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

ШКОЛА ЕСТЕСТВЕННЫХ НАУК

«СОГЛАСОВАНО»
Руководитель образовательной программы

_____ А.С. Величко

«30» июня 2016 г.

«УТВЕРЖДАЮ»
Врио заведующего кафедрой
математических методов в экономике

_____ А.С. Величко

«30» июня 2016 г.

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ
Программирование для ЭВМ
Направление подготовки 01.03.04 Прикладная математика

Форма подготовки очная

курс 1 семестр 1, 2
лекции 0 час.
практические занятия 0 час.
лабораторные работы 72 час.
в том числе с использованием МАО лек. 0 час. / пр. 0 час. / лаб. 72 час.
всего часов аудиторной нагрузки 72 час.
в том числе с использованием МАО 72 час.
самостоятельная работа 72 час.
в том числе на подготовку к экзамену 27 час.
контрольные работы (количество) 6
курсовая работа / курсовой проект не предусмотрены
зачет 2 семестр
экзамен 1 семестр

Рабочая программа учебной дисциплины (РПУД) составлена в соответствии с требованиями образовательного стандарта по направлению 01.03.04 «Прикладная математика», самостоятельно устанавливаемого ДВФУ, утвержденного приказом ректора от 18.02.2016 № 12-13-235

Рабочая программа обсуждена на заседании кафедры математических методов в экономике, протокол № 12 от «30» июня 2016 г.

Врио заведующего кафедрой математических методов в экономике, к.ф.-м.н., доцент А.С. Величко

Составитель:
старший преподаватель кафедры математических методов в экономике Е.А. Воронцова

АННОТАЦИЯ

Дисциплина «Программирование для ЭВМ» предназначена для студентов направления подготовки 01.03.04 «Прикладная математика».

Общая трудоемкость освоения дисциплины составляет 4 зачетные единицы (144 часа). Дисциплина реализуется на 1 курсе в 1-м и во 2-м семестре. Дисциплина входит в обязательные дисциплины вариативной части блока «Дисциплины (модули)».

Особенности построения курса: лабораторные работы (72 часа), самостоятельная работа (45 часов), подготовка к экзамену (27 часов).

Основная тематика курса определяется потребностями в базовых теоретических и прикладных знаниях студентов в предметной области. Содержание лабораторных работ призвано сформировать у студентов полноценное и единое представление о предмете.

Цели освоения дисциплины: научиться проектировать, разрабатывать, отлаживать и тестировать программное обеспечение для ЭВМ, изучить классические алгоритмы и разобрать их реализацию. Умение разрабатывать компьютерные приложения, полученное в результате освоения данного курса, будет необходимо при выполнении различных заданий и курсовых работ по дисциплинам, изучаемым на последующих курсах.

Курс «Программирование для ЭВМ» охватывает следующие разделы: элементы языка программирования C++, методы построения и анализа алгоритмов.

Задачи:

- развитие способности осваивать современные технологии программирования;
- развитие способности отлаживать и тестировать прикладное программное обеспечение;
- развитие способность настраивать, тестировать и осуществлять проверку вычислительной техники и программных средств;

- развитие готовности настраивать, тестировать и осуществлять проверку вычислительной техники и программных средств; .
- развитие способности демонстрировать знания современных языков программирования;
- развитие готовности демонстрировать знания современных языков программирования.

Для успешного изучения дисциплины «Программирование для ЭВМ» у обучающихся должны быть сформированы следующие предварительные компетенции:

- способность выявить сущность проблем, возникающих в ходе профессиональной деятельности, готовность использовать для их решения соответствующий аппарат;
- способность использовать для работы операционную систему ЭВМ Microsoft Windeows, информационно-телекоммуникационную сеть "Интернет".

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

Код и формулировка компетенции	**Этапы формирования компетенции**	
ОПК-2 - способность использовать современные математические методы и современные прикладные программные средства и осваивать современные технологии программирования	Знает	современные технологии программирования на языке С++
	Умеет	разрабатывать программные продукты с помощью современных технологий программирования
	Владеет	эффективными методами разработки программных продуктов с помощью современных технологий программирования

ПК-1 - способность использовать стандартные пакеты прикладных программ для решения практических задач на электронных вычислительных машинах, отлаживать, тестировать прикладное программное обеспечение	Знает	технологии отладки и тестирования прикладного программного обеспечения
	Умеет	отлаживать и тестировать прикладное программное обеспечение
	Владеет	методами отладки и тестирования прикладного программного обеспечения
ПК-2 - способность и готовность настраивать, тестировать и осуществлять проверку вычислительной техники и программных средств	Знает	способы тестирования программных средств, в том числе, разработанных самостоятельно
	Умеет	находить и исправлять ошибки в тестируемых программных продуктах
	Владеет	навыками настройки вычислительной техники для работы в интегрированной среде разработки, навыками тестирования и проверки программных продуктов
ПК-3 – способность и готовность демонстрировать знания современных языков программирования, операционных систем, офисных приложений, информационно-телекоммуникационной сети "Интернет", способов и механизмов управления данными, принципов организации, состава и схемы работы операционных систем	Знает	современные языки программирования (в частности, языки С и С++)
	Умеет	применять знания современных языков программирования для разработки программных продуктов
	Владеет	навыками разработки программ на современных языках программирования

Для формирования вышеуказанных компетенций в рамках дисциплины «Программирование для ЭВМ» применяются следующие методы активного/интерактивного обучения: работа в малых группах, занятие-дискуссия, обсуждение и разрешение проблем при создании программного обеспечения с использованием компьютерных технологий.

I. СТРУКТУРА И СОДЕРЖАНИЕ ТЕОРЕТИЧЕСКОЙ ЧАСТИ КУРСА

Поскольку лекции по дисциплине не предусмотрены, теоретическая

часть курса изучается студентами в рамках самостоятельной работы.

