Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования «Новосибирский национальный исследовательский
государственный университет» (Новосибирский государственный университет, НГУ)

Факультет информационных технологий

СОГЛАСОВАНО
Декан ФИТ НГУ
_____ М.М. Лаврентьев
«03» июля 2019 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Парадигмы программирования

Направление подготовки: 09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА
Направленность (профиль): Программная инженерия и компьютерные науки
Форма обучения: очная
Год обучения: 4, семестр: 7

№	Вид деятельности	Семестр
		7
1	Лекции, час.	16
2	Практические занятия, час.	32
3	Лабораторные занятия, час.	
4	Занятий в контактной форме без учета промежуточной аттестации, час, из них	50
5	в электронной форме, час.	
6	из них аудиторных занятий, час.	48
7	из них в активной и интерактивной форме, час.	48
8	консультаций, час.	2
9	Самостоятельная работа, час.	92
10	в том числе на выполнение письменных работ, час	40
11	Форма аттестации (экзамен, зачет, дифференцированный зачет), час	Э 2
12	Всего зачетных единиц[1]	4

Новосибирск 2019

[1] С учетом выделенных часов на промежуточную аттестацию

Рабочая программа дисциплины составлена на основании федерального государственного образовательного стандарта (ФГОС) высшего образования - бакалавриат по направлению подготовки 09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА.

Федеральный государственный образовательный стандарт (ФГОС) высшего образования - бакалавриат по направлению подготовки 09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА введен в действие приказом Минобрнауки от 19.09.2017 № 929.

Место дисциплины в структуре учебного плана: Блок 1 Дисциплины (модули); часть, формируемая участниками образовательных отношений, дисциплина по выбору

Рабочая программа дисциплины утверждена решением Ученого совета факультета информационных технологий от 02.07.2019, протокол № 75.

Программу разработал:

доцент кафедры систем информатики ФИТ,
кандидат физико-математических наук  Л.В.Городняя

Заведующий кафедрой систем информатики ФИТ,
доктор физико-математических наук  М.М. Лаврентьев

Ответственный за образовательную программу:

доцент кафедры систем информатики ФИТ,
кандидат технических наук  А.А. Романенко

Аннотация к рабочей программе дисциплины
«Парадигмы программирования»

Дисциплина «Парадигмы программирования» реализуется в рамках образовательной программы высшего образования – программы бакалавриата 09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА, направленность (профиль): ПРОГРАММНАЯ ИНЖЕНЕРИЯ И КОМПЬЮТЕРНЫЕ НАУКИ по очной форме обучения на русском языке.

Место в образовательной программе: Дисциплина «Парадигмы программирования» развивает знания, умения и навыки, сформированные у обучающихся по результатам изучения следующих дисциплин: «Программирование», «Методы трансляции и компиляции», «Операционные системы», «Основы объектно-ориентированного программирования».

Дисциплина «Парадигмы программирования» является базовой для прохождения учебной/производственной практики и написания выпускной квалификационной работы.

Дисциплина «Парадигмы программирования» реализуется в 7 семестре в рамках части, формируемой участниками образовательных отношений, дисциплин (модулей) Блока 1 и является дисциплиной по выбору.

Дисциплина «Парадигмы программирования» направлена на формирование компетенций:

Способен применять естественнонаучные и общеинженерные знания и методы теоретического и экспериментального исследования в профессиональной деятельности (ПКС-2), в части следующих индикаторов достижения компетенции:

ПКС-2.1 Владеть: навыками разработки программ на языках высокого уровня.

ПКС-2.7 Уметь: проводить объектную декомпозицию информационной системы, вырабатывать и обосновывать архитектурные решения.

Перечень основных разделов дисциплины:

При освоении дисциплины студенты выполняют следующие виды учебной работы: лекции, практические занятия, консультации, самостоятельная работа. Содержание дисциплины охватывает круг вопросов, связанных с разнообразием моделей вычислений, включая модели параллелизма, а также с методами программирования параллельных процессов и проблемами организации высокопроизводительных вычислений.

Ядро курса включает в себя следующие темы:

- Классификация языков и парадигм программирования.
- Методы определения и спецификации языков программирования
- Семантика параллелизма.
- Учебные и модельные языки
- Функциональное программирование параллельных вычислений
- Верификация программ
- Функции высших порядков
- Взаимодействие процессов
- Многоуровневая память
- Языки параллельного программирования

Самостоятельная работа включает: подготовку к практическим занятиям по разделам дисциплины, подготовку презентаций докладов, написание рефератов, подготовку к экзамену.

Общий объем дисциплины – 4 зачетных единиц (144 часа).

Правила аттестации по дисциплине. Текущий контроль по дисциплине «Парадигмы программирования» осуществляется на практических занятиях и заключается в проведении учебных работ по основным разделам дисциплины и итоговой контрольной работы по всему пройденному материалу. Контрольная работа проводится в письменной форме и содержат 3 задачи. Максимальное количество баллов за решенную задачу – 5. Кроме того, выполняется подготовка презентации и защита доклада по основным разделам дисциплины, по результатам которых выставляется оценка «зачтено» или «не зачтено». Оценка «зачтено» по результатам защиты докладов является одним из условий успешного прохождения промежуточной аттестации.

Оценка "зачтено" за освоение дисциплины "Парадигмы программирования" выставляется при наличии следующих условий:

1) презентации и доклады на темы, соответствующие разделам дисциплины в каждом семестре, выполнены и защищены в полном соответствии с предъявляемыми требованиями (оценка "зачтено").

Промежуточная аттестация по дисциплине «Парадигмы программирования» проводится по завершению семестра. Промежуточная аттестация по дисциплине проводится в два этапа:

1) Оценочное портфолио по результатам работы в семестре, которое включает: 1 домашнее задание из 3 задач, презентации к докладу и 1 контрольная работа.
2) Устный экзамен. В каждом экзаменационном билете два вопроса. Во время ответа обучающемуся могут быть заданы дополнительные вопросы, в зависимости от вопросов, образующих билет.

Результаты промежуточной аттестации по дисциплине оцениваются по шкале «неудовлетворительно», «удовлетворительно», «хорошо», «отлично». Оценки «отлично», «хорошо», «удовлетворительно» означают успешное прохождение промежуточной аттестации.

Оценка «отлично» соответствует продвинутому уровню сформированности компетенции.

Оценка «хорошо» соответствует базовому уровню сформированности компетенции.

Оценка «удовлетворительно» соответствует пороговому уровню сформированности компетенции.

Учебно-методическое обеспечение дисциплины.

