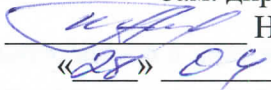


МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ
И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ
Северо-Кавказский филиал
ордена Трудового Красного Знамени федерального государственного
бюджетного образовательного учреждения высшего образования
«Московский технический университет связи и информатики»

УТВЕРЖДАЮ
Зам. директора по УР
Н.А. Андреева

«28» 09 2026 г.

Б1.В.10 Функциональное программирование

рабочая программа дисциплины

Кафедра	Информатики и вычислительной техники
Направление подготовки	09.03.01. Информатика и вычислительная техника
Профиль	Искусственный интеллект и машинное обучение
Формы обучения	очная, заочная

**Распределение часов дисциплины по семестрам (для очной формы обучения (ОФО), курсам
(для заочной формы обучения (ЗФО))**

Вид учебной работы	ОФО		ЗФО	
	ЗЕ	часов/ семестр	ЗЕ	часов/ курс
Общая трудоемкость дисциплины, в том числе (по семестрам, курсам):	3	108/6	3	108/4
Контактная работа, в том числе (по семестрам, курсам):		56/6		20/4
Лекции		14/6		6/4
Лабораторные работы		28/6		6/4
Практические занятия		14/6		8/4
Семинарские занятия				
Самостоятельная работа студентов		52/6		88/4
Контроль				
Число контрольных работ (по семестрам, курсам)				
Число КР (по семестрам, курсам)				
Число КП (по семестрам, курсам)				
Число зачетов (с разбивкой по семестрам, курсам)		1/6		1/4
Число экзаменов (с разбивкой по семестрам, курсам)				

Программу составил:
ст. преподаватель кафедры ИВТ Осмоловская Н.С.

Рабочая программа дисциплины
«Функциональное программирование»

Разработана в соответствии с ФГОС ВО
направления подготовки **09.03.01. Информатика и вычислительная техника**, утвержденным
приказом Министерства образования и науки Российской Федерации
от 19 сентября 2017 г. № 929 .

Составлена на основании учебного плана
направления **09.03.01. Информатика и вычислительная техника** профиля «Искусственный
интеллект и машинное обучение», одобренного Учёным советом СКФ МТУСИ, протокол № 8
от 31.03. 2026 г., и утвержденного директором СКФ МТУСИ 31.03. 2026 г.

Рассмотрена и одобрена на заседании кафедры
«Информатика и вычислительная техника»

Протокол от «28» 04 2026 г. № 9

Зав. кафедрой  Соколов С.В.

1. Цели изучения дисциплины

Целями изучения дисциплины «Функциональное программирование» являются:

- освоение основ и методов функционального программирования и выработка практических навыков применения этих знаний;
- изучение основных принципов функционального программирования, их основных применений в современном программировании;
- дать ориентиры в дальнейшем углубленном изучении отдельных вопросов в специализированных разделах математической логики и функционального программирования;
- формирование у студентов профессиональных знаний и практических навыков по разработке и созданию моделей интеллектуальных систем с помощью языков функционального программирования.

2. Планируемые результаты обучения

Изучение дисциплины направлено на формирование у обучающегося способности решать профессиональные задачи в соответствии с проектной деятельностью.

Результатом освоения дисциплины являются сформированные у обучающегося следующие компетенции:

Компетенции обучающегося, формируемые в результате освоения дисциплины (в части, обеспечиваемой дисциплиной)	
ПК-3: Способен разрабатывать и внедрять программное обеспечение с применением современных средств и технологий, осуществлять его поддержку	
Знать:	методы и приемы формализации, алгоритмизации, программирования и оформления программного кода; компоненты программно-технических архитектур, существующие приложения и интерфейсы взаимодействия с ними.
Уметь:	разрабатывать программное обеспечение с использованием языков и сред программирования, выполнять определение и манипулирование данными.
Владеть:	приемами анализа возможностей и разработки требований к программному обеспечению; методами и средствами интеграции модулей и компонент программного обеспечения, приемами развертывания и обновления программного обеспечения.

3. Место дисциплины в структуре образовательной программы

Требования к предварительной подготовке обучающегося (предшествующие дисциплины, модули, темы):	
1	Б1.О.30 «Структуры и алгоритмы обработки данных»
2	Б1.О.08 «Информационные технологии и программирование»

3	Б1.О.11 «Дискретная математика»
4	Б1.О.14 «Управление ИТ-проектами»
5	Б1.О.15 «Теория информации, данные, знания»
6	Б1.О.19 «Высшая математика»
7	Б1.О.26 «Введение в информационные технологии»
Последующие дисциплины и практики, для которых освоение данной дисциплины необходимо:	
1	Б1.В.ДВ.02.01 Программирование мобильных устройств
2	Б1.В.ДВ.03.01 WEB-программирование
3	Б2.В.02(П) Производственная практика (системы искусственного интеллекта)
4	Б3.01 Выполнение и защита выпускной квалификационной работы

4. Структура и содержание дисциплины

4.1 Очная форма обучения, 4 года (всего 108 часов, 56 часов контактной работы)

Код зан.	Тема и краткое содержание занятия	Вид зан.	Кол. часов	Компетенции	УМИО
1	2	3	4	5	6
Курс 3, Семестр 6					
Модуль 1 – Концепция функционального программирования – 54 (8 Лек+14 ЛР+6 ПЗ) часов контактной работы – 26 часов СРС					
1.1	Введение в функциональное программирование. Соотношение функционального программирования с объектно-ориентированным. Абстрактная и конкретная формы программ. Проблема связывания значений и переменных.	Лек.	2	ПК-3	Л1.1 Л2.1 Л2.2 Л2.3 Л2.4
1.2	Сравнение императивной и декларативной парадигм. Техника интерпретации функциональных программ. Особенности интерпретирования императивных программ.	СРС	4	ПК-3	Л1.1 Л2.1 Л2.4
1.3	Эволюция C++ как языка функционального программирования	СРС	4	ПК-3	Л1.1 Л2.1
1.4	Основы функционального программирования. Выполнение работ по индивидуальным заданиям.	ЛР	6	ПК-3	Л1.2 Л2.1
1.5	Преимущества функционального программирования. Краткость и удобочитаемость кода. Параллельная обработка и синхронизация. Непрерывная оптимизация. Функциональные эквиваленты императивных программ. Преобразование императивных программ в функциональные. Аппаратное обеспечение функциональных программ.	Лек.	2	ПК-3	Л1.1 Л1.2 Л2.1 Л2.2 Л2.3 Л2.4 Л3.1
1.6	Преимущества функционального программирования. Модульность и возможность повторного использования. Уменьшение побочных эффектов. Конкурентность и параллелизм. Выразительный и читаемый код	СРС	6	ПК-3	Л1.1 Л2.1 Л2.3 Л2.4
1.7	Разработка приложений на основе функций по индивидуальным заданиям	ЛР	4	ПК-3	Л1.2 Л2.1
1.8	Синтаксис функции в языке программирования. Виды функций. Возвращаемое значение. Локальные и	Лек.	2	ПК-3	Л1.1 Л2.1

