Негосударственное аккредитованное некоммерческое частное образовательное учреждение высшего образования
«Академия маркетинга и социально-информационных технологий – ИМСИТ»
(г. Краснодар)

(НАН ЧОУ ВО Академия ИМСИТ)

УТВЕРЖДАЮ
Проректор по учебной работе, доцент

_____Н.И. Севрюгина

20.11.2023

Б1.О.18
Логическое программирование
рабочая программа дисциплины (модуля)

Закреплена за кафедрой **Кафедра математики и вычислительной техники**

Учебный план 45.03.04 Интеллектуальные системы в гуманитарной сфере

Квалификация **бакалавр**
Форма обучения **очная**
Общая трудоемкость **3 ЗЕТ**

Часов по учебному плану 108
в том числе:
аудиторные занятия 48
самостоятельная работа 59,8
контактная работа во время промежуточной аттестации (ИКР) 0

Виды контроля в семестрах:
зачеты с оценкой 5

Распределение часов дисциплины по семестрам

Семестр (<Курс>.<Семестр на курсе>)	**5 (3.1)**		Итого	
Недель	16 5/6			
Вид занятий	УП	РП	УП	РП
Лекции	16	16	16	16
Практические	32	32	32	32
Контактная работа на аттестации	0,2	0,2	0,2	0,2
Итого ауд.	48	48	48	48
Контактная работа	48,2	48,2	48,2	48,2
Сам. работа	59,8	59,8	59,8	59,8
Итого	108	108	108	108

Программу составил(и):
к. ф.-м. н., доцент, Бужан В.В.

Рецензент(ы):
д.т.н., профессор кафедры информационных систем и программирования КубГТУ, Видовский Л.А.;директор АО «ЮГ-СИСТЕМА ПЛЮС», Глебов О.В.

Рабочая программа дисциплины
Логическое программирование

разработана в соответствии с ФГОС ВО:
Федеральный государственный образовательный стандарт высшего образования - бакалавриат по направлению подготовки 45.03.04 Интеллектуальные системы в гуманитарной сфере (приказ Минобрнауки России от 24.04.2018 г. № 324)

составлена на основании учебного плана:
45.03.04 Интеллектуальные системы в гуманитарной сфере
утвержденного учёным советом вуза от 20.11.2023 протокол № 3.

Рабочая программа одобрена на заседании кафедры
Кафедра математики и вычислительной техники

Протокол от 13.10.2023 г. № 3

Зав. кафедрой Исикова Наталья Павловна

Согласовано с представителями работодателей на заседании НМС, протокол № 3 от 20.11.2023.

Председатель НМС проф. Павелко Н.Н.

1. ЦЕЛИ ОСВОЕНИЯ ДИСЦИПЛИНЫ (МОДУЛЯ)	
1.1	изучение парадигм функционального и логического программирования, используемых при решении задач искусственного интеллекта и элементов инженерии знаний, знакомство с теоретической базой, используемой при решении неформализуемых или плохо формализуемых задач
Задачи: развитие современного профессионального мировоззрения и знакомство с нестандартными подходами к решению задач на компьютерах, получение первичных навыков построения моделей на основе логической и функциональной парадигм, знакомство с техникой программирования задач искусственного интеллекта	

2. МЕСТО ДИСЦИПЛИНЫ (МОДУЛЯ) В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ	
Цикл (раздел) ОП:	Б1.О
2.1	**Требования к предварительной подготовке обучающегося:**
2.1.1	Алгоритмизация и программирование
2.2	**Дисциплины (модули) и практики, для которых освоение данной дисциплины (модуля) необходимо как предшествующее:**
2.2.1	Подготовка к процедуре защиты и защита выпускной квалификационной работы
2.2.2	Производственная практика: Преддипломная практика
2.2.3	Производственная практика: Технологическая (проектно-технологическая) практика

3. ФОРМИРУЕМЫЕ КОМПЕТЕНЦИИ, ИНДИКАТОРЫ ИХ ДОСТИЖЕНИЯ и планируемые результаты обучения	
ОПК-3: Способен разрабатывать алгоритмы и компьютерные программы, пригодные для практического применения.	
ОПК-3.1: Знает основные языки программирования и работы с базами данных, операционные системы и оболочки, современные программные среды разработки информационных систем и технологий	
Знать	
Уровень 1	обладать минимально допустимым уровнем знаний, допускать много негрубых ошибок
Уровень 2	обладать уровнем знаний в объёме, соответствующем программе подготовки, допускать несколько негрубых ошибок
Уровень 3	обладать знаниями в объёме, соответствующем программе подготовки, без ошибок
ОПК-3.2: Умеет применять языки программирования и работы с базами данных, современные программные среды разработки информационных систем и технологий для автоматизации бизнес-процессов, решения прикладных задач различных классов, ведения баз данных и информационных хранилищ	
Уметь	
Уровень 1	демонстрировать основные умения, решать типовые задачи с негрубыми ошибками и выполнять все задания, но не в полном объеме
Уровень 2	демонстрировать умения решать все основные задачи с негрубыми ошибками, выполнять все задания в полном объёме, но некоторые с недочётами
Уровень 3	демонстрировать умения решать все основные задачи с отдельными несущественными недочётами в полном объёме
ОПК-3.3: Владеет навыками программирования, отладки и тестирования прототипов программно-технических комплексов задач	
Владеть	
Уровень 1	минимальным опытом профессиональной деятельности и слабо выраженной личностной готовностью к профессиональному самосовершенствованию
Уровень 2	базовыми навыками решения стандартных задач с некоторыми недочётами
Уровень 3	навыками решения нестандартных задач без ошибок и недочётов

ПК-3: Способен разрабатывать и тестировать новые программы и интерфейсы систем	
ПК-3.1: Знает технологии разработки и тестирования программ, языки программирования и стандарты на представления результатов анализа и проектирования	
Знать	
Уровень 1	обладать минимально допустимым уровнем знаний, допускать много негрубых ошибок
Уровень 2	обладать уровнем знаний в объёме, соответствующем программе подготовки, допускать несколько негрубых ошибок
Уровень 3	обладать знаниями в объёме, соответствующем программе подготовки, без ошибок
ПК-3.2: Умеет использовать интегрированные среды разработки, включая средства визуального программирования, умеет использовать средства автоматизации этапов анализа и проектирования	
Уметь	
Уровень 1	демонстрировать основные умения, решать типовые задачи с негрубыми ошибками и выполнять все задания, но не в полном объеме

