МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ

«БРАТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра математики и физики

УТВЕРЖДАЮ:
Проректор по учебной работе
_____Е.И.Луковникова
«_____» декабря 2018 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

Б1.В.11

НАПРАВЛЕНИЕ ПОДГОТОВКИ

01.03.02 Прикладная математика и информатика

ПРОФИЛЬ ПОДГОТОВКИ
Инженерия программного обеспечения

Программа академического бакалавриата

Квалификация (степень) выпускника: бакалавр

СОДЕРЖАНИЕ ПРОГРАММЫ **Стр.**

1. ПЕРЕЧЕНЬ ПЛАНИРУЕМЫХ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ, СООТНЕСЕННЫХ С ПЛАНИРУЕМЫМИ РЕЗУЛЬТАТАМИ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Вид деятельности выпускника

Дисциплина охватывает круг вопросов, относящихся к научно-исследовательскому виду профессиональной деятельности выпускника в соответствии с компетенциями и видами деятельности, указанными в учебном плане.

Цель дисциплины

Целью изучения дисциплины является: ознакомление обучающихся с различными методами, приемами функционального программирования, приемами интеграции одних программных пакетов в другие и использованию результатов интеграции при создании собственных сложных универсальных программных комплексов.

Задачи дисциплины

- обучение методам компьютерного формализованного представления знаний и реализации выводов для последующей выработки и принятия человеком вариантов принимаемого решения;
- формирование умения и навыков самостоятельного исследования и решения различного рода задач путем применения средств функционального программирования совместно с другими видами программного обеспечения;
- формирование и развитие умений и навыков, позволяющих применять современные математические методы и программное обеспечение для решения задач науки и техники.

Код компетенции	Содержание компетенций	Перечень планируемых результатов обучения по дисциплине
1	2	3
ОПК-1	Способность использовать базовые знания естественных наук, математики и информатики, основные факты, концепции, принципы теорий, связанных с прикладной математикой и информатикой	**знать:** - математические основы функционального программирования **уметь:** – использовать технологии решения интеллектуальных задач в функциональном стиле **владеть:** - методами решения задач профессиональной деятельности
ОПК-3	Способность к разработке алгоритмических и программных решений в области системного и прикладного программирования, математических, информационных и имитационных моделей, созданию информационных ресурсов глобальных сетей, образовательного контента, прикладных баз данных, тестов и средств тестирования систем и средств на соответствие стандартам и исходным требованиям	**знать:** - основные алгоритмы решения задач **уметь:** – разрабатывать алгоритмы решения задач из области системного и прикладного программирования, математических, информационных и имитационных моделей ; **владеть:** - приемами построения алгоритмических и программных решений.

ПК-7	Способность к разработке и применению алгоритмических и программных решений в области системного и прикладного программного обеспечения	**знать:** - принципы построения программных решений на функциональных языках; **уметь:** - разрабатывать программы на функциональных языках **владеть:** – навыками разработки программных решений в функциональном стиле

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Дисциплина Б1.В.11 Функциональное программирование относится к дисциплинам базовой части.

Дисциплина Функциональное программирование базируется на знаниях, полученных при изучении таких учебных дисциплин, как: Математическая логика, Теория алгоритмов.

Основываясь на изучении перечисленных дисциплин, Функциональное программирование представляет основу для преддипломной практики и подготовки к государственной итоговой аттестации.

Такое системное междисциплинарное изучение направлено на достижение требуемого ФГОС уровня подготовки по квалификации бакалавр.

3. РАСПРЕДЕЛЕНИЕ ОБЪЕМА ДИСЦИПЛИНЫ

3.1. Распределение объема дисциплины по формам обучения

Форма обучения	*Курс*	*Семестр*	*Трудоемкость дисциплины в часах*						*Контрольная работа*	*Вид промежуточной аттестации*
			Всего часов (с экз.)	*Аудиторных часов*	*Лекции*	*Лабораторные работы*	*Семинары Практические занятия*	*Самостоятельная работа*		
1	2	3	4	5	6	7	8	9	10	11
Очная	4	7	144	68	17	51	-	40	кр	Экзамен
Заочная	-	-	-	-	-	-	-	-	-	-
Заочная (ускоренное обучение)	-	-	-	-	-	-	-	-	-	-
Очно-заочная	-	-	-	-	-	-	-	-	-	-

3.2. Распределение объема дисциплины по видам учебных занятий и трудоемкости

Вид учебных занятий	*Трудоемкость (час.)*	*в т.ч. в интерактивной, активной, иннова-циионной формах, (час.)*	*Распреде-ление по семестрам, час*
			7
1	2	3	4
I. Контактная работа обучающихся с преподавателем (всего)	68	26	68
Лекции (Лк)	17	6	17
Лабораторные работы (ЛР)	51	20	51
Контрольная работа*	+	-	+
Групповые (индивидуальные) консультации*	+	-	+
II .Самостоятельная работа обучающихся (СР)	40	-	40
Подготовка к лабораторным работам	10	-	10
Подготовка к экзамену в течение семестра	20	-	20
Выполнение контрольной работы	10	-	10
III. Промежуточная аттестация экзамен	36	-	36
Общая трудоемкость дисциплины час.	144	-	144
зач. ед.	4	-	4,0

4. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

4.1. Распределение разделов дисциплины по видам учебных занятий

- для очной формы обучения:

№ раздела и темы	*Наименование раздела и тема дисциплины*	*Трудоемкость, (час.)*	*Виды учебных занятий, включая самостоятельную работу обучающихся и трудоемкость; (час.)*		
			учебные занятия		*самостоятельная работа обучающихся*
			лекции	*лабораторные работы*	
1	2	3	4	5	6
1.	**Основные элементы функционального языка**	**64**	**10**	**30**	**24**
1.1.	Синтаксис и семантика языка Lisp	12	2	6	4
1.2.	Списочная организация данных	12	2	6	4
1.3.	Конструкции ветвления	12	2	6	4
1.4.	Организация циклов	12	2	6	4
1.5.	Свойства символа	11	1	6	4
1.6.	Ввод/вывод	5	1	-	4
2.	**Функциональный подход к решению задач**	**44**	**7**	**21**	**16**
2.1.	Функции пользователя	9	1	4	4
2.2.	Рекурсии	11	2	5	4
2.3.	Функционалы	12	2	6	4
2.4.	Макросы	12	2	6	4
	ИТОГО	**108**	**17**	**51**	**40**

4.2. Содержание дисциплины, структурированное по разделам и темам

№ раздела и темы	*Наименование раздела и темы дисциплины*	*Содержание лекционных занятий*	*Вид занятия в интерактивной, активной, инновационной формах, (час.)*
1	**2**	**3**	**4**
1.	**Основные элементы функционального языка**		
1.1.	Синтаксис и семантика языка Lisp	Особенности языка Lisp. Префиксная нотация записи функций. Функции с побочным эффектом. Математические функции.	-

1.2.	Списочная организация данных	Выражения, атомы и списки. Внутреннее представление списков. Базовые функции обработки списков.	Проблемная лекция (1 час)
1.3.	Конструкции ветвления	Конструкции if, when, until.Сложное ветвление cond. Форма выбора case.	-
1.4.	Организация циклов	Цикл loop. Форма do.Формы динамического прекращения вычислений.	Проблемная лекция (1 час)
1.5.	Свойства символа	Последовательные вычисления. Символ и его свойства. Функции для обращения к свойствам. Ассоциативные списки.	-
1.6.	Ввод/вывод	Форматный вывод .Файловые потоки. Работа с файлами.Макрос with-open-file.	-
2.	**Функциональный подход к решению задач**		
2.1.	Функции пользователя	Определение функции пользователя. Фиктивные и фактические параметры. Параметры функций.	-
2.2.	Рекурсии	Простая рекурсия. Рекурсии высоких порядков. Применение рекурсии для обработки списков.	Проблемная лекция (1 час)
2.3.	Функционалы	Функционалы: основные определения. Отображающие функционалы. Применяющие функционалы. Конструирование функционалов.	Проблемная лекция (1 час)
2.4.	Макросы	Понятие макроса. Контекст вычисления макроса. Лямбда-список и ключевые слова макросов. Обратная блокировка.	Проблемная лекция (1 час)

4.3. Лабораторные работы

№ п/п	*Номер раздела дисциплины*	*Наименование лабораторной работы*	*Объем (час.)*	*Вид занятия в интерактивной, активной, инновационной формах, (час.)*
1	**1.**	Синтаксические особенности языка Lisp. Математические функции .	14	Работа в малых группах (4 час)
2		Основные синтаксические конструкции	13	Работа в малых группах (4 час)

3	**2.**	Функции пользователя и рекурсии	12	Работа в малых группах (6 час)
4		Макросы и функционалы	12	Работа в малых группах (6 час)
		ИТОГО	**51**	**20**

4.4. Семинары/ практические занятия

учебным планом не предусмотрено.

