МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«Дальневосточный федеральный университет»
(ДВФУ)
ШКОЛА ЕСТЕСТВЕННЫХ НАУК

«СОГЛАСОВАНО»
Руководитель образовательной программы

_____ И.Л. Артемьева
«28» 08 2015 г.

«УТВЕРЖДАЮ»
Заведующая кафедрой прикладной математики, механики, управления и программного обеспечения

_____ И.Л. Артемьева
«28» 08 2015 г.

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ
Функциональное и логическое программирование

Направление подготовки – 02.03.03 «Математическое обеспечение и администрирование информационных систем»
профиль «Технология программирования»
Форма подготовки (очная)

курс 4 семестр 7, 8
лекции 30 час.
практические занятия _____час.
лабораторные работы 42 час.
в том числе с использованием МАО лек._____/пр._____/лаб 30 час.
в том числе в электронной форме лек._____/пр._____/лаб. 30 час.
всего часов аудиторной нагрузки 72 час.
в том числе с использованием МАО 30 час.
в том числе в электронной форме 30 час.
самостоятельная работа 72 час.
в том числе на подготовку к экзамену _____ час.
курсовая работа / курсовой проект – не предусмотрен
зачет 7, 8 семестр
экзамен – не предусмотрен

Рабочая программа составлена в соответствии с требованиями федерального государственного образовательного стандарта высшего образования, утвержденного приказом Министерства образования и науки РФ от 12 марта 2015 г. № 222

Рабочая программа обсуждена на заседании кафедры прикладной математики, механики, управления и программного обеспечения, протокол № _7__ от «_4__» _июля_ 2015_г.

Заведующая кафедрой прикладной математики, механики, управления и программного обеспечения Артемьева И.Л., д.т.н., профессор
Составитель (ли): зав. кафедрой ПММУиПО И.Л.Артемьева д.т.н., профессор,
доценг кафедры ПММУиПО Крылов Д.А, к.т.н.

Оборотная сторона титульного листа РПУД

I. Рабочая программа пересмотрена на заседании кафедры:

Протокол от «_____» _____ 20___ г. № _____

Заведующий кафедрой _____ _____
(подпись) (И.О. Фамилия)

II. Рабочая программа пересмотрена на заседании кафедры:

Протокол от «_____» _____ 20___ г. № _____

Заведующий кафедрой _____ _____
(подпись) (И.О. Фамилия)

ABSTRACT

Bachelor's/Specialist's/Master's degree in 02.03.03 – Software and Administration of Information Systems

Study profile/ Specialization/ Master's Program "Title" Programming technology

Course title: Functional and logic programming

Basic part of Block 1, _4_credits

Instructor: Artemeva I., Krylov D.

At the beginning of the course a student should be able to: apply program development methods.

Learning outcomes: ability to use knowledge of main conceptual provisions of functional, logic, object-oriented and visual trends of programming, methods and tools of the development of programs in the framework of these trends.

Course description: functional and logic programming languages, programming in logic and functional languages, modern programming languages with the elements of functional languages.

Main course literature:

1. Ezdakov A.L. Funktsional'noe i logicheskoe programmirovanie [Functional and logic programming]. Moscow, BINOM, 2009. 119 p.

http://lib.dvfu.ru:8080/lib/item?id=chamo:277585&theme=FEFU

2. Cherkashin E.A. Rekursivno-logicheskoe programmirovanie [Recursive and logic programming]. Irkutsk, Irkutsk State University Publ., 2013. 109 p.

https://lib.dvfu.ru:8443/lib/item?id=chamo:748386&theme=FEFU

3. Markov V.N. Sovremennoe logicheskoe programmirovanie na yazyke Visual Prolog 7.5 [Modern logic programming in Visual Prolog 7.5]. Saint Petersburg, BHV-Petersburg, 2016. 541 p.

https://lib.dvfu.ru:8443/lib/item?id=chamo:823207&theme=FEFU

4. Artemeva I.L. Funktsional'noe programmirovanie [Functional programming]. Vladivostok, Far Eastern Federal University Publ., 2011. 38 p.

5. Artemeva I.L. Rekursivno-logicheskoe programmirovanie [Recursive and logic programming]. Vladivostok, Far Eastern Federal University Publ., 2011. 32 p.

6. Andreeva V.V. Logicheskoe programmirovanie na yazyke Visual Prolog [Logic programming in Visual Prolog]. Tomsk, National Research Tomsk State University Publ., 2013. 104 p.

https://e.lanbook.com/book/44911

7. Dushkin R.V. Funktsional’noe programmirovanie na yazyke Haskell [Functional programming in Haskell]. Moscow, DMK Press, 2008. 609 p.
http://znanium.com/catalog.php?bookinfo=409306

Form of final control: pass-fail exam.

АННОТАЦИЯ

Рабочая программа дисциплины «Функциональное и логическое программирование» разработана для студентов 4 курса, обучающихся по направлению 02.03.03 «Математическое обеспечение и администрирование информационных систем», профиль «Технология программирования». Трудоемкость дисциплины 4 зачетных единицы (144 часа). Дисциплина базируется на дисциплинах «Математическая логика», «Теория вычислительных процессов и структур».

Дисциплина реализуется в 7 и 8 семестрах. В 7 семестре дисциплина содержит 18 часов лекций, 0 часов практических занятий, 18 часов лабораторных работ, из них 0 часов лекций, 0 часов практических занятий, 18 часов лабораторных работ с использованием методов активного обучения. На самостоятельную работу студентов отводится 36 часов. В 8 семестре дисциплина содержит 12 часов лекций, 0 часов практических занятий, 24 часов лабораторных работ, из них 0 часов лекций, 0 часов практических занятий, 12 часов лабораторных работ с использованием методов активного обучения. На самостоятельную работу студентов отводится 36 часов.

Цель дисциплины - познакомить студентов с логическими и функциональными языками программирования и программными системами, в основе которых лежит ламбда исчисление, порождающие модели и исчисление предикатов, а также с методами реализации таких систем, с особенностями программирования на языках данных классов.

Задачи дисциплины:

- изучение класса функциональных языков программирования;
- изучение класса логических языков программирования;
- получение навыков программирования на логических и функциональных языках
- изучение современных языков программирования с элементами функциональных языков.

Для успешного изучения дисциплины «Функциональное и логическое программирование» обучающиеся должны овладеть методами разработки программ.

В результате изучения данной дисциплины у обучающихся формируются следующие общекультурные/ общепрофессиональные/ профессиональные компетенции (элементы компетенций).

Код и формулировка компетенции	Этапы формирования компетенции	
ОПК7 Способность использовать знания основных концептуальных положений функционального, логического, объектно-ориентированного и визуального направлений программирования, методов, способов и средств разработки программ в рамках этих направлений	Знает	языки функционального и логического программирования отличие семантики логических и функциональных языков от семантики алгоритмических
	Умеет	программировать на функциональных и логических языках
	Владеет	Навыками использования языковых процессоров логических и функциональных языков

Для формирования вышеуказанных компетенций в рамках дисциплины «Функциональное и логическое программирование» применяются следующие методы активного/ интерактивного обучения: метод проектов.

- **СТРУКТУРА И СОДЕРЖАНИЕ ТЕОРЕТИЧЕСКОЙ ЧАСТИ КУРСА**

Лекционный материал (30 час.)

Раздел 1. Функциональное программирование и функциональные языки (18 часов)

Тема 1. Отличие функциональных языков от алгоритмических (императивных). Программирование с помощью функций и процедур. Строгий и нестрогий функциональный язык. Функциональные языки строгой и нестрогой типизации. Отличие семантики. Примеры языков: Лисп, Haskell. Примеры языков с элементами функциональных (3 час.)

Тема 2. Ламбда исчисление А.Черча. Понятие функции в функциональных языках. Определение функции, описание функции, вызов функции в функциональных языках. Способы программирования в функциональных обозначениях. (2 час.)

Тема 3. Рекурсивные функции, правила их записи в функциональных языках. Приемы программирования с использованием рекурсивных функций. Работа со списками в функциональных языках. (2 час.)

Тема 4. Функции высших порядков. Их использование в функциональных языках. Отображающие и применяющие функционалы в Лисп. (2 час.)

Тема 5. Переменные в функциональных языках. Область действия переменных. Типы формальных параметров в Лисп. Квалификационные

выражения как способ сокращения записи. Примеры определения квалификационных выражений. (2 час.)

Тема 6. Методы реализации функциональных языков. Интерпретаторы функциональных программ. Сборщики «мусора». (2 час.)

Тема 7. Использование концепций функциональных языков в современных языках программирования. Языки Python, Ruby, и другие. (5 час.)

Раздел 2. Рекурсивно-логическое программирование (12 часов)

Тема 1. Языки, основанные на исчислении предикатов. (10 час.)

Общие сведения о языке логического программирования; основные элементы языка и приемы программирования; согласование целевых утверждений. Хорновские дизъюнкты в Пролог. Вывод от фактов к цели (восходящий процесс) и от цели к фактам (нисходящий процесс) для логических языков. Вычислительная и декларативная семантика языка Пролог. Метод резолюций. Примитивные программы в Пролог. Основные и универсальные факты. Простой и составной вопросы. Средства представления и обработки списков в Пролог. Рекурсивные представления данных и программ. Арифметика в языке логического программирования. Рекурсивные представления данных и программ. Возврат и отсечение в Пролог. Методы его использования. Встроенные предикаты. Особенности отладки программ; примеры использования языка логического программирования для решения задач искусственного интеллекта.

Концепция "открытого" и "замкнутого" миров в логических языках при работе с негативными фактами. Работа с негативными фактами в Пролог. Логика второго порядка в Пролог.

Тема 2. Языки, основанные на порождающих моделях. (2 час.)

Структура состояний рабочей среды. Операции над рабочей средой. Синтаксис проблемно–ориентированных правил. Правила вывода и правила остановки. Свойства процесса вывода.

СТРУКТУРА И СОДЕРЖАНИЕ ПРАКТИЧЕСКОЙ ЧАСТИ КУРСА

Практические занятия (0 час.)

Практические занятия учебным планом не предусмотрены.

Лабораторные работы (42 час.)

f

Лабораторная работа 1. Программирование на языке Лисп с использованием некоторого языкового процессора (6 час.)

Лабораторная работа 2. Программирование на другом функциональном языке (по выбору студентов) с использованием некоторого языкового процессора (8 час.)

Лабораторная работа 3. Выполнение зачетных заданий на функциональных языках (4 часа)

Лабораторная работа 4. Программирование на современных языках программирования с элементами функциональных (по выбору студентов) (12 час.)

Лабораторная работа 5. Программирование на языке Пролог с использованием одного из существующих языковых процессоров. Изучение методов программирования и отладки программ (24 час.)

Ш. УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ

Учебно-методическое обеспечение самостоятельной работы обучающихся по дисциплине «Функциональное и логическое программирование» представлено в Приложении 1 и включает в себя: план-график выполнения самостоятельной работы по дисциплине, в том числе примерные нормы времени на выполнение по каждому заданию; характеристика заданий для самостоятельной работы обучающихся и методические рекомендации по их выполнению; требования к представлению и оформлению результатов самостоятельной работы; критерии оценки выполнения самостоятельной работы.

