

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
СТАРООСКОЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ ИНСТИТУТ ИМ. А.А. УГАРОВА
(филиал) федерального государственного автономного образовательного учреждения
высшего образования
«Национальный исследовательский технологический университет «МИСИС»
СТИ НИТУ «МИСИС»

Рабочая программа утверждена
решением Ученого совета
СТИ НИТУ «МИСИС»
от «23» июня 2026 г.
протокол № 38

Рабочая программа дисциплины
Программирование на языках высокого уровня

Закрепленная кафедра **Кафедра автоматизированных и информационных систем управления им. Ю.И. Еременко**

Квалификация **Бакалавр**

Форма обучения **Очная**

Общая трудоемкость **4** ЗЕТ

Формы контроля в семестре: 4

Часов по учебному плану 144

в том числе:

аудиторные занятия 51

самостоятельная работа 48

часов на контроль 45

Семестр изучения 4

Экзамен, Курсовая работа, 4

Распределение часов дисциплины по курсам

Семестр	4		Итого	
Вид занятий	УП	РП	УП	РП
Лекции	17	17	17	17
Лабораторные	17	17	17	17
Практические занятия.	17	17	17	17
Сам. работа	48	48	48	48
Часов на контроль	45	45	45	45
Итого:	144	144	144	144

Год набора 2026г.

Программу составил: ассистент кафедры. АИСУ им.
Ю.И. Еременко,
к.т.н., Жуков Пётр Игоревич

Должность, уч.ст., уч.зв. ФИО полностью



подпись

Рабочая программа дисциплины

Программирование на языках высокого уровня

наименование

Разработана в соответствии с ОС ВО НИТУ «МИСИС»:

Самостоятельно устанавливаемый образовательный стандарт высшего образования – бакалавриат
Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский технологический университет «МИСИС» по направлению подготовки
09.03.02 Информационные системы и технологии (приказ от 05.03.2020 г. № 95 о.в.)

Составлена на основании учебного плана 2026 года набора.

09.03.02 Информационные системы и технологии

Профиль: Информационные системы и технологии, утвержденного Ученым советом СТИ НИТУ
«МИСИС» 23.06.2026 г., протокол № 38.

Рабочая программа одобрена на заседании кафедры Автоматизированные и информационные системы
управления им. Ю.И. Еременко.

Протокол от «22 июня 2026 г. № 6.

и.о. зав.
кафедрой

АИСУ им. Ю.И.
Еременко

аббревиатура наименования кафедры

«22» июня 2026 г.



подпись

Д.А. Полещенко

И.О. Фамилия

Руководитель ОПОП ВО

и.о. зав. кафедрой, кандидат технических
наук, доцент

должность, уч.ст., уч.зв.

«22» июня 2026 г.



подпись

Д.А. Полещенко

И.О. Фамилия

1. ЦЕЛИ ОСВОЕНИЯ	
Цель дисциплины получить теоретические знания и практические навыки по основам объектно-ориентированного программирования на языке высокого уровня C#. Познакомить обучающихся с понятием методологии разработки и паттернами проектирования. Научить взаимодействовать с тестовым окружением и тестировать написанный код.	
Задачи дисциплины:	
1. Научить основным подходам объектно-ориентированного программирования; 2. Развить алгоритмическое мышление; 3. Научить взаимодействовать со средами разработки программ; 4. Научить кодировать программы в соответствии с объектно-ориентированной парадигмой.	

2. МЕСТО В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ	
Цикл (раздел) ОП:	Б1.В.ДВ.
2.1	Требования к предварительной подготовке обучающихся:
2.1.1	Компьютерное обеспечение специальности
2.1.2	Проектирование информационных систем
2.1.3	Современные информационные технологии
2.1.4	Системный анализ
2.1.5	Интеллектуальные системы управления
2.2	Дисциплины и практики, для которых освоение данной дисциплины необходимо как предшествующее:
2.2.1	Научно-исследовательская работа
2.2.2	Преддипломная практика для выполнения выпускной квалификационной работы
2.2.3	Защита выпускной квалификационной работы, включая подготовку к процедуре защиты и процедуру защиты

3. ИНДИКАТОРЫ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ, СОВМЕЩЕННЫЕ С РЕЗУЛЬТАТАМИ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ	
УК-2: Способен собирать и интерпретировать данные и определять круг задач в рамках поставленной цели, выбирать оптимальные способы решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений, умение обосновывать принятые решения	
Знать:	УК-2-31 – Методики и способы тестирования программного кода
Уметь:	УК-2-У1 – Применять современные инструментальные средства юнит- и интеграционного тестирования
Владеть:	УК-2-В1 – Навыками интерпретации результатов тестирования программного кода
ОПК-6 Способен выбирать и применять методики проектирования и актуальные инструментальные средства, разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий	
Знать:	ОПК-6-31 – Структуру и состав современных средств разработки ПО на основе языков высокого уровня (IDE)
Уметь:	ОПК-6-У1 – Применять встроенные инструменты работы с языком высокого уровня, предоставляемые средством разработки ПО (IDE)
Владеть:	ОПК-6-В1 – Навыками чистого программирования и построения эффективной программной архитектуры
ОПК-8: Способен применять математические модели, методы и средства проектирования информационных и автоматизированных систем, демонстрировать практические навыки для решения задач и реализации проектов, в области, соответствующей профилю подготовки, применять знание экономических, организационных и управленческих вопросов на всех этапах жизненного цикла информационных систем	
Знать:	ОПК-8-31 – Паттерны и методологии разработки ПО на языках высокого уровня в объектно-ориентированной парадигме.
Уметь:	ОПК-8-У1 – Определять области применения паттернов проектирования ПО на языках высокого уровня ОПК-8-У2 – Интерпретировать паттерны в уже написанном программном коде.
Владеть:	ОПК-8-В1 – Навыками применения объектно-ориентированного подхода в решении практических задач
ПК-1: Способен участвовать в работах по:	
• доводке и освоению информационных технологий в ходе внедрения и эксплуатации информационных технологий; • поддержанию работоспособности информационных систем и технологий в заданных функциональных характеристиках и в соответствии с критериями качества; • управлению программно-аппаратными средствами информационных служб инфокоммуникационной системы организации; - обеспечению безопасности и целостности данных информационных систем и технологий, а также условий жизненного цикла информационных систем; • адаптации приложений к изменяющимся условиям функционирования; • администрированию сетевой подсистемы инфокоммуникационной системы организации.	
Использовать технологии разработки объектов профессиональной деятельности в различных областях.	
Знать:	ПК-1-31 – Синтаксис языка высокого уровня (C#) и его основных библиотек для разработки прикладных программ
Уметь:	ПК-1-У1 – Тестировать ПО, написанное на языке высокого уровня, и выявлять уязвимые места
Владеть:	ПК-1-В1 – Инструментами отладки и тестирования кода языков высокого уровня.

4. СТРУКТУРА И СОДЕРЖАНИЕ						
Код занятия	Наименование разделов и тем /вид занятия/	Семестр	Часов	Компетенции	Литература и эл. ресурсы	Примечание
Раздел 1. Поведенческие паттерны						
1.1.	Паттерны поведения из числа GoF / Лек /	4	4	ОПК-8-31 ОПК-8-У1 ОПК-8-У2	Л 1.1, Л 1.2, Л 2.1, Л 2.2, Л 2.3	
1.2.	Паттерны поведения: «Стратегия», «Наблюдатель», «Команда», «Посредник», «Хранитель» и «Цепочка обязанностей» / Лаб /	4	2	ОПК-6-31 ОПК-6-У1 ОПК-6-В1	Л 3.3	
1.3.	Паттерны поведения: «Шаблонный метод», «Итератор», «Состояние», «Интерпретатор» и «Посетитель» / Лаб /	4	2	ОПК-8-У1 ОПК-8-У2 ОПК-8-В1	Л 3.3	
1.4.	Проработка лекционного материала и дополнительной литературы. Подготовка в защите лабораторных работ / С.р. /	4	9	ОПК-6-31 ОПК-6-У1 ОПК-6-В1 ОПК-8-У1	Л 1.1, Л 1.2, Л 2.1, Л 2.2, Л 2.3	
1.5.	Контроль знаний по разделку. Защита лабораторных работ / К.р. /	4	4	ОПК-8-У2 ОПК-8-В1 ОПК-8-31 ОПК-8-У1 ОПК-8-У2	Л 1.1, Л 1.2, Л 2.1, Л 2.2, Л 2.3	
Раздел 2. Структурные паттерны						
2.1.	Паттерны структуры из числа GoF / Лек /	4	2	ОПК-8-31 ОПК-8-У1 ОПК-8-У2	Л 1.1, Л 1.2, Л 2.1, Л 2.2, Л 2.3	
2.2.	Паттерны структуры: «Декоратор», «Адаптер», «Фасад», «Компоновщик», «Прокси», «Мост», «Приспособленец» / Лаб /	4	4	ОПК-6-31 ОПК-6-У1 ОПК-6-В1 ОПК-8-У1 ОПК-8-У2 ОПК-8-В1	Л 3.3	
2.3.	Проработка лекционного материала и дополнительной литературы. Подготовка в защите лабораторных работ / С.р. /	4	4	ОПК-6-31 ОПК-6-У1 ОПК-6-В1 ОПК-8-У1	Л 1.1, Л 1.2, Л 2.1, Л 2.2, Л 2.3	
2.4.	Контроль знаний по разделку. Защита лабораторной работы / К.р. /	4	4	ОПК-8-У2 ОПК-8-В1 ОПК-8-31 ОПК-8-У1 ОПК-8-У2	Л 1.1, Л 1.2, Л 2.1, Л 2.2, Л 2.3	
Раздел 3. Порождающие паттерны						
3.1.	Паттерны порождения из числа GoF / Лек /	4	2	ОПК-8-31 ОПК-8-У1 ОПК-8-У2	Л 1.1, Л 1.2, Л 2.1, Л 2.2, Л 2.3	
3.2.	Паттерны порождения: «Фабричный метод», «Абстрактная фабрика», «Синглтон», «Прототип» и «Строитель» / Лаб /	4	4	ОПК-6-31 ОПК-6-У1 ОПК-6-В1 ОПК-8-У1 ОПК-8-У2 ОПК-8-В1	Л 3.3	
3.3.	Проработка лекционного материала и дополнительной литературы. Подготовка в защите лабораторных работ / С.р. /	4	4	ОПК-6-31 ОПК-6-У1 ОПК-6-В1 ОПК-8-У1	Л 1.1, Л 1.2, Л 2.1, Л 2.2, Л 2.3	
3.4.	Контроль знаний по разделку. Защита лабораторной работ / К.р. /	4	4	ОПК-8-У2 ОПК-8-В1 ОПК-8-31 ОПК-8-У1 ОПК-8-У2	Л 1.1, Л 1.2, Л 2.1, Л 2.2, Л 2.3	
Раздел 4. Паттерн MVVM						
4.1.	Архитектурный паттерн «Model-View-View Model» / Лек /	4	2	ОПК-8-31 ОПК-8-У1 ОПК-8-У2 ПК-1-31	Л 1.1, Л 1.2, Л 2.1, Л 2.2, Л 2.3	
4.2.	Введение в технологию WPF и Bindings / Лек /		4	ОПК-8-31 ОПК-8-У1 ОПК-8-У2 ПК-1-31 ОПК-6-31	Л 1.1, Л 1.2, Л 2.1, Л 2.2, Л 2.3	
4.3.	Приложение WPF в архитектуре MVVM / Лаб /	4	2	ОПК-6-31 ОПК-6-У1	Л 3.3	
4.4.	Работа с бинарными хранилищами. Сериализация и десериализация. Применение	4	3	ОПК-6-В1 ОПК-8-У1	Л 3.3	

