**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];

int modified\_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";

}

}

for (int i = 0; i < ROWS; ++i)

{

MPI\_Bcast(matrix[i], COLUMNS, MPI\_INT, 0, MPI\_COMM\_WORLD);

}

for (int i = 0; i < ROWS; i++)

{

for (int j = 0; j < COLUMNS; j++)

{

modified\_matrix[i][j] = matrix[i][j];

}

}

int local\_unchanged\_count = 0;

for (int i = rank; i < ROWS; i = i + size)

{

for (int j = 0; j < COLUMNS; j++)

{

if (matrix[i][j] == 0)

{

local\_unchanged\_count++;

}

if (modified\_matrix[i][j] > 0)

{

if (matrix[i][j] == 1)

{

local\_unchanged\_count++;

}

modified\_matrix[i][j] = 1;

}

else if (modified\_matrix[i][j] < 0)

{

if (matrix[i][j] == -1)

{

local\_unchanged\_count++;

}

modified\_matrix[i][j] = -1;

}

}

}

for (int i = 0; i < ROWS; ++i)

{

MPI\_Bcast(modified\_matrix[i], COLUMNS, MPI\_INT, i % size, 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 << modified\_matrix[i][j] << " ";

}

std::cout << "\n";

}

std::cout << "\nNumber of unchanged elements: " << global\_unchanged\_count << std::endl;

}

MPI\_Finalize();

return 0;

}

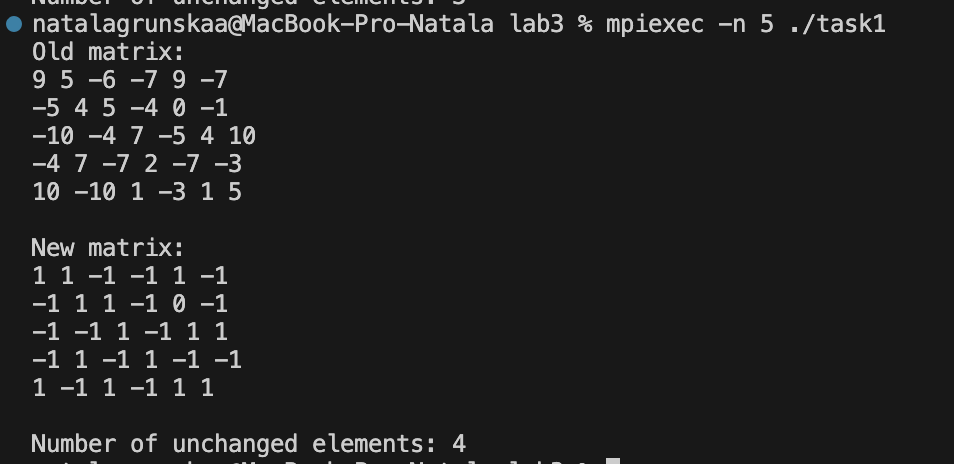


Рисунок 1. Запуск программы на 5-ти процессах.

**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.