Учебно-методический комплекс по дисциплине «Парадигмы программирования»:

1. Л.В. Городняя. Парадигмы программирования: анализ и сравнение. Из-во СО РАН РФФИ 17-11-00042. 2017 - 216 с. http://www.rfbr.ru/rffi/ru/books/o_2043045
2. Л.В. Городняя. Парадигма программирования (курс лекций) Новосиб. гос. Ун-т. – Новосибирск : РИЦ НГУ, 2015. – 206 с. https://www.iis.nsk.su/files/book/file/FIT-Gor-PP3.pdf
3. Городняя, Л.В. Основы функционального программирования: курс / Л.В. Городняя ; Национальный Открытый Университет "ИНТУИТ". – Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2004. – 217 с. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=233773
4. Иртегов Д.В. Многопоточное программирование с использованием POSIX Threads - http://www.intuit.ru/department/se/posixthreads/

1. Внешние требования к дисциплине

Таблица 1.1

Компетенция ПКС-2 Способен применять естественнонаучные и общеинженерные знания, методы математического анализа и моделирования, теоретического и экспериментального исследования в профессиональной деятельности, *в части следующих индикаторов достижения компетенции:*
ПКС-2.1 Владеть: навыками разработки программ на языках высокого уровня.
ПКС-2.7 Уметь: проводить объектную декомпозицию информационной системы, вырабатывать и обосновывать архитектурные решения.

2. Требования к результатам освоения дисциплины

Таблица 2.1

Результаты изучения дисциплины по уровням освоения (иметь представление, знать, уметь, владеть)	**Формы организации занятий**		
	Лекции	**Практики / семинары /лабораторные работы**	**Самостоятельная работа**
ПКС-2.1 Владеть: навыками разработки программ на языках высокого уровня.			
1. Знать базовые парадигмы программирования.	+	+	+
2. Иметь навыки выбирать парадигму программирования по форме постановки задачи.	+	+	+
ПКС-2.7 Уметь: проводить объектную декомпозицию информационной системы, вырабатывать и обосновывать архитектурные решения.			
3. Знать типовые архитектурные решения, предоставляемые системами программирования.	+	+	+
4. Уметь обосновывать выбор системы программирования для решения конкретной задачи.		+	+

3. Содержание и структура учебной дисциплины

Таблица 3.1

Темы лекций	**Активные формы, час.**	**Часы**	**Ссылки на результаты обучения**
Семестр: 7			
3) О классификации языков и моделей параллельного программирования. Начинается изложение с необходимых пояснений к терминам, т.к. исторически сложилась весьма пёстрая речевая практика, дающая пищу техническим разночтениям общеизвестных понятий (поток, процесс, действие, операция и т.д.). Рассматриваются такие понятия как условие готовности и срабатывание, завершение и оттеснение, время и длительность,	1	1	1

пропускная способность и производительность, события и исключения, одновременность, ускорение вычислений. Понятия «параллелизм» и «параллельные алгоритмы» рассматриваются с точки зрения измеримости производительности систем. Проводится предварительный инструктаж по практике на C# и F# с библиотекой .Net. (установка интерпретатора и компилятора, прогон программ, диагностика ошибок) 4) Методы определения и спецификации языков программирования Рассматривается Венская методика определения языков программирования и техника символьной обработки разных структур данных, включая списки и множества, используемые при реализации программ и организации вычислений (очереди, протоколы и др.). Символьные вычисления рассматриваются как важный этап разработки эффективных и надежных программ и удобных языков программирования.			
5) Неимперативные вычисления. Небольшой обзор парадигм программирования, показывающий разнообразие подходов к программированию вообще и к параллельному программированию в частности. Особое внимание уделено неимперативным вычислениям, преимущественно функциональному программированию, показавшему свои преимущества в параллельном программировании. Парадигмы программирования анализируются в русле функционального подхода к определению семантики языков программирования. 6) .Преобразования программ и процессов. Краткий обзор средств и методов представления программ и процессов в форме определения языков и изображения эквивалентных им сетевых граф-схем с целью демонстрации возможностей декомпозиции параллельных программ и реорганизации процессов их выполнения, включая распараллеливание.	1	1	1
7) Семантика параллелизма. Абстрактные сети Ознакомление с аппаратом сетей, приспособленном к достаточно тонкой детализации семантики параллелизма. Изучается алгебра процессов, иерархические сети, уровни определений и конкретизация понятий. Рассматривается проблема синхронизации независимых сетей. Представление сетевых и распределенных систем рассматриваются как обоснование к выбору единого сетевого представления семантики программ, процессов, языков и систем программирования. 8) Параллельное программирование. Временные отношения Изучаются простейшие средства представления временных отношений между процессами. Рассматриваются процессы уровня ОС, допускающие явное и неявное задание времени и приоритетов. Время	1	1	1, 3

как ресурс. Распределение времени. В качестве примеров используются автоматы Т.Хоара, при программировании которых используются потоки, счетчики, барьеры. Отмечается подобие средств управления очередностью процессов и управления памятью.			
9) Учебные и модельные языки Рассматриваются примеры языков, конструируемых специально в целях изучения отдельных идей, стилей, техник программирования. 10) Функциональное программирование параллельных вычислений Изучаются особенности функционального программирования, позволяющие рассматривать его как базовую парадигмы параллельного программирования. 11) Верификация программ Ознакомление с результатами исследований в области верификации и моделирования программ, способными дать перспективу надежности параллельного программирования через разработку верифицирующих компиляторов. 12) Полный жизненный цикл программ (ЖЦП) Рассматривается схема ЖЦП и его конкретизация на случай разработки параллельных программ.	1	1	1, 2, 3
13) Функции высших порядков Изучается методика программирования с использованием функций высших порядков в качестве конструкторов программ. 14) Процессы в сетях Объектно-ориентированный подход к программированию сделал популярным представление программ в виде компонент, взаимосвязанных с помощью событий и сообщений. Рассматривается техника декомпозиции программ на такие компоненты и доопределения программ обработчиками исключений. Используются в качестве примеров задачи на клетчатой доске (взаимодействие процессов). Управление памятью рассматривается как доминирующая линия в определении семантики программ.	1	1	1, 3
15) Взаимодействие процессов Изучаются альтернативные подходы к определению условий готовности компонент к срабатыванию. Пространства имен (барьеры) как инструмент синхронизации. Завершение или готовность результата. Бесконечные и пошаговые процессы. Совмещение и слияние процессов. Управление памятью и контроль типов данных при разработке программ и компиляторов для языков программирования. 16) Макрообработка данных Рассматривается макротехника на разных уровнях представления данных, используемая с целью повышения точности их обработки, зашкаливающей за типичные границы применимости более надежных средств.	1	1	1, 2

