Arrays

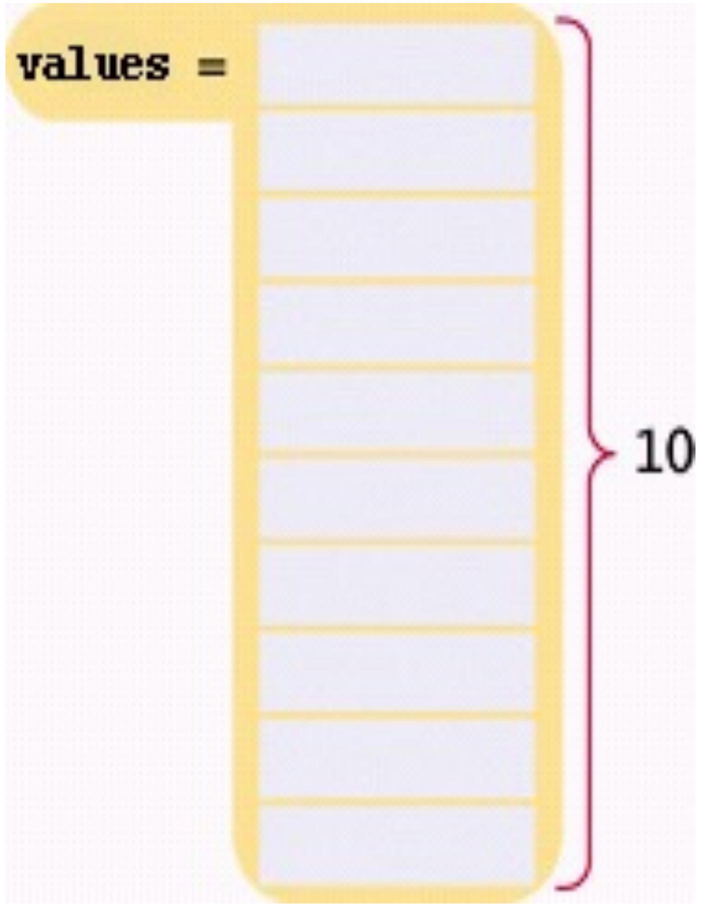
Using Arrays

32   54   67.5   29   35   80   115   44.5   100   65

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

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

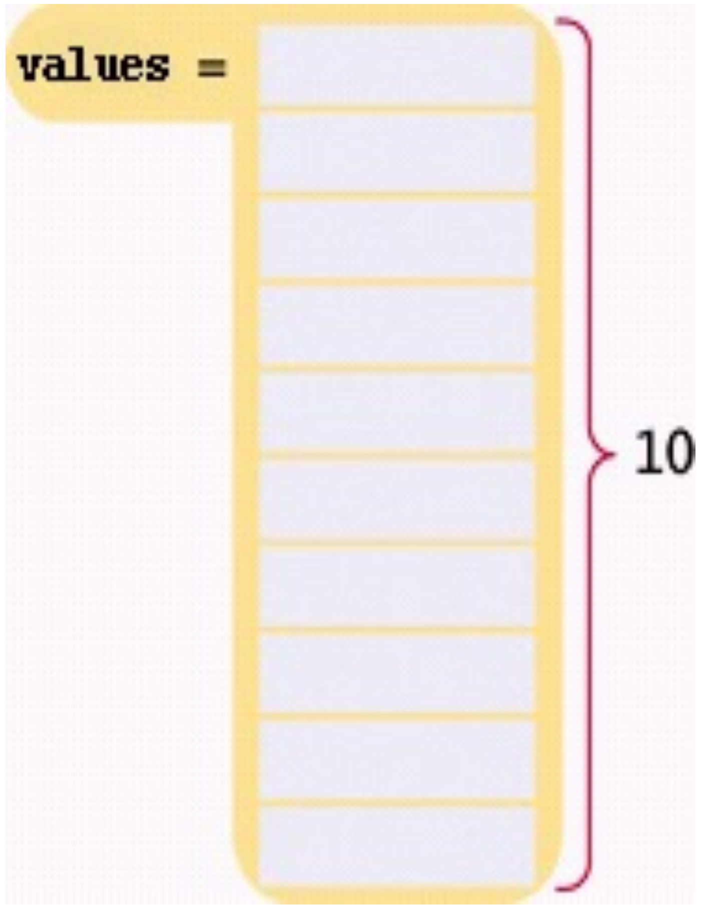
Using Arrays



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

Defining Arrays

An "array of double"



Ten elements of **double** type can be stored under one name as an array.

