Computer Science 1: Starting Computing
CSCI 1300

University of Colorado
Boulder

Computer Science 1: Starting Computing CSCI 1300

Dr. Ioana Fleming
Spring 2019
Lecture 14

Reminders

Submissions:

- Homework 5: due 2/24 at 6pm
- Homework 6: due Monday 3/4 at 6pm

Readings:

- Ch. 6 – arrays
- Ch. 8 – streams

Practicum I: Wed 2/20 at 5:30 pm

Should have received Moodle links?

Practice, practice, practice!

- Review all previous Moodle programming questions from previous recitation and homework assignments
- Review examples we did in class
- Time is short; prepare accordingly.

Tips for Timed Exam

- Read the Questions
  - read them not once, but **TWICE** before starting the code
  - follow all the instructions explicitly (especially for names and order of parameters)
- Create or Modify a Code
  - know your C++ syntax
- Create and Use an IF, IF ELSE
  - know your C++ syntax
  - know how to create a condition
- Create and Use a WHILE or FOR loop
  - know your C++ syntax
- Passing parameters
  - know your C++ syntax

Missed the Grading Interview?

- You are allowed to reschedule one interview in a semester without any penalty.
- Second time 25% penalty
- Third time 50% penalty
- And so on …

© traveler1116/iStockphoto.

Chapter Six: Arrays and Vectors

University of Colorado Boulder
Slides by Evan Gallagher

Chapter Goals

- To become familiar with using arrays ~~and vectors~~ to collect values
- To learn about common algorithms for processing arrays ~~and vectors~~
- To write functions that receive and return arrays ~~and vectors~~
- To be able to use two-dimensional arrays

Vectors – later in the semester

Topic 1

1. Arrays
2. Common array algorithms
3. Arrays / functions
4. Problem solving: adapting algorithms
5. Problem solving: discovering algorithms
6. 2D arrays
7. Vectors
8. Chapter Summary

Using Arrays

Think of a sequence of data:

32 54 67.5 29 35 80 115 44.5 100 65

(all of the same type, of course)
(storable as **`double`**s)

Using Arrays

32 54 67.5 29 35 80 115 44.5 100 65

Which is the largest in this set?

(You must look at every single value to decide.)

Using Arrays

32 54 67.5 29 35 80 115 44.5 100 65

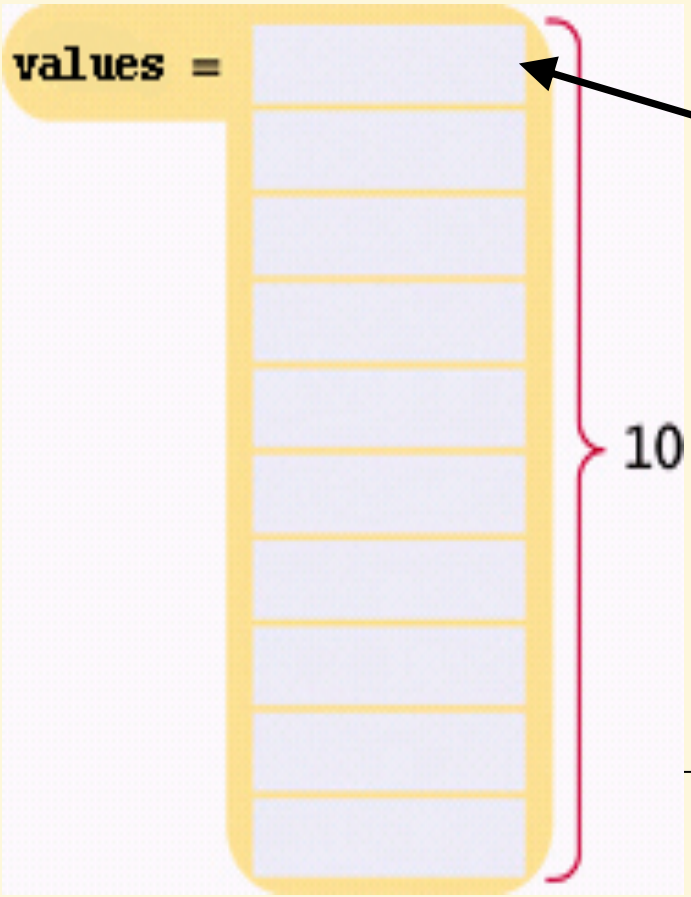
So you would create a variable for each, of course!

```
int n1, n2, n3, n4, n5, n6, n7, n8, n9, n10;
```

Then what ???