17) Управление процессами Рассматриваются проблемы организации асинхронных процессов, возникающие при разработке распределенных систем. Анализ «живучести». Ленивые и энергичные вычисления. Синхросети и мониторы. Иллюстративные примеры (задачи о философах, читателях-писателях и т.п.) Представление сетевых и распределенных систем рассматриваются как основа для разработки сложных информационных систем и языков программирования.			
18) Векторная обработка Изучаются модели параллелизма для матричных вычислений. Идеальный параллелизм. Генерация параллельных итераций. Локальные обмены данными. Неограниченное число процессоров над одноуровневой общей памятью. Оценка ускорения вычислений и производительности систем в рамках программ над системами типов, допускающими статический контроль. (Примеры программ: сортировка, умножение матриц, факториал.) 19) Распараллеливание Рассматривается зависимость ускорения вычислений от числа процессоров и объема общей и распределенной памяти. Циклы и рекурсия. Однородное пространство процессоров, общая память, быстрые обмены, соседство. Оценка производительности систем для высокопроизводительных вычислений. 20) Многоуровневая память Изучается влияние дисциплины работы с памятью на характеристики параллельных процессов. Защищенная и размазанная память. Решения, принятые в разных языках программирования, по работе с многоуровневой и разнородной памятью (доступ, побочный эффект, дубли и копии). Обработка транзакций становится одной из типовых семантик работы с памятью в языках программирования. 21) Типы управления Рассматриваются средства и методы типизации управления процессами, удобными для подготовки программ, ориентированных на исполнение с помощью Open MP или MPI. Идеи языков программирования по сетевому представлению типов управления и дисциплины работы с памятью. Компонентно-ориентированная разработка ПО может следовать единому сетевому определению семантики языков программирования и процесса разработки программ	1	1	1, 2
22) Модели параллелизма в языках программирования Рассматриваются модели параллелизма, встроенные в наиболее известные языки программирования. Рассмотрены языки APL, Algol-68, Setl, Ada и др. Кроме того, рассмотрены guarded-command Э.Дейкстры и модели взаимодействующих процессов Т.Хоара и Р.Милнера, дающие теоретическую основу языковым и системным исследованиям. Имеется и небольшой экскурс в отечественные разработки в области языков параллельного программирования (БАРС, Поляр, Норма,	4	4	1, 3

mpC и др.). 23) Языки параллельного программирования Параллельное программирование на языка F# с библиотеками .Net Изучаются средства организации параллельных процессов средствами языка функционального программирования F# с библиотеками .Net. На этом языке рассматривается возможность реорганизация программ, что поддержано в языке специальными средствами типа Quotation – доступ к внутреннему представлению программ в виде структуры данных. Анализируется вклад типизации данных в построение эффективных программ. Использование модифицируемой и защищенной памяти. Прикладные аспекты работы с информацией, отражающей специфику области приложения программ (measure)			
24) Параллельное программирование на языке C#. C# дает возможность встраивать функциональные построения в контекст привычных императивных программ Рассматривается стыковка программ на новых языках с производственными системами, разрабатываемыми на базе библиотек .Net. Возможность оперирования деревом разбора программы и ее исполнимым кодом. Вопросы защиты программ и данных. Управление дисциплиной доступа к памяти и представление запросов к базам данных Параллельное программирование на языке Haskell. Рассматривается строго функциональный подход к спецификации параллельных программ и типов данных в языке с предпочтением так называемой «ленивой» схемы вычислений. Сопоставление достоинств и недостатков ленивых и энергичных методов вычислений. Концепция «монад» в строго функциональном языке программирования. Особенности определения семантики языковых конструкций. Методика обеспечения удобочитаемости программ и их отладки. Мемоизация как практичный инструмент снижения сложности вычислений. Параллельное программирование на языке Python. Язык Python зарекомендовал себя как удобное средство разработки распределенных систем и сетевого программирования. Рассматриваются особенности представления программ на языке Python и реализованные в нем решения по организации процессов в сетях и доступа к базам данных. История языка и особенности быстрой разработки программ Параллельное программирование на языке Sisal. Авторы строго типизированного функционального языка параллельного программирования Sisal создали интересный прецедент по расширению и уточнению системы понятий программирования для нужд представления и реализации масштабируемых параллельных вычислений. Программа в этом языке строится из участков с однократными присваиваниями,	5	5	1, 3

удобными для оптимизирующих преобразований, включая распараллеливание. В структуре цикла выделены позиции для формирования пространства параллельных итераций, фильтрации или сборки параллельно полученных результатов и обработки потоков данных при развитой системе работы с векторами, включая методику распространения скалярных операций на структуры данных. Рассматриваются примеры программ на языке Sisal. 25) Разработка параллельных программ и распределенных систем Проблемы разработки, отладки, тестирования и верификации параллельных программ по сложности весьма превосходят обычное программирование. Именно здесь сосредоточена исследовательская активность современного языкотворчества и системного программирования, поиск средств и методов интеграции удачного опыта параллельного программирования и успешного обучения методам параллельных вычислений. Акцент на анализ подходящих решений в области разработки переносимых программ, средств визуализации процессов, типизации управления и дисциплины доступа к памяти.			
Итого	**16**	**16**	

Таблица 3.2

Темы практических занятий *(лабораторных)*	**Активные формы, час.**	**Часы**	**Ссылки на результаты обучения**	**Учебная деятельность**
Семестр: 7				
Тема 1. Программирование макета, демонстрирующего проблемы синхронизации процессов (дедлоки и их исключение) на примере простой задачи управления автоматами.	10	10	1,2,3,4	Разбор теоретического материала, представленного на лекции, решение задач
Тема 2. Разработка макетных образцов отдельных фрагментов языков и моделей вычислений и параллельного программирования.	4	4	1,2,3,4	Разбор теоретического материала, представленного на лекции, решение задач
Тема 3. Экспериментальные макеты тренажеров для обучения программированию, включая параллельное.	18	18	1,2,3,4	Разбор теоретического материала, представленного на лекции, решение задач
Итого	**32**	**32**		

4. Самостоятельная работа студентов

Таблица 4.1

№	**Виды самостоятельной работы**	**Ссылки на результаты обучения**	**Часы на выполнение**	**Часы на консультации**