	Bindings / Лаб /			ОПК-8-У2 ОПК-8-В1		
4.5.	Проработка лекционного материала и дополнительной литературы. Подготовка в защите лабораторных работ / С.р. /	4	4	ОПК-6-31 ОПК-6-У1 ОПК-6-В1 ОПК-8-У1	Л 1.1, Л 1.2, Л 2.1, Л 2.2, Л 2.3	
4.6.	Контроль знаний по разделку. Защита лабораторной работ / К.р. /	4	4	ОПК-8-У2 ОПК-8-В1 ОПК-8-31 ОПК-8-У1 ОПК-8-У2	Л 1.1, Л 1.2, Л 2.1, Л 2.2, Л 2.3	
Раздел 5. Тестирование программного кода						
5.1.	Введение в юнит и интеграционное тестирование. Классы эквивалентности в тестировании / Лек /	4	3	УК-3-31 ОПК-6-31 ПК-1-У1 УК-2-У1	Л 1.1, Л 1.2, Л 2.1, Л 2.2, Л 2.3	
5.2.	Введение в юнит-тесты при помощи технологии "xUnit" / Пр. /	4	2	УК-2-У1 УК-2-В1 ОПК-6-У1 ОПК-6-В1 ПК-1-У1 ПК-1-В1	Л 3.1	
5.3.	Простые тесты в "xUnit" / Пр. /	4	2		Л 3.1	
5.4.	Тестирование коллекций в "xUnit" / Пр. /	4	2		Л 3.1	
5.5.	Тестирование строк в "xUnit" / Пр. /	4	2		Л 3.1	
5.6.	Тестирование возврата исключений в "xUnit" / Пр. /	4	2		Л 3.1	
5.7.	Интеграционные тесты в "xUnit". Моск-объекты и их применение / Пр. /	4	7		Л 3.1	
5.8.	Проработка лекционного материала и дополнительной литературы. Подготовка в защите практических работ / С.р. /	4	10	УК-2-31 ОПК-6-31 ПК-1-31	Л 1.1, Л 1.2, Л 2.1, Л 2.2, Л 2.3	
5.9.	Контроль знаний по разделку. Защита практических работ. / К.р. /	4	12	УК-2-31 ОПК-6-31 ПК-1-31 УК-2-У1 УК-2-В1 ОПК-6-У1 ОПК-6-В1 ПК-1-У1 ПК-1-В1	Л 1.1, Л 1.2, Л 2.1, Л 2.2, Л 2.3	
Раздел 6. Курсовая работа						
6.1.	Подготовка к выполнению курсовой работы. Согласование темы курсовой работы / С.р. /	4	4	ПК-1-У1 ПК-1-В1	Л 3.2	
6.2.	Выполнение курсовой работы / С.р. /	4	10	ОПК-8-У1 ОПК-8-У2	Л 3.2	
6.3.	Подготовка к защите курсовой работы / С.р. /	4	3	ОПК-8-В1 ОПК-6-У1	Л 1.1, Л 1.2, Л 2.1, Л 2.2, Л 2.3	
6.4.	Сдача курсовых работ. Консультация по этапам курсовой работы. / К.р./	4	33	ПК-1-31 ПК-1-У1 ПК-1-В1 ОПК-8-31 ОПК-8-У1 ОПК-8-У2 ОПК-8-В1 ОПК-6-31 ОПК-6-У1	Л 1.1, Л 1.2, Л 2.1, Л 2.2, Л 2.3	

5. ФОНД ОЦЕНОЧНЫХ МАТЕРИАЛОВ

5.1. Вопросы для самостоятельной подготовки к экзамену (зачёту с оценкой)

5.1.1. Перечень вопросов для подготовки к экзамену. (промежуточный контроль успеваемости)

Раздел №1-3 Паттерны проектирования в ООП

1. Что такое паттерн Singleton и когда его следует использовать в C#?
2. Как реализовать паттерн Factory Method в C#?
3. В чем отличие между Factory Method и Abstract Factory?
4. Как использовать паттерн Builder для создания сложных объектов?
5. В каких случаях применяется паттерн Prototype?
6. Как работает паттерн Adapter и когда он полезен?
7. В чем преимущества использования паттерна Decorator?
8. Как паттерн Facade упрощает взаимодействие с комплексной системой?

9. В чем суть паттерна Bridge и когда его следует применять?
10. Как работает паттерн Proxy и для чего он используется?
11. Как реализовать паттерн Observer для отслеживания изменений состояния объектов?
12. В чем преимущества использования паттерна Strategy?
13. Как паттерн Command инкапсулирует запросы?
14. Как использовать паттерн Iterator для последовательного доступа к элементам коллекции?
15. В чем заключается паттерн Template Method и как его применять?
16. Как паттерн Singleton гарантирует наличие только одного экземпляра класса?
17. Как паттерн Factory Method позволяет подклассам изменять тип создаваемых объектов?
18. Как паттерн Abstract Factory помогает создавать семейства взаимосвязанных объектов?
19. Как паттерн Prototype позволяет создавать новые объекты на основе прототипа?
20. Как паттерн Adapter позволяет использовать несовместимые интерфейсы?
21. Как паттерн Decorator добавляет новое поведение объекту без изменения его структуры?
22. Как паттерн Facade скрывает сложность системы от пользователя?
23. Как паттерн Bridge разделяет абстракцию от реализации?
24. Как паттерн Proxy контролирует доступ к объекту?
25. Как паттерн Observer позволяет объектам реагировать на изменения состояния других объектов?
26. Как паттерн Visitor позволяет добавлять новое поведение к классу без изменения его структуры?
27. В чем преимущества использования паттерна State для управления состоянием объекта?
28. Как паттерн Memento позволяет сохранять и восстанавливать состояние объекта?
29. Как паттерн Interpreter упрощает обработку языковых структур?
30. Как паттерн Factory Method помогает в создании объектов на основе параметров, переданных во время выполнения?

Раздел №4 Паттерн MVVM

1. Что такое паттерн MVVM и какие его основные компоненты?
2. Каковы преимущества использования MVVM в C# по сравнению с другими паттернами?
3. Как ViewModel взаимодействует с View и Model в паттерне MVVM?
4. Какие основные интерфейсы и классы используются в MVVM?
5. Как осуществляется привязка данных в MVVM и какие инструменты для этого предоставляются в C#?
6. Как обрабатывать события в паттерне MVVM?
7. Что такое команды в MVVM и как их реализовать на C#?
8. Как использовать INotifyPropertyChanged в ViewModel для уведомления View об изменениях?
9. Как реализовать валидацию данных в ViewModel согласно паттерну MVVM?
10. Как организовать тестирование ViewModel в паттерне MVVM?
11. Какие лучшие практики следует соблюдать при работе с MVVM в C#?
12. Как MVVM помогает в разделении ответственности между слоями приложения?
13. Какие распространенные ошибки следует избегать при реализации MVVM в C#?
14. Какие основные типы привязок существуют в WPF и как их использовать в зависимости от задачи?
15. Как реализовать двустороннюю привязку в WPF и в каких случаях она может быть полезна?

