МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования

«Дальневосточный федеральный университет»

(ДВФУ)

ШКОЛА ЕСТЕСТВЕННЫХ НАУК ДВФУ

«СОГЛАСОВАНО»
Руководитель ОП
д.ф.-м.н., профессор, академик РАН, Гузев М.А.

_____
(подпись) (Ф.И.О. рук. ОП)
«9» июля 2018 г.

«УТВЕРЖДАЮ»
Заведующая (ий) кафедрой
информатики, математического и компьютерного моделирования
(название кафедры)

_____ Чеботарев А.Ю.
(подпись) (Ф.И.О. зав. каф.)
«9» июля 2018 г.

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ (РПУД)

Технология программирования

Направление подготовки 09.03.03 Прикладная информатика

Профиль: «Прикладная информатика в компьютерном дизайне»

Форма подготовки очная

курс 3 курс 5,6 семестр
лекции 54 час.
практические занятия час.
лабораторные работы 90 час.
в том числе с использованием МАО лек.0/ пр. 0, лаб. 0
всего часов аудиторной нагрузки 144 час.
в том числе с использованием МАО 50
самостоятельная работа 70 час.
курсовая работа/курсовой проект 6 семестр
зачет 3,4 семестр

Рабочая программа составлена в соответствии с требованиями образовательного стандарта, самостоятельно установленного ДВФУ, принятого решением Ученого совета Дальневосточного федерального университета, протокол от 28.01.2016 № 01-16, и введенного в действие приказом ректора ДВФУ от 18.02.2016 № 12-13-235.

Рабочая программа обсуждена на заседании кафедры информатики, математического и компьютерного моделирования, протокол №18 «9» июля 2018 г..

Заведующий кафедрой профессор Чеботарев А.Ю.

Составитель: старший преподаватель Кленина Н.В.

Оборотная сторона титульного листа РПУД

I. Рабочая программа пересмотрена на заседании кафедры:

Протокол от «_____» _____ 200 г. № _____

Заведующий кафедрой _____ _____

(подпись) (И.О. Фамилия)

II. Рабочая программа пересмотрена на заседании кафедры:

Протокол от «_____» _____ 200 г. № _____

Заведующий кафедрой _____ _____

(подпись) (И.О. Фамилия)

АННОТАЦИЯ

Учебно-методический комплекс дисциплины «Технология программирования» разработан для студентов 3 курса по направлению 09.03.03 «Прикладная информатика».

Дисциплина «Технология программирования» входит в базовую часть профессионального цикла.

Общая трудоемкость освоения дисциплины составляет 6 зачетных единиц, 216 часов. Учебным планом предусмотрены лекционные занятия (54 часов), лабораторные занятия (90 часов) самостоятельная работа студента (72 часа).

Цели освоения дисциплины

Содержание дисциплины охватывает знания о технологических принципах разработки и сопровождения программных систем среднего и большого размера, в том числе в составе коллектива разработчиков.

Рассматриваются основные цели технологического подхода к программированию — повышение воспроизводимости, надежности и эффективности процесса разработки программного обеспечения.

Уделяется внимание глубокому изучению наиболее распространённых конкретных технологий программирования, используемых ими организационных и технических инструментов.

Также поверхностно рассматриваются юридические, экономические, этические и философские аспекты деятельности программиста.

Задачи дисциплины

Дисциплина должна:

1. познакомить студентов с общими технологическими принципами разработки и сопровождения программных систем;

2. познакомить студентов с наиболее распространёнными современными технологиями программирования;
3. углубить знания студентов о типичных для данной темы организационных и технических инструментах
4. научить студентов достижению высоких показателей оценки процесса разработки программного обеспечения;
5. научить студентов проектировать БД;
6. научить студентов коллективной разработке сетевых прикладных программ.

Место дисциплины в структуре ООП бакалавриата

Дисциплина «Технология программирования» относится к циклу профессиональных дисциплин ОП. Дисциплина направлена на формирование общекультурных и профессиональных компетенций выпускника.

Дисциплина «Технология программирования» логически и содержательно связана с такими курсами, как «Web-программирование», «Практикум на ЭВМ», «Базы данных».

Дисциплина направлена на формирование общекультурных и профессиональных компетенций выпускника.

Для изучения дисциплины студент должен:

Знать:

– основы алгоритмизации и программирования;

– базовые инструменты проектирования и структурирования программных продуктов.

Уметь:

– программировать нескольких алгоритмических языках;

– вести индивидуальную разработку программных систем небольшой сложности.

Владеть:

– методами алгоритмизации и программирования;

– навыками разработки, отладки и сопровождения небольших приложений;

– навыками коммуникации, как очной так и с помощью электронных средств связи.

