Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Самарский государственный технический университет»
(ФГБОУ ВО «СамГТУ»)
Филиал ФГБОУ ВО «СамГТУ» в г. Белебее Республики Башкортостан

УТВЕРЖДАЮ
Директор филиала ФГБОУ ВО «СамГТУ»
в г. Белебее Республики Башкортостан
_____ Л.М. Инаходова
25.05.2023 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

Б1.О.03.02 «Технологии программирования»

Код и направление подготовки (специальность)	**09.03.02 Информационные системы и технологии**
Направленность (профиль)	**Информационные системы и технологии**
Квалификация	**Бакалавр**
Форма обучения	**Заочная**
Год начала подготовки	**2023**
Выпускающая кафедра	**Инженерные технологии**
Кафедра-разработчик	**Инженерные технологии**
Объем дисциплины, ч. / з.е.	**180 / 5**
Форма контроля (промежуточная аттестация)	**Экзамен**

Белебей 2023 г.

Рабочая программа дисциплины (далее – РПД) разработана в соответствии с требованиями ФГОС ВО по направлению подготовки (специальности) 09.03.02 «Информационные системы и технологии», утвержденного приказом Министерства образования и науки РФ от 19 сентября 2017 г. № 926, и соответствующего учебного плана.

Разработчик РПД:

доцент, к.т.н., доцент		З.Ф. Камальдинова
(должность, степень, ученое звание)	(подпись)	(ФИО)

РПД рассмотрена и одобрена на заседании кафедры 25.05.2023 г., протокол № 6.

Заведующий кафедрой	к.т.н., доцент	А.А. Цынаева
	(степень, ученое звание, подпись)	(ФИО)

СОГЛАСОВАНО:

Руководитель образовательной программы	доцент, к.т.н.	З.Ф. Камальдинова
	(степень, ученое звание, подпись)	(ФИО)

СОДЕРЖАНИЕ

1. Перечень планируемых результатов обучения по дисциплине (модулю), соотнесенных с планируемыми результатами освоения образовательной программ

Универсальные компетенции

Таблица 1

Наименование категории (группы) компетенций	Код компетенции	Наименование компетенции	Код и наименование индикатора достижения компетенции	Результаты обучения
не предусмотрены учебным планом				

Общепрофессиональные компетенции

Таблица 2

Код компетенции	Наименование компетенции	Код и наименование индикатора достижения компетенции	Результаты обучения
ОПК-2	Способен понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности	**ОПК-2.1** Использует и понимает принципы работы информационных технологий и программных средств при решении задач в сфере информационных систем и технологий	**З2 ОПК-2.1** Знать: теорию программно-аппаратного обеспечения информационных систем, автоматизирующих задачи организационного управления и бизнес-процессы в организациях различных форм собственности с целью повышения эффективности деятельности организаций.
		ОПК-2.2 Применяет современные информационные технологии и программные средства отечественного производства при решении задач в сфере информационных систем и технологий	**У2 ОПК-2.2** Уметь: выбирать и применять программно-аппаратное обеспечение информационных систем, в том числе отечественного производства при решении задач профессиональной деятельности **В2 ОПК-2.2** Владеть: методами и средствами обеспечения работы программно-аппаратного обеспечения информационных систем, в том числе отечественного производства, при решении задач профессиональной деятельности
ОПК-6	Способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий;	**ОПК-6.1** Разрабатывает алгоритмы и программы, пригодные для практического применения	**З1 ОПК-6.1** Знать: основы алгоритмизации, языки программирования, методы реализации алгоритмов, операционные системы и оболочки, современные программные среды разработки информационных систем и технологий
		ОПК-6.3 Ведет и использует базы данных и информационные хранилища	**У1 ОПК-6.3** Уметь: применять языки программирования, работать с базами данных, разрабатывать информационные системы и технологии для автоматизации бизнес-процессов, решать прикладные задачи различных классов

Профессиональные компетенции

Таблица 3

Код компетенции	Наименование компетенции	Код и наименование индикатора достижения компетенции	Результаты обучения
не предусмотрены учебным планом			

2. Место дисциплины (модуля) в структуре образовательной программы

Место дисциплины в структуре образовательной программы: обязательная часть.

Таблица 4

Код компетенции	Предшествующие дисциплины	Параллельно осваиваемые дисциплины	Последующие дисциплины
ОПК-2	Информационные	Управление данными	Методы и средства проектирования

	технологии и программирование; Языки и методы программирования; Учебная практика: ознакомительная практика		информационных систем и технологий
ОПК-6	Языки и методы программирования; Системы искусственного интеллекта	Управление данными	Методы и средства проектирования информационных систем и технологий

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

Таблица 5

Вид учебной работы	Всего часов	Курс 3
Аудиторная контактная работа (всего), в том числе:	**12**	**12**
лекционные занятия (ЛЗ)	4	4
лабораторные работы (ЛР)	8	8
практические занятия (ПЗ)	0	0
Внеаудиторная контактная работа, КСР	**5**	**5**
Самостоятельная работа (всего), в том числе:	**154**	**154**
подготовка к лабораторным работам	77	77
самостоятельное изучение материала	77	77
Формы текущего контроля успеваемости	Вопросы к устному опросу, тестовые задания	Вопросы к устному опросу, тестовые задания
Формы промежуточной аттестации	**экзамен**	**экзамен**
Контроль	**9**	**9**
ИТОГО: час.	**180**	**180**
ИТОГО: з.е.	**5**	**5**

4. Содержание дисциплины, структурированное по темам (разделам), с указанием отведенного на них количества академических часов и видов учебных занятий

Таблица 6

№ раздела	Наименование раздела дисциплины	Виды учебной нагрузки и их трудоемкость, часы						
		ЛЗ	ЛР	ПЗ	СРС	КСР	Конт-роль	Всего часов
1	Введение	2	4	-	77	2	5	90
2	Технология программирования	2	4	-	77	3	4	90
Итого:		**4**	**8**	**0**	**154**	**5**	**9**	**180**

4.1. Содержание лекционных занятий

Таблица 7

№ ЛЗ	Наименование раздела	Тема лекции	Содержание лекции (перечень дидактических единиц: рассматриваемых подтем, вопросов)	Кол-во часов
Курс 3				
1	Введение	Развитие технологии программирования	Обзор основных этапов развития технологии программирования: «стихийное» программирование, структурный подход, объектный подход к программированию, компонентный подход и CASE-технологии. Рейтинг языков программирования.	2
2	Технология программирования	Некоторые специальные средства C++ Расширенные возможности работы с функциями.	Объединения. Перечисляемые типы. Ссылочные переменные. Указатели на функцию. Аргументы по умолчанию. Подставляемые функции. Перегрузка функций. Шаблоны функций. Перегрузка операторов.	2
Итого за курс:				**4**
Итого:				**4**

4.2. Содержание лабораторных занятий

Таблица 8

№ ЛР	Наименование раздела	Наименование лабораторной работы	Содержание лабораторной работы (перечень дидактических единиц: рассматриваемых подтем, вопросов)	Кол-во часов
Курс 3				
1	Введение	Основные понятия	Установка среды программирования на ПК, основные режимы работы, запуск программы. Выполнение программы, правила оформления отчетов, как сдавать лабораторные работы	2
2	Введение	Документирование и стандартизация	Понятие технического задания. Стили оформления листинга. Состав документации.	2
3	Технология программирования	Оформление листинга и блок-схемы. Отработка навыков чтения листингов программ.	Оформление блок-схемы и листинга индивидуального задания с использованием функций и процедур. Отчет по ЛЗ. По листингу программы восстановить укрупненный алгоритм, найти в листинге фрагмент, реализующий заданный элемент, показать в листинге интересный фрагмент, который можно использовать в дальнейшей работе. Отчет по ЛЗ	2
4	Технология программирования	Объединения и перечисляемые типы	Разработка программы с использованием перечисляемых типов и объединений.	2
			Итого за курс:	**8**
			Итого:	**8**

4.3. Содержание практических занятий

Таблица 9

№ ПЗ	Наименование раздела	Тема практического занятия	Содержание практического занятия (перечень дидактических единиц: рассматриваемых подтем, вопросов)	Кол-во часов
не предусмотрены учебным планом				

4.4. Содержание самостоятельной работы

Таблица 10

№ п/п	Наименование раздела	Вид самостоятельной работы	Содержание самостоятельной работы (перечень дидактических единиц: рассматриваемых подтем, вопросов)	Кол-во часов
Курс 3				
1	Введение	самостоятельное изучение материала	Обзор современного состояние языков программирования	77
2	Введение	подготовка к лабораторным занятиям	Установка среды программирования на персональном компьютере для выполнения лабораторных работ	
3	Технология программирования	самостоятельное изучение материала	Синтаксис и семантика языка программирования	77
4	Технология программирования	подготовка к лабораторным занятиям	Подготовка к лабораторным работам, выполнение заданий	
			Итого за курс:	**154**
			Итого:	**154**

5. Методические указания для обучающихся по освоению дисциплины (модуля)

Методические указания при работе на лекции

До лекции студент должен просмотреть учебно-методическую и научную литературу по теме лекции для того, чтобы иметь представление о проблемах, которые будут подняты в лекции.

