# **Lab 9: Arrays**

This lab accompanies Chapter 8 of *Starting Out with Programming Logic & Design*.

Name: David White

**Lab 9.1 – Arrays and Pseudocode**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Critical Review  An array allows you to store a group of items of the same data type together in memory.  A variable stores just a single value, and oftentimes can be cumbersome to work with when your program has similar values.  Values stored in an array are called elements. Each element has a subscript that makes it unique.  An array is defined as follows:  Declare Integer numbers[10]  Integer defines the type of numbers that can be stored, numbers is the name of the array, and [10] is how many numbers can be stored.  In most languages, the first element is assigned the subscript of 0, so the above array actually runs from 0 to 9.  Constant variables can also be used to declare the size of an array.  Constant Integer SIZE = 5  Declare Integer numbers[SIZE] = 847, 1238, 48, 123, 840  Elements in the array   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 847 | 1238 | 48 | 123 | 840 | | 0 | 1 | 2 | 3 | 4 |   Subscript or Index starting a 0  Loops are generally used to step through an array. This can be done using any type of loop and for any process such as filling, calculating, searching, sorting, or outputting elements of the array. |

This lab examines the various ways of working with arrays by writing pseudocode. Read the following programming problem prior to completing the lab.

The American Red Cross wants you to write a program that will calculate the average pints of blood donated during a blood drive. The program should take in the number of pints donated during the drive, based on a seven hour drive period. The average pints donated during that period should be calculated and displayed. Additionally, the highest and the lowest number of pints donated should be determined and displayed. Write a loop around the program to run multiple times.

**Step 1:** Declare the following variables:

* An array named pints of the data type Real of size 7
* A variable named totalPints of the data type Real
* A variable named averagePints of the data type Real initialized to 0
* A variable named highPints of the data type Real initialized to 0
* A variable named lowPints of the data type Real initialized to 0

Module main()

//Declare local variables

Declare String again = “no”

Declare Real pints[7]

Declare real totalPints

Declare real averagePints = 0

Declare real highPints = 0

Declare real lowPints = 0

While again == “no”

//module calls below

Display “Do you want to run again: yes or no”

Input again

End While

End Module

**Step 2:** Write a module call to a module named getPints that passes the pints array. Additionally, write a module header named getPints that accepts the pints array. (Reference: Passing an Array as an Argument to a Module or Function, page 376).

//Module call

Call getPints(pints)

//Module header

Module getPints(Real pints[7])

**Step 3:** Write a for loop that runs 7 times using the counter variable. Inside the for loop, allow the user to enter values into the array. (Reference: Using a Loop to Step Through an Array, page 355).

Declare Integer counter = 0

For counter = 0 to 6

Display “Enter pints collected:”

Input pints[counter]

End For

**Step 4:** Write a function call to a module named getTotal that passes the pints array and the totalPints variable. Additionally, write a function header named getTotal that accepts the pints array and the totalPints variable.

//Function call

totalPints = getTotal(pints, totalPints)

//Function header

Function getTotal(real pints[7], real totalPints)

**Step 5:** Write a for loop that runs 7 times using the counter variable. Inside the for loop, total up the values of the array and store in the variable totalPints. Also, return the correct variable from the function. (Reference: Totaling the Values in an Array, page 371).

Declare Integer counter = 0

Set totalPints = 0

For counter = 0 to 6

Set totalPints = totalPints + pints[counter]

End For

Return totalPints

**Step 6:** Write a function call to a module named getAverage that passes the totalPints variable and the averagePints variable. Additionally, write a function header named getAverage that accepts the totalPints variable and the averagePints variable.

//Function call

averagePints = getAverage(totaPints, averagePints)

//Function header

Function getAverage(real totalPints, real averagePints)

**Step 7:** Write a statement that will calculate the average pints donated over the drive period. Also, return the correct variable from the function. (Reference: Averaging the Values in an Array, page 372).

averagePints = totalPints / 7

Return averagePints

**Step 8:** Write a function call to a module named getHigh that passes the highPints variable and the pints array. Additionally, write a function header named getHigh that accepts the highPints variable and the pints array.

//Function call

highPints = getHigh(highPints, pints)

//Function header

Function getHigh(Real highPints, Real pints[7])

**Step 9:** Write the code that will determine the highest value in an array. Also, return the correct variable from the function. (Reference: Finding the Highest Value in an Array, page 373).

Set highPints = pints[index]

Set index = 0

For index = 0 to 6

If pints[index] > highPints Then

Set highPints = pints[index]

End If

End For

Return highPints

**Step 10:** Write a function call to a module named getLow that passes the lowPints variable and the pints array. Additionally, write a function header named getLow that accepts the lowPints variable and the pints array.

//Function call

lowPints = getLow(lowPints, pints)

//Function header

Function getLow(real lowPints, real pints[7]

**Step 11:** Write the code that will determine the lowest value in an array. Also, return the correct variable from the function. (Reference: Finding the Lowest Value in an Array, page 374).