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

Код и формулировка компетенции	Этапы формирования компетенции	
ПК-4 способностью документировать процессы создания информационных систем на стадиях жизненного цикла	Знает	подходы к формированию функциональных, технических и программных требований к разрабатываемому продукту, методику и нотации описания процессов проектирования и реализации информационных систем
	Умеет	проводить описание прикладных процессов и информационного обеспечения в соответствии с современными требованиями
	Владеет	навыками описания прикладных процессов и информационного обеспечения в соответствии с современными требованиями
ПК-9 способностью составлять техническую документацию проектов автоматизации и информатизации прикладных процессов	Знает	инструментальное обеспечение для разработки программных приложений, языки и средства программирования и сопровождения процесса программирования.
	Умеет	применять методы алгоритмизации, методологию коллебктивной реализации приложений
	Владеет	способностью программировать приложения и создавать программные прототипы решения прикладных задач
ПК-16 способностью осуществлять тестирование компонентов информационных систем по заданным сценариям	Знает	технологию и методику тестирования компонентов программного обеспечения
	Умеет	организовывать все современный технологии тестирование компонентов программного обеспечения
	Владеет	навыками разработки стратегии тестирование компонентов программного обеспечения
ОК-13 способностью работать в коллективе, толерантно воспринимая социальные, этнические, конфессиональные и культурные различия	Знает	принципы коллективной разработки программной системы высокого уровня сложности, меру ответственности и ценность инициативы, экономические и юридические аспекты профессиональной деятельности программиста
	Умеет	вести разработку в составе коллектива программистов с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности, осуществлять профессиональную деятельность программиста в соответствии с существующими нормами
	Владеет	Способн----------остью проявлять инициативу и принимать ответственные решения, осознавая ответственность за результаты своей профессиональной деятельности

Для формирования вышеуказанных компетенций в рамках дисциплины «Технология программирования» применяются следующие методы интерактивного обучения: лекция-беседа, метод автоматизированного обучения, метод коллективной разработки.

При выполнении различных видов работ используются следующие технологии:

1. *Проблемное обучение* – стимулирование обучающихся к самостоятельному приобретению знаний, необходимых для решения конкретной проблемы.

3. *Контекстное обучение* – мотивация студентов к усвоению знаний путём выявление связей между конкретным знанием и его применением.

4. *Обучение на основе опыта* – активизация познавательной деятельности студентов бакалавриата за счёт ассоциации и собственного опыта с предметом обучения.

I. СТРУКТУРА И СОДЕРЖАНИЕ ТЕОРЕТИЧЕСКОЙ ЧАСТИ КУРСА

Тема 1. Сложные системы (6 час.)

• Требования к отчётности. Обзор курса.

• Понятие сложности и сложной системы.

• Математические и физические основы теории сложных систем.

• Теория хаоса. Непредсказуемость сложных систем.

• Борьба со сложностью как фундаментальная проблема программирования.

Тема 2. Технологии (6 час.)

• Понятие и назначение технологии.

• Примеры технологий.

• Технологии программирования как частный случай технологий.

• Классификация и исторический обзор технологий программирования.

Тема 3. Коллективная разработка (6 час.)

• Классификация коллективной деятельности, особенности творческих коллективов.

• Сублинейный рост производительности труда как основная проблема крупных творческих коллективов.

• Методы и трудности объективной оценки и мотивации творческой деятельности.

• Конкретные критерии оценки производительности программистов и их недостатки, на примере технологии CMM.

• Открытая разработка как пример эффективной организации больших коллективов программистов.

Тема 4. Жизненный цикл программного продукта. (6 час.)

• Понятие жизненного цикла, основные этапы.

• Анализ предметной области.

• Проектирование программного продукта.

• Разработка.

• Тестирование.

• Сопровождение.

Тема 5. Технологии управления жизненным циклом (6 час.)

• Технология waterfall, кризис её применения.

• Повторяемость жизненного цикла.

• Инкрементная и итеративная разработка.

• Технологии короткого цикла, Agile, SCRUM.

Тема 6. Интеллектуальная собственность. (2 час.)

• Виды интеллектуальной собственности.

• Авторское право, лицензии.

• Закрытые и открытые лицензии.

• Свободное программное обеспечение.

• Этические и философские аспекты разработки и распространения программного обеспечения.

Тема

7. Контроль версий. (4 час.)

• Математический сопроцессор;

II. СТРУКТУРА И СОДЕРЖАНИЕ ПРАКТИЧЕСКОЙ ЧАСТИ КУРСА

Практические работы организованы в виде практических работ по выполнению этапов полного жизненного цикла сложной сетевой системы в рамках единого для каждой команды семестрового задания. Цель – практическая реализация изложенных на лекциях принципов коллективной разработки.

Практические работы (108 час.)

Задание 1. Формирование творческих коллективов, выбор руководителей. Распределение заданий между коллективами. Взаимодействие с заказчиком. Описание и анализ предметной области. (14/0 час.)

Задание 2. Формализация задания. Формирование требований (функциональных, программных, технических, к пользователю, к надежности и пр.) (10/0 час.)

Задание 3. Распределение ролей этапа проектирования в коллективе, выбор средств проектирования программных систем. Принятие основных проектных решений, (14/0 час.)

Задание 4. Выполнение проектных работ коллективом разработчиков. Ведение общей документации, управление проектированием со стороны руководителя коллектива (20/0 час.)

Задание 5. Коллективная реализация проекта системы. Мобильное распределение ролей этапа реализации в коллективе с использованием системы контроля версий. Разработка протокола взаимодействия компонент системы. Контроль реализацией со стороны руководителя коллектива (24/0 час.)

Задание 6. Разработка технологии тестирования реализованной системы. Авторское тестирование, отладка системы (10/0 час.)

