**Andrew Tran**

**Lab Experience Thirteen**

**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:** **int fun[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(int i = 0; i < 5 ; i++)**

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

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

**for(int i = 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(int i = 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: **int fun[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. const int MAXELEMENTS = 8;

4. typedef int integerArray[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(int i = 0; i < MAXELEMENTS ; i++){

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

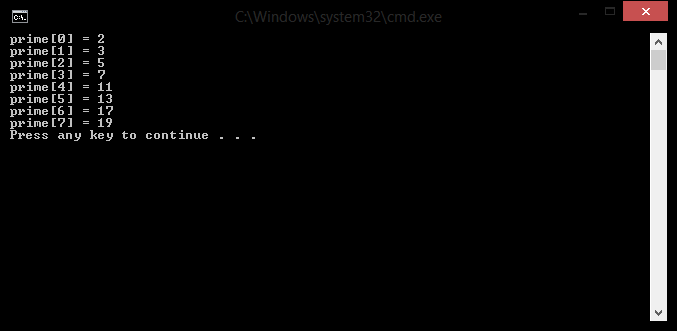
}

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 typedef specifier 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.



1. 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.

It does give a “too many initializers” error.

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

It fills in the remaining 4 slots with zero.

1. When completed close the solution.

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

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

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

#include <iostream>

using namespace std;

void showValues(int [], int); // Function prototype

const int 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. \*

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

void showValues(int nums[], 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 void showValues(int [], int); to void showValues(int , int); and execute your program.

I get an error “cannot convert argument 1 from 'int [100]' to 'int'”

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

I get an error “subscript requires array or pointer type”

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

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

It added a zero at the end of the array.

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

I get a syntax error.

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

I get an error “cannot convert argument 1 from 'int' to 'int []'”

1. Change void showValues(int [], int); to void showValues(int &, int); and change

void showValues(int nums[], int size) to void showValues(int &nums, int size) and execute your program.

I get an error saying that “cannot convert argument 1 from 'int [100]' to 'int &'”

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



**Exercise 3:**

Do problem 9 on pages 450-451. Functions are not required for this program.

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

#include<iostream>

#include<iomanip>

using namespace std;

int main()

{

int i;

int empId[7] = { 5658845,4520125,7895122,8777541,8451277,1302850,7580489 };

double rate, hour, wages[7];

for (i = 0; i <= 6; i++)

{

cout << "Employee number: " << empId[i] << endl;

cout << "\nEnter employee's hours: ";

cin >> hour;

while (hour < 0)

{

cout << "Invalid Input. Hours cannot be negative." << endl;

cout << "Enter employee's hours: ";

cin >> hour;

}

cout << "Enter employee's pay rate: ";

cin >> rate;

while (rate < 15)

{

cout << "Invalid Input. Employee's pay rate must be higher than $15." << endl;

cout << "Enter employee's pay rate: ";

cin >> rate;

}

wages[i] = hour \* rate;

cout << "\n";

}

for (i = 0; i <= 6; i++)

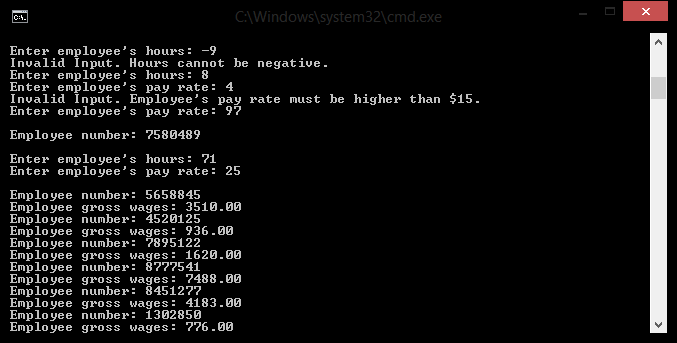
{

cout << "Employee number: " << empId[i] << endl;

cout << "Employee gross wages: " << fixed << setprecision(2) << wages[i] << endl;

}

}



**Due Date:**

As indicated on the Lab Thirteen Dropbox Folder.

**What to hand in:**

1. Save your word document as yournamelab11.docx, i.e. timwrennlab13.docx.
2. Compress your program from .cpp file and your word file into a single compressed folder called yournamelab13.zip.
3. Hand in printouts of your program from exercise 3 and your word document.
4. Place the compressed file into the lab 13 drop box folder.