

Государственное автономное образовательное учреждение  
высшего образования Ленинградской области  
«Гатчинский государственный университет»

Утверждаю  
Проректор по образовательной  
деятельности и цифровой  
трансформации  
Е.В. Карпичев  
«19» декабря 2025 г.



## **РАБОЧАЯ ПРОГРАММА ПО ДИСЦИПЛИНЕ «ОСНОВЫ ПРОГРАММИРОВАНИЯ»**

Направление подготовки:

**44.03.05 Педагогическое образование (с двумя профилями подготовки)**  
(уровень бакалавриата)

Направленность (профиль) образовательной программы  
«Технология и организация производства»

Форма обучения  
очная

Гатчина  
2025

Рабочая программа по дисциплине «Основы программирования» разработана на основе федерального государственного образовательного стандарта высшего образования (далее ФГОС ВО) по направлению подготовки 44.03.05 Педагогическое образование (с двумя профилями подготовки) направленность (профиль) образовательной программы «Технология и организация производства»

Уровень: бакалавриат

Организация-разработчик: ГАОУ ВО ЛО «Гатчинский государственный университет»

Разработчик: преподаватель Иванова Ю.А.

Рассмотрена и одобрена на заседании профессионального и технологического образования «17» октября 2025 г. Протокол №2.

СОГЛАСОВАНО:

Руководитель ОП  / Талалай Г.С.

## СОДЕРЖАНИЕ

1. Пояснительная записка (цели и задачи) освоения дисциплины (модуля)	4
2. Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы.....	6
3. Место дисциплины в структуре образовательной программы .....	7
4. Объем дисциплины (модуля) в зачетных единицах с указанием количества академических или астрономических часов, выделенных на контактную работу обучающихся с преподавателем (по видам учебных занятий) и на самостоятельную работу обучающихся.....	9
5. Содержание дисциплины (модуля), структурированное по темам (разделам) с указанием отведенного на них количества академических или астрономических часов и видов учебных занятий .....	10
6. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине (модулю) .....	27
7. Фонд оценочных и методических материалов для проведения промежуточной аттестации обучающихся по дисциплине .....	28
8. Перечень основной, дополнительной учебной литературы, ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины (модуля).....	40
9. Методические указания для обучающихся по освоению дисциплины (модуля).....	41
10. Особенности освоения дисциплины для инвалидов и лиц с ограниченными возможностями здоровья.....	43
11. Перечень информационных технологий, профессиональных баз данных, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень программного обеспечения и информационных справочных систем.....	44
12. Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине (модулю) .....	44

## **1. Пояснительная записка (цели и задачи) освоения дисциплины (модуля)**

Курс «*Основы программирования*» занимает важное место при подготовке бакалавров по направлению 44.03.05 – Педагогическое образование (с двумя профилями подготовки). Это связано с тем, что дисциплина «*Основы программирования*» включена в структуру образовательной программы и относится к дисциплинам части формируемой участниками образовательных отношений. Она осваивается на 3 курсе, в 6 семестре. Изучение дисциплины «*Основы программирования*» — основа для прохождения студентами педагогической практики и подготовки к государственной итоговой аттестации.

Цель освоения дисциплины «*Основы программирования*» заключается: формирование у обучающихся систематизированных знаний и навыков в области программирования, ознакомление с различными парадигмами программирования. Рабочая программа учебной дисциплины направлена на воспитание и приобретение обучающимися теоретических знаний, необходимых для успешного освоения иных учебных дисциплин, составляющих профессиональный цикл основной образовательной программы.

*Задачами освоения дисциплины являются:*

- ознакомление с основными конструкциями языка программирования. Например, изучение синтаксиса, основных типов данных, операторов, структур управления (линейные, разветвляющиеся и циклические алгоритмы);
- изучение типов и структур данных. Освоение базовых структур данных, таких как массивы, строки, списки, а также других типов данных, используемых в программировании;
- формирование умений разработки, отладки и тестирования программ. Включает умение строить математические модели, составлять алгоритмы решения задач, записывать их на языках программирования высокого уровня, анализировать и исправлять синтаксические и семантические ошибки, составлять систему тестов для проверки корректности программы;
- ознакомление с различными парадигмами программирования. Например, изучение процедурного, объектно-ориентированного, функционального и визуального программирования;
- развитие навыков работы с файлами. Работа с файлами, включая их открытие, построчную обработку, закрытие и другие операции;
- формирование умения оценивать сложность алгоритмов и анализировать полученные результаты;
- развитие логического и алгоритмического мышления, умения планировать действия и структурировать информацию для решения поставленных задач.

При изучении данной дисциплины «*Основы программирования*» обучающийся должен знать:

- основные понятия, используемые в программировании;
- базовый синтаксис одного из языков программирования;
- основы структурного программирования;
- средства и возможности языков программирования высокого уровня;
- возможности современных информационных технологий поддержки технологий программирования;
- профессиональная лексика в области программирования;
- теоретические основы объектно-ориентированного анализа, проектирования и программирования (в зависимости от программы обучения);
- основные типы и структуры данных (списки, деревья, множества и т. п.), методы их обработки и способы реализации в инструментальных средах;
- основные алгоритмы решения основных классов задач;
- методы и технологии программирования в объектно-ориентированных инструментальных средах.

При изучении данной дисциплины «*Основы программирования*» обучающийся должен уметь:

- читать программный код, написанный на одном из языков программирования;
- создавать реализации базовых алгоритмов на одном из языков программирования;
- программно работать с данными, хранящимися в файлах;
- демонстрировать способность и готовность: создавать программы для решения некоторых математических задач; составлять алгоритмы и записывать их на одном из языков программирования.

При изучении данной дисциплины «*Основы программирования*» обучающийся должен владеть навыками:

- базовые компьютерные технологии и программные средства. Владение Технология и организация производствами обработки и отображения информации, навыками использования программных средств и работы в компьютерных сетях;
- методы составления блок-схем алгоритмов. Умение работать с алгоритмическими структурами и их визуальным представлением;
- навыки создания программ. Способность разрабатывать программы для решения типовых учебных или профессиональных задач с применением инструментальных средств поддержки технологий программирования;
- применение изученных средств и возможностей языков программирования. Умение использовать возможности языков программирования для создания программ, включая работу с данными, хранящимися в файлах;
- навыки разработки программного кода и его тестирования. Включают умение применять различные подходы к составлению алгоритмов и проектированию программного обеспечения;

- работа с научно-технической литературой и технической документацией по программному обеспечению.

## **2. Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы**

Процесс изучения дисциплины (модуля) направлен на формирование следующей компетенции (следующих компетенций):

<b>Компетенция (и)</b>	<b>Индикатор (ы)</b>
ПК-2 Способен использовать в профессиональной деятельности знания основных положений и концепций в области технологии, а также смежных метапредметных дисциплин	ПК-2.1 Знает особенности основных положений и концепций в области технологии, а также смежных метапредметных дисциплин
	ПК-2.2 Умеет толковать основные положения и концепции в области технологии, а также смежных метапредметных дисциплин
	ПК-2.3 Владеет навыками передачи общего содержания положений и концепций в области технологии, а также смежных метапредметных дисциплин

### 3. Место дисциплины в структуре образовательной программы

«Основы программирования» является дисциплиной обязательной части для подготовки студентов по направлению 44.03.05 Педагогическое образование (с двумя профилями подготовки).

Шифр компетенции	Предшествующие дисциплины (модули), практики учебного плана, в которых осваивается компетенция	Дисциплины (модули), практики учебного плана, в которых компетенция осваивается параллельно с изучаемой дисциплиной	Последующие дисциплины (модули), практики учебного плана, в которых осваивается компетенция
ПК-2	Материаловедение, Теоретическая механика, Инженерная графика, Практикум по обработке пищевых продуктов, Основы робототехники, Основы электротехники, Практикум по обработке текстильных материалов, Практикум по обработке конструкционных материалов, Теория и методика обучения робототехнике, Методы производственного обучения, Черчение, Физика, Компьютерная графика, Образовательная робототехника	Основы арт-дизайна кулинарной и кондитерской продукции, Основы автоматики и электроники	Основы технического творчества, Основы технического предпринимательства, Основы мехатроники, Теория и методика обучения технологии, Технологии современного производства, Предметно-методический модуль (профиль: Организация производства), Scratch-программирование, Технологии лазерной обработки материалов, Прототипирование и макетирование, Программирование на языке C++, Программирование на языке Python, Художественная обработка материалов, Декоративная отделка материалов, Современные технологии художественной обработки материалов, Современные технологии декоративной отделки материалов, Производственная практика (педагогическая)

			<p>практика),          Производственная          практика (преддипломная          практика), Подготовка к          сдаче и сдача          государственного          экзамена,          Выполнение и защита          выпускной          квалификационной          работы.</p>
--	--	--	--



**4. Объем дисциплины (модуля) в зачетных единицах с указанием количества академических или астрономических часов, выделенных на контактную работу обучающихся с преподавателем (по видам учебных занятий) и на самостоятельную работу обучающихся**

Общая трудоемкость освоения учебной дисциплины «*Основы программирования*» составляет 3 зачетных единицы или 108 академических часа.

