**Задачи по параллельному программированию**

**Решить средствами OpenMP**

1. Разработайте программу для вычисления скалярного произведения двух векторов.
2. Умножение матрицы на вектор при разделении данных по строкам.
3. Умножение матрицы на вектор при разделении данных по столбцам.
4. Умножение матрицы на вектор при блочном разделении данных.
5. Умножение матриц, ленточный алгоритм.
6. Умножение матриц, блочные схемы распределения данных.
7. Умножение матриц (варианты распараллеливания вложенных циклов).
8. Разработайте программу для задачи вычисления определённого интеграла непрерывной функции f(x) по отрезку [a,b], с использованием метода прямоугольников.
9. Разработайте программу для задачи вычисления определённого интеграла непрерывной функции f(x) по отрезку [a,b] , с использованием формулы трапеций.
10. Разработайте программу для задачи вычисления определённого интеграла непрерывной функции f(x) по отрезку [a, b], с использованием формулы Симпсона.
11. Требуется реализовать вычисление числа ln 2 с помощью формулы:
12. Требуется реализовать вычисление числа с помощью формулы:
13. Требуется реализовать вычисление числа с помощью формулы:
14. Требуется реализовать вычисление числа с помощью формулы Валлиса:
15. Требуется реализовать вычисление числа π с помощью формулы Лейбница:

**Решить средствами MPI**

**Задачи на пересылку данных**

1. В каждом из процессов, входящих в коммуникатор MPI\_COMM\_WORLD, прочесть одно вещественное число и вывести его противоположное значение.
2. В каждом из процессов, входящих в коммуникатор MPI\_COMM\_WORLD, прочесть одно целое число A и вывести его удвоенное значение. Кроме того, для главного процесса (процесса ранга 0) вывести количество процессов, входящих в коммуникатор MPI\_COMM\_WORLD.
3. В главном процессе прочесть вещественное число X и вывести его противоположное значение, в каждом из остальных процессов (подчинённых процессов, ранг которых больше 0) вывести его ранг.
4. В процессах чётного ранга (включая главный) ввести целое число и вывести его удвоенное значение. В процессах нечётного ранга не выполнять никаких действий.
5. В процессах чётного ранга (включая главный) ввести целое число, в процессах нечётного ранга ввести вещественное число. В каждом процессе вывести удвоенное значение введённого числа.
6. В подчинённых процессах чётного ранга ввести целое число, в процессах нечётного ранга ввести вещественное число. В каждом подчинённом процессе вывести удвоенное значение введённого числа. В главном процессе не выполнять никаких действий.
7. В каждом процессе чётного ранга (включая главный) дано целое число N (> 0) и набор из N вещественных чисел. Вывести в каждом из этих процессов сумму чисел из данного набора. В процессах нечётного ранга не выполнять никаких действий.
8. В каждом процессе дано целое число N (> 0) и набор из N вещественных чисел. В процессах чётного ранга (включая главный) вывести сумму чисел из данного набора, в процессах нечётного ранга вывести среднее арифметическое чисел из данного набора.
9. В каждом процессе дано целое число N (> 0) и набор из N вещественных чисел. В подчинённых процессах чётного ранга вывести сумму чисел из данного набора, в процессах нечётного ранга вывести среднее арифметическое чисел из данного набора, в главном процессе вывести произведение чисел из данного набора.
10. В каждом процессе дано целое число N (> 0) и набор из N чисел, причём в подчинённых процессах нечётного ранга (1, 3, ...) набор содержит вещественные числа, в подчинённых процессах чётного ранга (2, 4, ...) — целые числа, а тип элементов в главном наборе зависит от общего количества процессов: если количество процессов нечётное, то набор содержит целые числа, а если чётное, то вещественные. В процессах чётного ранга (включая главный) вывести минимальный элемент из данного набора, в процессах нечётного ранга вывести максимальный элемент.