	глобальные переменные. Использование адресации памяти при работе с функциями.				Л2.3 Л2.4
1.9	Перегрузка функций. Концепция перегрузки функций. Перегрузка стандартных функций. Перегрузка операторов. Использование перегруженных операторов в программе. Передача функции в качестве аргумента. Реализация на основе циклов. Рекурсия и оптимизация хвостового вызова	Лек.	2	ПК-3	Л1.1 Л2.1 Л2.3 Л2.4
1.10	Функции: объявления и определение. Передача аргументов функции. Стандартная библиотека функций. Библиотека стандартного потокового ввода, вывода. Форматированный ввод, вывод. Файловые потоки.	СРС	6	ПК-3	Л1.1 Л1.2 Л2.4 Л3.1
1.11	Рекурсивные функции. Аргументы по умолчанию. Перегрузка функций. Шаблоны функций.	СРС	6	ПК-3	Л1.1 Л2.4
1.12	Применение перегрузки функций. Выполнение работ по индивидуальным заданиям.	ЛР	4	ПК-3	Л1.2 Л2.1
1.13	Объявление и определение абстрактных типов данных.	ПЗ	2	ПК-3	Л1.2 Л2.1
1.14	Реализация абстрактных типов данных. Перегрузка операторов.	ПЗ	2	ПК-3	Л1.2 Л2.1
1.15	Разработка и использование шаблонов абстрактных типов данных.	ПЗ	2	ПК-3	Л1.2 Л2.1
Модуль 2 – Функциональные структуры данных – 54 (6 Лек+14 ЛР+8 ПЗ) часов контактной работы – 26 часов СРС					
2.1	Структуры данных: связный список. Представление связного списка. Типы связных списков. Базовые операции: вставка, удаление, обратная операция, отображение, поиск. Добавление и удаление элемента в начале, конце, середине списка.	Лек.	2	ПК-3	Л1.1 Л1.2 Л2.2 Л2.4 Л3.1
2.2	Списки. Операции над односвязными списками. Двухнаправленные и кольцевые списки. Операции над кольцевыми списками.	СРС	6	ПК-3	Л1.1 Л1.2 Л2.2
2.3	Неизменяемые векторы. Поиск элементов в префиксном дереве. Добавление элементов в конец префиксного дерева. Изменение элементов. Удаление элемента из конца префиксного дерева. Другие операции и общая эффективность префиксных деревьев	Лек.	2	ПК-3	Л1.1 Л1.2 Л2.2 Л2.4 Л3.1
2.4	Деревья. Определение и построение. Таблицы. Двоичное (бинарное) дерево: создание и обход. Функция сравнения или получение ключа. Поиск в глубину (Depth-first search, DFS). Поиск в ширину (breadth-first search, BFS)	СРС	10	ПК-3	Л1.1 Л1.2 Л2.2 Л2.4 Л3.1
2.5	Алгебраические типы данных. Определение типов-сумм через наследование. Определение типов-сумм с использованием объединений. Реализация конкретных состояний. Особый тип-сумма: необязательные значения. Типы-суммы для обработки ошибок. Моделирование предметной области с алгебраическими типами.	Лек.	2	ПК-3	Л1.1 Л2.2 Л2.3 Л2.4
2.6	Алгебраические типы данных (ADT и GADT). Математические свойства. Реляционная алгебра и алгебраические типы данных. Модель ошибок	СРС	10	ПК-3	Л1.1 Л2.2 Л2.4

2.7	Функции высших порядков. Композиции. Рекурсия.	ЛР	4	ПК-3	Л1.1 Л1.2
2.8	Разработка библиотеки функций высших порядков. Составление и использование библиотек.	ЛР	4	ПК-3	Л1.1 Л1.2
2.9	Разработка и сборка проектов с использованием библиотек	ЛР	6	ПК-3	Л1.1 Л1.2
2.10	Контейнеры и итераторы в библиотеке STL. Выполнение работ по индивидуальным заданиям.	ПЗ	2	ПК-3	Л 1.2 Л 2.1
2.11	Вектор. Очереди. Стек. Список. Выполнение работ по индивидуальным заданиям.	ПЗ	4	ПК-3	Л 1.2 Л 2.1
2.12	Ассоциативные массивы. Объекты-функции и предикаты.	ПЗ	2	ПК-3	Л 1.2 Л 2.1
Итого – 108 часов					

4.2 Заочная форма обучения, 5 лет (всего 108 часов, 20 часов контактной работы)

Код зан.	Тема и краткое содержание занятия	Вид зан.	Кол. часов	Компетенции	УМИО
1	2	3	4	5	6
Курс 5					
Модуль 1 – Концепция функционального программирования – 54 (4 Лек+4 ЛР+2 ПЗ) часов контактной работы – 44 часов СРС					
1.1	Введение в функциональное программирование. Соотношение функционального программирования с объектно-ориентированным. Абстрактная и конкретная формы программ. Проблема связывания значений и переменных.	Лек.	2	ПК-3	Л1.1 Л2.1 Л2.2 Л2.3 Л2.4
1.2	Сравнение императивной и декларативной парадигм. Техника интерпретации функциональных программ. Особенности интерпретирования императивных программ.	СРС	12	ПК-3	Л1.1 Л2.1 Л2.3 Л2.4
1.3	Эволюция C++ как языка функционального программирования	СРС	10	ПК-3	Л1.1 Л2.1
1.4	Основы функционального программирования. Выполнение работ по индивидуальным заданиям.	ЛР	2	ПК-3	Л 1.2 Л 2.1
1.5	Преимущества функционального программирования. Модульность и возможность повторного использования. Уменьшение побочных эффектов. Конкурентность и параллелизм. Выразительный и читаемый код	СРС	12	ПК-3	Л1.1 Л2.1 Л2.3 Л2.4
1.6	Перегрузка функций. Концепция перегрузки функций. Перегрузка стандартных функций. Перегрузка операторов. Использование перегруженных операторов в программе. Передача функции в качестве аргумента. Реализация на основе циклов. Рекурсия и оптимизация хвостового вызова	Лек.	2	ПК-3	Л1.1 Л2.1 Л2.3 Л2.4
1.7	Функции: объявления и определение. Передача аргументов а функции. Стандартная библиотека функций. Библиотека стандартного потокового ввода, вывода. Форматированный ввод, вывод. Файловые потоки.	СРС	10	ПК-3	Л1.1 Л1.2 Л2.2 Л2.4 Л3.1

1.8	Применение перегрузки функций. Выполнение работ по индивидуальным заданиям.	Лаб.	2	ПК-3	Л 1.2 Л 2.1
1.9	Объявление и определение абстрактных типов данных. Реализация абстрактных типов данных. Перегрузка операторов.	ПЗ	2	ПК-3	Л 1.2 Л 2.1
Модуль 2 – Функциональные структуры данных – 54 (2 Лек+2 ЛР+6 ПЗ) часов контактной работы – 44 часов СРС					
2.1	Структуры данных: связный список. Представление связного списка. Типы связных списков. Базовые операции: вставка, удаление, обратная операция, отображение, поиск. Добавление и удаление элемента в начале, конце, середине списка.	Лек.	2	ПК-3	Л1.1 Л1.2 Л2.2 Л2.4 Л3.1
2.2	Деревья. Определение и построение. Таблицы. Двоичное (бинарное) дерево: создание и обход. Функция сравнения или получение ключа. Поиск в глубину (Depth-first search, DFS). Поиск в ширину (breadth-first search, BFS)	СРС	22	ПК-3	Л1.1 Л1.2 Л2.2 Л2.4 Л3.1
2.3	Разработка библиотеки функций высших порядков. Составление и использование библиотек. Разработка и сборка проектов с использованием библиотек	ЛР	2	ПК-3	Л1.1 Л1.2
2.4	Алгебраические типы данных (ADT и GADT). Математические свойства. Реляционная алгебра и алгебраические типы данных. Модель ошибок	СРС	22	ПК-3	Л1.1 Л1.2
2.5	Контейнеры и итераторы в библиотеке STL. Выполнение работ по индивидуальным заданиям.	ПЗ	2	ПК-3	Л 1.2 Л 2.1
2.6	Вектор. Очереди. Стек. Список. Выполнение работ по индивидуальным заданиям.	ПЗ	2	ПК-3	Л 1.2 Л 2.1
2.7	Ассоциативные массивы. Объекты-функции и предикаты.	ПЗ	2	ПК-3	Л 1.2 Л 2.1
Итого – 108 часов					

5. Учебно-методическое и информационное обеспечение дисциплины

5.1 Рекомендуемая литература				
5.1.1. Основная литература				
Код	Авторы, составители	Заглавие	Издательство, год	Кол.
Л1.1	И. Чукич, В. Ю. Винник, А. Н. Киселев.	Функциональное программирование на языке C++	Москва: ДМК Пресс, 2020	Э1
Л1.2	А.С. Фарафонов	Программирование на языке высокого уровня: методические указания к проведению лабораторных работ	Липецк: ЛГТУ ЭБС АСВ, 2013	Э2
5.1.2 Дополнительная литература				
Код	Авторы, составители	Заглавие	Издательство, год	Кол.
Л2.1	Конев С.И.	Функциональное программирование. Ч.1: учебное пособие	Ростов-на-Дону, СКФ МТУСИ, 2018	Э3
Л2.2	Доткулова А.С.	Практикум по дисциплине «Логическое и функциональное программирование»	Москва, МТУСИ, 2016	Э4

Л2.3	Яшина М.В., Барков В.В., Украинский С.В.	Учебно-методическое пособие по дисциплине: «Логическое и функциональное программирование»	Москва, МТУСИ, 2016	Э5
Л2.4	Козырева Г.Ф.	Функциональное и логическое программирование: учебно-методическое пособие	Саратов: Ай Пи Эр Медиа, 2018	Э6