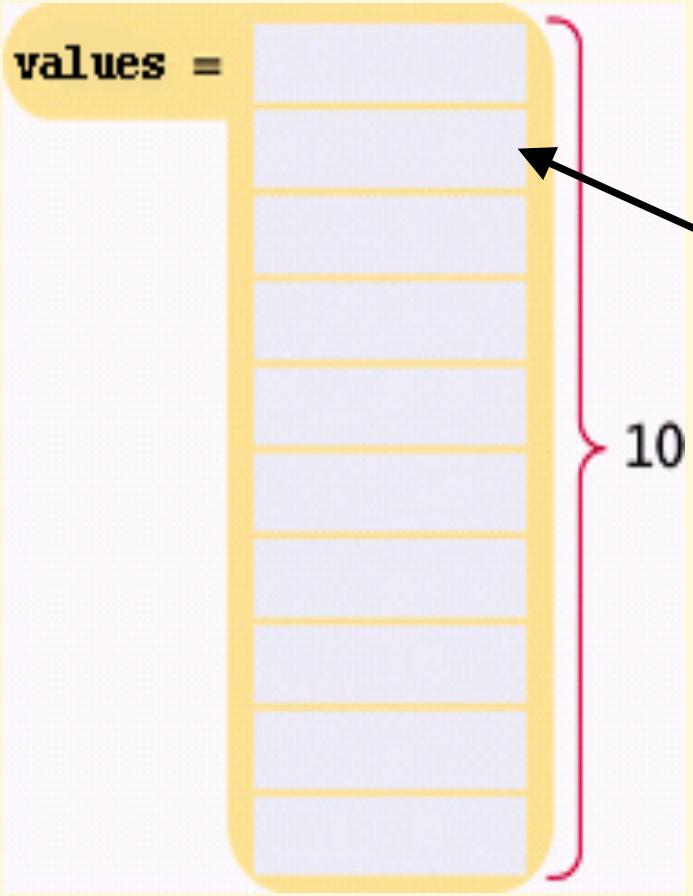
Using Arrays

You can easily visit each element in an array,
checking and updating a variable holding the current maximum.



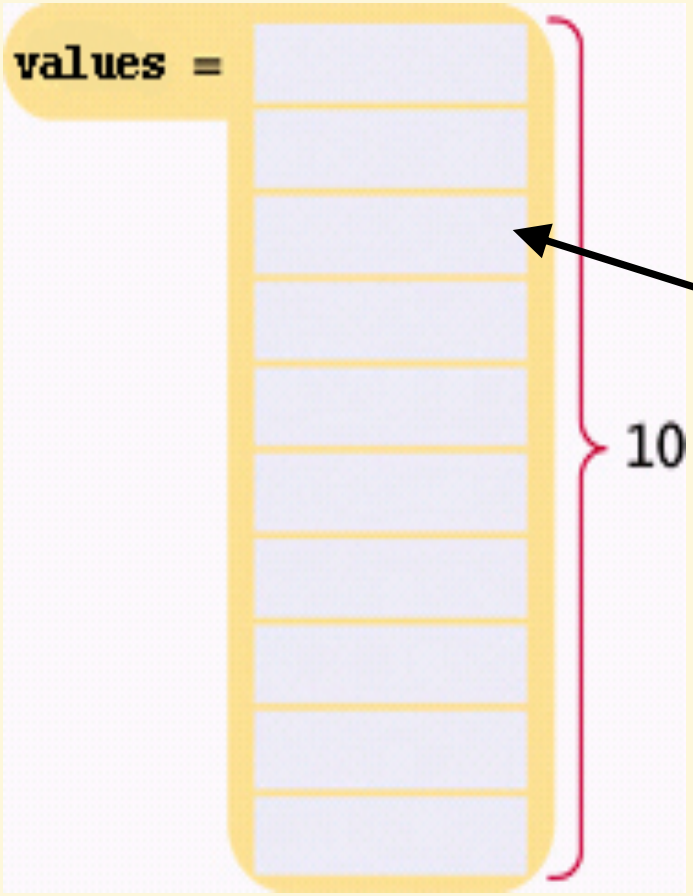
Hm. Is this the max, so far?

Using Arrays

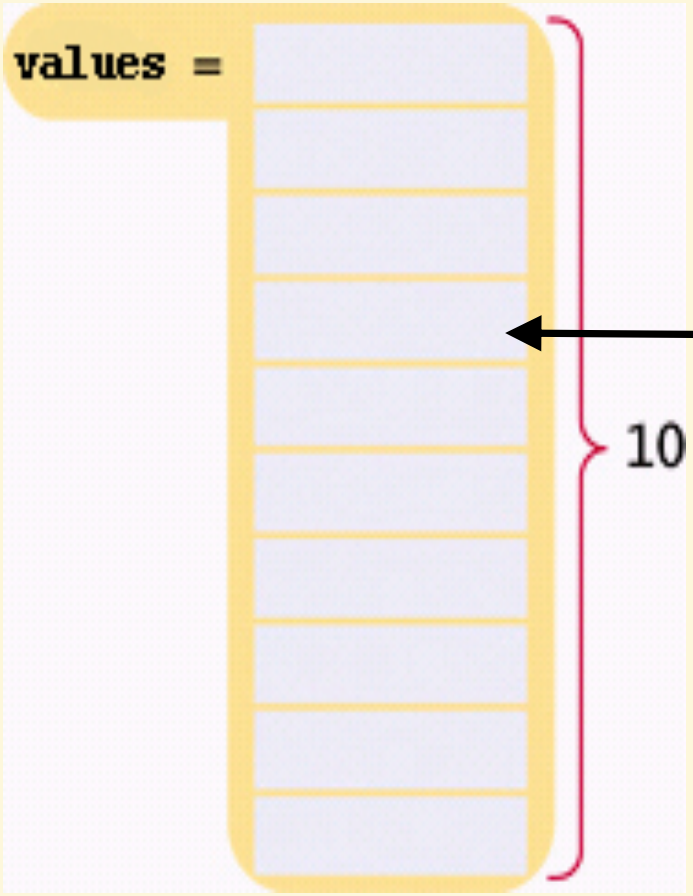


Maybe this one?