IV. КОНТРОЛЬ ДОСТИЖЕНИЯ ЦЕЛЕЙ КУРСА

№ п/п	Контролируемые разделы/темы	Коды и этапы формирования	Оценочные средства - наименование

	дисциплины	компетенций		текущий контроль	промежуточная аттестация
1	Раздел 1. Функциональное программирование и функциональные языки	ОПК7	Знает	ПР1 тест	Зачет, семестр 7, вопросы № 1-15
			Умеет, владеет	ПР6 лабораторная работа ПР11 задания для самостоятельного выполнения	
2	Раздел 2. Рекурсивно-логическое программирование	ОПК7	Знает	ПР1тест	Зачет, семестр 8, вопросы № 1-16
			Умеет, владеет	ПР6 лабораторная работа ПР11 задания для самостоятельного выполнения	

Типовые контрольные задания, методические материалы, определяющие процедуры оценивания знаний, умений и навыков и (или) опыта деятельности, а также критерии и показатели, необходимые для оценки знаний, умений, навыков и характеризующие этапы формирования компетенций в процессе освоения образовательной программы, представлены в Приложении 2.

V. СПИСОК УЧЕБНОЙ ЛИТЕРАТУРЫ И ИНФОРМАЦИОННО МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Основная литература

1. Программирование на F# / Крис Смит; [пер. с англ. А. Киселева]. Санкт-Петербург Москва: Символ-Плюс, 2011. 447 с. https://lib.dvfu.ru:8443/lib/item?id=chamo:786184&theme=FEFU
2. Ездаков А.Л. Функциональное и логическое программирование. [Электронный ресурс]. Учебное пособие. М.:БИНОМ. Лаборатория знаний, 2009.- 119 с. http://lib.dvfu.ru:8080/lib/item?id=chamo:277585&theme=FEFU
3. Рекурсивно-логическое программирование: учебное пособие / Е. А. Черкашин; Иркутский государственный университет, Институт динамики систем и теории управления Сибирского отделения РАН. Иркутск: Изд-во Иркутского университета, 2013. - 109 с. https://lib.dvfu.ru:8443/lib/item?id=chamo:748386&theme=FEFU
4. Современное логическое программирование на языке Visual Prolog 7.5: учебник для вузов / В. Н. Марков. Санкт-Петербург : БХВ-Петербург, 2016. 541 с. https://lib.dvfu.ru:8443/lib/item?id=chamo:823207&theme=FEFU

5. Артемьева И.Л. Функциональное программирование: курс лекций. Владивосток: Изд. ДВФУ, 20011. – 38с.
6. Артемьева И.Л. Рекурсивно-логическое программирование: Учебно-методическое пособие. – Владивосток: Издательство ДВФУ.- 2011. – 32 с.
7. Ездаков А.Л. Функциональное и логическое программирование. [Электронный ресурс]. Учебное пособие. М.:БИНОМ. Лаборатория знаний, 2011.- 119 с. http://www.znanium.com/catalog.php?bookinfo=366490
8. Python на практике / М. Саммерфилд: Пер. с англ. Слинкин А.А. М.: ДМК Пресс, 2014. – 338 с. https://e.lanbook.com/book/66480
9. Логическое программирование на языке Visual Prolog / В.В. Андреева. Томск: Национальный исследовательский Томский государственный университет, 2013. – 104 с. https://e.lanbook.com/book/44911#authors
10. Душкин Р. В. Функциональное программирование на языке Haskell / Гл. ред. Д. А. Мовчан;. — М.: ДМК Пресс, 2008. — 609 с., ил. с. http://znanium.com/catalog.php?bookinfo=409306

Дополнительная литература

(печатные и электронные издания)

1. Теория и практика логического программирования на языке Visual Prolog 7.Учебное пособие для вузов. Издательство "Горячая линия-Телеком", 2013. – 232 с. https://e.lanbook.com/book/11847#authors
2. Душкин, Р. В. 14 занимательных эссе о языке Haskell и функциональном программировании [Электронный ресурс] / Р. В. Душкин. - М.: ДМК Пресс, 2011. - 288 с.: ил. - ISBN 978-5-94074-691-1. http://znanium.com/catalog.php?bookinfo=408876
3. Душкин, Р. В. Справочник по языку Haskell [Электронный ресурс] / Р. В. Душкин. - М.: ДМК Пресс, 2009. - 544 с.: ил. - ISBN 5-94074-410-9. http://znanium.com/catalog.php?bookinfo=407123
4. Душкин, Р. В. Практика работы на языке Haskell [Электронный ресурс] / Р. В. Душкин. - М.: ДМК Пресс, 2010. - 288 с.: ил. - ISBN 978-5-94074-588-4. http://znanium.com/catalog.php?bookinfo=408467
5. Лутц М. Изучаем Python. 4-е издание. Пер. с англ. СПб: Символ-Плюс. 2011. 1280 с.
6. Братко И. Алгоритмы искусственного интеллекта на языке PROLOG = Prolog Programming For Artificial Intelligence — М.:«Вильямс», 2004. — С. 640.

7. Адаменко А., Кучуков А. Логическое программирование и Visual Prolog-СПб: БХВ-Петербург, 2003.- 990 с.
8. Солдатова О.П., Лёзина И.В. Логическое программирование на языке Visual Prolog.- Учебное пособие - Самара: СНЦ РА, 2010. - 81 с.
9. Роганова Н.А. Функциональное программирование. М:ГИНФО, 2002. - 260с.
10. Душкин, Р. В. Практика работы на языке Haskell [Электронный ресурс] / Р. В. Душкин. - М.: ДМК Пресс, 2010. - 288 с.: ил. - ISBN 978-5-94074-588-4.
11. Саммерфилд М. Программирование на Python 3. Подробное руководство.- Пер. с англ. СПб: Символ-Плюс. 2009. 608 с.
12. Флэнаган Д., Мацумото Ю. Язык программирования Ruby. – СПб:Питер, 2011 – 496 с.
13. http://fprog.ru/Практика функционального программирования, журнал, 2009-2011, №№ 1-7.
14. Братко И. Программирование на языке Пролог для искусственного интеллекта. М: Мир, 1990. - 559 с. https://lib.dvfu.ru:8443/lib/item?id=chamo:29454&theme=FEFU
15. Стерлинг Л., Шапиро Э. Искусство программирования на языке Пролог. М: Мир, 1990. – 333 с. https://lib.dvfu.ru:8443/lib/item?id=chamo:679623&theme=FEFU
16. Фитцджеральд, М. Изучаем Ruby [Электронный ресурс] / М. Фитцджеральд - СПб.: БХВ-Петербург, 2008. - 336 с.: ил. - ISBN 978-5-9775-0225-2. http://znanium.com/catalog.php?bookinfo=489640
17. В.Г.Крылов - Основы логического программирования. Учебное пособие Екатеринбург, 2009 г. - 62 стр.
18. Братко И. Алгоритмы искусственного интеллекта на языке PROLOG = Prolog Programming For Artificial Intelligence — М.:«Вильямс», 2004. — 640. С.

Перечень ресурсов информационно-телекоммуникационной сети «Интернет»

1. http://progbook.ru/hope/733-fild-funktsionalnoe-programmirovanie.htmlФилд А., Харрисон П. Функциональное программирование. М.: Мир, 1993. 637 с.
2. http://www.proklondike.com/books/codingproch/huvenen_semplyanen_mir_lispa_1990.html Хювенен Э., Сеппянен Й. Мир Лиспа. Методы и системы программирования: М:Мир, 1990. 2 т.

3. http://www.intuit.ru/studies/courses/49/49/info Сузи Р.А. Язык программирования Python: учебное пособие. - 2-е изд., испр. М.: БИНОМ. Лаборатория знаний. Интернет-университет информационных технологий. - 2007 – 328 с.
4. http://window.edu.ru/library/pdf2txt/485/60485/30391 Роганов Е.А., Роганова Н. А. Программирование на языке Ruby. Учебное пособие. — М.: МГИУ, 2008. — 56 с.
5. http://window.edu.ru/resource/726/41726 Городняя Л.В. Основы функционального программирования. М.: Изд. «Интернет университет информационных технологий -ИНТУИТ. - 2004.- 280 с.
6. http://window.edu.ru/resource/726/41726/files/funprog.pdf Городняя Л.В. Основы функционального программирования. М.: Изд. «Интернет университет информационных технологий -ИНТУИТ. - 2004.- 280 с.
7. http://www.studentlibrary.ru/book/ISBN9785970602997.html Программирование на Clojure [Электронный ресурс] / Эмерик Ч., Карпер Б., Гранд К. - М.: ДМК Пресс, 2015. -
8. http://www.studentlibrary.ru/book/ISBN5940743579.html Программирование на языке Ruby [Электронный ресурс] / Фултон Х. - М.: ДМК Пресс, 2007. -
9. http://www.studentlibrary.ru/book/ISBN5940743358.html Функциональное программирование на языке Haskell [Электронный ресурс] / Душкин Р.В. - М.: ДМК Пресс, 2008. -
10. http://www.studentlibrary.ru/book/ISBN9785940746898.html Функциональное программирование на F# [Электронный ресурс] / Сошников Д.В. - М.: ДМК Пресс, 2011. -
11. http://www.studentlibrary.ru/book/ISBN9785991201940.html Теория и практика логического программирования на языке Visual Prolog 7 [Электронный ресурс]: Учебное пособие для вузов / Цуканова Н.И., Дмитриева Т.А. - М.: Горячая линия - Телеком, 2013. -
12. http://fprog.ru/ Практика функционального программирования, журнал, 2009-2011, №№ 1-7.

Перечень информационных технологий и программного обеспечения

Лекции проводятся с использованием проектора и мультимедийного комплекса для проведения лекций внутренней системы портала ДВФУ. Лабораторные занятия проводятся в специализированном компьютерном классе. Для составления документации используется текстовый процессор (LibreOffice

или MicrosoftWord).

VI. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Дисциплина изучается в следующих организационных формах: лекционное занятие; лабораторное занятие; самостоятельное изучение теоретического материала; самостоятельное выполнение индивидуального проекта; индивидуальные и групповые консультации.

Основной формой самостоятельной работы студента является изучение конспекта лекций, их дополнение рекомендованной литературой, выполнение проекта, а также активная работа на лабораторных занятиях.

К прослушиванию лекции следует готовиться, для этого необходимо знать программу курса и рекомендованную литературу. Тогда в процессе лекции легче отделить главное от второстепенного, легче сориентироваться: что записать, что самостоятельно проработать, что является трудным для понимания, а что легко усвоить.

Контроль за выполнением самостоятельной работы студента производится в виде контроля каждого этапа работы, отраженного в документации и защиты проекта.

Студент должен планировать график самостоятельной работы по дисциплине и придерживаться его.

VII. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Лекции проводятся с использованием проектора и внутренней системы портала ДВФУ. Лабораторные занятия проводятся в компьютерном классе.