Задание 7. Документирование и защита проекта (10/0 час.)

Задание 8. Сдача системы с документацией заказчику. Внедрение и сопровождение системы. (6/0 час.)

III. УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ.

Учебно-методическое обеспечение самостоятельной работы обучающихся по дисциплине «Информатика и программирование» представлено в Приложении 1 и включает в себя:

план-график выполнения самостоятельной работы по дисциплине, в том числе примерные нормы времени на выполнение по каждому заданию;

характеристика заданий для самостоятельной работы обучающихся и методические рекомендации по их выполнению;

требования к представлению и оформлению результатов самостоятельной работы;

критерии оценки выполнения самостоятельной работы.

ХАРАКТЕРИСТИКА ЗАДАНИЙ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Содержание самостоятельной работы студентов по дисциплине

1. Изучение необходимых для реализации системы технических аспектов и программных средств, выходящий за рамки предыдущих курсов обучения.
2. Разработка алгоритмов и программ при выполнении лабораторных заданий.
3. Подготовка системы к тестированию.
4. Подготовка студента к экзамену.

Текущая СРС.

- работа с лекционным материалом, поиск и обзор литературы и электронных источников информации по программным и техническим средствам реализации программных продуктов;
- изучение тем, вынесенных на самостоятельную проработку;
- подготовка к лабораторным работам;
- подготовка к экзамену.

Творческая проблемно-ориентированная самостоятельная работа (ТСР).

ТСР направлена на развитие интеллектуальных умений, комплекса универсальных (общекультурных) и профессиональных компетенций, повышение творческого потенциала бакалавров и заключается в:

– поиске, анализе, структурировании и документации информации;

– разработке сложного программного продукта;

– исследовательской работе и участии в творческих студенческих коллективах;

– анализе научных публикаций по заранее определенной руководителем коллектива теме.

ТРЕБОВАНИЯ К ПРЕДСТАВЛЕНИЮ И ОФОРМЛЕНИЮ РЕЗУЛЬТАТОВ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Устный ответ по вопросу экзамена.
2. Отчет по коллективному проекту системы, оформленный согласно установленным стандартам.
3. Исходный код разработанной системы.

КРИТЕРИИ ОЦЕНКИ ВЫПОЛНЕНИЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Оценка результатов самостоятельной работы организуется как единство двух форм: самоконтроль и контроль со стороны преподавателя.

Критерии оценки:

1. Анализ предметной области и обзор существующих решений (10 баллов)
2. Коллективный выбор и обоснование проектных решений (10 баллов)
3. Разработка коллективного проекта системы (10 баллов)
4. Процесс и качество реализации проекта (10 баллов)
5. Качество документации реализованной системы (10 баллов)

6. Надежность системы и соответствие требованиям заказчика (10 баллов)

IV. КОНТРОЛЬ ДОСТИЖЕНИЯ ЦЕЛЕЙ КУРСА

Изучение дисциплины «Технология программирования» предусматривает:

- предоставление теоретического материала в соответствии с программой, с указанием материала по списку электронных источников для каждой темы;
- взаимодействие с членами творческого коллектива, выполнение индивидуальных заданий руководителя коллектива;
- обязательное согласование проектных, технических и прочих решений с руководителем коллектива.

Текущий контроль. Предусматривает учет руководителем коллектива своевременности и качества выполненных студентами

Итоговый контроль. Предусматривает рейтинговую оценку по учебной дисциплине в течение семестра (распределение баллов осуществлется руководителем коллектива) и экзамен.

№ п/п	Контролируемые разделы / темы дисциплины	Коды и этапы формирования компетенций		Оценочные средства	
				текущий контроль	промежуточная аттестация
1	Сложные программные системы	ОК-3 ПК-7 ПК-8 ПК-13 ПК-27	знает понятие сложности и особенности сложной системы, математические и физические основы теории сложных систем, требования к отчётности и документированию программных систем, непредсказуемость, теорию хаоса. сложных систем, фундаментальность проблемы борьбы со сложностью	Устный опрос	1 - 4

			умеет определять уровень сложности программной системы, использовать современные методы борьбы со сложностью системы	Лабораторная работа (ПР-6)	Отчет по лабораторной работе
			владеет навыками коллективной разработки сложных программных систем	Лабораторная работа (ПР-6)	Отчет по лабораторной работе
2	Технологии коллективной разработки сложных программных систем	ОК-3 ПК-7 ПК-8 ПК-13 ПК-27	Знает понятие и назначение технологии, в том числе технологии программирования, принципы классификации и исторические предпосылки современных технологий программирования	коллоквиум (УО-2).	*5 - 11*
			Умеет выполнять классификацию и представить исторический обзор технологий программирования, привести примеры современных технологий	Лабораторная работа (ПР-6)	Отчет по лабораторной работе
			владеет языком и средствами формализации описания процесса разработки и модификации сложной программной системы	Лабораторная работа (ПР-6)	Отчет по лабораторной работе
3	Технологии управления жизненным циклом программного продукта	ОК-3 ПК-7 ПК-8 ПК-13 ПК-27	•знает понятие и структуру жизненного цикла, основные его этапы, понимает свойство повторяемости жизненного цикла, инкрементный и итеративный характер жизненного цикла	коллоквиум (УО-2).	12 – 14
			умеет, осуществлять анализ предметной области, проектирование,	Лабораторная работа (ПР-6)	Отчет по лабораторной работе

