# Objectives: Lab 02

The purpose of the first lab of BTP500 is to familiarize yourself with the Visual Studio User Interface and write a simple C++ program with functions. By the end of this lab, you should be able to:

* Successfully ran 3 algorithms to learn about different sorting methods using C++ functions.

# LAB 02 – SUBMISSION

# **4 files.**

# **1.   Word document BTP500-LAB2-NAME.DOCX filled with answers and screenshots**

# **2.   BTP500-LAB2-STUDENTNAME-InsertSort.cpp**

# **3.   BTP500-LAB2-STUDENTNAME-BubbleSort.cpp**

# **4.   BTP500-LAB2-STUDENTNAME-QuickSort.cpp**

Explore the Sorting of data in C++.

This is the main function code.

A computer screen shot of a computer program

Description automatically generated

1. Write a C++ recursive or non-recursive function to sort an unsorted array using Bubble Sort algorithm and submit the .cpp file separately. Paste the screenshot of C++ here below and give an example of bubble sort here solving step by step with 7 un-sorted nodes. Take any values of unsorted nodes.

A screenshot of a computer

Description automatically generated

**Example of bubble sort:**

Example unsorted array: 3, 4, 7, 2, 5, 1, 6

I = n-1 = 6, J = 0

First iteration:

I = 6, j = 0; 3 < 4, nothing happens

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 7 | 2 | 5 | 1 | 6 |

I = 6, j = 1; 4 < 7 nothing happens

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 7 | 2 | 5 | 1 | 6 |

I = 6, j = 2; 7 > 2, swap

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 7 | 5 | 1 | 6 |

I = 6, j = 3; 7 > 5, swap

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 5 | 7 | 1 | 6 |

I = 6, j = 4; 7 > 1, swap

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 5 | 1 | 7 | 6 |

I = 6, j = 5; 7 > 6, swap

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 5 | 1 | 6 | 7 |

I--, j = 0

Second iteration:

I = 5, j = 0; 3 < 4, nothing happens

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 5 | 1 | 6 | 7 |

I = 5, j = 1; 4 > 2, swap

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 2 | 4 | 5 | 1 | 6 | 7 |

I = 5, j = 2; 4 < 5, nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 2 | 4 | 5 | 1 | 6 | 7 |

I = 5, j = 3; 5 > 1, swap

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 2 | 4 | 1 | 5 | 6 | 7 |

I = 5, j = 4; 5 < 6, nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 2 | 4 | 1 | 5 | 6 | 7 |

I--, j = 0

Third iteration:

I = 4, j = 0; 3 > 2,swap

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 2 | 4 | 1 | 5 | 6 | 7 |

I = 4, j = 1; 2 < 4, nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 4 | 1 | 5 | 6 | 7 |

I = 4, j = 2; 4 > 1, swap

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 1 | 4 | 5 | 6 | 7 |

I = 4, j = 3; 4 < 5, nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 1 | 4 | 5 | 6 | 7 |

I--, j = 0

Fourth iteration:

I = 3, j = 0; 2 < 3, nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 1 | 4 | 5 | 6 | 7 |

I = 3, j = 1; 3 > 1, swap

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 1 | 3 | 4 | 5 | 6 | 7 |

I = 3, j = 2; 3 < 4, nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 1 | 3 | 4 | 5 | 6 | 7 |

I--, j = 0

Fifth iteration:

I = 2, j = 0; 2 > 1, swap

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 1 | 3 | 4 | 5 | 6 | 7 |

I = 2, j = 1; 2 < 3, nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

I--, j = 0

Sixth iteration:

I = 1, j = 0; 1 < 2, nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

1. Write a C++ recursive or non-recursive function to sort an unsorted array using Insertion algorithm and submit the .cpp file separately. Paste the screenshot of C++ here below and give an example of Insertion sort here solving step by step with 7 un-sorted nodes. Take any values of unsorted nodes.

A screenshot of a computer

Description automatically generated

Example of insert sort:

Example unsorted array: 3, 4, 7, 2, 5, 1, 6

I = 1

J = I

First iteration:

I = 1, j = 1; 4 > 3, nothing happens

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 7 | 2 | 5 | 1 | 6 |

I++, j = i

Second iteration:

I = 2, j = 2; 7 > 4, nothing happens

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 7 | 2 | 5 | 1 | 6 |

I = 2, j = 1; 4 > 3, nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 7 | 2 | 5 | 1 | 6 |

I++, j = i

Third iteration:

I = 3, j = 3; 2 < 7, swap

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 7 | 5 | 1 | 6 |

I = 3, j = 2; 2 < 4, swap

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 2 | 4 | 7 | 5 | 1 | 6 |

I = 3, j = 1; 2 < 3, swap

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 4 | 7 | 5 | 1 | 6 |

I++, j = i

Fourth iteration:

I = 4, j = 4; 5 < 7 and 5 > 4, swap

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 4 | 5 | 7 | 1 | 6 |

I = 4, j = 3; 5 > 4, nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 4 | 5 | 7 | 1 | 6 |

I = 4, j = 2; 4 > 3, nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 4 | 5 | 7 | 1 | 6 |

I = 4, j = 1; 3 > 2, nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 4 | 5 | 7 | 1 | 6 |

I++, j = i

Fifth iteration:

I = 5, j = 5; 1 < 7, swap

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 4 | 5 | 1 | 7 | 6 |

I = 5, j = 4; 1 < 5, swap

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 4 | 1 | 5 | 7 | 6 |

I = 5, j = 3; 1 < 4, swap

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 1 | 4 | 5 | 7 | 6 |

I = 5, j = 2; 1 < 3, swap

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 1 | 3 | 4 | 5 | 7 | 6 |

I = 5, j = 1; 1 < 2, swap

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 7 | 6 |

I++, j = i

Sixth iteration:

I = 6, j = 6; 6 < 7 and 6 > 5, swap

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

I = 6, j = 5; 6 > 5, nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

I = 6, j = 4; 5 > 4, nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

I = 6, j = 3; 4 > 3, nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

I = 6, j = 2; 3 > 2, nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

I = 6, j = 1; 2 > 1, nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

1. Write a C++ recursive or non-recursive function to sort an unsorted array using Quick Sort algorithm and submit the .cpp file separately. Paste the screenshot of C++ here below and give an example of Quick sort here solving step by step with 7 un-sorted nodes. Take any values of unsorted nodes.

A screenshot of a computer

Description automatically generated

Example of quick sort:

Example unsorted array: 3, 4, 7, 2, 5, 1, 6

quickSort(array, 7);

Start = 0, n = 7

pivotIndex = start, I = start

pivot is end-1 (end of the array), in this case 6.

pivotIndex = 0, I = 0; 3 < 6, swap current with pivot index, pivotIndex++

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 7 | 2 | 5 | 1 | 6 |

pivotIndex = 1, I = 1; 4 < 6, swap current with pivot index, pivotIndex++

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 7 | 2 | 5 | 1 | 6 |

pivotIndex = 2, I = 2; 7 > 6, do nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 7 | 2 | 5 | 1 | 6 |

pivotIndex = 2, I = 3; 2 < 6, swap current with pivot index, pivotIndex++

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 7 | 5 | 1 | 6 |

pivotIndex = 3, I = 4; 5 < 6, swap current with pivot index, pivotIndex++

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 5 | 7 | 1 | 6 |

pivotIndex = 4, I = 5; 1 < 6, swap current with pivot index, pivotIndex++

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 5 | 1 | 7 | 6 |

pivotIndex = 5

swap last element with pivotIndex

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 5 | 1 | 6 | 7 |

quickSort(array, pivotIndex(5), start(0));

quickSort(array, n(7), pivotIndex+1(6)); (start >= n-1) do nothing, return to main caller;

quickSort(array, 5, 0);

Start = 0, n = 5

pivotIndex = start, I = start

pivot is end-1 (end of the array) in this case, 1

pivotIndex = 0, I = 0; 3 > 1, do nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 5 | 1 | 6 | 7 |

pivotIndex = 0, I = 1; 4 > 1, do nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 5 | 1 | 6 | 7 |

pivotIndex = 0, I = 2; 2 > 1, do nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 5 | 1 | 6 | 7 |

pivotIndex = 0, I = 3; 5 > 1, do nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 5 | 1 | 6 | 7 |

pivotIndex = 0

swap last element with pivotIndex

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 4 | 2 | 5 | 3 | 6 | 7 |

quickSort(array, pivotIndex(0), start(0)); (start >= n-1) do nothing, return to main caller.

quickSort(array, n(5), pivotIndex+1(1));

quickSort(array, 5, 1);

Start = 1, n = 5

pivotIndex = start, I = start

pivot is end-1 (end of the subarray) in this case, 3

pivotIndex = 1, I = 1; 4 > 3, do nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 4 | 2 | 5 | 3 | 6 | 7 |

pivotIndex = 1, I = 2; 2 < 3, swap current with pivot index, pivotIndex++

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 4 | 5 | 3 | 6 | 7 |

pivotIndex = 2, I = 3; 5 > 3, do nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 4 | 5 | 3 | 6 | 7 |

pivotIndex = 2

swap last element with pivotIndex

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 5 | 4 | 6 | 7 |

quickSort(array, pivotIndex(2), start(1)); (start >= n-1) do nothing, return to main caller.

quickSort(array, n(5), pivotIndex+1(3));

quickSort(array, 5, 3);

Start = 3, n = 5

pivotIndex = start, I = start

pivot is n-1 (end of the subarray) in this case, 5

pivotIndex = 3, I = 3; 5 > 4, do nothing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 5 | 4 | 6 | 7 |

pivotIndex = 3

swap pivot with pivotIndex

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Do not submit a .zip or RAR file.