5.2 Электронные образовательные ресурсы

Э1	https://www.iprbookshop.ru/124744.html
Э2	https://www.iprbookshop.ru/22912.html
Э3	https://www.iprbookshop.ru/89511.html
Э4	https://www.iprbookshop.ru/61489.html
Э5	https://www.iprbookshop.ru/61490.html
Э6	https://www.iprbookshop.ru/71596.html

5.3 Информационно-справочные системы и профессиональные базы данных

5.3.1	Консультант Плюс – https://www.consultant.ru/
5.3.2	Гарант – https://www.garant.ru/
5.3.3	Консорциум Кодекс – https://docs.cntd.ru/
5.3.4	Научная электронная библиотека – https://www.elibrary.ru/
5.3.5	Профессиональная база данных: Федеральный портал «Российское образование»: [Электронный ресурс] – Режим доступа: http://www.edu.ru/ (открытый доступ)
5.3.6	Профессиональная база данных: Федеральный центр информационно-образовательных ресурсов: [Электронный ресурс] – Режим доступа: http://fcior.edu.ru/ (открытый доступ)

5.4 Программное обеспечение

П.1	CodeBlocks Среда программирования на языке C/C++
П.2	IDLE Среда программирования на языке Python
П.3	LibreOffice Кроссплатформенный, свободно распространяемый офисный пакет
П.4	Яндекс браузер
П.6	Антивирус Касперского

6. Материально-техническое обеспечение дисциплины

6.1 МТО лекционных занятий	
1	Лекционная аудитория, оснащенная интерактивной доской, проектором, ПК (ноутбуком), экраном.
6.2 МТО лабораторных работ и практических занятий	
1	Учебные аудитории, оборудованные компьютерной техникой, с возможностью подключения к сети «Интернет», и обеспечением доступа в электронную информационно-образовательную среду СКФ МТУСИ
6.3 МТО рубежных контролей, зачетов, экзаменов	
1	Учебные аудитории, оборудованные компьютерной техникой с возможностью выхода в локальную сеть Филиала и Интернет.
6.4 МТО самостоятельной работы обучающихся	
1	Помещение для самостоятельной работы обучающихся оснащено компьютерной техникой с возможностью подключения к сети Интернет и обеспечением доступа в электронную информационно-образовательную среду СКФ МТУСИ

7. Оценочные материалы

Оценочные материалы и перечень видов оценочных средств текущего контроля и промежуточной аттестации представлены в Приложении 1 к рабочей программе дисциплины.

8. Методические рекомендации по освоению дисциплины

Методические рекомендации по освоению дисциплины представлены в Приложении 2 к рабочей программе дисциплины.

9. Особенности реализации дисциплины (модуля) при обучении инвалидов и лиц с ограниченными возможностями здоровья

Рабочая программа дисциплины (модуля) при необходимости может быть адаптирована для обучения (в том числе с применением дистанционных образовательных технологий) лиц с ограниченными возможностями здоровья, инвалидов. Для этого требуется заявление обучающихся, являющихся лицами с ограниченными возможностями здоровья, инвалидами, или их законных представителей и рекомендации психолого-медико-педагогической комиссии. При обучении лиц с ограниченными возможностями здоровья учитываются их индивидуальные психофизические особенности. Обучение инвалидов осуществляется также в соответствии с индивидуальной программой реабилитации инвалида (при наличии).

Для лиц с нарушением слуха возможно предоставление учебной информации в визуальной форме (краткий конспект лекций; тексты заданий, напечатанные увеличенным шрифтом), на аудиторных занятиях допускается присутствие ассистента, а также сурдопереводчиков и тифлосурдопереводчиков.

Текущий контроль успеваемости осуществляется в письменной форме: обучающийся письменно отвечает на вопросы, письменно выполняет практические задания.

Доклад (реферат) также может быть представлен в письменной форме, при этом требования к содержанию остаются теми же, а требования к качеству изложения материала (понятность, качество речи, взаимодействие с аудиторией и т. д.) заменяются на соответствующие требования, предъявляемые к письменным работам (качество оформления текста и списка литературы, грамотность, наличие иллюстрационных материалов и т.д.).

Промежуточная аттестация для лиц с нарушениями слуха проводится в письменной форме, при этом используются общие критерии оценивания. При необходимости время подготовки к ответу может быть увеличено.

Для лиц с нарушением зрения допускается аудиальное предоставление информации, а также использование на аудиторных занятиях звукозаписывающих устройств (диктофонов и т.д.). Допускается присутствие на занятиях ассистента (помощника), оказывающего обучающимся необходимую техническую помощь.

Текущий контроль успеваемости осуществляется в устной форме.

При проведении промежуточной аттестации для лиц с нарушением зрения тестирование может быть заменено на устное собеседование по вопросам.

Для лиц с ограниченными возможностями здоровья, имеющих нарушения опорно-двигательного аппарата, на аудиторных занятиях, а также при проведении процедур текущего контроля успеваемости и промежуточной аттестации могут быть предоставлены необходимые технические средства (персональный компьютер, ноутбук или другой гаджет); допускается присутствие ассистента (ассистентов), оказывающего обучающимся необходимую техническую помощь (занять рабочее место, передвигаться по аудитории, прочитать задание, оформить ответ, общаться с преподавателем).

10. Дополнения и изменения в Рабочей программе

Лист актуализации рабочей программы дисциплины «Б1.В.10 Функциональное программирование» для использования в 20___/20___ учебном году

Утверждаю

Зам. директора по УР _____
«__» _____ 20__ г.

Направление: 09.03.01. Информатика и вычислительная техника

Профиль: Искусственный интеллект и машинное обучение

Форма обучения: очная, заочная

(Возможны следующие варианты):

а) Рабочая программа действует без изменений.

б) В рабочую программу вносятся следующие изменения:

- 1)
- 2)
- 3)

Разработчик (и): _____
(ФИО, ученая степень, ученое звание)

«__» _____ 20__ г.

Рабочая программа пересмотрена и одобрена на заседании кафедры _____

Протокол № _____ от «__» _____ 20__ г.

Заведующий кафедрой _____

Оценочные материалы для проведения текущего контроля и промежуточной аттестации по дисциплине

1. Оценочные материалы для проведения промежуточной аттестации по дисциплине

1.1 Шкала оценивания компетенций

Шкала оценивания компетенций		
Оценка	Уровень освоения компетенции	Критерии оценивания
«Отлично»	Высокий уровень	Обучающийся показывает всестороннее, систематическое и глубокое знание основного и дополнительного учебного материала, дает полные, профессиональные и грамотные ответы на поставленные вопросы, что свидетельствует об освоении обучающимся взаимосвязи основных понятий дисциплины; усвоил основную и знаком с дополнительной рекомендованной литературой; проявляет творческие способности в понимании, изложении и использовании учебно-программного материала.
«Хорошо»	Повышенный уровень	Обучающийся показывает достаточный уровень знания основного и дополнительного учебного материала, дает в целом грамотные ответы на поставленные вопросы, что свидетельствует об освоении обучающимся взаимосвязи основных понятий дисциплины; усвоил основную и знаком с дополнительной рекомендованной литературой. Допускает не существенные погрешности в ответах, устраняет их без помощи преподавателя.
«Удовлетворительно»	Пороговый уровень	Обучающийся демонстрирует знание основного учебного материала в минимальном объеме, необходимом для дальнейшей учебы; в целом отвечает на поставленные вопросы, допуская не принципиальные ошибки; знаком с основной литературой. Допускает ошибки в ответах различного уровня, которые исправляет после наводящих вопросов и под руководством преподавателя.
«Неудовлетворительно»	Минимальный уровень не достигнут	Обучающийся обнаруживает существенные пробелы в знаниях основного учебно-программного материала, допустившему принципиальные ошибки в выполнении, предусмотренных программой, заданий; не знаком с рекомендованной литературой, не может исправить допущенные ошибки. Как правило, оценка «неудовлетворительно» ставится обучающимся, которые не могут продолжить обучение или приступить к профессиональной деятельности по окончании вуза без дополнительных занятий по соответствующей дисциплине.

