Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Курганский государственный университет»
(КГУ)

Кафедра «Программное обеспечение автоматизированных систем»

УТВЕРЖДАЮ:

Ректор
_____ / Н.В. Дубив/

«31» августа 2020 г.

.

Рабочая программа учебной дисциплины

ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

образовательной программы высшего образования –
программы бакалавриата

09.03.03 – Прикладная информатика
Направленность:
Интеллектуальные информационные системы и технологии

Форма обучения: очная

Курган 2020

Рабочая программа дисциплины «Функциональное программирование» составлена в соответствии с учебными планами по программе бакалавриата «Прикладная информатика» (Интеллектуальные информационные системы и технологии) утвержденными для очной формы обучения «28» августа 2020 года.

Рабочая программа дисциплины одобрена на заседании кафедры «Программное обеспечение автоматизированных систем» «28» августа 2020 года, протокол № 1.

Рабочую программу составили:

Доцент кафедры, к.т.н. Н.В. Агапова

Согласовано:

Заведующий кафедрой
«Программное обеспечение
автоматизированных систем»
к.т.н., доцент Т.Р. Змызгова

Специалист по учебно-методической
работе Учебно-методического отдела Г.В. Казанкова

Начальник Управления образовательной
деятельности С.Н. Синицын

1. ОБЪЕМ ДИСЦИПЛИНЫ

Всего: 3 зачетных единиц трудоемкости (108 академических часов)

Очная форма обучения

Вид учебной работы	На всю дисциплину	Семестр
		6
Аудиторные занятия (контактная работа с преподавателем), всего часов в том числе:	**48**	**48**
Лекции	16	16
Лабораторные работы	32	32
Аудиторные занятия в интерактивной форме, часов	-	-
Самостоятельная работа, всего часов в том числе:	**60**	**60**
Подготовка к зачету	18	18
Другие виды самостоятельной работы	24	24
Контрольная работа	18	18
Вид промежуточной аттестации	**зачет**	**зачет**
Общая трудоемкость дисциплины и трудоемкость по семестрам, часов	**108**	**108**

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Дисциплина «Функциональное программирование» относится к базовой части, модуль блока 1 «Программирование».

Программа составлена с учетом межпредметных связей с учебными дисциплинами. Основой для изучения учебной дисциплины являются следующие учебные дисциплины первой ступени высшего образования: «Информатика», «Основы программирования», «Дискретная математика» и «Математическая логика», «Алгоритмы и структуры данных».

Учебная дисциплина «Функциональное программирование» знакомит студентов с парадигмой функционального программирования (ФП) в общем виде, её особенностями, выгодно отличающими её от императивной и объектно-ориентированной парадигм, языком Haskell как наиболее развитыми современным языком ФП, связанной с ФП парадигмой реактивного программирования, а также с математическими основаниями ФП: λ-исчислением и теорией категорий.

Результаты обучения по дисциплине необходимы для изучения дисциплин: «Архитектура ЭВМ», «Теория систем и системный анализ», «Администрирование программных систем», «Технологии параллельного программирования» и выполнения выпускной квалификационной работы.

3. ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ

Цель преподавания учебной дисциплины – овладение обучающимися бурно развивающимся и набирающим популярность стилем программирования, подготовка к его практическому использованию.

Особое внимание уделено общезначимым, независимым от конкретного языка, принципам, методам и понятиям ФП, что позволит обучающимся применять полученные знания при работе на множестве различных языков.

Задачи учебной дисциплины:

– дать обучающимся основу, необходимую для успешного усвоения перспективных технологий и методов программирования;

– дать обучающимся более глубокую и полную картину программирования за счёт альтернативной точки зрения на программы и процессы их выполнения;

– приобретенные знания позволяют понять современные тенденции в языках программирования, не относящихся к ФП (например, популярность Linq в платформе .Net, введение лямбда-выражений в язык C++ и т. д.).