Приложение 1

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

«Дальневосточный федеральный университет»

(ДВФУ)

ШКОЛА ЕСТЕСТВЕННЫХ НАУК

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ

по дисциплине «Функциональное и логическое программирование»

Направление подготовки – 02.03.03 «Математическое обеспечение и администрирование информационных систем»

профиль «Технология программирования»

Форма подготовки (очная)

Владивосток
2015

План-график выполнения самостоятельной работы по дисциплине

№ п/п	Дата/сроки выполнения	Вид самостоятельной работы	Примерные нормы времени на выполнение	Форма контроля
1	1-18 недели	Составление программ с использованием функциональных языков	36	Защита индивидуального задания
2	19-24 недели	Изучение одного из современных языков с элементами функциональных. Составление программ с использованием выбранного языка	18	Защита индивидуального задания
3	25-30	Составление программ с использованием логических языков	18	Защита индивидуального задания

Рекомендации по самостоятельной работе студентов

Самостоятельная работа (72 час.)

В процессе самостоятельной работы студенты выполняют задания на компьютере по программированию с использованием языковых процессоров функциональных и логических языков. Также студенты самостоятельно изучают современные языки с элементами функциональных и выполняют задания на компьютере с использованием языкового процессора одного из таких языков.

Примеры задач для самостоятельного решения с использованием функциональных языков

1. Задан текст, состоящий из символов. Текст содержит предложения. Признаком конца предложения является точка. Построить список предложений. Для каждого предложения посчитать количество слов (слова отделяются пробелом). Найти предложение, число слов в котором наибольшее. Найти в каждом предложении слово максимальное длины и построить список таких слов.

2. Отсортировать список чисел в порядке возрастания

3. Задан список студентов. Элемент списка содержит фамилию студента и результаты сдачи всех экзаменов. Посчитать для каждого студента средний бал, полученный после сдачи экзаменов. Результат представить в виде списка

4. Написать функцию, значение которой равно "истина", если в заданном списке содержится заданный элемент.

5. Написать функцию, которая по списку вида:

– фамилия

– улица

– номер дома

сформирует список соседей, живущих на заданной улице и в указанном доме

6. Написать функцию, которая строит разность двух списков одинакового типа.

7. В игре участвует два игрока. Множество карт игрока представлено списком. Элемент списка имеет следующую структуру: название масти, название карты. Также задан второй список, в котором каждой карте колоды сопоставляется "стоимость" – некоторое число. Написать функции языка HOPE, позволяющие определить, у какого игрока суммарная стоимость карт выше.

8. Покупателю требуется закупить некоторый набор товаров. У него имеется определенная сумма денег. Известна цена каждого товара. Написать функции, которые позволяли выяснить, может ли покупатель закупить весь требуемый ему товар.

9. Про каждого студента известна следующая информация: какой предмет он сдавал, каковы были результаты (оценка от 2 до 5), когда происходил экзамен или пересдача. Сформировать список тех студентов, кто уже предпринял три неудачные попытки сдачи хотя бы одного из предметов.

10. Про каждое блюдо известен набор продуктов, требуемый для его изготовления, и количество каждого продукта. Задан список продуктов, имеющихся в наличии: какой продукт, сколько. Написать функции, формирующие возможное меню. Учитывать, что суммарное количество одного и того же продукта в меню не может превышать имеющееся количество продукта.

11. Определить функцию, которая из списка выбирает все элементы – списки чисел и формирует список списков в обратном порядке.

12. Определить функцию, которая к каждому нечетному элементу исходного списка прибавляет заданное число, а затем полученный список упорядочивает по возрастанию элементов.

13. Определить функцию, которая из списка вида (((1 2) (3 4)) ((5 6) (7 8)) …) (т.е. элемент списка – список, состоящий из двух элементов, каждый из которых также список, состоящий из двух элементов) формирует список вида (…(7 8) (5 6) (3 4) (1 2)).

14. Определить функцию, которая к каждому четному элементу исходного списка прибавляет заданное число, а затем полученный список упорядочивает по возрастанию элементов.

15. Задан список производителей товаров, элементы которого имеют следующую структуру: название товара, стоимость изготовления товара, адрес производителя (город, улица). Задан список требуемых товаров. Сформировать список тех производителей каждого из требуемых товаров, у которых стоимость производства ниже, чем у других.

16. Задан список, элементами которого являются списки чисел. Сформировать список, содержащий максимальные элементы каждого из подсписков. Найти максимальный элемент из этих максимальных.

17. Использовать функции высших порядков при решении следующих задач:

- к каждому элементу списка прибавить заданное число;
- каждый элемент списка умножить на заданное число;
- по исходному списку сформировать список пар, где вторым элементом списка является заданное число.

18. Использовать функции высших порядков при решении следующих задач:

- из исходного списка сформировать список списков;
- из исходного списка сформировать список пар, где первый элемент каждой пары равен соответствующему элементу исходного списка, а второй равен числу вхождений этого элемента в исходный список;
- из исходного списка сформировать список, в котором каждый элемент равен разности между соседними элементами исходного списка;
- из исходного списка сформировать список, в котором каждый элемент равен сумме соседних элементов исходного списка.

Примеры задач для самостоятельного решения с использованием логических языков

1. Задача о миссионерах и людоедах:

На левом берегу реки n миссионеров и n людоедов, а также лодка вместимостью m мест. Необходимо перевезти миссионеров и людоедов на другой берег, так, чтобы людоеды не съели миссионеров;

2. Задача о волке, козе и капусте:
   На левом берегу волк, коза и капуста и фермер. Фермер должен перевезти всех на правый берег так, чтобы волк не съел козу, а коза не съела капусту.
3. Расставить 8 ферзей по полю 8*8 так, чтобы они не били друг друга;
4. Задача о емкостях:
   Даны три пустых емкости и бак с водой. В первую емкость входит 5 л воды, во вторую 3 л воды, а в третью 4 л. Заполнить третью емкость, используя первые две;
5. Найти в графе путь из заданной вершины в заданную вершину;
6. Написать программу синтаксического анализа выражений;
7. Написать программу вычисления значения арифметического выражения, операндами которого является числа;
8. Найти в графе цикл;
9. Выделить в графе компоненту связности.

Критерии оценки отчетов по самостоятельной работе

– 100-86 баллов выставляется, если содержание и составляющие части соответствуют выданному заданию. Продемонстрировано владение навыками подготовки документа по теме. Фактических ошибок, связанных с пониманием структуры и содержания задания нет.

– 85-76 - баллов выставляется, если при выполнении задания допущено не более одной ошибки. Продемонстрировано владение навыками подготовки документа по теме. Фактических ошибок, связанных с пониманием структуры и содержания задания нет.

– 75-61 балл выставляется, если при выполнении задания допущено не более двух ошибок. Продемонстрировано навыки подготовки документа по теме. Допущено не более 2 ошибок, связанных с пониманием структуры и содержания задания.

60-50 баллов - если структура и содержание задания не соответствуют требуемым

Шкала оценивания

Менее 60 баллов	незачтено	неудовлетворительно

От 61 до 75 баллов	зачтено	удовлетворительно
От 76 до 85 баллов	зачтено	хорошо
От 86 до 100 балов	зачтено	отлично

Приложение 2

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«Дальневосточный федеральный университет»
(ДВФУ)

ШКОЛА ЕСТЕСТВЕННЫХ НАУК

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

по дисциплине «Функциональное и логическое программирование»

Направление подготовки – 02.03.03 «Математическое обеспечение и администрирование информационных систем»

профиль «Технология программирования»

Форма подготовки (очная)

Владивосток
2015

Паспорт ФОС

Код и формулировка компетенции	Этапы формирования компетенции	
ОПК7 Способность использовать знания основных концептуальных положений функционального, логического, объектно-ориентированного и визуального направлений программирования, методов, способов и средств разработки программ в рамках этих направлений	Знает	языки функционального и логического программирования отличие семантики логических и функциональных языков от семантики алгоритмических
	Умеет	программировать на функциональных и логических языках
	Владеет	Навыками использования языковых процессоров логических и функциональных языков

№ п/п	Контролируемые разделы/темы дисциплины	Коды и этапы формирования компетенций		Оценочные средства - наименование	
				текущий контроль	промежуточная аттестация
1	Раздел 1. Функциональное программирование и функциональные языки	ОПК7	Знает	ПР1 тест	Зачет, семестр 7, вопросы № 1-15
			Умеет, владеет	ПР6 лабораторная работа ПР11 задания для самостоятельного выполнения	
2	Раздел 2. Рекурсивно-логическое программирование	ОПК7	Знает	ПР1тест	Зачет, семестр 8, вопросы № 1-16
			Умеет, владеет	ПР6 лабораторная работа ПР11 задания для самостоятельного выполнения	

Шкала оценивания уровня сформированности компетенций

Код и формулировка компетенции	Этапы формирования компетенции		критерии	показатели
ОПК7 Способность использовать знания основных концептуальных положений	знает (пороговый уровень)	языки функционального и логического программирования отличие семантики	Знание синтаксиса и семантики функциональных и логических языков	Способность дать ответы на вопросы

функционального, логического, объектно-ориентированного и визуального направлений программирования, методов, способов и средств разработки программ в рамках этих направлений		логических и функциональных языков от семантики алгоритмических		
	умеет (продвинутый)	программировать на функциональных и логических языках	Умение использовать модели данных и методы создания программ с использованием функциональных и логических языков	Наличие программ
	владеет (высокий)	Навыками использования языковых процессоров логических и функциональных языков	Владение методами создания и отладки программ средствами языковых процессоров функциональных и логических языков	Способность пояснить возможности языковых процессоров

Методические рекомендации, определяющие процедуры оценивания результатов освоения дисциплины

Текущая аттестация студентов. Текущая аттестация студентов по дисциплине «Функциональное и логическое программирование» проводится в соответствии с локальными нормативными актами ДВФУ и является обязательной.

Текущая аттестация по дисциплине «Функциональное и логическое программирование» проводится в форме контрольных мероприятий:

защиты индивидуальных заданий, тестирования.

Объектами оценивания выступают:

- учебная дисциплина (активность на занятиях, своевременность выполнения различных видов заданий, посещаемость всех видов занятий по аттестуемой дисциплине);

- степень усвоения теоретических знаний;

- уровень овладения практическими умениями и навыками по всем видам учебной работы;

- результаты самостоятельной работы.

Критерии оценки (устный ответ)

100-85 баллов - если ответ показывает прочные знания основных процессов изучаемой предметной области, отличается глубиной и полнотой раскрытия темы; владение терминологическим аппаратом; умение объяснять сущность, явлений, процессов, событий, делать выводы и обобщения, давать аргументированные ответы, приводить примеры; свободное владение монологической речью, логичность и последовательность ответа; умение приводить примеры современных проблем изучаемой области.

85-76 - баллов - ответ, обнаруживающий прочные знания основных процессов изучаемой предметной области, отличается глубиной и полнотой раскрытия темы; владение терминологическим аппаратом; умение объяснять сущность, явлений, процессов, событий, делать выводы и обобщения, давать аргументированные ответы, приводить примеры; свободное владение монологической речью, логичность и последовательность ответа. Однако допускается одна - две неточности в ответе.

75-61 - балл - оценивается ответ, свидетельствующий в основном о знании процессов изучаемой предметной области, отличающийся недостаточной глубиной и полнотой раскрытия темы; знанием основных вопросов теории; слабо сформированными навыками анализа явлений, процессов, недостаточным умением давать аргументированные ответы и приводить примеры; недостаточно свободным владением монологической речью, логичностью и последовательностью ответа. Допускается несколько ошибок в содержании ответа; неумение привести пример развития ситуации, провести связь с другими аспектами изучаемой области.

60-50 баллов - ответ, обнаруживающий незнание процессов изучаемой предметной области, отличающийся неглубоким раскрытием темы; незнанием основных вопросов теории, несформированными навыками анализа явлений, процессов; неумением давать аргументированные ответы, слабым владением монологической речью, отсутствием логичности и последовательности. Допускаются серьезные ошибки в содержании ответа; незнание современной проблематики изучаемой области.

Критерии оценки программы по лабораторным работам (проектов)

– 100-86 баллов выставляется, если содержание и составляющие части соответствуют выданному заданию. Продемонстрировано владение навыками

разработки, тестирования программ на языке программирования. Программа правильно работает на всех наборах входных данных. Текст программы содержит комментарии.

– 85-76 - баллов выставляется, если при выполнении задания допущено не более одной ошибки. Продемонстрировано владение навыками разработки программ на языке программирования. Программа правильно работает не на всех наборах входных данных (90%). Текст программы содержит комментарии.

– 75-61 балл выставляется, если при выполнении задания допущено не более двух ошибок. Продемонстрировано знание методов разработки программ на языке программирования Программа правильно работает не на всех наборах входных данных (70%). В тексте программы комментарии отсуствуют.

60- 0 баллов - если структура и содержание задания не соответствуют требуемым

Шкала оценивания

Менее 60 баллов	незачтено	неудовлетворительно
От 61 до 75 баллов	зачтено	удовлетворительно
От 76 до 85 баллов	зачтено	хорошо
От 86 до 100 балов	зачтено	отлично

Промежуточная аттестация студентов. Промежуточная аттестация студентов по дисциплине «Функциональное и логическое программирование» проводится в соответствии с локальными нормативными актами ДВФУ и является обязательной.

Промежуточная аттестация по дисциплине предусмотрена в виде зачета в устной форме (устный опрос в форме ответов на вопросы)

Критерии выставления оценки студенту на зачете (экзамене)

Баллы	**Оценка зачета/ экзамена**	**Требования к сформированным компетенциям**

86-100	«зачтено»/ «отлично»	Оценка «отлично» выставляется студенту, если он глубоко и прочно усвоил программный материал, исчерпывающе, последовательно, четко и логически стройно его излагает, умеет тесно увязывать теорию с практикой, свободно справляется с задачами, вопросами и другими видами применения знаний, причем не затрудняется с ответом при видоизменении заданий, правильно обосновывает принятое решение, владеет разносторонними навыками и приемами выполнения практических задач.
Баллы	**Оценка зачета/ экзамена** (стандартная)	**Требования к сформированным компетенциям**
76-85	«зачтено»/ «хорошо»	Оценка «хорошо» выставляется студенту, если он твердо знает материал, грамотно и по существу излагает его, не допуская существенных неточностей в ответе на вопрос, правильно применяет теоретические положения при решении практических вопросов и задач, владеет необходимыми навыками и приемами их выполнения.
61-75	«зачтено»/ «удовлетвори тельно»	Оценка «удовлетворительно» выставляется студенту, если он имеет знания только основного материала, но не усвоил его деталей, допускает неточности, недостаточно правильные формулировки, нарушения логической последовательности в изложении программного материала, испытывает затруднения при выполнении практических работ.

0-60	«не зачтено»/ «неудовлетворительно»	Оценка «неудовлетворительно» выставляется студенту, который не знает значительной части программного материала, допускает существенные ошибки, неуверенно, с большими затруднениями выполняет практические работы. Как правило, оценка «неудовлетворительно» ставится студентам, которые не могут продолжить обучение без дополнительных занятий по соответствующей дисциплине.

Оценочные средства для промежуточной аттестации

Вопросы к зачету (семестр 7)

1. Отличие функциональных языков от алгоритмических.
2. Понятие функции в функциональных языках.
3. Функциональные языки строгой и нестрогой типизации. Отличие семантики. Примеры языков.
4. Определение функции, описание функции, вызов функции.
5. Определение функции, описание функции, вызов функции.
6. Переменные в функциональных языках. Область действия переменных.
7. Типы формальных параметров в функциональных языках.
8. Работа со списками в функциональных языках.
9. Квалификационные выражения как способ сокращения записи.
10. Квалификационные выражения в функциональных языках.
11. Функции высших порядков.
12. Функционалы в функциональных языках.
13. Рекурсивные функции, правила их записи в функциональных языках.
14. Функции высших порядков. Их использование в функциональных языках.
15. Методы реализации функциональных языков.

Вопросы к зачету (семестр 8)

1. Вывод от фактов к цели (восходящий процесс) и от цели к фактам (нисходящий процесс) для логических языков.
2. Вычислительная и декларативная семантика языка Пролог.
3. Примитивные программы в Пролог.
4. Основные и универсальные факты.

5. Простой и составной вопросы.
6. Средства представления и обработки списков в Пролог.
7. Расширение языка Пролог функциями и предикатами для работы с числовыми данными.
8. Возврат и отсечение в Пролог.
9. Концепция "открытого" и "замкнутого" миров в логических языках при работе с негативными фактами.
10. Работа с негативными фактами в Пролог.
11. Логика второго порядка в Пролог.
12. Пример системы продукций. Структура состояний рабочей среды. Операции над рабочей средой.
13. Правила вывода и правила остановки. Свойства процесса вывода.
14. Методы реализации систем продукций.
15. Методы оптимизации в системе продукций.
16. Немонотонные системы продукций.

Оценочные средства для текущей аттестации

Примеры индивидуальных заданий для текущего контроля. Функциональные языки

Вариант 1.

Написать на функциональном языке функции, выполняющие указанные действия.

1. Добавление элемента в список
2. Проверка на принадлежность элемента
3. Вычисление длины списка
4. Вычисление минимального элемента списка
5. Вычисление максимального элемента списка
6. Сортировка элементов списка
7. Вычисление суммы элементов списка

Вариант 2.

Написать на функциональном языке функции, выполняющие указанные действия.

1. Вычисление разности соседних элементов списка
2. Проверка на принадлежность элемента списку разностей

3. Вычисление длины списка разностей
4. Вычисление минимальной разности
5. Вычисление произведений соседних элементов списка
6. Сортировка элементов списка произведений по убыванию
7. Сортировка элементов списка разностей по возрастанию

Вариант 3.

Написать на функциональном языке функции, выполняющие указанные действия.

1. Конкатенация двух списков
2. Инверсия списка
3. Вывод на экран элементов списка
4. Подсчет числа вхождений в список некоторого элемента.
5. Проверка, является ли один список подсписком другого списка.
6. Выделение подсписка
7. Удаление повторяющихся элементов списка с сохранением порядка.

Вариант 4

Написать на функциональном языке функции, выполняющие указанные действия.

1. Добавление элемента.
2. Удаление элемента.
3. Проверка на принадлежность.
4. Объединение.
5. Пересечение.
6. Разность.
7. Вывод на экран элементов множества в порядке возрастания

Примечание.

Множества представлять в виде списков.

Вариант 5.

Написать на функциональном языке функции, выполняющие указанные действия.

1. Проверка включения множества в другое.
2. Проверка равенства двух множеств.
3. Подсчет числа элементов множества.
4. Вывод на экран элементов множества.

5. Поиск минимального элемента множества
6. Поиск максимального элемента множества
7. Разбиение множества на классы эквивалентности (при условии что задано отношение эквивалентности двух элементов множества p(x,y)).

Примечание.
Множества представлять в виде списков.

Вариант 6.
Написать на функциональном языке функции, выполняющие указанные действия.
1. Проверка включения множества в другое.
2. Проверка равенства двух множеств.
3. Подсчет числа элементов множества.
4. Вывод на экран элементов множества.
5. Поиск минимального элемента множества
6. Поиск максимального элемента множества
7. Разбиение множества на классы эквивалентности (при условии что задано отношение эквивалентности двух элементов множества p(x,y)).

Вариант 7.
Написать на функциональном языке функции, выполняющие указанные действия.
1. Добавление элемента.
2. Удаление элемента.
3. Проверка на принадлежность.
4. Объединение.
5. Пересечение.
6. Разность.
7. Вывод на экран элементов множества в порядке возрастания

Примечание.
Для представления множеств использовать двоичные деревья.

Вариант 8.
Написать на функциональном языке функции, выполняющие указанные действия.
1. Проверка включения множества в другое.
2. Проверка равенства двух множеств.

3. Подсчет числа элементов множества.
4. Вывод на экран элементов множества.
5. Поиск минимального элемента множества
6. Поиск максимального элемента множества
7. Разбиение множества на классы эквивалентности (при условии что задано отношение эквивалентности двух элементов множества p(x,y)).

Примечание.
Для представления множеств использовать двоичные деревья.

Вариант 9
Написать на функциональном языке функции, выполняющие указанные действия.
1. Перестановка четных и нечетных элементов списка
2. Проверка на не принадлежность элемента списку
3. Сформировать из исходного списка такой, в котором не содержатся элементы, входящие в первый список более, чем один раз
4. Вычисление длины нового списка
5. Вычисление минимального элемента нового списка
6. Вычисление сумм элементов исходного и нового списка
7. Вычисление максимального элемента из двух сумм

Вариант 10.
Написать на функциональном языке функции, выполняющие указанные действия.
1. Вычисление разности соседних четных элементов списка
2. Проверка на принадлежность каждого элемента списка разностей исходному списку
3. Вычисление длины списка разностей
4. Вычисление минимальной разности
5. Вычисление произведений соседних элементов списка
6. Сортировка элементов списка произведений по убыванию
7. Сортировка элементов списка разностей по возрастанию

Вариант 11.
Написать на функциональном языке функции, выполняющие указанные действия.

1. Конкатенация двух упорядоченных по возрастанию списков так, чтобы полученный список также был упорядочен по возрастанию
2. Инверсия списка
3. Вывод на экран элементов списка
4. Сформировать для заданного списка список, элементами которого являются числа, задающие число вхождений каждого элемента исходного списка в этот список.
5. Проверка, является ли один список подсписком другого списка, причем таким, что все его элементы являются последними элементами исходного списка. Например, для двух списков [a,d,c,b,d,f,a] и [d,f,a] результат yes.
6. Выделение подсписка, первым элементом которого будет заданный элемент
7. Удаление повторяющихся элементов списка с сохранением порядка.

Вариант 12

Написать на функциональном языке функции, выполняющие указанные действия.

1. Перестановка четных и нечетных элементов списка
2. Проверка на не принадлежность элемента списку
3. Сформировать из исходного списка такой, в котором не содержатся элементы, входящие в первый список более, чем один раз
4. Вычисление длины нового списка
5. Вычисление минимального элемента нового списка
6. Вычисление сумм элементов исходного и нового списка
7. Вычисление максимального элемента из двух сумм

Для представления списков использовать двоичные деревья

Вариант 13.

Написать на функциональном языке функции, выполняющие указанные действия.

