

**Федеральное государственное автономное образовательное
учреждение высшего образования
«Московский физико-технический институт
(национальный исследовательский университет)»**

УТВЕРЖДЕНО

**Директор физтех-школы
прикладной математики и
информатики
А.М. Райгородский**

	Рабочая программа дисциплины (модуля)
по дисциплине:	Функциональное программирование
по направлению:	Информатика и вычислительная техника
профиль подготовки:	Физтех-школа Прикладной Математики и Информатики кафедра алгоритмов и технологий программирования
курс:	3
квалификация:	бакалавр

Семестр, формы промежуточной аттестации: 6 (весенний) - Дифференцированный зачет

Аудиторных часов: 60 всего, в том числе:

лекции: 30 час.

семинары: 30 час.

лабораторные занятия: 0 час.

Самостоятельная работа: 75 час.

Всего часов: 135, всего зач. ед.: 3

Количество контрольных работ, заданий: 1

Программу составил: Д.В. Сошников, канд. физ.-мат. наук, доцент, доцент

Программа обсуждена на заседании кафедры алгоритмов и технологий программирования 04.06.2020

Аннотация

Курс посвящен знакомству с функциональной парадигмой программирования в общем и с языками программирования F#, Haskell, LISP.

Студенты получают навык написания эффективных функциональных программ.

Здесь находят приложение идеи, освещаемые в курсе математической логики, что способствует укреплению изучаемого параллельно теоретического материала.

1. Цели и задачи

Цель дисциплины

изучение студентами парадигмы функционального программирования, знакомство с языками функционального программирования F#, Haskell, LISP, получение навыков написания эффективных функциональных программ.

Задачи дисциплины

В результате прохождения учебного курса студенты должны:

- быть в состоянии использовать функциональный подход и функциональные языки для решения практических задач в тех областях, где это представляется удобным и практичным
- самостоятельно выделять такие задачи и оценивать преимущества использования функционального подхода, проектировать программные системы и проекты на основе мультипарадигмального подхода
- понимать взаимосвязь лямбда-исчисления как теоретической модели вычислений с практическими аспектами функционального программирования
- использовать более чистый (свободный от побочных эффектов) стиль программирования с высоким уровнем абстракции, научиться эффективно использовать новые функциональные возможности современных императивных языков (LINQ, лямбда-выражения и т.д.).

2. Перечень формируемых компетенций

Освоение дисциплины направлено на формирование следующих компетенций:

Код и наименование компетенции	Индикаторы достижения компетенции
УК-1 Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	УК-1.2 Находит, критически анализирует и выбирает информацию, необходимую для решения поставленной задачи
	УК-1.1 Анализирует задачу, выделяя этапы ее решения, действия по решению задачи
	УК-1.3 Рассматривает различные варианты решения задачи, оценивает их преимущества и недостатки
	УК-1.4 Грамотно, логично, аргументированно формирует собственные суждения и оценки
	УК-1.5 Определяет и оценивает практические последствия возможных вариантов решения задачи
ОПК-1 Способен применять фундаментальные знания, полученные в области физико-математических и (или) естественных наук, и использовать их в профессиональной деятельности	ОПК-1.1 Способен анализировать поставленную задачу, намечать пути ее решения
	ОПК-1.2 Способен строить математические модели, производить количественные расчеты и оценки
	ОПК-1.3 Способен определять границы применимости полученных результатов
ОПК-2 Способен использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности, соблюдая требования информационной безопасности	ОПК-2.1 Способен применять современные вычислительную технику и сервисы сети Интернет в области (сфере) профессиональной деятельности
	ОПК-2.2 Знает и умеет применять численные математические методы и прикладное программное обеспечение для решения научных задач в профессиональной области

