МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Астраханский государственный университет имени В. Н. Татищева»
(Астраханский государственный университет им. В. Н. Татищева)

| | |
|---|---|
| СОГЛАСОВАНО<br>Руководитель ОПОП | УТВЕРЖДАЮ<br>Заведующий кафедрой ФМО |
| _____ М.В. Коломина | _____ И.А. Байгушева |
| «29» августа 2023 г. | «29» августа 2023 г. |

# РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

# «СУПЕРКОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ»

| | |
|---|---|
| Составитель(и) | **Гордеев И.И., к. ф.-м. н., доцент кафедры ФМО** |
| Направление подготовки / специальность | **01.03.02 Прикладная математика и информатика** |
| Направленность (профиль) ОПОП | |
| Квалификация (степень) | **бакалавр** |
| Форма обучения | **очная** |
| Год приёма | **2021** |
| Курс | **3** |
| Семестр(ы) | **5-6** |

Астрахань – 2022

## 1. ЦЕЛИ И ЗАДАЧИ ОСВОЕНИЯ ДИСЦИПЛИНЫ (МОДУЛЯ)

**1.1. Целью освоения дисциплины (модуля)** «Суперкомпьютерные технологии программирования» является:

- формирование у студентов представления о порядке постановки и решения задач для ЭВМ, функционирующих в современных информационно-вычислительных системах;
- изучение студентами одного из алгоритмических языков высокого уровня, в том числе получение практических навыков программирования задач в интегрированной среде разработчика;
- ознакомление с основами разработки программного, аппаратного и пользовательского интерфейса программных продуктов;
- приобретение опыта постановки и решения задач, связанных со сбором, обработкой и представлением данных в современных информационно-вычислительных системах.

**1.2. Задачи освоения дисциплины (модуля):**

- изучение теоретических основ современных технологий программирования и получение практических навыков их реализации;
- изучение технологий параллельного программирования.

## 2. МЕСТО ДИСЦИПЛИНЫ (МОДУЛЯ) В СТРУКТУРЕ ОПОП

**2.1. Учебная дисциплина (модуль)** «Суперкомпьютерные технологии программирования» относится к обязательной части учебного плана.

**2.2. Для изучения данной учебной дисциплины (модуля) необходимы следующие знания, умения, навыки, формируемые предшествующими учебными дисциплинами (модулями):**

- Архитектура компьютера,
- Объектно-ориентированное программирование.

**Знания**: представление целочисленных и вещественных данных на компьютере.

**Умения**: выбирать типы данных, соответствующие решаемой задаче.

**Навыки**: чтение и запись данных в шестнадцатеричном виде.

**2.3. Последующие учебные дисциплины (модули) и (или) практики, для которых необходимы знания, умения, навыки, формируемые данной учебной дисциплиной (модулем):**

- Технологии высокопроизводительных вычислений,
- Современные методы анализа данных.

## 3. ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

Процесс изучения дисциплины (модуля) направлен на формирование элементов следующих компетенций в соответствии с ФГОС ВО и ОПОП ВО по данному направлению подготовки (специальности):

***а) универсальных (УК):***

- способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач (УК-1);

***б) профессиональных (ПК):***

- способность собирать, обрабатывать, интерпретировать данные и проводить под руководством научное исследование в области математических и (или) естественных наук (ПК-1);
- способность применять высокопроизводительные суперкомпьютерные технологии при решении задач в профессиональной деятельности (ПК-2);

- способен разрабатывать требования и проектировать программное обеспечение (ПК-4).

**Таблица 1 – Декомпозиция результатов обучения**

| Код и наименование компетенции | Планируемые результаты обучения по дисциплине (модулю) | | |
|---|---|---|---|
| | Знать (1) | Уметь (2) | Владеть (3) |
| УК-1. Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач | ИУК-1.1.1 способы поиска, критического анализа и синтеза информации. | ИУК-1.2.1 выполнять поиск необходимой информации, её критический анализ и обобщать результаты анализа для решения поставленной задачи. | ИУК-1.3.1 использованием системного подхода для решения поставленных задач. |
| ПК-1. Способность собирать, обрабатывать, интерпретировать данные и проводить под руководством научное исследование в области математических и (или) естественных наук | ИПК-1.1.1 обладает базовыми знаниями, полученными в области математических и (или) естественных наук. Знает современные методы сбора и анализа полученного материала. | ИПК-1.2.1 собирать и обрабатывать статический, экспериментальный, теоретический, графический и т.п. материал, необходимый для построения математических моделей, расчетов и конкретных практических выводов, использовать методы прикладной математики и информатики для решения научно-исследовательских и прикладных задач. | ИПК-1.3.1 практическим опытом оформления результатов научного труда в соответствии с современными требованиями. |
| ПК-2. Способность применять высокопроизводительные суперкомпьютерные технологии при решении задач в профессиональной деятельности | ИПК-2.1.1 основные алгоритмы высокопроизводительных вычислений. | ИПК-2.2.1 разрабатывать и реализовывать алгоритмические и программные решения в области высокопроизводительных суперкомпьютерных технологий. | ИПК-2.3.1 практическим опытом применения высокопроизводительных суперкомпьютерных технологий при решении задач в профессиональной деятельности. |
| ПК-4. Способен разрабатывать требования и проектировать программное обеспечение | ИПК-4.1.1 современные технологии проектирования и производства программного продукта, методику анализа требований и вариантов реализации программного продукта. | ИПК-4.2.1 использовать подобные технологии при создании программных продуктов. | ИПК-4.3.1 практическим опытом применения подобных технологий. |

## 4. СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

Объем дисциплины (модуля) в 5 семестре составляет 6 зачетных единиц, в том числе 119 часов, выделенных на контактную работу обучающихся с преподавателем (из них 34 часа – лекции, 68 часов – лабораторные работы), и 114 часов – на самостоятельную работу обучающихся; в 6 семестре составляет 7 зачетных единиц, в том числе 108 часов, выделенных на контактную работу обучающихся с преподавателем (из них 18 часов – лекции, 72 часа – лабораторные работы), и 162 часа – на самостоятельную работу обучающихся.

**Таблица 2 – Структура и содержание дисциплины (модуля)**

| Раздел, тема дисциплины (модуля) | Семестр | Контактная работа (в часах) | | | Самост. работа | | Форма текущего контроля успеваемости, форма промежуточной аттестации |
|---|---|---|---|---|---|---|---|
| | | Л | ПЗ | ЛР | КР | СР | |
| Раздел 1. Общая характеристика процесса разработки ПС | 5 | 8 | | 16 | | 28 | Опрос. Отчет по лабораторной работе |
| Раздел 2. Понятие жизненного цикла ПО. Модели жизненного цикла. | | 8 | | 16 | | 28 | Опрос. Отчет по лабораторной работе |
| Раздел 3. Типовое проектирование | | 6 | | 12 | | 20 | Опрос. Отчет по лабораторной работе |
| Раздел 4. Структурная модель предметной области | | 4 | | 8 | | 14 | Опрос. Отчет по лабораторной работе |
| Раздел 5. Потоки данных. Моделирование данных. | | 4 | | 8 | | 12 | Опрос. Отчет по лабораторной работе |
| Раздел 6. Планирование и оценка проекта | | 4 | | 8 | | 12 | Опрос. Отчет по лабораторной работе |
| Раздел 7. Технологии параллельного программирования. | 6 | 4 | | 16 | | 32 | Опрос. Отчет по лабораторной работе |
| Раздел 8. Технология MPI. | | 6 | | 24 | | 48 | Опрос. Контрольная работа №1 |
| Раздел 9. Технология OpenMP. | | 4 | | 16 | | 32 | Опрос. Отчет по лабораторной работе |
| Раздел 10. Средства отладки параллельных программ. | | 4 | | 16 | | 32 | Опрос. Контрольная работа №2 |
| Курсовая работа | | | | | 18 | | Курсовая работа |
| **Итого** | | **52** | | **140** | **18** | **258** | **Экзамен, экзамен** |

Примечание: Л – лекция; ПЗ – практическое занятие, семинар; ЛР – лабораторная работа; КР – курсовая работа; СР – самостоятельная работа.

**Таблица 3 – Матрица соотнесения разделов, тем учебной дисциплины (модуля) и формируемых компетенций**

| Раздел, тема дисциплины (модуля) | Кол-во часов | Код компетенции | | | | Общее количество компетенций |
|---|---|---|---|---|---|---|
| | | УК-1 | ПК-1 | ПК-2 | ПК-4 | |
| Раздел 1. Общая характеристика процесса разработки ПС | 52 | + | + | + | + | *3* |
| Раздел 2. Понятие жизненного цикла ПО. Модели жизненного цикла. | 52 | + | + | + | + | *3* |
| Раздел 3. Типовое проектирование | 39 | + | + | + | + | *3* |
| Раздел 4. Структурная модель предметной области | 25 | + | + | + | + | *3* |
| Раздел 5. Потоки данных. Моделирование данных. | 24 | + | + | + | + | *3* |
| Раздел 6. Планирование и оценка проекта | 24 | + | + | + | + | *3* |
| Раздел 7. Технологии параллельного программирования. | 52 | + | + | + | + | *3* |
| Раздел 8. Технология MPI. | 78 | + | + | + | + | *3* |
| Раздел 9. Технология OpenMP. | 52 | + | + | + | + | *3* |
| Раздел 10. Средства отладки параллельных программ. | 52 | + | + | + | + | *3* |
| Курсовая работа | 18 | + | + | + | + | *3* |

| Раздел, тема дисциплины (модуля) | Кол-во часов | Код компетенции | | | | Общее количество компетенций |
|---|---|---|---|---|---|---|
| | | УК-1 | ПК-1 | ПК-2 | ПК-4 | |
| **Итого** | **468** | | | | | **3** |

**Краткое содержание каждой темы дисциплины (модуля)**

**Раздел 1. Общая характеристика процесса разработки ПС**

Особенности современных программных проектов. Этапы создания ПС.

**Раздел 2. Понятие жизненного цикла ПО. Модели жизненного цикла**

Процессы жизненного цикла: основные, вспомогательные, организационные. Содержание и взаимосвязь процессов жизненного цикла ПО. Модели жизненного цикла: каскадная, модель с промежуточным контролем, спиральная. Стадии жизненного цикла ПО. Регламентация процессов проектирования в национальных и международных стандартах. Технологии проектирования ПС. Каноническое проектирование.

**Раздел 3. Типовое проектирование**

Типовое проектирование. Формирование системы требований к ПО. Формирование и анализ первичных требований. Методы углубленного анализа требований.

**Раздел 4. Структурная модель предметной области**

Объектная структура. Функциональная структура. Структура управления. Организационная структура. Методология функционального моделирования. Состав функциональной модели. Иерархия диаграмм. Типы связей между функциями. Моделирование потоков данных. Диаграммы потоков данных. Внешние сущности. Системы и подсистемы. Процессы. Накопители данных.

**Раздел 5. Потоки данных. Моделирование данных**

Диаграммы потоков работ. Создание логической и физической модели.

**Раздел 6. Планирование и оценка проекта**

Формирование и анализ первичных требований. Углубленный анализ требований. Надежность ИС. Понятие надежности программного обеспечения ИС. Верификация. Обзор традиционных и современных методов верификации. Тестирование. Основные понятия. Обзор методов тестирования.