1. Вычисление разности соседних четных элементов списка
2. Проверка на принадлежность каждого элемента списка разностей исходному списку
3. Вычисление длины списка разностей
4. Вычисление минимальной разности
5. Вычисление произведений соседних элементов списка
6. Сортировка элементов списка произведений по убыванию

7. Сортировка элементов списка разностей по возрастанию

Для представления списков использовать двоичные деревья

Вариант 14.

Написать на функциональном языке функции, выполняющие указанные действия.

1. Конкатенация двух упорядоченных по возрастанию списков так, чтобы полученный список также был упорядочен по возрастанию
2. Инверсия списка
3. Вывод на экран элементов списка
4. Сформировать для заданного списка список, элементами которого являются числа, задающие число вхождений каждого элемента исходного списка в этот список.
5. Проверка, является ли один список подсписком другого списка, причем таким, что все его элементы являются последними элементами исходного списка. Например, для двух списков [a,d,c,b,d,f,a] и [d,f,a] результат yes.
6. Выделение подсписка, первым элементом которого будет заданный элемент
7. Удаление повторяющихся элементов списка с сохранением порядка.

Для представления списков использовать двоичные деревья

Вариант 15

Написать на функциональном языке функции, выполняющие указанные действия.

1. Добавление заданных элементов.
2. Удаление заданных элементов.
3. Проверка на принадлежность.
4. Объединение трех множеств.
5. Пересечение трех множеств.
6. Вывод на экран элементов множества в порядке возрастания

Примечание. Множества представлять в виде списков.

Вариант 16.

Написать функциональном языке функции, выполняющие указанные действия.

1. Для заданных трех множеств проверить, что первое является подмножеством второго, а второе – подмножеством третьего.
2. Проверка равенства двух множеств.

3. Подсчет числа положительных элементов множества.
4. Вывод на экран отрицательных элементов множества.
5. Поиск минимального положительного элемента множества
6. Поиск максимального отрицательного элемента множества

Примечание.
Множества представлять в виде списков.

Вариант 17.
Написать на функциональном языке функции, выполняющие указанные действия.
1. Для произвольных двух множеств проверить, что все положительные элементы первого множества содержатся во втором.
2. Для двух заданных множеств проверить, что все отрицательные элементы первого не содержатся во втором.
3. Подсчет числа элементов множества, больших всех элементов второго множества.
4. Вывод на экран элементов множества.
5. Поиск максимального среди тех элементов множества, которые больше заданного числа

Примечание.
Множества представлять в виде списков.

Вариант 18.
Написать на функциональном языке функции, выполняющие указанные действия.
1. Для заданных трех множеств проверить, что первое является подмножеством второго, а второе – подмножеством третьего.
2. Проверка равенства двух множеств.
3. Подсчет числа положительных элементов множества.
4. Вывод на экран отрицательных элементов множества.
5. Поиск минимального положительного элемента множества
6. Поиск максимального отрицательного элемента множества

Примечание.
Для представления множеств использовать двоичные деревья.

Вариант 19.

Написать на функциональном языке функции, выполняющие указанные действия.

1. Для произвольных двух множеств проверить, что все положительные элементы первого множества содержатся во втором.
2. Для двух заданных множеств проверить, что все отрицательные элементы первого не содержатся во втором.
3. Подсчет числа элементов множества, больших всех элементов второго множества.
4. Вывод на экран элементов множества.
5. Поиск максимального среди тех элементов множества, которые больше заданного числа

Примечание.

Для представления множеств использовать двоичные деревья.

Вариант 20.

Написать на функциональном языке функции, выполняющие указанные действия.

1. Для двух заданных множеств проверить, что каждый третий элемент первого множества принадлежит второму, а каждый второй элемент второго множества не принадлежит первому
2. Добавить ко второму множеству те четные элементы первого множества, которые больше заданного числа
3. Вычисление числа элементов полученного второго множества
4. Вычисление минимального элемента среди тех элементов первого множества, которые меньше заданного числа
5. Вычисление максимального элемента среди тех элементов второго множества, которые меньше максимального элемента первого множества
6. Вычисление суммы тех элементов, которые больше заданного числа

Множество представляется списком

Вариант 21.

Написать на функциональном языке функции, выполняющие указанные действия.

1. Вычисление разности четных элементов списка
2. Исключить из списка все положительные элементы
3. Вычисление длины полученного списка

4. Вычисление минимального среди тех элементов полученного списка, которые больше заданного числа
5. Вычисление произведений нечетных элементов списка
6. Сортировка элементов списка произведений по убыванию

Вариант 22.

Написать на функциональном языке функции, выполняющие указанные действия.

1. Конкатенация двух упорядоченных по убыванию списков так, чтобы результирующий список также был упорядочен по убыванию
2. Из исходного списка сформировать список, в котором нечетный и четный элементы поменялись местами
3. Вывод на экран элементов списка
4. Для заданного списка сформировать список, который содержит те элементы исходного списка, число вхождений которых в исходный список превышает заданного числа.
5. Проверка, является ли один список подсписком другого списка.
6. Выделение подсписка, начинающегося заданным элементом и заканчивающимся другим заданным элементом

Вариант 23.

Написать на функциональном языке функции, выполняющие указанные действия.

1. Для двух заданных множеств проверить, что каждый третий элемент первого множества принадлежит второму, а каждый второй элемент второго множества не принадлежит первому
2. Добавить ко второму множеству те четные элементы первого множества, которые больше заданного числа
3. Вычисление числа элементов полученного второго множества
4. Вычисление минимального элемента среди тех элементов первого множества, которые меньше заданного числа
5. Вычисление максимального элемента среди тех элементов второго множества, которые меньше максимального элемента первого множества
6. Вычисление суммы тех элементов, которые больше заданного числа

Множество представляется бинарным деревом

Вариант 24.

Написать на функциональном языке функции, выполняющие указанные действия.

1. Вычисление разности четных элементов списка
2. Исключить из списка все положительные элементы
3. Вычисление длины полученного списка
4. Вычисление минимального среди тех элементов полученного списка, которые больше заданного числа
5. Вычисление произведений нечетных элементов списка
6. Сортировка элементов списка произведений по убыванию

Вариант 25.

Написать на функциональном языке функции, выполняющие указанные действия.

1. Конкатенация двух упорядоченных по убыванию списков так, чтобы результирующий список также был упорядочен по убыванию
2. Из исходного списка сформировать список, в котором нечетный и четный элементы поменялись местами
3. Вывод на экран элементов списка
4. Для заданного списка сформировать список, который содержит те элементы исходного списка, число вхождений которых в исходный список превышает заданного числа.
5. Проверка, является ли один список подсписком другого списка.
6. Выделение подсписка, начинающегося заданным элементом и заканчивающимся другим заданным элементом

Вариант 26.

Написать на функциональном языке функции, выполняющие указанные действия.

1. Для двух заданных множеств проверить, что каждый нечетный элемент первого множества принадлежит второму, а каждый четный элемент второго множества не принадлежит первому
2. Добавить к первому множеству те нечетные элементы второго множества, которые больше заданного числа
3. Вычисление числа элементов полученного первого множества
4. Вычисление минимального элемента среди тех элементов первого множества, которые положительные и меньше заданного числа

5. Вычисление максимального элемента среди тех элементов второго множества, которые отрицательные и меньше максимального элемента первого множества
6. Вычисление суммы тех отрицательных элементов, которые больше заданного числа

Для представления множеств использовать списки

Вариант 27.

Написать на функциональном языке функции, выполняющие указанные действия.

1. Для двух заданных множеств проверить, что каждый нечетный элемент первого множества принадлежит второму, а каждый четный элемент второго множества не принадлежит первому
2. Добавить к первому множеству те нечетные элементы второго множества, которые больше заданного числа
3. Вычисление числа элементов полученного первого множества
4. Вычисление минимального элемента среди тех элементов первого множества, которые положительные и меньше заданного числа
5. Вычисление максимального элемента среди тех элементов второго множества, которые отрицательные и меньше максимального элемента первого множества
6. Вычисление суммы тех отрицательных элементов, которые больше заданного числа

Для представления множеств использовать бинарные деревья

Вариант 28.

Написать на функциональном языке функции, выполняющие указанные действия.

1. Вычисление суммы нечетных элементов списка
2. Исключить из списка все отрицательные элементы, большие заданного числа
3. Вычисление длины полученного списка
4. Вычисление максимального среди тех элементов полученного списка, которые меньше заданного числа
5. Вычисление произведений четных элементов списка, больших заданного числа
6. Сортировка элементов списка произведений по возрастанию

Вариант 29.

Написать на функциональном языке функции, выполняющие указанные действия.

1. Вычисление суммы нечетных элементов списка
2. Исключить из списка все отрицательные элементы, большие заданного числа
3. Вычисление длины полученного списка
4. Вычисление максимального среди тех элементов полученного списка, которые меньше заданного числа
5. Вычисление произведений четных элементов списка, больших заданного числа
6. Сортировка элементов списка произведений по возрастанию

Примеры индивидуальных заданий для текущего контроля. Логический язык

Вариант 1.

Используя стандартное представление для списков, запрограммировать на Прологе следующие предикаты работы со списками. Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Добавление элемента в список
2. Проверка на принадлежность элемента
3. Вычисление длины списка
4. Вычисление минимального элемента списка
5. Вычисление максимального элемента списка
6. Сортировка элементов списка
7. Вычисление суммы элементов списка

Вариант 2.

Используя стандартное представление для списков, запрограммировать на Прологе следующие предикаты работы со списками. Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Вычисление разности соседних элементов списка
2. Проверка на принадлежность элемента списку разностей

3. Вычисление длины списка разностей
4. Вычисление минимальной разности
5. Вычисление произведений соседних элементов списка
6. Сортировка элементов списка произведений по убыванию
7. Сортировка элементов списка разностей по возрастанию

Вариант 3.

Используя стандартное представление для списков, запрограммировать на Прологе следующие предикаты работы со списками. Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Конкатенация двух списков
2. Инверсия списка
3. Вывод на экран элементов списка
4. Подсчет числа вхождений в список некоторого элемента.
5. Проверка, является ли один список подсписком другого списка.
6. Выделение подсписка
7. Удаление повторяющихся элементов списка с сохранением порядка.

Вариант 4

Реализовать на Прологе набор предикатов работы с множествами.
Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Добавление элемента.
2. Удаление элемента.
3. Проверка на принадлежность.
4. Объединение.
5. Пересечение.
6. Разность.
7. Вывод на экран элементов множества в порядке возрастания

Примечание.
Множества представлять в виде списков.

Вариант 5.

Реализовать на Прологе набор предикатов работы с множествами.
Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Проверка включения множества в другое.
2. Проверка равенства двух множеств.
3. Подсчет числа элементов множества.
4. Вывод на экран элементов множества.
5. Поиск минимального элемента множества
6. Поиск максимального элемента множества
7. Разбиение множества на классы эквивалентности (при условии что задано отношение эквивалентности двух элементов множества p(x,y)).

Примечание.
Множества представлять в виде списков.

Вариант 6.
Реализовать на Прологе набор предикатов работы с множествами.
Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.
1. Проверка включения множества в другое.
2. Проверка равенства двух множеств.
3. Подсчет числа элементов множества.
4. Вывод на экран элементов множества.
5. Поиск минимального элемента множества
6. Поиск максимального элемента множества
7. Разбиение множества на классы эквивалентности (при условии что задано отношение эквивалентности двух элементов множества p(x,y)).

