|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CS 1400 Lab #23**  **Bubble Sort**  **Version 1.0**  **Objectives:**  The objective of this lab is to build your understanding of sorting arrays.  After completing this lab, you should be able to   1. Explain how a bubble sort works. Be able to describe the performance issues with a bubble sort. Correctly write the code to do a bubble sort.   **Study Material**   |  | | --- | | **Designing a Bubble Sort**  **Swapping Array Elements**  A bubble sort requires that we start by comparing the first two elements of an array, and then put those two elements in the correct order with respect to one another; then moving to the next pair and so on. So, one of the key methods that we will need to write for the bubble sort, is a **swap( )** method. You have seen the code to do this before. We will need to pass the elements to be swapped by ref (reference). Let's assume, for this exercise, that we are sorting an array of integers. The Swap method would look like  **void Swap(ref int a, ref int b) {      int temp = a;      a = b;      b = temp; }**  **Sorting**  blocksConsider the blocks shown in the figure to the left. Our objective in this example is to put the blocks in order from the smallest to the largest. Let's begin by comparing the first two blocks. Clearly the first block is larger than the second one, so we need to swap the position of these two blocks. The code to do this might look something like   **if ( block [1] > block [2])      swap (ref block [1], ref block [2]);**    blocks  This looks right. Now lets move on and compare the second block with the third. The second is bigger than the first, so we also swap these two. This code will look like   **if ( block [2] > block [3])      swap (ref block [2], ref block [3]);**    blocksNote that after this swap the large black has moved one more place to the right. This is what we want. Now lets move on and compare the third block with the last one. The third is bigger than the fourth, so we also swap these two. This code will look like   **if ( block [3] > block [4])      swap (ref block [3], ref block [4]);**    blocksAt this point, we have been through the entire array one time. After this last swap, the largest block has moved all of the way to the right side. It has moved one position for each comparison that we made. This "bubbling" the small elements up and the largest element to where it belongs, is a key characteristic of a bubble sort. We can express all of the above steps in a for loop of the form   **for (int i = 0; i < 3; i++) {      if ( block [i] > block [i+1])           swap (ref block [i], ref block [i + 1]); }**  Notice the index in the loop. We stop when *i* is less than 3, not 4, as you might expect. This is because once we compare the next to last element of the array (in this case element 3) with the last one, we are done.  blocks  We have now moved through the array one time. However, as you can see by looking at the picture, the array is not sorted. What this means is that we have to go through the array a second time. The first comparison will not result in a swap, since the first element (3) is smaller than the second (4).  However, when we compare the second (4) with the third (1), the second (3) is larger than the third (1) and we will swap them. Finally, the third (1) is smaller than the last (5) , and no swap occurs. The result is  blocks   The array is still not sorted, but it is close. The last two blocks are just where they belong. So, we will have to go through the array yet again. To be sure that every element gets swapped, a bubble sort therefore contains two loops, an outer loop and inner loop, that look like  **for (int j = 0; j < 3; j++) {      for (int i = 0; i < 3; i++)      {           if ( block [i] > block [i+1])                swap (ref block [i], ref block [i + 1]);      } }** |  1. [**Animation of a Bubble Sort**](http://debryro.tc.uvu.edu/1400/labs/lab23/bubbleSort.swf) (click on the green button to start the animation or see Dr. deBry’s website)   **Programming Exercise**  You should have reviewed the material on designing a bubble sort before starting this assignment. Create a new project in Visual Studio for C#. Use the program below [bsort.cs](http://debryro.tc.uvu.edu/1400/labs/lab23/bsort.cs) and add it to the project.  // This is a Bubble Sort Routine  // CS 1400 Lab #19  // Author: Roger deBry  // Date: March 2014  // -----------------------------------  using System;  static class Program  {  static void Main()  {  // this is the date to be sorted  int[ ] theData = {45, 12, 23, 34};  // sort the array in ascending order  // print out lots of messages so we can see the sort work  for (int j = 0; j < theData.Length-1; j++) // index for outer loop is j  {  Console.WriteLine("Iteration {0} for the outer loop", j );  for (int i = 0; i < theData.Length-1; i++) // index for inner loop is i  {  Console.WriteLine("\nIteration {0} for the inner loop", i  if (theData[ i ] > theData[ i + 1] )  Swap( ref theData[ i ], ref theData[ i + 1] );  }  }    // print out the sorted array  Console.WriteLine("\n\n\*\*\*\*\* The sorted array is: \*\*\*\*\*");  for (int i = 0; i < theData.Length; i++)  {  Console.Write("{0} ", theData[ i ]);  }  Console.WriteLine( );  Console.ReadLine( );  }  // method prototype for the swap routine  // parameters: two integers, passed by reference  // routines: none  // The two integer values are swapped  // ---------------------------------------------------  static void Swap (ref int a, ref int b)  {  Console.WriteLine("Swapping {0} and {1}", a, b);  int temp = a;  a = b;  b = temp;  }  }  **Program Specification**  The program, as written, sorts an array of integers in ascending order (the smallest number is first). For this exercise, all you need to do is to change the program so that it sorts the array in descending order as well (the biggest number is first).  **File(s) to Submit:**  Place your complete project folder in a zip file and name the zip file Lab\_23\_your-initials\_V1.0.zip. For example, I would name my file Lab\_23\_DAF\_V1.0.zip. Submit this assignment as Lab #23 on Canvas.   |  |  |  | | --- | --- | --- | |  | **Grading Checklist** |  | | # | Program | C(correct)  X(incorrect) | | 1 | Meets & works to specifications | 6 points | | 2 | Error Free, elegant & efficient | 4 points | | 3 | Pseudo-Code | -3 points | | 4 | Style Guidelines | -2 points | | 6 | Source Files(s) & Formatting | -2 points | | 7 | Project Prolog | -1 points | | 8 | Function Prologs | -1 points | | 9 | Zip Filename | -1 points | | 10 | Lab & Project Names | -1 points | | 11 | Zip File is invalid or will not unzip | Lab = 0 pts | |  | Total Points | 10 | 0-9 | |