VISUALIZING PROGRAM DYNAMICS ARRAYS

Problem Solving with Computers-I

https://ucsb-cs16-wi17.github.io/

Reflecting on the midterm

- The question paper is on the course website: https://ucsb-cs16-wi17.github.io/exam/e01/
- Overall – it was a good performance! Mean: 85.57%, median 87.33%, std. deviation: 9.68%
- Lab04 is now available – all about arrays!
- Hw08 is also all about arrays and tracing code!

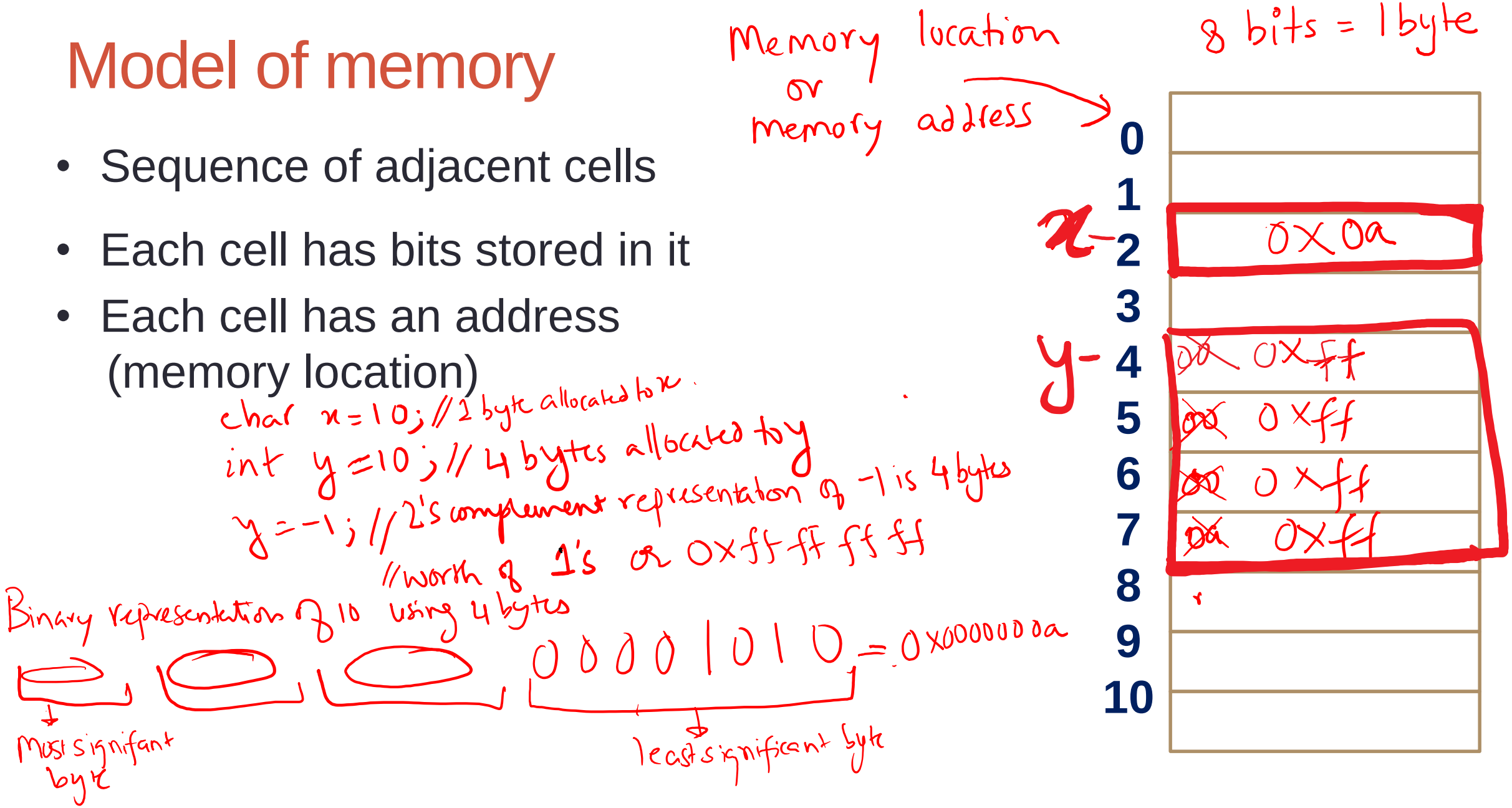
Memory and C++ programs

"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)