			разработку, тестирование, сопровождение программного продукта		
			владеет навыками использования технология waterfall, информацией о кризисе её применения, навыками технологии короткого цикла SCRUM	Лабораторная работа (ПР-6)	Отчет по лабораторной работе
4	Аспекты создания интеллектуальной собственности	ОК-3 ПК-7 ПК-8 ПК-13 ПК-27	Знает виды интеллектуальной собственности, юридические аспекты установления авторского права, получения лицензии, этические и философские аспекты разработки и распространения программных продуктов	коллоквиум (УО-2).	15 – 17
			умеет применять и распространять свободное программное обеспечение	Лабораторная работа (ПР-6)	Отчет по лабораторной работе
			владеет навыками оформления открытых (закрытых) лицензий, соблюдения авторских прав на программные продукты,	Лабораторная работа (ПР-6)	Отчет по лабораторной работе

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

V. СПИСОК УЧЕБНОЙ ЛИТЕРАТУРЫ И ИНФОРМАЦИОННО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Основная литература
(электронные и печатные издания)

1. Технология программирования [Электронный ресурс]: учебное пособие/ Ю.Ю. Громов [и др.].— Электрон. текстовые данные.— Тамбов: Тамбовский государственный технический университет, ЭБС АСВ, 2013.— 173 с.— Режим доступа: http://www.iprbookshop.ru/63910.html.— ЭБС «IPRbooks»
2. Технология программирования: учебник / Г.С. Иванова. — Москва : КноРус, 2011. — 333 с. — ISBN 978-5-406-00519-4.
3. Котляров В.П. Основы тестирования программного обеспечения [Электронный ресурс]/ Котляров В.П.— Электрон. текстовые данные.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.— 334 с.— Режим доступа: http://www.iprbookshop.ru/62820.html.— ЭБС «IPRbooks»
4. Сергеев С.Ф. Методы тестирования и оптимизации интерфейсов информационных систем [Электронный ресурс]: учебное пособие/ Сергеев С.Ф.— Электрон. текстовые данные.— СПб.: Университет ИТМО, 2013.— 117 с.— Режим доступа: http://www.iprbookshop.ru/68664.html.— ЭБС «IPRbooks»
5. **Информационные системы**: Учебное пособие / О.Л. Голицына, Н.В. Максимов, И.И. Попов. - 2-е изд. - М.: Форум: НИЦ ИНФРА-М, 2014. - 448 с.: ил.; 60x90 1/16. - (Высшее образование). (переплет) ISBN 978-5-91134-833-5 - Режим доступа: http://znanium.com/catalog/product/435900
6. Битюцкая Н.И. Разработка программных приложений [Электронный ресурс]: лабораторный практикум/ Битюцкая Н.И.— Электрон. текстовые данные.— Ставрополь: Северо-Кавказский федеральный университет, 2015.— 140 с.— Режим доступа: http://www.iprbookshop.ru/63128.html.— ЭБС «IPRbooks»
7. Грэхем, Л. Разработка через тестирование для iOS [Электронный ресурс] / Л. Грэхем ; пер. с англ. Киселев А.Н.. — Электрон. дан. — Москва :

ДМК Пресс, 2013. — 272 с. — Режим доступа: https://e.lanbook.com/book/63183. — Загл. с экрана.]

Дополнительная литература
(печатные и электронные издания)

1. Долженко А.И. Технологии командной разработки программного обеспечения информационных систем [Электронный ресурс]/ Долженко А.И.— Электрон. текстовые данные.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.— 300 с.— Режим доступа: http://www.iprbookshop.ru/39569.html.— ЭБС «IPRbooks»
2. Введение в программные системы и их разработку [Электронный ресурс]/ С.В. Назаров [и др.].— Электрон. текстовые данные.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.— 649 с.— Режим доступа:

Перечень ресурсов информационно-телекоммуникационной сети «Интернет»

1. Литвиненко Н. А.Технология программирования на C++. Win32 API-приложения: Учебное пособие / Литвиненко Н.А. - СПб:БХВ-Петербург, 2010. - 280 с. ISBN 978-5-9775-0600-7 - Режим доступа: http://znanium.com/catalog/product/351463
2. Сайт AnandTech [http://anandtech.com].
3. Сайт Tom's Hardware [http://tomshardware.com].
4. Сайт Ars Technica CPU and Chipset Guide [http://arstechnica.com/cpu/index.html].

Перечень информационных технологий и программного обеспечения

1. Электронная рассылка
2. Электронный журнал успеваемости
3. Электронные поисковые приложения
4. Система контроля версий

VI. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

На изучение дисциплины отводится 144 часа аудиторных (лекционных и практических) занятий. На занятиях перед выдачей коллективных заданий преподаватель вводит основные требования к его выполнению, формирует творческие коллективы, назначает руководителей коллективов, озвучивает полномочия, права и обязанности руководителей и рядовых членов коллективов. Преподаватель объясняет теоретический материал об очередном аспекте разработки программной системы. Приводит примеры, поддерживает непрерывный контакт с аудиторией, отвечает на возникающие у студентов вопросы. На практических занятиях преподаватель контролирует деятельность коллектива через еего руководителя, оказывает текущую организационную и профессиональную поддержку, уточняет особенности выполнения очередного этапа разработки системы.