4.5. Контрольные мероприятия: контрольная работа

Все контрольные работы выполняются как индивидуальные домашние задания. Зачтенные работы оформляются и включаются в портфолио обучающегося.

7 семестр. Контрольная работа «Элементы функционального программирования»

Цель: Проверить знания и умения программирования на языке Lisp

Структура:

Задача 1. Внутреннее представление списков;

Задача 2. Применение функционала

Задача 3. Разработка функционала;

Задача 4. Разработка макроса.

Основная тематика: математические функции, списки, организация вычислений.

Рекомендуемый объем: 4 задания.

Выдача задания , прием кр проводится в соответствии с календарным учебным графиком.

Оценка	Критерии оценки контрольной работы
отлично	Верное выполнение всех заданий с подробным пояснением их решения.
хорошо	Выполнение 3 заданий верно или всех заданий с незначительными ошибками или с неточностями в пояснениях.
удовлетворительно	Выполнение 2 заданий верно или3 заданий с неточностями и ошибками.
неудовлетворительно	Выполнение менее 2 заданий.

5. МАТРИЦА СООТНЕСЕНИЯ РАЗДЕЛОВ УЧЕБНОЙ ДИСЦИПЛИНЫ К ФОРМИРУЕМЫМ В НИХ КОМПЕТЕНЦИЯМ И ОЦЕНКЕ РЕЗУЛЬТАТОВ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Компетенции / *№, наименование разделов дисциплины*	*Кол-во часов*	*Компетенции*			*Σ комп.*	$t_{ср}$, *час*	*Вид учебных занятий*	*Оценка результатов*
		ОПК		*ПК*				
		1	*3*	*7*				
1	2	3	4	5	6	7	8	9
1. Основные элементы функционального языка	64	+	+	-	2	32	Лк, ЛР	кр, экзамен
2. Функциональный подход к решению задач	44	-	-	+	1	44	Лк, ЛР	кр, экзамен
всего часов	**108**	**32**	**32**	**44**	**3**	**36**		

6. ПЕРЕЧЕНЬ УЧЕБНО-МЕТОДИЧЕСКОГО ОБЕСПЕЧЕНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

1. Шичкина Ю.А. Создание приложений на языке Visual C# в среде программирования Visual Studio : учебное пособие / Ю. А. Шичкина. - Братск : БрГУ, 2011. - 210 с.

7. ПЕРЕЧЕНЬ ОСНОВНОЙ И ДОПОЛНИТЕЛЬНОЙ ЛИТЕРАТУРЫ, НЕОБХОДИМОЙ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

№	*Наименование издания*	*Вид заня-тия*	*Количество экземпляров в библиотеке*, **шт.**	*Обеспеченность*, **(*экз./чел.*)**
1	**2**	**3**	**4**	**5**
Основная литература				
1.	Информатика. Базовый курс : учебник для бакалавров и специалистов / Под ред. С. В. Симоновича. - 3-е изд. - Санкт-Петербург : Питер, 2015. - 640 с. - (Учебник для вузов. Стандарт третьего поколения)	Лк, ЛР, СР	123 включая аналоги	1,0
2.	Салмина, Н.Ю. Функциональное программирование и интеллектуальные системы : учебное пособие / Н.Ю. Салмина ; Министерство образования и науки Российской Федерации, Томский Государственный Университет Систем Управления и Радиоэлектроники (ТУСУР), Факультет дистанционного обучения. - Томск : ТУСУР, 2016. - 100 с. : ил. - Библиогр.: с.97 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=480936	Лк, СР	ЭР	1,0
Дополнительная литература				
3.	Сергиевский, Г. М. Функциональное и логическое программирование : учебное пособие / Г. М. Сергиевский, Н. Г. Волченков. - Москва : Академия, 2010. - 320 с.	Лк, СР	5	0,25
4.	Рогозин, О.В. Функциональное и рекурсивно-логическое программирование : учебно-методический комплекс / О.В. Рогозин. - Москва : Евразийский открытый институт, 2009. - 139 с. - ISBN 978-5-374-00182-2 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=90927	Лк, СР	ЭР	1,0

8. ПЕРЕЧЕНЬ РЕСУРСОВ ИНФОРМАЦИОННО-ТЕЛЕКОММУНИКАЦИОННОЙ СЕТИ «ИНТЕРНЕТ» НЕОБХОДИМЫХ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

1.Электронный каталог библиотеки БрГУ
http://irbis.brstu.ru/CGI/irbis64r_15/cgiirbis_64.exe?LNG=&C21COM=F&I21DBN=BOOK&P21DBN=BOOK&S21CNR=&Z21ID=.
2. Электронная библиотека БрГУ
http://ecat.brstu.ru/catalog .
3. Электронно-библиотечная система «Университетская библиотека online»
http://biblioclub.ru .
4. Электронно-библиотечная система «Издательство «Лань»
http://e.lanbook.com .
5. Информационная система "Единое окно доступа к образовательным ресурсам"
http://window.edu.ru .
6. Научная электронная библиотека eLIBRARY.RU http://elibrary.ru .

7. Университетская информационная система РОССИЯ (УИС РОССИЯ) https://uisrussia.msu.ru/ .
8. Национальная электронная библиотека НЭБ http://xn--90ax2c.xn--p1ai/how-to-search /.
Специальные тематические сайты
1. Сайт по программированию http://life-prog.ru ;
2. Электронный журнал «Типичный программист» https://tproger.ru .
3. Сайт по программированию http://www.w3ii.com/ru/lisp/lisp_quick_guide.html .

9. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Обучающийся должен разработать собственный режим равномерного освоения дисциплины. Подготовка студента к предстоящей лекции включает в себя ряд важных познавательно-практических этапов:

– чтение записей, сделанных в процессе слушания и конспектирования предыдущей лекции, вынесение на поля всего, что требуется при дальнейшей работе с конспектом и учебником;

– техническое оформление записей (подчеркивание, выделение главного, выводов, доказательств);

– выполнение практических заданий преподавателя;

– знакомство с материалом предстоящей лекции по учебнику и дополнительной литературе.

Успешность выполнения лабораторных работ определяется подготовкой к ним. Подготовка к лабораторным работам содержит:

- изучение теоретического материала, содержащегося в учебной литературе, изучение лекционного материала,

- знакомство с заданиями на лабораторную работу;

- составление плана выполнения лабораторной работы.

Наиболее продуктивной является самостоятельная работа в библиотеке, где доступны основные и дополнительные печатные и электронные источники.

При выполнении приведенных выше рекомендаций подготовка к зачету сведется к повторению изученного и совершенствованию навыков применения теоретических положений и различных методов решения к стандартным и нестандартным заданиям.

9.1. Методические указания для обучающихся по выполнению лабораторных работ

Лабораторная работа №1

Синтаксические особенности языка Lisp. Математические функции

Цель работы:
Изучить особенности программирования на языке Lisp. Освоить префиксную нотацию записи функций. Приобрести навыки написания простейших программ на языке Lisp.
Теоретические сведения:

1. **Написание и запуск LisP-программ.**

LisP-программы пишутся в обычном текстовом редакторе, таком, как Kate и сохраняются обычно с расширением .lsp или .lisp.
Для запуска программы в терминале набирают команду
clisp имя_программы
Можно также вычислять выражения в стиле LisP прямо в терминале, набрав команду clisp.
Пример программы «Hello, word!»:
(print ″Hello, word!″)

2. Функции и псевдофукнкции

ЛИСП-функциональный язык. Поэтому любая конструкция в нем по умолчанию интерпретируется как функция. Допустимые конструкции – это выражения в скобках.

В ЛИСПе используется префиксная нотация функций.

a+b sin(x) x^2

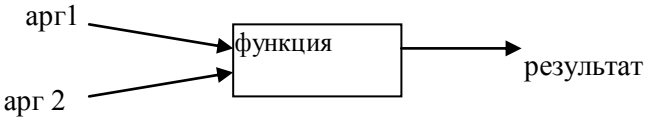
В ЛИСПе все функции записываются в префиксной нотации (f x1 x2…)

Таблица 1. Примеры записи функций в префиксной нотации

Обычная запись	Запись в стиле LisP
a+b	(+ a b)
1+3+5+7	(+ 1 3 5 7)
100-23-12	(- 100 23 12)
sin x	(sin x)
x<y	(< x y)
max(x,y,z)	(max x y z)
ln(1+2x)	(log (+ 1 (* 2 x)))

Задание. Записать в стиле ЛИСП и вычислить в терминале следующие выражения
а) 3.234*(45.6+2.43) б) 55+21.3+1.54*2-32 в) (34-21.5676-43)/(342+32*4.1)

Функция – это такой объект, который принимает входящие значения (аргументы) и вычисляет результат. Делать что-то ещё, например сообщать нам полученный результат, функция не обязана.



Если функция выполняет еще какие-то действия, то она называется **псевдофункцией** или функцией с побочным эффектом.