Семестр: 7				
1	Подготовка к практическим занятиям.	1,2,3,4	10	
	Обучающиеся изучают введение в язык Clisp и особенности работы с системой программирования, на базе которой предстоит выполнять практические задания			
2	Выполнение домашних заданий в рамках портфолио	2,3,4	40	
	Обучающиеся решают практические задачи, входящие в портфолио. Он проводит программную реализацию выбранных алгоритмов по решению выбранной задачи. Методические рекомендации по выполнению домашнего задания представлены в приложении к рабочей программе дисциплины.			
3	Подготовка презентации доклада.	1,2,3,4	8	
	Обучающийся формулирует и конкретизирует тему из списка, указанного в рабочей программе дисциплины, представляет доклад и презентацию для обсуждения и защиты на практическом занятии. Методические рекомендации по подготовке презентаций представлены в приложении к рабочей программе дисциплины.			
6	Подготовка реферата.	1,4	10	
	Обучающийся готовит реферат по теме из списка, указанного в рабочей программе дисциплины. Методические рекомендации по подготовке реферата представлены в приложении к рабочей программе дисциплины.			
7	Подготовка к экзамену	1,3	24	2
	Подготовка к экзамену по вопросам, представленным в фонде оценочных средств, являющихся приложением к рабочей программе дисциплины.			
	Итого		**92**	**2**

5. Образовательные технологии

В ходе реализации учебного процесса по дисциплине проводятся лекционные и семинарские занятия. Темы, рассматриваемые на лекциях и изучаемые самостоятельно, закрепляются на семинарах, по вопросам, вызывающим затруднения, проводятся консультации.

В ходе реализации учебного процесса по дисциплине применяются такие формы проведения практических занятий, как дискуссии, обсуждение и защита результатов работы, а также применяются следующие интерактивные формы организации учебных занятий (таблица 5.1).

Таблица 5.1

1	Лекция в форме дискуссии	ПКС-2.1, ПКС-2.7
Формируемые умения: уметь применять различные парадигмы программирования на разных фазах жизненного цикла программ и при решении задач организации высокопроизводительных вычислений. Совместная деятельность студентов в группе под руководством лидера, направленная на решение общей задачи путем творческого сложения результатов индивидуальной работы членов команды с делением полномочий и ответственности.		
Краткое описание применения: Обсуждение, в контексте изученной теории, специфики различных парадигм программирования на разных фазах жизненного цикла программ		
2	Портфолио	ПКС-2.1, ПКС-2.7

Формируемые умения: Уметь применять различные парадигмы программирования на разных фазах жизненного цикла программ и при решении задач организации высокопроизводительных вычислений
Краткое описание применения: бакалавры ведут портфолио (оценки за задания), которое является основой для проведения аттестации по дисциплине

Для организации и контроля самостоятельной работы студентов, а также проведения консультаций применяются информационно-коммуникационные технологии

Таблица 5.2

Информирование	lidvas@gmail.com
Консультирование	lidvas@gmail.com
Контроль	lidvas@gmail.com
Размещение учебных материалов	lidvas@gmail.com

6. Правила аттестации студентов по учебной дисциплине

По результатам освоения дисциплины «Парадигмы программирования» выставляется оценка «неудовлетворительно», «удовлетворительно», «хорошо», «отлично». Оценки «отлично», «хорошо», «удовлетворительно» означают успешное прохождение промежуточной аттестации

Текущая аттестация по дисциплине «Парадигмы программирования» осуществляется на практических занятиях и заключается в презентации и защите докладов по каждой теме практических занятий. В ходе обучения каждый студент должен подготовить презентации докладов по каждому разделу самостоятельной работы и публично выступить с ними, защищая полученные результаты в ходе обсуждения и дискуссии. По результатам текущей аттестации выставляется оценка «зачтено» или «не зачтено». Оценка «зачтено» по результатам защиты докладов является одним из условий успешного прохождения промежуточной аттестации.

Для получения оценки «зачтено» презентация и доклад на каждую тему, соответствующую разделам дисциплины в каждом семестре, должна быть выполнена и защищена в полном соответствии с предъявляемыми требованиями.

Промежуточная аттестация (итоговая по дисциплине) проводится по завершению каждого периода ее освоения (семестра) в виде защиты индивидуального проекта в формате портфолио, в состав которого включаются все работы, выполненные студентом в ходе изучения дисциплины. Завершает портфолио итоговая рефлексивная работа, направленная на переосмысление и оценку содержания дисциплины «Парадигмы программирования» и реализованной в его рамках учебной деятельности.

По результатам освоения дисциплины «Парадигмы программирования» результаты промежуточной аттестации определяются оценками «отлично», «хорошо», «удовлетворительно», «неудовлетворительно». Оценки «отлично», «хорошо», «удовлетворительно» означают успешное прохождение промежуточной аттестации.

Таблица 6.1

Коды компетенций ФГОС	**Результаты обучения**	**Формы аттестации**	
		1 этап - портфолио	**2 этап - экзамен**

ПКС.2	**ПКС-2.1** Владеть: навыками разработки программ на языках высокого уровня.	+	+
	ПКС-2.7 Уметь: проводить объектную декомпозицию информационной системы, вырабатывать и обосновывать архитектурные решения.	+	+

Требования к структуре и содержанию портфолио, оценочные средства, а также критерии оценки сформированности компетенций и освоения дисциплины в целом, представлены в Фонде оценочных средств, являющемся приложением 1 к настоящей рабочей программе дисциплины.

7. Литература

1. Страуструп, Б. Язык программирования C++ для профессионалов / Б. Страуструп. - Москва : Интернет-Университет Информационных Технологий, 2006. - 568 с. ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=234816

Интернет-ресурсы

Таблица 7.1

№ п/п	Наименование Интернет-ресурса	Краткое описание
1	http://www.hpcu.ru/ -	Интернет-Университет Суперкомпьютерных Технологий
2	http://www.cons.org/cmucl/	Сайт с материалами по особо эффективной реализации Lisp-а CMUCL

8. Учебно-методическое и программное обеспечение дисциплины

8.1. Учебно-методическое обеспечение

Учебно-методический комплекс по дисциплине «Парадигмы программирования»:

1. Л.В. Городняя. Парадигмы программирования: анализ и сравнение. Из-во СО РАН РФФИ 17-11-00042. 2017 - 216 с. http://www.rfbr.ru/rffi/ru/books/o_2043045
2. Л.В. Городняя. Парадигма программирования (курс лекций) Новосиб. гос. Ун-т. – Новосибирск : РИЦ НГУ, 2015. – 206 с. https://www.iis.nsk.su/files/book/file/FIT-Gor-PP3.pdf
3. Городняя, Л.В. Основы функционального программирования: курс / Л.В. Городняя ; Национальный Открытый Университет "ИНТУИТ". – Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2004. – 217 с. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=233773
4. Иртегов Д.В. Многопоточное программирование с использованием POSIX Threads - http://www.intuit.ru/department/se/posixthreads/

8.2. Программное обеспечение

Для обеспечения реализации дисциплины используется стандартный комплект программного обеспечения (ПО), включающий регулярно обновляемое лицензионное ПО Windows и MS Office.

