**I'd be glad to explain the code in detail:**

**Header Inclusions:**

* **#include <iostream>: This line includes the iostream header file, which provides essential functionalities for input (using cin) and output (using cout).**
* **#include <omp.h>: This line includes the omp.h header file, which is necessary for using OpenMP directives for parallel programming.**

**Namespace Declaration:**

* **using namespace std;: This statement introduces the std namespace into the current scope, allowing you to use objects like cin, cout, and OpenMP constructs without explicitly prefixing them with std::.**

**Function Definitions:**

* **minval(int arr[], int n):**
  + **Takes an integer array arr and its size n as input.**
  + **Initializes minval to the first element of the array arr[0].**
  + **Uses the OpenMP directive #pragma omp parallel for reduction(min : minval):**
    - **parallel for: Creates a team of threads and distributes the loop iterations across them in parallel.**
    - **reduction(min : minval): Applies a reduction operation to find the minimum value. Here, the minimum of minval across all threads is identified.**
  + **Iterates through the array using a for loop (0 to n-1).**
  + **Inside the loop:**
    - **Checks if the current element arr[i] is less than the current minimum minval.**
    - **If so, updates minval with the smaller value.**
  + **Returns the final minimum value found.**
* **maxval(int arr[], int n): This function is very similar to minval, but it uses the reduction(max : maxval) clause to find the maximum element in the array using parallel processing.**
* **sum(int arr[], int n):**
  + **Takes an integer array arr and its size n as input.**
  + **Initializes sum to 0.**
  + **Uses the #pragma omp parallel for reduction(+ : sum) directive:**
    - **Parallelizes the loop for faster summation.**
    - **The reduction(+ : sum) clause accumulates the partial sums from each thread into the final sum.**
  + **Iterates through the array (0 to n-1).**
  + **Inside the loop, adds the current element arr[i] to the running total sum.**
  + **Returns the final sum of all elements.**
* **average(int arr[], int n):**
  + **Takes an integer array arr and its size n as input.**
  + **Casts the result of sum(arr, n) to a double to ensure floating-point division for calculating the average.**
  + **Divides the sum by the number of elements n using type casting to double.**
  + **Returns the calculated average as a double.**

**Main Function:**

* **int main(): This is the entry point of the program.**
  + **Declares an integer variable n and initializes it to 5 (the size of the array).**
  + **Creates an integer array arr with values {1, 2, 3, 4, 5}.**
  + **Calls the minval function and prints the minimum value using cout.**
  + **Calls the maxval function and prints the maximum value using cout.**
  + **Calls the sum function and prints the sum of the elements using cout.**
  + **Calls the average function and prints the average value using cout.**
  + **Returns 0 to indicate successful program execution.**

**Key Points:**

* **This code demonstrates parallel processing using OpenMP directives to find the minimum, maximum, sum, and average of an array concurrently.**
* **The reduction clauses in the #pragma omp parallel for directives ensure that the final results (minimum, maximum, sum) are correctly computed across all threads.**
* **The code includes comments to explain its functionality.**