Таблица 2. Основные математические функции

Запись	Интерпретация	Запись	Интерпретация
(+ x1 x2 ..xn)	x1+x1+…+xn	(max x1 x2 ..xn)	max(x1, x1,…,xn)
(- x1 x2 ..xn)	x1-x1-…-xn	(min x1 x2 ..xn)	min(x1, x1,…,xn)
(* x1 x2 ..xn)	x1+*x1*…*xn	(mod m n)	остаток от деления m на n
(/ x1 x2 ..xn)	$\frac{x1}{x2 \cdot \ldots xn}$	(gsd x1 x2 ..xn)	НОД (x1, x1,…,xn)
(sin x)	sin x	(lsm x1 x2 ..xn)	НОК (x1, x1,…,xn)
(cos x)	cos x	(numerator n/m)	сокращает дробь n/m и возвращает её числитель
(tan x)	tg x	(abs x)	модуль x
(asin x)	arcsin x	(signum x)	знак числа x
(acos x)	arcos x	(round x)	округление
(atan x)	arctg x	(floor x)	целая часть числа x

(sqrt x)	$\sqrt{x}$	(celing x)	наименьшее целое число, не меньшее x
(exp x)	e^x	(random n)	случайное число из диапазона от 0 до n
(expt x y)	x^y	(- x)	-x
(log x)	ln x	(1+ x)	x+1
(log x y)	$\log_y x$	(1- x)	x-1
Сравнение чисел и предикаты			
(= x y)	равенство x= y?	(zerop x)	x=0?
(= x1 x2 …xn)	x1=x2=…=xn?	(numberp x)	x- число?
(/= x y)	$x \neq y$?	(integerp x)	x- целое
(>= x1 x2 …xn)	$x1 \geq x2 \geq ... \geq xn$?	(oddp x)	x-нечетное?
(<= x1 x2 …xn)	$x1 \leq x2 \leq ... \leq xn$?	(evenp x)	x-четное?

Базовые псевдофункции:

1) Вывод на экран: **print, princ, prin1**

Например
(print ″Привет!″) выводит строку;
(print x) выводит значение переменной x;
(print 'x) выводит «X»;
Сравните (print (+ 12 6)) и (print ‘(+ 12 6))

2) Присвоение переменной значения

Обычно языки программирования для присвоения переменной значения используют знак «=» (равно) или обозначение «:=». Но в Лиспе знак «=» применяется для сравнения, а для присвоения используют псевдофункции set, setq и setf. Их назначение несколько различается, но мы в начале будем пользоваться только **setq**.

Пример
(setq x 9) присваивает переменной x значение 9

3) Ввод данных: **read**

Функция не показывает, что она ждет ввода выражения. Она лишь читает выражение и возвращает в качестве значения само это выражение, после чего вычисления продолжаются.

Если прочитанное выражение необходимо сохранить для дальнейшего использования, то вызов READ должен быть аргументом какой-нибудь формы, например присваивания (SETQ), которая свяжет полученное выражение:

(setq x (read))
или так
(setq x (read) y (read) z (read))

Запрет на вычисления

Как уже говорилось, любое предложение, заключенное в скобки, считается функцией и выполняется сразу.

Чтобы предотвратить вычисление значения выражения, нужно перед этим выражением поставить апостроф «’». Апостроф перед выражением - это на самом деле сокращение лисповской функции **quote.**

Следующие две записи эквивалентны
(setq x (quote (* 3 4)) и (setq x '(* 3 4)).
В обоих случаях значением переменной x будет **выражение (* 3 4).**

Системный оператор **quote** подавляет вычисление своего аргумента. Если он не используется, то интерпретатор Лиспа, получив на входе список или символ, пытается его вычислить. Для символа

возвращается его значение, для списка — результат вызова функции, имя которой находится в голове списка, с параметрами — хвостом списка. Если же нужно, чтобы интерпретатор не вычислял значения, а взял символ или список «как есть», к нему применяют quote.

eval

Эта функция, по сути, и есть интерпретатор Лиспа. Она является противоположностью quote — она вычисляет значение своего аргумента.

(eval '(+ 12 5))) ==> 17

Возможность прямого и непосредственного вызова интерпретатора вкупе с идентичностью структуры программы и данных позволяет без каких-либо ограничений порождать и непосредственно исполнять в системе любые Лисп-программы.

Фрагмент кода

```
(setq x '(* 3 4)).
;;Сравните
(print x))   ,будет напечатано (* 3 4)
(print (eval x)) ,будет напечатано 12
```

Комментарии

В Лиспе однострочные и отбиваются точкой с запятой (;)

Задание:

1. Проделайте следующие вычисления с помощью интерпретатора Лиспа:

а) 3.234*(45.6+2.43)

б) 55+21.3+1.54*2.5432-32

в) (34-21.5676-43)/(342+32*4.1)

2. Написать программу, решающую квадратные уравнения без учета знака дискриминанта.
3. Написать программу, вычисляющую значение функции в точке, заданной пользователем

Вариант	Функция	Вариант	Функция
1	$f(x,y,z)=\frac{x^z-2e^y}{\ln(x^2+y^2+z^2+2)}$	6	$f(x,y,z)=\frac{x^{z-y}-\sqrt{5z+2}}{x^2+y^2}$
2	$f(x,y,z)=\frac{\sin 3x-y/29}{1+\sqrt{x^2+z^2}}$	7	$f(x,y,z)=\frac{\ln z+y^{\sin x}}{\sqrt{3x+1}}$
3	$f(x,y,z)=\frac{\sqrt{6x-z-y-8}}{\sin(3xy+z)}$	8	$f(x,y,z)=\frac{x^y}{x^2+2y^2+4z^2}$
4	$f(x,y,z)=\frac{x^{2+z}+\sqrt{y}}{y^{2\ln x}}$	9	$f(x,y,z)=\frac{3x}{y^z}+\frac{z}{\cos x}$
5	$f(x,y,z)=\frac{xe^y}{x^4+3y^2+1}$	10	$f(x,y,z)=\frac{3x}{y^2}+\frac{\ln(z+3)}{\sqrt{x}}$

4. Что больше e^{π} или π^e?
5. Напишите программу, которая запрашивает 3 числа a, b, c и находит наибольший остаток от деления этих чисел на 11.
6. Напишите строковый калькулятор на Lisp
7. Напишите программу, которая генерирует случайное число x и подставляет его в функцию, введенную пользователем с терминала.

Порядок выполнения:

1. Изучить теоретический материал;
2. Выполнить задания 1-7;
3. Запустить программы и протестировать их;
4. Составить отчет по лабораторной работе.

Форма отчетности:

Отчет по работе содержит:

1. Наименование лабораторной работы;
2. Разработанную программу;

3. Результаты её тестирования;
4. Выводы по работе.

Основная литература

1. Информатика. Базовый курс : учебник для бакалавров и специалистов / Под ред. С. В. Симоновича. - 3-е изд. - Санкт-Петербург : Питер, 2014. - 640 с.

Контрольные вопросы для самопроверки

1. Что такое префиксная нотация?
2. Как синтаксически оформляются комментарии?
3. Каковы особенности использования функций?
4. Перечислите базовые псевдо функции и кратко охарактеризуйте каждую из них.
5. Для чего используется функция eval?

Лабораторная работа №2

Основные синтаксические конструкции

Цель работы:

Изучить особенности программирования на языке Lisp. Изучить конструкции ветвления, циклы. Освоить приемы реализации алгоритмов на языке Lisp.

Теоретические сведения:

Разветвление вычислений: условное предложение COND

Предложение cond является основным средством разветвления вычислений. Это синтаксическая форма, позволяющая управлять вычислениями на основе определяемых предикатами условий. Структура условного предложения такова:

(nil cond
(p_1 a_1)
(p_2 a_2)
…
(p_n a_n))

Предикатами p_i и результирующими выражениями a_i могут быть произвольные формы. Значение предложения cond определяется следующим образом:

1. Выражения p_i, выполняющие роль предикатов, вычисляются пошагово слева направо (сверху вниз) до тех пор, пока не встретится выражение, значением которого не является ни nil ни false, т.е. логическим значением которого является истина.
2. Вычисляется выражение, соответствующее этому предикату, и полученное значение возвращается в качестве значения всего предложения cond.
3. Если истинного предиката нет, то значением cond будет nil.

Рекомендуется в качестве последнего предиката использовать символ true, и соответствующее ему выражение будет вычисляться всегда в тех случаях, когда ни одно другое условие не выполняется.