Уровень 2	демонстрировать умения решать все основные задачи с негрубыми ошибками, выполнять все задания в полном объёме, но некоторые с недочётами
Уровень 3	демонстрировать умения решать все основные задачи с отдельными несущественными недочётами в полном объёме
ПК-3.3: Владеет методами разработки и тестирования прикладных программ	
Владеть	
Уровень 1	минимальным опытом профессиональной деятельности и слабо выраженной личностной готовностью к профессиональному самосовершенствованию
Уровень 2	базовыми навыками решения стандартных задач с некоторыми недочётами
Уровень 3	навыками решения нестандартных задач без ошибок и недочётов

ПК-5: Способен использовать технические, программные средства и языки программирования для разработки алгоритмов и программ в области интеллектуального анализа данных, интеллектуальных и информационных систем	
ПК-5.1: Знает синтаксис, семантику, возможности и ограничения языков программирования, применяемых для разработки программных средств интеллектуального анализа данных, интеллектуальных и информационных систем	
Знать	
Уровень 1	обладать минимально допустимым уровнем знаний, допускать много негрубых ошибок
Уровень 2	обладать уровнем знаний в объёме, соответствующем программе подготовки, допускать несколько негрубых ошибок
Уровень 3	обладать знаниями в объёме, соответствующем программе подготовки, без ошибок
ПК-5.2: Умеет применять современные интегрированные среды разработки для создания систем интеллектуального анализа данных и интеллектуальных информационных систем	
Уметь	
Уровень 1	демонстрировать основные умения, решать типовые задачи с негрубыми ошибками и выполнять все задания, но не в полном объеме
Уровень 2	демонстрировать умения решать все основные задачи с негрубыми ошибками, выполнять все задания в полном объёме, но некоторые с недочётами
Уровень 3	демонстрировать умения решать все основные задачи с отдельными несущественными недочётами в полном объёме
ПК-5.3: Владеет методами разработки систем интеллектуального анализ данных, интеллектуальных и информационных систем	
Владеть	
Уровень 1	минимальным опытом профессиональной деятельности и слабо выраженной личностной готовностью к профессиональному самосовершенствованию
Уровень 2	базовыми навыками решения стандартных задач с некоторыми недочётами
Уровень 3	навыками решения нестандартных задач без ошибок и недочётов

4. СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

Код занятия	Наименование разделов и тем /вид занятия/	Семестр / Курс	Часов	Компетен-ции	Литература и эл. ресурсы	Практ. подг.
	Раздел 1. Методологические основы функционального программирования					
1.1	Функциональный подход к обработке информации /Лек/	5	2	ОПК-3.1 ОПК-3.2 ОПК-3.3	Л1.1 Л1.2 Л1.3Л2.1 Л2.2 Э1 Э2 Э3 Э4 Э5 Э6	
1.2	Функциональный подход к обработке информации /Пр/	5	4	ПК-3.1 ПК-3.2 ПК-3.3	Л1.1 Л1.2 Л1.3Л2.1 Л2.2 Э1 Э2 Э3 Э4 Э5 Э6	
1.3	Функциональный подход к обработке информации /Ср/	5	8	ПК-3.1 ПК-3.2 ПК-3.3	Л1.1 Л1.2 Л1.3Л2.1 Л2.2 Э1 Э2 Э3 Э4 Э5 Э6	
1.4	Реализации языков функционального программирования /Лек/	5	2	ПК-5.1 ПК-5.2 ПК-5.3	Л1.1 Л1.2 Л1.3Л2.1 Л2.2 Э1 Э2 Э3 Э4 Э5 Э6	
1.5	Реализации языков функционального программирования /Пр/	5	4	ПК-3.1 ПК-3.2 ПК-3.3	Л1.1 Л1.2 Л1.3Л2.1 Л2.2 Э1 Э2 Э3 Э4 Э5 Э6	
1.6	Реализации языков функционального программирования /Ср/	5	12	ПК-3.1 ПК-3.2 ПК-3.3	Л1.1 Л1.2 Л1.3Л2.1 Л2.2 Э1 Э2 Э3 Э4 Э5 Э6	
1.7	Лямбда-исчисление Черча /Лек/	5	2	ПК-3.1 ПК-3.2 ПК-3.3	Л1.1 Л1.2 Л1.3Л2.1 Л2.2 Э1 Э2 Э3 Э4 Э5 Э6	

1.8	Лямбда-исчисление Черча /Пр/	5	4	ПК-3.1 ПК-3.2 ПК-3.3	Л1.1 Л1.2 Л1.3Л2.1 Л2.2 Э1 Э2 Э3 Э4 Э5 Э6	
1.9	Лямбда-исчисление Черча /Ср/	5	12	ПК-5.1 ПК-5.2	Л1.1 Л1.2 Л1.3Л2.1 Л2.2 Э1 Э2 Э3 Э4 Э5 Э6	
1.10	Рекурсивные функции. Списки /Лек/	5	4	ПК-3.1 ПК-3.2 ПК-3.3	Л1.1 Л1.2 Л1.3Л2.1 Л2.2 Э1 Э2 Э3 Э4 Э5 Э6	
1.11	Рекурсивные функции. Списки /Пр/	5	8	ПК-3.1 ПК-3.2 ПК-3.3	Л1.1 Л1.2 Л1.3Л2.1 Л2.2 Э1 Э2 Э3 Э4 Э5 Э6	
1.12	Рекурсивные функции. Списки /Ср/	5	10,8	ПК-5.1 ПК-5.2 ПК-5.3	Л1.1 Л1.2 Л1.3Л2.1 Л2.2 Э1 Э2 Э3 Э4 Э5 Э6	
	Раздел 2. Программирование на функциональном языке					
2.1	Язык функционального программирования F# /Лек/	5	2	ОПК-3.1 ОПК-3.2 ОПК-3.3 ПК-3.1 ПК-3.2 ПК-3.3 ПК-5.1 ПК-5.2 ПК-5.3	Л1.1 Л1.2 Л1.3Л2.1 Л2.2 Э1 Э2 Э3 Э4 Э5 Э6	
2.2	Язык функционального программирования F# /Пр/	5	4	ОПК-3.1 ОПК-3.2 ОПК-3.3 ПК-3.1 ПК-3.2 ПК-3.3 ПК-5.1 ПК-5.2 ПК-5.3	Л1.1 Л1.2 Л1.3Л2.1 Л2.2 Э1 Э2 Э3 Э4 Э5 Э6	
2.3	Язык функционального программирования F# /Ср/	5	11	ОПК-3.1 ОПК-3.2 ОПК-3.3 ПК-3.1 ПК-3.2 ПК-3.3 ПК-5.1 ПК-5.2 ПК-5.3	Л1.1 Л1.2 Л1.3Л2.1 Л2.2 Э1 Э2 Э3 Э4 Э5 Э6	
2.4	Приёмы программирования на F# /Лек/	5	2	ОПК-3.1 ОПК-3.2 ОПК-3.3 ПК-3.1 ПК-3.2 ПК-3.3 ПК-5.1 ПК-5.2 ПК-5.3	Л1.1 Л1.2 Л1.3Л2.1 Л2.2 Э1 Э2 Э3 Э4 Э5 Э6	
2.5	Приёмы программирования на F# /Пр/	5	4	ОПК-3.1 ОПК-3.2 ОПК-3.3 ПК-3.1 ПК-3.2 ПК-3.3 ПК-5.1 ПК-5.2 ПК-5.3	Л1.1 Л1.2 Л1.3Л2.1 Л2.2 Э1 Э2 Э3 Э4 Э5 Э6	
2.6	Приёмы программирования на F# /Ср/	5	2	ОПК-3.1 ОПК-3.2 ОПК-3.3 ПК-3.1 ПК-3.2 ПК-3.3 ПК-5.1 ПК-5.2 ПК-5.3	Л1.1 Л1.2 Л1.3Л2.1 Л2.2 Э1 Э2 Э3 Э4 Э5 Э6	
	Раздел 3. Методологические основы логического программирования					
3.1	Анализ структуры термов; металогические предикаты; внелогические предикаты /Лек/	5	2	ПК-5.1 ПК-5.2 ПК-5.3	Л1.1 Л1.2 Л1.3Л2.1 Л2.2 Э1 Э2 Э3 Э4 Э5 Э6	
3.2	Анализ структуры термов; еталогические предикаты; внелогические предикаты /Пр/	5	4	ПК-5.1 ПК-5.2 ПК-5.3	Л1.1 Л1.2 Л1.3Л2.1 Л2.2 Э1 Э2 Э3 Э4 Э5 Э6	
3.3	Анализ структуры термов; еталогические предикаты; внелогические предикаты /Ср/	5	4	ПК-5.1 ПК-5.2 ПК-5.3	Л1.1 Л1.2 Л1.3Л2.1 Л2.2 Э1 Э2 Э3 Э4 Э5 Э6	