Вариант 7.
Реализовать на Прологе набор предикатов работы с множествами.
Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.
1. Добавление элемента.
2. Удаление элемента.
3. Проверка на принадлежность.
4. Объединение.
5. Пересечение.
6. Разность.
7. Вывод на экран элементов множества в порядке возрастания

Примечание.
Для представления множеств использовать двоичные деревья.

Вариант 8.

Реализовать на Прологе набор предикатов работы с множествами.

Каждый предикат должен вычислять сложность выполнения самого себя.

1. Проверка включения множества в другое.
2. Проверка равенства двух множеств.
3. Подсчет числа элементов множества.
4. Вывод на экран элементов множества.
5. Поиск минимального элемента множества
6. Поиск максимального элемента множества
7. Разбиение множества на классы эквивалентности (при условии что задано отношение эквивалентности двух элементов множества p(x,y)).

Примечание.

Для представления множеств использовать двоичные деревья.

Вариант 9

Используя стандартное представление для списков, запрограммировать на Прологе следующие предикаты работы со списками. Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Перестановка четных и нечетных элементов списка
2. Проверка на не принадлежность элемента списку
3. Сформировать из исходного списка такой, в котором не содержатся элементы, входящие в первый список более, чем один раз
4. Вычисление длины нового списка
5. Вычисление минимального элемента нового списка
6. Вычисление сумм элементов исходного и нового списка
7. Вычисление максимального элемента из двух сумм

Вариант 10.

Используя стандартное представление для списков, запрограммировать на Прологе следующие предикаты работы со списками. Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Вычисление разности соседних четных элементов списка
2. Проверка на принадлежность каждого элемента списка разностей исходному списку

3. Вычисление длины списка разностей
4. Вычисление минимальной разности
5. Вычисление произведений соседних элементов списка
6. Сортировка элементов списка произведений по убыванию
7. Сортировка элементов списка разностей по возрастанию

Вариант 11.

Используя стандартное представление для списков, запрограммировать на Прологе следующие предикаты работы со списками. Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Конкатенация двух упорядоченных по возрастанию списков так, чтобы полученный список также был упорядочен по возрастанию
2. Инверсия списка
3. Вывод на экран элементов списка
4. Сформировать для заданного списка список, элементами которого являются числа, задающие число вхождений каждого элемента исходного списка в этот список.
5. Проверка, является ли один список подсписком другого списка, причем таким, что все его элементы являются последними элементами исходного списка. Например, для двух списков [a,d,c,b,d,f,a] и [d,f,a] результат yes.
6. Выделение подсписка, первым элементом которого будет заданный элемент
7. Удаление повторяющихся элементов списка с сохранением порядка.

Вариант 12

Используя стандартное представление для списков, запрограммировать на Прологе следующие предикаты работы со списками. Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Перестановка четных и нечетных элементов списка
2. Проверка на не принадлежность элемента списку
3. Сформировать из исходного списка такой, в котором не содержатся элементы, входящие в первый список более, чем один раз
4. Вычисление длины нового списка
5. Вычисление минимального элемента нового списка
6. Вычисление сумм элементов исходного и нового списка

7. Вычисление максимального элемента из двух сумм

Для представления списков использовать двоичные деревья

Вариант 13.

Используя стандартное представление для списков, запрограммировать на Прологе следующие предикаты работы со списками. Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Вычисление разности соседних четных элементов списка
2. Проверка на принадлежность каждого элемента списка разностей исходному списку
3. Вычисление длины списка разностей
4. Вычисление минимальной разности
5. Вычисление произведений соседних элементов списка
6. Сортировка элементов списка произведений по убыванию
7. Сортировка элементов списка разностей по возрастанию

Для представления списков использовать двоичные деревья

Вариант 14.

Используя стандартное представление для списков, запрограммировать на Прологе следующие предикаты работы со списками. Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Конкатенация двух упорядоченных по возрастанию списков так, чтобы полученный список также был упорядочен по возрастанию
2. Инверсия списка
3. Вывод на экран элементов списка
4. Сформировать для заданного списка список, элементами которого являются числа, задающие число вхождений каждого элемента исходного списка в этот список.
5. Проверка, является ли один список подсписком другого списка, причем таким, что все его элементы являются последними элементами исходного списка. Например, для двух списков [a,d,c,b,d,f,a] и [d,f,a] результат yes.
6. Выделение подсписка, первым элементом которого будет заданный элемент
7. Удаление повторяющихся элементов списка с сохранением порядка.

Для представления списков использовать двоичные деревья

Вариант 15

Реализовать на Прологе набор предикатов работы с множествами.

Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Добавление заданных элементов.
2. Удаление заданных элементов.
3. Проверка на принадлежность.
4. Объединение трех множеств.
5. Пересечение трех множеств.
6. Вывод на экран элементов множества в порядке возрастания

Примечание. Множества представлять в виде списков.

Вариант 16.

Реализовать на Прологе набор предикатов работы с множествами.

Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Для заданных трех множеств проверить, что первое является подмножеством второго, а второе – подмножеством третьего.
2. Проверка равенства двух множеств.
3. Подсчет числа положительных элементов множества.
4. Вывод на экран отрицательных элементов множества.
5. Поиск минимального положительного элемента множества
6. Поиск максимального отрицательного элемента множества

Примечание.

Множества представлять в виде списков.

Вариант 17.

Реализовать на Прологе набор предикатов работы с множествами.

Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Для произвольных двух множеств проверить, что все положительные элементы первого множества содержатся во втором.
2. Для двух заданных множеств проверить, что все отрицательные элементы первого не содержатся во втором.
3. Подсчет числа элементов множества, больших всех элементов второго множества.

4. Вывод на экран элементов множества.
5. Поиск максимального среди тех элементов множества, которые больше заданного числа

Примечание.
Множества представлять в виде списков.

Вариант 18.
Реализовать на Прологе набор предикатов работы с множествами.
Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.
1. Для заданных трех множеств проверить, что первое является подмножеством второго, а второе – подмножеством третьего.
2. Проверка равенства двух множеств.
3. Подсчет числа положительных элементов множества.
4. Вывод на экран отрицательных элементов множества.
5. Поиск минимального положительного элемента множества
6. Поиск максимального отрицательного элемента множества

Примечание.
Для представления множеств использовать двоичные деревья.

Вариант 19.
Реализовать на Прологе набор предикатов работы с множествами.
Каждый предикат должен вычислять сложность выполнения самого себя.
1. Для произвольных двух множеств проверить, что все положительные элементы первого множества содержатся во втором.
2. Для двух заданных множеств проверить, что все отрицательные элементы первого не содержатся во втором.
3. Подсчет числа элементов множества, больших всех элементов второго множества.
4. Вывод на экран элементов множества.
5. Поиск максимального среди тех элементов множества, которые больше заданного числа

Примечание.
Для представления множеств использовать двоичные деревья.

Вариант 20.

Реализовать на Прологе предикаты работы с множествами
Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Для двух заданных множеств проверить, что каждый третий элемент первого множества принадлежит второму, а каждый второй элемент второго множества не принадлежит первому
2. Добавить ко второму множеству те четные элементы первого множества, которые больше заданного числа
3. Вычисление числа элементов полученного второго множества
4. Вычисление минимального элемента среди тех элементов первого множества, которые меньше заданного числа
5. Вычисление максимального элемента среди тех элементов второго множества, которые меньше максимального элемента первого множества
6. Вычисление суммы тех элементов, которые больше заданного числа

Множество представляется списком

Вариант 21.

Используя стандартное представление для списков, запрограммировать на Прологе следующие предикаты работы со списками. Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Вычисление разности четных элементов списка
2. Исключить из списка все положительные элементы
3. Вычисление длины полученного списка
4. Вычисление минимального среди тех элементов полученного списка, которые больше заданного числа
5. Вычисление произведений нечетных элементов списка
6. Сортировка элементов списка произведений по убыванию

Вариант 22.

Используя стандартное представление для списков, запрограммировать на Прологе следующие предикаты работы со списками. Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Конкатенация двух упорядоченных по убыванию списков так, чтобы результирующий список также был упорядочен по убыванию

2. Из исходного списка сформировать список, в котором нечетный и четный элементы поменялись местами
3. Вывод на экран элементов списка
4. Для заданного списка сформировать список, который содержит те элементы исходного списка, число вхождений которых в исходный список превышает заданного числа.
5. Проверка, является ли один список подсписком другого списка.
6. Выделение подсписка, начинающегося заданным элементом и заканчивающимся другим заданным элементом

Вариант 23.

Реализовать на Прологе предикаты работы с множествами
Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Для двух заданных множеств проверить, что каждый третий элемент первого множества принадлежит второму, а каждый второй элемент второго множества не принадлежит первому
2. Добавить ко второму множеству те четные элементы первого множества, которые больше заданного числа
3. Вычисление числа элементов полученного второго множества
4. Вычисление минимального элемента среди тех элементов первого множества, которые меньше заданного числа
5. Вычисление максимального элемента среди тех элементов второго множества, которые меньше максимального элемента первого множества
6. Вычисление суммы тех элементов, которые больше заданного числа

Множество представляется бинарным деревом

Вариант 24.

Используя представление списков в виде бинарного дерева, запрограммировать на Прологе следующие предикаты работы со списками. Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Вычисление разности четных элементов списка
2. Исключить из списка все положительные элементы
3. Вычисление длины полученного списка
4. Вычисление минимального среди тех элементов полученного списка, которые больше заданного числа

5. Вычисление произведений нечетных элементов списка
6. Сортировка элементов списка произведений по убыванию

Вариант 25.

Используя представление списков в виде бинарного дерева, запрограммировать на Прологе следующие предикаты работы со списками. Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Конкатенация двух упорядоченных по убыванию списков так, чтобы результирующий список также был упорядочен по убыванию
2. Из исходного списка сформировать список, в котором нечетный и четный элементы поменялись местами
3. Вывод на экран элементов списка
4. Для заданного списка сформировать список, который содержит те элементы исходного списка, число вхождений которых в исходный список превышает заданного числа.
5. Проверка, является ли один список подсписком другого списка.
6. Выделение подсписка, начинающегося заданным элементом и заканчивающимся другим заданным элементом

Вариант 26.

Реализовать на Прологе предикаты работы с множествами

Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Для двух заданных множеств проверить, что каждый нечетный элемент первого множества принадлежит второму, а каждый четный элемент второго множества не принадлежит первому
2. Добавить к первому множеству те нечетные элементы второго множества, которые больше заданного числа
3. Вычисление числа элементов полученного первого множества
4. Вычисление минимального элемента среди тех элементов первого множества, которые положительные и меньше заданного числа
5. Вычисление максимального элемента среди тех элементов второго множества, которые отрицательные и меньше максимального элемента первого множества
6. Вычисление суммы тех отрицательных элементов, которые больше заданного числа

Для представления множеств использовать списки

Вариант 27.