Компетенции, формируемые в результате освоения дисциплины:

- способность использовать современные информационные технологии и программные средства, в том числе отечественного производства, при решении задач профессиональной деятельности (ОПК-2);

- способность разрабатывать алгоритмы и программы, пригодные для практического применения (ОПК-7);

В результате изучения дисциплины обучающийся должен

Знать

- главные характеристики парадигмы ФП, отличающие её от других распространённых парадигм (для ОПК 2);
- карринг (преобразование функции многих аргументов в функцию высшего порядка) (для ОПК 2);
- ленивый порядок вычислений (для ОПК 2);
- рекурсивный подход к определению алгоритмов, особенности хвостовой рекурсии (для ОПК 2);
- операции свёртки и развёртки, их применение для решения прикладных задач (для ОПК 2);
- алгебраические типы данных, бесконечные структуры данных (для ОПК 2);
- понятие класса типов; классы типов в стандартной библиотеке языка Haskell (для ОПК 2);
- программирование в терминах функторов и аппликативных функторов (для ОПК 2);
- программирование с использованием монад (для ОПК 2);
- основы λ-исчисления и теории категорий, связанные с ФП (для ОПК2).

Уметь:

- осуществлять разработку программного продукта по заданной спецификации на языке ФП (для ОПК 7);
- преобразовывать постановку задачи к виду, удобному для реализации в функциональном стиле (для ОПК 7);
- производить декомпозицию задач в терминах функций, свёрток, отображений, функторов, монад и других понятий ФП (для ОПК 7);
- находить в стандартной библиотеке и сторонних модулях функции, пригодные для повторного использования в своих задачах (для ОПК 7).

Владеть:

- компилятором GHC, диалоговым интерпретатором GHCi (для ОПК 2);
- системами управления проектами cabal и stack (для ОПК 2);
- репозиторием пакетов hackage (для ОПК 7);
- системой поиска функций по сигнатуре hoogle (для ОПК 2).

4. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

4.1. Учебно-тематический план

Очная форма обучения

№	Наименование раздела	Количество часов контактной работы с преподавателем	
		Лекции	Лабораторные работы
1	Введение. Элементарные основы ФП	2	4
2	Синтаксис и идиомы языка	4	8
3	Элементы средней сложности	2	4
	Рубежный контроль №1		2
4	Классы типов и экземпляры классов	6	8
5	Анализ данных	2	4
	Рубежный контроль №2		2
	Всего:	**16**	**32**

4.2. Содержание лекционных занятий

Тема 1. Введение. Элементарные основы ФП

Понятие парадигмы программирования. Принципы функциональной парадигмы. Обзор истории функциональных языков. Современное состояние ФП.

Базовые синтаксические конструкции и семантические элементы языка Haskell: определение значения и функции, применение функции к аргументам. Начальные сведения о сопоставлении с образцом. Понятие о чистой функции. Простейшие средства ввода-вывода.

Тема 2. Синтаксис и идиомы языка

Списки и функции над ними. Сопоставление аргумента-списка с образцом. Конкатенация. Списки кортежей. Бесконечные списки.
Система типов. Базовые типы. Простейшие конструкторы типов: кортеж и список. Типы функций. Переменные-типы. Понятие о классе типов. Начальные сведения об автоматическом выводе типов. Рекурсия. Понятие хвостовой рекурсии и её преимущества перед рекурсией общего вида.

Тема 3. Элементы средней сложности

Функции высшего порядка. Карринг. Частичное применение функции к аргументам. Сечение бинарной операции. Сопоставление с образцом. Выражения if, case, охраняемые выражения. Лямбда-функции. Композиция функций. Импорт модулей. Обзор часто используемых стандартных модулей. Создание собственных модулей

Тема 4. Классы типов и экземпляры классов

Типы данных. Конструкторы значений. Альтернативы. Синтаксис записей. Параметры типов. Конструкторы типов. Рекурсивные типы. Синонимы типов. Включение своего типа данных в имеющийся класс. Создание собственных классов типов. Функторы. Тип функции как функтор. Применение завёрнутой функции к завёрнутому аргументу. Аппликативные функторы. Монады. Законы монад

Тема 5. Анализ данных

Аппликативные функторы. Определение аппликативного функтора. Представители класса типов Applicative. Примеры. Законы аппликативных функторов. Аппликативный парсер Parsec. Аппликативный парсер своими руками. Композиция на уровне типов.
Управление эффектами