1.2 Показатели, критерии и шкалы оценивания компетенций

Показатели компетенции	Критерии оценивания	Шкала оценивания
ПК-3: Способен разрабатывать и внедрять программное обеспечение с применением современных средств и технологий, осуществлять его поддержку		
Знать:		
– методы и приемы формализации, алгоритмизации, программирования и оформления программного кода;	Контрольная работа	Модуль 2 0÷14 «Неудовлетворительно» - 0-7 «Удовлетворительно» - 8-10 «Хорошо» - 11-12 «Отлично» - 13-14
– компоненты программно-технических архитектур, существующие приложения и интерфейсы взаимодействия с ними.	Контрольная работа	Модуль 2 0÷14 «Неудовлетворительно» - 0-7 «Удовлетворительно» - 8-10 «Хорошо» - 11-12 «Отлично» - 13-14
Уметь:		
– разрабатывать программное обеспечение с использованием языков и сред программирования;	Лабораторная работа 1 Практические занятия 1-2	Модуль 1 0÷6 «Неудовлетворительно» - 0-2 «Удовлетворительно» - 3 «Хорошо» - 4-5 «Отлично» - 6
– выполнять определение и манипулирование данными.	Лабораторная работа 4 Практические занятия 4-5	Модуль 2 0÷6 «Неудовлетворительно» - 0-2 «Удовлетворительно» - 3 «Хорошо» - 4-5 «Отлично» - 6
Владеть:		
– приемами анализа возможностей и разработки требований к программному обеспечению;	Лабораторные работы 2-3 Практическое занятие 3	Модуль 1 0÷6 «Неудовлетворительно» - 0-2 «Удовлетворительно» - 3 «Хорошо» - 4-5 «Отлично» - 6
– методами и средствами интеграции модулей и компонент программного обеспечения, приемами развертывания и обновления программного обеспечения.	Лабораторные работы 5-6 Практическое занятие 6	Модуль 2 0÷6 «Неудовлетворительно» - 0-2 «Удовлетворительно» - 3 «Хорошо» - 4-5 «Отлично» - 6
	Зачет	0-100 «Не зачтено» - 0-40 «Зачтено» - 41-100

1.3 Оценочные материалы: типовые контрольные задания, иные материалы

1.3.1 Оценочные материалы для очной и заочной форм обучения

Модуль 1 (50 баллов):

Модуль содержит 4 лекционных занятия, 3 практических и 3 лабораторных работы. Знание лекционного материала оценивается по контрольной работе, состоящей из двух вопросов, максимальное количество баллов за контрольную работу составляет 14. За выполненные и защищенные практические и лабораторные занятия студент получает максимум 36 баллов. Общее максимальное количество баллов за модуль 1 составляет 50 баллов.

Вопросы для контрольной работы (ПК-3):

1. Что такое функциональное программирование и чем оно отличается от императивного подхода? От объектно-ориентированного?
2. Назовите три ключевых принципа функционального программирования.
3. Дайте определение чистой функции и объясните её важность в функциональном подходе.
4. В чём состоит различие между абстрактной и конкретной формами программы?
5. Что представляет собой проблема связывания значений и переменных в функциональном программировании?
6. Назовите пять преимуществ функционального программирования перед традиционным императивным стилем.
7. Каким образом достигается высокая читаемость и лаконичность кода в функциональном подходе?
8. Почему параллелизм и синхронизация проще решаются средствами функционального программирования?
9. Как аппаратное обеспечение современных компьютеров влияет на эффективность исполнения функциональных программ?
10. Какие существуют специализированные языки и среды для поддержки функционального стиля программирования?
11. Что такое концепция перегрузки функций и как она реализуется в C++?
12. В чём суть перегрузки стандартных функций и какие ограничения здесь имеются?
13. Приведите пример перегрузки оператора в C++ и объясните его пользу.
14. Как передать функцию в качестве параметра другому методу в C++?
15. В чём преимущество передачи функций как аргументов по сравнению с традиционными способами организации кода?
16. Что такое рекурсивная функция и как её эффективно использовать в функциональном стиле?
17. Что такое хвостовая рекурсия и как она связана с оптимизацией производительности?
18. Как компилятор реализует оптимизацию хвостового вызова в C++?
19. Какие встроенные средства C++ поддерживают функциональный стиль программирования?
20. Приведите пример реализации функционального подхода в стандартной библиотеке STL.

Практическое занятие №1. Объявление и определение абстрактных типов данных.

Контрольные вопросы (ПК-3):

1. Что такое абстрактный тип данных (ADT)?
2. Какие основные компоненты входят в объявление класса в C++?
3. Перечислите и поясните назначение каждого компонента.
4. Как объявляется класс в C++, содержащий приватные члены-данные и публичные методы?
5. Для чего используются ключевые слова `public`, `private` и `protected` при определении классов?
6. Опишите различия между этими спецификаторами доступа и области их применения.
7. Объясните разницу между определением класса и созданием объекта этого класса.
8. Почему рекомендуется скрывать внутренние детали реализации класса (инкапсуляция)?
9. Какие преимущества даёт использование закрытых членов класса?
10. Приведите пример реализации простейшего абстрактного типа данных (например, стек или очередь) на языке C++.

Практическое занятие №2. Реализация абстрактных типов данных. Перегрузка операторов.

Контрольные вопросы (ПК-3):

1. Что такое перегрузка операторов в C++ и зачем она применяется?
2. Расскажите, как работает механизм перегрузки операторов и какие преимущества он даёт при проектировании собственных классов.
3. Какой оператор нельзя перегружать в C++?
4. Чем отличается перегрузка бинарных операторов от перегрузки унарных операторов?
5. Почему иногда полезно перегружать оператор вывода (<<) и ввода (>>) для потоковых операций?
6. Можно ли перегружать операторы сравнения (==, !=, <, <=, >, >=)? И как это сделать грамотно?
7. Зачем нужна перегрузка арифметических операторов (+, -, *) для пользовательских классов?
8. Каковы особенности реализации перегрузки оператора new и delete?
9. Когда и зачем целесообразно переопределять стандартные операторы выделения памяти?

Практическое занятие №3. Разработка и использование шаблонов абстрактных типов данных.

Контрольные вопросы (ПК-3):

1. Что такое шаблон класса в C++ и зачем он используется?
2. Как выглядит синтаксис объявления шаблона класса в C++?
3. Как создать экземпляр шаблона класса с конкретизированным типом данных?
4. Может ли шаблон класса содержать статические переменные и методы?
5. Если да, покажите пример реализации статического члена и метода в шаблоне класса.
6. Как организовать специализацию шаблона класса для определенного типа данных?
7. Как работают шаблоны функций в сочетании с шаблонами классов?
8. Как избежать дублирования кода при создании похожих реализаций для разных типов данных?
9. Как обеспечить безопасность типов при использовании шаблонов классов?
10. Какова разница между шаблоном класса и обычным классом в C++?

Лабораторная работа №1. Основы функционального программирования. Выполнение работ по индивидуальным заданиям.

Контрольные вопросы (ПК-3):

1. Что такое функциональное программирование и как оно связано с математическими и вычислительными теориями?
2. Как реализовать передачу данных между функциями без изменения внешнего состояния системы?
3. Какие стандартные библиотечные функции и алгоритмы C++ способствуют применению функционального программирования?
4. Как применять рекурсию вместо циклических конструкций в функциональном программировании на C++?
5. Как интегрировать обработку ошибок и исключений в функциональном стиле программирования на C++?
6. Как преобразовать существующие императивные программы в функциональный стиль на языке C++?

Лабораторная работа №2. Разработка приложений на основе функций по индивидуальным заданиям.

Контрольные вопросы (ПК-3):

1. Что такое функция в C++ и каково её предназначение в структуре программы?
2. Как объявляются и определяются функции в C++? Приведите примеры.
3. Как осуществляется передача параметров в функции по значению и по ссылке? В чём отличия?

4. Как вернуть значение из функции в C++?
5. Возможны ли множественные возвраты?
6. Что такое перегрузка функций в C++ и как она реализуется?
7. Как использовать рекурсию в C++ для решения задач? Приведите пример рекурсивной функции.
8. Как организовывать взаимодействие между несколькими функциями в приложении на C++?
9. Как обрабатываются ошибки и исключительные ситуации в функциях C++?
10. Как оптимизировать производительность функций в приложениях на C++?

Лабораторная работа №3. Применение перегрузки функций. Выполнение работ по индивидуальным заданиям.

