

Министерство науки и высшего образования Российской Федерации
Муромский институт (филиал)
федерального государственного бюджетного образовательного учреждения высшего образования
**«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»**
(МИ ВлГУ)

Кафедра *ИС*

«УТВЕРЖДАЮ»
Заместитель директора по УР
Д.Е. Андрианов
_____ 19.05.2026

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Объектно-ориентированное программирование

Направление подготовки

09.03.02 Информационные системы и технологии

Профиль подготовки

Информационные системы и технологии

Семестр	Трудоемкость, час./зач. ед.	Лекции, час.	Практические занятия, час.	Лабораторные работы, час.	Консультация, час.	Контроль, час.	Всего (контактная работа), час.	СРС, час.	Форма промежуточного контроля (экз., зач., зач. с оц.)
3	234 / 6,5	48	16	32	6,8	0,35	103,15	95,2	Экз.(35,65)
4	54 / 1,5		30			2,25	32,25	21,75	Зач.
Итого	288 / 8	48	46	32	6,8	2,6	135,4	116,95	35,65

Муром, 2026 г.

1. Цель освоения дисциплины

Цель дисциплины: ознакомление студентов с методикой объектно-ориентированного программирования; формирование способностей осуществлять сертификацию проектов, проектировать базовые и прикладные информационные технологии и разрабатывать средства реализации информационных технологий.

Задачи дисциплины:

- Изучение основных концепций объектно-ориентированного программирования;
- Формирование представлений о стандартах качества программных продуктов;
- Приобретение навыков выполнять проектирование и разработку информационных технологий.

2. Место дисциплины в структуре ОПОП ВО

Дисциплина «Объектно-ориентированное программирование» — это методика программирования, которая обеспечивает понимание основ и концепций объектно-ориентированного программирования, а также подходов при разработке объектно-ориентированных программ. Курс базируется на знаниях, полученных студентами в процессе изучения дисциплин: информатика, технологии программирования. Углубление и расширение вопросов, изложенных в данном курсе, будет осуществляться во время работы студентов над дисциплинами: основы теории алгоритмов, управление данными, информационные технологии и многих других, а также при написании бакалаврских работ.

3. Планируемые результаты обучения по дисциплине

Планируемые результаты обучения по дисциплине, соотнесенные с планируемыми результатами освоения ОПОП (компетенциями и индикаторами достижения компетенций)

Формируемые компетенции (код, содержание компетенции)	Планируемые результаты обучения по дисциплине, в соответствии с индикатором достижения компетенции		Наименование оценочного средства
	Индикатор достижения компетенции	Результаты обучения по дисциплине	
ОПК-6 Способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий;	ОПК-6.1 Демонстрирует знания алгоритмизации, языков и технологий программирования, пригодных для практического применения в области информационных систем и технологий	Знать методы алгоритмизации, языки и технологии программирования, пригодные для практического применения в области информационных систем и технологий (ОПК-6.1)	вопросы к устному опросу, тесты
	ОПК-6.2 Применяет методы алгоритмизации, языки и технологии программирования при решении профессиональных задач в области информационных систем и технологий	Уметь применять методы алгоритмизации, языки и технологии программирования при решении профессиональных задач в области информационных систем и технологий (ОПК-6.2)	
	ОПК-6.3 Программирует, отлаживает и тестирует прототипы программно-технических комплексов	Иметь навыки программирования, отладки и тестирования прототипов программно-технических комплексов задач (ОПК-6.3)	

4. Структура и содержание дисциплины

Общая трудоемкость дисциплины составляет 8 зачетных единиц, 288 часов.

4.1. Форма обучения: очная

Уровень базового образования: среднее общее.

Срок обучения 4г.

4.1.1. Структура дисциплины

№ п\п	Раздел (тема) дисциплины	Семестр	Контактная работа обучающихся с педагогическим работником							Самостоятельная работа	Форма текущего контроля успеваемости (по неделям семестра), форма промежуточной аттестации(по семестрам)
			Лекции	Практические занятия	Лабораторные работы	Контрольные работы	КП / КР	Консультация	Контроль		
1	Методы объектно-ориентированного программирования	3	48	16	32					95,2	устный опрос, тестирование
Всего за семестр		234	48	16	32			6,8	0,35	95,2	Экз.(35,65)
2	Объектно-ориентированное проектирование	4		30						21,75	устный опрос, тестирование
Всего за семестр		54		30			+	0	2,25	21,75	Зач.
Итого		288	48	46	32			6,8	2,6	116,95	35,65

4.1.2. Содержание дисциплины

4.1.2.1. Перечень лекций

Семестр 3

Раздел 1. Методы объектно-ориентированного программирования

Лекция 1.

Новейшие направления в области создания технологий программирования. Законы эволюции программного обеспечения (2 часа).

Лекция 2.

Программирование в средах современных информационных систем: создание модульных программ, элементы теории модульного программирования объектно-ориентированные проектирование и программирование (2 часа).

Лекция 3.

Среда Net. Framework. Общеязыковая исполняющая среда (CLR). Стандарты C++. Технологии отладки (2 часа).

Лекция 4.

Объектно-ориентированный подход к проектированию и разработке программ. Сущность объектно-ориентированного подхода (2 часа).

Лекция 5.

Инкапсуляция. Понятие класса. Управление доступом к элементам класса. Конструкторы и деструкторы. Переменные объектного типа (2 часа).

Лекция 6.

Друзья класса (2 часа).

Лекция 7.

Наследование. Базовый класс. Управление доступом при наследовании (2 часа).

Лекция 8.

Простое наследование. Виртуальные функции (2 часа).

Лекция 9.

Чистые виртуальные функции. Абстрактные классы. Непрямые базовые классы. Виртуальные деструкторы. Виртуальные базовые классы (2 часа).

Лекция 10.

Полиморфизм (2 часа).

Лекция 11.

Методы общего полиморфизма: перегрузка операций. Реализация перегруженной операции. Перегрузка операции присваивания, инкремента, декремента, бинарных арифметических операций (2 часа).

Лекция 12.

Методы общего полиморфизма: перегрузка операций. Перегрузка операций ввода-вывода в потоки, индексации массивов, операций выделения и освобождения памяти (2 часа).

Лекция 13.

Методы общего полиморфизма: Преобразования типов, определяемые классом (2 часа).

Лекция 14.

Методы общего полиморфизма: перегрузка функций. Чистый полиморфизм (2 часа).

Лекция 15.

Параметрический полиморфизм. Шаблоны функций (2 часа).

Лекция 16.

Параметрический полиморфизм. Шаблоны классов (2 часа).

Лекция 17.

Классы в C++. Управление доступом. Конструкторы. Деструкторы. Наследование (2 часа).

Лекция 18.

Свойства классов в C++. Скалярные и индексированные свойства (2 часа).

Лекция 19.

Перегрузка операций в C++. Преобразование типов в C++ (2 часа).

Лекция 20.

Делегаты. События. Интерфейсы (2 часа).

Лекция 21.

Лямбда-выражения. Основы LINQ (2 часа).

Лекция 22.

Обработка исключительных ситуаций в C++ (2 часа).

Лекция 23.

Обобщения в C++ (2 часа).

Лекция 24.

Основные структуры данных: очередь, стек, двусторонняя очередь, связный список (2 часа).

4.1.2.2. Перечень практических занятий

Семестр 3

Раздел 1. Методы объектно-ориентированного программирования

Практическое занятие 1

Виджеты и компоновки (2 часа).

Практическое занятие 2

Работа с виджетами (2 часа).

Практическое занятие 3

Обработка событий в Qt (2 часа).

Практическое занятие 4

Работа с файлом ресурсов в Qt (2 часа).

Практическое занятие 5

Сигналы и слоты (2 часа).

Практическое занятие 6

Работа с файлами в Qt (2 часа).

Практическое занятие 7

MessageBox в Qt (2 часа).

Практическое занятие 8

Расширенные функции калькулятора (2 часа).

Семестр 4

Раздел 2. Объектно-ориентированное проектирование

Практическое занятие 9

Коллективная разработка приложения с использованием GitHub (2 часа).

Практическое занятие 10

Создание DLL-библиотек в C# (2 часа).

Практическое занятие 11

События в C#. Обработка событий (2 часа).

Практическое занятие 12

Работа с каталогами и файлами в C# (2 часа).

Практическое занятие 13

Запросы LINQ (2 часа).

Практическое занятие 14

Регулярные выражения в C# (2 часа).

Практическое занятие 15

Лямбда-выражения в C# (2 часа).

Практическое занятие 16

Объектная модель Word. Работа с документами Word (2 часа).

Практическое занятие 17

Создание документов Excel средствами языка C# (2 часа).

Практическое занятие 18

Создание коммерческого приложения (2 часа).

Практическое занятие 19

Разработка многопоточного приложения в C# (2 часа).

Практическое занятие 20

Создание приложений WPF. Стили и шаблоны в WPF (2 часа).

