Arrays

Due this week

- **Project 1**
  - Write solutions in VSCode and paste in Autograder, **Project1-CodeRunner**
  - **Honor Code MCQ**
  - **Project1-CodeRunnerEC**
  - Zip your .cpp files and submit on canvas **Project 1**. Check the due date! **No late submissions!!**
  - **Mandatory Grading Interview** next week!
  - Follow instructions from write-up
- No Quiz this week

Today

- What are arrays?
- Initializing arrays
- Common array algorithms

Arrays

Using Arrays

Think of a list of values - sales per month in 2021

```
32 54 67.5 29 35 80.3 115 98 100 65 210.5 140
```

(all of the same type, of course)

(storable as **`double`**s)

Using Arrays

```
32 54 67.5 29 35 80.3 115 98 100 65 210.5 140
```

Which is the largest value in this set?

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

Using Arrays

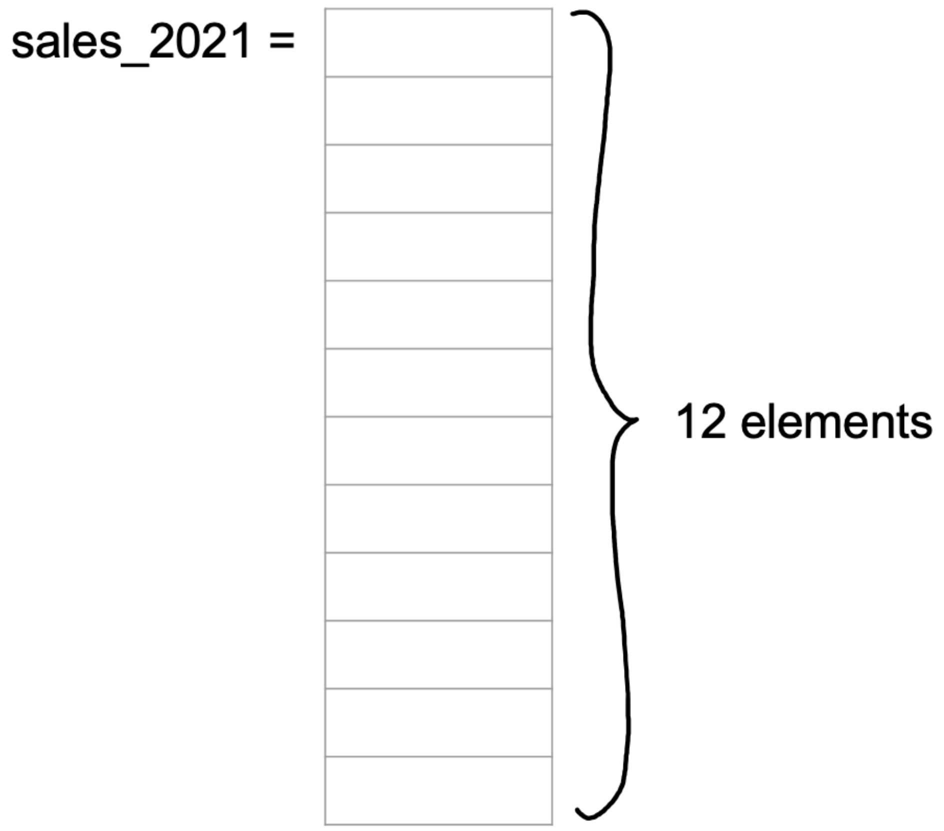
32 54 67.5 29 35 80.3 115 98 100 65 210.5 140

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

```
double jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec;
```

Then what ???

Defining Arrays



An "array of double"

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

```
double sales_2021[12];
```

type of each element

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

Brief C++ by Cay Horstmann
Copyright © 2017 by John Wiley & Sons. All rights reserved

