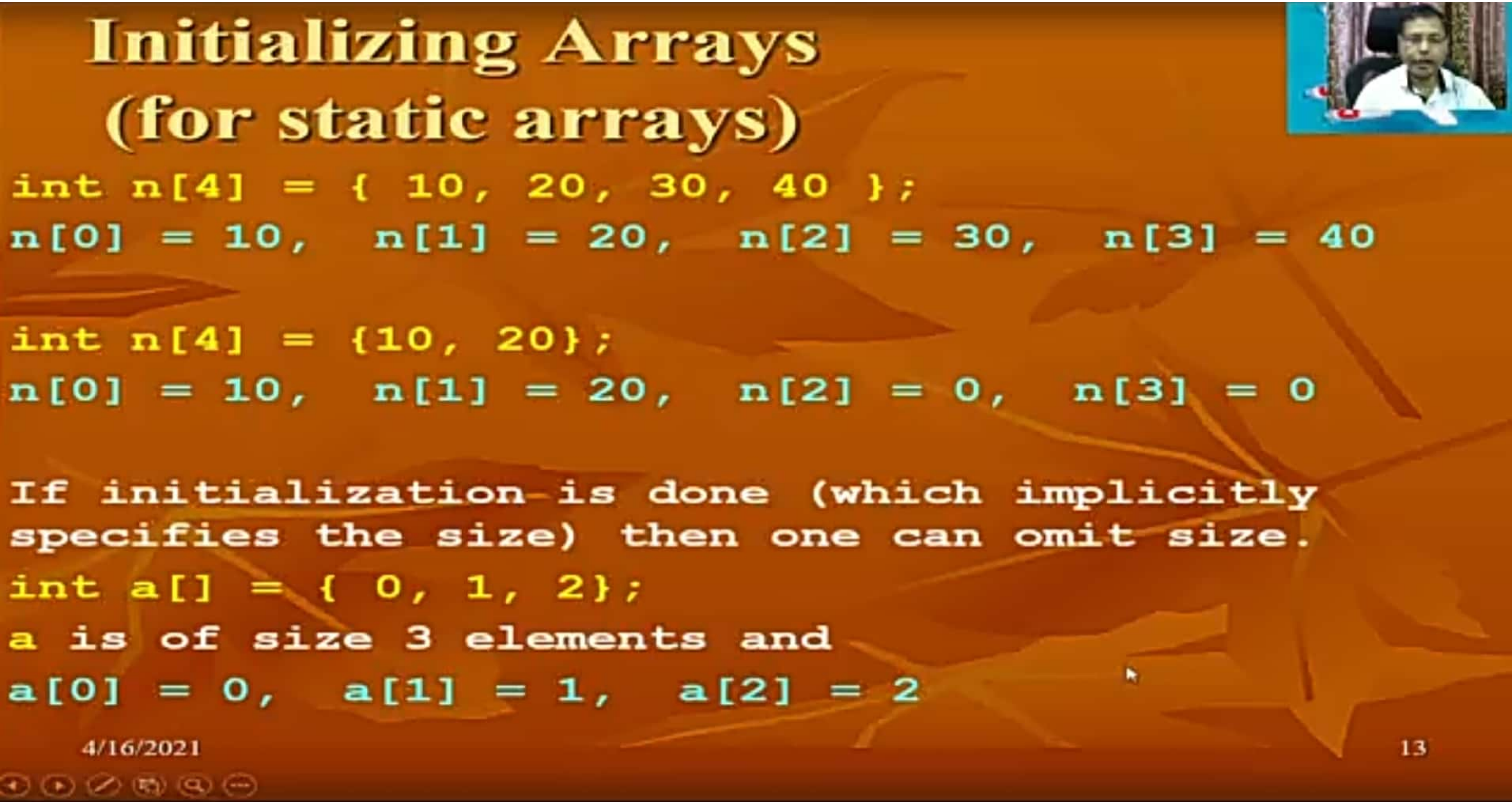
4/16/2021



Conquering Arrays

- A sequence of elements of the same type which share a single name.
- Two categories of arrays exist:
  - 1. Static arrays
  - 2. Dynamic arrays
- Declaration for static array specifies the array size, which **cannot be altered** afterwards.
- But for dynamic arrays the size can be changed.
  - There is something called as dynamic memory using which size of an array can be modified dynamically.
  - This will be discussed only after pointers, post midsem.

