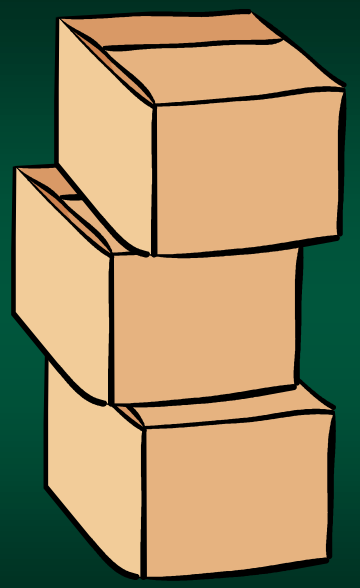
Arrays

Chapter 8

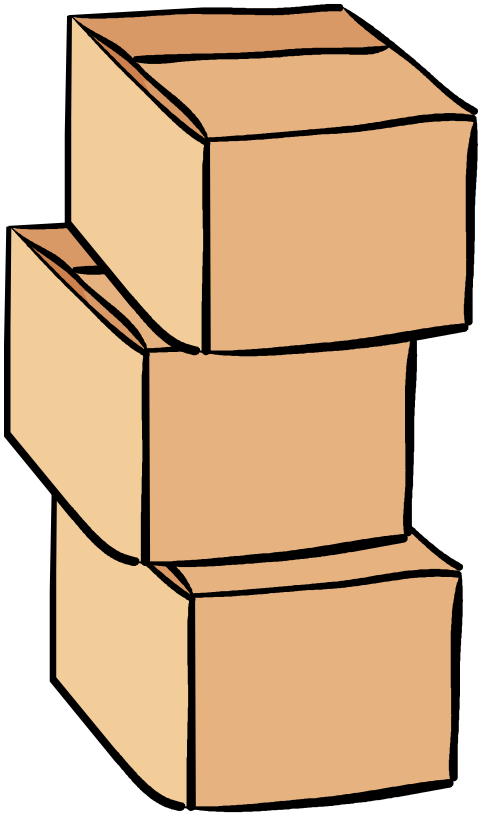


Array Basics

Chapter 8.1

Array Basics

- Normally, variables only have one piece of data associated with them
- An *array* allows you to store a group of items of the same data type together in memory

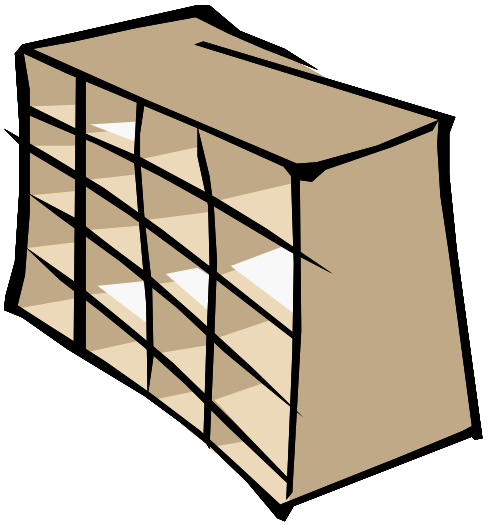


Array Basics

- Why? Instead of creating multiple similar variables such as employee1, employee2, employee3 and so on…
- It's more efficient to create just one variable – with a shared, but multiple values

Metaphor for Arrays

- Think of an array as a *set of mailboxes*
- Each mailbox belongs to the same variable
- Each mailbox has a unique number

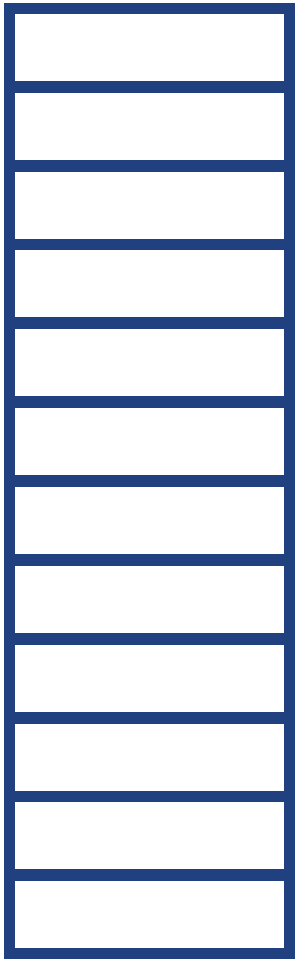


Metaphor for Arrays

- ... or think of arrays as a *group of boxes*
- Each box belongs to the same variable
- Each box has a unique number

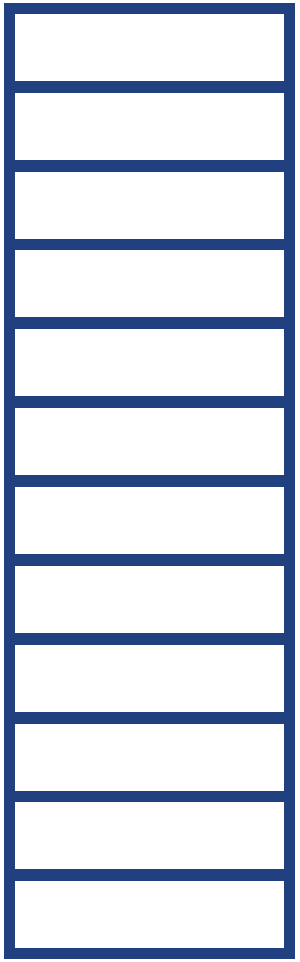
Array Terminology

- Each value located in an array is called an *element*
- Each can be accessed using an unique number called an *index* (also called a *subscript*)



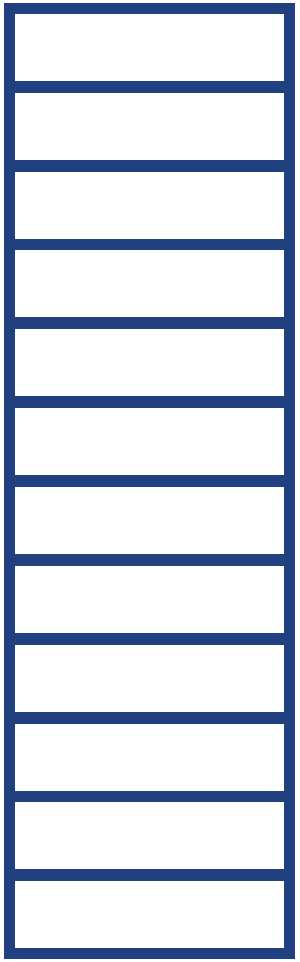
What Value Do We Start With?

