Программа утверждена на заседании кафедры вычислительной математики
Протокол № 1 от 7 сентября 2015 г.

Рабочая программа дисциплины (модуля)

1. Код и наименование дисциплины (модуля): СПЕЦКУРС (годовой). ОСНОВЫ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ.

2. Уровень высшего образования – специалитет.

3. Направление подготовки: 01.05.01 Фундаментальные математика и механика. Специализация: Фундаментальная математика.

4. Место дисциплины (модуля) в структуре ООП: вариативная часть ООП. Является специальной дисциплиной (спецкурсом) для студентов 3-6 годов обучения, специализирующихся в данной научной области или смежной научной области, спецкурсом по выбору студента.
Освоение дисциплины необходимо для последующего изучения дисциплин образовательной программы: курсовая работа, научно-исследовательская практика, преддипломная практика, выпускная квалификационная работа.

5. Планируемые результаты обучения по дисциплине (модулю), соотнесенные с планируемыми результатами освоения образовательной программы (компетенциями выпускников)

Формируемые компетенции	**Планируемые результаты обучения по дисциплине (модулю)**
СПК-1 владение специальными разделами фундаментальной математики, методами анализа и решения задач специализации	Знание специальных разделов теории построения программного обеспечения и примеров реализации и использования этой теории на практике. Умение применять методы построения параллельных алгоритмов и параллельного программирования при решении задач специализации

6. Объем дисциплины (модуля) в зачетных единицах с указанием количества академических или астрономических часов, выделенных на контактную работу обучающихся с преподавателем (по видам учебных занятий) и на самостоятельную работу обучающихся:

Объем дисциплины (модуля) составляет 4 зачетных единиц, всего 144 часов, из которых 88 часов составляет контактная работа студента с преподавателем (72 часов занятия лекционного типа, 16 часов мероприятия промежуточной аттестации), 56 часов составляет самостоятельная работа студента.

7. Входные требования для освоения дисциплины (модуля), предварительные условия.

Для того чтобы изучение дисциплины было возможно, обучающийся должен

1) освоить дисциплины базовой части образовательной программы специалиста 1 и 2-го годов обучения
2) обладать следующими компетенциями:

Знать: основные направления, проблемы, теории и методы фундаментальной математики.

Уметь: решать стандартные задачи программирования (работа со структурами данных) и реализации базовых численных методов на языке C++, и применять идеи, использованные в их решениях, для решения аналогичных задач.

Владеть: основными понятиями и теоремами из курса «Работа на ЭВМ и программирование»

8. Формат обучения.

Очная форма обучения, лекционные занятия.

9. Содержание дисциплины (модуля), структурированное по темам* (Перечень тем см. Приложения).

Наименование и краткое содержание разделов и тем дисциплины (модуля), **форма промежуточной аттестации по дисциплине (модулю)**	**Всего (часы)**	В том числе								
		Контактная работа (работа во взаимодействии с преподавателем), часы из них						**Самостоятельная работа обучающегося, часы** из них		
		Занятия лекционного типа	Занятия семинарского типа	Групповые консультации	Индивидуальные консультации	Учебные занятия, направленные на проведение текущего контроля успеваемости, промежуточной аттестации	**Всего**	Выполнение домашних заданий	Подготовка рефератовит.п..	**Всего**

Тема 1	4	2					2	2		2
Тема 2	4	2					2	2		2
Тема 3	4	2					2	2		2
Тема 4	4	2					2	2		2
Тема 5	4	2					2	2		2
Тема 6	4	2					2	2		2
Тема 7	4	2					2	2		2
Тема 8	4	2					2	2		2
Текущий контроль успеваемости	10					2	2	8		8
Тема 9	4	2					2	2		2
Тема 10	4	2					2	2		2
Тема 11	4	2					2	2		2
Тема 12	4	2					2	2		2
Тема 13	4	2					2	2		2
Тема 14	4	2					2	2		2
Тема 15	4	2					2	2		2
Тема 16	4	2					2	2		2

