**Lab Experience Twelve**

**Objectives:**

1. Understanding arrays.
2. Understanding how arrays are passed to functions.

**Background**

Arrays in Computer Science are data structures used to hold large amounts of the same type of data instead of using multiple variables. An array represents consecutive memory locations referenced by the same name with an offset into memory which is represented by the subscript.

**Arrays**

An array in C++ is represented by a single identifier followed by a constant value used to allocate consecutive memory locations. Arrays can only contain the same data type, a mixture of data types is not allowed. Arrays have already been used to represent character data, i.e. strings.

To declare a variable as an array the following declaration is used:

**datatype identifier[sizeofArray]; // datatype is any valid C++ data type**

**// identifier is any valid C++ identifier**

**// sizeofArray-- amount of memory to allocate**

**Example:intfun[5]; // allocates 5 storage locations for fun all containing type int.**

The identifier **fun** contains the starting address of the array. To access each element of the array a subscript is used as an offset value from the starting address. The subscript has to be an ordinal type (int or char) in the range 0 to **sizeofArray**-1. C++ does not perform range checking of the subscript. If a subscript is used that is outside of the array, execution will continue or will terminate abnormally. It is the programmer’s responsibility to verify subscript usage. If char is used the ASCII value of the character is used to offset into the array.

Examples: Using the array **fun** defined above what is contained in **fun** after the code segment below executes?

Contents of fun after the code have executed.

0 1 2 3 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 6 | 7 | 8 | 9 |

**fun[0] = 5;**

**fun[1] = 6;**

**fun[2] = 7;**

fun

**fun[3] = 8;**

**fun[4] = 9;**

What if fun is defined as a 100 storage locations? The method demonstrated above for assigning values to the array fun is tedious and time consuming. Fortunately, C++ allows the usage of ordinal variables to be used as subscripts.

The above code can be re-written as:

Contents of fun after the code have executed.

0 1 2 3 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 6 | 7 | 8 | 9 |

**for(inti = 0; i< 5 ; i++)**

**fun[i] = i +5;**

The same code segment can be used to output the array as follows:

**for(inti = 0; i< 5 ; i++)**

**cout<< “fun[“ <<i<< “] = “ << fun[i] <<endl;**

The output will be:

fun[0] = 5

fun[1] = 6

etc. Note since fun is not a character array the subscript is necessary to access the contents of each memory location individually. With character arrays this is not necessary because the null character indicates the end of the string, whereas with all other arrays this information is not available.

This does not prevent the programmer from accessing each element of a character array. To access individual characters of the array, just use a subscript.

Since arrays represent storage locations and can contain numeric values, we can perform operations with an array as long as the subscript is used to reference specific elements of the array. The code shown below will sum up all elements of the array.

**sum = 0;**

**for(inti = 0; i< 5 ; i++)**

**sum += fun[i];**

Comparison between two storage locations in the array can also be used. For example:

**if(fun[3] > fun[4])**

**cout<< “The value in location 3 is larger than the value contained in location 4”**

**<<endl;**

**else**

**cout<< “The value in location 4 is larger than the value contained in location 3”**

**<<endl;**

Arrays can be initialized during declaration, but the number of values used to initialize the array must be less than or equal to size of the array. If the number of values is less than the size, the rest of the array contains zeros.

Example: **intfun[5] = {5, 6, 7, 8, 9}; // initialize the array.**

The values are stored from left to right in the array.

**Lab Exercises**

**Directions:**

Start Microsoft word and record the questions and answers to all of the exercises in the word document   
Answer the following questions based on material presented in lecture and found in chapters 1-7 of the textbook.

**Exercise 1**

Download the program arrayExample1 from D2L. The file is listed below with the line numbers being used as a reference.

//

// An array manipulation example

// Programmer: your name here

// Subject: CSCI 1106

//

1. #include <iostream>

2. #include <iomanip>

using namespace std;

3. constint MAXELEMENTS = 8;

4. typedefintintegerArray[MAXELEMENTS]; // create an alias for a one dimensional

// array containing integers

5. int main(){

6. integerArray prime = {2, 3, 5, 7, 11, 13, 17, 19};

7. for(inti = 0; i< MAXELEMENTS ; i++){

8. cout<<setw(3) << "prime[" <<i<<"] = " << prime[i] <<endl;

}

9. return 0;

}

Statement #4 creates an alias for the array name. The statement is called a **typedef statement** and it defines a synonym for the specified type-declaration. The identifier in the type-declaration becomes another name for the type, instead of naming an instance of the type. **You cannot use the typedefspecifier inside a function definition.**

1. Statement #6 initializes the array elements to whatever is declared in the list. Execute the program to see what the output is.
2. It is an error to give too many values in an initializer list of an array. Increase the number of initializers in statement #6 to determine if our compiler catches the error. Describe what happens.