Harrison Ford & the legendary Sir Sean Connery



Initializing Arrays (for static arrays)

```
int n[4] = { 10, 20, 30, 40 };
n[0] = 10,  n[1] = 20,  n[2] = 30,  n[3] = 40
```

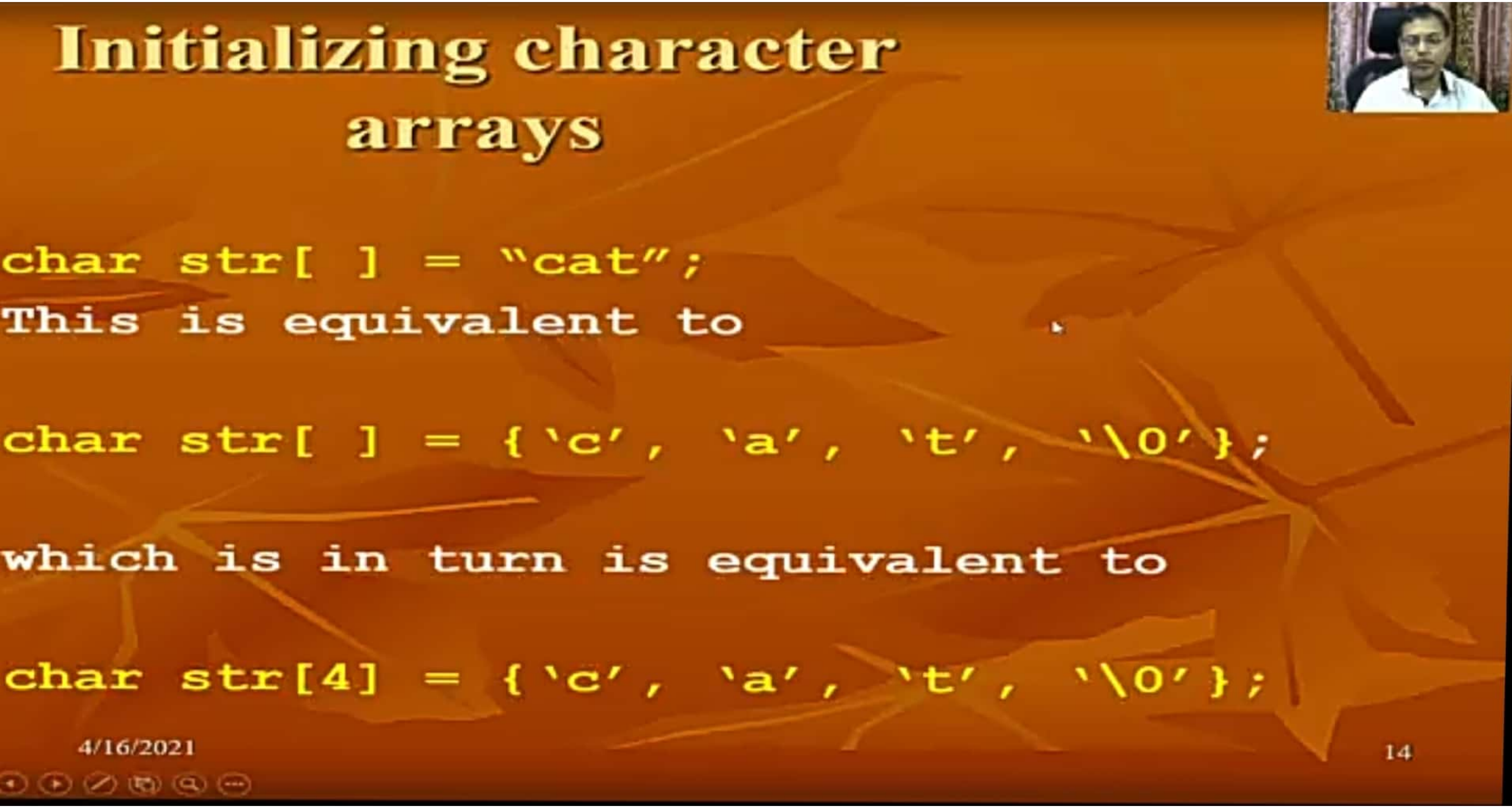
```
int n[4] = {10, 20};
n[0] = 10,  n[1] = 20,  n[2] = 0,  n[3] = 0
```

If initialization is done (which implicitly specifies the size) then one can omit size.

```
int a[] = { 0, 1, 2};
```