Перечень специализированного программного обеспечения для изучения дисциплины представлен в таблице 8.1.

Специализированное программное обеспечение

Таблица 8.1

№	Наименование ПО	Назначение
1	Microsoft Visual Studio Professional 2019	Среда разработки приложений

9. Профессиональные базы данных и информационные справочные системы

1. Полнотекстовые журналы Springer Journals за 1997-2015 г., электронные книги (2005-2016 гг.), коллекция научных биомедицинских и биологических протоколов SpringerProtocols, коллекция научных материалов в области физических наук и инжиниринга SpringerMaterials, реферативная БД по чистой и прикладной математике zbMATH.
2. Электронная библиотека диссертаций Российской государственной библиотеки (ЭБД РГБ)
3. Электронные ресурсы Web of Science Core Collection (Thomson Reuters Scientific LLC.), Journal Citation Reports + ESI
4. БД Scopus (Elsevier)
5. Лицензионные материалы на сайте eLibrary.ru

10. Материально-техническое обеспечение

Таблица 10.1

№	Наименование	Назначение
1	Презентационное оборудование (мультимедиа-проектор, экран, компьютер для управления)	Для проведения лекционных занятий
2	Компьютерный класс (с выходом в Internet)	Для проведения практических занятий и организации самостоятельной работы

Материально-техническое обеспечение образовательного процесса по дисциплине для обучающихся из числа лиц с ограниченными возможностями здоровья осуществляется согласно «Порядку организации и осуществления образовательной деятельности по образовательным программам для инвалидов и лиц с ограниченными возможностями здоровья в Новосибирском государственном университете».

Лист актуализации рабочей программы дисциплины
«Парадигмы программирования»

№	Характеристика внесенных изменений (с указанием пунктов документа)	Дата и № протокола Ученого совета ФИТ	Подпись ответственного
1.	Актуализирована на 2020-2021 уч.год	22.07.2020 №77	

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования «Новосибирский национальный исследовательский
государственный университет» (Новосибирский государственный университет, НГУ)

Факультет информационных технологий

СОГЛАСОВАНО
Декан ФИТ НГУ
М.М. Лаврентьев
«03» июля 2019 г.

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ
по дисциплине Парадигмы программирования

Направление подготовки: 09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

Направленность (профиль): Программная инженерия и компьютерные науки
Квалификация: бакалавр

Форма обучения: очная Год обучения: 4, семестр 7

Форма аттестации	Семестр
Экзамен	7

Новосибирск 2019

Фонд оценочных средств промежуточной аттестации по дисциплине является **Приложением 1** к рабочей программе дисциплины «Парадигмы программирования», реализуемой в рамках образовательной программы высшего образования – программы бакалавриата 09.03.01 Информатика и вычислительная техника, направленность (профиль): Программная инженерия и компьютерные науки.

Фонд оценочных средств промежуточной аттестации по дисциплине утвержден решением ученого совета факультета информационных технологий, протокол № 75 от 02.07.2019.

Разработчики:
доцент кафедры систем информатики ФИТ,
кандидат физико-математических наук Л.В.Городняя

Заведующий кафедрой систем информатики ФИТ,
доктор физико-математических наук М.М. Лаврентьев

Ответственный за образовательную программу:
доцент кафедры систем информатики ФИТ,
кандидат технических наук А.А. Романенко

1. Содержание и порядок проведения промежуточной аттестации по дисциплине

1.1. Общая характеристика содержания промежуточной аттестации

Промежуточная аттестация по дисциплине «Парадигмы программирования» проводится по завершению периода освоения образовательной программы (семестра) для оценки сформированности компетенций в части следующих индикаторов достижения компетенции (таблица П1.1).

Таблица П1.1

Код	Компетенции, формируемые в рамках дисциплины «Парадигмы программирования»	Семестр 7	
		Портфолио	Экзамен
	ПКС-2 Способен разрабатывать компоненты системных программных продуктов		
ПКС-2.1	Владеть: навыками разработки программ на языках высокого уровня	+	+
ПКС-2.7	Уметь: проводить объектную декомпозицию информационной системы, вырабатывать и обосновывать архитектурное решение	+	+

Тематика вопросов экзамена соответствует избранным разделам (темам) дисциплины «Парадигмы программирования»:

О классификации языков и моделей параллельного программирования.
Термины и специальные понятия
Методы определения и спецификации языков программирования
Неимперативные вычисления.
Преобразования программ и процессов.
Семантика параллелизма.
Абстрактные сети
Параллельное программирование.
Временные отношения
Учебные и модельные языки
Функциональное программирование параллельных вычислений
Верификация программ
Полный жизненный цикл программ (ЖЦП)
Функции высших порядков
Макрообработка данных
Процессы в сетях
Взаимодействие процессов
Управление процессами
Векторная обработка
Распараллеливание
Многоуровневая память
Типы управления
Модели параллелизма в языках программирования
Языки параллельного программирования
Разработка параллельных программ и распределенных систем

1.2. Порядок проведения промежуточной аттестации по дисциплине

Промежуточная аттестация проводится в форме экзамена и включает 2 этапа: портфолио и экзамен. Необходимым условием для прохождения промежуточной аттестации является оценка «зачтено» по результатам выполненного портфолио. Для оценивания портфолио студенту необходимо сдать все работы, входящие в структуру портфолио.

Экзамен проводится в аудитории, студентам разрешено пользоваться бумагой для записей и авторучкой. Справочной, учебной и другой литературой пользоваться не разрешается. Использование электронных устройств (телефоны, любые виды компьютеров, т.д.) запрещено.

2. Требования к структуре и содержанию фонда оценочных средств промежуточной аттестации по дисциплине

Перечень оценочных средств, применяемых на каждом этапе проведения промежуточной аттестации по дисциплине, представлен в таблице П1.3.

Таблица П1.3

№ п/п	Наименование оценочного средства	Краткая характеристика оценочного средства	Представление оценочного средства в фонде
Семестр 7 Этап 1 - Портфолио			
1	Портфолио	Целевая подборка работ студента, раскрывающая его индивидуальные образовательные достижения в одной или нескольких учебных дисциплинах.	Требования к структуре и содержанию портфолио
Этап 2 – Экзамен			
2	Билет для экзамена	Комплекс вопросов	Список теоретических вопросов

2.1. Требования к структуре и содержанию оценочных средств аттестации в семестре

Программой дисциплины предусмотрены следующие виды контроля: текущий контроль успеваемости в 7 семестре в форме портфолио (оценки за задания, контрольную работу), промежуточная аттестация в 7 семестре в форме экзамена.