**Intellesense and the compiler errors saying that there are too many initializer values in the list.**

1. Change the number of initializers in statement #6 to only 4 items and describe what happens when you run your program.

**The 4 items in the list will be outputted, and the last 4 remaining outputs are zero, since the array is filled with 0's when there are less items in initializer list than the total size of the array.**

1. When completed close the solution.

**Exercise 2: Common errors involved in passing arrays to functions.**

Download the following program, lab11Array from D2L and load it into your compiler.

// This program demonstrates an array being passed to a function.

#include<iostream>

usingnamespacestd;

voidshowValues(int [], int); // Function prototype

constint ARRAY\_SIZE = 100;

int main()

{

int numbers[ARRAY\_SIZE] = {5, 10, 15, 20, 25, 30, 35, 40};

showValues(numbers, 8);

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Definition of function showValue. \*

// This function accepts an array of integers and \*

// the array's size as its arguments. The contents \*

// of the array are displayed. \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

voidshowValues(intnums[], int size)

{

for (int index = 0; index < size; index++)

cout<<nums[index] <<" ";

cout<<endl;

}

Perform the following operations and record your results (error messages, etc.) in your word document. **After you have recorded your results from the error message restore the program back to its original code.**

1. Change voidshowValues(int [], int); to voidshowValues(int , int); and execute your program.

**An error occurs saying that parameter1 cannot be converted from int[100] to int.**

1. Change voidshowValues(intnums[], int size) to voidshowValues(int nums, int size) and execute your program.

**An error occurs saying that the expression must have a pointer to object type.**

1. Change for (int index = 0; index < size; index++) to

for (int index = 0; index < size; index++) and execute your program.

**All the initialized values in the list were shown as well as a 0 at the end, since the array is filled with 0's after the initialization list.**

1. Change showValues(numbers, 8); to showValues(numbers[], 8); and execute your program.

**There was error that there was an expression expected.**

1. Change showValues(numbers, 8); to showValues(numbers[0], 8); and execute your program.

**There was an error saying that shows that the parameter 1 cannot be converted from int to int[]**

1. Change voidshowValues(int [], int); to voidshowValues(int&, int); andchange

voidshowValues(intnums[], int size) to voidshowValues(int&nums, int size) and execute your program.

**There was an error saying that showValues cannot convert parameter 1 from int[100] to int &**

1. Change showValues(numbers, 8); to showValues(numbers, ARRAY\_SIZE); and execute your program.

**The output was the initializer list along with 92 zeroes following the list, since the array is filled with zeroes after the initializer list.**

**Exercise 3:**

Do problem 9 on pages 450-451.

Copy and paste your program below along with screen shots of the execution of your program.

