Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Алтайский государственный технический университет

им. И.И. Ползунова»

Факультет (институт) Информационных технологий

Кафедра Прикладная математика

Отчет защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_

А.И.Потупчик

(подпись преподавателя) (инициалы, фамилия)

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Отчет

по лабораторной (практической) работе №\_\_10\_\_

Использование модели MPI

(название лабораторной (практической) работы)

по дисциплине Архитектура ЭВМ

(наименование дисциплины)

ЛР 09.03.04.21.000 ОТ

(обозначение документа)

Студент группы ПИ-02 Р.А Чередов

(инициалы, фамилия)

Преподаватель доцент, доцент А.И.Потупчик

(должность, ученое звание) (инициалы, фамилия)

Барнаул 2022

Задание:

**21. Дан двумерный массив b. Рассматривая его как матрицу, транспонировать b.**

Решение:

#include <iostream>

#include <mpi.h>

#include <deque>

using namespace std;

// Функция заполнения матрицы

void input(int \*arr, int row,int column) {

for (int i = 0; i < row \* column; ++i) {

arr[i] = i+1;

}

}

// Функция транспонирования матрицы

void Trans(int\* arr, int &row, int &column) {

int temp[100][100];

int a;

// Копирование данных

for (int i = 0; i < row; ++i) {

for (int j = 0; j < column; ++j) {

temp[i][j] = arr[i\* column + j];

}

}

// Транспонирование

for (int i = 0; i < row; ++i) {

for (int j = 0; j < column; ++j) {

arr[i + j \* row] = temp[i][j];

}

}

// row и column меняются местами

a = row;

row = column;

column = a;

}

// Вывод матрицы в консоль

void print(int\* arr, int row, int column) {

for (int i = 0; i < row; ++i) {

for (int j = 0; j < column; ++j) {

cout << arr[i \* column + j] << "\t";

}

cout << endl;

}

cout << endl;

}

int main(int\* argc, char\*\* argv) {

int numRow = 4; // Количество строк в матрице

int numColumn = 2; // Количество столбцов в матрице

const int size = 4; // Необходимо чтобы size всегда был больше или равен чем numRow и numColumn.

// Ограничение size = 100 (если нужно больше меняй в функции Trans размер temp)

double t1, t2, dt; // Переменные для хранения времени выполнения

int numtasks, rank;

int matrix[size][size]; // Матрица

// Обязательные MPI компоненты

MPI\_Init(argc, &argv);

MPI\_Status status;

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &numtasks);

t1 = MPI\_Wtime(); // Запуск таймера

// Процесс 0

// Заполняет матрицу значениями и отправляет её процессу 1 и 2

if (rank == 0) {

input(\*matrix, numRow, numColumn); // Заполнение матрицы

MPI\_Send(matrix, numRow \* numColumn, MPI\_INT, 1, 1, MPI\_COMM\_WORLD); // Матрица отправляется 1 процессу

MPI\_Send(matrix, numRow \* numColumn, MPI\_INT, 2, 2, MPI\_COMM\_WORLD); // Матрица отправляется 2 процессу

t2 = MPI\_Wtime(); // Конец таймера

dt = t2 - t1; // Расчитываем время работы области программы

MPI\_Send(&dt, 1, MPI\_DOUBLE, 4, 6, MPI\_COMM\_WORLD); // Время выполнения работы процесса отправляется 4 процессу

}

// Процесс 1

// Печатает матрицу в консоль

else if (rank == 1) {

MPI\_Recv(matrix, numRow \* numColumn, MPI\_INT, 0, 1, MPI\_COMM\_WORLD, &status); // Принимает заполненую матрицу от процесса 0

print(\*matrix, numRow, numColumn); // Вывод матрицы в консоль

t2 = MPI\_Wtime(); // Конец таймера

dt = t2 - t1; // Расчитываем время работы области программы

MPI\_Send(&dt, 1, MPI\_DOUBLE, 4, 7, MPI\_COMM\_WORLD); // Время выполнения работы процесса отправляется 4 процессу

}

// Процесс 2

// Транспонирует матрицу и отправляет её процессу 3

// Так же отправляет новые значение строк (row) и колонок (column) процессу 3

else if (rank == 2) {

MPI\_Recv(matrix, numRow \* numColumn, MPI\_INT, 0, 2, MPI\_COMM\_WORLD, &status); // Принимает заполненую матрицу от процесса 0

Trans(\*matrix, numRow, numColumn); // Транспонирование матрицы

MPI\_Send(matrix, numRow \* numColumn, MPI\_INT, 3, 3, MPI\_COMM\_WORLD); // Отправляет транспонированную матрицу процессу 3

MPI\_Send(&numRow, 1, MPI\_INT, 3, 4, MPI\_COMM\_WORLD); // Отправляет новое значение строк процессу 3

MPI\_Send(&numColumn, 1, MPI\_INT, 3, 5, MPI\_COMM\_WORLD); // Отправляет новое значение колонок процессу 3

t2 = MPI\_Wtime(); // Конец таймера

dt = t2 - t1; // Расчитываем время работы области программы

MPI\_Send(&dt, 1, MPI\_DOUBLE, 4, 8, MPI\_COMM\_WORLD); // Время выполнения работы процесса отправляется 4 процессу

}

// Процесс 3

// Печатает матрицу в консоль

else if (rank == 3) {

MPI\_Recv(&numRow, 1, MPI\_INT, 2, 4, MPI\_COMM\_WORLD, &status); // Принимает новое значение строк от процесса 2

MPI\_Recv(&numColumn, 1, MPI\_INT, 2, 5, MPI\_COMM\_WORLD, &status); // Принимает новое значение строк от процесса 2

MPI\_Recv(matrix, numRow \* numColumn, MPI\_INT, 2, 3, MPI\_COMM\_WORLD, &status); // Принимает заполненую матрицу от процесса 2

print(\*matrix, numRow, numColumn); // Вывод матрицы в консоль

t2 = MPI\_Wtime(); // Конец таймера

dt = t2 - t1; // Расчитываем время работы области программы

MPI\_Send(&dt, 1, MPI\_DOUBLE, 4, 9, MPI\_COMM\_WORLD); // Время выполнения работы процесса отправляется 4 процессу

}

// Процесс 4

// Печатает время работы каждого процесса и общее количество запущенных процессов

if (rank == 4) {

double time[4];

MPI\_Recv(&time[0], 1, MPI\_DOUBLE, 0, 6, MPI\_COMM\_WORLD, &status); // Принимает время работы процесса 0

MPI\_Recv(&time[1], 1, MPI\_DOUBLE, 1, 7, MPI\_COMM\_WORLD, &status); // Принимает время работы процесса 1

MPI\_Recv(&time[2], 1, MPI\_DOUBLE, 2, 8, MPI\_COMM\_WORLD, &status); // Принимает время работы процесса 2

MPI\_Recv(&time[3], 1, MPI\_DOUBLE, 3, 9, MPI\_COMM\_WORLD, &status); // Принимает время работы процесса 3

// Печатает время каждого принятого процесса

for (int i = 0; i < 4; ++i) {

cout << "RunTime MPI Process " << i << " = " << time[i] << endl;

}

cout << "Total number of process: " << numtasks << endl << endl;

}

MPI\_Finalize(); // Обязательные MPI компонент

}

Тесты программы:





