МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ярославский государственный университет им. П.Г. Демидова

Кафедра вычислительных и программных систем

УТВЕРЖДАЮ
Декан факультета ИВТ
_____ Д.Ю. Чалый
«_24_» ___мая_____ 2022 г.

Рабочая программа дисциплины
«Основы программирования»

Направление подготовки
02.03.02 Фундаментальная информатика и информационные технологии

Профиль
«Информатика и компьютерные науки»

Квалификация выпускника
Бакалавр

Форма обучения
очная

Программа рассмотрена
на заседании кафедры
от 17 марта 2022 г.,
протокол № 7

Программа одобрена НМК
факультета ИВТ
протокол № 6 от
18 апреля 2022 г.

Ярославль

1. Цели освоения дисциплины

Целями дисциплины «Основы программирования» являются ознакомление студентов с понятием алгоритма, способами и средствами их представления, классификацией и эволюцией языков программирования и современными тенденциями их развития, а также формирование устойчивых практических навыков алгоритмизации и разработки программ решения вычислительных задач и задач обработки информации с использованием одного из языков высокого уровня (язык С).

2. Место дисциплины в структуре ОП бакалавриата

Дисциплина «Основы программирования» относится к базовой части ОП бакалавриата.

Для освоения данной дисциплиной студенты должны обладать знаниями по математике и информатике в объеме школьной программы, проявлять настойчивость, целеустремленность и инициативу в процессе обучения.

Знания и навыки, полученные при ее изучении, представляют собой основу для изучения учебных дисциплин компьютерной направленности. Они используются учащимися при изучении последующих общепрофессиональных и специальных дисциплин компьютерного цикла, в частности дисциплин "Объектно-ориентированное программирование", "Операционные системы", "Системы программирования", "Функциональное программирование", "Разработка программных проектов", а также ряда других учебных дисциплин.

3. Планируемые результаты обучения по дисциплине, соотнесенные с планируемыми результатами освоения ОП бакалавриата

Процесс изучения дисциплины направлен на формирование следующих элементов компетенций в соответствии с ФГОС ВО, ОП ВО и приобретения следующих знаний, умений, навыков и (или) опыта деятельности:

Формируемая компетенция (код и формулировка)	**Индикатор достижения компетенции (код и формулировка)**	**Перечень планируемых результатов обучения**
Общепрофессиональные компетенции		
ОПК-3 Способен к разработке алгоритмических и программных решений в области системного и прикладного программирования, математических, информационных и имитационных моделей, созданию информационных ресурсов	ОПК-3.2 Знает современные языки программирования	**Знать:** – средства описания алгоритмов; – назначение и категории языков программирования – основы программирования на языке С; – принципы разработки программ и отдельных программных модулей. **Уметь:** – пользоваться современными IDE для разработки прикладных приложений. – составлять программы для решения вычислительных задач и задач обработки информации, используя при этом одну из распределённых систем управления версиями (Git); – выполнять отладку и тестирование программ;

глобальных сетей, образовательного контента, прикладных баз данных, тестов и средств тестирования систем и средств на соответствие стандартам и исходным требованиям		– оформлять программную документацию с использованием современных офисных средств. **Владеть навыками:** **–** использования инструментальных средств современными IDE для создания, отладки и тестирования программ и отдельных модулей; **–** использования современных коммуникативных средств, включая Zoom и Moodle; – работы с научно-технической литературой и технической документацией по программному обеспечению; – поиска необходимой информации в глобальной сети Интернет

4. Объем, структура и содержание дисциплины

Общая трудоемкость дисциплины составляет 4 зач. ед., 144 акад. час.

№ п/п	Темы (разделы) дисциплины, их содержание	Семестр	Виды учебных занятий, включая самостоятельную работу студентов, и их трудоемкость (в академических часах)						Формы текущего контроля успеваемости Форма промежуточной аттестации *(по семестрам)*
			Контактная работа						
			лекции	практические	лабораторные	консультации	аттестационные испытания	самостоятельная работа	
1.1	Введение	1	6			2		6	
2.2	Базовые конструкции программирования на языке С	1	14			4		12	
3.1	Функции, классы памяти, конструирование типов, управление процессом компиляции	1	12			4	**8**	10	**Коллоквиум**
4.2	Стандартные библиотеки языка С	1	10			2		8	
5.2	Типовые структуры данных и приемы программирования	1	9			2		9	
							26		**Экзамен**
	Всего за 1 семестр		**51**			**14**	**34**	**45**	

	Всего		**51**			**14**	**34**	**45**	

Для самостоятельной работы используются задания из учебного пособия [3]. Примерные задания для контроля текущей успеваемости приведены в приложении.

Экзамен проводится в письменной форме. Примерные задания приведены в приложении.

Содержание разделов дисциплины:

1. ***Введение***
   - Задачи курса и порядок его изучения.
   - Понятие алгоритма. Сущность алгоритмизации вычислительных процессов.
   - Данные и алгоритмический процесс. Представление алгоритмов.
   - Понятие языка программирования. Синтаксис и семантика.
   - Классификация языков, история разработки языков программирования. Парадигмы программирования.
   - Общая характеристика языка программирования С, сходство и отличия от других языков.
   - Понятие операционной системы, текстового редактора, среды разработки.
   - Процесс создания и исполнения программы на компьютере. Интерпретация и компиляция.
   - Процесс отладки и тестирования.

2. ***Базовые конструкции программирования на языке С***
   - Введение в язык С. Пример простой программы на С. Структура простой программы. Алфавит языка С, идентификаторы. Использование комментариев.
   - Понятие типа данных. Переменные и константы. Базовые типы данных языка С, их размер, машинное представление. Операция sizeof.
   - Основные арифметические операции, порядок их выполнения. Операции ++ и --. .Преобразование типов, операция приведения.
   - Операции, выражения и операторы в языке С. Составной оператор, пустой оператор.
   - Простейшие средства ввода-вывода. Функции getchar(), putchar(), printf(), scanf(). Спецификации преобразования, используемые в функции printf(). Строковые константы, символьные строки.
   - Выбор вариантов. Оператор if в полной и сокращенной форме. Вложенные условные операторы.
   - Операции отношения. Понятие "истина" в языке С. Логические операции.
   - Множественный выбор: операторы switch и break.
   - Побитовые операции. Условная операция ?:.
   - Программирование циклических процессов. Операторы циклов while, for, do. Ключевые слова break, continue. Операция запятая. Вложенные циклы. Оператор перехода goto.
   - Массивы: описание, использование, доступ к элементам, ввод и вывод. Многомерные массивы.

3. ***Функции, классы памяти, конструирование типов, управление процессом компиляции***
   - Функции в языке С. Формальные и фактические параметры. Возвращение значения функцией. Описание типа функции. Оператор return. Локальные переменные функции.

- Нахождение адреса переменной: операция &. Первое знакомство с указателями. Описание указателей. Операция косвенной адресации *. Использование указателей для связи между функциями.
- Структура программы, состоящей из нескольких функций. Равноправие функций. Рекурсия, ее использование. Косвенная рекурсия. Механизм вызова функции, понятие стека.
- Классы памяти и области видимости. Автоматические, внешние, статические, регистровые переменные. Инициализация переменных, массивов, многомерных массивов.
- Массивы и указатели. Использование имен массивов в качестве аргументов функции. Использование указателей для работы с массивами. Операции с указателями. Указатели и многомерные массивы.
- Функции и многомерные массивы. Указатели на функции.
- Массивы указателей. Указатели на указатели. Аргументы функции main().
- Структуры, их описание, инициализация, доступ к элементам. Вложенные структуры. Указатели на структуры, их описание и использование. Битовые поля в структурах.
- Объединения. Перечисляемые типы данных, их назначение и примеры использования.
- Оператор typedef.
- Препроцессор языка С. Директива #define. Макроопределения с параметрами, их отличие от функций. Директива #include. Заголовочные файлы. Директивы #undef, #ifdef, #ifndef, #if, #else, #endif. Понятие условной компиляции.
- Раздельная компиляция, организация использования общих данных. Понятие проекта, в частности, устройство файлов решения и проекта в среде разработки Microsoft Visual Studio.
- Создание и использование собственных статических библиотек. Понятие многокомпонентной разработки.

4. Стандартные библиотеки языка С

- Библиотеки ввода-вывода. Понятие потока ввода-вывода. Типы потоков. Стандартные потоки. Перенаправление потоков.
- Структуры данных и функции для работы с потоками ввода-вывода.
- Функции обработки строк, преобразования строки в число и обратно.
- Функции динамического распределения и освобождения памяти.
- Функции для работы с логическими дисками, директориями, функции поиска. Функции запуска процессов.

5. Типовые структуры данных и приемы программирования

- Однонаправленные, двунаправленные, циклические списки, бинарные деревья, их применение для решения типовых задач. Эффективная работа с памятью при реализации динамических структур данных. Минимизация фрагментации динамически распределяемой памяти.
- Использование рекурсии для повышения эффективности программирования. Примеры рекурсивной организации процесса вычислений, оценка трудоемкости.

5. Образовательные технологии, используемые при осуществлении образовательного процесса по дисциплине

В процессе обучения используются следующие образовательные технологии:

Вводная лекция – дает первое целостное представление о дисциплине и ориентирует студента в системе изучения данной дисциплины. Студенты знакомятся с назначением и задачами курса, его ролью и местом в системе учебных дисциплин и в

системе подготовки в целом. Дается краткий обзор курса, история развития науки и практики, достижения в этой сфере, имена известных ученых, излагаются перспективные направления исследований. На этой лекции высказываются методические и организационные особенности работы в рамках данной дисциплины, а также дается анализ рекомендуемой учебно-методической литературы.

Академическая лекция (или лекция общего курса) – последовательное изложение материала, осуществляемое преимущественно в виде монолога преподавателя. Требования к академической лекции: современный научный уровень и насыщенная информативность, убедительная аргументация, доступная и понятная речь, четкая структура и логика, наличие ярких примеров, научных доказательств, обоснований, фактов.

Занятия по данной дисциплине проводятся в форме традиционной лекции, поскольку для студентов младших курсов необходимо обеспечить возможность тесного общения с преподавателем. Он имеет возможность в любой момент задать вопрос и немедленно получить ответ, причем вопрос и ответ должна слышать вся аудитория. Практическое применение полученных знаний также отрабатывается в рамках курса "Практикум ЭВМ по основам программирования", занятия по этому курсу проводятся в компьютерном классе и в значительной мере синхронизированы с лекционным материалом. Большинство преподавателей имеет опыт коммерческой разработки программных средств, а часть из них и в настоящий момент совмещает работу в коммерческих и промышленных организациях с преподаванием в университете.

Лабораторное занятие – занятие в компьютерном классе, посвященное освоению конкретных умений и навыков и закреплению полученных на лекции знаний.

Основной формой практической работы студентов по усвоению данного курса является выполнения ими самостоятельных проектов в рамках занятий по дисциплине "Практикум ЭВМ по основам программирования". Все задания и вспомогательные учебные материалы предоставляются студентам как в электронном, так и в напечатанном виде [3, 4]. Указанные издания содержат также необходимый теоретический материал и примеры кода для усвоения всех тем данного курса.

Промежуточная аттестация производится в форме отчетов студентов по самостоятельно выполненным проектам в рамках курса "Практикум ЭВМ по основам программирования" и при проведении коллоквиума, окончательная аттестация в форме экзамена.

6. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень лицензионного программного обеспечения и информационных справочных систем (при необходимости)

В процессе осуществления образовательного процесса используются:

– Windows 10 в качестве основной операционной системы;

– Интегрированная среда разработки Microsoft Visual Studio в сочетании со средствами Git для подготовки демонстрационных примеров, выполнения лабораторных работ и учебных упражнений;

– Microsoft OfficeStd 2013 RUS OLP NL Acdmc 021-10232 для формирования инструкций по выполнению учебных упражнений, а также текстов материалов для промежуточной и текущей аттестации;

– Программа для организации видеоконференций Zoom для проведения удаленных (при необходимости) занятий и консультаций;

– LMS "Электронный университет Moodle ЯрГУ" для размещения дополнительных материалов, связи с преподавателями и при необходимости приема выполненных заданий;

– Автоматизированная библиотечная информационная система "БУКИ-NEXT" (АБИС "Буки-Next") для поиска учебной литературы в библиотеке ЯрГУ.

7. Перечень основной и дополнительной учебной литературы, ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

а) основная:

1. В.В.Васильчиков, Н.С.Лагутина, Ю.А.Ларина Основы программирования на языке С: учебное пособие. - Ярославль.: ЯрГУ, 2006.
2. Васильчиков, В. В., Основы программирования на языке С [Электронный ресурс] : учеб. пособие для вузов / В. В. Васильчиков, Н. С. Лагутина, Ю. А. Ларина ; Яросл. гос. ун-т, Ярославль, ЯрГУ, 2006, 79с.
3. Лагутина, Н. С., С++. Примеры и задачи [Электронный ресурс] : практикум / Н. С. Лагутина ; Яросл. гос. ун-т, Ярославль, ЯрГУ, 2011, 47с

б) дополнительная:

1. Подбельский, В. В., Практикум по программированию на языке Си (+CD) : учеб. пособие, М., Финансы и статистика, 2004, 575с
2. Болски, М. И., Язык программирования Си : справочник : пер. с англ., М., Радио и связь, 1988, 99с
3. Джехани, Н., Программирование на языке Си : пер. с англ., М., Радио и связь, 1988, 270с
4. Уэйт, М., Язык СИ : руководство для начинающих / М. Уэйт, С. Прата, Д. Мартин ; пер. с англ., М., Мир, 1988, 512с

в) ресурсы сети «Интернет»

1. Информация по языкам программирования, операционным системам, примеры программ: www.firststeps.ru, www.corp7.ivt.uniyar.ac.ru
2. Электронная библиотека учебных материалов ЯрГУ: http://www.lib.uniyar.ac.ru/opac/bk_cat_find.php
3. https://docs.microsoft.com/ru-ru/

8. Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине

Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине включает в свой состав специальные помещения:
– учебные аудитории для проведения занятий лекционного типа и практических занятий (семинаров);
– учебные аудитории для проведения лабораторных занятий;
– учебные аудитории для проведения групповых и индивидуальных консультаций,
– учебные аудитории для проведения текущего контроля и промежуточной аттестации;
– помещения для самостоятельной работы;
– помещения для хранения и профилактического обслуживания технических средств обучения.

Специальные помещения укомплектованы средствами обучения, служащими для представления учебной информации большой аудитории.

Для проведения занятий лекционного типа предлагаются наборы демонстрационного оборудования и учебно-наглядных пособий, хранящиеся на электронных носителях и обеспечивающие тематические иллюстрации, соответствующие рабочим программам дисциплин.

Помещения для лабораторных занятий и самостоятельной работы обучающихся оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и

обеспечением доступа в электронную информационно-образовательную среду организации.

Число посадочных мест в лекционной аудитории больше либо равно списочному составу потока, а в аудитории для практических занятий (семинаров) – списочному составу группы обучающихся.

Автор(ы) :

Зав. кафедрой
вычислительных и программных систем, к.т.н. _____ В.В. Васильчиков

Приложение №1
к рабочей программе дисциплины
"Основы программирования"

Фонд оценочных средств
для проведения текущей и промежуточной аттестации студентов
по дисциплине

1. Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

1.1. Контрольные задания и иные материалы, используемые в процессе текущей аттестации

Задания для самостоятельной работы

Для самостоятельной работы используются задания из учебного пособия [3]: В.В.Васильчиков, Н.С.Лагутина, Ю.А.Ларина "Основы программирования на языке С: учебное пособие" - Ярославль.: ЯрГУ, 2006.

Примерное распределение их по темам указано в графе "Форма промежуточной аттестации" таблицы "Структура и содержание дисциплины" раздела 4 данной рабочей программы.

Текущая аттестация происходит в форме отчета по выполнению индивидуального задания по заданной теме. Набор тестов для задания разрабатывается студентом и согласуется с преподавателем, который может их модифицировать или дополнить.

Задания по теме «Программирование циклических процессов»

Для изучения данной темы используются задания A1.1-A1.2, B1.3-B1.6, C1.7-C1.11 из учебного пособия [3].

Критерии оценивания

Оценка	Критерии
Отлично	ОПК-3: Знает все операторы цикла языка С, умеет при необходимости изменить программу, использовав другой оператор цикла. Программа выполняется для всех тестов. Создает полноценное консольное приложение в среде разработки. Поясняет код и изменяет его при необходимости
Хорошо	ОПК-3: Знает операторы цикла языка С, умеет при необходимости изменить программу, использовав другой оператор цикла, в некоторых случаях с подсказкой преподавателя. Программа выполняется для всех тестов, если какой-либо из тестов не выполняется программа дорабатывается. Создает консольное приложение в среде разработки. Поясняет код и изменяет его при необходимости с небольшими неточностями
Удовлетворительно	ОПК-3: Знает операторы цикла языка С, но не может без подсказки выбрать методы и полностью самостоятельно составить алгоритм для решения задачи либо решает только самые простые задачи. Программа выполняется для всех тестов, возможно кроме одного-двух крайних случаев. Создает консольное приложение в среде разработки. С трудом поясняет код, не может изменить код при усложнении или существенном

	дополнении задачи
Неудовлетворительно	ОПК-3: Не знает операторов цикла. Не может составить работающий алгоритм решения задачи. Программа не выполняется для большинства тестов. Не может создать консольное приложение в среде разработки или создает неработающий проект. Не может пояснить код и изменить его.

Задания по теме «Массивы и указатели, указатели на указатели»

Для изучения данной темы используются задания A2.1-A2.2, B2.3-B2.6, C2.7-C2.10 из учебного пособия [3].

Критерии оценивания

Оценка	Критерии
Отлично	ОПК-3: Умеет работать со всеми типами массивов. Свободно использует косвенную адресацию для доступа к элементам. Может использовать указатели на указатели, в том числе для доступа к аргументам командной строки. Программа выполняется для всех тестов. Создает полноценное консольное приложение в среде разработки. Поясняет код и изменяет его при необходимости
Хорошо	ОПК-3: Умеет работать с разными типами массивов. Может использовать косвенную адресацию для доступа к элементам. Может использовать аргументы командной строки. Программа выполняется для всех тестов, если какой-либо из тестов не выполняется программа дорабатывается. ПК-7: Создает консольное приложение в среде разработки. Поясняет код и изменяет его при необходимости с небольшими неточностями
Удовлетворительно	ОПК-3: Умеет работать с массивами разной размерности. Может получить информацию через аргументы командной строки. Программа выполняется для всех тестов, возможно кроме одного-двух крайних случаев. Создает консольное приложение в среде разработки. С трудом поясняет код, не может изменить код при усложнении или существенном дополнении задачи
Неудовлетворительно	ОПК-3: Не умеет работать с массивами. Не может получить информацию через аргументы командной строки. Программа не выполняется для большинства тестов. Не может создать консольное приложение в среде разработки или создает неработающий проект. Не может пояснить код и изменить его.

Задания по теме «Стандартные средства ввода и вывода»

Для изучения данной темы используются задания A5.1-A5.7, B5.8-B5.16, C5.17-C5.21 из учебного пособия [3].

Критерии оценивания

Оценка	Критерии
Отлично	ОПК-3: Знает, что такое поток ввода-вывода. Умеет использовать потоки для ввода и вывода информации из файлов, а также перенаправления ввода и вывода. Может организовать ввод и вывод максимально эффективным способом. Программа

	выполняется для всех тестов. Создает полноценное консольное приложение в среде разработки. Поясняет код и изменяет его при необходимости
Хорошо	ОПК-3: Умеет использовать потоки для ввода и вывода информации из файлов, а также перенаправления ввода и вывода. Может организовать ввод и вывод достаточно эффективным способом. Программа выполняется для всех тестов, если какой-либо из тестов не выполняется программа дорабатывается. Создает консольное приложение в среде разработки. Поясняет код и изменяет его при необходимости с небольшими неточностями
Удовлетворительно	ОПК-3: Умеет использовать потоки для ввода и вывода информации из файлов. Программа выполняется для всех тестов, возможно кроме одного-двух крайних случаев. Создает консольное приложение в среде разработки. С трудом поясняет код, не может изменить код при усложнении или существенном дополнении задачи
Неудовлетворительно	ОПК-3: Не умеет потоки для ввода и вывода информации из файлов. Программа не выполняется для большинства тестов. Не может создать консольное приложение в среде разработки или создает неработающий проект. Не может пояснить код и изменить его.

Задания по теме «Динамические структуры данных»

Для изучения данной темы используются задания А6.1-А6.8, В6.9-В6.11, С6.12-С6.14 из учебного пособия [3].

Критерии оценивания

Оценка	Критерии
Отлично	ОПК-3: Знает все основные динамические структуры данных. Может реализовать их программно и выбрать наиболее эффективную структуру для решения заданной задачи. Умеет эффективно реализовать эти структуры данных через динамическое выделение памяти. Программа выполняется для всех тестов. Создает полноценное консольное приложение в среде разработки. Поясняет код и изменяет его при необходимости
Хорошо	ОПК-3: Знает основные динамические структуры данных. Может реализовать их программно. Умеет реализовать эти структуры данных через динамическое выделение памяти. Программа выполняется для всех тестов, если какой-либо из тестов не выполняется программа дорабатывается. Создает консольное приложение в среде разработки. Поясняет код и изменяет его при необходимости с небольшими неточностями
Удовлетворительно	ОПК-3: Знает хотя бы две динамические структуры данных. Может реализовать их программно. Программа выполняется для всех тестов, возможно кроме одного-двух крайних случаев. Создает консольное приложение в среде разработки. С трудом поясняет код, не может изменить код при усложнении или существенном дополнении задачи
Неудовлетворительно	ОПК-3: Не знает динамических структур данных, не умеет их

	программно реализовать. Программа не выполняется для большинства тестов. Не может создать консольное приложение в среде разработки или создает неработающий проект. Не может пояснить код и изменить его.

