HPC Lab Exp No.3

#include <iostream>

#include <vector>

#include <omp.h>

using namespace std;

long parallelSum(const vector<int>& arr) {

long sum = 0;

#pragma omp parallel for reduction(+:sum)

for (size\_t i = 0; i < arr.size(); ++i) {

sum += arr[i];

}

return sum;

}

int parallelMax(const vector<int>& arr) {

int max\_val = arr[0];

#pragma omp parallel for reduction(max:max\_val)

for (size\_t i = 1; i < arr.size(); ++i) {

if (arr[i] > max\_val) {

max\_val = arr[i];

}

}

return max\_val;

}

int parallelMin(const vector<int>& arr) {

int min\_val = arr[0];

#pragma omp parallel for reduction(min:min\_val)

for (size\_t i = 1; i < arr.size(); ++i) {

if (arr[i] < min\_val) {

min\_val = arr[i];

}

}

return min\_val;

}

double parallelAverage(const vector<int>& arr) {

long sum = parallelSum(arr);

return static\_cast<double>(sum) / arr.size();

}

int main() {

int n;

cout << "Enter the number of elements: ";

cin >> n;

vector<int> arr(n);

cout << "Enter the elements: ";

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

cin >> arr[i];

}

long sum = parallelSum(arr);

int max\_val = parallelMax(arr);

int min\_val = parallelMin(arr);

double average = parallelAverage(arr);

cout << "Sum: " << sum << endl;

cout << "Max: " << max\_val << endl;

cout << "Min: " << min\_val << endl;

cout << "Average: " << average << endl;

return 0;

}

OUTPUT:

