МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное учреждение высшего образования
«Чувашский государственный университет И.Н. Ульянова»
Факультет информатики и вычислительной техники
Кафедра вычислительной техники

Параллельное программирование
Лабораторная работа 3
Выполнение заданий с 15 - 20

Выполнил:

Студент группы ИВТ-41-20
Галкин Д.С.

Проверил:

Ковалев С.В.

Чебоксары, 2024

Цель работы (Технология программирования в MPI):

Задание 15. Программа "I am!":

> ⓘ **Задание для выполнения**
>
> Напишите программу, в которой каждый процесс выводит на экран свой номер и общее количество процессов в приложении в формате:
>
> I am <Номер процесса> process from <Кол-во процессов> processes!

Входные данные: нет
Выходные данные: строка в формате "I am <Номер процесса> process from <количество процессов> processes!"

Входные данные	Выходные данные
3	I am 0 process from 3 processes! I am 1 process from 3 processes! I am 2 process from 3 processes!

Полный текст программы:

1. Интерфейс ILab (Используется для всех лабораторных работ)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ProgramMPI.Interfaces
{
    internal interface ILab
    {
        /// <summary>
        /// Задание лабораторной работы
        /// </summary>
        void ProccessStart(string[] args);

        /// <summary>
        /// Запуск лабораторной работы (Proccess)
        /// </summary>
        void Start();
    }
}
```

2. Класс Main

```
var lab1 = new Lab1("Lab1");
lab1.Start();
```

3. Класс Lab1

```
using MPI;
using ProgramMPI.Interfaces;

namespace ProgramMPI.Labs
{
    public class Lab1 : ILab
    {
        public string Name { get; set; }

        public string[] _args { get; set; }

        public Lab1 (string name, string[] args)
        {
            Name = name;
            _args = args;
        }

        public void ProccessStart(string[] args)
        {
            using (new MPI.Environment(ref args))
            {
                Intracommunicator comm = Communicator.world;

                if (comm.Rank == 0)
                {
                    Console.WriteLine(this);
                }

                Console.WriteLine($"I am {comm.Rank} process from {comm.Size}
processes!");
            }
        }

        public void Start()
        {
            ProccessStart(_args);

            Console.WriteLine("\n");
        }

        public override string ToString() => $"{Name} started classes:" ;

        private void Print(int rank, int size) => Console.WriteLine($"I am {rank}
proccess from {size} proccesses!");

    }
}
```

Результат:

```
Lab1 started classes:
I am 0 process from 3 processes!
I am 2 process from 3 processes!
I am 1 process from 3 processes!
```

Задание 2. Программа "На первый-второй рассчитайся!":

> **Задание для выполнения**
>
> Напишите программу, в которой каждый процесс с четным номером выводит на экран строку "I am <Номер процесса>: FIRST!"
>
> А каждый процесс с нечетным номером - "I am <Номер процесса>: SECOND!"
>
> Процесс с номером 0 должен вывести на экран общее количество процессов в приложении в формате "<Количество процессов> processes."

Входные данные: нет
Выходные данные: строки в формате "I am <Номер процесса>: SECOND!" или "I am <Номер процесса>: FIRST!" или "<Количество процессов> processes."

Входные данные	Выходные данные
4	4 processes. I am 0 process: FIRST! I am 1 process: SECOND! I am 2 process: FIRST! I am 3 process: SECOND!

Полный текст программы:

1. Класс Main

```
var lab2 = new Lab2("Lab2");
lab2.Start();
```

2. Класс Lab2

```
using MPI;
using ProgramMPI.Interfaces;

namespace ProgramMPI.Labs
{
    public class Lab2 : ILab
    {
        public string Name { get; set; }
        public string[] _args { get; set; }

        public Lab2(string name, string[] args)
        {
            Name = name;
            _args = args;
        }

        public void ProccessStart(string[] args)
        {
            using (new MPI.Environment(ref args))
            {
                Intracommunicator comm = Communicator.world;

                if (comm.Rank == 0)
                {
                    Console.WriteLine(this);
                    Console.WriteLine($"{comm.Size} processes.");
                }
                Print(comm.Rank, comm.Size);
            }
        }

        public void Start()
        {
            ProccessStart(_args);

