МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Новосибирский национальный исследовательский государственный университет»
(Новосибирский государственный университет, НГУ)

Физический факультет
Кафедра физико-технической информатики

УТВЕРЖДАЮ
Декан ФФ, д.ф.-м.н
В.Е.Блинов
«__» _____ 2022 г.

Рабочая программа дисциплины

СОВРЕМЕННЫЕ ТЕХНОЛОГИИ, МЕТОДЫ И ЯЗЫКИ ПРОГРАММИРОВАНИЯ

направление подготовки: **03.04.02 Физика**
направленность (профиль): **Информационные процессы и системы**

Форма обучения
Очная

Семестр	**Общий объем**	**Виды учебных занятий (в часах)**				**Промежуточная аттестация (в часах)**				
		Контактная работа обучающихся с преподавателем			Самостоятельная работа, не включая период сессии	Самостоятельная подготовка к промежуточной аттестации	Контактная работа обучающихся с преподавателем			
		Лекции	Практические занятия	Лабораторные занятия			Консультации	Зачет	Дифференцированный зачет	Экзамен
1	2	3	4	5	6	7	8	9	10	11
		32	32		42				2	
Всего 108 часов / 3 зачётные единицы, из них: - контактная работа 66 часов										
Компетенции ПК-1										

Руководитель программы
д.ф.-м.н. И. Б. Логашенко

Новосибирск, 2022

Содержание

1. Перечень планируемых результатов обучения по дисциплине, соотнесённых с планируемыми результатами освоения образовательной программы.

Дисциплина «Современные технологии, методы и языки программирования» нацелена на ознакомление студентов с рядом широко используемых современных технологий программирования, ознакомление их с различными парадигмами программирования, а также повышение профессиональной эрудиции.

Дисциплина нацелена на формирование у обучающегося профессиональной компетенции:

Результаты освоения образовательной программы (компетенции)	Индикаторы	Результаты обучения по дисциплине
ПК-1 Способен использовать специализированные знания в области физики при решении поставленных задач в научно-исследовательской деятельности в соответствии с профилем подготовки в зависимости от специфики объекта исследования.	**ПК 1.1** Применяет специализированные знания в области физики при решении конкретных задач в области научных исследований в соответствии с профилем подготовки в зависимости от специфики объекта исследования. **ПК 1.2** Выбирает наиболее эффективные методы решения конкретных задач в области научных исследований в соответствии с профилем подготовки в зависимости от специфики объекта исследования.	**Знать** основные функции высшего порядка, принципы работы с бесконечными списками. **Уметь** выражать основные функции высшего порядка, выражать итеративные алгоритмы через комбинации функций высшего порядка, работать с бесконечными списками, реализовывать простые программы на чистом функциональном языке с использованием ввода/вывода, создавать простые клиент-серверные приложения с веб-интерфейсом. **Владеть** регулярными выражениями, семейством XML-технологий, словарем аспектно-ориентированной парадигмы, способами реализации асинхронных вычислений.

2. Место дисциплины в структуре образовательной программы.

Дисциплина «Современные технологии, методы и языки программирования» реализуется в осеннем семестре для студентов первого курса магистратуры, обучающихся по направлению подготовки **03.04.02 Физика, направленность «Информационные процессы и системы»**. Курс является одной из профессиональных дисциплин по выбору, реализуемых кафедрой физико-технической информатики. Для достижения поставленной цели выделяются задачи курса:

1. сравнительное рассмотрение основных парадигм программирования;
2. знакомство с языком программирования, использующим динамическую проверку типов;
3. изучение регулярных выражений;
4. знакомство с семейством технологий XML;
5. знакомство с функциональным программированием и с языком программирования, использующим эту парадигму;
6. знакомство с инструментами для использования аспектно-ориентированного программирования;

7. знакомство некоторыми с языками и технологиями, использующимися в создании клиент-серверных приложений с веб-интерфейсом.

3. Трудоёмкость дисциплины в зачётных единицах с указанием количества академических часов, выделенных на контактную работу обучающегося с преподавателем (по видам учебных занятий) и на самостоятельную работу.

