Quinn Roemer

CISP - 430

Assignment 0

1/24/2018

Program 0.1 - BubbleSort

Description:

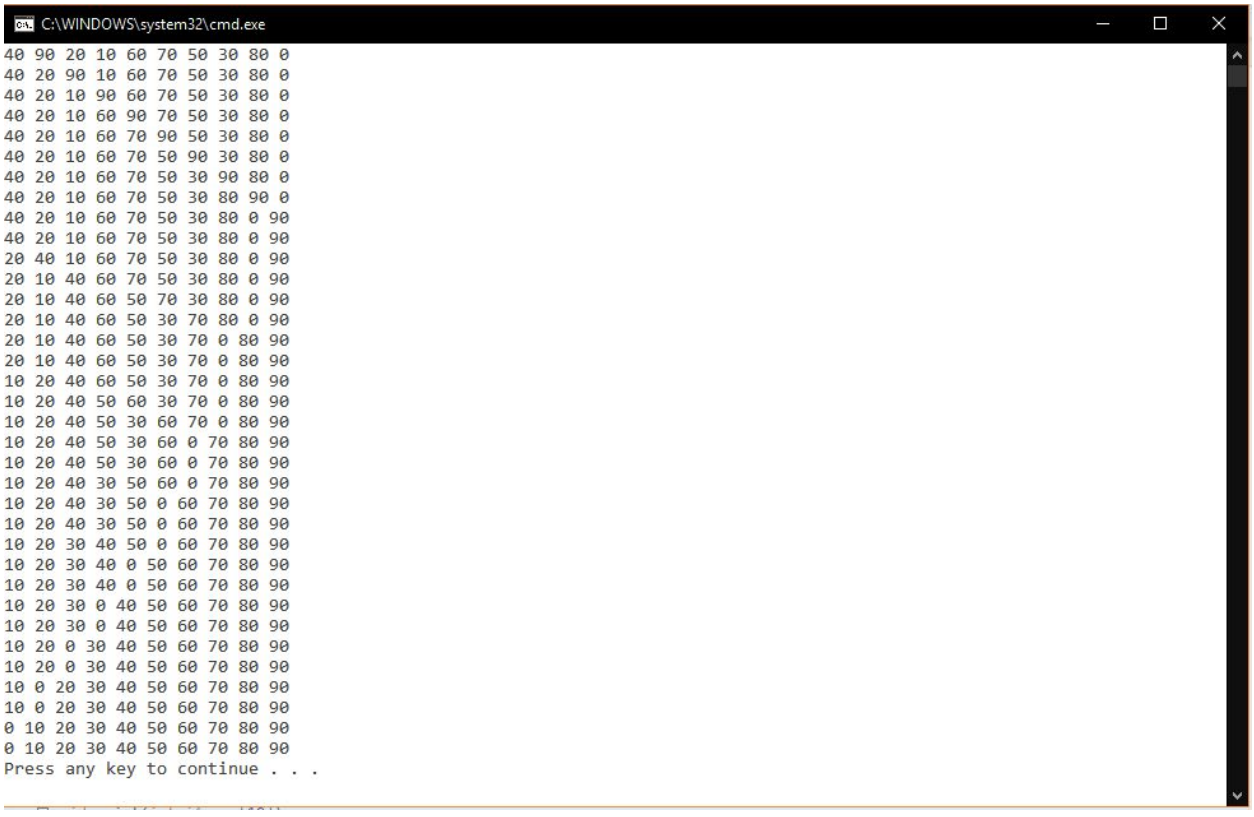
This is a simple program that demonstrates how the BubbleSort algorithm operates. Note, this code was designed by myself. I did not use the provided source code.

Source Code:

```
//Code written by Quinn Roemer

#include <iostream>

using namespace std;

void sort(int iArray[10]);
void print(int iArray[10]);

int main()
{
   int iArray[10] = { 40, 90, 20, 10, 60, 70, 50, 30, 80, 0 };
   sort(iArray);
   return 0;
}
void sort(int iArray[10])
{
   int counter = 9;    int temp;
   bool isSort = false;
   for (int count = 0; count < 10; count++)
   {
            print(iArray);
            if (counter == 0)
            {
                     break;
            }
            for (int index = 0; index < counter; index++)
            {
                     if (iArray[index] > iArray[index + 1])
                     {
                              temp = iArray[index];
                              iArray[index] = iArray[index + 1];
                              iArray[index + 1] = temp;
                              isSort = true;
                             print(iArray);
                     }
```

```
            }
            counter--;
            if (isSort == false)
            {
                        break;
            }
            isSort = false;
   }
}
void print(int iArray[10])
{
   for (int count = 0; count < 10; count++)
   {
            cout << iArray[count] << " ";
   }
   cout << endl;
}
```

Output:



Output further explained:

Note, yellow highlighted text represents the sorted part of the array. Also, blue highlighted text represents the numbers that were swapped.