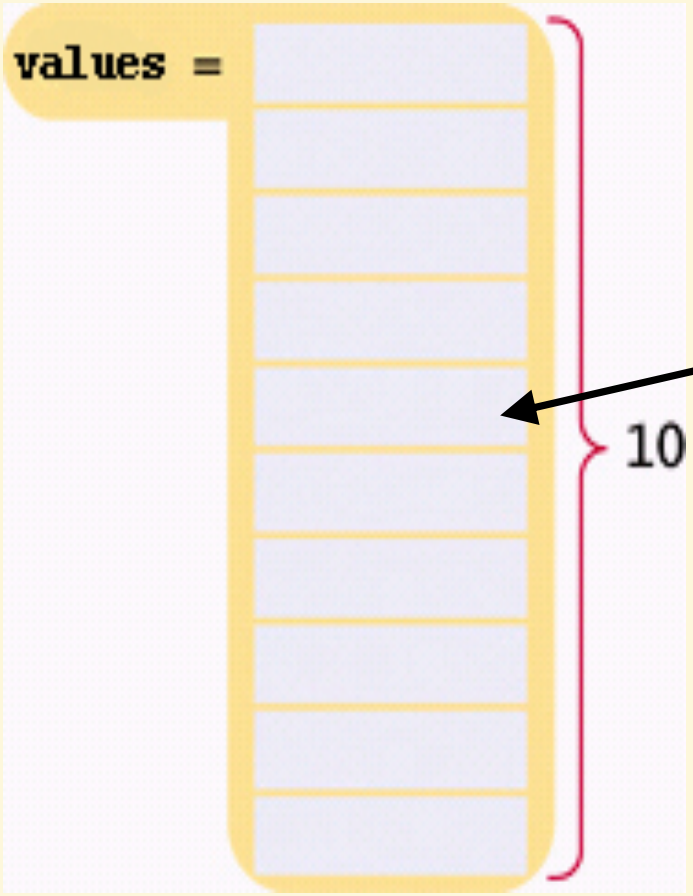
Using Arrays



Using Arrays

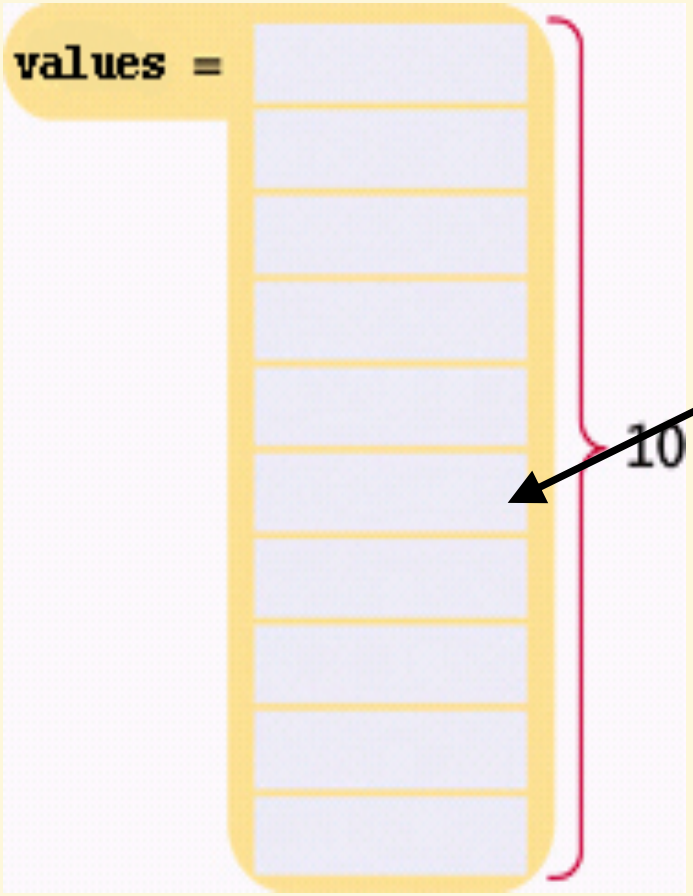


Using Arrays



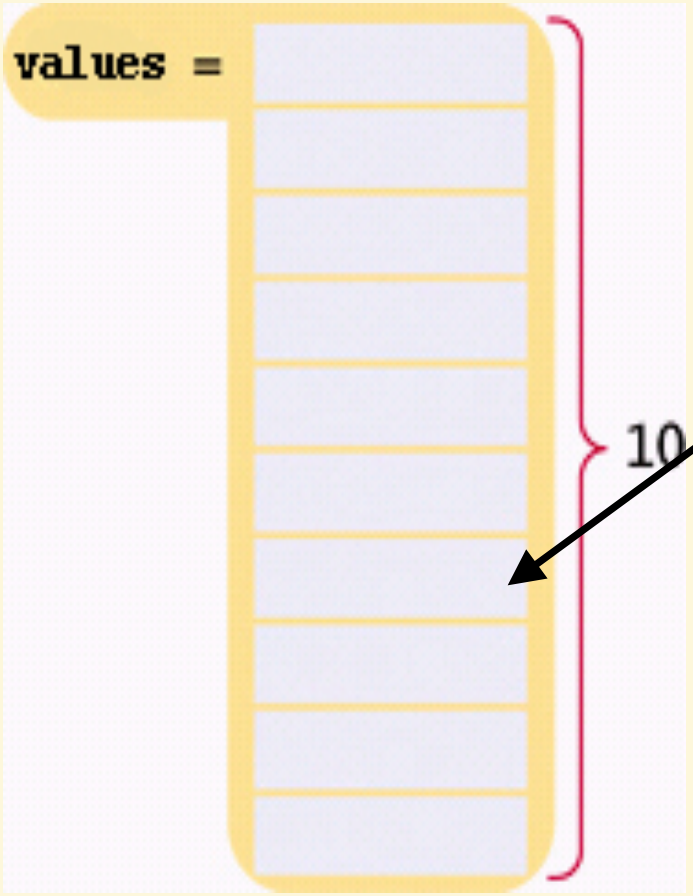
How about here?