a is of size 3 elements and

```
a[0] = 0,  a[1] = 1,  a[2] = 2
```



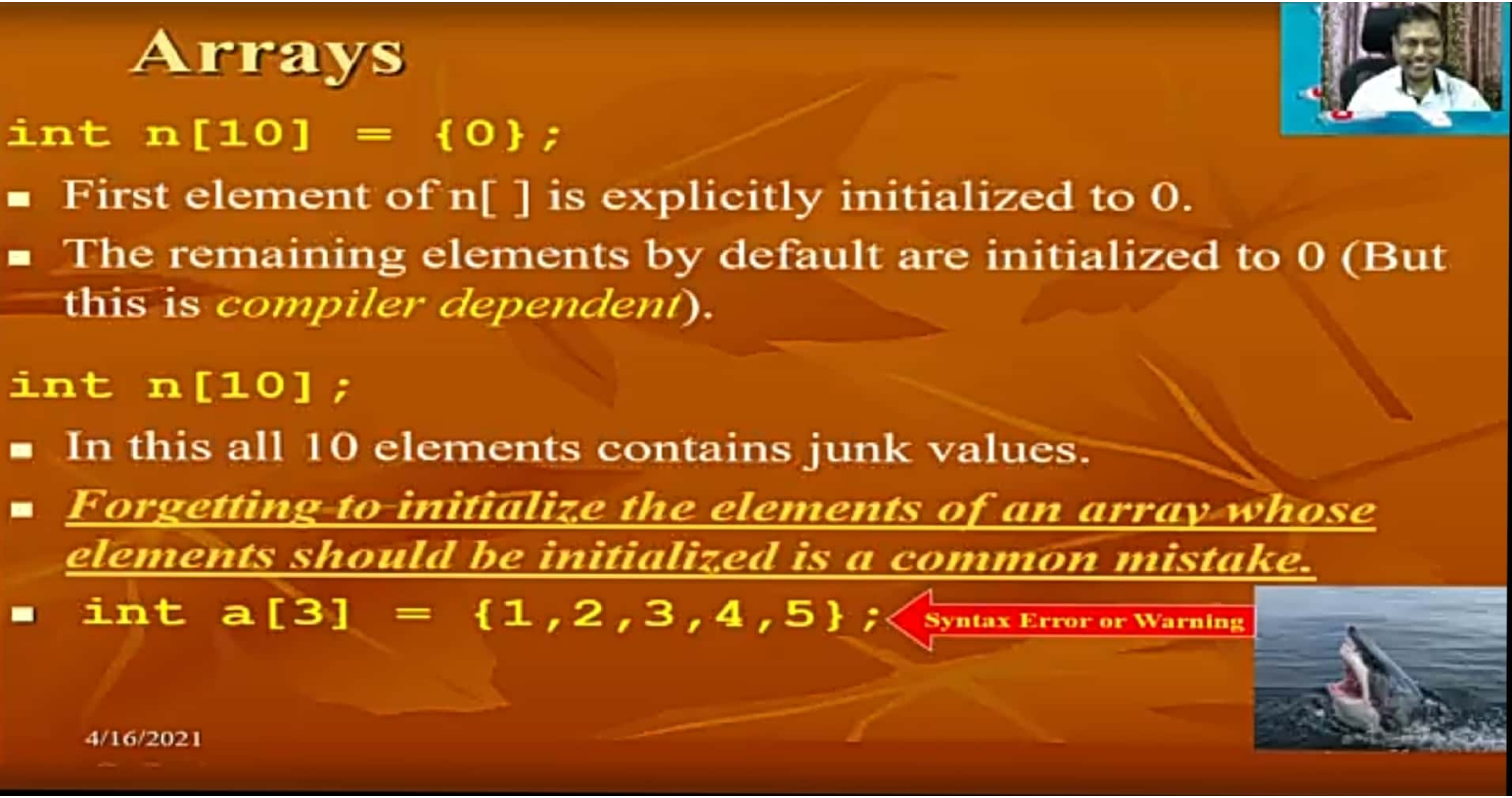
Initializing character arrays

```
char str[ ] = “cat”;
This is equivalent to

char str[ ] = {‘c’, ‘a’, ‘t’, ‘\0’};

which is in turn is equivalent to

char str[4] = {‘c’, ‘a’, ‘t’, ‘\0’};
```