Перед началом лекции обучающимся сообщается тема лекции, план, вопросы, подлежащие рассмотрению, доводятся основные литературные источники. Весь учебный материал, сообщаемый преподавателем, должен не просто прослушиваться. Он должен быть активно воспринят, т. е. услышан, осмыслен, понят, зафиксирован на бумаге и закреплен в памяти. Приступая к слушанию нового учебного материала, полезно мысленно установить его связь с ранее изученным. Следя за техникой чтения лекции (акцент на существенном, повышение тона, изменение ритма, пауза и т. п.), необходимо вслед за преподавателем уметь выделять основные категории, законы и определять их содержание, проблемы, предполагать их возможные решения, доказательства и выводы. Осуществляя такую работу, можно значительно облегчить себе понимание учебного материала, его конспектирование и дальнейшее изучение.

Методические указания при работе на лабораторном занятии

Проведение лабораторной работы делится на две условные части: теоретическую и практическую.

Необходимыми структурными элементами занятия являются проведение лабораторной работы, проверка усвоенного материала, включающая обсуждение теоретических основ выполняемой работы.

Перед лабораторной работой, как правило, проводится технико-теоретический инструктаж по использованию необходимого оборудования. Преподаватель корректирует деятельность обучающегося в процессе выполнения работы (при необходимости). После завершения лабораторной работы подводятся итоги, обсуждаются результаты деятельности.

Возможны следующие формы организации лабораторных работ: фронтальная, групповая и индивидуальная. При фронтальной форме однотипная работа выполняется всеми обучающимися одновременно. При групповой форме работа выполняется группой (командой). При индивидуальной форме обучающимися выполняются индивидуальные работы.

По каждой лабораторной работе имеются методические указания по их выполнению, включающие необходимый теоретический и практический материал, содержащие элементы и последовательную инструкцию по проведению выбранной работы, индивидуальные варианты заданий, требования и форму отчетности по данной работе.

Методические указания по самостоятельной работе

Организация самостоятельной работы обучающихся ориентируется на активные методы овладения знаниями, развитие творческих способностей, переход от поточного к индивидуализированному обучению с учетом потребностей и возможностей обучающегося.

Самостоятельная работа с учебниками, учебными пособиями, научной, справочной литературой, материалами периодических изданий и Интернета является наиболее эффективным методом получения дополнительных знаний, позволяет значительно активизировать процесс овладения информацией, способствует более глубокому усвоению изучаемого материала. Все новые понятия по изучаемой теме необходимо выучить наизусть.

Самостоятельная работа реализуется:

- непосредственно в процессе аудиторных занятий;
- на лекциях, практических занятиях;
- в контакте с преподавателем вне рамок расписания;
- на консультациях по учебным вопросам, в ходе творческих контактов, при ликвидации задолженностей, при выполнении индивидуальных заданий и т. д.;
- в методическом кабинете, дома, на кафедре при выполнении обучающимся учебных и практических задач.

Эффективным средством осуществления обучающимся самостоятельной работы является электронная информационно-образовательная среда университета, которая обеспечивает доступ к учебным планам, рабочим программам дисциплин (модулей), практик, к изданиям электронных библиотечных систем.

Методические указания по подготовке к устному опросу

Самостоятельная работа обучающихся включает подготовку к устному опросу на семинарских занятиях. Для этого обучающийся изучает лекции, основную и дополнительную литературу, публикации, информацию из Интернет-ресурсов. Темы и вопросы к семинарским занятиям, вопросы для самоконтроля доводятся до обучающихся заранее. Эффективность подготовки обучающихся к устному опросу зависит от качества ознакомления с рекомендованной литературой. Для подготовки к устному опросу необходимо ознакомиться с материалом по теме семинар и обратить внимание на усвоение основных понятий изучаемой темы, выявить неясные вопросы и подобрать дополнительную литературу для их освещения, составить тезисы выступления по отдельным проблемным аспектам. В среднем, подготовка к устному опросу по одному семинарскому занятию занимает от 2 до 4 часов.

Методические указания по подготовке к тестированию

Тестовые задания – система стандартизированных заданий, позволяющая автоматизировать процедуру измерения уровня знаний и умений обучающегося.

Успешное выполнение тестовых заданий является необходимым условием итоговой положительной оценки. Выполнение тестовых заданий предоставляет обучающимся возможность самостоятельно контролировать уровень своих знаний, обнаруживать пробелы в знаниях и принимать меры по их ликвидации. Форма изложения тестовых заданий позволяет закрепить и восстановить в памяти пройденный материал. Тестовые задания охватывают основные вопросы по изучаемой теме. Для формирования заданий использована как закрытая, так и открытая форма. У обучающегося есть возможность выбора правильного ответа или нескольких правильных ответов из числа предложенных вариантов. Для выполнения тестовых заданий обучающиеся должны изучить лекционный материал по теме, соответствующие разделы литературы по дисциплине. Контрольный тест выполняется обучающимся самостоятельно во время практических занятий.

6. Перечень учебной литературы и учебно-методического обеспечения для самостоятельной работы

Таблица 11

№ п/п	Автор(ы), наименование, место, год издания (если есть, указать «гриф»)	Книжный фонд (КФ) или	Литература	
			учебная	для самост.

		электрон. ресурс (ЭР)		работы
1.	Русанова Я.М., Чердынцева М.И. *С++ как второй язык в обучении приемам и технологиям программирования; Издательство Южного федерального университета, 2010.- Режим доступа* https://elib.samgtu.ru/getinfo?uid=els_samgtu||iprbooks||47120	ЭР	+	
2.	Фридман А.Л. Язык программирования С++; Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021.- Режим доступа: https://elib.samgtu.ru/getinfo?uid=els_samgtu||iprbooks||102076	ЭР	+	
3.	Ковшов В.И. Технология программирования : лаб.практикум / В. И. Ковшов; Самар.гос.техн.ун-т, Вычислительная техника .- 2-е изд..- Самара, 2013.- 58 с..- Режим доступа: https://elib.samgtu.ru/getinfo?uid=els_samgtu||elib||891	ЭР	+	
4.	Кулямин В.В. Технологии программирования. Компонентный подход; Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021.- Режим доступа: https://elib.samgtu.ru/getinfo?uid=els_samgtu||iprbooks||102071	ЭР		+
5.	Терехов А.Н. Технология программирования; Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020.- Режим доступа: https://elib.samgtu.ru/getinfo?uid=els_samgtu||iprbooks||97587	ЭР		+
6.	Технологии программирования: практикум / Мишова В.В., Кемеровский государственный институт культуры: 2016.- Режим доступа: https://elib.samgtu.ru/getinfo?uid=els_samgtu||iprbooks||66371	ЭР		+
7.	Современные технологии программирования: практикум / Зайцев М.Г., Сибирский государственный университет телекоммуникаций и информатики: 2008.- Режим доступа: https://elib.samgtu.ru/getinfo?uid=els_samgtu||iprbooks||55460	ЭР		+

Доступ обучающихся к ЭР НТБ СамГТУ (elib.samgtu.ru) осуществляется посредством электронной информационной образовательной среды университета и сайта НТБ СамГТУ по логину и паролю.

7. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень программного обеспечения

При проведении лекционных занятий используется мультимедийное оборудование. Организовано взаимодействие обучающегося и преподавателя с использованием электронной информационной образовательной среды университета.

Программное обеспечение

Таблица 12

№ п/п	Название	Способ распространения (лицензионное или свободно распространяемое)	Правообладатель (производитель)	Страна происхождения (иностранное или отечественное)
1.	Пакет офисных программ LibreOffice	свободно распространяемое	The Document Foundation	иностранное
2.	Пакет офисных программ Microsoft Office	лицензионное	Microsoft	иностранное
3.	Adobe Reader	свободно распространяемое	Adobe Systems Incorporated	иностранное
4.	Справочно-правовая система «Консультант Плюс»	лицензионное	НПО «ВМИ»	отечественное
5.	Антивирус Касперского	лицензионное	Лаборатория Касперского	отечественное
6.	Операционная система Microsoft Windows	лицензионное	Microsoft	иностранное
7.	Операционная система семейства Unix	свободно распространяемое	The Linux Foundation	иностранное
8.	Яндекс.Браузер	свободно распространяемое	Яндекс	отечественное

8. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», профессиональных баз данных, информационно-справочных систем

Таблица 13

№ п/п	Наименование	Краткое описание	Режим доступа
1	Электронно-библиотечная система IPRbooks	Электронно-библиотечная система	http://www.iprbookshop.ru/
2	Электронно-библиотечная система СамГТУ	Электронная библиотека СамГТУ	https://elib.samgtu.ru/
3	eLIBRARY.RU	Научная электронная библиотека	http://www.elibrary.ru/

9. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине

Лекционные занятия

Аудитории для лекционных занятий укомплектованы мебелью и техническими средствами обучения, служащими для представления учебной информации большой аудитории (наборы демонстрационного оборудования (проектор, экран, компьютер/ноутбук).