Семестр	Общий объем	Виды учебных занятий (в часах)				Промежуточная аттестация (в часах)				
		Контактная работа обучающихся с преподавателем			Самостоятельная	Самостоятельная	Контактная работа обучающихся с преподавателем			
		Лекции	Практические занятия	Лабораторные занятия			Консультации	Зачет	Дифференцированный зачет	Экзамен
1	2	3	4	5	6	7	8	9	10	11
		32	32		42				2	
Всего 108 часов / 3 зачётные единицы, из них: - контактная работа 66 часов										
Компетенции ПК-1										

Реализация дисциплины предусматривает практическую подготовку при проведении следующих видов занятий, предусматривающих участие обучающихся в выполнении отдельных элементов работ, связанных с будущей профессиональной деятельностью: лекции, практические занятия, самостоятельная работа студента и её контроль преподавателями с помощью заданий, дифференцированный зачет.

Программой дисциплины предусмотрены следующие виды контроля:

- текущий контроль успеваемости: задания для самостоятельного решения;
- промежуточная аттестация: дифференцированный зачёт.

Общая трудоемкость рабочей программы дисциплины составляет 3 зачетные единицы.

- занятия лекционного типа – 32 часа;
- практические занятия – 32 часа;
- самостоятельная работа обучающегося в течение семестра, не включая период сессии – 42 часа;
- промежуточная аттестация (дифференцированный зачёт) – 2 часа.

Объём контактной работы обучающегося с преподавателем (лекции, практические занятия, дифференцированный зачёт) составляет 66 часов.

4. Содержание дисциплины, структурированное по темам (разделам) с указанием отведённого на них количества академических часов и видов учебных занятий.

Общая трудоемкость дисциплины составляет 3 зачётные единицы, 108 академических часов.

№ п/п	Раздел дисциплины	Неделя семестра	Виды учебной работы, включая самостоятельную работу студентов и трудоёмкость (в часах)	Промежуточная аттестация (в часах)

			Всего	Аудиторные часы		Сам. работа во время занятий (не включая период сессии	Сам. работа во время промежуточной аттестации	
				Лекции	Практические занятия			
1	2	3	4	5	6	7	8	9
1.	Современные динамические языки.	1-3	16	6	6	4		
2.	XML-технологии	4-6	16	6	6	4		
3.	Функциональное программирование.	7-10	22	8	8	6		
4.	Аспектно-ориентированное программирование. Классы задач, решаемых с помощью АОП: персистентность, журналирование, авторизация, транзакционность. Проблема модульности. Понятие сквозной функциональности (cross-cutting concerns). Понятие аспекта, аспектная декомпозиция сложных систем. Основы языка AspectJ. Модель компиляции и исполнения.	11-12	14	4	4	6		
5.	Языки и технологии Web-программирования.	13-16	22	8	8	6		
6.	Подготовка к дифференцированному зачёту	16	16			16		
7.	Дифференцированный зачёт		2					2
Всего			**108**	**32**	**32**	**42**		**2**

Программа и основное содержание лекций (32 часа)

Раздел 1. Современные динамические языки (6 часов).

Классификация языков программирования по назначению и модели исполнения. Языки сценариев. Встраиваемые языки. Современное представление об интерпретируемых языках. Байт-код. JIT-компиляция. Динамическая типизация. Язык JavaScript: общие характеристики, историческая сводка, система типов, конструкции и операторы языка, функции как объекты первого класса, замыкания. Элементы функционального программирования в JavaScript. Отладка программ на JavaScript. Объектная модель JavaScript: прототипно-ориентированный стиль, понятие

прототипа, создание и модификация объектов, использование замыканий для инкапсуляции, наследование, полиморфизм. Регулярные выражения в JavaScript и других языках.

Раздел 2. XML-технологии (6 часов)

Структура и назначение языка XML. Well-formed и valid документы. Модели синтаксического анализа XML: SAX и DOM. Валидация XML. DTD. XML-Schema. XSL-преобразования. Обзор технологий и языков, основанных на XML.