Текущий контроль успеваемости	10					2	2	8		8
Тема 17	4	2					2	2		2
Тема 18	4	2					2	2		2
Тема 19	4	2					2	2		2
Тема 20	4	2					2	2		2
Тема 21	4	2					2	2		2
Тема 22	4	2					2	2		2
Тема 23	4	2					2	2		2
Тема 24	4	2					2	2		2
Текущий контроль успеваемости	10					2	2	8		8
Тема 25	4	2					2	2		2
Тема 26	4	2					2	2		2
Тема 27	4	2					2	2		2
Тема 28	4	2					2	2		2
Тема 29	4	2					2	2		2
Тема 30	4	2					2	2		2

Тема 31	4	2					2	2		2
Тема 32	2						0	2		2
Промежуточная аттестация *Экзамен (зачет)*	24					2	2	22		22
Итого	180	62				8	70	110		110

10. Перечень учебно-методического обеспечения для самостоятельной работы студентов по дисциплине (модулю):
Конспекты лекций, списки задач к лекциям, основная и дополнительная учебная литература.

11. Фонд оценочных средств для промежуточной аттестации по дисциплине (модулю).

- Перечень компетенций: ПСК-1.
- Описание шкал оценивания*:*

экзамен с оценкой по пятибалльной шкале

- Критерии и процедуры оценивания результатов обучения по дисциплине (модулю), характеризующих этапы формирования компетенций.

РЕЗУЛЬТАТ ОБУЧЕНИЯ по дисциплине (модулю)	**КРИТЕРИИ ОЦЕНИВАНИЯ РЕЗУЛЬТАТА ОБУЧЕНИЯ по дисциплине (модулю) и ШКАЛА оценивания**					**ПРОЦЕДУРЫ ОЦЕНИВАНИЯ**
	1	2	3	4	5	
Знание специальных разделов теории построения программного обеспечения и примеров	Отсутствие знаний	Фрагментарные знания специальных разделов теории построения программного обеспечения	Общие, но не структурированные знания специальных разделов теории построения программного обеспечения	Сформированные, но содержащие отдельные пробелы знания специальных разделов теории построения программного обеспечения	Сформированные систематические знания специальных разделов теории построения программного обеспечения	Экзамен в форме индивидуального собеседования

реализации и использования этой теории на практике.						
Умение применять методы построения параллельных алгоритмов и параллельного программирования при решении задач специализации	Отсутствие умений	Частично освоенное умение применять методы построения параллельных алгоритмов и параллельного программирования	В целом успешное, но не систематически осуществляемое умение применять методы построения параллельных алгоритмов и параллельного программирования	В целом успешное, но содержащие отдельные пробелы умение применять методы построения параллельных алгоритмов и параллельного программирования	Сформированное умение применять методы построения параллельных алгоритмов и параллельного программирования	Практическая реализация программных алгоритмов

- Типовые контрольные задания или иные материалы, необходимые для оценки результатов обучения, характеризующих этапы формирования компетенций. См. Приложения.

12. Ресурсное обеспечение:

Перечень основной учебной литературы: см. Приложение

Перечень дополнительной учебной литературы: см. Приложения

Перечень ресурсов информационно-телекоммуникационной сети «Интернет»: см. Приложения.

Описание материально-технической базы: аудитории для проведения лекционных занятий.

13. Язык преподавания: русский (при необходимости – английский).