Контрольные вопросы (ПК-3):

1. Что такое перегрузка функций в C++ и для чего она используется?
2. Как объявлять и определять перегружаемые функции в C++?
3. Как компилятор выбирает подходящую перегруженную функцию при вызове?
4. Какие ограничения накладываются на параметры перегружаемых функций?
5. Могут ли перегруженные функции иметь одинаковое количество параметров?
6. Как использовать константные параметры и ссылки в перегруженных функциях?
7. Как реализовать перегрузку функций с различными типами возврата?
8. Как протестировать правильность работы перегруженных функций в приложении на C++?

Модуль 2 (50 баллов):

Модуль содержит 3 лекционных занятий, 3 практических и 3 лабораторных работы. Знание лекционного материала оценивается по контрольной работе, состоящей из двух вопросов, максимальное количество баллов за контрольную работу составляет 14. За выполненные и защищенные практические и лабораторные занятия студент получает максимум 36 балла. Общее максимальное количество баллов за модуль 1 составляет 50 баллов.

Вопросы для контрольной работы (ПК-3):

1. Что такое связный список и как он представляется в памяти компьютера?
2. Какие бывают типы связных списков (одно-, двунаправленные, кольцевые)?
3. Как выполняется вставка нового элемента в начало однонаправленного связного списка?
4. Как удалить элемент из середины связного списка?
5. Как реализовать операцию обращения связного списка (reverse)?
6. Как осуществить эффективный поиск элемента в связном списке?
7. Как добавить элемент в конец двусвязного списка?
8. Каковы преимущества и недостатки связных списков по сравнению с массивами?
9. Как изменить значение элемента в середине связного списка?
10. Как эффективно удалять элемент из начала связного списка?
11. Что такое неизменяемый вектор и как он отличается от обычного массива?
12. Как обновляют содержимое неизменяемых векторов?
13. Какие преимущества имеют неизменяемые векторы перед изменяемыми структурами данных?
14. Как реализуют эффективную вставку и удаление элементов в неизменяемых векторах?
15. Что такое префиксное дерево (trie) и для чего оно используется?
16. Как добавлять элементы в конец префиксного дерева?
17. Как искать элементы в префиксном дереве?
18. Как удаляется элемент из конца префиксного дерева?
19. Какие дополнительные операции возможны в префиксных деревьях помимо добавления и удаления?
20. Как оценивается общая эффективность операций в trie относительно других структур данных?

21. Что такое алгебраические типы данных и как они отличаются от примитивных типов?
22. Как определяется тип-сумма через наследование в языках программирования?
23. Как определить тип-сумму с использованием объединений (union)?
24. Как реализовать конкретные состояния алгебраических типов данных?
25. Что представляют собой особые типы суммы — необязательные значения (optional values)?
26. Как типы суммы помогают обрабатывать ошибки в программах?
27. Как моделировать предметную область с помощью алгебраических типов данных?
28. Какие преимущества даёт использование алгебраических типов данных перед простыми перечислениями?
29. Как выражать сложные бизнес-правила с помощью алгебраических типов данных?
30. Как сочетать алгебраические типы данных с другими структурами данных (списки, деревья) для эффективного решения прикладных задач?

Практическое занятие №4. Контейнеры и итераторы в библиотеке STL. Выполнение работ по индивидуальным заданиям.

Контрольные вопросы (ПК-3):

1. Что такое контейнеры в библиотеке STL и какие виды контейнеров существуют?
2. Как инициализировать и заполнить контейнеры (вектор, список, карта) элементами?
3. Как использовать итераторы для обхода элементов контейнера?
4. В чём разница между прямым и обратным итератором?
5. Как добавить и удалить элементы из контейнера vector и list?
6. Как отсортировать элементы контейнера set и map?
7. Как реализовать поиск элемента в контейнере с помощью алгоритмов STL?
8. Как выбрать подходящий контейнер для хранения большого количества данных?
9. Как эффективно копировать элементы из одного контейнера в другой?
10. Как обработать исключение при попытке обратиться к несуществующему элементу контейнера?

Практическое занятие №5. Вектор. Очереди. Стек. Список. Выполнение работ по индивидуальным заданиям.

Контрольные вопросы (ПК-3):

1. Что такое вектор в C++ и каковы его основные свойства?
2. Как добавить и удалить элементы из вектора в C++?
3. Что такое очередь и как она реализуется в C++?
4. Как использовать очередь для реализации алгоритма BFS (широкий поиск)?
5. Что такое стек и каковы его базовые операции?
6. Как реализовать стек с использованием стандартного контейнера C++?
7. Что такое список и каковы его преимущества перед вектором?
8. Как добавить элемент в начало и конец односвязного списка?
9. Как реализовать двунаправленный список в C++?
10. Как эффективно перебирать элементы в стеке, очереди и списке с помощью итераторов?

Практическое занятие №6. Ассоциативные массивы. Объекты-функции и предикаты.

Контрольные вопросы (ПК-3):

1. Что такое ассоциативный массив в C++ и каковы его основные характеристики?
2. Как создаются и используются контейнеры map и unordered_map в STL?
3. Как добавить, обновить и удалить элементы из ассоциативного массива?
4. Что такое объект-функция в C++ и как его реализовать?
5. Как использовать объекты-функции в качестве предикатов для сортировки и фильтрации данных?
6. Как создать собственный предикат для нестандартной сортировки элементов контейнера?
7. Как работают стандартные предикаты вроде less, greater, equal_to в C++?
8. Как использовать лямбда-функции в качестве предикатов в операциях с контейнерами?

9. Как эффективно осуществлять поиск по ключу в ассоциативных массивах?
10. Как выбирать между `map` и `unordered_map` в зависимости от требований приложения?

Лабораторная работа №4. Функции высших порядков. Композиции. Рекурсия.

Контрольные вопросы (ПК-3):

1. Что такое функции высшего порядка в C++ и как они реализуются?
2. Как передать функцию в качестве аргумента другой функции в C++?
3. Что такое композиция функций и как её реализовать в C++?
4. Как использовать лямбда-функции для реализации функций высшего порядка?
5. Что такое рекурсия и как она применяется в C++?
6. Как реализовать хвостовую рекурсию и почему она предпочтительнее обычной рекурсии?
7. Как избежать переполнения стека при глубокой рекурсии в C++?
8. Как использовать стандартные алгоритмы STL совместно с функциями высшего порядка?
9. Как эффективно комбинировать рекурсию и функции высшего порядка для решения задач?
10. Как протестировать корректность работы рекурсивных и композиционных функций в C++?

Лабораторная работа №5. Разработка библиотеки функций высших порядков. Составление и использование библиотек.

Контрольные вопросы (ПК-3):

1. Что такое функции высшего порядка и какова их роль в функциональном программировании?
2. Как реализовать функцию высшего порядка, принимающую другую функцию в качестве аргумента в C++?
3. Как создать универсальный интерфейс для функций высшего порядка в собственной библиотеке?
4. Как использовать лямбда-функции и `std::function` для реализации функций высшего порядка?
5. Как спроектировать собственную библиотеку функций высшего порядка с учётом повторного использования и расширения функционала?
6. Как реализовать композицию функций (*chain of functions*) в своей библиотеке?
7. Как обеспечить совместимость вашей библиотеки с существующими стандартными алгоритмами STL?
8. Как организовать тестирование и проверку работоспособности функций высшего порядка в разрабатываемой библиотеке?
9. Как оформить документацию и справочные материалы для пользователей вашей библиотеки функций высшего порядка?
10. Как оптимизировать производительность функций высшего порядка в вашей библиотеке для больших объёмов данных?

Лабораторная работа №6. Разработка и сборка проектов с использованием библиотек.

Контрольные вопросы (ПК-3):

1. Что такое библиотека в C++ и для чего она предназначена?
2. Как создать статическую библиотеку (`*.lib` или `*.a`) в C++?
3. Как создать динамическую библиотеку (`*.dll` или `*.so`) в C++?
4. Как подключить стороннюю библиотеку к проекту в среде разработки (например, Visual Studio или CLion)?
5. Как экспортировать функции и классы из библиотеки для дальнейшего использования?
6. Как настроить сборку проекта с использованием нескольких библиотек?
7. Как разрешить конфликты имен при подключении нескольких библиотек?
8. Как организовать структуру каталогов и файлов для удобной сборки проекта с библиотеками?
9. Как автоматизировать сборку проекта с библиотеками с помощью инструментов `make`, `CMake` или другого инструмента сборки?
10. Как упаковать готовую библиотеку вместе с документацией и примерами использования для распространения?