Раздел №5 Тестирование программного кода

1. Какие основные соглашения следует соблюдать при именовании тестовых проектов и пространств имен в xUnit?
2. Как правильно структурировать тестовые проекты в рамках решения и на диске?
3. Какие основные отличия xUnit от других фреймворков, таких как MSTest и NUnit?
4. Что такое стиль AAA (Arrange-Act-Assert) и как его применять в тестах xUnit?
5. Как использовать атрибуты Fact, Theory и Fixture в xUnit для различных типов тестов?
6. Как проверять исключения в тестах xUnit с помощью ассертов?
7. Какие преимущества дает использование IClassFixture и ICollectionFixture в xUnit?
8. Как организовать выполнение тестов в одном классе и коллекции в xUnit?
9. Как обрабатывать общие ресурсы при выполнении тестов в xUnit?
10. Что такое MemberData и InlineData в контексте Theory тестов в xUnit?
11. Как использовать коллекции для выполнения тестов параллельно в xUnit?
12. Какие лучшие практики следует соблюдать при написании тестов в xUnit для обеспечения их надежности и понятности?
13. Как использовать класс-маркер в xUnit для управления контекстами тестов?
14. Как в xUnit организовать параметризованные тесты с использованием атрибутов, таких как MemberData и Theory?
15. Какие инструменты и подходы можно использовать для генерации тестовых данных в xUnit, чтобы тесты были более реалистичными и полезными?

5.1.2. Перечень контрольных заданий, используемых при формировании экзаменационных билетов при оценке знаний обучающихся на экзамене

Раздел 1-4 Паттерны проектирования в ООП и MVVM

1. Реализуйте фабрику, которая создает различные типы документов (например, WordDocument и PdfDocument). Используйте интерфейс IDocument для создания абстрактного класса фабрики и конкретных классов документов.
2. Реализуйте паттерн “Одиночка” для управления доступом к единственному экземпляру базы данных в приложении. Убедитесь, что только один экземпляр может быть создан и предоставлен всем клиентам.

3. Создайте систему уведомлений, где подписчики получают сообщения при изменении состояния объекта. Реализуйте интерфейсы `IObserver` и `INewsPublisher`, а также классы `NewsSubscriber` и `NewsPublisher`, чтобы продемонстрировать зависимость “один ко многим”.
4. Реализуйте различные стратегии расчета стоимости доставки для интернет-магазина. Используйте интерфейс `IShippingStrategy` для определения методов расчета стоимости и классы, реализующие этот интерфейс для разных методов доставки (например, `StandardShipping`, `ExpressShipping`).
5. Реализуйте фабрику для создания различных транспортных средств (например, `Car`, `Bus`, `Truck`). Создайте базовый класс `Vehicle` и фабричный метод, который будет создавать объекты транспортных средств на основе переданного типа.
6. Создайте структуру каталогов файлов в системе. Используйте паттерн “Компоновщик” для построения древовидной структуры, где каждый узел может быть как листом (файл), так и ветвью (каталог). Реализуйте классы `File` и `Directory`.
7. Реализуйте систему управления пользователями, где пользователи могут иметь различные роли (например, `Admin`, `User`, `Guest`). Используйте паттерн “Мост” для разделения логики управления пользователями и их ролями, чтобы можно было независимо изменять структуру пользователей и их роли.
8. Создайте коллекцию объектов `Book` и реализуйте паттерн “Итератор” для обхода этой коллекции. Реализуйте интерфейсы `IEnumerator` и `IEnumerable` для обеспечения возможности итерации по коллекции с помощью оператора `foreach`.
9. Реализуйте обработку заказов в интернет-магазине, где заказ проходит через несколько этапов обработки (например, `Validation`, `Shipping`, `Billing`). Используйте паттерн “Цепочка ответственности” для последовательной передачи заказа от одного обработчика к другому, пока заказ не будет полностью обработан.
10. Создайте простой язык программирования (команды сложения, умножения, вычитания и деления) с использованием стека для хранения выражений. Реализуйте интерпретатор для выполнения этих выражений, используя паттерн “Интерпретатор”. Определите классы для различных типов выражений и реализуйте интерпретацию выражений в классе интерпретатора.
11. Создайте класс `Person` с полями имя, фамилия и возраст. Напишите метод для сериализации объекта `Person` в бинарный формат и сохраните его в файл. Реализуйте метод для десериализации бинарного файла обратно в объект `Person`.
12. Определите класс `Book` с полями название, автор и год издания. Создайте список объектов `Book` и сериализуйте его в строку `JSON`. Десериализуйте строку `JSON` обратно в список объектов `Book`.
13. Создайте классы `Author` и `Book` с соответствующими полями. Создайте объект `Book` с вложенным объектом `Author` и сериализуйте его в `JSON`. Десериализуйте `JSON` обратно в объект `Book` с корректным восстановлением вложенного объекта `Author`.
14. Определите базовый класс `Animal` и производные классы `Dog` и `Cat`. Сериализуйте список животных (смесь собак и кошек) в бинарный формат. Десериализуйте бинарный файл обратно в список объектов `Animal`.
15. Подключите библиотеку `Newtonsoft.Json` для работы с `JSON`. Создайте класс `Customer` с полями имя, адрес и номер телефона. Напишите методы для сериализации и десериализации объектов `Customer` в `JSON` и обратно, используя методы из подключенной библиотеки.

Раздел 4 Тестирование программного кода

1. Используя фреймворк `xUnit`, создайте тест для метода, который принимает на вход два целых числа, и возвращает результат их сложения в двоичной форме в виде массива;
2. Используя фреймворк `xUnit`, напишите тест для проверки исключения, возникающего при делении на ноль, при обращении к несуществующему индексу массива и при попытке обратиться к `NULL`-объекту;
3. Используя фреймворк `xUnit`, проверьте, что метод изменения пароля пользователя вызывает исключение при вводе старого пароля.
4. Используя фреймворк `xUnit`, создайте интеграционный тест для проверки взаимодействия двух классов в проекте;
5. Используя фреймворк `xUnit`, напишите тест для метода сортировки массива пузырьком, убедившись, что элементы отсортированы по возрастанию.
6. Используя фреймворк `xUnit`, создайте тест для проверки работы метода, удаляющего элемент из списка.
7. Используя фреймворк `xUnit`, напишите тест для метода, который проверяет, является ли строка палиндромом.
8. Используя фреймворк `xUnit`, проверьте, что метод сложения матриц возвращает правильную сумму.
9. Используя фреймворк `xUnit`, создайте тест для проверки работы метода округления числа до двух знаков после запятой не используя класс `Math`;
10. Используя фреймворк `xUnit`, напишите тест для класса, который отвечает за авторизацию пользователя по логину и паролю и в случае успешной авторизации генерирует 512-битный строковый ключ, а в случае ошибки исключение. Обеспечьте максимальное покрытие тестами этого класса;
11. Используя фреймворк `xUnit`, проверьте, что метод генерации случайного числа в заданном диапазоне работает корректно, без использования `Random`;
12. Используя фреймворк `xUnit`, создайте тест для метода, возвращающего коллекцию дат ближайших дней недели относительно введенной даты. Обеспечьте покрытие тестами.
13. Используя фреймворк `xUnit`, напишите тест для метода, проверяющего, является ли переданное значение в метод 192-битным ключом;
14. Используя фреймворк `xUnit`, напишите тесты с использованием атрибута `[Theory]` и `[InlineData]` для метода, который возвращает оценку на основе набранных баллов из 10. Протестируйте несколько значений входных параметров и убедитесь, что результаты соответствуют ожидаемым строковым значениям;
15. Используя фреймворк `xUnit`, напишите тест для метода, который принимает на вход массив строк произвольной длины и возвращает обратно массив индексов строк, которые при конкатенации образуют тандем (четная строка, у которой левая половина равна правой половине).

5.1.3. ПЕРЕЧЕНЬ ВОПРОСОВ К ЗАЩИТЕ ЛАБОРАТОРНЫХ РАБОТ

Лабораторная работа №1 «Паттерны поведения: «Стратегия», «Наблюдатель», «Команда», «Посредник», «Хранитель» и «Цепочка обязанностей»

1. Как паттерн «Стратегия» помогает выбирать различные подходы к выполнению задачи в зависимости от заданных условий? Приведите пример использования этого паттерна в реальной жизни.
2. Как работает паттерн «Наблюдатель» и в каких ситуациях он может быть полезен? Приведите пример, где этот паттерн используется для подписки на события.
3. В чем заключается основная идея паттерна «Команда» и как он позволяет абстрагироваться от деталей выполнения действий? Приведите пример использования этого паттерна в быту.
4. Какова роль паттерна «Посредник» и как он управляет взаимодействием между объектами? Приведите пример, где посредник может быть полезен в организации процессов.
5. В каких случаях используется паттерн «Хранитель» и как он помогает сохранять и восстанавливать состояние объекта? Приведите пример из повседневной жизни, где этот паттерн может быть применен.
6. Как работает паттерн «Цепочка обязанностей» и в каких ситуациях он может быть полезен? Приведите пример процесса, где используется этот паттерн для обработки запросов.

Лабораторная работа №2 «Паттерны поведения: «Шаблонный метод», «Итератор», «Состояние», «Интерпретатор» и «Посетитель»

1. Как паттерн «Шаблонный метод» позволяет определить основу алгоритма и предоставить подклассам возможность переопределять отдельные шаги? Приведите пример использования этого паттерна в программировании.
2. В чем заключается основная идея паттерна «Итератор» и как он позволяет абстрагировать процесс обхода коллекций? Приведите пример использования итератора в реальных проектах.
3. Как паттерн «Состояние» помогает управлять изменением поведения объекта в зависимости от его состояния? Приведите пример, где этот паттерн может быть полезен в повседневной жизни.
4. Какова роль паттерна «Интерпретатор» и как он используется для интерпретации языковых конструкций? Приведите пример применения этого паттерна в разработке программного обеспечения.
5. Как паттерн «Посетитель» позволяет отделить алгоритм от структуры объектов и добавить новые операции без изменения классов объектов? Приведите пример использования посетителя в объектно-ориентированном программировании.

