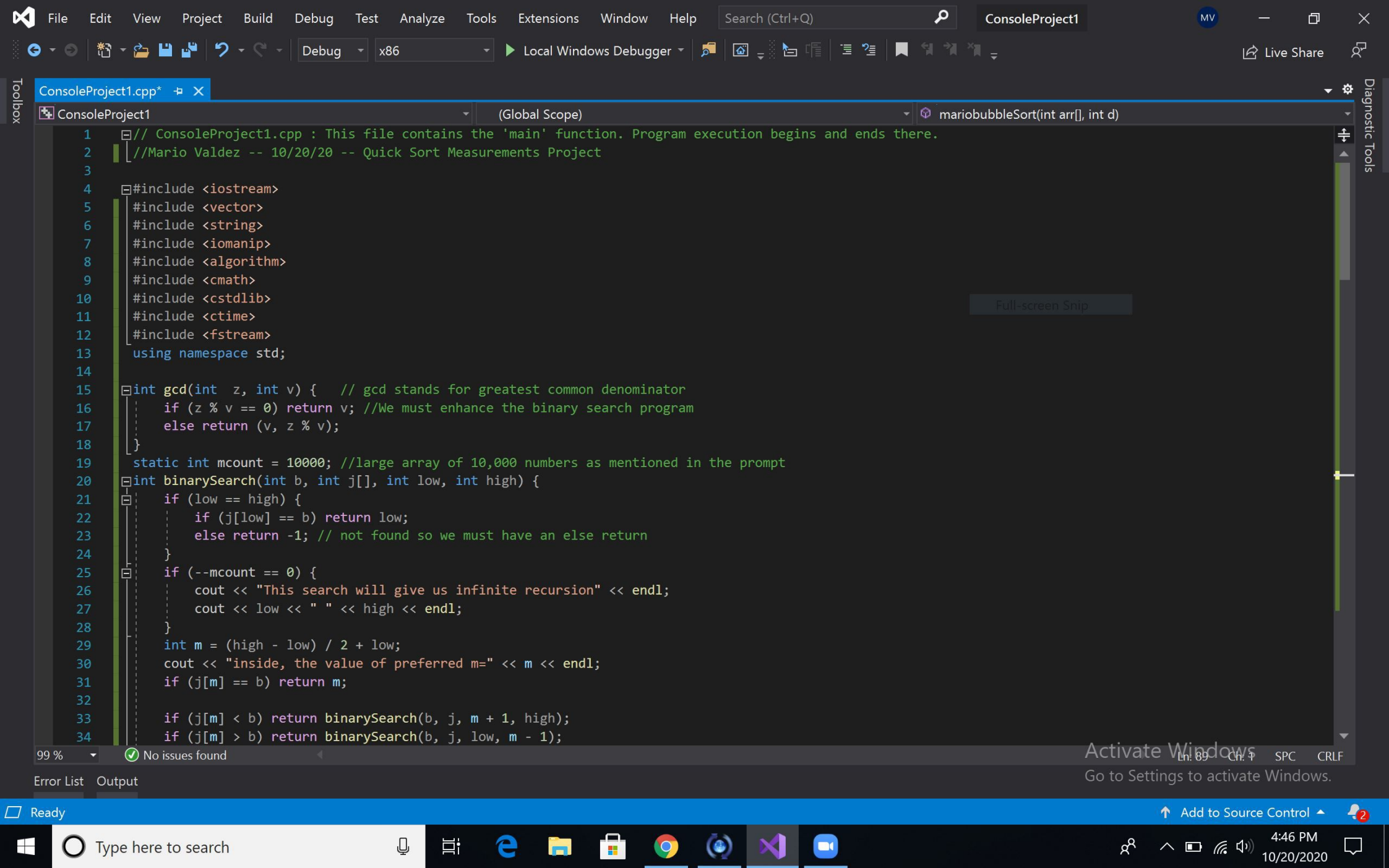
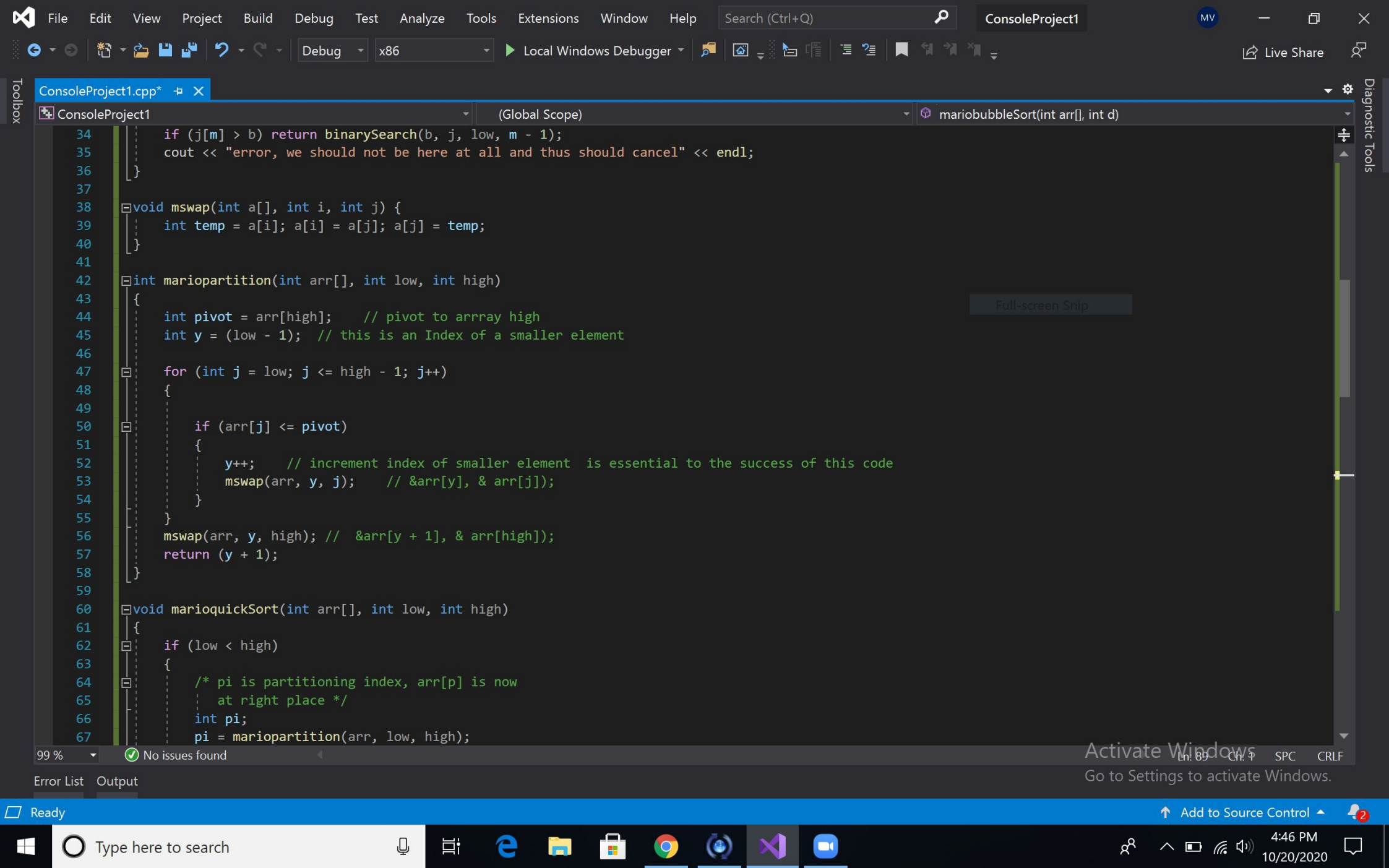
Mario Valdez
15th October 2020

QuickSort and BubbleSort Measurements