В следующем примере с помощью предложения cond определена функция, определяющая тип выражения:

```
>(nil defmethod тип (l)
  (nil cond
    ((nil = l) 'пусто)
    ((nil atom l) 'атом)
    (true 'список)))
(lambda (l) (nil cond ((nil = l) 'пусто) ((nil atom l) 'атом) (true 'список)))
>(nil тип '(a b c))
список
>(nil тип (nil atom '(а т о м)))
атом
```

В условном предложении может отсутствовать выражение a_i или на его месте часто может быть несколько форм:

(nil cond
(p_1 $a_{1,1}$)
…
(p_i); выражение отсутствует
…
(p_k $a_{k,1}$ $a_{k,2}$ … $a_{k,n}$); несколько форм в качестве результата
…)

Если условию не ставится в соответствие выражение, то в качестве результата предложения cond при истинности предиката выдаётся само значение предиката. Если же условию соответствует несколько форм, то при его истинности формы вычисляются пошагово слева направо и результатом предложения cond будет значение последней формы (неявный progn).

В качестве примера использования условного предложения определим логические действия логики высказываний and, or, not, => (импликация) и <=> (тождество):

```
>(('logic defclass) defvar  value)
logic
>('logic  defmethod set (x)
  ('valuef  set x))
(lambda (x) ('value set x))
>('logic  defmethod and (x)
  (nil  cond
    (value  x)
    (true false)))
(lambda (x) (nil  cond  (value  x)  (true false)))
>('x  set ('logic newobject))
logic(EnvironmentLayer((this .  .. logic(EnvironmentLayer((this .  … ) (value . nil) ))) (value
. nil) ))
>(x set true)
true
>(x and false)
false
>('logic  defmethod or  (x)
  (nil  cond
    (value  true)
    (true x)))
(lambda (x) (nil  cond  (value  true) (true x)))
>(x or  false)
true
>('logic  defmethod not ()
  (nil  not value))
(lambda nil (nil  not value))
>(x not)
false
>('logic  defmethod =>  (x)
  (nil  cond
    (value  x)
    (true true)))
(lambda (x) (nil  cond  (value  x)  (true true)))
>(x set false)
false
>(x =>  true)
true
```

Импликацию можно определить и через другие операции:

```
('logic defmethod =>  (x)
  (this or  (x  not)))
('logic defmethod <=> (x)
  ((this  =>  x)  and (x  =>  this)))
```

Предикаты and и or входят в состав встроенных функций Лиспа. Число их аргументов может быть произвольным.

```
>(nil and (nil  atom  nil)  (nil  = nil)  (nil  eq  nil nil))
true
```

Предложения cond можно комбинировать таким же образом, как и вызовы функций. Например, предикат xor, который истинен при наличии единственного утвердительного аргумента, можно определить следующим образом:

```
>('logic  defmethod xor (x)
  (nil  cond
    (value
      (nil  cond
        (x  false)
        (true value)))
    (true x)))
(lambda (x) (nil  cond  (value  (nil  cond  (x  false)  (true value)))  (true x)))
>(x set true)
true
>(x xor false)
true
>(x set false)
false
>(x xor false)
false
```

В этой функции на месте выражения первого условия вновь стоит предложение cond. На месте, определённом условию, также можно использовать ещё одно условное предложение, и в этом случае мы получим условное условие. Такие построения очень быстро приводят к трудным определениям.

Другое условное предложение: IF

Предложение cond представляет собой наиболее общую условную структуру. Её критикуют за обилие скобок и за то, что структура этого предложения совершенно оторвана от естественного языка. Поэтому в Лисп-системах используются и другие, в различных отношениях более естественные, условные предложения. Но можно обойтись и с помощью лишь cond предложения.

В простом случае можно воспользоваться вполне естественной и содержащей мало скобок формой if.

```
(nil if условие то-форма иначе-форма)
≡
(nil cond
(    условие то-форма)
(true     иначе-форма))
>(nil if  (nil  atom  true) 'атом 'список)
атом
```

Циклические вычисления: предложения FOR, FOR*, WHILE и DO-WHILE

В случае повторяющихся вычислений в Лиспе используются известные в основном по процедурным языкам циклы.

```
(число for* переменная {участок для вычислений})
```

Пошагово локальной переменной присваиваются числа от 0 до число-1. При каждом таком значении вычисляется тело цикла.

Для примера с помощью предложения for* определим функцию, вычисляющую n-ую степень числа (n - целое, положительное):

```
>('number defmethod expt (n)
 (nil let ((результат 1))
  (n for* i
   (nil setq результат (результат * this)))
  результат))
(lambda (n) (nil let ((результат 1)) (n for* i (nil setq результат (результат * this))) результат))
>(2 expt 3)8
```

Цикл for отличается от for* тем, что в нём всё тело вычисляется параллельно, и каждый процесс имеет свою собственную переменную. Применяется данный цикл при обслуживании элементов вектора (в случае, если вычисления независимы от их порядка). Данные циклы определены ещё для обслуживания элементов списка и строк.

Следующий цикл while предназначен для пошагово выполняемых действий до получения удовлетворяющих результатов.

```
(nil while условие {участок для вычислений})
>(nil setq x 10 s 0)
nil
>(nil while (x > 0)
 (nil setq s (s + x))
 (nil setq x (x - 1)))
false
>s
55
```

Цикл do-while отличается от while только порядком проверки и действий:

(nil do-while {участок для вычислений} условие)

Задание:

1. Напишите программу, вычисляющую значения функции f(x,y) с применением cond:

$$f(x,y)=\begin{cases}\sqrt{x-y} & x>y,\\ x^2+y^2 & y>x,\\ 1 & x=y.\end{cases}$$

2. Напишите программу, вычисляющую значения функции f(x)= arccos (2*x*), используя функцию when или unless

С помощью функции if определите функцию xor(x,y).

3. Напишите программу, которая, для каждого дня недели выводит список предметов из Вашего расписания. В программе используйте функцию case.
4. Напишите игру «Угадай число». Программа «загадывает» число от 1 до 100, а пользователь пытается угадать его. Игра продолжается до тех пор, пока число не будет угадано.
5. Напишите игру «Камень-ножницы-бумага» . Игра продолжается до тех пор, пока в ней не выиграет пользователь.

Порядок выполнения:

5. Изучить теоретический материал;
6. Выполнить задания 1-7;
7. Запустить программы и протестировать их;
8. Составить отчет по лабораторной работе.

Форма отчетности:

Отчет по работе содержит:
1. Наименование лабораторной работы;
2. Разработанную программу;
3. Результаты её тестирования;
4. Выводы по работе.

Основная литература

1. Информатика. Базовый курс : учебник для бакалавров и специалистов / Под ред. С. В. Симоновича. - 3-е изд. - Санкт-Петербург : Питер, 2014. - 640 с.

Контрольные вопросы для самопроверки
1. Что такое префиксная нотация?
2. Как синтаксически оформляются комментарии?
3. Каковы особенности использования функций?
4. Перечислите базовые псевдо функции и кратко охарактеризуйте каждую из них.
5. Для чего используется функция eval?

Лабораторная работа № 3

Функции пользователя и рекурсии

Цель работы:

Изучить приемы разработки новых функций, их использование в программе. Разработать программы, использующие функции пользователя.

Теоретические сведения:

В Лиспе пользователь может создавать свои функции трех классов - EXPR, FEXPR и MACRO.

Для создания функции класса EXPR служит специальная встроенная функция DEFUN (определить функцию). Сама функция DEFUN принадлежит классу FSUBR. Вызов этой функции требует трех аргументов:

- Первый аргумент должен атомом;
- Второй аргумент должен быть списком атомов;
- Третий аргумент может быть произвольной формой (S-выражением, имеющим значение

Атом, который задан первым аргументом функции DEFUN, после возврата "превратится в функцию". Второй аргумент функции DEFUN называется списком формальных параметров. Третий аргумент функции DEFUN называется телом функции или определяющим выражением.

Если вызов функции DEFUN завершился удачно, то в качестве результата возвращается атом, заданный первым параметром, а в системе появляется новая функция, требующая при вызове столько аргументов, сколько элементов содержалось в списке формальных параметров. При вычислении этой новой функции значения аргументов будут вычисляться.

Итак, чтобы создать функцию, необходимо обратиться к порождающей функции DEFUN:

(DEFUN имя_функции (список_параметров) (тело_функции))

Построим простейшую арифметическую функцию, которая вычисляет разность квадратов своих аргументов. Для имени функции выберем атом QDIF (полагая, что такой функции еще нет). Список параметров нашей функции будет содержать два формальных параметра: (x y). Остается написать тело функции (S-выражение, вычисляющее разность квадратов значений x и y:

(* (- x y) (+ x y))

Собирая все вместе, получим:

(DEFUN QDIF (x y) (* (- x y) (+ x y)))

Пусть читатель обратит внимание на то, что тело функции содержит атомы, входящие в список формальных параметров. Можно создать функцию, тело которой не обращается к формальным параметрам. Однако, такая функция при любых параметрах давала бы один и

тот же результат...
Как работает функция? При вызове:

(QDIF Арг-1 Арг-2)

сначала значение Арг-1 присваивается атому x, значение Арг-2 - атому y. Затем вычисляется тело функции. Результат вычисления возвращается в качестве результата вызова. Важно подчеркнуть, что значения аргументов вычисляет не наша функция QDIF, а ядро Лиспа. Поскольку функция QDIF, создана вызовом DEFUN, то она принадлежит к классу EXPR, т.е. ядро Лиспа "знает", что значения аргументов функции QDIF перед вызовом нужно вычислять.