	Раздел 4. Промежуточная аттестация					
4.1	Зачёт /КА/	5	0,2	ОПК-3.1 ОПК-3.2 ОПК-3.3 ПК-3.1 ПК-3.2 ПК-3.3 ПК-5.1 ПК-5.2 ПК-5.3	Л1.1 Л1.2 Л1.3Л2.1 Л2.2 Э1 Э2 Э3 Э4 Э5 Э6	

5. ОЦЕНОЧНЫЕ МАТЕРИАЛЫ

5.1. Контрольные вопросы и задания

Список вопросов по дисциплине

1. Особенности функционального программирования, его отличие от императивного программирования. Понятие функции.
2. Понятие рекурсии и рекурсивных функций. Виды рекурсии.
3. Понятие лямбда-исчисления.
4. Свойства, особенности и назначение языка функционального программирования.
5. Понятие типа и структуры данных, их представление в памяти компьютера и основные операции над ними.
6. Определение функции для работы со списками: функции создания.
7. Понятие функции, способы записи функций в F#, иерархия вызовов.
8. Основная цель использования рекурсии в функциональном программировании.
9. Определение и назначение списков в функциональном программировании.
10. Понятие функционального подхода к решению задачи.
11. Понятие логического программирования.
12. Особенности логического программирования и его отличие от императивного и функционального программирования.
13. Понятие исчисления высказываний.
14. Понятие исчисления предикатов.
15. Понятие процедуры доказательства, связывания и унификации.
16. Понятие механизма возврата в логическом программировании.
17. Понятие операции отсечения в логическом программировании.
18. Реализация вычислений в логическом программировании.
19. Реализация рекурсивных функций в логическом программировании на примере вычисления факториала.
20. Области применения логического программирования

Задания для проведения текущего контроля

Вариант 0
Написать программу для реверса списка. Например: список [1, 2, 3] преобразуется в список [3, 2, 1].
Вариант 1
Написать программу для получения значения n-го элемента списка. Например: в списке [three, one, two] второй элемент равен one.
Вариант 2
Написать программу для удаления из списка всех элементов, равных 0. Например: список [1, 0, 2, 0, 0, 3] преобразуется в список [1, 2, 3].
Вариант 3
Написать программу для циклического сдвига списка вправо на заданное число элементов. Например: список [6, 5, 4, 3, 2, 1], циклически сдвинутый вправо на 2 элемента, преобразуется в список [2, 1, 6, 5, 4, 3].
Вариант 4
Написать программу для удаления из списка 2-ого, 4-ого и т.д. элементов. Например: список [6, 5, 4, 3, 2, 1] преобразуется в список [6, 4, 2].
Вариант 5
Написать программу для замены в списке всех элементы, равные 0, на -1. Например: список [1, 0, 0] преобразуется в список [1, -1, -1].
Вариант 6
Написать программу для перевода списка арабских чисел (от 1 до 10) в список римских чисел. Например: список [1, 2, 3] преобразуется в список [“I”, “II”, “III”].
Вариант 7
Написать программу для подсчёта количества определённых элементов в списке. Например: в списке [1, 2, 1, 3, 1] три единицы.
Вариант 8
Написать программу для подсчёта количества элементов списка без какого-либо указываемого элемента. Например: в списке [1, 2, 1, 3, 1] два элемента без учета единиц.
Вариант 9
Написать программу для подсчёта количества элементов списка, значения которых лежат в определённом диапазоне.

Например: в списке [10, 20, 10, 30, 15] два элемента, значения которых больше 10 и меньше 30.

