



Частное профессиональное образовательное учреждение
«Северо-Кавказский межотраслевой колледж»

ИНН/КПП 0600001944/060001001 ОГРН 1220600000455
386101, Республика Ингушетия, г. Назрань, пр-т. Базоркина, д. 3

ПРИНЯТО
на заседании учебно-методического
совета Протокол
от «13» апреля 2026 г. № 3

КОМПЛЕКТ КОНТРОЛЬНО-ОЦЕНОЧНЫХ СРЕДСТВ МДК 01.01 Разработка программных модулей

Специальность: 09.02.07 Информационные системы и программирование

*Тип образовательной программы: Программа подготовки специалистов
среднего звена*

Квалификация: Программист

Форма обучения: очная

Назрань, 2026

Содержание

1. ОБЩИЕ ПОЛОЖЕНИЯ.....	3
2. ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОСВОЕНИЯ МДК	4
3. ФОРМЫ И МЕТОДЫ КОНТРОЛЯ И ОЦЕНИВАНИЯ	5
4. ПАСПОРТ КОМПЛЕКТА КОНТРОЛЬНО-ОЦЕНОЧНЫХ СРЕДСТВ ПО МДК.01.01....	6
5. ОЦЕНОЧНЫЕ СРЕДСТВА ТЕКУЩЕГО КОНТРОЛЯ ПО МДК.....	8
6. ОЦЕНОЧНЫЕ СРЕДСТВА ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО МДК	22
7. ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ	26

1. ОБЩИЕ ПОЛОЖЕНИЯ

Комплект контрольно-оценочных средств (КОС) междисциплинарного курса МДК 01.01 Разработка программных модулей является частью ПМ.01 программы подготовки специалистов среднего звена по специальности 09.02.07 Информационные системы и программирование (квалификация «Программист»).

КОС предназначен для оценки достижения запланированных результатов обучения по междисциплинарному курсу и включает оценочные средства для проведения текущего контроля успеваемости и промежуточной аттестации.

Текущий контроль успеваемости осуществляется в пределах учебного времени, отведённого на изучение МДК. Результаты текущего контроля фиксируются в журнале учебных занятий по пятибалльной системе («5», «4», «3», «2»).

Промежуточная аттестация по МДК проводится в форме экзамена. Экзамен проводится после завершения освоения МДК в сроки, установленные календарным учебным графиком. Вопросы и задания доводятся до сведения обучающихся в течение первых двух месяцев от начала обучения.

2. ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОСВОЕНИЯ МДК

Общие компетенции (ОК)

Код	Формулировка компетенции
ОК 01	Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам
ОК 02	Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности
ОК 03	Планировать и реализовывать собственное профессиональное и личностное развитие
ОК 04	Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами
ОК 05	Осуществлять устную и письменную коммуникацию на государственном языке
ОК 09	Использовать информационные технологии в профессиональной деятельности
ОК 10	Пользоваться профессиональной документацией на государственном и иностранном языках

Профессиональные компетенции

Код	Формулировка компетенции
ПК 1.1	Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием
ПК 1.2	Разрабатывать программные модули в соответствии с техническим заданием
ПК 1.5	Осуществлять рефакторинг и оптимизацию программного кода

В результате освоения МДК обучающийся должен:

Иметь практический опыт:

- разработки кода программного продукта на основе готовой спецификации на уровне модуля;
- использования инструментальных средств на этапе разработки;
- оптимизации и рефакторинга программного кода.

Уметь:

- осуществлять разработку кода программного модуля на языках высокого уровня;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять оптимизацию и рефакторинг программного кода;
- оформлять документацию на программные средства.

Знать:

- основные этапы разработки программного обеспечения;
- основные принципы технологии структурного и объектно-ориентированного программирования;
- способы оптимизации и приёмы рефакторинга.

3. ФОРМЫ И МЕТОДЫ КОНТРОЛЯ И ОЦЕНИВАНИЯ

Формы текущего контроля по МДК:

- устный опрос (фронтальный, индивидуальный, комбинированный);
- тестирование (письменное или компьютерное);
- письменная проверка (ответы на вопросы, решение задач, составление схем, выполнение заданий для самостоятельной работы);
- практическая проверка (при проведении практических и лабораторных занятий);
- самоконтроль и взаимопроверка.

Критерии оценки профессиональных компетенций (фрагмент)

Компетенция	Критерии оценки	Методы оценки
ПК 1.1 Формировать алгоритмы	«Отлично» – алгоритм разработан в соответствии с ТЗ, оценена сложность, оформлен по стандартам.	Экзамен (практическое задание), защита отчётов по работам, наблюдение.
ПК 1.2 Разрабатывать программные модули	«Отлично» – модуль разработан по алгоритму, соответствует ТЗ, документация оформлена.	Экзамен, защита отчётов, курсовой проект.
ПК 1.5 Осуществлять рефакторинг и оптимизацию	«Отлично» – выявлены фрагменты некачественного кода, выполнен рефакторинг, проведена оптимизация.	Экзамен, защита отчётов.

Оценки по ОК выставляются на основе экспертного наблюдения за выполнением работ.