Наименование и содержание лекции	Часов контактной работы с преподавателем
Элементарные основы ФП	
Лекция 1. Введение. Элементарные основы ФП Понятие парадигмы программирования. Принципы функциональной парадигмы. Обзор истории функциональных языков. Современное состояние ФП. Базовые синтаксические конструкции и семантические элементы языка Haskell: определение значения и функции, применение функции к аргументам. Начальные сведения о сопоставлении с образцом. Понятие о чистой функции. Простейшие средства ввода-вывода.	2
Синтаксис и идиомы языка	
Лекция 2. Списки и функции над ними. Система типов Тип списка. Строки как списки символов. Генераторы списков. Создание функций над списками, сопоставление аргумента-списка с образцом. Функции head и tail, last и init, их реализация своими руками. Конкатенация. Функции elem, length, drop, sum, maximum. Функция fmap. Списки кортежей. Бесконечные списки. Базовые типы. Простейшие конструкторы типов: кортеж и список. Типы функций. Переменные-типы. Понятие о классе типов. Классы Eq, Ord, Show, Read, Enum, Bounded. Начальные сведения об автоматическом выводе типов.	3
Лекция 3. Рекурсия Создание рекурсивных реализаций для арифметических алгоритмов. Понятие хвостовой рекурсии и её преимущества перед рекурсией общего вида. Преобразование рекурсивных алгоритмов к хвостовому виду с помощью аккумулятора	1
Элементы средней сложности	
Лекция 4. Элементы средней сложности. Модули Функции высшего порядка. Карринг. Частичное применение функции к аргументам. Сечение бинарной операции. Выражения where и let. Сопоставление с образцом. Выражения if, case, охраняемые выражения. Лямбда-функции. Композиция функций. Импорт модулей. Обзор часто используемых стандартных модулей. Создание собственных модулей.	2

Наименование и содержание лекции	Часов контактной работы с преподавателем
Классы типов и экземпляры классов	
Лекция 5. Типы данных Создание собственных алгебраических типов данных. Конструкторы значений. Альтернативы. Синтаксис записей. Параметры типов. Конструкторы типов. Рекурсивные типы. Синонимы типов. Включение своего типа данных в имеющийся класс. Создание собственных классов типов.	2
Лекция 6. Функторы Список как пример функтора. Общее понятие о функторе. Трактовка функтора как контейнера, управляющего преобразованием своих элементов. Класс Functor. Законы функторов. Функторы Identity, Empty, Maybe, Either. Тип функции как функтор. Потребность в обобщении понятия функтора на функции многих аргументов. Понятие о применении завёрнутой функции к завёрнутому аргументу. Аппликативные функторы и класс Applicative. Примеры. Законы аппликативных функторов.	2
Лекция 7. Монады Ввод-вывод как пример монады, решение проблемы чистоты функций. Понятие о монаде в языке Haskell. Списки, функции, Maybe, Either, Reader, Writer, State как монады. Законы монад.	2
Анализ данных	
Лекция 8. Аппликативные функторы. Управление эффектами Определение аппликативного функтора. Представители класса типов Applicative. Примеры. Законы аппликативных функторов. Аппликативный парсер Parsec. Аппликативный парсер своими руками. Композиция на уровне типов. Класс типов Foldable. Класс типов Traversable. Законы и свойства класса Traversable. Связь классов Monad и Applicative. Классы типов Alternative и MonadPlus	2
Итого:	16

4.3. Лабораторные работы

Номер раздела, темы	Наименование раздела	Наименование практической работы	Норматив времени, час.
1	Элементарные основы ФП	Установка и настройка среды. Знакомство с компилятором ghc, диалоговым интерпретатором ghci и др. инструментарием. Функции. Операторы. Базовые типы. Рекурсия. Локальные связывания и правила отступов	4
2	Синтаксис и идиомы языка	Параметрический полиморфизм. Классы типов. Стандартные классы типов. Нестрогая семантика. Модули и компиляция	4
		Рекурсия: рекурсивные реализации для арифметических алгоритмов. Хвостовая рекурсия. Преобразование рекурсивных алгоритмов к хвостовому виду с помощью аккумулятора	2
		Функции для работы со списками. Функции высших порядков над списками. Генераторы списков. Правая свертка. Левая свертка и ее сравнение с правой. Родственные сверткам функции	2
3	Элементы средней сложности	Типы перечислений. Типы произведений и сумм произведений. Синтаксис записей. Типы с параметрами. Рекурсивные типы данных. Синонимы и обертки для типов	4
	Рубежный контроль №1		2