Arrays

```
int n[10] = {0};
```

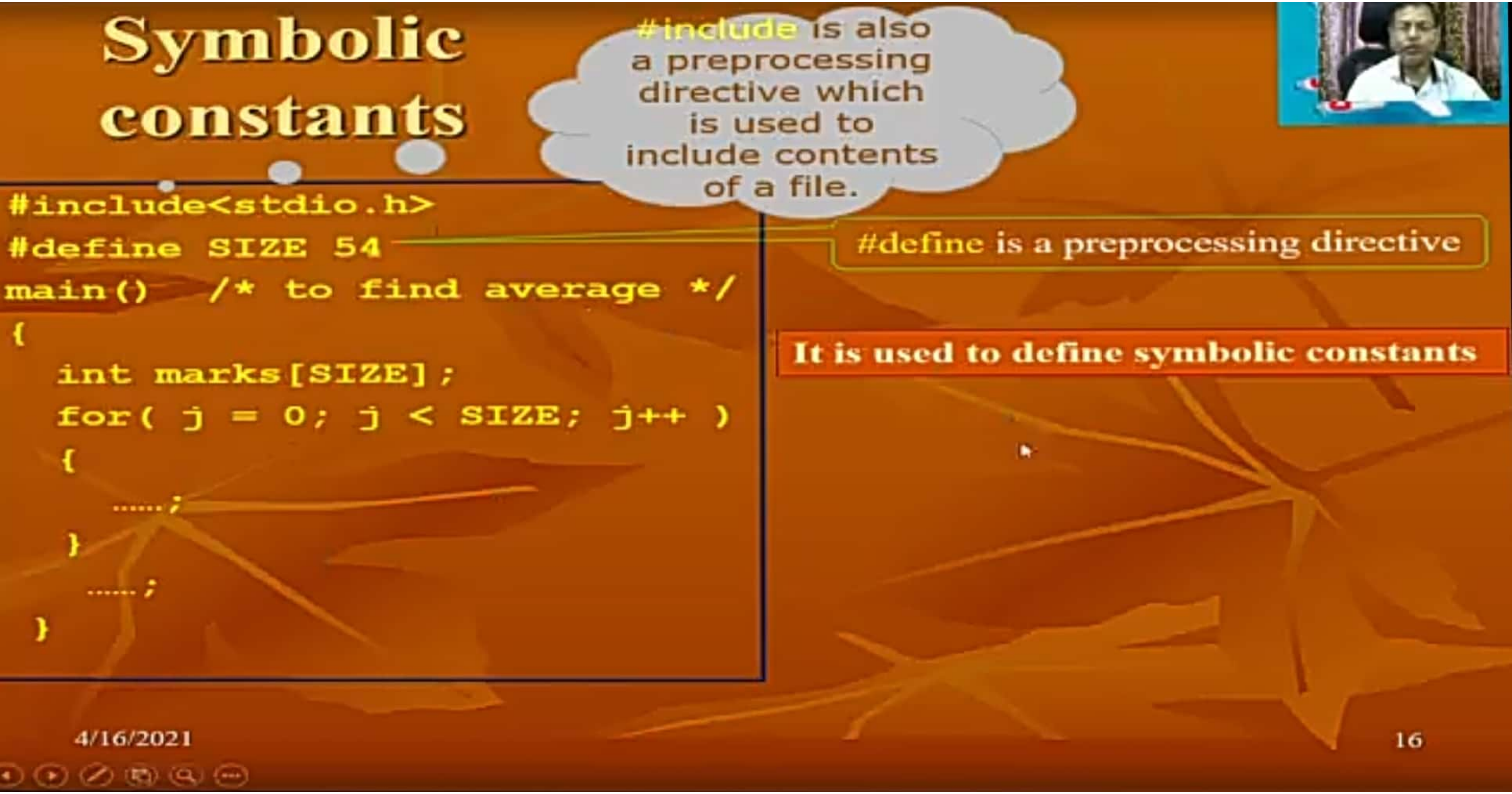
- First element of n[ ] is explicitly initialized to 0.
- The remaining elements by default are initialized to 0 (But this is *compiler dependent*).

```
int n[10];
```

- In this all 10 elements contains junk values.
- ***Forgetting to initialize the elements of an array whose elements should be initialized is a common mistake.***
- `int a[3] = {1,2,3,4,5};`

Syntax Error or Warning

4/16/2021



Symbolic constants

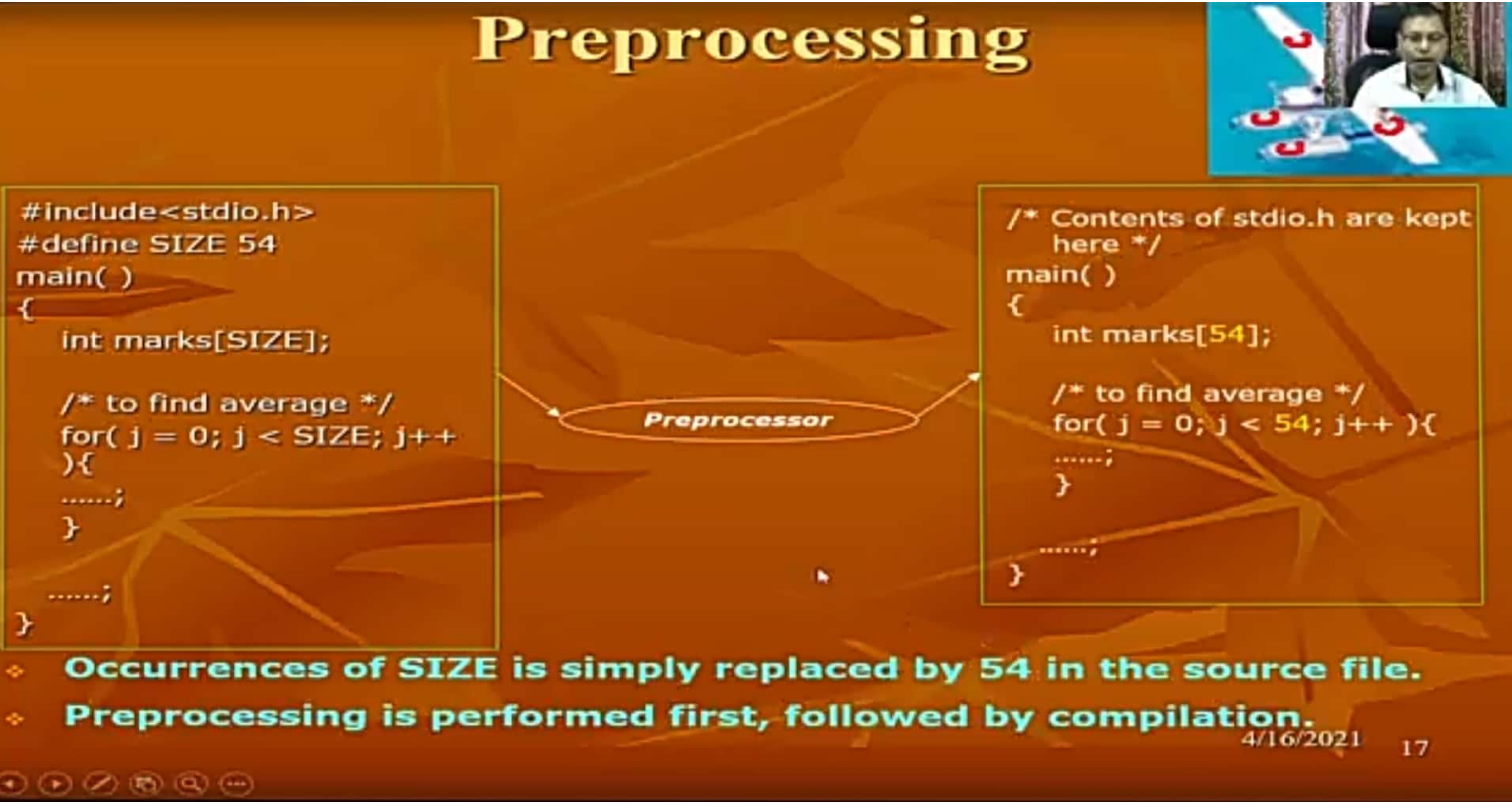
#include is also a preprocessing directive which is used to include contents of a file.

```
#include<stdio.h>
#define SIZE 54
main()   /* to find average */
{
  int marks[SIZE];
  for( j = 0; j < SIZE; j++ )
  {
     ......;
  }
    ......;
 }
```

