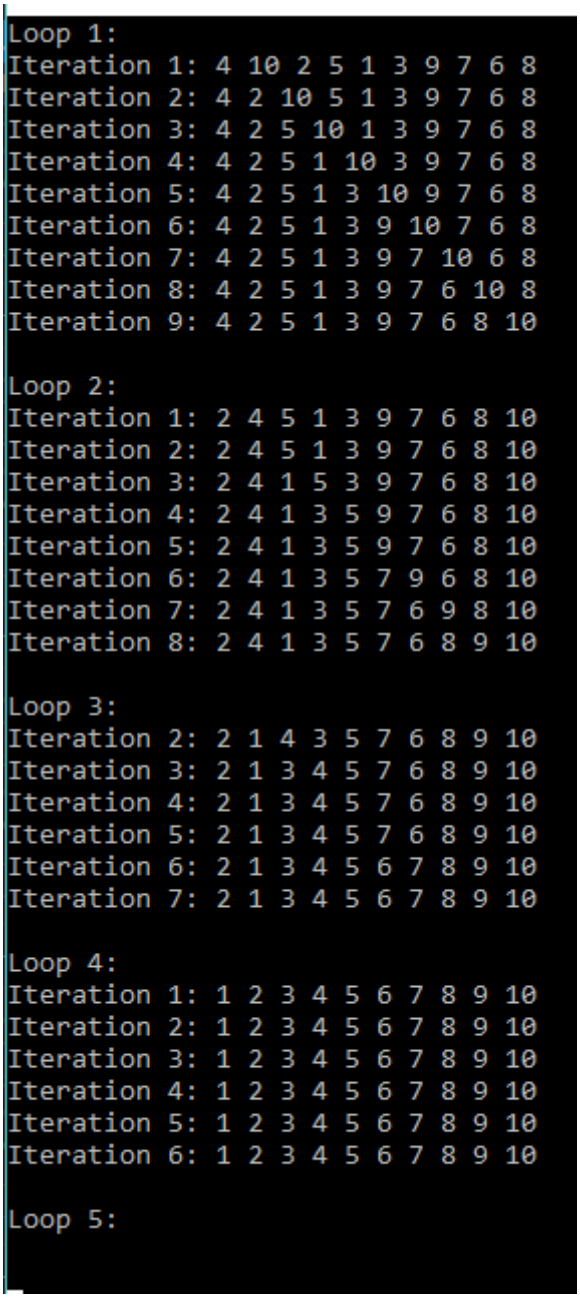
C# - Bubblesortering

Bubble sort är en sorteringsalgoritm som är enkel att implementera men är inte så effektiv. Algoritmen fungerar genom att gå igenom en samling och jämför ett par åt gången, sorterar, och går till nästa par och jämför.
Vi tar ett exempel där vi har 5 siffror i olika storlek, vi vill att dessa ska gå från lägst till högst.
Exempel: 3, 5, 1, 4, 2
Med bubbelsortering kommer vi först jämföra vilket värde som är högst av 3 och 5. 5 är högst så vi behöver inte flytta någon, sen kollar vi vilken som är störst av 5 och 1. Nu måste vi flytta på dessa två eftersom 5 är större. Nu ser listan ut så här: 3, 1, 5, 4, 2
Vi fortsätter nu att kolla nästa par som är 5 och 4, där är 5 större så vi flyttar på dom, då ser listan ut så här: 3, 1, 4, 5, 2. Nu kollar vi sista paret som är 5 och 2 och byter plats på dessa så att listan blir 3, 1, 4, 2, 5. Sedan går bubbelsortering tillbaka från början och flyttar på talen tills listan blir 1, 2, 3, 4, 5.