Model of memory

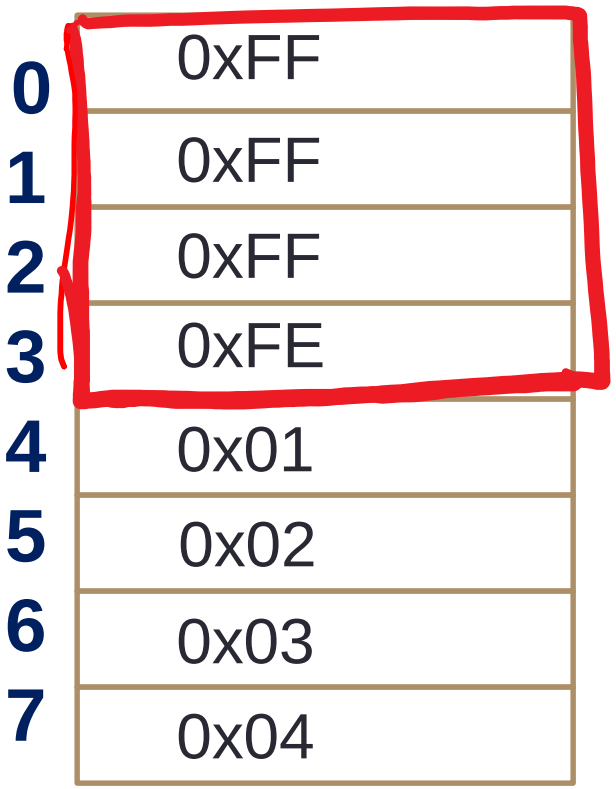
- Sequence of adjacent cells
- Each cell has bits stored in it
- Each cell has an address (memory location)



Interaction of programs with memory

Consider the declaration: `int x; //Assume starting location of 'x' is 0`

Memory map below would result from which of the following C++ statements?



State of memory after code execution

A. `int x = 0xFF;`

B. `int x = 0xFE;`

C. `int x = 0xFFFFFFFE;`

D. `int x = -2;`

E. **Both C and D**

00000FF

2's complement representation of -2

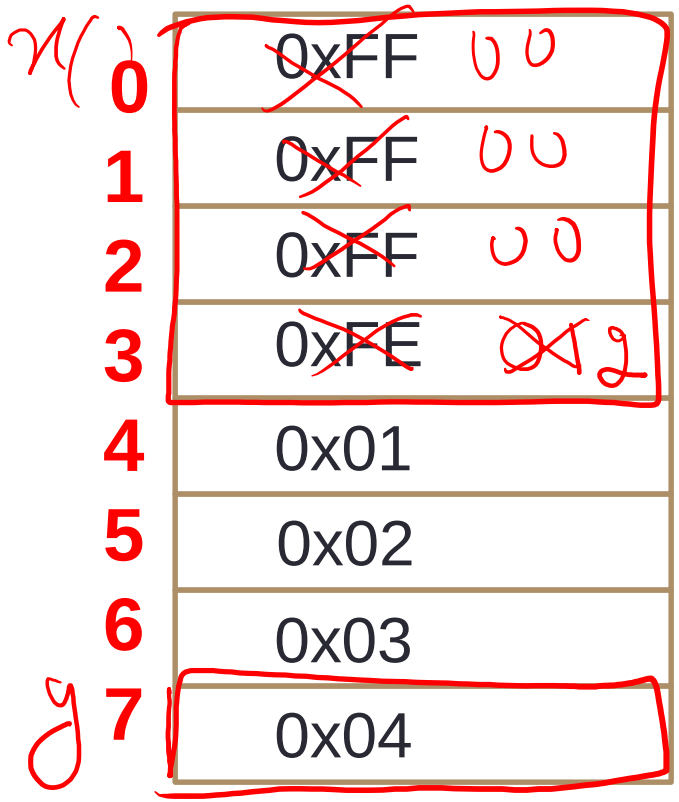
-1 ≡ 11111111 11111111 11111111 11111111

= 0xffffffff

-2 = -1 - 1 = 0xfffffffe

Tracing code

Show how the state of memory is modified when the following C++ code is executed?