	ОПК-2.3 Знает основные требования информационной безопасности
ОПК-3 Способен составлять и оформлять научные и (или) технические (технологические, инновационные) отчеты (публикации, проекты)	ОПК-3.1 Знает основные правила оформления научных публикаций и научно-технической документации, в том числе с использованием прикладного программного обеспечения
	ОПК-3.2 Владеет на практике методологией составления научно-технических отчетов (проектов)
	ОПК-3.3 Владеет методами визуального и графического представления результатов научной (научно-технической, инновационной технологической) деятельности в виде отчетов, научных публикаций
ПК-1 Способен ставить, формализовывать и решать задачи, в том числе разрабатывать и исследовать математические модели изучаемых явлений и процессов, системно анализировать научные проблемы, получать новые научные результаты	ПК-1.3 Способен применять теоретические и (или) экспериментальные методы исследований к конкретной научной задаче и интерпретировать полученные результаты
	ПК-1.2 Способен выдвигать гипотезы, строить математические модели для описания изучаемых явлений и процессов, оценить качество разработанной модели
	ПК-1.1 Способен находить, анализировать и обобщать информацию об актуальных результатах исследований в рамках тематической области своей профессиональной деятельности
ПК-2 Способен самостоятельно или в качестве члена (руководителя) малого коллектива организовывать и проводить научные исследования и их апробацию	ПК-2.1 Знает принципы построения научной работы, методы сбора и анализа полученного материала, способы аргументации
	ПК-2.2 Способен планировать и проводить научные исследования самостоятельно или в качестве члена (руководителя) малого научного коллектива
	ПК-2.3 Способен проводить апробацию результатов научно-исследовательской работы посредством публикации научных статей и участия в конференциях

3. Перечень планируемых результатов обучения по дисциплине (модулю)

В результате освоения дисциплины обучающиеся должны

знать:

- общие принципы функционального подхода к программированию, преимущества и недостатки функционального подхода для реализации программных систем;
- алгоритмические модели;
- исчисления и комбинаторной логики, лежащие в основе функционального программирования;
- инструментальные средства и основные языки функционального программирования;
- использование функционального стиля программирования и элементов функционального программирования в традиционных императивных языках (C++, C# 3.0), в языках трансформации XSLT и др.;
- подходы и средства к построению трансляторов с функциональных языков, на основе интерпретации и компиляции в код абстрактной машины;
- подходы к описанию семантики функциональных языков на основе денотационной семантики и операционной семантики абстрактной машины.

уметь:

- разрабатывать, кодировать, тестировать и отлаживать программы на языках функционального программирования или в функциональном стиле;
- использовать функции высших порядков, функции-как-данные и замыкания;
- использовать языки функционального программирования для реализации известных алгоритмов информатики;
- выделять характерные задачи для применения функционального подхода и предлагать способы их решения;
- использовать подходы и языки логического программирования при построении программных систем, в том числе совместно с традиционными системами программирования.

владеть:

- как минимум одним из существующих наиболее распространенных языков функционального программирования: F#, Haskell, LISP/Scheme, OCaml, Erlang.

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

4.1. Разделы дисциплины (модуля) и трудоемкости по видам учебных занятий

№	Тема (раздел) дисциплины	Трудоемкость по видам учебных занятий, включая самостоятельную работу, час.			
		Лекции	Семинары	Лаборат. работы	Самост. работа
1	Парадигмы программирования. Примеры функционального подхода к программированию. Использование функционального подхода в современной программной инженерии.	4	7		5
2	Аппликативная модель вычислений. Исчисление и комбинаторная логика.	4	6		10
3	Исчисление как язык программирования.	2			5
4	Типизация в языках функционального программирования.	4	6		5
5	Рекурсия и рекурсивные структуры данных.	2	3		10
6	Языки функционального программирования.	4			5
7	Анализ естественных и искусственных языков.	2			10
8	Операции над функциональными программами. Доказательство программ. Семантика языков функционального программирования.	4	4		5
9	Современные направления развития функционального программирования. Функциональное программирование в промышленном масштабе.	2			10
10	Объектное и объектно-ориентированное программирование.	2	4		10
Итого часов		30	30		75
Подготовка к экзамену		0 час.			
Общая трудоёмкость		135 час., 3 зач.ед.			

4.2. Содержание дисциплины (модуля), структурированное по темам (разделам)

Семестр: 6 (Весенний)

1. Парадигмы программирования. Примеры функционального подхода к программированию. Использование функционального подхода в современной программной инженерии.