**Задачи на тему «Процессы и их ранги»**

1. В каждом подчинённом процессе дано целое число. Переслать эти числа в главный процесс, используя функции MPI\_Send и MPI\_Recv (стандартные блокирующие функции для передачи и приема сообщения), и вывести их в главном процессе. Полученные числа выводить в порядке возрастания рангов переславших их процессов.
2. В главном процессе дан набор вещественных чисел; количество чисел равно количеству подчинённых процессов. С помощью функции MPI\_Send переслать по одному числу в каждый из подчинённых процессов (первое число в процесс 1, второе — в процесс 2, и т. д.) и вывести в подчинённых процессах полученные числа.
3. В каждом подчинённом процессе даны четыре целых числа. Переслать эти числа в главный процесс, используя по одному вызову функции MPI\_Send для каждого передающего процесса, и вывести их в главном процессе. Полученные числа выводить в порядке возрастания рангов переславших их процессов.
4. В главном процессе дано целое число *N* и набор из *N* чисел; *K* − 1 ≤ *N* < 10, где *K* — количество процессов. С помощью функции MPI\_Send переслать по одному числу их данного набора в процессы 1, 2, …, *K* − 2, а оставшиеся числа — в процесс *K* − 1, и вывести полученные числа. В процессе *K* − 1 для определения количества полученных чисел использовать функцию MPI\_Get\_count.
5. В каждом подчинённом процессе дано целое число, причём только для одного процесса это число отлично от нуля. Переслать ненулевое число в главный процесс и вывести в главном процессе полученное число и ранг процесса, переславшего это число. Для приёма сообщения в главном процессе использовать функцию MPI\_Recv с параметром MPI\_ANY\_SOURCE.
6. В каждом подчинённом процессе дано целое число *N*, причём для одного процесса это число больше нуля, а для остальных равно нулю. В процессе с ненулевым *N* дан также набор из *N* чисел. Переслать данный набор чисел в главный процесс и вывести в главном процессе полученные числа и ранг процесса, переславшего этот набор. При приёме сообщения использовать параметр MPI\_ANY\_SOURCE.
7. В каждом подчинённом процессе дано целое число *N*, в главном процессе дано целое число *K* (> 0), равное количеству тех подчинённых процессов, в которых даны положительные числа *N*. Переслать все положительные числа *N* в главный процесс и вывести в нем сумму полученных чисел. Для приёма сообщений в главном процессе использовать функцию MPI\_Recv с параметром MPI\_ANY\_SOURCE.
8. В каждом процессе даны два числа: вещественное *A* и целое *N*, причём набор чисел *N* содержит все значения от 0 до *K* − 1, где *K* — количество процессов. Используя функции MPI\_Send и MPI\_Recv (с параметром MPI\_ANY\_SOURCE), выполнить в каждом процессе пересылку числа *A* в процесс *N* и вывести полученное число, а также ранг процесса, из которого число было получено.
9. В каждом процессе дано целое число *N*, причём для одного процесса значение *N* равно 1, а для остальных равно 0. В процессе с *N* = 1 дан также набор из *K* − 1 числа, где *K* — количество процессов. Переслать из этого процесса по одному из чисел данного набора в остальные процессы, перебирая ранги получателей в возрастающем порядке, и вывести в каждом из них полученное число.
10. В каждом процессе дан набор из *K* − 1 целого числа, где *K* — количество процессов. Для каждого процесса переслать по одному из данных в нем чисел в остальные процессы, перебирая ранги процессов-получателей в возрастающем порядке, и вывести полученные числа в порядке возрастания рангов переславших их процессов.
11. В каждом подчинённом процессе дано вещественное число *A* и его порядковый номер *N* (целое число); набор всех номеров *N* содержит все целые числа от 1 до *K* − 1, где *K* — количество процессов. Переслать числа *A* в главный процесс и вывести их в порядке, соответствующем возрастанию их номеров *N*. Массивы не использовать; для передачи номера *N* указывать его в качестве параметра tag функции MPI\_Send.
12. В каждом подчинённом процессе дано целое число *L* (≥ 0) и набор из *L* пар чисел (*A*, *N*), где *A* — вещественное число, а *N* — его порядковый номер. Все числа *L* в сумме равны 2*K*, где *K* — количество процессов; набор номеров *N*, данных во всех процессах, содержит все целые числа от 1 до 2*K*. Переслать числа *A* в главный процесс и вывести их в порядке, соответствующем возрастанию их номеров *N*. Массивы не использовать; для передачи номера *N* указывать его в качестве параметра tag функции MPI\_Send.
13. В каждом подчинённом процессе дано вещественное число. Переслать эти числа в главный процесс, используя функции MPI\_Bsend (посылка сообщения с буферизацией) и MPI\_Recv, и вывести их в главном процессе. Полученные числа выводить в порядке убывания рангов переславших их процессов. Для задания буфера использовать функцию MPI\_Buffer\_attach.
14. В главном процессе дан набор вещественных чисел; количество чисел равно количеству подчинённых процессов. С помощью функции MPI\_Bsend переслать по одному числу в каждый из подчинённых процессов, перебирая процессы в обратном порядке (первое число в последний процесс, второе — в предпоследний процесс, и т. д.), и вывести в подчинённых процессах полученные числа.
15. В каждом подчинённом процессе дано целое число *N* (0 < *N* < 5) и набор из *N* целых чисел. Переслать данные наборы в главный процесс, используя по одному вызову функции MPI\_Bsend для каждого передающего процесса, и вывести наборы в главном процессе в порядке возрастания рангов переславших их процессов. Для определения размера пересланного набора использовать функцию MPI\_Get\_count.
16. В каждом процессе дано вещественное число. Переслать число из главного процесса во все подчинённые процессы, а все числа из подчинённых процессов — в главный, и вывести в каждом процессе полученные числа (в главном процессе числа выводить в порядке возрастания рангов переславших их процессов). Для отправки сообщений использовать функцию MPI\_Ssend.