**Раздел 7. Технологии параллельного программирования**

**Раздел 8. Технология MPI**

**Раздел 9. Технология OpenMP**

**Раздел 10. Средства отладки параллельных программ**

## 5. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ПРЕПОДАВАНИЮ И ОСВОЕНИЮ ДИСЦИПЛИНЫ (МОДУЛЯ)

### 5.1. Указания для преподавателей по организации и проведению учебных занятий по дисциплине (модулю)

**Лекционные занятия**

Основной формой реализации теоретического обучения является лекция, которая представляет собой систематическое, последовательное изложение преподавателем-лектором учебного материала теоретического характера. Цель лекции – организация целенаправленной познавательной деятельности студентов по овладению программным материалом учебной дисциплины.

Порядок подготовки лекционного занятия включает в себя выполнение следующих этапов:

- изучение требований программы дисциплины,

- определение целей и задач лекции,
- разработка плана проведения лекции,
- подбор литературы (ознакомление с методической литературой, публикациями периодической печати по теме лекционного занятия),
- отбор необходимого и достаточного по содержанию учебного материала,
- определение методов, приемов и средств поддержания интереса, внимания, стимулирования творческого мышления студентов,
- написание конспекта лекции.

Лекция должна включать следующие разделы:

- формулировку темы лекции;
- указание основных изучаемых разделов или вопросов и предполагаемых затрат времени на их изложение;
- изложение вводной части;
- изложение основной части лекции;
- краткие выводы по каждому из вопросов;
- заключение;
- рекомендации литературных источников по излагаемым вопросам.

**Лабораторные занятия**

Лабораторное занятие − целенаправленная форма организации педагогического процесса, направленная на углубление научно-теоретических знаний и овладение определенными методами работы, в процессе которых вырабатываются умения и навыки выполнения тех или иных учебных действий в данной сфере науки. Они развивают научное мышление и речь, позволяют проверить знания студентов и выступают как средства оперативной обратной связи.

Правильно организованные лабораторные занятия ориентированы на решение следующих задач:

- обобщение, систематизация, углубление, закрепление полученных в процессе самостоятельной работы теоретических знаний по дисциплине (предмету);
- формирование практических умений и навыков, необходимых в будущей профессиональной деятельности, реализация единства интеллектуальной и практической деятельности;
- выработка при решении поставленных задач таких профессионально значимых качеств, как самостоятельность, ответственность, точность, творческая инициатива.

Состав заданий для практического занятия должен быть спланирован с расчетом, чтобы за отведенное время они могли быть качественно выполнены большинством обучающихся.

Лабораторные занятия должны так быть организованы, чтобы студенты ощущали нарастание сложности выполнения заданий, испытывали бы положительные эмоции от переживания собственного успеха в учении, поисками правильных и точных решений.

**Самостоятельная работа**

Самостоятельная работа – это вид учебной деятельности, которую студент совершает в установленное время и в установленном объеме индивидуально или в группе, без непосредственной помощи преподавателя (но при его контроле), руководствуясь сформированными ранее представлениями о порядке и правильности выполнения действий.

В учебном процессе образовательного учреждения выделяются два вида самостоятельной работы:

1) аудиторная – выполняется на учебных занятиях, под непосредственным руководством преподавателя и по его заданию (выполнение самостоятельных работ; выполнение контрольных и практических работ; решение задач).

2) внеаудиторная – выполняется по заданию преподавателя, но без его непосредственного участия (подготовка к аудиторным занятиям; изучение учебного материала, вынесенного на

самостоятельную проработку; выполнение домашних заданий разнообразного характера; выполнение индивидуальных заданий, направленных на развитие у студентов самостоятельности и инициативы; подготовка к контрольной работе). Внеаудиторные самостоятельные работы представляют собой логическое продолжение аудиторных занятий, проводятся по заданию преподавателя, который инструктирует студентов и устанавливает сроки выполнения задания.

**5.2. Указания для обучающихся по освоению дисциплины (модулю)**

**Лекция. Как ее слушать и записывать**

1. Лекция основной вид обучения в вузе.
2. В лекции излагаются основные положения теории, ее понятия и законы, приводятся факты, показывающие связь теории с практикой.
3. Накануне лекции необходимо повторить содержание предыдущей лекции, а затем посмотреть тему очередной лекции по программе (по плану лекций).
4. Полезно вести записи (конспекты) лекций: для непонятных вопросов оставлять место при работе над темой лекции с учебными пособиями.
5. Записи лекций следует вести в отдельной тетради, оставляя место для дополнений во время самостоятельной работы.
6. При конспектировании лекций выделяйте главы и разделы, параграфы, подчеркивайте основное.

**Лабораторное занятие**

Лабораторное занятие – наиболее активный вид учебных занятий в вузе. Он предполагает самостоятельную работу над учебными пособиями.

К каждому лабораторному занятию нужно готовиться. Подготовку следует начинать с повторения теории (по учебному пособию). После этого нужно решать задачи из предложенного домашнего задания.

**Организация самостоятельной работы**

1. Бюджет времени студента определяется временем, отведенным на занятия по расписанию и на самостоятельную работу. Задание и материал для самостоятельной работы дается во время учебных занятий, на этих же занятиях преподаватель осуществляет контроль за самостоятельной работой.
2. Для выполнения объема самостоятельной работы необходимо заниматься в среднем 4 часа (академических) ежедневно.
3. Начинать самостоятельные занятия следует с первых же дней семестра, установив определенный порядок, равномерный ритм на весь семестр. Полезно для этого составить расписание порядка дня.

**Таблица 4 – Содержание самостоятельной работы обучающихся**

| Вопросы, выносимые на самостоятельное изучение | Кол-во часов | Форма работы |
|---|---|---|
| Раздел 1. Общая характеристика процесса разработки ПС | 24 | Подготовка информационного сообщения |
| Раздел 2. Понятие жизненного цикла ПО. Модели жизненного цикла. | 24 | Подготовка информационного сообщения |
| Раздел 3. Типовое проектирование | 18 | Подготовка информационного сообщения |
| Раздел 4. Структурная модель предметной области | 11 | Подготовка информационного сообщения |
| Раздел 5. Потоки данных. Моделирование данных. | 10 | Подготовка информационного сообщения |
| Раздел 6. Планирование и оценка проекта | 10 | Подготовка информационного сообщения |
| Раздел 7. Технологии параллельного программирования. | 28 | Подготовка информационного сообщения |

| Вопросы, выносимые на самостоятельное изучение | Кол-во часов | Форма работы |
|---|---|---|
| Раздел 8. Технология MPI. | 42 | Подготовка информационного сообщения |
| Раздел 9. Технология OpenMP. | 28 | Подготовка информационного сообщения |
| Раздел 10. Средства отладки параллельных программ. | 28 | Подготовка информационного сообщения |
| Курсовая работа | 18 | Курсовая работа |

**5.3. Виды и формы письменных работ, предусмотренных при освоении дисциплины (модуля), выполняемые обучающимися самостоятельно**

Дисциплиной «Технологии программирования» предусмотрены такие формы самостоятельной работы, как работа с источниками информации (литературой, электронными пособиями и т.д.), написание информационного сообщения, написание курсовой работы.

При написании курсовой работы необходимо придерживаться требований по оформлению, представленных в Приложении 1. Титульный лист оформляется в соответствии с Приложением 2. Темы курсовых работ представлены в пункте 7.3.

**Примерные темы информационного сообщения:**

1. Особенности современных программных проектов
2. Каноническое проектирование
3. Методы углубленного анализа требований
4. Типы связей между функциями
5. Создание логической и физической модели
6. Обзор методов тестирования
7. Параллельное программирование на графических процессорах
8. Группы процессов в MPI
9. Управление данными для параллельно выполняемых потоков
10. Отладка параллельных программ в Visual Studio

**Требования к оформлению информационного сообщения**

1. Формат страницы: А4.
2. Поля: левое - 3 см, правое - 1,5 см, верхнее - 2 см, нижнее - 2 см.
3. Текст, отформатированный с помощью стилей: Заголовок 1: шрифт - Times New Roman; размер шрифта - 16; начертание - полужирный; все буквы ПРОПИСНЫЕ; выравнивание - по центру; межстрочный интервал - полуторный; интервал после абзаца - 0,21. Заголовок 2: шрифт - Times New Roman; размер шрифта - 14; начертание - полужирный; выравнивание - по левому краю; отступ первой строки - 1,25 см; интервал перед абзацем - 0,42 см; интервал после абзаца - 0,21 см; межстрочный интервал - полуторный. Основной текст: шрифт - Times New Roman; размер шрифта - 14; межстрочный интервал - полуторный; отступ первой строки - 1,25 см; интервал после абзаца - 0,21 см; выравнивание - по ширине.
4. Создать автоматическую нумерацию глав и подглав. Разделы «Введение», «Заключение» и «Список литературы» не нумеруются.
5. Формулы должны быть набраны с помощью редактора формул.
6. Изображения, формулы, таблицы, схемы, диаграммы должны быть подписаны и пронумерованы (автоматическая подпись) с указанием ссылок на них.
7. Документ должен содержать:
    1) Титульный лист.
    2) Содержание.
    3) Основной текст
        - введение,
        - основная часть,
        - заключение.

4) Список использованной литературы.

ТИТУЛЬНЫЙ ЛИСТ

Титульный лист должен содержать:

1. Наименование организации, где выполнялась работа.
2. Наименование работы.
3. Сведения об авторе (должность, Ф.И.О.).
4. Место и дата выполнения работы.

СОДЕРЖАНИЕ

Содержание работы разместить на отдельном листе. Содержание должно быть сформировано автоматически и содержать все заголовки и подзаголовки с указанием номера страницы.

ОСНОВНОЙ ТЕКСТ

Введение. В аннотации (3-5 предложений) кратко указывается, какой проблеме посвящается методическая разработка, какие вопросы раскрывает, кому может быть полезна.

Основная часть. Количество разделов в основной части работы может изменяться в зависимости от объема имеющегося материала и поставленной перед собой целью. В этом разделе подробно рассматриваются все вопросы, внесенные в содержание. По ходу изложения можно представлять необходимые таблицы и рисунки. Нумерация по мере появления в тексте (например, рис. 1, таблица 3. и т. д.). Таблица должна иметь название и «шапку» с наименованием колонок.

СПИСОК ЛИТЕРАТУРЫ

В список литературы по порядку включаются те источники, которые использовались при написании работы. На все перечисленные в «Списке литературы» источники должны быть ссылки в основном тексте работы в виде номеров из списка, заключенных в квадратные скобки. Пример: [5], где 5 это номер по порядку в списке использованных источников.