Практическое занятие 21

Обработка исключительных ситуаций (2 часа).

Практическое занятие 22

Использование визуальных и не визуальных компонентов при разработке приложений (2 часа).

Практическое занятие 23

Создание собственных компонент в C# (2 часа).

4.1.2.3. Перечень лабораторных работ

Семестр 3

Раздел 1. Методы объектно-ориентированного программирования

Лабораторная 1.

Классы и объекты в C++. Инкапсуляция (4 часа).

Лабораторная 2.

Классы и объекты в C++. Наследование (4 часа).

Лабораторная 3.

Классы и объекты в C++. Полиморфизм (4 часа).

Лабораторная 4.

Классы и объекты в C++. Виртуальные методы (4 часа).

Лабораторная 5.

Абстрактные классы и полиморфизм в C++ (4 часа).

Лабораторная 6.

Композиция классов и объектов в C++ (4 часа).

Лабораторная 7.

Динамические объекты в C++ (4 часа).

Лабораторная 8.

Использование объектов в качестве входных и выходных аргументов функций (4 часа).

4.1.2.4. Перечень тем и учебно-методическое обеспечение самостоятельной работы

Перечень тем, вынесенных на самостоятельное изучение:

1. Преимущества перед процедурным программированием.
2. Принципы разработки иерархии классов и ее использования.
3. Разрешение вопросов, связанных с неоднозначностью доступа.
4. Иерархия классов, используемая в среде Microsoft Visual Studio.
5. Программирование графики на C++.
6. Объектно-ориентированное программирование как основа современного программирования.
7. Совместное использование функций и перегрузка операторов.
8. Использование динамической памяти.
9. Механизм исключительных ситуаций.
10. Поддержка модульности. Разбиение программы на единицы компиляции.
11. Шаблонные классы и стандартные контейнеры.
12. Организация пользовательского интерфейса.

Для самостоятельной работы используются методические указания по освоению дисциплины и издания из списка приведенной ниже основной и дополнительной литературы.

4.1.2.5. Перечень тем контрольных работ, рефератов, ТР, РГР, РПР

Не планируется.

4.1.2.6. Примерный перечень тем курсовых работ (проектов)

1. Разработка программы, реализующей построение и движение простых фигур в C++/C#.
2. Разработка графического редактора в C++/C#.
3. Разработка тестирующей программы по дисциплине «Объектно-ориентированное программирование» в C++/C#.
4. Разработка текстового редактора в C++/C#.
5. Разработка иерархии классов по заданной тематике.
6. Разработка объектно-ориентированной программы разбора математических выражений.
7. Разработка программы, определяющей аппаратные характеристики ПК.
8. Разработка игры «Сапер».
9. Разработка игры «Крестики-нолики».
10. Разработка игры «Пазлы».

11. Разработка программы "Файловый менеджер".
12. Разработка программы воспроизведения аудио и видео файлов.
13. Разработка программы обработки изображений.

4.2 Форма обучения: заочная

Уровень базового образования: среднее профессиональное.

Срок обучения 3г бм.

Семестр	Трудоёмкость, час./ зач. ед.	Лекции, час.	Практические занятия, час.	Лабораторные работы, час.	Консультация, час.	Контроль, час.	Всего (контактная работа), час.	СРС, час.	Форма промежуточного контроля (экз., зач., зач. с оц.)
3	144 / 4	10	2	8	5	0,6	25,6	109,75	Экз.(8,65)
4	144 / 4		10			2,25	12,25	128	Зач.(3,75)
Итого	288 / 8	10	12	8	5	2,85	37,85	237,75	12,4

4.2.1. Структура дисциплины

№ п/п	Раздел (тема) дисциплины	Семестр	Контактная работа обучающихся с педагогическим работником							Самостоятельная работа	Форма текущего контроля успеваемости (по неделям семестра), форма промежуточной аттестации(по семестрам)
			Лекции	Практические занятия	Лабораторные работы	Контрольные работы	КП / КР	Консультация	Контроль		
1	Методы объектно-ориентированного программирования	3	10	2	8					109,75	контрольная работа, тестирование
Всего за семестр		144	10	2	8	+		5	0,6	109,75	Экз.(8,65)
2	Объектно-ориентированное проектирование	4		10						128	контрольная работа, тестирование
Всего за семестр		144		10			+	0	2,25	128	Зач.(3,75)
Итого		288	10	12	8			5	2,85	237,75	12,4

4.2.2. Содержание дисциплины

4.2.2.1. Перечень лекций

Семестр 3

Раздел 1. Методы объектно-ориентированного программирования

Лекция 1.

Классы и объекты в C++. Инкапсуляция (2 часа).

Лекция 2.

Наследование в C++. Простое наследование. Управление доступом (2 часа).

Лекция 3.

Наследование. Множественное наследование. Полиморфизм. Виды полиморфизма (2 часа).

Лекция 4.

Полиморфизм. Перегрузка операций. Преобразование типов (2 часа).

Лекция 5.

Полиморфизм. Виртуальные функции. Чистые виртуальные функции. Абстрактные классы (2 часа).

4.2.2.2. Перечень практических занятий

Семестр 3

Раздел 1. Методы объектно-ориентированного программирования

Практическое занятие 1.

Создание приложения с графическим интерфейсом в Qt Creator (2 часа).

Семестр 4

Раздел 2. Объектно-ориентированное проектирование

Практическое занятие 2.

Виджеты и компоновки (2 часа).

Практическое занятие 3.

Работа с виджетами (2 часа).

Практическое занятие 4.

Обработка событий в Qt (2 часа).

Практическое занятие 5.

Работа с файлом ресурсов в Qt (2 часа).

Практическое занятие 6.

Сигналы и слоты (2 часа).

4.2.2.3. Перечень лабораторных работ

Семестр 3

Раздел 1. Методы объектно-ориентированного программирования

Лабораторная 1.

Классы и объекты в C++ (4 часа).

Лабораторная 2.

Наследование (4 часа).

4.2.2.4. Перечень тем и учебно-методическое обеспечение самостоятельной работы

Перечень тем, вынесенных на самостоятельное изучение:

1. Новейшие направления в области создания технологий программирования. Законы эволюции программного обеспечения.

2. Программирование в средах современных информационных систем: создание модульных программ, элементы теории модульного программирования объектно-ориентированные проектирование и программирование.

3. Среда Net. Framework. Общеязыковая исполняющая среда (CLR). Стандарты C++. Технологии отладки.

4. Объектно-ориентированный подход к проектированию и разработке программ. Сущность объектно-ориентированного подхода.

5. Инкапсуляция. Понятие класса. Управление доступом к элементам класса. Конструкторы и деструкторы. Переменные объектного типа.

6. Друзья класса.

7. Наследование. Базовый класс. Управление доступом при наследовании.

8. Простое наследование. Виртуальные функции.

9. Чистые виртуальные функции. Абстрактные классы. Непрямые базовые классы. Виртуальные деструкторы. Виртуальные базовые классы.

10. Полиморфизм.
 11. Методы общего полиморфизма: перегрузка операций. Реализация перегруженной операции. Перегрузка операции присваивания, инкремента, декремента, бинарных арифметических операций.
 12. Методы общего полиморфизма: перегрузка операций. Перегрузка операций ввода-вывода в потоки, индексации массивов, операций выделения и освобождения памяти.
 13. Методы общего полиморфизма: Преобразования типов, определяемые классом.
 14. Методы общего полиморфизма: перегрузка функций. Чистый полиморфизм.
 15. Параметрический полиморфизм. Шаблоны функций.
 16. Параметрический полиморфизм. Шаблоны классов.
 17. Классы в C++. Управление доступом. Конструкторы. Деструкторы. Наследование.
 18. Свойства классов в C++. Скалярные и индексированные свойства.
 19. Перегрузка операций в C++. Преобразование типов в C++.
 20. Делегаты. события. Интерфейсы.
 21. Концепция объектно-ориентированного проектирования. Проектирование структурной схемы программы. Составление начальной иерархии и структуры классов. Реорганизация иерархии и структуры классов. Организационная структура программы. Проектирование файлов интерфейсов классов. Файл интерфейса базового класса. Файл интерфейса производных классов. Файл управления.
 22. Концепции программирования для Windows. Структура Windows программ. Использование Windows Forms для создания приложений с графическим интерфейсом пользователя.
 23. Создание элемента управления Windows Forms.
 24. Разработка справочной системы для Windows-приложений.
 25. Преимущества перед процедурным программированием.
 26. Принципы разработки иерархии классов и ее использования.
 27. Разрешение вопросов, связанных с неоднозначностью доступа.
 28. Иерархия классов, используемая в среде QT.
 29. Программирование на C++/CLI.
 30. Объектно-ориентированное программирование как основа современного программирования.
 31. Совместное использование функций и перегрузка операторов.
 32. Использование динамической памяти.
 33. Механизм исключительных ситуаций.
 34. Поддержка модульности. Разбиение программы на единицы компиляции.
 35. Шаблоны классов и стандартные контейнеры.
 36. Организация пользовательского интерфейса.
- Для самостоятельной работы используются методические указания по освоению дисциплины и издания из списка приведенной ниже основной и дополнительной литературы.