1.3.2 Вопросы, выносимые на зачет по дисциплине «Функциональное программирование» (ПК-3)

1. Что такое функциональное программирование и чем оно отличается от императивного подхода? От объектно-ориентированного?
2. Назовите три ключевых принципа функционального программирования.
3. Дайте определение чистой функции и объясните её важность в функциональном подходе.
4. В чём состоит различие между абстрактной и конкретной формами программы?
5. Что представляет собой проблема связывания значений и переменных в функциональном программировании?
6. Назовите пять преимуществ функционального программирования перед традиционным императивным стилем.
7. Каким образом достигается высокая читаемость и лаконичность кода в функциональном подходе?
8. Почему параллелизм и синхронизация проще решаются средствами функционального программирования?
9. Как аппаратное обеспечение современных компьютеров влияет на эффективность исполнения функциональных программ?
10. Какие существуют специализированные языки и среды для поддержки функционального стиля программирования?
11. Что такое концепция перегрузки функций и как она реализуется в C++?
12. В чём суть перегрузки стандартных функций и какие ограничения здесь имеются?
13. Приведите пример перегрузки оператора в C++ и объясните его пользу.
14. Как передать функцию в качестве параметра другому методу в C++?
15. В чём преимущество передачи функций как аргументов по сравнению с традиционными способами организации кода?
16. Что такое рекурсивная функция и как её эффективно использовать в функциональном стиле?
17. Что такое хвостовая рекурсия и как она связана с оптимизацией производительности?
18. Как компилятор реализует оптимизацию хвостового вызова в C++?
19. Какие встроенные средства C++ поддерживают функциональный стиль программирования?
20. Приведите пример реализации функционального подхода в стандартной библиотеке STL.
21. Что такое связный список и как он представляется в памяти компьютера?
22. Какие бывают типы связных списков (одно-, двунаправленные, кольцевые)?
23. Как выполняется вставка нового элемента в начало однонаправленного связного списка?
24. Как удалить элемент из середины связного списка?
25. Как реализовать операцию обращения связного списка (reverse)?
26. Как осуществить эффективный поиск элемента в связном списке?
27. Как добавить элемент в конец двусвязного списка?
28. Каковы преимущества и недостатки связных списков по сравнению с массивами?
29. Как изменить значение элемента в середине связного списка?
30. Как эффективно удалять элемент из начала связного списка?
31. Что такое неизменяемый вектор и как он отличается от обычного массива?
32. Как обновляют содержимое неизменяемых векторов?
33. Какие преимущества имеют неизменяемые векторы перед изменяемыми структурами данных?
34. Как реализуют эффективную вставку и удаление элементов в неизменяемых векторах?
35. Что такое префиксное дерево (trie) и для чего оно используется?
36. Как добавлять элементы в конец префиксного дерева?
37. Как искать элементы в префиксном дереве?
38. Как удаляется элемент из конца префиксного дерева?
39. Какие дополнительные операции возможны в префиксных деревьях помимо добавления и удаления?

40. Как оценивается общая эффективность операций в `trie` относительно других структур данных?
41. Что такое алгебраические типы данных и как они отличаются от примитивных типов?
42. Как определяется тип-сумма через наследование в языках программирования?
43. Как определить тип-сумму с использованием объединений (`union`)?
44. Как реализовать конкретные состояния алгебраических типов данных?
45. Что представляют собой особые типы суммы — необязательные значения (`optional values`)?
46. Как типы суммы помогают обрабатывать ошибки в программах?
47. Как моделировать предметную область с помощью алгебраических типов данных?
48. Какие преимущества даёт использование алгебраических типов данных перед простыми перечислениями?
49. Как выражать сложные бизнес-правила с помощью алгебраических типов данных?
50. Как сочетать алгебраические типы данных с другими структурами данных (списки, деревья) для эффективного решения прикладных задач?

2. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

2.1 Порядок и методика проведения текущего контроля и промежуточной аттестации

Предусмотрены следующие виды контроля:

- текущий контроль по каждому модулю – в форме написания контрольной работы для оценки теоретических знаний;
- текущий контроль по каждому модулю – в форме отчетов по лабораторным работам и практическим занятиям для оценки практических навыков;
- промежуточная аттестация по дисциплине – в форме зачета.

Текущий контроль успеваемости в форме защиты лабораторных и практических проводится по окончании каждой лабораторной работы (практического занятия) в форме индивидуального собеседования по выполненным исследованиям или расчетам. Проводится с целью контроля закрепления теоретической части материала и степени отработки студентом практических навыков исследования на аппаратуре.

Контрольные работы выполняются в виде короткого письменного ответа на один вопрос, изученный на предыдущей лекции в начале каждой последующей лекции. Ответ на вопрос дается в течение 5-10 минут. Таким образом, после лекционного курса каждого модуля формируется общая оценка за теоретические знания.

С целью повышения качества обучения за счет побуждения студентов к активной текущей учебной работе, четкого и оперативного контроля всего хода учебного процесса, снижения роли случайных и субъективных факторов при оценивании учебной деятельности студентов в образовательном процессе реализована модульно-рейтинговая система.

Правила ее использования прописаны в «Положении об МРС».

Набранные обучающимися в рамках текущего контроля баллы могут быть переведены в оценку:

- «не зачтено» - от 0 до 40 от максимального количества баллов;
- «зачтено» - от 41 до 100 максимального количества баллов.

Соотношения максимального количества баллов, полученных студентом по блокам модулей, показаны в Таблице.

Таблица - Распределение баллов по модулям дисциплины «Функциональное программирование»

Модуль	Всего баллов (Максимальное значение)	Теоретический блок (Контрольная работа)	Практический блок
Модуль 1	50	14	36
Модуль 2	50	14	36
Зачет	100		

Как правило, теоретический блок оценивается по результатам контрольной работы. Практический блок оценивается по результатам выполнения лабораторных работ и заданий на практических занятиях.

На зачете производится оценка тех компетенций, которые должны быть в той или иной форме освоены в процессе изучения. Рекомендуется формировать вопросы в билетах к зачету таким образом, чтобы преподаватель смог оценить все компетенции данной дисциплины.

2.2 Методика проведения зачета в группах очной и заочной форм

Зачет по дисциплине «Функциональное программирование» у студентов очной и заочной форм обучения проводится письменно по классической методике. Вопросы сгруппированы в билетах. В каждом билете содержится по 2-3 вопроса, скомпонованные таким образом, чтобы преподаватель смог оценить все компетенции данной дисциплины. Количество билетов должно быть не менее числа студентов в группе. Студенты готовят письменные ответы на вопросы.

3 Промежуточная аттестация

Промежуточная аттестация проводится в форме зачета.

Зачет проводится по расписанию экзаменационной сессии в письменном виде. Количество вопросов в билете к зачету – 2. Проверка ответов и объявление результатов производится в день зачета. Результаты аттестации заносятся в зачетную ведомость и зачетную книжку студента.

Студенты, не прошедшие промежуточную аттестацию по графику сессии, должны ликвидировать задолженность в установленном порядке.

Образец билета к зачету

	МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ Северо-Кавказский филиал ордена Трудового Красного Знамени федерального государственного бюджетного образовательного учреждения высшего образования «Московский технический университет связи и информатики»	Утверждаю Зав. кафедрой «ИВТ» _____ Соколов С.В. «___» _____ 20___ г.
Направление подготовки: 09.03.01. Информатика и вычислительная техника Курсы: 3 Дисциплина: Функциональное программирование		
Билет №1 1. Что такое рекурсивная функция и как её эффективно использовать в функциональном стиле? 2. Назовите три ключевых принципа функционального программирования. ст. преподаватель кафедры ИВТ _____ Осмоловская Н.С. «___» _____ 20___ г.		

Один комплект отпечатанных билетов к зачету, подписанных преподавателем кафедры и утвержденных заведующим кафедрой хранится у заведующего кафедрой, другой комплект – у преподавателя, ведущего дисциплину.

4 Тестовые задания для проведения оценки сформированности компетенций (ПК-3)

Данная дисциплина формирует у обучающегося ПК-3 компетенцию, объективно оценить которую можно при ответе студента на тест или при выполнении практического или лабораторного задания.

Один комплект тестовых вопросов с указанием правильных ответов хранится у заведующего кафедрой, другой комплект – у преподавателя, ведущего дисциплину.

Тестовые задания позволяют оценить уровень сформированности компетенции ПК-3.

Задания распределены по блокам в соответствии с уровнем сложности. Каждый блок содержит номер задания, и текст задания.

Базовый уровень содержит примерно 50 % заданий.