Раздел 3. Функциональное программирование (8 часов)

Идеи функционального программирования. Лямбда-исчисление Черча. Аппликация и абстракция. β-редукция. η-преобразование. Функции высшего порядка в функциональных языках программирования, каррирование. Комбинаторы. Нетипизированное и типизированное λ-исчисление. Введение в язык Haskell, основные языковые конструкции, сравнение с образцом. Типы, классы типов. Списки. Бесконечные структуры данных, ленивые вычисления. Эмуляция ленивых вычислений в императивных языках программирования. Введение в функторы и монады. do-нотация и написание программ с побочными эффектами.

Раздел 4. Аспектно-ориентированное программирование (4 часа)

Классы задач, решаемых с помощью АОП: персистентность, журналирование, авторизация, транзакционность. Проблема модульности. Понятие сквозной функциональности (cross-cutting concerns). Понятие аспекта, аспектная декомпозиция сложных систем. Основы языка AspectJ. Модель компиляции и исполнения.

Раздел 5. Языки и технологии Web-программирования (8 часов)

История WWW. HTML-страница, сценарии, стили. Введение в CSS: селекторы, свойства, box-model, форматирование веб-документа. DOM-модель в JavaScript: доступ к XML/HTML-дереву, модификация дерева, управление атрибутами, события. Событийно-ориентированная архитектура. Коллбеки, промисы, конструкции async/await Язык представления структурированных данных JSON. Возможности новых стандартов ECMAScript. Классификация инструментов веб-разработчика, примеры библиотек. Технология AJAX: общие принципы работы, паттерны использования. Библиотека jQuery: селекторы, манипуляции с CSS и DOM, обход DOM-дерева. Использование JavaScript на сервере. Простой сервер на Node.js.

Программа практических занятий (32 часа)

Раздел 1. Современные динамические языки (6 часов).

Занятие 1. Организационные моменты. Вход на учебные ПК, проверка наличия инструментов для работы (2 часа)

Занятие 2. Язык программирования JavaScript и стандартная библиотека, преобразования гомогенных коллекций – map, reduce, filter (2 часа)

Занятие 3. Perl-совместимые регулярные выражения, реализация в JavaScript (2 часа)

Раздел 2. XML-технологии (6 часов)

Занятие 4. Язык разметки XML, SAX-парсер, работа с объектной моделью документа (DOM) (2 часа)

Занятие 5. Схема документа XML - технология XSD (2 часа)

Занятие 6. Преобразования XML- технология XSLT (2 часа)

Раздел 3. Функциональное программирование (8 часов)

Занятие 7. Преобразование бесконечных гомогенных коллекций в функциональном стиле на языке JavaScript (2 часа)

Занятие 8. Принципы реализации преобразований гомогенных коллекций на языке Haskell (2 часа)

Занятие 9. Реализация простых функций с монадическим вводом-выводом (2 часа)

Занятие 10. Написание простых программ в функциональном стиле (2 часа)

Раздел 3. Аспектно-ориентированное программирование (4 часа)

Занятие 11. Аспектно-ориентированное программирование (2 часа)

Занятие 12. Разбор вопросов по функциональному программированию, аспектно-ориентированному программированию (2 часа)

Раздел 5. Языки и технологии Web-программирования (8 часов)

Занятие 14. Работа с функциями обратного вызова и абстракциями над ними – Promise, async/await (2 часа)

Занятие 15. Простые клиент-серверные веб-приложения (2 часа)

Занятие 16. Разбор вопросов по темам курса (2 часа)

Самостоятельная работа студентов (42 часа)

Перечень занятий на СРС	Объем, час
Решение задач из задания для самостоятельного решения	20
Изучение теоретического материала, не освещаемого на лекциях	6
Подготовка к дифференцированному зачёту	16

5. Перечень учебной литературы.

1. Абельсон, Харольд. Структура и интерпретация компьютерных программ : [пер. с англ.] / Харольд Абельсон, Джеральд Джей Сассман, при участии Джули Сассман. [2-е изд.]. Москва : Добросвет : КДУ, 2011. 608 с., ISBN 978-5-98227-829-6 (10 экз)

2. Городняя, Лидия Васильевна. Парадигма программирования. Новосибирск : Институт систем информатики им. А.П. Ершова, 2014-21 см(Препринт / Рос. акад. наук, Сиб. отд-ние, Ин-т систем информатики им. А.П. Ершова ; ...)
(Ч1- 1 экз, Ч2-1 экз, Ч3- 1 экз, Ч4- 1 экз, Ч5- 1экз)

6. Перечень учебно-методических материалов по самостоятельной работе обучающихся.