4.2.2.5. Перечень тем контрольных работ, рефератов, ТР, РГР, РПР

1. Новейшие направления в области создания технологий программирования. Законы эволюции программного обеспечения.
2. Программирование в средах современных информационных систем: создание модульных программ, элементы теории модульного программирования объектно-ориентированное проектирование и программирование.
3. Среда Net. Framework. Общеязыковая исполняющая среда (CLR). Стандарты C++. Технологии отладки.
4. Объектно-ориентированный подход к проектированию и разработке про-грамм. Сущность объектно-ориентированного подхода .
5. Методы общего полиморфизма: Преобразования типов, определяемые классом.
6. Методы общего полиморфизма: перегрузка функций. Чистый полиморфизм.
7. Параметрический полиморфизм. Шаблоны функций.
8. Параметрический полиморфизм. Шаблоны классов .

9. Классы в C++. Управление доступом. Конструкторы. Деструкторы. Наследование.
10. Свойства классов в C++. Скалярные и индексированные свойства.
11. Перегрузка операций в C++. Преобразование типов в C++.
12. Делегаты. события. Интерфейсы.
13. Концепция объектно-ориентированного проектирования. Проектирование структурной схемы программы. Составление начальной иерархии и структуры классов. Реорганизация иерархии и структуры классов. Организационная структура программы. Проектирование файлов интерфейсов классов. Файл интерфейса базового класса. Файл интерфейса производных классов. Файл управления.
14. Концепции программирования для Windows. Структура Windows программ. Использование Windows Forms для создания приложений с графическим интерфейсом пользователя.
15. Создание элемента управления Windows Forms .
16. Разработка справочной системы для Windows-приложений.

4.2.2.6. Примерный перечень тем курсовых работ (проектов)

1. Разработка программы, реализующей построение и движение простых фигур в C++.
2. Разработка графического редактора в C++.
3. Разработка тестирующей программы по дисциплине «Объектно-ориентированное программирование» в C++.
4. Разработка текстового редактора в C++.
5. Разработка иерархии классов по заданной тематике.
6. Разработка объектно-ориентированной программы разбора математических выражений .
7. Разработка программы, определяющей аппаратные характеристики ПК.
8. Разработка игры «Сапер».
9. Разработка игры «Крестики-нолики».
10. Разработка игры «Пазлы».
11. Разработка программы "Файловый менеджер".
12. Разработка программы воспроизведения аудио и видео файлов.
13. Разработка программы обработки изображений.

5. Образовательные технологии

В процессе изучения дисциплины применяется контактная технология преподавания (за исключением самостоятельно изучаемых студентами вопросов).. При проведении практических и лабораторных работ применяется имитационный или симуляционный подход, когда преподавателем разбирается на конкретном примере проблемная ситуация, все шаги решения задачи студентам демонстрируются при помощи мультимедийной техники. Затем студенты самостоятельно решают аналогичные задания.

Во время выполнения лабораторных и практических работ каждому студенту выдается конкретное задание, тем самым формируется способность обучающихся к самостоятельной работе при решении определенных задач, связанных с изучением принципов объектно-ориентированного программирования

6. Оценочные средства для текущего контроля успеваемости, промежуточной аттестации по итогам освоения дисциплины.

Фонды оценочных материалов (средств) приведены в приложении.

7. Учебно-методическое и информационное обеспечение дисциплины.

7.1. Основная учебно-методическая литература по дисциплине

1. Объектно-ориентированное программирование на C++ : учебник / И.В. Баранова [и др.].. — Красноярск : Сибирский федеральный университет, 2019. — 288 с. — ISBN 978-5-

7638-4034-6. — Текст : электронный // IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/100067.html> (дата обращения: 11.12.2022). — Режим доступа: для авторизир. пользователей - <https://www.iprbookshop.ru/100067.html>

2. Объектно-ориентированное программирование. В 3-х частях. Ч.1 : учебное пособие / П.П. Степанов [и др.]. — Омск : Омский государственный технический университет, 2021. — 112 с. — ISBN 978-5-8149-3301-0 (ч.1), 978-5-8149-3300-3. — Текст : электронный // IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/124850.html> (дата обращения: 13.12.2022). — Режим доступа: для авторизир. пользователей - <https://www.iprbookshop.ru/124850.html>

7.2. Дополнительная учебно-методическая литература по дисциплине

1. Достовалов, Д. Н. Объектно-ориентированный анализ и проектирование. Задачи и примеры на C++ : учебное пособие / Д. Н. Достовалов, О. В. Лауферман. — Новосибирск : Новосибирский государственный технический университет, 2022. — 74 с. — ISBN 978-5-7782-4708-6. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/126577.html> (дата обращения: 09.03.2023). — Режим доступа: для авторизир. пользователей - <https://www.iprbookshop.ru/126577.html>

2. Демяненко, Я. М. Программирование на языке C++ : учебное пособие / Я. М. Демяненко, М. И. Чердынцева. — Москва : Ай Пи Ар Медиа, 2022. — 398 с. — ISBN 978-5-4497-2008-5. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/128007.html> (дата обращения: 14.02.2023). — Режим доступа: для авторизир. пользователей. - DOI: <https://doi.org/10.23682/128007> - <https://www.iprbookshop.ru/128007.html>

7.3. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень программного обеспечения и информационных справочных систем

В образовательном процессе используются информационные технологии, реализованные на основе информационно-образовательного портала института (www.mivlgu.ru/iop), и инфокоммуникационной сети института:

- предоставление учебно-методических материалов в электронном виде;
- взаимодействие участников образовательного процесса через локальную сеть института и Интернет;
- предоставление сведений о результатах учебной деятельности в электронном личном кабинете обучающегося.

Информационные справочные системы:

- электронная библиотечная системы "IPRBooks" (<http://www.iprbookshop.ru/>);

Программное обеспечение:

LibreOffice (Mozilla Public License v2.0)

Microsoft Visual Studio (Программа Microsoft Azure Dev Tools for Teaching (Order Number: IM126433))

QT Creator ((L)GPL)

GNU Debugger (GPL 3+)

Double Commander (GNU GPL 2+)

7.4. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

iprbookshop.ru

mivlgu.ru/iop

8. Материально-техническое обеспечение дисциплины

Лаборатория распределенных систем

12 персональных компьютеров; проектор Nec V300X; экран настенный Lumien Master Picture

9. Методические указания по освоению дисциплины

Для успешного освоения теоретического материала обучающийся: знакомится со списком рекомендуемой основной и дополнительной литературы; уточняет у преподавателя, каким дополнительным пособиям следует отдать предпочтение; ведет конспект лекций и прорабатывает лекционный материал, пользуясь как конспектом, так и учебным пособием по курсу "Объектно-ориентированное программирование".

На практических занятиях пройденный теоретический материал подкрепляется решением задач по основным темам дисциплины. Занятия проводятся в компьютерном классе, используя специальное программное обеспечение. Каждой подгруппе обучающихся преподаватель выдает задачу, связанную с разработкой и программной реализацией алгоритмов обработки информации. В конце занятия обучающие демонстрируют полученные результаты преподавателю и при необходимости делают работу над ошибками.

До выполнения лабораторных работ обучающийся изучает соответствующий раздел теории. Перед занятием студент знакомится с описанием заданий для выполнения работы, внимательно изучает содержание и порядок проведения лабораторной работы. Лабораторная работа проводится в компьютерном классе. Обучающиеся выполняют индивидуальную задачу компьютерного моделирования в соответствии с заданием на лабораторную работу. Полученные результаты исследований сводятся в отчет и защищаются по традиционной методике в классе на следующем лабораторном занятии. Необходимый теоретический материал, индивидуальное задание, шаги выполнения лабораторной работы и требование к отчету приведены в методических указаниях, размещенных на информационно-образовательном портале института.

Самостоятельная работа оказывает важное влияние на формирование личности будущего специалиста, она планируется обучающимся самостоятельно. Каждый обучающийся самостоятельно определяет режим своей работы и меру труда, затрачиваемого на овладение учебным содержанием дисциплины. Он выполняет внеаудиторную работу и изучение разделов, выносимых на самостоятельную работу, по личному индивидуальному плану, в зависимости от его подготовки, времени и других условий.

Курсовая работа выполняется в соответствии с методическими указаниями на курсовую работу. Обучающийся выбирает одну из указанных в перечне тем курсовых работ, исходя из своих интересов, наличия соответствующих литературных и иных источников. В ходе выполнения курсовой работы преподаватель проводит консультации обучающегося. На заключительном этапе обучающийся оформляет пояснительную записку к курсовой работе и выполняет ее защиту в присутствии комиссии из преподавателей кафедры.