## 6. ОБРАЗОВАТЕЛЬНЫЕ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

### 6.1. Образовательные технологии

**Таблица 5 – Образовательные технологии, используемые при реализации учебных занятий**

| Раздел, тема дисциплины (модуля) | Форма учебного занятия | | |
|---|---|---|---|
| | Лекция | Практическое занятие, семинар | Лабораторная работа |
| Раздел 1. Общая характеристика процесса разработки ПС | *Обзорная лекция* | *Не предусмотрено* | *Выполнение лабораторной работы №1* |
| Раздел 2. Понятие жизненного цикла ПО. Модели жизненного цикла. | *Интерактивная лекция* | *Не предусмотрено* | *Выполнение лабораторной работы №2* |
| Раздел 3. Типовое проектирование | *Интерактивная лекция* | *Не предусмотрено* | *Выполнение лабораторной работы №3* |
| Раздел 4. Структурная модель предметной области | *Интерактивная лекция* | *Не предусмотрено* | *Выполнение лабораторной работы №4* |
| Раздел 5. Потоки данных. Моделирование данных. | *Интерактивная лекция* | *Не предусмотрено* | *Выполнение лабораторной работы №5* |
| Раздел 6. Планирование и оценка проекта | *Интерактивная лекция* | *Не предусмотрено* | *Выполнение лабораторной работы №6* |

| | | | |
|---|---|---|---|
| Раздел 7. Технологии параллельного программирования. | *Интерактивная лекция* | *Не предусмотрено* | *Выполнение лабораторной работы №7* |
| Раздел 8. Технология MPI. | *Интерактивная лекция* | *Не предусмотрено* | *Выполнение лабораторной работы №8 и №9* |
| Раздел 9. Технология OpenMP. | *Интерактивная лекция* | *Не предусмотрено* | *Выполнение лабораторной работы №10* |
| Раздел 10. Средства отладки параллельных программ. | *Интерактивная лекция* | *Не предусмотрено* | *Выполнение лабораторной работы №11* |

**6.2. Информационные технологии**

При реализации различных видов учебной и внеучебной работы используются следующие информационные технологии:

1) использование возможностей интернета в учебном процессе (использование сайта преподавателя (рассылка заданий, предоставление выполненных работ, ответы на вопросы, ознакомление обучающихся с оценками и т. д.));

2) использование электронных учебников и различных сайтов (например, электронных библиотек, журналов и т. д.) как источников информации;

3) использование возможностей электронной почты преподавателя;

4) использование средств представления учебной информации (электронных учебных пособий и практикумов, применение новых технологий для проведения очных (традиционных) лекций и семинаров с использованием презентаций и т. д.);

5) использование интегрированных образовательных сред, где главной составляющей являются не только применяемые технологии, но и содержательная часть, т. е. информационные ресурсы (доступ к мировым информационным ресурсам, на базе которых строится учебный процесс);

6) использование виртуальной обучающей среды (LMS Moodle «Цифровое обучение») или иных информационных систем, сервисов и мессенджеров.

**6.3. Программное обеспечение, современные профессиональные базы данных и информационные справочные системы**

**6.3.1. Программное обеспечение**

Перечень программного обеспечения (***состав подлежит обновлению при необходимости)***

| **Наименование программного обеспечения** | **Назначение** |
|---|---|
| Adobe Reader | Программа для просмотра электронных документов |
| WinDjView | Программа для просмотра электронных документов |
| LMS Moodle | Образовательный портал ФГБОУ ВО «АГУ» |
| Microsoft Office | Пакет офисных программ |
| LibreOffice | Пакет офисных программ |
| 7-zip | Архиватор |
| Microsoft Windows | Операционная система |
| Kaspersky Endpoint Security | Средство антивирусной защиты |
| Google Chrome | Браузер |
| Opera | Браузер |
| Python 3.10 | Язык программирования |
| Idle | Среда разработки для языка Python |
| Microsoft Visual Studio | Среда программирования |
| Code ::Blocks 20.03 | Среда программирования |

| | |
|---|---|
| MinGW | Компилятор для языка программирования C++ |
| Microsoft MPI | Среда выполнения параллельных процессов |
| Microsoft MPI SDK | Пакет разработчика MPI |

**6.3.2. Современные профессиональные базы данных и информационные справочные системы**

1. Электронная библиотека «Астраханский государственный университет» собственной генерации на платформе ЭБС «Электронный Читальный зал – БиблиоТех». https://biblio.asu.edu.ru.
2. Электронно-библиотечная система (ЭБС) ООО «Политехресурс» «Консультант студента». https://www.studentlibrary.ru.
3. Электронная библиотечная система издательства ЮРАЙТ, раздел «Легендарные книги». https://www.biblio-online.ru, https://urait.ru.
4. Электронный каталог Научной библиотеки АГУ на базе MARK SQL НПО «Информ-систем». https://library.asu.edu.ru.

## 7. ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ПРОВЕДЕНИЯ ТЕКУЩЕГО КОНТРОЛЯ И ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

### 7.1. Паспорт фонда оценочных средств

При проведении текущего контроля и промежуточной аттестации по дисциплине (модулю) «Суперкомпьютерные технологии программирования» проверяется сформированность у обучающихся компетенций, указанных в разделе 3 настоящей программы. Этапность формирования данных компетенций в процессе освоения образовательной программы определяется последовательным освоением дисциплин (модулей) и прохождением практик, а в процессе освоения дисциплины (модуля) – последовательным достижением результатов освоения содержательно связанных между собой разделов, тем.

**Таблица 6 – Соответствие разделов, тем дисциплины (модуля), результатов обучения по дисциплине (модулю) и оценочных средств**

| Контролируемый раздел, тема дисциплины (модуля) | Код контролируемой компетенции | Наименование оценочного средства |
|---|---|---|
| Раздел 1. Общая характеристика процесса разработки ПС | УК-1, ПК-1, ПК-2, ПК-4 | Лабораторная работа |
| Раздел 2. Понятие жизненного цикла ПО. Модели жизненного цикла | УК-1, ПК-1, ПК-2, ПК-4 | Лабораторная работа |
| Раздел 3. Типовое проектирование | УК-1, ПК-1, ПК-2, ПК-4 | Лабораторная работа |
| Раздел 4. Структурная модель предметной области | УК-1, ПК-1, ПК-2, ПК-4 | Лабораторная работа |
| Раздел 5. Потоки данных. Моделирование данных | УК-1, ПК-1, ПК-2, ПК-4 | Лабораторная работа |
| Раздел 6. Планирование и оценка проекта | УК-1, ПК-1, ПК-2, ПК-4 | Лабораторная работа |
| Раздел 7. Технологии параллельного программирования | УК-1, ПК-1, ПК-2, ПК-4 | Лабораторная работа |
| Раздел 8. Технология MPI | УК-1, ПК-1, ПК-2, ПК-4 | Лабораторная работа. Контрольная работа |
| Раздел 9. Технология OpenMP | УК-1, ПК-1, ПК-2, ПК-4 | Лабораторная работа |
| Раздел 10. Средства отладки параллельных программ | УК-1, ПК-1, ПК- | Лабораторная работа. |

| Контролируемый раздел, тема дисциплины (модуля) | Код контролируемой компетенции | Наименование оценочного средства |
|---|---|---|
| | 2, ПК-4 | Контрольная работа |
| Курсовая работа | УК-1, ПК-1, ПК-2, ПК-4 | Курсовая работа |

**7.2. Описание показателей и критериев оценивания компетенций, описание шкал оценивания**

**Таблица 7 – Показатели оценивания результатов обучения в виде знаний**

| Шкала оценивания | Критерии оценивания |
|---|---|
| 5 «отлично» | демонстрирует глубокое знание теоретического материала, умение обоснованно излагать свои мысли по обсуждаемым вопросам, способность полно, правильно и аргументированно отвечать на вопросы, приводить примеры |
| 4 «хорошо» | демонстрирует знание теоретического материала, его последовательное изложение, способность приводить примеры, допускает единичные ошибки, исправляемые после замечания преподавателя |
| 3 «удовлетворительно» | демонстрирует неполное, фрагментарное знание теоретического материала, требующее наводящих вопросов преподавателя, допускает существенные ошибки в его изложении, затрудняется в приведении примеров и формулировке выводов |
| 2 «неудовлетворительно» | демонстрирует существенные пробелы в знании теоретического материала, не способен его изложить и ответить на наводящие вопросы преподавателя, не может привести примеры |

**Таблица 8 – Показатели оценивания результатов обучения в виде умений и владений**

| Шкала оценивания | Критерии оценивания |
|---|---|
| 5 «отлично» | демонстрирует способность применять знание теоретического материала при выполнении заданий, последовательно и правильно выполняет задания, умеет обоснованно излагать свои мысли и делать необходимые выводы |
| 4 «хорошо» | демонстрирует способность применять знание теоретического материала при выполнении заданий, последовательно и правильно выполняет задания, умеет обоснованно излагать свои мысли и делать необходимые выводы, допускает единичные ошибки, исправляемые после замечания преподавателя |
| 3 «удовлетворительно» | демонстрирует отдельные, несистематизированные навыки, испытывает затруднения и допускает ошибки при выполнении заданий, выполняет задание по подсказке преподавателя, затрудняется в формулировке выводов |
| 2 «неудовлетворительно» | не способен правильно выполнить задания |

**7.3. Контрольные задания и иные материалы, необходимые для оценки результатов обучения по дисциплине (модулю)**

**Раздел 1. Общая характеристика процесса разработки ПС**

**Лабораторная работа № 1**

Реализовать первый алгоритм сортировки методом выборки. Предполагаем, что сортируемые элементы расположены в памяти в виде массива. Идея заключается в следующем: путем просмотра всех элементов массива среди них находится наименьший и запоминается его номер, а затем элемент с запомненным номером обменивается с нулевым элементом массива. В результате наименьший элемент оказывается на своем месте, т.е. в начале массива. Затем среди оставшихся элементов (начиная с первого) опять ищется наименьший и запоминается его номер, этот элемент обменивается уже с первым элементом, в результате нулевой и первый элементы массива уже стоят на своих местах в порядке сортировки. Также делаем выборку начиная со второго, третьего и так далее элементов, которые ставятся на свои места, пока не будет отсортирован весь массив.

## Раздел 2. Понятие жизненного цикла ПО. Модели жизненного цикла

**Лабораторная работа № 2**

Реализовать второй алгоритм сортировки – «Метод вставки». Идея метода вставки основывается на следующих соображениях: если имеется уже отсортированный кусок массива длины $m$, то можно добавить к этому массиву один элемент и, получив отсортированный кусок длины $m+1$, если при этом предварительно определить, между какими элементами должен стоять новый, чтобы после вставки нового элемента массив оставался отсортированным. Используя эту идею можно отсортировать весь массив. При вставке последние элементы сдвигаются к концу массива, образуя свободное место для вставляемого. Таким образом, в самом начале можно считать, что у нас есть отсортированный массив из одного элемента, затем мы берем следующий элемент и, выполняя его вставку в отсортированную часть, получаем отсортированный массив из двух элементов и так далее, пока не получим отсортированный массив. Трудоемкость этого метода сортировки также пропорциональна $n^2$. В связи с тем, что может потребоваться сдвигать достаточно много элементов, т.е. при вставке в массив из одного элемента может потребоваться сдвинуть этот один элемент, при вставке в массив из двух элементов, может потребоваться сдвинуть оба элемента и так далее, при вставке последнего $n$-го элемента может потребоваться сдвинуть $n-1$ предыдущий элемент.

## Раздел 3. Типовое проектирование

**Лабораторная работа № 3**

Реализовать третий алгоритм – сортировка методом пузырька. Трудоемкость $O(n^2)$. Основная идея метода пузырька заключается в том, чтобы обменивать соседние элементы, стоящие в «неправильном» порядке. Например, можно просмотреть весь массив, сравнивая соседние пары элементов, сначала нулевой с первым, если первый меньше нулевого, то обмениваем их, затем сравниваем первый со вторым, если второй меньше первого, то обмениваем и т.д. После одного такого прохода по всему массиву можно гарантировать, что на свое место станет наибольший элемент в конце массива. Затем можно сделать такой же проход по массиву, обрабатывая уже $n-1$ элемент. После каждого очередного прохода очередной элемент с конца встает на свое место и так, пока не будет отсортирован весь массив.