Курс / семестр		3 курс / 6 семестр	Всего, часов
Общая трудоемкость (всего ак. часов / з.ед)		108 / 3	108 / 3
Контактная работа	Лекции	16	16
	Практические занятия	16	16
	Лабораторные занятия	16	16
Самостоятельная работа		42	42
Вид промежуточной аттестации	Зачет с оценкой	18	18

**5. Содержание дисциплины (модуля), структурированное по темам (разделам) с указанием отведенного на них количества академических или астрономических часов и видов учебных занятий**

№	Наименование раздела дисциплины (тема)	Трудоемкость				СРС	Содержание
		Всего	Контактная работа <sup>1</sup>				
			Л	ПЗ	ЛЗ		
6 семестр							
1.	Тема 1. Введение в програм- мирование. Базовые понятия и к- онструкции.	14	2	2	2	8	<i>Лекция:</i> <b>Введение в программирование:</b> 1. Классификация языков программирования. 2. Историческая справка и эволюция языков программирования. 3. Основные свойства языков программирования высокого уровня. 4. Знакомство со средой программирования (установка, настройка, интер- фейс). <b>Базовые понятия и конструкции:</b> 1. Основне понятия языков программирования: алфавит, синтаксис, семан- тика. 2. Типы данных: – простые типы (целые, вещественные, логические, символьные); – составные типы. 3. Переменные и константы: объявление, инициализация, область видимо- сти. 4. Операции и выражения: арифметические, логические, отношения. 5. Функции и процедуры стандартной библиотеки. 6. Организация ввода-вывода данных. 7. Оператор присваивания и его особенности. <i>Практическое занятие:</i>

<sup>1</sup> Л. – лекция. ПЗ – практическое занятие. ЛЗ – лабораторное занятие. СРС – самостоятельная работа студента

						<p><b>Начальный (знакомство с синтаксисом, вводом-выводом, простыми операциями):</b></p> <p><b>Калькулятор двух чисел.</b> Напишите программу, которая запрашивает у пользователя два числа и выводит результаты их сложения, вычитания, умножения и деления.  <i>Обрабатывает:</i> ввод данных (input()), вывод (print()), арифметические операции, преобразование типов (int(), float()).</p> <p><b>Конвертер единиц.</b> Создайте программу для перевода:          – километров в мили;          – градусов Цельсия в градусы Фаренгейта;          – рублей в доллары/евро (по фиксированному курсу).  <i>Обрабатывает:</i> формулы, переменные, ввод/вывод.</p> <p><b>Площадь и периметр.</b> Напишите программу, вычисляющую площадь и периметр прямоугольника по заданным пользователем длине и ширине.  <i>Обрабатывает:</i> базовые математические формулы, работу с переменными.</p> <p><b>Приветствие.</b> Программа запрашивает имя пользователя и выводит сообщение вида «Привет, [Имя]! Добро пожаловать в мир программирования!».  <i>Обрабатывает:</i> строковые операции, конкатенацию строк, input() и print().</p> <p><b>Средний (условные операторы, простые циклы):</b></p> <p><b>Проверка чётности.</b> Напишите программу, которая определяет, является ли введённое число чётным или нечётным.  <i>Обрабатывает:</i> оператор if-else, оператор остатка от деления (%).</p> <p><b>Калькулятор возраста.</b> Программа запрашивает год рождения пользователя и текущий год, затем выводит возраст и сообщает, совершеннолетний пользователь или нет (<math>\geq 18</math> лет).  <i>Обрабатывает:</i> условные конструкции, арифметику.</p> <p><b>Таблица умножения.</b> Выведите на экран таблицу умножения для числа, введённого пользователем (например, для числа 5: <math>5 \times 1 = 5</math>, <math>5 \times 2 = 10</math>, ..., <math>5 \times 10 = 50</math>).  <i>Обрабатывает:</i> цикл for с range().</p>
--	--	--	--	--	--	--

						<p><b>Сумма чисел.</b> Напишите программу, которая суммирует все целые числа от 1 до N, где N — число, введённое пользователем.</p> <p><i>Отрабатывает:</i> цикл for, накопление результата в переменной.</p> <p><b>Обратная строка.</b> Пользователь вводит строку, программа выводит её в обратном порядке.</p> <p><i>Отрабатывает:</i> работу со строками, срезы ([::-1]).</p> <p><b>Продвинутый (комбинация конструкций, работа с данными):</b></p> <p><b>Угадай число (упрощённая версия).</b> Программа «загадывает» случайное число от 1 до 10 (используйте random.randint(1, 10)). Пользователь пытается его угадать. После каждой попытки программа сообщает «больше» или «меньше».</p> <p><i>Отрабатывает:</i> import, генератор случайных чисел, цикл while, условные операторы.</p> <p><b>Анализ чисел.</b> Пользователь вводит несколько чисел (например, 5). Программа находит и выводит:</p> <ul style="list-style-type: none"> <li>– наибольшее число;</li> <li>– наименьшее число;</li> <li>– среднее арифметическое.</li> </ul> <p><i>Отрабатывает:</i> циклы, списки (или накопление в переменных), встроенные функции max(), min(), sum().</p> <p><b>Счётчик гласных.</b> Программа запрашивает строку и подсчитывает количество гласных букв (а, е, и, о, у) в ней (можно на русском: а, е, ё, и, о, у, ы, э, ю, я).</p> <p><i>Отрабатывает:</i> цикл for, проверку вхождения (in), работу со строками.</p> <p><b>Простые числа.</b> Напишите программу, которая проверяет, является ли введённое число простым.</p> <p><i>Отрабатывает:</i> вложенные циклы, условные операторы, оптимизацию (проверка до n).</p> <p><b>Генератор паролей (базовый).</b> Программа генерирует простой пароль заданной длины, состоящий из букв и цифр.</p> <p><i>Отрабатывает:</i> модуль random, строки, циклы, конкатенацию.</p> <p><b>Интерактивное меню.</b> Создайте простую программу с меню:</p>
--	--	--	--	--	--	---

							1. Калькулятор (задача 1). 2. Конвертер (задача 2). 3. Выход. Пользователь выбирает пункт, выполняет действие, возвращается в меню. <i>Отрабатывает:</i> структуру while True с break, вложенные условные операторы. <i>Лабораторные работы:</i> Знакомство с Python. Линейные программы. <i>Самостоятельная работа:</i> подготовиться к устному опросу, конспект, доклад, реферат, подготовка к зачету с оценкой.
2.	Тема 2. Алгоритмические конструкции.	18	4	2	4	8	<i>Лекция:</i> <b>Алгоритмические конструкции:</b> 1. Программирование линейных алгоритмов. 2. Операторы ветвления: – условный оператор (if); – цепочка вложенных условных операторов; – реализация множественного выбора (switch / case). 3. Программирование разветвляющихся алгоритмов. 4. Операторы повторения: – цикл с предусловием (while); – цикл с постусловием (do-while); – цикл с параметром (for). 5. Программирование циклических алгоритмов. 6. Вложенные циклы и их применение. 7. Отладка программ: поиск и устранение ошибок, использование отладчика. <i>Практическое занятие:</i> <b>1. Конструкция «следование»:</b> 1. Составить программу вычисления площади и периметра прямоугольника по заданным сторонам.

						<p>2. Разработать алгоритм перевода температуры из градусов Цельсия в градусы Фаренгейта и обратно.</p> <p>3. Написать программу расчёта параметров прямоугольного треугольника (гипотенуза, углы) по двум катетам.</p> <p>4. Реализовать алгоритм вычисления объёма и площади поверхности цилиндра по радиусу основания и высоте.</p> <p>5. Составить программу конвертации единиц измерения (метры → сантиметры → миллиметры).</p> <p><b>2. Конструкция «ветвление»:</b></p> <p>1. Разработать программу определения, является ли введённое число чётным/нечётным.</p> <p>2. Создать алгоритм проверки, может ли существовать треугольник с заданными сторонами.</p> <p>3. Написать программу решения квадратного уравнения <math>ax^2+bx+c=0</math> с учётом всех случаев (2 корня, 1 корень, нет корней).</p> <p>4. Реализовать калькулятор скидок: в зависимости от суммы покупки применять разные процентные ставки.</p> <p>5. Составить алгоритм определения принадлежности точки с координатами <math>(x,y)</math> заданной области на плоскости (например, внутри круга или за его пределами).</p> <p><b>3. Циклы с предусловием и постусловием («пока», «до»):</b></p> <p>1. Написать программу вычисления суммы первых <math>N</math> натуральных чисел с использованием цикла <code>while</code>.</p> <p>2. Реализовать алгоритм нахождения наибольшего общего делителя (НОД) двух чисел методом Евклида.</p> <p>3. Создать программу, которая определяет количество цифр в введённом целом числе.</p> <p>4. Разработать алгоритм вычисления факториала числа с использованием цикла с постусловием.</p> <p>5. Написать программу поиска первого числа Фибоначчи, превышающего заданное значение.</p> <p><b>4. Цикл со счётчиком («для»):</b></p>
--	--	--	--	--	--	---

