Saint Petersburg National Research University of Information Technologies, Mechanics and Optics (ITMO University)

Report

**about laboratory works**

**Assignment 3**

**Assignment 4.**

**Assignment 5.**

**Student** Pogrebnoy D.A. j4132c

Saint-Petersburg, 2021

# Assignment 3.

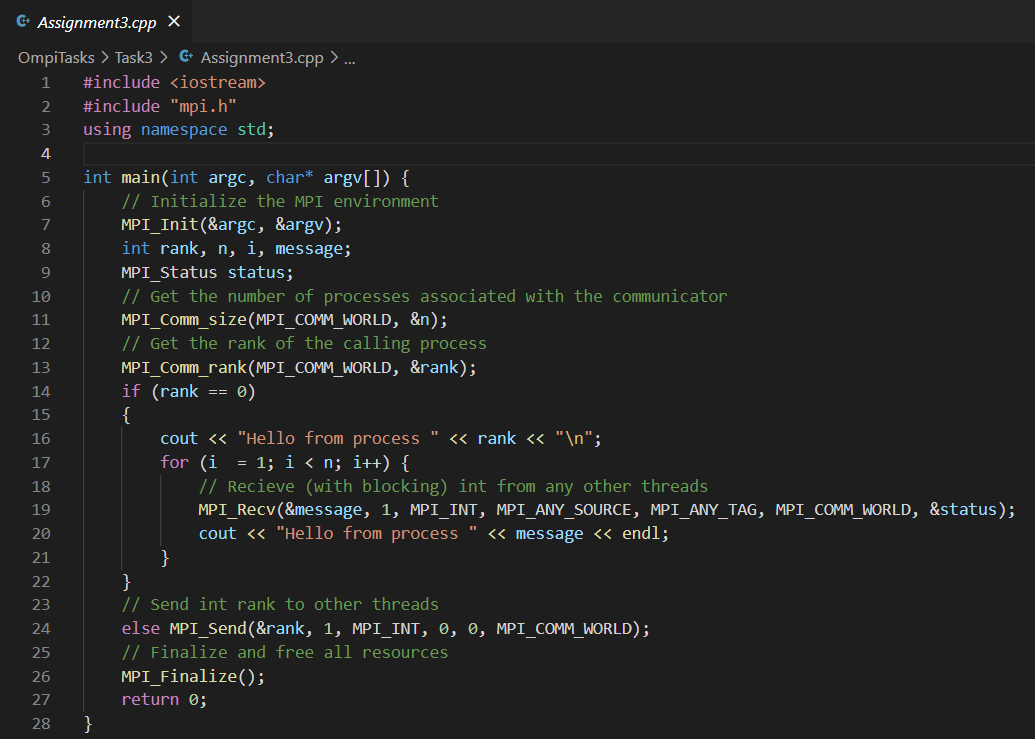
**Task**

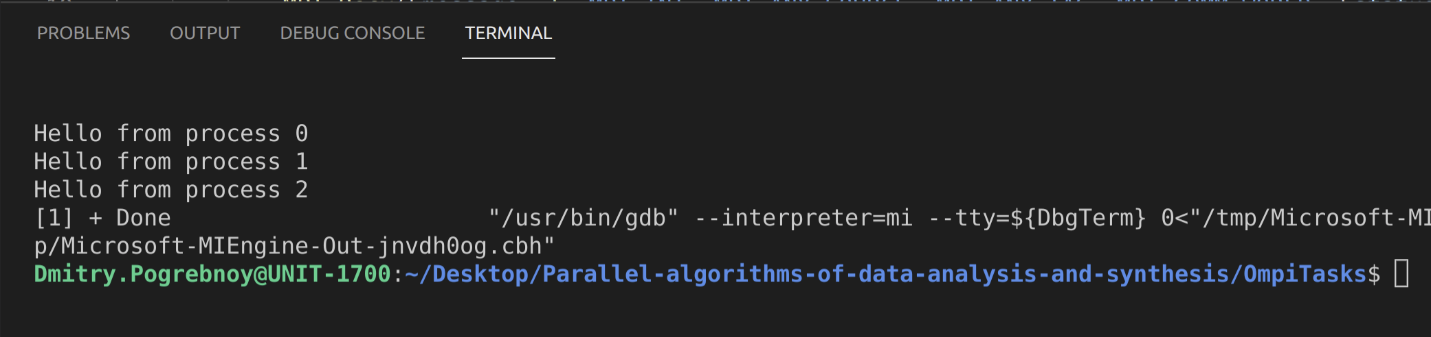
Compile and run Assignment3.c program. Explain in detail how it works.

