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

УТВЕРЖДЕНО
Директор высшей школы программной инженерии
А.В. Малеев

Рабочая программа дисциплины (модуля)

по дисциплине: Функциональное программирование
по направлению: Программная инженерия
профиль подготовки: Разработка программно-информационных систем
высшая школа программной инженерии
высшая школа программной инженерии МФТИ - Яндекс
курс: 3
квалификация: бакалавр

Семестр, формы промежуточной аттестации: 5 (осенний) - Дифференцированный зачет

Аудиторных часов: 60 всего, в том числе:
лекции: 30 час.
семинары: 30 час.
лабораторные занятия: 0 час.

Самостоятельная работа: 48 час.

Всего часов: 108, всего зач. ед.: 3

Программу составил: А.В. Созыкин, канд. техн. наук, доцент

Программа обсуждена на заседании высшей школы программной инженерии МФТИ - Яндекс 17.03.2023

Аннотация

Курс посвящен знакомству с функциональной парадигмой программирования в общем и с языком программирования Haskell в частности. Haskell является языком с высокой теоретической нагрузкой, что заставляет обучающихся активно использовать абстрактное мышление с одной стороны, и не забывать основы – с другой. Здесь находят приложение идеи, освещаемые в курсе математической логики, что способствует укреплению изучаемого параллельно теоретического материала.

1. Цели и задачи

Цель дисциплины

- знакомство студентов с основами и методами функционального программирования и выработки практических навыков применения этих знаний.

Задачи дисциплины

- изложение основных принципов функционального программирования, их основных применений в современном программировании;
- предоставление студенту ориентиров для дальнейшего самостоятельного изучения отдельных вопросов в специализированных разделах математической логики и функционального программирования.

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-2 Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений	УК-2.1 Формулирует совокупность взаимосвязанных задач в рамках поставленной цели работы, обеспечивающих ее достижение. Определяет ожидаемые результаты решения поставленных задач
	УК-2.2 Проектирует решение конкретной задачи проекта, выбирая оптимальный способ ее решения, исходя из действующих правовых норм и имеющихся ресурсов и ограничений
ОПК-5 Способен инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем	ОПК-5.1 Владеет на практике методологией составления научно-технических отчетов (проектов)
ОПК-6 Способен разрабатывать алгоритмы и программы, пригодные для практического использования, применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов	ОПК-6.4 Имеет навыки программирования и тестирования программных продуктов
ОПК-7 Способен применять в практической деятельности основные концепции, принципы, теории и факты, связанные с информатикой	ОПК-7.1 Обладает навыками создания и выполнения тестовых сценариев для выявления ошибок в программном обеспечении
	ОПК-7.3 Умеет выявлять узкие места в процессе разработки и предлагать методы и инструменты для его оптимизации
ПК-2 Способен формализовать и алгоритмизировать поставленную задачу	ПК-2.1 Способен формализовать и алгоритмизировать поставленную задачу
	ПК-2.2 Владеет методами и приемами формализации и алгоритмизации задач

ПК-3 Способен проектировать, разрабатывать, интегрировать, проверять на работоспособность программное обеспечение	ПК-3.3 Умеет излагать основные принципы построения и виды архитектуры программного обеспечения, методы и средства проектирования программного обеспечения, методологии разработки программного обеспечения и технологии программирования
	ПК-3.2 Умеет выбирать языки программирования для написания программного кода с учетом технического задания

3. Перечень планируемых результатов обучения по дисциплине (модулю)

В результате освоения дисциплины обучающиеся должны

знать:

- роль функционального программирования в решениизадач искусственного интеллекта;
- существующий наборы инструментальных средств функционального программирования, а также тенденции и перспективы их развития;
- теория и практика лямбда – исчислений.

уметь:

- разрабатывать программные приложения для решения поставленных задач на функциональном языке программирования;
- разрабатывать алгоритмы решения задач для функционального программирования.

владеть:

- Актуальными знаниями в области функционального программирования;
- Знаниями основ лямбда – исчислений;
- Навыками по применению лямбда - исчисления как языка программирования;
- Навыками в основах объектно-ориентированного программирования в функциональном программировании.

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	Лямбда-исчисление	2	2		4
2	Рекурсия. Редукция	4	4		4
3	Просто типизированное лямбда-исчисление	2	2		4
4	Введение в Haskell	2	2		4
5	Программирование на языке Haskell	4	4		4
6	Аппликативные функторы и свёртки	2	2		5
7	Монады	6	6		5
8	Вывод типов	4	4		6
9	Параметричность	2	2		6
10	Чисто функциональные структуры данных	2	2		6
Итого часов		30	30		48
Подготовка к экзамену		0 час.			
Общая трудоёмкость		108 час., 3 зач.ед.			