**Указание.** Функция MPI\_Ssend обеспечивает *синхронный режим* пересылки данных, при котором операция отправки сообщения будет завершена только после начала приема этого сообщения процессом-получателем. В случае пересылки данных в синхронном режиме возникает опасность *взаимных блокировок* (deadlocks) из-за неправильного порядка вызова функций отправки и получения сообщений.

1. В каждом процессе дано целое число. С помощью функций MPI\_Ssend и MPI\_Recv осуществить для всех процессов циклический сдвиг данных с шагом 1, переслав число из процесса 0 в процесс 1, из процесса 1 в процесс 2, …, из последнего процесса в процесс 0. В каждом процессе вывести полученное число. **Указание. См. указание к задаче 16.**
2. В каждом процессе дано целое число. С помощью функций MPI\_Ssend и MPI\_Recv осуществить для всех процессов циклический сдвиг данных с шагом −1, переслав число из процесса 1 в процесс 0, из процесса 2 в процесс 1, …, из процесса 0 в последний процесс. В каждом процессе вывести полученное число. **Указание. См. указание к задаче 16.**
3. В каждом процессе даны два целых числа. С помощью функций MPI\_Ssend и MPI\_Recv переслать первое число в предыдущий процесс, а второе — в последующий процесс (для процесса 0 считать предыдущим последний процесс, а для последнего процесса считать последующим процесс 0). В каждом процессе вывести числа, полученные от предыдущего и последующего процесса (в указанном порядке). **Указание. См. указание к задаче 16.**
4. Количество процессов — чётное число. В каждом процессе дано целое число *N* (0 < *N* < 5) и набор из *N* чисел. С помощью функции MPI\_Sendrecv выполнить обмен исходными наборами между парами процессов 0 и 1, 2 и 3, и т. д. В каждом процессе вывести полученный набор чисел.
5. В каждом процессе дано вещественное число. С помощью функции MPI\_Sendrecv\_replace поменять порядок исходных чисел на обратный (число из процесса 0 должно быть передано в последний процесс, число из процесса 1 — в предпоследний процесс, …, число из последнего процесса — в процесс 0). В каждом процессе вывести полученное число.

