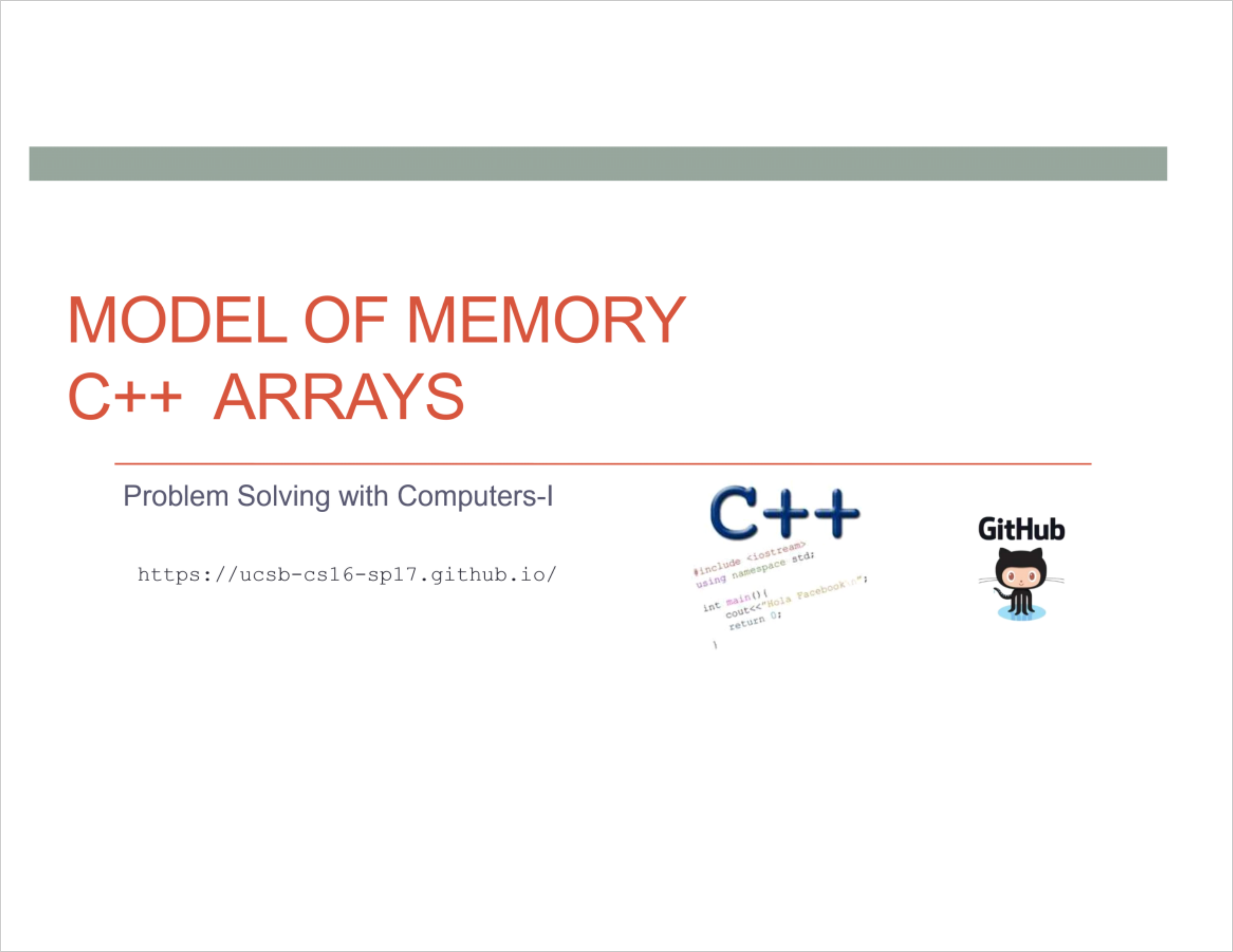
# Lecture 8

Monday, May 1, 2017 1:56 PM



## MODEL OF MEMORY
## C++ ARRAYS

Problem Solving with Computers-I

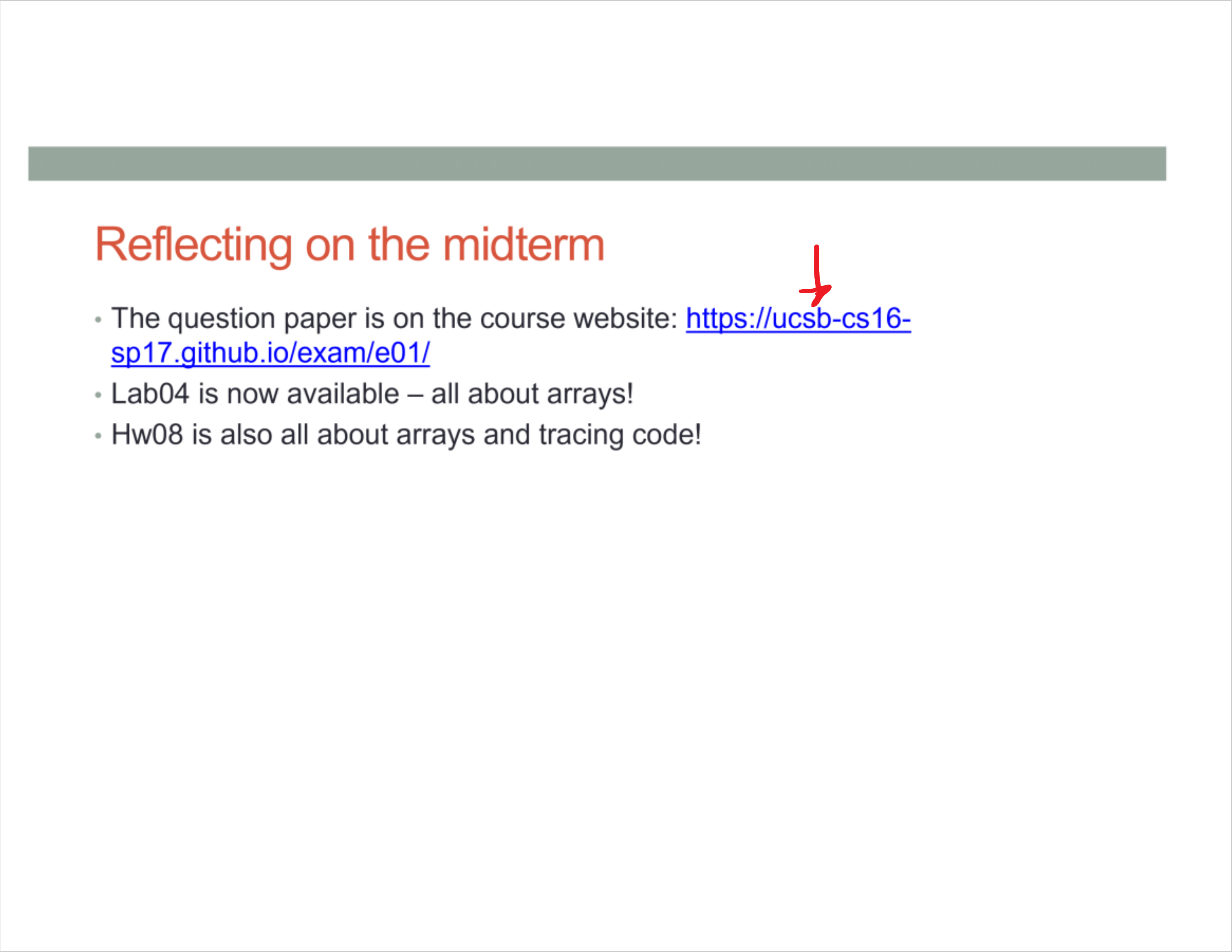
https://ucsb-cs16-sp17.github.io/

```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```

# Page 2

Monday, May 1, 2017 1:56 PM

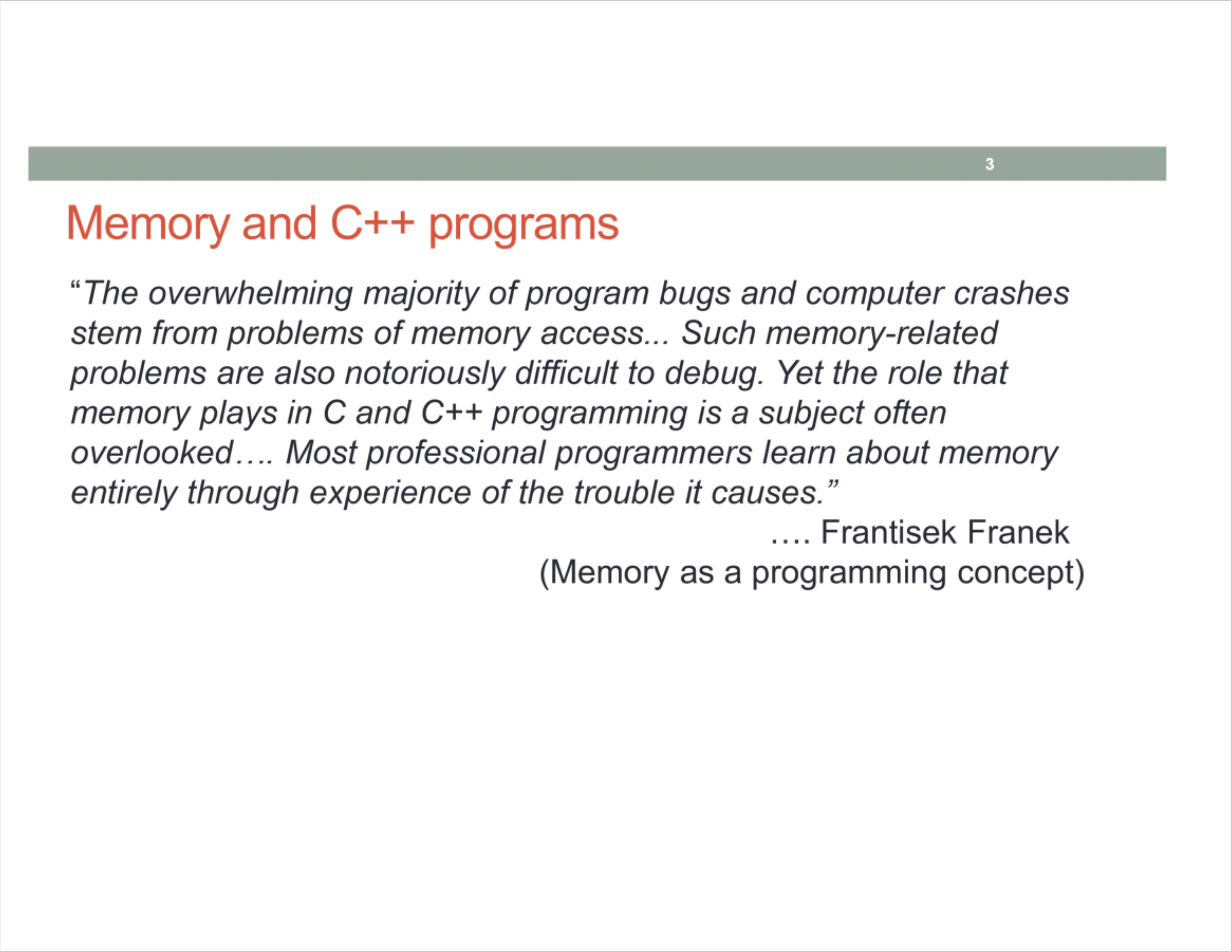


## Reflecting on the midterm

- The question paper is on the course website: https://ucsb-cs16-sp17.github.io/exam/e01/
- Lab04 is now available – all about arrays!
- Hw08 is also all about arrays and tracing code!

# Page 3

Monday, May 1, 2017 1:56 PM

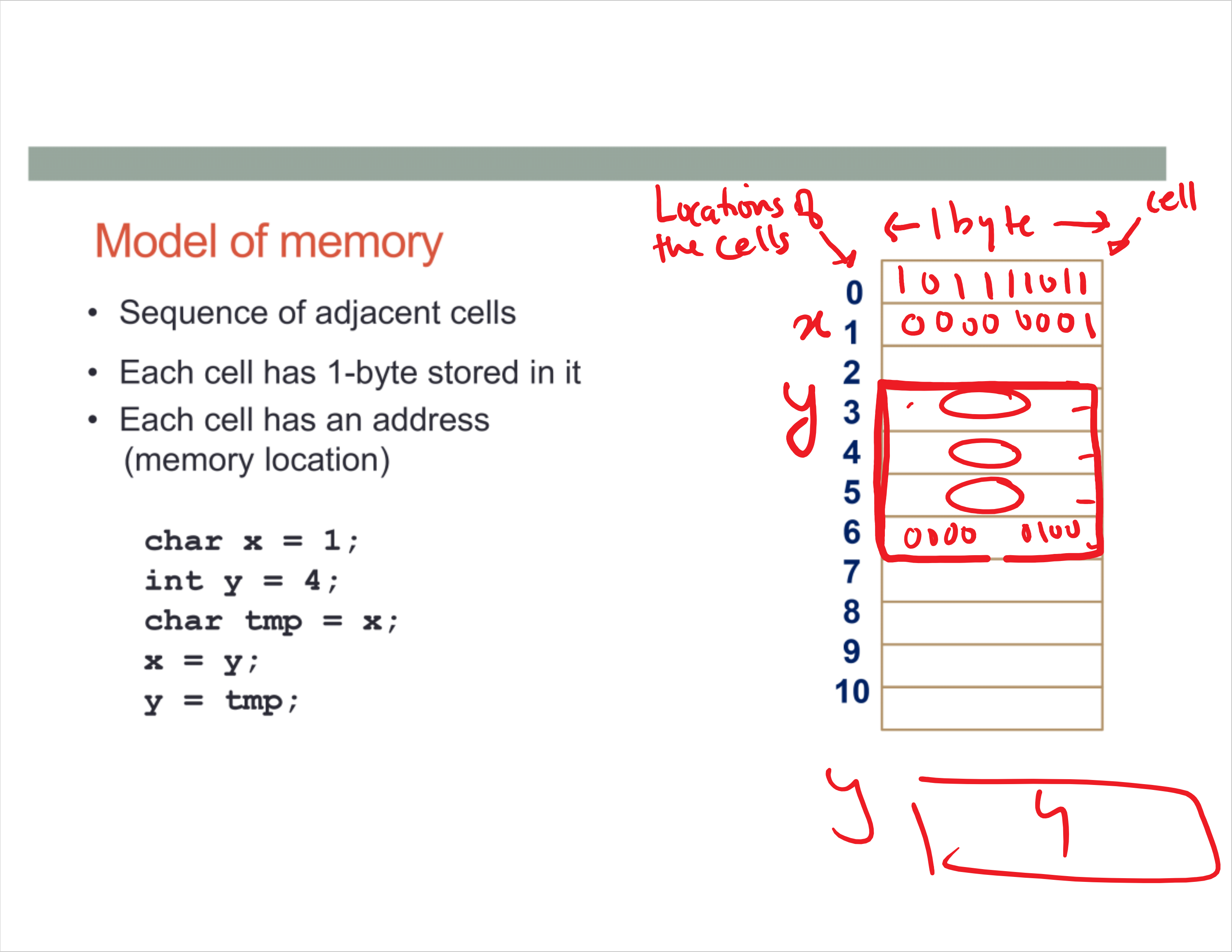


