

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

ОТЧЕТ
по лабораторной работе №2
по дисциплине «Параллельные вычисления»
Тема: «Передача данных по процессам»

Студентка гр. 1307

Грунская Н.Д.

Преподаватель

Манжиков Л.П.

Санкт-Петербург

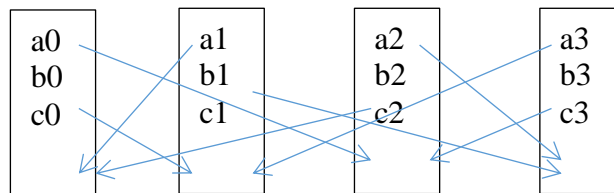
2025

Цель работы.

Освоить функции передачи данных между процессами.

Задание 1.

- 1) Запустить 4 процесса.
- 2) На каждом процессе создать переменные: a_i, b_i, c_i , где i – номер процесса. Инициализировать переменные. Вывести данные на печать.
- 3) Передать данные на другой процесс. Напечатать номера процессов и поступившие данные. Найти: $c_0 = a_1 + b_2$; $c_1 = a_3 + b_0$; $c_2 = a_0 + b_3$; $c_3 = a_2 + b_1$.



Текст программы task1.cpp.

```
#include <cstdlib>
#include <iostream>
#include <ctime>
#include "mpi.h"
#define MAX 10
using namespace std;

int main(int argc, char *argv[])
{
    const int tag = 198;
    int rank, size, n, i, ibeg, iend, ai, bi, ci;
    MPI_Status status;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    srand(rank);
    ai = (rand() % 100) + 1;
    bi = (rand() % 100) + 1;
```

```

printf("rank = %d, a%d = %d, b%d = %d\n", rank,rank,ai,rank,bi);
printf("Sending: a%d = %d, b%d = %d\n",rank,ai,rank,bi);
switch (rank){
    case 0:
    {
        MPI_Send(&bi, 1, MPI_INT, 1, tag, MPI_COMM_WORLD);
        MPI_Send(&ai, 1, MPI_INT, 2, tag, MPI_COMM_WORLD);
        MPI_Recv(&ai, 1, MPI_INT, 1, tag, MPI_COMM_WORLD,&status);
        MPI_Recv(&bi, 1, MPI_INT, 2, tag, MPI_COMM_WORLD,&status);
        break;
    }
    case 1:
    {
        MPI_Send(&ai, 1, MPI_INT, 0, tag, MPI_COMM_WORLD);
        MPI_Send(&bi, 1, MPI_INT, 3, tag, MPI_COMM_WORLD);
        MPI_Recv(&ai, 1, MPI_INT, 3, tag, MPI_COMM_WORLD,&status);
        MPI_Recv(&bi, 1, MPI_INT, 0, tag, MPI_COMM_WORLD,&status);
        break;
    }
    case 2:
    {
        MPI_Send(&ai, 1, MPI_INT, 3, tag, MPI_COMM_WORLD);
        MPI_Send(&bi, 1, MPI_INT, 0, tag, MPI_COMM_WORLD);
        MPI_Recv(&ai, 1, MPI_INT, 0, tag, MPI_COMM_WORLD,&status);
        MPI_Recv(&bi, 1, MPI_INT, 3, tag, MPI_COMM_WORLD,&status);
        break;
    }
    case 3:
    {
        MPI_Send(&ai, 1, MPI_INT, 1, tag, MPI_COMM_WORLD);
        MPI_Send(&bi, 1, MPI_INT, 2, tag, MPI_COMM_WORLD);
        MPI_Recv(&ai, 1, MPI_INT, 2, tag, MPI_COMM_WORLD,&status);
        MPI_Recv(&bi, 1, MPI_INT, 1, tag, MPI_COMM_WORLD,&status);
        break;
    }
}

