CHAPTER 6
ARRAYS

CMPS 148

Lab #3

°F to °C	Deduct 32, then multiply by 5, then divide by 9
°C to °F	Multiply by 9, then divide by 5, then add 32

Write a program to generate the following table using two functions

```
double celsiusToFahrenheit(double c)
double fahrenheitToCelsius(double f)
```

Implement these two functions in a header/implementation file set (temp.h, temp.cpp)

Celsius	Fahrenheit	Fahrenheit	Celsius
40.0	104.0	120.0	48.89
39.0	102.2	110.0	43.33
38.0	100.4	100.0	37.78
37.0	98.6	90.0	32.22
36.0	96.8	80.0	26.67
35.0	95.0	70.0	21.11
34.0	93.2	60.0	15.56
33.0	91.4	50.0	10.00
32.0	89.6	40.0	4.44
31.0	87.8	30.0	-1.11

Today's Topics

- Working with Arrays
- Arrays with Functions
- Sorted Lists
  - Sort while reading
  - Sort after reading them all
- Binary Search

Why do we need arrays?

- Programs often need to store collections of items (numbers, characters, etc)

- Keeping track of many variables is error-prone and a headache

C++ Arrays

- Arrays have a type, name, and size
- Array Declaration:

```
int myArray[10] // 10 integers
double x[20] // 20 doubles
char y[1000]// 1000 characters
```

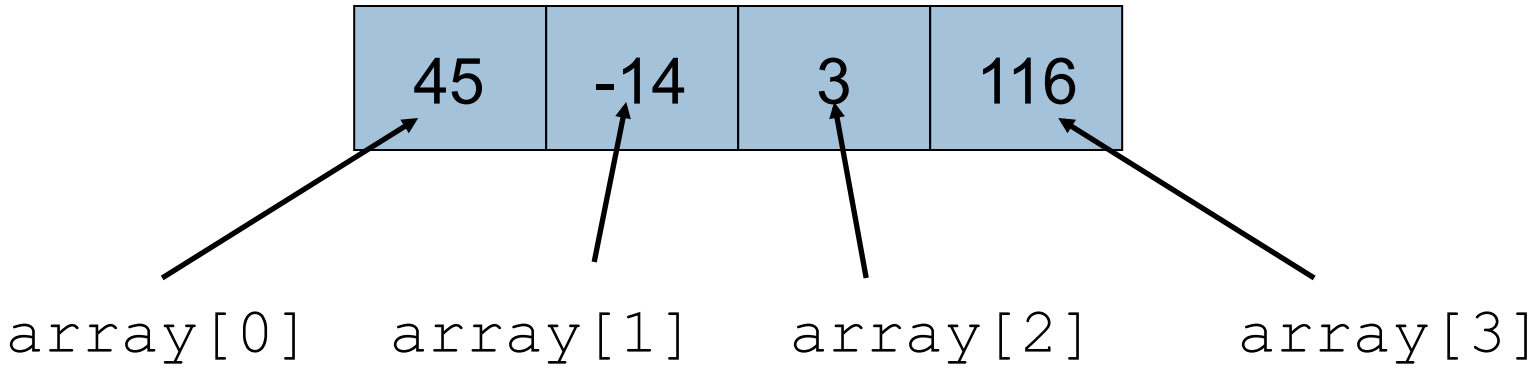
- Initialization:

```
int array[3] = { 1, 12, 65 };
```

Using Arrays

- You can access an individual *element* in an array using its ***index***.

```
int array[4] = {45, -15, 3, 116};
```



- The index **ALWAYS** starts at 0

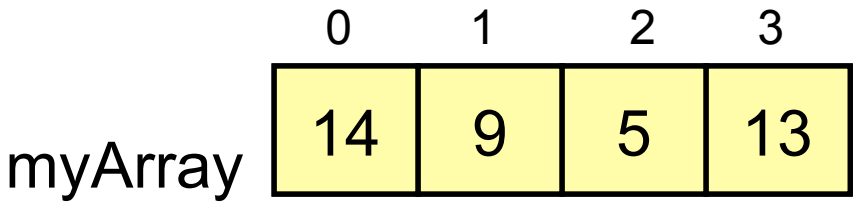
Syntax Rules

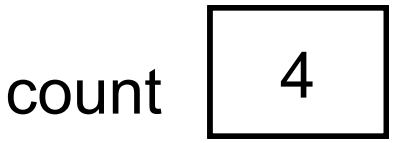
```
const int SIZE = 6;
int x = 5;
int array1[3];     // OKAY
int array2[x];    // NOT OKAY!
int array3[SIZE];  // GOOD
int array4[0];  // NO!

array1 = 5;  // VERY BAD!
array1[0] = 5;  // OK
array1[3] = 6;  //  ?
```

Partially filled Arrays

- Often, we do not know in advance *exactly* how many items the user will enter…
- We **cannot** change the size of the array - we must declare a size that is *big enough* for most situations
- Instead, we allocate a maximum size,
  - Use an independent variable (often called "count" to keep track of how many elements are used
  - array[count++] = 5;