## Раздел 4. Структурная модель предметной области

**Лабораторная работа № 4**

Реализовать сортировку с использованием библиотечной функции qsort. Существует большое количество модификаций алгоритмов сортировки, в частности разработано большое количество алгоритмов, имеющих более высокую эффективность, чем $n^2$, но трудоемкость, соответствующую $n^2$ в худшем случае. В частности, в стандартной библиотеке языка C++ имеется функция qsort, объявленная в библиотеке stdlib.h, имеющая среднюю трудоемкость $n \cdot \log_2 n$.

## Раздел 5. Потоки данных. Моделирование данных

### Лабораторная работа № 5

Реализовать сортировку слиянием с использованием алгоритма Джона фон Неймана. Алгоритм фон Неймана требует удвоенного объема памяти, поскольку предполагает использование второй копии массива и работает с гарантированной трудоемкостью $O(n \cdot \log_2 n)$.

Основная идея алгоритма заключается в том, что если у нас есть два отсортированных массива (оба имеют длину $k$), то мы можем получить из двух отсортированных массивов один общий отсортированный массив длины $2k$ за $2k$ элементарных операций. Предполагая, что оба массива отсортированы в порядке возрастания, начинаем просматривать оба массива, которые нужно объединить, запоминая в каждом массиве место, до которого мы дошли и сравниваем очередную пару элементов из первого и второго массивов. Определяем, какой из них меньше, и ставим меньший элемент в качестве очередного элемента в новом формируемом массиве. В том массиве, из которого взяли элемент, переходим к следующему элементу. Продолжаем так, пока не дойдем до конца одного из массивов. Тогда оставшиеся элементы второго массива добавляются в конец нового. В результате из двух отсортированных в порядке возрастания массивов получаем один отсортированный в порядке возрастания массив.

Для данного алгоритма несущественно, чтобы объединяемые массивы были одинакового размера. Алгоритм может точно также работать на двух массивах разной длины, например, с длинами $k1$ и $k2$. За $k1 + k2$ элементарных операций будет сформирован новый, отсортированный массив, поскольку после каждого сравнения двух элементов один из них становится в новый массив.

На основе идеи слияния двух отсортированных массивов можно реализовать сортировку любого массива. Возможны два подхода к описанию подобной сортировки: циклический и рекурсивный.

## Раздел 6. Планирование и оценка проекта

### Лабораторная работа № 6

Реализовать программу, которая генерирует массив из ста тысяч случайных целых чисел и записывает его в текстовый файл. Затем написать две версии программы сортировки. Обе версии программы читают массив целых чисел из файла, затем сортируют его в порядке возрастания и затем записывают отсортированный массив в новый текстовый файл. Различие между сортирующими программами состоит в том, что первая программа использует для сортировки метод пузырька, а вторая программа использует стандартную функцию qsort. В каждой из программ замеряется как общее выполнение всех действий, так и время сортировки массива.

Для оценки соответствия времени работы программы желаемым параметрам можно использовать в C++ функцию для замера прошедшего времени. Стандартные функции для работы со временем объявлены в заголовочном файле time.h. Если в программе требуется замерять очень большие промежутки времени, то можно использовать функцию time, которая возвращает количество секунд, прошедших с 1 января 1970 года. Для замеров небольших промежутков времени можно использовать функцию clock(). Данная функция возвращает целое количество прошедших «тиков» – временных тактов. 1 тик = некоторая доля секунды. На многих системах 1 тик может составлять 0,001 долю секунды. Узнать точное количество тиков в одной секунде можно при помощи константы CLOCKS_PER_SEC, объявленной в файле time.h. Формально в файле time.h функция clock() объявленная как возвращающая значения типа clock_t. Тип clock_t является целым типом со знаком (аналог size_t), однако он не привязан к разрядности программы, а зависит от технической реализации функции clock_t в конкретной среде программирования. Функция clock_t возвращает целое количество прошедших с момента запуска программы тиков. Чтобы при помощи данной функции замерить время выполнения некоторых операторов, можно применять схему с повторными вызовами функции clock_t и вычисления разности между значениями.

## Раздел 7. Технологии параллельного программирования

### Лабораторная работа № 7

На использование аргументов функции main. Реализовать программу, которая распечатывает свое имя и свои параметры с указанием номеров параметра.

Параметры запускаемой программы можно задавать разными способами, например, самым очевидным вариантом является прямое использование командной строки.

Второй способ более удобен для работы с файлами .bat, когда нужно многократно запускать программу с одними и теми же параметрами. Файлы .bat называют пакетными файлами, поскольку в них можно вставить сразу несколько команд или пакет команд.

Также аргументы командной строки могут задаваться в свойствах ярлыка для запуска программы.

При отладке программы в среде программирования удобно, чтобы запуск с параметрами мог происходить прямо из среды программирования. Для этого можно воспользоваться параметрами проекта.

## Раздел 8. Технология MPI

### Лабораторная работа № 8

Запуск требуемого количества процессов на компьютере. Чтобы запустить требуемое число процессов на компьютере, можно использовать команду **mpiexec** с параметрами:

**mpiexec –n 3 myprog.exe**, где 3 – это количество процессов, myprog.exe – программа, которую нужно запустить.

При тестировании MPI программ желательно проверять работу всего лишь одной копии MPI процесса, в таком случае можно указать значение количества процессов 1. При запуске одной копии процесса из-под среды выполнения MPI могут быть выявлены ошибки, которые не обнаруживаются при запуске из-под среды программирования или просто при запуске одной копии программы. Ошибки могут обнаруживаться благодаря тому, что среда выполнения MPI контролирует завершение всех стандартных операций, в частности среда выполнения MPI сообщает об ошибке, если процесс не сделал вызов MPI_Finalize(), либо были ошибки при операциях передачи данных. В частности, для обобщения и реализации произвольных схем организации программ, MPI процесс может посылать сообщение самому себе. При отладке MPI программ желательно проверять работу на 1-4 процессах.

Для подобного тестирования удобно использовать **bat файл**, в котором вставлены:

```
mpiexec -n 1 myprog.exe
mpiexec -n 2 myprog.exe
mpiexec -n 3 myprog.exe
mpiexec -n 4 myprog.exe
pause
```

При необходимости для тестирования можно на одном компьютере запускать больше процессов, чем имеется ядер, указав нужное число процессов. В этом случае следует иметь ввиду, что выигрыша от параллельной работы процессов может практически не быть, поскольку, когда процессов больше, чем ядер, то некоторые процессы вынуждены простаивать, ожидая, пока им будет выделено процессорное время.

### Лабораторная работа № 9

Задание к лабораторной работе «Параллельный запуск MPI программы на нескольких компьютерах»

Показать запуск **ipconfig /all**, скриншоты с ip адресом и именем своего компьютера. Создать папку со своей фамилией на сетевом диске и скопировать туда MPI программу показывающую имя узла с использованием функции MPI_Get_processor_name (пример 4 из Антонова), сделать скриншот. Создать конфигурационный файл для параллельного запуска, вставить в отчет. Выполнить параллельный запуск MPI программы на своем и одном или двух других компьютерах, сделать скриншот.

**Контрольная работа № 1**

Напишите программу, выполняющую параллельное вычисление заданной в Вашем варианте математической функции с использованием функций MPI_Send, MPI_Recv, MPI_Isend, MPI_Irecv.

**Формат входного файла:**

В первой строке два целых числа: начальное и конечное значение, для которых вычисляется функция.

**Формат выходного файла:**

В первой строке три числа: количество процессов, номер управляющего процесса и количество значений k, вычисляемых каждым процессом.

Далее выводятся номера вычислявших процессов и результаты вычислений для каждого процесса в следующем формате. Строка с номером процесса, затем k строк, содержащих по два числа, значение аргумента и значение функции.

Пример входного файла input.txt:

2 7

Пример соответствующего выходного output3.txt файла для функции f(n)=2n (всего 3 процесса, все вычисляют, управляет процесс 1):

3 1 2
1
4 8
5 10
0
2 4
3 6
2
6 12
7 14

## Раздел 9. Технология OpenMP

### Лабораторная работа № 10

Применение операции редукции в OpenMP. Для применения операции редукции после директивы omp, содержащей for или parallel, пишется ключевое слово reduction (операция: список переменных) если для разных переменных требует разные операции редукции, то ключевое слово reduction можно использовать несколько раз используя разные операции и разные списки переменных. Например, требуется распараллелить цикл, котором считается две суммы и произведение. Например, для определенности sum1 считается сумма положительных элементов, а в sum2 сумма отрицательных. Произведение считается для всех не нулевых элементов.

```
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
#include<omp.h>
void main()
{
int sum1=5,sum2=10,p=2;
int a[]={-2,1,-1,2,-1 };
int n=sizeof(a)/sizeof(a[0]);

#pragma omp parallel for reduction(+:sum1,sum2)reduction (*:p)
for(int i=0;i<n;i++)
{
	if(a[i]>0)
	sum1+=a[i];
	if(a[i]<0)
	sum2+=a[i];
	if(a[i])
	p*=a[i];
```

```
}
printf("sum1=%d,sum2=%d,p=%d\n",sum1,sum2,p);
}
```

Если в параллельной области указано использование редукции, то компилятор создает локальные копии переменных, указанные в скобках после ключевого слово reduction. Каждая локальная переменная инициализируется соответственно указанной операции. Операция + предполагает инициализацию локальной переменой 0.

Операция * предполагает инициализацию локальной переменной 1.

## Раздел 10. Средства отладки параллельных программ.

### Лабораторная работа № 11

Задание к лабораторной работе «Динамическая и статическая сборки, отладочная и итоговая версии».

Попробовать для простейшей программы (например, выводящей просто «Привет, мир») все 4 варианта сборки. Для каждого варианта сборки зафиксировать размер выполняемого файла с точностью до байта и для каждой версии исполняемого файла запустить программу dumpbin с параметром /DEPENDENTS и вставить скриншот программы dumpbin с результатом просмотра зависимостей в отчет по лабораторной. В отчет также вставить код программы, которая компилировалась.

### Контрольная работа № 2

Напишите программу, выполняющую параллельное вычисление заданной в Вашем варианте математической функции с использованием функций MPI_Scatter, MPI_Gather.

**Формат входного файла:**

В первой строке три целых числа: начальное и конечное значение аргумента, шаг, для которых вычисляется функция.

**Формат выходного файла:**

В первой строке три числа: количество процессов NumP, номер управляющего процесса и количество значений, вычисляемых всеми процессами.

Во второй строке NumP чисел: количество значений, вычисляемых каждым процессом, в порядке номеров процессов.

Далее выводятся номера вычислявших процессов и результаты вычислений для каждого процесса в следующем формате. Строка с номером процесса, затем k строк, содержащих по два числа, значение аргумента и значение функции.