Bubble Sort i C# (en variation)

```
static void Main(string[] args)
{
    // En osorterad array/vektor
    int[] unsortedArray = new int[10] { 10, 4, 2, 5, 1, 3, 9, 7, 6, 8 };
    // Kontrollvariabel som bestämmer om vi behöver loopa mer eller inte
    bool keepSorting = true;

    // Iterera så länge i < längden på vår array OCH keepSorting är true
    for (int i = 0; i < unsortedArray.Length && keepSorting; i++)
    {
        keepSorting = false;    // Ändra till false, om vi inte sorterat klart
                                // ändras denna till true i loopen.

        Console.Write("Loop " + (i+1) + ":\n");     // Skriver ut vilken Iteration vi är på samt ny rad
        for (int j = 0; j < unsortedArray.Length - i - 1; j++)
        {
            // Om denna position har högre värde än nästa position
            if (unsortedArray[j] > unsortedArray[j + 1])
            {
                // Temp variabel som temporärt håller nuvarande position
                int temp = unsortedArray[j];

                unsortedArray[j] = unsortedArray[j + 1];    // Sätter nästa positions värde som denna
                unsortedArray[j + 1] = temp;                // Sätter denna positions värde som nästa

                keepSorting = true; // Eftersom vi sorterade denna iteration, se till att vi kollar
            }                       // värdena igen och sorterar.

            // Om keepSorting är true, skriv ut och visa föregående iteration
            if (keepSorting)
            {
                Console.Write("Iteration " + (j + 1) + ": ");
                // Iterera igenom och skriv ut varje position på samma rad
                foreach (int pos in unsortedArray)
                {
                    Console.Write(pos + " "); // Position + mellanrum
                }
                Console.WriteLine();    // Ny rad.
            }
        }
        Console.WriteLine();    // Ny rad.
    }
    Console.ReadLine(); // Hindra att programmet inte stänger ned
}
```

Vårat output:



```
Loop 1:
Iteration 1: 4 10 2 5 1 3 9 7 6 8
Iteration 2: 4 2 10 5 1 3 9 7 6 8
Iteration 3: 4 2 5 10 1 3 9 7 6 8
Iteration 4: 4 2 5 1 10 3 9 7 6 8
Iteration 5: 4 2 5 1 3 10 9 7 6 8
Iteration 6: 4 2 5 1 3 9 10 7 6 8
Iteration 7: 4 2 5 1 3 9 7 10 6 8
Iteration 8: 4 2 5 1 3 9 7 6 10 8
Iteration 9: 4 2 5 1 3 9 7 6 8 10

Loop 2:
Iteration 1: 2 4 5 1 3 9 7 6 8 10
Iteration 2: 2 4 5 1 3 9 7 6 8 10
Iteration 3: 2 4 1 5 3 9 7 6 8 10
Iteration 4: 2 4 1 3 5 9 7 6 8 10
Iteration 5: 2 4 1 3 5 9 7 6 8 10
Iteration 6: 2 4 1 3 5 7 9 6 8 10
Iteration 7: 2 4 1 3 5 7 6 9 8 10
Iteration 8: 2 4 1 3 5 7 6 8 9 10

Loop 3:
Iteration 2: 2 1 4 3 5 7 6 8 9 10
Iteration 3: 2 1 3 4 5 7 6 8 9 10
Iteration 4: 2 1 3 4 5 7 6 8 9 10
Iteration 5: 2 1 3 4 5 7 6 8 9 10
Iteration 6: 2 1 3 4 5 6 7 8 9 10
Iteration 7: 2 1 3 4 5 6 7 8 9 10

Loop 4:
Iteration 1: 1 2 3 4 5 6 7 8 9 10
Iteration 2: 1 2 3 4 5 6 7 8 9 10
Iteration 3: 1 2 3 4 5 6 7 8 9 10
Iteration 4: 1 2 3 4 5 6 7 8 9 10
Iteration 5: 1 2 3 4 5 6 7 8 9 10
Iteration 6: 1 2 3 4 5 6 7 8 9 10

Loop 5:
```

Efter loop 4 har vi sorterat vår lista och den går nu från lägst till högst.