ПРИЛОЖЕНИЕ

1. Спецкурс программы специалитета, годовой. ОСНОВЫ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ (на русском языке).
2. Преподаватель - доц. К.Ю. Богачев
3. Аннотация курса: строение параллельных ЭВМ, методы их программирования, распараллеливание базовых алгоритмов линейной алгебры.
4. Тематическое содержание курса

Тема 1	Архитектуры современных процессоров. Конвейеры, независимо работающие устройства.
Тема 2	Кэш память и ее организация. Оптимизация кода для современных процессоров.
Тема 3	Виды многопроцессорных архитектур. Общее строение современных кластерных систем.
Тема 4	Типы топологий соединений узлов. Графические и вычислительные сопроцессоры.
Тема 5	Процессы, состояния, механизмы взаимодействия и управления.
Тема 6	Виды ресурсов, приоритеты.
Тема 7	Межпроцессное взаимодействие стандарта IPC.
Тема 8	Разделяемая память, семафоры, очереди сообщений.
Тема 9	Потоки исполнения.

Тема 10	Управление и объекты синхронизации типов mutex и condvar.
Тема 11	Влияние дисциплины доступа к оперативной памяти на эффективность работы.
Тема 12	Message Passing Interface (MPI). Сообщения и их виды
Тема 13	. Коммуникаторы. Попарный обмен сообщениями.
Тема 14	Коллективный обмен сообщениями.
Тема 15	. Передача строк и столбцов матриц при коллективных обменах
Тема 16	Ограничение коллективного обмена на подмножество процессов.
Тема 17	Методы построения триангуляции двумерных областей. Метод наименьших квадратов.
Тема 18	Общая схема построения матрицы системы для базиса из функций Куранта.
Тема 19	Разреженные матрицы и методы их хранения в оперативной памяти.
Тема 20	Общий вид одношаговых итерационных методов.
Тема 21	Понятие предобуславливателя. Пример подпрограммы вычисления предобуславливателя в методе Якоби для разреженной матрицы в формате MSR для систем с общей памятью.
Тема 22	Автоматический выбор итерационного параметра
Тема 23	Метод наименьших невязок..
Тема 24	Организация вычислений для минимизации количества умножений матрицы на вектор
Тема 25	Вычислительная устойчивость итерационных алгоритмов.

Тема 26	Зависимость результата от числа логических процессоров.
Тема 27	Основные этапы разработки программы построения матрицы системы в формате MSR для метода наименьших квадратов для систем с общей и распределенной памятью.
Тема 28	Обобщенные производные. Пространства Соболева. Теоремы вложения и о следах.
Тема 29	Обобщенное решение задачи Дирихле для уравнения Пуассона. Теорема существования и единственности, априорная оценка.
Тема 30	Метод конечных элементов. Общая схема построения матрицы системы для задачи Дирихле для уравнения Пуассона.
Тема 31	Основные этапы разработки программы построения матрицы системы в формате MSR.
Тема 32	Разработка программы построения матрицы системы в формате MSR для задачи Дирихле для уравнения Пуассона для систем с общей и распределенной памятью.

5. Типовые контрольные задания или иные материалы, необходимые для оценки результатов обучения, характеризующих этапы формирования компетенций.

   Контрольные вопросы и задания по обязательной и вариативной частям дисциплины для промежуточной аттестации по итогам освоения дисциплины:

   1. CISC и RISC процессоры. Основные черты RISC архитектуры.
   2. Повышение производительности процессоров за счет конвейеризации. Условия оптимального функционирования конвейера.
   3. Суперконвейерные и суперскалярные процессоры. Выделение независимо работающих устройств: IU, FPU, MMU, BU.
   4. Методы уменьшения негативного влияния инструкций перехода на производительность процессора.
   5. Повышение производительности процессоров за счет введения кэш памяти. Кэши: единый, Гарвардский, с прямой записью, с обратной записью.
   6. Организация кэш-памяти. Алгоритмы замены данных в кэш памяти. Специальные кэши.
   7. Согласование кэшей в мультипроцессорных системах с общей памятью.