## Memory and C++ programs

*“The overwhelming majority of program bugs and computer crashes stem from problems of memory access... Such memory-related problems are also notoriously difficult to debug. Yet the role that memory plays in C and C++ programming is a subject often overlooked…. Most professional programmers learn about memory entirely through experience of the trouble it causes.”*

…. Frantisek Franek
(Memory as a programming concept)

Monday, May 1, 2017 1:56 PM



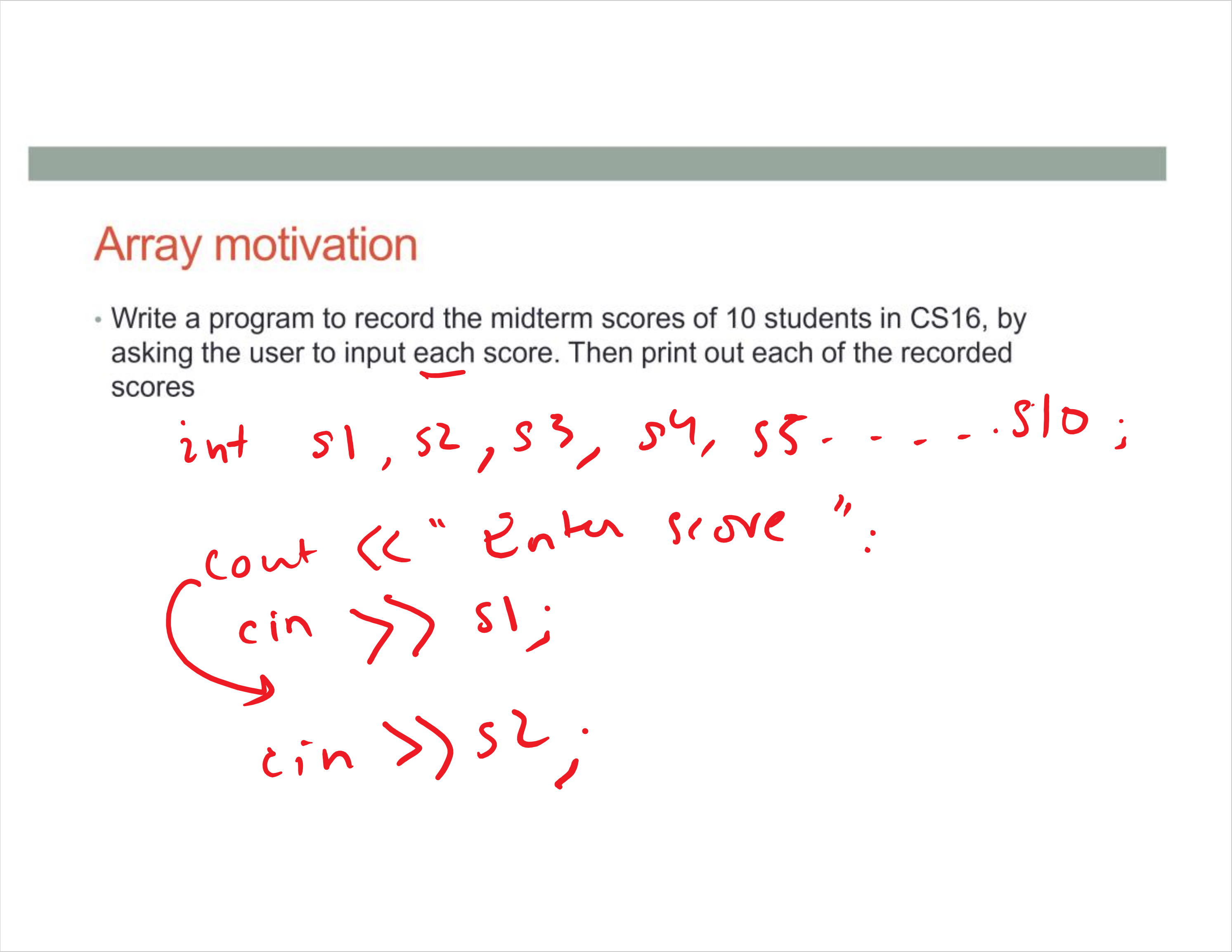
## Model of memory

- Sequence of adjacent cells
- Each cell has 1-byte stored in it
- Each cell has an address (memory location)

