Write a program to implement Parallel Bubble using OpenMP. Use existing algorithms and measure the performance of sequential and parallel algorithms.

```
#include <iostream>
#include <vector>
#include <omp.h>
#include <chrono>

void sequentialBubbleSort(std::vector<int>& arr) {
    int n = arr.size();
    for (int i = 0; i < n - 1; ++i) {
        for (int j = 0; j < n - i - 1; ++j) {
            if (arr[j] > arr[j + 1]) {
                std::swap(arr[j], arr[j + 1]);
            }
        }
    }
}

void parallelBubbleSort(std::vector<int>& arr) {
    int n = arr.size();
    bool sorted = false;

    while (!sorted) {
        sorted = true;

        #pragma omp parallel for shared(arr, sorted)
        for (int i = 0; i < n - 1; ++i) {
            if (arr[i] > arr[i + 1]) {
                std::swap(arr[i], arr[i + 1]);
                sorted = false;
            }
        }
    }
}

int main() {
    std::vector<int> arr = {9, 4, 2, 7, 5, 1, 8, 3, 6};

    std::cout << "Sequential Bubble Sort:" << std::endl;
    std::vector<int> arrSeq = arr;
    auto startSeq = std::chrono::steady_clock::now();
    sequentialBubbleSort(arrSeq);
    auto endSeq = std::chrono::steady_clock::now();
    std::chrono::duration<double> durationSeq = endSeq - startSeq;

    for (int num : arrSeq) {
        std::cout << num << " ";
    }

    std::cout << "\n\nParallel Bubble Sort:" << std::endl;
    std::vector<int> arrPar = arr;
    auto startPar = std::chrono::steady_clock::now();
    parallelBubbleSort(arrPar);
    auto endPar = std::chrono::steady_clock::now();
    std::chrono::duration<double> durationPar = endPar - startPar;

    for (int num : arrPar) {
        std::cout << num << " ";
    }

    std::cout << "\n\nSequential Bubble Sort Duration: " << durationSeq.count() << " seconds";
    std::cout << "\nParallel Bubble Sort Duration: " << durationPar.count() << " seconds";

    return 0;
}
```