Понятие о программировании и алгоритмической модели. История языков программирования. Традиционный подход к программированию машин с архитектурой фон Неймана. Языки низкого и высокого уровня. Функциональная, объектно-ориентированная абстракции и абстракция данных при декомпозиции задач. Основные парадигмы программирования: императивная, декларативная, аппликативная и др. Пример решения задач на языке функционального программирования (факториал, сумма натуральных чисел от 1 до N, преобразование изображений, построение графического изображения множества Мандельброта, построение трехмерного графика функции). Другие парадигмы программирования. Мультипарадигмальные языки (Oz, Mercury). Примеры программных систем, разработанных на функциональных языках (Emacs, HeVeA, ...). Функциональное программирование в современных промышленных языках (C# 3.0, F#). Пример программирования в функциональном стиле. Особенности функционального стиля программирования (отсутствие побочных эффектов, функции как данные, immutability и др.).

2. Аппликативная модель вычислений. Исчисление и комбинаторная логика.

Основные идеи и нотация λ -исчисления. Каррирование. Парадокс Рассела. λ -исчисление как формальная система. Синтаксис. Свободные и связанные переменные. Правила подстановки и λ -конверсия. Экстенциональность. Редукция и стратегии редукции. Жадные и ленивые вычисления. Мемоизация. Теорема Черча-Россера. Комбинаторная логика и комбинаторы. Чистое λ -исчисление и прикладные теории. Пополнение семантического базиса для реализации различных информационных объектов.

3. Исчисление как язык программирования.

От формальной системы к языку программирования. Язык FP Дж.Бэкурса. Основы синтаксиса функционального языка семейства ML: запись λ -выражений, пары и n-ки, логические значения, натуральные числа. Представление данных в λ -исчислении. Теорема о полноте по Тьюрингу. let-выражения. λ -исчисление как декларативный язык. Отображения и функционалы. Контекст вычислений и лексическое замыкание. Использование замыканий для реализации потоков и ленивых (отложенных) вычислений. Вычисления с бесконечными списками.

4. Типизация в языках функционального программирования.

Понятие о бестиповых языках и языках со слабой и строгой типизацией. Статическая и динамическая типизация. Понятие типа данных в императивных языках и формальных аксиоматических системах. Реализация динамической типизации в бестиповых языках. Типизированное λ -исчисление. Типизация по Черчу и по Карри. Полиморфизм. Теорема о сохранении типов. let-полиморфизм. Конструкторы типов. Наиболее общий тип и алгоритм Милнера. Система типов Хиндли-Милнера. Вывод типов. Типизированная комбинаторная логика.

5. Рекурсия и рекурсивные структуры данных.

Рекурсивные функции. Комбинатор неподвижной точки. Определение рекурсивных структур данных. Операция сопоставления с образцом. Списки и основные операции со списками: определение длины, принадлежность элемента списку, конкатенации двух списков, удаления элемента из списка, перестановки, определение подсписка. Пример: алгоритмы сортировки. Деревья. Пример: сортировка списка при помощи упорядоченного дерева. Использование функциональной абстракции и функционалов высших порядков для унификации функций обработки данных: итераторы и другие списковые комбинаторы. Хвостовая рекурсия и аккумуляторы.

6. Языки функционального программирования.

Дополнительные возможности ML. Императивные элементы: исключения, ссылки и массивы, ввод-вывод. Диалекты ML (OCaml, SML, F#). Языки с ленивой стратегией вычислений Miranda и Haskell, их особенности и диалекты. Модули и монады. Классические функциональные языки: LISP и Scheme. Функциональное программирование на Python.

7. Анализ естественных и искусственных языков.

Лексический и синтаксический анализ. Грамматики. Контекстно-свободные грамматики. Поверхностные и глубинные структуры фраз, приведение их к канонической форме. Представление предложения в контекстно-свободной грамматике в виде дерева разбора. Разбор предложений в контекстно-свободной грамматике. Разбор предложений методом рекурсивного спуска. Использование расширенной сети переходов для представления более богатых грамматик. Разбор предложений на естественном языке. Подход к представлению и интерпретации сообщений на естественном языке. Инструментарий fslex/fsyacc для построения компиляторов.