8. Виды многопроцессорных архитектур. Системы с разделяемой и распределенной памятью. Гибридные системы.
9. Типы топологий соединений узлов кластерных систем. Взаимодействие Infiniband и Ethernet каналов. Коммуникационный сопроцессор платы ввода-вывода, его взаимодействие с основной системой.
10. Общее строение современных кластерных систем, их подсистем коммуникации и ввода-вывода. Проблемы масштабирования в больших системах.
11. Графические и вычислительные сопроцессоры. Методы программирования и взаимодействия с основной вычислительной системой.
12. Методы повышения производительности и емкости оперативной и дисковой памяти.
13. Поддержка многозадачности и многопроцессорности специальными инструкциями процессора. Организация данных во внешней памяти.
14. Программа, процессор, процесс. Основные составляющие процесса, состояния процесса. Стек, виртуальная память, механизмы трансляции адреса.
15. Механизмы взаимодействия процессов. Разделяемая память, семафоры, сигналы, почтовые ящики, события. Задачи (threads). Сравнение с процессами. Ресурсы, приоритеты. Параллельные процессы. Связывание. Статическое и динамическое связывание.
16. Виды ресурсов: аппаратные, программные, активные, пассивные, локальные, разделяемые, постоянные, временные, не критичные, критичные.
17. Типы взаимодействия процессов: сотрудничающие и конкурирующие процессы. Критические секции, взаимное исключение процессов (задач).
18. Проблемы, возникающие при синхронизации задач и идеи их разрешения.
19. Состояния процесса и механизмы перехода из одного состояния в другое.
20. Стандарты на UNIX системы. Стандарты на работу с объектами в разделяемой и распределенной памяти.
21. Управление процессами. Основные этапы создания нового процесса. Отношения между родительским и порожденным процессами (функции fork, execl, execv, waitpid).
22. Работа с сигналами. Идентификация сигналов, методы реакции на сигналы и инициации сигналов (функции signal, kill). Оповещение процесса о наступлении аварийной ситуации.
23. Межпроцессное взаимодействие стандарта IPC: разделяемая память (функции shmget, shmat, shmctl); семафоры (функции semget, semop, semctl); очереди сообщений (функции msgget, msgsnd, msgrcv, msgctl). Роль ядра ОС в механизме IPC.
24. Управление задачами (threads). Функции pthread_create, pthread_join, sched_yield.

25. Объекты синхронизации типа mutex. Функции pthread_mutex_init, pthread_mutex_lock, pthread_mutex_trylock, pthread_mutex_unlock, pthread_mutex_destroy.
26. Объекты синхронизации типа condvar. Функции pthread_cond_init, pthread_cond_signal, pthread_cond_broadcast, pthread_cond_wait, pthread_cond_destroy.
27. Влияние дисциплины доступа к оперативной памяти на эффективность работы.
28. Message Passing Interface (MPI). Общая структура MPI-программы. Функции MPI_Init, MPI_Finalize. Сообщения и их виды.
29. Коммуникаторы. Функции MPI_Comm_size, MPI_Comm_rank.
30. Попарный обмен сообщениями. Функции MPI_Send, MPI_Recv.
31. Операции ввода-вывода в MPI программах.
32. Дополнительные возможности для попарного обмена сообщениями. Функции MPI_Sendrecv, MPI_Sendrecv_replace.
33. Дополнительные возможности для попарного обмена сообщениями. Функции MPI_Isend, MPI_Irecv, MPI_Test, MPI_Testany, MPI_Wait, MPI_Waitany.
34. Коллективный обмен сообщениями. Функции MPI_Barrier, MPI_Abort, MPI_Bcast.
35. Коллективный обмен сообщениями. Функции MPI_Reduce, MPI_Allreduce, MPI_Op_create, MPI_Op_free.
36. Время в MPI программах. Астрономическое и процессорное время. Функции MPI_Wtime, MPI_Wtick.
37. Дополнительные возможности для коллективного обмена массивами данных. Функции MPI_Gather, MPI_Allgather, MPI_Scatter.
38. Дополнительные возможности для коллективного обмена массивами данных. Передача строк и столбцов матриц. Функции MPI_Type_vector, MPI_Type_commit.
39. Ограничение коллективного обмена на подмножество процессов. Функции MPI_Comm_group, MPI_Group_incl, MPI_Comm_create, MPI_Comm_free.
40. Методы построения триангуляции двумерных областей.
41. Метод наименьших квадратов. Общая схема построения матрицы системы для базиса из функций Куранта.
42. Разреженные матрицы и методы их хранения в оперативной памяти.
43. Общий вид одношаговых итерационных методов. Оценка падения невязки за k шагов. Оптимальный выбор параметра для одношагового метода с самосопряженной матрицей.
44. Понятие предобуславливателя. Влияние на скорость сходимости. Методы Якоби, Зейделя, верхней релаксации.
45. Автоматический выбор итерационного параметра. Метод наименьших невязок. Организация вычислений для минимизации количества умножений матрицы на вектор.
46. Вычислительная устойчивость итерационных алгоритмов. Зависимость результата от числа логических процессоров.