Самостоятельная работа студентов поддерживается следующими учебными пособиями:

1.Абельсон, Харольд. Структура и интерпретация компьютерных программ : [пер. с англ.] / Харольд Абельсон, Джеральд Джей Сассман, при участии Джули Сассман. [2-е изд.]. Москва : Добросвет : КДУ, 2011. 608 с.

7. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины.

Для освоения дисциплины используются следующие ресурсы:
- электронная информационно-образовательная среда НГУ (ЭИОС);
- образовательные интернет-порталы;
- информационно-телекоммуникационная сеть Интернет.

Интернет-ресурсы:

- Пугачев К.В. Современные технологии, методы и языки программирования. Электронный лекционный курс / Новосибирск: ИЯФ, 2020 [https://wwwsnd.inp.nsk.su/~pugachev/nsu-prog/mtmpl/]

7.1 Современные профессиональные базы данных

Не используются.

7.2. Информационные справочные системы

Не используются

8. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине.

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

Могут применяться инструменты, используемые через Интернет-браузер (онлайн среды разработки, онлайн компиляторы, веб-интерфейсы систем управления версиями — [https://repl.it/], [http://aspectjml.cin.ufpe.br/], [https://github.com/] или аналогичные) при их доступности в сети "Интернет" на момент проведения занятий курса.

При отсутствии онлайн инструментов используется дополнительное бесплатное ПО или дополнительное свободное ПО: текстовый редактор Notepad++ [https://notepad-plus-plus.org/], среда Eclipse [https://www.eclipse.org/] с плагином AspectJ [https://www.eclipse.org/aspectj/], Node.js [https://nodejs.org/], Haskell Stack [https://www.haskellstack.org/] (или аналоги).

9. Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине.

Для реализации дисциплины используются специальные помещения:

1. Учебные аудитории для проведения занятий лекционного типа, практических занятий, курсового проектирования (выполнения курсовых работ), групповых и индивидуальных консультаций, текущего контроля, промежуточной и итоговой аттестации.

2. Помещения для самостоятельной работы обучающихся.

Учебные аудитории укомплектованы специализированной мебелью и техническими средствами обучения, служащими для представления учебной информации большой аудитории.

Помещения для самостоятельной работы обучающихся оснащены компьютерной техникой с возможностью подключения к сети "Интернет" и обеспечением доступа в электронную информационно-образовательную среду НГУ.

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».

10. Оценочные средства для проведения текущего контроля и промежуточной аттестации по дисциплине.

10.1 Порядок проведения текущего контроля и промежуточной аттестации по дисциплине

Текущий контроль

Текущий контроль осуществляется в ходе семестра: во время практических занятий студентам необходимо сдавать задания.

Промежуточная аттестация

Освоение компетенций оценивается согласно шкале оценки уровня сформированности компетенции. Положительная оценка по дисциплине выставляется в том случае, если заявленная

компетенция ПК-1 сформирована не ниже порогового уровня в части, относящейся к формированию способности использовать специализированные знания в области физико-технической информатики.

Окончательная оценка работы студента в течение семестра происходит на дифференцированном зачёте. Оценка по дисциплине проводится по результатам работы в семестре. Итоги дифференцированного зачета оцениваются по количеству сданных задач из заданий для самостоятельного решения. Незнание технических деталей и понимания решения задачи расценивается как попытка списывания и данная задача не засчитывается.

- оценка «отлично» ставится если сданы все задания (продвинутый уровень освоения компетенций);

- оценка «хорошо» ставится, если не меньше половины заданий сданы полностью, а из остальных сдано всё, кроме одной задачи (базовый уровень освоения компетенций);

- оценка «удовлетворительно» ставится если сдано не менее одной задачи из каждого задания (пороговый уровень освоения компетенций);

- оценка «неудовлетворительно» ставится если сдано менее одной задачи в каком-либо задании (пороговый уровень освоения компетенций не сформирован).