Портфолио включает 3 задания и контрольную работу, состоящую из 1 задачи.
Студентам предлагается выполнить 3 практических задания. Выполненные задания сдаются преподавателю на занятии.

Оценка ответа обучающегося по дисциплине «Парадигмы программирования» является положительной («удовлетворительно» и выше) только в случае положительных оценок по всем вопросам и задачам.

Оценка за портфолио по дисциплине «Парадигмы программирования» является положительной («удовлетворительно» и выше) только в случае положительных оценок по всем вопросам и задачам.

По результатам освоения дисциплины «Парадигмы программирования» выставляется оценка «неудовлетворительно», «удовлетворительно», «хорошо», «отлично». Оценки «отлично», «хорошо», «удовлетворительно» означают успешное прохождение промежуточной аттестации

По результатам освоения дисциплины «Парадигмы программирования» выставляется оценка «неудовлетворительно», «удовлетворительно», «хорошо», «отлично». Оценки «отлично», «хорошо», «удовлетворительно» означают успешное прохождение промежуточной аттестации

2.1.1. Примерные задания по дисциплине
Список задач

1) Написать рекурсивную функцию построения множества слов, встречающихся в двух заданных предложениях.
2) Написать рекурсивную функцию построения множества слов, встречающихся в каждом из двух заданных предложений.
3) Написать рекурсивную функцию построения множества слов, встречающихся в первом, но не во втором из двух заданных предложений.
4) Промоделировать на списках предикат для выяснения принадлежности атома множеству.
5) Промоделировать на списках предикат для выяснения принадлежит ли произвольный элемент, возможно не атом, множеству.
6) Промоделировать на списках объединение системы множеств.
7) Промоделировать на списках пересечение системы множеств.
8) Промоделировать на списках дополнение одного множества до другого.
9) Написать функцию, печатающую значения полей записи в указанном порядке.
10) Написать предикат, выясняющий, имеются ли в записях одноименные поля.
11) Промоделировать на списках доступ к полю записи по заданному ключу.
12) Промоделировать на списках замену поля записи на заданное значение.
13) Промоделировать на списках добавление к записи нового имени поля и значения.
14) Промоделировать на списках выборку элемента вектора по индексу
15) Промоделировать на списках присваивание заданному элементу вектора нового значения.
16) Промоделировать на списках вырезку части вектора по заданным границам индексов.
17) Промоделировать на списках слияние двух векторов без потери элементов.
18) Промоделировать на списках слияние произвольного числа векторов.
19) Промоделировать на списках поиск заданного элемента в векторе – определить его индекс.
20) Промоделировать на списках выборку элемента матрицы по списку его индексов
21) Промоделировать на списках присваивание заданному списком индексов элементу матрицы.
22) Промоделировать на списках вырезка минора по заданным координатам элемента.
23) Промоделировать на списках транспонирование матриц.
24) Промоделировать на списках выделение диагонали матрицы
25) Промоделировать на списках поиск координат заданного элемента в матрице
26) Построить список имен функций, входящих в представленное списком выражение (пересечение двух множеств)

27) Выполнить подсчет числа всех атомов в списке с помощью Лисп-функции.
28) Распространить суммирование на две произвольных матрицы одинаковой размерности (матрица сумм соответствующих элементов).
29) Написать функцию, строящую список всех атомов многоуровневого списка.
Пример: (а (в б) е) -> (а в б е)
30) Для списка произвольного типа подсчитать сумму всех чисел, входящих в него.
Например: (а 1 б 3) -> 4
31) Построить список первых (самых левых в записи) атомов подсписков первого уровня с помощью обычных функций. Пример: ((а ь) ((с) (д е)) -> (а с)
32) Подстановка заданного числа вместо атома на верхнем уровне списка (замена атома на число)
33) Подстановка заданного числа вместо атома на всех уровнях произвольного списка (замена атома на число).
34) Задан список пар из атомов и чисел. Отобразить список из атомов в список из соответствующих чисел.
35) Написать рекурсивную функцию подсчета вхождений заданного слова в предложение.

b. задачи для контрольных работ.

1) Алгоритм приведения логической формулы к ДНФ.

Пример: (A & (A v C) & (B v C)) => ((A & B) v (A & C))

---

2) Алгоритм приведения логической формулы к КНФ.

Пример: ((A & B) v (A & C)) => (A & (B v C))

---

3) Алгоритм расстановки в арифметическом выражении скобок в соответствии с порядком выполнения операций

Пример: (a + b * c – e / f) => ((a + (b * c)) – (e / f))

---

4) Алгоритм исключения из арифметического выражения «лишних» скобок, не влияющие на результат вычислений. Приоритеты операций заданы в виде ассоциативного списка.

Пример: (a + (b * c) – (e + f)) => (a + b * c – e - f)
(a + (b * c) – (e - f)) => (a + b * c – (e - f)) => (a + b * c – e + f))
(a + (b * c) + (e - f)) => (a + b * c + e - f)

---

5) Алгоритм проверки корректности химической формулы (соответствие валентностей). Валентности атомов или радикалов заданы в виде ассоциативного списка.

---

6) Алгоритм проверки корректности физической формулы (размерности), считая, что задано соответствие обозначений и размерностей в виде ассоциативного списка.

---

7) Алгоритм представления графа, заданного дугами, списками смежных вершин.

Пример: (смежные ‘((1 2)(2 1)(2 3)(3 4)(1 5))) ; = ((1 2 5) (2 1 3)(3 4))

---

8) Представить дугами граф, заданный списками смежных вершин.

Пример: (дуги ‘((1 2 5) (2 1 3)(3 4))) ; = ((1 2)(2 1)(2 3)(3 4)(1 5))

---

9) Алгоритм "снятия" структуры S-выражения, т.е. построить список, включающий все атомы в той же последовательности, что и в исходной структуре.

Примеры: (уравниловка ‘(A . B)) ;= (A B)
(уравниловка ‘(A (B))) ;= (A B)
(уравниловка ‘(A (B) C ((D)E))) ;= (A B C D E)
(уравниловка ‘A) ;= (A)

---

10) Алгоритм, выясняющий частоту вхождения атома в список. Дать алгоритм построения гистограммы вхождения атомов в список и довести его до обработки произвольного S-выражения.

Примеры: (частота ‘A ‘(A B A C)) ; = 2
(гистограмма ‘ (A B A C)) ; = ((A . 2)(B . 1)(C . 1))

---

11) Алгоритм вычисления индекса первого вхождения в список заданного атома. Затем определить список индексов, показывающих позицию атома в произвольном S-выражении.