40 90 20 10 60 70 50 30 80 0
40 20 90 10 60 70 50 30 80 0
40 20 10 90 60 70 50 30 80 0
40 20 10 60 90 70 50 30 80 0
40 20 10 60 70 90 50 30 80 0
40 20 10 60 70 50 90 30 80 0
40 20 10 60 70 50 30 90 80 0
40 20 10 60 70 50 30 80 90 0
40 20 10 60 70 50 30 80 0 90
40 20 10 60 70 50 30 80 0 90
20 40 10 60 70 50 30 80 0 90
20 10 40 60 70 50 30 80 0 90
20 10 40 60 50 70 30 80 0 90
20 10 40 60 50 30 70 80 0 90
20 10 40 60 50 30 70 0 80 90
20 10 40 60 50 30 70 0 80 90
10 20 40 60 50 30 70 0 80 90
10 20 40 50 60 30 70 0 80 90
10 20 40 50 30 60 70 0 80 90
10 20 40 50 30 60 0 70 80 90
10 20 40 50 30 60 0 70 80 90
10 20 40 30 50 60 0 70 80 90
10 20 40 30 50 0 60 70 80 90
10 20 30 40 50 0 60 70 80 90
10 20 30 40 0 50 60 70 80 90
10 20 30 40 0 50 60 70 80 90
10 20 30 0 40 50 60 70 80 90
10 20 0 30 40 50 60 70 80 90
10 0 20 30 40 50 60 70 80 90
0 10 20 30 40 50 60 70 80 90
0 10 20 30 40 50 60 70 80 90

Program 0.2 - SelectionSort

Description:

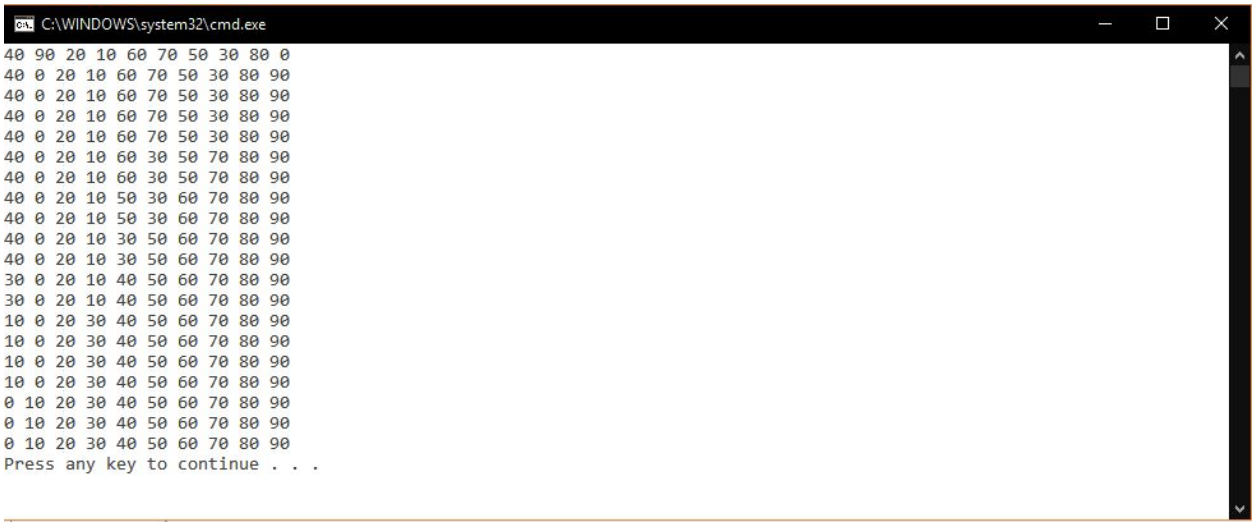
This is a simple program that demonstrates how the SelectionSort algorithm operates. Note, this code was designed by myself. I did not use the provided source code.