Он формируется из заданий с выбором одного или нескольких верных ответов из предложенных или воспроизведения фактического материала (терминология, факты, классификация, параметры и др.);

Повышенный уровень в среднем составляет 35 % заданий.

В нем используются задания на сопоставление, сравнение, установление последовательности;

Высокий уровень в среднем составляют – 15 % заданий.

В задании используются: решения нетиповых задач, алгоритмы, доказательства, задания с развернутым ответом – определения, перечисление, изображение схемы, структуры и др.

Количество заданий по компетенции равно 20.

Один комплект тестовых вопросов с указанием правильных ответов хранится у заведующего кафедрой, другой комплект – у преподавателя, ведущего дисциплину.

ПК-3: Способен разрабатывать и внедрять программное обеспечение с применением современных средств и технологий, осуществлять его поддержку.

БЛОК А (базовый уровень) – задание с выбором одного или нескольких верных ответов из предложенных.

Инструкция по тестам Блока А: Прочитайте текст и выберите один или несколько правильных ответов.

№ задания	Тексты заданий
1	Основная задача конструктора класса – это... 1. освобождение памяти, занятой объектом 2. инициализация объекта при его создании 3. определение типа объекта 4. преобразование объекта в другой тип
2	В чем основное отличие ссылки от указателя в C++? 1. ссылка не может быть перенаправлена на другой объект после инициализации 2. указатель не может быть константным 3. для ссылки не требуется оператор разыменования 4. ссылка может хранить значение nullptr
3	Какой оператор используется для доступа к статическим членам класса через его имя (без создания объекта)? 1. : 2. -> 3. :: 4. .

№ задания	Тексты заданий
4	Какое объявление корректно создаёт ссылку на целое число? 1. <code>int& ref = 5;</code> 2. <code>int x = 5; int& ref = x;</code> 3. <code>int* ref = new int(5);</code> 4. <code>int ref& = 10;</code>
5	Какое ключевое слово используется для объявления пространства имён? 1. <code>class</code> 2. <code>scope</code> 3. <code>using</code> 4. <code>namespace</code>
6	Какой из способов передачи параметра позволяет изменять исходную переменную в вызывающей функции? 1. <code>void foo(int x)</code> 2. <code>void foo(const int& x)</code> 3. <code>void foo(int& x)</code> 4. <code>void foo(int&& x)</code>
7	Объявление <code>int const* p</code> означает... 1. константный указатель на <code>int</code> (значение <code>p</code> нельзя изменить) 2. указатель на константный <code>int</code> (значение по адресу <code>p</code> нельзя изменить) 3. константный указатель на константный <code>int</code> 4. указатель на указатель типа <code>int</code>
8	Как правильно объявить константный метод, который гарантирует, что не будет изменять поля класса? 1. <code>void method() const;</code> 2. <code>void const method();</code> 3. <code>void method() static;</code> 4. <code>const void method();</code>
9	Конструктор копирования вызывается при... 1. создании нового объекта с помощью оператора <code>new</code> 2. присваивании одного объекта другому 3. передаче объекта в функцию по значению 4. явном вызове деструктора
10	Указатель <code>this</code> внутри метода класса содержит... 1. адрес таблицы виртуальных методов класса 2. адрес объекта, для которого вызван метод 3. адрес статического поля класса 4. адрес родительского класса в иерархии наследования

БЛОК Б (повышенный уровень) – задание закрытого типа на установление соответствия
Инструкция: Прочитайте текст и установите соответствие.

№ задания	Тексты заданий
11	Установите соответствие между модификатором доступа и его особенностью: 1. <code>Public</code> 2. <code>Private</code> 3. <code>Protected</code> А) Ограничивает доступность атрибутов, оставляя детализацию внутренней реализации скрытым

№ задания	Тексты заданий
	Б) Позволяет контролировать доступ к внутренним аспектам родительского класса, открывая доступ только подклассам. В) Позволяют внешнему коду взаимодействовать с объектом
12	Установите правильное соответствие между определениями ООП и их описаниями: 1. Агрегация 2. Наследование 3. Полиморфизм 4. Инкапсуляция А) сокрытие внутренних деталей реализации объекта и предоставление контролируемого внешнего интерфейса для взаимодействия с ним. Б) разрешение одному классу использовать переменные и методы другого класса как свои собственные. В) отношение между двумя классами, при котором один класс содержит экземпляр другого класса Г) возможность объектов с одинаковым интерфейсом демонстрировать разное поведение в зависимости от их типа.
13	Установите соответствие между языком программирования и его преимуществом: 1. C++ 2. Java 3. Ruby 4. PHP А) славится своей лаконичной и ясной структурой, что ускоряет разработку и повышает читаемость кода. Б) компилируемый язык, что позволяет добиться высокой скорости выполнения программ, близких к возможностям языков низкого уровня. В) Сборщик мусора освобождает разработчика от ручной очистки памяти, снижая риск утечек памяти и прочих проблем. Г) Большинство веб-хостингов его поддерживают, что облегчает развертывание и администрирование веб-приложений.
14	Установите соответствие между конструктором и его действиями: 1. Конструктор объектов 2. Конструктор копирования 3. Конструктор по умолчанию А) не принимает аргументов и создает объект с начальными значениями по умолчанию. Б) активируется, когда объект передается по значению или возвращается из функции. В) устанавливает начальные значения полей класса и подготавливает объект к дальнейшей работе.

БЛОК Б (повышенный уровень) – задание закрытого типа на установление последовательности.

Инструкция: Прочитайте текст и установите правильную последовательность.

№ задания	Тексты заданий
15	Расположите операторы согласно их приоритета от низшего к высшему: 1. a++ 2. a+b 3. &&

№ задания	Тексты заданий
	4. :: 5. = 6. ++a
16	Установите правильную последовательность этапов разрешения проблемы ромбовидного наследования в языке C++: 1. Используйте виртуальное наследование для устранения дублирования общей базы. 2. Определите порядок вызова конструкторов базовых классов. 3. Устраните неоднозначность путем явного указания пути вызова методом. 4. Проверьте, какая версия метода будет вызвана при обращении к методу общего предка. 5. Решите, как поступить с конфликтующими членами (методами или полями) из разных ветвей наследования.
17	Установите правильную последовательность пунктов при написании программы на C++: 1. Определение основной функции main() 2. Заголовочный комментарий и директива include 3. Прототипы функций и объявлений классов 4. Реализация классов и методов 5. Тестирование и отладка 6. Подключение нужных библиотек 7. Определение класса и его методов

БЛОК В (высокий уровень) – задание открытого типа с развернутым ответом.
Инструкция: Прочитайте текст и запишите обоснованный ответ.

№ задания	Тексты заданий
18	Программа должна создать динамический массив целых чисел. Какая строка пропущена? <pre> #include <iostream> int main() { int size = 5; int* arr = new int[size]; for (int i = 0; i < size; ++i) { arr[i] = i * 10; } // ... использование массива ... // <i>Пропущена строка</i> return 0; } </pre>
19	Специальная функция в C++, которая автоматически вызывается при создании объекта этого класса – это _____ класса.
20	Дружественные функции – это функции, которые не являются членами класса, однако имеют доступ к его закрытым членам – переменным и функциям, которые имеют спецификатор _____.

Система оценивания выполнения тестовых заданий по компетенции ПК-3.

Номер задания	Указания по оцениванию	Результат оценивания (баллы, полученные за выполнение задания/характеристика правильности ответа)
Задания 1-10	Задания с выбором одного или нескольких верных ответов из предложенных считается верным, если правильно указана цифра или цифры, означающие верные ответы.	Совпадение с верным ответом оценивается 1 баллом; неверный ответ или его отсутствие - 0 баллов.
Задания 11-14	Задания закрытого типа на установление соответствия считается верным, если правильно установлены все соответствия (позиции из одного столбца верно сопоставлены с позициями другого)	Полное совпадение с верным ответом оценивается 1 баллом; неверный ответ или его отсутствие - 0 баллов.
Задания 15-17	Задания закрытого типа на установление последовательности считается верным, если правильно указана вся последовательность цифр	Полное совпадение с верным ответом оценивается 1 баллом; если допущены ошибки или ответ отсутствует - 0 баллов.
Задания 18-20	Задание открытого типа с развернутым ответом считается верным, если ответ совпадает с эталонным по содержанию и полноте.	Полный правильный ответ на задание оценивается 3 баллами; если допущена одна ошибка/неточность/ответ правильный, но не полный - 1 балл, если допущено более одной ошибки/ответ неправильный/ ответ отсутствует- 0 баллов.