							<ol style="list-style-type: none"> <li>1. Составить программу вывода таблицы умножения для заданного числа <math>a</math>.</li> <li>2. Реализовать алгоритм вычисления суммы ряда <math>1+2!+3!+\dots+N!</math>.</li> <li>3. Написать программу построения таблицы значений функции <math>y=f(x)</math> на заданном интервале с фиксированным шагом.</li> <li>4. Создать алгоритм вычисления суммы квадратов первых <math>N</math> натуральных чисел.</li> <li>5. Разработать программу подсчёта количества положительных, отрицательных и нулевых элементов в последовательности из <math>N</math> чисел.</li> </ol> <p><b>5. Комбинированные задачи (несколько конструкций):</b></p> <ol style="list-style-type: none"> <li>1. Написать программу поиска всех простых чисел в заданном диапазоне (использовать циклы и ветвления).</li> <li>2. Реализовать алгоритм сортировки одномерного массива методом пузырька (циклы + ветвления).</li> <li>3. Создать программу вычисления приближённого значения числа <math>\pi</math> методом Монте-Карло (циклы, ветвления, генерация случайных чисел).</li> <li>4. Разработать калькулятор с меню выбора операций (сложение, вычитание, умножение, деление), работающий до тех пор, пока пользователь не выберет выход.</li> <li>5. Написать программу анализа последовательности чисел: найти максимум, минимум, среднее арифметическое (использовать циклы, ветвления и накопление результатов).</li> </ol> <p><b>6. Работа с блок-схемами:</b></p> <ol style="list-style-type: none"> <li>1. Построить блок-схему алгоритма вычисления площади круга по заданному радиусу.</li> <li>2. Составить блок-схему для алгоритма проверки введённого пароля (с ограничением числа попыток).</li> <li>3. Разработать блок-схему вычисления суммы арифметической прогрессии с использованием цикла со счётчиком.</li> <li>4. Построить блок-схему алгоритма поиска максимального элемента в массиве.</li> </ol>
--	--	--	--	--	--	--	---

							<p>5. Создать блок-схему программы, которая определяет тип треугольника (остроугольный, прямоугольный, тупоугольный) по трём сторонам.</p> <p><i>Лабораторные работы:</i></p> <p>Разветвляющиеся вычислительные процессы.</p> <p><i>Самостоятельная работа:</i> подготовиться к устному опросу, конспект, доклад, реферат, подготовка к зачету с оценкой.</p>
3.	<p>Тема 3.</p> <p>Структуры данных.</p> <p>Модульное программирование.</p>	20	4	4	4	8	<p><i>Лекция:</i></p> <p><b>Структуры данных:</b></p> <ol style="list-style-type: none"> <li>Одномерные массивы: <ul style="list-style-type: none"> <li>представление и объявление;</li> <li>инициализация и заполнение;</li> <li>алгоритмы обработки (поиск, сортировка, суммирование и т.д.).</li> </ul> </li> <li>Двумерные массивы: <ul style="list-style-type: none"> <li>представление и объявление;</li> <li>работа с матрицами;</li> <li>алгоритмы обработки матриц.</li> </ul> </li> <li>Множества: <ul style="list-style-type: none"> <li>определение и представление;</li> <li>операции и функции обработки;</li> <li>использование в программах.</li> </ul> </li> <li>Неоднородные структуры данных: <ul style="list-style-type: none"> <li>записи (структуры): определение и особенности;</li> <li>работа с полями записей.</li> </ul> </li> <li>Строки: <ul style="list-style-type: none"> <li>представление и обработка;</li> <li>основные операции со строками;</li> <li>алгоритмы обработки строк.</li> </ul> </li> </ol> <p><b>Модульное программирование:</b></p> <ol style="list-style-type: none"> <li>Создание подпрограмм: функции и процедуры.</li> <li>Параметры подпрограмм: <ul style="list-style-type: none"> <li>параметры-значения;</li> <li>параметры-переменные;</li> </ul> </li> </ol>



						<ul style="list-style-type: none"> <li>– передача массивов в подпрограммы.</li> </ul> <p>3. Область видимости переменных: локальные и глобальные переменные.</p> <p>4. Рекурсия: понятие, примеры рекурсивных алгоритмов.</p> <p><i>Практическое занятие:</i></p> <p><b>Базовый уровень</b></p> <p>1. <b>Реализация и работа с массивами:</b></p> <ul style="list-style-type: none"> <li>– создать модуль для работы с одномерным массивом целых чисел (функции добавления, удаления, поиска элемента);</li> <li>– написать программу, использующую этот модуль для сортировки массива методом «пузырька».</li> </ul> <p>2. <b>Работа со структурами (структуры/записи):</b></p> <ul style="list-style-type: none"> <li>– определить структуру Student с полями: имя, фамилия, номер группы, оценки;</li> <li>– реализовать модуль с функциями создания, редактирования и вывода данных о студентах;</li> <li>– составить программу, которая заполняет массив из 5 студентов и выводит отличников.</li> </ul> <p>3. <b>Статические структуры данных:</b></p> <ul style="list-style-type: none"> <li>– разработать модуль для работы со стеком на основе массива (операции push, pop, peek, isEmpty);</li> <li>– использовать модуль для проверки корректности расстановки скобок в арифметическом выражении.</li> </ul> <p>4. <b>Модули для обработки строк:</b></p> <ul style="list-style-type: none"> <li>– создать модуль с функциями: подсчёт слов в строке, замена подстроки, проверка на палиндром;</li> <li>– написать тестовую программу, демонстрирующую работу всех функций модуля.</li> </ul> <p>5. <b>Файловые структуры:</b></p> <ul style="list-style-type: none"> <li>– спроектировать модуль для работы с файлом записей (добавление, поиск, удаление записей);</li> <li>– применить модуль для создания базы данных книг (автор, название, год издания).</li> </ul>
--	--	--	--	--	--	--

						<p><b>Средний уровень</b></p> <p><b>1. Динамические массивы:</b></p> <ul style="list-style-type: none"> <li>– реализовать модуль для динамического массива с автоматическим изменением размера;</li> <li>– включить функции: добавление в конец/начало, вставка в позицию, удаление по индексу;</li> <li>– протестировать модуль на примере хранения и обработки последовательности чисел.</li> </ul> <p><b>2. Очередь на основе массива:</b></p> <ul style="list-style-type: none"> <li>– создать модуль очереди с циклическим буфером (операции enqueue, dequeue, front, isFull);</li> <li>– использовать очередь для моделирования системы обработки запросов (например, печати документов).</li> </ul> <p><b>3. Ассоциативный массив (хеш-таблица):</b></p> <ul style="list-style-type: none"> <li>– разработать простой модуль хеш-таблицы с разрешением коллизий методом цепочек;</li> <li>– функции: вставка, поиск, удаление по ключу (ключ — строка, значение — целое число);</li> <li>– продемонстрировать работу на примере словаря частот слов в тексте.</li> </ul> <p><b>4. Двумерные массивы и матрицы:</b></p> <ul style="list-style-type: none"> <li>– создать модуль для операций с матрицами (сложение, умножение, транспонирование);</li> <li>– написать программу решения СЛАУ методом Гаусса с использованием модуля.</li> </ul> <p><b>5.</b></p> <p><b>Модульное программирование с разделением интерфейса и реализации:</b></p> <ul style="list-style-type: none"> <li>– разделить модуль работы с динамическим списком на заголовочный файл (.h) и файл реализации (.c/.cpp);</li> <li>– обеспечить инкапсуляцию данных (скрытые поля структуры списка);</li> <li>– предоставить набор функций для управления списком без доступа к внутренней структуре.</li> </ul>
--	--	--	--	--	--	--

						<p><b>Продвинутый уровень</b></p> <p><b>1. Бинарное дерево поиска:</b></p> <ul style="list-style-type: none"> <li>– реализовать модуль бинарного дерева с операциями вставки, поиска, удаления, обхода (in-order, pre-order);</li> <li>– добавить функцию балансировки дерева (AVL-дерево или красно-чёрное дерево);</li> <li>– применить дерево для построения индекса слов в текстовом файле.</li> </ul> <p><b>2. Графы:</b></p> <ul style="list-style-type: none"> <li>– создать модуль графа на основе списков смежности (добавление вершин/рёбер, поиск в глубину/ширину);</li> <li>– решить задачу поиска кратчайшего пути (алгоритм Дейкстры) с использованием модуля;</li> <li>– визуализировать граф и путь в консольном режиме.</li> </ul> <p><b>3. Многомодульная система учёта:</b></p> <ul style="list-style-type: none"> <li>– спроектировать систему из 3–4 модулей: работа с данными (структуры), хранение (файлы/БД), интерфейс (меню), логика обработки;</li> <li>– пример: система учёта товаров на складе (добавление, поиск по категории, формирование отчётов);</li> <li>– обеспечить независимую компиляцию и подключение модулей.</li> </ul> <p><b>4. Шаблоны (для языков C++/Java):</b></p> <ul style="list-style-type: none"> <li>– реализовать параметризованный модуль стека/очереди (шаблонный класс);</li> <li>– протестировать работу с разными типами данных (int, string, пользовательская структура);</li> <li>– сравнить с нешаблонной реализацией по гибкости и объёму кода.</li> </ul> <p><b>5. Интеграция с GUI (по желанию):</b></p> <ul style="list-style-type: none"> <li>– расширить один из предыдущих модулей (например, стек или очередь) для работы с графическим интерфейсом (Lazarus/Qt);</li> <li>– создать форму с кнопками для выполнения операций и отображением состояния структуры в реальном времени.</li> </ul> <p><i>Лабораторные работы:</i></p>
--	--	--	--	--	--	--