8. Операции над функциональными программами. Доказательство программ. Семантика языков функционального программирования.

Функциональные программы как математические объекты. Построение расширяемого мета-транслятора языка функционального программирования. Алгоритмическая неразрешимость проблемы корректности. Тестирование и верификация, пределы верификации. Примеры доказательства корректности программ (возведение в степень, GCD, конкатенация списков). Понятие семантики языков программирования. Подходы к определению семантики: операционный, денотационный, пропозиционный. Денотационная семантика и теория вычислений Д.Скотта. Семантика абстрактных машин: SECD-машина, КАМ. Преобразование функциональных программ в инструкции КАМ. Код де Брейна.

9. Современные направления развития функционального программирования. Функциональное программирование в промышленном масштабе.

Особенности функционального программирования для разработки пользовательских интерфейсов, ориентированных на события систем, систем реального времени, систем с параллелизмом. Пример: определение функции параллельного агрегирования элементов списка на F#. Возможности современных систем функционального программирования (F#, SML, Haskell/Hugs, Mercury). Пример: 3D-визуализация с использованием F# и Managed DirectX. Функциональное программирование в современных императивных и объектно-ориентированных языках и средах: C# 2.0, C# 3.0 (функциональные типы и делегаты, лямбда-нотация, анонимные типы и вывод типов, унификация доступа к данным LINQ и др.). Построение систем на основе интероперабельности функциональных (F#) и императивных языков на платформе Microsoft .NET. Функциональный язык преобразования слабоструктурированных данных XSLT. Использование функционального подхода для быстрого прототипирования программных систем. Подход к отладке и тестированию функциональных программ.

10. Объектное и объектно-ориентированное программирование.

Основные концепции объектного подхода к программированию. Объектно-ориентированный анализ, моделирование и программирование. Объектная природа функциональных языков. Моделирование объектности на чистых функциональных языках. Объектные расширения функциональных языков: CLOS, Flavors, Objective Caml, F#, SML. Формализация объектных моделей.

5. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине (модулю)

Компьютерный класс, оснащенный персональными компьютерами с установленным программным обеспечением Visual Studio 2013, мультимедиапроектор и экран.

6. Перечень рекомендуемой литературы

Основная литература
не предусмотрено

Дополнительная литература
не предусмотрено

7. Перечень ресурсов информационно-телекоммуникационной сети "Интернет", необходимых для освоения дисциплины (модуля)

- Сошников Д.В. Видео-курс «Функциональное программирование» интернет-университета информационных технологий ИНТУИТ.РУ (<http://www.intuit.ru/department/pl/funcprog/>)
- Электронный журнал «Практика функционального программирования»: <http://fprog.ru>
- <http://blogs.msdn.com/dsyme>
- <http://www.codeplex.com/fsharpsamples>
- Функциональное программирование на C#: <http://kuklaora.blogspot.com/2007/03/c.html>
- LINQ как шаг к функциональному программированию:
<http://www.rsdn.ru/article/mag/200802/LinqAsStapToFp.xml>

1. Chris Okasaki, Purely Functional Data Structures (Ph.D. Thesis): <http://lib.mexmat.ru/books/12772>
2. E. Chailloux, P. Manoury, B. Pagano. Разработка программ с помощью Objective Caml. O'Reilly. Русский перевод: <http://shamil.free.fr/comp/ocaml/>

8. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень необходимого программного обеспечения и информационных справочных систем (при необходимости)

На лекционных занятиях используются мультимедийные технологии, включая демонстрацию презентаций и выполнение программного кода в среде программирования.

В процессе самостоятельной работы и семинарских работ обучающимися предполагается использование среды программирования на языке F#, например, Visual Studio 2013, Xamarin Studio, Monodevelop и др.