II. СТРУКТУРА И СОДЕРЖАНИЕ ПРАКТИЧЕСКОЙ ЧАСТИ КУРСА

Лабораторные работы (72 часа)

Модуль 1 (36 часов)

Лабораторная работа № 1. Работа в среде визуального программирования Google Blockly (2 часа)

Лабораторная работа № 2. Программирование на C++. Арифметические операции. Инкремент и декремент. Приоритеты. Условная операция (8 часов)

Лабораторная работа № 3. Написание программы вычисления сложного математического выражения (2 часа)

Лабораторная работа № 4. Программирование на C++. Условный оператор if…else и переключатель switch (6 часов)

Лабораторная работа № 5. Программирование на C++. Операторы цикла (6 часов)

Лабораторная работа № 6. Написание программы форматного вывода таблицы значений функции (4 часа)

Лабораторная работа № 7. Указатели и адреса (8 часов)

Модуль 2 (36 часов)

Лабораторная работа № 1. Программирование на C++. Многомерные массивы (12 часов)

Лабораторная работа № 2. Трассировка быстрой, шейкерной сортировки, сортировки Шелла и сортировки «Простой выбор» (2 часа)

Лабораторная работа № 3. Вычислительные эксперименты с различными видами сортировок массивов (4 часа)

Лабораторная работа № 4. Программирование на C++. Функции (8 часов)

Лабораторная работа № 5. Программирование на C++. Структуры (8 часов)

Лабораторная работа № 6. Работа с хеш-таблицами (2 часа)

III. УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ

Учебно-методическое обеспечение самостоятельной работы обучающихся по дисциплине «Программирование для ЭВМ» представлено в Приложении 1 и включает в себя:

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

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

требования к представлению и оформлению результатов самостоятельной работы;

критерии оценки выполнения самостоятельной работы.

IV. КОНТРОЛЬ ДОСТИЖЕНИЯ ЦЕЛЕЙ КУРСА

Контролируемые разделы дисциплины, этапы формирования компетенций, виды оценочных средств, зачетно-экзаменационные материалы, комплекты оценочных средств для текущей аттестации, описание показателей и критериев оценивания компетенций на различных этапах их формирования, 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. Дейл Н. Программирование на C++ [Электронный ресурс] : учебник / Дейл Н., Уимз Ч., Хедингтон М. — Электрон. дан. — М. : ДМК Пресс, 2007. — 672 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1_id=1219.

2. Тяпичев Г.А. Быстрое программирование на C++ [Электронный ресурс]. — Электрон. дан. — М.: СОЛОН-Пресс, 2008. — 373 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1_id=13688.

3. Подбельский В.В. Курс программирования на языке Си [Электронный ресурс]: учебник / Подбельский В.В., Фомин С.С. — Электрон. текстовые данные. — М.: ДМК Пресс, 2012. —384 с.— Режим доступа: http://e.lanbook.com/books/element.php?pl1_id=4148.

Дополнительная литература

(печатные и электронные издания)

1. Златопольский Д.М. Программирование: типовые задачи, алгоритмы, методы [Электронный ресурс]. — Электрон. дан. — М.: "Лаборатория знаний" (ранее "БИНОМ. Лаборатория знаний"), 2015. — 224 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1_id=70753.

2. Степанов В.П. Лабораторный практикум по программированию на языке Си [Электронный ресурс] : учебное пособие. — Электрон. дан. — М.: МГТУ им. Н.Э. Баумана, 2007. — 48 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1_id=52383.

3. Арипова О.В. Программирование на языке высокого уровня: лабораторный практикум для вузов [Электронный ресурс] / Арипова О.В., Гущин А.Н., Палехова О.А. — Электрон. дан. — СПб. : БГТУ "Военмех" им. Д.Ф. Устинова (Балтийский государственный технический университет «Военмех» имени Д.Ф. Устинова), 2014. — 96 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1_id=63671.

4. Грузина Э.Э. Практикум по программированию. – Ч. I [Электронный ресурс] : учебное пособие / Грузина Э.Э., Черноусова Н.Л.. — Электрон. дан. — Кемерово : Издательство КемГУ (Кемеровский государственный университет), 2013. — 100 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1_id=58312.

Перечень ресурсов информационно-телекоммуникационной сети «Интернет»

1. Язык программирования C++. Учебник [Электронный ресурс]. URL: http://cppstudio.com/cat/274/
2. Visual C++ Guided Tour. URL: http://msdn.microsoft.com/en-us/library/vstudio/ms235630.aspx
3. Онлайн-курс «Введение в программирование (C++)». Академия Яндекса, Высшая школа экономики (НИУ ВШЭ). https://stepic.org/course/Введение-в-программирование-(C++)-363/syllabus
4. Онлайн-курс «Основы алгоритмов» University of California, San Diego, Higher School of Economics. https://www.coursera.org/learn/algorithmic-toolbox

Перечень дополнительных информационно-методических материалов

1. Подбельский В.В. Практикум по программированию на языке Си. М.: Финансы и статистика, 2004.
2. Подбельский В.В. Язык Си++. 5-е изд. М.: Финансы и статистика, 2007.
3. Павловская Т.А., Щупак Ю.А. C/C++. Структурное программирование: Практикум. СПб.: Питер, 2003.
4. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы. Построение и

анализ. 3-е изд. М.: Вильямс, 2013.
5. Керниган Б., Пайк Р. Практика программирования. М.: Вильямс, 2004.
6. Прата С. Язык программирования С. Лекции и упражнения, 5-е издание. М.: Вильямс, 2013.
7. Вирт Н. Алгоритмы и структуры данных. М.: ДМК Пресс, 2010.

Перечень информационных технологий и программного обеспечения

При осуществлении образовательного процесса по дисциплине используется среда для разработки программ Microsoft Visual Studio одной из последних версий.

VI. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Рекомендации по планированию и организации времени, отведенного на изучение дисциплины, описание последовательности действий обучающихся

Освоение дисциплины следует начинать с изучения рабочей учебной программы, которая содержит основные требования к знаниям, умениям и навыкам. Обязательно следует учитывать рекомендации преподавателя, данные в ходе установочных занятий. Затем – приступать к изучению отдельных разделов и тем в порядке, предусмотренном программой.

Получив представление об основном содержании раздела, темы, необходимо изучить материал с помощью рекомендуемой основной литературы. Целесообразно составить краткий конспект или схему, отображающую смысл и связи основных понятий данного раздела и включенных в него тем. Обязательно следует записывать возникшие вопросы, на которые не удалось ответить самостоятельно.

Подготовку к началу обучения включает несколько необходимых пунктов:

1) Необходимо создать для себя рациональный и эмоционально достаточный уровень мотивации к последовательному и планомерному изучению дисциплины.

2) Необходимо изучить список рекомендованной основной и дополнительной литературы и убедиться в её наличии у себя дома или в библиотеке в бумажном или электронном виде.