Пример входного файла input.txt:

1 9 2

Пример соответствующего выходного output3.txt файла для функции f(n)=2n (всего 3 процесса, все вычисляют, управляет процесс 1):

3 1 5
2 2 1
0
1 2
3 6
1
5 10
7 14
2
9 12

## Примерные мини-проекты

### Исследование реализации MPI_Ibcast и MPI_Bcast

Следующая некорректная программа отрабатывает без ошибок при запуске нескольких процессов в реализации Microsoft MPI.

```
#include<iostream>
#include<fstream>
#include<mpi.h>
#include<sstream>

using namespace std;

int f(int x)
{
	return x*x;
}

int main (int argc, char *argv[])
{
	int nProc, rProc, root, kz;
	MPI_Init (&argc, &argv);
	MPI_Comm_size(MPI_COMM_WORLD, &nProc);
	MPI_Comm_rank (MPI_COMM_WORLD, &rProc);
	root=nProc/2;
	int *x=new int[2];
	if(rProc==root)
	{
		int x1, x2;
		fstream inFile;
		MPI_Request *zad_req;
		zad_req=new MPI_Request[nProc];
		inFile.open("input.txt");
		inFile>>x1>>x2;
		inFile.close();
		if(nProc==1) kz=x2-x1+1;
		else kz=(x2-x1+1)/nProc;
		int *y=new int[x2-x1+1];
		x[0]=x1;
		x[1]=x2;
		MPI_Ibcast(x,2,MPI_INT,root,MPI_COMM_WORLD,zad_req);
		MPI_Request gl_zad_req;
		MPI_Ibcast(x,2,MPI_INT,root,MPI_COMM_WORLD,&gl_zad_req);
		MPI_Wait(&gl_zad_req,MPI_STATUS_IGNORE);
		int *ot=new int[kz];
		int l=x[0]+rProc*kz;
		for(int i=l;i<l+kz;i++)
			ot[i-l]=f(i);
		MPI_Gather(ot,kz,MPI_INT,y,kz,MPI_INT,root,MPI_COMM_WORLD);
		stringstream name;
		name<<"output"<<nProc<<".txt";
		ofstream OutFile(name.str().c_str());
		OutFile<<nProc<<' '<<x2-x1+1<<endl;
		for(int i=0,l=x[0]; i<nProc; i++)
		{
			OutFile<<i<<endl;
			for(int j=0; j<kz; j++)
				OutFile<<l+j<<' '<<y[i*kz+j]<<endl;
			l+=kz;
		}
		OutFile.close();
		delete[]zad_req;
	}
	else
	{
		MPI_Request rab_zad_req, rab_otv_req;
		MPI_Ibcast(x,2,MPI_INT,root,MPI_COMM_WORLD,&rab_zad_req);
		MPI_Wait(&rab_zad_req,MPI_STATUS_IGNORE);
```

```
            kz=(x[1]-x[0]+1)/nProc;
            int *ot=new int[kz];
            int l=x[0]+rProc*kz;
            for(int i=l;i<l+kz;i++)
                  ot[i-l]=f(i);
            MPI_Gather(ot,kz,MPI_INT,ot,kz,MPI_INT,root,MPI_COMM_WORLD);
            delete []ot;
      }
      MPI_Finalize();
      delete[]x;
      return 0;
}
```

**Задание 1 (на 10 баллов).** Некорректность программы заключается в том, что у второго вызова MPI_Ibcast на управляющем процессе нет соответствующего вызова MPI_Ibcast на управляемых процессах. Для второго вызова MPI_Ibcast запускается MPI_Wait, который требует обязательного завершения неблокирующей операции. Возникает вопрос, каким же образом происходит завершение второго MPI_Ibcast на управляющем процессе?

*Предположение.* Завершение второй операции MPI_Ibcast на управляющем процессе возможно, если реализация MPI имеет внутреннюю буферизацию. При небольшом объеме посылаемой информации результат вызова MPI_Ibcast на отправляющем процессе может сводиться к тому, что оправляемые данные, указанные в первом аргументе, будут просто скопированы в буфер. После копирования отправляемых данных в буфер операция может считаться завершенной, поскольку теперь разрешается повторное использование буфера отправки для других целей. При этом, для отправляющего процесса, когда существует буферизация, может быть несущественно, принял ли кто-нибудь вообще эти данные. Если это предположение верно, то в реализации MPI должен быть максимальный размер буфера, который допускает такую ситуацию. При отправке большого объема данных буфера в реализации MPI может не хватить, и тогда MPI_Wait будет ждать реальной отправки данных получающим процессам, и на этом месте управляющий процесс зависнет.

*Задание.* Проверить, при каком максимальном объеме отправляемых данных программа еще не виснет.

**Задание 2 (на 10 баллов).** Попытаться увеличить некорректность программы, делая на управляющем процессе несколько лишних вызовов MPI_Ibcast, у которых нет соответствующих вызовов MPI_Ibcast на управляемых процессах. Для лишних вызовов MPI_Ibcast запускать MPI_Wait. Исследовать при небольшом количестве отправляемых данных, сколько можно сделать лишних вызовов MPI_Ibcast без зависания на управляющем процессе?

**Задание 3 (на 10 баллов).** Проверить, как будет работать подобный некорректный подход при наличии лишних вызовов блокирующей функции MPI_Bcast.

**Темы курсовых работ**

1. Одновременное применение OpenMP и MPI в параллельных методах Монте-Карло.
2. Применение возможностей стандарта C++11 в параллельных методах Монте-Карло.
3. Применение OpenMP в параллельных вычислениях для решения задач теории перколяции.
4. Применение возможностей стандарта C++11 для параллельных вычислений в прямых методах решения СЛАУ.
5. Применение возможностей стандарта C++11 в параллельных вычислениях для решения систем обыкновенных дифференциальных уравнений.
6. Применение параллельных вычислений для нахождения мостов в графе.
7. Применение MPI для параллельного поиска для нахождения шарниров в графе.
8. Применение параллельных вычислений для проверки графа на рёберную k-связность.
9. Применение параллельных вычислений для проверки графа на планарность.

10. Одновременное применение OpenMP и MPI для параллельного поиска кратчайших путей между вершинами в графе.
11. Одновременное применение OpenMP и MPI параллельных вычислений в алгоритме Хошена-Копельмана разметки связных компонент графа.
12. Применение OpenMP для параллельного поиска наибольшего независимого множества в графе.
13. Применение MPI для параллельного поиска наибольшего паросочетания в графе.
14. Применение возможностей стандарта C++11 для параллельной раскраски графов.
15. Применение параллельных вычислений для поиска сильно связных компонент в орграфах.
16. Применение MPI для параллельной проверки графа на вершинную k-связность.
17. Применение параллельных вычислений для поиска сильно связных компонент в орграфах.
18. Применение параллельных вычислений для проверки изоморфизма графов.
19. Применение возможностей стандарта C++11 для параллельной реализации проверки гомеоморфизма графов.
20. Применение параллельных вычислений для поиска граней в плоском графе.
21. Применение возможностей стандарта C++11 для параллельной реализации поиска гамильтоновых путей в графах.
22. Применение MPI для параллельного решения задачи коммивояжера.
23. Применение параллельных вычислений для реализации алгоритм Лиса (Leath) генерации и анализа случайных графов.
24. Применение возможностей стандарта C++11 для параллельной реализации волнового алгоритм Ли.
25. Применение параллельных вычислений для решения задачи о китайском почтальоне.
26. Применение параллельных вычислений для построения паросочетаний.
27. Применение параллельных вычислений для реализации алгоритма Тарьяна-Вишкина.
28. Применение параллельных вычислений для анализа генераторов псевдослучайных чисел.

**Перечень вопросов и заданий,**
**выносимых на экзамен / зачёт / дифференцированный зачёт**
**Вопросы к экзамену (5 семестр)**

1. Обзор технологий программирования. Процесс разработки программного обеспечения (ПО).
2. Этапы разработки ПО.
3. Гибкая методология управления разработкой программного продукта.
4. Анализ требований к спецификациям по времени выполнения программы и спецификаций оборудования на котором выполняется программа. Анализ возможности выполнения требований по времени.
5. Конвейер процессора. Увеличение частоты процессора в сравнении с увеличением числа ядер.
6. Сортировка методом пузырька, методом выборки и методом вставки. Ручное прослеживание программы.
7. Выбор предпочтительного направления организации программы. Анализ сложности выполнения алгоритмов на примере различных алгоритмов сортировки.
8. Асимптотические оценки сложности: Омега большое, О большое, Тета большое и соответствующие им графики. Худший, лучший случаи.
9. Профилировка программы. Среднестатистическая оценка сложности.
10. Замеры времени работы программы. Функции clock() и std::chrono::now(). Арифметический тип std::ratio.

11. Функции QueryPerformanceCounter и QueryPerformanceFrequency. Точность замеров времени.
12. Эффективные способы сортировки. Сортировка методом слияния.
13. Устойчивость сортировки. Быстрая сортировка(Quicksort) или сортировка Хоара.
14. Возможное обобщение подпрограмм в языках Си и C++. Использование шаблонов и указателей на функцию. Использование библиотечных функции для быстрого написания программ. Функции sort и qsort.
15. Отладочная и итоговая версия программы. Утилита командной строки dumpbin. Перенос выполняемых файлов на другие компьютеры. Изменение разрядности программы.
16. Оптимизация машинного кода. Статическая сборка. Разделение программы на несколько cpp-файлов. Схема сборки программы в машинном коде на C++. .lib и .obj файлы.
17. Функция main и ее аргументы. Использование аргументов командной строки и разные способы запуска программы.

**Вопросы к экзамену (6 семестр)**

1. Технологии параллельного программирования. Закон Амдала. Пиковая и максимальная производительность.
2. Популярные технологии распараллеливания программ. Стандарт MPI (Message Passing Interface – Интерфейс передачи сообщений). Системы с общей памятью. Библиотеки для работы с нитями, OpenMP (Open Multi Processing – Открытая многопроцессорная обработка).
3. Параллельное программирование графических процессоров. OpenGL, CUDA, DirectX, C++ AMP, OpenCL, SYCL.
4. Среда выполнения MPI. MPI SDK (software development kit - набор инструментов разработки). Компиляция и запуск MPI программ. Запуск нескольких копий программы (mpiexec).
5. Общая структура MPI программ. Понятие коммуникатора. Идентификация отдельных процессов. Основные функции MPI (MPI_Init, MPI_Comm_size, MPI_Comm_rank, MPI_Finalize, MPI_Get_processor_name). Вспомогательные функции MPI (MPI_Initialized, MPI_Finalized).
6. Использование номеров процессов для разделения вычислений между процессами, случай некратности числа значений числу процессов (третий пример у Антонова).
7. Использование системного таймера в MPI. Определение имени компьютера, на котором запущен процесс. Функции MPI_Wtick, MPI_Wtime, MPI_Get_processor_name. Запуск нескольких MPI процессов на нескольких компьютерах (smpd).
8. Логика передачи и приема сообщений с блокировкой без блокировки. Функции MPI_Send, MPI_Recv. Смысл MPI_Datatype, MPI_Status и других параметров. Допустимость сообщения нулевой длины. Использование констант MPI_ANY_SOURCE и MI_ANY_TAG.
9. Соответствие между операцией приема и операцией отправки. Номера процесса. Код, выполняемый на всех процессах и лишь на некоторых. Применение схемы «мастер-рабочие» с использованием функций MPI_Send и MPI_Recv.
10. Порядок вызова отправки и получения для блокирующих операций. Неявная буферизация сообщений.
11. Выяснение количества принятых данных. Функция MPI_Get_count, значение MPI_UNDEFINED.
12. Пересылка сообщений с несуществующим процессом, значение MPI_PROC_NULL.
13. Буферизированная посылка данных. Функции MPI_Bsend, MPI_Buffer_attach. Определение размера отправляемых сообщений. Значение MPI_BSEND_OVERHEAD. Функция MPI_Pack_size. Функция MPI_Buffer_detach и ее использование.
14. Получение информации об атрибутах сообщения, функция MPI_Probe.
15. Резервируемый размер стека. Тестирование скорости передачи и задержки при передаче сообщений (латентность и пропускная способность).