Номер раздела, темы	Наименование раздела	Наименование практической работы	Норматив времени, час.
4	Классы типов и экземпляры классов	Список как пример функтора. Функтор как контейнера. Законы функторов. Функторы Identity, Empty, Maybe, Either. Тип функции как функтор. Применение завёрнутой функции к завёрнутому аргументу. Аппликативные функторы и класс Applicative.	4
		Класс типов Functor и законы для него. Определение монады. Монада Identity. Список и Maybe как монады. Монада IO. Монада Reader. Монада Writer. Монада State	4
5	Анализ данных	Аппликативный функтор. Представители класса типов Applicative. Аппликативный парсер Parsec. Аппликативный парсер своими руками. Композиция на уровне типов. Классы типов Foldable и Traversable. Связь классов Monad и Applicative. Классы типов Alternative и MonadPlus	4
	Рубежный контроль №2		2
		Всего:	**32**

4.4 Контрольная работа

4.4.1 Назначение, цели и задачи контрольной работы

Контрольная работа выполняется обучающимися по вариантам заданий или по теме, предложенной обучающимся и согласованной с преподавателем.

Основная учебная цель: закрепление теоретических знаний, полученных в процессе изучения дисциплины и приобретение практических навыков по разработке программ на языке Haskell.

Основные задачи, решаемые обучающимися:

- проектирование и реализация программного приложения;
- оформление комплекта документации.

4.4.2 Требования к содержанию контрольной работы

Контрольная работа должна содержать визуальное приложение и комплект документации:

- опись альбома;

- пояснительная записка, содержащая разделы: введение, постановка задачи, описание исходных данных, описание алгоритма решения задачи, диаграмма классов (при наличии), описание структуры программного приложения, заключение, список использованных источников, содержание).

Варианты заданий

Разработать техническое задание и разработать программу:

1. Создать музыкальный секвенсор
2. Создать парсер
3. Создать простой компилятор
4. Обход и фильтрация дерева в Haskell
5. Рекурсивно найти всех детей в дереве. Сравнить производительность с программой на другом языке.
6. Интеграция Haskell и C
7. Целочисленное деление, в том числе и на отрицательные числа.
8. Задача разложения числа на степени двойки и подобные.
9. Игра «Путешествие во времени»
10. Напишите функцию, которая выдает список машин, уменьшая их цену на заданный процент.
11. Игра судоку, выводящая возможные решения.
12. Игра “крестики-нолики”
13. Игра в пятнашки
14. Игра «Жизнь»
15. Визуальная среда обучения программированию на Haskell
16. Поиск маршрутов в метро
17. Реализация алгоритмов дискретной математики
18. Реализация алгоритмов синтаксического разбора
19. Реализация алгоритмов криптографии на языке Haskell
20. Реализация на языке Haskell реактивного микросервиса

5. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Во время лекций по дисциплине обучающимся рекомендуется конспектировать теоретический материал, отмечая важные моменты, на которые заострил внимание преподаватель, участвовать в опросах и дискуссиях. Перед лекцией необходимо повторить выданный материал, зафиксировать непонятные места, чтобы обсудить их на занятии. Конспект лекций представлен в виде мультимедийных презентаций и включен в состав методического комплекса дисциплины.

Лабораторный практикум включает практические задания по четырем разделам дисциплины: «Элементарные основы ФП», «Синтаксис и идиомы языка», «Элементы средней сложности», «Классы типов и экземпляры классов», «Анализ данных». Все работы выполняются в соответствие с заданием, выданным преподавателем.