Using Arrays



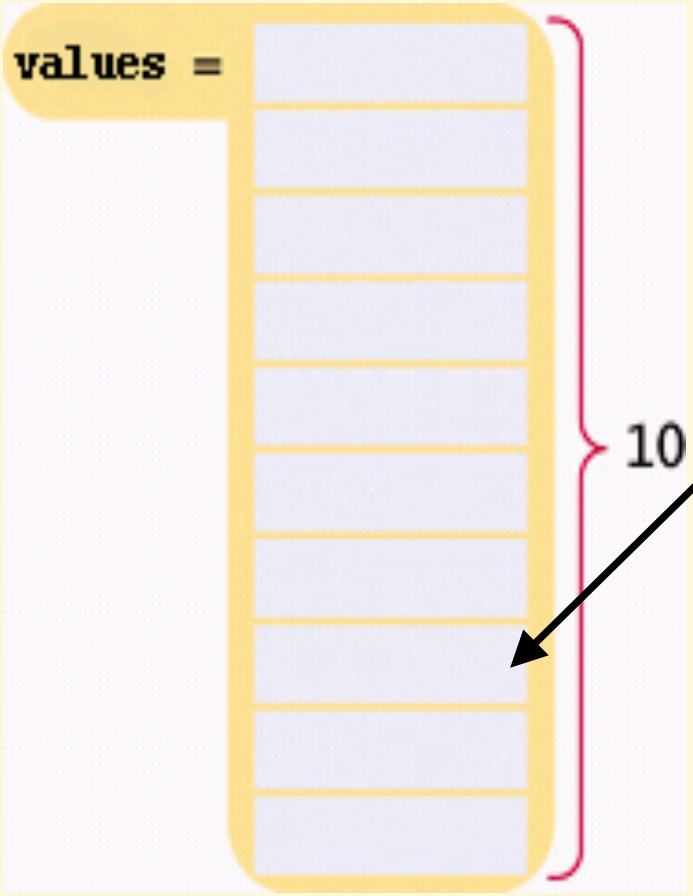
Gotta check here too!

Using Arrays



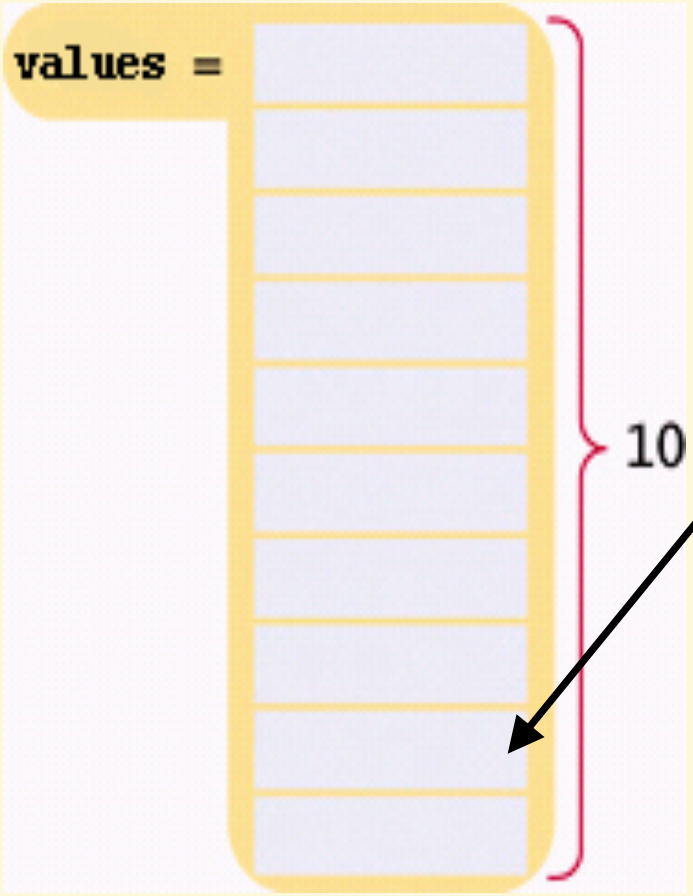
Again, maybe this one?

Using Arrays



Or this one?

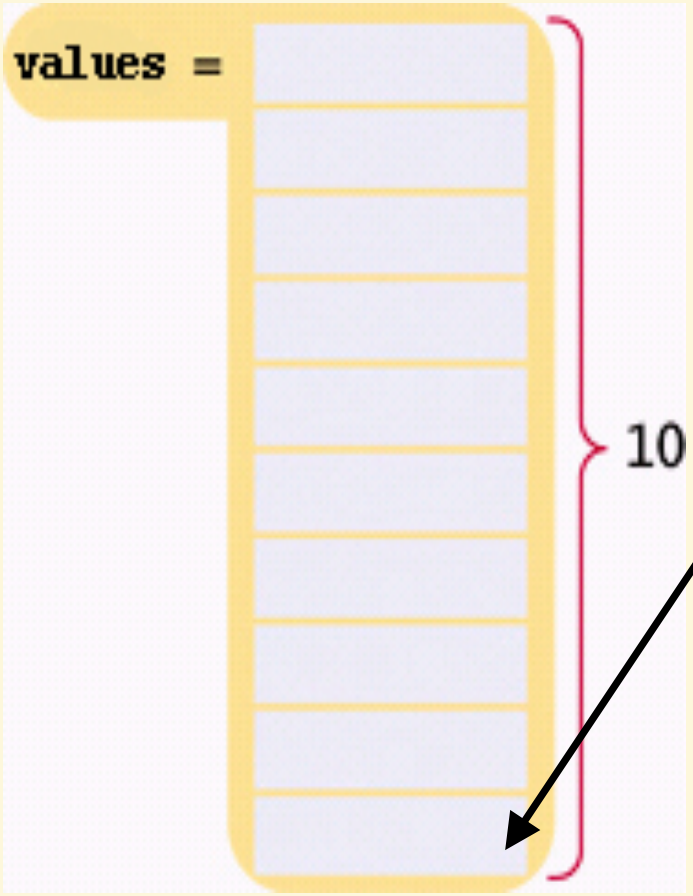
Using Arrays



Or this one?