16. Возможные схемы обмена сообщениями при организации параллельных вычислений с MPI. Варианты схемы «мастер-рабочие». Функции управляющего процесса. обмен между «соседними» процессами. Общая структура MPI программ.
17. Обмен по кольцевой топологии при помощи неблокирующих операций. Реализация цепочки, замкнутой в кольцо, при использовании блокирующих операций. Функции MPI_Irecv, MPI_Isend, тип MPI_Request.
18. Четыре режима отправки (стандартный, буферизированный, синхронный, по готовности), функции MPI_Ssend и MPI_Rsend. Функции MPI_Isend, MPI_Ibsend, MPI_Issend, MPI_Irsend. Функция MPI_Recv, приём сообщений для разных режимов отправки, рекомендации по использованию режимов.
19. Функции MPI_Test и MPI_Wait. Завершение неблокирующих операций. Сравнение функций MPI_Wait, MPI_Test, MPI_Waitall, MPI_Testall, MPI_Waitany, MPI_Testany, MPI_Waitsome, MPI_Testsome.
20. Отмена неблокирующих операций, функции MPI_Cancel, MPI_Test_cancelled. Описание функций MPI_Wait и MPI_Waitall из стандарта.
21. Параллельная обработка матриц с разбивкой на блоки (пример 13 у Антонова). Понятие ореола.
22. Одновременное выполнение приема и передачи. Функции MPI_Sendrecv, MPI_Sendrecv_replace. Посылка сообщения самому себе.
23. Синхронизация вычислений в MPI, функция MPI_Barrier.
24. Коллективные операции передачи данных в MPI. Функции MPI_Bcast, MPI_Ibcast.
25. Функции для коллективного сбора информации на одном процессе MPI_Gather, MPI_Igather, MPI_Gatherv, MPI_Igatherv.
26. Функции для коллективной рассылки разной информации процессам MPI_Scatter, MPI_Scatterv.
27. Функции для обобщённого коллективного сбора информации на всех процессах MPI_Allgather, MPI_Allgatherv, MPI_Alltoall, MPI_Alltoallv.
28. Коллективные операции редукции. Функции MPI_Reduce, MPI_Allreduce.
29. Производные типы в MPI Аналоги структур в MPI. Функции MPI_Type_struct, MPI_Get_address, MPI_Type_commit, MPI_Type_free.
30. Параллельное программирование для систем с общей памятью. Библиотеки Windows Thread API и PThread API. Технология OpenMP.
31. Параллельные программы в OpenMP. Общий вид директив OpenMP. Директивы #pragma omp parallel, #pragma omp critical. Поддержка OpenMP в Microsoft Visual Studio.
32. Использование стандартных библиотек OpenMP. Функции omp_get_thread_num, omp_get _num_threads, omp_get_wtime, omp_get_wtick.
33. Распараллеливание по данным для циклов. Директива pragma omp parallel for. Информационная зависимость по данным. Общие (разделяемые) и частные (принадлежащие нити) переменные. Опции shared, private. Опции threadprivate, copyin, copyprivate, default. Переменные в стеке и в динамической памяти.
34. Управление распределением итераций цикла между потоками Опция schedule, способ распределения, автоматический, статический, динамический, управляемый (auto, static, dynamic, guided). Размеры кусков.
35. Выбор способа распределения во время запуска, schedule(runtime). Переменная окружения OMP_SCHEDULE.
36. Управление порядком выполнения вычислений. Параметр ordered директивы for и директива ordered. Комбинирование директив parallel и for.
37. Синхронизация вычислений по окончании цикла, параметр nowait. Добавление условий для определения параллельных фрагментов, параметр if.
38. Гонка потоков. Взаимное исключение при работе с общими данными. Директивы #pragma omp critical, #pragma omp atomic. Именование критических секций.
39. Распараллеливание по схеме, аналогичной MPI. Директивы master, barrier, single.

40. Редукция в OpenMP.

**Таблица 9 – Примеры оценочных средств с ключами правильных ответов**

| № п/п | Тип задания | Формулировка задания | Правильный ответ | Время выполнения (в минутах) |
|---|---|---|---|---|
| ***Код и наименование проверяемой компетенции***<br>***УК-1. Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач.*** | | | | |
| 1. | Задание закрытого типа | Какие функции MPI можно вызывать до вызова процедуры MPI_Init? (Ответ считается верным, если отмечены все правильные варианты ответов.)<br>1) MPI_Get_version<br>2) MPI_Initialized<br>3) MPI_Finalized<br>4) MPI_Finalize | 1,2,3 | 1 |
| 2. | | Какие функции MPI можно вызывать после вызова процедуры MPl_Finalize? (Ответ считается верным, если отмечены все правильные варианты ответов.)<br>1) MPI_Init<br>2) MPI_Get_version<br>3) MPI_Initialized<br>4) MPI_Finalized | 2,3,4 | 1 |
| 3. | | Сколько аргументов имеет функция MPI_Send? | 6 | 1 |
| 4. | | Сколько аргументов имеет функция MPI_Recv? | 7 | 1 |
| 5. | | Какие функции MPI возвращают значение типа double? (Ответ считается верным, если отмечены все правильные варианты ответов.)<br>1) MPI_Wtime<br>2) MPl_Wtick<br>3) MPI_Comm_rank<br>4) MPI_Get_processor_name | 1,2 | 1 |
| 6. | Задание открытого типа | Дайте определение коммуникатора в MPI. | Коммуникатор в MPI - специально создаваемый служебный объект, объединяющий в своем составе группу процессов и ряд дополнительных параметров (контекст). Коммуникатор предоставляет собой отдельный контекст обмена процессов некоторой группы. | 5 |
| 7. | | Для чего предназначен тег сообщения в MPI? | По тегу процесс, принимающий сообщение может, например, различить два | 5 |

| № п/п | Тип задания | Формулировка задания | Правильный ответ | Время выполнения (в минутах) |
|---|---|---|---|---|
| | | | сообщения, пришедшие к нему от одного и того же процесса. | |
| 8. | | Что делает функция MPI_Comm_size? | Функция MPI_Comm_size возвращает число параллельных процессов в коммуникаторе. | 5 |
| 9. | | Что делает функция MPI_Comm_rank? | Функция MPI_Comm_rank возвращает номер (ранг) процессов в коммуникаторе. | 5 |
| 10. | | Что делает функция MPI_Get_processor_name? | Функция MPI_Get_processor_name возвращает в строке имя узла, на котором запущен вызвавший процесс. | 5 |
| ***Код и наименование проверяемой компетенции***<br>***ПК-1. Способность собирать, обрабатывать, интерпретировать данные и проводить под руководством научное исследование в области математических и (или) естественных наук.*** | | | | |
| 11. | Задание закрытого типа | Коммуникатор, который включает в себя все запущенные параллельно процессы MPI, обозначается:<br>1) MPI_COMM_WORLD<br>2) MPI_COMM_SELF<br>3) MPI_COMM_NULL<br>4) Не имеет специального обозначения | 1 | 1 |
| 12. | | Коммуникатор, который включает в себя только один текущий процесс MPI, обозначается:<br>1) MPI_COMM_WORLD<br>2) MPI_COMM_SELF<br>3) MPI_COMM_NULL<br>4) Не имеет специального обозначения | 2 | 1 |
| 13. | | Недопустимый коммуникатор в MPI обозначается:<br>1) MPI_COMM_WORLD<br>2) MPI_COMM_SELF<br>3) MPI_COMM_NULL<br>4) Не имеет специального обозначения | 3 | 1 |
| 14. | | В каких из функций MPI_Send или MPI_Recv можно использовать в четвёртом аргументе значение MPI_ANY_SOURCE?<br>1) MPI_Send и MPI_Recv<br>2) MPI_Send<br>3) MPI_Recv | 3 | 1 |

| № п/п | Тип задания | Формулировка задания | Правильный ответ | Время выполнения (в минутах) |
|---|---|---|---|---|
| | | 4) Ни в одной нельзя | | |
| 15. | | В каких из функций MPI_Send или MPI_Recv можно использовать в пятом аргументе значение MPI_ANY_TAG?<br>1) MPI_Send и MPI_Recv<br>2) MPI_Send<br>3) MPI_Recv<br>4) Ни в одной нельзя | 3 | 1 |
| 16. | Задание открытого типа | В чём разница в MPI между функциями обмена с блокировкой и функциями без блокировки (асинхронными).. | Функции обмена с блокировкой приостанавливают работу процесса до выполнения некоторого условия, а возврат из асинхронных процедур происходит немедленно после инициализации соответствующей коммуникационной операции. | 5 |
| 17. | | В чём суть гибкой методологии управления разработкой программного обеспечения? | При использовании гибкой методологии работа над программным обеспечением разбивается на небольшие по продолжительности этапы (от одной недели до одного месяца) и ставятся конкретные задачи по разработке в течении очередного этапа. По прошествии очередного этапа, совместно с заказчиком, делается оценка достигнутых результатов. | 5 |
| 18. | | Что делает функция QueryPerformanceCounter? | Функция QueryPerformanceCounter считывает счетчик производительности и возвращает общее количество тактов, произошедших с момента запуска Windows операционной системы, в том числе время, когда компьютер находился в спящем состоянии, например в режиме ожидания, гибернации или | 5 |

| № п/п | Тип задания | Формулировка задания | Правильный ответ | Время выполнения (в минутах) |
|---|---|---|---|---|
| | | | подключенного режима ожидания. | |
| 19. | | В чём заключается идея сортировки слиянием? | Идея сортировки методом слияния заключается в том, что из двух отсортированных массивов можно легко за один проход получить общий отсортированный массив. | 5 |
| 20. | | В чём заключается идея сортировки вставкой? | Идея сортировки методом вставки заключается в том, чтобы, рассматривая начальную часть массива как отсортированную, выполнять вставку следующего после отсортированной части элемента в нужное место в уже отсортированной части массива. В результате каждой такой вставки отсортированная часть массива удлиняется на 1 элемент. | 5 |
| ***Код и наименование проверяемой компетенции***<br>***ПК-2. Способность применять высокопроизводительные суперкомпьютерные технологии при решении задач в профессиональной деятельности.*** | | | | |
| 21. | Задание закрытого типа | Какие функции MPI можно вызывать до вызова процедуры MPI_Init? (Ответ считается верным, если отмечены все правильные варианты ответов.)<br>1) MPI_Get_version<br>2) MPI_Initialized<br>3) MPI_Finalized<br>4) MPI_Finalize | 1,2,3 | 1 |
| 22. | | Какие функции MPI можно вызывать после вызова процедуры MPl_Finalize? (Ответ считается верным, если отмечены все правильные варианты ответов.)<br>1) MPI_Init<br>2) MPI_Get_version<br>3) MPI_Initialized<br>4) MPI_Finalized | 2,3,4 | 1 |
| 23. | | Сколько аргументов имеет функция MPI_Send? | 6 | 1 |
| 24. | | Сколько аргументов имеет функция MPI_Recv? | 7 | 1 |
| 25. | | Какие функции MPI возвращают | 1,2 | 1 |