Преподавателем запланировано применение на лабораторных занятиях технологий развивающейся кооперации, коллективного взаимодействия, разбора конкретных ситуаций. Поэтому приветствуется групповой метод выполнения лабораторных работ и защиты отчетов, а также взаимооценка и обсуждение результатов выполнения лабораторных работ.

Для текущего контроля успеваемости по очной форме обучения преподавателем используется балльно-рейтинговая система контроля и оценки академической активности.

Выполнение самостоятельной работы подразумевает самостоятельное изучение разделов дисциплины, подготовку к лабораторным занятиям, к рубежным контролям (для обучающихся очной формы обучения), выполнение контрольной работы, подготовку к зачету.

Рекомендуемый режим самостоятельной работы

Наименование вида самостоятельной работы	**Рекомендуемая трудоемкость, акад. час.**
	Очная форма обучения
Самостоятельное изучение тем дисциплины:	**6**
Элементарные основы ФП	1
Синтаксис и идиомы языка	1
Элементы средней сложности	1
Классы типов и экземпляры классов	1
Анализ данных	2
Подготовка к лабораторным работам (по 2 ч. на каждое занятие)	**16**
Подготовка к рубежным контролям (по 1 часу на каждый рубеж)	**2**
Подготовка к контрольной работе	**18**
Подготовка к зачету	**18**
Всего:	**60**

6. ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ АТТЕСТАЦИИ ПО ДИСЦИПЛИНЕ

6.1. Перечень оценочных средств

1. Балльно-рейтинговая система контроля и оценки академической активности студентов в КГУ (для очной формы обучения)
2. Отчеты студентов по лабораторным работам
3. Тестовые задания
4. Банк заданий к рубежным контролям № 1, № 2 (для очной формы обучения).
5. Контрольная работа
6. Вопросы к зачету

6.2. Система балльно-рейтинговой оценки работы студентов по дисциплине

Очная форма обучения

<table>
<tr><td>№</td><td>Наименование</td><td colspan="7">Содержание</td></tr>
<tr><td rowspan="3">1</td><td rowspan="3">Распределение баллов за семестры по видам учебной работы, сроки сдачи учебной работы (доводятся до сведения обучающихся на первом учебном занятии)</td><td colspan="7">Распределение баллов,
6 семестр</td></tr>
<tr><td>Вид учебной работы:</td><td>Посещение лекций</td><td>Выполнение и защита результатов лабораторных работ</td><td>Выполнение и защита контрольной работы</td><td>Рубежный конкконтроль №1</td><td>Рубежный конкконтроль №2</td><td>Зачет</td></tr>
<tr><td>Балльная оценка:</td><td>До 8</td><td>44 б</td><td>8 б</td><td>5</td><td>5</td><td>30</td></tr>
<tr><td></td><td>Примечания:</td><td>8 лекций по 1 баллу</td><td>6 б. за 4часов, 4 б. за 2часов</td><td></td><td></td><td></td><td></td></tr>
<tr><td>2</td><td colspan="2">Критерий пересчета баллов в традиционную оценку по итогам работы в семестре и экзамена</td><td colspan="5">60 и менее баллов – незачтено;
61…73 – удовлетворительно; зачтено.
74… 90 – хорошо;
91…100 – отлично.</td></tr>
<tr><td>3</td><td colspan="2">Критерии допуска к промежуточной аттестации, возможности получения автоматического зачета (экзаменационной оценки) по дисциплине, возможность получения бонусных баллов</td><td colspan="5">Для допуска к промежуточной аттестации (зачету) обучающийся должен набрать не менее 50 баллов и выполнить все лабораторные работы (для очной) и контрольную работу.
Для получения «автоматической» оценки «зачтено» обучающемуся необходимо набрать 61 балл.
По согласованию с преподавателем обучающемуся могут быть добавлены дополнительные (бонусные) баллы за активность на консультациях, активное участие в научной и методической работе, оригинальность принятых решений в ходе выполнения лабораторных работ, за участие в значимых учебных и внеучебных мероприятиях кафедры.</td></tr>
</table>