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**//**

**// Programmer: Chris Dang Class: CSCI 1106 Fall 2014**

**//**

**// Description: This program displays an employees ID number and will then ask**

**// for the number of hours worked, followed by the pay rate for the employee.**

**// The program will then calculate the gross wage, and then display all of the**

**// employee's ID's along with their wage.**

**//**

**//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**#include <iostream>**

**#include <iomanip> // for formatting**

**#include <string>**

**using namespace std;**

**const int NUM\_EMPLOYEES = 7 ;**

**const double MIN\_WAGE = 15.00 ; //minumum wage is $15**

**//Employee ID numbers:**

**const long emp1 = 5658845, emp2 = 4520125, emp3 = 7895122, emp4 = 8777541,**

**emp5 = 8451277, emp6 = 1302850, emp7 = 7580489 ;**

**const string ENTER\_HOURS =**

**"Please enter the number of hours worked for employee #" ;**

**const string ENTER\_PAY =**

**"Please enter the pay rate for employee #" ;**

**int main () {**

**long empId[NUM\_EMPLOYEES] = {emp1, emp2, emp3, emp4, emp5, emp6, emp7} ;**

**double hours[NUM\_EMPLOYEES] ;**

**double payRate[NUM\_EMPLOYEES] ;**

**double wages[NUM\_EMPLOYEES] ;**

**// Loops the get input cycles**

**for (int i = 0; i < NUM\_EMPLOYEES; i++) {**

**cout << ENTER\_HOURS << empId[i] << ": ";**

**cin >> hours[i] ;**

**// input validating while. Traps user until non-negative number is entered**

**while (hours[i] < 0) {**

**cout << "Hours cannot be less than 0.\n" << ENTER\_HOURS << empId[i]**

**<< ": " ;**

**cin >> hours[i] ;**

**} // end while**

**cout << ENTER\_PAY << empId[i] << ": ";**

**cin >> payRate[i] ;**

**// input validating while. Traps user until wage is >= $15**

**while (payRate[i] < MIN\_WAGE) {**

**cout << "Pay rate cannot be less than 15.\n" << ENTER\_PAY << empId[i]**

**<< ": " ;**

**cin >> payRate[i] ;**

**} // end while**

**//calculates gross wage**

**wages[i] = hours[i] \* payRate[i] ;**

**} // end for**

**//displays employee number and gross wage**

**cout << setprecision(2) << fixed << showpoint << "Gross pay for each employee is:\n";**

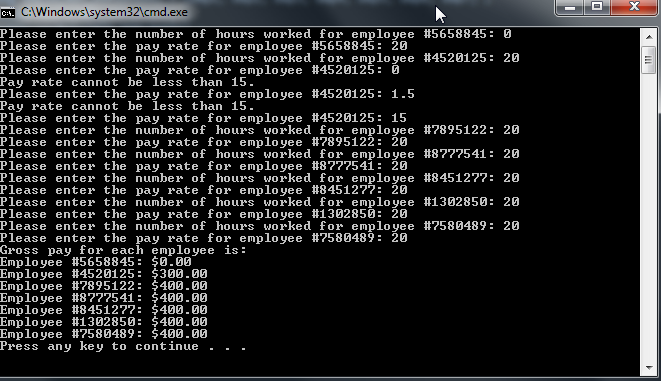
**for (int i = 0; i < NUM\_EMPLOYEES; i++) {**

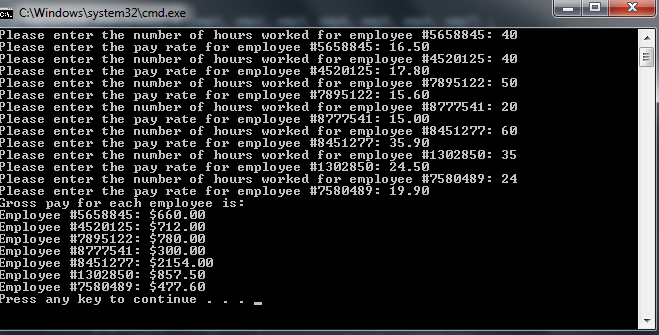
**cout << "Employee #" << empId[i] << ": $" << wages[i] << endl ;**

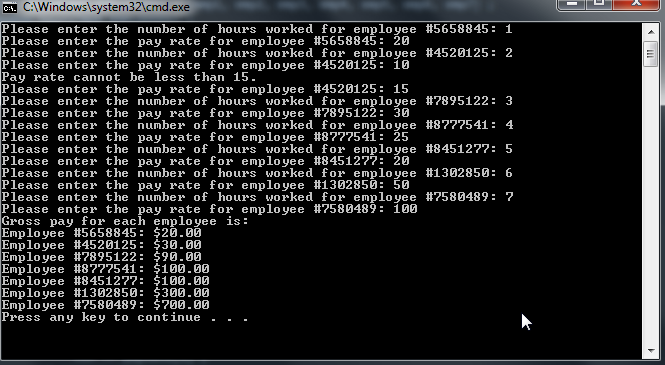
**} // end for**

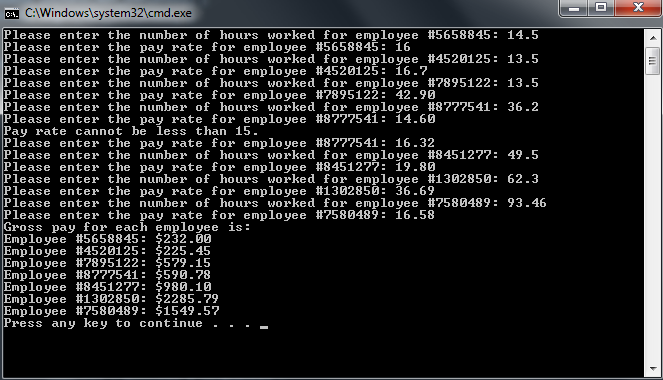
**return 0;**

**}**









**Due Date:**

As indicated on the Lab Twelve Dropbox Folder.

**What to hand in:**

1. Save your word document as yournamelab11.docx, i.e. timwrennlab12.docx.
2. Compress your program from .cpp file and your word file into a single compressed folder called yournamelab12.zip.
3. Hand in printouts of your program from exercise 3 and your word document.
4. Place the compressed file into the lab 12 dropbox folder.