МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра вычислительной техники

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Параллельные вычисления»
Тема: «Передача данных по процессам»

Студентка гр. 1307 _____ Грунская Н.Д.

Преподаватель _____ Манжиков Л.П.

Санкт-Петербург
2025

Цель работы.

Освоить функции передачи данных между процессами.

Задание 1.

3) Заменить в прямоугольной матрице все положительные элементы на 1, отрицательные на -1;

Задание 2.

В полученной матрице (по результатам выполнения задания 1) найти:

3) Количество элементов, оставшихся неизменными в новой матрице.

Для выполнения задания написан код, в константы которого вписаны размеры матрицы.

Программа не меняет 0, так как он является неположительным и неотрицательным.

Текст программы task1.cpp.

```
#include <stdio.h>
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>
#include "mpi.h"

#define ROWS 5
#define COLUMNS 6

int main(int argc, char **argv)
{
   int rank, size;
   int unchanged_count = 0;

   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   MPI_Comm_size(MPI_COMM_WORLD, &size);

   int matrix[ROWS][COLUMNS];
```

```
if (rank == 0)
{
    srand(time(NULL));
    std::cout << "Old matrix:\n";
    for (int i = 0; i < ROWS; i++)
    {
        for (int j = 0; j < COLUMNS; j++)
        {
            matrix[i][j] = (rand() % 21) - 10;
            std::cout << matrix[i][j] << " ";
        }
        std::cout << "\n";
    }
}

int *send_counts = new int[size];
int *displs = new int[size];

int rows_per_process = ROWS / size;
int remainder = ROWS % size;

for (int i = 0; i < size; i++)
{
    send_counts[i] = (i < remainder) ? (rows_per_process + 1) * COLUMNS : rows_per_process * COLUMNS;
    displs[i] = (i == 0) ? 0 : displs[i - 1] + send_counts[i - 1];
}

int local_rows = (rank < remainder) ? rows_per_process + 1 : rows_per_process;
int local_elements = local_rows * COLUMNS;
int local_matrix[local_rows][COLUMNS];
MPI_Scatterv(matrix, send_counts, displs, MPI_INT, local_matrix, local_elements, MPI_INT, 0, MPI_COMM_WORLD);

int local_unchanged_count = 0;
for (int i = 0; i < local_rows; i++)
{
    for (int j = 0; j < COLUMNS; j++)
    {
        if (local_matrix[i][j] == 0)
        {
            local_unchanged_count++;
```

```
            }
            if (local_matrix[i][j] > 0)
            {
                if (local_matrix[i][j] == 1)
                {
                    local_unchanged_count++;
                }
                local_matrix[i][j] = 1;
            }
            else if (local_matrix[i][j] < 0)
            {
                if (local_matrix[i][j] == -1)
                {
                    local_unchanged_count++;
                }
                local_matrix[i][j] = -1;
            }
        }
    }
    MPI_Gatherv(local_matrix, local_elements, MPI_INT, matrix, send_counts, displs, MPI_INT, 0, MPI_COMM_WORLD);

    int global_unchanged_count = 0;
    MPI_Reduce(&local_unchanged_count, &global_unchanged_count, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);

    if (rank == 0)
    {
        std::cout << "\nNew matrix:\n";
        for (int i = 0; i < ROWS; i++)
        {
            for (int j = 0; j < COLUMNS; j++)
            {
                std::cout << matrix[i][j] << " ";
            }
            std::cout << "\n";
        }
        std::cout << "\nNumber of unchanged elements: " << global_unchanged_count << std::endl;
    }

    MPI_Finalize();
    return 0;
}
```

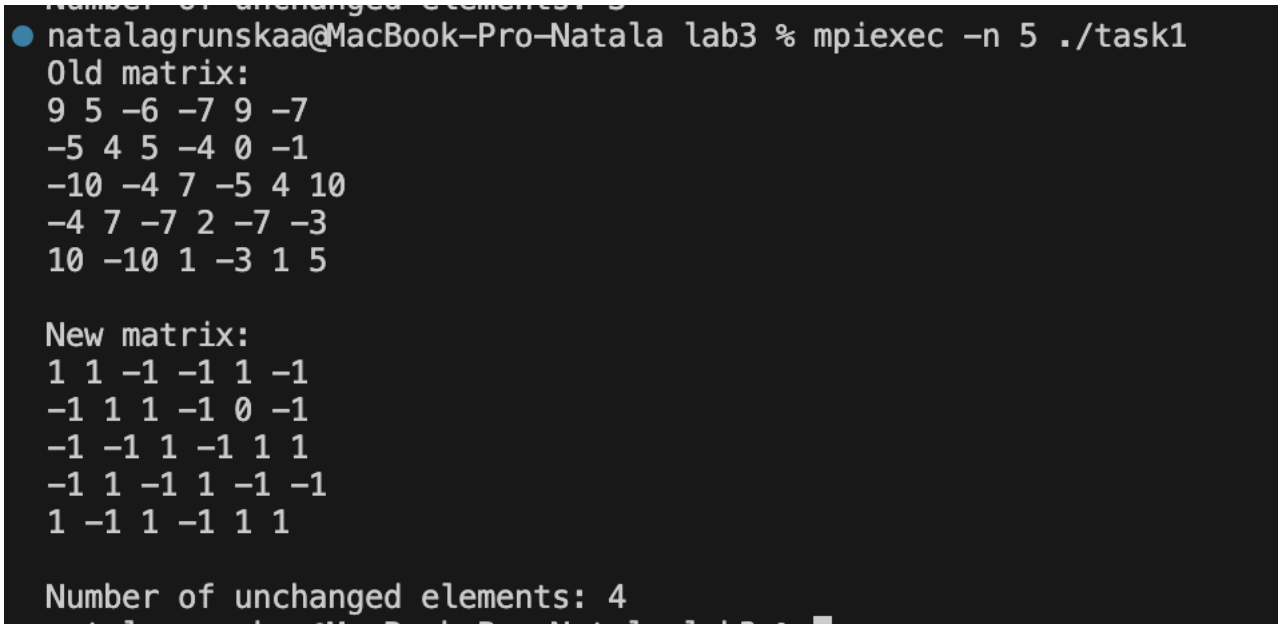

Рисунок 1. Запуск программы на 5-ти процессах.

Выводы.

В результате выполнения лабораторной работы были освоены основные концепции и функции MPI, необходимые для реализации параллельных программ, включая инициализацию, определение количества и рангов процессов, отправку и прием данных, распределение вычислений и сбор результатов. Были получены практические навыки работы с базовыми функциями MPI для построения параллельных приложений, что является фундаментом для разработки более сложных параллельных алгоритмов. Также получен опыт разработки, тестирования и отладки параллельных программ.