3) Необходимо иметь «под рукой» специальные и универсальные словари, справочники и энциклопедии, для того, чтобы постоянно уточнять значения используемых терминов и понятий. Пользование словарями и справочниками необходимо сделать привычкой. Опыт показывает, что неудовлетворительное усвоение предмета зачастую коренится в неточном, смутном или неправильном понимании и употреблении понятийного аппарата учебной дисциплины.

4) Желательно в самом начале периода обучения возможно тщательнее спланировать время, отводимое на работу с источниками и литературой по дисциплине, представить этот план в наглядной форме (график работы с датами) и в дальнейшем его придерживаться, не допуская срывов графика индивидуальной работы и «аврала» в предсессионный период. Пренебрежение этим пунктом приводит к переутомлению и резкому снижению качества усвоения учебного материала.

Рекомендации по работе с литературой

1) Всю учебную литературу желательно изучать «под конспект». Чтение литературы, не сопровождаемое конспектированием, даже пусть самым кратким – бесполезная работа. Цель написания конспекта по дисциплине – сформировать навыки по поиску, отбору, анализу и формулированию учебного материала. Эти навыки обязательны для любого специалиста с высшим образованием независимо от выбранной специальности.

2) Написание конспекта должно быть творческим – нужно не переписывать текст из источников, но пытаться кратко излагать своими словами содержание ответа, при этом максимально структурируя конспект, используя символы и условные обозначения. Копирование и «заучивание» неосмысленного текста трудоемко и по большому счету не имеет большой познавательной и практической ценности.

3) При написании конспекта используется тетрадь, поля в которой обязательны. Страницы нумеруются, каждый новый вопрос начинается с нового листа, для каждого экзаменационного вопроса отводится 1-2 страницы конспекта. На полях размещается вся вспомогательная информация – ссылки, вопросы, условные обозначения и т.д.

4) В итоге данной работы «идеальным» является полный конспект по программе дисциплины, с выделенными определениями, узловыми пунктами, примерами, неясными моментами, проставленными на полях вопросами.

5) При работе над конспектом обязательно выявляются и отмечаются трудные для самостоятельного изучения вопросы, с которыми уместно обратиться к преподавателю при посещении установочных лекций и консультаций, либо в индивидуальном порядке.

6) При чтении учебной и научной литературы всегда следить за точным и полным пониманием значения терминов и содержания понятий, используемых в тексте. Всегда следует уточнять значения по словарям или энциклопедиям, при необходимости записывать.

7) При написании учебного конспекта обязательно указывать все прорабатываемые источники, автор, название, дата и место издания, с указанием использованных страниц.

Подготовка к промежуточной аттестации по дисциплине: экзамену (зачету)

К аттестации допускаются студенты, которые систематически в течение всего семестра посещали и работали на занятиях и показали уверенные знания в ходе выполнении практических заданий и лабораторных работ.

Непосредственная подготовка к аттестации осуществляется по вопросам, представленным в рабочей учебной программе. Тщательно изучите формулировку каждого вопроса, вникните в его суть, составьте план ответа. Обычно план включает в себя:

— определение сущности рассматриваемого вопроса, основных положений, утверждений, определение необходимости их доказательства;

— запись обозначений, формул, необходимых для полного раскрытия вопроса;

— графический материал (таблицы, рисунки, графики), необходимые для раскрытия сущности вопроса;

— роль и значение рассматриваемого материала для практической деятельности, примеры использования в практической деятельности.

VII. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Для осуществления образовательного процесса по дисциплине необходим компьютерный класс с персональными компьютерами с доступом в сеть «Интернет» и установленной средой разработки Microsoft Visual Studio. В компьютерном классе должно быть оборудование мультимедийного типа (мультимедийный проектор, настенный экран) и пластиковая доска.

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 «Программирование для ЭВМ»
Направление подготовки 01.03.04 Прикладная математика

Форма подготовки очная

Владивосток
2016

План-график выполнения самостоятельной работы по дисциплине

№ п/п	Дата/сроки выполнения	Вид самостоятельной работы	Примерные нормы времени на выполнение	Форма контроля
1	Модуль 1. 4 неделя	Повторение практического материала дисциплины, изученного в ходе аудиторных занятий; изучение основной и дополнительной литературы, указанной в рабочей учебной программе дисциплины,	3 часа	Работа над проектами
2	Модуль 1. 8 неделя	Самостоятельный разбор заданий и задач, решаемых на практических занятиях; самостоятельный повтор действий, осуществляемых в ходе выполнения лабораторных работ, в том числе при работе со специальным программным обеспечением	3 часа	Работа над проектами
3	Модуль 1. 12 неделя	Самостоятельный разбор заданий и задач, решаемых на практических занятиях; самостоятельный повтор действий, осуществляемых в ходе выполнения лабораторных работ, в том числе при работе со специальным программным обеспечением	3 часа	Работа над проектами
4	Модуль 1. 16 неделя	Подготовка к экзамену	27 часов	Экзамен
5	Модуль 2. 4 неделя	Самостоятельный разбор заданий и задач, решаемых на практических занятиях; самостоятельный повтор действий, осуществляемых в ходе выполнения лабораторных работ, в том числе при работе со	6 часов	Работа над проектами

		специальным программным обеспечением		
6	Модуль 2. 8 неделя	Самостоятельный разбор заданий и задач, решаемых на практических занятиях; самостоятельный повтор действий, осуществляемых в ходе выполнения лабораторных работ, в том числе при работе со специальным программным обеспечением	6 часов	
7	Модуль 2. 12 неделя	Самостоятельный разбор заданий и задач, решаемых на практических занятиях; самостоятельный повтор действий, осуществляемых в ходе выполнения лабораторных работ, в том числе при работе со специальным программным обеспечением	6 часов	Работа над проектами
8	Модуль 2. 16 неделя	Повторение практического материала дисциплины, изученного в ходе аудиторных занятий; изучение основной и дополнительной литературы, указанной в рабочей учебной программе дисциплины, самоконтроль ответов на основные вопросы по темам лабораторных работ	18 часов	Собеседование

Характеристика заданий для самостоятельной работы обучающихся и методические рекомендации по их выполнению

Задания и методические рекомендации по теме «Программирование на C++»

Помимо основных арифметических операций, в языке C++ определено некоторое количество специфических операций. К ним относятся операция инкремент (++) – увеличение значения на 1, операция декремент (—) – уменьшение значения на 1, операция sizeof – операция вычисления размера в байтах. Следует обратить внимание на операцию деления (/) – в зависимости от типов операндов она обозначает либо обычное деление, либо деление нацело.