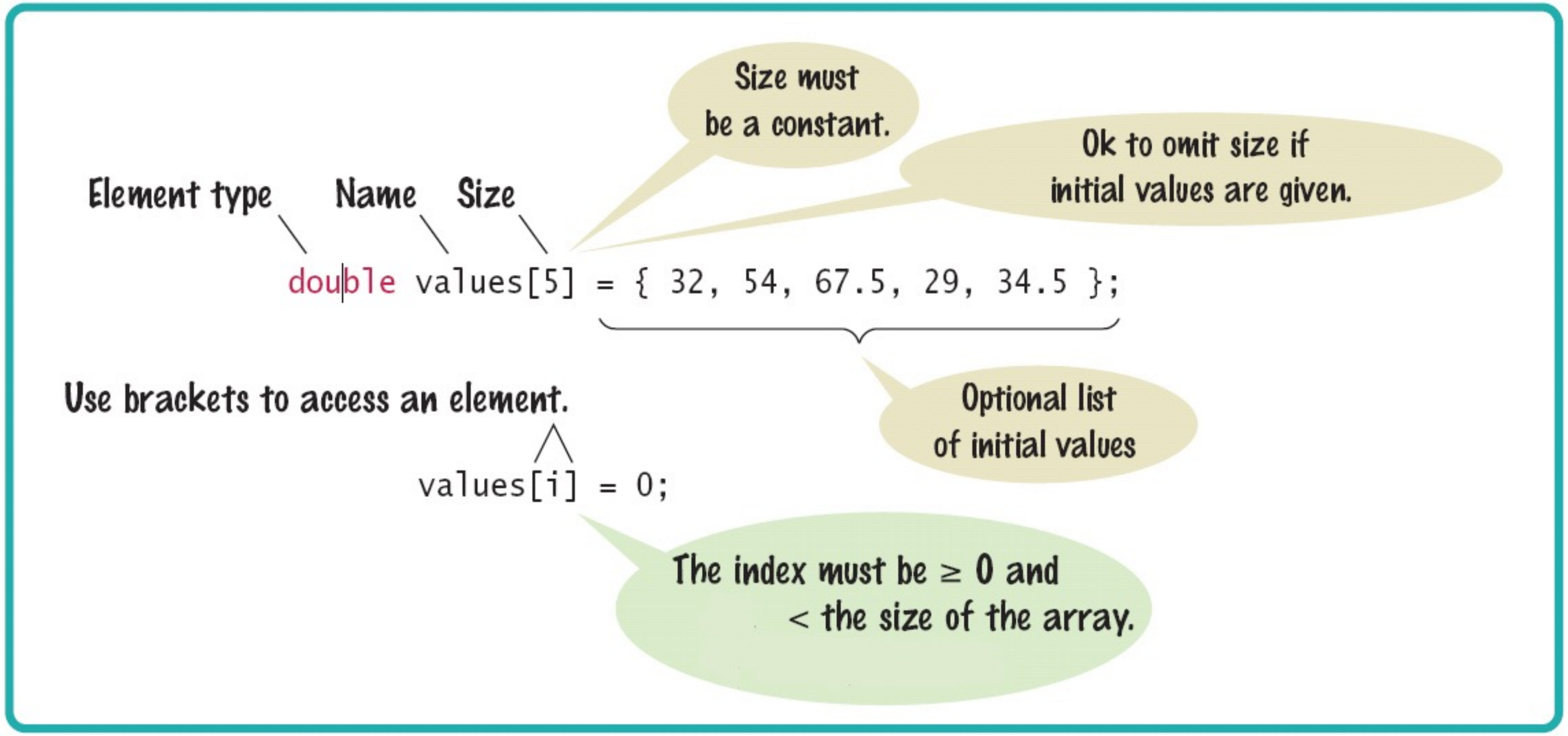
```
double values[10];
```

type of each element

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

Array Syntax

Defining an Array


Size must be a constant.
Ok to omit size if initial values are given.
Element type
Name
Size
Optional list of initial values
Use brackets to access an element.
The index must be ≥ 0 and < the size of the array.

```
double values[5] = { 32, 54, 67.5, 29, 34.5 };

values[i] = 0;
```

Introduction to Arrays

Definition: An array is a collection of data of the same type, referenced as different elements of the same name.