47. Основные этапы разработки программы построения матрицы системы в формате MSR для метода наименьших квадратов в прямоугольнике для систем с общей памятью.
48. Обобщенные производные. Пространства Соболева. Неравенства Фридрихса и Пуанкаре. Нормы в пространствах Соболева.
49. Обобщенное решение задачи Дирихле для уравнения Пуассона. Теорема существования и единственности, априорная оценка.
50. Теоремы вложения для пространств Соболева. Теоремы о следах функций из пространств Соболева.
51. Метод конечных элементов. Общая схема построения матрицы системы для задачи Дирихле для уравнения Пуассона.
52. Основные этапы разработки программы построения матрицы системы в формате MSR для задачи Дирихле для уравнения Пуассона в прямоугольнике для систем с общей памятью.
53. Основные этапы разработки программы построения матрицы системы в формате MSR для задачи Дирихле для уравнения Пуассона в прямоугольнике для систем с распределенной памятью.

Примеры задач:

1. Написать программу, вычисляющую сумму элементов N последовательностей вещественных чисел, находящихся в N файлах, имена которых заданы массивом A. Ответ должен быть записан в файл res.txt. Программа запускает N процессов, вычисляющих ответ для каждого из файлов и прибавляющих его к результату, находящемуся в выходном файле. Взаимное исключение при доступе к файлу обеспечивается с помощью семафора.
2. Написать функцию, получающую в качестве аргументов целое число N и массив длины N с именами файлов, содержащих единую последовательность вещественных чисел неизвестной длины, и возвращающую количество участков постоянства этой последовательности. Функция возвращает -1, -2 и т.д., если она не смогла открыть какой-либо файл, прочитать элемент и т.д.. Функция должна запускать N процессов, обрабатывающих свой файл и передающих результаты в основной процесс через очередь сообщений для формирования ответа для последовательности в целом.
3. Написать программу, осуществляющую мониторинг и перезапуск в случае завершения работы заданного количества приложений. Приложения задаются массивом строк, являющихся их полным путевым именем, и не имеют аргументов.
4. Написать реализацию стека строк в разделяемой памяти. При запуске программа создает блок разделяемой памяти (если его еще нет) или присоединяется к существующему блоку. Программа должна обеспечивать основные функции работы со стеком (добавить и удалить элемент) и возможность запуска себя во многих экземплярах.
5. Написать реализацию набора конфигурационных параметров для многих одновременно работающих экземпляров программы. Набор параметров задается некоторой структурой данных и хранится в разделяемой памяти. Блок разделяемой памяти создается при запуске первого экземпляра программы и заполняется из файла. Перед окончанием работы последнего экземпляра набор

параметров сохраняется в файле, а блок разделяемой памяти удаляется. Программа должна обеспечивать основные функции работы с набором параметров (прочитать и изменить элемент), причем в случае изменения данных одним из экземпляров он оповещает все остальные экземпляры с помощью сигнала.

