МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный университет имени М.В.Ломоносова»

«Утверждаю»

Декан факультета ВМК МГУ
имени М.В. Ломоносова

академик _____ Е. И. Моисеев

«___»_____ 2017 г.

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ

«Введение в функциональное программирование»

Уровень высшего образования – подготовка научно-педагогических кадров в аспирантуре

Направление подготовки–09.06.01 «Информатика и вычислительная техника»

Направленность (профиль)–«Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей» (05.13.11)

2017 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

1. НАИМЕНОВАНИЕ ДИСЦИПЛИНЫ

Введение в функциональное программирование

2. УРОВЕНЬ ВЫСШЕГО ОБРАЗОВАНИЯ

Подготовка научно-педагогических кадров в аспирантуре.

3. НАПРАВЛЕНИЕ ПОДГОТОВКИ, НАПРАВЛЕННОСТЬ (ПРОФИЛЬ) ПОДГОТОВКИ

Направление 09.06.01 «Информатика и вычислительная техника». Направленность (профиль) «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей» (05.13.11).

4. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОСНОВНОЙ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Дисциплина относитсяк специальным дисциплинам вариативной части образовательной программы.

5.ПЕРЕЧЕНЬ ПЛАНИРУЕМЫХ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ

Дисциплинаучаствуетв формировании следующих компетенцийобразовательной программы:

Формируемые компетенции	Планируемые результаты обучения
ПК-1 Владение современными методами построения и анализа математических моделей, возникающих при решении естественнонаучных задач, а также методами разработки и реализации алгоритмов их решения на основе фундаментальных знаний в области математики и информатики	З1 (ПК-1) Знать: современные методы построения и анализа математических моделей, возникающих при решении задач обеспечения пользовательского интерфейса, а также современные методы разработки и реализации алгоритмов их решения У1 (ПК-1) Уметь: применять современные методы построения и анализа математических моделей, возникающих при решении задач обеспечения пользовательского интерфейса, а также современные методы разработки и реализации алгоритмов их решения В1 (ПК-1) Владеть:

	навыками оптимального выбора современных методов построения и анализа математических моделей, возникающих при решении задач обеспечения человеко-машинного интерфейса, а также современными методами разработки и реализации алгоритмов их решения
Владение современными методами разработки программного обеспечения вычислительных комплексов (ПК-3).	Знать Современные методы разработки программного обеспечения вычислительных комплектов Владеть Современными методами разработки программного обеспечения вычислительных комплексов Уметь Разрабатывать программное обеспечение для вычислительных комплексов
Способность самостоятельно осуществлять научно-исследовательскую деятельность в соответствующей профессиональной области с использованием современных методов исследования и информационно-коммуникационных технологий (ОПК-1)	З1(ОПК-1) ЗНАТЬ: современные методы исследования и информационно-коммуникационные технологии в профессиональной области построения эффективного и удобного для человека пользовательского интерфейса У1(ОПК-1) УМЕТЬ: уметь самостоятельно осуществлять научно-исследовательскую деятельность в профессиональной области построения эффективного и удобного для человека пользовательского интерфейса с использованием современных методов исследования и информационно-коммуникационных технологий

Оценочные средства для промежуточной аттестации приведены в Приложении.

6. ОБЪЕМ ДИСЦИПЛИНЫ

Объем дисциплины составляет 3 зачетных единицы, всего 108 часов.

44 часа составляет контактная работа с преподавателем – 36 часов занятий лекционного типа, 0 часов занятий семинарского типа (семинары, научно-практические занятия, лабораторные работы и т.п.), 0 часов индивидуальных консультаций, 2 часа мероприятий текущего контроля успеваемости, 4 часа групповых консультаций, 2 часа мероприятий промежуточной аттестации.

64 часа составляет самостоятельная работа аспиранта.

7. ВХОДНЫЕ ТРЕБОВАНИЯ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Учащиеся должны владеть знаниями поалгоритмам, алгоритмическим языкам и программированию в объеме, соответствующемосновным образовательным программам бакалавриата и магистратуры по укрупненным группам направлений и специальностей 01.00.00 «Математика и механика», 02.00.00 «Компьютерные и информационные науки».

8. ОБРАЗОВАТЕЛЬНЫЕ ТЕХНОЛОГИИ

В процессе обучения каждое лекционное занятие начинается с проведения анкетирования. Аспиранты заполняют анкеты с вопросами и/или задачами по материалу предыдущих лекций. При изложении материала лекций предполагается диалог со слушателями. На каждой лекции выделяется время для ответов на вопросы по текущему материалу и его обсуждения. Материалы лекций демонстрируются аспирантам в виде презентаций, сопровождаемых комментариями лектора. По ходу чтения слайды прочитанных лекций выкладываются в специально созданной учебной онлайновой группе в социальной сети Вконтакте. Каждая прочитанная лекция может быть обсуждена в учебной онлайновой группе. Дополнительно каждый аспирант может дистанционно получить разъяснения преподавателя по электронной почте.Чтобы стимулировать творческую активность аспирантов и повысить их интерес, в рамках задания «Доктор» предложено творческое упражнение №8, успешное выполнение которого приносит дополнительные баллы.

9. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

Курс посвящён знакомству слушателей с парадигмой функционального программирования на примере языка Scheme. В рамках курса рассматриваются как базовые средства языка, относящиеся к «чистому» функциональному программированию, так и дополнительные его возможности: мутируемые структуры данных, макросы, потоки, средства объектно-ориентированного программирования. Также даётся короткий обзор математических основ функционального программирования.

Наименование и краткое содержа-	**Всего**	В том числе

ние разделов и тем дисциплины (модуля), форма промежуточной аттестации по дисциплине (модулю)	**(часы)**	**Контактная работа (работа во взаимодействии с преподавателем), часы** из них						**Самостоятельная работа обучающегося, часы** из них		
		Занятия лекционного типа	Занятия семинарского типа	Групповые консультации	Индивидуальные консультации	Учебные занятия, направленные на проведение текущего контроля успеваемости (коллоквиумы, практические контрольные занятия и др)*	**Всего**	Выполнение домашних заданий	Подготовка рефратов и т. п..	**Всего**
Раздел 1. ***Базовые средства языка функционального программирования Scheme***	36	16	0	2	0	2	20	16	0	16
Тема 1.1. ***Основные сведения о языке Scheme*** (основные его отличия от императивных языков программирования; основные его отличия от диалектов языка Лисп; имена, связывания, окружения; классификация выражений в Scheme; внешнее представление значений; именованные функции и специальная форма `define`; подстановочная модель вычислений; аппликативный и нормальный порядки вычислений; анонимные функции и специальная форма `lambda`; специальные формы: `cond`, `if`, `case`, `and`, `or`, `begin`, `let`, `let*`, `quote`; базовые типы данных:	6	4	0	0	0	0	4	2	0	2