По количеству операндов, которые необходимы для выполнения операции, все операции можно разделить на три типа: унарные, бинарные и тернарные. К унарным операциям относятся, например, инкремент, декремент и арифметическое отрицание. К бинарным – умножение, сложение, деление, вычитание, операции отношения, операции сдвига (<< и >>), остаток от деления (%), ряд логических и поразрядных операций. Единственная тернарная операция (т.е., требующая для своего выполнения трех операндов) в C++ – условная операция ? :, являющаяся в каком-то смысле аналогом условного оператора if...else.

1. Операции присваивания, инкремента и декремента

Составьте программу, которая:

а) выполняет арифметические операции над двумя целыми числами и выводит их результат (применить сокращенные арифметические операции, инкремент и декремент).

б) выполняет аналогичные действия с двумя вещественными числами двойной точности.

Вопросы

1. Опишите назначение операций типа знак =, ++ и --.
2. Введите:
   а) вместо целого числа – вещественное число;
   б) вместо вещественное числа – целое;
   Каковы будут результаты и почему?
3. Какую роль выполняет заголовочный файл iostream? Указать потоки ввода-вывода.

2. Свойства операций инкремента и декремента в постфиксной и префиксной формах

Пример программы.

```
#include <iostream>
using namespace std;
int main()
{
  int i, k = 10;
  i = 5 * (k++);
  cout << " i = " << i << endl << '\n';
  k--;
  i = 5 * ++k;
  cout << " i = " << i << "\n" << endl;
  return 0;
}
```

Вопросы

1. Опишите отличие и сходство '\n', "\n" и endl.
2. Изменится ли результат, если убрать скобки в (k++) или, наоборот, взять ++k в скобки?
3. Попробуйте вставить пробел в k--:

а) написать k-- в виде k --; б) написать k-- в виде k- -.

Что при этом произойдет?

4. Укажите различие приоритетов префиксной и постфиксной форм инкремента и декремента? Продемонстрируйте их в программе.

3. Приоритеты арифметических операций. Преобразование типа выражения из целого в вещественный

Для произвольных целых чисел i, j, x и y вычислите выражения:

-10 +++ j % i--

(-10 +++i +++j) / (3 * j + 3)

x / y + y / x +i * x + j * y

(i +++++x) / (--j---y)

Вопросы

1. Объясните порядок выполнения операций. Что делает операция % ?
2. Дополните программу операцией преобразования типа для вывода точного результата.
3. Вычислите вручную значения данных выражений и сравните с полученным в программе результатом.

4. Условная операция, операции отношения (сравнения) и логические бинарные операции

```
#include <iostream>
using namespace std;
void main ()
{
  int i = 5;
  int j = 10;
  cout << ((i < j) ? "TRUE" : "FALSE") << endl;
  cout << ((i > 0 || j < 100) ? "TRUE" : "FALSE") << endl;
```

```
  cout << ((i > 0 && j <= 10) ? "TRUE" : "FALSE") << endl;
  (i == 5 && i == j) ? cout << "TRUE"  << endl : cout << "FALSE" << endl;
  int k = (i == 5 || i != j) ? (i > 0 || j < 100) : ((i > 0 && j <= 10);
  cout << (k ? "TRUE" : "FALSE") << endl;
 }
```

Вопросы

1. Уточните текст и объясните смысл программы.
2. Укажите, где в этой программе условные операции, операции отношения и логические бинарные операции.

5. Анализ сортировок 1

Реализовать два метода сортировки – сортировку прямым выбором и шейкерную сортировку. Сделать сравнительный анализ двух реализованных методов по времени работы и по количеству итераций на нескольких наборах данных. Вывести результаты в виде таблицы.

6. Анализ сортировок 2

Реализовать два метода сортировки – пузырьковую сортировку и сортировку Шелла. Сделать сравнительный анализ двух реализованных методов по времени работы и по количеству итераций на нескольких наборах данных. Вывести результаты в виде таблицы.

7. Задача про бензин

Напишите программу, которая просит пользователя ввести количество миль, которые проехал автомобиль пользователя и количество галлонов израсходованного бензина. Затем эта программа должна рассчитать и отобразить на экране количество миль, пройденных на одном галлоне горючего, с одним знаком после десятичной точки. Используя тот факт, что один галлон приблизительно равен 3.785 литра и одна миля = 1.609 километра, ваша программа должна перевести значение в милях на галлон в литры на 100 километров и вывести результат с одним знаком после десятичной точки. (Обратите внимание, что американская схема измеряет затраты горючего, необходимого для преодоления заданного расстояния, тогда как европейская схема измеряет пройденный путь на единицу горючего.) Используйте символьные константы для этих двух параметров преобразования.

Требования к представлению и оформлению результатов самостоятельной работы

Самостоятельная работа включает в себя повторение теоретического и практического материала дисциплины, заслушиваемого и конспектируемого в ходе аудиторных занятий; изучение основной и дополнительной литературы, указанной в рабочей учебной программе дисциплины, самоконтроль ответов на основные проблемные вопросы по темам занятий; самостоятельный повтор действий, осуществляемых в ходе выполнения лабораторных работ, в том числе при работе со специальным программным обеспечением — Microsoft Visual Studio.

Результаты самостоятельной работы представляются и оформляются в виде ответов на основные положения теоретического и практического материала дисциплины по темам; работающего исходного кода результата решения практических заданий и задач; собственных действий, осуществляемых в ходе выполнения лабораторных работ.

Критерии оценки выполнения самостоятельной работы

Общие критерии оценки выполнения самостоятельной работы – правильность ответов на вопросы по темам теоретической части дисциплины, верность получаемых ответов в ходе решения практических заданий и задач, достижение правильного результата при осуществлении собственных действий по лабораторным работам.

Оценивание знаний в форме собеседования проводится по критериям:

- логичность изложения, знание и понимание основных теоретических и практических положений и концепций по теме;
- владение теоретическими и практическими приёмами написания, отладки и тестирования программ на языке программирования C++.

Оценивание знаний в форме работы над проектами проводится по критериям:

- завершенность и полнота выполненных заданий в рамках работы над проектами;
- владение методами и приемами решения конкретных задач и самостоятельность использования специализированного программного обеспечения;
- качество оформления письменного отчета в соответствии с правилами и стандартами оформления документации разработанных программ.