Соответствие индикаторов и результатов освоения дисциплины

Таблица 10.1

Индикатор	**Результат обучения по дисциплине**	**Оценочные средства**
ПК 1.1 Применяет специализированные знания в области физики при решении конкретных задач в области научных исследований в соответствии с профилем подготовки в зависимости от специфики объекта исследования.	**Знать** основные функции высшего порядка, принципы работы с бесконечными списками.	Выполнение практические занятия, дифференцированный зачет.
ПК 1.2 Выбирает наиболее эффективные методы решения конкретных задач в области научных исследований в соответствии с профилем подготовки в зависимости от специфики объекта исследования.	**Уметь** выражать основные функции высшего порядка, выражать итеративные алгоритмы через комбинации функций высшего порядка, работать с бесконечными списками, реализовывать простые программы на чистом функциональном языке с использованием ввода/вывода, создавать простые клиент-серверные приложения с веб-интерфейсом. **Владеть** регулярными выражениями, семейством XML-технологий, словарем аспектно-ориентированной парадигмы, способами реализации асинхронных вычислений.	Выполнение практические занятия, дифференцированный зачет.

10.2 Описание критериев и шкал оценивания индикаторов достижения результатов обучения по дисциплине «Современные технологии, методы и языки программирования».

Таблица 10.2

Критерии оценивания результатов обучения	Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенций)	Уровень освоения компетенции			
		Не сформирован (0 баллов)	Пороговый уровень (3 балла)	Базовый уровень (4 балла)	Продвинутый уровень (5 баллов)
1	2	3	4	5	6
Полнота знаний	ПК 1.1	Уровень знаний ниже минимальных требований. Имеют место грубые ошибки.	Минимально допустимый уровень знаний. Допускается значительное количество негрубых ошибок.	Уровень знаний соответствует программе подготовки по темам/разделам дисциплины. Допускается несколько негрубых/ несущественных ошибок. Не отвечает на дополнительные вопросы.	Уровень знаний соответствует программе подготовки по темам/разделам дисциплины. Свободно и аргументированно отвечает на дополнительные вопросы.
Наличие умений	ПК 1.2	Отсутствие минимальных умений. Не умеет решать стандартные задачи. Имеют место грубые ошибки.	Продемонстрированы частично основные умения. Решены типовые задачи. Допущены негрубые ошибки.	Продемонстрированы все основные умения. Решены все основные задания с негрубыми ошибками или с недочетами.	Продемонстрированы все основные умения. Решены все основные задания в полном объеме без недочетов и ошибок.
Наличие навыков (владение опытом)	ПК 1.2	Отсутствие владения материалом по темам/разделам дисциплины. Нет навыков в решении стандартных задач. Наличие грубых ошибок.	Имеется минимальный набор навыков при решении стандартных задач с некоторыми недочетами.	Имеется базовый набор навыков при решении стандартных задач с некоторыми недочетами.	Имеется базовый набор навыков при решении стандартных задач без ошибок и недочетов. Продемонстрированы знания по решению нестандартных задач.

10.3 Типовые контрольные задания и материалы, необходимые для оценки результатов обучения

Перечень типовых заданий для самостоятельного решения

ЗАДАНИЕ №1. Динамические ЯП

Задача 1. Реализация основных функций высшего порядка (ФВП) – часть 1

Задача 2. Реализация основных ФВП – часть 2

Задача 3. Применение ФВП для написания алгоритма

Задача 4. Разбор текстовых данных регулярными выражениями

ЗАДАНИЕ №2. Энтерпрайз-инструменты

Задача 5. Работа с XML с использованием SAX- и DOM-парсеров, XPath
Задача 6. Работа с XML с использованием XSD, XSLT
Задача 7. Вынесение сквозной ответственности (cross-cutting concern) в аспект

ЗАДАНИЕ №3. Функциональное программирование
Задача 8. Реализация основных ФВП на функциональном ЯП
Задача 9. Реализация бесконечных структур данных
Задача 10. Написание функции с побочными эффектами
Задача 11. Написание программы с ФВП и побочными эффектами

ЗАДАНИЕ №4. Веб-программирование
Задача 12. Реализация цепочки асинхронных действий с использованием нескольких подходов
Задача 13. Реализация простого клиент-серверного приложения с веб-интерфейсом

Расшифровка условий задач

Задача 1. Реализовать filter, map и reduce без циклов, переменных и нехвостовой рекурсии.