Методика проведения тестирования

Максимальное время выполнения теста - 60 минут.

Количество тестовых заданий в тесте не менее 60.

Для проведения контроля остаточных знаний компьютер формирует в случайном порядке 60 тестовых заданий.

При прохождении тестового контроля знаний студент должен выбрать один или несколько правильных ответов в каждом тестовом задании.

В случае правильного ответа на 49 и более тестовых заданий (81-100% правильных ответов) студент подтверждает освоение компетенций с уровнем «отлично». При ответе на 48-37 тестовых заданий (61-80% правильных ответов) студент подтверждает освоение компетенций с уровнем «хорошо». При ответе на 36-25 тестовых заданий (41-60% правильных ответов) студент подтверждает освоение компетенций с уровнем «удовлетворительно». При ответе менее чем на 25 тестовых заданий (0-40% правильных ответов) студент не подтверждает необходимый уровень знаний и оценивается «неудовлетворительно».

Методические рекомендации по освоению дисциплины

1. Методические рекомендации преподавателю

Дисциплина «Функциональное программирование» включает в себя:

- лекционные занятия – 14 часов;
- практические занятия – 14 часов;
- лабораторные работы – 28 часов;
- промежуточную аттестацию – зачет.

Перед началом изучения дисциплины преподаватель должен ознакомить студентов с рабочей программой и оценочными материалами по дисциплине, с видами учебной и самостоятельной работы, перечнем литературы и интернет-ресурсов, с формами текущей и промежуточной аттестации, с критериями оценки качества знаний для итоговой оценки по дисциплине.

При проведении лекций, преподаватель:

- 1) формулирует тему и цель занятия, объявляет учебные вопросы;
- 2) излагает основные теоретические положения;
- 3) с помощью технических средств обучения и/или под запись дает определения основных понятий, расчетных формул;
- 4) проводит примеры из отечественного и зарубежного опыта, дает текущие статистические данные для наглядного и образного представления изучаемого материала;
- 5) в конце занятия выдает вопросы для самостоятельного изучения.

На занятиях лабораторного цикла следует обратить внимание на соответствие выбираемых студентом средств выполнения решаемым в работе задачам.

Каждая лабораторная работа должна быть оформлена и защищена в соответствии с требованиями. Защита производится после оформления отчета по работе.

Во время выполнения заданий в учебной аудитории студент может консультироваться с преподавателем, определять наиболее эффективные методы решения поставленных задач. Если какая-то часть задания остается не выполненной, студент может продолжить её выполнение во время внеаудиторной самостоятельной работы.

Для оценки полученных знаний и освоения учебного материала по каждому разделу и в целом по дисциплине преподаватель использует формы текущего контроля и промежуточной аттестации знаний обучающихся.

2. Методические рекомендации студентам

Приступая к изучению новой учебной дисциплины, студенты должны ознакомиться с рабочей программой, оценочными материалами, учебной, научной и методической литературой, имеющейся в ЭИОС СКФ МТУСИ,

Глубина усвоения дисциплины зависит от активной и систематической работы студента на лекциях и практических занятиях, а также в ходе самостоятельной работы, по изучению рекомендованной литературы.

2.1 Методические указания для обучающихся по лекционным занятиям

Важно сосредоточить внимание на содержании лекции. Это поможет лучше воспринимать учебный материал и уяснить взаимосвязь проблем по всей дисциплине.

Основное содержание лекции целесообразнее записывать в тетради в виде ключевых фраз, понятий, тезисов, обобщений, схем, опорных выводов. Необходимо обращать внимание на термины, формулировки, раскрывающие содержание тех или иных явлений и процессов, научные выводы и практические рекомендации. Желательно оставлять в конспектах поля, на которых делать пометки из рекомендованной литературы, дополняющей материал прослушанной лекции, а также подчеркивающие особую важность тех или иных теоретических положений.

С целью уяснения теоретических положений, разрешения спорных ситуаций необходимо задавать преподавателю уточняющие вопросы. Для закрепления содержания лекции в памяти, необходимо во время самостоятельной работы внимательно прочесть свой конспект и дополнить его записями из учебников и рекомендованной литературы.

Конспектирование читаемых лекций и их последующая доработка способствует более глубокому усвоению знаний, и поэтому являются важной формой учебной деятельности студентов.

2.2 Методические указания для обучающихся по подготовке к лабораторным работам

Лабораторные занятия по любой учебной дисциплине проводятся в соответствии с рабочей программой этой учебной дисциплины в установленные расписанием часы.

Инструктаж по технике безопасности, правилам пожарной безопасности и правилам внутреннего распорядка в лаборатории проводится преподавателем на первом лабораторном занятии. На этом же занятии студентам сообщаются:

- программа всего предстоящего лабораторного цикла;
- условия взаимодействия студентов с преподавателем в процессе выполнения лабораторных работ;
- условия контроля самостоятельной работы студентов, включая правила оформления отчетов по лабораторным работам и их последующей защиты;
- другая необходимая информация.

Организация лабораторных работ включает:

- самостоятельную внеаудиторную подготовку студента к выполнению каждой отдельной лабораторной работы в соответствии с ее программой в рамках часов, выделенных на самостоятельную проработку материала в программе дисциплины;
- входной контроль преподавателем степени подготовленности каждого студента к выполнению лабораторных работ;
- выполнение программы лабораторных работ в полном объеме;
- оформление отчета и его защиту каждым студентом в установленные сроки;
- формирование преподавателем рейтингов каждого из студентов по результатам выполнения и защиты им отдельных лабораторных работ и их циклов (используется Модульно-рейтинговая система).

Оценка качества выполнения лабораторных работ каждым студентом производится преподавателем отдельно за подготовку к работе, ее выполнение и защиту.

2.3 Методические указания для обучающихся по подготовке к практическим занятиям

Целью практических занятий является закрепление теоретических знаний, полученных при изучении дисциплины.

При подготовке к практическому занятию целесообразно выполнить следующие рекомендации:

- изучить основную литературу; ознакомиться с дополнительной литературой, новыми публикациями в периодических изданиях: журналах, газетах и т.д.;
- при необходимости доработать конспект лекций. При этом учесть рекомендации преподавателя и требования учебной программы.

При выполнении практических занятий основным методом обучения является самостоятельная работа студента под управлением преподавателя. На них пополняются теоретические знания студентов, их умение творчески мыслить, анализировать, обобщать изученный материал, проверяется отношение студентов к будущей профессиональной деятельности.

Оценка выполненной работы осуществляется преподавателем комплексно: по результатам выполнения заданий, устному сообщению и оформлению работы.

После подведения итогов занятия студент обязан устранить недостатки, отмеченные преподавателем при оценке его работы.

Методические указания по выполнению лабораторных и практических работ по дисциплине «Функциональное программирование» для студентов очной формы обучения, размещенные на сайте СКФ МТУСИ в разделе «Методические материалы»:

3. Методические рекомендации для обучающихся по самостоятельной работе

Самостоятельная работа студентов является составной частью учебной работы и имеет целью закрепление и углубление полученных знаний и навыков, поиск и приобретение новых знаний, в том числе с использованием автоматизированных обучающих курсов (систем), а также выполнение учебных заданий, подготовку к предстоящим занятиям, зачетам и экзаменам.

Самостоятельная работа во внеаудиторное время включает в себя:

- повторение лекционного материала;
- подготовки к лабораторным работам (практическим занятиям);
- изучения учебной и научной литературы;
- решения задач, выданных на практических занятиях;
- подготовки к тестированию;
- выделение наиболее сложных и проблемных вопросов по изучаемой теме, получение разъяснений и рекомендаций по данным вопросам с преподавателями кафедры на их еженедельных консультациях.
- проведение самоконтроля путем ответов на вопросы текущего контроля знаний, решения представленных в учебно-методических материалах кафедры задач, тестов, написания рефератов по отдельным вопросам изучаемой дисциплины.

Перед выполнением внеаудиторной самостоятельной работы преподаватель проводит инструктаж (консультацию) с определением цели задания, его содержания, сроков выполнения, основных требований к результатам работы, критериев оценки, форм контроля и перечня источников и литературы.

Обычно постановку задачи обучаемым на проведение самостоятельной работы преподаватель осуществляет на одном из занятий, предшествующем данному.

Методику самостоятельной работы все обучаемые выбирают индивидуально.