Лабораторные занятия

Для лабораторных занятий используется аудитория, оснащенная компьютерной техникой с возможностью подключения к сети «Интернет» и доступом к электронной информационно-образовательной среде СамГТУ**.**

Самостоятельная работа

Помещения для самостоятельной работы оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и доступом к электронной информационно-образовательной среде СамГТУ:

методический кабинет (ауд. 9);

компьютерные классы (ауд. 6, 15).

10. Фонд оценочных средств по дисциплине

Фонд оценочных средств для проведения текущего контроля успеваемости и промежуточной аттестации представлен в Приложении 1.

Полный комплект контрольных заданий или иных материалов, необходимых для оценивания результатов обучения по дисциплине, практике хранится на кафедре-разработчике в бумажном и электронном виде.

Приложение 1

Фонд оценочных средств для проведения текущего контроля успеваемости и промежуточной аттестации

по дисциплине

Б1.О.03.02 «Технологии программирования»

Код и направление подготовки (специальность)	**09.03.02 Информационные системы и технологии**
Направленность (профиль)	**Информационные системы и технологии**
Квалификация	**бакалавр**
Форма обучения	**заочная**
Год начала подготовки	**2023**
Выпускающая кафедра	**Инженерные технологии**
Кафедра-разработчик	**Инженерные технологии**
Объем дисциплины, ч. / з.е.	**180 / 5**
Форма контроля (промежуточная аттестация)	**экзамен**

1. **Перечень компетенций, индикаторов достижения компетенций и признаков проявления компетенций (дескрипторов), которыми должен овладеть обучающийся в ходе освоения образовательной программы**

Универсальные компетенции

Таблица 1

Наименование категории (группы) компетенций	Код компетенции	Наименование компетенции	Код и наименование индикатора достижения компетенции	Результаты обучения
не предусмотрены учебным планом				

Общепрофессиональные компетенции

Таблица 2

Код компетенции	Наименование компетенции	Код и наименование индикатора достижения компетенции	Результаты обучения
ОПК-2	Способен понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности	**ОПК-2.1** Использует и понимает принципы работы информационных технологий и программных средств при решении задач в сфере информационных систем и технологий	**З2 ОПК-2.1** Знать: теорию программно-аппаратного обеспечения информационных систем, автоматизирующих задачи организационного управления и бизнес-процессы в организациях различных форм собственности с целью повышения эффективности деятельности организаций.
		ОПК-2.2 Применяет современные информационные технологии и программные средства отечественного производства при решении задач в сфере информационных систем и технологий	**У2 ОПК-2.2** Уметь: выбирать и применять программно-аппаратное обеспечение информационных систем, в том числе отечественного производства при решении задач профессиональной деятельности **В2 ОПК-2.2** Владеть: методами и средствами обеспечения работы программно-аппаратного обеспечения информационных систем, в том числе отечественного производства, при решении задач профессиональной деятельности
ОПК-6	Способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий;	**ОПК-6.1** Разрабатывает алгоритмы и программы, пригодные для практического применения	**З1 ОПК-6.1** Знать: основы алгоритмизации, языки программирования, методы реализации алгоритмов, операционные системы и оболочки, современные программные среды разработки информационных систем и технологий
		ОПК-6.3 Ведет и использует базы данных и информационные хранилища	**У1 ОПК-6.3** Уметь: применять языки программирования, работать с базами данных, разрабатывать информационные системы и технологии для автоматизации бизнес-процессов, решать прикладные задачи различных классов

Профессиональные компетенции

Таблица 3

Код компетенции	Наименование компетенции	Код и наименование индикатора достижения компетенции	Результаты обучения
не предусмотрены учебным планом			

Матрица соответствия оценочных средств запланированным результатам обучения

Таблица 4

Код и индикатор	Оценочные средства		
	Раздел 1.	**Раздел 2.**	**Промежуточная**

достижения компетенции	Название		аттестация
	Вопросы к устному опросу, тестовые задания		экзамен
ОПК-2.1	З2 ОПК-2.1	З2 ОПК-2.1	З2 ОПК-2.1
ОПК-2.2	У2 ОПК-2.2 В2 ОПК-2.2	У2 ОПК-2.2 В2 ОПК-2.2	У2 ОПК-2.2 В2 ОПК-2.2
ОПК-6.1	З1 ОПК-6.1	З1 ОПК-6.1	З1 ОПК-6.1
ОПК-6.3	У1 ОПК-6.3	У1 ОПК-6.3	У1 ОПК-6.3

2. Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующие процесс формирования компетенций в ходе освоения образовательной программы

2.1. Формы текущего контроля успеваемости

Промежуточная аттестация проводится в виде письменного/устного опроса, тестирования и представляет собой ответы на 2 вопроса и выполнение тестовых заданий.

Примерный перечень тестовых заданий

Номер задания	Правильный ответ	Содержание вопроса	Компетенция	Время выполнения задания, мин
1.	А	Параметры объекта, которые характеризуют возможные действия над ними, называются а) свойствами б) методами в) событиями г) характеристиками	ОПК-2	2
2.	Б	Способность объекта скрывать внутреннее устройство своих свойств и методов, называется а) Абстрагирование б) Инкапсуляция в) Наследование г) Полиморфизм	ОПК-2	2
3.	Г	Выделение характеристик и свойств объекта, которые позволяют его однозначно отделить от других объектов, называется .. а) Инкапсуляция б) Наследование в) Полиморфизм г) Абстрагирование	ОПК-2	2
4.	А	Основным понятием объектно-ориентированного программирования является: а) объект б) модуль в) структура г) функция	ОПК-2	2
5.	Б	Для чего предназначен оператор **namespace?** а) для заключения в группу объявлений классов, переменных и функций для использования только в текущем модуле б) для заключения в группу объявлений классов, переменных и функций в отдельный контекст со своим именем в) для использования классов, переменных и функций из других модулей программы без использования заголовочных файлов	ОПК-6	2
6.	Б	Функция вычисляет произведение двух чисел. Исходные данные вводятся с клавиатуры. Какие проверки целесообразно ввести в программе? а) проверка, что исходные данные являются числами и эти числа больше нуля б) проверка, что исходные данные являются числами в) проверка исходных данных на равенство нулю г) проверки не нужны, все возможные ошибки отловит компилятор	ОПК-6	2
7.	В	Что будет выведено в результате выполнения данного кода? int f1(int x1, int &x2) { return ++x1 + (++x2); } int main() { int a = 7, k = 1; k = f1(a, k);	ОПК-6	2

		cout << a << " " << k; } а) 7 1 б) 7 10 в) 8 10		
8.	Б	Какой из наборов перечисляемых значений записан правильно? а) enum { a, b = 3, c = 4, 3 }; б) enum {a, b = 3, c, d }; в) enum { a, b, 3, 4 };	ОПК-6	2

9.	Б	Существует файл «test.dat» в котором записано «Hello World». Каково будет содержимое файла после выполнения кода:
		ofstream outfile("c:\\test.dat");
		if (!outfile)
		{ cout << "Ошибка создания файла";
		return 1; }
		outfile << "!!!" << endl;
		outfile.close();
		а) !!!
		б) Hello World!!!
		в) Hello World
			ОПК-6	2

10.	Б	Что выведет следующая программа?
		#include <iostream.h>
		int main() {
		int 1_i ;
		for(1_i = 0; 1_i < 9; 1_i++)
		cout << 1_i +1;
		return 1;}
		а) цифры от 0 до 8
		б) программа не будет построена из-за ошибок
		в) цифры от 1 до 9
			ОПК-6	2

11.	Г	Что означает константа **ios_base::ate**, передаваемая в качестве аргумента? а) Открыть файл только для чтения. б) Открыть файл, не создавая его. в) Открыть файл, предварительно создав его. г) При открытии переместить указатель в конец файла.	ОПК-6	2
12.	Б	В каком из следующих вариантов ответов выполнен корректный доступ к переменной структуры, причём структура объявлена через указатель? а) b.var; б) b>var; в) b-var	ОПК-6	2
13.	Г	Словосочетание "Hello world!" может быть сохранено в символьном массиве размером n элементов. Укажите чему равно n? а) 11 б) 13 в) 10 г) 12	ОПК-6	2
14.	Г	Укажите корректное определение строковой переменной а) string mystr[20]; б) string[20] mystr; в) char mystr[20]; г) string mystr	ОПК-6	2
15.	Б	В каком из вариантов ответов объявлен двумерный массив? а) array anarray[20][20]; б) int anarray[20][20]; в) int array[20, 20]; г) char array[20]	ОПК-6	2