            Console.WriteLine("\n");
        }

        public override string ToString() => $"{Name} started classes:";

        private void Print(int rank, int size) =>
            Console.WriteLine($"I am {rank} process: {(rank % 2 != 0 ? "FIRST!" :
"SECOND!")}");

    }
}
```

Результат:

```
Lab2 started classes:
4 processes.
I am 1 process: FIRST!
I am 0 process: SECOND!
I am 3 process: FIRST!
I am 2 process: SECOND!
```

Задание 3. Коммуникация "точка-точка" (простые блокирующие обмены):

> **Задание для выполнения**
>
> Напишите основные MPI-функции блокирующей передачи сообщений точка-точка **MPI_send** и **MPIRecy**. Напишите MPI-программу, в которой с помощью данных функций процесс с номером 0 отправляет сообщение процессу с номером 1. Процесс 1 выводит полученное сообщение на экран.

Входные данные: нет
Выходные данные: "receive message '<сообщение>'"

Входные данные	Выходные данные
	receive message '45'

Полный текст программы:

1. Класс Main

```
var lab3 = new Lab3("Lab3");
lab3.Start();
```

2. Класс Lab3

```
using MPI;
using ProgramMPI.Interfaces;

namespace ProgramMPI.Labs
{
    public class Lab3 : ILab
    {
        public string Name { get; set; }
        public string[] _args { get; set; }

        public Lab3(string name, string[] args)
        {
            Name = name;
            _args = args;
        }

        public void ProccessStart(string[] args)
        {
            using (new MPI.Environment(ref args))
            {
                Intracommunicator comm = Communicator.world;

                if (comm.Rank == 0)
                {
                    Console.WriteLine(this);
                    // Процесс с номером 0 отправляет сообщение процессу с номером 1
                    //string message = "Hello from process 0";
                    //byte[] messageBytes = Encoding.UTF8.GetBytes(message);
                    comm.Send(45, 1, 0); // Отправляем массив байтов процессу 1 с
тегом 0
                }
                else if (comm.Rank == 1)
                {
                    // Процесс с номером 1 получает сообщение от процесса с номером 0
                    int receivedBytes = comm.Receive<int>(0, 0); // Получаем массив
байтов от процесса 0 с тегом 0
                    //string receivedMessage = Encoding.UTF8.GetString(receivedBytes);
                    Print(receivedBytes); ;
                }
            }
        }

        public void Start()
        {
            ProccessStart(_args);

            Console.WriteLine("\n");
        }

        public override string ToString() => $"{Name} started classes:";

        private void Print(int receivedMessage) =>
            Console.WriteLine($"Received message: '{receivedMessage}'");
```

```
    }
}
```

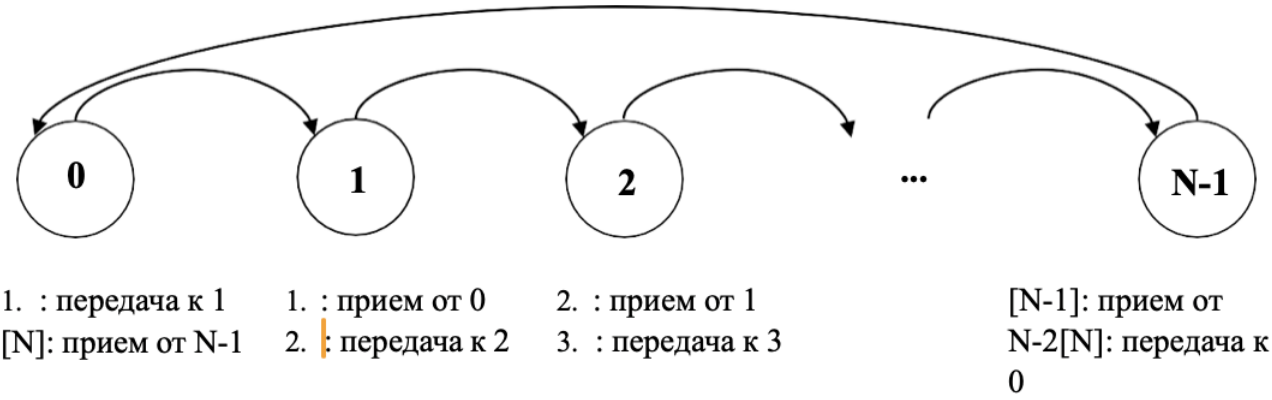
Результат:

```
Lab3 started classes:
Received message: '45'
```

Задание 4. Коммуникации "точка-точка" (схема "эстафетная палочка"):

Задание для выполнения

Напишите MPI-программу, реализующую при помощи блокирующих функций посылки сообщений типа точка-точка схему коммуникации процессов "эстафетная палочка", в которой каждый процесс дожидается сообщения от предыдущего и потом посылает следующему. В качестве передаваемого сообщения используйте на процесс 0 его номер, на остальных процессах - инкрементированное полученное сообщение.



- Процесс с номером X

[t] - Последовательность выполнения действия,где t – порядковый номером

N - Общее количество процессов

Входные данные: нет
Выходные данные: "[<номер_процесса>]: receive message <'сообщение'>"

Входные данные	Выходные данные
	1. : receive message '3' 2. : receive message '0' 3. : receive message '1' 4. : receive message '2'

Полный текст программы:

1. Класс Main

```
var lab4 = new Lab4("Lab4");
lab4.Start();
```

2. Класс Lab4

```
using MPI;
using ProgramMPI.Interfaces;

namespace ProgramMPI.Labs
{
    public class Lab4 : ILab
    {
        public string Name { get; set; }
        public string[] _args { get; set; }

        public Lab4(string name, string[] args)
        {
            Name = name;
            _args = args;
        }

        public void ProccessStart(string[] args)
        {
            using (new MPI.Environment(ref args))
            {
                Intracommunicator comm = Communicator.world;

                int message;
                if (comm.Rank == 0)
                {
                    // Процесс 0 начинает эстафету, отправляя свой номер процессу 1
                    message = comm.Rank;
                    comm.Send(message, (comm.Rank + 1) % comm.Size, 0);
                }

                // Все процессы, кроме процесса 0, получают сообщение
                if (comm.Rank != 0)
                {
                    message = comm.Receive<int>(comm.Rank - 1, 0);
                    Print(comm.Rank, message);
                }
                else
                {
                    // Процесс 0 также ожидает получение сообщения от последнего
процесса
                    message = comm.Receive<int>(comm.Size - 1, 0);
                    Print(comm.Rank, message);
                }

                // Каждый процесс, получив сообщение, отправляет его следующему,
инкрементировав на 1
                // Процесс 0 уже отправил сообщение, поэтому его исключаем
                if (comm.Rank != 0)
                {
                    comm.Send(message + 1, (comm.Rank + 1) % comm.Size, 0);
                }
            }
        }
```

```
        public void Start()
        {
            ProccessStart(_args);

            Console.WriteLine("\n");
        }

        public override string ToString() => $"{Name} started classes:";

        private void Print(int rank, int receivedMessage) =>
            Console.WriteLine($"[{rank + 1}]: received message: '{receivedMessage}'");

    }
}
```

Результат:

```
[2]: received message: '0'
[3]: received message: '1'
[4]: received message: '2'
[1]: received message: '3'
```

Задание 5. Коммуникации "точка-точка" (схема "мастер-рабочие"):

> **Задание для выполнения**
>
> Напишите MPI-программу, реализующую при помощи блокирующих функций посылки сообщений типа точка-точка схему коммуникации процессов "master-slave", в которой один процесс, называемый master, принимает сообщение от остальных процессов, называемых slave. В качестве передаваемого сообщение используйте номер процесса. Master-процесс должен вывести на экран все полученные сообщение.

Входные данные: нет
Выходные данные: "<receive message '<сообщение>'> from <номер процесса>"

Входные данные	Выходные данные
	receive message '1' receive message '2' receive message '3'

Полный текст программы:

1. Класс Main