```
char x = 1;
int y = 4;
char tmp = x;
x = y;
y = tmp;
```

# Page 5

Monday, May 1, 2017 1:56 PM



## Array motivation

- Write a program to record the midterm scores of 10 students in CS16, by asking the user to input each score. Then print out each of the recorded scores

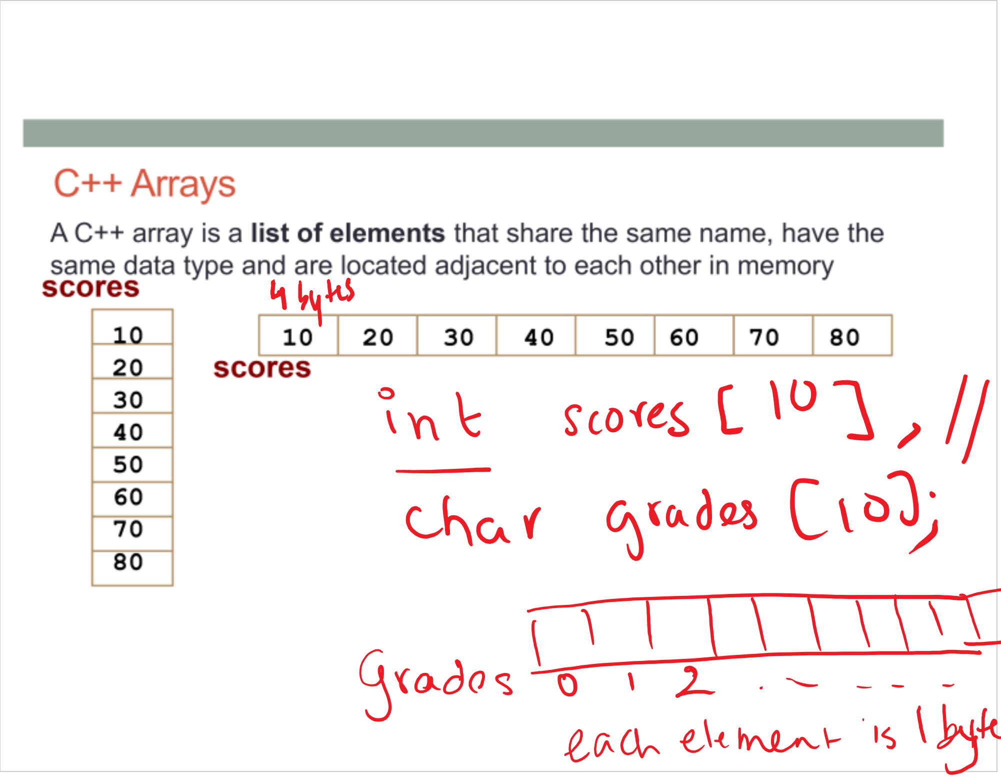
```
int s1, s2, s3, s4, s5 - - - - - .s10 ;

cout << " Enter score ";
  cin >> s1;