6. Написать многопоточную функцию, получающую в качестве аргументов NxN массив A вещественных чисел, целое число N, номер потока K, общее количество потоков P, и возвращающую ненулевое значение, если массив A симметричен (т.е. A_{ij}=A_{ji}), 0 в противном случае. При этом должна быть обеспечена равномерная загрузка всех потоков. Основная программа должна вводить числа P, N и массив A (из файла или по заданной формуле), запускать потоки, вызывать эту функцию и выводить на экран результат ее работы.
7. Написать многопоточную подпрограмму, получающую в качестве аргументов NxN массив A вещественных чисел, целое число N, номер потока K, общее количество потоков P, и заменяющую матрицу A на ее транспонированную. При этом должна быть обеспечена равномерная загрузка всех потоков. Основная программа должна вводить числа P, N и массив A (из файла или по заданной формуле), запускать потоки, вызывать эту подпрограмму и выводить на экран результат ее работы.
8. Написать многопоточную подпрограмму, получающую в качестве аргументов NxN массив A вещественных чисел, целое число N, номер потока K, общее количество потоков P, и заменяющую матрицу A на матрицу (A+A^t)/2, где A^t - транспонированная матрица A. При этом должна быть обеспечена равномерная загрузка всех задач. Основная программа должна вводить числа P, N и массив A (из файла или по заданной формуле), запускать потоки, вызывать эту подпрограмму и выводить на экран результат ее работы.
9. Написать многопоточную подпрограмму, получающую в качестве аргументов массив A вещественных чисел, целое число N, являющееся длиной этого массива, номер потока K, общее количество потоков P, и заменяющую каждый элемент массива (для которого это возможно) на среднее арифметическое соседних элементов. При этом должна быть обеспечена равномерная загрузка всех задач. Основная программа должна вводить числа P, N и массив A (из файла или по заданной формуле), запускать потоки, вызывать эту подпрограмму и выводить на экран результат ее работы.
10. Написать многопоточную функцию, получающую в качестве аргументов массив A вещественных чисел, целое число N, являющееся длиной этого массива, номер потока K, общее количество потоков P, и заменяющую элементы массива, принадлежащие участку строго возрастания, на среднее арифметическое элементов во всех таких участках массива. При этом должна быть обеспечена равномерная загрузка всех потоков. Основная программа должна вводить числа P, N и массив A (из файла или по заданной формуле), запускать потоки, в которых вызывается эта функция, выводить на экран результат, а также процессорное время, затраченное на каждый поток и суммарно на все потоки.

11. Написать многопоточную подпрограмму, получающую в качестве аргументов NxN массив A вещественных чисел, вспомогательный массив B вещественных чисел длины N (в каждом потоке свой), целое число N, номер потока K, общее количество потоков P, и заменяющую каждый элемент A_{ij} матрицы A (для которого это возможно) на (A_{i+1,j}+A_{i-1,j}+A_{i,j+1}+A_{i,j-1}-4A_{i,j}). При этом должна быть обеспечена равномерная загрузка всех задач. Основная программа должна вводить числа P, N и массив A (из файла или по заданным формулам), запускать потоки, вызывать эту подпрограмму и выводить на экран результат ее работы.
12. Написать многопоточную подпрограмму, получающую в качестве аргументов NxN массив A вещественных чисел, вспомогательный массив B вещественных чисел длины 2N (в каждом потоке свой), целое число N, номер потока K, общее количество потоков P, и заменяющую каждый элемент A_{ij} матрицы A (для которого это возможно) на (A_{i+2,j}+A_{i-2,j}+A_{i,j+2}+A_{i,j-2}-4A_{i,j}). При этом должна быть обеспечена равномерная загрузка всех задач. Основная программа должна вводить числа P, N и массив A (из файла или по заданным формулам), запускать потоки, вызывать эту подпрограмму и выводить на экран результат ее работы.
13. Написать MPI функцию, получающую в качестве аргументов соответствующую часть (блок) NxN массива A вещественных чисел, целое число N, номер процесса K, общее количество процессов P, и возвращающую ненулевое значение, если массив A симметричен (т.е. A_{ij}=A_{ji}), 0 в противном случае. При этом должна быть обеспечена равномерная загрузка всех процессов. Основная программа должна начинать работу с MPI, вводить число N и массив A (из файла или по заданной формуле), вызывать эту функцию и выводить на экран результат ее работы.
14. Написать MPI подпрограмму, получающую в качестве аргументов соответствующую часть (блок) NxN массива A вещественных чисел, целое число N, номер процесса K, общее количество процессов P, и заменяющую матрицу A на ее транспонированную. При этом должна быть обеспечена равномерная загрузка всех процессов. Основная программа должна начинать работу с MPI, вводить число N и массив A (из файла или по заданной формуле), вызывать эту подпрограмму и выводить на экран результат ее работы.
15. Написать MPI подпрограмму, получающую в качестве аргументов соответствующую часть (блок) NxN массива A вещественных чисел, целое число N, номер процесса K, общее количество процессов P, и заменяющую матрицу A на матрицу (A+A^t)/2, где A^t -- транспонированная матрица A. При этом должна быть обеспечена равномерная загрузка всех процессов. Основная программа должна начинать работу с MPI, вводить число N и массив A (из файла или по заданной формуле), вызывать эту подпрограмму и выводить на экран результат ее работы.
16. Написать MPI подпрограмму, получающую в качестве аргументов соответствующую часть (блок) массива A вещественных чисел, целое число N, являющееся длиной этого массива, номер процесса K, общее количество процессов P, и заменяющую

каждый элемент массива (для которого это возможно) на среднее арифметическое соседних элементов. При этом должна быть обеспечена равномерная загрузка всех процессов. Основная программа должна начинать работу с MPI, вводить число N и массив A (из файла или по заданной формуле), вызывать эту подпрограмму и выводить на экран результат ее работы.

17. Написать MPI подпрограмму, получающую в качестве аргументов соответствующие части (блоки) NxN массива A вещественных чисел, вспомогательный массив B вещественных чисел длины N, целое число N, номер процесса K, общее количество процессов P, и заменяющую каждый элемент A_{ij} матрицы A (для которого это возможно) на (A_{i+1,j}+A_{i-1,j}+A_{i,j+1}+A_{i,j-1}-4A_{i,j}). При этом должна быть обеспечена равномерная загрузка всех процессов. Основная программа должна начинать работу с MPI, вводить число N и массив A (из файла или по заданной формуле), вызывать эту подпрограмму и выводить на экран результат ее работы.
18. Написать MPI подпрограмму, получающую в качестве аргументов соответствующие части (блоки) NxN массива A вещественных чисел, вспомогательный массив B вещественных чисел длины 2N, целое число N, номер процесса K, общее количество процессов P, и заменяющую каждый элемент A_{ij} матрицы A (для которого это возможно) на (A_{i+2,j}+A_{i-2,j}+A_{i,j+2}+A_{i,j-2}-4A_{i,j}). При этом должна быть обеспечена равномерная загрузка всех процессов. Основная программа должна начинать работу с MPI, вводить число N и массив A (из файла или по заданной формуле), вызывать эту подпрограмму и выводить на экран результат ее работы.

6. Перечень основной и дополнительной учебной литературы, ресурсов информационно-телекоммуникационной сети «Интернет»:

1. К.Ю. Богачев. Основы параллельных вычислений. Москва: ЦПИ при механико-математическом ф-те МГУ им. М.В.Ломоносова, 2002. 352 с.
2. К.Ю. Богачев. Основы параллельного программирования. Москва: Бином, 2003. 342 с. ISBN: 5-94774-037-0.
3. В.В.Воеводин, Вл.В.Воеводин. Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.

Приложение утверждено на заседании кафедры вычислительной математики
Протокол № 1 от 7 сентября 2015 г.