Set lowPints = pints[index]

Set index = 1

For index = 1 to 6

If pints[index] < lowPints Then

Set lowPints = pints[index]

End If

End For

Return lowPints

**Step 12:** Write a module call to a module named displayInfo. Pass the necessary variable to the functions that are needed to display the averagePints, the highPints, and the lowPints. Also, write the module header that accepts the same variables.

//Module call

Call displayInfo(averagePints, highPints, lowPints)

//Module header

Module displayInfo(real averagePints, real highPints, real lowPints)

**Lab 9.2 – Checking the Work**

Using the program from Lab 9.1, complete the following checks for a better understanding of your work.

**Step 1:** Imagine the following number of pints were entered into the array.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Element | 34 | 39 | 25 | 18 | 43 | 31 | 12 |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

**Step 2:** Recall Step 5 of Lab 9.1 that accumulates the pints collected.

Declare Integer counter = 0

Set totalPints = 0

For counter = 0 to 6

Set totalPints = totalPints + pints[counter]

End For

**Step 3:** Complete the following chart by writing what the counter and the totalPints value stores on each iteration of the loop.

|  |  |
| --- | --- |
| **Counter** | **totalPints** |
| 0 | 34 |
| 1 | 73 |
| 2 | 98 |
| 3 | 116 |
| 4 | 159 |
| 5 | 190 |
| 6 | 202 |

**Step 4:** Recall Step 9 from Lab 9.1 that determines the high value.

Set highPints = pints[0]

Set index = 1

For index = 1 to 6

If pints[index] > highPints Then

Set highPints = pints[index]

End If

End For

**Step 5:** Complete the following chart by writing what the highPints and the pints array value stores on each iteration of the loop. Also conclude whether it will be True or False.

|  |  |  |
| --- | --- | --- |
| **Pints** | **highPints** | **True or False** |
| 39 | 34 | TRUE |
| 25 | 39 | FALSE |
| 18 | 39 | FALSE |
| 43 | 39 | TRUE |
| 31 | 43 | FALSE |
| 12 | 43 | FALSE |

**Step 6:** Recall Step 11 from Lab 9.1 that determines the low value.

Set lowPints = pints[0]

Set index = 1

For index = 1 to 6

If pints[index] < lowPints Then

Set lowPints = pints[index]

End If

End For

**Step 7:** Complete the following chart by writing what the lowPints and the pints array value stores on each iteration of the loop. Also conclude whether it will be True or False.

|  |  |  |
| --- | --- | --- |
| **Pints** | **lowPints** | **True or False** |
| 39 | 34 | FALSE |
| 25 | 34 | TRUE |
| 18 | 25 | TRUE |
| 43 | 18 | FALSE |
| 31 | 18 | FALSE |
| 12 | 18 | TRUE |

**Lab 9.3 – Arrays and Flowchart**

This lab requires you to create a flowchart for the blood drive program in Lab 9.1. Use an application such Visio, Draw.io (must convert to a PDF file before submitting), or simply draw the flowchart by hand and upload a picture of it in JPG format.

**The final step is to upload your finished flowchart to Blackboard.** You may paste your flowchart here as well for reference.



 **Lab 9.4 – Arrays and Python Code**

Critical Review

In Python, arrays are native objects called lists.

List index starts at 0.

The following is a method used when you know the elements of the array.

even\_numbers = [2, 4, 6, 8, 10]

You can use the print statement to display an entire list, as shown here:

print(even\_numbers)

The following is a method used when you do not know what the elements should be, but you know the size.

numbers = [0] \* 5

A loop can also be used to print the elements of the array. An example is as follows:

//A for in loop

for n in numbers:

print(n)

//A while loop

index = 0

while index < 5:

print(numbers [index])

index = index + 1

The goal of this lab is to convert the blood drive program from Lab 9.1 to Python code.

**Step 1:** Start the IDLE Environment for Python. Prior to entering code, save your file by clicking on File and then Save. Select your location and save this file as *Lab*9-4*.py*. Be sure to include the .py extension.

**Step 2:** Document the first few lines of your program to include your name, the date, and a brief description of what the program does.

**Step** **3:** Start your program with the following code for main:

#Lab 9-4 Blood Drive

#the main function

def main():

endProgram = 'no'

print()

while endProgram == 'no':

print()

# declare variables

# function calls

endProgram = input('Do you want to end program? (Enter no or yes): ')

while not (endProgram == 'yes' or endProgram == 'no'):

print('Please enter a yes or no')

endProgram = input('Do you want to end program? (Enter no or yes): ')

#the getPints function

#the getTotal function

#the getAverage function

#the getHigh function

#the getLow function

#the displayInfo function

# calls main

main()

**Step 4:** Under the documentation for declaring variables, declare your variables and initialize them to 0. The array/list should be declared as follows:

pints = [0] \* 7

**Step 5:** Write a function call to the getPints function and pass it pints. The function will return pints, so set it equal to the function call. This should be done as follows:

pints = getPints(pints)

**Step 6:** Under the documentation for the getPints function, write a while or a for in loop that will allow the user to enter pints into the array. This function might be written as follows.