Форма заключительного контроля при промежуточной аттестации – зачет. Для проведения промежуточной аттестации по дисциплине разработаны фонд оценочных средств и балльно-рейтинговая система оценки учебной деятельности студентов. Оценка по дисциплине выставляется в информационной системе и носит интегрированный характер, учитывающий результаты оценивания участия студентов в аудиторных занятиях, качества и своевременности выполнения заданий в ходе изучения дисциплины и промежуточной аттестации.

Программа составлена в соответствии с требованиями ФГОС ВО по направлению *09.03.02 Информационные системы и технологии* и профилю подготовки *Информационные системы и технологии*

Рабочую программу составил *к.т.н., доцент, Еремеев С.В.*_____

Программа рассмотрена и одобрена на заседании кафедры *ИС*

протокол № 21 от 05.05.2026 года.

Заведующий кафедрой *ИС* _____ *Андреианов Д.Е.*

(Подпись)

Рабочая программа рассмотрена и одобрена на заседании учебно-методической комиссии факультета

протокол № 9 от 12.05.2026 года.

Председатель комиссии ФИТР _____ *Кутарова Е.И.*

(Подпись)

(Ф.И.О.)

Фонд оценочных материалов (средств) по дисциплине
Объектно-ориентированное программирование

1. Оценочные материалы для проведения текущего контроля успеваемости по дисциплине

Семестр 3.

Рейтинг-контроль №1

1. Определение класса. Данные и методы класса.
2. Конструктор и деструктор класса. Определение. Виды конструкторов.
3. Указатели на компонентные функции.
4. Объекты класса.
5. Массив объектов класса.
6. Статические члены класса.
7. Наследование. Простое и множественное наследование
8. Конструкторы производных классов. Особенности вызова
9. Виртуальные функции. Назначение
10. Абстрактные классы. Чистые виртуальные функции.

Рейтинг-контроль №2

10. Абстрактные классы. Чистые виртуальные функции.
11. Базовые и производные классы. Атрибуты наследования.
12. Перегрузка операций. Назначение
12. Какие операции не перегружаются
13. Перегрузка унарных операций в C++ функцией-членом класса
14. Перегрузка унарных операций в C++ дружественной функцией
15. Перегрузка бинарных операций в C++ функцией-членом класса
16. Перегрузка бинарных операций в C++ дружественной функцией
17. Какие виды преобразований можно реализовать в C++
18. Преобразование из типа класса к скалярному типу
19. Преобразование из скалярного типа к типу класса
20. Преобразование из из типа одного класса к типу другого класса.
21. Конструкторы преобразований. Назначение
22. Операции преобразований. Назначений.

Рейтинг-контроль №3

1. Преимущества перед процедурным программированием.
2. Принципы разработки иерархии классов и ее использования.
3. Разрешение вопросов, связанных с неоднозначностью доступа.
4. Иерархия классов, используемая в среде Microsoft Visual Studio.
5. Программирование на C++/CLI.
6. Объектно-ориентированное программирование как основа современного программирования.
7. Совместное использование функций и перегрузка операторов.
8. Использование динамической памяти.
9. Механизм исключительных ситуаций.
10. Поддержка модульности. Разбиение программы на единицы компиляции.
11. Шаблонные классы и стандартные контейнеры.
12. Организация пользовательского интерфейса.

Семестр 3.

Вопросы к устному зачету.

Рейтинг-контроль №1

1. Новейшие направления в области создания технологий программирования. Законы эволюции программного обеспечения.
2. Программирование в средах современных информационных систем: создание модульных программ, элементы теории модульного программирования объектно-ориентированные проектирование и программирование
3. Среда Net. Framework. Общезыковая исполняющая среда (CLR). Стандарты C++. Технологии отладки
4. Объектно-ориентированный подход к проектированию и разработке программ. Сущность объектно-ориентированного подхода
5. Инкапсуляция. Понятие класса. Управление доступом к элементам класса. Конструкторы и деструкторы. Переменные объектного типа.
6. Друзья класса.
7. Наследование. Базовый класс. Управление доступом при наследовании.
8. Простое наследование. Виртуальные функции.
9. Чистые виртуальные функции. Абстрактные классы. Непрямые базовые классы. Виртуальные деструкторы. Виртуальные базовые классы
10. Полиморфизм.

Рейтинг-контроль №2

11. Методы общего полиморфизма: перегрузка операций. Реализация перегруженной операции. Перегрузка операции присваивания, инкремента, декремента, бинарных арифметических операций.
12. Методы общего полиморфизма: перегрузка операций. Перегрузка операций ввода-вывода в потоки, индексации массивов, операций выделения и освобождения памяти.
13. Методы общего полиморфизма: Преобразования типов, определяемые классом.
14. Методы общего полиморфизма: перегрузка функций. Чистый полиморфизм.
15. Параметрический полиморфизм. Шаблоны функций.
16. Параметрический полиморфизм. Шаблоны классов

Рейтинг-контроль №3

17. Классы в C#. Управление доступом. Конструкторы. Деструкторы. Наследование.
18. Свойства классов в C#. Скалярные и индексированные свойства.
19. Перегрузка операций в C#. Преобразование типов в C#.
- 20 Делегаты. события. Интерфейсы
21. Концепции программирования для Windows. Структура Windows программ. Использование Windows Forms для создания приложений с графическим интерфейсом пользователя.
22. Создание элемента управления Windows Forms

Общее распределение баллов текущего контроля по видам учебных работ для студентов

Рейтинг-контроль 1	Контрольная работа, результаты защиты практических работ	до 20 баллов
Рейтинг-контроль 2	Контрольная работа, результаты защиты практических работ	до 20 баллов
Рейтинг-контроль 3	Контрольная работа, результаты защиты практических работ	до 20 баллов
Посещение занятий студентом	Отметка в журнале посещений	до 5 баллов

Дополнительные баллы (бонусы)		до 15 баллов
Выполнение семестрового плана самостоятельной работы		до 20 баллов

2. Промежуточная аттестация по дисциплине

Перечень вопросов к экзамену / зачету / зачету с оценкой.

Перечень практических задач / заданий к экзамену / зачету / зачету с оценкой (при наличии)

ЗНАТЬ:

- С помощью чего реализуется принцип полиморфизма в C ++?
 - наличия множественного наследования.
 - наличия виртуальных методов. +
 - Использование виртуального наследования.
 - наличия абстрактных классов.
- В программе описано класс и объект `class A {public: int a, b, c; }; A * obj;` Как обратиться к атрибуту `c`?
 - `obj.c`
 - `obj->c`
 - `obj.A ->c`
 - `obj->A.c`
- Какая из перечисленных функций не может быть конструктором?
 - `void String ()` +
 - `String ();`
 - `String (String & s)`
 - `String (const int a)`
- Отметьте правильное утверждение для абстрактного класса для языка C ++.
 - Класс, у которого все методы чисто виртуальные, называется абстрактным.
 - Абстрактный базовый класс навязывает определенный интерфейс всем производным из него классам.
 - Невозможно создать объект абстрактного класса. +
 - В абстрактном классе не описываются методы вообще.
- Если в программе на языке C ++ в производном классе переопределена операция `new` то ...
 - все объекты этого класса и все объекты классов, выведенных из него, будут использовать эту операцию независимо от зоне видимости, в которой она переопределена.
 - производные от этого класса могут использовать глобальную операцию применив операцию `базовый_класс :: new.` +
 - операцию `new` нельзя переопределить.
 - в любом случае эта операция будет доступна только в пределах класса-потомка.
- Какой из перечисленных методов может быть конструктором для класса `String` в языке C ++?
 - `String * String ();`
 - `void String ();`
 - `String (String & s);` +
 - `const String (int a);`
- Какая функция, не будучи компонентом класса, имеет доступ к его защищенным и внутренним компонентам?
 - Шаблонная.
 - Полиморфная.

- Дружеская. +
- . Статическая.

8 Вызовет данный код ошибку компиляции? `class Rectangle public: int a, b; int sum (); int square (); ~Rect ();`;

- Ошибки нет, все записано верно.
- Ошибка являются: имя деструктора должно совпадать с именем класса. +
- . Ошибка являются: имя деструктора не может начинаться с маленькой буквы.
- Ошибка являются: никакой идентификатор в C ++ не может начинаться со знака «~».

9 Укажите правильное объявление виртуального метода, который принимает одно целочисленное значение и возвращает void.

- `virtual void SomeFunction (int x);` +
- `void SomeFunction (int x) virtual;`
- `virtual SomeFunction (int x);`
- `virtual void SomeFunction (int * x);`

10. Укажите правильное использование оператора friend.

- `class A {int_friend CountPass (); private: short i;};`
- . `class A {public: friend int H :: CountPass (); private: short i;};` +
- . `class A {public: int A1 :: CountPass (); friend: short i;};`
- `class A {public: friend int H :: q; short i;};`

11. Принцип объектно-ориентированного программирования, заключается в объединении атрибутов и методов объекта с целью обеспечения сохранности данных, называется:

- Наследование.
- Сочетание.
- Инициализация.
- . Инкапсуляция. +

12 В программе описано абстрактный класс A и производный от этого класса класс A1. Какой из записей заведомо неверный?