По всем аспектам студентам предлагается проявлять самостоятельность в рамках коллективной работы. Преподаватель контролирует работу студентов, в части случаев отвечает на возникающие вопросы, в других случаях рекомендует выполнить самостоятельный обзор возможных источников информации.

После выполнения задания, коллектив оформляет отчет и документацию в соответствии с указанным в методических материалах требованиями, предоставляет на проверку преподавателю код программного продукта на компьютере во время занятия. Руководитель отвечает на вопросы преподавателя, дает необходимые пояснения по системе.

По данному курсу разработаны методические материалы. Для успешного достижения учебных целей занятий должны выполняться следующие основные требования:

-соответствие действий обучающихся ранее изученным на лекционных и семинарских занятиях методикам и технологиям.

-максимальное приближение действий студентов к реальным, соответствующим будущим функциональным обязанностям.

- формирование умений и навыков по каждому этапу разработки системы, т.е. каждый студент должен выполнять практические задания из всех этапов.

-использование при работе фактических документов, технологических карт, бланков и т.п.

-выработка соответствующих индивидуальных и коллективных умений и навыков.

Студент должен:

-научиться самостоятельно предлагать обоснованные варианты решений, работать с бумажными и электронными источниками, , пользоваться справочной и научной литературой.

-формировать умение учиться самостоятельно, т.е. овладевать методами, способами и приемами самообучения, саморазвития и самоконтроля.

Рекомендации по подготовке к экзамену:

Рекомендуется еще раз самостоятельно ответить на теоретические вопросы для самопроверки, приведенные в учебных материалах, уже после приобретения практических умений и навыков в ходе разработки программной системы.

При ответе на каждый вопрос экзамена студент должен продемонстрировать знание определения указанного понятия, связанных с ним особенностей реализации и применения, умение реализовать указанную операцию, а также навыки иллюстрации теоретических принципов на предложенных простых примерах.

VII. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Лекционная аудитория: мультимедийный проектор Optima EX542I – 1 шт.; аудио усилитель QVC RMX 850 – 1 шт.; колонки – 1 шт.; ноутбук; ИБП – 1 шт.; настенный экран; микрофон – 1 шт.

Компьютерные классы ДВФУ (кампус на о. Русском, Аякс 10, корпус D, ауд. 733, 733а) по 15 персональных компьютеров Extreme DOU E 8500/500 GB/ DVD+RW.

Системное и прикладное обеспечение ПЭВМ.

Приложение 1

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

ШКОЛА ЕСТЕСТВЕННЫХ НАУК

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ

по дисциплине «Технология программирования»

Направление подготовки— 09.03.03 Прикладная информатика

Владивосток
2017

План-график выполнения самостоятельной работы по дисциплине

№ п/п, название	Дата/сроки выполнения	Вид СРС	Примерные нормы времени на выполнение	Форма контроля
1. Сложные программные системы. Аспекты технологии их коллективной разработки	Четвертая неделя семестра	ИДЗ	2 недели	Коллоквиум
2. Технологии управления жизненным циклом программного продукта. Аспекты создания интеллектуальной собственности	Пятая неделя семестра	ИДЗ	1 неделя	Коллоквиум
3. Формирование требований к программной системе	Шестая неделя семестра	ИДЗ	2 недели	Документальный отчет
4. Разработка проекта программной системы	Восьмая неделя семестра	ИДЗ	3 неделя	Документальный отчет
5. Реализация, тестирование и отладка проекта программной системы	Одиннадцатая неделя семестра	ИДЗ	4 недели	Программный код
6. Подготовка	Сессия	ИДЗ	1 неделя	Итоговое

программной системы к внедрению				тестирование кода

Критерии оценивания

В течение семестра студентам последовательно выдаются практические задания по семи темам, каждая из которых имеет вес от 10%. Своевременность выполнения заданий также учитывается и имеет вес 10%. Для получения оценки «отлично» необходимо иметь итоговый балл не ниже 80%, оценки «хорошо» – 60 %, оценки «удовлетворительно» – 50 %,

Глоссасарий дисциплины «Технология программирования»

- Технология программирования - дисциплина, изучающая технологические процессы программирования и порядок их прохождения.
- Аспектно-ориентированное сборочное программирование - разновидность сборочного программирования, основанная на сборке полнофункциональных приложений из многоаспектных компонентов, инкапсулирующих различные варианты реализации.
- Восходящее программирование
- Программирование "снизу вверх"
- Восходящее программирование - методика разработки программ, при которой крупные блоки собираются из ранее созданных мелких блоков.
- Восходящее программирование начинается с разработки ключевых процедур и подпрограмм, которые затем постоянно модифицируются.
- Диаграмма функционального моделирования
- Structured analysis and design technique (SADT)
- Диаграмма функционального моделирования - инструмент разработки функциональных спецификаций в виде диаграмм, фрагментов текста и глоссария, связанных перекрестными ссылками. В состав диаграммы входят:

- - блоки, изображающие активность моделируемой системы; и
- - дуги, связывающие блоки вместе и изображающие взаимодействия и взаимосвязи между ними.
- Место соединения дуги с блоком определяет тип интерфейса:
- - управляющая информация входит в блок сверху;
- - входная информация, подвергающаяся обработке, показывается с левой стороны блока;
- - выходная информация показывается с правой стороны;
- - механизм, осуществляющий операцию, представляется дугой, входящей в блок снизу.
- Заглушка - структурном программировании - процедура, представленная точной спецификацией заголовка и пустым телом. Заглушка позволяет компилировать и выполнять программу в отладочном режиме.
- Императивное программирование - технология программирования, характеризующаяся принципом последовательного изменения состояния вычислителя пошаговым образом. При этом управление изменениями полностью определено и полностью контролируемо.
- Инструментарий технологии программирования
- Инструментарий технологии программирования - программные продукты, предназначенные для поддержки технологии программирования.
- Компонентное сборочное программирование
- Компонентное сборочное программирование - объектно-ориентированное сборочное программирование, основанное на распространении классов в бинарном виде и предоставление доступа к методам класса через строго определенные интерфейсы.
- Компонентное сборочное программирование поддерживают технологические подходы COM, CORBA, .Net.
- Компьютерный дарвинизм

- Компьютерный дарвинизм - подход к разработке программных систем, основанный на принципе восходящей разработки при интенсивном тестировании. Подход состоит из трех основных процессов: макетирования, тестирования и отладки.
- Логическое программирование
- Логическое программирование - программирование в терминах фактов и правил вывода, с использованием языка, основанного на формальных исчислениях.
- Метод восходящего проектирования
- Метод восходящего проектирования - подход, при котором в первую очередь определяются вспомогательные модули, которые потребуются для проектируемой программы.
- Метод расширения ядра
- Метод расширения ядра - метод восходящего программирования, при котором основное внимание уделяется выявлению множества вспомогательных модулей, а не определению функции всей программы в целом.
- Модульное программирование
- Модульное программирование - метод разработки программ, предполагающий разбиение программы на независимые модули. Считается, что:
- - оптимальный по размерам модуль целиком помещается на экране дисплея;
- - разделение большой программы на модули облегчает ее разработку, отладку и сопровождение.
- Модульное сборочное программирование
- Модульное сборочное программирование - разновидность сборочного программирования, основанная на процедурах и функциях методологии структурного императивного программирования.
- Нисходящее программирование

- Программирование "сверху вниз"
- Нисходящее программирование - методика разработки программ, при которой разработка начинается с определения целей решения проблемы, после чего идет последовательная детализация, заканчивающаяся детальной программой.
- Объектно-ориентированное программирование - технология программирования, при которой программа рассматривается как набор дискретных объектов, содержащих, в свою очередь, наборы структур данных и процедур, взаимодействующих с другими объектами.
- Объектно-ориентированное сборочное программирование
- Объектно-ориентированное сборочное программирование - разновидность сборочного программирования:
- - основанная на методологии объектно-ориентированного программирования; и
- - предполагающая распространение библиотек классов в виде исходного кода или упаковку классов в динамически компонуемую библиотеку.
- Сборочное программирование
- Сборочное программирование - технология программирования, при которой программа собирается посредством повторного использования уже известных фрагментов программ.
- Синтезирующее программирование - программирование, предполагающее синтез программы по ее спецификации.
- Структурное программирование - методология и технология разработки программных комплексов, основанная на принципах:
- - программирования "сверху-вниз";
- - модульного программирования.
- При этом логика алгоритма и программы должны использовать три основные структуры: последовательное выполнение, ветвление и повторение.
- Компонент - Составная часть распределенного приложения

- Инкапсуляция, encapsulation - механизм, который объединяет данные и код, манипулирующий зтими данными, а также защищает и то, и другое от внешнего вмешательства или неправильного использования. В объектно-ориентированном программировании код и данные могут быть объединены вместе; в этом случае говорят, что создаётся так называемый "чёрный ящик". Когда коды и данные объединяются таким способом, создаётся объект (object). Другими словами, объект - это то, что поддерживает инкапсуляцию.
- Внутри объекта коды и данные могут быть закрытыми (private). Закрытые коды или данные доступны только для других частей этого объекта. Таким образом, закрытые коды и данные недоступны для тех частей программы, которые существуют вне объекта. Если коды и данные являются открытыми, то, несмотря на то, что они заданы внутри объекта, они доступны и для других частей программы. Характерной является ситуация, когда открытая часть объекта используется для того, чтобы обеспечить контролируемый интерфейс закрытых элементов объекта.
- На самом деле объект является переменной определённого пользователем типа. Может показаться странным, что объект, который объединяет коды и данные, можно рассматривать как переменную. Однако применительно к объектно-ориентированному программированию это именно так. Каждый элемент данных такого типа является составной переменной.
- Полиморфизм, polymorphism - (от греческого polymorphos) - это свойство, которое позволяет одно и то же имя использовать для решения двух или более схожих, но технически разных задач. Целью полиморфизма, применительно к объектно-ориентированному программированию, является использование одного имени для задания общих для класса действий. Выполнение каждого конкретного действия будет определяться типом данных. Например для языка Си, в котором

полиморфизм поддерживается недостаточно, нахождение абсолютной величины числа требует трёх различных функций: abs(), labs() и fabs(). Эти функции подсчитывают и возвращают абсолютную величину целых, длинных целых и чисел с плавающей точкой соответственно. В C++ каждая из этих функций может быть названа abs(). Тип данных, который используется при вызове функции, определяет, какая конкретная версия функции действительно выполняется. В C++ можно использовать одно имя функции для множества различных действий. Это называется перегрузкой функций (function overloading).