Тесты

1. Какое из приведённых S-выражений может играть роль представления функций в Лисп-программе?
a. (defun fn (x) (cons x x))
b. (lambda car x)
c. (Label Первый Car)
2. По какой причине не может быть вычислена форма ((cons x ‘два) ‘три). Что в ней надо подправить, чтобы добиться вычислимости?
a. первый элемент списка имеет значение, но оно не представляет собой функцию. Например, можно перед ним вставить lambda (x). Это даст форму ((lambda (x)(cons x ‘два)) ‘три) и ее значение (три . два)
b. первый элемент формы должен быть атомом, так что можно убрать скобки из аргументов, что дает (cons x ‘два ‘три)
c. надо изображение переменной x заменить на конкретное значение, например ‘шесть, что даст нечто вроде ((cons ‘шесть ‘два) ‘три)
3. Какое из приведённых данных представляет собой составное S-выражение?
a. atom
b. (car (quote (cons a b)))
c. (cons (a b) NIL)
4. Какой из приведенных текстов не представляет собой ни список, ни сложное S-выражение?
a. (один, два, три, четыре, два)
b. (eq a (b . c))
c. ;; (a b c)
5. В каком из приведённых S-выражений представления функций расположены корректно?
a. (defun fn (x) (cons x x)) (fn 123)
b. ((cons ‘один ‘сто) ‘два ‘двести)
c. (((lambda (x) (car x))) ‘(сто двадцать три))
d. (два плюс три минус семь)
6. Что можно подправить, чтобы форма (cons ‘один ‘два ‘три) имела значение?
a. оставить в списке только два аргумента
b. заменить атомы списками
c. поставить апострофы перед атомами
7. Как можно воздействовать на форму (cons (a ‘два) (b ‘семь)), чтобы интерпретатор обязательно выдал ее значение, а не диагностическое сообщение?
a. символы a и b можно заменить на имена унарных функций над атомами, например atom или quote, что даёт (cons (atom ‘два) (quote ‘семь))
b. в начало каждого из аргументов следует вставить бинарную функцию, например, (eq a ‘два), (cons b ‘семь) в результате получится (cons (eq a ‘два) (cons b ‘семь))
c. заменить a и b на атом car, что даёт (cons (car ‘два) (car ‘семь))
8. Почему форма ((cons ‘a1 ‘b2) ‘c3) не может быть вычислена?
a. первый элемент формы должен быть атомом
b. функция cons не приспособлена к работе с атомами
c. потерян головной элемент списка, возможно, было (cons (cons ‘a1 ‘b2) ‘c3)
d. первый элемент списка имеет значение, но оно не представляет собой функцию
9. Какие из приведённых данных не представляют собой ни список, ни S-выражение?
a. (a . (b . c))
b. ;; (a b c)
c. ((cons (a b) NIL)
d. (list a1 b2 c31 d42)
10. Какие из данных текстов изображают перечень ветвей условного выражения?
a. (((eval(caar c)a)(eval(cadar c)a))(T(evcon(cdr c)a)))
b. (((null m)(cons(eval(car m)a)(evlis(cdr m)a)))
c. ((eq x (CAAR x)) (CAR al)) ((QUOTE T) (assoc x (CDR al)))
11. В каком случае правильно указано число обращений к CONS, которое произойдёт
a. при выполнении функции append (соединение двух списков в один) на заданных аргументах?
b. 5 (z x y) (w v)
c. 2 (a b) (c d)
d. 1 (a d c)(a)
e. 4 (a b) (a c d e)
12. На каком наборе данных функция member (поиск заданного элемента из списка) выполнит ровно одно обращение к самой себе?
a. d (a b c d e f)
b. f (a)
c. a (a c d e a f)
13. Какое из заданных выражений имеет определенный результат?
a. (pairlis – наращивает список пар, соединяя в пары элементы первых двух аргументов)
b. (pairlis (cons a b) ‘(t d f) ‘((k . y)(p . y)))

c. (pairlis () () ‘((k . y) (p . y)))
d. (pairlis (cons a b) (t d f) ((ten . x)(two . y)))

14. На каком наборе данных функция insert (вставляет в список перед заданным элементом третий аргумент) сделает менее трёх обращений к самой себе?
a. (a b c d) a c
b. (a b c d e) f c
c. (b c d e a) a b

15. Отметьте вариант, в котором правильно указано число обращений к себе функции equal, сравнивающей две структуры для выяснения, совпадают ли они?
a. 2 ((a b) c) ((a b) d)
b. 1 (a) (a)
c. 4 (a b c) (a b c d)

16. В каких случаях правильно указано число обращений к CONS, которое произойдёт при выполнении функции append (сцепление списков) на указанных аргументах?
a. append 2 (a b) (c d)
b. append 1 (a d c)(a)
c. append 3 (z x y) (w v)
d. append 4 (a b) (a c d e)

17. Какое из выражений при вычислении не искажает исходные данные?
a. (mapc #’list ‘(1 2 3) ‘(4 5 6) ‘(7 8 9))
b. (mapcon #’+ ‘(1 2 3) ‘(4 5 6) ‘(7 8 9))
c. (mapcan #’+ ‘(1 2 3) ‘(4 5 6) ‘(7 8 9))

18. Результат какой из форм совпадает с фактически построенной структурой?
a. (mapcar #’list ‘(1 2 3) ‘(4 5 6) ‘(7 8 9))
b. (mapc #’list ‘(1 2 3) ‘(4 5 6) ‘(7 8 9))
c. (mapl #’list ‘(1 2 3) ‘(4 5 6) ‘(7 8 9))

19. В какой из форм несоответствия типов значений помешает выполнению отображающей функции?
a. (mapcar #’+ ‘(1 2 3) ‘(4 5 6) ‘(7 8 9))
b. (maplist #’list ‘(1 2 3) ‘(4 5 6) ‘(7 8 9))
c. (mapc #’+ ‘(1 2 3) ‘(4 5 6) (+ 7 8 9))

20. В какой из форм выполнение отображающей функции не зависит от числа аргументов?
a. (mapcar #’CAR ‘(4 5 6) ‘(7 8 9))
b. (mapcar #’list ‘(1 2 3) ‘(4 5 6) ‘(7 8 9))
c. (maplist #’CONS ‘(7 8 9))

21. Какая из форм опасна для сохранения исходных данных?
a. (mapcon #’+ ‘(1 2 3) ‘(4 5 6) ‘(7 8 9))
b. (mapc #’list ‘(1 2 3) ‘(4 5 6) ‘(7 8 9))
c. (maplist #’list ‘(1 2 3) ‘(4 5 6) ‘(7 8 9))

22. Какая из форм формально дает результат, отличающийся от построенной структуры данных?
a. (mapcar #’list ‘(1 2 3) ‘(4 5 6) ‘(7 8 9))
b. (mapcar #’+ ‘(1 2 3) ‘(4 5 6) ‘(7 8 9))
c. (mapc #’+ ‘(1 2 3) ‘(4 5 6) ‘(7 8 9))

23. Какая из форм сохраняет исходные данные?
a. (mapl #’list ‘(1 2 3) ‘(4 5 6) ‘(7 8 9))
b. (mapcon #’list ‘(1 2 3) ‘(4 5 6) ‘(7 8 9))
c. (mapcan #’+ ‘(1 2 3) ‘(4 5 6) ‘(7 8 9))

24. Какое из S-выражений не равносильно (A (B C) D (E))?
a. (A . (B . ((C . (D . Nil)) . (E . Nil))))
b. (A (B C) . (D (E . Nil)))
c. (A (B . (C . Nil)) D (E))