#define is a preprocessing directive

It is used to define symbolic constants

4/16/2021



Preprocessing

```
#include<stdio.h>
#define SIZE 54
main( )
{
    int marks[SIZE];

    /* to find average */
    for( j = 0; j < SIZE; j++
    ){
    ......;
    }


   ......;
}
```

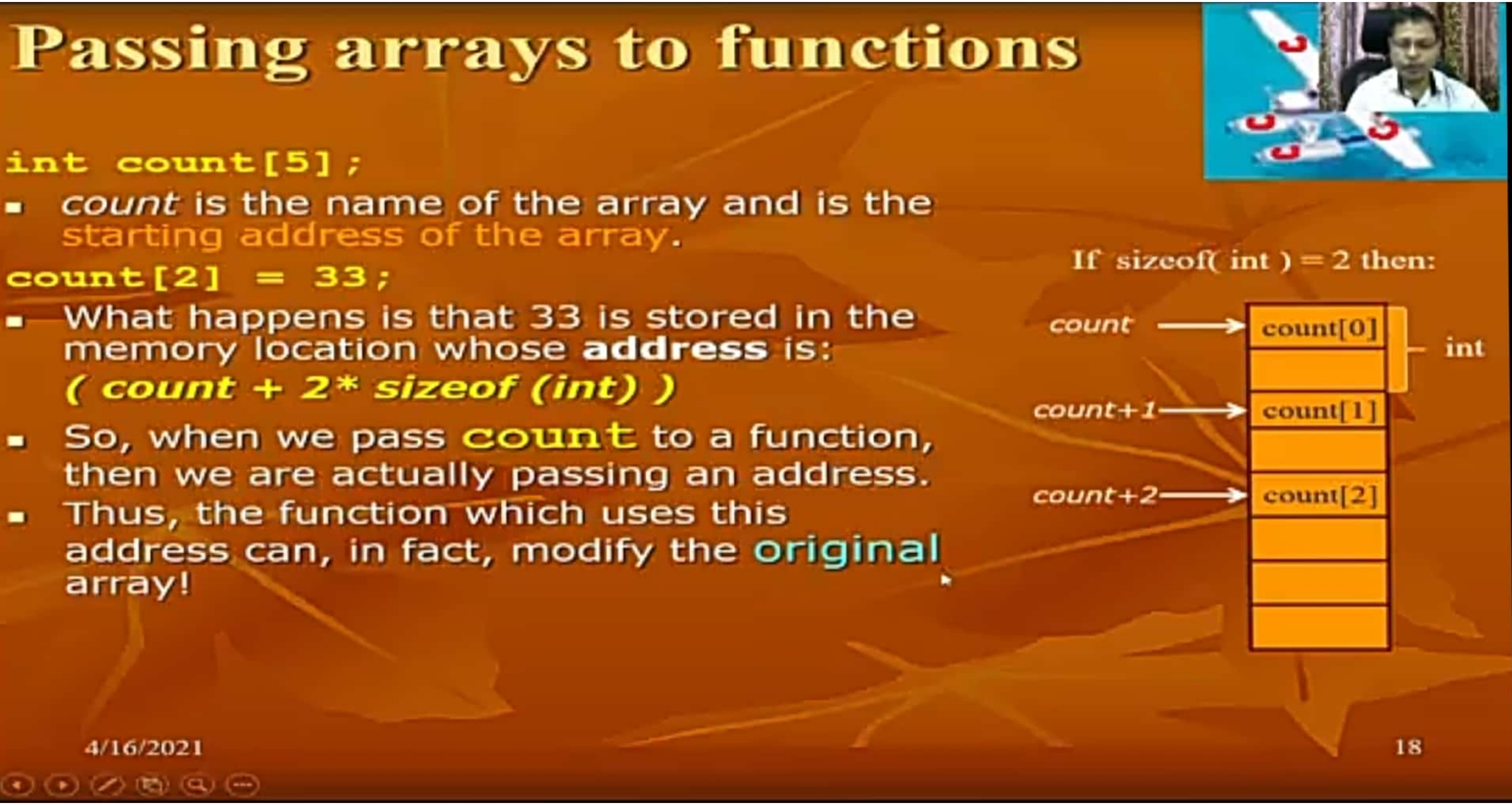
```
/* Contents of stdio.h are kept
   here */
main( )
{
   int marks[54];

   /* to find average */
   for( j = 0; j < 54; j++ ){
   ......;
   }

   ......;
}
```