Примеры: (индекс ‘A ‘(B A C)) ;= 1
(индекс ‘A ‘(A C)) ;= 0
(индекс ‘A ‘(B C A)) ;= 2

(позиция ‘A ‘(B A C)) ;= (1)
(позиция ‘A ‘(B (A) C)) ;= (1 0)

---

12) По данному списку индексов выбрать из структуры соответствующий элемент. Нумеруем с нуля.

Примеры: (выбор ‘(1 2) ‘(a (b c d e) f)) ; = d
(выбор ‘(0) ‘(a (b c d e) f)) ; = a
(выбор ‘(1 2 0) ‘(a (b c (d e)) f)) ; = d

---

13) Из системы уравнений убрать несущественные. Уравнения представлять списками переменных.

Пример: x = y + z
Y = t + 5

Z = 6 + y
T = z – y
Уравнение для X не влияет на решение

---

14) Из последовательности убрать безрезультатные присваивания

Пример: ((set x 1) (set y 2) (set z y) (set x (+ z y))) – первое присваивание бессмысленно.

---

15) Перечислить в графе маршруты заданной длины (число дуг)
Пример: в графе ((1 2)(2 3)(3 4)(1 5)) длину 3 имеет маршрут (1 2 3 4)

---

16) Проверить достижимость в графе одной вершины из другой. Граф задан списком дуг.
Примеры: в графе ((1 2)(2 3)(3 4)(1 5)) вершина 4 достижима из вершины 2, а 5 – нет.

---

17) Алгоритм приведения алгебраической формулы к упрощенному виду.

Примеры: (+(* 2 x y)(* 3 x z)(* x y)) => (+(* 3 x y)(* 3 x z)) => (* 3 x (+ y z))

---

18) Упрощение арифметической формулы, учитывая свойства + - 0 1

Примеры: ((a – a) * x) + y => y
(a + b) * (x / x) => (a + b)

---

19) Проверить соответствие списка заданному образцу. Пусть специальный атом _ELEM_ означает в образце вхождение произвольного элемента списка, атом _ANY_ - произвольное число элементов списка.

Примеры: (a b c d e) соответствует образцам (a _ELEM_ c d e), (a _ANY_ e),
Не соответствует образцам (a _ELEM_ e), (a _ANY_ c).

---

20) Алгоритм, строящий список, представляющий множество всех атомов многоуровневого списка.

Примеры: (а (в б) е) -> (а в б е)
(а (в а б) е) -> (а в б е)

---

21) Алгоритм для функции инфикс [польская; табл] преобразует выражение из польской в инфиксную запись, используя таблицу соответствия префиксных и инфиксных операций.

Примеры: (инфикс ‘(МНОЖ 3 A B) ‘((ПЛЮС . +) (МНОЖ . *)))
; = (3 * A * B)

(инфикс '(ДЕЛИ (МНОЖ C A B)(ПЛЮС 3 (МНОЖ A 2) X Y))

‘((ПЛЮС . +) (МНОЖ . *) (ДЕЛИ . /)))
; = ((C * A * B) / (3 + (A * 2) + X + Y))

---

22) Функция (префикс инфикс табл) выполняет обратное к функции «инфикс» преобразование, т.е. преобразует выражение из инфиксной записи в префиксную, используя таблицу соответствия префиксных и инфиксных операций.

Примеры: (префикс ‘(3 * A * B) ‘((+ . ПЛЮС) (* . МНОЖ)))
; = (МНОЖ 3 A B)

(префикс ‘((C * A * B) / (3 + (A * 2) + X + Y)) ‘((+ . ПЛЮС) (* . МНОЖ) (/ . ДЕЛИ)))
; = (ДЕЛИ (МНОЖ C A B) (ПЛЮС 3 (МНОЖ A 2) X Y))

---

23) Алгоритм для перестановки строк строит список всех перестановок символов заданной строка в предположении, что все элементы строки уникальны. /Строки представить как список однобуквенных атомов./

Пример: (перестановки ‘(A B C))
; = ((A B C)(A C B)(B C A)(B A C)(C A B)(C B A))

---

24) Алгоритм для перегруппировки строк делает перестановку символов строки без предположений относительно уникальности элементов. Это трудно сделать эффективным образом, без повторов. /Строки представить как список однобуквенных атомов./

Пример: (перегруппировка ‘(A A B)) ; = ((A A B)(A B A)(B A A))

---

25) Подобрать необходимое символьное представление и предложить ветвь для включения в функцию ДИФФ обработчика правила Лейбница, сводящего производную интеграла к разности производных от границ интегрирования.

$$\frac{d}{dy}\int_{a(y)}^{b(y)} f(x,y)\,dx = \int_{a}^{b} \frac{df}{dy}\,dx + f(b,y)\frac{db}{dy} - f(a,x)\frac{da}{dy}$$

---

26) Дан предикат (перед а в), упорядочивающий атомы. Это значит, если (перед а в) = T, то (перед в а) = Nil, где "а" и "в" произвольные различные атомы. Напишите предикат (раньше а в), упорядочивающий структуры с учетом упорядочения атомов.

Пример: (раньше ‘(a b) (b a)) = T
(раньше ‘(a b a) (a b)) = Nil

---

27) Одна структура называется делителем другой, если вторая включает первое как подструктуру. Аналог наибольшего общего делителя для структур - общий делитель, не яв-

ляющийся делителем другого общего делителя, не обязательно единственный. Напишите функцию поиска наибольшего общего делителя для структур.

Пример: (нод ‘(A . ((B . C) . Д)) ‘(E . (B . C))) ; = (B . C)

Найдите все наибольшие общие делители.

Пример: (внод ‘(X . (У . (A . B))) ‘(B . (X . (A . B)))) ; = ((A . B) X)

---

28) Функция (постфикс инфикс табл) преобразует выражение из инфиксной записи в постфиксную, используя таблицу соответствия префиксных и инфиксных операций.

Пример: (постфикс ‘(3 * A * B) ‘((+ . ПЛЮС) (* . МНОЖ))) ; = (3 A B МНОЖ)

(постфикс ‘((C * A * B) / (3 + (A * 2) + X + Y)) ‘((+ . ПЛЮС) (* . МНОЖ) (/ . ДЕЛИ))) ; = ((C A B МНОЖ) (3 (A 2 МНОЖ) X Y ПЛЮС) ДЕЛИ)

2.2.2 Форма и перечень вопросов билета для экзамена 7 семестра

Форма экзаменационного билета

Таблица П1.3

Новосибирский государственный университет
Экзамен

Парадигмы программирования
наименование дисциплины

09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА
Программная инженерия и компьютерные науки
наименование образовательной программы

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ №

1. Вопрос из категории 1
2. Вопрос из категории 2

Составитель

_____
(подпись)

Ответственный за образовательную программу

_____ А.А. Романенко
(подпись)

«____»_____20 г.