Перечень вопросов к устному опросу

Номер задания	Правильный ответ	Содержание вопроса	Компетенция	Время выполнения задания, мин
1.	Структурный стиль программирования способствует созданию кода, который проще понять, изменить, тестировать и поддерживать. Он	Основные преимущества структурного стиля программирования.	ОПК-2	2

	позволяет управлять сложностью программы и обеспечивает более эффективное использование ресурсов программиста. Основные преимущества структурного стиля программирования включают следующее: читаемость и понятность, понимание кода сверху вниз, модульность и повторное использование, отладка и тестирование, управление сложностью, расширяемость			
2.	C++ - это высокоуровневый язык программирования, разработанный на основе языка C, с добавлением различных расширений и возможностей, таких как классы и объекты.	Что представляет собой язык программирования C++?	ОПК-2	2
3.	C++ добавляет в C новые возможности, такие как объектно-ориентированное программирование, исключения, шаблоны, перегрузка операторов и пространства имен.	Какие основные особенности различают C++ от C?	ОПК-2	2
4.	Класс - это шаблон или форма для создания объектов в C++. Он определяет состояние (поля) и поведение (методы) объектов.	Что такое класс в C++?	ОПК-2	2
5.	Объект - это экземпляр класса. Он содержит данные (поля) и может выполнять определенные действия (методы), определенные в классе.	Что такое объект в C++?	ОПК-2	2
6.	В C++ структура и класс похожи, но имеют некоторые различия по умолчанию: поля структуры являются открытыми (public) по умолчанию, в то время как поля класса по умолчанию являются закрытыми (private).	Чем отличается структура от класса в C++?	ОПК-2	2
7.	Наследование - это механизм в C++, который позволяет создавать новые классы (подклассы) на основе существующих классов (базовых классов), наследуя их данные и поведение.	Что такое наследование в C++?	ОПК-2	2
8.	Полиморфизм - это возможность объектов разных классов обладать разным поведением при вызове одного и того же метода. Это достигается с помощью виртуальных функций и наследования.	Что такое полиморфизм в C++?	ОПК-2	2
9.	Исключения - это механизм для обработки и управления ошибками в программе. Код, который может вызывать исключения, помещается в блок try, а обработка исключений осуществляется в блоках catch.	Что такое исключения в C++?	ОПК-2	2
10.	Шаблоны (template) - это механизм, который позволяет создавать обобщенные типы и функции в C++. Они позволяют писать код, который может работать с различными типами данных.	Что такое шаблоны (template) в C++?	ОПК-2	2

2.2. Формы промежуточной аттестации

Промежуточная аттестация проводится в виде письменного/устного опроса, тестирования и представляет собой ответы на 2 вопроса и выполнение тестовых заданий.

Примерный перечень вопросов к экзамену

Номер задания	Правильный ответ	Содержание вопроса	Компетенция	Время выполнения задания, мин
1.	Указатель - это переменная, которая содержит адрес памяти другой переменной. Указатели используются для работы с динамической памятью, передачи аргументов по ссылке и других операций, связанных с адресами памяти.	Что такое указатель в C++?	ОПК-2	2
2.	Объектно-ориентированное программирование (сокр. ООП) — методология программирования, основанная на представлении программы в виде совокупности взаимодействующих объектов,	Методология объектно-ориентированного программирования:	ОПК-2	2

	каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования	определение, базовые принципы.		
3.	Директива — специальная команда, указывающая компилятору на особенности обработки кода при компиляции.	Что такое директива?	ОПК-2	2
4.	Директивы препроцессора позволяют изменить текст программы, например, заменить некоторые лексемы в тексте, вставить текст из другого файла, запретить трансляцию части текста и т. п.	Когда выполняются директивы препроцессора?	ОПК-2	2
5.	В языках программирования Си и C++ заголовочные файлы — основной способ подключить к программе типы данных, структуры, прототипы функций, перечисляемые типы и макросы, используемые в другом модуле. По умолчанию используется расширение . h; иногда для заголовочных файлов языка C++ используют расширение . hpp.	Для чего нужны заголовочные файлы C++?	ОПК-2	2
6.	В языке программирования С классы и объекты используются для определения набора связанных данных и функций. Класс — это шаблон, определяющий структуру объекта. Используется для создания объектов класса. Объекты создаются с использованием имени класса, за которым следуют круглые скобки. Например, если класс называется «Человек», объект этого класса можно создать с использованием синтаксиса «Человек p1 = новый Человек();» Ключевое слово «new» используется для создания нового экземпляра класса.	Определение классов и объектов в программе на языке Си++.	ОПК-2	2
7.	Перечислимый тип данных, или перечисление (enum), в C++ является пользовательским типом данных, который позволяет определить набор значений, называемых элементами перечисления. Он используется для объявления переменных, которые могут принимать только определенные значения из предопределенного списка.	Перечислимый тип данных в C++: определение.	ОПК-6	2
8.	Назначение перечислимых типов данных: Улучшение читаемости кода: перечисления позволяют использовать именованные константы вместо числовых значений, что делает код более понятным и читаемым. Ограничение допустимых значений: перечисления могут использоваться для ограничения значений переменных, чтобы они принимали только предопределенные значения из списка элементов перечисления. Повышение безопасности: перечисления могут быть полезны для предотвращения ошибок, связанных с использованием неправильных значений, так как они предоставляют ограниченный и контролируемый набор допустимых значений.	Перечислимый тип данных в C++: назначение.	ОПК-6	2

9.	Пример использования перечислимого типа данных в C++:
	
	#include <iostream>
	
	enum Color {
	RED,
	GREEN,
	BLUE
	};
	
	int main() {
	Color myColor = GREEN;
	
	if (myColor == RED) {
	std::cout << "The color is red." << std::endl;
	} else if (myColor == GREEN) {
	std::cout << "The color is green." << std::endl;
	} else if (myColor == BLUE) {
	std::cout << "The color is blue." << std::endl;
	}
	
	return 0;
	}
	В этом примере определен перечислимый тип данных Color, который может принимать значения RED, GREEN и BLUE. В функции main() переменная myColor объявлена типа Color и установлена равной GREEN. Затем используется условное выражение для проверки значения myColor и выводится соответствующее сообщение.
		Перечислимый тип данных в C++: пример использования.	ОПК-6	2

10.	В C++ потоковый ввод-вывод (I/O) реализуется с использованием классов и объектов из библиотеки <iostream>. Эта библиотека предоставляет возможности работы с различными типами потоков, включая стандартные потоки (stdin, stdout и stderr) и пользовательские потоки.	Потоковый ввод-вывод в C++.	ОПК-6	2

11.	В C++ поток представляет собой последовательный поток символов, через который данные могут быть переданы между различными источниками и приемниками, такими как консоль, файлы или другие программы. Потоки поддерживают операции ввода и вывода.	Определение потока в C++	ОПК-6	2
12.	Стандартные потоки ввода-вывода: C++ предоставляет три стандартных потока: std::cin: Стандартный поток ввода (stdin). Он используется для получения данных из внешнего источника, например, с клавиатуры. std::cout: Стандартный поток вывода (stdout). Он используется для вывода данных во внешний приемник, например, на экран. std::cerr: Стандартный поток ошибок (stderr). Он также используется для вывода данных, но обычно для сообщений об ошибках или отладочной информации. Отличие от std::cout заключается в том, что std::cerr обрабатывается без буферизации, что означает, что данные выводятся немедленно.	Стандартные потоки в C++.	ОПК-6	2
13.	Типы потоков: В C++ потоки могут быть различных типов, таких как: std::ofstream: Поток для записи данных в файл. std::ifstream: Поток для чтения данных из файла. std::fstream: Поток для чтения и записи данных в файл. std::stringstream: Поток, который работает со строками в памяти. Он позволяет выполнить ввод-вывод операций с использованием строки вместо файла.	Типы потоков в C++	ОПК-6	2

14.	Пример использования стандартных потоков ввода-вывода:
	
	#include <iostream>
	
	int main() {
	int number;
	std::cout << "Enter a number: ";
	std::cin >> number;
	std::cout << "You entered: " << number << std::endl;
	
	std::cerr << "This is an error message." << std::endl;
	
	return 0;
	}
	В этом примере используются стандартные потоки ввода-вывода. Сначала программа запрашивает у пользователя ввод числа с помощью std::cin, а затем выводит число обратно на экран с использованием std::cout. В случае ошибки, она выводит сообщение об ошибке с помощью std::cerr.
	Вы также можете использовать пользовательские потоки, такие как std::ofstream или std::ifstream, для работы с файлами.
		Пример использования стандартных потоков ввода-вывода	ОПК-6	2