- 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

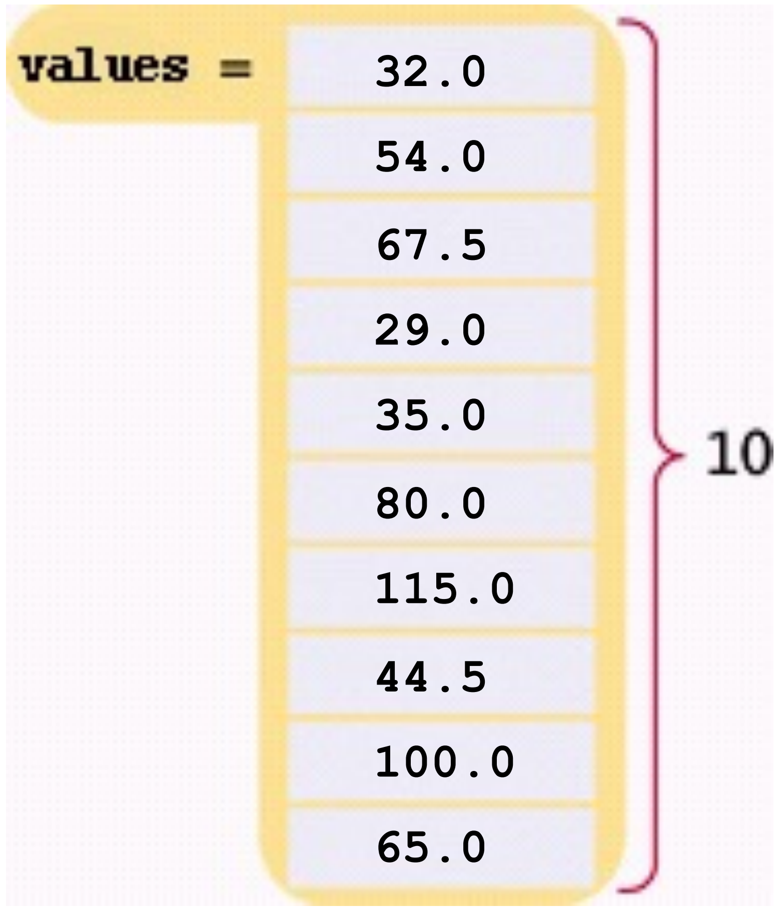
```
int score[5];
```

- 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 };
```



Accessing Arrays

- Access using index/subscript

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

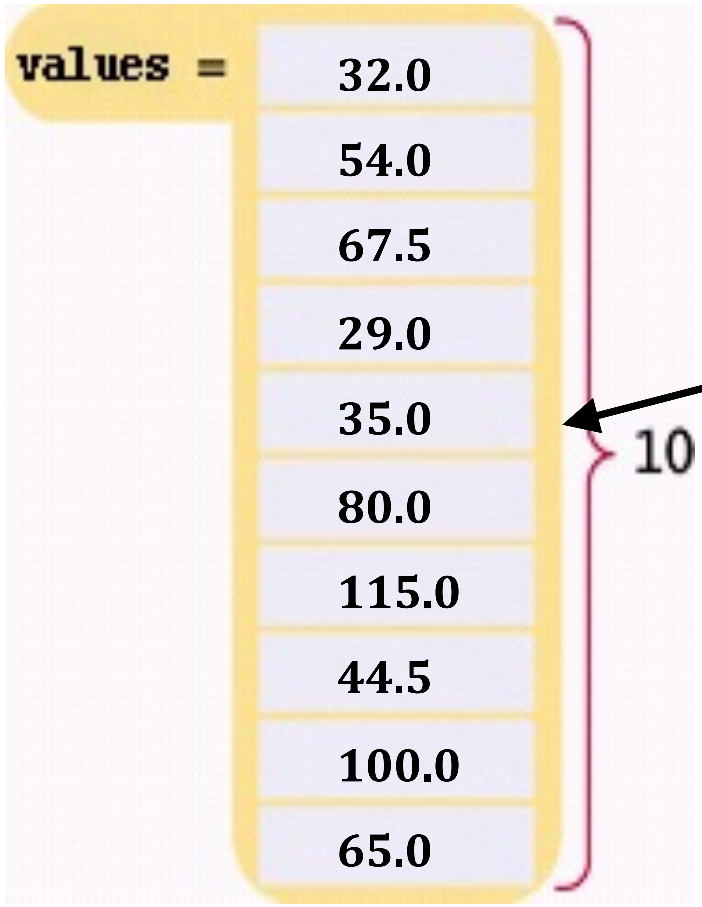
- Note two uses of brackets:
  - In declaration, specifies SIZE of array
  - Anywhere else, specifies a subscript