							<p>Организация циклов».</p> <p><i>Самостоятельная работа:</i> подготовиться к устному опросу, конспект, доклад, реферат, подготовка к зачету с оценкой.</p>
4.	<p>Тема 4.</p> <p>Работа с файлами.</p>	19	2	4	4	9	<p><i>Лекция:</i></p> <p><b>Работа с файлами:</b></p> <ol style="list-style-type: none"> <li>1. Основные термины и понятия: файл, поток, дескриптор.</li> <li>2. Типы файлов в программировании: <ul style="list-style-type: none"> <li>– текстовые файлы;</li> <li>– бинарные (типизированные) файлы.</li> </ul> </li> <li>3. Основные операции с файлами: <ul style="list-style-type: none"> <li>– открытие и закрытие файлов;</li> <li>– чтение и запись данных;</li> <li>– позиционирование в файле.</li> </ul> </li> <li>4. Стандартные функции для работы с файлами.</li> <li>5. Практические примеры работы с файлами разного типа.</li> </ol> <p><i>Практическое занятие:</i></p> <p><b>Базовый уровень</b></p> <ol style="list-style-type: none"> <li>1. <b>Чтение и вывод содержимого текстового файла</b>  Написать программу, которая: <ul style="list-style-type: none"> <li>– запрашивает у пользователя путь к текстовому файлу;</li> <li>– считывает содержимое файла;</li> <li>– выводит его на экран.</li> </ul> Обработать возможные ошибки (файл не найден, нет прав доступа).</li> <li>2. <b>Копирование текстового файла</b>  Создать программу, которая копирует содержимое одного текстового файла в другой. Предусмотреть проверку существования исходного файла.</li> <li>3. <b>Подсчёт строк, слов и символов</b>  Разработать программу, которая анализирует текстовый файл и выводит: <ul style="list-style-type: none"> <li>– количество строк;</li> </ul> </li> </ol>

						<ul style="list-style-type: none"> <li>– количество слов;</li> <li>– общее количество символов (с пробелами и без).</li> </ul> <p><b>4. Поиск строки в файле</b> Написать приложение, которое ищет заданную строку в текстовом файле и выводит номера строк, где она встречается. Реализовать опцию учёта/неучёта регистра.</p> <p><b>5. Создание лог-файла</b> Реализовать программу, которая записывает в лог-файл текущую дату и время при каждом запуске. Каждая запись должна быть на новой строке.</p> <p><b>Средний уровень</b></p> <p><b>1.Обработка числовых данных из файла</b> Программа должна:</p> <ul style="list-style-type: none"> <li>– считывать из файла последовательность чисел (каждое число на отдельной строке);</li> <li>– вычислять среднее арифметическое, минимум и максимум;</li> <li>– записывать результаты в отдельный файл.</li> </ul> <p><b>2.Объединение нескольких файлов</b> Написать скрипт, который объединяет содержимое нескольких текстовых файлов (имена файлов передаются как аргументы) в один итоговый файл. Добавить возможность сортировки строк перед объединением.</p> <p><b>3.Замена текста в файле</b> Создать утилиту, которая заменяет все вхождения одной строки на другую в указанном файле. Предусмотреть создание резервной копии исходного файла перед изменениями.</p> <p><b>4.Анализ CSV-файла</b> Обработать CSV-файл с данными (например, «имя,возраст,город»). Программа должна:</p> <ul style="list-style-type: none"> <li>– выводить записи, где возраст больше заданного значения;</li> <li>– сохранять отфильтрованные данные в новый файл.</li> </ul> <p><b>5.Шифрование/дешифрование файла</b> Реализовать простой алгоритм шифрования (например, сдвиг символов</p>
--	--	--	--	--	--	--

						<p>по алфавиту) для содержимого файла. Программа должна иметь два режима: шифрование и дешифрование.</p> <p><b>Продвинутый уровень</b></p> <p><b>1.Индексация файлов в каталоге</b>  Написать программу, которая:</p> <ul style="list-style-type: none"> <li>– обходит указанный каталог и все его подкаталоги;</li> <li>– создаёт индексный файл со списком всех найденных файлов (путь, размер, дата изменения);</li> <li>– позволяет искать файлы по имени в созданном индексе.</li> </ul> <p><b>2.Архивация файлов</b>  Реализовать упрощённую версию архиватора:</p> <ul style="list-style-type: none"> <li>– пользователь указывает файлы для архивации;</li> <li>– программа объединяет их в один бинарный файл с заголовком (метаданные о файлах);</li> <li>– добавить функцию распаковки архива.</li> </ul> <p><b>3.Обработка JSON-файла</b>  Создать приложение для работы с конфигурационным файлом в формате JSON:</p> <ul style="list-style-type: none"> <li>– чтение настроек из файла;</li> <li>– изменение отдельных параметров через интерфейс командной строки;</li> <li>– сохранение обновлённого файла.</li> </ul> <p><b>4.Мониторинг изменений в файле</b>  Разработать программу, которая отслеживает изменения в указанном текстовом файле (добавление новых строк) и выводит их в реальном времени. Использовать механизм опроса файла или системные уведомления (если поддерживается языком).</p> <p><b>5.Пакетная обработка файлов</b>  Написать скрипт для массовой обработки файлов в каталоге:</p> <ul style="list-style-type: none"> <li>– переименование файлов по шаблону;</li> <li>– конвертация кодировки текста;</li> <li>– добавление префикса/суффикса к содержимому.</li> </ul> <p>Реализовать логирование выполненных операций.</p>
--	--	--	--	--	--	--

							<p><i>Лабораторные работы:</i>  Одномерные массивы.  Функции в программировании.  <i>Самостоятельная работа:</i> подготовиться к устному опросу, конспект, доклад, реферат, подготовка к зачету с оценкой.</p>
5.	<p>Тема 5.  Основы парадигм программирования.  Практические навыки и инструменты.</p>	19	4	4	2	9	<p><i>Лекция:</i>  <b>Основы парадигм программирования:</b>  1. Структурное программирование: принципы и подходы.  2. Основы объектно-ориентированного программирования (ООП):  – классы и объекты;  – инкапсуляция, наследование, полиморфизм (базовые понятия).  3. Сравнение подходов: преимущества и недостатки.  <b>Практические навыки и инструменты:</b>  1. Инструментальные средства разработчика: среды разработки (IDE), компиляторы, отладчики.  2. Составление блок-схем алгоритмов.  3. Тестирование и отладка программ: методы и приёмы.  4. Оформление кода: стиль, комментарии, читаемость.  <i>Практическое занятие:</i>  <b>Процедурное программирование</b>  1. Написать программу, которая запрашивает у пользователя два числа и выводит их сумму, разность, произведение и частное (с округлением до трёх знаков после запятой).  2. Реализовать программу проверки, можно ли из трёх заданных отрезков построить треугольник (по длинам сторон).  3. Написать программу для подсчёта количества букв, цифр, знаков препинания и прочих символов во введённой строке.  4. Реализовать алгоритм сортировки массива методом пузырька или выбором — с выводом промежуточных шагов.  5. Написать программу вычисления факториала числа с использованием цикла и рекурсии; сравнить подходы.  <b>Объектно-ориентированное программирование (ООП)</b></p>

						<ol style="list-style-type: none"> <li>1. Создать класс Rectangle с полями ширины и высоты, методами расчёта площади и периметра. Добавить конструктор и метод вывода информации о прямоугольнике.</li> <li>2. Разработать иерархию классов Animal → Dog и Cat: реализовать полиморфизм через переопределение метода makeSound().</li> <li>3. Реализовать класс BankAccount с инкапсуляцией: приватное поле баланса, методы deposit(), withdraw(), getBalance(). Добавить проверку на отрицательный баланс.</li> <li>4. Создать абстрактный класс Shape с абстрактным методом area(). Реализовать подклассы Circle, Rectangle, Triangle, переопределив метод расчёта площади.</li> <li>5. Разработать класс Student с полями имени, списка оценок и методом расчёта среднего балла. Использовать списки/коллекции для хранения оценок.</li> </ol> <p><b>Функциональное программирование</b></p> <ol style="list-style-type: none"> <li>1. Написать функцию высшего порядка, которая принимает функцию и список чисел, применяя функцию к каждому элементу (аналог map).</li> <li>2. Реализовать функцию фильтрации списка чисел по условию (аналог filter) — например, отобрать чётные числа или числа больше заданного значения.</li> <li>3. Использовать лямбда-выражения для сортировки списка строк по длине или списка чисел по убыванию.</li> <li>4. Написать рекурсивную функцию вычисления чисел Фибоначчи и сравнить её эффективность с итеративным вариантом.</li> <li>5. Применить функции reduce (или аналоги) для вычисления произведения всех элементов списка.</li> </ol> <p><b>Логическое программирование (базовый уровень)</b></p> <ol style="list-style-type: none"> <li>1. Описать факты и правила на псевдоязыке (или Prolog) для простой базы знаний «Семья»: определить отношения родитель, брат, сестра и вывести всех дедушек.</li> </ol>
--	--	--	--	--	--	--



						<p>2. Создать набор правил для решения головоломки «Кто держит рыбу?» (задача Эйнштейна) с использованием логических ограничений.</p> <p><b>Сравнительный анализ и переход между парадигмами</b></p> <p>1. Решить одну задачу (например, подсчёт частоты слов в тексте) тремя способами:</p> <ul style="list-style-type: none"> <li>– процедурно (циклы, массивы);</li> <li>– ООП (класс WordCounter с методами);</li> <li>– функционально (функции map, filter, reduce). Сравнить код и эффективность.</li> </ul> <p>2. Переписать фрагмент кода из процедурного стиля в ООП-стиль: например, программу управления библиотекой (книги, выдача, возврат) с использованием классов.</p> <p>3. Проанализировать готовый код (предоставленный преподавателем) и определить, какие парадигмы в нём используются. Указать плюсы и минусы выбранного подхода.</p> <p><b>Инструменты и отладка</b></p> <p>1. Настроить среду разработки (IDE: PyCharm, VS Code, Eclipse и т.д.) для выбранного языка. Написать и отладить программу с использованием точек останова и пошагового выполнения.</p> <p>2. Написать юнит-тесты (например, с использованием unittest в Python или JUnit в Java) для функций из предыдущих заданий (сортировка, расчёт площади).</p> <p>3. Использовать систему контроля версий Git: создать репозиторий, зафиксировать изменения кода по мере выполнения задания, сделать ветку для экспериментальной функции.</p> <p>4. Проанализировать производительность двух реализаций алгоритма (например, рекурсивной и итеративной) с помощью профилировщика (например, cProfile в Python).</p> <p>5. Исправить ошибки в предоставленном фрагменте кода (синтаксические, логические, проблемы производительности), используя отладчик и статический анализатор (например, pylint).</p> <p><i>Лабораторные работы:</i></p>
--	--	--	--	--	--	--