State of memory

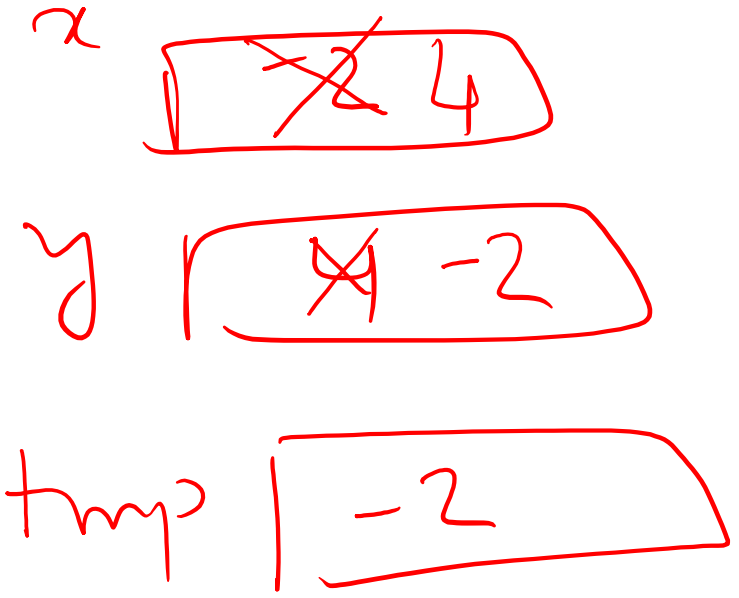
```
int x = 1;  // Assume x is at location 0
char y;     // Assume y is at location 7
if(x>0)
    x++;
else
    y++;
```

Drawing memory maps to trace code

- Trace the code below by drawing memory diagrams
- Choose the level of abstraction in your diagram that's right for this context!

0	0xFF
1	0xFF
2	0xFF
3	0xFE
4	0xA1
5	0xC2
6	0x00
7	0x04

```
char x = -2;
char y = 4;
char tmp = x;
x = y;
y = tmp;
```

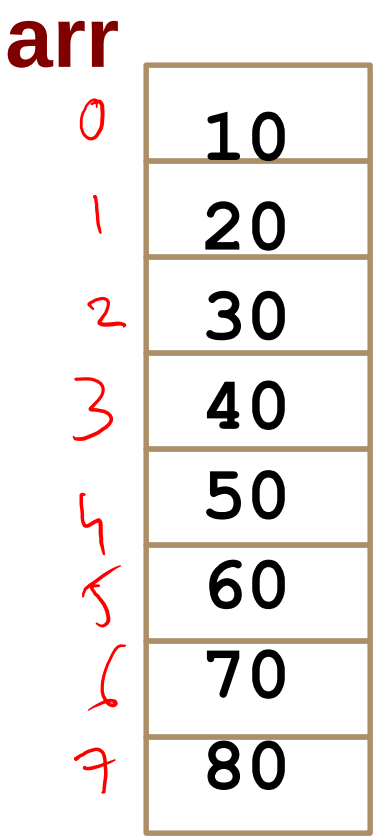


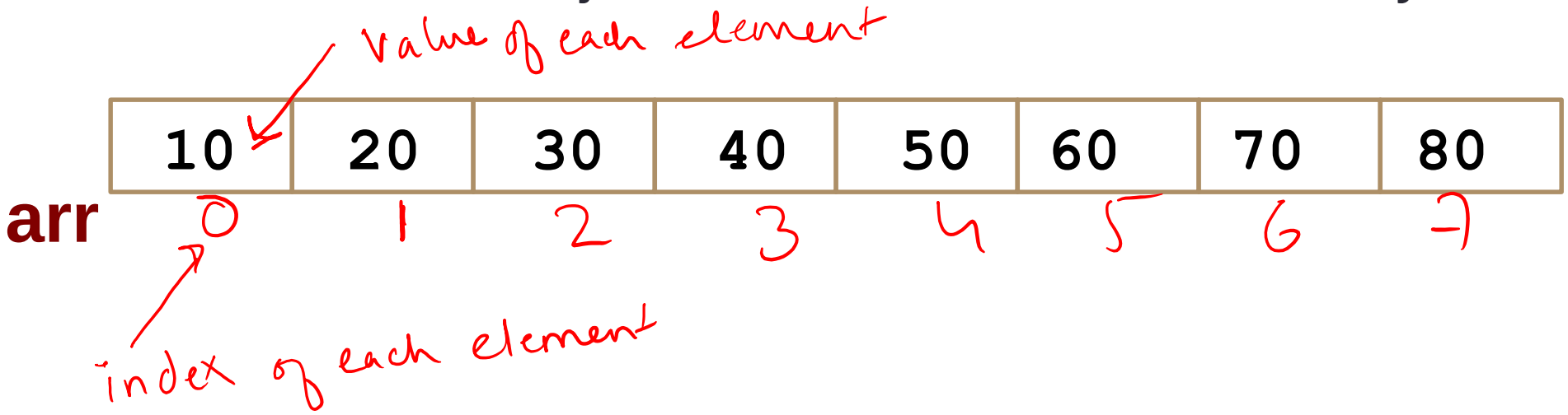
By tracing the code you can deduce what the code is doing