25. Какое из перечисленных S-выражений равносильно (A B (C D) E)?
a. (A B (C . D) E)
b. (A B (C D) . E)
c. (A . (B . ((C . (D . Nil)) . (E . Nil))))

26. Отметьте функцию со свободной переменной
a. (lambda (x y) (cons z y))
b. (lambda (x y) (cons y x))
c. (lambda (x) (quote (A B C D)))

27. Которое из выражений даст результат (Альфа Центавра)?
a. ((lambda (x y) (list ‘x ‘y)) ‘ Альфа ‘ Центавра)
b. ((lambda (x y) (cons x y)) ‘ Альфа ‘ Центавра)
c. ((lambda (x) (list x ‘ Центавра)) ‘ Альфа)
d. ((lambda (x y) (list x ‘ Центр)) ‘ Альфа ‘ Центавра)

28. Какое выражение даст результат (A . B)?
a. ((lambda (y x) (cons x y)) ‘a ‘b)
b. (cons ‘a ‘b)
c. ((lambda (x) (cons x ‘A)) ‘b)

29. Которое из выражений построит список (Альфа Центавра)?

a. ((lambda (x y) (list y x)) ‘ Альфа ‘ Центавра)
b. ((lambda (x y) (cons x y)) ‘ Альфа ‘ Центавра)
c. ((lambda (x y) (list x y)) ‘ Альфа ‘ Центавра)
30. Которое из выражений не может дать результат (Альфа Центавра) независимо от значения переменной?
a. ((lambda (y) (cons y x)) ‘ Альфа)
b. ((lambda (b) (list b x)) ‘ Альфа)
c. ((lambda (b) (list x b)) ‘(Центавра))
d. (cons ‘(Центавра) x)

5.2. Темы письменных работ

Рефераты

Формой осуществления контроля выполнения самостоятельной работы является подготовки рефератов на актуальные темы, т. е. изучение с помощью научных методов явлений и процессов, анализа влияния на них различных факторов, а также, изучение взаимодействия между явлениями, с целью получения убедительно доказанных и полезных для науки и практики решений с максимальным эффектом.
Цель реферата – определение конкретного объекта и всестороннее, достоверное изучение его структуры, характеристик, связей на основе разработанных в науке принципов и методов познания, а также получение полезных для деятельности человека результатов, внедрение в производство с дальнейшим эффектом.
Основой разработки каждой темы является методология, т. е. совокупность методов, способов, приемов и их определенная последовательность, принятая при разработке научного исследования. В конечном счете, методология – это схема, план решения поставленной научно-исследовательской задачи.
Процесс подготовки реферат состоит из следующих основных этапов:
1. Выбор темы и обоснование ее актуальности.
2. Составление библиографии, ознакомление с законодательными актами, нормативными документами и другими источниками, относящимися к теме проекта (работы).
3. Разработка алгоритма исследования, формирование требований к исходным данным, выбор методов и инструментальных средств анализа.
4. Сбор фактического материала.
5. Обработка и анализ полученной информации с применением современных методов анализа.
6. Формулировка выводов и выработка рекомендаций.
7. Оформление работы в соответствии с установленными требованиями.

Темы рефератов

1. Профилирование программ на языке F#.
2. Профилирование программ на языке F#.
3. Автоматизированное тестирование программ на языке Haskell.
4. Отладка программ на языке Haskell.
5. Взаимодействие Haskell с платформой .NET.
6. Механизмы синхронизации параллельных вычислений в программах на языке F#.
7. Механизмы распределённых вычислений в программах на языке F#.
8. Представление и обработка сложных структур данных в программах на F#.
9. Средства работы с регулярными выражениями.
10. Использование F# для решения математических задач.
11. Обработка статистических данных.
12. Работа с низкоуровневыми функциями операционной системы

5.3. Фонд оценочных средств

Задание №1. Разработать рекурсивный вариант программы в функциональном стиле для решения предложенной ниже задачи.

Текст задания. Объединять два упорядоченных в порядке возрастания чисел списка в один упорядоченный

Пример:

> (name '(1 3 5 7) '(2 4 6))

(1 2 3 4 5 6 7)
Введение. Разрабатываемая функция выполняет селективное слияние списков. Аналогом является стандартная функция слияния списков append. Функция от двух аргументов, являющихся списками. Значением функции также является список. Поскольку требуется чисто функциональное решение, то в программе не допускается использование статических связей переменных (псевдофункции set, setq), форм организации циклических вычислений (do, do*), структуроразрушающего присваивания (rplaca, rplacd?, nconc, и т. д.). Функция должна поэлементно копировать элементы исходных списков в новую структуру, проверяя соблюдение условия упорядочения. Допускаются к использованию селекторы car, cdr и конструктор cons.
Декларативное описание решения. Решение обладает следующими свойствами:

• функция упорядоченного слияния name(list,list) -> list;

• если первый список пустой, то результат слияния – второй список (начальные условия рекурсии);

• если голова второго списка меньше головы первого, то результат – список, состоящий из головы второго и результата слияния хвоста второго списка с первым;

• иначе результирующий список состоит из головы первого и результата слияния хвоста первого списка со вторым.

Комментированный текст программы:

```
(defun name (list1 list2)
(cond ( (null list1) list2) ; НАЧАЛЬНЫЕ УСЛОВИЯ РЕКУРСИИ
;-Если первый  список пустой, то результат-второй  список
( (< (car list2) (car list1))
; если голова  второго списка меньше головы  первого, то результат
( cons (car list2)
; список, состоящий  из головы второго
(name (cdr list2) list1)))
; и результата  слияния хвоста второго списка  с первым списком
( t
; иначе
( cons (car list1)
; Искомый список  состоит из головы первого  списка
(name (cdr list1) list2)))
; и результата  слияния хвоста первого со  вторым списком
))
```

Комментированное тестирование программы

```
>(trace name)
```

Запускаем трассировку функции name

```
> (name '(1 3 5 7 9) '(2 4 6 8))
```

при каждом обращении к функции будет выдаваться список

аргументов

```
Entering: NAME, Argument list: ((1 3 5 7 9) (2 4 6 8))
Entering: NAME, Argument list: ((3 5 7 9) (2 4 6 8))
Entering: NAME, Argument list: ((4 6 8) (3 5 7 9))
Entering: NAME, Argument list: ((5 7 9) (4 6 8))
Entering: NAME, Argument list: ((6 8) (5 7 9))
Entering: NAME, Argument list: ((7 9) (6 8))
Entering: NAME, Argument list: ((8) (7 9))
```

Entering: NAME, Argument list: ((9) (8))

Выполнение начальных условий – первый список пустой

Entering: NAME, Argument list: (NIL (9))

Формирование результата

Exiting: NAME, Value: (9)

Exiting: NAME, Value: (8 9)

Exiting: NAME, Value: (7 8 9)

Exiting: NAME, Value: (6 7 8 9)

Exiting: NAME, Value: (5 6 7 8 9)

Exiting: NAME, Value: (4 5 6 7 8 9)

Exiting: NAME, Value: (3 4 5 6 7 8 9)

Exiting: NAME, Value: (2 3 4 5 6 7 8 9)

Exiting: NAME, Value: (1 2 3 4 5 6 7 8 9)

(1 2 3 4 5 6 7 8 9)
Частный случай применения: функция, определенная выше, может использоваться и для внедрения в упорядоченный список числа. При этом число должно подаваться как элемент списка, являющегося аргументом функции.