15.	В C++ для открытия файловых потоков используется класс std::fstream из библиотеки <fstream>. При открытии потоков важно указать режимы открытия файла, которые определяют, как данные будут читаться или записываться. Режимы открытия файловых потоков: std::fstream::in - Поток открыт только для чтения. Если файл не существует, открытие завершится неудачей. std::fstream::out - Поток открыт только для записи. Если файл не существует, создается новый. Если файл существует, его содержимое удаляется и записывается новое. std::fstream::app - Поток открыт для записи в конец файла. Если файл не существует, создается новый. std::fstream::ate - Поток открыт для записи, указатель позиции устанавливается в конец файла. std::fstream::trunc - Поток открыт для записи, и если файл существует, его содержимое удаляется. std::fstream::binary - Открывает поток в двоичном режиме. Этот режим позволяет работать с двоичными данными, включая чтение и запись в виде байт.	Открытие потоков в C++: режимы открытия	ОПК-6	2
16.	Различия между текстовыми и двоичными потоками: Текстовые потоки (std::ifstream, std::ofstream, std::fstream без флага std::fstream::binary) используются для чтения и записи текстовых данных. При чтении, текстовый файл считывается как последовательность символов, а при записи символы интерпретируются в соответствии с кодировкой, такой как ASCII или UTF-8. При работе с текстовыми потоками, трансляция символов новой строки между операционной системой (например, \n в Unix и \r\n в Windows) может происходить автоматически.	Различия текстовых и двоичных потоков.	ОПК-6	2

	Двоичные потоки (std::ifstream, std::ofstream, std::fstream с флагом std::fstream::binary) используются для чтения и записи двоичных данных без изменений и интерпретаций. Они позволяют работать с данными в виде байт и не выполняют автоматическую трансляцию символов новой строки.			

17.	Пример открытия файловых потоков:
	
	#include <fstream>
	
	int main() {
	std::ofstream outfile("output.txt", std::fstream::out);
	if (outfile.is_open()) {
	// Здесь можно выполнять операции записи в файл с помощью outfile
	outfile << "Hello, World!";
	outfile.close();
	}
	
	std::ifstream infile("input.txt", std::fstream::in);
	if (infile.is_open()) {
	// Здесь можно выполнять операции чтения из файла с помощью infile
	std::string data;
	infile >> data;
	infile.close();
	}
	
	return 0;
	}
	В этом примере создаются объекты std::ofstream и std::ifstream для записи и чтения файлов соответственно. Режимы открытия std::fstream::out и std::fstream::in указывают, что потоки открыты только для записи и чтения соответственно.
		Пример открытия файловых потоков в C++	ОПК-6	2

18.	В C++ для контроля состояния потоков используются различные функции и методы классов потоков (std::istream и std::ostream). Некоторые из основных функций контроля состояния потоков: std::ios::good() или streamObject.good() - возвращает true, если поток находится в "хорошем" состоянии, то есть нет ошибок и достигнут конец файла. Это эквивалентно проверке !streamObject.fail(). std::ios::bad() или streamObject.bad() - возвращает true, если в потоке произошла "критическая" ошибка, которая, скорее всего, не может быть восстановлена, например, ошибка ввода-вывода. std::ios::fail() или streamObject.fail() - возвращает true, если в потоке произошла ошибка. Это может быть вызвано, например, некорректным вводом или выводом, таким как попытка чтения числа из строки, содержащей символы. std::ios::eof() или streamObject.eof() - возвращает true, если был достигнут конец файла. std::ios::clear() или streamObject.clear() - сбрасывает флаги ошибок в потоке, восстанавливая его состояние. std::ios::rdstate() или streamObject.rdstate() - возвращает битовую маску, представляющую текущие состояния потока. Может быть использована для проверки конкретных флагов ошибок. std::ios::setstate(std::ios::iostate state) или streamObject.setstate(std::ios::iostate state) - устанавливает состояние потока, установив соответствующие флаги ошибок.	Основные функции контроля состояния потоков.	ОПК-6	2

19.	Пример использования этих функций:
	
	#include <iostream>
	
	int main() {
	std::ifstream infile("input.txt");
	if (!infile) {
	// Ошибка открытия файла
	std::cerr << "Ошибка открытия файла\n";
	return 1;
	}
	
	int value;
	infile >> value;
	
	if (infile.bad()) {
		Пример использования функции контроля состояния потоков.	ОПК-6	2

	// Критическая ошибка ввода-вывода
	std::cerr << "Критическая ошибка ввода-вывода\n";
	return 1;
	}
	
	if (infile.fail()) {
	// Ошибка чтения значения
	std::cerr << "Ошибка чтения значения\n";
	infile.clear(); // Очищаем флаги ошибок
	infile.ignore(std::numeric_limits<std::streamsize>::max(), '\n'); // Пропускаем оставшуюся часть строки
	}
	
	if (infile.eof()) {
	// Достигнут конец файла
	std::cout << "Достигнут конец файла\n";
	}
	
	return 0;
	}
	В этом примере используются функции контроля состояния потока для обработки ошибок ввода и вывода. Проверяется состояние потока после операции чтения значения и принимается соответствующая реакция на ошибки.			

20.	В C++ для ввода строковой информации в поток и вывода информации из потока используются следующие функции и методы:
	
	Для ввода информации в поток:
	
	std::getline(std::istream& stream, std::string& str) - функция, которая позволяет считывать строку из потока stream и сохранять ее в переменную str. Она считывает строку до символа новой строки ('\n') и удаляет символ новой строки из потока.
	
	std::istream& operator>>(std::istream& stream, std::string& str) - перегруженный оператор >>, который позволяет считывать строку из потока stream и сохранять ее в переменную str. Он считывает строку до первого пробела, табуляции или символа новой строки и оставляет символ новой строки в потоке.
	
	Для вывода информации из потока:
	
	std::ostream& operator<<(std::ostream& stream, const std::string& str) - перегруженный оператор <<, который позволяет выводить строку str в поток stream. Он записывает содержимое строки в поток.
	Пример использования этих функций:
	
	#include <iostream>
	#include <string>
	
	int main() {
	std::string name;
	std::cout << "Введите ваше имя: ";
	std::getline(std::cin, name);
	std::cout << "Привет, " << name << "!\n";
	
	std::string city;
	std::cout << "Введите ваш город: ";
	std::cin >> city;
	std::cout << "Вы живете в городе " << city << ".\n";
	
	return 0;
	}
	В этом примере используются функции std::getline и оператор >> для ввода строковой информации из стандартного ввода (std::cin). Функция std::getline используется для ввода имени, позволяя ввести строку с пробелами. Оператор >> используется для ввода города, считывая строку до первого пробела или символа новой строки.
	
	Затем используется оператор << для вывода информации в стандартный вывод (std::cout).
		Основные функции ввода строковой информации в поток и вывода информации из потока.	ОПК-6	2

21.	Формальные параметры (также называемые формальными	Формальные и	ОПК-6	2

	аргументами или просто параметрами) функции - это переменные, которые объявлены в определении функции и используются для представления аргументов, передаваемых в функцию при ее вызове. Формальные параметры представляют собой список переменных, указанных в объявлении функции с их соответствующими типами данных (если применимо). Фактические параметры (также называемые фактическими аргументами) функции - это реальные значения или переменные, которые передаются функции в момент ее вызова. Фактические параметры представляют собой значения, которые передаются в функцию и используются внутри функции для выполнения определенных операций. Связь между формальными и фактическими параметрами заключается в том, что при вызове функции фактические параметры копируются в формальные параметры функции. Таким образом, значения, передаваемые в функцию как фактические параметры, становятся доступными внутри функции через соответствующие формальные параметры. Внутри функции можно использовать формальные параметры для выполнения необходимых вычислений или операций.	фактические параметры. Связь формальных и фактических параметров функций.		

22.	В C++ можно передавать массивы в функции как параметры. Когда массив передается в функцию, он передается по указателю на его первый элемент. Это означает, что функция имеет доступ к оригинальному массиву и может изменять его содержимое.
	Для передачи массива в функцию в C++ следует указать имя массива в списке формальных параметров функции с квадратными скобками пустой размерности или с указанием фиксированного размера массива. Например:
	
	`void printArray(int arr[], int size) {`
	`  for (int i = 0; i < size; ++i) {`
	`  cout << arr[i] << " ";`
	`  }`
	`}`
	В приведенном примере функция printArray принимает массив arr вместе с параметром size, который указывает размер массива. Внутри функции мы можем обращаться к элементам массива с использованием индексов.
	