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

Семестр: 5 (Осенний)

1. Лямбда-исчисление

Лямбда-исчисление. Функциональное vs императивное программирование. Введение в λ-исчисление. Подстановка и преобразования. Расширения чистого λ-исчисления.

2. Рекурсия. Редукция

Рекурсия. Редукция: Теорема о неподвижной точке. Редексы и нормальная форма. Теорема Чёрча-Россера. Стратегии редукции.

3. Просто типизированное лямбда-исчисление

Просто типизированное лямбда-исчисление: Понятие типа. Просто типизированное λ-исчисление. Формализм систем λ→. Свойства λ→.

4. Введение в Haskell

Введение в Haskell: Язык Haskell. Основы программирования. Базовые типы. Система модулей. Операторы и сечения.

5. Программирование на языке Haskell

Программирование на языке Haskell: Ленивость и строгость. Алгебраические типы данных и сопоставление с образцом. Списки и работа с ними.

Система типов Haskell: Виды полиморфизма. Классы типов. Стандартные классы типов. Реализация классов типов.

6. Аппликативные функторы и свёртки

Аппликативные функторы и свёртки: Функторы. Класс типов Pointed. Аппликативные функторы. Класс типов Traversable. Свёртки. Моноиды. Класс типов Foldable. Свойство слияния для foldr.

7. Монады

Монады:Класс типов Monad. Монада Maybe. Список как монада.

Использование монад: Класс типов Monad. Монада Maybe. Список как монада.

Трансформеры монад:Моноиды, Alternative, MonadPlus. Мультипараметрические классы типов. Монады с обработкой ошибок. Трансформеры монад.

8. Вывод типов

Вывод типов: Главный тип. Подстановка типа и унификация. Теорема Хиндли-Милнера.

Полиморфные системы типов: Сильный и слабый полиморфизм. Let-полиморфизм. Полиморфизм высших рангов. Универсальные абстракция и применение. Импредикативность. Сильная нормализация. Программирование в полиморфных системах. Система с зависимыми типами. Семейства типов. Виды для семейств типов. Тип зависимого произведения.

9. Параметричность

Параметричность: Параметричность как свойство полиморфных систем. Теорема Рейнольдса. Свободные теоремы для полиморфных типов.

10. Чисто функциональные структуры данных

Чисто функциональные структуры данных: Зипперы. Алгебра и анализ зипперов. Линзы и призмы. Пользовательские линзы.

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

- проектор с возможностью подключения через HDMI и/или VGA);
- доска с мелом или whiteboard с фломастерами;
- компьютерный класс, оснащенные ПЭВМ.

6.Перечень рекомендуемой литературы

Основная литература

1. Введение в программирование , Электрон. версия печ. публикации / И. Ю. Баженова, В. А. Сухомлин. — Москва, ИНТУИТ, 2016

2. Паттерны проектирования, [учеб. пособие для вузов] / Эр. Фримен, Эл. Фримен . — Санкт-Петербург, Питер, 2017.— URL: https://ibooks.ru/bookshelf/354827/reading (дата обращения: 26.11.2020). - Полный текст (Режим доступа : из сети МФТИ / Удаленный доступ)

3. Функциональное программирование [Текст] : [учеб. пособие для вузов] / А. Филд, П. Харрисон ; пер. с англ. М. В. Горбатовой [и др.] ; под ред. В. А. Горбатова .— М. : Мир, 1993 .— 638 с.

Рекомендуемая литература д ля самостоятельного изучения:

- Barendregt, H., Dekkers, W., Statman, R. Lambda calculus with types. – Cambridge University Press, 2013.

Дополнительная литература

-

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

1. SimonPeytonJones (editor), Haskell 98 LanguageandLibraries (TheRevisedReport)
2. JohnHughes, IntroductiontoProgramminginHaskell, www.cs.chalmers.se/~rjmh
3. PaulHudak, JohnPeterson, Joseph H. Fasel , A GentleIntroductiontoHaskell 98
4. CordeliaHall, JohnHugs, ThelittleHaskeller
5. PhilipWadler, Monadsforfunctionalprogramming, DepartmentofComputingScience, UniversityofGlasgow
6. AllAboutMonads , http://www.nomaware.com/monads/html/
7. EmeryBerger , FP + OOP = Haskell, DepartmentofComputerScience, TheUniversityofTexasatAustin
8. RexPage, TwoDozenShortLessonsinHaskell, a participatorytextbookonfunctionalprogramming, SchoolofComputerScience, UniversityofOklahoma
9. DamirMedak, GerhardNavratil, Haskell-Tutorial, InstituteforGeoinformationTechnicalUniversityVienna
10. Главная страница языка Haskell, http://haskell.org.

