**Лабораторная работа №2: Коллективные операции в MPI на C++. Широковещательная передача, синхронизация и коллективные вычисления.**

**Теоретическая часть**

**MPI (Message Passing Interface)** — это стандарт, который описывает взаимодействие между процессами в распределённых системах. В этой лабораторной работе мы сосредоточимся на **MPI\_Barrier**, **MPI\_Bcast** и **MPI\_Reduce**, а также их использовании в программировании с MPI.

**MPI\_Barrier** — это функция, которая используется для синхронизации всех процессов в коммуникаторе. Когда процесс достигает MPI\_Barrier, он ожидает, пока все остальные процессы в том же коммуникаторе не достигнут этой точки. Это гарантирует, что все процессы завершили выполнение определенной части кода перед тем, как продолжить выполнение.

**MPI\_Bcast** — это функция, которая используется для широковещательной передачи данных от одного процесса (обычно с рангом 0) всем остальным процессам в коммуникаторе. Она позволяет одному процессу расслать массив данных, чтобы все остальные процессы могли его получить.

**MPI\_Reduce** — это функция, которая используется для объединения данных от всех процессов в одно значение с помощью заданной операции (например, суммы, максимума и т.д.). Результат сохраняется в одном процессе (обычно с рангом 0).

**Пример, когда требуется MPI\_Barrier:** Предположим, у вас есть несколько процессов, которые выполняют параллельные вычисления, и затем они должны объединить свои результаты для дальнейшей обработки. Если один процесс завершает свои вычисления быстрее других и начинает обрабатывать данные, не дожидаясь остальных, это может привести к ошибкам или некорректным результатам. Использование MPI\_Barrier гарантирует, что все процессы завершили свои вычисления перед началом объединения результатов.

**Задание**

1. Процесс с рангом 0 создаёт массив данных (например, 10 целых чисел) и рассылает его всем процессам с помощью MPI\_Bcast.
2. Каждый процесс изменяет данные массива, умножая их на свой ранг. После этого каждый процесс вычисляет сумму элементов массива.
3. Каждый процесс находит максимальный элемент массива.
4. Используя MPI\_Reduce, соберите суммы со всех процессов и передайте общую сумму в процесс 0. Также соберите максимальные значения от всех процессов и передайте результат в процесс 0.
5. Все процессы должны быть синхронизированы перед выводом результатов, используя MPI\_Barrier. Процесс 0 выводит на экран итоговую сумму и максимальный элемент массива, найденный среди всех процессов.

Ожидаемый вывод программы при запуске с 4 процессами.

*Process 0 broadcasting data...*

*Total sum of all elements: 330*

*Maximum element: 30*