

BUBBLE SORT

```
#include <bits/stdc++.h>

using namespace std;

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

void bubbleSort(int arr[], int n)
{
    int i, j;
    bool swapped;
    for (i = 0; i < n-1; i++)
    {
        swapped = false;
        for (j = 0; j < n-i-1; j++)
        {
            if (arr[j] > arr[j+1])
            {
                swap(&arr[j], &arr[j+1]);
                swapped = true;
            }
        }
        // IF no two elements were swapped by inner loop, then break
        if (swapped == false)
            break;
    }
}
```

```

void printArray(int arr[], int size)
{
    int i;
    for (i=0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

int main()
{
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr)/sizeof(arr[0]);

    bubbleSort(arr, n);

    printf("Sorted array: \n");
    printArray(arr, n);

    return 0;
}

```

[edit](#) [fork](#) [download](#)

[copy](#)

```

1. #include <bits/stdc++.h>
2. using namespace std;
3. void swap(int *xp, int *yp)
4. {
5.     int temp = *xp;
6.     *xp = *yp;
7.     *yp = temp;
8. }
9. void bubbleSort(int arr[], int n)
10. {
11.     int i, j;
12.     bool swapped;
13.     for (i = 0; i < n-1; i++)
14.     {
15.         swapped = false;
16.         for (j = 0; j < n-i-1; j++)
17.         {
18.             if (arr[j] > arr[j+1])
19.             {
20.                 swap(&arr[j], &arr[j+1]);
21.                 swapped = true;
22.             }
23.         }
24.         // IF no two elements were swapped by inner loop, then break
25.         if (swapped == false)
26.             break;
27.     }
28. }
29. void printArray(int arr[], int size)
30. {
31.     int i;
32.     for (i=0; i < size; i++)
33.         printf("%d ", arr[i]);
34.     printf("\n");
35. }
36. int main()
37. {
38.     int arr[] = {64, 34, 25, 12, 22, 11, 90};
39.     int n = sizeof(arr)/sizeof(arr[0]);
40.     bubbleSort(arr, n);
41.     printf("Sorted array: \n");
42.     printArray(arr, n);
43.     return 0;
44. }

```

 stdin copy

Standard input is empty

 stdout copy

Sorted array:

11 12 22 25 34 64 90 n

Time Complexity:

Worst and Average Case Time Complexity: $O(n*n)$. Worst case occurs when array is reverse sorted.

Best Case Time Complexity: $O(n)$. Best case occurs when array is already sorted.

INSERTION SORT

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

```
void insertionSort(int arr[], int n)
```

```
{
```

```
    int i, key, j;
```

```
    for (i = 1; i < n; i++)
```

```
    {
```

```
        key = arr[i];
```

```
        j = i - 1;
```

```
        while (j >= 0 && arr[j] > key)
```

```
        {
```

```
            arr[j + 1] = arr[j];
```

```
            j = j - 1;
```

```
        }
```

```
        arr[j + 1] = key;
```

```
    }
```

```
}
```

```
void printArray(int arr[], int n)
```

```
{
```

```
    int i;
```

```
    for (i = 0; i < n; i++)
```

```
        cout << arr[i] << " ";
```

```
    cout << endl;
```

```
}
```

```
int main()
```

```

{
    int arr[] = { 12, 11, 13, 5, 6 };
    int n = sizeof(arr) / sizeof(arr[0]);
    insertionSort(arr, n);
    printArray(arr, n);
    return 0;
}

```

Time Complexity: $O(n^2)$

[edit](#) [fork](#) [download](#)

[copy](#)

```

1. #include <bits/stdc++.h>
2. using namespace std;
3. void insertionSort(int arr[], int n)
4. {
5.     int i, key, j;
6.     for (i = 1; i < n; i++)
7.     {
8.         key = arr[i];
9.         j = i - 1;
10.        while (j >= 0 && arr[j] > key)
11.        {
12.            arr[j + 1] = arr[j];
13.            j = j - 1;
14.        }
15.        arr[j + 1] = key;
16.    }
17. }
18. void printArray(int arr[], int n)
19. {
20.     int i;
21.     for (i = 0; i < n; i++)
22.         cout << arr[i] << " ";
23.     cout << endl;
24. }
25. int main()
26. {
27.     int arr[] = { 12, 11, 13, 5, 6 };
28.     int n = sizeof(arr) / sizeof(arr[0]);
29.     insertionSort(arr, n);
30.     printArray(arr, n);
31.     return 0;
32. }

```

Success #stdin #stdout 0s 5460KB

[comments \(0\)](#)

[stdin](#)

[copy](#)

Standard input is empty

[stdout](#)

[copy](#)

5 6 11 12 13