**Code**

#include <iostream>

#include <vector>

#include <omp.h>

#include <climits>

using namespace std;

void min\_reduction(vector<int>& arr) {

int min\_value = INT\_MAX;

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

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

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

min\_value = arr[i];

}

}

cout << "Minimum value: " << min\_value << endl;

}

void max\_reduction(vector<int>& arr) {

int max\_value = INT\_MIN;

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

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

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

max\_value = arr[i];

}

}

cout << "Maximum value: " << max\_value << endl;

}

void sum\_reduction(vector<int>& arr) {

int sum = 0;

#pragma omp parallel for reduction(+: sum)

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

sum += arr[i];

}

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

}

void average\_reduction(vector<int>& arr) {

int sum = 0;

#pragma omp parallel for reduction(+: sum)

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

sum += arr[i];

}

cout << "Average: " << (double)sum / arr.size() << endl;

}

int main() {

vector<int> arr;

arr.push\_back(5);

arr.push\_back(2);

arr.push\_back(9);

arr.push\_back(1);

arr.push\_back(7);

arr.push\_back(6);

arr.push\_back(8);

arr.push\_back(3);

arr.push\_back(4);

min\_reduction(arr);

max\_reduction(arr);

sum\_reduction(arr);

average\_reduction(arr);

}

**Output:**

****