Приложение 2

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

ШКОЛА ЕСТЕСТВЕННЫХ НАУК

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

по дисциплине «Программирование для ЭВМ»
Направление подготовки 01.03.04 Прикладная математика

Форма подготовки очная

Владивосток
2016

Паспорт
фонда оценочных средств
по дисциплине «Программирование для ЭВМ»

Код и формулировка компетенции	Этапы формирования компетенции	
ОПК-2 - способность использовать современные математические методы и современные прикладные программные средства и осваивать современные технологии программирования	Знает	современные технологии программирования на языке С++
	Умеет	разрабатывать программные продукты с помощью современных технологий программирования
	Владеет	эффективными методами разработки программных продуктов с помощью современных технологий программирования
ПК-1 - способность использовать стандартные пакеты прикладных программ для решения практических задач на электронных вычислительных машинах, отлаживать, тестировать прикладное программное обеспечение	Знает	технологии отладки и тестирования прикладного программного обеспечения
	Умеет	отлаживать и тестировать прикладное программное обеспечение
	Владеет	методами отладки и тестирования прикладного программного обеспечения
ПК-2 - способность и готовность настраивать, тестировать и осуществлять проверку вычислительной техники и программных средств	Знает	способы тестирования программных средств, в том числе, разработанных самостоятельно
	Умеет	находить и исправлять ошибки в тестируемых программных продуктах
	Владеет	навыками настройки вычислительной техники для работы в интегрированной среде разработки, навыками тестирования и проверки программных продуктов
ПК-3 – способность и готовность демонстрировать знания современных языков программирования, операционных систем, офисных приложений, информационно-телекоммуникационной сети "Интернет", способов и механизмов управления данными, принципов организации, состава и схемы работы операционных систем	Знает	современные языки программирования (в частности, языки С и С++)
	Умеет	применять знания современных языков программирования для разработки программных продуктов
	Владеет	навыками разработки программ на современных языках программирования

№ п/п	Контролируемые разделы дисциплины	Коды и этапы формирования компетенций		Оценочные средства - наименование	
				текущий контроль	промежуточная аттестация
1	Программирование на языке С++	ОПК-2	Знает	Собеседование (УО-1)	Экзамен, вопросы 1-21, зачёт (вопросы 1-24)
			Умеет	Проект (ПР-9)	Проект (темы 1-20)
			Владеет	Проект (ПР-9)	Проект (темы 1-20)
		ПК-1	Знает	Собеседование (УО-1)	Зачёт, вопросы 20-21
			Умеет	Проект (ПР-9)	Проект (темы 1-20)
			Владеет	Проект (ПР-9)	Проект (темы 1-20)
		ПК-2	Знает	Собеседование (УО-1)	Зачёт, вопросы 20-21
			Умеет	Проект (ПР-9)	Проект (темы 1-20)
			Владеет	Проект (ПР-9)	Проект (темы 1-20)
		ПК-3	Знает	Собеседование (УО-1)	Экзамен, вопросы 1-21, зачёт (вопросы 1-19, 22-24)
			Умеет	Проект (ПР-9)	Проект (темы 1-20)
			Владеет	Проект (ПР-9)	Проект (темы 1-20)

Зачётно-экзаменационные материалы

Вопросы для подготовки к экзамену
по дисциплине «Программирование для ЭВМ»

1. Алфавит и лексемы языка C++. Идентификаторы и служебные слова.

2. Символьные и строковые константы. Разница между ними. ESC-последовательности.

3. Расширенная таблица ASCII. Пример программы.

4. Целочисленные типы данных. Особенности и примеры их применения. Представление в памяти.

5. Типы данных с плавающей точкой. Особенности и примеры их применения. Представление в памяти.

6. Оператор присваивания. Пример линейной программы.

7. Синтаксис и семантика операторов выражения, возврата, ветвления и выбора. Пример программы ветвления.

8. Синтаксис и семантика операторов цикла, прерывания и продолжения. Пример циклической программы.

9. Классификация типов данных. Основные и производные типы данных. Правила преобразования арифметических типов: по умолчанию и явное.

10. Арифметические операции и их приоритеты.

11. Логические операции. Синтаксис и семантика. Условная операция.

12. Поразрядные целочисленные операции.

13. Виды выражений. Приоритеты операций.

14. Символьный тип данных. Операции над данными символьного типа.

15. Одномерные массивы. Индексация выражения.

16. Многомерные массивы. Индексация выражения.

17. Массивы с элементами типа char и тип string. Разница между ними.

18. Указатели: понятие, синтаксис и семантика объявления. Операция разыменования. Типы указателей.

19. Адресная арифметика.

20. Взаимодействие между массивами и указателями. Массивы указателей. Примеры.

21. Динамические массивы. Операции выделения и освобождения памяти.

Комплекты оценочных средств для текущей аттестации

Вопросы для собеседования

по дисциплине «Программирование для ЭВМ»

I. Вопросы, обязательные для изучения. Элементы языка C/C++

1. Алфавит и лексемы языка C++. Идентификаторы и служебные слова.
2. Символьные и строковые константы. Разница между ними. ESC-последовательности.
3. Расширенная таблица ASCII. Пример программы.
4. Целочисленные типы данных. Особенности и примеры их применения. Представление в памяти.
5. Типы данных с плавающей точкой. Особенности и примеры их применения. Представление в памяти.
6. Оператор присваивания. Пример линейной программы.
7. Синтаксис и семантика операторов выражения, возврата, ветвления и выбора. Пример программы ветвления.
8. Синтаксис и семантика операторов цикла, прерывания и продолжения. Пример циклической программы.
9. Классификация типов данных. Основные и производные типы данных. Правила преобразования арифметических типов: по умолчанию и явное.
10. Арифметические операции и их приоритеты.
11. Логические операции. Синтаксис и семантика. Условная операция.
12. Поразрядные целочисленные операции.
13. Виды выражений. Приоритеты операций.
14. Символьный тип данных. Операции над данными символьного типа.
15. Одномерные массивы. Индексация выражения. Массив как параметр функций.
16. Многомерные массивы. Индексация выражения. Массив как параметр функций.
17. Массивы с элементами типа char и тип string. Разница между ними.
18. Указатели: понятие, синтаксис и семантика объявления. Операция разыменования. Типы указателей.
19. Адресная арифметика.
20. Взаимодействие между массивами и указателями. Массивы указателей. Примеры.
21. Динамические массивы. Операции выделения и освобождения памяти.
22. Тип данных «структура». Объявление, синтаксис и особенности использования.
23. Массивы структур. Пример.

24. Тип данных «ссылка». Объявление, синтаксис и особенности использования.
25. Функции. Объявление, определение, сигнатура, тело функции. Параметры функции. Примеры.
26. Функции. Способы и особенности передачи значений параметров. Примеры.
27. Указатель на функцию. Примеры использования.
28. Оператор typedef. Особенности применения.
29. Глобальные и локальные переменные. Область видимости.
30. Чем плох оператор goto? Мнение Н. Вирта.
31. История создания языка C++ и его связь с языком C. Почему язык так назван?

II. Блок вопросов по выбору студента. Модуль 2. Элементы технологии программирования
32. Формальная постановка задачи. Анализ предметной области (бизнес-моделирование).
33. Алгоритм. Блок-схема. Примеры.
34. Программа. Программное обеспечение (ПО). Классификация ПО.
35. Технология программирования. Технологическая операция.
36. Основные этапы развития технологии программирования.
37. Проблемы разработки сложных программных систем.
38. Блочно-иерархический подход к созданию сложных систем.
39. Декомпозиция предметной области. Методы декомпозиции.
40. Жизненный цикл и этапы разработки программного обеспечения.
41. Анализ требований. Спецификации. Проектирование.
42. Эволюция моделей жизненного цикла ПО. Каскадная модель и модель с промежуточным контролем.
43. Спиральная модель жизненного цикла ПО.
44. Тестирование. Цель тестирования. Зависимость вероятности правильного исправления
и его стоимости от этапа разработки.
45. Стратегии тестирования. Структурный подход.
46. Ручной контроль ПО.
47. Функциональное тестирование. Эквивалентное разбиение. Анализ граничных значений.

III. Вопросы, обязательные для изучения. Модуль 2. Алгоритмы сортировки данных

48. Сортировка «простой выбор». Примеры.
49. Пузырьковая сортировка. Примеры.
50. Шейкерная сортировка. Примеры.
51. Сортировка Шелла. Примеры.
52. Быстрая сортировка. Примеры.
53. Метод суммирующего множителя.
54. Полный анализ быстрой сортировки с помощью метода суммирующего множителя.

IV. Блок вопросов по выбору студента. Модуль 2. Динамические структуры данных
55. Список. Основные операции над элементами списка. Реализация на С. Примеры.
56. Очередь. Основные операции над элементами очереди. Реализация на С. Примеры.
57. Стек. Основные операции над элементами стека. Реализация на С. Примеры.
58. Хеширование. Хеш-функции. Коллизии. Метод деления. Метод умножения.
59. Хеш-функции для строк. Пример кода.
60. Хеш-таблицы. Метод цепочек.
61. Хеш-таблицы. Метод открытой адресации.
62. Деревья. Определение. Примеры. Число вершин и рёбер.
63. Двоичные, n-арные деревья. Реализация.
64. Обход двоичного дерева в глубину. Прямой, обратный, поперечный.
65. Обход двоичного дерева в ширину.
66. Двоичное дерево поиска. Операции: поиск по ключу, вставка, удаление узла, поиск минимального ключа.

V. Блок вопросов по выбору студента. Модуль 2. Методы построения, анализа алгоритмов, теория трансляции программ
67. Компилируемые и интерпретируемые языки программирования. Разница. Примеры.
68. Основные фазы компиляции. Лексический анализатор. Синтаксический анализатор.
69. Жадные алгоритмы.
70. Понятие рекурсии. Примеры использования.
71. О-символика для оценки быстродействия. Примеры.

Вопросы к зачёту

по дисциплине «Программирование для ЭВМ»

I. Вопросы, обязательные для изучения. Модуль 2. Алгоритмы сортировки данных

1. Сортировка «простой выбор». Примеры.
2. Пузырьковая сортировка. Примеры.
3. Шейкерная сортировка. Примеры.
4. Сортировка Шелла. Примеры.
5. Быстрая сортировка. Примеры.
6. Метод суммирующего множителя.
7. Полный анализ быстрой сортировки с помощью метода суммирующего множителя.

II. Блок вопросов по выбору студента. Модуль 2. Динамические структуры данных

8. Список. Основные операции над элементами списка. Реализация на С. Примеры.
9. Очередь. Основные операции над элементами очереди. Реализация на С. Примеры.
10. Стек. Основные операции над элементами стека. Реализация на С. Примеры.
11. Хеширование. Хеш-функции. Коллизии. Метод деления. Метод умножения.
12. Хеш-функции для строк. Пример кода.
13. Хеш-таблицы. Метод цепочек.
14. Хеш-таблицы. Метод открытой адресации.
15. Деревья. Определение. Примеры. Число вершин и рёбер.
16. Двоичные, n-арные деревья. Реализация.
17. Обход двоичного дерева в глубину. Прямой, обратный, поперечный.
18. Обход двоичного дерева в ширину.
19. Двоичное дерево поиска. Операции: поиск по ключу, вставка, удаление узла, поиск минимального ключа.

III. Блок вопросов по выбору студента. Модуль 2. Методы построения, анализа алгоритмов, теория трансляции программ

20. Компилируемые и интерпретируемые языки программирования. Разница. Примеры.
21. Основные фазы компиляции. Лексический анализатор. Синтаксический анализатор.

22. Жадные алгоритмы.
23. Понятие рекурсии. Примеры использования.
24. О-символика для оценки быстродействия. Примеры.

Критерии оценки:

✓ 100-86 баллов - если ответ показывает глубокое и систематическое знание всего программного материала и структуры конкретного вопроса, а также основного содержания и новаций лекционного курса по сравнению с учебной литературой. Студент демонстрирует отчетливое и свободное владение концептуально-понятийным аппаратом, научным языком и терминологией соответствующей научной области. Знание основной литературы и знакомство с дополнительно рекомендованной литературой. Логически корректное и убедительное изложение ответа.

✓ 85-76 - баллов - знание узловых проблем программы и основного содержания лекционного курса; умение пользоваться концептуально-понятийным аппаратом в процессе анализа основных проблем в рамках данной темы; знание важнейших работ из списка рекомендованной литературы. В целом логически корректное, но не всегда точное и аргументированное изложение ответа.