числа, литеры, строки, списки; блоки и блочная структура программы).										
Тема 1.2. ***Рекурсия и итерация, рекурсивные и итеративные процессы***(функции и процессы, которые они порождают; характеристики сложности процесса: количество шагов и размер памяти; хвостовая рекурсия; «именованный» `let` – средство реализации итеративных процессов; примеры реализации функций, порождающих рекурсивные и итеративные процессы).	5	3	0	0	0	0	3	2	0	2
Тема 1.3. ***Функции высшего порядка***(функции в роли аргументов других функций; функции как обобщённые схемы вычислений при обработке списков: `map`, `filter`, `foldl`, `foldr`, `andmap`, `ormap`, `filter-map`, `count`, `argmin`, `argmax`, `append-map`, `filter-not`; функции как возвращаемые значения).	5	3	0	0	0	0	3	2	0	2
Тема 1.4. ***Структуры данных, их использование и реализация в программах на языке Scheme*** (проектирование структур данных; типовые операции структуры данных: конструкторы, селекторы, чеккеры; по-	5	3	0	0	0	0	3	2	0	2

строение слоистых систем с помощью структур данных; точечные пары – основа реализации структур данных; функции работы с точечными парами: `cons`, `car`, `cdr`; вектора и их использование при реализации структур данных; стрелочные диаграммы; деревья и бинарные деревья поиска в Scheme).										
Тема 1.5. ***Программирование в стиле передачи остаточных вычислений***(остаточные вычисления; управление порядком вычисления выражения при помощи явного выписывания остаточных вычислений в виде анонимной функции;преобразование программы, порождающей рекурсивный процесс, в программу с явной передачей остаточных вычислений; примеры программ, составленных в стиле передачи остаточных вычислений).	5	3	0	0	0	0	3	2	0	2
1.6. Текущий контроль успеваемости:*письменная контрольная работа*	10	0	0	2	0	2	4	6		
Раздел 2. ***Дополнительные возможности языка программирования Scheme***	24	16	0	0	0	0	16	8	0	8
Тема 2.1. ***Присваивание и модель вычислений с окружениями*** (специаль-	6	4	0	0	0	0	4	2	0	2

ная форма `set!`; преимущества и издержки присваивания; мутируемые точечные пары и мутируемые векторы; основные понятия модели вычислений с окружениями: связывание, кадр, окружение; правила вычисления выражений в модели с окружениями; стрелочные диаграммы окружений; реализация в Scheme стеков, очередей, таблиц; мемоизация; реализация хеш-таблиц в Scheme).										
Тема 2.2. ***Макросы в языке Scheme*** (система макросов `syntax-rules`; основания для использования макросов в программе; характеристики системы `syntax-rules`: гигиеничность, прозрачность ссылок, закрытость; язык образцов системы `syntax-rules`; язык шаблонов системы `syntax-rules`; спецсимволы в образцах; реализация пользовательских специальных форм макросами).	5	3	0	0	0	0	3	2	0	2
Тема 2.3. ***Программирование с потоками*** (ленивые вычисления, строгость / нестрогость функции по параметру; санки и функции работы с санками: `delay` и `force`; мемоизациясанков; реализация потоков как	5	3	0	0	0	0	3	2	0	2

«ленивых списков»; функции работы с потоками: `stream-cons`, `stream-first`, `stream-rest`, `stream`, `stream*`, `stream-empty?`										
Тема 2.4. ***Объектно-ориентированное программирование в Scheme*** (обобщённые операции; объекты данных как альтернатива обобщённым операциям; базовые понятия ООП: класс, экземпляр класса, механизм передачи сообщений, наследование, множественное наследование, ассоциация; Три точки зрения на ООП: модель, использование, реализация; UML-диаграммы классов и UML-диаграммы объектов; объектное расширение Scheme и его элементы: функции `create-instance`, `ask`, `get-method` и схема реализации обработчика сообщений; структура экземпляра класса, `self`.	8	6	0	0	0	0	6	2	0	2
Раздел 3. ***Математические основы функционального программирования***	8	4	0	0	0	0	4	0	4	4
Тема 3.1. ***Основные сведения о λ-исчислении*** (λ-нотация; классическое λ-исчисление; λ-выражения; свободные и связанные переменные; подстановки; β-редукция и α-редукция; экви-	8	4	0	0	0	0	4	0	4	4

валентность и λ-редукция; нормальная форма; стратегии редукции при поиске нормальной формы; теорема Черча-Россера и её следствия; комбинаторы: I, S, K, B, S, W, Y).										
4. Промежуточная аттестация: ***письменная экзаменационная работа***	40	0	0	2	0	2	4	36		
Итого	108	36	0	4	0	4	44	64		

10. УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ УЧАЩИХСЯ

Самостоятельная работа учащихся состоит в изучении лекционного материала, учебно-методической литературы, выполнения домашних заданий, выполнения самостоятельных практических заданий, составления отчёта по второму практическому заданию, подготовки к текущему контролю и к промежуточной аттестации:

• Домашнее задание по теме 1.1: упражнения №№ 1.1-1.8 из книги [Абельсон, Сассман и др. 2006]; упражнения №№ 1 и 2 практического задания «Доктор».

• Домашнее задание по теме 1.2: упражнения №№ 1.9-1.20 из книги [Абельсон, Сассман и др. 2006]; упражнения №№ 3 и 4 практического задания «Доктор».

• Домашнее задание по теме 1.3: упражнения №№ 1.30-1.43, 2.33-2.39 из книги [Абельсон, Сассман и др. 2006]; упражнения №№ 5 и 6 практического задания «Доктор».

• Домашнее задание по теме 1.4: упражнения №№ 2.1-2.30 из книги [Абельсон, Сассман и др. 2006]; упражнение № 7 практического задания «Доктор»).

• Домашнее задание по теме 1.5: упражнение № 8 практического задания «Доктор».

• Подготовка к письменной контрольной работе.

• Домашнее задание по теме 2.1: упражнения 3.1-3.20 из книги [Абельсон, Сассман и др. 2006]; упражнение № 8 практического задания «Доктор».

• Домашнее задание по теме 2.2: упражнение № 8 практического задания «Доктор».

• Домашнее задание по теме 2.3: упражнения 3.50-3.57 из книги [Абельсон, Сассман и др. 2006]; упражнение № 8 практического задания «Доктор».

• Домашнее задание по теме 2.4: второе практическое задание «Создание игровой программы на основе минимаксного алгоритма».
• Подготовка отчёта по второму практическому заданию.
• Подготовка к письменной экзаменационной работе.

Указания по выполнению упражнений первого практического задания «Доктор»

«Доктор» (или ELIZA) — это название программы, созданной Джозефом Вейценбаумом. Эта программа имитирует (или пародирует) психоаналитика, ведущего диалог с пациентом. Программа принимает реплики пациента (в виде списков символов) и генерирует ответные реплики (также в виде списков). Для простоты сокращена пунктуация в записи реплик.

В рамках выполнения практического задания Вы создадите собственную версию «Доктора», которая основана на некоторых (но не всех!) идеях, лежащих в основе исходной программы.

Упражнения по «Доктору» делятся на 4 блока:

1-й блок — упражнения с 1 по 4.

2-й блок — упражнения 5 и 6.

3-й блок — упражнение 7.

4-й блок — дополнительный.