```

```

}

printf("Received: a = %d, b = %d\n",ai,bi);

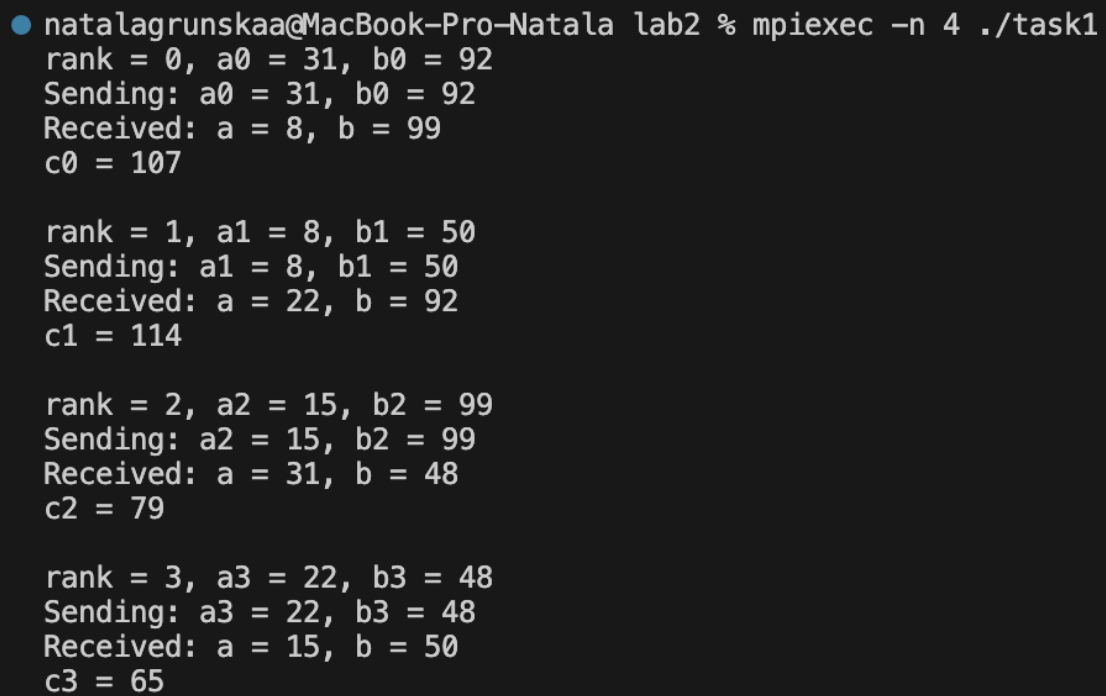
printf("c%d = %d\n\n",rank,ai+bi);

MPI_Finalize();

return 0;

}

```



```

● natalagrunskaa@MacBook-Pro-Natala lab2 % mpiexec -n 4 ./task1
rank = 0, a0 = 31, b0 = 92
Sending: a0 = 31, b0 = 92
Received: a = 8, b = 99
c0 = 107

rank = 1, a1 = 8, b1 = 50
Sending: a1 = 8, b1 = 50
Received: a = 22, b = 92
c1 = 114

rank = 2, a2 = 15, b2 = 99
Sending: a2 = 15, b2 = 99
Received: a = 31, b = 48
c2 = 79

rank = 3, a3 = 22, b3 = 48
Sending: a3 = 22, b3 = 48
Received: a = 15, b = 50
c3 = 65

```

Рисунок 1. Запуск программы на 4-х процессах.

Задание 2.

Запустить n процессов и найти по вариантам:

1. Сумму нечетных элементов вектора;
2. Минимальный элемент;
- 3. Максимальный элемент;**
4. Среднее арифметическое элементов вектора;
5. Сумму элементов кратных 3;
6. Количество четных положительных элементов;
7. Максимальный элемент среди отрицательных чисел;
8. Минимальный элемент среди положительных чисел;
9. Сумму элементов из заданного пользователем диапазона.

Текст программы task2.cpp

//3. Максимальный элемент;

```
#include <cstdlib>
#include <iostream>
#include <ctime>
#include "mpi.h"
using namespace std;

int main(int argc, char *argv[])
{
    int rank, size;
    MPI_Status status;
    int n= 20;
    int vector[n];
    srand(time(NULL));
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    int local_max=0, max=0;
    if (rank==0){
        for (int i=0;i<n;i++){
            vector[i]= (rand() % 100);
            printf("%d ",vector[i]);
        }
        printf("\n");
    }
    MPI_Bcast(&vector,n,MPI_INT,0,MPI_COMM_WORLD);
    for (int i=rank;i<n;i=i+size){
        if(vector[i]> local_max){
            local_max = vector[i];
        }
    }
    MPI_Reduce(&local_max,&max,1,MPI_INT,MPI_MAX,0, MPI_COMM_WORLD);
    if(rank==0){
        printf("max = %d\n",max);
    }
    MPI_Finalize();
    return 0;
}
```

```

● natalagrunskaa@MacBook-Pro-Natala lab2 % mpic++ task2New.cpp -o task2New
● natalagrunskaa@MacBook-Pro-Natala lab2 % mpiexec -n 8 ./task2New
76 28 57 68 47 62 9 55 83 22 65 29 71 75 14 77 88 54 58 71
max = 88
● natalagrunskaa@MacBook-Pro-Natala lab2 % mpiexec -n 2 ./task2New
4 24 8 6 73 3 85 73 34 70 78 95 98 96 62 95 55 70 71 72
max = 98
● natalagrunskaa@MacBook-Pro-Natala lab2 % mpiexec -n 10 ./task2New
31 79 27 66 8 50 93 61 14 69 4 59 12 23 93 83 26 92 97 34
max = 97
● natalagrunskaa@MacBook-Pro-Natala lab2 %

```

Рисунок 2. Запуск программы на 2-х, 8-и, 10-и 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.