1.2. Примерные варианты заданий для проведения письменного экзамена по курсу "Основы программирования "

ВАРИАНТ 1

ЗАДАНИЕ № 1 Найти синтаксические ошибки в программе

```
#define B 20                                  /*  1 */
#define B color                               /*  2 */
#define A B                                   /*  3 */
main (int argc, int argv)                     /*  4 */
{ int argc; char **argv;                      /*  5 */
  float f1, f2; int A;                        /*  6 */
  f2=function(A=3);                           /*  7 */
  printf("Привет!",f1=f2+A);                  /*  8 */
}                                             /*  9 */
#define break qq                              /* 10 */
float function(int break)                     /* 11 */
{                                             /* 12 */
  int i=0, B=10;                              /* 13 */
  for (i=0, break=color; i<b; i+=A)           /* 14 */
    break*=.1e-2;                             /* 15 */
  return }                                    /* 16 */
```

ЗАДАНИЕ № 2 Что напечатает программа ?

```
#include <stdio.h>

int A[6];

int func (int a, int *c)
{
  a+=c[3]; c[0]=c[1]++;
  if (a%3) return a; else return *c+3;
}

void main()
{ int i,*p;
  for (i=1;i<=4;i++) A[i]=5-i;
  *A=func(A[5]++,A+1);
  for(p=A,i=4;i;) printf("%3d",p[i--]);
  printf("\n");
}
```

ЗАДАНИЕ № 3

(Работа должна быть оформлена как функция, должен быть приведен пример ее вызова и описано, где содержится результат ее работы)

Дан односвязный список элементов, описанных посредством структуры

```
struct List
  { int Value;
    struct List *Next;
  } *First;
```

Поле Value содержит некоторое целое значение, поле Next используется для связи элементов в списке и содержит указатель на следующий элемент либо нулевой указатель, если следующего элемента нет. Глобальная переменная First содержит указатель на первый элемент списка либо нулевой указатель, если список пуст.

Написать функцию, которая переупорядочивает список задом наперед.

ВАРИАНТ 2

ЗАДАНИЕ № 1 Найти синтаксические ошибки в программе

```
#define A 10                                    /*  1 */
#define B A+20                                  /*  2 */
#ifndef C                                       /*  3 */
#undef A                                        /*  4 */
#endif                                          /*  5 */
                                                /*  6 */
int main (int ac)                               /*  7 */
{                                               /*  8 */
  int A;                                        /*  9 */
  struct { int i, j, k } *a;                    /* 10 */
  int i, j; float f=1;                          /* 11 */
  for (a->i=0; a->j<a->k)                       /* 12 */
  { a<<=ac; ac<<=f;                             /* 13 */
    a->j=(a->i==a->k); }                        /* 14 */
  if (A=15) printf('%d',A=B);                   /* 15 */
  if (B=30) ac=A;                               /* 16 */
}                                               /* 17 */
```

ЗАДАНИЕ № 2 Что напечатает программа ?

```
#include <stdio.h>

int A[6];

int func (int a, int *c)
{
  c[0]=c[-1]++; a-=c[-3];
  return (a > *c ? a+1 : a-1);
}

void main()
{ int i,*p;
  for (i=4;i>0;--i) A[i]=2-i;
  A[2]=func((*A)++,A+5);
  for(p=A+1,i=-1;i++<3;) printf("%3d",p[i]);
  printf("\n");
}
```

ЗАДАНИЕ № 3

(Работа должна быть оформлена как функция, должен быть приведен пример ее вызова и описано, где содержится результат ее работы)

Дан односвязный список элементов, описанных посредством структуры

```
struct List
   { int Value;
     struct List *Next;
   } *First;
```

Поле Value содержит некоторое целое значение, поле Next используется для связи элементов в списке и содержит указатель на следующий элемент либо нулевой указатель, если следующего элемента нет. Глобальная переменная First содержит указатель на первый элемент списка либо нулевой указатель, если список пуст.

Написать функцию, которая создает копию N–го члена (если он есть) и помещает ее в конец списка.

Показатели и критерии, используемые при выставлении оценки:

Номер задачи	Критерии	Шкала оценивания
1	ОПК-3: Знать: ● синтаксис языка программирования С Уметь: ● находить и объяснять синтаксические ошибки в тексте программы	0 баллов – студент нашел менее половины синтаксических ошибок, при этом обозначил как ошибочные значительное количество синтаксически верных конструкций; 1 балл – студент нашел не менее половины синтаксических ошибок, при этом, возможно, обозначил как ошибочные несколько синтаксически верных конструкций; 2 балла – студент нашел все синтаксические ошибки, при этом не обозначил как ошибочные синтаксически верные конструкции.
2	ОПК-3: Знать: ● синтаксис языка программирования С ● механизм вызова функций и передачи им параметров ● механизм косвенной адресации	0 баллов – студент не разобрался в процессе исполнения полученного кода, вывод программы описал неправильно; 1 балл – студент в основном полностью процесс исполнения представленного кода, допустив небольшое количество неточностей в отчете о состоянии использованных в коде переменных, верно указал, что выведет программа;