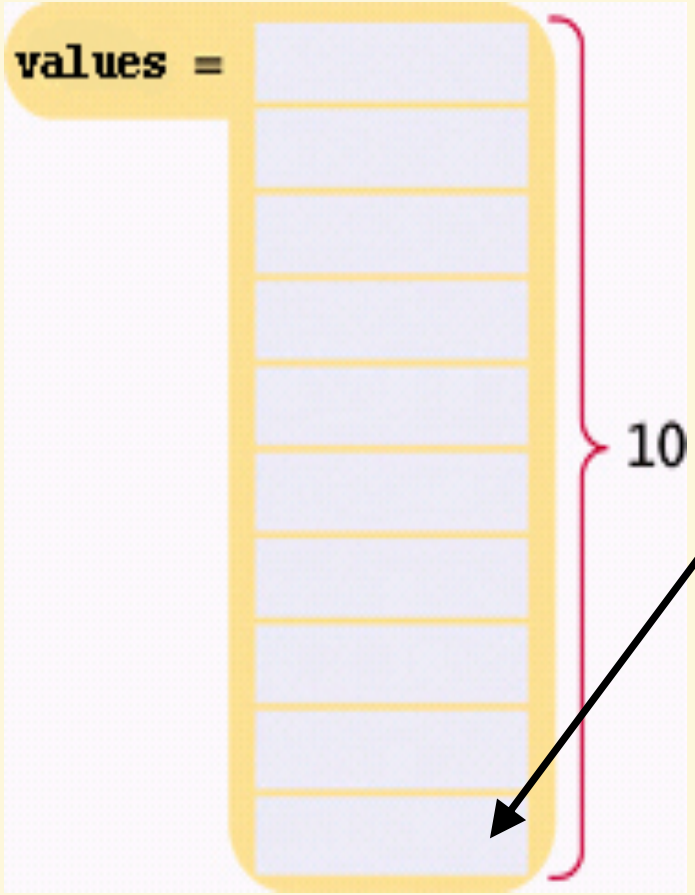
Will this never end?

Using Arrays



Or the last one? Finally!

Using Arrays and Vectors



Or the last one? ***Finally!***

Using Arrays and Vectors

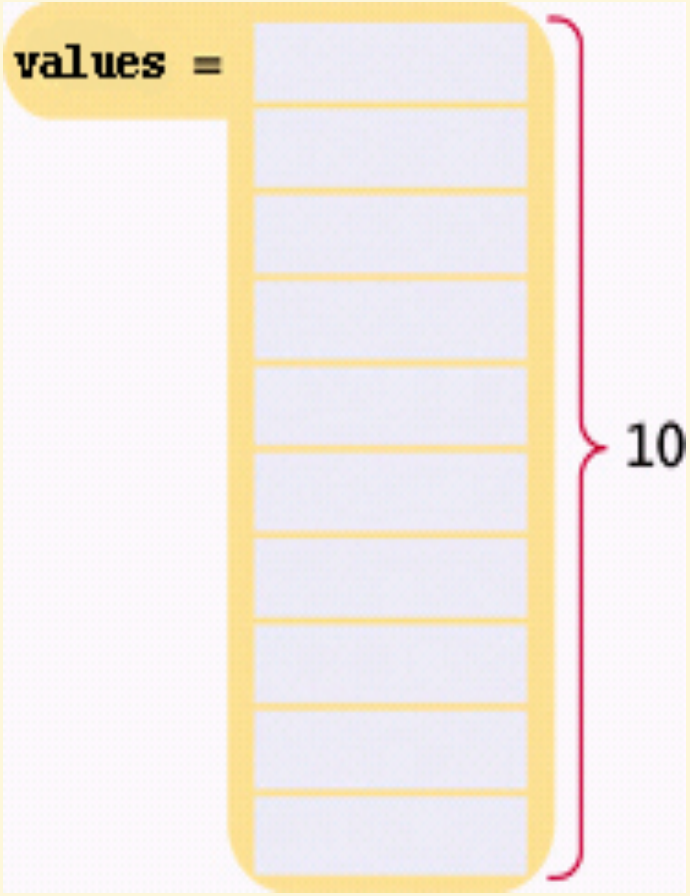
That would have been impossible with ten separate variables!

```
int n1, n2, n3, n4, n5, n6, n7, n8, n9, n10;
```

And what if there needed to be more double values in the set?

ARGH!

