

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

**ОТЧЕТ**  
**по лабораторной работе №1**  
**по дисциплине «Параллельные вычисления»**  
**Тема: «Запуск параллельной программы»**

Студентка гр. 1307

\_\_\_\_\_

Грунская Н.Д.

Преподаватель

\_\_\_\_\_

Манжиков Л.П.

Санкт-Петербург

2025

### **Цель работы.**

Освоить процесс запуска программы на C++ с применением библиотеки MPICH2. Научиться получать сведения о количестве запущенных процессов и номере отдельного процесса.

**Задание 1.** Создать и запустить программу на 2-х процессах с применением функций `int MPI_Init( int* argc, char*** argv)` и `int MPI_Finalize( void )`.

Текст программы task1.cpp.

```
#include <stdio.h>
#include "mpi.h"
#include <sys/types.h>
#include <unistd.h>

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

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    printf("Size:%i, Process rank:%i, Start \n", size,rank);
    printf("Size:%i, Process rank:%i, End \n", size,rank);
    MPI_Finalize();

    return 0;
}
```

```

● natalagrunskaa@MacBook-Pro-Natala lab1 % mpic++ test.cpp -o test
● natalagrunskaa@MacBook-Pro-Natala lab1 % mpiexec -n 5 ./test
Size:5, Process rank:0, Start
Size:5, Process rank:0, End
Size:5, Process rank:1, Start
Size:5, Process rank:1, End
Size:5, Process rank:2, Start
Size:5, Process rank:2, End
Size:5, Process rank:3, Start
Size:5, Process rank:3, End
Size:5, Process rank:4, Start
Size:5, Process rank:4, End
○ natalagrunskaa@MacBook-Pro-Natala lab1 %

```

Рисунок 1. Запуск программы на 5-х процессах.

**Задание 2.** Создать и запустить программу на 3-х процессах с применением функций:

- 1) `int MPI_Init( int* argc, char*** argv);`
- 2) `int MPI_Finalize( void );`
- 3) `int MPI_Comm_size( MPI_Comm comm, int* size)`
- 4) `int MPI_Comm_rank( MPI_Comm comm, int* rank)`

Программа должна выводить на экран номер процесса и какой-либо идентификатор процесса.

Текст программы task2.cpp

```

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

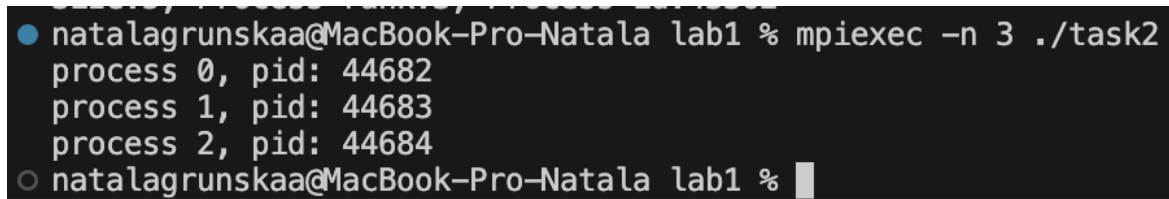
int main(int argc, char *argv[])
{
    int rank, size, n, i, ibeg, iend;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

```

```

printf ("process %d, pid: %d\n", rank, getpid());
MPI_Finalize();
return 0;
}

```



```

natalagrunskaa@MacBook-Pro-Natala lab1 % mpiexec -n 3 ./task2
process 0, pid: 44682
process 1, pid: 44683
process 2, pid: 44684
natalagrunskaa@MacBook-Pro-Natala lab1 %

```

Рисунок 2. Запуск программы на 3-х процессах

**Задание 3.** Создать и запустить программу на n-х процессах печати таблицы умножения.

Текст программы task3.cpp

```

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

int main(int argc, char *argv[])
{
    int rank, size, n, i, j, ibeg, iend;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    n=(MAX-1)/size+1;
    ibeg=rank*n+1;
    iend=(rank+1)*n;
    for(j=1; j<=10; j++){
        printf ("process %d, %d ^ %d = %d \n", rank, rank, j,rank*j);}
    printf ("\n");
    MPI_Finalize();
    return 0;
}

```

```

● natalagrunskaa@MacBook-Pro-Natala lab1 % mpiexec -n 5 ./task3
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40

```

```
9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 10 = 50
10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54 x 7 = 56
8 x 8 = 64
8 x 9 = 72
8 x 10 = 80
36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
5 = 50
10 x 6 = 60
10 x 7 = 70
10 x
6 x 10 = 60
21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
79 x 9 = 81
9 x 10 = 90
x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
8 = 80
10 x 9 = 90
10 x 10 = 100
natalagrunskaa@MacBook-Pro-Natala lab1 %
```

Рисунок 3. Запуск программы на 5-х процессах.

### Выводы.

Лабораторная работа позволила освоить базовые принципы параллельного программирования с использованием MPI. Полученные навыки инициализации процессов, распределения задач и управления ими являются фундаментом для разработки сложных распределенных приложений, требующих оптимизации вычислений за счет параллелизма.