In this case its swapping the values of x and y

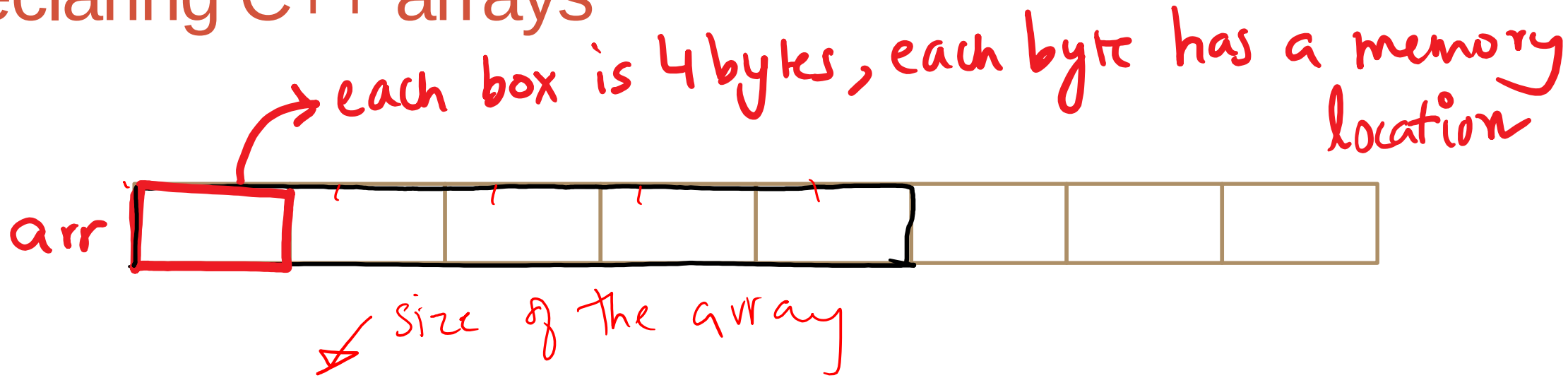
C++ Arrays

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





Declaring C++ arrays

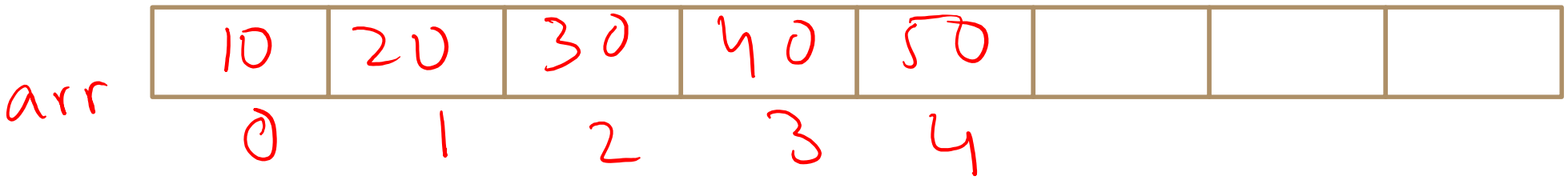



```
int arr[5];          // declares a 5-element integer array
```

type of each element

name of the array

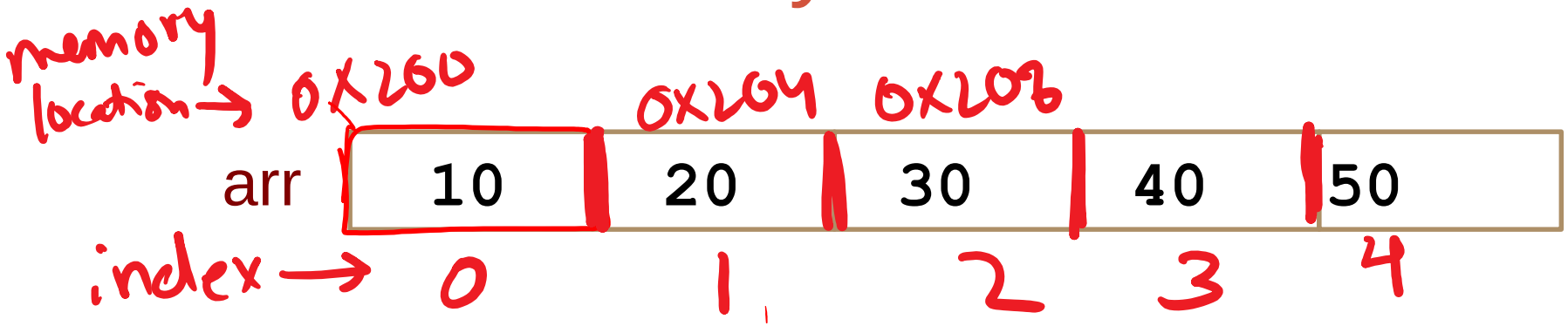
Declaring and initializing C++ arrays



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

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

What is the memory location of each element?




```
int arr[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

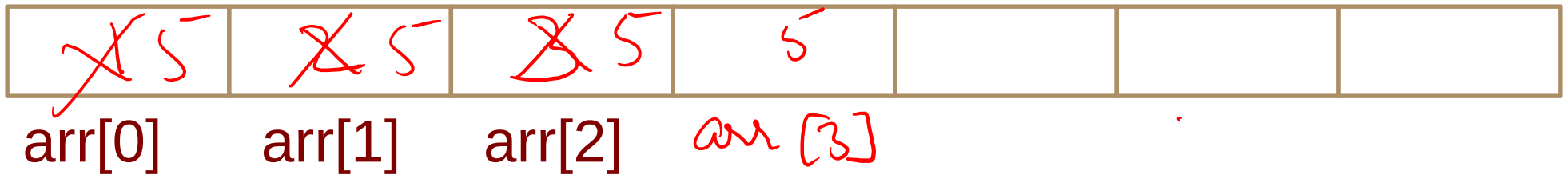
C. 0x204

D. 0x208 = 0x200 + 2*4

The starting location of an array is the location/address of element at index 0

Given the starting location, the address of any element "i" can be computed

Accessing elements of an array

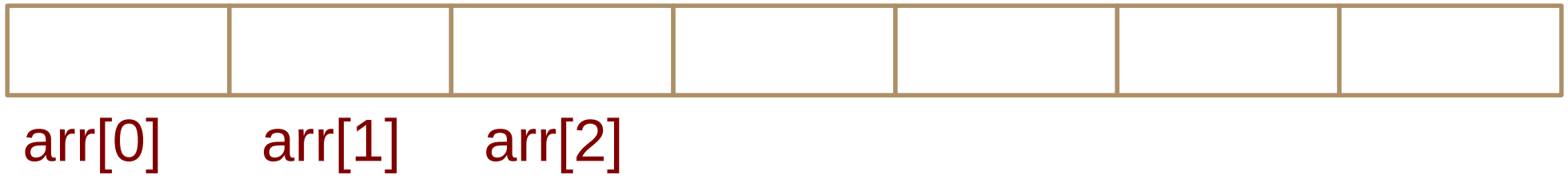



```
int arr[]={1,2,3}; // declare an initialize
```

//Access each element and reassign its value to 5

```
arr[0] = 5;
arr[1] = 5;
arr[2] = 5;
arr[3] = 5; //out of bound array access - bad things will happen!
```

Most common array pitfall- out of bound access




```
int arr[]={1,2,3}; // declare an initialize
arr[3] = 5;
```

Using variables as array subscripts


```
int arr[]={1,2,3};
//increment each element of the array
```

arr[0] = arr[0] + 1;

or arr[0]++;

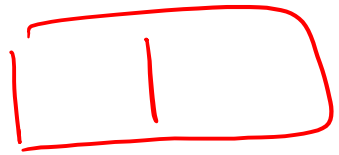
Use a loop to iterate through the array

```
for(int i=0;i<3;i++)
    arr[i]++;
```

Loop variable "i" is used to index into the array

Tracing code involving arrays

tmp




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

// Make a deduction about what the code does based on the outcome of your trace

Choose the resulting array after the code is executed

A.

B.

C.

D. None of the above

Arrays - motivation

DEMO: Write a program to store 5 scores and calculate the average of the 5 scores.

Next time

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