	Владеть навыками: ● понимания представленного кода и процесса его исполнения	2 балла – студент полностью описал процесс исполнения представленного кода с подробным отчетом о состоянии использованных в нем переменных в каждый момент времени, верно и в правильном формате указал, что выведет программа.
3	ОПК-3: Знать: ● синтаксис языка программирования С ● механизм вызова функций и передачи им параметров ● механизм косвенной адресации Уметь: ● разработать алгоритм решения поставленной задачи ● реализовать его как функцию на языке программирования С	0 баллов – студент не сумел разработать правильный алгоритм решения задачи, в представленном коде функции имеются грубые синтаксические ошибки; 1 балла – студент составил и реализовал алгоритм (необязательно эффективный) в виде функции, частично спроектировал передачу ей исходных данных и получение результата вызывающей функцией, функция правильно работает не на всех корректных исходных данных, возможно, допущены некритичные синтаксические ошибки; 2 балла – студент составил и реализовал алгоритм (необязательно эффективный) в виде функции, спроектировал (необязательно эффективным образом) передачу ей исходных данных и получение результата вызывающей функцией, функция правильно работает почти на всех корректных исходных данных, на некорректных может выдавать неправильный результат, возможно, допущены некритичные синтаксические ошибки; 3 балла – студент составил и реализовал алгоритм (необязательно эффективный) в виде функции, спроектировал передачу ей исходных данных и получение результата вызывающей функцией, функция правильно работает на всех корректных исходных данных, на некорректных может выдавать неправильный результат; 4 балла – студент правильно составил и реализовал необходимый алгоритм в виде функции, грамотно спроектировал передачу ей исходных данных и получение результата вызывающей функцией, функция правильно работает на всех корректных исходных данных, на некорректных - выдает соответствующее сообщение.

Максимальное количество баллов по ОПК-3 – 8 баллов

Набранное количество баллов соответствует оценке за выполнение экзаменационной работы:

- менее 2 баллов за задание №3 — оценка «неудовлетворительно», в общей сумме менее 3 баллов
- не менее 2 баллов за задание №3 и не менее 1 балла за остальные задания, в общей сумме от 3 до 4 баллов — оценка «удовлетворительно», пороговый уровень формирования компетенции,
- не менее 2 баллов за задание №3 и не менее 2 баллов за остальные задания, в общей сумме от 5 до 6 баллов — оценка «хорошо», продвинутый уровень формирования компетенции,
- от 3 до 4 баллов за задание №3 и не менее 3 баллов за остальные задания, в общей сумме от 6 до 8 баллов — оценка «отлично», высокий уровень формирования компетенции.

Методические указания по выставлению итоговой оценки за экзамен.

Итоговая оценка по дисциплине «Основы программирования» формируется в два этапа.

Первый этап – оценивание работы студента в течение изучения курса на основе средней оценки за самостоятельные и контрольные работы. Если на этом этапе все аттестационные задания выполнены в срок и средний балл за текущую аттестацию больше трех студент допускается ко второму этапу. Если все аттестационные задания имели повышенную сложность, выполнены в срок с надлежащим качеством, по представлению преподавателя, ведущего лабораторные работы, студенту может быть выставлена итоговая оценка «отлично» автоматом досрочно.

Второй этап – проведение экзаменационной работы. Для получения положительной оценки за экзамен студент должен сдать письменную экзаменационную работу на положительную оценку. При выполнении этого условия оценка за экзаменационную работу считается итоговой.

2. Перечень компетенций, этапы их формирования, описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкалы оценивания

2.1. Шкала оценивания сформированности компетенций и ее описание

Оценивание уровня сформированности компетенций в процессе освоения дисциплины осуществляется по следующей трехуровневой шкале:

Пороговый уровень - предполагает отражение тех ожидаемых результатов, которые определяют минимальный набор знаний и (или) умений и (или) навыков, полученных студентом в результате освоения дисциплины. Пороговый уровень является обязательным уровнем для студента к моменту завершения им освоения данной дисциплины.

Продвинутый уровень - предполагает способность студента использовать знания, умения, навыки и (или) опыт деятельности, полученные при освоении дисциплины, для решения профессиональных задач. Продвинутый уровень превосходит пороговый уровень по нескольким существенным признакам.

Высокий уровень - предполагает способность студента использовать потенциал интегрированных знаний, умений, навыков и (или) опыта деятельности, полученных при освоении дисциплины, для творческого решения профессиональных задач и самостоятельного поиска новых подходов в их решении путем комбинирования и использования известных способов решения применительно к конкретным условиям. Высокий уровень превосходит пороговый уровень по всем существенным признакам.

2.2. Перечень компетенций, этапы их формирования, описание показателей и критериев оценивания компетенций на различных этапах их формирования

Код компе-тенции	Форма контроля	Этапы форми-рования (№ темы (раздела)	Показатели оценивания	Шкала и критерии оценивания компетенций на различных этапах их формирования		
				Пороговый уровень	Продвинутый уровень	Высокий уровень
Общепрофессиональные компетенции						
ОПК-3	Проверка выполнения лабораторных работ по темам № 1-7 [3] Экзамен	1-7 [3]	**Знать:** –средства описания алгоритмов; – основы программирования на языке С; – принципы разработки программ и отдельных программных модулей. **Уметь:** – пользоваться инструментальными средствами Visual Studio для разработки прикладных приложений; – составлять программы для решения вычислительных задач и задач	1. Знание понятия алгоритма, средств его записи. 2. Знание основных конструкций языка С. 3. Умение описать необходимые данные и функциональные требования к программе. 4. Программирование требуемой функциональности. 5. Реализация ввода и вывода в консольном	1. Знание понятия алгоритма, различных средств его представления. 2. Знание основных конструкций языка С. Четкое представление о его синтаксисе. 3. Умение описать необходимые данные и функциональные требования к программе. 4. Программирование требуемой функциональности. Умение оценить трудоемкость выполнения кода. 5. Реализация ввода и	1. Знание понятия алгоритма, различных средств его представления. 2. Знание основных конструкций языка С. Четкое представление о его синтаксисе. Четкое представление о процессе компиляции и использовании препроцессора. 3. Умение описать необходимые данные и функциональные требования к программе. Умение составить техническое задание. 4. Программирование требуемой функциональности. Умение оценить трудоемкость выполнения кода и сравнивать по этому показателю различные алгоритмы решения задачи. 5. Реализация ввода и вывода в