   cin >> s2;
```

# Page 6

Monday, May 1, 2017 1:56 PM



## C++ Arrays

A C++ array is a **list of elements** that share the same name, have the same data type and are located adjacent to each other in memory

scores

10
20
30
40
50
60
70
80

4 bytes

10 20 30 40 50 60 70 80

scores

int scores [ 10 ] , // Declaration

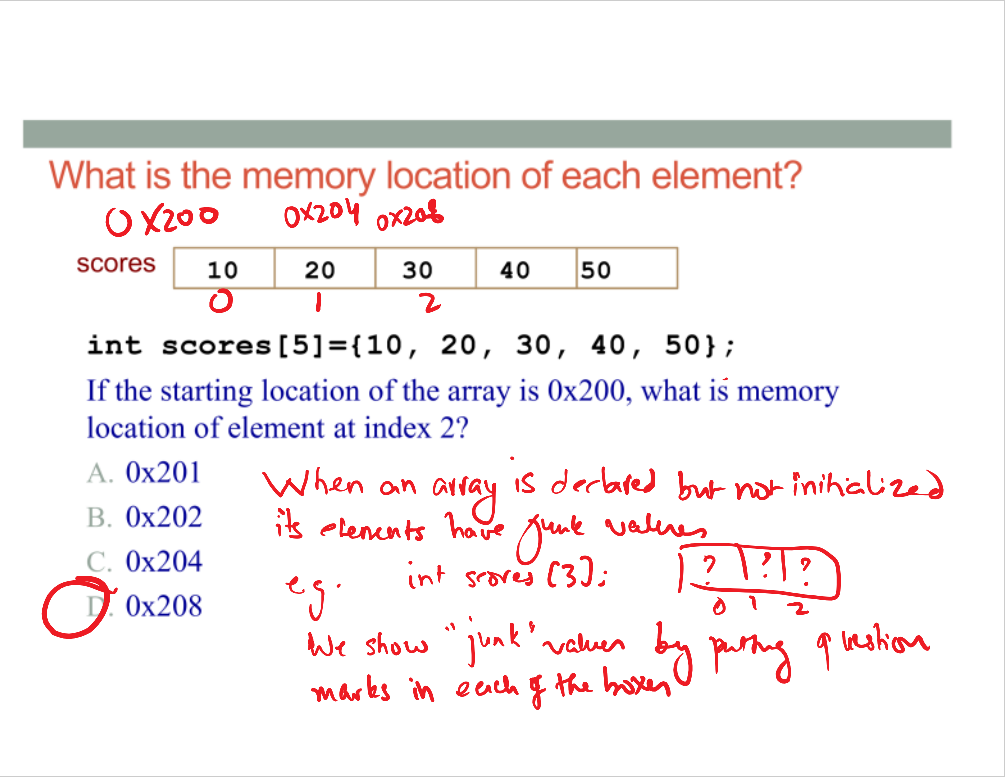
char grades [10];

grades 0 1 2 . . . . . 9

each element is 1 byte . Each byte has an address

# Page 7

Monday, May 1, 2017 1:56 PM



## What is the memory location of each element?

0x200 0x204 0x208

| scores | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|
| | 0 | 1 | 2 | | |

```
int scores[5]={10, 20, 30, 40, 50};
```

If the starting location of the array is 0x200, what is memory location of element at index 2?

A. 0x201

B. 0x202

C. 0x204

[✓ D.] 0x208

When an array is declared but not initialized its elements have junk values

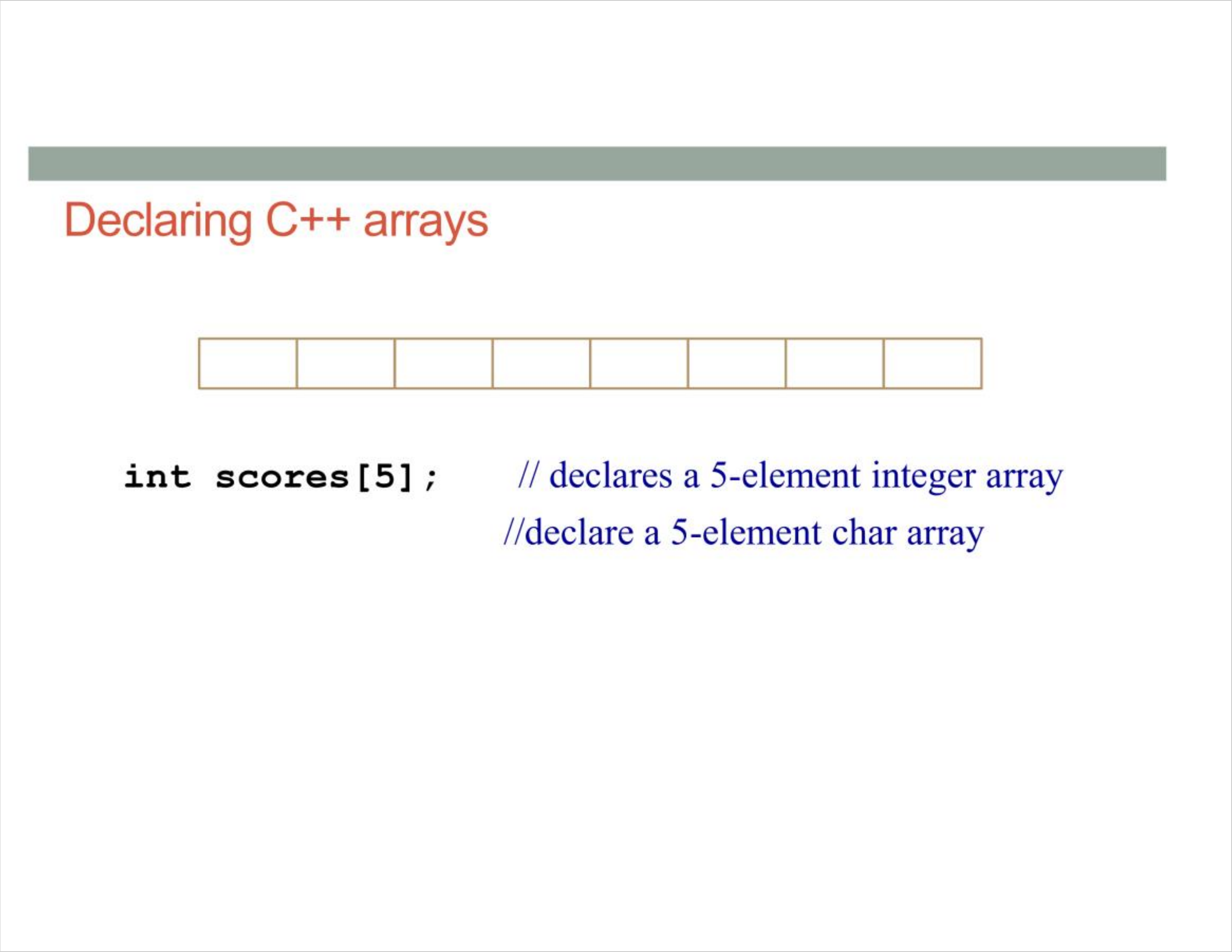
e.g. int scores[3];

| ? | ? | ? |
|---|---|---|
| 0 | 1 | 2 |

We show "junk" values by putting question marks in each of the boxes

# Page 8

Monday, May 1, 2017 1:56 PM

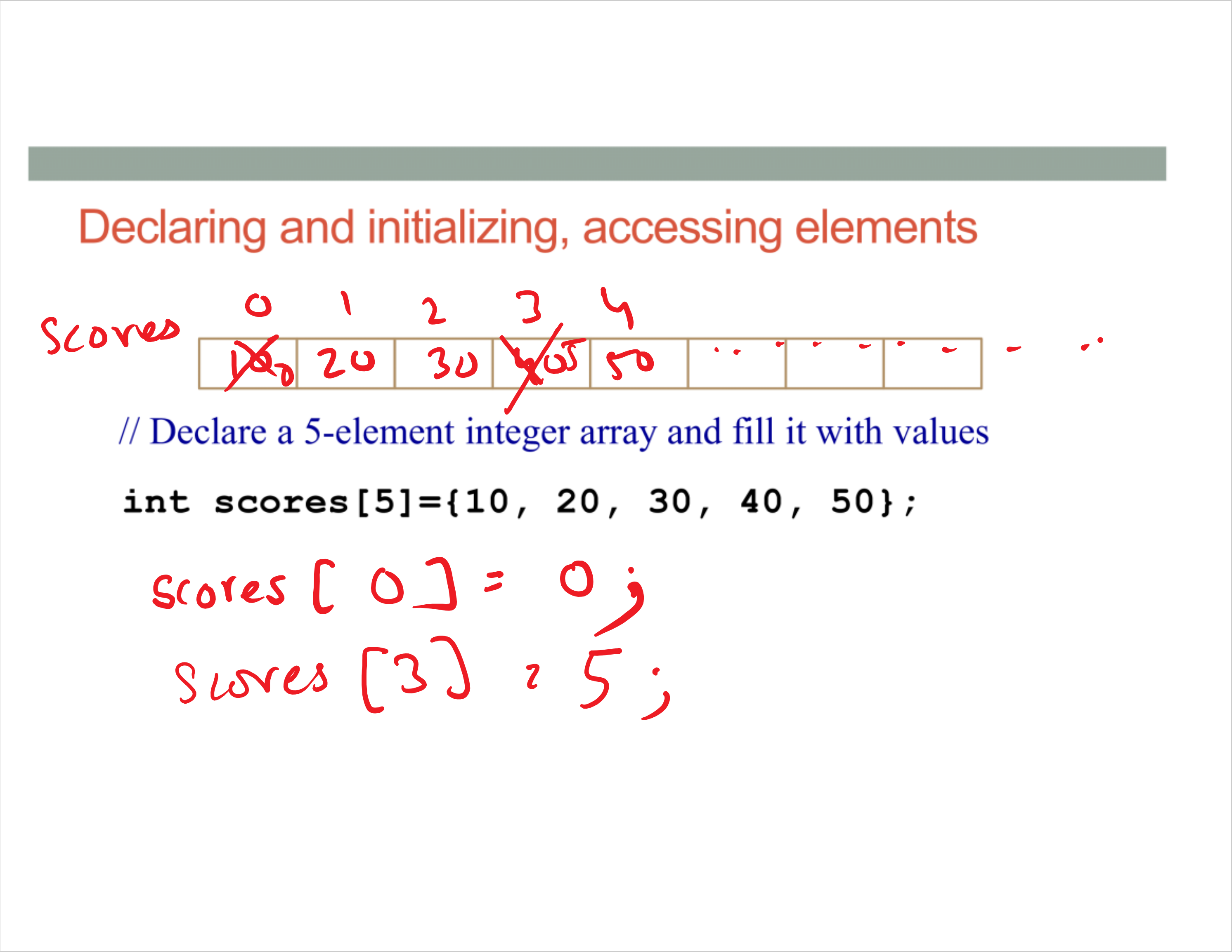


## Declaring C++ arrays