Задача 2. Выразить через filter/map/reduce (и приделать к Array.prototype при отсутствии):
a. some и every;
b. аналоги enumerate и zip из Python, не используя Array.keys или второй аргумент аргумента map;
c. аналоги uniq, group_by и each_slice из Ruby.

Задача 3. Для заданного набора символов получить все возможные строки длиной n, состоящие из этих символов, и не содержащие двух одинаковых символов, идущих подряд. Не допускается использовать циклы и переменные, выразите все через filter/map/reduce. Для символов ['a', 'b', 'c'] и n=2 результат должен быть ["ab", "ac", "ba", "bc", "ca", "cb"] с точностью до перестановки.

Задача 4. Написать программу, разбирающую журнал событий (например, лог web-сервера, syslog, детализацию звонков и т. д.) и вычисляющий ежемесячную/ежедневную/почасовую/... статистику в зависимости от содержимого выбранного журнала (например, обращения с различных IP-адресов, ошибки различных типов, аутентификацию пользователей и т.д.)

Задача 5. Сочинить XML-документ. Разобрать его с помощью SAX-парсера, подсчитать количество элементов, сгруппированное по названиям. Разобрать его с помощью DOM-парсера, модифицировать содержимое и сохранить результат. Извлечь значение атрибута при помощи XPath.

Задача 6. Определить XSD-схему для документа из задачи 5. Валидировать документ на основе схемы. Определить и применить стиль XSL для трансформации в HTML.

Задача 7 (вариант 1). Покопайтесь в вашей коллекции кода на Java. Выберите одну программу, в которой четко прослеживается cross-cutting concern (или несколько): персистентность, версионирование данных, валидация, кеширование, … или какой-нибудь особо сквозной и обособленный кусок бизнес-логики. Выделите выбранный (выбранные) cross-cutting concern в аспект AspectJ.

Задача 7 (вариант 2). Реализуйте по аналогии с примером Observer универсальный абстрактный аспект, который обеспечивал бы функциональность undo/redo. Привязка к конкретным классам

должна осуществляться за счет аспектов-наследников. Продемонстрируйте работу аспекта на примере собственной реализации класса, представляющего двумерную точку (либо любого другого класса данных). Класс должен реализовать только функции бизнес-логики (аксессоры к атрибутам).

Задача 7 (вариант 3). В приведенном ниже тексте выделите разными цветами разные cross-cutting concerns, проранжируйте их по степени удаленности от бизнес-логики. Набросайте UML-диаграмму классов описанной системы. Для каждого из cross-cutting concerns проведите анализ, выделять его в аспект или нет, решение обоснуйте. Дорисуйте соответствующими цветами аспекты, словами распишите соответствующие pointcut'ы и advices. Выберите шаблон проектирования и вынесите его в аспект, абстрагируйте этот аспект.

Требуется спроектировать универсальный коммутатор голосовых вызовов и текстовых сообщений. Должны поддерживаться телефонная сеть / SMS, SIP, Skype, Jingle, система должна легко расширяться поддержкой новых протоколов. Звонок или сообщение на один из выделенных номеров в одной из сетей должны коммутироваться на номер/handle в другой сети согласно таблице коммутации. Факты установки, завершения или разрыва соединения, а также данные и метаданные клиентских и сервисных (бесплатных) сообщений должны протоколироваться в соответствии с форматом MADEUP 4.7. Невозможность установления соединения или отправки сообщения должна сопровождаться бесплатным текстовым сообщением инициатору с пояснением причины. Запрещено перенаправнять звонки или сообщения на номера/handles, уже перенаправляемые системой куда-либо еще. Авторизация клиентов происходит по номеру/handle инициатора звонка/сообщения. Звонки и сообщения должны тарифицироваться согласно гибкому набору правил (таблица поминутных расценок из каждой сети в каждую + скидки и акции), неавторизованные клиенты или клиенты с отрицательным балансом не могут устанавливать новые звонки, отправлять или получать платные сообщения. Изменение баланса на счете клиента должно сопровождаться бесплатным сообщением на номер/handle, указанный клиентом как основной способ связи. Таблица коммутации, счета и контактная информация клиентов хранятся в БД.