| № п/п | Тип задания | Формулировка задания | Правильный ответ | Время выполнения (в минутах) |
|---|---|---|---|---|
| | | значение типа double? (Ответ считается верным, если отмечены все правильные варианты ответов.)<br>1) MPI_Wtime<br>2) MPl_Wtick<br>3) MPI_Comm_rank<br>4) MPI_Get_processor_name | | |
| 26. | Задание открытого типа | Дайте определение коммуникатора в MPI. | Коммуникатор в MPI - специально создаваемый служебный объект, объединяющий в своем составе группу процессов и ряд дополнительных параметров (контекст). Коммуникатор предоставляет собой отдельный контекст обмена процессов некоторой группы. | 5 |
| 27. | | Для чего предназначен тег сообщения в MPI? | По тегу процесс, принимающий сообщение может, например, различить два сообщения, пришедшие к нему от одного и того же процесса. | 5 |
| 28. | | Что делает функция MPI_Comm_size? | Функция MPI_Comm_size возвращает число параллельных процессов в коммуникаторе. | 5 |
| 29. | | Что делает функция MPI_Comm_rank? | Функция MPI_Comm_rank возвращает номер (ранг) процессов в коммуникаторе. | 5 |
| 30. | | Что делает функция MPI_Get_processor_name? | Функция MPI_Get_processor_name возвращает в строке имя узла, на котором запущен вызвавший процесс. | 5 |
| ***Код и наименование проверяемой компетенции***<br>***ПК-4. Способен разрабатывать требования и проектировать программное обеспечение.*** | | | | |
| 31. | Задание закрытого типа | Коммуникатор, который включает в себя все запущенные параллельно процессы MPI, обозначается:<br>1) MPI_COMM_WORLD<br>2) MPI_COMM_SELF<br>3) MPI_COMM_NULL<br>4) Не имеет специального | 1 | 1 |

| № п/п | Тип задания | Формулировка задания | Правильный ответ | Время выполнения (в минутах) |
|---|---|---|---|---|
| | | обозначения | | |
| 32. | | Коммуникатор, который включает в себя только один текущий процесс MPI, обозначается:<br>1) MPI_COMM_WORLD<br>2) MPI_COMM_SELF<br>3) MPI_COMM_NULL<br>4) Не имеет специального обозначения | 2 | 1 |
| 33. | | Недопустимый коммуникатор в MPI обозначается:<br>1) MPI_COMM_WORLD<br>2) MPI_COMM_SELF<br>3) MPI_COMM_NULL<br>4) Не имеет специального обозначения | 3 | 1 |
| 34. | | В каких из функций MPI_Send или MPI_Recv можно использовать в четвёртом аргументе значение MPI_ANY_SOURCE?<br>1) MPI_Send и MPI_Recv<br>2) MPI_Send<br>3) MPI_Recv<br>4) Ни в одной нельзя | 3 | 1 |
| 35. | | В каких из функций MPI_Send или MPI_Recv можно использовать в пятом аргументе значение MPI_ANY_TAG?<br>1) MPI_Send и MPI_Recv<br>2) MPI_Send<br>3) MPI_Recv<br>4) Ни в одной нельзя | 3 | 1 |
| 36. | Задание открытого типа | В чём разница в MPI между функциями обмена с блокировкой и функциями без блокировки (асинхронными).. | Функции обмена с блокировкой приостанавливают работу процесса до выполнения некоторого условия, а возврат из асинхронных процедур происходит немедленно после инициализации соответствующей коммуникационной операции. | 5 |
| 37. | | В чём суть гибкой методологии управления разработкой программного обеспечения? | При использовании гибкой методологии работа над программным обеспечением разбивается на небольшие по продолжительности этапы (от одной недели до одного месяца) и | 5 |

| № п/п | Тип задания | Формулировка задания | Правильный ответ | Время выполнения (в минутах) |
|---|---|---|---|---|
| | | | ставятся конкретные задачи по разработке в течении очередного этапа. По прошествии очередного этапа, совместно с заказчиком, делается оценка достигнутых результатов. | |
| 38. | | Что делает функция QueryPerformanceCounter? | Функция QueryPerformanceCounter считывает счетчик производительности и возвращает общее количество тактов, произошедших с момента запуска Windows операционной системы, в том числе время, когда компьютер находился в спящем состоянии, например в режиме ожидания, гибернации или подключенного режима ожидания. | 5 |
| 39. | | В чём заключается идея сортировки слиянием? | Идея сортировки методом слияния заключается в том, что из двух отсортированных массивов можно легко за один проход получить общий отсортированный массив. | 5 |
| 40. | | В чём заключается идея сортировки вставкой? | Идея сортировки методом вставки заключается в том, чтобы, рассматривая начальную часть массива как отсортированную, выполнять вставку следующего после отсортированной части элемента в нужное место в уже отсортированной части массива. В результате каждой такой вставки отсортированная часть массива удлиняется на 1 элемент. | 5 |

Полный комплект оценочных материалов по дисциплине (модулю) (фонд оценочных

средств) хранится в электронном виде на кафедре, утверждающей рабочую программу дисциплины (модуля), и в Центре мониторинга и аудита качества обучения.

### 7.4. Методические материалы, определяющие процедуры оценивания результатов обучения по дисциплине (модулю)

**Таблица 10 – Технологическая карта рейтинговых баллов по дисциплине (модулю)**

| № п/п | Контролируемые мероприятия | Количество мероприятий / баллы | Максимальное количество баллов | Срок представления |
|---|---|---|---|---|
| **Основной блок** | | | | |
| ***1.*** | *Выполнение лабораторных работ* | 4 / 11 | 44 | Указан в Moodle |
| ***2.*** | *Выполнение контрольных работ* | 2 / 13 | 26 | В день проведения |
| ***3.*** | *Выполнение мини-проектов* | 2 /10 | 20 | В течение семестра |
| **Всего** | | | **90** | |
| **Блок бонусов** | | | | |
| **4.** | *Посещение всех занятий* | 6 | 6 | В расписании |
| **5.** | *Своевременное выполнение всех заданий* | 4 | 4 | Указан в Moodle |
| **Всего** | | | **10** | |
| **ИТОГО** | | | **100** | |

**Таблица 11 – Шкала перевода рейтинговых баллов в итоговую оценку за семестр по дисциплине (модулю)**

| Сумма баллов | Оценка по 4-балльной шкале | |
|---|---|---|
| 90–100 | 5 (отлично) | Зачтено |
| 85–89 | 4 (хорошо) | |
| 75–84 | | |
| 70–74 | | |
| 65–69 | 3 (удовлетворительно) | |
| 60–64 | | |
| Ниже 60 | 2 (неудовлетворительно) | Не зачтено |

При реализации дисциплины (модуля) в зависимости от уровня подготовленности обучающихся могут быть использованы иные формы, методы контроля и оценочные средства, исходя из конкретной ситуации.

## 8. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

### 8.1. Основная литература

1. Давыдов В.Г. Технологии программирования. С++: рек. УМО вузов РФ по образованию в области радиотехники, электроники, биомедицинской техники и автоматизации в качестве учеб. пособ. для студентов вузов, обучающихся по специальности 210100 «Управление и информатика в технических системах». - СПб.: БХВ-Петербург, 2005. - 672 с.+1 электрон. диск (CD-ROM). - ISBN 5-94157-605-6: 204-26: 204-26. (7 экз.)

2. Малявко, А. А. Параллельное программирование на основе технологий openmp, cuda, opencl, mpi : учебное пособие для вузов / А. А. Малявко. — 3-е изд., испр. и доп. — Москва : Издательство Юрайт, 2023. — 135 с. — (Высшее образование). — ISBN 978-5-534-14116-0. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://www.biblio-online.ru/bcode/514199 (Электронная библиотечная система издательства ЮРАЙТ)
3. Арыков, С. Б. Параллельное программирование над общей памятью. OpenMP: учебное пособие / С. Б. Арыков, М. А. Городничев, Г. А. Щукин. - Новосибирск : НГТУ, 2019. - 95 с. - ISBN 978-5-7782-3796-4. - Текст : электронный // ЭБС "Консультант студента" : [сайт]. - URL : https://www.studentlibrary.ru/book/ISBN9785778237964.html (ЭБС «Консультант студента»)
4. Основы многопоточного и параллельного программирования [Электронный ресурс]: учеб. пособие / Карепова Е.Д. - Красноярск: СФУ, 2016. URL: http://www.studentlibrary.ru/book/ISBN9785763833850.html (ЭБС «Консультант студента»)
5. Технология программирования: метод. указания к лабораторному практикуму. Ч. 2 [Электронный ресурс] / Т.И. Вишневская, Т.Н. Романова. - М.: Издательство МГТУ им. Н. Э. Баумана, 2010. URL: http://www.studentlibrary.ru/book/bauman_0553.html (ЭБС «Консультант студента»)
6. Лаврищева, Е. М. Программная инженерия и технологии программирования сложных систем : учебник для вузов / Е. М. Лаврищева. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2023. — 432 с. — (Высшее образование). — ISBN 978-5-534-07604-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://www.biblio-online.ru/bcode/513067 (Электронная библиотечная система издательства ЮРАЙТ)

### 8.2. Дополнительная литература

1. Антонов А.С. Параллельное программирование с использованием технологии MPI / Антонов А.С. - М.: Национальный Открытый Университет «ИНТУИТ», 2016. URL: http://www.studentlibrary.ru/book/intuit_240.html (ЭБС «Консультант студента»)
2. Гергель В.П. Теория и практика параллельных вычислений / Гергель В.П. - М.: Национальный Открытый Университет «ИНТУИТ», 2016. URL: http://www.studentlibrary.ru/book/ISBN9785947746457.html (ЭБС «Консультант студента»)
3. Левин М.П. Параллельное программирование с использованием OpenMP / Левин М.П. - М.: Национальный Открытый Университет «ИНТУИТ», 2016. URL: http://www.studentlibrary.ru/book/ISBN9785947748574.html (ЭБС «Консультант студента»)
4. Немнюгин С.А. Введение в программирование на Intel Cilk Plus / Немнюгин С.А. - М.: Национальный Открытый Университет «ИНТУИТ», 2016. URL: http://www.studentlibrary.ru/book/intuit_072.html (ЭБС «Консультант студента»)
5. Камаев В.А. Технологии программирования: доп. М-вом образования и науки РФ в качестве учебника для студентов вузов ... «Информатика и вычислительная техника». - изд. 2-е; перераб. и доп. - М.: Высш. шк., 2006. - 454 с. - ISBN 5-06-004870-5: 363-00: 363-00. (11 экз.)

### 8.3. Интернет-ресурсы, необходимые для освоения дисциплины (модуля)

1. Электронная библиотечная система издательства ЮРАЙТ раздел «Легендарные книги».
2. Электронный каталог «Научные журналы АГУ»: http://journal.asu.edu.ru/.
3. Научная электронная библиотека eLIBRARY.ru ООО «РУНЭБ» - крупнейший российский информационный портал: http://elibrary.ru
4. ИНТУИТ(национальный открытый университет ) http://www.intuit.ru/department/se/oip/

5. Электронно-библиотечная система (ЭБС) ООО «Политехресурс» «Консультант студента». www.studentlibrary.ru.

## 9. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

Для проведения лекционных занятий используется аудитория, оборудованная современной презентационной техникой (доска /интерактивная доска).

Для проведения практических занятий используется компьютерный класс, оснащенный персональными компьютерами класса PC с выходом в Интернет.

Рабочая программа дисциплины (модуля) при необходимости может быть адаптирована для обучения (в том числе с применением дистанционных образовательных технологий) лиц с ограниченными возможностями здоровья, инвалидов. Для этого требуется заявление обучающихся, являющихся лицами с ограниченными возможностями здоровья, инвалидами, или их законных представителей и рекомендации психолого-медико-педагогической комиссии. Для инвалидов содержание рабочей программы дисциплины (модуля) может определяться также в соответствии с индивидуальной программой реабилитации инвалида (при наличии).