	Чтобы передать массив в функцию, просто передайте его имя как фактический параметр при вызове функции. Например:
	
	`int main() {`
	`  int myArray[] = {1, 2, 3, 4, 5};`
	`  int size = sizeof(myArray) / sizeof(myArray[0]);`
	
	`  printArray(myArray, size);  // Вызов функции с передачей массива`
	`  return 0;`
	`}`
	В данном случае мы создаем массив myArray и находим его размер, используя операторы sizeof. Затем мы передаем массив myArray в функцию printArray вместе с его размером.
	
	Когда массив передается в функцию, изменения, внесенные в массив внутри функции, будут видны за ее пределами. Это происходит потому, что массив передается по указателю на первый элемент, а не копируется. Если вы хотите избежать изменений оригинального массива, можно использовать ключевое слово const в объявлении параметра функции:
	
	`void printArray(const int arr[], int size) {`
	`  //...`
	`}`
	Такое объявление указывает, что функция не изменяет содержимое массива arr.
		Функции и массивы: массивы как параметры функций.	ОПК-6	2

23.	Указатель в C++ - это переменная, которая содержит адрес в памяти другой переменной или объекта. Он "указывает" на местонахождение другого объекта в памяти. Описывается указатель с использованием звездочки * перед именем переменной. Например, следующая строка объявляет указатель на целочисленную переменную: `int* ptr;`	Указатели. Описание указателя. Операции над указателем. Указатель и одномерный массив. Доступ к элементам массива через	ОПК-6	2

	Для работы с указателями доступны следующие операции:
	Взятие адреса: Используя оператор взятия адреса &, можно получить адрес переменной или объекта. Например:
	int var = 42;
	int* ptr = &var; // ptr указывает на адрес переменной var
	Разыменование: Используя оператор разыменования *, можно получить доступ к значению, на которое указывает указатель. Например:
	int value = *ptr; // value будет содержать значение переменной var
	Изменение указателя: Указатели также могут быть перенаправлены, чтобы указывать на другие переменные или объекты. Например:
	int a = 10;
	int b = 20;
	int* ptr = &a; // ptr указывает на переменную a
	ptr = &b; // теперь ptr указывает на переменную b
	Указатели могут быть использованы для доступа к элементам одномерного массива. Если объявить указатель на тип элемента массива, то он может указывать на первый элемент этого массива. С помощью арифметики указателей можно получить доступ к остальным элементам массива. Например:
	int arr[] = {10, 20, 30, 40, 50};
	int* ptr = arr; // указатель ptr указывает на первый элемент массива arr
	
	for (int i = 0; i < 5; ++i) {
	  cout << *(ptr + i) << " "; // вывод элементов массива с использованием указателя
	}
	В данном примере, указатель ptr указывает на первый элемент массива arr. Затем мы используем арифметику указателей, добавляя i к указателю ptr, чтобы получить доступ к каждому элементу массива. Оператор разыменования * используется для получения значения элемента.
	
	Также можно использовать квадратные скобки для доступа к элементам массива через указатель:
	
	int arr[] = {10, 20, 30, 40, 50};
	int* ptr = arr;
	
	for (int i = 0; i < 5; ++i) {
	  cout << ptr[i] << " "; // вывод элементов массива через указатель с использованием квадратных скобок
	}
	Оба этих подхода дают возможность получить доступ к элементам массива через указатель.
		указатель.		

24.	Алгоритм быстрой сортировки (также известный как сортировка Хоара) является одним из популярных алгоритмов сортировки в C++ и других языках. Он основан на принципе "разделяй и властвуй" и использует стратегию разделения массива на две части: элементы, меньшие опорного элемента, и элементы, большие опорного элемента. Рекурсивно применяя эту стратегию к каждой из двух частей, массив упорядочивается.
	
	Вот рекурсивная реализация алгоритма быстрой сортировки на C++:
	
	#include <iostream>
	#include <vector>
	
	// Функция для обмена элементов
	void swap(int& a, int& b) {
	  int temp = a;
	  a = b;
	  b = temp;
	}
	
	// Функция для разделения массива и возврата индекса опорного элемента
	int partition(std::vector<int>& arr, int low, int high) {
	  int pivot = arr[high]; // Опорный элемент
	  int i = (low - 1); // Индекс меньшего элемента
	
	  for (int j = low; j <= high - 1; j++) {
		Алгоритм быстрой обменной сортировки, его рекурсивная реализация	ОПК-6	2

	// Если текущий элемент меньше или равен опорному
	if (arr[j] <= pivot) {
	i++; // Увеличиваем индекс меньшего элемента
	swap(arr[i], arr[j]);
	}
	}
	swap(arr[i + 1], arr[high]);
	return (i + 1); // Возвращаем индекс опорного элемента
	}
	
	// Функция быстрой сортировки
	void quickSort(std::vector<int>& arr, int low, int high) {
	if (low < high) {
	// Поиск индекса опорного элемента
	int pivotIndex = partition(arr, low, high);
	
	// Рекурсивно сортируем элементы до и после опорного элемента
	quickSort(arr, low, pivotIndex - 1);
	quickSort(arr, pivotIndex + 1, high);
	}
	}
	
	// Функция для вывода массива на экран
	void printArray(const std::vector<int>& arr) {
	for (int i = 0; i < arr.size(); i++) {
	std::cout << arr[i] << " ";
	}
	std::cout << std::endl;
	}
	
	int main() {
	std::vector<int> arr = {9, 2, 5, 1, 7, 4, 8, 6, 3};
	int n = arr.size();
	
	std::cout << "Исходный массив: ";
	printArray(arr);
	
	quickSort(arr, 0, n - 1);
	
	std::cout << "Упорядоченный массив: ";
	printArray(arr);
	
	return 0;
	}
	В этом примере функция quickSort выполняет основную логику быстрой сортировки: выбор опорного элемента, разделение массива и рекурсивные вызовы для подмассивов до и после опорного элемента. Функции swap используется для обмена элементов, а функция partition разделяет массив на две части и возвращает индекс опорного элемента. Функция printArray используется только для вывода массива на экран.
	
	Пример вывода:
	
	Исходный массив: 9 2 5 1 7 4 8 6 3
	Упорядоченный массив: 1 2 3 4 5 6 7 8 9
	Этот пример демонстрирует рекурсивную реализацию алгоритма быстрой сортировки на C++. Вы можете изменить входной массив (arr), чтобы опробовать алгоритм на других данных.			

25.	Векторы являются одним из наиболее удобных и часто используемых контейнеров в C++. Они представляют собой динамический массив, который автоматически расширяется при добавлении элементов. Векторы обеспечивают эффективное хранение и манипулирование данными. Для использования векторов при обработке данных вы должны включить заголовочный файл <vector>. Вот примеры некоторых основных операций с векторами: Создание вектора и добавление элементов: #include <iostream> #include <vector>	Использование векторов при обработке данных	ОПК-6	2

```
int main() {
    std::vector<int> vec;  // Создание пустого вектора

    vec.push_back(10);  // Добавление элемента в конец вектора
    vec.push_back(20);
    vec.push_back(30);

    for (int i : vec) {
        std::cout << i << " ";  // Вывод элементов вектора: 10 20 30
    }

    return 0;
}
```

Обращение к элементам вектора:

```
#include <iostream>
#include <vector>

int main() {
    std::vector<int> vec = {10, 20, 30};

    std::cout << vec[0] << std::endl;  // Вывод первого элемента: 10
    std::cout << vec.at(1) << std::endl;  // Вывод второго элемента: 20

    return 0;
}
```

Размер вектора и перебор его элементов:

```
#include <iostream>
#include <vector>

int main() {
    std::vector<int> vec = {10, 20, 30};

    std::cout << "Размер вектора: " << vec.size() << std::endl;  // Вывод
размера: 3

    for (int i = 0; i < vec.size(); i++) {
        std::cout << vec[i] << " ";  // Вывод элементов вектора: 10 20 30
    }

    return 0;
}
```

Изменение элементов вектора:

```
#include <iostream>
#include <vector>

int main() {
    std::vector<int> vec = {10, 20, 30};

    vec[1] = 40;  // Изменение второго элемента

    for (int i : vec) {
        std::cout << i << " ";  // Вывод элементов вектора: 10 40 30
    }

    return 0;
}
```

Удаление элементов из вектора:

```
#include <iostream>
#include <vector>

int main() {
    std::vector<int> vec = {10, 20, 30};

    vec.pop_back();  // Удаление последнего элемента

    for (int i : vec) {
        std::cout << i << " ";  // Вывод элементов вектора: 10 20
    }

    return 0;
}
```

Векторы также предоставляют множество других методов и функций для удобной работы с данными, таких как insert, erase, clear и т.д.

	Более подробную информацию вы можете найти в документации по C++ или в онлайн-ресурсах, посвященных векторам в C++.			