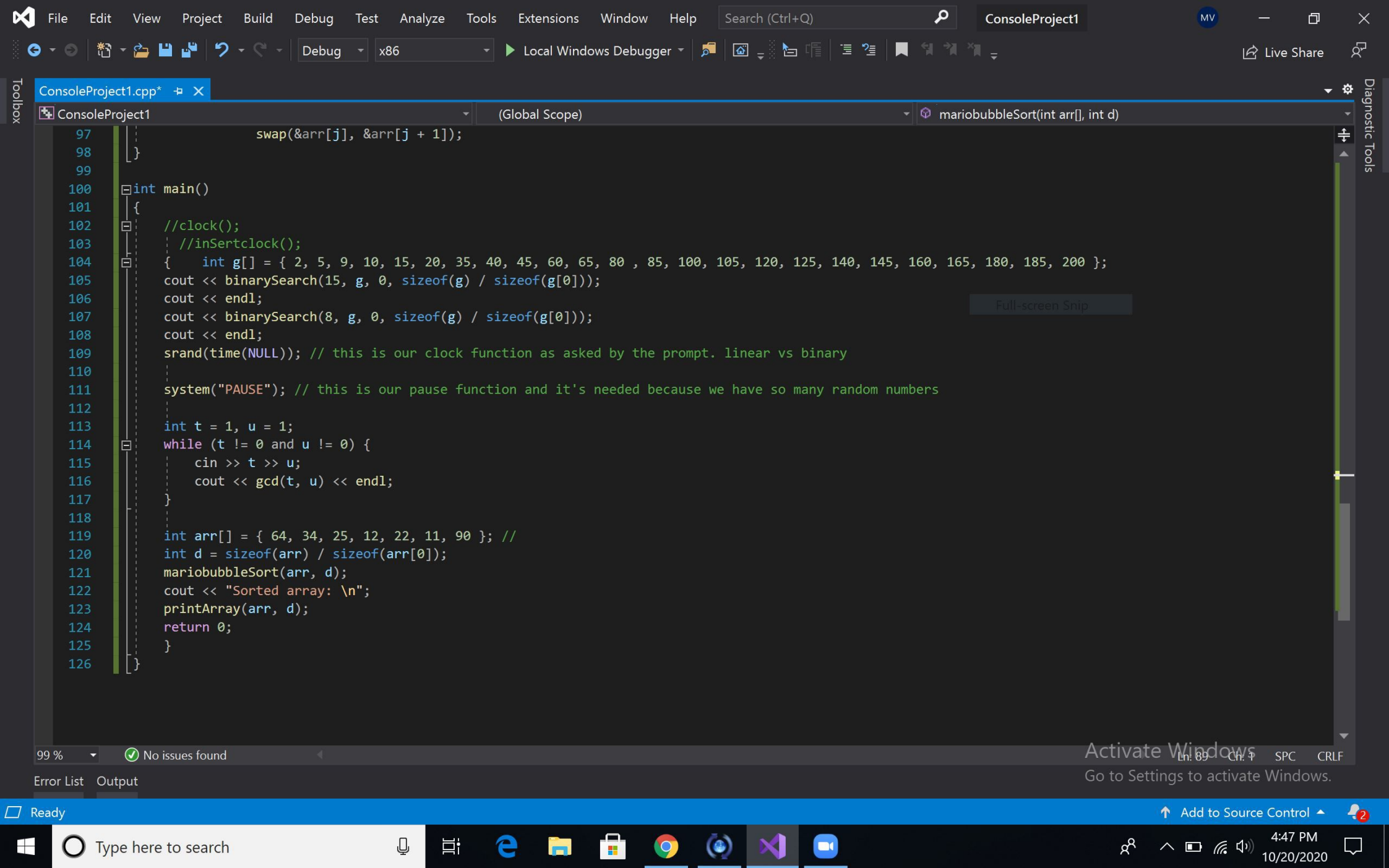
Introduction: For our first project, we will generate a large random table, 10,000 numbers, sort it with QuickSort, and measure it. Afterward, we can then sort it with the Bubble Sor algorithm and measure it. And for advanced options, it is advised to sort it with STL standard library function QuickSort and compare it all.

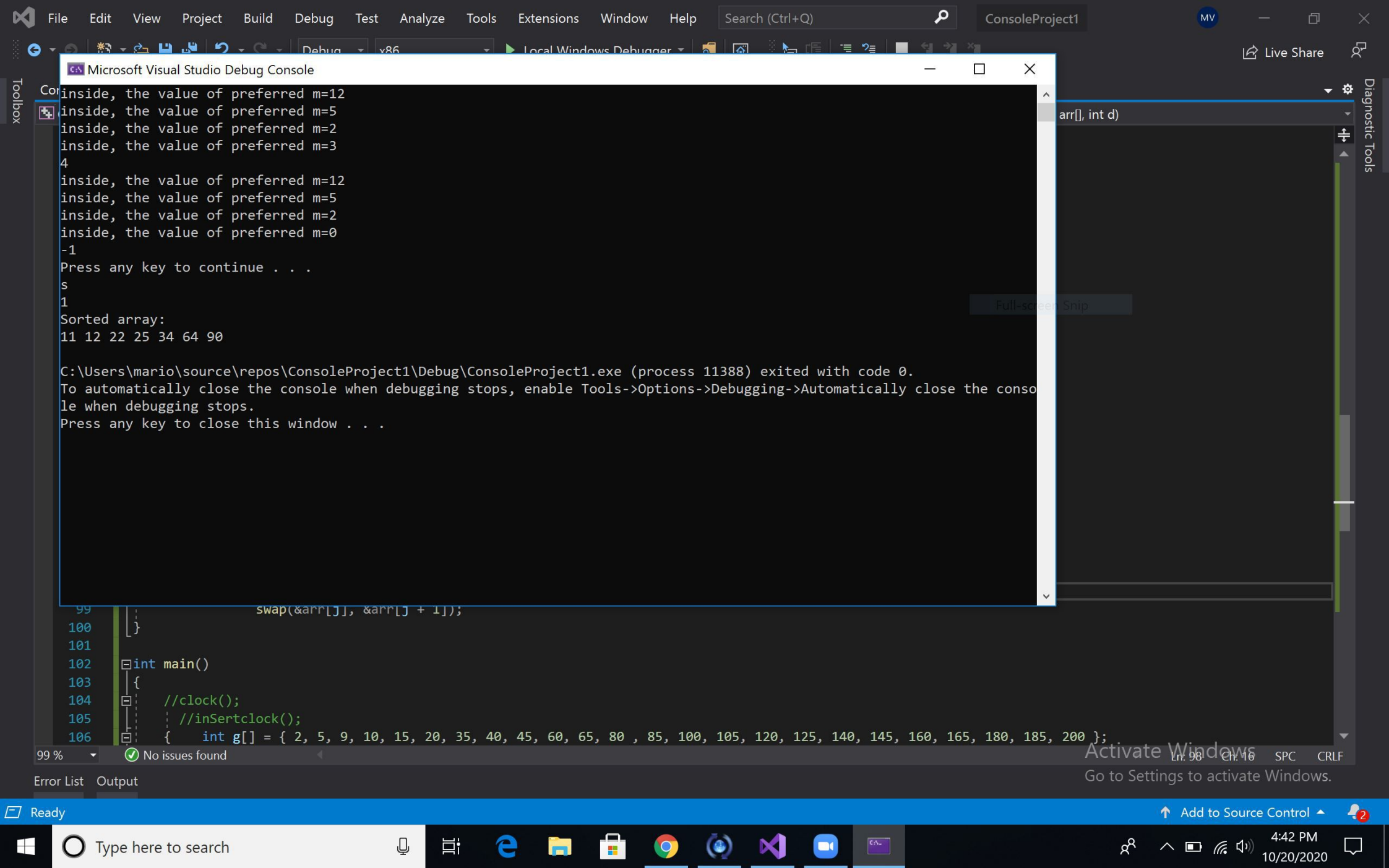
Detailed Analysis and Summary: There should be a basic idea presented of what exactly QuickSort is, and how it works in relation to the project at hand. First off, the QuickSort program itself is logical and requires a great amount of skill to use. It follows the concept of recursion which when a function is called within the same function, and in an advanced way the function is really calling itself. It's both a fruitful endeavor and fascinating when recursive functions can accomplish, with the most immediate being the abilities of QuickSort. While this is being discussed, it is important to note that generating double random numbers is slightly different from generating integer randoms. This will be illustrated in the project to a very understandable degree. Bubblesort and QuickSort both share similarities and differences. I am using Bubblesort in code lines 91-100, and I am also testing QuickSort in code lines 60 -71. In the beginning, I did a couple of test runs while decreasing and increasing the number. The original prompt states that one must generate a random table of 10,000 numbers, but I first went with 100,000 numbers to see what I would get. I should note in advance, that the compiler states what the original prompt requested, but I mentioned my test runs so that I have an idea of how I came to understand the concepts of BubbleSort and QuickSort. For my QuickSort, I utilized pi which is very possible in the application of Visual Studio. I assigned int pi to partition with parentheses. This located in code line 60. I also listed an array numerically as well as integers low and high in the QuickSort function. I also used arr[] quite commonly and this is short for array since this is crucial for measuring the 10,000 numbers. This is project deals heavily with sorting algorithms so it was a way to keep the code in check and working properly. I made use of a clock function as asked by the prompt. linear vs binary, as well as a pause function, and it's needed because we have so many random numbers and the table generated is 10,000 numbers. This going to have to be sorted and measured.


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
ConsoleProject1
Debug
x86
Local Windows Debugger
Live Share
ConsoleProject1.cpp*
ConsoleProject1
(Global Scope)
mariobubbleSort(int arr[], int d)
// ConsoleProject1.cpp : This file contains the 'main' function. Program execution begins and ends there.
//Mario Valdez -- 10/20/20 -- Quick Sort Measurements Project