```
int scores[5];      // declares a 5-element integer array
                    //declare a 5-element char array
```

Page 9

Monday, May 1, 2017 1:56 PM



## Declaring and initializing, accessing elements

Scores

| 0 | 1 | 2 | 3 | 4 | | | |
|---|---|---|---|---|---|---|---|
| ~~10~~ 0 | 20 | 30 | ~~40~~ 5 | 50 | . . | . . | . . |

// Declare a 5-element integer array and fill it with values

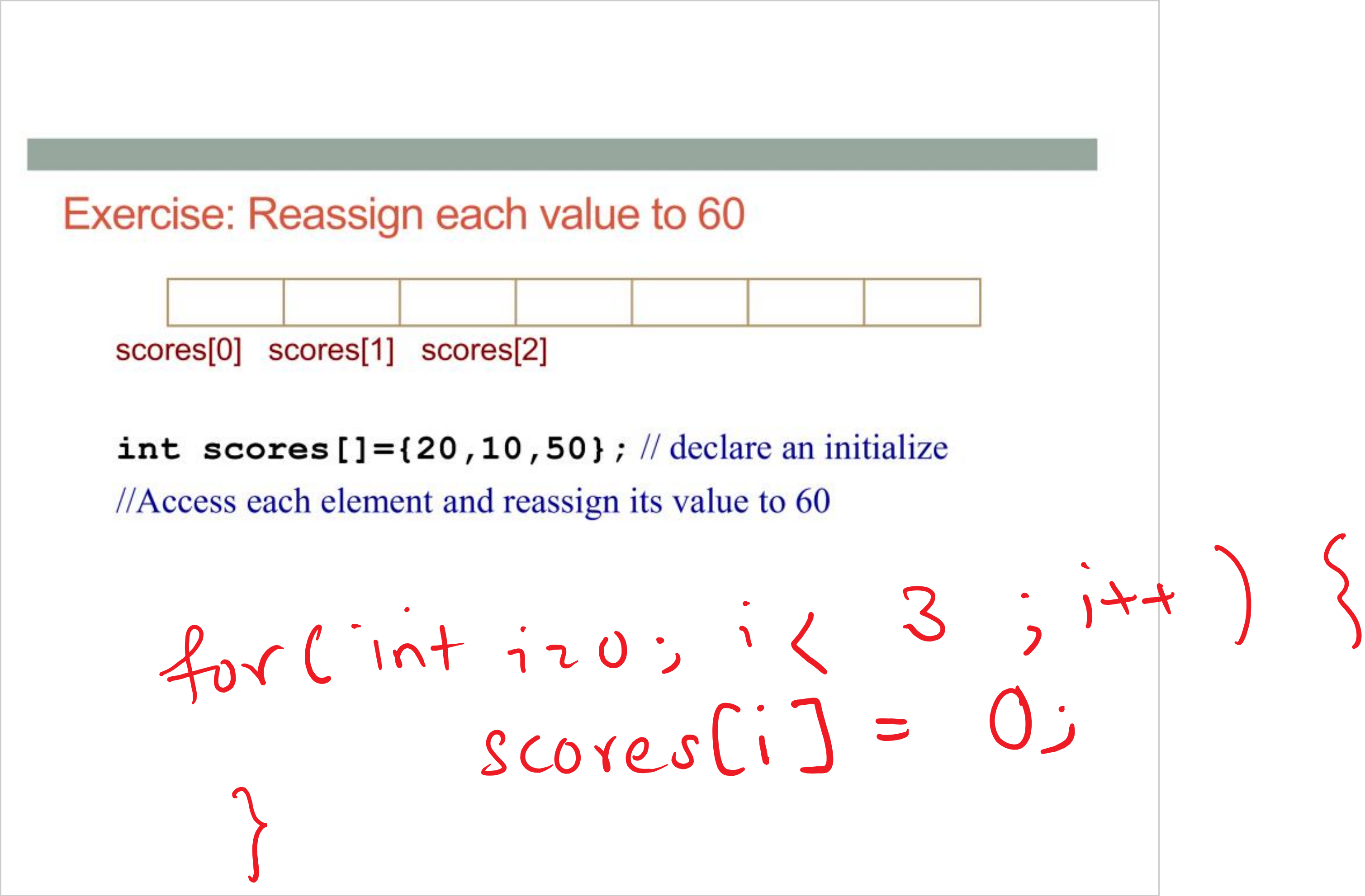
```
int scores[5]={10, 20, 30, 40, 50};
```

scores [ 0 ] = 0 ;

scores [3] = 5;

# Page 10

Monday, May 1, 2017 1:56 PM

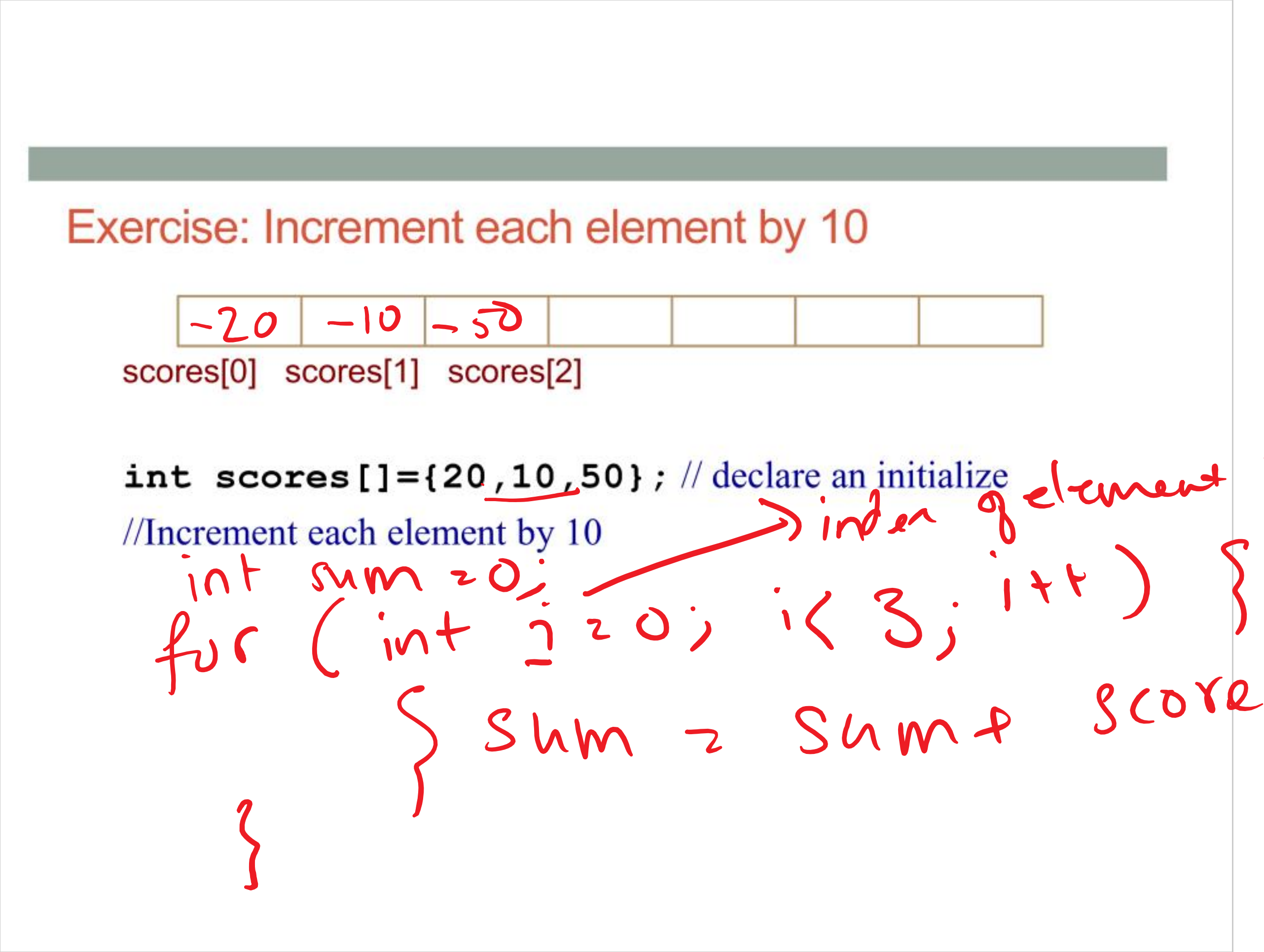


## Exercise: Reassign each value to 60

scores[0]
scores[1]
scores[2]