- Occurrences of SIZE is simply replaced by 54 in the source file.
- Preprocessing is performed first, followed by compilation.

4/16/2021



Passing arrays to functions

```
int count[5];
```

- *count* is the name of the array and is the starting address of the array.

```
count[2] = 33;
```

- What happens is that 33 is stored in the memory location whose **address** is:
  (count + 2* sizeof (int))
- So, when we pass `count` to a function, then we are actually passing an address.
- Thus, the function which uses this address can, in fact, modify the original array!

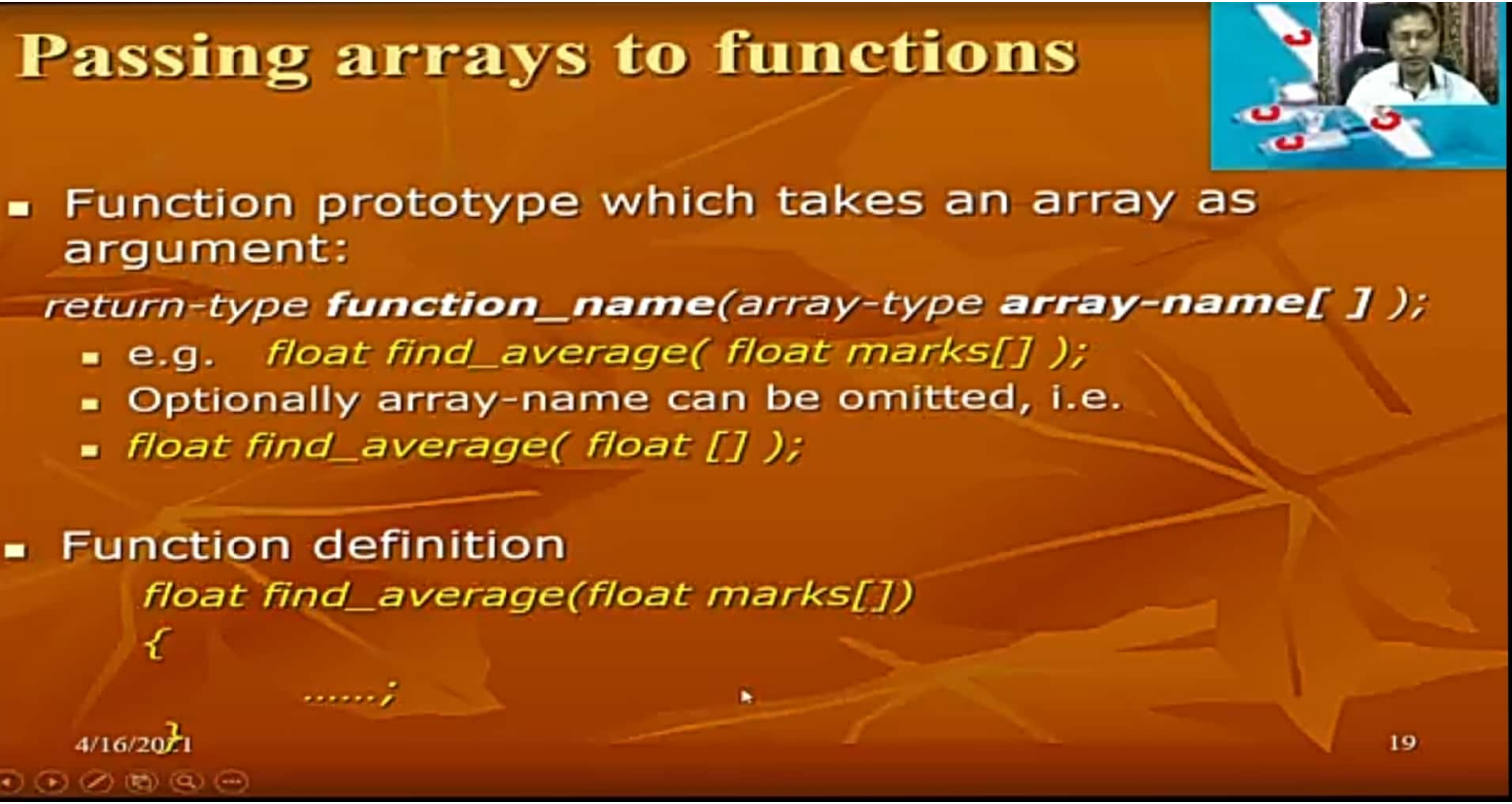
If sizeof(int) = 2 then:

count → count[0]

int

count+1 → count[1]

count+2 → count[2]



Passing arrays to functions

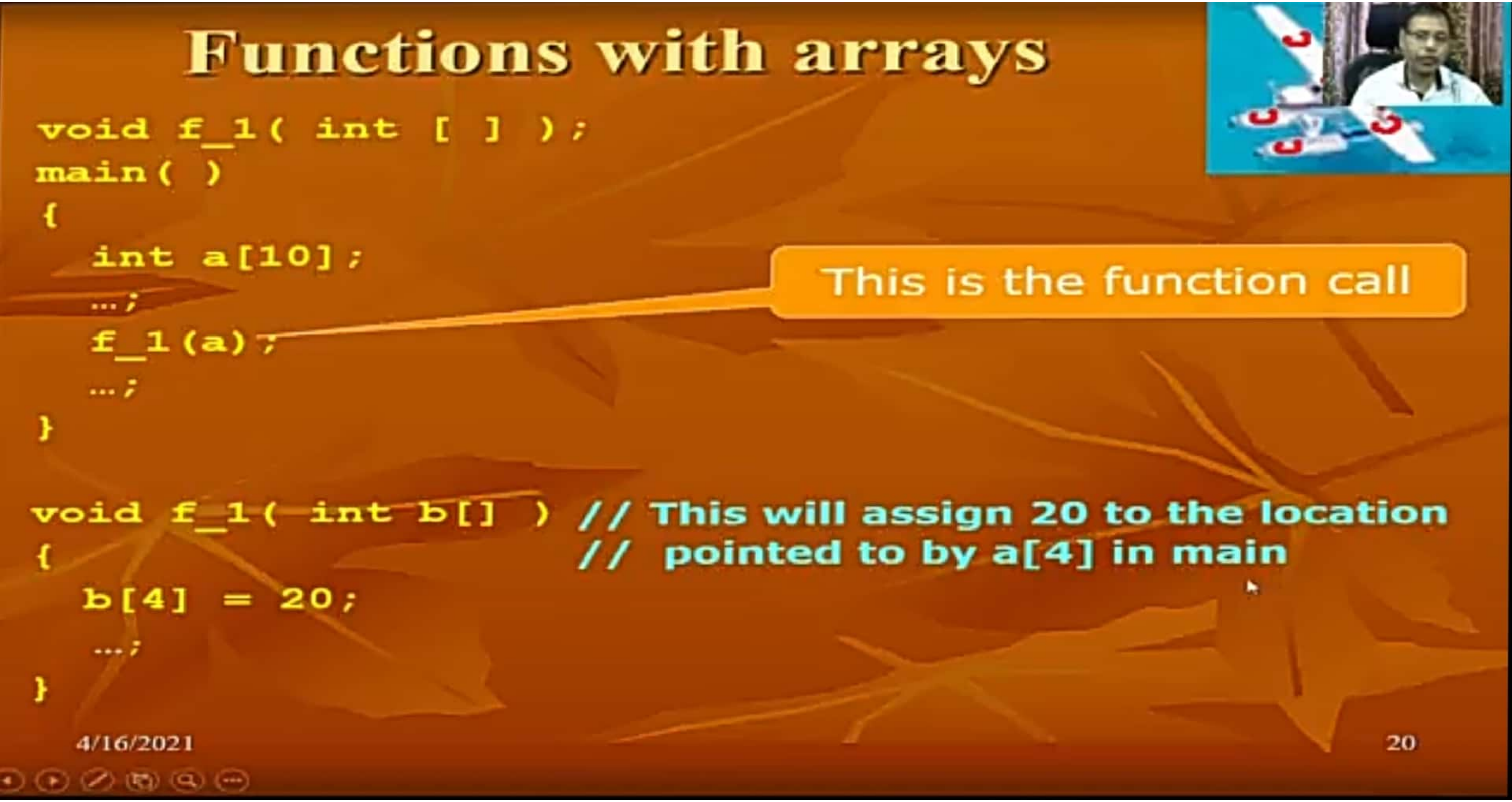
- Function prototype which takes an array as argument:

*return-type **function_name**(array-type **array-name[ ]**);*

  - e.g. *float find_average(float marks[]);*
  - Optionally array-name can be omitted, i.e.
  - *float find_average(float []);*