Задача 8. Реализуйте filter, map, foldl, foldr. на языке Haskell. Реализуйте факториал через свёртку бесконечного списка [1..] : fac n = fold? ??? [1..].
Примечание: чтобы не было конфликтов со стандартной библиотекой, достаточно импотировать её явно, скрывая реализуемые функции:

```
import Prelude hiding (filter, map, foldl, foldr)
```

Задача 9. Реализуйте бесконечный список на языке JavaScript и факториал через свёртку первых натуральных чисел. Можно ли обойтись без использования take?
Вспомогательный код:

```
const cons, head, tail, map, take; // TODO: implement
const nil = {}; // why nil's implemented this way?
const nats = cons(() => 1, () => map(x => x+1, nats));
const foldl1 = (f, xs) => tail(xs) === nil ?
  head(xs) : f(head(xs), foldl1(f, tail(xs)));
console.log(head(nats)); // 1
console.log(head(tail(nats))); // 2
const fac = n => foldl1((acc, x) => acc*x, take(n, nats));
console.log(fac(5)); // 120
```

Примечание: для реализации бесконечного списка достаточно иметь ленивый конструктор. В императивных языках ленивость можно эмулировать через оборачивание выражения в функцию (JS, C++11) или объект, имеющий геттер (JS, C#) или оператор неявного преобразования типа (JS, C++).

Задача 10. Напишите монадическую функцию fileHash :: FilePath -> IO FileHash

вычисляющую какую-нибудь хэш-функцию от файла.
Функция должна вернуть монадическое значение, которое

- считает файл с указанным именем
- посчитает хэш (некоторый тип FileHash)
- вернёт этот хэш

Для проверки работы возьмите список из некоторых имён, например

```
names :: [FilePath]
names = [ "file1.txt", "file2.txt", file3.txt"]
```

и распечатайте хэши этих файлов используя монадический аналог map.
Примечания:

- считать файл можно с помощью Data.ByteString.readFile;
- пакет SHA позволяет считать хэши одноимённого семейства;
- Data.ByteString.Lazy.fromChunks преобразует строгую строку в ленивую.

Задача 11. Напишите программу, которая анализирует содержимое каталога, указанного в аргументах командной строки, и отображает информационную сводку.

- Вывести распределение файлов по типам данных.

В отдельном месте указано соответствие типов файлов и расширений:

```
contentTypes = [
  ("Image", ["jpg","png","gif","bmp"]),
  ("Document", ["odt","doc","docx","txt"]),
  ("Audio", ["mp3","wav","flac"]),
  ("Video", ["avi","mp4","mkv"]),
  ("Source", ["c","cpp","hs","js","py"])
  ]
```

8. Вывести список продублированных файлов (разрешается включить в список не только файлы-копии, но и оригиналы).

Примечания:

- в пакете containers есть ассоциативный массив Map;
- пакет SHA позволяет считать хэши одноимённого семейства;
- Получить список всех файлов внутри каталога можно с помощью функции:

```
-- import System.Directory (listDirectory, doesDirectoryExist)
-- import System.FilePath.Posix (joinPath, FilePath)
-- import Data.Functor ((<$>))
listDirectoryDeep :: FilePath -> IO [FilePath]
listDirectoryDeep dir = listDirectory dir >>= traverse where
  traverse dirs = concat <$> mapM traverseItem dirs
  traverseItem name = do
    let path = joinPath [dir, name]
    e <- doesDirectoryExist path
    if e
      then listDirectoryDeep path
      else return [path]
```

Возможно, написание следующих функций облегчит решение задачи:

```
typeof :: FilePath -> Maybe ContentTypeName
stats :: [FilePath] -> [(ContentTypeName, Int)]
fileHash :: FilePath -> IO FileHash
duplicates :: [(FilePath, FileHash)] -> [FilePath]
```

Задача 12. Реализуйте функцию для асинхронного решения линейного уравнения с помощью (а) коллбеков, (б) промисов и (в) async/await.
В синхронном случае функция выглядит следующим образом:

```
function solve(a, b) { // ax+b = 0
  if (!a && !b) return [];
  if (!a) return null;
  return [-b/a];
```

```
}
```

