**Practical 2 : Bubble Sort**

#include <iostream>

#include <omp.h>

#include<cstdlib>

using namespace std;

void sequentialBubbleSort(int \*, int);

void parallelBubbleSort(int \*, int);

void swap(int &, int &);

void sequentialBubbleSort(int \*a, int n)

{

int swapped;

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

{

swapped = 0;

for (int j = 0; j < n - 1; j++)

{

if (a[j] > a[j + 1])

{

swap(a[j], a[j + 1]);

swapped = 1;

}

}

if (!swapped)

break;

}

}

void parallelBubbleSort(int \*a, int n)

{

int swapped;

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

{

swapped = 0;

#pragma omp parallel for shared(a)

for (int j = 0; j < n - 1; j++)

{

if (a[j] > a[j + 1])

{

swap(a[j], a[j + 1]);

swapped = 1;

}

}

if (!swapped)

break;

}

}

void swap(int &a, int &b)

{

int test;

test = a;

a = b;

b = test;

}

int main()

{

int \*a, \*b, n;

cout << "\nEnter total no of elements : ";

cin >> n;

a = new int[n];

b = new int[n];

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

{

int random\_value = rand()%1000000;

a[i] = random\_value;

b[i] = random\_value;

}

double start\_time = omp\_get\_wtime(); // start timer for sequential algorithm

sequentialBubbleSort(a, n);

double end\_time = omp\_get\_wtime(); // end timer for sequential algorithm

double seq\_time = end\_time - start\_time;

cout << "\nSorted array ( first 20 elements ) : \n";

for (int i = 0; i < 20; i++)

{

cout << a[i] <<", ";

}

start\_time = omp\_get\_wtime(); // start timer for parallel algorithm

parallelBubbleSort(b, n);

end\_time = omp\_get\_wtime(); // end timer for parallel algorithm

double parallel\_time = end\_time - start\_time;

cout << "\n\nSorted array ( first 20 elements using parallel algorithm ) : \n";

for (int i = 0; i < 20; i++)

{

cout << b[i] <<", ";

}

cout << "\n\nTime taken by sequential algorithm: " << seq\_time << " seconds" << endl;

cout << "\n\nTime taken by parallel algorithm: " << parallel\_time << " seconds" << endl;

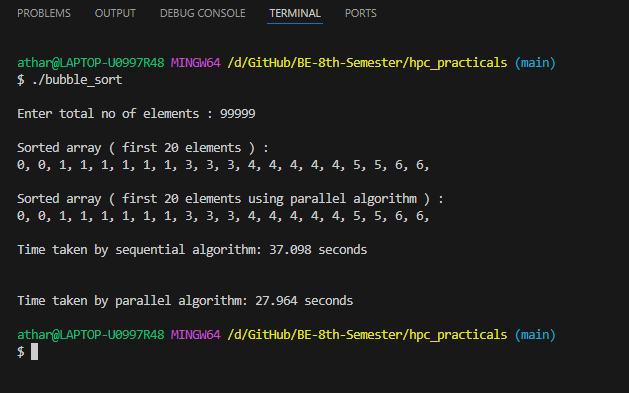
delete[] a; // Don't forget to free the allocated memory

delete[] b;

return 0;

}

**Output :**

****