			обработки информации; – выполнять отладку и тестирование программ; – оформлять программную документацию. **Владеть:** **–** навыками использования инструментальных средств Visual Studio для создания, отладки и тестирования программ и отдельных модулей; – навыками работы с научно-технической литературой и технической документацией по программному обеспечению	режиме. 6. Понимание способа хранения данных программы. 7. Знание основных приемов отладки и тестирования: расстановка точек остановки, пошаговое выполнение, отслеживание значений переменных.	вывода в консольном режиме. Умение перенаправлять ввод и вывод, использовать файлы. 6. Понимание способа хранения данных программы. Понимание классов памяти. Понимание косвенной адресации и работы с динамически выделяемой памятью. 7. Знание основных приемов отладки и тестирования: расстановка точек остановки, пошаговое выполнение, отслеживание значений переменных, использование условных точек остановки, отслеживание состояния стека.	консольном режиме. Умение перенаправлять ввод и вывод, программировать работу с файлами. 6. Понимание способа хранения данных программы. Четкое представление о классах памяти и особенностях их использования. Эффективное использование динамически выделяемой памяти. 7. Знание основных приемов отладки и тестирования: расстановка точек остановки, пошаговое выполнение, отслеживание значений переменных, использование условных точек остановки, отслеживание состояния стека.

3. Методические рекомендации преподавателю по процедуре оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Целью процедуры оценивания является определение степени овладения студентом ожидаемыми результатами обучения (знаниями, умениями, навыками и (или) опытом деятельности).

Процедура оценивания степени овладения студентом ожидаемыми результатами обучения осуществляется с помощью методических материалов, представленных в разделе «Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций»

3.1 Критерии оценивания степени овладения знаниями¸ умениями, навыками и (или) опытом деятельности, определяющие уровни сформированности компетенций

Пороговый уровень (общие характеристики):

- владение основным объемом знаний по программе дисциплины;
- знание основной терминологии данной области знаний, стилистически грамотное, логически правильное изложение ответа на вопросы без существенных ошибок;
- владение инструментарием дисциплины, умение его использовать в решении стандартных (типовых) задач;
- способность самостоятельно применять типовые решения в рамках рабочей программы дисциплины;
- усвоение основной литературы, рекомендованной рабочей программой дисциплины;
- знание базовых теорий, концепций и направлений по изучаемой дисциплине;
- самостоятельная работа на практических и лабораторных занятиях, периодическое участие в групповых обсуждениях, достаточный уровень культуры исполнения заданий.

Продвинутый уровень (общие характеристики):

- достаточно полные и систематизированные знания в объёме программы дисциплины;
- использование основной терминологии данной области знаний, стилистически грамотное, логически правильное изложение ответа на вопросы, умение делать выводы;
- владение инструментарием дисциплины, умение его использовать в решении учебных и профессиональных задач;
- способность самостоятельно решать сложные задачи (проблемы) в рамках рабочей программы дисциплины;
- усвоение основной и дополнительной литературы, рекомендованной рабочей программой дисциплины;
- умение ориентироваться в базовых теориях, концепциях и направлениях по изучаемой дисциплине и давать им сравнительную оценку;
- самостоятельная работа на практических и лабораторных занятиях, участие в групповых обсуждениях, высокий уровень культуры исполнения заданий.

Высокий уровень (общие характеристики):

- систематизированные, глубокие и полные знания по всем разделам дисциплины;
- точное использование терминологии данной области знаний, стилистически грамотное, логически правильное изложение ответа на вопросы, умение делать обоснованные выводы;

- безупречное владение инструментарием дисциплины, умение его использовать в постановке и решении научных и профессиональных задач;
- способность самостоятельно и творчески решать сложные задачи (проблемы) в рамках рабочей программы дисциплины;
- полное и глубокое усвоение основной и дополнительной литературы, рекомендованной рабочей программой дисциплины;
- умение ориентироваться в основных теориях, концепциях и направлениях по изучаемой дисциплине и давать им критическую оценку;
- активная самостоятельная работа на практических и лабораторных занятиях, творческое участие в групповых обсуждениях, высокий уровень культуры исполнения заданий.

3.2 Описание процедуры выставления оценки

В зависимости от уровня сформированности каждой компетенции по окончании освоения дисциплины студенту выставляется оценка. Для дисциплин, изучаемых в течение нескольких семестров, оценка может выставляться не только по окончании ее освоения, но и в промежуточных семестрах. Вид оценки («отлично», «хорошо», «удовлетворительно», «неудовлетворительно», «зачтено», «не зачтено») определяется рабочей программой дисциплины в соответствии с учебным планом.

Высокий уровень формирования компетенций соответствует оценке «отлично» за самостоятельные, контрольные работы и экзаменационную работу.

Продвинутый уровень формирования компетенций соответствует оценке «хорошо» за самостоятельные, контрольные работы и экзаменационную работу.

Пороговый уровень формирования компетенций соответствует оценке «удовлетворительно» за самостоятельные, контрольные работы и экзаменационную работу.

Оценка «отлично» выставляется студенту, у которого каждая компетенция (полностью или частично формируемая данной дисциплиной) сформирована на высоком уровне.

Оценка «хорошо» выставляется студенту, у которого каждая компетенция (полностью или частично формируемая данной дисциплиной) сформирована не ниже, чем на продвинутом уровне.