4. ПАСПОРТ КОМПЛЕКТА КОНТРОЛЬНО-ОЦЕНОЧНЫХ СРЕДСТВ ПО МДК.01.01

№	Наименование темы	Результаты обучения (умения, знания)	ПК, ОК	Текущий контроль успеваемости
Тема 1.1.1 Жизненный цикл ПО				
1	Понятие ЖЦ ПО. Этапы ЖЦ ПО	31	ОК 01, ОК 02	Устный опрос, тестирование
Тема 1.1.2 Структурное программирование				
2	Технология структурного программирования	32, У1	ПК 1.1	Устный опрос, тестирование
3	Инструментальные средства оформления и документирования алгоритмов	У2, 33	ПК 1.1	Практическая работа
4	Оценка сложности алгоритмов	У3, 34	ПК 1.1	Лабораторная работа
Тема 1.1.3 Объектно-ориентированное программирование				
5	Основные принципы ООП (инкапсуляция, наследование, полиморфизм)	35, У4	ПК 1.2	Устный опрос, тестирование
6	Классы, объекты, конструкторы, перегрузка методов	У5, 36	ПК 1.2	Лабораторная работа
7	Иерархия классов, интерфейсы, наследование	У6, 37	ПК 1.2	Лабораторная работа
8	Коллекции, параметризованные классы (обобщения)	У7, 38	ПК 1.2	Лабораторная работа
Тема 1.1.4 Паттерны проектирования				
9	Назначение и виды паттернов. Порождающие паттерны	У8, 39	ПК 1.2	Устный опрос, тестирование
10	Структурные и поведенческие паттерны	У9, 310	ПК 1.2	Лабораторная работа
Тема 1.1.5 Событийно-управляемое программирование				
11	Событийно-управляемое программирование. Обработчики событий	У10, 311	ПК 1.2	Устный опрос, тестирование
12	Элементы управления, диалоговые окна, графика	У11, 312	ПК 1.2	Лабораторная работа
Тема 1.1.6 Оптимизация и рефакторинг кода				
13	Методы оптимизации программного кода	У12, 313	ПК 1.5	Устный опрос, тестирование
14	Цели и методы рефакторинга	У13, 314	ПК 1.5	Лабораторная работа
Тема 1.1.7 Разработка пользовательского интерфейса				
15	Правила разработки интерфейсов пользователя	У14, 315	ПК 1.2	Устный опрос, тестирование
Тема 1.1.8 Основы ADO.Net				
16	Работа с базами данных. Доступ к данным	У15, 316	ПК 1.2	Лабораторная работа

17	Создание таблиц, запросов, хранимых процедур	У16, 317	ПК 1.2	Лабораторная работа
Курсовое проектирование				
18	Выполнение и защита курсового проекта	У1-У16, 31-317	ПК 1.1, 1.2, 1.5	Курсовой проект

5. ОЦЕНОЧНЫЕ СРЕДСТВА ТЕКУЩЕГО КОНТРОЛЯ ПО МДК

5.1. Типовые задания для оценки знаний и умений

Тема 1.1.1. Жизненный цикл ПО

Вопросы для устного опроса:

1. Что такое жизненный цикл программного обеспечения?
2. Перечислите основные этапы ЖЦ ПО.
3. Какие модели ЖЦ ПО вы знаете (каскадная, итеративная, спиральная)?
4. В чём преимущества и недостатки каскадной модели?
5. Что такое «водопадная» модель?
6. Для каких проектов подходит спиральная модель?
7. Что такое Agile? Назовите основные принципы.
8. Какие фазы включает классический ЖЦ по ГОСТ 19?
9. Что такое техническое задание (ТЗ) и его роль в ЖЦ?
10. Как связаны ЖЦ ПО и управление конфигурацией?

Тестовые задания:

1. **Какой этап ЖЦ ПО следует после этапа проектирования?**
а) Сбор требований
б) Реализация (кодирование)
в) Тестирование
г) Сопровождение
Ответ: б
2. **Какая модель ЖЦ предполагает возврат на предыдущий этап при обнаружении ошибок?**
а) Каскадная
б) Спиральная
в) Итеративная
г) Водопадная
Ответ: в
3. **Что является результатом этапа анализа требований?**
а) Исходный код
б) Спецификация требований (SRS)
в) Тестовые сценарии
г) Инсталляционный пакет
Ответ: б
4. **Какая модель ЖЦ наиболее часто используется в современных гибких методологиях?**
а) Каскадная
б) V-образная
в) Итеративная (Agile)
г) Инкрементная
Ответ: в
5. **Что такое «сопровождение» ПО?**
а) Разработка новой версии
б) Исправление ошибок и добавление новых функций после выпуска
в) Тестирование продукта
г) Написание документации
Ответ: б
6. **Какой документ определяет цели, требования и ограничения проекта?**
а) Техническое задание (ТЗ)
б) Руководство пользователя

- c) Тест-план
- d) Программа работ

Ответ: a

7. **Какая модель ЖЦ характерна для проектов с высокой степенью неопределённости требований?**