#include <iostream>
#include <vector>
#include <string>
#include <iomanip>
#include <algorithm>
#include <cmath>
#include <cstdlib>
#include <ctime>
#include <fstream>
using namespace std;
int gcd(int z, int v) { // gcd stands for greatest common denominator
if (z % v == 0) return v; //We must enhance the binary search program
else return (v, z % v);
}
static int mcount = 10000; //large array of 10,000 numbers as mentioned in the prompt
int binarySearch(int b, int j[], int low, int high) {
if (low == high) {
if (j[low] == b) return low;
else return -1; // not found so we must have an else return
}
if (--mcount == 0) {
cout << "This search will give us infinite recursion" << endl;
cout << low << " " << high << endl;
}
int m = (high - low) / 2 + low;
cout << "inside, the value of preferred m=" << m << endl;
if (j[m] == b) return m;
if (j[m] < b) return binarySearch(b, j, m + 1, high);
if (j[m] > b) return binarySearch(b, j, low, m - 1);
99 %
No issues found
Ln: 89 Ch: 1 SPC CRLF
Error List Output
Ready
Add to Source Control
Type here to search
4:46 PM
10/20/2020


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
ConsoleProject1
Debug
x86
Local Windows Debugger
Live Share
ConsoleProject1.cpp*
ConsoleProject1
(Global Scope)
mariobubbleSort(int arr[], int d)
if (j[m] > b) return binarySearch(b, j, low, m - 1);
cout << "error, we should not be here at all and thus should cancel" << endl;
}
void mswap(int a[], int i, int j) {
int temp = a[i]; a[i] = a[j]; a[j] = temp;
}
int mariopartition(int arr[], int low, int high)
{
int pivot = arr[high]; // pivot to arrray high
int y = (low - 1); // this is an Index of a smaller element
for (int j = low; j <= high - 1; j++)
{
if (arr[j] <= pivot)
{
y++; // increment index of smaller element is essential to the success of this code
mswap(arr, y, j); // &arr[y], & arr[j]);
}
}
mswap(arr, y, high); // &arr[y + 1], & arr[high]);
return (y + 1);
}
void marioquickSort(int arr[], int low, int high)
{
if (low < high)
{
/* pi is partitioning index, arr[p] is now
at right place */
int pi;
pi = mariopartition(arr, low, high);
99 %
No issues found
Ln: 89 Ch: 1 SPC CRLF
Error List Output
Ready
Add to Source Control
Type here to search
4:46 PM
10/20/2020

```
        pi = mariopartition(arr, low, high);

        marioquickSort(arr, low, pi - 1);  // Before pi
        marioquickSort(arr, pi + 1, high); // After pi
    }
}

void swap(int* xp, int* yp) //
{
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}

void printArray(int arr[], int size) //
{
    int i;