Версии программы 1-3 блоков следует составлять на подмножестве языка scheme/base (начинайте свой код с директивы #langscheme/base). В них использование мутаторов (присваиваний и т. п.), мутируемых структур данных Racket, запрещено. Составленная в ходе выполнения упражнений, программа сдаётся. При переходе от начальных упражнений к последующим код следует дописывать так, чтобы функциональность программы расширялась (то, что было раньше, не портить). Не следует сдавать несколько разных версий программы, каждая из которых решает только одно из упражнений.

Предполагается, что прошлые годы учёбы привили Вам навык оформления кода. Сдаваемый код должен быть оформлен должным образом: быть читаемым, сопровождаться Вашими комментариями на русском языке.

Скачайте заготовку кода «Доктора» со страницы курса. Найдите дистрибутив среды DrRacket и установите её себе на компьютер. Запустите IDE DrRacket и откройте заготовку кода «Доктора». Запустите код в интерпретаторе (Ctrl+R или кнопкой Run). Проверьте, как он работает.

Упражнение 1. Измените функции qualifier-answer и hedge, добавив в каждую не менее трёх новых заготовленных фраз-реплик и/или фраз, с которых начинается ответ с заменой лица.

Упражнение 2. Напишите новую версию функции many-replace с хвостовой рекурсией и вызовите её в теле change-person. Составьте код нового many-replace без определения локальной вспомогательной функции, а с использованием "именованного" let. Далее всякий раз, когда для реализации итеративного процесса понадобится локальная вспомогательная функция, используйте вместо неё "именованный" let. Новая вер-

сия функции должна быть эффективной. Если Вы планируете использовать append для "сборки" результата, то оцените сложность решения. Как получить ответ без append?

Упражнение 3. Напишите ещё одну версию функции many-replace и замените вызов в теле change-person. Сделайте так, чтобы тело новой версии состояло только из вызова map. Дополнительную функцию не определяйте отдельно, а заведите как анонимную — результат вычисления спецформыlambda. Переписанная функция должна быть столь же эффективной, как результат выполнения упражнения 2. В ходе дальнейшей работы над «Доктором» всякий раз, когда описываете обработку списка, сделайте это с помощью подходящей функции высшего порядка (map, foldl, foldr, filter, andmap, ormap ...). В модуле, подключаемом директивой (requireracket/list) есть дополнительные функции высшего порядка для работы со списками (filter-map, count, append-map, filter-not, argmin, argmax, remf, remf*, ...), имейте это в виду.

Упражнение 4. Двух способов генерации ответов мало. Добавим третий. «Доктор» может запоминать все реплики пациента и в ходе беседы возвращаться к сказанному пациентом ранее. В этом случае реплика «Доктора» будет начинаться со слов earlieryousaidthat, а затем будет следовать одна из предыдущих реплик пациента, в которой выполнена замена лица. Например, если из предыдущих реплик пациента выбрана реплика (youarenotbeingveryhelpfultome), то по ней будет построен ответ (earlieryousaidthat i amnotbeingveryhelpfultoyou).

Измените программу таким образом, чтобы doctor-driver-loop сохранял список всех реплик пользователя. Замечание: не используйте присваивания (мутаторы вроде set!). В этом нет необходимости. Новая версия функции reply будет выбирать одну стратегию из трёх. Новую стратегию реализуйте как отдельную функцию history-answer. Обратите внимание, что hedge и qualifier-answer можно применять всегда, а history-answer — только при наличии предыдущих реплик. Когда пациент вводит первую реплику, эта стратегия не применима. Учтите это в реализации. Для случайного выбора из трёх альтернатив следует взять случайное число от 0 до 2: (random 3), и использовать его как номер выбранной стратегии, считая, что они нумеруются с нуля. Для вызова стратегии по выбранному номеру подходит спецформаcase.

Упражнение 5. Измените программу таким образом, чтобы «Доктор» автоматически переходил к приёму следующего пациента после прощания с предыдущим. Предусмотрите способ завершения работы многопользовательского «Доктора» или после использования стоп-слова в качестве имени очередного пациента, или при исчерпании количества принимаемых пациентов. Стоп-слово и максимальное количество пациентов передаются в обновлённый visit-doctor как значения его параметров.

Упражнение 6. Реализуйте в программе стратегию построения ответов по ключевым словам. Пополните набор групп ключевых слов (не менее чем двумя новыми группами). Пополните варианты ответов для каждой группы (не менее чем двумя вариантами). Построение реплик по ключевым словам должно быть реализовано так, чтобы можно было вносить изменения только в структуру данных, не исправляя код функции, составляющей реплику. Реализуйте случайный выбор ключевого слова для построения реплики, если этих слов несколько во фразе пользователя. Этот выбор должен учитывать количество вхождений слова в реплику пациента. Когда выбор ключевого слова сделан, по нему следует выбрать один из подходящих шаблонов для составления ответной реплики. Предусмотрите учёт ситуации, когда одно и то же ключевое слово относится к разным группам. Для таких слов следует выбирать шаблон из объединённого перечня всех шаблонов, относящихся к каждой группе, куда входит ключевое слово. В выбранном шаблоне следует заменить * на ключевое слово. При этом следует ис-

пользовать ту реализацию замен, которая уже есть в коде «Доктора». Поскольку построение реплики является обработкой списка, напишите код с использованием уместных функций высшего порядка (map, foldl, foldr, filter, andmap, ormap ...).

Стратегия построения реплики по ключевым словам применима только в том случае, когда в реплике есть какое-то ключевое слово. Следует описать отдельную функцию-предикат, проверяющую это. Функция-предикат должна быть эффективной. Она не должна делать лишних действий, после того как становится ясно, что стратегия применима. Если для работы функции-предиката не нужна вся структура данных, хранящая группы ключевых слов и привязанных к ним шаблонов, то следует однократно предобработать её, получив необходимые данные, и затем использовать результат однократной предобработки в вызовах функции-предиката.

Упражнение 7. Каждый раз при добавлении новой стратегии приходилось переписывать reply. Можно переписать его, создав обобщённую версию, которая будет работать с любым подаваемым ему на вход перечнем стратегий. Удобство обобщённого reply состоит в том, что при изменении стратегий построения ответных реплик будет достаточно менять сами данные о стратегиях, но не управляющий механизм. Этот универсальный механизм состоит в следующем: во-первых, строится список стратегий, применимых в текущей ситуации; во-вторых, если в построенном списке больше одной стратегии, то выбирается одна из них; в-третьих, выбранная стратегия применяется и её результат будет ответной репликой. На вход механизма подаётся структура данных со сведениями обо всех стратегиях «Доктора». Про каждую стратегию в этой структуре хранится: а) функция-предикат от реплики пользователя, от истории реплик пользователя, от, возможно, других параметров, которая возвращает не #f, только если стратегия применима, и #f — иначе (Внимание! Должна храниться именно функция!); б) вес стратегии — натуральное число, которое будет использовано при выборе одной из применимых стратегий (чем больше вес, тем выше вероятность применения стратегии, например, если есть применимые стратегии с весами 2, 3, 1, то первая будет выбрана с вероятностью 1/3, вторая — с вероятностью 1/2, третья — с вероятностью 1/6); тело стратегии — функция, которая по реплике пользователя, по истории реплик пользователя, по, возможно, другим параметрам строит ответную реплику (Внимание! Должна храниться именно функция!). Например, для стратегии "hedge" функция-предикат — это функция всегда возвращающая #t; вес этой стратегии = 1 (самый малый); тело этой стратегии — функция hedge.