В асинхронном случае коэффициенты и операции над ними вычисляются с помощью асинхронной функции math:

```
// math('a', (err, a) => {})
// math('b', (err, a) => {})
// math('/', a, b, (err, AdivB) => {})
// math('-', a, (err, negA) => {})
function math(what, ...args_) {
  var vars = {a: 3, b: 2};
  if (!args_.length) throw new Error('Callback not found!');
  var args = args_.slice(0, -1), cb = args_[args.length];
  function pass(error, value) {
    // An AJAX request to a real server could be here.
    if(Math.random() < 0.2) {
      error = new Error('Network error');
      value = null;
    }
    setTimeout(cb.bind(null, error, value), 500 + Math.random() * 500 | 0);
  }
  switch(what) {
    case 'a':
    case 'b': pass(null, vars[what]); break;
    case '/':
      if (args.length < 2) pass(new Error('Not enough arguments'));
      else pass(null, args[0] / args[1]);
      break;
    case '-':
      if (args.length < 1) pass(new Error('Not enough arguments'));
      else pass(null, -args[0]);
      break;
    default: pass(new Error('Invalid command'));
  }
}
```

- Для случаев промисов и async/await понадобится обёртка над math.
- Пользователь функции должен иметь возможность обработать ошибки.
- Уделите особое внимание параллельному выполнению. В нашем случае коэффициенты a и b могут запрашиваться одновременно, что сокращает время ожидания результата пользователем. Какие средства лучше всего справляются с этой задачей?

Задача 13. Напишите веб-чат (клиент-серверное приложение).

- Клиент – браузерный JS, сервер - приложение под Node.js.
- Сервер логически содержит главную страницу, статические файлы, API.

Примерный интерфейс сервера

URL	type	method	комментарий
/	HTML	GET	
/static/index.css	CSS	GET	кэширование
/static/index.js	JS	GET	защита от XSS, кэширование
/static/ding.mp3	audio	GET	кэширование
/api/messages	JSON	GET	запрашивать только нужные сообщения
/api/post-message	JSON	POST	защита от CSRF

- Клиент содержит как минимум
  - поле ввода имени пользователя,
  - список пользователей чата,
  - последние сообщения,
  - поле ввода сообщения и кнопку отправки.
- Элементы интерфейса клиента визуально разделены (рамками/цветом фона/...).

- Используется блочная вёрстка, семантический HTML. Стили и скрипты вынесены в отдельные файлы со статическим содержимым, в основной HTML включаются только их динамически генерируемые части.
- Клиент автоматически следит за появлением новых сообщений и пользователей (как вариант – постоянно держит соединение с сервером до получения обновлений) и автоматически обновляет интерфейс без перезагрузки страницы.
- При упоминании пользователя в сообщении выдаётся звуковой сигнал.
- Клиент, подключившийся во время беседы, может получить N предыдущих сообщений.
- Клиент корректно обрабатывает ошибки сети, неудачные коды состояния HTTP, разъединение с сервером; при невозможности отправки сообщения его текст не теряется и может быть отправлен заново, в т.ч. для отправки нескольких сообщений, список загруженных сообщений не содержит внутренних разрывов.
- Сервер корректно обрабатывает ошибки сети, неудачные коды состояния HTTP, разъединение с клиентом, некорректные HTTP-запросы клиентов;
- Аутентификация не обязательна, можно считать любого клиента авторизованным для написания комментариев.

Использование фреймворков и инструментов не запрещается. Однако, при упрощении задачи или существовании библиотечного решения в неё могут быть добавлены дополнительные условия.

Оценочные материалы по промежуточной аттестации, предназначенные для проверки соответствия уровня подготовки по дисциплине требованиям СУОС, хранятся на кафедре-разработчике РПД в печатном и электронном виде.

Лист актуализации рабочей программы
по дисциплине «Современные технологии, методы и языки программирования»
по направлению подготовки 03.04.02 Физика
Профиль «Информационные процессы и системы»

№	Характеристика внесенных изменений (с указанием пунктов документа)	Дата и № протокола Учёного совета ФФ НГУ	Подпись ответственного