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

не требуется

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

Студент, изучающий дисциплину, должен с одной стороны, овладеть общим понятийным аппаратом, а с другой стороны, должен научиться применять теоретические знания на практике.

В результате изучения дисциплины студент должен знать основные определения, понятия, аксиомы.

Успешное освоение курса требует напряжённой самостоятельной работы студента. В программе курса приведено минимально необходимое время для работы студента над темой. Самостоятельная работа включает в себя:

- чтение и конспектирование рекомендованной литературы;
- проработку учебного материала (по конспектам лекций, учебной и научной литературе), подготовку ответов на вопросы, предназначенных для самостоятельного изучения, доказательство отдельных утверждений, свойств;
- подготовку к дифференциальному зачету.

Руководство и контроль за самостоятельной работой студента осуществляется в форме индивидуальных консультаций.

Важно добиться понимания изучаемого материала, а не механического его запоминания. При затруднении изучения отдельных тем, вопросов, следует обращаться за консультациями к лектору.

ПРИЛОЖЕНИЕ

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

по направлению: Программная инженерия
профиль подготовки: Разработка программно-информационных систем
высшая школа программной инженерии МФТИ - Яндекс
высшая школа программной инженерии
курс: 3
квалификация: бакалавр

Семестр, формы промежуточной аттестации: 5 (осенний) - Дифференцированный зачет

Разработчик: А.В. Созыкин, канд. техн. наук, 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-2 Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений	УК-2.1 Формулирует совокупность взаимосвязанных задач в рамках поставленной цели работы, обеспечивающих ее достижение. Определяет ожидаемые результаты решения поставленных задач
	УК-2.2 Проектирует решение конкретной задачи проекта, выбирая оптимальный способ ее решения, исходя из действующих правовых норм и имеющихся ресурсов и ограничений
ОПК-5 Способен инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем	ОПК-5.1 Владеет на практике методологией составления научно-технических отчетов (проектов)
ОПК-6 Способен разрабатывать алгоритмы и программы, пригодные для практического использования, применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов	ОПК-6.4 Имеет навыки программирования и тестирования программных продуктов
ОПК-7 Способен применять в практической деятельности основные концепции, принципы, теории и факты, связанные с информатикой	ОПК-7.1 Обладает навыками создания и выполнения тестовых сценариев для выявления ошибок в программном обеспечении
	ОПК-7.3 Умеет выявлять узкие места в процессе разработки и предлагать методы и инструменты для его оптимизации
ПК-2 Способен формализовать и алгоритмизировать поставленную задачу	ПК-2.1 Способен формализовать и алгоритмизировать поставленную задачу
	ПК-2.2 Владеет методами и приемами формализации и алгоритмизации задач
ПК-3 Способен проектировать, разрабатывать, интегрировать, проверять на работоспособность программное обеспечение	ПК-3.3 Умеет излагать основные принципы построения и виды архитектуры программного обеспечения, методы и средства проектирования программного обеспечения, методологии разработки программного обеспечения и технологии программирования
	ПК-3.2 Умеет выбирать языки программирования для написания программного кода с учетом технического задания

2. Показатели оценивания компетенций

В результате изучения дисциплины «Функциональное программирование» обучающийся должен:

знать:

- роль функционального программирования в решениизадач искусственного интеллекта;
- существующий наборы инструментальных средств функционального программирования, а также тенденции и перспективы их развития;
- теория и практика лямбда – исчислений.

уметь:

- разрабатывать программные приложения для решения поставленных задач на функциональном языке программирования;
- разрабатывать алгоритмы решения задач для функционального программирования.

владеть:

- Актуальными знаниями в области функционального программирования;
- Знаниями основ лямбда – исчислений;
- Навыками по применению лямбда - исчисления как языка программирования;
- Навыками в основах объектно-ориентированного программирования в функциональном программировании.

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

С целью контроля освоения обучающимися учебного материала проводится устный опрос в начале занятия по теме прошлой лекции или в конце занятия по пройденной теме.