- В более общем смысле, концепцией полиморфизма является идея "один интерфейс, множество методов". Это означает, что можно создать общий интерфейс для группы близких по смыслу действий. Преимуществом полиморфизма является то, что он помогает снижать сложность программ, разрешая использование того же интерфейса для задания единого класса действий. Выбор же конкретного действия, в зависимости от ситуации, возлагается на компилятор. Вам, как программисту, не нужно делать этот выбор самому. Нужно только помнить и использовать общий интерфейс. Пример из предыдущего абзаца показывает, как, имея три имени для функции определения абсолютной величины числа вместо одного, обычная задача становится более сложной, чем это действительно необходимо.
- Полиморфизм может применяться также и к операторам. Фактически во всех языках программирования ограниченно применяется полиморфизм, например, в арифметических операторах. Так, в Си, символ + используется для складывания целых, длинных целых, символьных переменных и чисел с плавающей точкой. В этом случае компилятор автоматически определяет, какой тип арифметики требуется. В C++ вы можете применить эту концепцию и к другим, заданным вами, типам данных. Такой тип полиморфизма называется перегрузкой операторов (operator overloading).

- Ключевым в понимании полиморфизма является то, что он позволяет вам манипулировать объектами различной степени сложности путём создания общего для них стандартного интерфейса для реализации похожих действий.
- Наследование, inheritance - это процесс, посредством которого один объект может приобретать свойства другого. Точнее, объект может наследовать основные свойства другого объекта и добавлять к ним черты, характерные только для него. Наследование является важным, поскольку оно позволяет поддерживать концепцию иерархии классов (hierarchical classification).
- Применение иерархии классов делает управляемыми большие потоки информации. Например, подумайте об описании жилого дома. Дом - это часть общего класса, называемого строением. С другой стороны, строение - это часть более общего класса - конструкции, который является частью ещё более общего класса объектов, который можно назвать созданием рук человека. В каждом случае порождённый класс наследует все, связанные с родителем, качества и добавляет к ним свои собственные определяющие характеристики. Без использования иерархии классов, для каждого объекта пришлось бы задать все характеристики, которые бы исчерпывающи его определяли. Однако при использовании наследования можно описать объект путём определения того общего класса (или классов), к которому он относится, с теми специальными чертами, которые делают объект уникальным. Наследование играет очень важную роль в ООП.
- Контейнер в программировании - структура, позволяющая инкапсулировать в себя объекты разных типов.
- Среди "широких масс" программистов наиболее известны контейнеры, построенные на основе шаблонов, однако существуют и реализации в виде библиотек (наиболее широко известна библиотека GLib). Кроме того, применяются и узкоспециализированные решения. Примерами

контейнеров являются контейнеры из стандартной библиотеки (STL) - map, vector и др. В контейнерах часто встречается реализация алгоритмов для них. В ряде языков программирования (особенно скриптовых типа Perl или PHP) контейнеры и работа с ними встроена в язык.

- Контейнер, в отличие от коллекции, в общем случае, обычно не допускает явного задания числа элементов и обычно не поддерживает ветвистой структуры. Впрочем, это сильно зависит от реализации, поскольку многие реализации (особенно ориентированные на persistant storage) позволяют задавать размеры при создании контейнера.
- Распределенные вычисления - Парадигма организации приложений, в которой различные части программы могут исполняться на разных компьютерах в сети.
- Active Directory - Сетевая служба каталогов Microsoft, которую корпорация включила в состав Windows с версии 2000.
- ActiveX - Предлагаемый Microsoft способ разработки программных компонентов; название группы технологий, разработанных Microsoft для программирования компонентных объектных приложений на основе модели COM.
- ActiveX control - управляющий элемент ActiveX; введенное в 1996 г. Microsoft новое название независимых программируемых компонентов, ранее называемых OLE controls, OCXs, OLE custom controls; в отличие от последних позволяют работать с Internet.
- CDN, Content Delivery Network, Content Distribution Network, Сеть доставки и дистрибуции контента - географически распределённая сетевая инфраструктура, позволяющая оптимизировать доставку и дистрибуцию контента конечным пользователям в сети Интернет. Использование контент-провайдерами CDN способствует увеличению скорости загрузки интернет-пользователями аудио-, видео-,

программного, игрового и других видов цифрового контента в точках присутствия сети CDN.