Лабораторная работа №3 «Паттерны структуры: «Декоратор», «Адаптер», «Фасад», «Компоновщик», «Прокси», «Мост», «Приспособленец»

1. Что такое паттерн «Декоратор» и как он используется в программировании?
2. В чем основное отличие паттерна «Адаптер» от других структурных паттернов и как он помогает в интеграции несовместимых интерфейсов?
3. Как паттерн «Фасад» упрощает взаимодействие с большим и сложным набором классов, и какие преимущества он предоставляет?
4. В чем заключаются ключевые особенности паттерна «Компоновщик» и как он отличается от других структурных паттернов при работе с древовидными структурами?
5. Как паттерн «Прокси» используется для управления доступом к ресурсу и какие преимущества он предоставляет по сравнению с прямым доступом?
6. В чем заключается суть паттерна «Мост» и как он помогает разделять абстракцию и реализацию, улучшая гибкость и поддерживаемость кода?
7. Как работает паттерн «Приспособленец» и в каких случаях его использование наиболее целесообразно?

Лабораторная работа №4 «Паттерны порождения: «Фабричный метод», «Абстрактная фабрика», «Синглтон», «Прототип» и «Строитель»

1. В чем заключается суть паттерна «Фабричный метод» и как он позволяет создавать объекты без указания конкретных классов?
2. Как паттерн «Абстрактная фабрика» помогает в создании семейств связанных продуктов, и чем он отличается от других порождающих паттернов?
3. В чем основные преимущества использования паттерна «Синглтон» и как он гарантирует, что будет создан только один экземпляр класса?
4. Как паттерн «Прототип» позволяет создавать новые объекты путем клонирования существующих, и в каких случаях это особенно полезно?
5. В чем заключаются ключевые особенности паттерна «Строитель» и как он отделяет конструирование сложного объекта от его представления?

Лабораторная работа №5 «Приложение WPF в архитектуре MVVM»

1. Что такое архитектура MVVM и какие основные компоненты она включает?
2. Как в архитектуре MVVM осуществляется связывание данных между View и ViewModel?
3. Какие преимущества предоставляет использование MVVM в WPF-приложениях по сравнению с другими архитектурными моделями?
4. Как реализуется команда (Command) в WPF-приложении с использованием MVVM и для чего она нужна?
5. Какие инструменты и библиотеки часто используются в WPF-приложениях с архитектурой MVVM для упрощения разработки и тестирования?
6. Какие лучшие практики следует соблюдать при разработке WPF-приложений с использованием архитектуры MVVM для обеспечения их гибкости и поддерживаемости?

Лабораторная работа №6 «Работа с бинарными хранилищами. Сериализация и десериализация. Применение Bindings»

1. Что такое сериализация и десериализация, и какие основные методы используются для работы с бинарными хранилищами в программировании?
2. Какие преимущества предоставляет использование бинарных хранилищ по сравнению с текстовыми форматами данных?
3. Как работают привязки (Bindings) в контексте сериализации и десериализации объектов, и какие механизмы используются для обеспечения корректной работы с данными?
4. Какие библиотеки и фреймворки часто используются для сериализации и десериализации объектов в бинарные хранилища, и какие особенности они предоставляют?
5. Какие проблемы могут возникнуть при работе с бинарными хранилищами и как их можно решить с помощью правильной реализации сериализации и десериализации?
6. Как осуществляется управление версиями объектов при сериализации и десериализации в бинарных хранилищах, и какие подходы могут быть использованы для минимизации проблем, связанных с изменениями в структуре данных?
7. Какие лучшие практики следует соблюдать при работе с бинарными хранилищами для обеспечения безопасности и производительности при сериализации и десериализации данных?

5.1.4. ПЕРЕЧЕНЬ ВОПРОСОВ К ЗАЩИТЕ ПРАКТИЧЕСКИХ РАБОТ

Практическая работа №1 «Введение в юнит-тесты при помощи технологии “xUnit”»

1. Что такое xUnit и какие основные преимущества он предоставляет по сравнению с другими фреймворками для юнит-тестирования?
2. Как создать базовый юнит-тест с использованием xUnit, и какие ключевые атрибуты используются для обозначения тестов и настройки тестового окружения?
3. Какие основные методы и подходы используются в xUnit для написания параметризованных тестов и как они помогают в повышении эффективности тестирования?
4. Как xUnit обеспечивает изоляцию тестов и какие механизмы используются для управления состоянием между тестами в одном классе?
5. Какие встроенные утверждения (assertions) предоставляет xUnit и как их использовать для проверки результатов юнит-тестов?

Практическая работа №2 «Простые тесты в “xUnit”»

1. Как создать и структурировать простой юнит-тест с использованием xUnit, и какие атрибуты необходимы для его обозначения?
2. Какие основные методы используются в xUnit для написания и выполнения простых юнит-тестов, и как они помогают в проверке логики кода?
3. Как в xUnit обеспечить изоляцию тестов, чтобы каждый тест выполнялся в отдельном окружении?
4. Как организовать и структурировать простые тесты в xUnit для улучшения читаемости и поддержки кода?
5. Какие лучшие практики следует соблюдать при написании простых юнит-тестов в xUnit для обеспечения их надежности и понятности?
6. Как в xUnit реализовать параметризованные тесты для проверки различных сценариев с использованием одного теста?

Практическая работа №3 «Тестирование коллекций в “xUnit”»

1. Как в xUnit тестировать методы, возвращающие коллекции, и какие атрибуты и методы использовать для проверки содержимого этих коллекций?
2. Какие подходы существуют в xUnit для параметризации тестов коллекций, и как они помогают в написании более гибких и повторно используемых тестов?
3. Как реализовать тестирование добавления и удаления элементов в коллекции с использованием xUnit, и какие методы утверждений (assertions) при этом применяются?
4. Как тестировать сортировку и фильтрацию коллекций в xUnit, и какие инструменты и атрибуты фреймворка помогают в этом процессе?
5. Как в xUnit тестировать итераторы и генераторы коллекций, и какие методы использовать для проверки их выходных данных?

Практическая работа №4 «Тестирование строк в “xUnit”»

1. Как в xUnit тестировать методы, возвращающие строки, и какие атрибуты и методы использовать для проверки содержимого строк?
2. Какие встроенные утверждения (assertions) предоставляет xUnit для проверки равенства и содержимого строк, и как их правильно применять?
3. Как реализовать тестирование форматирования строк в xUnit, и какие подходы использовать для проверки корректности формата?
4. Как в xUnit тестировать методы, изменяющие строки, такие как обрезка пробелов или замена подстрок, и какие атрибуты использовать для этих тестов?
5. Какие лучшие практики следует соблюдать при тестировании строк в xUnit для обеспечения надежности и понятности тестов?

Практическая работа №5 «Тестирование возврата исключений в “xUnit”»

1. Как в xUnit тестировать методы, выбрасывающие исключения, и какие атрибуты использовать для обозначения

таких тестов? 2. Какие подходы существуют в xUnit для проверки, что метод выбрасывает ожидаемое исключение при определенных условиях? 3. Как реализовать тестирование обработки исключений в xUnit, и какие методы утверждений (assertions) при этом применяются? 4. Как в xUnit тестировать различные типы исключений, которые могут быть выброшены методом, и как проверять их свойства? 5. Какие лучшие практики следует соблюдать при тестировании выбрасывания исключений в xUnit для обеспечения надёжности и понятности тестов? 6. Как в xUnit тестировать обработку исключений с использованием параметризованных тестов, и какие преимущества это даёт? 7. Как в xUnit реализовать тестирование, чтобы удостовериться, что исключения не выбрасываются в некорректных условиях, и какие атрибуты и методы для этого использовать? Практическая работа №6 «Интеграционные тесты в “xUnit”. Mock-объекты и их применение» 1. Как в xUnit использовать mock-объекты для изоляции тестируемого кода и уменьшения зависимостей на внешние сервисы? 2. Какие библиотеки для создания mock-объектов поддерживаются в xUnit, и как их интегрировать в тестовый проект? 3. Как в xUnit настроить и использовать mock-объекты для проверки взаимодействия с зависимостями в модульных тестах? 4. Какие преимущества даёт использование mock-объектов при написании модульных тестов в xUnit, и как это влияет на их надёжность и скорость выполнения? 5. Как в xUnit реализовать тестирование логики кода, взаимодействующего с внешними системами, с использованием mock-объектов для имитации ответов этих систем? 6. Какие лучшие практики следует соблюдать при использовании mock-объектов в xUnit для обеспечения понятности и поддерживаемости тестов? 7. Как в xUnit тестировать асинхронные методы с использованием mock-объектов, и какие подходы существуют для проверки их поведения и взаимодействия с зависимостями?	5.2. Перечень работ, выполняемых по дисциплине В перечень работ, выполняемых в процессе изучения дисциплины в семестре 4 включены: 1. Лабораторные работы a. Лабораторная работа №1 «Паттерны поведения: «Стратегия», «Наблюдатель», «Команда», «Посредник», «Хранитель» и «Цепочка обязанностей»; b. Лабораторная работа №2 «Паттерны поведения: «Шаблонный метод», «Итератор», «Состояние», «Интерпретатор» и «Посетитель»; c. Лабораторная работа №3 «Паттерны структуры: «Декоратор», «Адаптер», «Фасад», «Компоновщик», «Прокси», «Мост», «Приспособленец»; d. Лабораторная работа №4 «Паттерны порождения: «Фабричный метод», «Абстрактная фабрика», «Синглтон», «Прототип» и «Строитель»; e. Лабораторная работа №5 «Приложение WPF в архитектуре MVVM»; f. Лабораторная работа №6 «Работа с бинарными хранилищами. Сериализация и десериализация. Применение Bindings»; 2. Практические работы a. Практическая работа №1 «Введение в юнит-тесты при помощи технологии “xUnit”»; b. Практическая работа №2 «Простые тесты в “xUnit”»; c. Практическая работа №3 «Тестирование коллекций в “xUnit”»; d. Практическая работа №4 «Тестирование строк в “xUnit”»; e. Практическая работа №5 «Тестирование возврата исключений в “xUnit”»; f. Практическая работа №6 «Интеграционные тесты в “xUnit”. Mock-объекты и их применение». 3. Курсовая работа a. Курсовая работа «Проектная реализация программного продукта по индивидуальному ТЗ»; <u>Тема: «Проектная реализация программного продукта по индивидуальному ТЗ»</u> Задание для данной курсовой работы предполагает проверку навыков применения паттернов проектирования, умений работать с тестовым окружением и применять инструментальные средства интегрированных средств разработки. Курсовая работа предполагает творческое задание на подобию реальной задачи программирования, упрощенной до базового уровня владения языком. Выдача академических «учебных» задач и распределение по вариантам не предусматривается. Пример ТЗ на естественном языке: 1. Спроектировать и реализовать автоматизированное рабочее место для специалиста логистики поставок с использованием технологии WPF и паттерна MVVM. Программа должна содержать интуитивно понятный графический интерфейс, иметь возможность авторизации и настройки интерфейса. В основе данной программы должен находиться механизм решения транспортной задачи поставок. Подход к проектированию свободный, язык программирования – C#. Курсовая работа должна содержать: Отчет должен включать: 1. Титульный лист 2. Лист консультаций;
---	--

