```cpp
//---------------------------------------------------------------------------
// 2C_Sum_of_Numbers.cpp
//---------------------------------------------------------------------------
#if (defined __linux__) || (defined _AIX) || (defined __APPLE__)
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#elif (defined _WIN32) || (defined _WIN64)
#include <conio.h>
#include <direct.h>
#endif

#include <mpi.h>
#include <iostream>
using namespace std;

#define N 10000

int main(int argc, char** argv) {
    int i, rank, size, start_number, end_number, borders[64][2];
    long unsigned sum = 0, total_sum = 0, accum[64];
    double time;
    MPI_Status status;

    MPI_Init(&argc, &argv);
    MPI_Barrier(MPI_COMM_WORLD);
    time = -MPI_Wtime();

    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    if (rank == 0)
        printf("\nThere are %d processes.\n", size);

    start_number = rank * N / size + 1;
    end_number = (rank + 1) * N / size;

    for (i = start_number; i <= end_number; i++)
        sum += i;

    if (rank != 0) {
        MPI_Send(&start_number, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);
        MPI_Send(&end_number, 1, MPI_INT, 0, 1, MPI_COMM_WORLD);
        MPI_Send(&sum, 1, MPI_INT, 0, 2, MPI_COMM_WORLD);
    }
    else {
        borders[0][0] = start_number;
        borders[0][1] = end_number;
        total_sum = accum[0] = sum;
        for (i = 1; i < size; i++) {
            MPI_Recv(&borders[i][0], 1, MPI_INT, i, 0, MPI_COMM_WORLD, &status);
            MPI_Recv(&borders[i][1], 1, MPI_INT, i, 1, MPI_COMM_WORLD, &status);
            MPI_Recv(&accum[i], 1, MPI_INT, i, 2, MPI_COMM_WORLD, &status);
            total_sum += accum[i];
        }
    }

    if (rank == 0) {
        for (i = 0; i < size; i++)
            cout << "\nProcess " << i << ": The sum of numbers from " << borders[i][0]
            << " to " << borders[i][1] << " is: " << accum[i];

        cout << "\n\nThe sum of numbers from 1 to " << N << " is: " << total_sum;
    }
    time += MPI_Wtime();
    if (rank == 0)
        printf("\n\nTime = %10.6f s\n", time);

    fflush(stdout);
    MPI_Finalize();

    return 0;
}
```