							Работа с файлами. <i>Самостоятельная работа: подготовиться к устному опросу, конспект, доклад, реферат, подготовка к зачету с оценкой.</i>
<b>Зачет с оценкой</b>	<b>18</b>						-
<b>Итого</b>	<b>108</b>	<b>16</b>	<b>16</b>	<b>16</b>	<b>42</b>		-

## 6. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине (модулю)

№	Виды самостоятельной работы	Трудоемкость, ак. часы	Форма контроля
1.	Проработка теоретического материала по конспектам лекций, рекомендованной литературе, дополнительным источникам информации	21	Консультация преподавателя, устное собеседование
2.	Подготовка к практическим занятиям: поиск необходимой информации, обработка информации, написание доклада, подготовка к выступлению (дискуссии)	21	Выступление с докладом, презентация, ответы на дискуссионные вопросы
3.	Подготовка к зачету с оценкой	17,75	Устное собеседование

Для самостоятельной работы по дисциплине (модулю) обучающиеся используют следующее учебно-методическое обеспечение:

1. Карпович Е. Е. «Основы алгоритмизации и программирования». Учебное пособие предназначено для студентов технических специальностей вузов. В нём описаны основные понятия информатики, необходимые для изучения основ алгоритмизации и программирования. Рассмотрена методология структурного проектирования алгоритмов и способы их представления. Представлено описание синтаксиса и семантики конструкций языка программирования C/C++. Приведены многочисленные примеры программ, иллюстрирующие возможности этого языка, 2024.
2. Фонд оценочных и методических материалов по дисциплине «Основы программирования».

## **7. Фонд оценочных и методических материалов для проведения промежуточной аттестации обучающихся по дисциплине**

### **Темы конспекта**

#### **Введение в программирование**

5. Классификация языков программирования.
6. Историческая справка и эволюция языков программирования.
7. Основные свойства языков программирования высокого уровня.
8. Знакомство со средой программирования (установка, настройка, интерфейс).

#### **Базовые понятия и конструкции**

8. Основные понятия языков программирования: алфавит, синтаксис, семантика.
9. Типы данных:
  - простые типы (целые, вещественные, логические, символьные);
  - составные типы.
10. Переменные и константы: объявление, инициализация, область видимости.
11. Операции и выражения: арифметические, логические, отношения.
12. Функции и процедуры стандартной библиотеки.
13. Организация ввода-вывода данных.
14. Оператор присваивания и его особенности.

#### **Алгоритмические конструкции**

8. Программирование линейных алгоритмов.
9. Операторы ветвления:
  - условный оператор (if);
  - цепочка вложенных условных операторов;
  - реализация множественного выбора (switch / case).
10. Программирование разветвляющихся алгоритмов.
11. Операторы повторения:
  - цикл с предусловием (while);
  - цикл с постусловием (do-while);
  - цикл с параметром (for).
12. Программирование циклических алгоритмов.
13. Вложенные циклы и их применение.
14. Отладка программ: поиск и устранение ошибок, использование отладчика.

#### **Структуры данных**

6. Одномерные массивы:
  - представление и объявление;
  - инициализация и заполнение;
  - алгоритмы обработки (поиск, сортировка, суммирование и т.д.).
7. Двумерные массивы:
  - представление и объявление;

- работа с матрицами;
- алгоритмы обработки матриц.
- 8. Множества:
  - определение и представление;
  - операции и функции обработки;
  - использование в программах.
- 9. Неоднородные структуры данных:
  - записи (структуры): определение и особенности;
  - работа с полями записей.
- 10. Строки:
  - представление и обработка;
  - основные операции со строками;
  - алгоритмы обработки строк.

### **Модульное программирование**

- 5. Создание подпрограмм: функции и процедуры.
- 6. Параметры подпрограмм:
  - параметры-значения;
  - параметры-переменные;
  - передача массивов в подпрограммы.
- 7. Область видимости переменных: локальные и глобальные переменные.
- 8. Рекурсия: понятие, примеры рекурсивных алгоритмов.

### **Работа с файлами**

- 6. Основные термины и понятия: файл, поток, дескриптор.
- 7. Типы файлов в программировании:
  - текстовые файлы;
  - бинарные (типизированные) файлы.
- 8. Основные операции с файлами:
  - открытие и закрытие файлов;
  - чтение и запись данных;
  - позиционирование в файле.
- 9. Стандартные функции для работы с файлами.
- 10. Практические примеры работы с файлами разного типа.

### **Основы парадигм программирования**

- 4. Структурное программирование: принципы и подходы.
- 5. Основы объектно-ориентированного программирования (ООП):
  - классы и объекты;
  - инкапсуляция, наследование, полиморфизм (базовые понятия).
- 6. Сравнение подходов: преимущества и недостатки.

### **Практические навыки и инструменты**

- 5. Инструментальные средства разработчика: среды разработки (IDE), компиляторы, отладчики.
- 6. Составление блок-схем алгоритмов.
- 7. Тестирование и отладка программ: методы и приёмы.
- 8. Оформление кода: стиль, комментарии, читаемость.

## **Требования к конспекту**

Написание конспекта представляет собой деятельность студента по созданию обзора информации, содержащейся в объекте конспектирования, в более краткой форме. В конспекте должны быть отражены основные принципиальные положения источника, то новое, что внес его автор, основные методологические положения работы, аргументы, этапы доказательства и выводы.

## **Примерная тематика докладов, рефератов:**

### **История и основы программирования**

1. История развития языков программирования: от машинных кодов до современных языков.
2. Сравнение языков программирования низкого и высокого уровня: особенности и сферы применения.
3. Алгоритм и его основные свойства. Способы записи алгоритмов.
4. Роль и значение стандартов кодирования в программировании.
5. Эволюция парадигм программирования: процедурное, объектно-ориентированное, функциональное.

### **Основы синтаксиса и структуры программ**

6. Основные типы данных в языках программирования (на примере C/C++ и Python).
7. Переменные и константы: правила именования, область видимости, время жизни.
8. Операторы языка программирования: арифметические, логические, сравнения, присваивания.
9. Консольный ввод-вывод данных: организация и особенности реализации.
10. Построение линейных, разветвляющихся и циклических алгоритмов.

### **Процурное и модульное программирование**

11. Функции и процедуры: создание, описание, реализация и вызов.
12. Параметры функций: неизменяемые и изменяемые параметры, передача по значению и по ссылке.
13. Рекурсия: принципы работы, примеры использования и ограничения.
14. Модульное программирование: преимущества разделения кода на модули и функции.
15. Отладка и тестирование отдельных модулей программы.

### **Работа с данными и структурами**

16. Строки как массивы символов: посимвольная и словесная обработка строк.
17. Стандартные функции для работы со строками и их применение.
18. Массивы: одномерные и многомерные, статические и динамические массивы.
19. Файловый ввод-вывод: работа с текстовыми и двоичными файлами.
20. Обработка ошибок при работе с файлами: выявление и обход ошибок формата.

### **Сложные структуры данных**

21. Составные типы данных: записи (структуры) и множества, их описание и использование.
22. Указатели и их роль в работе с динамическими структурами данных.
23. Стек и очередь: реализация и основные операции.
24. Однонаправленные и двунаправленные списки: создание и использование.
25. Бинарные деревья: построение, обход и применение в программировании.

### **Практические аспекты и современные технологии**

26. Основы отладки программ: методы поиска и устранения ошибок.
27. Основы тестирования программного обеспечения: виды и методы тестирования.
28. Основы документирования кода: стандарты и инструменты.
29. Введение в системы контроля версий (на примере Git): базовые команды и принципы работы.
30. Основы разработки пользовательского интерфейса в консольных приложениях.

### **Требования к докладу**

Доклад – средство, позволяющее проводить самостоятельный поиск материалов по заданной теме, реферировать и анализировать их, и доносить полученную информацию до окружающих. Доклад готовится по одной из проблем, находящихся в пределах обсуждаемой темы. Студент должен показать, что известно по этому поводу в науке, какие вопросы еще не освещены. Одним из условий, обеспечивающих успех практических занятий, является совокупность определенных конкретных требований к докладам студентов. Эти требования должны быть достаточно четкими и в то же время не настолько регламентированными, чтобы сковывать творческую мысль, насаждать схематизм. Перечень требований к выступлению студента:

- связь выступления с предшествующей темой или вопросом;
- раскрытие сущности проблемы;
- методологическое значение для научной, профессиональной и практической деятельности.

Важнейшие требования к выступлениям студентов – самостоятельность в подборе фактического материала и аналитическом отношении к нему, умение рассматривать примеры и факты во взаимосвязи и взаимообусловленности, отбирать наиболее существенные из них. Приводимые студентом примеры и факты должны быть существенными, по возможности перекликаться с программой подготовки. Примеры из области наук, близких к программе подготовки студента, из сферы познания. Выступление студента должно соответствовать требованиям логики. Четкое вычленение излагаемой проблемы, ее точная формулировка, неукоснительная последовательность аргументации именно данной проблемы, без неоправданных отступлений от нее в процессе обоснования, безусловная доказательность,

непротиворечивость и полнота аргументации, правильное и содержательное использование понятий и терминов.