Оценка «удовлетворительно» выставляется студенту, у которого каждая компетенция (полностью или частично формируемая данной дисциплиной) сформирована не ниже, чем на пороговом уровне.

Оценка «неудовлетворительно» выставляется студенту, у которого хотя бы одна компетенция (полностью или частично формируемая данной дисциплиной) сформирована ниже, чем на пороговом уровне.

Оценка «зачет» выставляется студенту, у которого каждая компетенция (полностью или частично формируемая данной дисциплиной) сформирована не ниже, чем на пороговом уровне.

Оценка «незачтено» выставляется студенту, у которого хотя бы одна компетенция (полностью или частично формируемая данной дисциплиной) сформирована ниже, чем на пороговом уровне.

Приложение №2
к рабочей программе дисциплины
"Основы программирования"

Методические указания для студентов по освоению дисциплины

Основной формой изложения учебного материала по дисциплине "Основы программирования" являются традиционные лекции, которые включают в себя большое количество примеров кода, что позволяет сделать материал лекций более наглядными, улучшает информативность и понимание изучаемого курса.

По большинству тем предусмотрены лабораторные работы, на которых происходит закрепление лекционного материала путем применения его к конкретным задачам и получение навыков разработки и отладки программных приложений. Выполнение этих лабораторных работ, а также выполнение индивидуальных заданий позволяет не только понять и закрепить теоретический материал, но и приобрести навык анализа предметной области и создания прикладных приложений на языке С.

Для успешного освоения дисциплины очень важно решение достаточно большого количества задач, как в аудитории, так и самостоятельно в качестве домашних заданий. Примеры решения задач разбираются и обсуждаются на лекциях и лабораторных занятиях. Основная цель решения задач – помочь усвоить фундаментальные понятия и основы разработки и отладки программных приложений. Для решения всех задач необходимо знать и понимать лекционный материал. Поэтому в процессе изучения дисциплины рекомендуется регулярное повторение пройденного лекционного материала. Материал, законспектированный на лекциях, необходимо дома еще раз прорабатывать и при необходимости дополнять информацией, полученной при выполнении лабораторных работ или из учебной литературы.

Большое внимание должно быть уделено самостоятельной домашней работе. В качестве заданий для самостоятельной работы дома студентам предлагаются задачи, аналогичные разобранным на лекциях и лабораторных занятиях или немного более сложные, которые являются результатом объединения нескольких базовых задач.

Для проверки и контроля усвоения теоретического материала, приобретенных практических навыков программирования на языке С, в течение всего периода обучения проводятся консультации и разбор самостоятельного выполнения индивидуальных заданий.

В конце семестра студенты сдают экзамен. Экзамен проводится в письменной форме. Задания позволяют проверить знание синтаксиса языка, умение понимать чужой код и, конечно, умение построить и реализовать на С свой алгоритм.

Освоить вопросы, излагаемые в процессе изучения дисциплины "Основы программирования" самостоятельно первокурснику достаточно сложно. Хорошее усвоение материала подразумевает не только умение составлять и реализовывать алгоритмы, но и понимание механизмов взаимодействия программы с операционной системой, способов хранения данных и доступа к ним, в том числе с использованием косвенной адресации. Поэтому посещение всех аудиторных занятий является совершенно необходимым. Не в меньшей степени это относится и к лабораторным занятиям, поскольку одной из важнейших целей данного учебного курса является формирование именно практических навыков программирования. Так что без упорных и регулярных занятий в течение семестра сдать экзамен по итогам изучения дисциплины студенту будет сложно.

Учебно-методическое обеспечение
самостоятельной работы студентов по дисциплине

Для самостоятельной работы особенно рекомендуется использовать учебную литературу, указанную в разделе № 7 данной рабочей программы.

Также для подбора учебной литературы рекомендуется использовать широкий спектр интернет-ресурсов:

1. Электронно-библиотечная система «Университетская библиотека online» (www.biblioclub.ru) - электронная библиотека, обеспечивающая доступ к наиболее востребованным материалам-первоисточникам, учебной, научной и художественной литературе ведущих издательств (*регистрация в электронной библиотеке – только в сети университета. После регистрации работа с системой возможна с любой точки доступа в Internet.).

2. Для самостоятельного подбора литературы в библиотеке ЯрГУ рекомендуется использовать:

1. Личный кабинет (http://lib.uniyar.ac.ru/opac/bk_login.php) дает возможность получения on-line доступа к списку выданной в автоматизированном режиме литературы, просмотра и копирования электронных версий изданий сотрудников университета (учеб. и метод. пособия, тексты лекций и т.д.) Для работы в «Личном кабинете» необходимо зайти на сайт Научной библиотеки ЯрГУ с любой точки, имеющей доступ в Internet, в пункт меню «Электронный каталог»; пройти процедуру авторизации, выбрав вкладку «Авторизация», и заполнить представленные поля информации.

2. Электронная библиотека учебных материалов ЯрГУ (http://www.lib.uniyar.ac.ru/opac/bk_cat_find.php) содержит более 2500 полных текстов учебных и учебно-методических материалов по основным изучаемым дисциплинам, изданных в университете. Доступ в сети университета, либо по логину/паролю.

3. Электронная картотека «Книгообеспеченность» (http://www.lib.uniyar.ac.ru/opac/bk_bookreq_find.php) раскрывает учебный фонд научной библиотеки ЯрГУ, предоставляет оперативную информацию о состоянии книгообеспеченности дисциплин основной и дополнительной литературой, а также цикла дисциплин и специальностей. Электронная картотека «Книгообеспеченность» доступна в сети университета и через Личный кабинет.