```
int scores[]={20,10,50}; // declare an initialize
//Access each element and reassign its value to 60
```

```
for(int i=0; i< 3 ;i++){
        scores[i] = 0;
}
```



## Exercise: Increment each element by 10

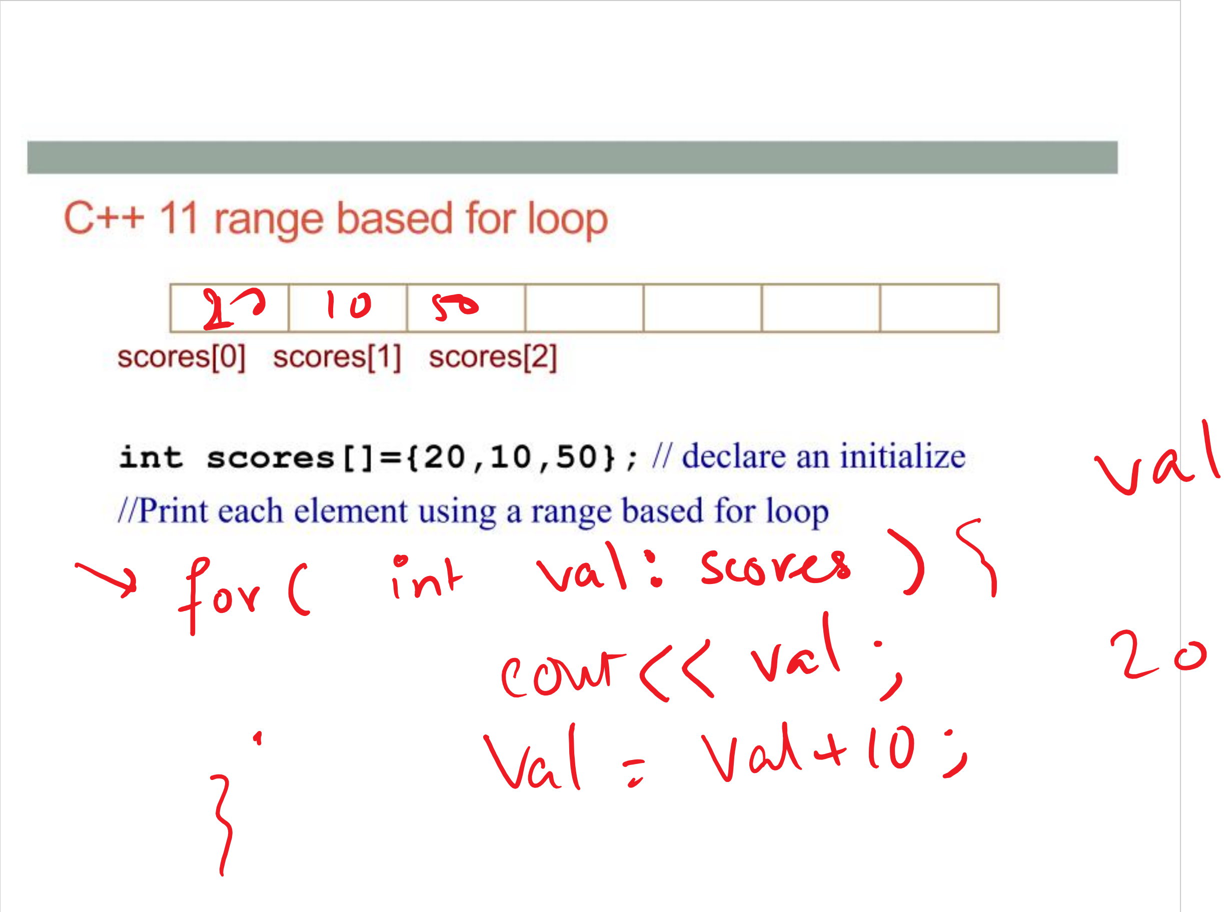
| | | | | | | |
|---|---|---|---|---|---|---|
| -20 | -10 | -50 | | | | |
| scores[0] | scores[1] | scores[2] | | | | |