9. Методические указания для обучающихся по освоению дисциплины (модуля)

Успешное освоение курса требует напряжённой самостоятельной работы студента. В программе курса приведено минимально необходимое время для работы студента над темой. Для полного освоения курса необходимо выполнение практических работ в объеме, превосходящем минимально предусмотренный, в частности, выполнение индивидуального или группового практического задания по созданию достаточно сложной программной системы, например:

- 1) Поисковая система в интернет
- 2) Агрегатор интернет-новостей с классификацией по темам

- 3) Система управления роботом с помощью Kinect на основе реактивного функционального программирования
- 4) Реализация системы программирования на модельном функциональном языке

Литература для самостоятельного изучения:

1. J.Harrop, F# for Scientists, Wiley, 2008.
2. Thompson S. Haskell: The Craft of Functional Programming. 2-nd edition, Addison-Wesley, 1999.
3. D.Syme, A.Granicz, A.Cisternio. Expert F# 2.0. Apress, 2010.
4. C. Smith, Programming F#: A comprehensive guide for writing simple code to solve complex problems. O'Reilly, 2010.
5. T.Neward, A.Erickson, T.Crowell, R.Minerich. Professional F# 2.0. Wiley Publishing, 2011.
6. T.Petricek, J.Skeet. Real World Functional Programming: With Examples in F# and C#. Manning Publications, 2010.

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

по направлению: Информатика и вычислительная техника

профиль подготовки: Физтех-школа Прикладной Математики и Информатики
кафедра алгоритмов и технологий программирования

курс: 3

квалификация: бакалавр

Семестр, формы промежуточной аттестации: 6 (весенний) - Дифференцированный зачет

Разработчик: Д.В. Сошников, канд. физ.-мат. наук, доцент, доцент

1. Компетенции, формируемые в процессе изучения дисциплины

Код и наименование компетенции	Индикаторы достижения компетенции
УК-1 Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	УК-1.2 Находит, критически анализирует и выбирает информацию, необходимую для решения поставленной задачи
	УК-1.1 Анализирует задачу, выделяя этапы ее решения, действия по решению задачи
	УК-1.3 Рассматривает различные варианты решения задачи, оценивает их преимущества и недостатки
	УК-1.4 Грамотно, логично, аргументированно формирует собственные суждения и оценки
	УК-1.5 Определяет и оценивает практические последствия возможных вариантов решения задачи
ОПК-1 Способен применять фундаментальные знания, полученные в области физико-математических и (или) естественных наук, и использовать их в профессиональной деятельности	ОПК-1.1 Способен анализировать поставленную задачу, намечать пути ее решения
	ОПК-1.2 Способен строить математические модели, производить количественные расчеты и оценки
	ОПК-1.3 Способен определять границы применимости полученных результатов
ОПК-2 Способен использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности, соблюдая требования информационной безопасности	ОПК-2.1 Способен применять современные вычислительную технику и сервисы сети Интернет в области (сфере) профессиональной деятельности
	ОПК-2.2 Знает и умеет применять численные математические методы и прикладное программное обеспечение для решения научных задач в профессиональной области
	ОПК-2.3 Знает основные требования информационной безопасности
ОПК-3 Способен составлять и оформлять научные и (или) технические (технологические, инновационные) отчеты (публикации, проекты)	ОПК-3.1 Знает основные правила оформления научных публикаций и научно-технической документации, в том числе с использованием прикладного программного обеспечения
	ОПК-3.2 Владеет на практике методологией составления научно-технических отчетов (проектов)
	ОПК-3.3 Владеет методами визуального и графического представления результатов научной (научно-технической, инновационной технологической) деятельности в виде отчетов, научных публикаций
ПК-1 Способен ставить, формализовывать и решать задачи, в том числе разрабатывать и исследовать математические модели изучаемых явлений и процессов, системно анализировать научные проблемы, получать новые научные результаты	ПК-1.3 Способен применять теоретические и (или) экспериментальные методы исследований к конкретной научной задаче и интерпретировать полученные результаты
	ПК-1.2 Способен выдвигать гипотезы, строить математические модели для описания изучаемых явлений и процессов, оценить качество разработанной модели
	ПК-1.1 Способен находить, анализировать и обобщать информацию об актуальных результатах исследований в рамках тематической области своей профессиональной деятельности
ПК-2 Способен самостоятельно или в качестве члена (руководителя) малого коллектива организовывать и проводить научные исследования и их апробацию	ПК-2.1 Знает принципы построения научной работы, методы сбора и анализа полученного материала, способы аргументации
	ПК-2.2 Способен планировать и проводить научные исследования самостоятельно или в качестве члена (руководителя) малого научного коллектива

