Programming Logic & Design

Chapter 8 – Arrays – Part 1

Queen's College CSD

1133 – CPCM -2023S

Chapter 8 – Arrays

CONCEPT: An array allows you to store a group of items of the same data type together in memory. Processing a large number of items in an array is usually easier than processing a large number of items stored in separate variables.

In the programs you have designed so far, you have used variables to store data in memory. In most programming languages, the simplest way to store a value in memory is to store it in a variable. Variables work well in many situations, but they have limitations. For example, they can hold only one value at a time. Consider the following pseudocode variable declaration:

Declare Integer number = 99
Consider what happens if the following statement appears later in the program:

Set number = 5

Chapter 8 – Arrays

Because variables hold only a single value, they can be cumbersome in programs that process lists of data. For example, suppose you are asked to design a program that holds the names of 50 employees. Imagine declaring 50 variables to hold all of those names:

```
Declare String employee1
Declare String employee2
Declare String employee3
and so on . . .
Declare String employee50
```

Then, imagine designing the code to input all 50 names:

```
// Get the first employee name.
Display "Enter the name of employee 1."
Input employee1

// Get the second employee name.
Display "Enter the name of employee 2."
Input employee2

// Get the third employee name.
Display "Enter the name of employee 3."
Input employee3
and so on . . .

// Get the fiftieth employee name.
Display "Enter the name of employee 50."
Input employee50
```

Chapter 8 – Arrays

As you can see, variables are not well-suited for storing and processing lists of data. Each variable is a separate item that must be declared and individually processed. Fortunately, most programming languages allow you to create arrays, which are specifically designed for storing and processing lists of data. Like a variable, an array is a named storage location in memory. Unlike a variable, an array can hold a group of values. All of the values in an array must be the same data type. You can have an array of Integers, an array of Reals, or an array of Strings, but you cannot store a mixture of data types in an array. The following example shows how we will declare an array in pseudocode:

```
Declare Integer units[10]
```

Notice that this statement looks like a regular Integer variable declaration except for the number inside the brackets. The number inside the brackets, called a size declarator, specifies the number of values that the array can hold

Chapter 8 – Arrays

In most languages, an array's size cannot be changed while the program is running. If you have written a program that uses an array and then find that you must change the array's size, you have to change the array's size declarator in the source code. Then you must recompile the program (or rerun the program if you are using an interpreted language) with the new size declarator. To make array sizes easier to maintain, many programmers prefer to use named constants as array size declarators.

Constant Integer SIZE = 10

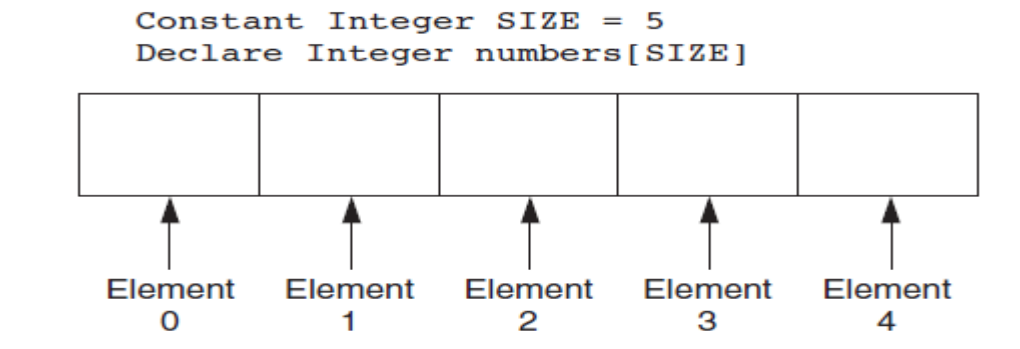
Declare Integer units[SIZE]

Array Elements and Subscripts

The storage locations in an array are known as elements. In memory, an array's elements are usually located in consecutive memory locations. Each element in an array is assigned a unique number known as a subscript. Subscripts are used to identify specific elements in an array. In most languages, the first element is assigned the subscript 0, the second element is assigned the subscript 1, and so forth

Chapter 8 – Arrays

As shown in following figure , the numbers array has five elements. The elements are assigned the subscripts 0 through 4. (Because subscript numbering starts at zero, the subscript of the last element in an array is one less than the total number of elements in the array.)