## **Требования к реферату**

### **Структура реферата**

Обязательные разделы (в строгой последовательности):

1. **Титульный лист** — первая страница с ключевыми данными:
  - полное название учебного заведения;
  - факультет, направление подготовки, курс;
  - вид работы («Реферат», выделяется жирным);
  - тема работы;
  - Ф. И. О. студента;
  - группа/курс;
  - Ф. И. О. научного руководителя/преподавателя;
  - город и год написания (в нижней части страницы).
2. **Содержание (оглавление)** — размещается после титульного листа:
  - заголовок «Содержание» по центру, прописными буквами;
  - перечисление всех разделов и подразделов с указанием страниц;
  - автоматическое форматирование нумерации;
  - выравнивание по ширине.
3. **Введение** (объёмом до 1 страницы):
  - актуальность темы (обоснование выбора и значимости);
  - цель работы (чётко сформулированная задача);
  - задачи (конкретные действия для достижения цели);
  - структура работы (краткий перечень разделов).
4. **Основная часть** (2–4 раздела):
  - каждый раздел посвящён отдельному аспекту темы и имеет собственное название;
  - ссылки на авторитетные источники (учебники, научные статьи и т.д.);
  - допустимо использование схем, таблиц, графиков;
  - краткие выводы в конце каждого раздела;
  - нумерация разделов — арабскими цифрами (1, 2, 3...), подразделов — с внутренней нумерацией (1.1, 1.2 и т.д.).
5. **Заключение** (1–2 страницы):
  - выводы по каждой поставленной задаче;
  - общий итог работы;
  - анализ достижения цели;
  - оценка значимости темы и личного вклада;
  - рекомендации для дальнейшего изучения (при необходимости).
6. **Список литературы** (оформляется по ГОСТу):
  - учебники, научные статьи, энциклопедии, справочники, официальные сайты, статистические сборники, документы;
  - заголовок «Список литературы» — жирным шрифтом, по центру;
  - источники нумеруются по алфавиту или по мере появления в тексте;



- отступ слева — 1,25 см, выравнивание — по левому краю;
- между записями — пустая строка.

#### 7. **Приложения** (если есть) — дополнительные материалы:

- таблицы, схемы, иллюстрации, фотоматериалы;
- на все приложения в основной части должны быть ссылки;
- номер приложения размещают в правом верхнем углу над заголовком после слова «Приложение».

#### **Технические требования к оформлению**

- **Формат страницы:** А4.
- **Шрифт:** Times New Roman, размер 14.
- **Межстрочный интервал:** 1,5.
- **Поля:**

левое — 3 см;

правое — 1 см;

верхнее и нижнее — по 2 см.

- **Абзацный отступ:** 1,25 см.
- **Выравнивание текста:** по ширине.
- **Нумерация страниц:** снизу, по центру (титальный лист не нумеруется, но считается первой страницей).
- **Формат файла:** .docx или .pdf.

**Объём:** 10–20 страниц (зависит от уровня подготовки и глубины темы).

#### **Дополнительные рекомендации:**

1. Используйте шаблоны из методических рекомендаций кафедры или сайта университета – они учитывают актуальные требования.
2. Проверяйте **идентичность заголовков** в содержании и в тексте работы.
- 3.

Следите за **грамотностью** и стилем изложения: текст должен быть лаконичным, чётким, без избыточных описаний и разговорных оборотов.

4.

При использовании **иллюстративного материала** (таблиц, графиков) обязательно подписывайте их и делайте ссылки в тексте.

5. Перед сдачей проверьте:

- сквозную нумерацию страниц;
- наличие всех обязательных разделов;
- корректность ссылок на источники и приложения;
- соответствие оформления ГОСТ и требованиям учебного заведения.

#### **Вопросы для самостоятельного изучения:**

##### **Базовые понятия и основы алгоритмизации**

1. Что такое алгоритм? Перечислите и охарактеризуйте его основные свойства.
2. Способы записи алгоритмов: блок-схемы, псевдокод, программный код. Приведите примеры.
3. Понятие исполнителя алгоритма. Приведите 2–3 примера собственных исполнителей и сформулируйте для них задачи.

4. Основные алгоритмические конструкции: линейные, разветвляющиеся, циклические. Приведите примеры реализации каждой на языке программирования.

5. Понятие сложности алгоритма. Как оценить временную и пространственную сложность?

### **Основы языков программирования и среды разработки**

6. Эволюция языков программирования: основные этапы и ключевые языки.

7. Виды трансляторов (компиляторы, интерпретаторы): в чём разница и где применяются?

8. Интегрированные среды разработки (IDE): назначение, основные функции. Приведите примеры популярных IDE.

9. Основы визуального программирования: что это такое и какие преимущества даёт?

10. Структура программы на примере Pascal или Python: из каких частей состоит, назначение каждой части.

### **Типы данных и операции**

11. Классификация типов данных в языках программирования (простые, составные, пользовательские).

12. Тожественность и совместимость типов данных: что это значит и как влияет на написание кода?

13. Выражения, операции и операнды: виды операций, приоритет выполнения.

14. Форматированный ввод-вывод данных: зачем нужен и как реализуется?

15. Перечисляемый и интервальный типы данных: особенности и примеры использования.

### **Операторы и управление выполнением программы**

16. Простые операторы: присваивание, безусловный переход, вызов процедуры

17. Условные операторы (if, switch): синтаксис, семантика, примеры использования.

18. Операторы цикла (for, while, do-while): различия, выбор подходящего цикла для задачи.

19. Правила пунктуации при записи операторов: типичные ошибки и способы их избежать.

20. Тестирование и отладка программ: основные методы и инструменты.

### **Подпрограммы и модульность**

21. Структурный подход в программировании: суть и преимущества.

22. Метод пошаговой детализации: как применять при разработке алгоритмов?

23. Процедуры и функции: различия, синтаксис объявления и вызова.

24. Механизм передачи параметров (по значению, по ссылке): как работает и когда какой способ предпочтительнее?

25. Стандартные библиотечные модули: назначение и примеры использования.

### **Работа с данными сложной структуры**

26. Массивы: описание типа, операции над массивами, обработка элементов.

27. Строки: строковые выражения, процедуры и функции для обработки строк.

28. Множества: особенности типа данных, операции над множествами.
29. Комбинированные типы данных (записи, структуры): описание, доступ к полям, примеры использования.
30. Файлы: описание типа «файл», средства обработки файлов (чтение, запись, поиск).

#### **Продвинутые темы**

31. Рекурсия: что это такое, формы рекурсивных процедур, примеры задач, решаемых рекурсивно.
32. Динамические структуры данных: понятие, виды (списки, стеки, очереди), реализация.
33. Графические возможности языков программирования: основы работы с графикой (рисование фигур, анимация).
34. Обработка исключений: что это и зачем нужно? Примеры обработки ошибок в коде.
35. Основы объектно-ориентированного программирования (ООП): классы, объекты, методы, инкапсуляция, наследование, полиморфизм.

#### **Практические задачи для закрепления**

1. Напишите программу, которая находит все делители заданного числа N.
2. Реализуйте алгоритм сортировки массива (пузырьком, выбором, вставками) и сравните их эффективность.
3. Создайте программу для поиска элемента в отсортированном массиве (бинарный поиск).
4. Напишите рекурсивную функцию для вычисления факториала числа.
5. Разработайте консольное приложение с меню для работы с массивом (добавление, удаление, поиск элементов).

#### **Требования для лабораторной работы**

##### **Цели лабораторных работ**

Лабораторные работы направлены на:

- закрепление теоретических знаний по алгоритмизации и программированию;
- приобретение практических навыков составления математических моделей, разработки алгоритмов, написания программ, их тестирования и отладки;
- формирование умений работать с современными Технологиями и организация производствами и инструментальными средствами программирования;
- развитие навыков решения типовых задач с использованием языков программирования.

##### **Структура и содержание работ**

Лабораторная работа включает несколько этапов:

1. Изучение теоретического материала — основы языка программирования, алгоритмы, структуры данных и т. д.
2. Получение индивидуального задания — каждому студенту выдаётся конкретное задание, которое он должен выполнить.
3. Разработка алгоритма решения задачи — может включать составление блок-схемы алгоритма.

4. Написание программного кода — реализация алгоритма на выбранном языке программирования (например, Pascal, Python, C# и др.).
5. Отладка и тестирование программы — проверка работы программы на тестовых примерах, устранение ошибок.
6. Подготовка отчёта — оформление результатов работы в соответствии с установленными требованиями.

Темы: лабораторных работ:

1. «Знакомство с Python».
2. «Линейные программы».
3. «Разветвляющиеся вычислительные процессы».
4. «Организация циклов».
5. «Одномерные массивы».
6. «Функции в программировании».
7. «Работа с файлами».

### **Требования к оформлению отчёта**

Отчёт по лабораторной работе обычно должен содержать:

- титульный лист;
- цель и задачи работы;
- индивидуальное задание;
- для каждой задачи: блок-схему алгоритма (если требуется), текст программы, тестовые примеры и результаты их выполнения;
- выводы.

К оформлению текста часто предъявляются следующие требования:

- использование текстового редактора (MS Word, OpenOffice/LibreOffice);
- шрифт — Times New Roman, размер — 14 пт, цвет — чёрный;
- межстрочный интервал — полуторный;
- отступ первой строки абзаца — 1,25 см;
- поля: левое — 3 см, верхнее, правое, нижнее — 1,5 см.