    for (i = 0; i < size; i++)
        cout << arr[i] << " ";
    cout << endl;
}

void mariobubbleSort(int arr[], int d) //
{
    int y, j;
    for (y = 0; y < d - 1; y++)

        // Last i elements are already in place
        for (j = 0; j < d - y - 1; j++)
            if (arr[j] > arr[j + 1])
                swap(&arr[j], &arr[j + 1]);
}

int main()
```


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
ConsoleProject1
Debug
x86
Local Windows Debugger
Live Share
ConsoleProject1.cpp*
ConsoleProject1
(Global Scope)
mariobubbleSort(int arr[], int d)
swap(&arr[j], &arr[j + 1]);
}
int main()
{
//clock();
//inSertclock();
{ int g[] = { 2, 5, 9, 10, 15, 20, 35, 40, 45, 60, 65, 80 , 85, 100, 105, 120, 125, 140, 145, 160, 165, 180, 185, 200 };
cout << binarySearch(15, g, 0, sizeof(g) / sizeof(g[0]));
cout << endl;
cout << binarySearch(8, g, 0, sizeof(g) / sizeof(g[0]));
cout << endl;
srand(time(NULL)); // this is our clock function as asked by the prompt. linear vs binary
system("PAUSE"); // this is our pause function and it's needed because we have so many random numbers
int t = 1, u = 1;
while (t != 0 and u != 0) {
cin >> t >> u;
cout << gcd(t, u) << endl;
}
int arr[] = { 64, 34, 25, 12, 22, 11, 90 }; //
int d = sizeof(arr) / sizeof(arr[0]);
mariobubbleSort(arr, d);
cout << "Sorted array: \n";
printArray(arr, d);
return 0;
}
}
99 %
No issues found
Ln: 89 Ch: 1 SPC CRLF
Activate Windows
Go to Settings to activate Windows.
Error List Output
Ready
Add to Source Control
Type here to search
4:47 PM
10/20/2020


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
ConsoleProject1
Live Share
Microsoft Visual Studio Debug Console
inside, the value of preferred m=12
inside, the value of preferred m=5
inside, the value of preferred m=2
inside, the value of preferred m=3
4
inside, the value of preferred m=12
inside, the value of preferred m=5
inside, the value of preferred m=2
inside, the value of preferred m=0
-1
Press any key to continue . . .
s
1
Sorted array:
11 12 22 25 34 64 90
C:\Users\mario\source\repos\ConsoleProject1\Debug\ConsoleProject1.exe (process 11388) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
arr[], int d)
swap(&arr[j], &arr[j + 1]);
}
int main()
{
//clock();
//inSertclock();
{ int g[] = { 2, 5, 9, 10, 15, 20, 35, 40, 45, 60, 65, 80 , 85, 100, 105, 120, 125, 140, 145, 160, 165, 180, 185, 200 };
99 %
No issues found
Ln: 98 Ch: 16 SPC CRLF
Error List Output
Activate Windows
Go to Settings to activate Windows.
Ready
Add to Source Control
Type here to search
4:42 PM
10/20/2020