✓ 75-61 - балл – фрагментарные, поверхностные знания важнейших разделов программы и содержания лекционного курса; затруднения с использованием научно-понятийного аппарата и терминологии учебной дисциплины; неполное знакомство с рекомендованной литературой; частичные затруднения с выполнением предусмотренных программой заданий; стремление логически определенно и последовательно изложить ответ.

✓ 60-50 баллов – незнание, либо отрывочное представление о данной проблеме в рамках учебно-программного материала; неумение использовать понятийный аппарат; отсутствие логической связи в ответе.

Темы проектов

по дисциплине «Программирование для ЭВМ»

1. Арифметические операции + системы счисления

Примеры заданий

1) Запишите результат работы данной программы:

```
#include <iostream>
using namespace std;
void main()
{
	int i = 5, j = 10;
	double x;
	x = 1. / 5 * j++;
	x--;
	cout << " x = " << x << endl << "\n";
	x = --i % 2 + 30./2;
	cout << " x = " << x << "\n" << endl;
}
```

2) Запишите результат работы данной программы:

```
#include <iostream>
using namespace std;
void main()
{
	int i, j, k;
	k = 0;
	j = 1;
	i = j + 1;
```

```
    cout << (k ? "Yes!" : "No(") << endl;
    (i == 2 && j == 1) ? cout << "Yes!" : cout << "No(" << endl;
}
```

3) а) Перевести из десятичной системы в двоичную число (101,7).

б) (101,01)2 = (?)10.

в) (1111)2 = (?)8.

2. Операторы цикла

Примеры заданий

1) Создайте программу, выводящую на экран таблицу символов с указанием их 16-ного кода.

3. Указатели (язык программирования C++) и адреса

Примеры заданий

1) Написать программу на C++, демонстрирующую возможности языка при работе с указателями, а также способы их применения.

4. Сортировка одномерного массива на языке программирования C++

Примеры заданий

1) Реализовать два метода сортировки – сортировку прямым выбором и шейкерную сортировку. Сделать сравнительный анализ двух реализованных методов по времени работы и по количеству итераций на нескольких наборах данных. Вывести результаты в виде таблицы.

2) Реализовать два метода сортировки – пузырьковую сортировку и сортировку Шелла. Сделать сравнительный анализ двух реализованных методов по времени работы и по количеству итераций на нескольких наборах данных. Вывести результаты в виде таблицы.

5. Функции (язык программирования C++)

Примеры заданий

1) Создайте функцию вычисления конечной суммы ряда $\mathbf{a^n}$ с заданной точностью **eps**. Условием достижения необходимой точности на n-м шаге вычисления является неравенство $|\mathbf{a^n - a^{n+1}}|/|\mathbf{a^{n+1}}| < \mathbf{eps}$ (если оно выполнено, то точность вычислений считается достигнутой).

6. Динамические структуры данных (язык программирования C++)

Примеры заданий

1) Реализуйте программу хранения данных, использующую хеш-таблицы и хеш-функции.

7. Дано целое число N, кратное 4. Требуется написать программу, выводящую N целых чисел, сумма которых равна 0, а произведение равно N.

8. Стрелки часов движутся с постоянными угловыми скоростями и показывают **n** часов **m** минут. Найти число полных минут до ближайшего момента, в который стрелки совпадут.

Вход. Два целых числа nи m.

Выход. Целое число минут (на экране).

Примеры. Вход: 0, 0; выход: 0. Вход: 1, 1; выход: 4.

9. Найти наибольший общий делитель двух положительных чисел с помощью алгоритма Евклида. Числа вводятся с клавиатуры.

10. Дано вещественное число **a** и натуральное число **n**. Определить цифры, стоящие в десятичной записи числа **a** под номером **n** и в целой, и в дробной части.

Пример. Вход: 23674.45198, 3; выход: 6, 1.

11. Определить количество пятниц, выпадающих на 13 число в текущем году.

12. Вывести на экран последовательность первых 100 чисел Фибоначчи. В последовательности Фибоначчи каждое последующее число равняется сумме двух предыдущих. Последовательность начинается с двух единиц.

13. Вывести на экран все числа-палиндромы (т.е. числа, символьные записи которых можно читать и слева направо, и справа налево, и эти две записи полностью совпадают) из заданного интервала натурального ряда **[n1, n2]**.

Вход. Интервал [n1, n2].

Выход. Числа-палиндромы из этого интервала, записанные в столбец.

Пример. Вход: [100, 130]; выход: 101, 111, 121.

14. В выражении **(((1 ? 2) ? 3) ? 4) ? 5) ? 6** вместо каждого вопросительного знака вставить знак одного из четырех арифметических действий: +, -, *, /; так, чтобы результат вычислений равнялся заданному пользователем числу. Достаточно найти одно решение, но нужно, чтобы программа сама подставляла в выражение разные знаки и их комбинации.

15. Представить заданное натуральное число в виде суммы двух квадратов, или выдать сообщение, что такое представление невозможно.

16. Вывести все числа из интервала **[1, 10000]**, сумма цифр которых равна заданному натуральному числу n.

Вход. Число n.

Выход. Ряд чисел, сумма цифр каждого из которых равна **n**.

Пример. Вход: 5; выход: 14, 23, 32, 41, 50, 104, 113, 122, 131, 140, 203, …

17. Напечатать все четырехзначные натуральные числа, в десятичной записи которых нет двух одинаковых цифр.

18. Напечатать в порядке возрастания все простые несократимые дроби, заключенные между 0 и 1, знаменатели которых не превышают 7.

19. Напишите программу, которая требует от пользователя ввести количетво дней, а затем переводит это значение в количество недель и дней. Например, она переводит 18 дней в 2 недели и 4 дня. Отобразите результаты в следующем формате:

18 дней составляют 2 недели и 4 дня.

Используйте цикла **while**, чтобы дать пользователю возможность многократного ввода количества дней. Чтобы закончить цикл, пользователь должен ввести неположительное значение, например, 0 или -20.

20. Напишите программу, печатающую таблицу, в каждой строке которой представлено целое число, его квадрат и его куб. Запросите у пользователя верхний и нижний пределы таблицы. Используйте цикл **for**.

Критерии оценки:

✓ 100-86 баллов выставляется, если студент/группа точно определили содержание и составляющие части задания, умеют аргументированно отвечать на вопросы, связанные с заданием. Продемонстрировано знание и владение навыками самостоятельной исследовательской работы по теме. Фактических ошибок, связанных с пониманием проблемы, нет.