Перечень вопросов экзамена, структурированный по категориям, представлен в таблице П1.4

Таблица П1.4

Семестр 7	Формулировка вопроса
Категория 1 (ПКС-2.1)	Примеры различных направлений, стилей и техник программирования
	Общая характеристика императивного программирования
	Общая характеристика функционального программирования
	Общая характеристика логического программирования
	Общая характеристика объектно-ориентированного программирования
	Операционная семантика языков программирования.
	Абстрактная машина и интерпретатор
	C# - механизмы функционального программирования в императивном языке
	F# - функционально-поизводственный язык программирования
	HomeLisp – интерфейс, язык, система
	Haskell – основные идеи
	Язык теоретико-множественного программирования SETL как пример языка сверхвысокого уровня.
	Параллельное программирование на языке APL
	Компиляция программ как синтаксически управляемый обработчик данных
	Организация ленивых (отложенных) вычислений
	Параллельное программирование на языке Sisal
	Обзор основных идей языков сверх высокого уровня
	Основные особенности языков низкого уровня
	Типичные механизмы языков высокого уровня
	Основные особенности языков параллельного программирования
	Средства и методы параллельного программирования в известных языках (примеры)
	C# - работа с недоопределенностью
	C# - Expression trees
	F# - типизация данных и контроль
	F# - изменяемые и защищенные переменных/контексты
	F# - ленивые вычисления
	F# - мемоизация
	F# - работа с внутренним представлением выражений
	F# - монады
	F# - асинхронность и .Net
	C# - управляемых и защищенный код
Категория 2 (ПКС-2.7)	Встроенные типы данных в Haskell: булевский тип, числа, символы, строки.
	Полиморфные типы в языке Haskell. Примеры.
	Списки, конструкторы списков, кортежи в языке Haskell.
	Определение функций в Haskell. Сопоставление с образцом.
	Использование let и where в описаниях функций в Haskell, отличие между ними.
	Лямбда-абстракции в Haskell (\p1...pn -> e). Сечения функций (inc = (+) 1).
	Типы данных в языке Haskell, определяемые пользователем (data).

	Модули в Haskell: списки экспорта, импортирование.
	Стандартные классы в Haskell: Eq, Ord, Num, Show и другие. Определение экземпляров (instance).
	Монада IO и функции ввода-вывода в Haskell.
	Монада Maybe в Haskell и её использование.
	Использование монад в Haskell для создания функций с побочными эффектами в функциональной парадигме.
	Ленивые вычисления в Haskell. Примеры использования.
	"Бесконечные" структуры данных в Haskell. Примеры использования.
	Использование массивов в Haskell.
	Компилятор Lisp-а.
	Функции высших порядков (синтаксический анализатор)
	Структуры данных для реализации списков и атомов Lisp-а
	Венский метод определения языков программирования
	Особенности программирования на ассемблере
	Программирование над стеком в языке Forth
	Макро-обработка текстов в GPM
	Язык управления процессами Bash
	Компьютерные языки, поддерживающие разработку распределенных информационных систем
	Макро-обработка текстов в TRAC
	Разностные списки в языке Prolog
	Бэктрекинг и логический вывод в языке Prolog
	Образовательная классификация парадигм программирования
	Учебные языки программирования XXI века

Набор вопросов формируется и утверждается в установленном порядке в начале учебного года при наличии контингента обучающихся, осваивающих дисциплину «Парадигмы программирования» в текущем учебном году.

3. Критерии оценки сформированности компетенций в рамках промежуточной аттестации по дисциплине

Таблица П1.7

Шифр компетенций	Структурные элементы оценочных средств	Показатель сформированности	Не сформирован	Пороговый уровень	Базовый уровень	Продвинутый уровень
ПКС-2	Портфолио (этап 1), Экзамен (этап 2)	ПКС-2.1 Владеть: навыками разработки программ на языках высокого уровня	Не умеет обоснованно выбирать парадигму программирования в зависимости от уровня изученности класса решаемых задач и модели жизненного цикла разрабатываемой информационной системы	Допускает грубые ошибки при выборе парадигмы программирования	Допускает незначительные ошибки при выборе парадигмы программирования	Умеет обоснованно выбирать парадигму программирования в зависимости от уровня изученности класса решаемых задач и модели жизненного цикла разрабатываемой информационной системы для реальных задач
ПКС-2	Портфолио (этап 1), Экзамен (этап 2)	ПКС-2.7 Уметь: проводить объектную декомпозицию информационной системы, вырабатывать и обосновывать архитектурное решение	Не умеет формулировать постановку задачи и ее решение с помощью	Допускает грубые ошибки, формулируя постановку задачи и ее	Умеет формулировать постановку задачи и ее решение с помощью разных средств, встречающихся в проектировании,	Умеет формулировать постановку задачи и ее решение с помощью разных средств, встречающихся в проектировании, реализации и сопровождения про-

			разных средств, встречающихся в проектировании, реализации и сопровождения программных проектов для особо сложных задач	решение с помощью разных средств, проектировании, реализации и сопровождения	реализации и сопровождения программных проектов для учебных задач	граммных проектов для особо сложных задач сложных реальных задач

4. Критерии выставления оценок по результатам промежуточной аттестации по дисциплине

Результаты промежуточной аттестации в семестре определяются оценками «отлично», «хорошо», «удовлетворительно», «неудовлетворительно». Оценки «отлично», «хорошо», «удовлетворительно» означают успешное прохождение промежуточной аттестации.

Решение об окончательной оценке принимается по результатам 2 этапа (экзамен).

Оценка «отлично» соответствует продвинутому уровню сформированности компетенции.

Оценка «хорошо» соответствует базовому уровню сформированности компетенции.

Оценка «удовлетворительно» соответствует пороговому уровню сформированности компетенции.

Оценка «неудовлетворительно» выставляется при неудовлетворительном прохождении одного или двух этапов промежуточной аттестации.

Лист актуализации фонда оценочных средств
промежуточной аттестации по дисциплине
«Парадигмы программирования»

№	Характеристика внесенных изменений (с указанием пунктов документа)	Дата и № протокола Ученого совета ФИТ	Подпись ответственного
1.	Актуализирован на 2020-2021 уч.год	22.07.2020 №77