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

1) Комбинаторы. Функции нескольких переменных, каррирование. Подстановка, лемма подстановки.
2) Бета-преобразование. Эта-преобразование. Расширение чистого лямбда-исчисления: дельта-преобразование.
3) Теорема о неподвижной точке, Y-комбинатор. Редексы. Одношаговая и многошаговая редукция. Нормальная форма. Редукционные графы.
4) Теорема Чёрча-Россера. Следствия: редуцируемость к нормальной форме, единственность нормальной формы.
5) Стратегии редукции. Теорема о нормализации. Механизмы вызова в функциональных языках.
6) Роль типов в языках программирования. Предтермы. Утверждения о типизации. Контексты.
7) Правила типизации по Карри и по Чёрчу. Деревья вывода типов. Свойства типизированного лямбда-исчисления. Связь между системами Карри и Чёрча.
8) Проблемы разрешимости. Сильная и слабая нормализация. Соответствие Карри-Говарда.
9) Стандарт языка. Компилятор GHC. Интерпретатор GHCi. Hoogle. Связывание. Кодирование функций. Рекурсия.
10) Основные синтаксические конструкции. Система модулей.
11) Частичное применение, каррирование. Инфиксные операторы, сечения. Бесточечный стиль.
12) Ошибки. Основание. Строгие и нестрогие функции. Ленивое и энергичное исполнение. Функция seq.
13) Алгебраические типы данных. Сопоставление с образцом, его семантика.
14) Объявления type и newtype. Метки полей.
15) Параметрический и специальный полиморфизм. Классы типов. Объявления представителей (instance declaration). Пример: классы Eq и Ord.
16) Операторы над типами как параметры в определении класса. Класс типов Functor.
17) Реализация классов типов.
18) Законы для функторов. Класс типов Applicative и его представители. Два способа объявления списка аппликативным функтором. Класс Alternative.
19) Моноиды. Представители класса типов Monoid.
20) Свёртки списков. Правая и левая свёртки. Ленивые и энергичные версии свёрток.
21) Класс типов Foldable.
22) Монады. Класс типов Monad. Пример: монада Identity.
23) Законы класса типов Monad. do-нотация.
24) Монада Maybe: вычисления, которые могут завершиться неудачей.
25) Монада списка: вычисления со множественными результатами.
26) Ввод-вывод в чистых языках. Монада IO. Взаимодействие с файловой системой. Монады для записи в лог, чтения из внешнего окружения и работы с изменяемым состоянием: Reader, Writer, State.
27) Класс Alternative. Парсеры регулярных выражений. Класс MonadPlus.
28) Монада Error: вычисление, которое может вызвать исключение.
29) Монада Cont: управление порядком вычислений.
30) Проблема комбинирования монадических эффектов. Трансформеры монад.

31) Вывод типов. Главный (наиболее общий) тип. Свойства подстановки типов.

32) Композиция подстановок.

33) Унификатор, теорема унификации. Главная пара. Алгоритм Хиндли-Милнера.

34) Сильный и слабый полиморфизм. Let-полиморфизм. Полиморфизм высших рангов. Универсальные абстракция и применение. Импредикативность. Сильная нормализация. Программирование в полиморфных системах.

35) Система с зависимыми типами. Семейства типов. Виды для семейств типов. Тип зависимого произведения.

36) Параметричность как свойство полиморфных систем. Теорема Рейнольдса. Свободные теоремы для полиморфных типов.

37) Неизменяемые структуры данных и эффективные алгоритмы для них. Роль ленивости.

38) Аммортизированная сложность. Зипперы. Список с произвольным доступом (flexible array).

Примеры билетов:

Билет №1

1. Теорема о неподвижной точке, Y-комбинатор. Редексы. Одношаговая и многошаговая редукция.

2. Неизменяемые структуры данных и эффективные алгоритмы для них.

Билет №2

1. Комбинаторы. Функции нескольких переменных, каррирование. Подстановка, лемма подстановки.

2. Параметричность как свойство полиморфных систем. Теорема Рейнольдса. Свободные теоремы для полиморфных типов.

Критерии оценивания

Во время семестра студенты выполняют достаточно практических заданий. За сделанное задание они получают от 1 до 10 баллов, за несделанное - -10 баллов. Студенты с отрицательным результатом не допускаются к зачету 20% обладателей топовых результатов получают «автомат» - 10 баллов. Финальный балл формируется из двух частей. За само задание студент может получить от 0 до 5 баллов. И за свою работу в течение семестра он получает от 0 до 5 баллов по формуле (((сумма баллов/максимальное число баллов) + 0.2) x 5.).

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

Время проведения дифференцированного зачета составляет 2 академических часа.

Во время проведения дифференцированного зачета обучающиеся могут пользоваться программой дисциплины и исходными текстами.