**Implementation**

Source code and data gathered are available on <https://github.com/DmitryPogrebnoy/Parallel-algorithms-of-data-analysis-and-synthesis/blob/master/OmpiTasks/Task3/Assignment3.cpp>

The description of the code is described in the comments. The main process waits for messages to be received, and the other processes send messages to it with their thread number.



Output example with 3 processes: 

# Assigments 4.

**Task**

Convert the code Assignment4.c to match your individual version of the assignment.

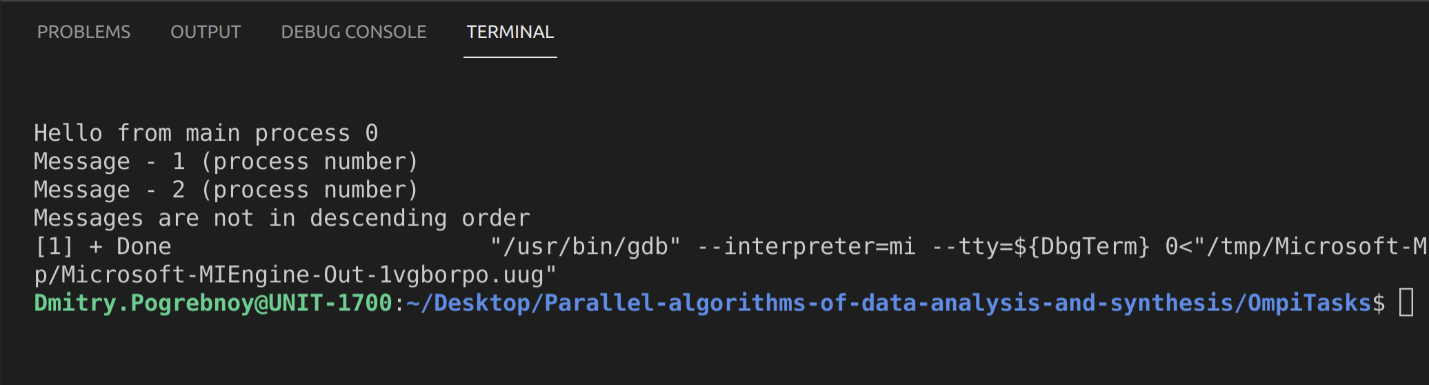
Option #21. The root process accepts messages from child processes and determines whether the sequence is strictly descending.