Последующие попытки вызвать функции QDIF с одним или тремя аргументами вызывают ошибку. Признак ошибки - возврат атома ERRSTATE в качестве результата. Как правило, ошибка сопровождается диагностическим сообщением (которое не нуждается в комментариях).
Далее заводятся две переменные x и y и им присваивается значение 6. Последующий вызов функции с аргументами 12 и 11 дает правильный результат. А вот проверка значений атомов x и y может удивить. Ведь, в соответствии со сказанным выше, переменным x и y при вызове функции должны были присвоиться значения 12 и 11. Тем не менее, значения переменных x и y остаются теми же, какими были до вызова функции!

Будем называть атомы, входящие в список формальных параметров какой-либо функции переменными, связанными в теле этой функции. Переменные, не входящие в список формальных параметров, но использующиеся в теле функции, будем называть свободными переменными.
А вот если функция каким-либо образом изменит значение свободной переменной, то это изменение останется и после выхода из функции. Как же функция может изменить значение свободной переменной? Употреблением функций SET/SETQ. Изменим тело нашей функции QDIF следующим образом:

(DEFUN QDIF (x y) (setq z (* (- x y) (+ x y))))

Здесь тело функции представляет собой вызов SETQ, который присваивает свободной переменной z значение разности квадратов, которое вычисляется по прежней формуле. Поскольку SETQ возвращает значение второго аргумента, функция QDIF не теряет работоспособности, однако, если присвоить атому z какое-либо значение, то после вызова оно будет заменено разностью квадратов:

```
(DEFUN QDIF (x y) (setq z (* (- x y) (+ x y))))
==> QDIF
(setq z 0)
==> 0
(qdif 5 3)
==> 16
z
==> 16
```

Задание:

1. Напишите функцию пользователя, сравнивающую остатки от деления N чисел x1, x2, …на число k и возвращающая число xi, дающее больший остаток.
2. Напишите рекурсивную функцию для вычисления n-й степени числа a
3. Напишите рекурсивную функцию, проверяющую, что все элементы списка S различны

Порядок выполнения:

1. Изучить теоретический материал;
2. Выполнить задания;
3. Запустить программы и протестировать их;
4. Составить отчет по лабораторной работе.

Форма отчетности:

Отчет по работе содержит:
1. Наименование лабораторной работы;
2. Разработанную программу;
3. Результаты её тестирования;
4. Выводы по работе.

Основная литература

1. Информатика. Базовый курс : учебник для бакалавров и специалистов / Под ред. С. В. Симоновича. - 3-е изд. - Санкт-Петербург : Питер, 2014. - 640 с.

Контрольные вопросы для самопроверки
1. Как объявляется функция пользователя?
2. Что означает ключевое слово defun?
3. Какие ветви должна содержать рекурсия?
4. Опишите принципы построения рекурсивных функций для работы со списками.

Лабораторная работа № 4

Макросы и функционалы

Цель работы:

Изучить приемы разработки функционалов, их использование в программе. Разработать программы, использующие макросы

Теоретические сведения:

Понятие функционала и отображения

Отображения - ключевой механизм информатики. Построение любой информационной системы сопровождается определением и реализацией большого числа отображений. Сначала выбираются данные, с помощью которых представляется информация. В результате по данным можно восстановить представленную ими информацию - извлечь информацию из данных (по записи числа восстановить его величину). Потом конструируется набор структур, достаточный для размещения и обработки данных и программ в памяти компьютера (по коду команды можно выбрать хранимую в памяти подпрограмму, которая построит новые коды чисел или структур данных).

Функционалы - это функции, которые используют в качестве аргументов или результатов другие функции. При построении функционалов переменные могут играть роль имен функций, определения которых находятся во внешних формулах, использующих функционалы.

Пример: Построить список из <голов> элементов списка

```
(defun 1st (xl)
(cond ; "головы" элементов = CAR пока список не пуст
(xl (cons (caar xl); выбираем CAR от его головы
(1st (cdr xl)) ; и переходим к остальным,
) ) ) ) ; собирая результаты в список
(1st`((один два)(one two)(1 2)) ) ; = (один one 1)
```

Пример: Выяснить длины элементов списка

```
(defun lens (xl) ; Длины элементов
(cond ; Пока список не пуст
(xl (cons (length (car xl))
; вычисляем длину его головы
(lens (cdr xl)); и переходим к остальным,
) ) ) ) ; собирая результаты в список
(lens `((1 2)()(a b c d)(1(a b c d)3))) ; = (2 0 4 3)
```

Внешние отличия в записи этих трех функций малосущественны, что позволяет ввести более общую функцию mapcar, в определении которой имена <car> и <length> могут быть заданы как значения параметра fn:

```
(defun mapcar(fn xl)
(cond ; Поэлементное преобразование XL с помощью функции FN пока XL не пуст
(xl (cons(funcall fn (car xl)) ; применяем FN как функцию голове XL
(mapcar fn (cdr xl))
; и переходим к остальным,
) ) ) ) ; собирая результаты в список
```

Эффект функций 1st и lens можно получить выражениями:

```
; (mapcar `car xl) ; "головы" элементов = CAR
; (mapcar `length xl) ; Длины элементов
(mapcar `car `((один два)(one two)(1 2)) ) ; = (один one 1)
(mapcar `length `((1 2)()(a b c d)(1(a b c d)3)) ) ; = (2 0 4 3)
```

Оба примера можно решить с помощью таких определяющих выражений:

```
(defun 1st(xl) (mapcar `car xl)) ; "головы" элементов = CAR
(defun lens(xl) (mapcar `length xl)) ; Длины элементов
```

Эти определения функций формально эквивалентны ранее приведенным - они сохраняют отношение между аргументами и результатами. Параметром функционала может быть любая вспомогательная функция.

Применяющие функционалы

Одним из видов функционалов, используемых в Лиспе являются применяющие функционалы. Они применяют функциональный аргумент к его параметрам. Так как применяющие функционалы вычисляют значение функции, в этом смысле они аналогичны функции EVAL, вычисляющей значение выражения.

Функционал APPLY

Предположим мы хотим объединить в один список несколько вложенных списков, т.е. из

```
((a b c) (d e f) (k l))
```

получить

```
(a b c d e f k l).
```

Для этой задачи используем функцию apply, которая имеет два аргумента: имя функции и список, и применяет названную функцию к элементам списка, как к аргументам функции.

Пример: Определим функцию, которая рассчитывает среднее списка чисел

```
(defun list-mean (x)
(/ (apply `+ x) (length x))
)
(list-mean `(1 2 3 4))
; 2.5
```

Часто apply используют вместе с mapcar.

Пример: найти общее число элемент в списках

```
(defun countall (lis)
(apply `+ (mapcar ` length lis))
)
(countall `((a b c) (d e f) (k l)))
; 8
```

Можно определить более сложную функцию countatom, которая считает элементы-атомы в любом списке.

```
(defun countatom (lis)
(cond((null lis) 0)((atom lis) 1)(t (apply `+ (mapcar `countatom lis)))))
(countatom `(a (a (b) c) (d) e (f g)));
```

Функционал FUNCALL

Применяющий функционал FUNCALL аналогичен APPLY, но аргументы он принимает, не в списке, а по отдельности:

```
(funcall fn x1 x2 ... xN) <=> (fn x1 x2 ... xN)
```

Здесь fn - функция с n аргументами.

(funcall '+ 1 2) ; <=> * (+ 1 2)
; 3
(funcall (car '(+ - / *)) 1 2)
; 3
Пример:
Map
(map result-type function sequences ...)

Функция function вызывается на всех первых элементах последовательностей, затем на всех вторых и т.д. Из полученных результатов function формируется результирующая последовательность, строение которой задается параметром result-type с допустимыми значениями cons, list, array, string, NIL.

(map `list `+ `(1 2 3) `(4 5 6))
; = (5 7 9)
Mapcar
(mapcar function list ...)

Функция function применяется к первым элементам списков, затем ко вторым и т.д. Другими словами, function применяется к <головам> методично сокращающихся списков, и результаты применения собираются в результирующий список.

(mapcar `list `(1 2 3) `(4 5 6))
; = ((1 4)(2 5)(3 6))
Maplist
(maplist function list ...)

Функционал аналогичен mapcar, но function применяется к <<хвостам>> списков list, начиная с полного списка.