- . `A * a = new A;` +
- `A1 a1;`
- `A1 a1; A & a = a1;`
- `A1 a1; A1 a2;`

13. Выберите верное утверждение о деструктор класса в C ++.

- Деструктор принимает в качестве параметра адрес того объекта, который нужно уничтожить.

- Деструктор принимает в качестве параметра указатель this.
- Деструктор не содержит параметров. +
- Деструктор принимает в качестве параметра тот объект, который нужно уничтожить.

14. Укажите правильный вариант доступа к членам объектов (язык C ++), описанных следующим образом: `class my {char s;public: double Z; int f (int c, int d) {return c + d;}; } T1, T2;`

- `T1.Z = 23.1;` +
- `T2-> f (2,1);`
- `T1.s = '#';`
- . `my.T2-> s = 'L';`

15 Какой из вариантов записи абстрактного класса в C ++ является правильным?

- `abstract class A {virtual int f () = 0;};`
- `class A {virtual int f () = 0;};` +
- `class A {virtual int f () = 0;} abstract;`
- `class A {virtual int f ();};`

16. Какое из свойств скрывает внутренние данные объекта?

- 1) Инкапсуляция
- 2) Полиморфизм
- 3) Наследование

4) Объектно-Ориентированные

17. Какое из свойств использует виртуальные или перегружаемые элементы?

- 1) Инкапсуляция
- 2) Полиморфизм
- 3) Наследование
- 4) Объектно-Ориентированные

18. Какое из свойств строит иерархию объектов?

- 1) Инкапсуляция
- 2) Полиморфизм
- 3) Наследование
- 4) Объектно-Ориентированные

19. Какое из свойств предназначено для улучшения интерфейса работы с объектами?

- 1) Инкапсуляция
- 2) Полиморфизм
- 3) Наследование
- 4) Объектно-Ориентированные

20. Какая из операций используется для обозначения указателя?

- 1) DIV
- 2) *
- 3) {\$
- 4) ^

21. Какое из названий обозначает завершение работы динамических методов объекта?

- 1) свойство
- 2) виртуальная функция
- 3) конструктор
- 4) деструктор

22. Какое из названий обозначает общедоступные элементы объекта?

- 1) public
- 2) published
- 3) protected
- 4) private

23. Какое из названий обозначает доступные только в модуле элементы класса?

- 1) public
- 2) published
- 3) protected
- 4) private

24. Какая из операций обозначает получение адреса?

- 1) @
- 2) &
- 3) ^
- 4) *

25. В какой из областей класса элементы недоступны для потомков вне данного модуля?

- 1) public
- 2) published

- 3) protected
- 4) private

26. Какой из терминов обозначает события объекта?

- 1) Canvas
- 2) Caption
- 3) Events
- 4) Enabled

27. Какое из свойств связано с соединением полей, методов и свойств в одном объекте?

- 1) Инкапсуляция
- 2) Наследование
- 3) Полиморфизм
- 4) Визуальность

28. Какое из названий обозначает создание объекта данного класса?

- 1) переопределение функции
- 2) виртуальная функция
- 3) конструктор
- 4) деструктор

29. Динамическая структура, которая имеет две основные операции: добавление в «хвост» и извлечение из «головы» является

- 1) очередью
- 2) стеком
- 3) списком
- 4) файлом

30. Динамическая структура, которая имеет одну точку доступа к его элементам («голова»), называется

- 1) очередью
- 2) стеком
- 3) списком
- 4) файлом

31. Упорядоченная динамическая структура, каждый элемент которой содержит ссылку, связывающую его со следующим элементом, называется

- 1) очередью
- 2) стеком
- 3) списком
- 4) файлом

УМЕТЬ

32. Что будет выведено в результате выполнения программы?

```
1     class Program
2     {
3         static void Main(string[] args)
4         {
5             try
6             {
7                 var a1 = new A1();
8                 Console.WriteLine("1");
9                 var a2 = new A2();
10            }

```

```

11.     catch (Exception)
12.     {
13.         Console.WriteLine("3");
14.     }
15.     Console.ReadLine();
16. }
17. }
18.
19. class A1
20. {
21.     public int B1;
22. }
23.
24. class A2
25. {
26.     public int B2;
27.
28.     public A2(int b2)
29.     {
30.         B2 = b2;
31.     }
32. }
1) 1
2) 3
3) 13
4) Возникнет ошибка на этапе компиляции

```

33. Что будет выведено в результате выполнения программы?

```

1.     class Program
2.     {
3.         static void Main(string[] args)
4.         {
5.             Console.WriteLine(A.B);
6.             var a1 = new A();
7.             Console.WriteLine(A.B);
8.             a1.Write();
9.             Console.ReadLine();
10.        }
11.    }
12.
13.    public class A
14.    {
15.        public static int B;
16.
17.        public A()
18.        {
19.            B = 3;
20.        }
21.
22.        public void Write()
23.        {
24.            Console.WriteLine(B);

```

```
25.     }
26.
27.     static A()
28.     {
29.         B = 5;
30.     }
31. }
```

- 1) 033
- 2) 533
- 3) 553
- 4) Возникнет ошибка на этапе компиляции

34. Что будет выведено в результате выполнения программы?

```
1.     class Program
2.     {
3.         static void Main(string[] args)
4.         {
5.             var a = new A(2) {B = 3};
6.             Console.Write(a.B);
7.             Console.ReadLine();
8.         }
9.     }
10.
11.     public class A
12.     {
13.         public int B { get; set; }
14.
15.         public A(int b)
16.         {
17.             Console.Write("1");
18.             B = b;
19.         }
20.     }
```

- 1) 3
- 2) 12
- 3) 13
- 4) Возникнет ошибка на этапе компиляции

35. Какие утверждения относительно языка C# верны?

- 1) Допустимо множественное наследование
- 2) Класс может реализовать несколько интерфейсов
- 3) Интерфейс может наследоваться от множества других интерфейсов
- 4) Нельзя наследовать от класса, помеченного ключевым словом sealed

36. Что будет выведено в результате выполнения программы?

```
1.     class Program
2.     {
3.         static void Main(string[] args)
4.         {
5.             var b = new B();
6.             var c = new C();
```

```

7.
8.     Console.Write(b.Sum(2, 3));
9.     Console.Write(c.Sum(2, 3));
10.    Console.ReadLine();
11.    }
12.  }
13.
14.  class A
15.  {
16.      public virtual int Sum(int a, int b)
17.      {
18.          return a + b;
19.      }
20.  }
21.
22.  class B : A
23.  {
24.
25.  }
26.
27.  class C : A
28.  {
29.      public override int Sum(int a, int b)
30.      {
31.          return a + b + 1;
32.      }
33.  }
1.    55
2.    56
3.    66
4.    Возникнет ошибка на этапе компиляции

```

37 Что будет выведено в результате выполнения программы?

```

1.    class Program
2.    {
3.        static void Main(string[] args)
4.        {
5.            try
6.            {
7.                var a1 = new A1();
8.                Console.Write("1");
9.                var a2 = new A2();
10.           }
11.           catch (Exception)
12.           {
13.               Console.Write("3");
14.           }
15.           Console.ReadLine();
16.       }
17.   }
18.
19.   class A1
20.   {
21.       public int B1;

```

```
22.     }
23.
24.     class A2
25.     {
26.         public int B2;
27.
28.         public A2(int b2)
29.         {
30.             B2 = b2;
31.         }
32.     }
```

1. 1
2. 3
3. 13
4. Возникнет ошибка на этапе компиляции

38. Будет ли компилироваться следующий фрагмент кода?

```
try
{
    FileStream F = new FileStream("myfile.txt");
    string s = F.ReadLine();
}
catch (IOException) { }
finally
{
    F.Close();
}
```

- Да;
- Нет.

39. Что делает оператор %?

- Переводит дробное число в проценты
- Возвращает остаток от деления
- Возвращает процентное соотношение двух операндов
- Форматирует значения разных типов в строку

40. Сколько родительских классов может иметь производный класс?

- Не больше одного
- Всегда один
- Не больше двух
- Любое количество

41. Какой класс является базовым для всех классов в C#?

Ответ:

42. Первый сценарий (компиляция без ключа /checked):

```
byte mask = 255;
mask++;
```

Второй сценарий:

```
byte mask = 255;
checked
{
```

```
    mask++;  
}
```

Третий сценарий (компиляция с ключом /checked):

```
byte mask = 255;  
mask++;
```

Четвертый сценарий:

```
byte mask = 255;  
unchecked  
{  
    mask++;  
}
```

Перечислите чему равно mask во всех 4 случаях.

-255, 0, 1, 0

-0, -1, -1, 0

-1, вызовется исключение OverflowException, 0, 256

-0, вызовется исключение OverflowException, вызовется исключение OverflowException,

0

-256, -1, -1, 0

43.