2. Показатели оценивания компетенций

В результате изучения дисциплины «Функциональное программирование» обучающийся должен:

знать:

- общие принципы функционального подхода к программированию, преимущества и недостатки функционального подхода для реализации программных систем;
- алгоритмические модели;
- исчисления и комбинаторной логики, лежащие в основе функционального программирования;
- инструментальные средства и основные языки функционального программирования;
- использование функционального стиля программирования и элементов функционального программирования в традиционных императивных языках (C++, C# 3.0), в языках трансформации XSLT и др.;
- подходы и средства к построению трансляторов с функциональных языков, на основе интерпретации и компиляции в код абстрактной машины;
- подходы к описанию семантики функциональных языков на основе денотационной семантики и операционной семантики абстрактной машины.

уметь:

- разрабатывать, кодировать, тестировать и отлаживать программы на языках функционального программирования или в функциональном стиле;
- использовать функции высших порядков, функции-как-данные и замыкания;
- использовать языки функционального программирования для реализации известных алгоритмов информатики;
- выделять характерные задачи для применения функционального подхода и предлагать способы их решения;
- использовать подходы и языки логического программирования при построении программных систем, в том числе совместно с традиционными системами программирования.

владеть:

- как минимум одним из существующих наиболее распространенных языков функционального программирования: F#, Haskell, LISP/Scheme, OCaml, Erlang.

3. Перечень типовых (примерных) вопросов, заданий, тем для подготовки к текущему контролю

Понятие о программировании и алгоритмической модели.

История языков программирования.

Традиционный подход к программированию машин с архитектурой фон Неймана.

Языки низкого и высокого уровня.

Функциональная, объектно-ориентированная абстракции и абстракция данных при декомпозиции задач.

Основные концепции объектного подхода к программированию.

Объектно-ориентированный анализ, моделирование и программирование.

Объектная природа функциональных языков.

Моделирование объектности на чистых функциональных языках.

4. Перечень типовых (примерных) вопросов и тем для проведения промежуточной аттестации обучающихся

Перечень контрольных вопросов:

1. Основные парадигмы программирования: императивная, декларативная, аппликативная и др. Понятие о математических моделях вычислений (λ -исчисление и рекурсивные функции, машины Тьюринга). Примеры реализации программ в различных парадигмах. Функциональный стиль программирования на императивных языках.

