Bucket Sort Code In C++

```
#include <iostream>
using namespace std;

// Function to perform insertion sort on a bucket
void insertionSort(float bucket[], int n) {
    for (int i = 1; i < n; i++) {
        float key = bucket[i];
        int j = i - 1;
        while (j >= 0 && bucket[j] > key) {
            bucket[j + 1] = bucket[j];
            j--;
        }
        bucket[j + 1] = key;
    }
}

// Function to perform Bucket Sort
void bucketSort(float arr[], int n) {
    // 1. Create n empty buckets (fixed-size arrays)
    const int BUCKET_SIZE = 10; // Max size of each bucket
    float buckets[n][BUCKET_SIZE];
    int bucketCount[n] = {0}; // Track number of elements in each bucket

    // 2. Distribute array elements into buckets
    for (int i = 0; i < n; i++) {
        int index = n * arr[i]; // Bucket index
        if (index >= n) index = n - 1; // Handle edge case when arr[i] = 1
        buckets[index][bucketCount[index]++] = arr[i];
    }

    // 3. Sort individual buckets using insertion sort
    for (int i = 0; i < n; i++) {
        insertionSort(buckets[i], bucketCount[i]);
    }

    // 4. Concatenate buckets back into original array
    int k = 0;
```

```
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < bucketCount[i]; j++) {
            arr[k++] = buckets[i][j];
        }
    }
}

// Function to print array
void printArray(float arr[], int n) {
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;
}

// Driver code
int main() {
    float arr[] = {0.42, 0.32, 0.23, 0.52, 0.25, 0.47, 0.51};
    int n = sizeof(arr) / sizeof(arr[0]);

    cout << "Original array: ";
    printArray(arr, n);

    bucketSort(arr, n);

    cout << "Sorted array:   ";
    printArray(arr, n);

    return 0;
}
```

Output-

```
Original array: 0.42 0.32 0.23 0.52 0.25 0.47 0.51
Sorted array:   0.23 0.25 0.32 0.42 0.47 0.51 0.52
```