> (name '(1 2 3 4 5 6) '(3))

Entering: NAME, Argument list: ((1 2 3 4 5 6) (3))

Entering: NAME, Argument list: ((2 3 4 5 6) (3))

Entering: NAME, Argument list: ((3 4 5 6) (3))

Entering: NAME, Argument list: ((4 5 6) (3))

Entering: NAME, Argument list: (NIL (4 5 6))

Удовлетворение начальных условий рекурсии – обнаружение места, где должно находиться число

Exiting: NAME, Value: (4 5 6)

Exiting: NAME, Value: (3 4 5 6)

Exiting: NAME, Value: (3 3 4 5 6)

Exiting: NAME, Value: (2 3 3 4 5 6)

Exiting: NAME, Value: (1 2 3 3 4 5 6)

(1 2 3 3 4 5 6)

> (name '(4) '(1 2 3 5 6))

Entering: NAME, Argument list: ((4) (1 2 3 5 6))

Entering: NAME, Argument list: ((2 3 5 6) (4))

Entering: NAME, Argument list: ((3 5 6) (4))

Entering: NAME, Argument list: ((5 6) (4))

Entering: NAME, Argument list: (NIL (5 6))

Exiting: NAME, Value: (5 6)

Exiting: NAME, Value: (4 5 6)

Exiting: NAME, Value: (3 4 5 6)

Exiting: NAME, Value: (2 3 4 5 6)

Exiting: NAME, Value: (1 2 3 4 5 6)

(1 2 3 4 5 6)
Задание №2. Разработать итерационный вариант программы в императивном стиле для решения предложенной задачи.

Текст задания: Объединять два упорядоченных в порядке возрастания чисел списка в один упорядоченный

Пример:

> (name '(1 3 5 7) '(2 4 6))

(1 2 3 4 5 6 7)
Введение. Функция должна выполнять селективное объединение. В отличие задания №1 она должна быть выполнена в императивном стиле. Это означает следующее: вместо рекурсии для организации повторяющихся вычислений можно использовать конструкции организации повторяющихся вычислений (do, do*); допустимо использование разрушающего присваивания (set, nconc). Вместо декларативного описания свойств решения – словесное описание алгоритма.

Словесное описание алгоритма решения задачи.

• Функция двух формальных параметров list1и list2.

• Организуется цикл по локальной переменной list3 с начальным значением nil. Условие выхода из цикла – равенство nil переменных list1и list2. При этом возвращается значение переменной list3.

• Условное предложение.

Первое условие – если переменная list1 равна nil, то выполнить последовательность вычислений:

• выполнить слияние значений переменных list3 и list2;

• присвоить полученное значение переменной list3;

• присвоить переменной list2 значение nil.

Второе условие – если переменная list2 равна nil, то выполнить последовательность вычислений:

• выполнить слияние значений переменных list3 и list1;

• присвоить полученное значение переменной list3;

• присвоить переменной list1 значение nil.

Третье условие – если значение (car list1) меньше или равно значению (car list2), то выполнить последовательность вычислений:

• создать список из головы списка, представленного переменной list1;

• выполнить слияние списка, представленного переменной list3, и значения, полученного в предыдущем пункте;

• присвоить полученное значение переменной list3;

• присвоить переменной list1 значение cdr list1.

Четвертое условие – во всех остальных случаях выполнить последовательность вычислений:

• создать список из головы списка, представленного переменной list2;

• выполнить слияние списка, представленного переменной list3, и значения, полученного в предыдущем пункте;

• присвоить полученное значение переменной list3;

• присвоить переменной list2 значение cdr list2.
Комментированный текст программы

(defun name(list1 list2)

; Функция con двух формальных параметров list1и list2.

(do ((list3 nil))

; цикл по локальной переменной list3 с начальным значением nil.

((and (null list1) (null list2)) list3) ;УСЛОВИЕ ВЫХОДА ИЗ

ЦИКЛА

; условие выхода из цикла – равенство nil переменных list1и list2

; возвращается значение переменной list3 – результат всей программы

(cond

; условное предложение

((null list1)

; Первое условие – если переменная list1 равна nil

(progn

; выполнить последовательность вычислений:

(setq list3 (append list3 list2))

; выполнить слияние значений переменных list3 и list2

; присвоить полученное значение переменной list3

(setq list2 nil)))

; присвоить переменной list2 значение nil

((null list2)

; Второе условие – если переменная list2 равна nil

(progn (setq list3 (append list3 list1))

; выполнить слияние значений переменных list3 и list1

; присвоить полученное значение переменной list3

(setq list1 nil)))

; присвоить переменной list1 значение nil

((<= (car list1) (car list2))

; Третье условие – если значение (car list1) меньше или равно

; значения (car list2),

(progn

; то выполнить последовательность вычислений

(setq list3 (append list3 (cons (car list1) nil)))

```
; создать список  из головы списка, представленного  переменной
list1
; выполнить  слияние списка, представленного переменной list3, и
; значения, полученного  в предыдущем пункте
; присвоить  полученное значение переменной list3
(setq list1 (cdr list1))))
; присвоить  переменной list1 значение cdr list1
(t
; Четвертое условие – во всех остальных случаях(progn
; последовательность вычислений
(setq list3 ( append list3 (cons (car list2) nil) ))
; создать список  из головы списка, представленного  переменной
list2
; выполнить  слияние списка, представленного  переменной list3, и
; значения, полученного  в предыдущем пункте
; присвоить  полученное значение переменной list3
(setq list2 (cdr list2))
; присвоить  переменной list2 значение cdr list2
)))
))
```

Комментированное тестирование программы

```
> (trace name)
(NAME)
> (name '(1 3 5 7) '(2 4 6))
Entering: NAME, Argument list: ((1 3 5 7) (2 4 6))
Exiting: NAME, Value: (1 2 3 4 5 6 7)
(1 2 3 4 5 6 7)
```

5.4. Перечень видов оценочных средств

Задания со свободно конструируемым ответом (СКО) предполагает составление развернутого ответа на теоретический вопрос. Задание с выбором одного варианта ответа (ОВ, в задании данного типа предлагается несколько вариантов ответа, среди которых один верный. Задания со свободно конструируемым ответом (СКО) предполагает составление развернутого ответа, включающего полное решение задачи с пояснениями.

6. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

6.1. Рекомендуемая литература

6.1.1. Основная литература

	Авторы, составители	Заглавие	Издательство, год
Л1.1	Гордиенко А. П.	Функциональное программирование: Учебник	Москва: КноРус, 2022, URL: https://book.ru/book/942660

	Авторы, составители	Заглавие	Издательство, год
Л1.2	Голицына О. Л., Партыка Т. Л.	Языки программирования: Учебное пособие	Москва: Издательство "ФОРУМ", 2021, URL: http://znanium.com/catalog/document?id=367055
Л1.3	Пенькова Т.Г., Вайнштейн Ю.В.	Модели и методы искусственного интеллекта: Учебное пособие	Красноярск: Сибирский федеральный университет, 2019, URL: http://znanium.com/catalog/document?id=379870
6.1.2. Дополнительная литература			
	Авторы, составители	Заглавие	Издательство, год
Л2.1	Бедердинова О.И., Минеева Т.А.	Программирование на языках высокого уровня: Учебное пособие	Москва: ООО "Научно-издательский центр ИНФРА-М", 2019, URL: http://znanium.com/catalog/document?id=344897
Л2.2	Григорьев А.А., Исаев Е.А.	Методы и алгоритмы обработки данных: Учебное пособие	Москва: ООО "Научно-издательский центр ИНФРА-М", 2021, URL: http://znanium.com/catalog/document?id=361208
6.2. Электронные учебные издания и электронные образовательные ресурсы			
Э1	Интернет университе информационных технологий. - Режим доступа: https://www.intuit.ru/		
Э2	Естественно-научный образовательный портал. - Режим доступа: http://www.en.edu.ru/		
Э3	Электронная библиотечная система Znanium. - Режим доступа: http://znanium.com/%20		
Э4	Электронные ресурсы Академии ИМСИТ. - Режим доступа: http://eios.imsit.ru/		
Э5	Электронная библиотечная система Ibook. - Режим доступа: http://www.ibooks.ru/		
Э6	Электронная библиотечная система BOOK.ru. - Режим доступа: http://rpd.eios.imsit.ru:8080/RPD/Index/1636711/%20http://www.book.ru		
6.3.1. Лицензионное и свободно распространяемое программное обеспечение, в том числе отечественного производства			
6.3.1.1	Windows 10 Pro RUS Операцинная система – Windows 10 Pro RUS Подписка Microsoft Imagine Premium – Order №143659 от 12.07.2021		
6.3.1.2	Яндекс Браузер Браузер Яндекс Браузер Лицензионное соглашение на использование программ Яндекс Браузер https://yandex.ru/legal/browser_agreement/		
6.3.1.3	Mozilla Firefox Браузер Mozilla Firefox Программное обеспечение по лицензии GNU GPL		
6.3.1.4	LibreOffice Офисный пакет LibreOffice Программное обеспечение по лицензии GNU GPL		
6.3.1.5	MS Visio Pro 2016 Интегрированная средда разработки Microsoft Visio профессиональный 2016 Подписка Microsoft Imagine Premium – Order №143659 от 12.07.2021		
6.3.1.6	MS Visual Studio Community Edition Среда разработки Microsoft Visual Studio 2022 Программное обеспечение по лицензии GNU GPL		
6.3.2. Перечень профессиональных баз данных и информационных справочных систем			
6.3.2.1	ABOUT THE UNIFIED MODELING LANGUAGE SPECIFICATION https://www.omg.org/spec/UML		
6.3.2.2	ИСО Международная организация по стандартизации https://www.iso.org/ru/home.html		
6.3.2.3	РОССТАНДАРТ Федеральное агентство по техническому регулированию и метрологии https://www.gost.ru/portal/gost/		
6.3.2.4	Кодекс – Профессиональные справочные системы https://kodeks.ru		

7. МТО (оборудование и технические средства обучения)			
Ауд	Наименование	ПО	Оснащение
123	Лекционная аудитория	Windows 10 Pro RUS 7-Zip Яндекс Браузер Mozilla Firefox LibreOffice Kaspersky Endpoint Security MS Visio Pro 2016 Adobe Reader DC Klite Mega Codec Pack	Стол - 20 шт., стул - 21 шт., рабочее место преподавателя – 1 шт., персональный компьютер с выходом в интернет - 21 шт., доска учебная – 1 шт., многофункциональное устройство – 1 шт., мультимедийный проектор – 1 шт., проекционный экран – 1 шт., соответствующее программное обеспечение
125	Компьютерный класс	Windows 10 Pro RUS 7-Zip Яндекс Браузер Mozilla Firefox	Стол - 20 шт., стул - 21 шт., рабочее место преподавателя – 1 шт., персональный компьютер с выходом в интернет - 21 шт., доска учебная – 1 шт., многофункциональное устройство – 1 шт., мультимедийный проектор – 1 шт.,

		LibreOffice LibreCAD Inkscape Notepad++. 1С:Предприятие 8. Комплект Kaspersky Endpoint Security MS Access 2016 MS Project Pro 2016 MS SQL Server 2019 MS SQL Server Management Studio 18.8 MS Visio Pro 2016 MS Visual Studio Community Edition Visual Studio Code Gimp Maxima Oracle VM VirtualBox StarUML V1 PostgreSQL IntelliJ IDEA PyCharm Community Edition Eclips Adobe Reader DC Embarcadero RAD Studio XE8 Arduino Software (IDE) NetBeans IDE ZEAL ARIS Express Archimate Klite Mega Codec Pack Ramus Educational Micro-Cap Evaluation gvSIG Desktop Python	проекционный экран – 1 шт., соответствующее программное обеспечение
Читальный зал	Информационно-библиотечный центр (помещение для самостоятельной работы обучающихся)	7-Zip Яндекс Браузер Mozilla Firefox LibreOffice LibreCAD Inkscape Notepad++. Kaspersky Endpoint Security MS Access 2016 MS Project Pro 2016 MS Visio Pro 2016 Visual Studio Code Blender Gimp Maxima IntelliJ IDEA PyCharm Community Edition Adobe Reader DC MS Office Standart 2007 Windows 10 Pro	Стол - 20 шт., стул - 20 шт., рабочее место сотрудника - 2 шт., персональный компьютер с выходом в интернет и обеспечением доступа в электронную информационно-образовательную среду академии – 17 шт., многофункциональное устройство – 2 шт.

8. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ (МОДУЛЯ)

В соответствии с требованиями ФГОС ВО по направлению подготовки реализация компетентностного подхода предусматривает использование в учебном процессе активных и интерактивных форм проведения занятий (разбор конкретных задач, проведение блиц-опросов, исследовательские работы) в сочетании с внеаудиторной работой с целью формирования и развития профессиональных навыков обучающихся.

Лекционные занятия дополняются ПЗ и различными формами СРС с учебной и научной литературой В процессе такой работы студенты приобретают навыки «глубокого чтения» - анализа и интерпретации текстов по методологии и методике дисциплины.

Учебный материал по дисциплине «Логическое программирование». разделен на логически завершенные части (модули), после изучения, которых предусматривается аттестация в форме письменных тестов, контрольных работ.

Работы оцениваются в баллах, сумма которых дает рейтинг каждого обучающегося. В баллах оцениваются не только

знания и навыки обучающихся, но и их творческие возможности: активность, неординарность решений поставленных проблем. Каждый модуль учебной дисциплины включает обязательные виды работ – лекции, ПЗ, различные виды СРС (выполнение домашних заданий по решению задач, подготовка к лекциям и практическим занятиям).
Форма текущего контроля знаний – работа студента на практическом занятии, опрос. Форма промежуточных аттестаций – контрольная работа в аудитории, домашняя работа. Итоговая форма контроля знаний по модулям – контрольная работа с задачами по материалу модуля.
Методические указания по выполнению всех видов учебной работы размещены в электронной образовательной среде академии.
Методические указания и материалы по видам учебных занятий по дисциплине:
Вид учебных занятий, работ - Организация деятельности обучающегося
Лекция - Написание конспекта лекций: кратко, схематично, последовательно фиксировать основные положения, выводы, формулировки, обобщения, отмечать важные мысли, выделять ключевые слова, термины. Проверка терминов, понятий с помощью энциклопедий, словарей, справочников с выписыванием толкований в тетрадь. Обозначить вопросы, термины, материал, который вызывает трудности, попытаться найти ответ в рекомендуемой литературе, если самостоятельно не удаётся разобраться в материале, необходимо сформулировать вопрос и задать преподавателю на консультации, на практическом занятии.
Практические занятия - Конспектирование источников. Работа с конспектом лекций, подготовка ответов к контрольным вопросам, просмотр рекомендуемой литературы, работа с текстом. Выполнение практических задач в инструментальных средах. Выполнение проектов. Решение расчётно-графических заданий, решение задач по алгоритму и др.
Самостоятельная работа - Знакомство с основной и дополнительной литературой, включая справочные издания, зарубежные источники, конспект основных положений, терминов, сведений, требующихся для запоминания и являющихся основополагающими в этой теме. Составление аннотаций к прочитанным литературным источникам и др.

9. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ОБУЧАЮЩИМСЯ ПО ВЫПОЛНЕНИЮ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Самостоятельная работа студентов в ходе семестра является важной составной частью учебного процесса и необходима для закрепления и углубления знаний, полученных в период сессии на лекциях, практических и интерактивных занятиях, а также для индивидуального изучения дисциплины «Логическое программирование» в соответствии с программой и рекомендованной литературой.
Самостоятельная работа выполняется в виде подготовки домашнего задания или сообщения по отдельным вопросам, написание и защита научно-исследовательского проекта.
Контроль качества выполнения самостоятельной (домашней) работы может осуществляться с помощью устного опроса на лекциях или практических занятиях, обсуждения подготовленных научно-исследовательских проектов, проведения тестирования.
Устные формы контроля помогут оценить владение студентами жанрами научной речи (дискуссия, диспут, сообщение, доклад и др.), в которых раскрывается умение студентов передать нужную информацию, грамотно использовать языковые средства, а также ораторские приемы для контакта с аудиторией.
Письменные работы позволяют оценить владение источниками, научным стилем изложения, для которого характерны: логичность, точность терминологии, обобщённость и отвлеченность, насыщенность фактической информацией.
Выбор варианта задания может быть сделан из предложенного ниже списка:

Задачи по теме «Деревья (Lisp)»

Вариант 0
Написать программу для нахождения среднего арифметического литьевых вершин бинарного дерева.
Вариант 1
Написать программу для проверки упорядоченности бинарного дерева.
Вариант 2
Вывести бинарное дерево на экран в виде дерева.
Вариант 3
Написать программу для вычисления глубины бинарного дерева (глубина пустого дерева равна 0, глубина одноузлового дерева равна 1).
Вариант 4
Написать программу для подсчёта количества литьевых вершин дерева, значения которых лежат в определённом диапазоне.
Вариант 5
Написать программу для преобразования дерева в список.
Вариант 6
Написать программу для нахождения среднего арифметического отрицательных узлов дерева.
Вариант 7
Написать программу для подсчёта количества вершин бинарного дерева, значения которых не равны 0.
Вариант 8
Написать программу для нахождения среднего арифметического положительных узлов дерева.
Вариант 9
Написать программу для подсчёта количества вершин бинарного дерева, значения которых равны 0.

Задачи по теме «Основы функционального программирования на F#»

Вариант 0
Определить рекурсивную функцию, возвращающую значение n-го члена ряда Фибоначчи: f(0)=0, f(1)=1, f(n)=f(n-1)+f(n-2).
Вариант 1
Определить рекурсивную функцию для удаления последнего элемента списка.
Вариант 2
Определить рекурсивную функцию, возвращающую произведение двух целых положительных чисел (использовать суммирование).
Вариант 3
Определить рекурсивную функцию, возвращающую последний элемент списка.
Вариант 4
Определить рекурсивную функцию, возвращающую значение суммы ряда целых чётных чисел от 2 до n.
Вариант 5
Определить рекурсивную функцию, возвращающую список, из которого удалены 2- ой, 4-ый и т.д. элементы.
Вариант 6
Определить рекурсивную функцию, возвращающую количество элементов в списке без какого-либо указываемого элемента.
Вариант 7
Определить рекурсивную функцию, возвращающую количество определённых элементов в списке.
Вариант 8
Определить рекурсивную функцию для циклического сдвига списка вправо на один элемент.
Вариант 9
Определить рекурсивную функцию, возвращающую список, из которого удалены 1-й, 3-й и т.д. элементы.