**Chapter 8 Lab**

**Pointers**

# Lab Objectives

* Be able to create, use, and delete dynamic arrays
* Be able to diagram the memory in the computer as statements are executed
* Make an array of objects

**Introduction**

Static arrays can only be created if you know the size (as a constant) at the time of compiling so that the compiler knows how much memory to assign it on the stack. Memory in the stack is automatically reclaimed by the system when the program ends.

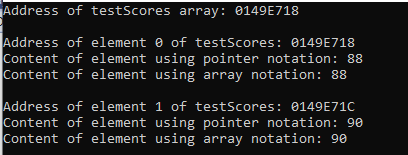
If you do not know the size of the array at compile time, you must use a dynamic array that will use the memory in the heap. This means that the memory will not be automatically reclaimed when the program ends, which can lead to memory leaks. Dynamic memory allocation from the heap is accomplished with the keyword “new”. Every “new” must be reclaimed with a “delete” before the program ends or your system will think that memory is still being used and cannot reallocate it.

# Task #1 Arrays as pointer parameters

1. Create a project in Visual Studio called Lab 8. Add the file ArraysAsParameters.cpp to the source files folder.
2. Compile and run. You should get the output **0 5 10 15 20.**
3. Write the function definition for the DoubleValue function following the prototype. It should double the value stored in each element of the array.
4. Write a function call, passing in the array and the size.
5. Print out the contents of the array as previously done. Your output should now have a second line, **0 10 20 30 40**.
6. Write the function definition for the TripleValue function following the prototype. It should triple the value stored in each element of the array. Be sure to use **pointer notation** within this function, NOT array notation with square braces.
7. Write a function call, passing in the array pointer and the size.
8. Print out the contents of the array as previously done. Your output should now have a third line, **0 30 60 90 120**.
9. Right click on the folder below the solution and rename it as PointerParameters.

# Task #2 Working with dynamic arrays of numbers

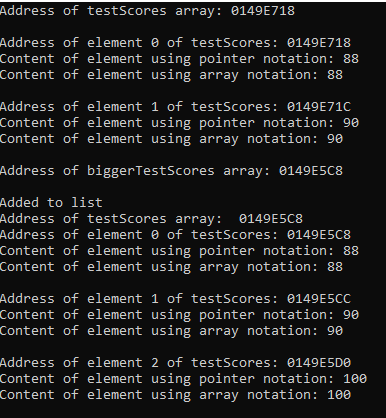
1. Add a project to the solution by right clicking on the solution folder and adding a new project called NumberDynamicArrays. Add the file DynamicArray.cpp to the source files folder.
2. Notice the PointerParameters folder is bold. This means that it is the active project and will run when you compile and run. Right click on the NumberDynamicArrays folder and set as the start up project (it will turn bold).
3. Compile and run. Your output should be similar, but you will have different addresses.



1. Notice these are in hexadecimal (base 16). Hexadecimal uses the characters {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f}. Take the difference between the memory address of element 1 and element 2 (you can just count along this list of characters to see how many bytes are saved for one element). **How many bytes are used for each element and why?**

**4 bytes because integers use only 4 bytes**

1. Declare a new integer pointer and set to nullptr. Create a new integer array that is one longer than the original array.
2. Write a loop to copy the contents of the original array to the new array.
3. Add the value 100 to the array in the extra spot at the end.
4. Reclaim the memory from the original array.
5. Increase the arraySize by one.
6. Reassign the larger array address to the original pointer variable.
7. Remove the address from the pointer that is no longer used.
8. Uncomment the lines used to print out the array information after we have added more information to the original array.
9. Compile, debug, and run. You should get output that is similar, but again will have different addresses.