Выполняя упражнение 7, следует переписать reply, добавив ему параметр — структуру данных о стратегиях построения реплик. Все стратегии ответов, имеющиеся в программе должны быть представлены в этой структуре данных. Каких-либо способов построения ответов вне структуры данных о стратегиях быть не должно. Структура данных о стратегиях не зависит от реплик пользователей и её значение не должно вычисляться более чем один раз. Так как речь снова идёт об обработке списков, пишите код с использованием уместных функций высшего порядка (map, foldl, foldr, filter, andmap, ormap ...). Назначая веса стратегиям, используйте разные значения веса, чтобы взвешенный поиск не превращался в случайный — pick-random. Рекомендуется реализовать аналог pick-random — pick-random-with-weight, который получает список элементов, имеющих веса, и выбирает случайный элемент с учётом весов. Рекомендуется перед реализацией «геометрически» решить задачу, т. е. рассмотреть случайный выбор части из составного отрезка (длина части отрезка = весу соответствующего элемента списка).

Упражнение 8 (дополнительное). Упражнение посвящено дальнейшему совершенствованию «Доктора». Одним из путей совершенствования является добавление, так называемого, метауровня. Ранее мы реализовали один из способов выбора стратегии построения ответов. Этот способ заключался в том, что среди всех применимых стратегий с учётом их весов случайно применяется одна. Обозначим этот способ выбора как управляющую стратегию №1. Мы могли бы использовать другой способ выбора (другую управляющую стратегию). Например, можно упорядочить стратегии по весу, проверять сначала применимость более тяжёлых и, как только какая-то стратегия применима, останавливать проверку и использовать её (т. н. управляющая cond-стратегия с весами). Другие управляющие стратегии могли бы учитывать историю стратегий (предыдущие использования стратегий построения ответов), сложность выполняемых проверок и др.. На метауровне «Доктор» перед тем как выбирать стратегию построения ответа, анализирует, какие управляющие стратегии применимы в текущий момент. Из применимых управляющих стратегий «Доктор» выбирает одну (случайно, с учётом веса). Далее он запускает выбранную управляющую стратегию для выбора стратегии построения ответа. У всех управляющих стратегий должны быть нетривиальные условия их применения. Решения с тривиальными условиями, тождественными не #f или всегда возвращающими не #f, не принимаются.

Второй путь совершенствования «Доктора» — это учёт истории в стратегии ответов по ключевым словам. «Доктор» мог бы запоминать, какое ключевое слово было использовано и к какой группе ключевых слов относилась выбранная реплика (ведь одно слово может входить в несколько групп). В последующем «Доктор» мог бы, с одной стороны поддерживать тему, выбирая реплики-ответы из предыдущей группы, если есть такая возможность, с другой стороны, избегать повторов, заменяя * в заготовке реплики не на использованное ранее ключевое слово, а на родственные ключевые слова из одной группы с ним. Например, если предыдущий обмен репликами, касался матери пациента, «Доктор» мог бы поддержать тему семьи, но переключить разговор на других родственников пациента.

Можно предлагать и реализовывать собственные стратегии построения ответных реплик, использовать идеи исходной программы ELIZA и т. п.. Можно попрактиковаться в cps-написании кода, переписав все функции, порождающие рекурсивные процессы, в их cps-версии.

Указание по выполнению второго практического задания «Создание игровой программы на основе минимаксного алгоритма»

Создайте игровую программу, используя минимаксный алгоритм. При решении второго задания можно использовать обычные библиотеки, мутируемые структуры, средства ООП, присваивание. Использование этих средств должно быть обосновано. При оценке игровой программы учитывается «сила» её игры, сложность выбранной игры, сложность реализации. Код второго задания должен быть оформлен как следует. Он должен содержать содержательные комментарии на русском языке. В них следует указать, как представлена игровая ситуация, какой способ решения игры (поиска хода) реализован, каковы его параметры. Как отправную точку в написании кода решающей части игровой программы можно рассматривать реализацию крестиков-ноликов. Этапы сдачи:

1й — создание основы игры. Следует придумать внутреннее представление для игровой ситуации. Реализовать основные функции: перейти в новую ситуацию из текущей согласно некоторому ходу игрока; проверить, окончена ли игра (победой, поражением, ничьей); вывести игровую ситуацию в текстовом виде. Следует реализовать двух "игроков", делающих ходы по вводу пользователя.

2й — создание итоговой версии. Следует реализовать решающие функции: эвристически оценить позицию; построить дерево с оценками, выбрать оптимальный ход по минимаксу (с альфа-бета-отсечением). Следует реализовать двух "игроков", делающих оптимальные ходы. Не рассматриваются «реализации», в которых под видом поиска по минимаксу с альфа-бета-отсечением осуществляется ординарный поиск и/или генерация случайных ходов.
3й – подготовка и сдача отчёта в электронном виде.
Варианты игр для второго практического задания:

1) Калах
2) Четыре в ряд
3) Крестики-нолики 3D
4) Шашки
5) Миниуголки
6) Реверси
7) Шахматы 5x5
8) Му-торере
9) Рэндзю
10) Четыре в ряд 3D
11) Пятипольное коно
12) Семь цветов
13) Точки и квадраты
14) L-игра
15) 8 пешек
16) Qubic
17) Аналогичная игра не из списка, по выбору.

Указания по составлению отчёта по второму практическому заданию

Отчёт пишется на русском языке. Вёрстку можно осуществлять в любой подходящей для Вас системе. Текст отчёта должен быть разбит на следующие части:

• Титульный лист, с «шапкой» – «Московский государственный университет имени М. В. Ломоносова, факультет Вычислительной математики и кибернетики». Далее следует заголовок: «Отчёт по второму заданию», номер и тема варианта задания, сведения об исполнителе (фа-

милия, имя и отчество полностью, номер группы). Внизу титульного листа указывается город и год. Нелишне обратить внимание на то, что точки после заголовков не ставятся.

• Содержание, которое состоит из перечня названий глав и подглав, сопровождаемых указанием номеров страниц, с которых они начинаются. Нумеруются все страницы, за исключением титульного листа. Номер страницы с содержанием: 2.

• Первая глава, названная «Постановка задачи», содержит формулировку задания и описание правил выбранной игры. Каждую главу следует начинать с новой страницы.

• Вторая глава, названная «Создание основы игры», содержит описание структур данных для представления игровой ситуации и основных функций.

• Третья глава, названная «Создание итоговой версии», содержит описания: выбранной эвристической функции и её обоснование; параметров поискового алгоритма (глубины раскрытия дерева, усовершенствование выбора ходов, улучшение минимакса и т. п.); реализаций решающих функций. Эти дополнительно был реализован GUI, то в текст главы следует включить его описание. Должно быть описано, как протекает одна игровая партия, должны быть приложены скриншоты, поясняющие особенности дизайна GUI.

