

```

1 // STATISTICS.CPP
2
3 #include <limits.h>
4 #include <omp.h>
5 #include <stdlib.h>
6
7 #include <array>
8 #include <chrono>
9 #include <functional>
10 #include <iostream>
11 #include <string>
12 #include <vector>
13
14 using std::chrono::duration_cast;
15 using std::chrono::high_resolution_clock;
16 using std::chrono::milliseconds;
17 using namespace std;
18
19 void s_avg(int arr[], int n) {
20     long sum = 0L;
21     int i;
22     for (i = 0; i < n; i++) {
23         sum = sum + arr[i];
24     }
25     // cout << "\nAverage = " << sum / long(n) << "\n";
26 }
27
28 void p_avg(int arr[], int n) {
29     long sum = 0L;
30     int i;
31 #pragma omp parallel for reduction(+ : sum) num_threads(16)
32     for (i = 0; i < n; i++) {
33         sum = sum + arr[i];
34     }
35     // cout << "\nAverage = " << sum / long(n) << "\n";
36 }
37
38 void s_sum(int arr[], int n) {
39     long sum = 0L;
40     int i;
41     for (i = 0; i < n; i++) {
42         sum = sum + arr[i];
43     }
44     // cout << "\nSum = " << sum << "\n";
45 }
46
47 void p_sum(int arr[], int n) {
48     long sum = 0L;
49     int i;
50 #pragma omp parallel for reduction(+ : sum) num_threads(16)
51     for (i = 0; i < n; i++) {
52         sum = sum + arr[i];
53     }
54     // cout << "\nSum = " << sum << "\n";
55 }
56
57 void s_max(int arr[], int n) {
58     int max_val = INT_MIN;
59     int i;
60     for (i = 0; i < n; i++) {
61         if (arr[i] > max_val) {
62             max_val = arr[i];
63         }
64     }
65     // cout << "\nMax value = " << max_val << "\n";
66 }
67
68 void p_max(int arr[], int n) {
69     int max_val = INT_MIN;
70     int i;
71 #pragma omp parallel for reduction(max : max_val) num_threads(16)
72     for (i = 0; i < n; i++) {
73         if (arr[i] > max_val) {
74             max_val = arr[i];

```

```

75     }
76 }
77 // cout << "\nMax value = " << max_val << "\n";
78 }
79
80 void s_min(int arr[], int n) {
81     int min_val = INT_MAX;
82     int i;
83     for (i = 0; i < n; i++) {
84         if (arr[i] < min_val) {
85             min_val = arr[i];
86         }
87     }
88     // cout << "\nMin value = " << min_val << "\n";
89 }
90
91 void p_min(int arr[], int n) {
92     int min_val = INT_MAX;
93     int i;
94     #pragma omp parallel for reduction(min : min_val) num_threads(16)
95     for (i = 0; i < n; i++) {
96         if (arr[i] < min_val) {
97             min_val = arr[i];
98         }
99     }
100     // cout << "\nMin value = " << min_val << "\n";
101 }
102
103 std::string bench_traverse(std::function<void()> traverse_fn) {
104     auto start = high_resolution_clock::now();
105     traverse_fn();
106     auto stop = high_resolution_clock::now();
107
108     // Subtract stop and start timepoints and cast it to required unit.
109     // Predefined units are nanoseconds, microseconds, milliseconds, seconds,
110     // minutes, hours. Use duration_cast() function.
111     auto duration = duration_cast<milliseconds>(stop - start);
112
113     // To get the value of duration use the count() member function on the
114     // duration object
115     return std::to_string(duration.count());
116 }
117
118 int main(int argc, const char **argv) {
119     if (argc < 2) {
120         std::cout << "Specify array length.\n";
121         return 1;
122     }
123     int *a, n, i;
124
125     n = stoi(argv[1]);
126     a = new int[n];
127
128     for (int i = 0; i < n; i++) {
129         a[i] = rand() % n;
130     }
131
132     cout << "Generated random array of length " << n << "\n\n";
133     omp_set_num_threads(16);
134
135     std::cout << "Sequential Min: " << bench_traverse([&] { s_min(a, n); }) << "ms\n";
136
137     std::cout << "Parallel (16) Min: " << bench_traverse([&] { p_min(a, n); }) << "ms\n";
138
139     std::cout << "\nSequential Max: " << bench_traverse([&] { s_max(a, n); }) << "ms\n";
140
141     std::cout << "Parallel (16) Max: " << bench_traverse([&] { p_max(a, n); }) << "ms\n";
142
143     std::cout << "\nSequential Sum: " << bench_traverse([&] { s_sum(a, n); }) << "ms\n";
144
145     std::cout << "Parallel (16) Sum: " << bench_traverse([&] { p_sum(a, n); }) << "ms\n";
146
147     std::cout << "\nSequential Average: " << bench_traverse([&] { s_avg(a, n); }) << "ms\n";
148
149     std::cout << "Parallel (16) Average: " << bench_traverse([&] { p_avg(a, n); }) << "ms\n";

```

```
150
151     // cout << "\nSorted array is =>";
152     // for (i = 0; i < n; i++) {
153     //     cout << "\n" << a[i];
154     // }
155     return 0;
156 }
157
158 /*
159
160 OUTPUT:
161
162 Generated random array of length 100000000
163
164 Sequential Min: 567ms
165 Parallel (16) Min: 49ms
166
167 Sequential Max: 568ms
168 Parallel (16) Max: 46ms
169
170 Sequential Sum: 579ms
171 Parallel (16) Sum: 46ms
172
173 Sequential Average: 579ms
174 Parallel (16) Average: 45ms
175
176 */
```