26.	В C++, указатели на функции представляют собой переменные, которые содержат адрес функции. Они позволяют программе передавать функции в качестве аргументов, хранить их в структурах или массивах и вызывать их динамически. Указатели на функции являются мощным механизмом, используемым, например, при использовании колбэков, обратных вызовов и в системах событий.
	Определение указателя на функцию:
	Указатель на функцию определяется с использованием синтаксиса, который указывает возвращаемый тип функции и ее сигнатуру (типы аргументов):
	
	returnType (*pointerName)(argType1, argType2, ...);
	Например, чтобы определить указатель на функцию, принимающую два целочисленных аргумента и возвращающую целое число:
	
	int (*sumPointer)(int, int);
	Назначение указателя на функцию:
	Для назначения указателю адреса функции используется оператор взятия адреса &. Например, чтобы назначить указателю sumPointer адрес функции sum, необходимо выполнить следующее присваивание:
	
	sumPointer = ∑ // sum - имя функции
	Использование указателей на функции:
	После назначения указателю адреса функции, можно вызвать функцию через указатель, используя оператор вызова функции ().
	Например:
	
	int result = (*sumPointer)(2, 3);
	Также можно использовать синтаксис указателей на функции, который автоматически выполняет разыменование указателя:
	
	int result = sumPointer(2, 3);
	Полный пример использования указателя на функцию:
	
	#include <iostream>
	
	int sum(int a, int b) {
	return a + b;
	}
	
	int main() {
	int (*sumPointer)(int, int); // определение указателя на функцию sum
	sumPointer = ∑ // назначение указателю адреса функции sum
	
	int result = sumPointer(2, 3); // вызов функции через указатель
	
	std::cout << "Result: " << result << std::endl;
	
	return 0;
	}
	В данном примере указатель на функцию sumPointer назначается адрес функции sum, и затем функция вызывается через указатель, возвращая результат сложения двух чисел.
	
	Указатели на функции предоставляют гибкость и возможность динамического выбора функций для выполнения определенных операций. Они широко используются в различных областях программирования, таких как разработка библиотек, реализация обратных вызовов и обработка событий.
		Указатели на функции: определение, назначение, примеры использования.	ОПК-6	2

27.	Перегрузка функций в C++ позволяет определить несколько функций с одним и тем же именем, но с разными параметрами. Компилятор различает эти функции на основе их сигнатур (типы и количество аргументов). Когда вызывается перегруженная функция, компилятор выбирает наиболее подходящий вариант функции на основе переданных аргументов. Основные преимущества перегрузки функций: Читаемость кода: Позволяет использовать одно имя для функций с похожими задачами, что делает код более понятным и читаемым.	Перегрузка функций: определение, достоинства, ограничения. Примеры реализации.	ОПК-6	2

	Гибкость: Позволяет разработчику выбрать наиболее подходящую функцию на основе типов и количества аргументов, без необходимости придумывать разные имена для каждой разновидности функции.
	Ограничения перегрузки функций:
	Только типы параметров учитываются: Перегрузка функций работает только на основе типов и количества параметров функции. Возвращаемое значение и модификаторы const/volatile не учитываются компилятором при выборе перегруженной функции.
	Единственный идентификатор функции: Несмотря на то, что у перегруженных функций одно имя, фактически они все равно считаются разными функциями, и каждая из них должна быть определена отдельно.
	Примеры реализации перегрузки функций:
	
	#include <iostream>
	
	// Перегруженные функции для сложения чисел различных типов
	int sum(int a, int b) {
	return a + b;
	}
	
	double sum(double a, double b) {
	return a + b;
	}
	
	int sum(int a, int b, int c) {
	return a + b + c;
	}
	
	int main() {
	int result1 = sum(2, 3); // вызывается функция sum(int, int)
	double result2 = sum(2.5, 3.7); // вызывается функция sum(double, double)
	int result3 = sum(2, 3, 4); // вызывается функция sum(int, int, int)
	
	std::cout << "Result1: " << result1 << std::endl;
	std::cout << "Result2: " << result2 << std::endl;
	std::cout << "Result3: " << result3 << std::endl;
	
	return 0;
	}
	В этом примере определены три перегруженные версии функции sum(), принимающие разные типы и количество параметров. В функции main() эти функции вызываются с различными аргументами. Компилятор выбирает наиболее подходящую версию функции на основе переданных аргументов и возвращает соответствующие результаты.
	Перегрузка функций является мощным инструментом в C++, который позволяет разработчикам создавать более гибкий и читаемый код за счет использования одного имени функции для нескольких вариантов ее реализации.			

28.	Препроцессор выполняет предварительные операции с файлами C и C++ перед их передачей компилятору. Препроцессор можно использовать для условной компиляции кода, вставки файлов, задания сообщений для ошибок времени компиляции, а также для применения правил, зависящих от компьютера, к разделам кода	Какую функцию выполняет препроцессор?	ОПК-6	2
29.	Препроцессорные команды, или директивы, управляют работой препроцессора. Таких команд немного, они все начинаются со знака решётки (#) и должны быть в начале строки исходного кода: #define.	Что такое препроцессорные директивы?	ОПК-6	2
30.	Директива #define заставляет компилятор подставлять строку токена для каждого вхождения идентификатора в исходном файле. Идентификатор заменяется только при создании маркера. То есть идентификатор не заменяется, если он отображается в комментарии, в строке или в составе более длинного идентификатора.	Как работает директива Define?	ОПК-6	2
31.	Директива using применяется для определения или разрешения использования типов как пространств имен.	В чем состоит назначение директивы using?	ОПК-6	2
32.	Директива препроцессора #include в языке программирования C++ используется для включения содержимого другого файла в исходный	Для чего применяется директива	ОПК-6	2

	код программы. Она позволяет расширить функциональность программы, добавив предварительно написанный код, который находится в другом файле. Обычно #include используется для включения заголовочных файлов (файлов с расширением .h или .hpp), которые содержат объявления функций, классов, констант и других элементов программы. Заголовочные файлы обычно содержат только объявления без определений (реализаций), поэтому они могут быть включены в несколько файлов, чтобы обеспечить доступность объявлений в различных частях программы.	препроцессора include?		

33.	В C++ можно определить функции с умалчиваемыми значениями параметров, что позволяет вызывать функцию с меньшим количеством аргументов, при этом используя значения по умолчанию для недостающих аргументов. Это удобно, когда вам нужно задать значения по умолчанию для некоторых параметров функции, но при необходимости можно явно указать другие значения при вызове функции.
	
	Для определения функции с умалчиваемыми значениями параметров вы можете указать значения по умолчанию в объявлении функции. Например:
	
	`void printMessage(const string& message, int count = 1) {`
	`  for (int i = 0; i < count; ++i) {`
	`  cout << message << endl;`
	`  }`
	`}`
	В данном примере функция printMessage принимает первым параметром строку message и вторым параметром целое число count, имеющее значение по умолчанию равное 1. Если при вызове функции не указывать значение для count, то будет использоваться значение по умолчанию, равное 1.
	
	Вызовы функции printMessage:
	
	`printMessage("Hello");  // будет выведено "Hello"`
	`printMessage("Hi", 3);  // будет выведено "Hi" три раза`
	В первом вызове функции будет использовано значение по умолчанию для параметра count, так как его значение не указано явно. Во втором вызове функции указано явное значение 3 для параметра count, и функция будет вызвана со значением count = 3.
	
	Стоит отметить, что при определении функции с умалчиваемыми значениями параметров значения по умолчанию должны быть указаны только в объявлении функции (обычно в заголовочном файле) или в определении функции вне класса (если функция не является членом класса). Один и тот же параметр не должен иметь значения по умолчанию в объявлении и в определении.
	
	Кроме того, в функции с умалчиваемыми значениями параметров значения по умолчанию должны быть указаны для последних параметров функции. Это означает, что вы не можете задать значение по умолчанию для параметра, идущего перед другим параметром без значения по умолчанию.
		Функции с умалчиваемыми значениями параметров.	ОПК-6	2

Образец экзаменационного билета

САМАРСКИЙ ПОЛИТЕХ
Опорный университет

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования «Самарский государственный технический университет» (ФГБОУ ВО «СамГТУ»)
Филиал ФГБОУ ВО «СамГТУ» в г. Белебее Республики Башкортостан

Кафедра «Инженерные технологии»

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 1

по дисциплине (модулю): «Технологии программирования»
Код направления подготовки (специальности), направленность (профиль): 09.03.02 Информационные системы и технологии, Информационные системы и технологии
Курс 3

1. Пример использования функции контроля состояния потоков.
2. Основные функции ввода строковой информации в поток и вывода информации из потока.

Составил:
доцент _____ З.Ф. Камальдинова
(подпись)
«____» _____ ____ г.

Утверждаю:
Заведующий кафедрой _____ А.А.Цынаева
(подпись)
«____» _____ ____ г.

3. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующие процесс формирования компетенций

3.1. Характеристика процедуры текущей и промежуточной аттестации по дисциплине

Таблица 5

№ п/п	Наименование оценочного средства	Периодичность и способ проведения процедуры оценивания	Методы оценивания	Виды выставляемых оценок	Способ учета индивидуальных достижений, обучающихся
1	Вопросы к устному опросу	систематически на всех видах занятий / устно	экспертный	По пятибалльной шкале	рабочая книжка преподавателя
2	Тестовые задания.	систематически на всех видах занятий /письменно и устно	экспертный	По пятибалльной шкале	рабочая книжка преподавателя
3	Промежуточная аттестация – вопросы к экзамену	по окончании изучения дисциплины/ устно и письменно	экспертный	По пятибалльной шкале	экзаменационная ведомость, зачетная книжка

3.2. Критерии и шкала оценивания результатов изучения дисциплины во время занятий (текущий контроль успеваемости)

Критерии оценки и шкала оценивания вопросов к устному опросу

Таблица 6

Шкала оценивания	Критерии оценки	Кол-во баллов
«Отлично»	Студент показывает полные и глубокие знания программного материала, логично и аргументировано отвечает на поставленный вопрос, а также дополнительные вопросы, показатели рейтинга (все предусмотренные РПД учебные задания выполнены, качество выполнения большинства из них оценено числом баллов, близким к максимальному).	(26-50) баллов
«Хорошо»	Студент показывает глубокие знания программного материала, грамотно его излагает, достаточно полно отвечает на поставленный вопрос и дополнительные вопросы, умело формулирует выводы, допуская незначительные погрешности, показатели рейтинга (все предусмотренные РПД учебные задания выполнены, качество выполнения ни одного из них не оценено максимальным числом баллов).	(11-25) баллов
«Удовлетворительно»	Студент показывает достаточные, но неглубокие знания программного материала; при ответе не допускает грубых ошибок или противоречий, однако в формулировании ответа отсутствует должная связь между анализом, аргументацией и выводами, для получения правильного ответа требуется уточняющие вопросы, достигнуты минимальные или выше показатели рейтинговой оценки при наличии выполнения предусмотренных РПД учебных заданий	(5-10) баллов
«Неудовлетворительно»	Ответы на вопросы даны не верно	(0-4)

		баллов

Критерии оценивания тестовых заданий

Таблица 7

Шкала оценивания	Критерии оценки	Кол-во баллов
«Отлично»	выше 84% правильных ответов	(26-50) баллов
«Хорошо»	от 68 до 84% правильных ответов	(11-25) баллов
«Удовлетворительно»	от 51 до 67% правильных ответов	(5-10) баллов
«Неудовлетворительно»	Менее 51% правильных ответов	(0-4) баллов

Общие критерии и шкала оценивания результатов для допуска к промежуточной аттестации

Таблица 8

Наименование оценочного средства		Балльная шкала
1.	Вопросы к устному опросу	0-50 баллов
2.	Тестовые задания	0-50 баллов
	Итого:	100 баллов

Максимальное количество баллов за семестр – 100. Обучающийся допускается к экзамену при условии 51 и более набранных за семестр баллов.

3.3 Критерии и шкала оценивания результатов изучения дисциплины на промежуточной аттестации

Основанием для определения оценки на экзамене служит уровень освоения обучающимися материала и формирования компетенций, предусмотренных программой учебной дисциплины.

Успеваемость определяется оценками: 5 «отлично»; 4 «хорошо»; 3 «удовлетворительно»; 2 «неудовлетворительно».

Оценку «отлично» получает обучающийся, освоивший компетенции дисциплины на всех этапах их формирования **на 85-100 %,** показавший всестороннее, систематическое и глубокое знание учебного материала, умение свободно выполнять задания, предусмотренные рабочей программой, усвоивший основную и ознакомленный с дополнительной литературой, рекомендованной программой. Как правило, оценка «отлично» выставляется обучающимся, усвоившим взаимосвязь основных положений учебной дисциплины, необходимых для приобретаемой профессии, проявившим творческие способности в понимании, изложении и использовании учебного материала.

Оценку «хорошо» заслуживает обучающийся, освоивший компетенции дисциплины на всех этапах их формирования **на 71-84 %,** обнаруживший полное знание учебного материала, успешно выполняющий предусмотренные рабочей программой задания, усвоивший основную литературу, рекомендованную в программе. Как правило, оценка «хорошо» выставляется обучающимся, продемонстрировавшим систематическое владение материалом дисциплины, способным к их самостоятельному пополнению и обновлению в ходе дальнейшей учебной работы и профессиональной деятельности, но допустившим несущественные неточности в ответе.

Оценку «удовлетворительно» получает обучающийся, освоивший компетенции дисциплины на всех этапах их формирования **на 51-70 %,** обнаруживший знание основного учебного материала в объёме, необходимом для дальнейшей учебы и предстоящей работы по профессии, справляющийся с выполнением заданий, предусмотренных рабочей программой, знакомый с основной литературой, рекомендованной программой. Как правило, оценка «удовлетворительно» выставляется обучающимся, допустившим погрешности в ответе на экзамене и при выполнении экзаменационных заданий, но обладающим необходимыми знаниями для устранения под руководством преподавателя допущенных недочетов.

Оценка «неудовлетворительно» выставляется обучающемуся, освоившему компетенции дисциплины на всех этапах их формирования менее чем **на 51%,** обнаружившему пробелы в знаниях основного учебного материала, допустившему принципиальные ошибки в выполнении предусмотренных рабочей программой заданий.

Шкала оценивания результатов

Таблица 9

Процентная шкала (при ее использовании)	Оценка в системе «неудовлетворительно – удовлетворительно – хорошо – отлично»
0-50%	Неудовлетворительно
51-70%	Удовлетворительно
71-84%	Хорошо
85-100%	Отлично

Приложение 2

УТВЕРЖДАЮ
Директор филиала ФГБОУ ВО «СамГТУ»
в г. Белебее Республики Башкортостан

_____ Л.М. Инаходова
«____» _____ 20__ г.

Дополнения и изменения к рабочей программе дисциплины (модуля)

Б1.О.03.02 «Технологии программирования»

по направлению подготовки (специальности) 09.03.02 «Информационные системы и технологии» по направленности (профилю) подготовки «Информационные системы и технологии»

на 20__/20__ учебный год

В рабочую программу вносятся следующие изменения:

1) .....;
2) .....;

Разработчик дополнений и изменений:

_____	_____	_____
(должность, степень, ученое звание)	(подпись)	(ФИО)

Дополнения и изменения рассмотрены и одобрены на заседании кафедры «____» _____ 20___ г., протокол № ____.

Заведующий кафедрой

_____	_____
(степень, звание, подпись)	(ФИО)

Приложение 3

Аннотация рабочей программы дисциплины
Б1.О.03.02 «Технологии программирования»

Код и направление подготовки (специальность) **09.03.02 Информационные системы и технологии**

Направленность (профиль) **Информационные системы и технологии**

Квалификация **бакалавр**

Форма обучения **заочная**

Год начала подготовки **2023**

Выпускающая кафедра **Инженерные технологии**

Кафедра-разработчик **Инженерные технологии**

Объем дисциплины, ч. / з.е. **180 / 5**

Форма контроля (промежуточная аттестация) **экзамен**

Курс	Час. / з.е.	Лек. зан., час.	Лаб. зан., час.	Практич. зан., час.	КСР	СРС	Контроль	Форма контроля
6	180 / 5	4	8	-	5	154	9	экзамен
Итого	180 / 5	4	8	-	5	154	9	экзамен

Универсальные компетенции:	
не предусмотрены учебным планом	
Общепрофессиональные компетенции:	
ОПК-2	Способен понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности
ОПК-2.1	Использует и понимает принципы работы информационных технологий и программных средств при решении задач в сфере информационных систем и технологий
ОПК-2.2	Применяет современные информационные технологии и программные средства отечественного производства при решении задач в сфере информационных систем и технологий
ОПК-6	Способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий;
ОПК-6.1	Разрабатывает алгоритмы и программы, пригодные для практического применения
ОПК-6.3	Ведет и использует базы данных и информационные хранилища
Профессиональные компетенции:	
не предусмотрены учебным планом	

Содержание дисциплины охватывает круг вопросов, связанных с развитием технологии программирования, документированием и стандартизацией, некоторыми специальными средствами C++, расширенными возможностями работы с функциями, операциями с контейнерами, файлами и векторами, динамическими структурами данных. Рассматриваются основные понятия, такие как инкапсуляция, абстрактные типы данных, наследование, полиморфизм.

Преподавание дисциплины предусматривает следующие формы организации учебного процесса: лекции, лабораторные занятия, самостоятельная работа студента.

Программой дисциплины предусмотрены следующие виды контроля: текущий контроль успеваемости в форме вопросов к устному опросу, тестовых заданий и промежуточный контроль в форме экзамена