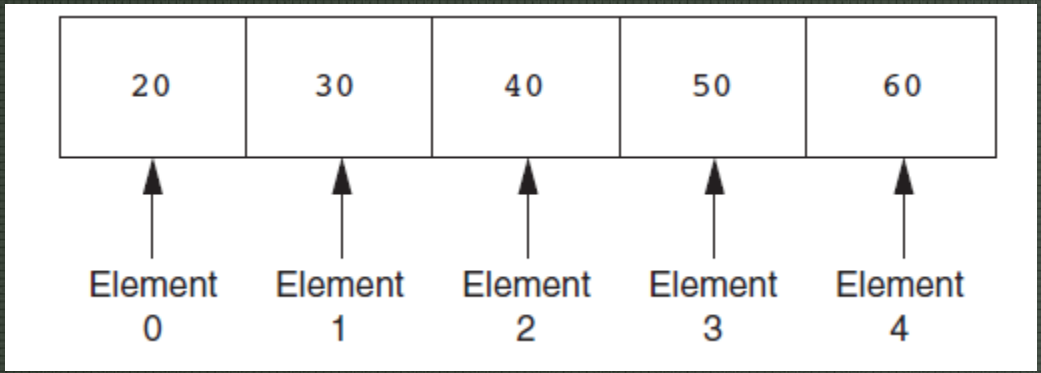
Assigning Values to Array Elements

You access the individual elements in an array by using their subscripts. For example,assuming numbers is the Integer array just described, the following pseudocode assigns the values to each of its five elements.

Chapter 8 – Arrays

```
Set numbers[0] = 20
Set numbers[1] = 30
Set numbers[2] = 40
Set numbers[3] = 50
Set numbers[4] = 60
```

This pseudocode assigns the value 20 to element 0, the value 30 to element 1, and so forth. Following figure shows the contents of the array after these statements execute.



Chapter 8 – Arrays

This pseudocode assigns the value 20 to element 0, the value 30 to element 1, and so forth. Following figure shows the contents of the array after these statements execute.

Inputting and Outputting Array Contents

You can read values from the keyboard and store them in an array element just as you can a regular variable. You can also output the contents of an array element. The pseudocode given below shows an array being used to store and display values entered by the user.

```
// Create a constant for the number of employees.
Constant Integer SIZE = 3

// Declare an array to hold the number of hours
// worked by each employee.
Declare Integer hours[SIZE]

// Get the hours worked by employee 1.
Display "Enter the hours worked by employee 1."
Input hours[0]

// Get the hours worked by employee 2.
Display "Enter the hours worked by employee 2."
Input hours[1]
```

Chapter 8 – Arrays

```
// Get the hours worked by employee 3.
Display "Enter the hours worked by employee 3."
Input hours[2]

// Display the values entered.
Display "The hours you entered are:"
Display hours[0]
Display hours[1]
Display hours[2]
```

Program Output (with Input Shown in Bold)

```
Enter the hours worked by employee 1.
40 [Enter]
Enter the hours worked by employee 2.
20 [Enter]
Enter the hours worked by employee 3.
15 [Enter]
The hours you entered are:
40
20
15
```

Using a Loop to Step Through an Array

Most programming languages allow you to store a number in a variable and then use that variable as a subscript. This makes it possible to use a loop to step through an entire array, performing the same operation on each element

```
// Declare an Integer array with 10 elements.
Declare Integer series[10]

// Declare a variable to use in the loop.
Declare Integer index

// Set each array element to 100.
For index = 0 To 9
   Set series[index] = 100
End For
```