Partially Filled Arrays





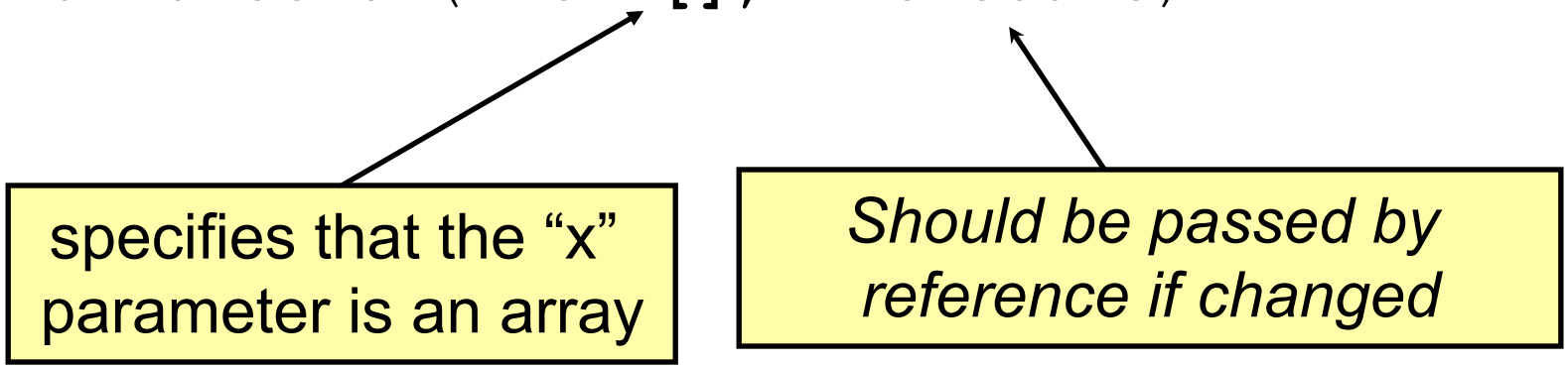
there is no need to fill up the entire array…

```
int myArray[4];
int count = 0;

myArray[count] = 14;
count++;

myArray[count++] = 9;

myArray[count++] = 5;

myArray[count++] = 13;

myArray[count] = 17;
```

Functions and Arrays

- We can write functions to accept and array as input (as a parameter)

```
void function(int x[], int count)
```



Exercise

- Write a program that reads in up to 15 numbers from user:
  - Print the list out
  - Let the user ask if a particular number is present in the array

Sorted Lists

- What if we want to sort the number the user has entered?
  - An easy solution is to insert each number the user enters ***in order***, rather than always at the end of the array…
  - This way, the array will always be in sorted order.

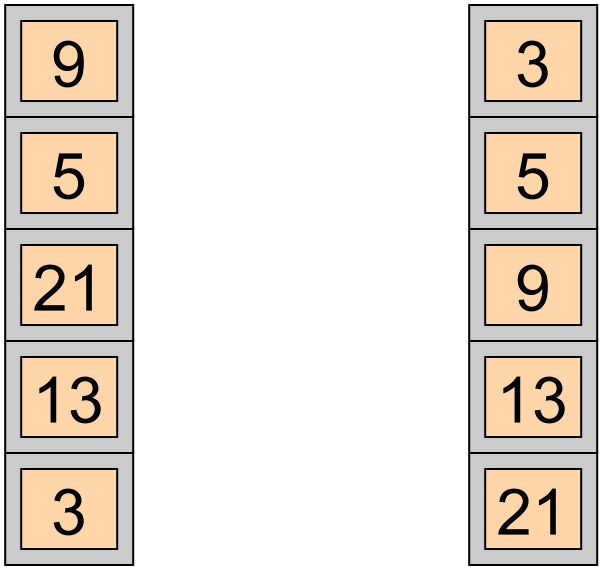
- 2 Cases:
  - Insert at end (easy)
  - Insert anywhere else…
  - Insert at beginning or middle means we need to "push" all the elements over to the right

Sorting After Reading

- We could also read all the numbers in first, and then sort them all once.
  - There are many, many sorting algorithms… lets look at one – *selection sort*.

Selection Sort

- Idea: Create a temporary array, move items from original to temporary in order:



Do we really need both arrays?

Selection Sort

- Algorithm: (selects in reverse order)
  - Starting at the end of the array
    - Scan all elements *before it (inclusive)*, remember *where* the largest is.
    - Swap the contents of the current element with the largest before it.
    - Move to the left one element.
  - Stop at element 1 (not 0)

A closer look at Find

- What if all the numbers in the array were in order?

- Binary Search
  - Similar to how you would look up a word in dictionary
  - Split the list in half, throw away the part that cannot contain what you are looking for.
    - **Repeat**

Time Complexity

Simulation Results

# of Iterations (Average)		
N	**Linear Search**	**Binary Search**
10	5.5	2.9
100	50.5	5.8
1000	500.5	9.0
10000	5000.5	12.4

- Clearly, Binary Search is far better…

Exercise

- Implement a binary search:
  - Read in each number from user
  - Insert *in order*.
  - Modify our existing "find" function to implement a binary search rather than linear search.

Try this on your own… it's a good study question …. Yes, there are many solutions online…