(maplist `list `(1 2 3) `(4 5 6))
; = (((1 2 3) (4 5 6)) ((2 3) (5 6)) ((3) (6)))
Mapc и Mapl

Оба функционала работают как mapcar и maplist, соответственно, за исключением того, что они в качестве формального результата выдают первый список (своеобразная аналогия с формальными аргументами).

(mapc `list `(1 2 3) `(4 5 6)) ; = (1 2 3)
(mapl `list `(1 2 3) `(4 5 6)) ; = (1 2 3)
Mapcan и Mapcon

И эти два функционала аналогичны mapcar и maplist, но формирование результатов происходит не с помощью операции cons, которая строит данные в новых блоках памяти, а с помощью деструктивной функции nconc, которая при построении новых данных использует память исходных данных, из-за чего исходные данные могут быть искажены.

(mapcan `list `(1 2 3 4)) ; (1 2 3 4)
(mapcon `list `(1 2 3 4)) ; ((1 2 3 4) (2 3 4) (3 4) (4))
Map-into

Функционал отображает результат в конкретную последовательность.

(setq a (list 1 2 3 4) b (list 10 10 10 10)) ; (10 10 10 10)
(map-into a `+ a b) ; (11 12 13 14)

Пример: Для заданного списка вычислим ряд его атрибутов, а именно - длина, первый элемент, остальные элементы списка без первого.

```
(defun mapf (fl el)
(cond ; Пока первый аргумент не пуст,
(fl (cons (funcall (car fl) el)
; применяем очередную функцию
; ко второму аргументу
(mapf (cdr fl) el)
; и переходим к остальным функциям,
) ) ) ) ; собирая их результаты в общий список
```

Рассмотрим еще один типичный вариант применения функционалов. Представим, что нас интересуют некие интегральные характеристики результатов, полученных при отображении, например, сумма полученных чисел, наименьшее или наибольшее из них и т.п. В таком случае говорят о свертке результата или его редукции. Редукция заключается в сведении множества элементов к одному элементу, в вычислении которого задействованы все элементы множества.

Пример: Подсчитать сумму элементов заданного списка.

```
(defun sum-el (xl)(cond((null xl) 0)(xl (+(car xl)(sum-el (cdr xl) ))))  )
(sum-el `(1 2 3 4) ) ; = 10
```

Перестроим такое определение, чтобы вместо <+> можно было использовать произвольную бинарную функцию:

```
(defun red-el (fn xl)(cond((null xl)0)(xl(funcall fn(car xl)(red-el fn (cdr xl))))))
(red-el `+ `(1 2 3 4) ) ; = 10
```

Задание:

1. Используя функционал, напишите функцию, которая по списку a составляет список из разностей соседних элементов
2. Определите на языке ЛИСП функционал, вставляющий перед каждым элементом списка, обладающим определенным свойством, символ *. Проверьте работу функционала для предикатов:
 -неотрицательное число (при вызове используйте лямбда-функцию);
 -четное число.
3. Реализовать функцию перемножения многочленов, если многочлены задаются в виде списков их коэффициентов.
4. Напишите функцию, которая преобразовывает линейные функции y=kx+b в обратные к ним. Результат должен представлять собой вычислимую функцию.
5. Напишите функцию, которая принимает в инфиксной нотации выражения вида (a + b), (a - b), (a * b), (a / b) и преобразует их в вычислимые выражения (+ a b) и т.д.

Порядок выполнения:

5. Изучить теоретический материал;
6. Выполнить задания;
7. Запустить программы и протестировать их;
8. Составить отчет по лабораторной работе.

Форма отчетности:

Отчет по работе содержит:

1. Наименование лабораторной работы;
2. Разработанную программу;
3. Результаты её тестирования;
4. Выводы по работе.

Основная литература

1. Информатика. Базовый курс : учебник для бакалавров и специалистов / Под ред. С. В. Симоновича. - 3-е изд. - Санкт-Петербург : Питер, 2014. - 640 с.

Контрольные вопросы для самопроверки

1. Что такое функционал?
2. Какова отличительная особенность функционалов семейства Map?
3. Какие вычисления выполняет Mapcar?
4. Как формируют результаты Mapcan и Mapcon?
5. Как с помощью функционалов подсчитать сумму элементов списка?

9.2. Методические указания по выполнению контрольной работы

Контрольная работа представляет собой способ проверки знаний студента, его умений и предполагают письменные ответы на поставленные вопросы, либо самостоятельное выполнение практических заданий. Подготовка к контрольной работе состоит в ответственном выполнении всех домашних заданий по дисциплине и самостоятельной проработке основной и дополнительной литературы.

Целью контрольной работы является приобретение навыков самостоятельной работы с литературой, закрепление умений работы со средой программирования, формирование навыков оценки результатов собственной деятельности.

Выполнения контрольной работы предполагает:

- анализ поставленных задач и выбор методов их решения;
- реализацию решения поставленных задач;
- проверку и анализ полученных результатов;
- оформление отчета.

Отчет по контрольной работе оформляется в печатном виде и содержит:

- формулировку заданий;
- описание их решений;
- полученные результаты;
-выводы.

10. ПЕРЕЧЕНЬ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, ИСПОЛЬЗУЕМЫХ ПРИ ОСУЩЕСТВЛЕНИИ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ

1. Microsoft Imagine Premium: Microsoft Windows Professional 7;
2. Microsoft Office 2007 Russian Academic OPEN No Level;
3. Антивирусное программное обеспечение Kaspersky Security;
4. ОС Linux;
5. LibreOfice
6. CLISP

11. ОПИСАНИЕ МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЙ БАЗЫ, НЕОБХОДИМОЙ ДЛЯ ОСУЩЕСТВЛЕНИЯ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ

Вид занятия	***Наименование аудитории***	***Перечень основного оборудования***	***№ ЛР***
1	**2**	**3**	**4**
Лк	Лекционная аудитория	-	-
ЛР	Лаборатория параллельных вычислений	Персональные компьютеры i5-2500/H67/4Gb/500Gb (монитор TFT19 Samsung E1920NR); интерактивная доска Smart Board X885ix со встроенным проектором UX60	№ 1-4
кр	Лаборатория параллельных вычислений	Персональные компьютеры i5-2500/H67/4Gb/500Gb (монитор TFT19 Samsung E1920NR);	-
СР	ЧЗ1	Оборудование 10 ПК i5-2500/H67/4Gb(монитор TFT19 Samsung); принтер HP LaserJet P2055D	-

Приложение 1

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

1. Описание фонда оценочных средств (паспорт)

№ компетенции	Элемент компетенции	Раздел	Тема	ФОС
ОПК-1	Способность использовать базовые знания естественных наук, математики и информатики, основные факты, концепции, принципы теорий, связанных с прикладной математикой и информатикой	**1**. Основные элементы функционального языка	1.1. Функциональная парадигма программирования	Индивидуальное задание, экзаменационный вопрос
			1.2. Синтаксис и семантика языка Lisp	Индивидуальное задание, экзаменационный вопрос
			1.3. Списочная организация данных	Индивидуальное задание, экзаменационный вопрос
ОПК-3	Способность к разработке алгоритмических и программных решений в области системного и прикладного программирования, математических, информационных и имитационных моделей		1.4. Конструкции ветвления	Индивидуальное задание, экзаменационный вопрос
			1.5. Организация циклов	Индивидуальное задание, экзаменационный вопрос
			1.6. Свойства символа	Индивидуальное задание, экзаменационный вопрос
			1.7. Ввод/вывод	Индивидуальное задание, экзаменационный вопрос
ПК-7	Способность к разработке и применению алгоритмических и программных решений в области системного и прикладного программного обеспечения	**2**.Функциональный подход к решению задач	2.1 Лямбда функции	Индивидуальное задание, экзаменационный вопрос
			2.2 Функции пользователя	Индивидуальное задание, экзаменационный вопрос
			2.3. Рекурсии	Индивидуальное задание, экзаменационный вопрос
			2.4. Функционалы	Индивидуальное задание, экзаменационный вопрос
			2.5. Макросы	Индивидуальное задание, экзаменационный вопрос