2. Неформальное определение функциональной парадигмы программирования. Основные отличия от императивных языков. Преимущества и недостатки функционального подхода в сравнении с императивным. Примеры. Мультипарадигмальные языки программирования.
3. Роль функционального и декларативного программирования в современной индустрии разработки ПО. Примеры языков, программных продуктов и проектов. Основные языки функционального программирования.
4. Функциональное программирование как способ борьбы с современными проблемами в области разработки ПО. Функциональная абстракция и декомпозиция. Параллельные вычисления в функциональной парадигме.
5. Основные конструкции языка F#: let, fun, function. Области видимости. Условный оператор. Сопоставление с образцом.
6. Рекурсия. Классификация рекурсии. Хвостовая рекурсия. Выделение функциональной абстракции циклов. Циклические конструкции в F#.
7. Рекурсивные структуры данных. Списки. Основные приемы обработки списков. Простейшие операции: принадлежность элемента списку, длина списка, конкатенация списков.
8. Функции высших порядков для работы со списками. map, filter, reduce, fold, find и др. Примеры описания.
9. Синтаксис порождения списка list comprehension и его сведение к функциям высших порядков. Приемы сведения функций над списками к хвостовой рекурсии. Примеры.
10. Порядковое представление списков. Представления матриц (списочные, покоординатные, встроенные алгебраические типы, разреженные матрицы). Массивы. Прямоугольные (regular) и непрямоугольные (jagged) многомерные массивы.
11. Деревья. Двоичные деревья и деревья общего вида. Сведение дерева общего вида к двоичному. Описание двоичных деревьев и деревьев общего вида на F#.
12. Понятие обхода дерева. Инфиксный, префиксный, постфиксный порядки обхода. Деревья выражений. Пример.
13. Деревья поиска. Вставка и удаление элемента из дерева поиска. Использование деревьев поиска для организации структур данных с прямым доступом. Сортировка списка с использованием дерева поиска.
14. Функциональные структуры данных. Реализация очереди. Zipper для списков и для деревьев.
15. Продолжения (continuations). Использование продолжений для сведения обхода дерева к хвостовой рекурсии.
16. Нормальный и аппликативный порядок редукции. Ленивые и энергичные вычисления. Эффект разделения. Вычисление с контекстом и мемоизация. Механизмы вызова и передачи параметров.
17. Контекст вычислений и замыкания (closures). Замыкания как объекты. Mutable-переменные. Частичное применение функций. Специализация. Проекция Футауры и суперкомпиляция.
18. Генераторы и отложенные вычисления. Пример реализации ленивой последовательности с использованием генератора.
19. Ленивые вычисления в F#. Lazy/Force. Реализация ленивой последовательности. Последовательности (sequence) в F#. Sequence comprehension.
20. Ленивые вычисления и мемоизация. Реализация мемоизации на F#.
21. Возможные подходы к реализации языков функционального программирования. Сравнение. Абстрактные синтаксические деревья. Пример описания абстрактного синтаксического дерева для простейшего языка функционального программирования.
22. Энергичный Eval/Apply-интерпретатор. Подход к построению ленивого интерпретатора. Реализация рекурсии в Eval/Apply-интерпретаторе.
23. SECD-машина Ландина. Пример вычисления выражения на SECD-машине. Правила перехода SECD-машины. Реализация рекурсии.
24. Редукция графов. Представление выражений в виде графов. Основные правила редукции. Примеры.
25. Другие подходы к реализации функциональных языков. Поточковые графы. Комбинаторные реализации. Примеры.
26. Метапрограммирование. Quotations. Примеры использования Quotations.
27. Императивное ядро в функциональном языке. Реализация ввода-вывода в чисто функциональных языках (Haskell). Монады. Монадические свойства. Пример простейшей монады вывода на F#.

28. Computational Expressions (Workflows) в F#. Монада недетерминированных вычислений и ее реализация в виде computational expression.
29. Реализация монады вычислений с продолжениями на F# в виде монадического выражения.
30. Параллельное и асинхронное программирование в F# с использованием Asynchronous Workflows.
31. Обзор языка программирования Haskell. Сравнение с энергичными функциональными языками (F#).
32. Стратегия вычислений map-reduce для обработки больших данных. Hadoop.
33. Реализация парсеров на основе комбинаторов на функциональных языках программирования.

Примеры экзаменационных заданий

Часть 1: Multiple Choice (10 баллов)

Выберите наиболее правильный вариант ответа на каждый из следующих вопросов:

1. Почему функциональные программы проще отлаживать, чем императивные?
 - a. Программный код получается короче
 - b. Нет побочных эффектов
 - c. Более удобный отладчик, основанный на поочередном вызове функций
2. Какой тип у функции сложения целых чисел, определённой как `let plus(x,y)=x+y`?
 - a. `int -> (int->int)`
 - b. `(int -> int) -> int`
 - c. `int * int -> int`
 - d. `int -> int`
3. Какой будет результат сопоставления `let x::y::z = [1;2]`?
 - a. Ошибка
 - b. `x=1, y=2, z=null`
 - c. `x=1, y=2, z=[]`
 - d. `x=[], y=1, z=2`
4. С помощью какой функции можно проще всего найти максимальный элемент целочисленного списка?
 - a. `iter`
 - b. `map`
 - c. `filter`
 - d. `fold`
 - e. ни одна из перечисленных
5. Какова сложность добавления элемента в начало списка длины `n`?
 - a. `O(1)`
 - b. `O(n)`
 - c. `O(n2)`
6. [2 балла] Какой тип будет у описанной ниже функции `f`? Что она делает?
`let f q x = List.map (fun z -> async { return q z }) x |> Async.Parallel |> Async.RunSynchronously`
7. [2 балла] Приведите описанную ниже функцию к хвостовой рекурсии с помощью продолжений:
`let rec fib = function`
 `| 0 | 1 -> 1`
 `| n -> fib(n-1)+fib(n-2)`
8. В функции из предыдущего примера избавьтесь от рекурсии с помощью комбинатора неподвижной точки, описанного на F#.