3. Лист задания;
4. Содержание;
5. Введение;
6. Аналитическая часть:
 - a. Постановка задачи;
 - b. Декомпозиция задачи на подзадачи;
 - c. Проектирование классов:
 - i. Разработка классовой структуры;
 - ii. Выделение основных классов;
 - iii. Создание UML диаграммы классов (статический вариант);
 - iv. Определения типов и видов наследования;
 - v. Формальное описание каждого класса на естественном языке с описанием полей, методов, свойств.
 - d. Проектирование бизнес-логики:
 - i. Разработка алгоритмов в соответствии с подзадачами (п.5b);
 - ii. Формальное описание алгоритмов с детализацией функционала (если имеются формулы для расчета, то указать их).
7. Практическая часть:
 - a. Описание используемых библиотек;
 - b. Листинг кода реализации отдельных классов (код должен быть закомментирован);
 - c. Листинг программы с точкой входа (код должен быть закомментирован);
8. Заключение;
9. Список использованных источников (не менее 10, включая корректно оформленные ссылки на интернет-источники. См. ГОСТ 7.1-2003 и ГОСТ 7.0.100-2018).

Защита курсовой работы осуществляется путем защиты кода программы и алгоритмов в нотации блок-схем.

5.2.1 Перечень тестовых вопросов

- 1) Какой паттерн используется для определения интерфейса для создания объектов, но позволяет подменять реализации на лету?
 - a) Singleton;
 - b) Factory Method;
 - c) Adapter;
 - d) Observer.
- 1) Какой паттерн позволяет объекту изменять свое поведение в зависимости от данных, с которыми он работает?
 - a) Strategy;
 - b) Decorator;
 - c) Composite;
 - d) Facade.
- 2) Какой паттерн предоставляет унифицированный интерфейс для работы с различными объектами, скрывая их реальные различия?
 - a) Proxy;
 - b) Bridge;
 - c) Singleton;
 - d) Interpreter.
- 3) Какой паттерн используется для отделения зависимости на интерфейс, чтобы упростить тестирование?
 - a) Dependency Injection;
 - b) Abstract Factory;
 - c) Prototype;
 - d) Visitor.
- 4) Какой паттерн позволяет объединять объекты в древовидные структуры и работать с ними как с единым целым?
 - a) Composite;
 - b) Singleton;
 - c) Proxy;
 - d) Bridge.
- 5) Какой паттерн предназначен для управления доступом к объекту через единый интерфейс, скрывая детали реализации?
 - a) Facade;
 - b) Adapter;
 - c) Decorator;
 - d) Proxy.
- 6) Какой паттерн позволяет динамически добавлять поведение к объекту, не изменяя его структуру?
 - a) Decorator;
 - b) Strategy;
 - c) Singleton;
 - d) Observer.
- 7) Какой паттерн используется для создания объектов, которые могут заменять друг друга на стадии выполнения программы?
 - a) Factory Method;
 - b) Prototype;
 - c) Singleton;

- d) Interpreter.
- 8) Какой паттерн позволяет объекту принимать различные формы, предоставляя общий интерфейс для разных классов?
- a) Adapter;
 - b) Bridge;
 - c) Composite;
 - d) Facade.
- 9) Какой паттерн используется для управления созданием объектов и обеспечения единого интерфейса для их получения?
- a) Singleton;
 - b) Abstract Factory;
 - c) Prototype;
 - d) Interpreter.
- 10) Какой паттерн используется для объединения объектов в древовидные структуры и работы с ними как с единым целым?
- a) Singleton;
 - b) Composite;
 - c) Adapter;
 - d) Observer.
- 11) Какой паттерн позволяет объединять объекты в древовидные структуры, но также может работать и с другими типами структур?
- a) Composite;
 - b) Singleton;
 - c) Proxy;
 - d) Bridge.
- 12) Какой паттерн предназначен для упрощения взаимодействия с библиотеками и фреймворками, которые имеют сложные интерфейсы?
- a) Adapter;
 - b) Bridge;
 - c) Singleton;
 - d) Interpreter.
- 13) Какой паттерн позволяет объектам взаимодействовать друг с другом без необходимости знать конкретные классы друг друга?
- a) Observer;
 - b) Strategy;
 - c) Singleton;
 - d) Interpreter.
- 14) Какой паттерн используется для определения интерфейса для создания объектов, но позволяет подменять реализации на лету?
- a) Singleton;
 - b) Factory Method;
 - c) Adapter;
 - d) Observer.
- 15) Какой паттерн позволяет динамически добавлять поведение к объекту, не изменяя его структуру?
- a) Decorator;
 - b) Strategy;
 - c) Singleton;
 - d) Observer.
- 16) Какой паттерн используется для отделения зависимости на интерфейс, чтобы упростить тестирование?
- a) Dependency Injection;
 - b) Abstract Factory;
 - c) Prototype;
 - d) Visitor.
- 17) Какой паттерн позволяет добавлять новые функции к объекту, не изменяя его существующую структуру?
- a) Decorator;
 - b) Strategy;
 - c) Singleton;
 - d) Observer.
- 18) Какой паттерн используется для создания объектов, которые могут заменять друг друга на стадии выполнения программы?
- a) Factory Method;
 - b) Prototype;
 - c) Singleton;
 - d) Interpreter.
- 19) Какой паттерн позволяет объекту принимать различные формы, предоставляя общий интерфейс для разных классов?
- a) Adapter;
 - b) Bridge;
 - c) Composite;
 - d) Facade.

- 20) Какой паттерн используется для определения семейства алгоритмов, позволяя подменять их на стадии выполнения программы?
- a) Singleton;
 - b) Strategy;
 - c) Observer;
 - d) Interpreter.
- 21) Какой паттерн позволяет объекту изменять свое поведение в зависимости от данных, с которыми он работает?
- a) Strategy;
 - b) Decorator;
 - c) Composite;
 - d) Facade.
- 22) Какой паттерн предоставляет унифицированный интерфейс для работы с различными объектами, скрывая их реальные различия?
- a) Proxy;
 - b) Bridge;
 - c) Singleton;
 - d) Interpreter.
- 23) Какой паттерн используется для отделения зависимости на интерфейс, чтобы упростить тестирование?
- a) Dependency Injection;
 - b) Abstract Factory;
 - c) Prototype;
 - d) Visitor.
- 24) Какой паттерн позволяет динамически добавлять поведение к объекту, не изменяя его структуру?
- a) Decorator;
 - b) Strategy;
 - c) Singleton;
 - d) Observer.
- 25) Какой паттерн используется для создания объектов, которые могут заменять друг друга на стадии выполнения программы?
- a) Factory Method;
 - b) Prototype;
 - c) Singleton;
 - d) Interpreter.
- 26) Какой паттерн позволяет объекту принимать различные формы, предоставляя общий интерфейс для разных классов?
- a) Adapter;
 - b) Bridge;
 - c) Composite;
 - d) Facade.
- 27) Какой паттерн используется для управления созданием объектов и обеспечения единого интерфейса для их получения?
- a) Singleton;
 - b) Abstract Factory;
 - c) Prototype;
 - d) Interpreter.
- 28) Какой паттерн позволяет добавлять поведение к объекту динамически, не изменяя его структуру?
- a) Decorator;
 - b) Strategy;
 - c) Singleton;
 - d) Observer.
- 29) Какой паттерн используется для обеспечения единого интерфейса для работы с различными системами?
- a) Facade;
 - b) Adapter;
 - c) Bridge;
 - d) Proxy.
- 30) Что является основной целью паттерна MVVM?
- a) Отделение логики приложения от пользовательского интерфейса;
 - b) Упрощение тестирования;
 - c) Повышение производительности;
 - d) Все вышеперечисленное.
- 31) Какие основные компоненты включает в себя паттерн MVVM?
- a) Model, View, ViewModel;
 - b) Model, View, ViewController;
 - c) Model, ViewModel, View;
 - d) Model, View, Controller.
- 32) Какая роль у компонента Model в паттерне MVVM?
- a) Управление данными и бизнес-логикой;
 - b) Отображение данных пользователю;
 - c) Связующее звено между Model и ViewModel;
 - d) Связующее звено между ViewModel и View.