Defining Arrays



An "array of double"

Ten elements of **`double`** type

can be stored under one name as an array.

```
double values[10];
```

type of each element

quantity of elements – the "size" of the array, must be a constant

Introduction to Arrays

- Array definition:
  - A collection of data of same type
- First "aggregate" data type
  - Means "grouping"
  - int, float, double, char are simple data types
- Used for lists of like items
  - Test scores, temperatures, names, etc.
  - Avoids declaring multiple simple variables
  - Can manipulate "list" as one entity

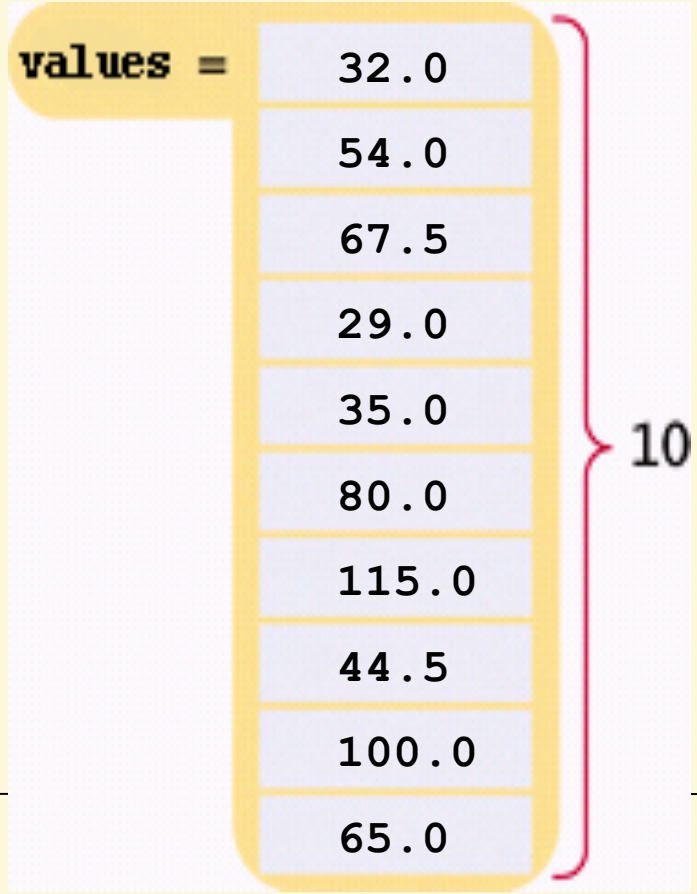
Declaring Arrays

- Declare the array → allocates memory
  `int score[5];`
  - Declares array of 5 integers named "score"
  - Similar to declaring five variables:
    `int score[0], score[1], score[2], score[3], score[4]`
- Individual parts can be called many things:
  - Indexed or subscripted variables
  - "Elements" of the array
  - Value in brackets is called index or subscript
    - Numbered from 0 to (size – 1)

Defining Arrays with Initialization

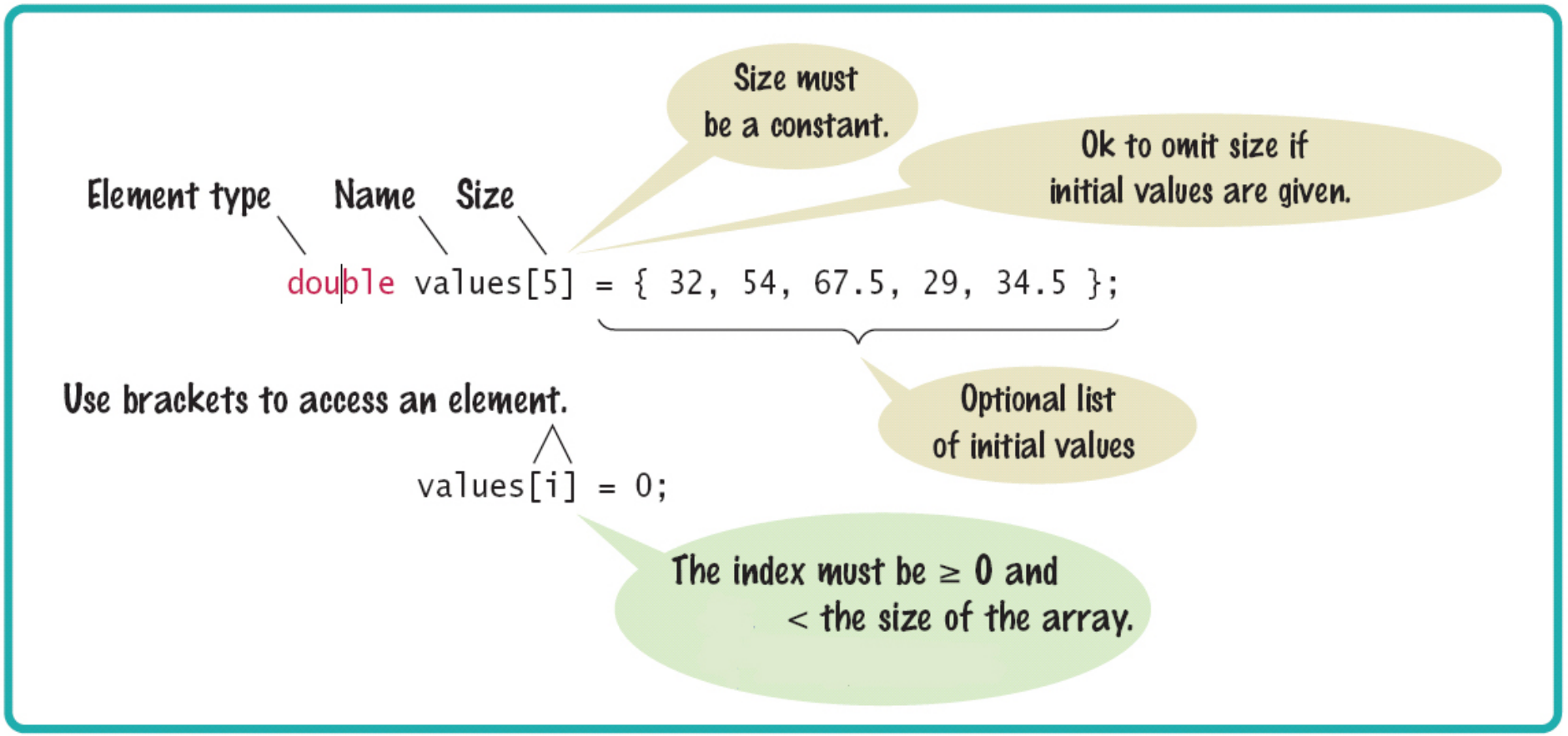
When you define an array, you can specify the initial values:

```
double values[] = { 32, 54, 67.5, 29, 35, 80, 115, 44.5, 100, 65 };
```