1. **Why is the address of the testScores array different after we added to the list?**

**The address is different because the location that the new array was created is in a different spot**

1. Uncomment the AddToListBad call.
2. Notice these are the same steps that we used in the main to successfully create add to the list. Compile, debug and run. Why did this not work? Let’s figure it out…
   1. **Draw the memory for lines 13-42.**
   2. **Draw an X in the diagram above to show what is deleted in line 44.**
   3. Draw the memory after lines 48-50 are executed.
   4. Remembering that parameter variables are local variables and are passed copies of the values (unless passed by reference using &), draw the memory for the parameter values for the function AddToListBad and pass them the values from the function call.
   5. Follow the function statements and **show the modification in the function variables in the above diagram**. **Why did the statements**

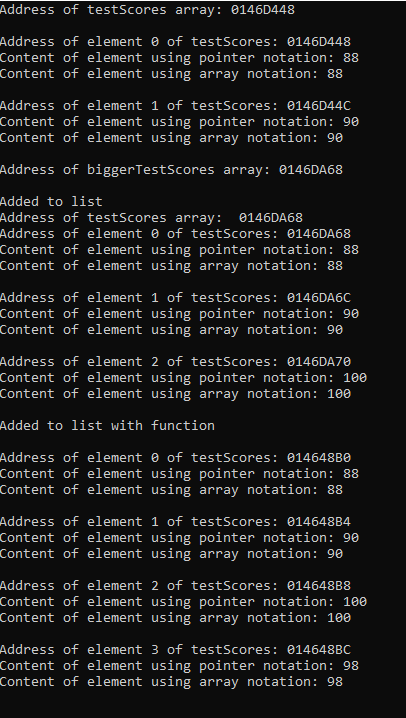
**delete[] originalArray;**

**originalArray = biggerArray;**

**not work the same as in the main?**

**They don’t work because originalArray is pointing to a pointer which causes originalArray to stop pointing to testScores and point to biggerArray which causes testScores not to change**

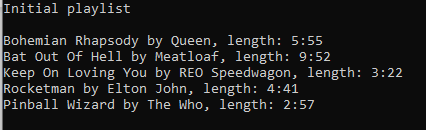
1. Write the function definition for the AddToList function following the prototype.
   1. Declare a new integer pointer and set to nullptr. Create a new integer array that is one longer than the original array.
   2. Write a loop to copy the contents of the original array to the new array.
   3. Add the value to the array in the extra spot at the end.
   4. Reclaim the memory from the original array.
   5. Increase the arraySize by one.
   6. Return the pointer to the new, bigger array.
2. Comment out the function call to AddToListBad. Uncomment the function call to AddToList.
3. Compile, debug and run. You should get similar output.



Notice that the address of the array again changed with testScores being reassigned the value of the pointer returned by the AddToList function.

# Task #3 Working with dynamic arrays of objects

1. Add another project to the current solution as in task #2. Add the files CD.cpp and Song.cpp to the source files folder, the Song.h file to the header files folder, and the cdTracks.txt file to the resources files folder.
2. Declare two Song pointers called playList1Ptr and playList2Ptr, initializing each to nullptr.
3. Create a dynamic array of Songs of size numTracks (after it is read in from the file) and assign it to the first playlist pointer.
4. After successfully reading the information from the file, construct a song from the information read and store it into the first playlist array.
5. Write a loop to display the songs in the first playlist and calling the Display member function for each song in the array.
6. At the bottom of the main, but before the return 0, reclaim the dynamic memory by deleting the dynamic array.
7. Set the pointer to nullptr.
8. Compile, run, and debug. The output should appear as:



## Task #4 Increasing the size of an array

1. Delete the 2 lines of stars. This should uncomment it all at once.
2. Create a dynamic array of Songs for the second playlist that is one longer than the first playlist and assign it to the second playlist pointer.
3. Copy the songs from the first playlist to the second playlist.
4. After you read in the information for the user’s song, construct a song from the information and put it in the second playlist (the last spot should be empty).
5. Write a loop to display the songs in the second playlist and calling the Display member function for each song in the array.
6. As done with the first playlist, reclaim the dynamic memory by deleting the dynamic array and set the pointer to nullptr.
7. Compile, run, and debug. You should get output that is similar to this (with the information that you enter for the last song):