**Теоретические вопросы для опроса**

1. Понятие параллельных вычислений. Необходимость параллельных вычислений. Сдерживающие факторы. Классификация параллельных аппаратных архитектур по Флинну (SISD, SIMD, MISD, MIMD). Характеристика последовательных программ, характеристика параллельных программ. Особенности параллельных программ.
2. Принципы разработки параллельных программ. Основные понятия. Концепция процесса. Определение потока. Понятие ресурса. Организация параллельных программ как системы потоков. Взаимодействие и взаимоисключение потоков. Принципы разработки параллельных методов. Разделение вычислений на независимые части. Выделение информационных зависимостей.
3. Классические задачи синхронизации "Производители-Потребители", "Читатели-Писатели", "Обедающие философы", "Спящий парикмахер".
4. Технология параллельного программирования OpenMP. Директивы OpenMP. Формат записи директив. Прагма omp parallel. Определение параллельной области. Параллельное программирование с использованием ОреnМР. Понятие параллельной программы.
5. Параллельное программирование с использованием OpenMP. Организация взаимодействия параллельных потоков. Структура OpenMP. Формат директив OpenMP. Выделение параллельно выполняемых фрагментов программного кода. Директива parallel для определения параллельных фрагментов. Пример первой параллельной программы. Основные понятия параллельной программы: фрагмент, область, секция.
6. Управляющие конструкции в OpenMP. Параметры директивы parallel. Определение времени выполнения параллельной программы. Распределение вычислительной нагрузки между потоками (распараллеливание по данным для циклов). Управление распределением итераций цикла между потоками. Управление порядком выполнения вычислений. Синхронизация вычислений по окончании выполнения цикла. Введение условий при определении параллельных фрагментов (параметр if директивы parallel). Управление данными для параллельно выполняемых потоков.
7. Организация взаимоисключения при использовании общих переменных. Определение общих и локальных переменных. Совместная обработка локальных переменных (операция редукции). Обеспечение атомарности (неделимости) операций. Использование критических секций. Применение переменных семафорного типа (замков). Распределение вычислительной нагрузки между потоками (распараллеливание по задачам при помощи директивы sections).
8. Определение однопотоковых участков для параллельных фрагментов (директивы single и master). Выполнение барьерной синхронизации (директива barrier). Синхронизация состояния памяти (директива flush). Определение постоянных локальных переменных потоков (директива threadprivate и параметр copyin директивы parallel). Управление количеством потоков. Задание динамического режима при создании потоков. Управление вложенностью параллельных фрагментов.
9. MPI: основные понятия и определения. Понятие параллельной программы. Операции передачи данных. Понятие коммуникаторов. Типы данных. Виртуальные топологии. Функции попарного обмена MPI\_Send, MPI\_Recv, MPI\_Wait, MPI\_Time и др. Примеры.
10. Режимы передачи данных. Функции MPI\_Ssend, MPI\_Rsend, MPI\_Bsend, MPI\_Buffer\_attach, MPI\_Buffer\_detach, MPI\_Sendrecv.
11. Коллективные операции передачи данных MPI. Обобщенная передача данных от одного процесса всем процессам. Обобщенная передача данных от всех процессов одному процессу. Общая передача данных от всех процессов всем процессам. Дополнительные операции редукции данных. Сводный перечень коллективных операций данных. Функции MPI\_Bcast, MPI\_Reduce, MPI\_Barrier, MPI\_Ssend, MPI\_Rsend, MPI\_Bsend. Функции MPI\_Buffer\_attach, MPI\_Buffer\_detach, MPI\_Sendrecv, MPI\_Scan, MPI\_Scatter, MPI\_Scatterv, MPI\_Gather, MPI\_Gatherv, MPI\_Allgather, MPI\_Allgatherv, MPI\_Alltoall, MPI\_Alltoallv.
12. Производные типы данных в MPI. Понятие производного типа данных. Способы конструирования производных типов данных. Объявление производных типов и их удаление. Формирование сообщений при помощи упаковки и распаковки данных. Управление группами. Управление коммуникаторами. Виртуальные топологии. Декартовы топологии (решетки). Топологии графа. Функции MPI\_Type\_contiguous, MPI\_Type\_vector, MPI\_Type\_hvector, MPI\_Type\_indexed, MPI\_Type\_struct , MPI\_Pack, MPI\_Pack\_size, MPI\_Unpack
13. Параллельные методы умножения матрицы на вектор. Принципы распараллеливания. Постановка задачи. Последовательный алгоритм. Умножение матрицы на вектор при разделении данных по строкам. Умножение матрицы на вектор при разделении данных по столбцам. Определение подзадач и выделение информационных зависимостей. Умножение матрицы на вектор при блочном разделении данных.