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

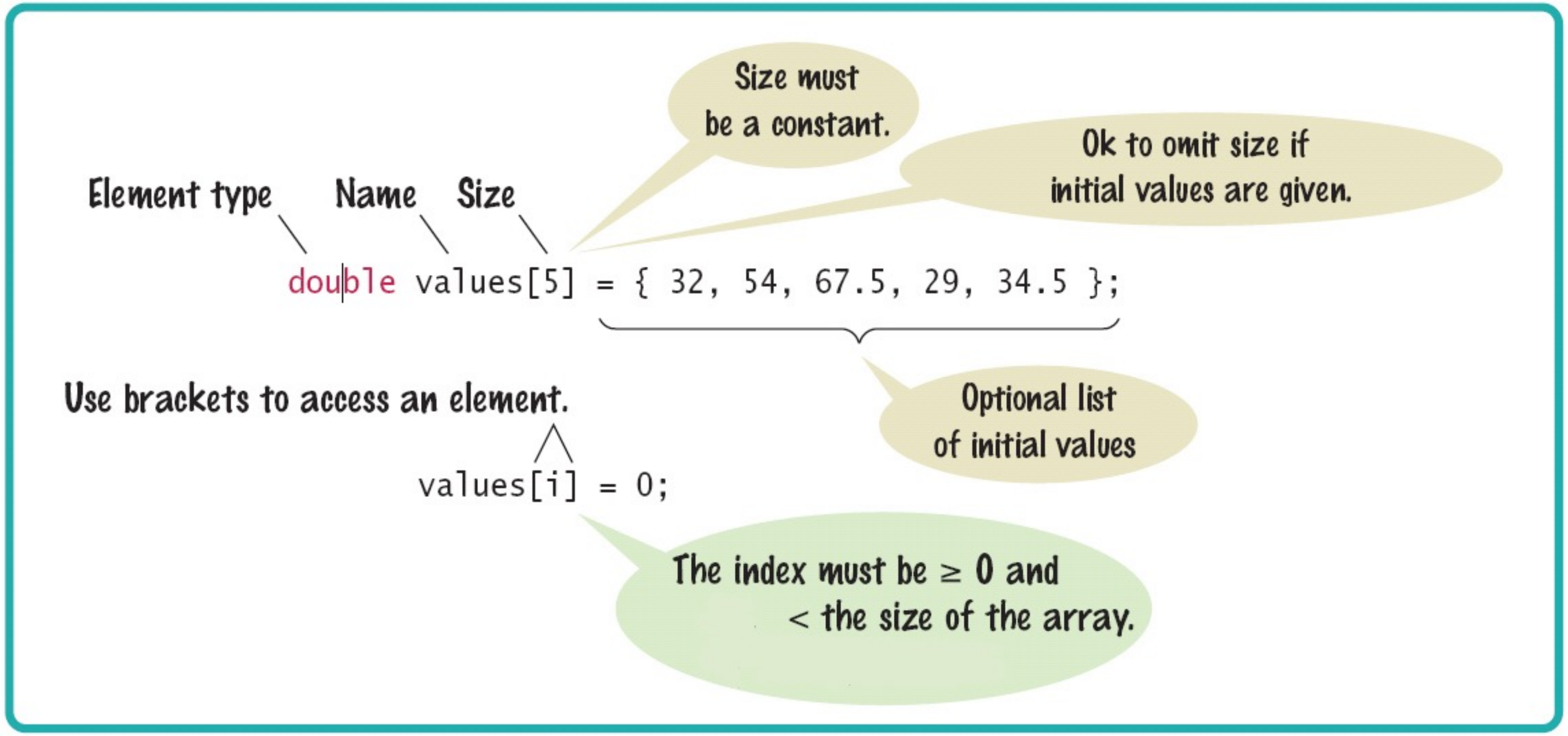
Declare the array → allocates memory

```
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)

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;
```

Brief C++ by Cay Horstmann
Copyright © 2017 by John Wiley & Sons. All rights reserved