#the getPints function

def getPints(pints):

counter = 0

while counter < 7:

pints[counter] = int(input('Enter pints collected: '))

counter = counter + 1

return pints

**Step 7:** Write a function call to the getTotal function and pass it pints and totalPints. This function should be set to the totalPints variable since it will be returned from the function. The call might look as follows:

totalPints = getTotal(pints, totalPints)

**Step 8:** Under the documentation for the getTotal function, add the following statements:

* Initialize counter back to 0
* Add a while loop that runs 7 iterations and includes:
  + Accumulate totalPints by setting totalPints = totalPints + pints[counter]
  + Increment counter
* Return totalPints

**Step 9:** Write a function call to the getAverage function and pass it totalPints and averagePints. This function should be set to the averagePints variable since it will be returned from the function. The call might look as follows:

averagePints = getAverage(totalPints, averagePints)

**Step 10:** Under the documentation for the getAverage function, add the following statements:

* A statement that will calculate averagePints as totalPints / 7
* Return averagePints

**Step 11:** Write a function call to the getHigh function and pass it pints and highPints. This function should be set to the highPints variable since it will be returned from the function. The call might look as follows:

highPints = getHigh(pints, highPints)

**Step 12:** Under the documentation for the getHigh function, add the following statements:

* Initialize highPints to pints[0]
* Set counter to 1
* Write a while loop that runs 7 iterations and includes:
  + An if statement that checks to see if pints[counter] > highPints
    - If it is true, set highPints to pints[counter]
  + Increment counter by 1
* Return highPints
* Be careful to watch your indentation on this function.

**Step 13:** Write a function call to the getLow function and pass it pints and lowPints. This function should be set to the lowPints variable since it will be returned from the function. The call might look as follows:

lowPints = getLow(pints, lowPints)

**Step 14:** Under the documentation for the getLow function, add the following statements:

* Initialize lowPints to pints[0]
* Set counter to 1
* Write a while loop that runs 7 iterations and includes:
  + An if statement that checks to see if pints[counter] < lowPints
    - If it is true, set lowPints to pints[counter]
  + Increment counter by 1
* Return lowPints
* Be careful to watch your indentation on this function.

**Step 15:** Write a function call to the displayInfo function and pass it averagePints, highPints, and lowPints.

**Step 16:** Under the documentation for the displayInfo function, write the statements that will do the following:

* Display the average pints donated
* Display the highest number of pints donated
* Display the lowest number of pints donated

**Step 17:** Run your program and check against the following output. If there are errors, go back through the steps to troubleshoot.

Enter pints collected: 43

Enter pints collected: 25

Enter pints collected: 64

Enter pints collected: 35

Enter pints collected: 19

Enter pints collected: 37

Enter pints collected: 46

The average number of pints donated is 38.4285714286

The highest pints donated is 64

The lowest pints donated is 19

Do you want to end program? (Enter no or yes): yes

**Step** **18:** Execute your program so that it works and paste the final code below

#######################################################

# Name: David White

# Class: CIS-1400

# Assignment: Lab 9-4

# File: lab9-4.py

# Purpose: calculate avg number of pints of blood donated

#######################################################

print('\n\*\*\*David White\*\*\*\n') # Display author's name

# the main function

def main():

endProgram = 'no'

print()

while endProgram == 'no':

print()

# declare variables

pints = [0] \* 7

# function calls

pints = getPints(pints)

totalPints = getTotal(pints)

avgPints = getAvg(totalPints)

maxPints = getMax(pints)

minPints = getMin(pints)

printInfo(maxPints, avgPints, minPints)

# input validation

endProgram = input('Do you want to end program? (Enter no or yes): ')

while not (endProgram == 'yes' or endProgram == 'no'):

print('Please enter a yes or no')

endProgram = input('Do you want to end program? (Enter no or yes): ')

# the getPints function

def getPints(pints):

# reset counter

counter = 0

while counter < 7:

# get pints

pints[counter] = int(input("Enter pints collected: "))

# increment counter

counter = counter + 1

# end while

return pints

# get total number of pints

def getTotal(pints):

# reset counter

counter = 0

# initialize variable

totalPints = 0

# get total

while counter < 7:

totalPints = totalPints + pints[counter]

# increment counter

counter = counter + 1

# end while

return totalPints

# get average number of pints

def getAvg(totalPints):

# calculate average

avgPints = totalPints / 7

return avgPints

# get highest number of pints

def getMax(pints):

minPints = pints[0]

# reset counter

counter = 1

while counter < 7:

if pints[counter] > minPints:

minPints = pints[counter]

# end if

# increment counter

counter = counter + 1

# end while

return minPints

# get lowest number of pints

def getMin(pints):

minPints = pints[0]

# reset counter

counter = 1

while counter < 7:

if pints[counter] < minPints:

minPints = pints[counter]

# end if

# increment counter

counter = counter + 1

# end while

return minPints

# the displayInfo function

def printInfo(maxPints, avgPints, minPints):

print()

print("the average number of pints donated is", avgPints)

print("the highest pints donated is", maxPints)

print("the lowest pints donated is", minPints)

# calls main

main()