- 33) Что делает компонент ViewModel в паттерне MVVM?
- a) Управляет данными и бизнес-логикой;
 - b) Отображает данные пользователю;
 - c) Связывает Model и View;
 - d) Связывает Controller и View.
- 34) Как ViewModel взаимодействует с View?
- a) Через прямые вызовы методов;
 - b) Через события и команды;
 - c) Через контроллер;
 - d) Через Model.
- 35) Какой инструмент часто используется для связывания данных в MVVM?
- a) ADO.NET;
 - b) Entity Framework;
 - c) Data Binding;
 - d) LINQ.
- 36) Что такое команды в контексте паттерна MVVM?
- a) Методы для управления данными;
 - b) События, которые вызывают действия;
 - c) Классы для работы с моделями;
 - d) Интерфейсы для взаимодействия с View.
- 37) Как осуществляется тестирование логики приложения при использовании MVVM?
- a) Через тестирование Model;
 - b) Через тестирование ViewModel;
 - c) Через тестирование View;
 - d) Через все компоненты.
- 38) Что такое DataContext в контексте MVVM?
- a) Контекст данных для Model;
 - b) Контекст данных для ViewModel;
 - c) Контекст данных для View;
 - d) Контекст данных для Controller.
- 39) Какие преимущества предоставляет паттерн MVVM при разработке приложений?
- a) Легкость тестирования и поддержка разделения ответственностей;
 - b) Ускорение выполнения кода;
 - c) Простота создания пользовательского интерфейса;
 - d) Все вышеперечисленное.
- 40) Что такое WPF?
- a) Windows Presentation Foundation;
 - b) Windows Process Framework;
 - c) Windows Protocol Foundation;
 - d) Windows Programming Framework.
- 41) Какой язык программирования используется для создания приложений с WPF?
- a) C#;
 - b) Python;
 - c) Java;
 - d) JavaScript.
- 42) Что такое XAML в контексте WPF?
- a) Язык для описания пользовательского интерфейса
 - b) Библиотека для работы с данными;
 - c) Фреймворк для веб-приложений;
 - d) Инструмент для тестирования.
- 43) Какие элементы управления включены в WPF?
- a) Button, TextBox, ListBox;
 - b) Form, Edit, List;
 - c) Menu, ComboBox, RadioButton;
 - d) Все вышеперечисленное.
- 44) Что такое DataBinding в WPF?
- a) Процесс связывания данных с элементами управления;
 - b) Метод шифрования данных;
 - c) Инструмент для создания анимации;
 - d) Технология для работы с базами данных.
- 45) Какая библиотека используется для создания анимаций в WPF?
- a) System.Windows.Media.Animation;
 - b) Microsoft.Animation;
 - c) WPF.Animation;
 - d) Windows.Animation.
- 46) Что такое Dependency Property в WPF?
- a) Свойство, которое зависит от других свойств;
 - b) Свойство, которое может быть изменено только в коде;
 - c) Свойство, которое автоматически обновляет пользовательский интерфейс;
 - d) Свойство, используемое только для хранения данных.

- 47) Какой элемент управления используется для отображения списков данных в WPF?
- a) ListView;
 - b) DataList;
 - c) GridView;
 - d) ListBox.
- 48) Что такое Styles в WPF?
- a) Шаблоны для оформления элементов управления;
 - b) Методы для обработки событий;
 - c) Инструменты для работы с базами данных;
 - d) Элементы пользовательского интерфейса.
- 49) Какая технология используется для создания интерактивных пользовательских интерфейсов в WPF?
- a) Commands;
 - b) Events;
 - c) Actions;
 - d) Triggers.
- 50) Что такое ControlTemplate в WPF?
- a) Шаблон для создания новых элементов управления;
 - b) Инструмент для работы с данными;
 - c) Метод для анимации элементов;
 - d) Механизм для обработки событий.
- 51) Что такое сериализация JSON?
- a) Процесс преобразования данных в строку;
 - b) Процесс преобразования данных в объект;
 - c) Процесс преобразования данных в массив;
 - d) Процесс преобразования данных в базу данных.
- 52) Что такое десериализация JSON?
- a) Процесс преобразования строки в объект;
 - b) Процесс преобразования объекта в строку;
 - c) Процесс преобразования массива в объект;
 - d) Процесс преобразования базы данных в объект.
- 53) Какая библиотека в C# используется для работы с JSON?
- a) Newtonsoft.Json;
 - b) System.Json;
 - c) Microsoft.Json;
 - d) Json.Net
- 54) Какой метод используется для сериализации объекта в строку JSON в C#?
- a) Serialize();
 - b) ToString();
 - c) JsonConvert.SerializeObject();
 - d) JsonSerializer().
- 55) Какой метод используется для десериализации строки JSON в объект в C#?
- a) Deserialize();
 - b) FromJson();
 - c) JsonConvert.DeserializeObject();
 - d) DeserializeString().
- 56) Какой тип данных в C# используется для десериализации JSON-строки в объект?
- a) String;
 - b) Object;
 - c) Dynamic;
 - d) JObject.
- 57) Как сериализовать список объектов в JSON в C#?
- a) Использовать метод SerializeList();
 - b) Использовать метод JsonConvert.SerializeList();
 - c) Использовать метод JsonConvert.SerializeObject() с параметром List<T>;
 - d) Использовать метод List.ToJson().
- 58) Как десериализовать JSON-строку в список объектов в C#?
- a) Использовать метод DeserializeList();
 - b) Использовать метод JsonConvert.DeserializeList();
 - c) Использовать метод JsonConvert.DeserializeObject() с параметром List<T>;
 - d) Использовать метод List.FromJson().
- 59) Как обработать JSON-объект, содержащий вложенные объекты и массивы, в C#?
- a) Использовать метод Flatten();
 - b) Использовать метод FlattenObject();
 - c) Использовать метод JsonConvert.DeserializeObject() с параметром Depth: 10;
 - d) Использовать метод DeserializeNested().
- 60) Как игнорировать определенные свойства при сериализации объекта в JSON в C#?
- a) Использовать атрибут Ignore;
 - b) Использовать метод IgnoreProperty();
 - c) Использовать атрибут Exclude;
 - d) Использовать метод ExcludeProperty().

- 61) Что такое бинарная сериализация?
- a) Преобразование данных в строку;
 - b) Преобразование данных в бинарный формат;
 - c) Преобразование данных в XML;
 - d) Преобразование данных в JSON.
- 62) Что такое десериализация?
- a) Процесс преобразования бинарных данных в объект;
 - b) Процесс преобразования объекта в бинарный формат;
 - c) Процесс преобразования строки в объект;
 - d) Процесс преобразования XML в объект.
- 63) Какая библиотека в C# используется для бинарной сериализации и десериализации?
- a) Newtonsoft.Json;
 - b) System.Text.Json;
 - c) BinaryFormatter;
 - d) DataContractSerializer.
- 64) Какой метод используется для сериализации объекта в бинарный формат в C#?
- a) Serialize();
 - b) BinarySerialize();
 - c) WriteObject();
 - d) SerializeToBinary().
- 65) Какой метод используется для десериализации бинарного файла в объект в C#?
- a) Deserialize();
 - b) ReadObject();
 - c) DeserializeFromBinary();
 - d) DeserializeBinary().
- 66) Какой атрибут используется для указания, что класс должен быть сериализован с помощью DataContractSerializer?
- a) [Serializable];
 - b) [DataContract];
 - c) [BinarySerialize];
 - d) [SerializableAttribute].
- 67) Как сериализовать список объектов в бинарный файл в C#?
- a) Использовать метод SerializeList();
 - b) Использовать метод WriteList();
 - c) Использовать метод BinarySerializeList();
 - d) Использовать метод SerializeObjectList().
- 68) Как десериализовать бинарный файл в список объектов в C#?
- a) Использовать метод DeserializeList();
 - b) Использовать метод ReadList();
 - c) Использовать метод BinaryDeserializeList();
 - d) Использовать метод DeserializeObjectList().
- 69) Как обрабатывать исключения при бинарной сериализации и десериализации в C#?
- a) Использовать блок try-catch;
 - b) Использовать метод TrySerialize();
 - c) Использовать атрибут [OnError];
 - d) Использовать метод HandleException().
- 70) Как игнорировать определенные свойства при бинарной сериализации объекта в C#?
- a) Использовать атрибут [Ignore];
 - b) Использовать атрибут [Exclude];
 - c) Использовать метод IgnoreProperty();
 - d) Использовать метод ExcludeProperty().
- 71) Что такое xUnit?
- a) Фреймворк для написания юнит-тестов;
 - b) Библиотека для работы с данными;
 - c) Инструмент для создания анимации;
 - d) Платформа для разработки веб-приложений.
- 72) Какой синтаксис используется в xUnit для написания тестов?
- a) test();
 - b) Assert.AreEqual();
 - c) Fact() и Theory();
 - d) runTest().
- 73) Как создать новый юнит-тест в xUnit?
- a) Использовать метод new Test();
 - b) Создать класс с методами Fact() и Theory();
 - c) Написать метод с атрибутом Test;
 - d) Вызвать функцию createTest().
- 74) Что такое Theory в xUnit?
- a) Метод для проверки условий;
 - b) Атрибут для тестирования;
 - c) Функция для генерации данных;
 - d) Класс для хранения тестов.