Array Syntax

Defining an Array



Accessing Arrays

- Access using index/subscript

```
cout << score[3];
```

- Note two uses of brackets:
  - In declaration, specifies SIZE of array
  - Anywhere else, specifies a subscript
- Size, subscript need not be literal

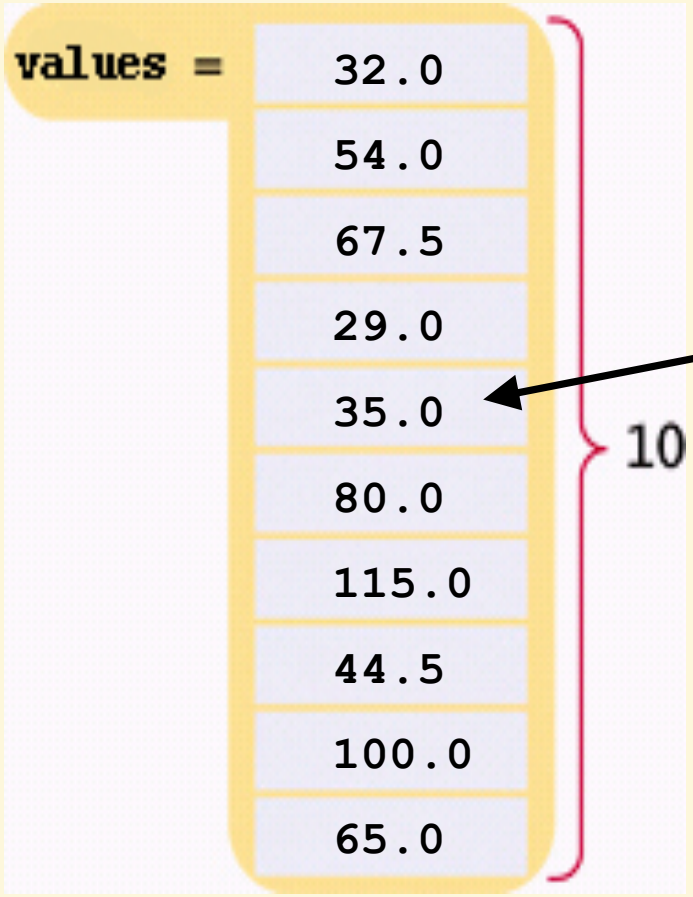
```
int score[MAX_SCORES];
score[n+1] = 99;
```

  - If n is 2, identical to: score[3]

Accessing an Array Element

To access the element at index 4 using this notation: **`values[4]`**

4 is the *index*.

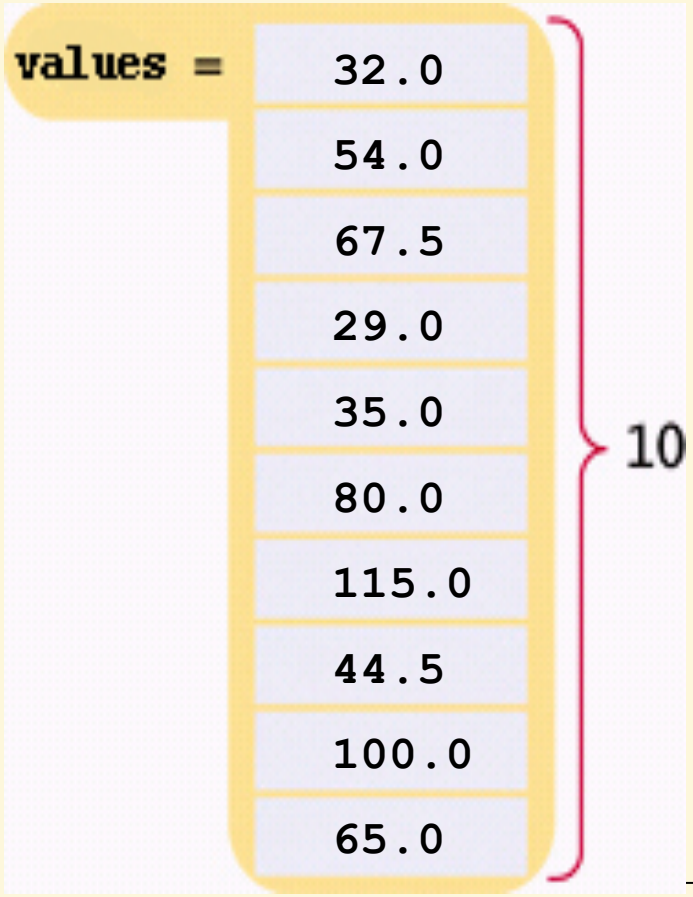



```
double values[10];
...
cout << values[4] << endl;
```

The output will be `35.0`.