Выберите допустимые прототипы метода Main.

-static int Main(string[] args) { }

-static int main() { }

-unsafe static void Main(int argc, char* argv[]) { }

-static void main() { }

-static void Main(string[] args) { }

static void Main() { }

static int Main() { }

44. Выберите элементы, которые нельзя пометить атрибутом.

-Интерфейсы

-Все перечисленное можно пометить атрибутом

-Возвращаемые значения

-Классы

-Методы

-Структуры

45 Выберите конструкцию, которая наиболее схожа по функциональности со следующим кодом:

```
using (StreamWriter writer = new StreamWriter("example.txt"))  
{  
}  
-StreamWriter writer = new StreamWriter("example.txt"); writer.Close();  
-StreamWriter writer; try { writer = new StreamWriter("example.txt"); } catch (Exception e)  
{ -Trace.WriteLine(e.ToString()); } finally { writer.Dispose(); }  
-Оператору using() { } нет альтернативы  
-StreamWriter writer; try { writer = new StreamWriter("example.txt"); } finally {  
writer.Dispose(); }
```

46. Что произойдет при исполнении следующего кода?

```
int i = 5;
```

```
object o = i;
```

```
long j = (long)o;
```

-Значение переменной j предсказать нельзя

-Средой исполнения будет вызвано исключение `InvalidCastException`

-Произойдет ошибка времени компиляции

-Ошибок не произойдет. Переменная j будет иметь значение 5

47. Что произойдет при компиляции проекта, где используется класс, структура, интерфейс или перечисление, помеченное атрибутом `Obsolete`?

-Что произойдет при компиляции проекта, где используется класс, структура, интерфейс или перечисление, помеченное атрибутом `Obsolete`?

-Будет выведено предупреждение о том, что данный тип устарел, но сборка будет создана

-Сборка будет создана, но при запуске произойдет ошибка времени выполнения

-Атрибут `Obsolete` никак не влияет на компиляцию

-Нет нужного варианта ответа

48. Выберите правильные объявления метода с корректным использованием ключевого слова `params`:

```
-public void Foo(params object[] objectArray, params int[] intArray) { }
```

```
-public void Foo(params double[] array) { }
```

```
-public void Foo(params int[] intArray, params double[] doubleArray) { }
```

```
-public void Foo(params double[] array, double volume) { }
```

```
-public void Foo(double volume, params double[] array) { }
```

49. Чем опасны подобные конструкции?

```
public class Widget
```

```
{
```

```
    private int width;
```

```
    public int IncrementWidth
```

```
    {
```

```
        get { return ++width; }
```

```
    }
```

```
}
```

```
class Program
```

```
{
```

```
    static void Main(string[] args)
```

```
    {
```

```
        Widget widget = new Widget();
```

```
        int width = widget.IncrementWidth;
```

```
    }
```

```
}
```

-Неправильно сделан инкремент поля `width`, нужно `++width` заменить на `width++`

-Ничем, код полностью корректен

-При отладке кода в режиме `Debug` при просмотре свойства `IncrementWidth` объекта `widget` будет увеличиваться значение `width`, что повлечет сложности при отладке

-Нет блока `set {}` в свойстве `IncrementWidth`

50. Какая утилита из .NET Framework служит для генерации XML-документации из комментариев `///` и `/** */` комментариев в коде?

- ilasm.exe с ключом `/doc:FileName.xml`
- tlbexp.exe
- xsd.exe с ключом `/doc:FileName.xml`
- csc.exe с ключом `/doc:FileName.xml`
- Нет правильного ответа

51. С помощью какой утилиты .NET Framework можно сгенерировать C# код класса по имеющейся XML схеме?

- xsd.exe с ключом `/c`
- xml.exe с ключом `/c`
- tlbimp.exe с ключом `/c`
- Автоматизировать это нельзя – код придется писать самому
- Используя утилиту `ildasm.exe` с ключом `/c`

52. Стоит задача обработать большой XML-файл. Какой класс для этого следует выбрать, чтобы обработка прошла наиболее быстро?

- XmlTextReader
- XmlDocument
- XmlPathDocument
- XmlSerializer

53. Чтобы использовать `unsafe` код в приложении, необходимо ...

- Пометить методы, где используется небезопасный код атрибутом `Unsafe`
- Пометить методы, где используется небезопасный код с помощью ключевого слова `unsafe`
- Пометить методы, где используется небезопасный код с помощью ключевого слова `fixed`
- Компилировать код приложения с ключом `/unsafe`

54. Каким образом нужно применить приведенный атрибут, чтобы одновременно присвоить значения свойствам `Description` и `Price`?

```
public class Product : Attribute {
    public Product(string description) {
        Description = description;
    }
    private string description;
    public string Description {
        get { return description; }
        set { description = value; }
    }
    private float price;
    public float Price {
        get { return price; }
        set { price = value; }
    }
}
```

- Это невозможно сделать
- `[Product(Price = 149.95f, "CPU")]`
- `[Product("CPU", Price = 149.95f)]`
- `[Product("CPU", 149.95f)]`

-[Product(Description = "CPU", Price = 149.95f)]

55. Что произойдет при Binary сериализации объекта widget?

```
public interface IWidget
{
    int Index
    {
        get;
        set;
    }
}
[Serializable]
public class Widget : IWidget
{
    private int index;
    public int Index
    {
        get { return index; }
        set { index = value; }
    }
}
// Метод Main()
IWidget widget = new Widget();
widget.Index = 10;
using (FileStream writer = new FileStream("data.dat", FileMode.Truncate))
{
    BinaryFormatter serializer = new BinaryFormatter ();
    serializer.Serialize(writer, widget);
}
```

-Ошибка времени компиляции: для сериализации нужно использовать не StreamWriter, а только BinaryWriter

-Ошибка времени исполнения: интерфейс IWidget должен быть помечен атрибутом Serializable

-Ошибка времени исполнения: класс Widget должен быть помечен атрибутом BinarySerializable, а не Serializable

-Будет вызвано исключение NotSupportedException с сообщением, что нельзя сериализовать интерфейс IWidget

-Объект widget будет успешно сериализован

56. Что произойдет при XML сериализации объекта widget?

```
public interface IWidget
{
    int Index
    {
        get;
        set; }
}
[Serializable]
public class Widget : IWidget
{
    private int index;
    public int Index
    {
        get { return index; }
        set { index = value; }
    }
}
```

```

}
// Метод Main()
IWidget widget = new Widget();
widget.Index = 10;
using (StreamWriter writer = new StreamWriter("data.xml"))
{
    XmlSerializer serializer = new XmlSerializer(typeof(IWidget));
    serializer.Serialize(writer, widget);
}

```

-Ошибка времени исполнения: класс Widget должен быть помечен атрибутом XmlSerializable, а не Serializable

-Ошибка времени компиляции: для сериализации нужно использовать не StreamWriter, а только XmlWriter

-Объект widget будет успешно сериализован

-Ошибка времени исполнения: интерфейс IWidget должен быть помечен атрибутом Serializable

-Будет вызвано исключение NotSupportedException с сообщением, что нельзя сериализовать интерфейс Iwidget

57. Как называется технология, благодаря которой возможно взаимодействие управляемого кода (managed code) с Win32 API функциями и COM-объектами?

-Remoting

-WebServices

-CodeDOM

-Interop

-Reflection

58. Какие операторы нужно переопределить у класса Digit, чтобы были возможными преобразования:

```

Digit digit = new Digit(100);
decimal decimalDigit = digit;
byte byteDigit = (byte)digit;
-public static implicit operator byte(Digit digit) { }
-public static explicit operator byte(Digit digit) { }
-Нужные операторы не приведены
-public static implicit operator decimal(Digit digit) { }

```

59. Какие ошибки присутствуют в следующем коде?

```

public interface IWidget {
    void Draw();
}
public interface IControl : IWidget {
    void Move(int x, int y);
}
public class Widget : IWidget {
    public void Draw() { }
}
public class Control : Widget, IControl {
    public void Move(int x, int y) { }
}

```

-Тела методов Move() и Draw() не должны оставаться пустыми

-Нельзя одновременно наследоваться от Widget, который реализует IWidget, и реализовывать IControl, который наследуется от Iwidget

- В Control не реализован метод Draw()
- Ошибок нет

60. Выберите средства, которые предоставляет C# для условной компиляции.

- Директива #endif
- Директива #typedef
- Директива #else
- Директива #elseif
- Атрибут Conditional
- Директива #define
- Директива #if

61. Выберите правильный вариант переопределенного метода GetHashCode() у класса Point (будем считать, что Equals() уже переопределено правильно)

```
public class Point
{
    public int x;
    public int y;
}
```

- public override int GetHashCode() { return x | y; }
- public override int GetHashCode() { return x * y; }
- Нет правильного ответа
- public override int GetHashCode() { return x & y; }
- public override int GetHashCode() { return x ^ y; }

62. Выберите операторы, которые необходимо переопределять попарно.