• Четвёртая глава, названная «Результаты», содержит анализ результатов работы программы. Следует охарактеризовать результаты, полученные на тестовых прогонах программы, оценить «силу» игры программы. Если доступны другие реализации выбранной игры, следует сравнить результаты их работы с результатами, демонстрируемыми Вашей программой.

• Заключение (которое не нумеруется, но номер на странице ставится), где подводится общий итог работы, завершает отчёт. В заключении можно указать характеристики написанного кода, привести соображения о том, насколько удачно удалось применить минимаксный алгоритм к решению доставшейся Вам задачи.

• Список использованной литературы приводится, если в ходе работы над заданием были использованы статьи и/или книги. Библиографические записи в списке следует оформлять по рекомендациям ГОСТ. Сделать это можно при помощи Google.Schollar, который умеет импортировать по ГОСТ. На каждую запись списка в тексте отчёта должна быть ссылка.

• Приложение, которое содержит Ваш код.

11.РЕСУРСНОЕ ОБЕСПЕЧЕНИЕ

Основная литература

1. Абельсон Х., Сассман Дж. и др. Структура и интерпретация компьютерных программ. М.: Добросвет. 2010. 608 с. [PDF] [http://newstar.rinet.ru/~goga/sicp/sicp.pdf]

2. Харрисон Дж. Введение в функциональное программирование. Перевод с английского. Новосибирск. 2009. [PDF]. [https://goo.gl/gDKPi5]
3. Малышко В. В. Слайды к лекциям по курсу "Введение в функциональное программирование". М.: ВМК. 2015. [PDF]. [http://vk.com/sp_scheme]

Дополнительная литература

1. Малышко В. В. Учебное пособие по выполнению задания «Доктор». 2015. [HTML]. [http://sp.cs.msu.ru/scheme/doctor.html]
2. Brown J. H. Advanced Scheme: Some Naughty Bits. Scheme Macros. MIT. 2003.[PDF].[http://web.mit.edu/sipb-iap/www/2003/scheme/macroslides.pdf]

Ресурсы информационно-телекоммуникационной сети «Интернет»

1. Сайт среды программирования DrRacket [http://drracket.org]
2. Веб-страница дисциплины «Введение в функциональное программирование» [http://sp.cs.msu.ru/scheme]
3. Веб-страница онлайновой учебной группы в социальной сети Вконтакте [http://vk.com/sp_scheme]
5. Лямбда-исчисление. СПбГУ ИТМО [neerc.ifmo.ru/wiki/index.php?title=Лямбда-исчисление]
6. Веб-страница курса SICPMIT 2005 [http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-001-structure-and-interpretation-of-computer-programs-spring-2005/]
7. Кирпичёв Е. Р. Веб-страница курса «Функциональное программирование». CS клуб. [http://logic.pdmi.ras.ru/csclub/courses/functional_programming]

Информационные технологии, используемые в процессе обучения

1. Программное обеспечение для подготовки слайдов лекцийOpenOfficeImpress.
2. Программное обеспечение для создания и просмотра pdf-документов AdobeReader.
3. Среда программирования DrRacket.
4. Веб-браузер для доступа к материалам, размещённым в WWW (MozillaFirefoxили аналогичный).
5. Программа-клиент электронной почты для онлайновых консультаций с лектором (MozillaThunderbirdили аналогичный).

Активные и интерактивные формы проведения занятия

№ п\п	Тип занятия или внеаудиторной работы	Вид и тематика (название) интерактивного занятия
1	Лекции по темам 1.1–1.5, 2.1–2.4, 3.1.	В рамках каждого лекционного занятия выделяется время (около 15 минут) для интерактивного обсуждения материала, а также проводится анкетирование студентов по ранее прочитанному материалу.

Материально-техническая база

Для преподавания дисциплины требуется класс, оборудованный маркерной или меловой доской и мультимедийным проектором. Для выполнения домашних заданий, а также для выполнения и сдачи практических заданий необходим персональный компьютер с доступом в Internet и установленной на нём средой программирования DrRacket.

12. ЯЗЫК ПРЕПОДАВАНИЯ

Русский

13. РАЗРАБОТЧИК ПРОГРАММЫ, ПРЕПОДАВАТЕЛИ

доцент, к.ф.-м.н.Малышко Виктор Васильевич

Приложение

ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО ДИСЦИПЛИНЕ

«Введение в функциональное программирование»

Средства для оценивания планируемых результатов обучения, критерии и показатели оценивания приведены ниже.

РЕЗУЛЬТАТ ОБУЧЕНИЯ по дисциплине (модулю)	**КРИТЕРИИ и ПОКАЗАТЕЛИ ОЦЕНИВАНИЯ РЕЗУЛЬТАТА ОБУЧЕНИЯ по дисциплине (модулю)** *(критерии и показатели берутся из соответствующих карт компетенций, при этом пользуются либо традиционной системой оценивания, либо БРС)*					**ОЦЕНОЧНЫЕ СРЕДСТВА**
	1	2	3	4	5	
	Неудовлетворительно	**Неудовлетворительно**	**Удовлетворительно**	**Хорошо**	**Отлично**	
ЗНАТЬ: современные методы построения и анализа математических моделей, возникающих при решении естественнонаучных задач, а также современные методы разработки и реализации алгоритмов их решения З1 (ПК-1)	Отсутствие знаний	Фрагментарные представления о современных методах построения и анализа математических моделей, возникающих при решении естественнонаучных задач, а также современных методах разработки и реализации алгоритмов их решения	В целом сформированные, но неполные знания о современных методах построения и анализа математических моделей, возникающих при решении естественнонаучных задач, а также современных методах разработки и реализации алгоритмов их решения	Сформированные, но содержащие отдельные пробелы знания о современных методах построения и анализа математических моделей, возникающих при решении естественнонаучных задач, а также современных методах разработки и реализации алгоритмов их решения	Сформированные систематические знания о современных методах построения и анализа математических моделей, возникающих при решении естественнонаучных задач, а также современных методах разработки и реализации алгоритмов их решения	Устный экзамен
УМЕТЬ: применять современные методы построения и анализа математических моделей, возникающих при ре-	Отсутствие умений	Фрагментарные умения применять современные методы построения и анализа математических моделей, возникающих при ре-	В целом успешное, но не систематическое умение применять современные методы построения и анализа матема-	Успешное, но содержащее отдельные пробелы умение применять современные методы построения и ана-	Сформированное умение применять современные методы построения и анализа математических моделей, возникающих при ре-	Устный экзамен