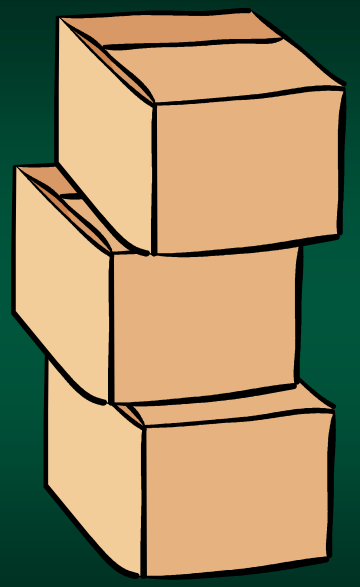
- So, what are the valid values for the index?
- Most languages use *0-indexing*
  - other languages use 1-indexing
  - the success of C set the standard
  - nowadays, most major languages use 0-indexing (e.g. Visual Basic)



Zero Indexing

- This means the first element in any array has the index 0
- So, even though this will be odd and strange, it is something you must learn to live with



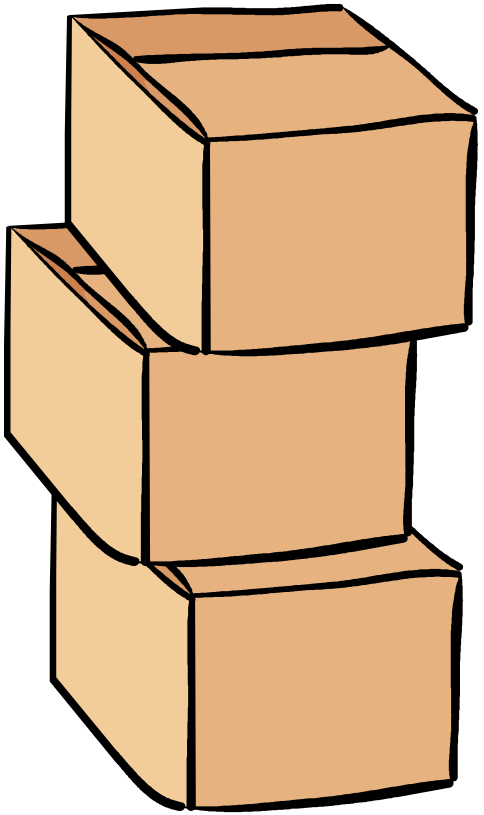


Creating Arrays

Chapter 8.1

Creating Arrays

- Arrays are created pretty much the same as any other variable
- However, since the array can contain multiple values, you must specify its *size*



Book Pseudocode: Array Declare

total number of elements

```
Declare type name [ size ]
```

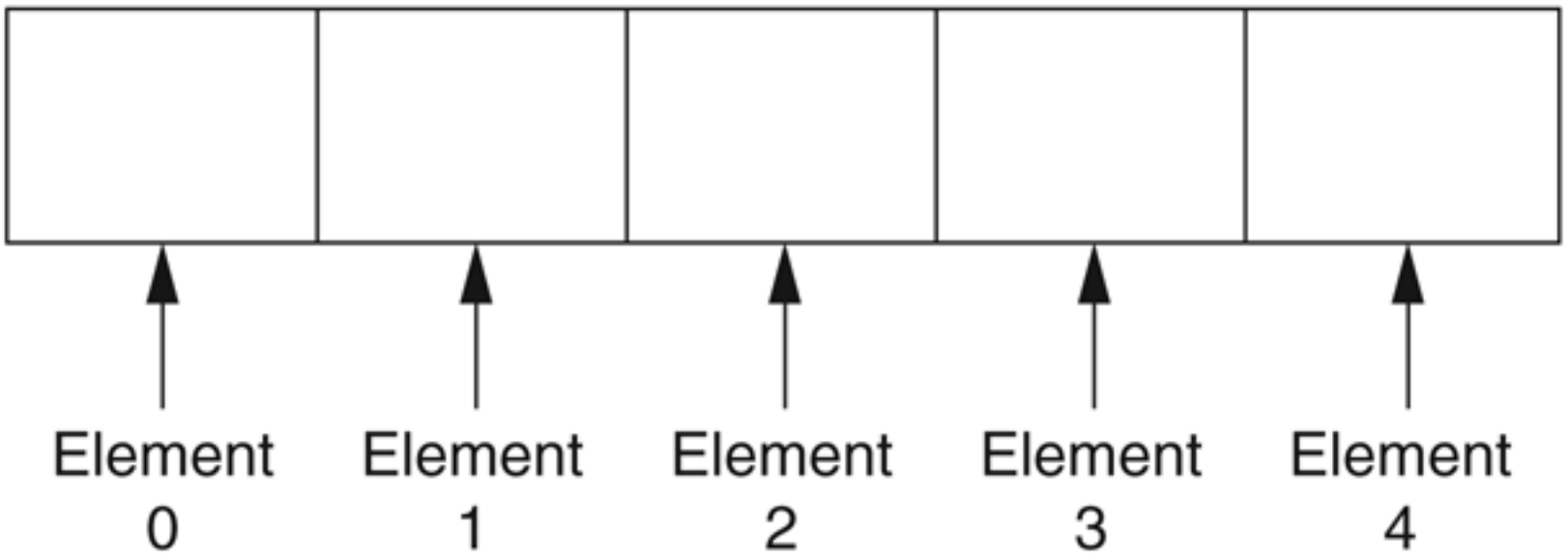
Real, Integer, or String

Examples

```
Declare String employees[50]
Declare Real salesAmounts[7]
```

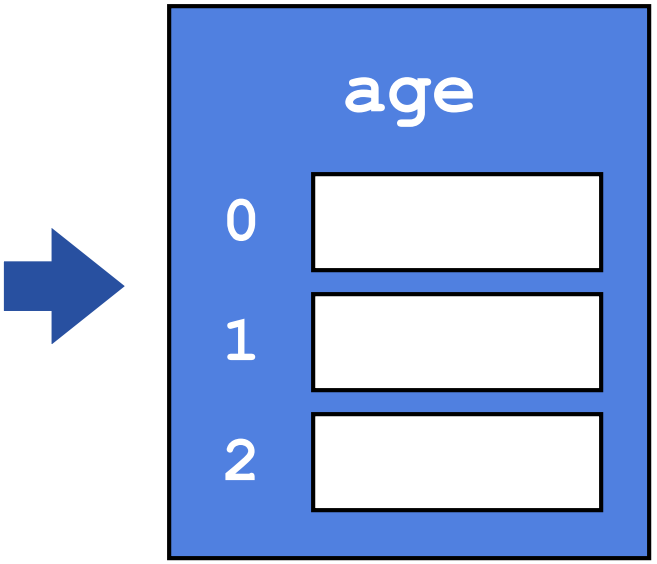
Accessing Elements

```
Constant Integer SIZE = 5
Declare Integer numbers[SIZE]
```



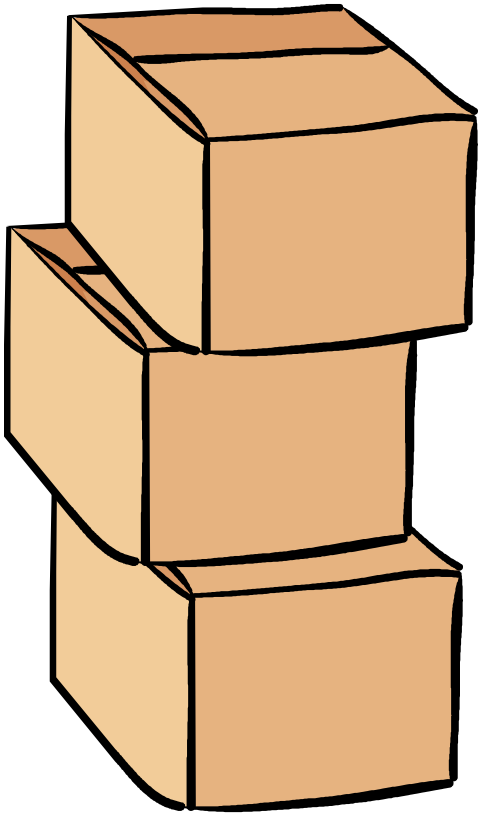
Array Declarations

```
Declare Real age[3]
```