Source Code:

```
//Code written by Quinn Roemer

#include <iostream>

using namespace std;

void sort(int iArray[10]);
void print(int iArray[10]);

int main()
{
   int iArray[10] = { 40, 90, 20, 10, 60, 70, 50, 30, 80, 0 };
   sort(iArray);
}
void sort(int iArray[10])
{
   int biggest = 0;
   int temp;
   int counter = 9;
   bool swapNeeded = false;

   for (int index = 0; index < 10; index++)
   {
            print(iArray);
            for (int count = 0; count <= counter; count++)
            {
                     if (iArray[biggest] < iArray[count])
                     {
                              biggest = count;
                              swapNeeded = true;
                     }
            }
            if (swapNeeded = true)
            {
                     temp = iArray[counter];
                     iArray[counter] = iArray[biggest];
```

```
                    iArray[biggest] = temp;
                    counter--;
                    biggest = 0;
                    print(iArray);
            }
            swapNeeded = false;
    }
}
void print(int iArray[10])
{
    for (int count = 0; count < 10; count++)
    {
            cout << iArray[count] << " ";
    }
    cout << endl;
}
```

Output:



Output further explained:

Note, yellow highlighted text represents the sorted part of the array. Also, blue highlighted text represents the next number to be included in the sorted part of the array.

40 90 20 10 60 70 50 30 80 0
40 0 20 10 60 70 50 30 80 90
40 0 20 10 60 70 50 30 80 90
40 0 20 10 60 30 50 70 80 90
40 0 20 10 60 30 50 70 80 90

40 0 20 10 50 30 60 70 80 90
40 0 20 10 50 30 60 70 80 90
40 0 20 10 30 50 60 70 80 90
40 0 20 10 30 50 60 70 80 90
30 0 20 10 40 50 60 70 80 90
30 0 20 10 40 50 60 70 80 90
10 0 20 30 40 50 60 70 80 90
10 0 20 30 40 50 60 70 80 90
10 0 20 30 40 50 60 70 80 90
0 10 20 30 40 50 60 70 80 90

Program 0.3 - InsertionSort

Description:

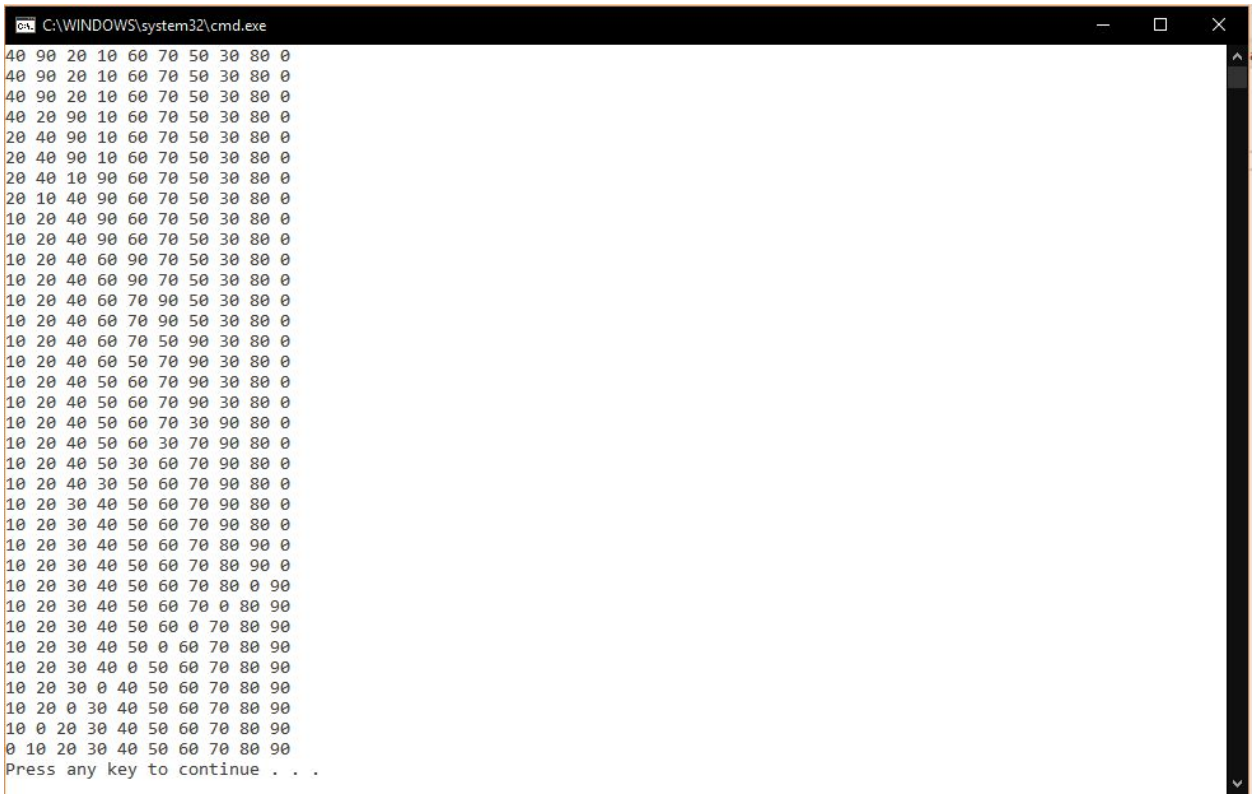
This is a simple program that demonstrates how the InsertionSort algorithm operates. Note, this code was designed by myself. I did not use the provided source code.