```
int scores[]={20,10,50}; // declare an initialize
//Increment each element by 10
int sum =0;
for ( int i =0; i< 3; i++ ) {
    { sum = sum+ scores[i];
}
```

index of element in array

Page 12

Monday, May 1, 2017 1:56 PM



## C++ 11 range based for loop

| | | | | | | |
|---|---|---|---|---|---|---|
| 20 | 10 | 50 | | | | |
| scores[0] | scores[1] | scores[2] | | | | |

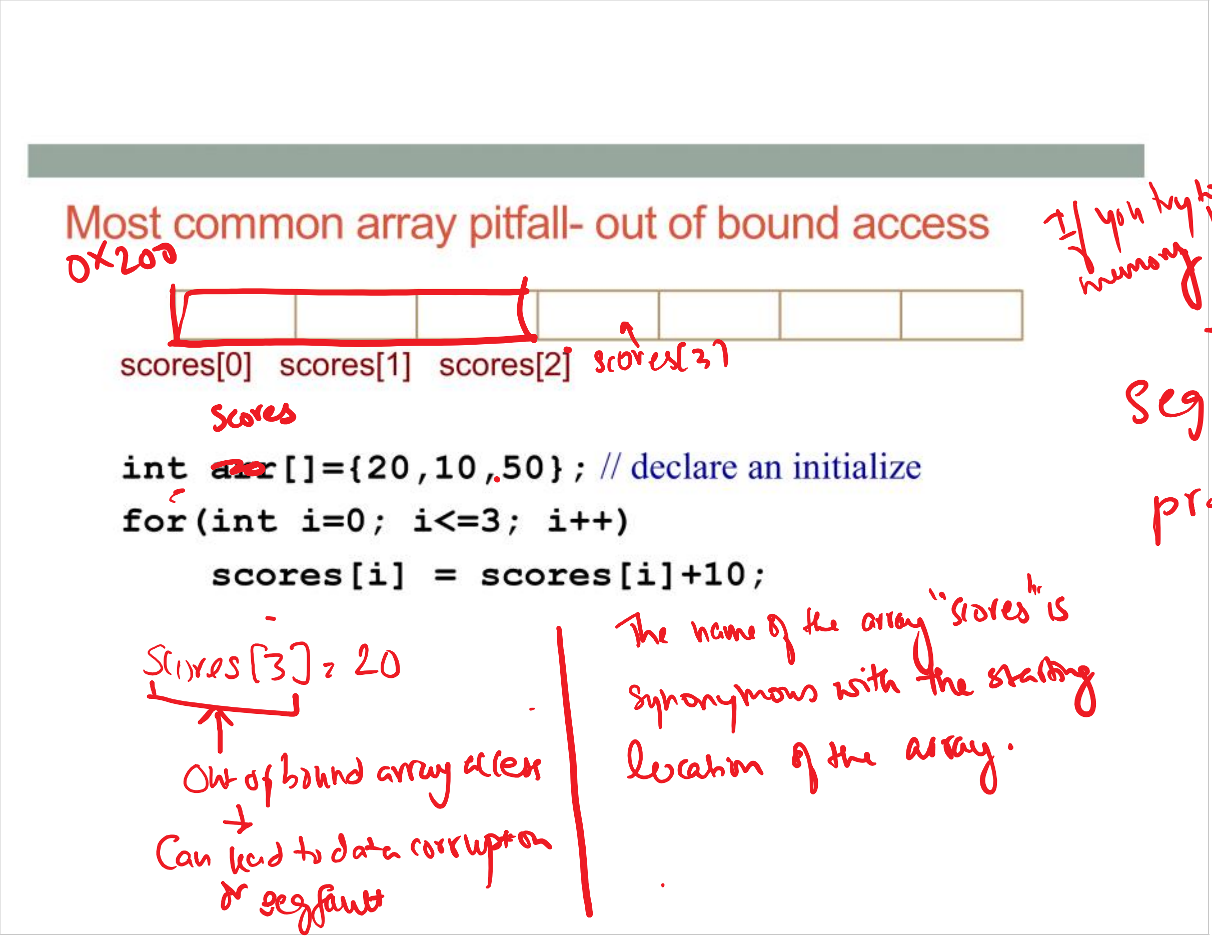
`int scores[]={20,10,50};` // declare an initialize

//Print each element using a range based for loop

```
→ for( int val: scores ) {
            cout<< val;
            Val = Val+10;
  }
```

val 1~~2~~010

20 10 50



## Most common array pitfall- out of bound access

scores

```
int arr[]={20,10,50}; // declare an initialize
for(int i=0; i<=3; i++)
    scores[i] = scores[i]+10;
```

scores[3] = 20

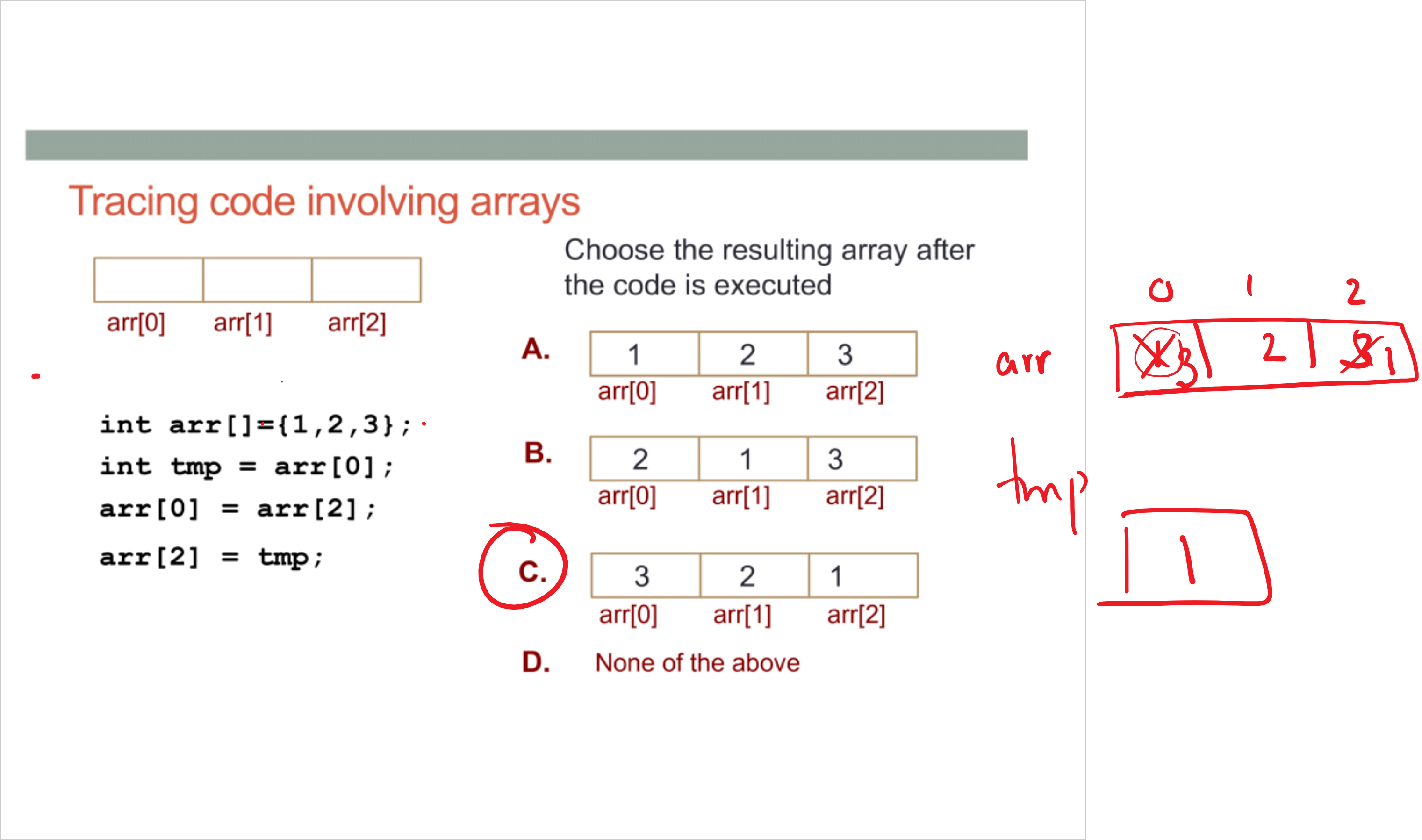
Out of bound array access
→ Can lead to data corruption or segfault

The name of the array "scores" is synonymous with the starting location of the array.

If you try to access a memory location that you don't have permission on
→ Segfault
→ program crashes

Page 14

Monday, May 1, 2017 1:56 PM



## Tracing code involving arrays