- > и <
- != и ==
- ++ и --
- * и /
- >= и <=
- + и -

63. Чем отличаются константы и доступные только для чтения поля?

- Ничем не отличаются
- Доступные только для чтения поля инициализируются во время компиляции, константы - во время выполнения
- Константы можно изменять, а доступные только для чтения поля нет
- Константы инициализируются во время компиляции, доступные только для чтения поля - во время выполнения

64. Когда вызываются статические конструкторы классов в C#?

- После каждого обращения к статическим полям, методам и свойствам
- Строгий порядок вызова не определен
- Статических конструкторов в C# нет
- Один раз при первом создании экземпляра класса или при первом обращении к статическим членам класса

65. Реализацией какого паттерна (шаблона проектирования) являются события в C#?

- Декоратор (Decorator)
- Шаблонный метод (Template Method)
- Посетитель (Visitor)
- Издатель-подписчик (Publisher-Subscriber)

66. Чтобы вывести на экран число типа float 10.56f с количеством цифр после запятой равным 3 и шириной равной 10 символам нужно написать:

```
-Console.WriteLine("{3:10F0}", 10.56f);  
-Console.WriteLine("{10:0:3F}", 10.56f);  
-Console.WriteLine("{0, 3F:10}", 10.56f);  
-Console.WriteLine("{0, 10:F3}", 10.56f);
```

67. По какой причине данный код не скомпилируется?

```
public delegate void EventHandler();  
public class Control  
{  
    public event EventHandler Invalidate;  
}  
public class Button : Control  
{  
    public void Draw()  
    {  
        if (Invalidate != null)  
        {  
            Invalidate();  
        }  
    }  
}
```

-Вызывать событие напрямую может только класс, в котором они объявлены

-Нельзя сравнивать событие с null

-Событие всегда равно null

-У события отсутствуют обязательные части add { } и remove { }

68. Чем отличаются перечисление State от перечисления StateFlags, помеченного атрибутом Flags, и что будет выведено на консоль при выполнении следующего кода:

```
public enum State  
{  
    ToolBox,  
    Menu,  
    StatusBar  
}  
[Flags]  
public enum StateFlags  
{  
    ToolBox,  
    Menu,  
    StatusBar  
}  
class Program  
{  
    static void Main(string[] args)  
    {  
        Console.WriteLine(State.Menu | State.StatusBar);  
        Console.WriteLine(StateFlags.Menu | StateFlags.StatusBar);  
    }  
}
```

- Будет выведено «Menu, StatusBar» и «3». StateFlags нельзя использовать как битовое поле, в отличие от State
- Код не откомпилируется
- Будет выведено «Menu, StatusBar» и «Menu, StatusBar». Как State, так и StateFlags можно использовать как битовые поля
- Будет выведено «3» и «Menu, StatusBar». Только перечисления, помеченные атрибутом Flags, можно использовать как битовые поля
- Будет выведено «3» и «3». StateFlags и State в любом случае нельзя использовать как битовые поля

69. Какие ошибки допущены в классе?

```
private class Money
{
    public virtual abstract void Transfer();
    public abstract Money Exchange();
    public abstract decimal Value
    {
        get;
        set;
    }
}
```

- Нельзя использовать private для классов
- Свойство Value не может быть помечено abstract
- В объявлении класса пропущено ключевое слово abstract
- Метод Transfer() не может быть одновременно помечен как виртуальный и абстрактный

70. Есть ли ошибки в следующем коде, и если да то какие?

```
public class Description : Attribute
{
    public Description(decimal money)
    {
        this.money = money;
    }
    private decimal money;
    public decimal Money
    {
        get { return money; }
    }
}
```

```
[Description(10)]
public class Widget
{
}
```

- В конструктор нужно передавать 10.0M, а не 10
- Недостает set { } для свойства Money
- Ошибок нет, код полностью корректен
- Тип decimal не разрешено использовать в атрибутах

71. Как правильно объявить implicit оператор, преобразующий объект класса Widget в int?

```
-public static implicit int(Widget widget) { }
```

```
-public static operator int(Widget widget) { }  
-public static implicit operator int(Widget widget) { }  
-public implicit operator int(Widget widget) { }
```

72. Что будет выведено на экран после выполнения кода и чему будет равно значение поля `Game.player.score`?

```
public class Player  
{  
    public Player(int score)  
    {  
        this.score = score;  
        Console.WriteLine(score);  
    }  
    public int score;  
}  
public class Game {  
    public static Player player = new Player(10);  
  
    static Game() {  
        player = new Player(20);  
    }  
}
```

// ...Метод Main()

```
Game game = new Game();
```

-Будет выведено 20, затем 10, а поле `score` равно 20

-Будет выведено 10, затем 20, а поле `score` равно 10

-Будет выведено 10, затем 20, а поле `score` равно 20

-Произойдет ошибка времени выполнения

-Нельзя сказать – строго не определен приоритет вызова

73. Что будет выведено на консоль при выполнении следующего кода?

```
int i = 0;  
while (i < 10)  
{  
    Console.WriteLine(i);  
    i++;  
}
```

```
while (i > 10);
```

-Это бесконечный цикл: будут выведены цифры 0, 1, 2, 3... до `Int32.MaxValue`

-Это бесконечный цикл: будут выведены цифры от 0 до 10, а далее будет повторяться цифра 10

-Код не скомпилируется

-Будут выведены цифры от 0 до 9

74. Выберите правильные варианты, в которых пространство имен `System` содержит пространство имен `Customizer`.

```
-namespace System::Customizer { }
```

```
-namespace System.Customizer { }
```

-Нельзя создавать собственные пространства имен в пространстве имен `System`

```
-namespace System { namespace Customizer { } }
```

75. Каковы будут результаты выполнения кода (консольное приложение)?

```

public class Author : Attribute
{
    public Author(string name)
    {
        Console.WriteLine(name);
    }
}
[Author("Daniel Pols")]
class Program
{
    static void Main(string[] args)
    {
    }
}

```

- На консоль будет выведено “Daniel Pols”
- Ошибка при компиляции кода: строка Console.WriteLine(name);
- На консоль ничего не будет выведено
- Ошибка при компиляции кода: строка [Author(

76. Каким образом можно перехватить добавление и удаление делегата из события?

- Использовать ключевые слова get и set
- Переопределить операторы + и – для делегата
- Такая возможность не предусмотрена
- Для этого существуют специальные ключевые слова add и remove

77.Какая технология позволяет производить компиляцию и генерировать IL код во время выполнения приложения?

- CodeDOM
- ADO .NET
- Remoting
- Такой технологии нет
- WebServices

78.Перечислите возможности структуры (value-типов).

- Определение индекса
- Переопределение конструктора по умолчанию
- Наследование от классов
- Использование событий
- Определение статического конструктора
- Множественное наследование
- Реализация интерфейсов
- Boxing и Unboxing

79. Есть ли ошибка в коде, и если да, то какая?

```

public class Base
{
    public const string Id = "123-982-232";
    public static readonly string Date = "12/03/2010";
}
public class Derive : Base
{
    public void ShowId()

```

```

        {
            Console.WriteLine(Id);
        }
        public void ShowDate()
        {
            Console.WriteLine(Date);
        }
    }

```

- Константные поля не наследуются, то есть поле Id не доступно из класса Derive
- Ошибок нет, Id и Date доступны из класса Derive
- Статические, доступные только для чтения поля не наследуются, то есть поле Date не доступно из класса Derive
- Доступные только для чтения уже статичны, поэтому слово static лишнее при описании поля Date

80. Что будет выведено на экран после выполнения следующего кода?

```

class Widget { }
class Program {
    static void Main(string[] args) {
        Widget widget = new Widget();
        WeakReference reference = new WeakReference(widget);
        Console.WriteLine(reference.Target == null);
        widget = null;
        GC.Collect();
        Console.WriteLine(reference.Target == null);
    }
}

```

- False False. До и после вызова на объект Widget будет ссылаться reference.Target
- False True. На объект Widget ссылается как слабая ссылка, так и widget. При вызове GC.Collect() сборщик проверяет, что на объект Widget ссылается только слабая ссылка и удаляет его
- True False. После вызова GC.Collect() сборщик проверяет, что на объект Widget ничто не ссылается и присваивает reference.Target ссылку на этот объект
- True True. Независимо от вызова сборщика мусора слабая ссылка не будет ссылаться на объект Widget

ВЛАДЕТЬ

81. Для определения иерархии классов связать отношением наследования классы: студент, преподаватель, персона, завкафедрой; Из перечисленных классов выбрать один, который будет стоять во главе иерархии. Это абстрактный класс. Написать программу, демонстрирующую работу с объектами созданных классов.

82. Для определения иерархии классов связать отношением наследования классы: служащий, персона, рабочий, инженер; Из перечисленных классов выбрать один, который будет стоять во главе иерархии. Это абстрактный класс. Написать программу, демонстрирующую работу с объектами созданных классов.