- Function definition

```
float find_average(float marks[])
{
        ......;
}
```

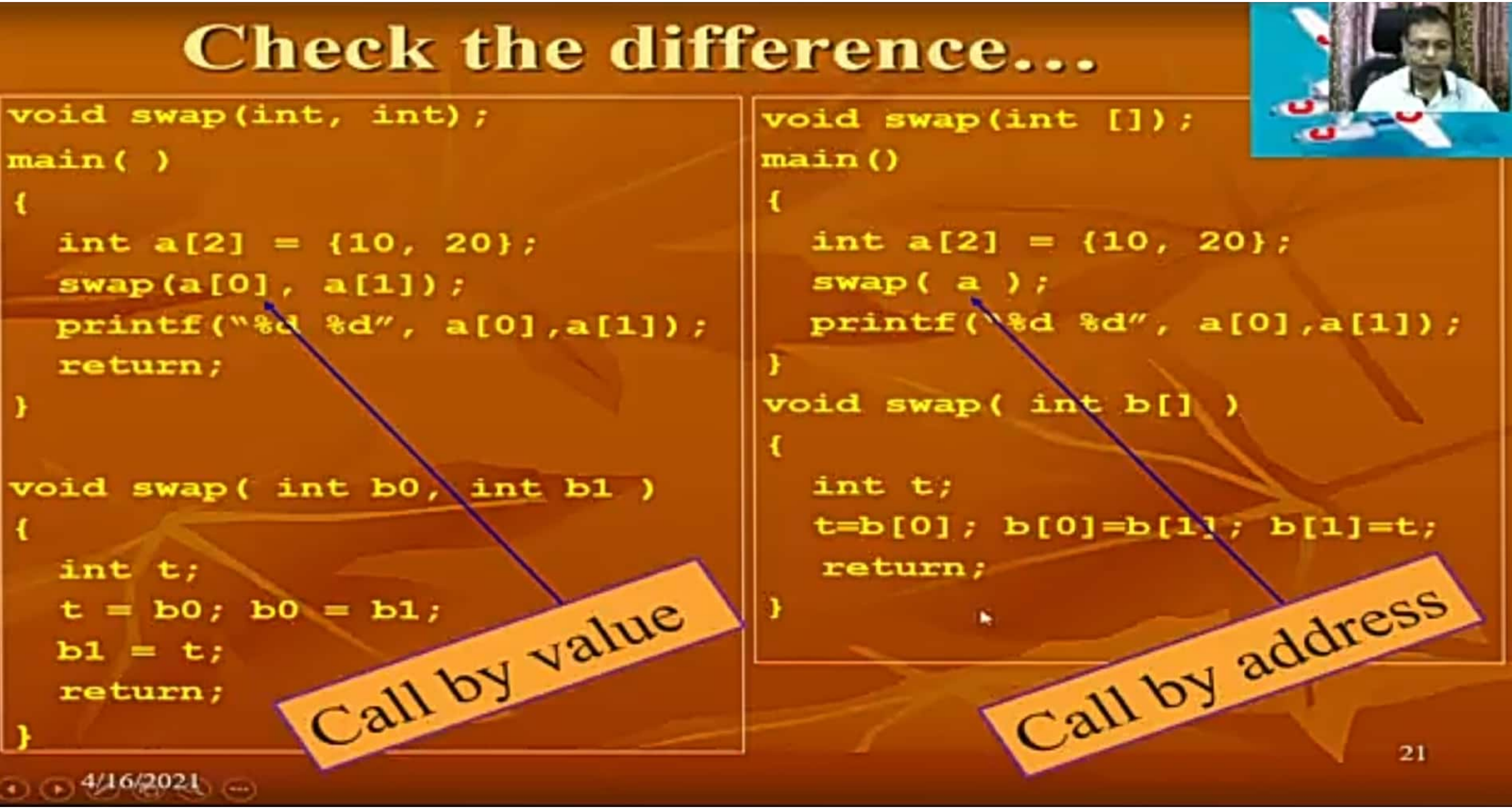


Functions with arrays

```
void f_1( int [ ] );
main( )
{
   int a[10];
   ...;
   f_1(a);
   ...;
}

void f_1( int b[] ) // This will assign 20 to the location
{                   //  pointed to by a[4] in main
  b[4] = 20;
   ...;
}
```

This is the function call



Check the difference…

```
void swap(int, int);
main( )
{
   int a[2] = {10, 20};
   swap(a[0], a[1]);
   printf(“%d %d”, a[0],a[1]);
   return;
}

void swap( int b0, int b1 )
{
   int t;
   t = b0; b0 = b1;
   b1 = t;
   return;
}
```

Call by value

```
void swap(int []);
main()
{
   int a[2] = {10, 20};
   swap( a );
   printf(“%d %d”, a[0],a[1]);
}
void swap( int b[] )
{
   int t;
   t=b[0]; b[0]=b[1]; b[1]=t;
   return;
}
```

Call by address

4/16/2021