Accessing Arrays

- Access using index/subscript

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

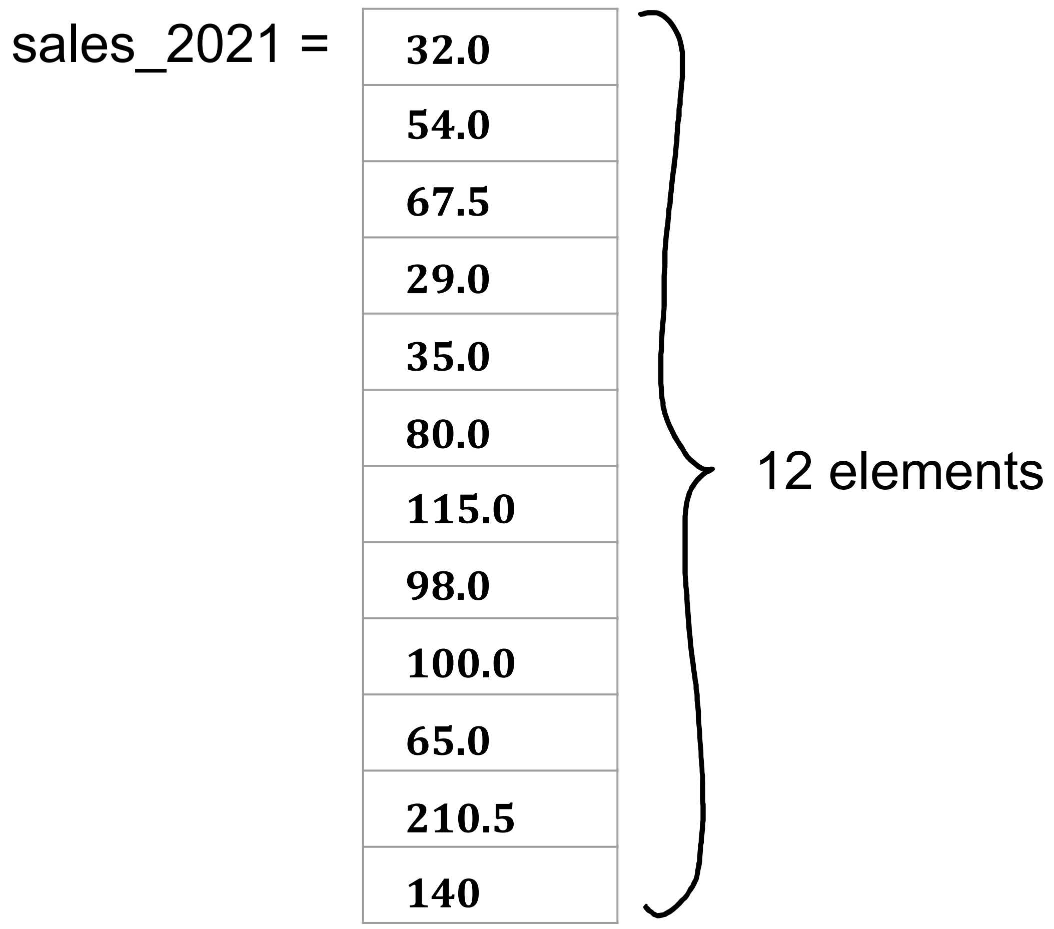
- Note two uses of brackets:
  - In declaration, specifies SIZE of array
  - Anywhere else, specifies a subscript