Реализовать на Прологе предикаты работы с множествами

Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Для двух заданных множеств проверить, что каждый нечетный элемент первого множества принадлежит второму, а каждый четный элемент второго множества не принадлежит первому
2. Добавить к первому множеству те нечетные элементы второго множества, которые больше заданного числа
3. Вычисление числа элементов полученного первого множества
4. Вычисление минимального элемента среди тех элементов первого множества, которые положительные и меньше заданного числа
5. Вычисление максимального элемента среди тех элементов второго множества, которые отрицательные и меньше максимального элемента первого множества
6. Вычисление суммы тех отрицательных элементов, которые больше заданного числа

Для представления множеств использовать бинарные деревья

Вариант 28.

Используя стандартное представление для списков, запрограммировать на Прологе следующие предикаты работы со списками. Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Вычисление суммы нечетных элементов списка
2. Исключить из списка все отрицательные элементы, большие заданного числа
3. Вычисление длины полученного списка
4. Вычисление максимального среди тех элементов полученного списка, которые меньше заданного числа
5. Вычисление произведений четных элементов списка, больших заданного числа
6. Сортировка элементов списка произведений по возрастанию

Вариант 29.

Используя стандартное представление для списков, запрограммировать на Прологе следующие предикаты работы со списками. Каждый предикат должен вычислять сложность (число шагов, необходимых для вычисления) для самого себя.

1. Вычисление суммы нечетных элементов списка
2. Исключить из списка все отрицательные элементы, большие заданного числа
3. Вычисление длины полученного списка
4. Вычисление максимального среди тех элементов полученного списка, которые меньше заданного числа
5. Вычисление произведений четных элементов списка, больших заданного числа
6. Сортировка элементов списка произведений по возрастанию

Примеры тестов для текущего контроля знаний. Логический язык

1. ПРОСТЕЙШАЯ ПРОГРАММА НА ЯЗЫКЕ ПРОЛОГ СОСТОИТ ИЗ МНОЖЕСТВА:

1) фактов
2) фактов и правил
3) фактов и вопросов
4) фактов, правил и вопросов
5) правил и вопросов

2. ПРОСТОЙ ВОПРОС В ЯЗЫКЕ ПРОЛОГ СОСТОИТ ИЗ:

1) одной цели, в которой не используются переменные
2) последовательности целей, в которых не используются переменные
3) одной цели, в которой используются переменные
4) последовательности целей, в которых используются переменные

3. СОСТАВНОЙ ВОПРОС В ЯЗЫКЕ ПРОЛОГ СОСТОИТ ИЗ:

1) одной цели, в которой не используются переменные
2) последовательности целей, в которых не используются переменные
3) одной цели, в которой используются переменные
4) последовательности целей, в которых используются переменные

4. ПРОСТОЙ ФАКТ В ЯЗЫКЕ ПРОЛОГ:

1) не содержит переменных

2) содержит переменные

5. УНИВЕРСАЛЬНЫЙ ФАКТ В ЯЗЫКЕ ПРОЛОГ:

1) не содержит переменных

2) содержит переменные

6. ОСНОВНОЙ ВОПРОС В ЯЗЫКЕ ПРОЛОГ:

1) не содержит переменных

2) содержит переменные

7. ОБЛАСТЬЮ ДЕЙСТВИЯ ПЕРЕМЕННЫХ В ПРОЛОГ ЯВЛЯЕТСЯ:

1) вся программа

2) отдельное правило или факт

8. А НАЗЫВАЕТСЯ В ПРОЛОГ ПРИМЕРОМ Б, ЕСЛИ СУЩЕСТВУЕТ ПОДСТАНОВКА ЗНАЧЕНИЙ ВМЕСТО ПЕРЕМЕННЫХ, ТАКАЯ, ЧТО:

1) А является результатом применения подстановки к Б

2) Б является результатом применения подстановки к А

9. С НАЗЫВАЕТСЯ В ПРОЛОГ ОБЩИМ ПРИМЕРОМ А И Б, ЕСЛИ СУЩЕСТВУЮТ ДВЕ ПОДСТАНОВКИ ЗНАЧЕНИЙ ВМЕСТО ПЕРЕМЕННЫХ, ТАКИЕ, ЧТО:

1) С синтаксически совпадает с результатом применения первой подстановки к А и С синтаксически совпадает с результатом применения второй подстановки к Б

2) А синтаксически совпадает с результатом применения первой подстановки к С и Б синтаксически совпадает с результатом применения второй подстановки к С

10. РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ НА ЯЗЫКЕ ПРОЛОГ ОТ ПОРЯДКА ЗАПИСИ ПРАВИЛ В ПРОГРАММЕ:

1) не зависит

2) зависит

11. РЕЗУЛЬТАТ ПОЛУЧЕНИЯ ОТВЕТА НА ВОПРОС В ЯЗЫКЕ ПРОЛОГ ОТ ПОРЯДКА ЦЕЛЕЙ В ВОПРОСЕ:

1) не зависит

2) зависит

11. ПРИ ПОИСКЕ ОТВЕТА НА ВОПРОС В ПРОЛОГ ПРОИСХОДИТ ПОИСК:

1) в глубину

2) в ширину

12. В ПРОЛОГ ОТРИЦАНИЕ РАССМАТРИВАЕТСЯ КАК:

1) неудача

2) отсутствие

13. В ЯЗЫКЕ ПРОЛОГ ЗАПРЕЩЕНА РЕКУРСИЯ:

1) левая

2) правая

3) любая

14.ОПЕРАТОР ! В ПРОЛОГ ОБОЗНАЧАЕТ:

1) отсечение

2) ответвление

3) откат

15. ДЛЯ ПРИВЕДЕННОЙ ПРОГРАММЫ НА ЯЗЫКЕ ПРОЛОГ
РОДИТЕЛЬ(ИВАН, ПЕТР).
ПРЕДОК(X, Z) :- ПРЕДОК(У, Z), РОДИТЕЛЬ(X,У).
ПРИ ПОИСКЕ ОТВЕТА НА ВОПРОС ПРЕДОК (ИВАН, X) БУДЕТ

1) получен бесконечный цикл

2) найден ответ

Примеры тестов для текущего контроля знаний. Функциональные языки

ОБВЕДИТЕ КРУЖКОМ НОМЕР ПРАВИЛЬНОГО ОТВЕТА:

1. ЯЗЫК ФУНКЦИОНАЛЬНОГО ПРОГРАММИРОВАНИЯ, ЯВЛЯЮЩИЙСЯ ЯЗЫКОМ СТРОГОЙ ТИПИЗАЦИИ

1) Норе
2) Лисп

2. ЯЗЫК ФУНКЦИОНАЛЬНОГО ПРОГРАММИРОВАНИЯ, ЯВЛЯЮЩИЙСЯ ЯЗЫКОМ НЕСТРОГОЙ ТИПИЗАЦИИ
1) Норе
2) Лисп

3. ФУНКЦИОНАЛЬНЫЕ ЯЗЫКИ ОСНОВАНЫ НА ИСЧИСЛЕНИИ
1) ламбда
2) предикатов
3) функций
4) высказываний

ОБВЕДИТЕ КРУЖКОМ НОМЕРА ВСЕХ ПРАВИЛЬНЫХ ОТВЕТОВ:

4.ФУНКЦИОНАЛЬНЫЕ ЯЗЫКИ
1) Норе
2) Лисп
3) Пролог
4) Паскаль
5) Симула

5. ЯЗЫК НОРЕ СРЕДСТВА ДЛЯ ОПИСАНИЯ ТИПОВ ДАННЫХ ПОЛЬЗОВАТЕЛЕМ
1) имеет
2) не имеет

6. КВАЛИФИКАЦИОННЫЕ ВЫРАЖЕНИЯ В НОРЕ СЛУЖАТ ДЛЯ
1) задания обозначений
2) задания имени функции
3) записи вызова функции

ОБВЕДИТЕ КРУЖКОМ НОМЕРА ВСЕХ ПРАВИЛЬНЫХ ОТВЕТОВ:

6. ЯЗЫК НОРЕ ПОЗВОЛЯЕТ ИСПОЛЬЗОВАТЬ В ВЫРАЖЕНИЯХ ЗАПИСЬ
1) инфиксную

2) постфиксную
3) префиксную

ОБВЕДИТЕ КРУЖКОМ НОМЕР ПРАВИЛЬНОГО ОТВЕТА:

7. ЯЗЫК ЛИСП ПОЗВОЛЯЕТ ИСПОЛЬЗОВАТЬ В ВЫРАЖЕНИЯХ ЗАПИСЬ
1) инфиксную
2) постфиксную
3) префиксную

8. ЗАПИСЬ DATA NUMLIST = = NIL ++ CONS(NUM # NUMLIST) НА НОРЕ ОЗНАЧАЕТ ОПРЕДЕЛЕНИЕ
1) типа списка, который может быть пустым, либо состоять из целых чисел
2) конструктора типа

9.ЗАПИСЬ DATA LIST(ALPHA) = = NIL ++ CONS(ALPHA # LIST(ALPHA)) НА НОРЕ ОЗНАЧАЕТ ОПРЕДЕЛЕНИЕ
1) типа списка, который может быть пустым, либо состоять из элементов произвольного типа
2) типа списка, который может быть пустым, либо состоять из элементов произвольного типа, причем все элементы списка имеют одинаковый тип
3) конструктора типа

10. ЯЗЫК НОРЕ ПОЛИМОРФИЗМ ОПЕРАЦИЙ
1) допускает
2) не допускает

11. ЗАПИСЬ INFIX :: : 7 ОПРЕДЕЛЯЕТ
1)инфиксную операцию
2) инфиксную операцию и ее приоритет
3) функцию INFIX

12. ЗАПИСЬ ВИДА TYPE COORDINATE = = REAL # REAL НА НОРЕ ОЗНАЧАЕТ ОПРЕДЕЛЕНИЕ
1) структуры, состоящей из двух вещественных чисел
2) конструктора типа
3) списка, элементами которого являются структурные объекты

13. ЗАПИСЬ ВИДА DEC JOIN: (LIST(ALPHA) # LIST(ALPHA) -> LIST(ALPHA) НА НОРЕ ОЗНАЧАЕТ
1) определение функции Join
2) объявление функции Join
3) вызов функции Join

14. ЗАПИСЬ ВИДА --- JOIN (NIL, L) <= L; --- JOIN (X::Y,L) <= X:: JOIN(Y,L) НА НОРЕ ОЗНАЧАЕТ
1) определение функции Join
2) объявление функции Join
3) вызов функции Join

15. ЗАПИСЬ ВИДА --- JOIN (X::Y,L) <= X:: JOIN(Y,L) ЗАДАЕТ
1) рекурсивное определение функции Join
2) объявление функции Join
3) вызов функции Join

16. ЗАПИСЬ ВИДА LAMBDA X ==> X+1 ЗАДАЕТ
1) оператор увеличения значения переменной x на 1
2) определение безымянной функции

17. ЗАПИСЬ ВИДА DEC MAP: (ALPHA -> BETA) # LIST(ALPHA) -> LIST(BETA) НА НОРЕ ОЗНАЧАЕТ
1) определение простой функции map
2) объявление функции высшего порядка map
3) определение функции высшего порядка map
4) объявление простой функции map