Для листинга программного кода может использоваться шрифт Courier New, размер — 12 пт, межстрочный интервал — одинарный.

### **Дополнительные требования**

Использование среды программирования. В зависимости от программы может потребоваться работа в конкретной среде разработки (например, Delphi, IDE для Python и т. д.).

Ответы на контрольные вопросы. В отчёте часто требуется дать краткие ответы на вопросы, представленные в методических указаниях.

Защита работы. Обычно включает демонстрацию работающей программы и устные пояснения студента (описание цели, структуры программы, выводов).

Соблюдение сроков. Лабораторные работы сдаются в установленные преподавателем сроки.

### **Критерии оценки**

При оценке лабораторной работы учитываются:

- правильность решения задачи;
- качество кода (читаемость, структурированность);

- полнота и корректность отчёта;
- умение объяснить алгоритм работы программы и ответить на вопросы при защите.

### **Рекомендации**

- перед выполнением работы тщательно изучите теоретический материал, на который она опирается;
- используйте рекомендованную литературу и методические указания;
- тестируйте программу на различных входных данных, включая граничные случаи;
- уделяйте внимание оформлению отчёта — несоблюдение требований может снизить оценку.

### **Примерные вопросы к зачету с оценкой:**

#### **Теоретические вопросы**

1. Понятие алгоритма. Свойства алгоритмов. Способы описания алгоритмов (словесный, блок-схемы, псевдокод).
2. Основные алгоритмические конструкции: следование, ветвление, цикл. Примеры реализации на языке программирования.
3. Понятие переменной. Типы данных: целочисленные, вещественные, символьные, логические. Приведение типов.
4. Операторы ввода-вывода данных. Особенности реализации в разных языках программирования.
5. Условные операторы (if, else, switch). Синтаксис и семантика. Вложенные условия.
6. Циклы: while, do-while, for. Отличия и области применения. Бесконечные циклы и способы их предотвращения.
7. Функции и процедуры: объявление, вызов, передача параметров. Возвращаемое значение. Рекурсия: понятие и примеры использования.
8. Массивы: одномерные и многомерные. Инициализация, доступ к элементам, основные операции обработки массивов.
9. Строки и операции над ними. Особенности работы со строками в разных языках программирования.
10. Указатели и ссылки: понятие, назначение, синтаксис. Динамическое выделение памяти.
11. Структуры данных: списки, стеки, очереди. Основные операции и реализация.
12. Файлы: типы файлов (текстовые, бинарные), операции открытия, чтения, записи, закрытия. Обработка ошибок при работе с файлами.
13. Объектно-ориентированное программирование (ООП): основные принципы (инкапсуляция, наследование, полиморфизм). Классы и объекты.
14. Исключения: понятие, обработка исключений. Блоки try-catch-finally.
15. Основы тестирования программ: виды тестирования (модульное, интеграционное, системное). Понятие тестового набора.

### **Практические задания и задачи**

1. Составить алгоритм и написать программу для вычисления суммы первых  $N$  натуральных чисел.
2. Написать программу, которая определяет, является ли введенное число простым.
3. Реализовать сортировку массива методом пузырька или выбором.
4. Написать функцию для поиска максимального элемента в массиве.
5. Составить программу для решения квадратного уравнения  $ax^2+bx+c=0$ .
6. Написать программу для подсчёта количества слов в текстовом файле.
7. Реализовать стек на основе массива с операциями добавления и удаления элемента.
8. Написать программу, которая преобразует строку в верхний регистр (или наоборот).
9. Составить алгоритм для поиска подстроки в строке (например, алгоритм прямого поиска).
10. Написать программу с использованием рекурсии для вычисления факториала числа.

### **Вопросы на понимание и анализ кода**

1. Проанализировать фрагмент кода и определить его результат при заданных входных данных.
2. Найти ошибки в представленном коде и предложить способы их исправления.
3. Оптимизировать предложенный алгоритм: уменьшить временную или пространственную сложность.
4. Сравнить два подхода к решению задачи (например, итеративный и рекурсивный) и обосновать выбор лучшего варианта.
5. Прокомментировать код: объяснить назначение функций, переменных, ключевых операций.

### **Рекомендации по подготовке к зачету с оценкой**

Промежуточная аттестация проводится в соответствии с Положением о проведении текущего контроля успеваемости и промежуточной аттестации обучающихся ГАОУ ВО ЛО «ГГУ». При подготовке к зачету с оценкой студент обязан повторить пройденный материал в строгом соответствии с учебной программой, примерным перечнем учебных вопросов, выносящихся на зачет с оценкой и содержащихся в данной программе. Для этой цели используется конспект лекций и литература, рекомендованная преподавателем. При необходимости студент может обратиться за консультацией и методической помощью к преподавателю. К зачету с оценкой допускается студент, выполнивший все практические и лабораторные задания. Зачет с оценкой проводится в форме как устного собеседования по заранее утвержденным на кафедре вопросам (теоретическим), так и выполнение предложенного практического задания и решения задачи.

(также утвержденного кафедрой), преподаватель задает дополнительно вопросы на понимание.

### **Требования к зачету с оценкой**

Выбор формы и порядок проведения зачета с оценкой осуществляется кафедрой. Оценка знаний студента в процессе зачета с оценкой осуществляется исходя из следующих критериев:

- умение сформулировать определения понятий, данных в вопросе, с использованием специальной лексики, показать связи между данными понятиями;
- способность дать развернутый ответ на поставленный вопрос с соблюдением логики изложения материала;
- проанализировать и сопоставить различные точки зрения на поставленную проблему;
- умение аргументировать собственную точку зрения, иллюстрировать высказываемые суждения и умозаключения практическими примерами.

### **Шкала оценивания зачета с оценкой**

Критерии оценки зачета с оценкой следующие:

**«Отлично»** — если обучающийся выполнил задания, сформулированные преподавателем, демонстрирует глубокие знания по теме (разделу) дисциплины, грамотно и логично излагает материал, даёт последовательный и исчерпывающий ответ на поставленные вопросы, делает обобщения и выводы. Освоен уровень всех составляющих компетенций: ПК-2., ПК-2.1; ПК-2.2; ПК-2.3.

**«Хорошо»** — если обучающийся выполнил задания, сформулированные преподавателем, демонстрирует прочные знания по теме (разделу) дисциплины, грамотно и логично излагает материал, даёт последовательный и полный ответ на поставленные вопросы, делает обобщения и выводы. Освоен уровень всех составляющих компетенций: ПК-2., ПК-2.1; ПК-2.2; ПК-2.3.

**«Удовлетворительно»** — если обучающийся частично выполнил задания, сформулированные преподавателем, демонстрирует знания основного материала по теме (разделу) дисциплины, даёт неполный, недостаточно аргументированный ответ, не делает правильные обобщения и выводы, ответил на дополнительные вопросы. Освоен уровень всех составляющих компетенций: ПК-2., ПК-2.1; ПК-2.2; ПК-2.3.

**«Неудовлетворительно»** — если обучающийся частично выполнил или не выполнил задания, сформулированные преподавателем, демонстрирует разрозненные знания по теме (разделу) дисциплины, допускает существенные ошибки и не корректирует ответ после дополнительных и уточняющих вопросов преподавателя, не делает обобщения и выводы, не ответил на дополнительные вопросы. Не освоен базовый уровень всех составляющих компетенций: ПК-2., ПК-2.1; ПК-2.2; ПК-2.3.

Комплект заданий и этапов формирования компетенции представлен в Фонде оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине, оформленный отдельным документом, представлен в приложении к РПД.

Работа с печатными изданиями для обучающегося может быть связана с трудностями в области доступа к современной научной печатной литературе. В связи с развитием научно-технического прогресса в такой ситуации надлежит воспользоваться материалами, находящимися в открытом доступе сети Internet. Также необходимо учитывать, что по состоянию на сегодняшний день многие справочные правовые системы содержат не только текст нормативных актов, но и научные статьи по различным вопросам (например, СПС «Консультант Плюс»). Одновременно следует обратить свое внимание на публичные библиотеки, предоставляющие возможность доступа к электронным версиям печатных источников. В силу кратковременности изучения и значительного объема данной учебной дисциплины кафедра настоятельно рекомендует систематически, а не эпизодически работать над изучением курса.

#### **8. Перечень основной, дополнительной учебной литературы, ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины (модуля)**

##### **а) основная литература:**

1. Бунаков П. Ю. «Основы алгоритмизации и программирования на языке Python». Учебное пособие имеет практическую направленность и включает теоретические сведения и большое количество практических примеров программ на языке Python. Особое внимание уделяется разбору примеров решения задач и написания программ, а также самостоятельной работе обучающихся — для этого разработано большое количество вариантов дополнительных заданий, 2024 г.
2. Бердникова А. А., Иванов С. Л., Лямин А. С., Рейн А. Д. «Основы алгоритмизации и программирования». В учебном пособии представлены теоретический материал и задания к практическим занятиям для изучения дисциплины «Основы алгоритмизации и программирования». Рассматривается алфавит языка C#, реализация основных типов алгоритмических структур (линейная, разветвлённая, циклическая), работа с массивами и функциями на этом языке. Особое внимание уделено работе со структурами, классами, файлами, 2024.

##### **б) дополнительная литература:**

1. Ковалёва З. А. «Основы программирования на языке PascalABC.NET. Основные управляющие структуры. Практикум». Издание содержит комплекс практических задач и рекомендаций по их выполнению, 2024 г.