шении естественнонаучных задач, а также современные методы разработки и реализации алгоритмов их решения У1 (ПК-1)		шении естественнонаучных задач, а также современные методы разработки и реализации алгоритмов их решения	тических моделей, возникающих при решении естественнонаучных задач, а также современные методы разработки и реализации алгоритмов их решения	лиза математических моделей, возникающих при решении естественнонаучных задач, а также современные методы разработки и реализации алгоритмов их решения	шении естественнонаучных задач, а также современные методы разработки и реализации алгоритмов их решения	
ВЛАДЕТЬ: навыками оптимального выбора современных методов построения и анализа математических моделей, возникающих при решении естественнонаучных задач, а также современных методов разработки и реализации алгоритмов их решения В1 (ПК-1)	Отсутствие навыков	Фрагментарное владение навыками оптимального выбора современных методов построения и анализа математических моделей, возникающих при решении естественнонаучных задач, а также современных методов разработки и реализации алгоритмов их решения	В целом успешное, но не полное владение навыками оптимального выбора современных методов построения и анализа математических моделей, возникающих при решении естественнонаучных задач, а также современных методов разработки и реализации алгоритмов их решения	Успешное, но содержащее отдельные пробелы владение навыками оптимального выбора современных методов построения и анализа математических моделей, возникающих при решении естественнонаучных задач, а также современных методов разработки и реализации алгоритмов их решения	Сформированное владение навыками оптимального выбора современных методов построения и анализа математических моделей, возникающих при решении естественнонаучных задач, а также современных методов разработки и реализации алгоритмов их решения	Устный экзамен
УМЕТЬ: самостоятельно осуществлять научно-исследовательскую деятельность в соответствующей профессиональной области с использованием современных методов исследования и информационно-коммуникационных	Отсутствие умений	Частично освоенное умение самостоятельно осуществлять научно-исследовательскую деятельность в соответствующей профессиональной области с использованием современных методов исследования и информационно-коммуникационных	В целом успешное, но не систематическое умение самостоятельно осуществлять научно-исследовательскую деятельность в соответствующей профессиональной области с использованием современных методов иссле-	В целом успешное, но содержащее отдельные пробелы умение самостоятельно осуществлять научно-исследовательскую деятельность в соответствующей профессиональной области с использованием современ-	Успешное и систематическое умение самостоятельно осуществлять научно-исследовательскую деятельность в соответствующей профессиональной области с использованием современных методов исследования и информационно-	устный экзамен

технологий У1 (ОПК-1)		технологий	дования и информационно-коммуникационных технологий	ных методов исследования и информационно-коммуникационных технологий	коммуникационных технологий	
УМЕТЬ: самостоятельно осуществлять научно-исследовательскую деятельность в соответствующей профессиональной области с использованием современных методов исследования и информационно-коммуникационных технологий У1 (ОПК-1)	Отсутствие умений	Частично освоенное умение самостоятельно осуществлять научно-исследовательскую деятельность в соответствующей профессиональной области с использованием современных методов исследования и информационно-коммуникационных технологий	В целом успешное, но не систематическое умение самостоятельно осуществлять научно-исследовательскую деятельность в соответствующей профессиональной области с использованием современных методов исследования и информационно-коммуникационных технологий	В целом успешное, но содержащее отдельные пробелы умение самостоятельно осуществлять научно-исследовательскую деятельность в соответствующей профессиональной области с использованием современных методов исследования и информационно-коммуникационных технологий	Успешное и систематическое умение самостоятельно осуществлять научно-исследовательскую деятельность в соответствующей профессиональной области с использованием современных методов исследования и информационно-коммуникационных технологий	устный экзамен
Знать Современные мето разработки программ го обеспечения выч лительных комплекто З1(ПК-3)	Отсутствие знаний	Фрагментарные предст ления о современных тодах разработки п граммного обеспече вычислительных к плектов	В целом сформированные, но неполные знания о современных методах разработки программного обеспечения вычислительных комплектов	Сформированные, но содержащие отдельные пробелы знания осовременных методах разработки программного обеспечения вычислительных комплектов	Сформированные систематические знания осовременных методах разработки программного обеспечения вычислительных комплектов	Устный экзамен
Владеть Современными мето ми разработки п	Отсутствие навыков	Фрагментарное владе навыками оптимальн выбора современны	В целом успешное, не полное владение выками оптимальн выбора современн	Успешное, но со жащее отдельные п белы владение навы ми оптимального	Сформированное владе навыками оптимальн выбора современных	Устный экзамен

граммного обеспече вычислительных к плексов В1(ПК-3)		методов разработки п граммного обеспече вычислительных к плексов	методов разрабо программного об печения вычис тельных комплекс	бора современн методов разрабо программного об печения вычис тельных комплекс	тодов разработки п граммного обеспече вычислительных к плексов	
Уметь Разрабатывать программное обеспечение для вычислительных комплексов У1(ПК-3)	Отсутствие умений	Фрагментарные умения разрабатывать программное обеспечение для вычислительных комплексов	В целом успешное, но не систематическое умение разрабатывать программное обеспечение для вычислительных комплексов	Успешное, но содержащее отдельные пробелы умение разрабатывать программное обеспечение для вычислительных комплексов	Сформированное умение разрабатывать программное обеспечение для вычислительных комплексов	Устный экзамен

Фонды оценочных средств, необходимые для оценки результатов обучения

Типовые контрольные задания или иные материалы для проведения текущего контроля успеваемости.
Примеры анкет, при помощи которых проводится регулярное анкетирование на лекциях:
Анкета 1

I. Значением выражения ((lambda (a b) (- (/ a b) (+ a b))) 18 2) является:

a) □ функция;
b) □ комбинация;
c) □ ничего;
d) □ выражение ошибочно;
e) □ число / укажите, какое:_____ ;
f) □ другое / укажите, что: _____ .

II. Значением выражения ((lambda (x + y) (+ x y)) 5 * 6) является:

a) □ выражение ошибочно;
b) □ число / укажите, какое:_____ ;
c) □ другое / укажите, что: _____ ;

d) □ функция;
e) □ комбинация;
f) □ ничего.

Анкета 2

Функция (sqlist n) для натурального n возвращает список квадратов первых n натуральных чисел (1 4 9 … n2), а для остальных целых n – пустой список. Дайте два определения этой функции:

А) Определение, порождающее линейно итеративный процесс: _____

Б) Определение, порождающее линейно рекурсивный процесс: _____

Анкета 3

С помощью функций высшего порядка map, foldl, foldr, filter опишите функцию (task22 lst) принимающую непустой список списков чисел. Функция находит среднее арифметическое максимумов модулей элементов вложенных списков.

Анкета 4

Опишите стиль передачи остаточных вычислений функцию (maxlist-cpslstcc), возвращающую максимум непустого списка чисел.

Анкета 5

I. В глобальном окружении GE есть описание:

```
(define (sum-list lst)
  (define (helper lst res)
    (if (equal? lst '()) res (helper (cdrlst) (+ res (car lst)))))
(helperlst 0))
```

Нарисуйте диаграмму модели вычислений с окружениями для вычисления выражения

(sum-list '(1 2))|GE

II. Основываясь на решении I, ответьте на вопрос: адекватно ли описывает модель вычислений с окружениями хвостовую рекурсию?

Анкета 6

Опишите спец. форму or макросом.

