OpenMP Parallel Reduction Code Explanation

Detailed Code Explanation

OpenMP Parallel Reduction Code Explanation

Headers and Setup

```cpp
#include <iostream>
#include <omp.h>
#include <climits>
```

- `iostream`: Standard input/output operations.
- `omp.h`: Required for OpenMP parallel programming features.
- `climits`: Provides `INT_MAX` and `INT_MIN` constants for finding min/max.

---

min_reduction Function

```cpp
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;
}
```

- Uses OpenMP's `reduction` to find the minimum value in the array in parallel.

OpenMP Parallel Reduction Code Explanation

- Each thread gets a local copy of `min_value`.
- After the loop, OpenMP combines them to find the global minimum.

---

max_reduction Function

```cpp
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;
}
```

- Works like `min_reduction`, but for the maximum value.

---

sum_reduction Function

```cpp
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;
```

OpenMP Parallel Reduction Code Explanation

}
```
- Uses `+` reduction to sum elements in parallel.

---

## average_reduction Function
```cpp
void average_reduction(int arr[], int n) {
    if (n <= 1) {
        cout << "Average: Cannot calculate (array size too small)" << endl;
        return;
    }
    int sum = 0;
    #pragma omp parallel for reduction(+ : sum)
    for (int i = 0; i < n; i++) {
        sum += arr[i];
    }
    cout << "Average: " << static_cast<double>(sum) / n << endl;
}
```
- Reuses the sum logic.
- Converts sum to `double` and divides by `n` for the average.

---

main() Function
```cpp
int main() {
    cout << "\n\nName: Girish Raut\nRoll No.39 \t Div.B\n\n";
    int *arr, n;

# OpenMP Parallel Reduction Code Explanation

```
    cout << "\nEnter total number of elements: ";
    cin >> n;

    if (n <= 0) {
        cerr << "Error: Array size must be positive" << endl;
        return 1;
    }

    arr = new int[n];
    cout << "\nEnter elements:\n";
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

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

    delete[] arr;
    return 0;
}
```

- Accepts number of elements and values.
- Calls each function to show min, max, sum, and average.
- Releases memory with `delete[]`.

Compile and Run

```bash
```

OpenMP Parallel Reduction Code Explanation

```
g++ -fopenmp -o parallel_reduction 5_Min_Max.cpp
./parallel_reduction
```