**Приложение 1**

## ОСНОВНЫЕ ТРЕБОВАНИЯ К КУРСОВОЙ РАБОТЕ

**Курсовая работа должна содержать в себе фрагменты, представленные ниже.**

**Обоснование актуальности** – объяснение необходимости изучения данной темы в контексте общего процесса научного познания. **Обоснование актуальности темы** излагается в тексте ВВЕДЕНИЯ.

**Обзор научной литературы** – основание для постановки проблемы.

**Задачи обзора литературы:**

- провести общее и детальное знакомство с темой исследования;
- выявить и сформулировать проблему исследования;
- определить цели и задачи курсовой работы.

**Формулировка цели** курсовой работы должна тесно соотноситься с темой, как правило, полностью включая её. После формулировки общей цели работы указываются конкретные **задачи**, которые являются своеобразными ступеньками-этапами на пути достижения цели. Обычно они даются в форме перечисления: «изучить...», «описать...», «раскрыть...», «выявить...», «определить...», «исследовать...», «выяснить...», «проанализировать...» и т. д. Формулировки задач необходимо делать как можно тщательнее, поскольку описание их решения должно составить содержание глав работы. Задач в работе не должно быть много.

Далее в соответствии с логической схемой исследования исследователем формулируются **объект** и **предмет исследования.**

**Объект** – это процесс или явление, порождающее проблемную ситуацию и избранное для изучения.

**Предмет** – это то, что находится в границах объекта.

Объект и предмет исследования как категории исследовательского процесса соотносятся между собой как общее и частное. В объекте выделяется та его часть, которая служит предметом исследования. Именно на него и направлено основное внимание исследователя, именно предмет исследования определяет тему исследования, которая обозначается на титульном листе как её заглавие.

Заключительным этапом являются **выводы**, которые содержат всё то новое и существенное, что составляет научные и практические результаты проведённой исследовательской работы.

## СТРУКТУРА КУРСОВОЙ РАБОТЫ

Структурными элементами курсовой работы являются:

- титульный лист;
- оглавление;
- введение;
- основная часть;
- заключение;
- список использованных источников;
- приложения.

**Титульный лист** является первой страницей курсовой работы и оформляется в соответствии с установленным образцом.

**Оглавление** содержит все заголовки разделов курсовой работы с указанием страницы, с которых они начинаются.

Во **введении** обосновывается актуальность выбранной темы, формулируется проблема, которую студент должен решить в данной работе, определяются цели и взаимосвязанный комплекс задач исследования, предмет и объект, методы исследования. Рекомендуемый объём введения – 2-3 страницы.

**Основная часть** носит содержательный характер, в ней решаются поставленные задачи, описывается ход и результаты научно-аналитической, экспериментальной работы.

Основную часть следует делить на главы и параграфы. Содержание глав основной части должно точно соответствовать теме работы и полностью её раскрывать. Рекомендуемое количество глав – 2-3, рекомендуемое количество параграфов – 2-3. Между параграфами и между главами необходимы смысловые связки, чтобы текст курсовой работы был логично выстроен и не содержал разрывов в изложении материала.

Работа пишется грамотным, литературным языком. Материал излагается последовательно и логично, в соответствии с намеченным планом работы и задачами исследования. Определения и формулировки, сделанные автором, должны быть точными и ясными, как без излишней детализации, так и чрезмерной краткости. Старайтесь избегать повторов слов и словосочетаний, речевых штампов.

В тексте работы допустимы некоторые сокращения, но они требуют пояснения. При первом их использовании словосочетание пишется полностью, а скобках указывается принятое сокращение, которое и будет использоваться в дальнейшем. Например, «самостоятельная работа школьников» (СРШ), «комплекс дидактических игр» (КДИ) и т. п. Следует иметь в виду, что таких сокращений не должно быть много, иначе это затрудняет восприятие материала.

При написании работы не рекомендуется вести изложение от первого лица («Я считаю...», «По моему мнению...») или от множественного лица («Мы полагаем...», «Мы наблюдаем…»). Лучше выразить мысль в безличной форме («Полученные данные свидетельствуют о том…», «Можно утверждать, что...», «Итоги эксперимента дают основание для...» и т. д.).

Если в тексте используется прямая цитата, она должна быть заключена в кавычки и сопровождаться ссылкой на автора. Для этого в скобках сразу после цитаты указываются инициалы и фамилия автора работы, год её издания. Например, «Самостоятельная работа учащихся, включаемая в процесс обучения, – это такая работа, которая выполняется без непосредственного участия учителя, но по его заданию, в специально предоставленное для это время» (Б. П. Есипов, 1961). Если цитата приводится не с начала (или без окончания), то перед ней (во втором случае после неё) ставится многоточие.

Не стоит увлекаться прямым цитированием. Мысль автора может быть передана своими словами (без искажения смысла), тем не менее, в этом случае ссылка на его фамилию необходима.

В **заключении** последовательно излагаются теоретические и практические результаты и суждения, к которым пришёл студент в результате исследования. Они должны быть краткими, чёткими, дающими полное представление о содержании, значимости, обоснованности и эффективности работы. Пишутся они тезисно, по пунктам. Результаты (выводы) исследования должны соответствовать поставленным цели и задачам.

После заключения приводится **список использованных источников** в установленном порядке. Каждый включённый литературный источник должен иметь отражение в тексте курсовой работы. Если автор делает ссылку на какие-либо заимствованные факты или цитирует работы других авторов, то он должен указать, откуда взяты приведённые материалы. Нельзя включать в библиографический список те работы, на которые нет ссылок в тексте работы и которые фактически не были использованы, включаются те источники, которые использовались при написании работы. На все перечисленные в списке источники должны быть ссылки в основном тексте работы в виде номеров из списка, заключенных в квадратные скобки. Пример: [5], где 5 это номер по порядку в списке использованных источников.

В **приложения** следует относить вспомогательный материал, который при включении в основную часть работы загромождает текст. К вспомогательному материалу относятся первичные таблицы, промежуточные расчёты, таблицы вспомогательных цифровых данных, инструкции, методики, иллюстрации вспомогательного характера, неопубликованные ранее тексты. Если приложений больше десяти, их следует объединить по видам.

Курсовая работа предоставляется в печатном виде и на электронном носителе на кафедру. Приложить программы (все проекты с исходными кодами) с тестовыми примерами в электронном виде, копии использованных источников в электронном виде.

Работа выполняется в любом текстовом редакторе. Формат страницы – А4, кегль – 14, межстрочный интервал – 1,5. Выравнивание по ширине, отступ слева (красная строка) – 1,5. Текст следует размещать на одной стороне листа бумаги с соблюдением следующих размеров полей: левое – 30 мм, правое – 15 мм, верхнее – 20 мм, нижнее – 20 мм. В тексте установить автоматические переносы, исключая заголовки. Общий объём 20-25 страниц. Описание практической части — не менее 6 страниц, описание теоретической части — не менее 6 страниц. Практическая часть: реализация на языке C++ программы, соответствующей теме, последовательная и параллельная версии программы, сравнение результатов и времени выполнения.

При оформлении работы необходимо соблюдать равномерную плотность, контрастность и чёткость изображения по всей работе. Не должно быть помарок, перечёркивания, сокращения слов, за исключением общепринятых.

Страницы текста нумеруют арабскими цифрами снизу по центру. По всему тексту соблюдается сквозная нумерация. Номер титульного листа и листа с содержанием **не проставляется**, но включается в общую нумерацию курсовой работы. Все структурные элементы работы: введение, главы основной части, заключение, список используемых источников, приложения, – должны начинаться **с новой страницы.**

**Оформление глав и параграфов.** Каждая глава курсовой работы начинается с новой страницы. Установить интервал после названия главы – 12 пт. Если глава имеет только один параграф, то выделять его не следует. Заголовки глав печатаются прописными буквами, заголовки параграфов пишутся строчными буквами (первая буква заголовка параграфа заглавная), шрифт полужирный. Точки в конце заголовков **не ставятся**, заголовки не подчёркиваются. Переносы слов во всех заголовках **не допускаются**.

**Оформление маркированных и нумерованных списков.** Нумерованные и маркированные списки во всей работе должны быть единообразными. Для маркированных списков используются одни и те же маркеры, выбран единый стиль правил написания (пунктуация). Для нумерованных списков применяется одинаковая нумерация, выбран единый стиль правил написания (пунктуация). Во всех списках устанавливаются одинаковые отступы до маркера (номера), от маркера (номера) до текста и отступ слева для последующих строк.

**Оформление табличного материала.** Цифровой материал, сопоставление и выявление определённых закономерностей оформляют в виде таблиц. Все таблицы, если их несколько, нумеруются арабскими цифрами в правом верхнем углу, например: «Таблица 1», нумерация сквозная во всей работе. На следующей строке по центру располагается заголовок таблицы. Таблица выполняется на одной странице. Если таблица не умещается на одной странице, то она переносится на другие, при этом заголовок таблицы помещается на первой странице, а на следующих страницах следует повторить шапку таблицы. Текст в таблице и подписи к ней имеют размер шрифта 12 пт, межстрочный интервал в тексте таблицы – одинарный.

**Оформление иллюстраций.** Иллюстрации (рисунки, графики, диаграммы, эскизы, чертежи и т. д.) располагаются в работе непосредственно после текста, в котором они упоминаются впервые, или на следующей странице. Все иллюстрации должны быть пронумерованы (снизу по центру). Нумерация сквозная, через всю работу. Надпись под рисунком: «Рис. 1: Название рисунка», размер шрифта 12 пт. Если иллюстрация в работе единственная, то она не нумеруется. В тексте на иллюстрации делаются ссылки, содержащие порядковые номера, под которыми иллюстрации помещены в курсовой работе.

В работе могут быть использованы фотоиллюстрации, сделанные автором самостоятельно. Они могут быть представлены в качестве приложения к курсовой работе, так же как и цифровые, табличные и прочие иллюстрированные материалы.

**Оформление формул.** Формулы, на которые необходимо обратить внимание, выделяются из текста в отдельную строку, располагаются по центру. Пояснение значений символов и числовых коэффициентов приводится непосредственно под формулой в той же последовательности, в которой они даны в формуле. Если формулу необходимо пронумеровать, то номер выравнивается по правому краю, нумерация сквозная.

**Оформление списка использованных источников.** Все источники, приведённые в списке, располагаются в едином нумерованном порядке. Основное условие правильного составления списка использованных источников – единообразное оформление и соблюдение государственных требований, предъявляемых к печати научных публикаций (ГОСТ Р 7.0.5—2008, Затекстовая библиографическая ссылка)

**Оформление приложений.** Все приложения, если их несколько, нумеруются арабскими цифрами в правом верхнем углу, записываются прописными буквами, полужирным шрифтом, размер 12 пт, например: «**Приложение 1**». Далее на следующей строке по правому краю располагается заголовок приложения, строчными буквами, полужирным шрифтом, размер 12 пт. Если приложение не умещается на одной странице, то оно переносится на другие, на следующих страницах следует повторить шапку приложения (номер, заголовок). Текст во всех приложениях имеет размер шрифта 12 пт, межстрочный интервал – одинарный.

**Приложение 2**

**МИНОБРНАУКИ РОССИИ**
**АСТРАХАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**
**ИМЕНИ В.Н. ТАТИЩЕВА**

Факультет _____

Кафедра _____

Фамилия Имя Отчество

_____

***название курсовой работы***

Курсовая работа выполнена в рамках изучения дисциплины
«Технологии программирования»

Направление подготовки: 01.03.02 Прикладная математика и информатика

Научный руководитель: _____
*ученая степень, звание, должность, Ф.И.О.*

Астрахань – 20__