Accessing an Array Element

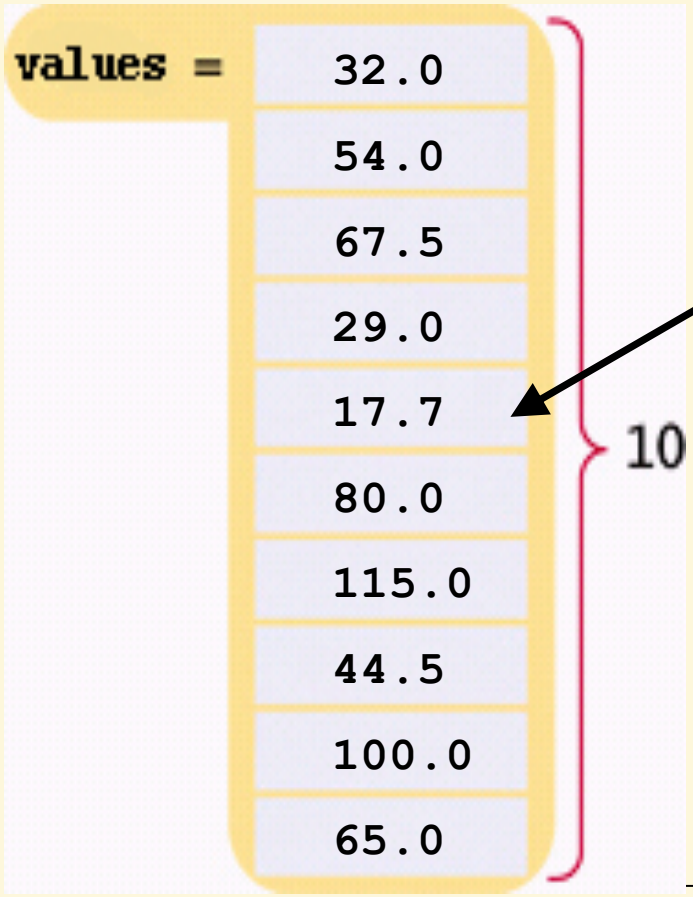
The same notation can be used to change the element.




```
values[4] = 17.7;
```

Accessing an Array Element

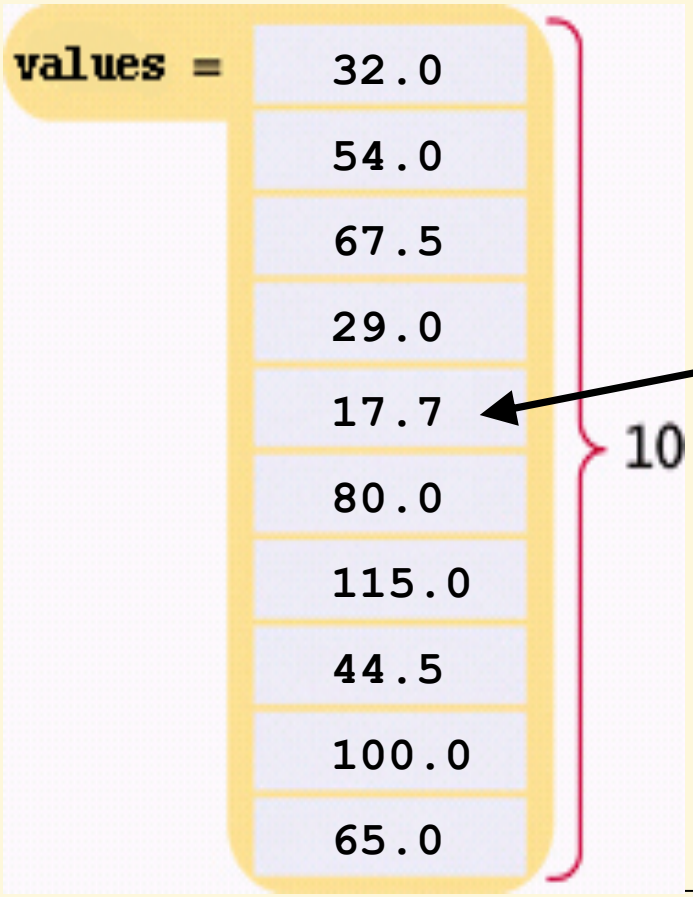
The same notation can be used to change the element.




```
values[4] = 17.7;
```

Accessing an Array Element

The same notation can be used to change the element.




```
values[4] = 17.7;
cout << values[4] << endl;
```

The output will be `17.7`.

Accessing an Array Element

That is, the legal elements for the **`values`** array are:

`values[0]`, the ***first*** element
`values[1]`, the second element
`values[2]`, the third element
`values[3]`, the fourth element
`values[4]`, the fifth element
`...`
`values[9]`, the tenth ***and last legal*** element
recall: **`double values[10];`**

The index must be **`>= 0`** and **`<= 9`**.
0, 1, 2, 3, 4, 5, 6, 7, 8, 9 is ... 10 numbers.

Array Usage

- Powerful storage mechanism
- Can issue commands like:
  - "Do this to i^{th} indexed variable", where i is computed by program
  - "Display all elements of array score"
  - "Fill elements of array score from user input"
  - "Find highest value in array score"
  - "Find lowest value in array score"
- Disadvantages: size MUST BE KNOWN at declaration

Demo

- Cloud9 – *largest.cpp*