4	Формы и виды учебной работы для неуспевающих (восстановившихся на курсе обучения) обучающихся для получения недостающих баллов в конце семестра	В случае если к промежуточной аттестации (зачету) набрана сумма менее 50 баллов, обучающемуся необходимо набрать недостающее количество баллов за счет выполнения дополнительных заданий, до конца последней (зачетной) недели семестра. При этом необходимо проработать материал всех пропущенных лабораторных работ. Формы дополнительных заданий (назначаются зачету): - выполнение и защита пропущенной лабораторной работы (при невозможности дополнительного проведения лабораторной работы преподаватель устанавливает форму дополнительного задания по тематике пропущенной лабораторной работы самостоятельно) – до 8 баллов. При прохождении рубежного контроля баллы ставятся в зависимости от рубежа. Ликвидация академических задолженностей, возникших из-за разности в учебных планах при переводе или восстановлении, проводится путем выполнения дополнительных заданий, форма и объем которых определяется преподавателем.

6.3. Процедура оценивания результатов освоения дисциплины

Рубежные контроли проводятся в форме письменного тестирования и решения задач, зачет в виде ответа на вопросы.

Перед проведением рубежного контроля преподаватель прорабатывает со обучающимися основной материал соответствующих разделов дисциплины в форме краткой лекции-дискуссии.

Варианты заданий для рубежных контролей № 1, № 2 состоят из 10 вопросов теста и одной практической задачи. На каждую подготовку к ответам по рубежному контролю студенту отводится 1 академический час.

Для определения баллов при проверке рубежных контролей используются интервальные оценки, представленные в таблице. Итоговая оценка получается суммированием баллов, набранных при ответе на тест и при решении задачи.

Количество правильных ответов	5-6	7-8	9-10	Решение задачи
Количество баллов	1	2	3	2

Преподаватель оценивает в баллах результаты рубежных контролей каждого студента по количеству правильных ответов и заносит в ведомость учета текущей успеваемости.

На зачете студенту предлагается ответить на 2 вопроса. Вопросы к зачету доводятся до студентов на последней лекции в семестре. Каждый вопрос

оценивается в 15 баллов. На подготовку ответа студенту отводится 1 астрономический час.

Результаты текущего контроля успеваемости и зачета заносятся преподавателем в ведомость, которая сдается в организационный отдел института в день зачета, а также выставляются в зачетную книжку студента.

6.4. Примеры оценочных средств для рубежных контролей и зачета

6.4.1 Примеры заданий для рубежного контроля №1

1. Подсчитать сумму каждого n-го элемента списка.
2. Найти факториал суммы элементов целочисленного списка.
3. Найти сумму факториалов элементов целочисленного списка.
4. Удалить отрицательные элементы из целочисленного списка.
5. Выполнить инверсию списка с использованием накапливающих параметров.
6. Выполнить построчную инверсию матрицы.
7. Найти максимальный/минимальный элементы матрицы.
8. Построить список из элементов главной диагонали матрицы.
9. Найти сумму элементов второстепенной диагонали матрицы.
10. Повернуть матрицу на 90 градусов по часовой стрелке.
11. Повернуть матрицу на 90 градусов против часовой стрелки.
12. Удалить из строки повторяющиеся элементы.
13. Произвести циклический сдвиг строки на n элементов вправо.
14. Произвести циклический сдвиг строки на n элементов влево.
15. Реализовать функцию двух аргументов, которая вычисляет длину двумерного вектора. Аргументы функции задают декартовы координаты конца вектора, его начало находится в начале координат.
16. Реализовать оператор |-|, который возвращает модуль разности переданных ему аргументов:
17. Определить функцию, вычисляющую двойной факториал, то есть произведение натуральных чисел, не превосходящих заданного числа и имеющих ту же четность. Н
18. Реализовать функцию трех аргументов, полиморфную по каждому из них, которая полностью игнорирует первый и третий аргумент, а возвращает второй. Укажите ее тип.
19. Реализовать функцию, которая бы добавляла два переданных ей значения в голову переданного списка.
20. Составить программу, которая будет спрашивать имя пользователя, а затем приветствовать его по имени. Причем, если пользователь не ввёл имя, программа должна спросить его повторно, и продолжать спрашивать, до тех пор, пока пользователь не представится.