Chapter 8 – Arrays

```
// Create a constant for the size of the array.
Constant Integer SIZE = 3

// Declare an array to hold the number of hours
// worked by each employee.
Declare Integer hours[SIZE]

// Declare a variable to use in the loops.
Declare Integer index

// Get the hours for each employee.
For index = 0 To SIZE - 1
   Display "Enter the hours worked by"
   Display "employee number ", index + 1
   Input hours[index]
End For

// Display the values entered.
Display "The hours you entered are:"
For index = 0 To SIZE - 1
   Display hours[index]
End For
```

```
Enter the hours worked by
employee number 1
40 [Enter]
Enter the hours worked by
employee number 2
20 [Enter]
Enter the hours worked by
employee number 3
15 [Enter]
The hours you entered are:
40
20
15
```

Chapter 8 – Arrays

Processing the Elements of an Array

Processing array elements is no different than processing other variables. In the previous programs we saw how we can assign values to array elements, store input in array elements, and display the contents of array elements. The following Program shows how array elements can be used in math expressions

Problem :

Megan owns a small neighborhood coffee shop, and she has six employees who work as baristas (coffee bartenders). All of the employees have the same hourly pay rate. Megan has asked you to design a program that will allow her to enter the number of hours worked by each employee and then display the amounts of all the employees' gross pay. You determine that the program should perform the following steps:

1. For each employee: get the number of hours worked and store it in an array element.
2. For each array element: use the value stored in the element to calculate an employee's gross pay. Display the amount of the gross pay.

Chapter 8 – Arrays

```
// Constant for the size of the array.
Constant Integer SIZE = 6

// Array to hold each employee's hours.
Declare Real hours[SIZE]

// Variable to hold the hourly pay rate.
Declare Real payRate

// Variable to hold a gross pay amount.
Declare Real grossPay

// Variable to use as a loop counter.
Declare Integer index

// Get each employee's hours worked.
For index = 0 To SIZE - 1
   Display "Enter the hours worked by"
   Display "employee ", index + 1
   Input hours[index]
End For
```

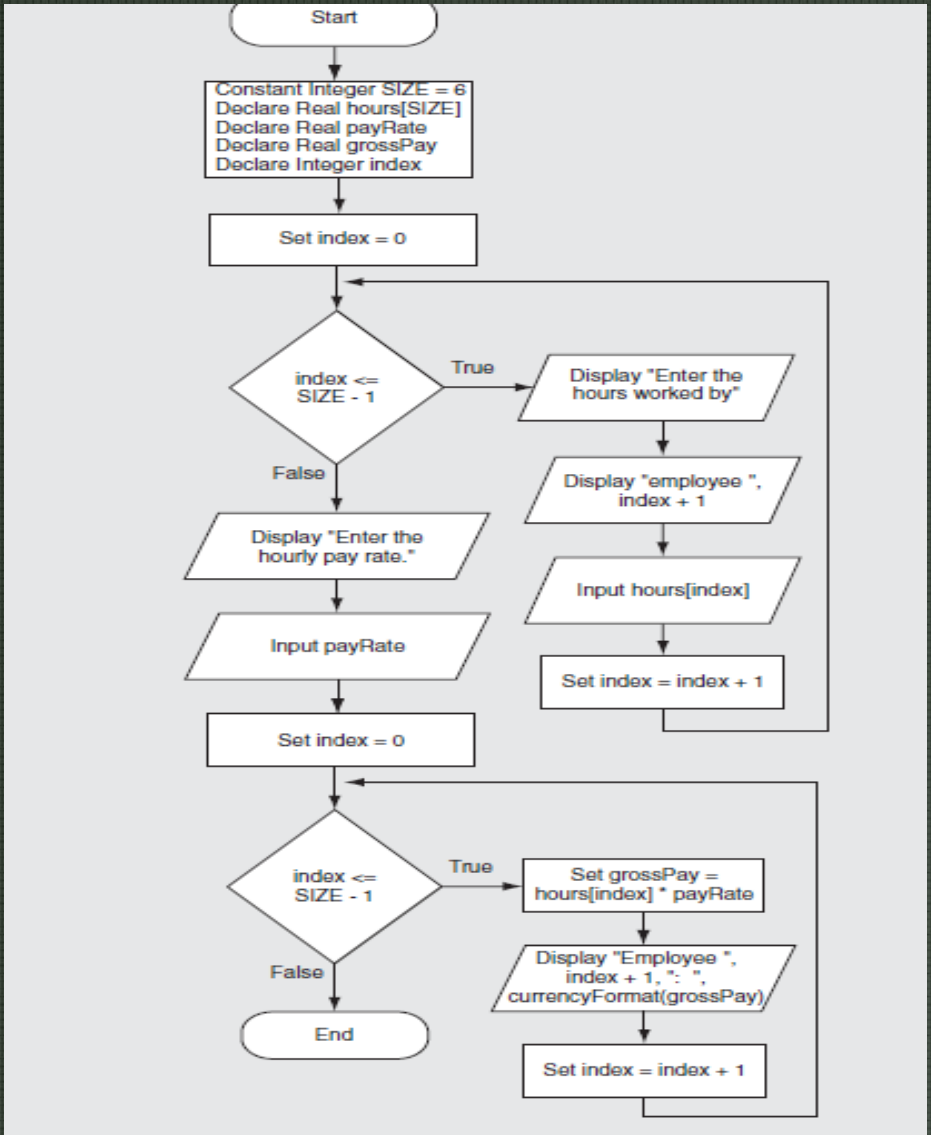
Chapter 8 – Arrays

```
// Get the hourly pay rate.
Display "Enter the hourly pay rate."
Input payRate

// Display each employee's gross pay.
Display "Here is each employee's gross pay."
For index = 0 To SIZE - 1
   Set grossPay = hours[index] * payRate
   Display "Employee ", index + 1, ": $",
           currencyFormat(grossPay)
End For
```

```
Enter the hours worked by employee 1.
10 [Enter]
Enter the hours worked by employee 2.
20 [Enter]
Enter the hours worked by employee 3.
15 [Enter]
Enter the hours worked by employee 4.
40 [Enter]
Enter the hours worked by employee 5.
20 [Enter]
Enter the hours worked by employee 6.
18 [Enter]
Enter the hourly pay rate.
12.75 [Enter]
Here is each employee's gross pay.
Employee 1: $127.50
Employee 2: $255.00
Employee 3: $191.25
Employee 4: $510.00
Employee 5: $255.00
Employee 6: $229.50
```