Пример варианта письменной контрольной работы (в составе которой тесты, задачи на составление программ, текстовые вопросы)

1. Тесты

1.I. Какое подвыражение вычисляется первым при вычислении (f a b):

a) □ a;
b) □ b;

c) □ f;
d) □ неизвестно (определяется интерпретатором).
1.II. Укажите, какие из перечисленных выражений являются литералами:
a) □ #t;
b) □ "abc";
c) □ 2/3;
d) □ (/ 2 3);
e) □ (quote (/ 2 3));
f) □ '(/ 2 3);
g) □ abc;
h) □ 'abc
i) □ (a b c).
1.III. Укажитерезультатвычисления ((lambda (x * y) (* x y)) (3 min 4)):
a) □ 3
b) □ 4
c) □ошибка
d) □ 12
e) □ (min 3 4)
1.IV. Пустьестьопределения: (define a (list (list ‘q) ‘r ‘s))
(define b (cons (list ‘q) (list ‘r ‘s)))
Укажите выражения, являющиеся истинными:
a) □ (eq? a b) b) □ (equal? a b)
c) □ (eq? (car a) (car b)) d) □ (= a b)
1.V. Пусть есть определения: (define (f n) (f (f n)))
(define (g n m) (if (= n 0) 1 m))
Укажите только верные утверждения
a) □ вычисление (g 0 (f 1)) зациклится при нормальном порядке вычислений;
b) □ вычисление (g 0 (f 1)) зациклится при аппликативном порядке вычислений;
c) □ вычисление (g 0 (f 1)) не зациклится независимо от порядка вычислений.
1.VI. Данноопределение (define (f a t) (cond ((= a 0) t)

(else (f (- a 1) (+ t (f (/ a 2) 1))))))

Укажите только верные утверждения:

a) □ вычисление функции f при ненулевом a порождает итеративный процесс;

b) □ вычисление функции f при ненулевом a порождает рекурсивный процесс.

1.VII. Укажите, что отличает спецформы от функций:

a) □ спецформам не нужно давать определения перед их использованием;

b) □ правила вычисления функций отличаются от правил вычисления спецформ;

c) □ при вычислении спецформ всегда используется нормальный порядок вычислений;

d) □ при вычислении спецформ всегда используется аппликативный порядок вычислений.

1.VIII. Каков результат вычисления (applymin '(5 3 7 2)):

a) □ошибка b) □ 5 c) □ 3 d) □ 7 e) 2

1.IX. Что сделать, чтобы программным способом убедиться, что if в Scheme является специальной формой:

a) □ вычислить if, указав среди аргументов вызов некоторой функции;

b) □ попытаться переопределить if;

c) □ попытаться использовать if внутри if;

d) □ использовать if внутри тела некоторой функции;

e) □ никак, в этом нельзя убедиться программным способом.

1.X. Значением выражения ((lambda (a b) (- (/ a b) (+ a b))) 8 2) является:

a) □ функция;

b) □ комбинация;

c) □ ничего;

d) □ выражение ошибочно;

e) □ число / укажите, какое: _____ ;

f) □ другое / укажите, что: _____ .

2. Задачи на программирование

2.I. Функции (abclst) в качестве аргумента подают список из чисел. Результатом функции является список из тех же элементов, отсортированный по неубыванию. Реализуйте функцию методом сортировки выбором.

2.II. «Сделай поровну». Пусть даны две операции, преобразующие пару чисел. Первая операция удваивает первое число и увеличивает на 1 второе. Вторая операция увеличивает на 1 первое число и удваивает второе. Напишите функцию, которая для заданной пары чисел находит последовательность применения операций, приводящую к паре равных чисел. Пример

(f (list 3 7)) ==> '(2 1 1)

2.III. Квадродеревом называется следующий способ представления растровых черно-белых изображений:

1) Если изображение целиком белое, то оно представляется квадродеревом из одной «белой» вершины. Линейная запись такого квадродерева: 0. [ноль]

2) Если изображение целиком чёрное, то оно представляется квадродеревом из одной «чёрной» вершины. Линейная запись такого квадродерева: 1. [единица]

3) Если на изображении есть и чёрные и белые участки, то оно делится на 4 равные части (верхнюю левую, верхнюю правую, нижнюю левую, нижнюю правую) и представляется квадродеревом, состоящим из корневой вершины и четырёх поддеревьев, которые описывают части изображения. Пусть линейные записи поддеревьев таковы: <верхлевдерево>, <верхправдерево>, <нижнлевдерево>, <нижнправдерево>; тогда запись всего дерева будет списком из четырёх элементов: (<верхлевдерево><верхпправдерево><нижнлевдерево><нижнправдерево>)

Составьте функцию (simplify t), которая упрощает запись квадродерева t, заменяя поддеревья вида (1 1 1 1) к 1, (0 0 0 0) к 0 до тех пор, пока не будет получено дерево с минимальной высотой, достигаемой упрощением.

Пример: (simplify '(1 1 1 (1 1 1 1))) ==> 1

2.IV. Функция (fib n cc) должна быть написана в стиле передачи остаточных вычислений. В качестве первого аргумента функции подаётся неотрицательное n. Результатом является n ое число Фибоначчи.

Примеры: (fib 0 (lambda (x) x)) ==> 0
(fib 1 (lambda (x) x)) ==> 1
(fib 2 (lambda (x) x)) ==> 1
(fib 3 (lambda (x) x)) ==> 2 ...

2.V. С использованием функций метапрограммирования реализуйте функцию (delete-longestlst), которая возвращает список, получаемый из исходного списка списков lst удалением из него подсписков имеющих наибольшую длину. Функция delete-longest не должна быть рекурсивной. Не используйте встроенные функции, кроме функций метапрограммирования (filter, foldl, foldr, map), базовых функций работы со списками (car, cdr, cons), арифметических функций, функций сравнения, max.

Пример: (delete-longest '((1) (2 3) (4 5) (6))) ==> '((1) (6))

2.VI Опишите функцию (fact2 n), вычисляющую двойной факториал. Порождаемый процесс должен быть итеративным.

3. Текстовые вопросы:

3.I. Расскажите о подстановочной модели вычислений. В чём она заключается? Каковы её ограничения? Проиллюстрируйте свой ответ примерами.

3.II. Нарисуйте стрелочную диаграмму для выражения

(list 1 (cons 2 3) (list 4 (list 5) (cons 6 7)))

3.III. Как работает специальная форма cond? Приведите пример её осмысленного использования.

Типовые контрольные задания или иные материалы для проведения промежуточного контроля успеваемости.
Пример варианта письменной экзаменационной работы (в составе которой тесты, задачи на составление программ, текстовые вопросы)

1. Тесты

1.I. Даны два определения: (define (enigma x) (x 2))
(define (square x) (* x x))
Укажите результат вычисления выражения: (enigma ‘square)

a) □ошибка b) □ ‘(square 2) c) □ 4 d) □ (‘square 2 2)

1.II. Укажите, какие из перечисленных выражений не могут быть идентификаторами в Scheme:

a) □ #true
b) □ true#
c) □ 4name
d) □ name4
e) □ x->y
f) □ x(y)
g) □ x[y]
h) □ x/y
i) □ x*y

1.III. Укажите только верные утверждения относительно (set! a b):

a) □ set! – это функция;
b) □ set! – это спецформа;
c) □ set! добавляет в окружение новое связывание имени a;
d) □ set! добавляет в окружение новое связывание имени a, только если ранее в нём не было связываний этого имени.