6.4.3 Примеры заданий для рубежного контроля №2

1. Определить представителя класса Functor для следующего типа данных, представляющего точку в трёхмерном пространстве: data Point3D a = Point3D a a a deriving Show
2. Построить частотный словарь слов из входного предложения.
3. Реализовать базовые логические операции между двумя множествами: объединение, пересечение, вычитание, обратное пересечение.
4. Найти неизвестные слова в исходном предложении. Предварительно сформировать словарь известных слов.
5. Найти самое длинное слово в предложении.
6. Упорядочить слова предложения по убыванию их длины. Предварительно удалить повторяющиеся слова.
7. Кодирование слов исходного предложения двоичным кодом.
8. Декодирование исходного двоичного кода по заранее определенным правилам, задаваемым преподавателем.
9. Исследовать граф

Пройти тест на знание основ теории функционального программирования на языке Haskell (примерный список вопросов):

1.Что подразумевается под условным обозначением tycon:
(1) переменные
(2) классы типов
(3) конструкторы типов
2.Какие из перечисленных идентификаторов являются зарезервированными:
(1) case
(2) import
(3) default
(4) deriving
(5) infixl
3.

```
length ::[a] → Integer
length   [] = 0
length  (x:xs) = 1 + length xs
```

Этот пример выполняет:
(1) сложение элемента к списку
(2) приписывает к списку 1
(3) подсчет количества элементов в списке
4.Какое из утверждений верно?

(1) описание конструктора списков начинается с ":"
(2) функции в Haskell начинаются обязательно с любого заразервированного оператора
(3) Haskell не умеет работать как с конструкторами, так и с типами

5.Какое из утверждений не верно?
(1) имя может иметь не обязательные квалификаторы, при определленных обстоятельствах
(2) класс типа может быть квалифицирован, если к нему присоединить слева идентификатор модуля
(3) переменная типа может быть квалифицирована, если к нему присоединить справа идентификатор модуля

6.Выберите квалифицированные имена:
(1) qconid → [modid.] conid
(2) qmodid → [modid.] modid
(3) qvarid → [modid.] varid
(4) qconsym → [modid.] consym

7.Выберите инфиксный оператор:
(1) Prelude.+
(2) -.Prelude
(3) +.Prelude

8.Выберите числовые литералы:
(1) decimal
(2) octal
(3) Integer
(4) exponent

9.Какой литерал относится к числовым:
(1) hexadecimal
(2) cntrl
(3) charesc

10.Выберите символьные литералы:
(1) char
(2) escape
(3) string
(4) charesc

6.4.4 Примерный перечень вопросов для зачета

1. Понятие о функциональной парадигме, её отличие от прочих парадигм программирования.
2. Базовые математические понятия, лежащие в основе функционального стиля: множество, кортеж, соответствие, функция.
3. Понятие о чистой функции и референциальной прозрачности.
4. Карринг, частичное применение и функции высшего порядка.
5. Рекурсия. Понятие о хвостовой рекурсии, метод приведения рекурсии общего вида к хвостовому.
6. Списки и деревья.
7. Алгебраические типы данных.
8. Классы типов в языке Haskell.
9. Понятия функтора и аппликативного функтора. Примеры из стандартной библиотеки.
10. Понятие монады. Примеры из стандартной библиотеки.
11. Функция как функтор и монада.
12. Понятие модуля в Haskell. Правила оформления и использования модулей.
13. Средства многопоточного программирования в языке Haskell.
14. Полиморфизм и его виды в функциональном программировании.
15. Определения категории, мономорфизма, эпиморфизма.
16. Функторы. Законы функторов. Вложения. Подкатегории.
17. Правая и левая свертка списков.
18. Категория типов. Произведения и копроизведения.
19. Синонимы типов и конструкторы данных.
20. Рекурсивные типы данных. Перечислимые и бесконечные типы данных.
21. Стандартные монады модуля Prelude. Монадические классы.
22. Императивные возможности в функциональных языках.
23. Реализация операций ввода/вывода в Haskell.