- 75) Какой атрибут используется для обозначения юнит-теста в xUnit?
- a) [Test];
 - b) [Theory];
 - c) [Fact];
 - d) [Example].
- 76) Как запустить юнит-тесты в xUnit?
- a) Вызвать метод runTests();
 - b) Использовать команду dotnet test;
 - c) Запустить проект в режиме отладки;
 - d) Нажать кнопку “Выполнить”.
- 77) Что такое Assert в xUnit?
- a) Метод для проверки условий;
 - b) Класс для хранения данных;
 - c) Функция для генерации тестов;
 - d) Атрибут для запуска тестов.
- 78) Как обозначить параметризованный тест в xUnit?
- a) Использовать метод param();
 - b) Создать класс с атрибутом Param;
 - c) Написать метод с атрибутом Theory;
 - d) Вызвать функцию parameterizeTest().
- 79) Что такое Fact в xUnit?
- a) Атрибут для проверки условий;
 - b) Метод для генерации данных;
 - c) Функция для запуска тестов;
 - d) Класс для хранения тестов.
- 80) Как обозначить тест, который не будет учитываться при подсчете покрытия кода в xUnit?
- a) Использовать атрибут [Ignore];
 - b) Написать метод с атрибутом [Fact];
 - c) Создать класс с атрибутом [Theory];
 - d) Вызвать функцию excludeTest().
- 81) Какой метод в xUnit используется для проверки равенства двух строк?
- a) Assert.Equal();
 - b) Assert.AreEqual();
 - c) Assert.Equals();
 - d) Assert.Same().
- 82) Как проверить, что строка содержит определенную подстроку в xUnit?
- a) Assert.Contains();
 - b) Assert.HasSubstring();
 - c) Assert.Include();
 - d) Assert.Substring().
- 83) Какой метод используется для проверки длины строки в xUnit?
- a) Assert.Length();
 - b) Assert.StringLength();
 - c) Assert.IsLength();
 - d) Assert.EqualLength().
- 84) Как проверить, что две строки не равны в xUnit?
- a) Assert.NotEqual();
 - b) Assert.AreNotEqual();
 - c) Assert.NotEquals();
 - d) Assert.Different().
- 85) Какой метод используется для проверки того, что строка начинается с определенного префикса?
- a) Assert.StartsWith();
 - b) Assert.BeginWith();
 - c) Assert.Starts();
 - d) Assert.Begin().
- 86) Как проверить, что строка заканчивается на определенный суффикс в xUnit?
- a) Assert.EndsWith();
 - b) Assert.End();
 - c) Assert.FinishWith();
 - d) Assert.EndString().
- 87) Какой метод используется для проверки того, что строка пуста?
- a) Assert.Empty();
 - b) Assert.Null();
 - c) Assert.NotEmpty();
 - d) Assert.NotNull().
- 88) Как проверить, что строка не пуста в xUnit?
- a) Assert.NotEmpty();
 - b) Assert.Null();
 - c) Assert.Empty();
 - d) Assert.NotNull().

- 89) Какой метод используется для проверки того, что строка состоит только из цифр?
- a) `Assert.IsNumeric();`
 - b) `Assert.IsDigit();`
 - c) `Assert.AllDigits();`
 - d) `Assert.DigitOnly();`
- 90) Как проверить, что строка соответствует регулярному выражению в xUnit?
- a) `Assert.Match();`
 - b) `Assert.RegexMatch();`
 - c) `Assert.Matches();`
 - d) `Assert.RegularMatch();`
- 91) Какой метод используется для проверки равенства двух коллекций в xUnit?
- a) `Assert.Equal();`
 - b) `Assert.AreEqual();`
 - c) `Assert.Equals();`
 - d) `Assert.Same();`
- 92) Как проверить, что коллекция содержит определенный элемент в xUnit?
- a) `Assert.Contains();`
 - b) `Assert.HasElement();`
 - c) `Assert.Include();`
 - d) `Assert.Member();`
- 93) Какой метод используется для проверки количества элементов в коллекции в xUnit?
- a) `Assert.Length();`
 - b) `Assert.Count();`
 - c) `Assert.Size();`
 - d) `Assert.NumberOfElements();`
- 94) Как проверить, что две коллекции не равны в xUnit?
- a) `Assert.NotEqual();`
 - b) `Assert.AreNotEqual();`
 - c) `Assert.NotEquals();`
 - d) `Assert.Different();`
- 95) Какой метод используется для проверки того, что все элементы коллекции удовлетворяют определенному условию в xUnit?
- a) `Assert.All();`
 - b) `Assert.Every();`
 - c) `Assert.Each();`
 - d) `Assert.AllElements();`
- 96) Как проверить, что коллекция пуста в xUnit?
- a) `Assert.Empty();`
 - b) `Assert.Null();`
 - c) `Assert.NotEmpty();`
 - d) `Assert.NotNull();`
- 97) Какой метод используется для проверки того, что коллекция не пуста в xUnit?
- a) `Assert.NotEmpty();`
 - b) `Assert.Null();`
 - c) `Assert.Empty();`
 - d) `Assert.NotNull();`
- 98) Как проверить, что коллекция отсортирована в xUnit?
- a) `Assert.Sorted();`
 - b) `Assert.Order();`
 - c) `Assert.IsSorted();`
 - d) `Assert.Ordered();`
- 99) Какой метод используется для проверки того, что коллекция не содержит дубликатов в xUnit?
- a) `Assert.Unique();`
 - b) `Assert.NoDuplicates();`
 - c) `Assert.Distinct();`
 - d) `Assert.UniqueElements();`
- 100) Как проверить, что коллекция является подмножеством другой коллекции в xUnit?
- a) `Assert.Subset();`
 - b) `Assert.IsSubset();`
 - c) `Assert.ContainsAll();`
 - d) `Assert.SubsetOf();`
- 101) Что такое mock-объекты?
- a) Реальные объекты, используемые в тестах;
 - b) Заглушки для тестирования взаимодействия;
 - c) Методы для генерации случайных данных;
 - d) Классы для создания новых проектов.
- 102) Какая библиотека часто используется для создания mock-объектов в .NET?
- a) `Moq;`
 - b) `NUnit;`
 - c) `MSTest;`

- d) xUnit.
- 103) Как создать mock-объект в xUnit?
- Использовать метод new Mock();
 - Использовать метод CreateMock();
 - Использовать атрибут Mock;
 - Использовать интерфейс IMock.
- 104) Какой метод используется для настройки поведения mock-объекта в xUnit?
- Setup();
 - Configure();
 - SetUp();
 - MockBehavior().
- 105) Как проверить, что метод был вызван на mock-объекте в xUnit?
- Assert.Called();
 - Verify.Method();
 - Assert.Invoke();
 - Verify.WasCalled().
- 106) Какой метод используется для проверки количества вызовов метода на mock-объекте в xUnit?
- Assert.Count();
 - Verify.Times();
 - Assert.NumberOfCalls();
 - Verify.CallCount().
- 107) Как задать возвращаемое значение метода на mock-объекте в xUnit?
- SetupReturn();
 - Return();
 - SetReturnValue();
 - MockReturnValue().
- 108) Какой метод используется для проверки того, что метод был вызван с определенными аргументами в xUnit?
- Assert.CalledWith();
 - Verify.WithArguments();
 - Assert.InvokeWith();
 - Verify.WasCalledWith().
- 109) Как игнорировать вызовы метода на mock-объекте в xUnit?
- Ignore();
 - SetupIgnore();
 - Verify.Ignore();
 - Mock.IgnoreCall().
- 110) Какой метод используется для очистки настроек mock-объекта в xUnit?
- Reset();
 - Clear();
 - Dispose();
 - Mock.ClearSetup().

5.3. Оценочные материалы, используемые для экзамена (описание билетов, тестов и т.п.)

Экзаменационный билет включает в себя два фундаментальных теоретических вопроса из установленного перечня контрольных вопросов, используемых при формировании экзаменационных билетов при оценке знаний обучающихся на экзамене, а также два практических задания из установленного перечня контрольных заданий, используемых при формировании экзаменационных билетов при оценке знаний обучающихся на экзамене

Пример экзаменационного билета:

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ СТАРООСКОЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ ИНСТИТУТ им. А.А. Угарова (филиал) — федерального государственного автономного образовательного учреждения высшего образования «Национальный исследовательский технологический университет «МИСИС» Кафедра «Автоматизированные и информационные системы управления им Ю.И. Еременко» 09.03.02 Информационные системы и технологии Профиль — Информационные системы и технологии Дисциплина «Программирование на языках высокого уровня» ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 1 Теоретические вопросы: → Как реализовать паттерн Factory Method в C#? → Как обрабатывать события в паттерне MVVM? Практические задачи: → Реализуйте паттерн «Одиночка» для управления доступом к единственному экземпляру базы данных в приложении. Убедитесь, что только один экземпляр может быть создан и предоставлен всем клиентам. → Создайте классы Author и Book с соответствующими полями. Создайте объект Book с вложенным объектом Author и сериализуйте его в JSON. Десериализуйте JSON обратно в объект Book с корректным восстановлением вложенного объекта Author. Экзаменатор: _____ П.И. Жуков Утверждено на заседании кафедры АИСУ им. Ю.И. Еременко Протоколом № _____ от « _____ » _____ 20__ г. И.о. заведующего кафедрой АИСУ им. Ю.И. Еременко: _____ Д.А. Полещенко	
---	--

Билеты в бумажном виде хранятся на кафедре АИСУ им. Ю.И. Еременко и утверждены ее заведующим (или заместителем зав. Кафедрой).

5.4. Методика оценки освоения дисциплины

№ п/п	Вид оценочного средства	Критерий	Оценка
1.	Выполнение и защита курсовой работы	Обучающийся самостоятельно выполняет полное и аргументированное решение индивидуальных заданий, не допустив ошибок. При защите заданий отвечает развернуто и исчерпывающе на все вопросы.	«Отлично»
		Обучающийся практически самостоятельно выполняет полное решение заданий, но не может аргументировать свое решение. При защите заданий допускает незначительные неточности.	«Хорошо»
		Обучающийся в целом правильно решает задание, но не может аргументировать свое решение. При защите заданий допускает значительные неточности. Обучающийся правильно понимает способ решения заданий, но допускает ошибки при их решении. Задание выполнено частично. При защите заданий допускает значительные неточности.	«Удовлетворительно»
		Обучающийся не может решить задание.	«Неудовлетворительно»
2.	Устный опрос по материалам лекционных занятий	Обучающийся уверенно, логически связно, динамично, грамотно отвечает на поставленные вопросы. Демонстрирует абсолютное понимание темы обсуждаемой предметной области, достаточно полные знания в объеме прослушанного теоретического материала. Грамотно владеет и адекватно использует терминологию предметной области.	«Зачтено»
		Обучающийся не верно интерпретирует поставленные вопросы, не владеет терминологией предметной области, не понимает сущности обсуждаемой проблемы. Невнятно, невразумительно, неуверенно, неверно формулирует ответы на дополнительные и наводящие вопросы и/или не отвечает на них.	«Не зачтено»
3.	Выполнение и защита лабораторных работ	Обучающийся показывает твердые и достаточно полные знания в объеме прослушанного теоретического материала, владеет требуемым математическим аппаратом, методиками решения, необходимыми остаточными знаниями по изученным фундаментальным дисциплинам (математика, физика); демонстрирует умения и практические навыки владения информационными технологиями, позволяющими оптимизировать экспериментальную и аналитическую часть лабораторного исследования. Логически связно, динамично, грамотно и последовательно излагает методику выполнения лабораторной работы и обработки результатов моделирования. Ошибаясь, уверенно исправляется после дополнительных и наводящих вопросов.	«Зачтено»
		Обучающийся допускает грубые ошибки в ответе, не понимает сущности излагаемого вопроса, не умеет применять теоретические знания на практике и/или не владеет требуемыми знаниями. Невнятно, невразумительно, неуверенно, неверно формулирует ответы на дополнительные и наводящие вопросы и/или не отвечает на них.	«Не зачтено»
4.	Экзамен	Компетенции УК-2, ПК-1 ОПК-6, ОПК-8 сформированы. Обучающийся демонстрирует: - глубокие знания содержания изученной дисциплины во взаимосвязи с другими дисциплинами; - способность использовать теоретические знания при выполнении практических заданий; - аргументированные, исчерпывающие ответы на все вопросы по билету, а также дополнительные вопросы экзаменатора; - умение выполнять и обосновывать решение практических заданий высокого уровня сложности; - наличие собственной обоснованной позиции по обсуждаемым вопросам; - свободное использование в ответах на вопросы материалов рекомендованной основной и дополнительной литературы.	«Отлично»

		Компетенции УК-2, ПК-1 ОПК-6, ОПК-8 сформированы. Обучающийся демонстрирует: - знание основных терминов по содержанию изученной дисциплины; - твердые знания теоретического материала; - умение дать четкие ответы на поставленные вопросы; - умение решать практические задания; - владение основной литературой, рекомендованной программой дисциплины. Допускаются незначительные неточности в ответах на теоретические вопросы и при выполнении практических заданий.	«Хорошо»
		Компетенции УК-2, ПК-1 ОПК-6, ОПК-8 сформированы. Обучающийся демонстрирует: - знания теоретического материала по изученной дисциплине; - неполные ответы на основные вопросы, допуская ошибки в ответе; недостаточное понимание сущности излагаемых вопросов; - неточные ответы на дополнительные вопросы; - умение выполнять практические задания без грубых ошибок; - недостаточное владение литературой, рекомендованной программой дисциплины.	«Удовлетворительно»
		Компетенции УК-2, ПК-1 ОПК-6, ОПК-8 не сформированы. Обучающийся демонстрирует: - существенные пробелы в знаниях учебного материала; - принципиальные ошибки при ответе на основные вопросы билета, отсутствие знаний и понимания основных терминов и определений; - непонимание сущности дополнительных вопросов в рамках заданий билета; - отсутствие навыка или существенные ошибки при выполнении практических заданий; - незнание литературы, рекомендованной программой дисциплины.	«Неудовлетворительно»
5	Оценка тестовых заданий	Обучающийся дал 90% и более правильных ответов	«Отлично»
		Обучающийся дал 65-89% правильных ответов	«Хорошо»
		Обучающийся дал 50-64% правильных ответов	«Удовлетворительно»
		Обучающийся дал менее 50% правильных ответов	«Неудовлетворительно»

6. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ				
6.1. Рекомендуемая литература				
6.1.1. Основная литература				
Обозначение	Авторы, составители	Заглавие	Библиотека	Издательство, год
Л 1.1	Чернышев, С.А.	Принципы, паттерны и методологии разработки программного обеспечения	ЭБС «Юрайт. Образовательная платформа» URL: https://urait.ru/bcode/544319	М: Юрайт, 2024
Л 1.2	Казанский А.А.	Программирование на C#	ЭБС «Юрайт. Образовательная платформа» URL: https://urait.ru/bcode/569864	М: Юрайт, 2025
6.1.2. Дополнительная литература				
Обозначение	Авторы, составители	Заглавие	Библиотека	Издательство, год
Л 2.1	Паронджанов В. Д.	Алгоритмические языки и программирование: ДРАКОН	ЭБС «Юрайт. Образовательная платформа» URL: https://urait.ru/bcode/567153	М: Юрайт, 2025
Л 2.2	Зыков С. В.	Объектно-ориентированное	Юрайт. Образовательная платформа URL: https://urait.ru/bcode/561434	М: Юрайт, 2025

		программирование		
Л 2.3	Галиаскаров Э. Г., Воробьев А. С.	Анализ и проектирование систем с использованием UML	ЭБС «Юрайт. Образовательная платформа» URL: https://urait.ru/bcode/568178	М: Юрайт, 2025
6.1.3. Методические разработки				
Обозначение	Авторы, составители	Заглавие	Библиотека	Издательство, год
Л 3.1	Жуков П.И.	Методические указания по выполнению практических работ	https://disk.yandex.ru/d/Pji31sUJCpdfjQ	
Л 3.2	Жуков П.И.	Методическое указание по выполнению курсовой работы	https://disk.yandex.ru/d/Pji31sUJCpdfjQ	
Л 3.3	Жуков П.И..	Программирование на языках высокого уровня. Учебное пособие для выполнения лабораторных работ	https://disk.yandex.ru/d/Pji31sUJCpdfjQ	

6.2. Перечень ресурсов информационно-телекоммуникационной сети «Интернет»	
Э 1	Открытое образование [Электронный ресурс]: https://openedu.ru/
Э 2	LMS Moodle [Электронный ресурс]: https://newlms.misis.ru
Э 3	ЭБС «Университетская библиотека онлайн» [Электронный ресурс]: https://biblioclub.ru
Э 4	Интерактивный учебник «Metanit. Язык C#» [Электронный ресурс]: https://metanit.com/sharp/
6.3. Перечень программного обеспечения	
П 1	Microsoft Windows
П 2	Microsoft Office
П 3	Microsoft Visual Studio 2022
6.4. Перечень информационных справочных систем и профессиональных баз данных	
И 1	eLIBRARY.RU - НАУЧНАЯ ЭЛЕКТРОННАЯ БИБЛИОТЕКА [Электронный ресурс]: https://elibrary.ru/

7. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ	
7.1	Аудитория №404 «Объектно-ориентированного программирования» Перечень основного оборудования, учебно-наглядных пособий: • 25 посадочных мест • 13 мониторов • 13 персональных компьютеров • Проектор Epson EH-TW750 • Экран настенный • Маркерная доска В помещении имеется подключение к сети «Интернет» и доступ в электронную информационно-образовательную среду организации.
7.2	Аудитория №405 «Прикладного программирования» Перечень основного оборудования, учебно-наглядных пособий: • 31 посадочное место • 13 мониторов • 13 персональных компьютеров • Проектор Epson EB-485W • Экран настенный • Доска настенная В помещении имеется подключение к сети «Интернет» и доступ в электронную информационно-образовательную среду организации.

8. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ИЗУЧЕНИЮ ДИСЦИПЛИНЫ
Материал в рамках данной дисциплины предполагает, что обучающийся уверенно владеет навыками структурного

программирования и объектно-ориентированного программирования и готов развивать эти навыки далее, осваивая конкретные паттерны и технологии.

Для успешного освоения курса крайне рекомендуется посещение всех практических и лабораторных работ и активная работа на них. В рамках курса предполагается, что упомянутые активности содержат параллельные аспекты искусства программирования, которые необходимо будет применить при написании курсовой работы.

Посещение лекционных занятий крайне желательно. Для обучающихся, плохо воспринимающих информацию на слух, рекомендуется к индивидуальному прочтению в свободное время (вместо лекций):

- 1) Принципы, паттерны и методологии разработки программного обеспечения за авторством Чернышева С.А.;
- 2) Чистая архитектура. Искусство разработки ПО за авторством Роберта Мартина;
- 3) Идеальный программист. Как стать профессионалом разработки ПО за авторством Роберта Мартина.

Рекомендуется держать «под рукой» интерактивный учебник *Metanit*:

- 1) Интерактивный учебник по C# (<https://metanit.com/sharp/>);
- 2) Интерактивный учебник по паттернам *GoF* (<https://metanit.com/sharp/patterns>)

Критерии успешного завершения дисциплины:

- Сдача экзамена на оценку «удовлетворительно» и выше;

Критерии допуска к экзамену:

- Успешная защита всех лабораторных работ;
- Успешная защита всех практических работ;
- Успешная защита курсовой работы.

Критерий успешной защиты лабораторных и практических работ:

- Полное выполнение задания в рамках работы;
- Устный ответ на 70% вопросов, представленных к конкретному виду работы.

Требования к оформлению отчетов по лабораторным и практическим работам:

- Наличие титульного листа с фамилией обучающегося и преподавателя дисциплины;
- Наличие листа с постановкой задачи;
- Наличие листа с результатами выполнения задания;
- Times New Roman, как основной шрифт, пт.14 или пт.12.