- 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

The same notation can be used to change the element.



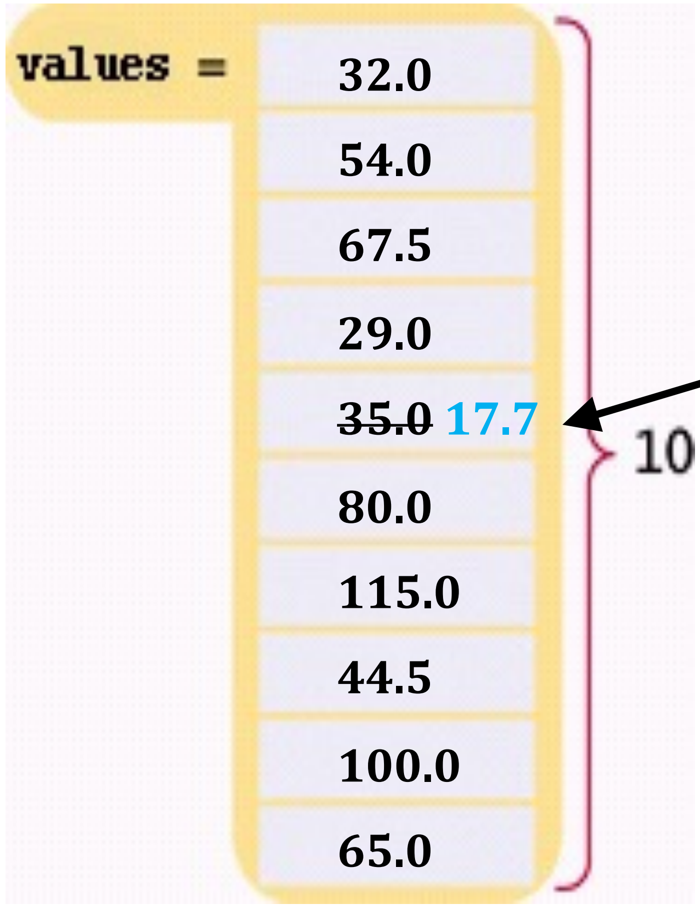
```
double values[10];
...
cout << values[4] << endl;
```

The output will be **35.0**.

Accessing an Array Element

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

4 is the *index*.



```
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 or < 10**

0, 1, 2, 3, 4, 5, 6, 7, 8, 9 is … 10 numbers.