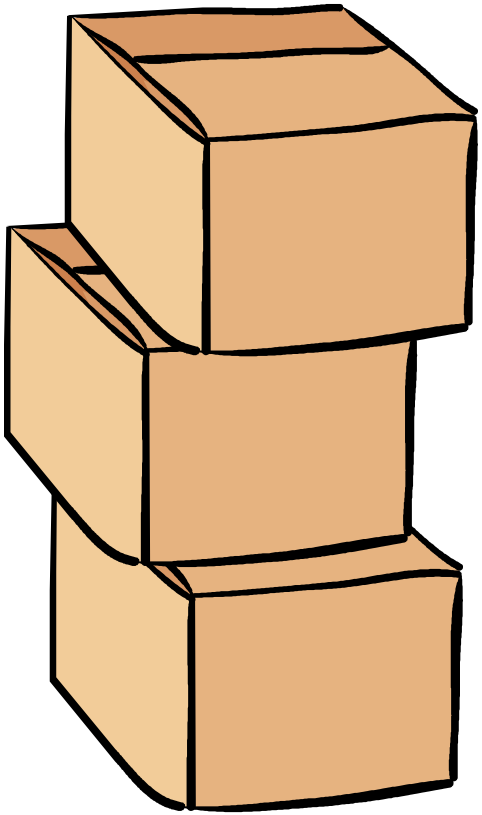
Accessing Each Cell

- After an array is created you can read/write any element
- You can also access the entire array using the variable name

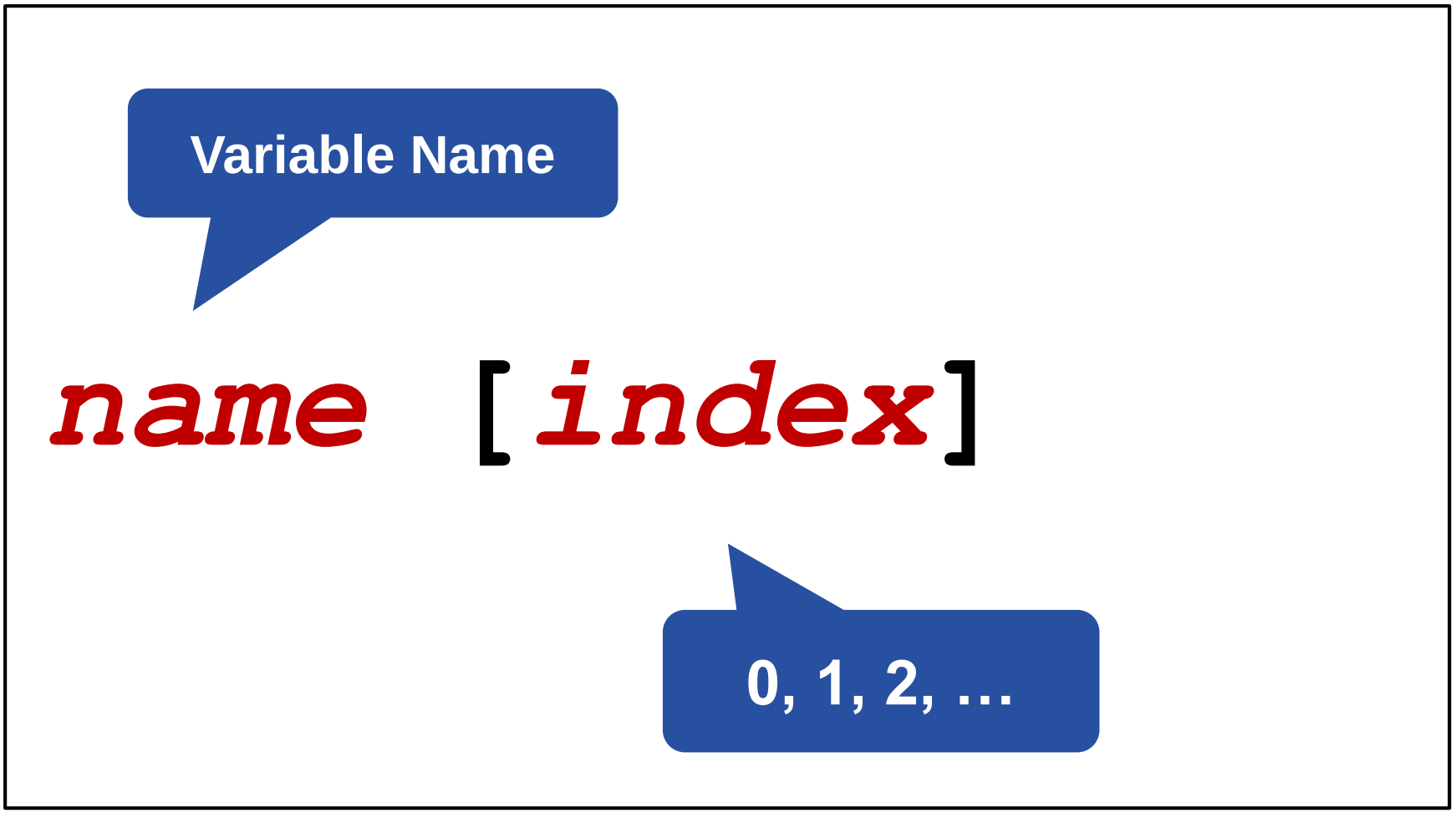


Accessing Each Cell

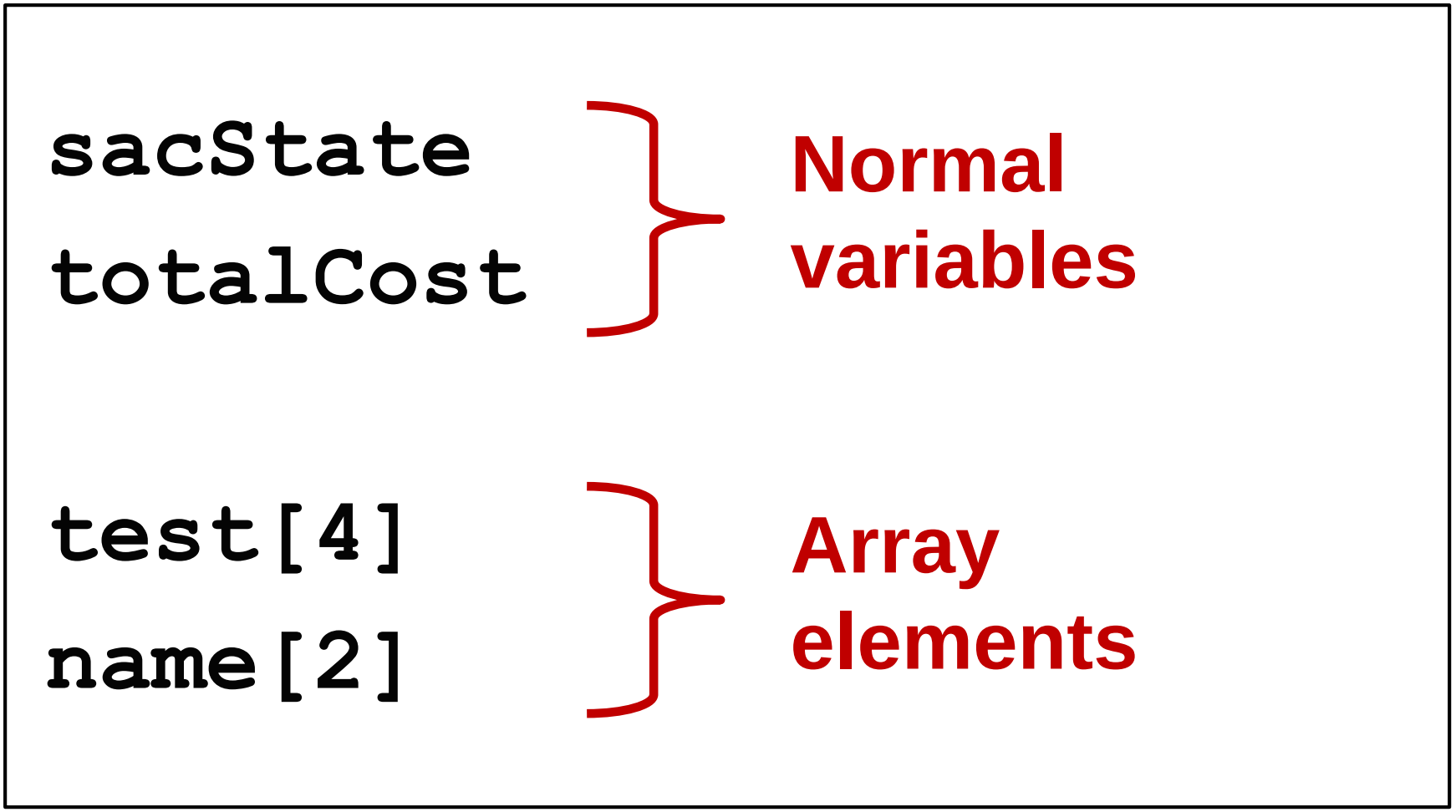
- The notation is incredibly simple
- Simply follow the array name by square brackets and the index of the element you want



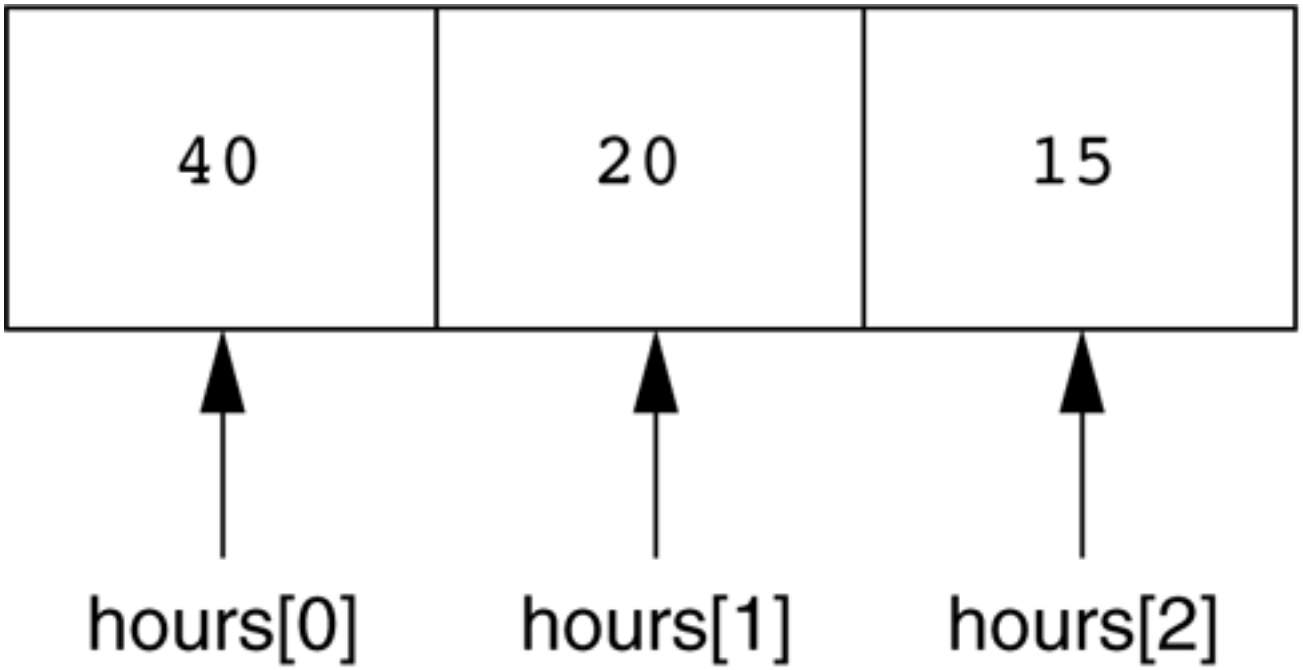
How You Access an Element



Example Variables



Array Elements

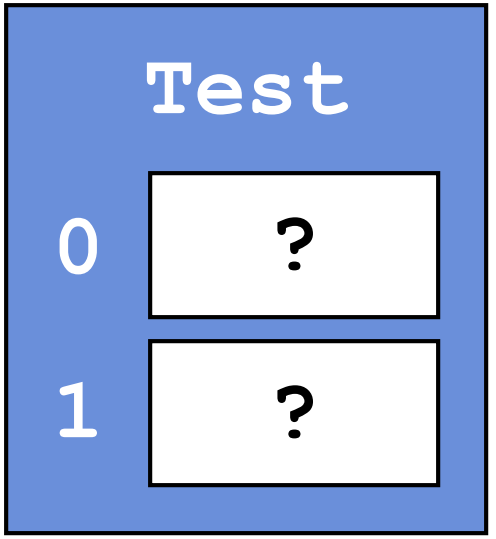


Array Example – What Happens?

```
Declare Integer test[2]

Set test[0] = 75
Set test[1] = 95

Display test[0]
Display test[1]
```