2. Экзаменационные вопросы

№ п/п	Код Компетенции	Определение	ЭКЗАМЕННАЦИОННЫЕ ВОПРОСЫ	№ и наименование раздела
1	2	3	4	5
1.	**ОПК-1**	Способность использовать базовые знания естественных наук, математики и информатики, основные факты, концепции, принципы теорий, связанных с прикладной математикой и информатикой	1. Классификация языков программирования по применяемой парадигме	**1.** Основные элементы функционального языка
			2. Предикаты проверки списков и атомов.	**1.** Основные элементы функционального языка
			3. Выражения, атомы и списки.	**1.** Основные элементы функционального языка
			4. Внутреннее представление списков.	**1.** Основные элементы функционального языка
			5. Базовые функции обработки списков.	**1.** Основные элементы функционального языка
2.	**ОПК-3**	Способность к разработке алгоритмических и программных решений в области системного и прикладного программирования, математических, информационных и имитационных моделей	6. Конструкции if, when, until.	**1.** Основные элементы функционального языка
			7. Сложное ветвление cond.	**1.** Основные элементы функционального языка
			8. Форма выбора case.	**1.** Основные элементы функционального языка
			9. Цикл loop.	**1.** Основные элементы функционального языка
			10. Форма do.	**1.** Основные элементы функционального языка
			11. Макрос with-open-file.	**1.** Основные элементы функционального языка
3.	**ПК-7**	Способность к разработке и применению алгоритмических и программных решений в области системного и прикладного программного обеспечения	12. Задание параметров в лямбда-списке.	**2.** Функциональный подход
			13. Определение функции пользователя.	**2.** Функциональный подход
			14. Применение рекурсии для обработки списков.	**2.** Функциональный подход
			15. Функционалы: основные определения. Отображающие функционалы.	**2.** Функциональный подход
			16. Применяющие функционалы.	**2.** Функциональный подход
			17. Понятие макроса. Контекст вычисления макроса.	**2.** Функциональный подход

3. Описание показателей и критериев оценивания компетенций

Показатели	Оценка	Критерии
Знать ПК-1: - способы сбора и обработки информации; ОПК-3: - основные алгоритмы решения задач ; ПК-7: - принципы построения программных решений на функциональных языках; **Уметь** ПК-1: - применять аппарат математической статистики для обработки данных; ОПК-3: - разрабатывать программные решения в области прикладного программного обеспечения **Владеть** ПК-1: – методами и приемами обработки данных и интерпретации результатов ОПК-3: - приемами построения алгоритмических и программных решений ПК-7: разрабатывать программы на функциональных языках ПК-7: – навыками разработки программных решений в функциональном стиле	**отлично**	Демонстрирует все показатели компетенций на высоком уровне, а именно: -знает математические основы функционального программирования; -знает основные алгоритмы решения задач ; -знает принципы построения программных решений на функциональных языках; -умеет использовать технологии решения интеллектуальных задач в функциональном стиле; -умеет разрабатывать алгоритмы решения задач из области системного и прикладного программирования; -умеет разрабатывать программы на функциональных языках ; -владеет методами решения задач профессиональной деятельности; .- приемами построения алгоритмических и программных решений.
	хорошо	Демонстрирует освоенность не менее 7 показателей компетенций:, а именно -знает математические основы функционального программирования; -знает основные алгоритмы решения задач ; -знает принципы построения программных решений на функциональных языках; -умеет использовать технологии решения интеллектуальных задач в функциональном стиле; -умеет разрабатывать алгоритмы решения задач из области системного и прикладного программирования; -умеет разрабатывать программы на функциональных языках ; -владеет методами решения задач профессиональной деятельности; или -знает математические основы функционального программирования; -знает основные алгоритмы решения задач ; -знает принципы построения программных решений на функциональных языках; -умеет использовать технологии решения интеллектуальных задач в функциональном стиле; -умеет разрабатывать алгоритмы решения задач из области системного и прикладного программирования; -умеет разрабатывать программы на функциональных языках ;

		-владеет приемами построения алгоритмических и программных решений; или -знает принципы построения программных решений на функциональных языках; -умеет использовать технологии решения интеллектуальных задач в функциональном стиле; -умеет разрабатывать алгоритмы решения задач из области системного и прикладного программирования; -умеет разрабатывать программы на функциональных языках ; -владеет методами решения задач профессиональной деятельности; -владеет приемами построения алгоритмических и программных решений; -владеет навыками разработки программных решений в функциональном стиле.
	удовлетворительно	Демонстрирует освоение не менее 5 параметров компетенций, а именно: -знает математические основы функционального программирования; -знает основные алгоритмы решения задач ; -знает принципы построения программных решений на функциональных языках; -умеет использовать технологии решения интеллектуальных задач в функциональном стиле; -умеет разрабатывать алгоритмы решения задач из области системного и прикладного программирования; или -знает математические основы функционального программирования; -знает основные алгоритмы решения задач ; -знает принципы построения программных решений на функциональных языках; -умеет использовать технологии решения интеллектуальных задач в функциональном стиле; -умеет разрабатывать программы на функциональных языках ; или -знает математические основы функционального программирования; -знает основные алгоритмы решения задач ; -знает принципы построения программных решений на функциональных языках; -умеет использовать технологии решения интеллектуальных задач в функциональном стиле; -владеет методами решения задач профессиональной деятельности; или -знает математические основы функционального программирования;

		-знает основные алгоритмы решения задач ; -знает принципы построения программных решений на функциональных языках; -умеет использовать технологии решения интеллектуальных задач в функциональном стиле; -владеет приемами построения алгоритмических и программных решений; или -знает математические основы функционального программирования; -знает основные алгоритмы решения задач ; -знает принципы построения программных решений на функциональных языках; -умеет использовать технологии решения интеллектуальных задач в функциональном стиле; -владеет навыками разработки программных решений в функциональном стиле; или -знает математические основы функционального программирования; -знает основные алгоритмы решения задач ; -знает принципы построения программных решений на функциональных языках; -умеет разрабатывать алгоритмы решения задач из области системного и прикладного программирования; -умеет разрабатывать программы на функциональных языках ; или -знает математические основы функционального программирования; -знает основные алгоритмы решения задач ; -знает принципы построения программных решений на функциональных языках; -умеет разрабатывать алгоритмы решения задач из области системного и прикладного программирования; -владеет методами решения задач профессиональной деятельности; -умеет разрабатывать алгоритмы решения задач из области системного и прикладного программирования; -умеет разрабатывать программы на функциональных языках ; -владеет методами решения задач профессиональной деятельности; -владеет приемами построения алгоритмических и программных решений; -владеет навыками разработки программных решений в функциональном стиле
	неудовлетворительно	Демонстрирует менее половины сформированных параметров компетенций

4. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и опыта деятельности

Дисциплина Функциональное программирование направлена на ознакомление обучающихся с парадигмой функционального программирования; на получение теоретических знаний и практических навыков разработки функциональных программ для идентификации, формулирования и решения проблем из различных областей науки и производства, , а также осуществления поиска, хранения, обработки и анализа информации из различных источников и представления ее в соответствующем виде и для их дальнейшего использования в практической деятельности.

Изучение дисциплины Функциональное программирование предусматривает:

- лекции,
- лабораторные работы;
- контрольную работу;
- экзамен;
- самостоятельную работу.

Для фиксирования успешности обучения предусматривается экзамен.

В ходе освоения раздела 1 «Основные элементы функционального языка» обучающиеся должны уяснить способы получения экспертных оценок, их анализ и обработку, научиться формулировать собственные идеи и представления об известной им предметной области.

В ходе освоения раздела 2 «Функциональный подход к решению задач» обучающиеся осваивают принципы проектирования и построения функциональных программ, формализацию представления знаний, идеи разработки программ в логическом стиле, основные конструкции языка программирования, способы представления данных и знаний.

Студентам необходимо овладеть навыками и умениями применения изученных методов для разработки и реализации профессионально ориентированных проектов в последующей учебной деятельности.

Овладение ключевыми понятиями является основой усвоения учебного материала по дисциплине.

При подготовке к экзамену особое внимание необходимо уделить рекомендациям и замечаниям преподавателей, ведущих аудиторные занятия по дисциплине

В процессе проведения лабораторных занятий происходит закрепление знаний, формирование умений и навыков применения различных методов решения стандартных математических ситуаций.

Самостоятельную работу необходимо начинать с чтения лекций и учебников.

В процессе консультации с преподавателем обучающийся выясняет наличие пробелов в знаниях и способах решения разных ситуаций.

Работа с литературой является важнейшим элементом в получении знаний по дисциплине. Прежде всего, необходимо воспользоваться списком рекомендуемой по данной дисциплине литературой. Дополнительные сведения по изучаемым темам можно найти в периодической печати и Интернете.

Предусмотрено проведение аудиторных занятий в виде разнообразных тренингов и ситуаций общения в сочетании с внеаудиторной работой.

Приложение 2

АННОТАЦИЯ
рабочей программы дисциплины
Функциональное программирование

1. Цель и задачи дисциплины

Целью изучения дисциплины является: ознакомление обучающихся с различными методами, приемами функционального программирования, приемами интеграции одних программных пакетов в другие и использованию результатов интеграции при создании собственных сложных универсальных программных комплексов.

Задачами дисциплины являются

- обучение методам компьютерного формализованного представления знаний и реализации выводов для последующей выработки и принятия человеком вариантов принимаемого решения;
- формирование умения и навыков самостоятельного исследования и решения различного рода задач путем применения средств функционального программирования совместно с другими видами программного обеспечения;
- формирование и развитие умений и навыков, позволяющих применять современные математические методы и программное обеспечение для решения задач науки и техники.

2. Структура дисциплины

2.1 Распределение трудоемкости по отдельным видам учебных занятий, включая самостоятельную работу: Лк.- 17 час., ЛР - 51 час.; СР - 40 час.