- Простыми словами, CDN-это промежуточный хостинг. Контент вашего сайта сначала загружается на CDN хостинг, а затем отдается пользователю. Часто используется в криминальных целях воровства контента, информационного подавления неугодных интернет-ресурсов, перехвата пользовательского трафика и позиций в поисковых системах.
- Популярные бесплатные CDN-сервисы: CloudFlare, Incapsula.
- COM, Component Object Model - Программная архитектура Microsoft, поддерживающая компонентный подход к разработке приложений; модель компонентных объектов Microsoft; стандартный механизм, включающий интерфейсы, с помощью которых одни объекты предоставляют свои сервисы другим; является основой многих объектных технологий, в том числе OLE и ActiveX).
- COM+ - Модернизация COM и Microsoft Transaction Server, который упрощает разработку сложных распределенных приложений.
- Common Object Request Broker Architecture, CORBA - Основной конкурент DCOM в области построения распределенных программных систем.
- DLL, Dynamic Link Library - динамически подключаемая библиотека, понятие операционной системы Microsoft Windows; динамическая библиотека, позволяющая многократное применение различными программными приложениями. K DLL иногда причисляют также элементы управления ActiveX и драйвера. В мире UNIX аналогичные функции выполняют т. н. shared objects (<разделяемые объекты>). Формат файлов *.dll придерживается тех же соглашений, что и формат исполняемых файлов *.exe, сочетая код, таблицы и ресурсы.
- Microsoft Transaction Server, MTS - Усовершенствование в составе COM, которое реализует поддержку транзакций баз данных.

- OLE, Object Linking and Embedding - общее название (до 1996 г.) группы объектно-ориентированных технологий Microsoft на основе COM (OLE 1, OLE 2, OLE automation, OLE Database и др.).
- OCX, OLE Custom eXtension - перемещаемые элементы управления, OLE custom control, OLE control. Упрощенно можно сказать, что файлы *.ocx - это элементы управления ActiveX, выполняющие примерно те же функции, что и файлы *.dll.
- - OLE custom control
- специализированный управляющий элемент OLE, OLE control.
- OLE control - управляющие элементы OLE, программируемые компоненты-приложения с интерфейсом на базе OLE, позволяющим легко включать их в другие приложения; с 1996 г. называются ActiveX control. Синонимы: OCX, OLE custom control.
- Remote Procedure Call, RPC - Удаленный вызов процедуры - сообщение, посылаемое по сети, которое позволяет программе, установленной на одном компьютере, инициировать выполнение необходимой операции на другом.

Приложение 2

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

ШКОЛА ЕСТЕСТВЕННЫХ НАУК

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

по дисциплине «Технология программирования»
Направление подготовки— 09.03.03 Прикладная информатика

Владивосток
2017

МАТЕРИАЛЫ ЭКЗАМЕНА

Вопросы к экзамену по дисциплине «Технология программирования»

1. Сложные системы. Понятие сложности. Источники сложности.
2. Свойства работоспособных сложных систем.
3. Методы борьбы со сложностью.
4. Понятие технологии. Особенности технологии программирования.
5. Жизненный цикл программного обеспечения.
6. Методологии управления разработкой программного обеспечения: обзор и история.
7. Методологии CMM/CMMI, RUP, ГОСТ.
8. Методологии Agile/Scrum, TDD, XP.
9. Управление коллективом разработчиков.
10. Требования к программной системе, управление требованиями.
11. Методы и инструменты проектирования программных систем.
12. Визуальное проектирование, язык UML: назначение и общие принципы.
13. Статические диаграммы UML.
14. Динамические диаграммы UML.
15. Риски, оценка и управление рисками.
16. Системы контроля версий: назначение, классификация, принципы работы.
17. Распределённые системы контроля версий на примере Git.
18. Управление изменениями и релизами.
19. Системы управления инцидентами: назначение, принципы работы.
20. Структура и жизненный цикл инцидента.
21. Управление качеством кода. Технический долг.
22. Формальные методики обеспечения корректности.
23. Неформальные методики обеспечения корректности: тестирование и отладка.
24. Документирование программных систем.

Вопросы к коллоквиуму по дисциплине «Технология программирования»

1. Предмет и объекты изучения дисциплины Технология программирования.
2. Пояснить понятие Аспектно-ориентированное сборочное программирование

3. Пояснить принципы технологии Восходящего программирования. Описать метод восходящего проектирования
4. Пояснить идею технологии Программирование "снизу вверх"
5. Диаграмма функционального моделирования – назначение и правила построения
6. Назначение и преимущества Заглушек
7. Пояснить принципы технологии Императивное программирование
8. Инструментарий технологии программирования
9. Инструментарий технологии программирования –назначение, примеры
10. Сборочное программирование – назначение технологии.
11. Общая концепция и область применения Компонентного сборочного программирования
12. Компьютерный дарвинизм
13. Описать подход –Компьютерный дарвинизм
14. Основные характеристики Логического программирования
15. Описать метод расширения ядра
16. Модульное программирование – основные концепции, преимущества
17. Модульное сборочное программирование – отличительные свойства.
18. Методика Нисходящего программирования – преимущества и недостатки
19. Технология Программирование "сверху вниз"
20. Методика Нисходящего программирования
21. Объектно-ориентированный подход к программированию
22. Объектно-ориентированное сборочное программирование – суть технологии
23. Идея Синтезирующего программирования
24. Основы Структурного программирования
25. Пояснить понятия: Компонент, Инкапсуляция, Полиморфизм, Наследование, Контейнер
26. Сфера применения технологии называемой "чёрный ящик".
27. Парадигма Распределенные вычисления