Source Code:

```
//Code written by Quinn Roemer

#include <iostream>

using namespace std;

void sort(int iArray[10]);
void print(int iArray[10]);

int main()
{
   int iArray[10] = { 40, 90, 20, 10, 60, 70, 50, 30, 80, 0 };
   sort(iArray);
}
void sort(int iArray[10])
{
   int sort, temp;

   for (int count = 0; count < 10; count++)
   {
            sort = count;
            print(iArray);

            while (sort > 0 && iArray[sort] < iArray[sort - 1])
            {
                     temp = iArray[sort];
                     iArray[sort] = iArray[sort - 1];
                     iArray[sort - 1] = temp;
                     sort--;
                     print(iArray);
            }
   }
}
void print(int iArray[10])
{
   for (int count = 0; count < 10; count++)
```

```
  {
        cout << iArray[count] << " ";
  }
  cout << endl;
}
```

Output:



Output further explained:

Note, yellow highlighted text represents the sorted part of the array. Also, blue highlighted text represents the next number to be included in the sorted part of the array.

40 90 20 10 60 70 50 30 80 0
40 90 20 10 60 70 50 30 80 0
40 20 90 10 60 70 50 30 80 0
20 40 90 10 60 70 50 30 80 0
20 40 10 90 60 70 50 30 80 0
20 10 40 90 60 70 50 30 80 0
10 20 40 90 60 70 50 30 80 0
10 20 40 60 90 70 50 30 80 0
10 20 40 60 70 90 50 30 80 0

10 20 40 60 70 50 90 30 80 0
10 20 40 60 50 70 90 30 80 0
10 20 40 50 60 70 90 30 80 0
10 20 40 50 60 70 30 90 80 0
10 20 40 50 60 30 70 90 80 0
10 20 40 50 30 60 70 90 80 0
10 20 40 30 50 60 70 90 80 0
10 20 30 40 50 60 70 90 80 0
10 20 30 40 50 60 70 80 90 0
10 20 30 40 50 60 70 80 0 90
10 20 30 40 50 60 70 0 80 90
10 20 30 40 50 60 0 70 80 90
10 20 30 40 50 0 60 70 80 90
10 20 30 40 0 50 60 70 80 90
10 20 30 0 40 50 60 70 80 90
10 20 0 30 40 50 60 70 80 90
10 0 20 30 40 50 60 70 80 90
0 10 20 30 40 50 60 70 80 90

Conclusion

This assignment proved to be enlightening. As hard to believe it as it is, through my time as a student of computer-science I have never actually had to program an insertion sort or selection sort algorithm. I had always used the bubble sort algorithm for any sorting requirements that were needed in my assignments for previous classes. That is not to say that I didn't know how to program it, I just decided to stick with the familiar. This assignment allowed me to expand my knowledge and actually get some hands-on experience with these particular sorting methods. Overall, an excellent first assignment for the course!