18. ЕСЛИ ЗАДАН ОПЕРАТОР ЯЗЫКА НОРЕ LET X = =1 IN (LET F = = LAMBDA Y => X+Y IN (LET X = = 2 IN F(X))), ТО ПЕРЕМЕННАЯ X В ЗАПИСИ X+Y ИМЕЕТ ЗНАЧЕНИЕ, РАВНОЕ
1) 2
2) 1
3) определенное вызовом функции f(x)

19. В ЯЗЫКЕ НОРЕ СВЯЗЫВАНИЕ ПЕРЕМЕННЫХ В ТЕЛЕ ФУНКЦИИ

1) статическое
2) динамическое

20. В ЯЗЫКЕ ЛИСП СВЯЗЫВАНИЕ ПЕРЕМЕННЫХ В ТЕЛЕ ФУНКЦИИ
1) статическое
2) динамическое

21. ФУНКЦИИ ВЫСШИХ ПОРЯДКОВ – ЭТО ФУНКЦИИ, ПРИ ОБЪЯВЛЕНИИ КОТОРЫХ
1) аргумент или результат является функцией
2) аргумент или результат содержит вызов этой же функции

ОБВЕДИТЕ КРУЖКОМ НОМЕРА ВСЕХ ПРАВИЛЬНЫХ ОТВЕТОВ:

22. ОСНОВНЫМИ ТИПАМИ ДАННЫХ ЯЗЫКА ЛИСП ЯВЛЯЮТСЯ
1) атомы
2) списки
3) массивы
4) структуры
5) указатели

23. ПАРАМЕТРЫ ФУНКЦИИ В ЛИСП МОГУТ БЫТЬ
1) необязательными
2) ключевыми
3) вспомогательными
4) позиционными
5) пропозициональными

ОБВЕДИТЕ КРУЖКОМ НОМЕР ПРАВИЛЬНОГО ОТВЕТА:

24. ЛОГИЧЕСКИЕ ЗНАЧЕНИЯ В ЛИСП
1) T и Nil
2) true и false

25. ЯЗЫК ЛИСП ЗАДАНИЕ ФУНКЦИЙ, ИМЕЮЩИХ ПЕРЕМЕННОЕ ЧИСЛО ПАРАМЕТРОВ
1) допускает

2) не допускает

26. СЛЕДУЮЩАЯ ФОРМА НА ЯЗЫКЕ ЛИСП
(DEFUN F (X Y) (CONS X (CONS Y NIL))) ЗАДАЕТ
1) определение безымянной функции
2) определение функции, имеющей имя
3) объявление безымянной функции
4) объявление функции, имеющей имя

27. В ЗАДАННОЙ ФОРМЕ (DEFUN FN X &OPTIONAL (Y (+ X 2)))
ФУНКЦИЯ FN ИМЕЕТ
1) один позиционный и один не обязательный параметры
2) один позиционный и один вспомогательный параметры
3) один позиционный и один ключевой параметры

28. В ЗАДАННОЙ ФОРМЕ НА ЯЗЫКЕ ЛИСП (DEFUN FN &KEY (X Y (Z 3)))
ФУНКЦИЯ FN ИМЕЕТ
1) два ключевых параметра и один не обязательный параметр
2) три ключевых параметра, причем при вызове функции значение параметра z может быть не задано
3) два ключевых параметра и один вспомогательный параметр
4) переменное число параметров

29. В ЗАДАННОЙ ФОРМЕ НА ЯЗЫКЕ ЛИСП (LAMBDA (X Y) (+ (* X X) (* Y Y))) ЗАДАНО
1) определение безымянной функции
2) определение тела некоторой функции
3) объявление безымянной функции

30. СЛЕДУЮЩИЕ ДВЕ ФОРМЫ НА ЯЗЫКЕ ЛИСП (LAMBDA (X Y) (CONS Y NIL))) И (LAMBDA (CAT DOG) (CONS CAT DOG))) ЗАДАЮТ
ОПРЕДЕЛЕНИЕ
1) одной и той же безымянной функции
2) двух разных безымянных функций

31. СЛЕДУЮЩАЯ ФОРМА НА ЯЗЫКЕ ЛИСП ((LAMBDA (X Y) (+ X Y)) 2 3)
ЗАДАЕТ

1) вызов безымянной функции
2) ламбда-выражение
3) определение безымянной функции

32. РЕЗУЛЬТАТОМ СЛЕДУЮЩЕЙ ФОРМЫ НА ЯЗЫКЕ ЛИСП ((LAMBDA (Y) ((LAMBDA (X) (LIST X Y)) 'ONE)) 'TWO) ЯВЛЯЕТСЯ:
1) список (one two)
2) список (two one)
3) список (one one)
4) список (two two)

33. РЕЗУЛЬТАТОМ СЛЕДУЮЩЕЙ ФОРМЫ НА ЯЗЫКЕ ЛИСП ((LAMBDA (X) (LIST X 'SECOND)) ((LAMBDA (Y) (LIST Y)) 'FIRST)) ЯВЛЯЕТСЯ:
1) список ((first) second)
2) список ((second) first)
3) список (first second))
4) список (second first)
5) список ((first) (second))
6) список ((second) (first))
7) список (first (second))
8) список (second (first))

34. В РЕЗУЛЬТАТЕ ВЫПОЛНЕНИЯ ПОСЛЕДНЕЙ ФОРМЫ СЛЕДУЮЩЕЙ ПРОГРАММЫ НА ЯЗЫКЕ ЛИСП (SETQ X 2) (QUOTE (EVAL X)) БУДЕТ ПОЛУЧЕН РЕЗУЛЬТАТ
1) 2
2) (eval x)

35. В РЕЗУЛЬТАТЕ ВЫПОЛНЕНИЯ ПОСЛЕДОВАТЕЛЬНОСТИ ФОРМ ЯЗЫКА ЛИСП (SETQ X 'Y) (SET X 2) (SET 'X 3) ПЕРЕМЕННЫЕ X И Y ИМЕЮТ СЛЕДУЮЩИЕ ЗНАЧЕНИЯ:
1) x = 3, y = 2
2) x = y, y = 2
3) x = 2, y = 3

36. В РЕЗУЛЬТАТЕ ВЫПОЛНЕНИЯ ПОСЛЕДОВАТЕЛЬНОСТИ ФОРМ ЯЗЫКА ЛИСП (SET 'Y 5) (SETQ X 'Y) (SETQ X 2) (SET 'X 3) ПЕРЕМЕННЫЕ X И Y ИМЕЮТ СЛЕДУЮЩИЕ ЗНАЧЕНИЯ:

1) x = 3, y = 5
2) x = y, y = 5
3) x = 2, y = 3
4) x = 3, y = 2

37. РЕЗУЛЬТАТОМ ВЫПОЛНЕНИЯ ФОРМЫ ЯЗЫКА ЛИСП ((LAMBDA (A &OPTIONAL (B 2)) (+ A (* B 3))) 4 5) БУДЕТ

1) 19
2) 10

38. РЕЗУЛЬТАТОМ ВЫПОЛНЕНИЯ ФОРМЫ ЯЗЫКА ЛИСП ((LAMBDA (A B &KEY C D) (LIST A B C D)) :A 1 :D 8 :C 6) БУДЕТ СПИСОК

1) (:a 1 6 8)
2) (1 nil 6 8)

39. РЕЗУЛЬТАТОМ ВЫПОЛНЕНИЯ ФОРМЫ ЯЗЫКА ЛИСП ((LAMBDA (A &OPTIONAL (B 3) & REST X &KEY C (D A)) (LIST A B C D X)) 1) БУДЕТ СПИСОК

1) (1 3 nil 1 nil)
2) (1 3 nil nil nil)
3) (nil 1 nil nil nil)
4) (1 nil nil 1 nil)

40. РЕЗУЛЬТАТОМ ВЫПОЛНЕНИЯ ФОРМЫ ЯЗЫКА ЛИСП ((LAMBDA (A &OPTIONAL (B 3) & REST X &KEY C (D A)) (LIST A B C D X)) 1 6 :C 7) БУДЕТ СПИСОК

1) (1 6 7 1 nil)
2) (1 6 7 1 (:c 7))

41. В РЕЗУЛЬТАТЕ ВЫПОЛНЕНИЯ ПОСЛЕДОВАТЕЛЬНОСТИ ФОРМ ЯЗЫКА ЛИСП (DEFUN F (X) (SETQ X 'NEW)) (SETQ X 'OLD) (F 'NEW) ПЕРЕМЕННАЯ X ИМЕЕТ ЗНАЧЕНИЕ:

1) new

2) old

42. В РЕЗУЛЬТАТЕ ВЫПОЛНЕНИЯ ПОСЛЕДОВАТЕЛЬНОСТИ ФОРМ ЯЗЫКА ЛИСП (SETQ X 2) (LET ((X 0)) (SETQ X 1)) ПЕРЕМЕННАЯ X ИМЕЕТ ЗНАЧЕНИЕ:
1) 1
2) 2

УПОРЯДОЧИТЕ:

43. В РЕЗУЛЬТАТЕ ВЫПОЛНЕНИЯ ФОРМЫ (LET ((М1 ЗНАЧ1) (М2 ЗНАЧ2)) ФОРМА1, ФОРМА 2)
1) вычисляются значения форм форма1 и форма2,
2) одновременно связываются переменные м1 и м2 со значениями знач1 и знач2,
3) вычисляются значения знач1 и знач 2,

3,2,1

44. В РЕЗУЛЬТАТЕ ВЫПОЛНЕНИЯ ФОРМЫ (LET* ((М1 ЗНАЧ1) (М2 ЗНАЧ2)) ФОРМА1, ФОРМА 2)
1) вычисляются значения форм форма1 и форма2
2) переменная м1 связывается со значением знач1
3) вычисляется значение знач1
4) переменная м2 связывается со значением знач2
5) вычисляется значение знач2,

3,2,5,4,1

ОБВЕДИТЕ КРУЖКОМ НОМЕР ПРАВИЛЬНОГО ОТВЕТА:

45. РЕЗУЛЬТАТОМ ФУНКЦИИ ЯЗЫКА ЛИСП (CADDR (LIST 1 (2 3 4) (5))) ЯВЛЯЕТСЯ:
1) 1
2) 2
d) 5

46. ФУНКЦИОНАЛЬНЫЙ АРГУМЕНТ ОПРЕДЕЛЕН В ФУНКЦИИ ЯЗЫКА ЛИСП:

1) (defun f … (g …) …)
2) (defun f … (f…) …)
3) (defun f … (f …(f…)…)…)
4) (defun f (… g…)…(apply g …)…)

47. РЕКУРСИВНЫЙ ФУНКЦИОНАЛЬНЫЙ АРГУМЕНТ ОПРЕДЕЛЕН В ФУНКЦИИ ЯЗЫКА ЛИСП:

1) (defun f … (g …) …)
2) (defun f … (f…) …)
3) (defun f … (f …(f…)…)…)
4) (defun f (… g…)…(apply g …)…)
5) (defun f (… f…)…(apply f …)…)

48. РЕЗУЛЬТАТОМ ВЫЗОВА ФУНКЦИОНАЛА ЯЗЫКА ЛИСП (SETQ X '(A B C)) (MAPCAR 'ATOM X) БУДЕТ СПИСОК:

1) (T T T)
2) (nil nil nil)
3) (nil T T)