**Implementation**

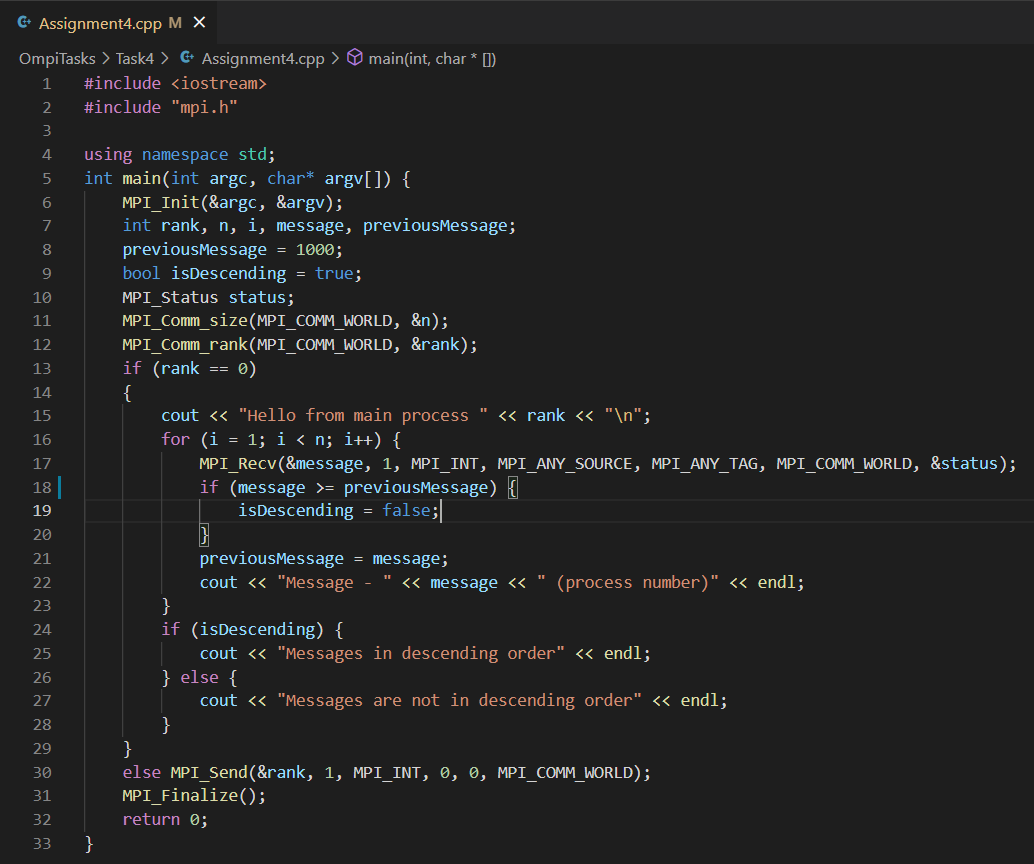
Source code and data gathered are available on <https://github.com/DmitryPogrebnoy/Parallel-algorithms-of-data-analysis-and-synthesis/blob/master/OmpiTasks/Task4/Assignment4.cpp>

The main process saves the previous message and compares it with the new one, if the order is broken, then the corresponding flag is set. And at the end, the main process outputs the corresponding message.

Output example with 3 processes:



The code looks like this:



# Assignment 5.

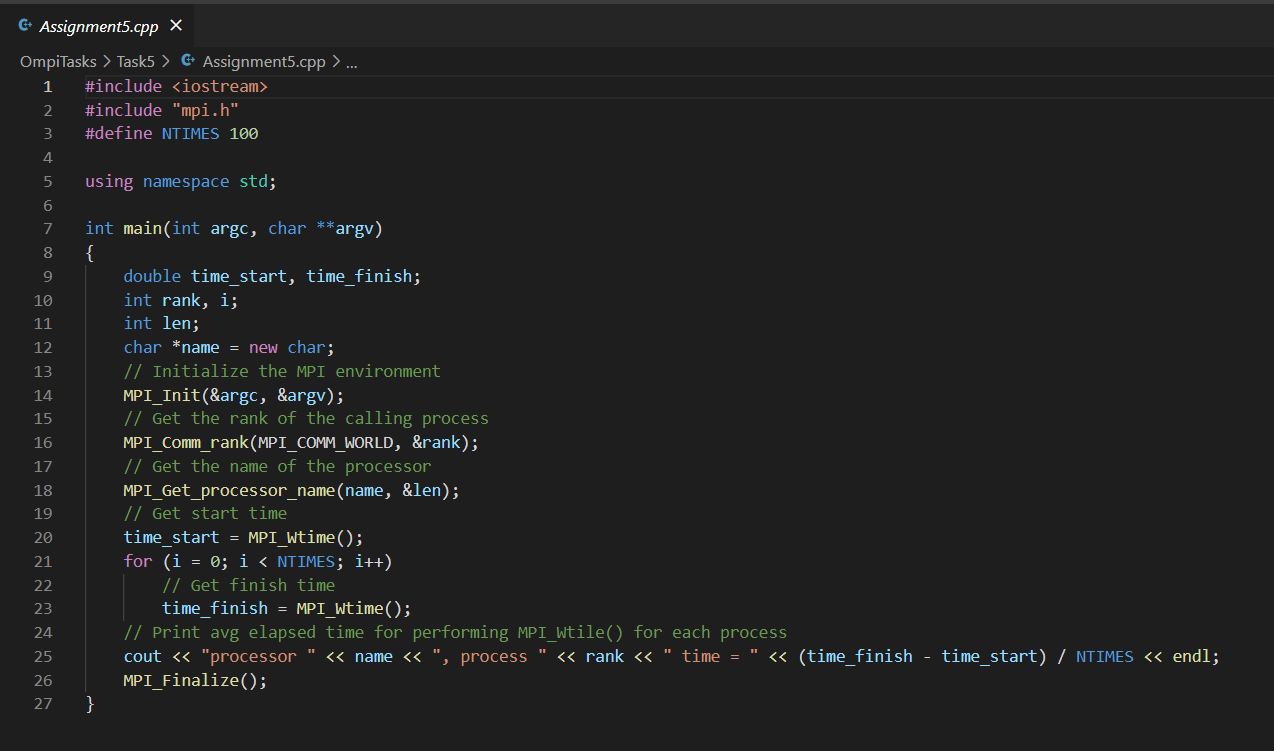
**Task**

Compile and run Assignment5.c program. Explain in detail how it works. Determine the execution time of the program from the previous task.

**Implementation**

Source code and data gathered are available on <https://github.com/DmitryPogrebnoy/Parallel-algorithms-of-data-analysis-and-synthesis/blob/master/OmpiTasks/Task5> .

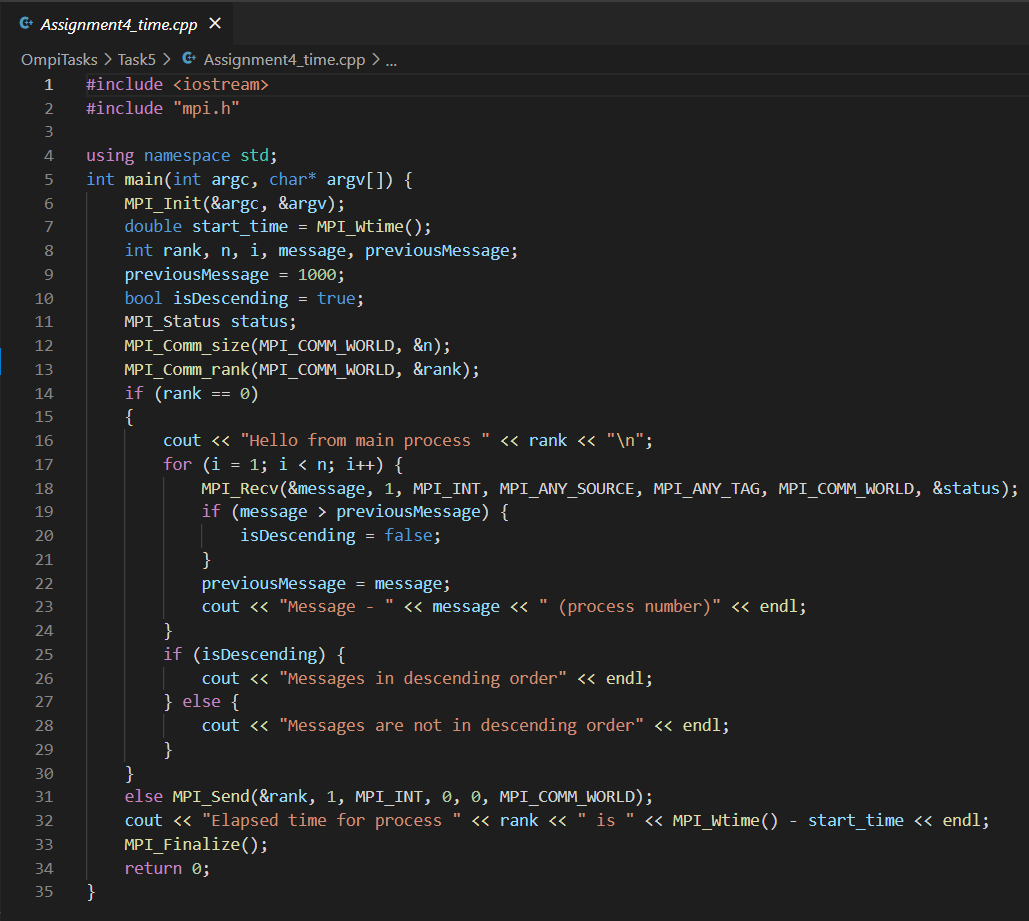
The description of the code is described in the comments. In each process, the time measurement is called 100 times using MPI\_Wtime() and the average time value is output.



Output example with 3 processes:



Assignment4 with time measurement looks like this:



Output example with 3 processes:

