МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №14

РАБОТА ЗАЩИЩЕНА С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

старший преподаватель		А. Ю. Сыщиков
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1
MPI, Пересылка данных

по дисциплине: Системы с параллельной обработкой информации

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. 1742 _____ В.А. Седов

подпись, дата — инициалы, фамилия

Санкт-Петербург 2021

1. Цель работы

Переслать вектор, размерности M, N процессам, используя различные виды связи между процессами. Элементы вектора задаются произвольно. Элементы вектора пересылаемого и принятого вектора, а также время выполнения должны быть выведены на экран.

№ варианта	M	N	Функция
16	10	3	MPI_Alltoall

2. Текст программы

```
#include "mpi.h"
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define M 2
static int gsize, myid;
void initArrayNew(int* a) {
	for (int i = 0; i < M; i++) {
		a[i] = rand() % 7 + (myid % 3 + 1) + rand() % 20 * (myid %2)+ (myid % 3
+1);
	}
}
void printArray(int* a, int R) {
	printf("[ ");
	for (int i = 0; i < R; i++) {
		printf("%d ", a[i]);
	}
	printf("]\n");
}
int main(int argc, char* argv[]) {
	int* rbuf;
	int* sendbuf;
	int namelen;
	double startwtime = 0.0, endwtime = 0.0;
	char processor_name[MPI_MAX_PROCESSOR_NAME];
	MPI_Init(&argc, &argv);
	MPI_Comm_size(MPI_COMM_WORLD, &gsize);
```

```
	MPI_Comm_rank(MPI_COMM_WORLD, &myid);
	MPI_Get_processor_name(processor_name, &namelen);
	//gsize = 9;
	startwtime = MPI_Wtime();
	sendbuf = (int*)malloc(M * sizeof(int));
	initArrayNew(sendbuf);
	rbuf = (int*)malloc(M * gsize * sizeof(int));
	printf("Process %d on %s\n", myid, processor_name);
	printf("Massiv: ");
	printArray(sendbuf, M);
	fflush(stdout);
	MPI_Allgather(sendbuf, M, MPI_INT, rbuf, M, MPI_INT, MPI_COMM_WORLD);
	//printf("id = %d, sendbuf = %d\n", myid, sendbuf[myid]);
	//printf("rbuf = %d\n", rbuf[0]);
	printf("Process %d on %s:", myid, processor_name);
	printArray(rbuf, M*gsize);
	fflush(stdout);
	if (myid == 0) {
		endwtime = MPI_Wtime();
		fprintf(stderr, "wall clock time = %f\n", endwtime - startwtime);
	}
	MPI_Finalize();
	return 0;
}
```

3. Результат работы программы

```
C:\Users\Admin\source\repos\MPITest\Debug>mpiexec.exe -n 4 MPITest.exe
Process 0 on Danila.MYDNS
Massiv: [ 8 8 ]
Process 1 on Danila.MYDNS
Massiv: [ 17 10 ]
Process 3 on Danila.MYDNS
Massiv: [ 15 8 ]
Process 2 on Danila.MYDNS
Massiv: [ 12 12 ]
Process 2 on Danila.MYDNS:[ 8 8 17 10 12 12 15 8 ]
Process 3 on Danila.MYDNS:[ 8 8 17 10 12 12 15 8 ]
Process 0 on Danila.MYDNS:[ 8 8 17 10 12 12 15 8 ]
Process 1 on Danila.MYDNS:[ 8 8 17 10 12 12 15 8 ]
wall clock time = 0.000576

C:\Users\Admin\source\repos\MPITest\Debug>
```

Рисунок 1. Результат работы программы.