Chapter 8 – Arrays



Chapter 8 – Arrays

Array Initialization

Most languages allow you to initialize an array with values when you declare it. In this book's pseudocode, we will initialize arrays in the following manner:

```
Constant Integer SIZE = 5
Declare Integer numbers[SIZE] = 10, 20, 30, 40, 50
```

The series of values separated with commas is called an initialization list. These values are stored in the array elements in the order they appear in the list. (The first value,10, is stored in numbers[0], the second value, 20, is stored in numbers[1], and so forth.) Here is another example:

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

This pseudocode declares days as an array of 7 Strings, and initializes days[0] with "Sunday", days[1] with "Monday", and so forth.

Chapter 8 – Arrays

Array Bounds Checking

Most programming languages perform array bounds checking, which means they do not allow a program to use an invalid array subscript. For example, look at the following pseudocode:

```
// Create an array.
Constant Integer SIZE = 5
Declare Integer numbers[SIZE]
// ERROR! This statement uses an invalid subscript!
Set numbers[5] = 99
```

This pseudocode declares an array with 5 elements. The subscripts for the array's elements are 0 through 4. The last statement will cause an error in most languages because it attempts to assign a value in numbers[5], a nonexistent element.

Chapter 8 – Arrays

Check Point

1 Can you store a mixture of data types in an array?

2 What is an array size declarator?

3 In most languages, can the size of an array be changed while the program is running?

4 What is an array element?

5 What is a subscript?

6 What is usually the first subscript in an array?

7 Look at the following pseudocode and answer questions a through d.

```
Constant Integer SIZE = 7
Declare Real numbers[SIZE]
```

a. What is the name of the array that is being declared?

b. What is the size of the array?

c. What data type are the array elements?

d. What is the subscript of the last element in the array?

8 What does “array bounds checking” mean?