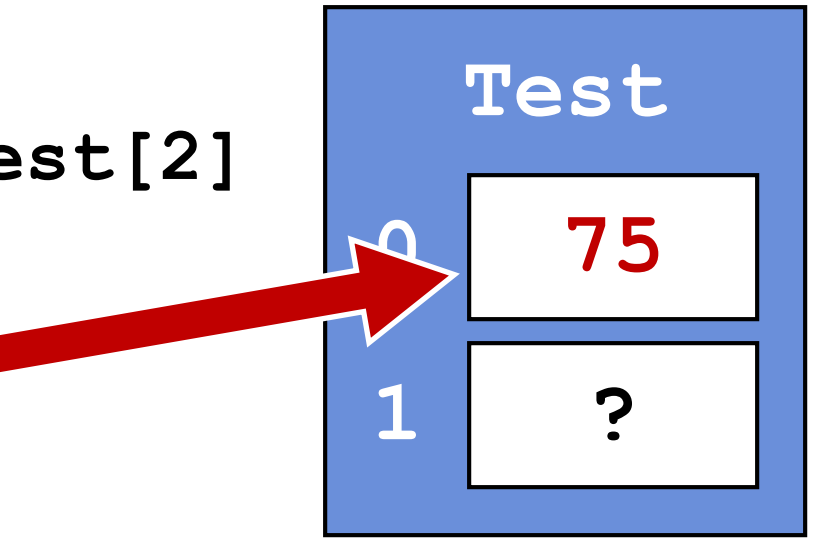
Array Example – What Happens?

```
Declare Integer test[2]

Set test[0] = 75
Set test[1] = 95
```



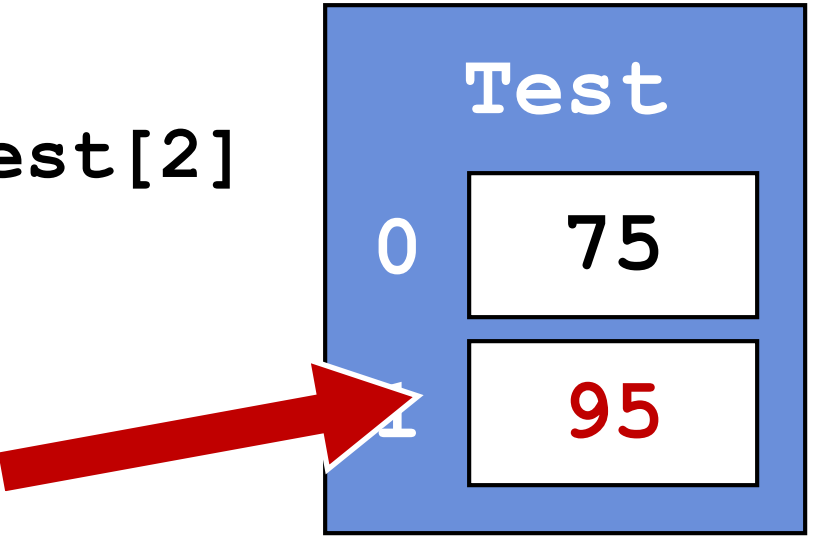
Array Example – What Happens?

```
Declare Integer test[2]

Set test[0] = 75
Set test[1] = 95
```



Array Example – What Happens?

```
Declare Integer test[2]

Set test[0] = 75
Set test[1] = 95
```



Array Example

```
Declare Integer test[2]

Set test[0] = 75
Set test[1] = 95

Display test[0]
Display test[1]
```

Array Example Output

```
75

95
```

Array Initialization

- Just like regular variables, arrays can be initialized to 0 or specific values
- Not all languages support this…
  - however, the big ones such as C#, Java, and Visual Basic do
  - even though the notation varies a bit

Example

```
Declare String days[7] = "Sunday",
    "Monday", "Tuesday", "Wednesday",
    "Thursday", "Friday", "Saturday"
```

Bounds Checking

- Sometimes the program will use an invalid index
- Naturally, this is attempting to access data that does not exist
- Array *bounds checking* prevents the use of an invalid subscript

Example

```
Declare String days[7]

days[7] = "Saturday"
```

Invalid because there is no 7 index

Bounds Checking

- A common error is running a loop one time more than is necessary, exceeding the bound of the array
- This is an *off-by-one error* and is happens quite often (especially so because we use 0-indexing)

Loops and Arrays

Chapter 8.1

For Loops and Arrays

- For Loops are extremely well suited for iterating through all the values of an array
- In fact, one of the reasons For Loops exists – is to interact with arrays

Loops and Arrays

- Using For Loops, it is easy to access all the elements of an array linearly
- The loop variable is used as the index in the array

Array Example

```
Declare String name[4]
Declare Integer n

name[0] = "Tappa Kegga Bru"
name[1] = "Cuppa Kappa Chino"
name[2] = "Hu Delta Phart"
name[3] = "Eta Lotta Pi"

For n = 0 TO 3
   Display name(n)
Next
```

Greek Example Output

```
Tappa Kegga Bru
Cuppa Kappa Chino
Hu Delta Phart
Eta Lotta Pi
```

Loop Example

```
Declare Integer n
Declare String days[7] = "Sunday",
    "Monday", "Tuesday", "Wednesday",
    "Thursday", "Friday", "Saturday"

For n = 0 to 6
    Display days[n]
End For
```

Loop Example Output

```
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
```

Loop Example 2

```
Declare Integer n
Declare String days[7] = "Sunday",
    "Monday", "Tuesday", "Wednesday",
    "Thursday", "Friday", "Saturday"

For n = 0 to 6
    Display n, days[n]
End For
```

Loop Example 2 Output

```
0 Sunday
1 Monday
2 Tuesday
3 Wednesday
4 Thursday
5 Friday
6 Saturday
```

Array - Scores

```
Declare Real score[3]

Set score[0] = 85
Set score[1] = 98
Set score[2] = 61

For n = 0 to 2
   If score[n] >= 70 Then
      Display score[n], " passes"
   Else
      Display score[n], " fails"
   End If
End For
```

Array - Scores Output

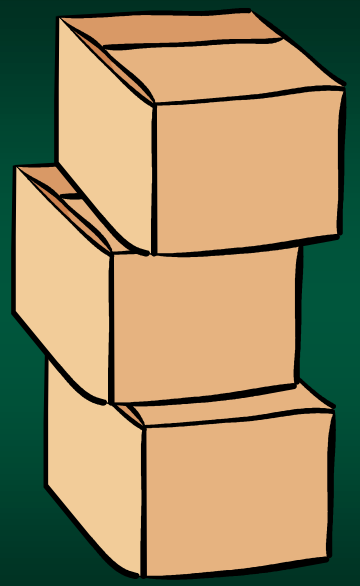
```
85 passes
98 passes
61 fails
```

The For Each Loop

- Some languages provide a For Each loop
- It works with an array, iterating once for each array element
- During each iteration, the loop copies an element's value to a variable.

For Each Example

```
Constant Integer SIZE = 4
Declare Integer numbers[SIZE] = 5, 10, 15, 20
Declare Integer num

For Each num In numbers
      Display num
End For
```

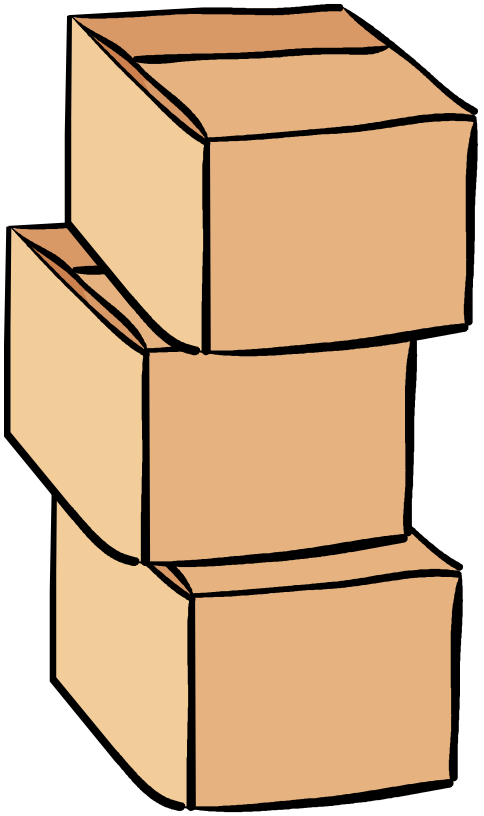


Partially Filled Array

Chapter 8.1

Partially Filled Array

- Sometimes an array is only *partially filled*
- To avoid processing the unfilled elements, you must use an integer variable that holds the number of items stored in the array



Partially Filled Array

- When the array is empty, 0 is stored in this variable
- The variable is incremented each time an item is added to the array
- The variable's value is used as the array's size when stepping through the array.

Partially Filled Arrays

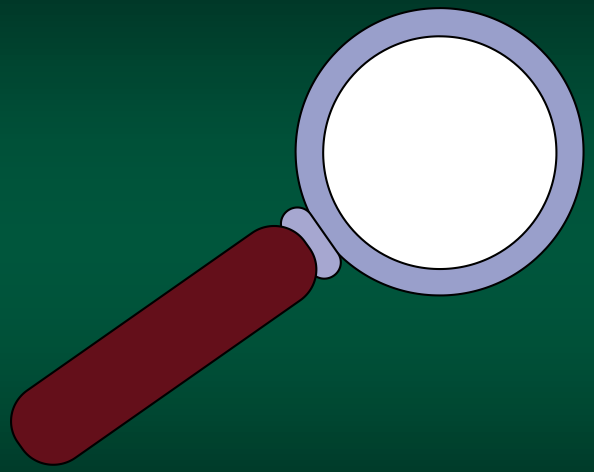
```
Declare Real score[100]
Declare Integer Count

Display "How many tests?"
Input Count

For n = 0 to Count - 1
    Input score[n]
End For
```

100 Capacity

Not using all of it

Array Example 2 Output

```
How many tests?
3
65
89
77
```

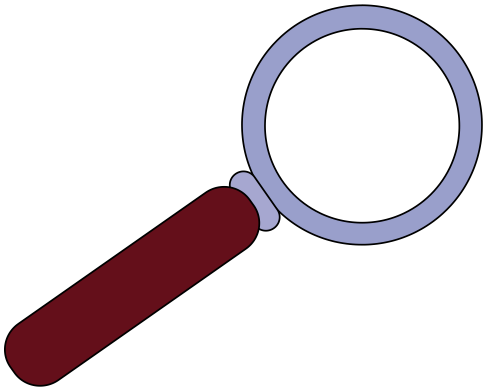


Sequentially Searching an Array

Chapter 8.2

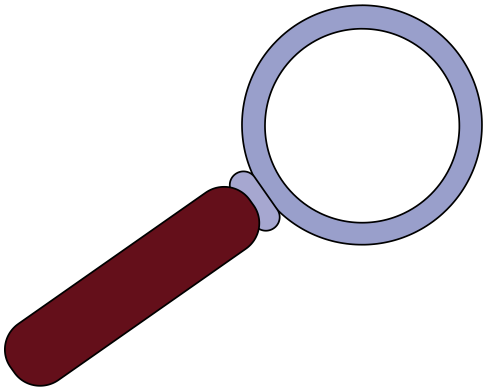
Sequentially Searching an Array

- A *sequential search* algorithm is a simple technique for finding an item in a string or numeric array
- One of the most common and ways of locating data



How it Works

- Uses a loop to sequentially step through an array
- Compares each element with the value being searched for
- Stops when the value is found or the end of the array is hit

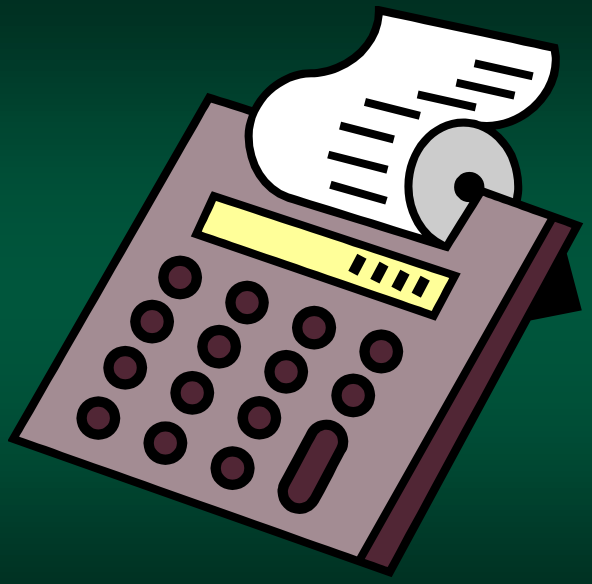


Example

```
Set found = False
Set index = 0

While found == False AND index <= SIZE -1
    If array[index] == searchValue Then
        Set found = True
    Else
        Set index = index + 1
    End If
End While
```

End if found

If not, look at next item (next index)



Processing the Contents of an Array

Chapter 8.3

Processing the Contents of an Array

- It is common to use an array to store multiple values to be analyzed
- The information is first stored, and then analyzed later by a different loop

Calculating the Average…

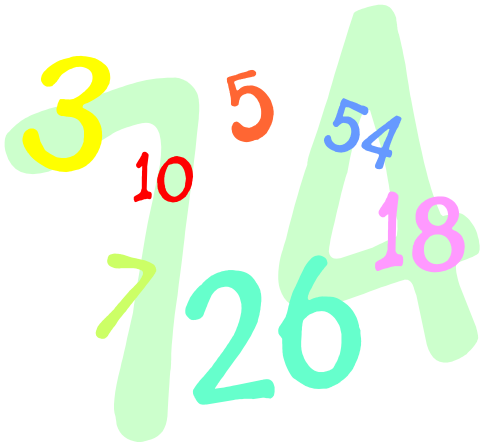
- Loops are used to accumulate the values – create a total
- Then, the total is simply divided
  by the size

Calculate the Average

```
Set total = 0

For n = 0 to SIZE - 1
    total = total + score[n]
End For

Set average = total / SIZE
```

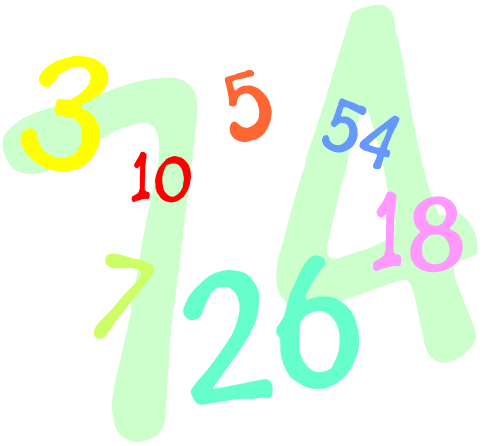
Highest and Lowest Value

- Finding the highest & lowest values in an array is common
- It basically works by scanning the array and setting a high (or low value) based on the current value