6.5. Фонд оценочных средств

Полный банк заданий для текущего, рубежных контролей и промежуточной аттестации по дисциплине, показатели, критерии, шкалы оценивания компетенций, методические материалы, определяющие процедуры оценивания образовательных результатов, приведены в учебно-методическом комплексе дисциплины.

7. ОСНОВНАЯ И ДОПОЛНИТЕЛЬНАЯ УЧЕБНАЯ ЛИТЕРАТУРА

7.1. Основная учебная литература

1. Холомьёв, А. Учебник по Haskell. //Личный сайт автора [Электронный документ]. – 2012. – Режим доступа: https://anton-k.github.io/ru-haskell-book/files/ru-haskell-book.pdf. – Дата доступа: 18.05.2021.

2.Миран Липовача Изучай Haskell во имя добра! / Пер. с англ. Леушина Д., Синицына А., Арсанукаева Я.– М.: ДМК Пресс, 2012. – 490 с.: ил.

3. Мена А. Изучаем Haskell. Библиотека программиста. — СПб.: Питер, 2015. — 464 с.: ил. — (Серия «Библиотека программиста»).

4. Душкин Р. В. Справочник по языку Haskell. М.: ДМК Пресс, 2008. 544 с., ил.

5. Душкин Р. В. 14 занимательных эссе о языке Haskell и функциональном программировании. – М.: ДМК Пресс, 2011. – 140 с., ил.

6. Д. Шевченко. О Haskell по-человечески. 2016. Электронный ресурс: https://www.ohaskell.guide/pdf/ohaskell.pd

7.2. Дополнительная учебная литература

1. Lemmer, R. Haskell Design Patterns./ R. Lemmer. – Packt,2015. – 166 p.

2. Bird R. Thinking Functionally in Haskell. / R. Bird. – Cambridge university, 2015. – 358 p.

3. H. Barendregt. Lambda calculus with types. (http://ttic.uchicago.edu/~dreyer/course/papers/barendregt.pdf)

4. G. Hutton. Programming in Haskell (2nd ed.). Cambridge Univ. Press, 2016 (https://people.southwestern.edu/~potter/HaskellCode/hutton.pdf)

8 МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

8.1 Техническое обеспечение

№	Наименование	Использование
1	Комплект: ноутбук, медиа-проектор, экран	Для демонстрации иллюстративного материала при чтении лекций.
2	Персональный компьютер стандартной комплектации	Используется в качестве инструмента и объекта исследования при выполнении лабораторных и контрольных работ.

8.2 Программное обеспечение

№	Наименование	Использование
1	Hackage: архив пакетов Haskell URL: https://hackage.haskell.org/	Hackage: архив пакетов Haskell URL: https://hackage.haskell.org/
2	Документация Haskell URL: https://www.haskell.org/documentation	Документация Haskell URL: https://www.haskell.org/documentation
3	GitHub: веб-сервис для хостинга IT-проектов и совместной разработки URL: https://github.com/	GitHub: веб-сервис для хостинга IT-проектов и совместной разработки URL: https://github.com/

Аннотация
рабочей программы учебной дисциплины

ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ
образовательной программы высшего образования –
программы бакалавриата

09.03.03 – Прикладная информатика
направленность:
Интеллектуальные информационные системы и технологии

формы обучения – очная
Трудоемкость освоения дисциплины – 3 зач. ед. (108 акад. часов)
Семестры: 6-й (для очной формы обучения)
Промежуточная аттестация: зачет (6-й семестр)

Содержание дисциплины

Цель преподавания учебной дисциплины – овладение обучающимися бурно развивающимся и набирающим популярность стилем программирования, подготовка к его практическому использованию.

Особое внимание уделено общезначимым, независимым от конкретного языка, принципам, методам и понятиям ФП, что позволит обучающимся применять полученные знания при работе на множестве различных языков.

Задачи учебной дисциплины:

– дать обучающимся основу, необходимую для успешного усвоения перспективных технологий и методов программирования;

– дать обучающимся более глубокую и полную картину программирования за счёт альтернативной точки зрения на программы и процессы их выполнения;

– приобретенные знания позволяют понять современные тенденции в языках программирования, не относящихся к ФП.