- Size, subscript need not be literal

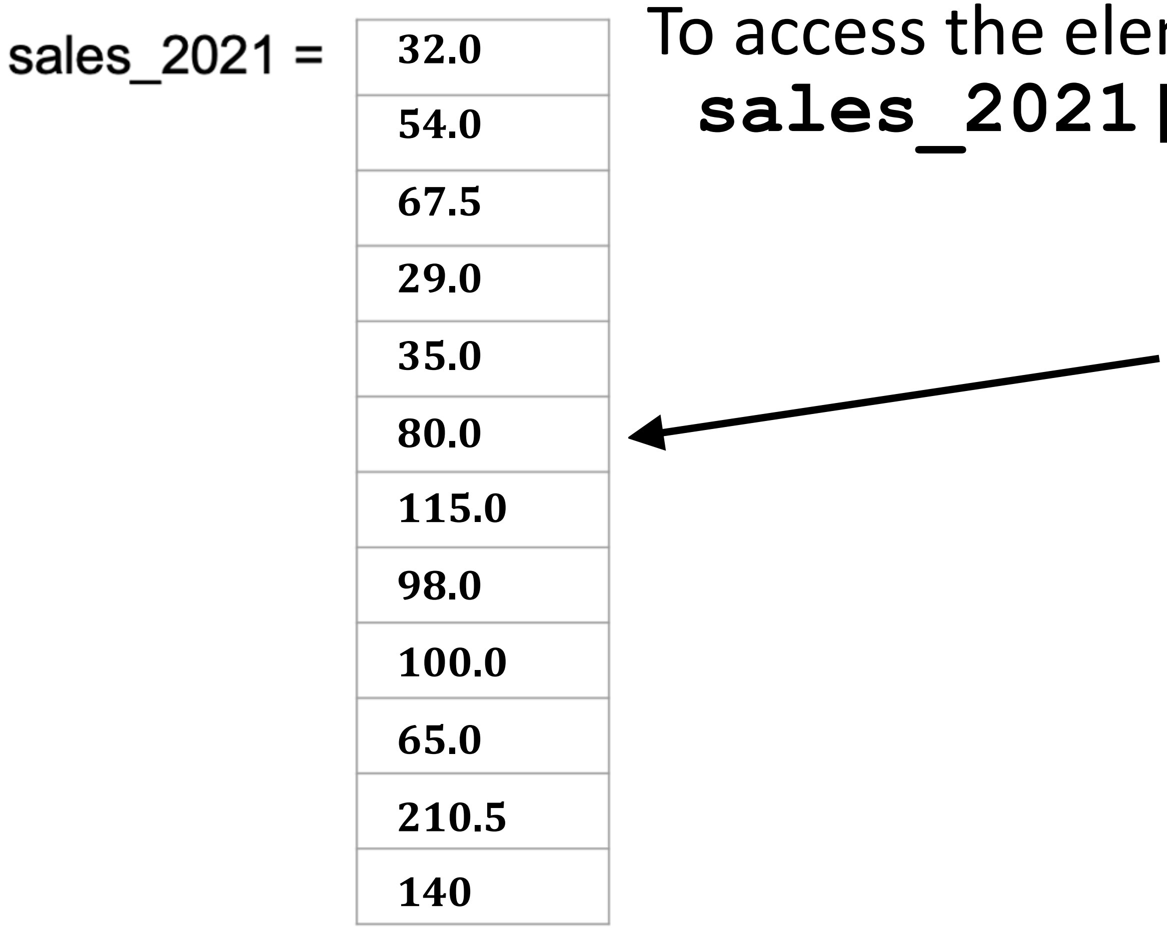
```
int score[MAX_SCORES];
score[n+1] = 99;   -->
```

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

Accessing Arrays



Accessing an Array Element



To access the element at index 5 using this notation:

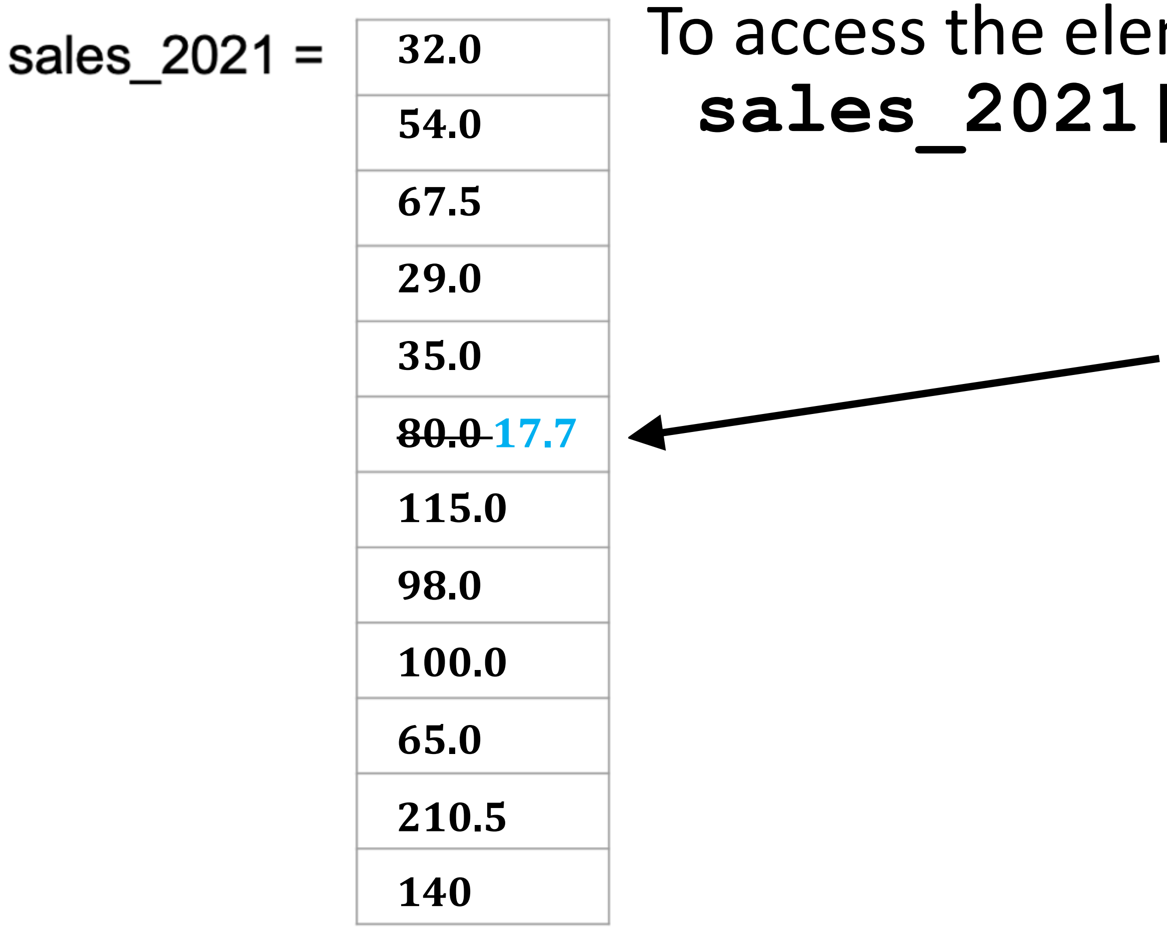
```
sales_2021[5]
```

5 is the *index*.

```
double sales_2021[12];
...
cout << sales_2021[5] << endl;
```

The output will be **80**

Accessing an Array Element



To access the element at index 5 using this notation:

```
sales_2021[5]
```

5 is the *index*.

```
sales_2021[5] = 17.7;
...
cout << sales_2021[5] << endl;
```

The output will be **17.7**

Accessing an Array Element

- That is, the legal elements for the **sales_2021** array are:
- sales_2021[0] , the first element
- sales_2021[1] , the second element
- sales_2021[2] , the third element
- sales_2021[3] , the fourth element
- sales_2021[4] , the fifth element
- ...
- sales_2021[11] , the twelfth and last legal element
- recall: double sales_2021[12];
- The index must be >= 0 and <= 11.
- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 is ... 12 numbers.

Array Usage

- Powerful storage mechanism
- Can issue commands like:
  - "Do this to ith 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

Common Array Algorithms

Common Algorithms – Filling

- This loop fills an array with zeros:

```
for (int i = 0; i < size; i++)
{
   values[i] = 0;
}
```

- To fill an array with squares (0, 1, 4, 9, 16, ...).

```
for (int i = 0; i < size; i++)
{
   squares[i] = i * i;
}
```

Brief C++ by Cay Horstmann
Copyright © 2017 by John Wiley & Sons. All rights reserved

Common Algorithms – Copying

- Consider these two arrays:

```
int squares[5] = { 0, 1, 4, 9, 16 };
int lucky_numbers[5];
```

- How can we copy the values from squares to lucky_numbers?
- Let’s try what seems right and easy…
  - `squares = lucky_numbers;`

    …and wrong!
  - *You cannot assign arrays!*
  - *The compiler will report a syntax error.*

Common Algorithms – Copying Requires a Loop

```
/* you must copy each element individually using a loop! */

int squares[5] = { 0, 1, 4, 9, 16 };
int lucky_numbers[5];

for (int i = 0; i < 5; i++)
{
   lucky_numbers[i] = squares[i];
}
```