83. Для определения иерархии классов связать отношением наследования классы: рабочий, кадры, инженер, администрация; Из перечисленных классов выбрать один, который будет стоять во главе иерархии. Это абстрактный класс. Написать программу, демонстрирующую работу с объектами созданных классов.

84. Для определения иерархии классов связать отношением наследования классы: деталь, механизм, изделие, узел;; Из перечисленных классов выбрать один, который будет

стоять во главе иерархии. Это абстрактный класс. Написать программу, демонстрирующую работу с объектами созданных классов.

85. Создать объект - однонаправленный список, в котором определены операции, ++ добавляет в конец списка, -- удаляет элемент из списка. (Как постфиксными так префиксными).

86. Создать объект стек, с перегруженными операциями +, *, =, +=, и для вытаскивания из стека --. () - выдает под-стек.

87. Определить класс - комплексные числа, перегрузив различные операторы, +, -, ++, -, +=, -=, *, ./, *=, /=, !, !=, ==, >, <, >=, <=, >, <, >=, <=, +, -. Ввода, вывода в поток. Сложение и вычитание должно производиться как с элементами данного класса так и со встроенными float.

88. Создать объект типа стек. Перегрузить оператор ++, --, !, !=, ==, >, <, >=, <=, +, -. Ввод, вывод в поток.

89. Создать объект - однонаправленный список, в котором определены операции, + - добавляет в конец списка, += добавляет в этот же список в конец списка. - удаляет указанный элемент из списка (номер элемента через параметр), = - присвоение списков, сравнение списков ==, !=, >, <, >=, <=, [] получение элемента списка, ++ - устанавливает указатель на следующий элемент. () выдать подсписок от первого до второго элемента.

90. Создать объект - очередь с перегруженными операциями ++ как функциями-членами, -- как дружественными функциями. (Как постфиксными так префиксными).

91. Создать класс вектор, содержащий ссылку на long double, размерность вектора и переменную ошибки. Класс имеет конструкторы по умолчанию, конструктор с одним и двумя параметрами, конструктор копирования и деструктор. Определить оператор +, -, *, - как дружественные функции, =, +=, -=, *=, [] - как функции-члены. Определить операторы =, +, -, *, +=, -=, *= с целым числом операторы ++ и --. Определить операторы =, +, -, *, +=, -=, *= с числом типа long double, операторы ++ и --. Определить функцию печати. Сравнить время работы созданного класса и встроенного массива типа long double. Перегрузить операторы вывода и ввода в поток.

92. Создать класс вещественных чисел. Определить оператор -, как функцию-член и + как дружественную функцию.

93. Создать два класса вектор (double *) и матрица (double **). Определить конструкторы - по умолчанию, с параметром, для класса матрица с двумя параметрами, копирования, деструкторы. Определить функцию умножения матрицу на вектор как дружественную.

94. Создать класс целых чисел Integer. Определить перегруженную функцию, возвращающую минимальное из двух аргументов. Функция не является членом класса целых чисел. Перегруженные функции имеют аргументы типа int, double, Integer. Тело перегруженных функций должны быть одинаковыми.

95. Определить два класса, строку с преобразованием из char * в строку и обратно и Double с преобразованием из double и обратно, а также взаимное преобразование String и Double.

96. Создать два класса вектор (float *) и матрица (float **). Определить конструкторы - по умолчанию, с параметром, для класса матрица с двумя параметрами, копирования, деструкторы. Определить функцию умножения матрицу на вектор как дружественную.

97. Создать объект-контейнер stack и заполнить его данными типа int.

98. Создать объект-контейнер queue и заполнить его данными типа int.

99. Создать элемент управления, определяющий число слов в строке, хранимой в некотором свойстве. Число слов отображается в отдельном свойстве.

100. Создать элемент управления, проверяющий парность круглых скобок, вводимых в свойство строкового типа. В случае парности в другом свойстве отображается значение «true» иначе «false».

Методические материалы, характеризующие процедуры оценивания

На основе типовых заданий из предыдущего раздела программным комплексом информационно-образовательного портала МИ ВлГУ формируются в автоматическом режиме тестовые задания для студентов: 15 вопросов в тесте (8 вопросов Блока 1, 4 вопроса Блока 2 и 3 вопроса Блока 3). Программный комплекс формирует индивидуальные задания для каждого зарегистрированного в системе студента и устанавливает время прохождения тестирования. Результатом тестирования является процент правильных ответов, с учетом индивидуального семестрового рейтинга студента формируется экзаменационная оценка за семестр 3.

На основе перечня вопросов к устному зачету, преподавателем формируются листы с двумя вопросами из перечня. В результате ответов на поставленные вопросы и с учетом индивидуального семестрового рейтинга студента проставляется зачет за семестр 4.

Максимальная сумма баллов, набираемая студентом по дисциплине равна 100.

Оценка в баллах	Оценка по шкале	Обоснование	Уровень сформированности компетенций
Более 80	«Отлично»	Содержание курса освоено полностью, без пробелов, необходимые практические навыки работы с освоенным материалом сформированы, все предусмотренные программой обучения учебные задания выполнены, качество их выполнения оценено числом баллов, близким к максимальному	Высокий уровень
66-80	«Хорошо»	Содержание курса освоено полностью, без пробелов, некоторые практические навыки работы с освоенным материалом сформированы недостаточно, все предусмотренные программой обучения учебные задания выполнены, качество выполнения ни одного из них не оценено минимальным числом баллов, некоторые виды заданий выполнены с ошибками	Продвинутый уровень

50-65	«Удовлетворительно»	Содержание курса освоено частично, но пробелы не носят существенного характера, необходимые практические навыки работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий, возможно, содержат ошибки	<i>Пороговый уровень</i>
Менее 50	«Неудовлетворительно»	Содержание курса не освоено, необходимые практические навыки работы не сформированы, выполненные учебные задания содержат грубые ошибки	<i>Компетенции не сформированы</i>

3. Задания в тестовой форме по дисциплине

Примеры заданий:

- С помощью чего реализуется принцип полиморфизма в C ++?
 - наличия множественного наследования.
 - наличия виртуальных методов. +
 - Использование виртуального наследования.
 - наличия абстрактных классов.
- В программе описано класс и объект `class A {public: int a, b, c; }; A * obj;` Как обратиться к атрибуту `c`?
 - `obj.c`
 - `obj-> c +`
 - `obj A -> -> c`
 - `obj-> A.c`
- Какая из перечисленных функций не может быть конструктором?
 - `void String () +`
 - `String ();`
 - `String (String & s)`
 - `String (const int a)`
- Отметьте правильное утверждение для абстрактного класса для языка C ++.
 - Класс, у которого все методы чисто виртуальные, называется абстрактным.
 - Абстрактный базовый класс навязывает определенный интерфейс всем производным из него классам.
 - Невозможно создать объект абстрактного класса. +
 - В абстрактном классе не описываются методы вообще.
- Если в программе на языке C ++ в производном классе переопределена операция `new` то ...
 - все объекты этого класса и все объекты классов, выведенных из него, будут использовать эту операцию независимо от зоне видимости, в которой она переопределена.
 - производные от этого класса могут использовать глобальную операцию применив операцию `базовый_класс :: new.` +
 - операцию `new` нельзя переопределить.
 - в любом случае эта операция будет доступна только в пределах класса-потомка.
- Какой из перечисленных методов может быть конструктором для класса `String` в языке C ++?
 - `String * String ();`

- void String ();
- . String (String & s); +
- . const String (int a);

7. Какая функция, не будучи компонентом класса, имеет доступ к его защищенным и внутренним компонентам?

- Шаблонная.
- Полиморфная.
- Дружеская. +
- . Статическая.

8 Вызовет данный код ошибку компиляции? class Rectangle public: int a, b; int sum (); int square (); ~Rect (); };

- Ошибки нет, все записано верно.
- Ошибка является: имя деструктора должно совпадать с именем класса. +
- . Ошибка является: имя деструктора не может начинаться с маленькой буквы.
- Ошибка является: никакой идентификатор в C ++ не может начинаться со знака «~».

9 Укажите правильное объявление виртуального метода, который принимает одно целочисленное значение и возвращает void.

- virtual void SomeFunction (int x); +
- void SomeFunction (int x) virtual;
- virtual SomeFunction (int x);
- virtual void SomeFunction (int * x);

10. Укажите правильное использование оператора friend.

- class A {int_friend CountPass (); private: short i;};
- . class A {public: friend int H :: CountPass (); private: short i;}; +
- . class A {public: int A1 :: CountPass (); friend: short i;};
- class A {public: friend int H :: q; short i;};

Полный перечень тестовых заданий с указанием правильных ответов, размещен в банке вопросов на информационно-образовательном портале института по ссылке <https://www.mivlgu.ru/iop/question/edit.php?cmid=5443&cat=38670%2C17725&qpage=0&recurse=1&showhidden=0&qshowtext=0>

Оценка рассчитывается как процент правильно выполненных тестовых заданий из их общего числа.