2. «Алгоритмизация и программирование» (В. В. Трофимов, Т. А. Павловская). Издательство: «Юрайт». Год издания: 2025 (4-е издание).
3. «Основы алгоритмизации и программирования» (Е. Е. Карпович). Издательство: «Лань». Год издания: 2026.

**в) ресурсы сети «Интернет»:**

- 1) Электронно-библиотечная система «Университетская библиотека онлайн». <https://biblioclub.ru/>
- 2) Электронно-библиотечная система «Лань». <https://e.lanbook.com/>
- 3) Электронно-библиотечная система «Znanium». <https://znanium.com/>
- 4) Научная электронная библиотека «eLIBRARY.RU». <https://www.elibrary.ru/>
- 5) Электронно-библиотечная система «Юрайт». <https://biblio-online.ru/>

## **9. Методические указания для обучающихся по освоению дисциплины (модуля)**

Самостоятельная подготовка обучающихся проводится для углубления и закрепления знаний, полученных на лекциях и других видах занятий, для выработки навыков самостоятельного применения новых, дополнительных знаний и подготовки к предстоящим учебным занятиям, зачету с оценкой.

Важным условием успешного изучения дисциплины является посещение лекций. Под посещением подразумевается не форма пассивного присутствия, а активная работа по изучению нового материала. Подготовка к лекционным занятиям включает в себя анализ предлагаемых для изучения вопросов, изучение нормативных источников и учебной и научной литературы по рассматриваемым вопросам лекции. В процессе лекции обучающийся может задавать уточняющие вопросы, осуществить взаимосвязь нового материала с уже изученным, подготовить базу для эффективного использования полученных знаний, облегчить подготовку к практическому занятию. Эффективным способом фиксации лекционного материала является конспектирование, представляющее собой не только фиксацию важнейших моментов лекции, но и указание примеров для понимания того или иного теоретического материала.

При подготовке к практическому занятию необходимо использовать конспектированные материалы лекций, учебную и научную литературу. Подготовка ответов по выносимым на обсуждение вопросам практического занятия включает в себя не только прочтение материала, но и его анализ и критическую оценку. Обучающемуся следует выявить малоизученные аспекты рассматриваемых вопросов, проявить инициативу при подготовке к практическому занятию.

При подготовке к практическим занятиям рекомендуется систематизировать знания, изображая их в табличном, графическом или схематичном виде. Это позволит установить взаимосвязь изучаемых явлений, упростит задачу запоминания материала, облегчит процесс практического применения полученных знаний.

Задачей практических занятий является выработка умения использовать теоретические знания, проявить наличие практических навыков. При подготовке к практическому занятию следует заблаговременно обеспечить наличие необходимо для данного занятия материала, самостоятельно повторить ранее изученные темы.

Для успешного освоения дисциплины важным является умение работать с терминами и их определениями. Для работы с терминологией эффективным является использование как учебной и научной литературы, так и словарей.

Работа с терминами может осуществляться в форме составления собственных тематических словариков для удобства и скорости поиска необходимого термина. С этой целью необходимо каждый новый встречающийся термин записывать и во время подготовки к семинарским и практическим занятиям указывать соответствующее определение. В случае возникновения сложности выбора определения из имеющегося объема в рамках научного знания необходимо задавать вопросы преподавателю в рамках лекционных, практических и лабораторных занятий.

Интерактивные формы проведения занятий по дисциплине «Основы программирования» включают в себя следующие виды занятий:

- *интерактивные лекции*, предполагают использование метода проблемного изложения. При таком подходе лекция становится похожей на диалог, преподавание имитирует исследовательский процесс (выдвигаются первоначально несколько ключевых постулатов по теме лекции, изложение выстраивается по принципу самостоятельного анализа и обобщения студентами учебного материала). Эта методика позволяет заинтересовать студента, вовлечь его в процесс обучения. Противоречия научного познания раскрываются посредством постановки проблемы. Учебная проблема и проблемная ситуация являются основными структурными компонентами проблемного обучения. Перед началом изучения определенной темы курса ставится перед студентами проблемный вопрос или дается проблемное задание. Стимулируя разрешение проблемы, преподаватель снимает противоречия между имеющимся ее пониманием и требуемыми от студента знаниями. Эффективность такого метода в том, что отдельные проблемы могут подниматься самими студентами. Главный успех данного метода в том, что преподаватель добивается от аудитории «самостоятельного решения» поставленной проблемы;

- *анализ задания*, когда используется метод индукции, т.е. при объяснении нового материала и формировании понятий, мысль студента движется от единичного к общему, от частных суждений к обобщениям. Подбирая задания, которые служат исходным материалом для выявления тех или иных закономерностей или вывода правил, преподаватель в интерактивной форме побуждает студентов к анализу предложенного материала. В ходе обсуждения студенты должны сделать необходимые обобщения и выводы.

Оценочные и методические материалы по дисциплине «Основы программирования» представлены в ФОММ.

При подготовке к промежуточному или итоговому тестированию необходимо изучить теоретический и практический материал. Открытые тестовые задания (без вариантов ответов) выявляют умение решать типовые задания. Закрытые тестовые задания (с перечнем возможных вариантов ответов, среди которых хотя бы один ответ является неверным) обеспечивают структурность мышления, вынужденного выбрать из предложенных вариантов ответ все правильные варианты. Тестовые задания на установление соответствия подразумевают необходимость проявления не только знания учебного материала, но и умения применять правила формальной логики.

Эффективным способом для подготовки к тестированию является работа обучающегося по решению тестовых заданий, предоставленных для самостоятельной работы. Также при подготовке к такой форме контроля знаний, как решение тестовых заданий, следует самостоятельно попытаться проработать рассматриваемые в дисциплине вопросы в форме составления тестовых заданий.

При подготовке к зачету с оценкой следует иметь в виду, что он является итоговой формой контроля по изучению данной учебной дисциплины. Зачет с оценкой подразумевает максимальную концентрацию знаний и умений, предполагающих полное изучение материала дисциплины.

Зачет с оценкой может проходить как в форме собеседования, так и в форме тестирования.

Решение преподавателя об итоговой аттестации (зачете) принимается по результатам всего собеседования на основе полноты и достоверности изложенного ответа и проявленных умений практического применения теоретических знаний.

Рекомендуется, наряду с печатными изданиями, использовать электронные библиотечные системы, а также ресурсы сети Интернет.

## **10. Особенности освоения дисциплины для инвалидов и лиц с ограниченными возможностями здоровья**

Обучение обучающихся с ограниченными возможностями здоровья при необходимости осуществляется на основе адаптированной рабочей программы с использованием специальных методов обучения и дидактических материалов, составленных с учетом особенностей психофизического развития, индивидуальных возможностей и состояния здоровья таких обучающихся (обучающегося).

В целях освоения учебной программы дисциплины «Основы программирования» инвалидами и лицами с ограниченными возможностями здоровья Институт обеспечивает:

- для инвалидов и лиц с ограниченными возможностями здоровья по зрению: размещение в доступных для обучающихся, являющихся слепыми или слабовидящими, местах и в адаптированной форме справочной информации о расписании учебных занятий; присутствие ассистента, оказывающего

- обучающемуся необходимую помощь; выпуск альтернативных форматов методических материалов (крупный шрифт или аудиофайлы);
- для инвалидов и лиц с ограниченными возможностями здоровья по слуху: надлежащими звуковыми средствами воспроизведение информации;
  - для инвалидов и лиц с ограниченными возможностями здоровья, имеющих нарушения опорно-двигательного аппарата: возможность беспрепятственного доступа обучающихся в учебные помещения, туалетные комнаты и другие помещения, а также пребывание в указанных помещениях. Обучающиеся из числа инвалидов и лиц с ОВЗ обеспечены печатными и (или) электронными образовательными ресурсами в формах, адаптированных к ограничениям их здоровья. Образование обучающихся с ограниченными возможностями здоровья может быть организовано как совместно с другими обучающимися, так и в отдельных группах или в отдельных организациях.

## **11. Перечень информационных технологий, профессиональных баз данных, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень программного обеспечения и информационных справочных систем**

- 1) Операционная система (Microsoft Windows Проприетарная);
- 2) Пакет офисных программ Microsoft Office (MS Word, MS Excel, MS Power Point, MS Access, MS Publisher и др. Проприетарная);
- 3) Программное обеспечение для просмотра электронных документов в стандарте PDF (Foxit Reader GNU Lesser General Public License);
- 4) Web-браузер (Mozilla Firefox GNU Lesser General Public License);
- 5) Автоматизированная информационная библиотечная система Marc21SQL;
- 6) Справочно-правовая система «Консультант Плюс»;
- 7) Реферативная и справочная база данных рецензируемой литературы Scopus <https://www.scopus.com>
- 8) Политематическая реферативно-библиографическая и наукометрическая (библио метрическая) база данных WebofScience <https://apps.webofknowledge.com>
- 9) Научная электронная библиотека [www.elibrary.ru](http://www.elibrary.ru)

## **12. Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине (модулю)**

<b>Наименование</b>
<b>Специализированные аудитории:</b>

Учебная аудитория для проведения занятий семинарского типа, курсового проектирования (выполнения курсовых работ), групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации / компьютерный класс / помещение для самостоятельной работы*
<b>Технические средства обучения:</b>
компьютеры с программным обеспечением, указанным в п.11
<b>Специализированные аудитории:</b>
Учебная аудитория для проведения занятий лекционного типа, занятий семинарского типа, курсового проектирования (выполнения курсовых работ), групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации*
<b>Технические средства обучения:</b>
экран настенный
мультимедийный проектор
компьютер с программным обеспечением, указанным в п.11

\* Аудитории конкретизируются в справке МТО