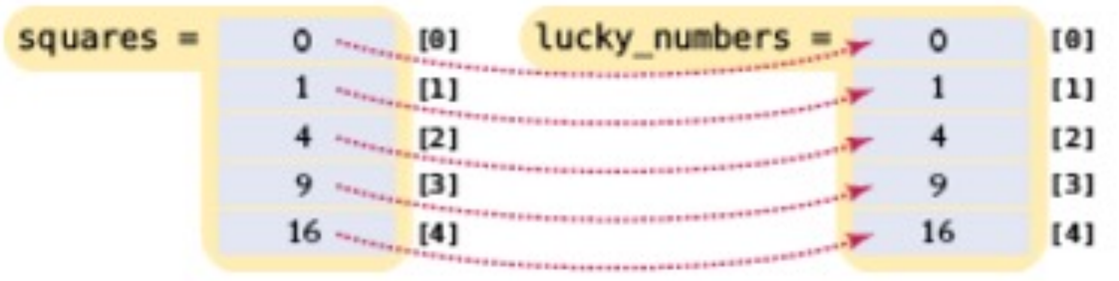


Figure 4 Copying Elements to Copy an Array

Brief C++ by Cay Horstmann
Copyright © 2017 by John Wiley & Sons. All rights reserved

Common Algorithms – Sum and Average Value

You have already seen the algorithm for computing the sum and average of a set of data. The algorithm is the same when the data is stored in an array.

```
double total = 0;
for (int i = 0; i < size; i++)
{
   total = total + values[i];
}
```

The average is just arithmetic:

```
double average = total / size;
```

Copyright © 2017 by John Wiley & Sons. All rights reserved

Common Algorithms – Maximum

To compute the largest value in a vector, keep a variable that stores the largest element that you have encountered, and update it when you find a larger one.

```
double largest = values[0];
for (int i = 1; i < size; i++)
{
    if (values[i] > largest)
    {
        largest = values[i];
    }
}
```

Copyright © 2017 by John Wiley & Sons. All rights reserved

Common Algorithms – Minimum

For the minimum, we just reverse the comparison.

```
double smallest = values[0];
for (int i = 1; i < size; i++)
{
     if (values[i] < smallest)
     {
          smallest = values[i];
     }
}
```

These algorithms require that the array contain at least one element.

Brief C++ by Cay Horstmann
Copyright © 2017 by John Wiley & Sons. All rights reserved

Common Algorithms – Linear Search

Find the position of a certain value, say 100, in an array:

```
int pos = 0;
bool found = false;
while (pos < size && !found)
{
   if (values[pos] == 100) // looking for 100
   {
      found = true;
   }
   else
   {
      pos++;
   }
}
```

Brief C++ by Cay Horstmann
Copyright © 2017 by John Wiley & Sons. All rights reserved

Common Algorithms – Swapping Elements

Suppose we need to swap the values at positions i and j in the array. Will this work?

```
values[i] = values[j];
values[j] = values[i];
```

- Look closely! In the first line you lost – forever! – the value at i, replacing it with the value at j.
- Then what?
- Put’ j’s value back in j in the second line?
- We end up with 2 copies of the [j] value, and have lost the [i]

Brief C++ by Cay Horstmann
Copyright © 2017 by John Wiley & Sons. All rights reserved

Code for Swapping Array Elements

```
//save the first element in
//  a temporary variable
// before overwriting the 1st

double temp = values[i];
values[i] = values[j];
values[j] = temp;
```