Общая трудоемкость дисциплины составляет 144 часов, 4 зачетных единиц.

2.2 Основные разделы дисциплины:

1 – Основные элементы функционального языка
2 – Функциональный подход к решению задач

3. Планируемые результаты обучения (перечень компетенций)

Процесс изучения дисциплины направлен на формирование следующих компетенций:

ОПК-1 – Способность использовать базовые знания естественных наук, математики и информатики, основные факты, концепции, принципы теорий, связанных с прикладной математикой и информатикой

ОПК-3 – Способность к разработке алгоритмических и программных решений в области системного и прикладного программирования, математических, информационных и имитационных моделей, созданию информационных ресурсов глобальных сетей, образовательного контента, прикладных баз данных, тестов и средств тестирования систем и средств на соответствие стандартам и исходным требованиям

ПК-7 – Способность к разработке и применению алгоритмических и программных решений в области системного и прикладного программного обеспечения

4. Вид промежуточной аттестации: экзамен.

Приложение 3

Протокол о дополнениях и изменениях в рабочей программе
на 20___-20___ учебный год

1. В рабочую программу по дисциплине вносятся следующие дополнения:

_____
_____

2. В рабочую программу по дисциплине вносятся следующие изменения:

_____
_____
_____

Протокол заседания кафедры №____ от «___» _____ 20 ____ г.,
(разработчик)

Заведующий кафедрой _____ _____
(подпись) *(Ф.И.О.)*

Приложение 4

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ ПО ДИСЦИПЛИНЕ

1. Описание фонда оценочных средств (паспорт)

№ компетенции	Элемент компетенции	Раздел	Тема	ФОС
ОПК-1	Способность использовать базовые знания естественных наук, математики и информатики, основные факты, концепции, принципы теорий, связанных с прикладной математикой и информатикой	1. Основные элементы функционального языка	1.2. Синтаксис и семантика языка Lisp	ЛР№1
			1.3. Списочная организация данных	ЛР№1, кр
ОПК-3	Способность к разработке алгоритмических и программных решений в области системного и прикладного программирования, математических, информационных и имитационных моделей		1.4. Конструкции ветвления	ЛР№2
			1.5. Организация циклов	ЛР№2
			2.2 Функции пользователя	ЛР№
			2.3. Рекурсии	ЛР№3
ПК-7	Способность к разработке и применению алгоритмических и программных решений в области системного и прикладного программного обеспечения	2.Функциональный подход к решению зада	2.4. Функционалы	ЛР№4, кр
			2.5. Макросы	ЛР№4, кр

2. Описание показателей и критериев оценивания компетенций

Показатели	Оценка	Критерии
Знать ПК-1: - способы сбора и обработки информации; ОПК-3:	**отлично**	Демонстрирует все показатели компетенций на высоком уровне, а именно: -знает математические основы функционального программирования; -знает основные алгоритмы решения задач ; -знает принципы построения программных решений на функциональных языках;

- основные алгоритмы решения задач ; ПК-7: - принципы построения программных решений на функциональных языках; **Уметь** ПК-1: - применять аппарат математической статистики для обработки данных; ОПК-3: - разрабатывать программные решения в области прикладного программного обеспечения **Владеть** ПК-1: – методами и приемами обработки данных и интерпретации результатов ОПК-3: - приемами построения алгоритмических и программных решений ПК-7: разрабатывать программы на функциональных языках ПК-7: – навыками разработки программных решений в функциональном стиле		-умеет использовать технологии решения интеллектуальных задач в функциональном стиле; -умеет разрабатывать алгоритмы решения задач из области системного и прикладного программирования; -умеет разрабатывать программы на функциональных языках ; -владеет методами решения задач профессиональной деятельности; .- приемами построения алгоритмических и программных решений.
	хорошо	Демонстрирует освоенность не менее 7 показателей компетенций:, а именно -знает математические основы функционального программирования; -знает основные алгоритмы решения задач ; -знает принципы построения программных решений на функциональных языках; -умеет использовать технологии решения интеллектуальных задач в функциональном стиле; -умеет разрабатывать алгоритмы решения задач из области системного и прикладного программирования; -умеет разрабатывать программы на функциональных языках ; -владеет методами решения задач профессиональной деятельности; или -знает математические основы функционального программирования; -знает основные алгоритмы решения задач ; -знает принципы построения программных решений на функциональных языках; -умеет использовать технологии решения интеллектуальных задач в функциональном стиле; -умеет разрабатывать алгоритмы решения задач из области системного и прикладного программирования; -умеет разрабатывать программы на функциональных языках ; -владеет приемами построения алгоритмических и программных решений; или -знает математические основы функционального программирования; -знает основные алгоритмы решения задач ; -знает принципы построения программных решений на функциональных языках; -умеет использовать технологии решения интеллектуальных задач в функциональном стиле; -умеет разрабатывать алгоритмы решения задач из области системного и прикладного программирования;

		-умеет разрабатывать программы на функциональных языках ; -владеет навыками разработки программных решений в функциональном стиле; -знает принципы построения программных решений на функциональных языках; -умеет использовать технологии решения интеллектуальных задач в функциональном стиле; -умеет разрабатывать алгоритмы решения задач из области системного и прикладного программирования; -умеет разрабатывать программы на функциональных языках ; -владеет методами решения задач профессиональной деятельности; -владеет приемами построения алгоритмических и программных решений; -владеет навыками разработки программных решений в функциональном стиле.
	удовлетво рительно	Демонстрирует освоение не менее 5 параметров компетенций, а именно: -знает математические основы функционального программирования; -знает основные алгоритмы решения задач ; -знает принципы построения программных решений на функциональных языках; -умеет использовать технологии решения интеллектуальных задач в функциональном стиле; -умеет разрабатывать алгоритмы решения задач из области системного и прикладного программирования; или -знает математические основы функционального программирования; -знает основные алгоритмы решения задач ; -з -знает математические основы функционального программирования; -знает основные алгоритмы решения задач ; -знает принципы построения программных решений на функциональных языках; -умеет использовать технологии решения интеллектуальных задач в функциональном стиле; -владеет приемами построения алгоритмических и программных решений; или -знает математические основы функционального программирования; -знает основные алгоритмы решения задач ; -знает принципы построения программных решений на функциональных языках; -умеет использовать технологии решения интеллектуальных задач в функциональном стиле;

		-владеет навыками разработки программных решений в функциональном стиле; или -знает математические основы функционального программирования; -знает основные алгоритмы решения задач ; -знает принципы построения программных решений на функциональных языках; -умеет разрабатывать алгоритмы решения задач из области системного и прикладного программирования; -умеет разрабатывать программы на функциональных языках ; или -знает математические основы функционального программирования; -знает основные алгоритмы решения задач ; -знает принципы построения программных решений на функциональных языках; -умеет разрабатывать алгоритмы решения задач из области системного и прикладного программирования; -владеет методами решения задач профессиональной деятельности; или -умеет разрабатывать алгоритмы решения задач из области системного и прикладного программирования; -умеет разрабатывать программы на функциональных языках ; -владеет методами решения задач профессиональной деятельности; -владеет приемами построения алгоритмических и программных решений; -владеет навыками разработки программных решений в функциональном стиле
	неудовлетворительно	Демонстрирует менее половины сформированных параметров компетенций

Программа составлена в соответствии с федеральным государственным образовательным стандартом высшего образования по направлению подготовки 01.03.02 Прикладная математика и информатика от «12» марта 2015 г. № 228

для набора 2015 года: и учебным планом ФГБОУ ВО «БрГУ» для очной формы обучения от «13 » июля 2015 г. № 475

для набора 2016 года: и учебным планом ФГБОУ ВО «БрГУ» для очной формы обучения от «06» июня 2016г. № 429

для набора 2017 года: и учебным планом ФГБОУ ВО «БрГУ» для очной формы обучения от «6» марта 2017г. № 125

для набора 2018 года и учебным планом ФГБОУ ВО «БрГУ» для очной формы обучения от «12» марта 2018г. №130

Программу составили:

Багинова Т.Г. , к.т.н, доцент каф. МиФ _____

Ратинская Е.В., ст. препод. каф. МиФ _____

Рабочая программа рассмотрена и утверждена на заседании кафедры МиФ

от «21» ноября 2018 г., протокол № 3

И.о. зав.выпускающей кафедрой _____ О.И.Медведева

СОГЛАСОВАНО:

И.о. зав.выпускающей кафедрой _____ О.И Медведева.

Директор библиотеки _____ Т.Ф.Сотник

Рабочая программа одобрена методической комиссией ЕН факультета

от «20 » декабря 2018 г., протокол № 4

Председатель методической комиссии факультета _____ М.А. Варданян

СОГЛАСОВАНО:

Начальник
учебно-методического управления _____ Г.П. Нежевец

Регистрационный №_____