Часть 2: Эссе (3 балла)

Опишите (на обороте) все преимущества и недостатки использования функционального языка программирования F# для решения задач вычислительной математики (напр., численное решение уравнения в частных производных).

Часть 3 (Вариант 10): Практическое задание (15 баллов)

Реализуйте на языке F# наиболее эффективным и функционально-чистым способом:

- Функцию, которая переводит число в *r*-ичной системе счисления, заданное последовательностью цифр, в чисто типа `int` [4 балла]
- Функцию, которая разбивает последовательность на подпоследовательности в зависимости от предикатов начала и конца подпоследовательности [5 баллов]

- Функцию, возвращающую из исходной последовательности символов последовательность чисел в р-ичной системе счисления в виде значений типа int [3 балла]
- Функцию, возвращающую максимальное значение числа в р-ичной системе счисления, содержащегося в текстовом файле [3 балла]

Критерии оценивания

- оценка «отлично (10)» выставляется студенту, показавшему всесторонние, систематизированные, глубокие знания учебной программы дисциплины и умение уверенно применять их на практике при решении конкретных задач, свободное и правильное обоснование принятых решений
- оценка «отлично (9)» выставляется студенту, показавшему всесторонние, систематизированные, глубокие знания учебной программы дисциплины и умение применять их на практике при решении конкретных задач, свободное и правильное обоснование принятых решений
- оценка «отлично (8)» выставляется студенту, показавшему всесторонние, систематизированные, глубокие знания учебной программы дисциплины и умение применять их на практике при решении конкретных задач, и правильное обоснование принятых решений
- оценка «хорошо (7)» выставляется студенту, если он твердо знает материал, грамотно и по существу излагает его, умеет применять полученные знания на практике, но допускает в ответе или в решении задач некоторые неточности;
- оценка «хорошо (6)» выставляется студенту, если он знает материал, грамотно и по существу излагает его, умеет применять полученные знания на практике, но допускает в ответе или в решении задач некоторые неточности;
- оценка «хорошо (5)» выставляется студенту, если он знает материал, и по существу излагает его, умеет применять полученные знания на практике, но допускает в ответе или в решении задач некоторые неточности;
- оценка «удовлетворительно (4)» выставляется студенту, показавшему фрагментарный, разрозненный характер знаний, недостаточно правильные формулировки базовых понятий, нарушения логической последовательности в изложении программного материала, но при этом он владеет основными разделами учебной программы, необходимыми для дальнейшего обучения и может применять полученные знания по образцу в стандартной ситуации;
- оценка «удовлетворительно (3)» выставляется студенту, показавшему фрагментарный, разрозненный характер знаний, недостаточно правильные формулировки базовых понятий, нарушения логической последовательности в изложении программного материала, но при этом он владеет фрагментарно основными разделами учебной программы, необходимыми для дальнейшего обучения и может применять полученные знания по образцу в стандартной ситуации;
- оценка «неудовлетворительно (2)» выставляется студенту, который не знает большей части основного содержания учебной программы дисциплины, допускает грубые ошибки в формулировках основных понятий дисциплины и не умеет использовать полученные знания при решении типовых практических задач
- оценка «неудовлетворительно (1)» выставляется студенту, который не знает формулировок основных понятий дисциплины.

5. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности

Дифференцированный зачет проводится в три этапа, каждый из которых оценивается в баллах:

- Тестирование (10 вопросов) по принципу multiple choice (с закрытым ответом) или вопросов с кратким открытым ответом (письменно)
- Мини-эссе на заданную тему (письменно)
- Выполнение серии заданий увеличивающейся сложности на ЭВМ

Во время проведения дифференцированного зачета, обучающиеся могут пользоваться программой дисциплины.