```
int arr[]={1,2,3};
int tmp = arr[0];
arr[0] = arr[2];
arr[2] = tmp;
```

Choose the resulting array after the code is executed

A.

| 1 | 2 | 3 |
|---|---|---|
| arr[0] | arr[1] | arr[2] |

B.

| 2 | 1 | 3 |
|---|---|---|
| arr[0] | arr[1] | arr[2] |

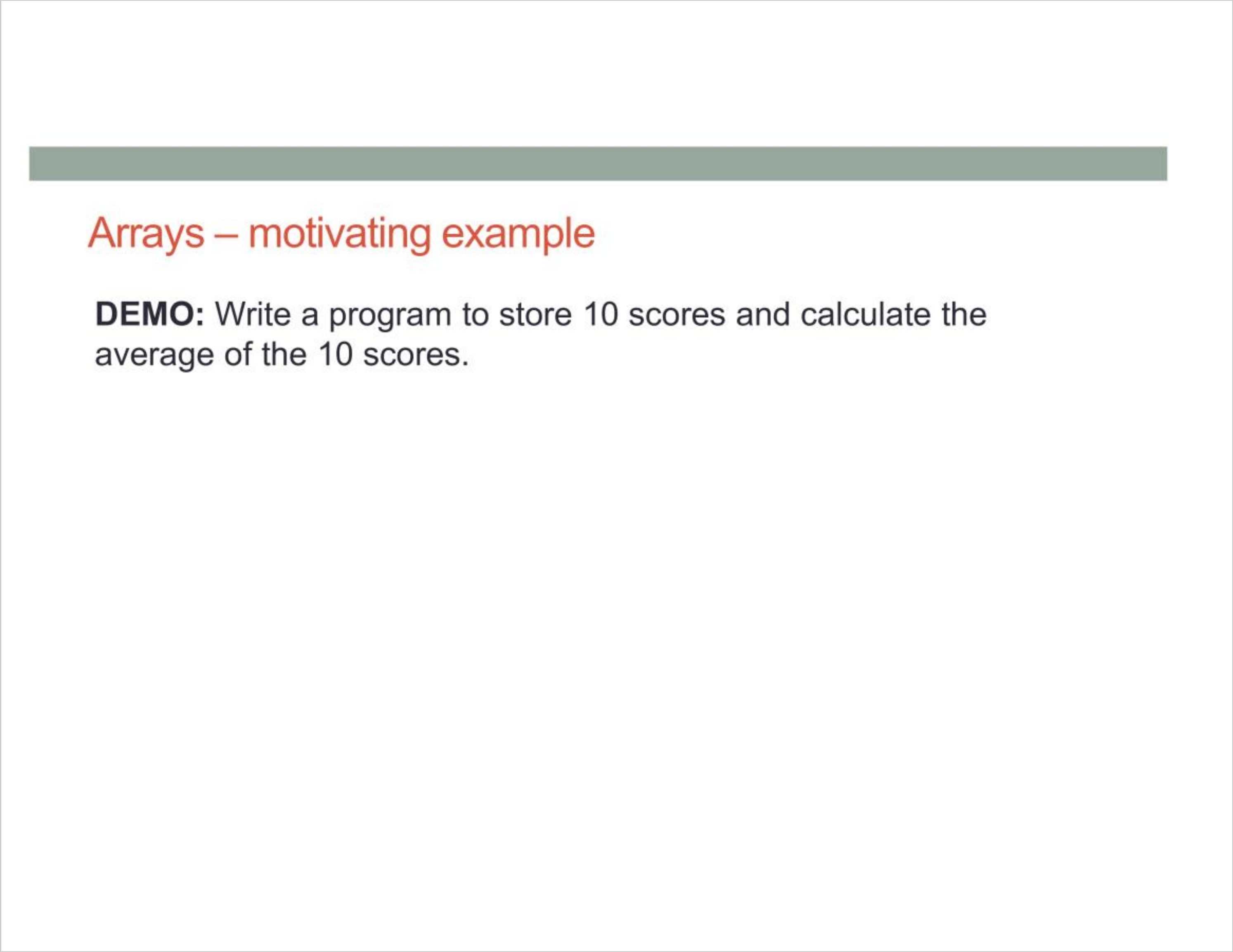
[✓ C.]

| 3 | 2 | 1 |
|---|---|---|
| arr[0] | arr[1] | arr[2] |

D. None of the above

# Page 15

Monday, May 1, 2017 1:56 PM

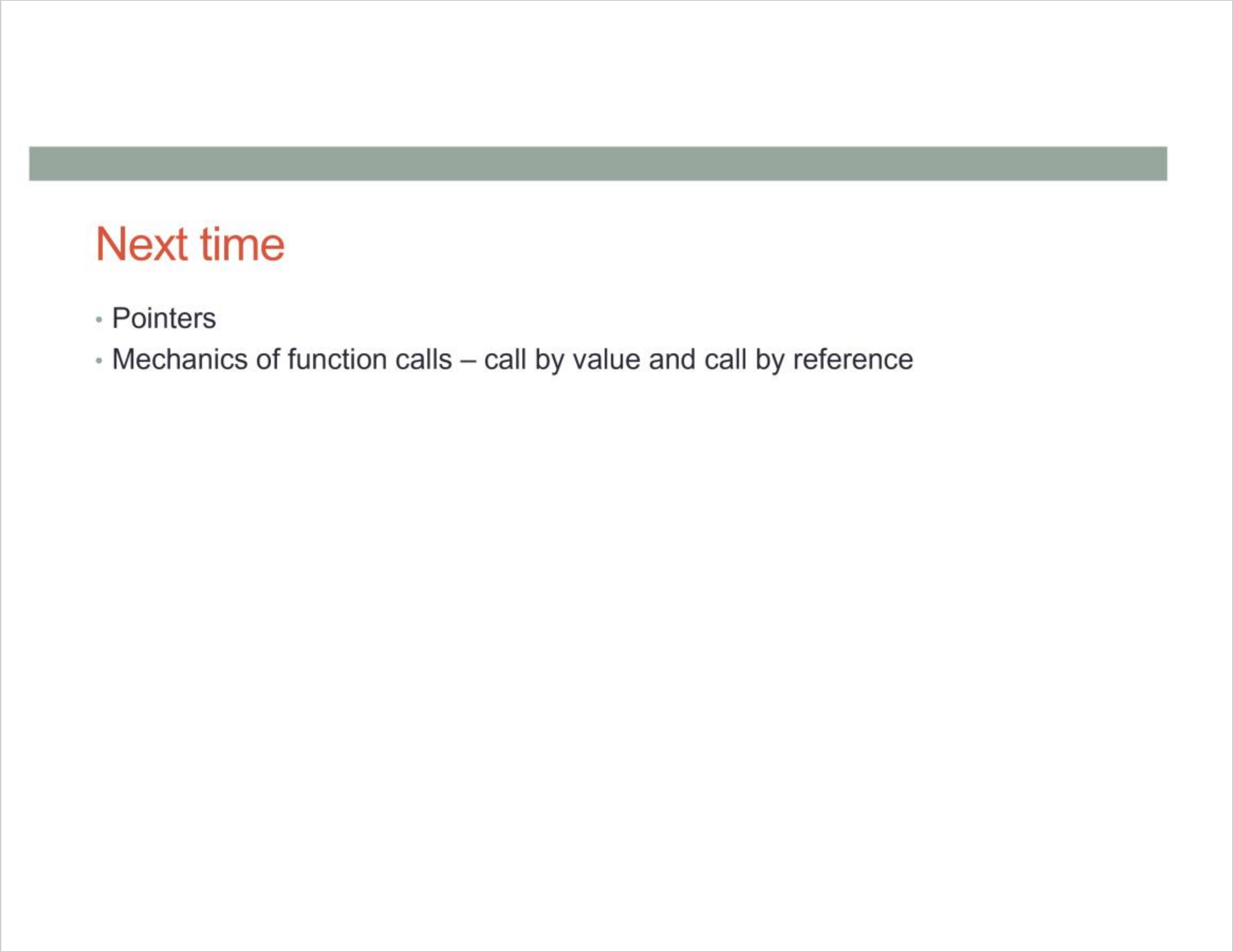


## Arrays – motivating example

**DEMO:** Write a program to store 10 scores and calculate the average of the 10 scores.

# Page 16

Monday, May 1, 2017 1:56 PM



## Next time

- Pointers
- Mechanics of function calls – call by value and call by reference