Steps Involved – Highest Value

- Create a variable to hold the highest value
- Assign the value at element 0 to the highest
- Use a loop to step through the rest of the elements
- Each iteration, a comparison is made to the highest variable
- If the element is greater than the highest value, that value is then the assigned to the highest variable



Highest Value

```
Set highest = scores[0]
For n = 1 to count - 1
    if score[n] > highest
        Set highest = score[n]
    end if
End For
```

Lowest Value

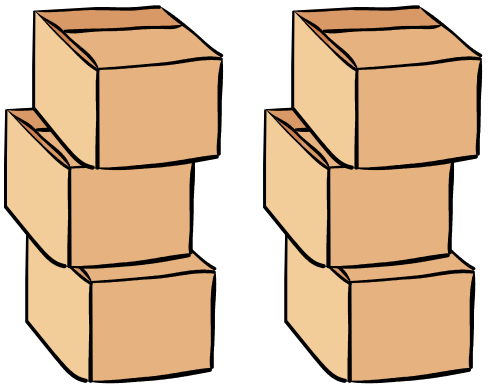
```
Set lowest = scores[0]
For n = 1 to count - 1
    if score[n] < lowest
        Set lowest = score[n]
    end if
End For
```

Parallel Arrays

Chapter 8.4

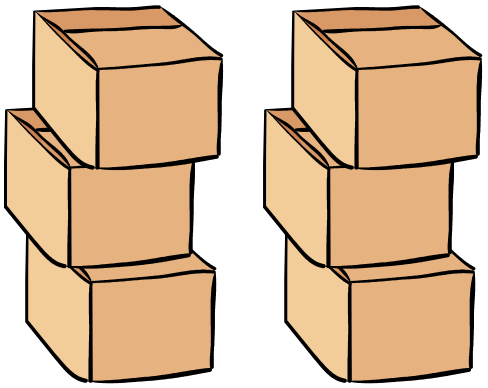
Parallel Arrays

- Often more than one piece of information needs to be saved for the same "object"
- One common approach is to use multiple arrays – one for each type of data

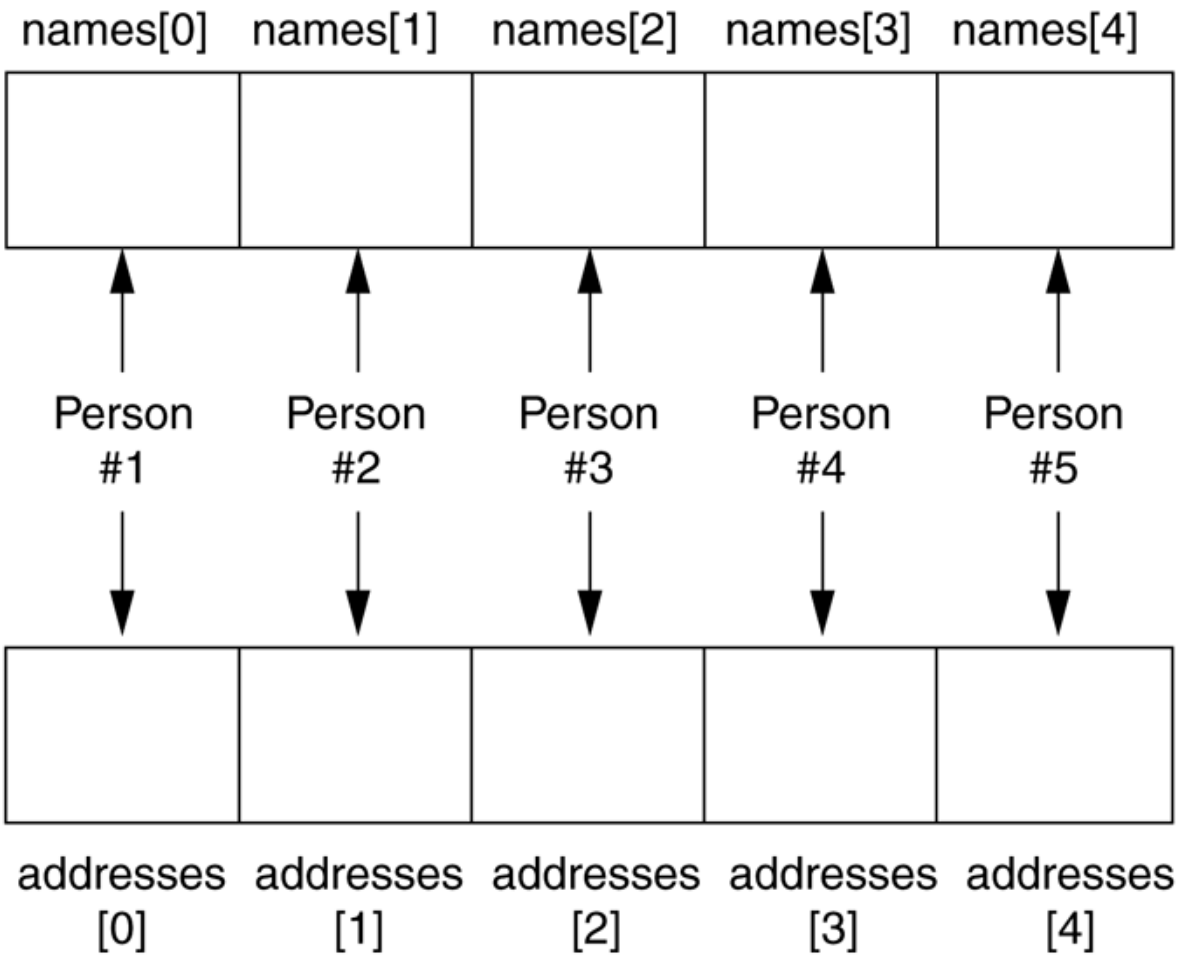


Parallel Arrays

- These are called *parallel arrays*
- They are separate arrays but we used the same index for each
- By using the same index, we can establish a relationship between them



Parallel Arrays



Example: Part 1 – Input

```
Constant SIZE = 10
Declare String name[SIZE]
Declare Integer cash[SIZE]

For n = 0 to SIZE - 1
    Input name[n]
    Input cash[n]
End For
```

Example: Part 2 – Search

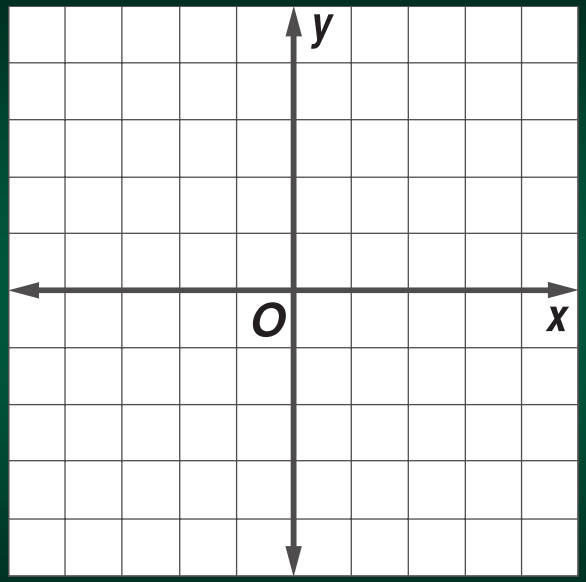
```
Set highest = 0
For n = 1 to SIZE - 1
    If points [n] > cash[highest]
        Set highest = n
    End If
End For
```