✓ 85-76 - баллов - работа студента/группы характеризуется смысловой цельностью, связностью и последовательностью изложения; допущено не более 1 ошибки при объяснении смысла или содержания проблемы. Продемонстрированы исследовательские умения и навыки. Фактических ошибок, связанных с пониманием проблемы, нет.

✓ 75-61 балл – проведен достаточно самостоятельный анализ основных этапов и смысловых составляющих проблемы; понимание базовых основ и теоретического обоснования выбранной темы. Привлечены основные источники по рассматриваемой теме. Допущено не более 2 ошибок в смысле или содержании проблемы

✓ 60-50 баллов - если работа представляет собой пересказанный или полностью переписанный исходный текст без каких бы то ни было комментариев, анализа. Не раскрыта структура и теоретическая составляющая темы. Допущено три или более трех ошибок смыслового содержание раскрываемой проблемы

Описание показателей и критериев оценивания компетенций, шкал оценивания

Критерии оценки собеседования

✓ 100-86 баллов - если ответ показывает глубокое и систематическое знание всего программного материала и структуры конкретного вопроса, а также основного содержания и новаций курса по сравнению с учебной литературой. Студент демонстрирует отчетливое и свободное владение концептуально-понятийным аппаратом, научным языком и терминологией соответствующей научной области. Знание основной литературы и знакомство с дополнительно рекомендованной литературой. Логически корректное и убедительное изложение ответа.

✓ 85-76 - баллов - знание узловых проблем программы и основного содержания курса; умение пользоваться концептуально-понятийным аппаратом в процессе анализа основных проблем в рамках данной темы; знание важнейших работ из списка рекомендованной литературы. В целом логически корректное, но не всегда точное и аргументированное изложение ответа.

✓ 75-61 - балл – фрагментарные, поверхностные знания важнейших разделов программы и содержания курса; затруднения с использованием научно-понятийного аппарата и терминологии учебной дисциплины; неполное знакомство с рекомендованной литературой; частичные затруднения с выполнением предусмотренных программой заданий; стремление логически определенно и последовательно изложить ответ.

✓ 60-50 баллов – незнание, либо отрывочное представление о данной проблеме в рамках учебно-программного материала; неумение использовать понятийный аппарат; отсутствие логической связи в ответе.

Итоговая шкала оценивания

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

Методические рекомендации, определяющие процедуры оценивания результатов освоения дисциплины

Текущая аттестация студентов. Текущая аттестация студентов по дисциплине «Программирование для ЭВМ» проводится в соответствии с локальными нормативными актами ДВФУ и является обязательной.

Текущая аттестация по дисциплине «Программирование для ЭВМ» проводится в форме собеседования и выполнения проектных работ и осуществляется ведущим преподавателем.

Объектами оценивания выступают:

– степень усвоения теоретических знаний - оценивается в форме собеседования;

– уровень овладения практическими умениями и навыками – оценивается в форме выполнения проектных работ.

Промежуточная аттестация студентов. Промежуточная аттестация студентов по дисциплине «Программирование для ЭВМ» проводится в соответствии с локальными нормативными актами ДВФУ и является обязательной.

По дисциплине предусмотрены экзамен в 1 семестре и зачёт во 2 семестре, которые проводится в устной форме.

Критерии выставления оценки студенту на зачёте/ экзамене по дисциплине «Программирование для ЭВМ»

Баллы (рейтинговой оценки)	**Оценка зачета/ экзамена** (стандартная)	**Требования к сформированным компетенциям**
	«зачтено»/ «отлично»	Оценка «отлично» выставляется студенту, если он глубоко и прочно усвоил программный материал, исчерпывающе, последовательно, четко и логически стройно его излагает, умеет тесно увязывать теорию с практикой, свободно справляется с задачами, вопросами и другими видами применения знаний, причем не затрудняется с ответом при видоизменении заданий, правильно обосновывает принятое решение, владеет разносторонними навыками и приёмами выполнения лабораторных работ; знает современные технологии программирования

		на языке C++; умеет разрабатывать программные продукты с помощью современных технологий программирования; владеет эффективными методами разработки программных продуктов с помощью современных технологий программирования; знает технологии отладки и тестирования прикладного программного обеспечения; умеет отлаживать и тестировать прикладное программное обеспечение; владеет методами отладки и тестирования прикладного программного обеспечения; знает способы тестирования программных средств, в том числе, разработанных самостоятельно; умеет находить и исправлять ошибки в тестируемых программных продуктах; владеет навыками настройки вычислительной техники для работы в интегрированной среде разработки, навыками тестирования и проверки программных продуктов; знает современные языки программирования (в частности, языки C и C++); умеет применять знания современных языков программирования для разработки программных продуктов; владеет навыками разработки программ на современных языках программирования.
	«зачтено»/ «хорошо»	Оценка «хорошо» выставляется студенту, если он твердо знает материал, грамотно и по существу излагает его, не допуская существенных неточностей в ответе на вопрос, правильно применяет теоретические положения при решении практических вопросов и задач, владеет необходимыми навыками и приемами их выполнения; знает современные технологии программирования на языке C++; умеет разрабатывать программные продукты с помощью современных технологий программирования; владеет эффективными методами разработки программных продуктов с помощью современных технологий программирования; умеет находить и исправлять ошибки в тестируемых программных продуктах; владеет навыками настройки вычислительной техники для работы в интегрированной среде разработки; знает современные языки программирования (в частности, языки C и C++); умеет применять знания современных языков

		программирования для разработки программных продуктов; владеет навыками разработки программ на современных языках программирования.
	зачтено»/ «удовлетворительно»	Оценка «удовлетворительно» выставляется студенту, если он имеет знания только основного материала, но не усвоил его деталей, допускает неточности, недостаточно правильные формулировки, нарушения логической последовательности в изложении программного материала, испытывает затруднения при выполнении лабораторных работ; частично владеет навыками разработки программ на современных языках программирования.
	«не зачтено»/ «неудовлетворительно»	Оценка «неудовлетворительно» выставляется студенту, который не знает значительной части программного материала, допускает существенные ошибки, неуверенно, с большими затруднениями выполняет лабораторные работы, практически не владеет навыками разработки программ на современных языках программирования. Как правило, оценка «неудовлетворительно» ставится студентам, которые не могут продолжить обучение без дополнительных занятий по соответствующей дисциплине.