

Grp A_Assignment 3

Implement Min, Max, Sum and Average operations using Parallel Reduction.

```
#include <iostream>
//#include <vector>
#include <omp.h>
#include <climits>
using namespace std;
void min_reduction(int arr[], int n) {
    int min_value = INT_MAX;
    #pragma omp parallel for reduction(min: min_value)
    for (int i = 0; i < n; i++) {
        if (arr[i] < min_value) {
            min_value = arr[i];
        }
    }
    cout << "Minimum value: " << min_value << endl;
}

void max_reduction(int arr[], int n) {
    int max_value = INT_MIN;
    #pragma omp parallel for reduction(max: max_value)
    for (int i = 0; i < n; i++) {
        if (arr[i] > max_value) {
            max_value = arr[i];
        }
    }
    cout << "Maximum value: " << max_value << endl;
}

void sum_reduction(int arr[], int n) {
    int sum = 0;
    #pragma omp parallel for reduction(+: sum)
    for (int i = 0; i < n; i++) {
        sum += arr[i];
    }
    cout << "Sum: " << sum << endl;
}

void average_reduction(int arr[], int n) {
    int sum = 0;
```

```

#pragma omp parallel for reduction(+: sum)
for (int i = 0; i < n; i++) {
    sum += arr[i];
}
cout << "Average: " << (double)sum / (n-1) << endl;
}

```

```

int main() {
    int *arr,n;
    cout<<"\n enter total no of elements=>";
    cin>>n;
    arr=new int[n];
    cout<<"\n enter elements=>";
    for(int i=0;i<n;i++)
    {
        cin>>arr[i];
    }
}

```

```

// int arr[] = {5, 2, 9, 1, 7, 6, 8, 3, 4};
// int n = size(arr);

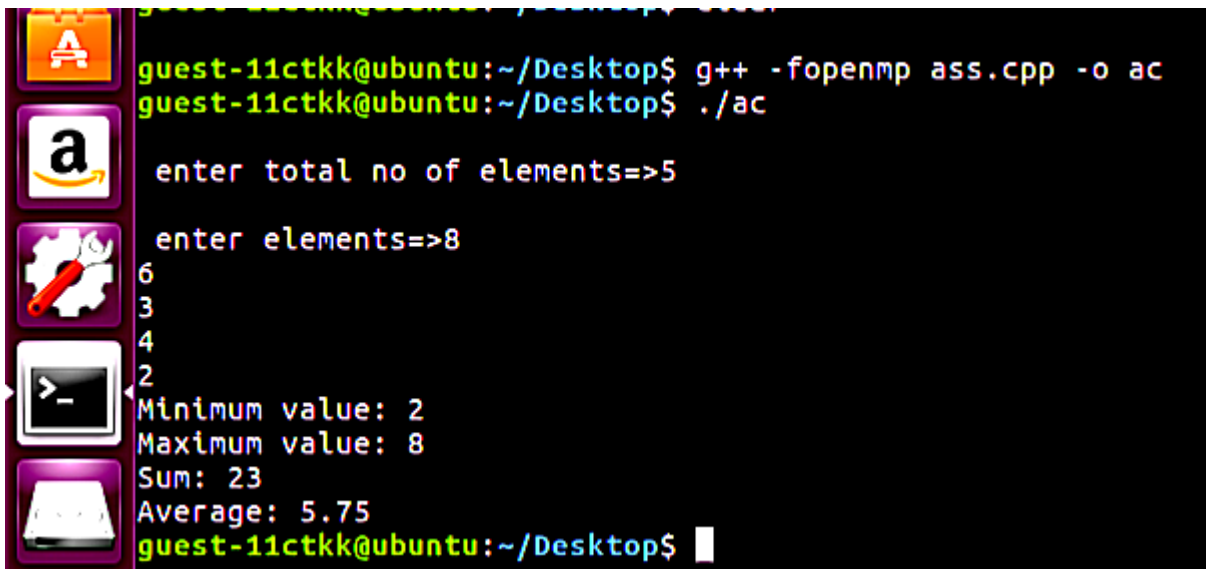
```

```

min_reduction(arr, n);
max_reduction(arr, n);
sum_reduction(arr, n);
average_reduction(arr, n);
}

```

Output



```

guest-11ctkk@ubuntu:~/Desktop$ g++ -fopenmp ass.cpp -o ac
guest-11ctkk@ubuntu:~/Desktop$ ./ac

enter total no of elements=>5

enter elements=>8
6
3
4
2
9
Minimum value: 2
Maximum value: 8
Sum: 23
Average: 5.75
guest-11ctkk@ubuntu:~/Desktop$

```

Void Min_reduction()

- void min_reduction(vector<int>& arr) declares a void function that takes a reference to an integer vector as its argument.
- int min_value = INT_MAX; initializes an integer variable min_value to the largest possible integer value using the INT_MAX constant from the <climits> header file. This is done to ensure that min_value is initially greater than any element in arr.

- `#pragma omp parallel for reduction(min: min_value)` is an OpenMP directive that specifies that the following loop should be executed in parallel using multiple threads. The `reduction(min: min_value)` clause indicates that each thread should maintain a private copy of `min_value` and update it with the minimum value it finds in its portion of the loop. Once the loop is complete, OpenMP will combine all the private copies of `min_value` into a single shared value that represents the minimum value in `arr`.
- `for (int i = 0; i < arr.size(); i++) {` is a loop that iterates over each element of `arr`.
- `if (arr[i] < min_value) { min_value = arr[i]; }` checks if the current element of `arr` is less than `min_value`. If so, it updates `min_value` to be the current element.
- `cout << "Minimum value: " << min_value << endl;` prints out the minimum value found in `arr`.

void max_reduction()

- `void max_reduction(vector<int>& arr)` declares a void function that takes a reference to an integer vector as its argument.
- `int max_value = INT_MIN;` initializes an integer variable `max_value` to the smallest possible integer value using the `INT_MIN` constant from the `<climits>` header file. This is done to ensure that `max_value` is initially smaller than any element in `arr`.
- `#pragma omp parallel for reduction(max: max_value)` is an OpenMP directive that specifies that the following loop should be executed in parallel using multiple threads. The `reduction(max: max_value)` clause indicates that each thread should maintain a private copy of `max_value` and update it with the maximum value it finds in its portion of the loop. Once the loop is complete, OpenMP will combine all the private copies of `max_value` into a single shared value that represents the maximum value in `arr`.
- `for (int i = 0; i < arr.size(); i++) {` is a loop that iterates over each element of `arr`.
- `if (arr[i] > max_value) { max_value = arr[i]; }` checks if the current element of `arr` is greater than `max_value`. If so, it updates `max_value` to be the current element.
- `cout << "Maximum value: " << max_value << endl;` prints out the maximum value found in `arr`.

#include <climits>

`<climits>` is a header file in C++ that contains constants related to integer types. This header file provides implementation-defined constants for minimum and maximum values of integral types, such as `INT_MAX` (maximum value of `int`) and `INT_MIN` (minimum value of `int`).

Using these constants instead of hardcoding the values of the minimum and maximum integer values is a good practice because it makes the code more readable and avoids the possibility of introducing errors in the code. The use of these constants also ensures that the code will work correctly across different platforms and compilers.

INT_MIN :

Minimum value for an object of type `int`

Value of `INT_MIN` is $-2^{15}+1$ or less*

INT_MAX :

Maximum value for an object of type `int`

Value of INT_MAX is 2147483647 (-2^{31} to $2^{31}-1$)