Example: Part 3 – Results

```
Display name[highest],
Display "got the most donations of $",
        points[highest]
```

Example Output

```
Tappa Kegga Bru
32.23
Lambda Lambda Lambda
432.11
Eta Lotta Pi
54.25

Lambda Lambda Lambda
got the most donations of $432.11
```

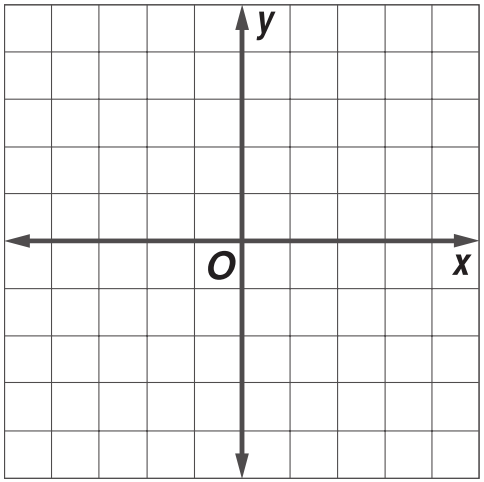


Two Dimension Arrays

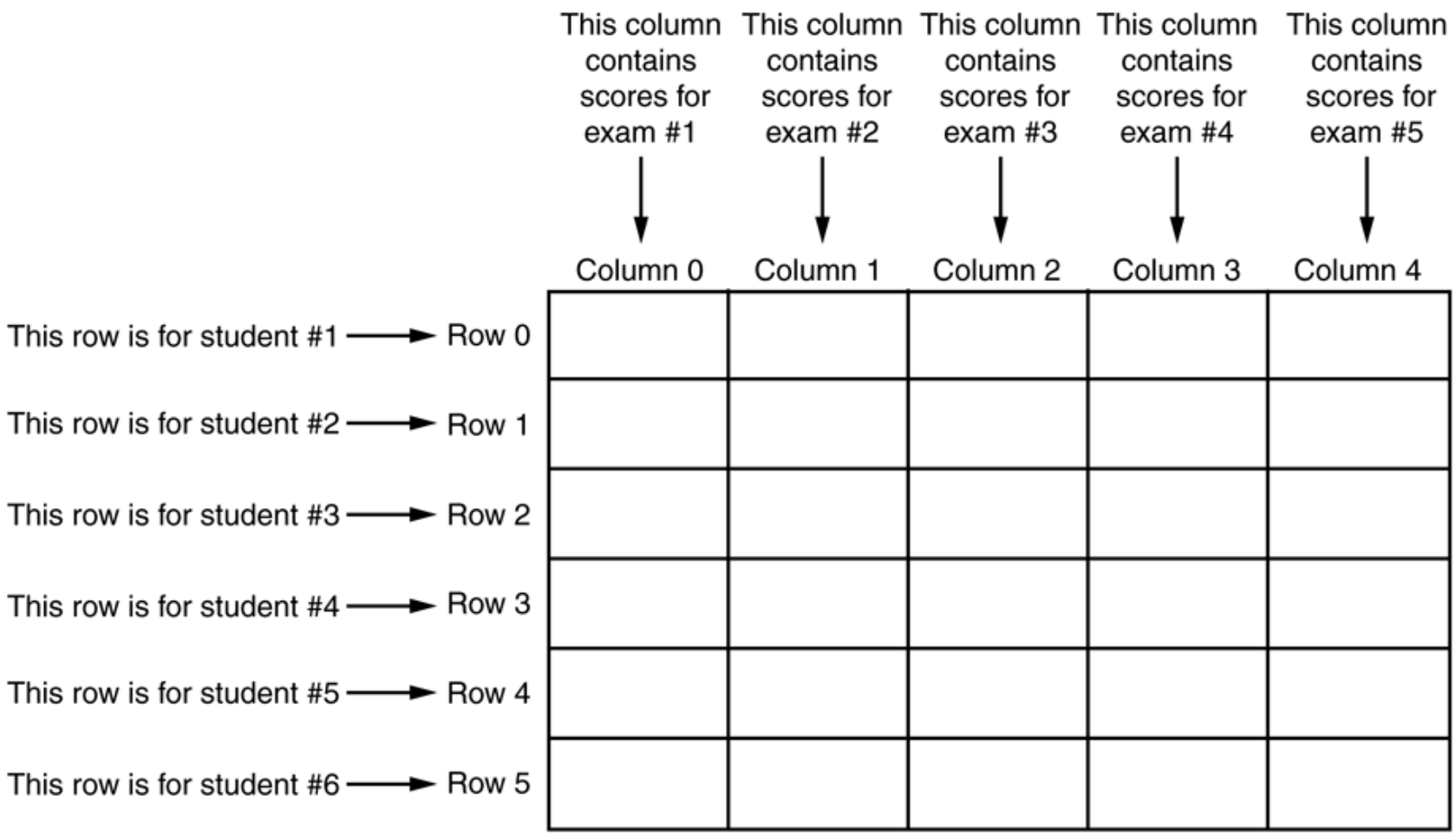
Chapter 8.5

Two Dimension Arrays

- A two-dimensional array is like several identical arrays put together
- Suppose a teacher has six students who take five tests



Two-Dimensional Arrays



Multiple Sizes Needed

- Two size variables are required when declaring two dimensional arrays
- Accessing is done with two loops, and both indexes

Declaring Arrays

```
Constant Integer ROWS = 3
Constant Integer COLS = 4

Declare Integer values[ROWS][COLS]
```

Using Loops

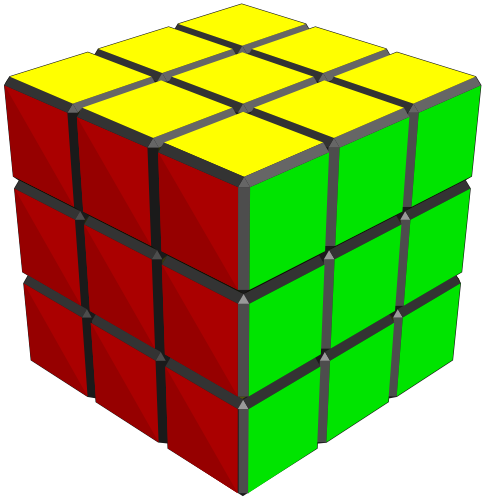
```
For row = 0 To ROWS -1
    For col = 0 To COLS - 1
        Display "Enter a number."
        Input values[row][col]
    End For
End For
```

Arrays of Three or More Dimensions

Chapter 8.6

Arrays of Three or More Dimensions

- Arrays can also be three or more dimensions
- In fact, there are no limitations once the number of dimensions



Arrays of Three or More Dimensions

```
Declare Real seats[3][5][8]
```

Arrays of Three or More Dimensions