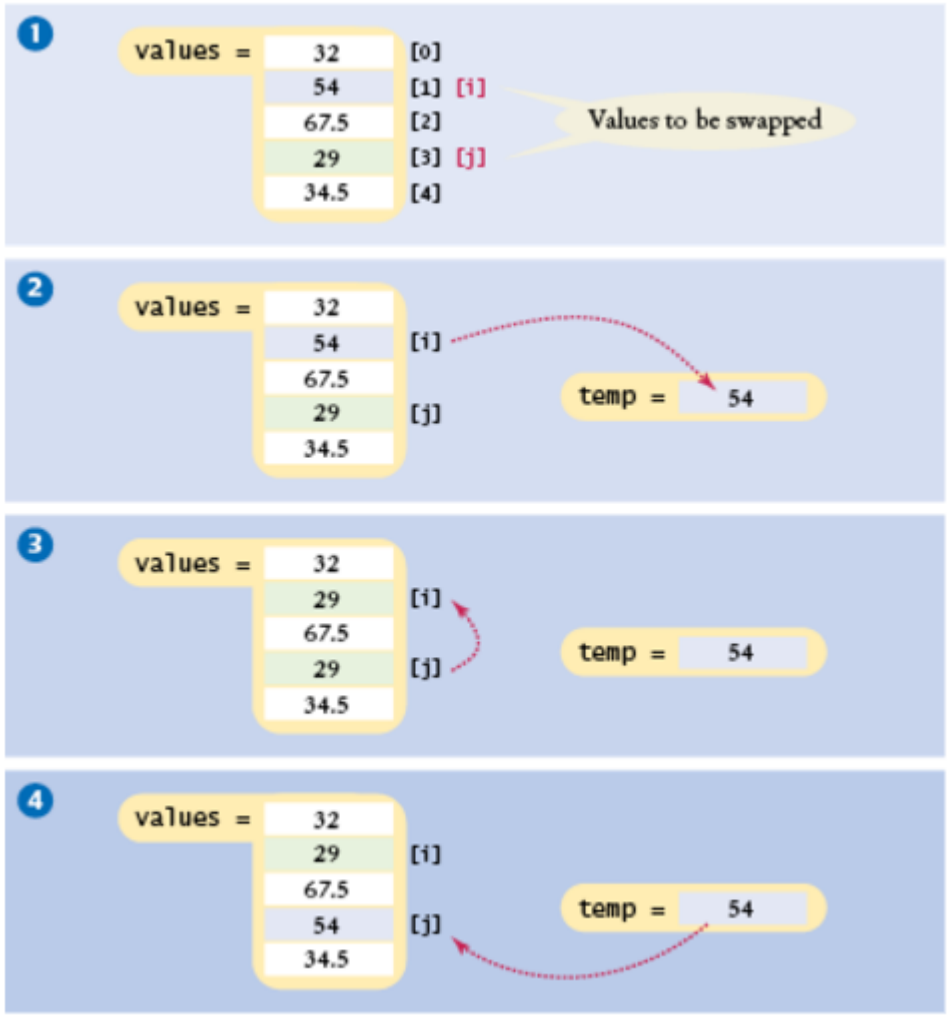

1
values = 32 54 67.5 29 34.5
[0] [1] [i] [2] [3] [j] [4]
Values to be swapped
2
values = 32 54 67.5 29 34.5
[i] [j]
temp = 54
3
values = 32 29 67.5 29 34.5
[i] [j]
temp = 54
4
values = 32 29 67.5 54 34.5
[i] [j]
temp = 54

Figure 9 Swapping Array Elements

Brief C++ by Cay Horstmann
Copyright © 2017 by John Wiley & Sons. All rights reserved

Common Algorithms – Reading Input

- If the know how many input values the user will supply, you can store them directly into the array:

```
double values[NUMBER_OF_INPUTS];
for (i = 0; i < NUMBER_OF_INPUTS; i++)
{
    cin >> values[i];
}
```

Common Algorithms – Reading Unknown # of Inputs

When there will be an arbitrary number of inputs, things get more complicated. But not hopeless. Add values to the end of the array until all inputs have been made. Again, the current_size variable will have the number of inputs.

```
double values[CAPACITY];
int current_size = 0;
double input;
while (cin >> input) //cin returns true until
  // invalid (non-numeric) char encountered
{
       if (current_size < CAPACITY)
       {
          values[current_size] = input;
          current_size++;
       }
}
```

Brief C++ by Cay Horstmann
Copyright © 2017 by John Wiley & Sons. All rights reserved

Complete Program to Read Inputs and Report the Maximum

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

int main() //read inputs, print out largest
{
   const int CAPACITY = 1000;
   double values[CAPACITY];
   int current_size = 0;

   cout << "Please enter values, Q to quit:" << endl;
   double input;
   while (cin >> input)
   {
      if (current_size < CAPACITY)
      {
         values[current_size] = input;
         current_size++;
      }
   }
```

Brief C++ by Cay Horstmann
Copyright © 2017 by John Wiley & Sons. All rights reserved

Complete Program to Read Inputs and Report the Maximum Part 2

```
double largest = values[0];
   for (int i = 1; i < current_size; i++)
   {
      if (values[i] > largest)
      {
         largest = values[i];
      }
   }
   for (int i = 0; i < current_size; i++)
   {  //print each element, highlighting largest
      cout << values[i];
      if (values[i] == largest)
      {
         cout << " <== largest value";
      }
      cout << endl;
   }
   return 0;
}
```

Brief C++ by Cay Horstmann
Copyright © 2017 by John Wiley & Sons. All rights reserved