```
var lab5 = new Lab5("Lab5");
lab5.Start();
```

2. Класс Lab5

```
using MPI;
using ProgramMPI.Interfaces;

namespace ProgramMPI.Labs
{
    public class Lab5 : ILab
    {
        public string Name { get; set; }
        public string[] _args { get; set; }

        public Lab5(string name, string[] args)
        {
            Name = name;
            _args = args;
        }

        public void ProccessStart(string[] args)
        {
            using (new MPI.Environment(ref args))
            {
                Intracommunicator comm = Communicator.world;

                if (comm.Rank == 0) // Master процесс
                {
                    for (int i = 1; i < comm.Size; i++)
                    {
                        int message = comm.Receive<int>(i, 0);
                        Print(message);
                    }
                }
                else // Slave процессы
                {
                    comm.Send(comm.Rank, 0, 0); // Отправка номера своего процесса
master процессу
                }
            }
        }

        public void Start()
        {
            ProccessStart(_args);

            Console.WriteLine("\n");
        }

        public override string ToString() => $"{Name} started classes:";

        private void Print(int receivedMessage) =>
            Console.WriteLine($"received message: '{receivedMessage}'");

    }
}
```

Результат:

```
received message: '1'
received message: '2'
received message: '3'
```

Задание 6. Коммуникации "точка-точка" (простые неблокирующие обмены):

> **Задание для выполнения**
>
> Изучите основные MPI-функции неблокирующие передачи сообщений точка-точка **MPI_Isend**, **MPI_Irecy**, **MPI_Wait**. Напишите MPI-программу, в которой с помощью данных функций процесс с номером 0 отправляет сообщение процессу с номером 1. Процесс 1 выводит полученное сообщение на экран.

Входные данные: нет
Выходные данные: "receive message '<сообщение>'"

Входные данные	Выходные данные
	receive message '45'

Полный текст программы:

1. Класс Main

```
var lab6 = new Lab6("Lab6");
lab6.Start();
```

2. Класс Lab6

```
using MPI;
using ProgramMPI.Interfaces;

namespace ProgramMPI.Labs
{
    public class Lab6 : ILab
    {
        public string Name { get; set; }
        public string[] _args { get; set; }

        public Lab6(string name, string[] args)
        {
            Name = name;
            _args = args;
        }

        public void ProccessStart(string[] args)
        {
            using (new MPI.Environment(ref args))
            {
                Intracommunicator comm = Communicator.world;

                if (comm.Rank == 0)
                {
                    // Процесс 0 отправляет число 45 процессу 1
                    int message = 45;
                    // Неблокирующая отправка сообщения
                    Request sendRequest = comm.ImmediateSend(new int[] { message }, 1,
0);

                    sendRequest.Wait(); // Ожидаем завершения отправки
                }
                else if (comm.Rank == 1)
                {
                    // Процесс 1 получает сообщение от процесса 0
                    int[] receivedMessage = new int[1]; // Буфер для получения числа

                    // Неблокирующее получение сообщения
                    Request receiveRequest = comm.ImmediateReceive<int>(0, 0,
receivedMessage);
                    receiveRequest.Wait(); // Ожидание получения сообщения
                    Print(receivedMessage[0]);
                }
            }
        }

        public void Start()
        {
            ProccessStart(_args);

            Console.WriteLine("\n");
        }

        public override string ToString() => $"{Name} started classes:";
```

```
        private void Print(int receivedMessage) =>

            Console.WriteLine($"received message: '{receivedMessage}'");
    }
}
```

Результат:

```
received message: '45'
```