1.IV. Пустьестьопределения: (define a (list (list ‘q) ‘r ‘s))
(define b (cons (list ‘q) (list ‘r ‘s)))
Укажите выражения, являющиеся истинными:

a) □ (eq? a b) b) □ (equal? a b)
c) □ (eq? (car a) (car b)) d) □ (= a b)

1.V. Укажите только верные утверждения

a) □ окружение является частью кадра;

b) □ связывание является частью кадра;

c) □ кадр является частью связывания;

d) □ кадр является частью окружения.

1.VI. Укажите только верные утверждения относительно (set! a b):

a) □ set! вычисляет все свои аргументы;

b) □ результат вычисления (set! a b) равен значению b;

c) □ результат вычисления не определён (зависит от реализации);

d) □ set! вычисляет лишь один из своих аргументов;

e) □ до вычисления (set! a b) в одном из кадров окружения обязательно должно быть связывание a.

1.VII. Диаграммы классов и диаграммы объектов описывают объектно-ориентированную систему с точки зрения:

a) □ её реализации на каком-либо языке программирования;

b) □ реализации базовых механизмов, обеспечивающих возможность её реализации на каком-либо языке программирования;

c) □ её модели;

d) □ другой точки зрения/укажите какой:

1.VIII. Теорема Черча-Россера утверждает, что:

a) □ если нормальная форма существует, то она единственна

b) □ если нормальная форма существует, то она единственна с точностью до α-редукций

c) □ что-то другое, отличное от a) и b)

1.IX. Набор обобщённых операций предпочтительнее набора объектов данных:

a) □ всегда;

b) □ никогда;

c) □ в случае, когда программа изменяется без добавления новых типов данных;

d) □ в случае, когда программа изменяется без добавления новых обобщённых функций.

1.X. Лексическая безопасность системы макросов syntax-rules обеспечивается за счёт:

a) □ сочетания гигиеничности с выразительностью языка образцов;

b) □ сочетания прозрачности ссылок и гигиеничности;

c) □ сочетания гигиеничности и закрытости;

d) □ сочетания прозрачности ссылок и закрытости;

e) сочетания выразительности языка образцов и закрытости;
f) только лишь гигиеничности.
2. Задачи на программирование
2.I. Функции (abc-vectvect) в качестве аргумента подают вектор из чисел. Результатом функции является вектор из тех же элементов, отсортированный по неубыванию. Реализуйте функцию методом сортировки выбором. Решение при помощи преобразования библиотечных функций сортировки не засчитывается.
2.II. «Сделай поровну». Пусть даны две операции, преобразующие пару чисел. Первая операция удваивает второе число и увеличивает на 1 первое. Вторая операция увеличивает на 1 второе число и удваивает первое. Напишите функцию, которая для заданной пары чисел находит последовательность применения операций, приводящую к паре равных чисел. Пример
(f (cond 3 7)) ==> '(1 2 2)
2.III. Реализуйте структуру данных очередь. Перечень операций: конструктор (make-queue); селектор (front-queue q); вставка (insert-queue! q e); удаление (delete-queue! q); проверка типа (queue? q); проверка на пустоту (empty-queue? q). Сложность вставки должна быть константой!
2.IV. Опишите простую функцию f, такую результат вычисления выражения (+ (f 1) (f 2)) зависел от порядка вычисления слагаемых и был равен номеру слагаемого, вычисленного первым.
2.V. Найдите нормальную форму (λx. (λy. (+ x ((λx. (- x 3)) y)))) 5 6
Обоснуйте ответ последовательным применением редукций. Примечание: +, - и числа имеют обычный смысл.
2.VI. Пустьвычислено:

```
(define a (list 1 2 3))
(define b (list 4 a a))
(set-cdr! a b)
(set-car! b a)
```

Нарисуйте стрелочную диаграмму, изображающую a и b после выполненных вычислений.
2.VII. Последовательность 0 1 2 3 4 5 6 7 8 9 1 0 1 1 1 2 1 3 … образована выписыванием подряд цифр из записи ряда неотрицательных целых чисел, упорядоченных по возрастанию (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, …). Определите функцию (make-digits n) возвращающую вектор длины n, элементы которого являются n первыми элементами данной последовательности.
3. Текстовые вопросы:
3.I. Расскажите о различиях между подстановочной моделью вычислений и моделью вычислений с окружениями. Перечислите элементы модели вычислений с окружениями и расскажите о них. В чём преимущество модели вычислений с окружениями над подстановочной моделью. Есть ли недостатки у модели вычислений с окружениями? Проиллюстрируйте свой ответ примерами.

3.II. Что такое мемоизация? Когда и в чём проявляется положительный эффект мемоизации? Приведите осмысленный код функции, использующий мемоизацию (функции для работы с таблицами считайте определенными).
3.III. Расскажите о макросах в Scheme (syntax-rules). Когда уместно использовать макрос, когда лучше использовать функцию? Перечислите характеристики системы макросов syntax-rules. Приведите осмысленный и уместный пример макроса.

Методические материалы для проведения процедур оценивания результатов обучения

Система контроля и оценивания

Оценка по курсу устанавливается в зависимости от суммы технических баллов, набранных студентом в течение семестра. Каждая анкета, заполненная на лекции, оценивается от 0 до 2 технических баллов. Общая сумма баллов по анкетам может составить от 0 до 16 технических баллов. За выполнение обязательной части первого задания по программированию («Доктор») студенту могут быть начислены от 0 до 20 технических баллов. За выполнение обязательной части второго задания («Создание игровой программы на основе минимаксного алгоритма») по индивидуальному варианту студент может получить от 0 до 30 баллов. За составление отчёта по выполненному второму заданию студент может получить от 0 до 5 баллов. За выполнение контрольной письменной работы студент может получить от 0 до 30 технических баллов. За выполнение итоговой экзаменационной письменной работы студент может получить от 0 до 45 баллов. За выполнение необязательных «творческих» упражнений первого и второго задания по программированию студент может получить от 0 до 15 баллов совокупно. Таким образом, максимально возможная сумма набранных технических баллов без учёта баллов за необязательные упражнения составляет 150. Оценка «отлично» ставится студентам, набравшим от 110 баллов и выше при условии сдачи всех обязательных упражнений по программированию. Оценка «хорошо» ставится студентам, набравшим от 80 до 109 технических баллов при условии сдачи всех обязательных упражнений по программированию. Оценка «удовлетворительно» ставится студентам, набравшим от 60 до 79 технических баллов при условии сдачи всех обязательных упражнений по программированию. Оценка «неудовлетворительно» ставится студентам, набравшим менее 60 технических баллов, а также студентам, не сдавшим хотя бы одно из обязательных упражнений по программированию.

Структура и график контрольных мероприятий

Письменная контрольная работа по завершении первого раздела, отчёт по второму практическому заданию в конце семестра, псьменный экзамен в конце семестра.