- a) Каскадная
- b) Спиральная
- c) V-образная
- d) Инкрементная

Ответ: b

8. **Что такое «верификация» в контексте ЖЦ?**

- a) Подтверждение того, что продукт соответствует требованиям
- b) Оценка стоимости проекта
- c) Управление рисками
- d) Написание кода

Ответ: a

9. **Какая фаза ЖЦ включает установку продукта у заказчика?**

- a) Разработка
- b) Тестирование
- c) Внедрение
- d) Проектирование

Ответ: c

10. **Что такое «Agile-манифест»?**

- a) Документ, описывающий ценности и принципы гибкой разработки
- b) Техническое задание
- c) План тестирования
- d) Руководство по архитектуре

Ответ: a

Тема 1.1.2. Структурное программирование

Вопросы для устного опроса:

1. Какие основные принципы структурного программирования?
2. Какие управляющие конструкции используются в структурном программировании?
3. Что такое «блок-схема»? Какие символы используются?
4. Что такое сложность алгоритма? Виды сложности (временная, ёмкостная).
5. Что такое «О-большое» (Big O notation)?
6. Оцените сложность пузырьковой сортировки в лучшем и худшем случае.
7. Какая сложность у бинарного поиска?
8. Что такое рекурсия? Приведите пример рекурсивного алгоритма.
9. Как оценить сложность рекурсивного алгоритма?
10. Что такое эвристический алгоритм? Пример.

Тестовые задания:

1. **Какая конструкция не относится к структурному программированию?**

- a) Следование
- b) Ветвление
- c) Цикл
- d) Безусловный переход (GOTO)

Ответ: d

2. **Какой сложностью обладает алгоритм пузырьковой сортировки в худшем случае?**

- a) $O(n)$
- b) $O(n \log n)$

с) $O(n^2)$

д) $O(1)$

Ответ: с

3. **Что означает $O(1)$ для алгоритма?**

а) Алгоритм выполняется за константное время

б) Алгоритм не выполняется

с) Алгоритм имеет логарифмическую сложность

д) Алгоритм имеет квадратичную сложность

Ответ: а

4. **Какой метод поиска имеет сложность $O(\log n)$?**

а) Линейный поиск

б) Бинарный поиск

с) Поиск перебором

д) Поиск в ширину

Ответ: б

5. **Что такое рекурсия?**

а) Вызов функцией самой себя

б) Цикл с предусловием

с) Ветвление

д) Последовательное выполнение

Ответ: а

6. **Какой элемент блок-схемы обозначает начало или конец алгоритма?**

а) Прямоугольник

б) Ромб

с) Овал

д) Параллелограмм

Ответ: с

7. **Что такое «базовый алгоритм» в структурном программировании?**

а) Алгоритм, использующий только три управляющие структуры

б) Алгоритм, написанный на ассемблере

с) Алгоритм с рекурсией

д) Алгоритм, содержащий GOTO

Ответ: а

8. **Какой алгоритм сортировки имеет наилучшую асимптотическую сложность в среднем?**

а) Пузырьковая

б) Быстрая сортировка (QuickSort)

с) Сортировка вставками

д) Сортировка выбором

Ответ: б

9. **Что такое «ёмкостная сложность» алгоритма?**

а) Количество операций

б) Объём используемой памяти

с) Время выполнения

д) Количество строк кода

Ответ: б

10. **Какая структура данных используется в рекурсивных алгоритмах?**

а) Массив

б) Стек

с) Очередь

д) Список

Ответ: б

Лабораторные работы:

1. Оценка сложности алгоритмов сортировки (пузырьком, вставками, выбором).
2. Оценка сложности алгоритмов поиска (линейный, бинарный).
3. Оценка сложности рекурсивных алгоритмов (факториал, числа Фибоначчи).
4. Оценка сложности эвристических алгоритмов (например, жадный алгоритм).

Тема 1.1.3. Объектно-ориентированное программирование

Вопросы для устного опроса:

1. Назовите три основных принципа ООП. Раскройте каждый.
2. Что такое класс и объект? Приведите пример.
3. Что такое конструктор? Какие виды конструкторов бывают?
4. Что такое перегрузка методов? Зачем она нужна?
5. Что такое наследование? Как реализовать наследование в C#/Java?
6. Что такое полиморфизм? Виды полиморфизма.
7. Что такое интерфейс? Чем он отличается от абстрактного класса?
8. Что такое инкапсуляция? Как реализуется в языках программирования?
9. Что такое коллекция? Приведите примеры (List, Dictionary).
10. Что такое обобщения (generics)? Зачем они нужны?

Тестовые задания:

1. **Какой принцип ООП означает сокрытие внутренних деталей реализации?**
a) Наследование
b) Полиморфизм
c) Инкапсуляция
d) Абстракция
Ответ: c
2. **Что такое класс?**
a) Экземпляр объекта
b) Шаблон для создания объектов
c) Функция
d) Переменная
Ответ: b
3. **Какой модификатор доступа обеспечивает доступ только внутри класса?**
a) public
b) protected
c) private
d) internal
Ответ: c
4. **Что такое конструктор класса?**
a) Метод, который вызывается при создании объекта
b) Метод, который удаляет объект
c) Поле класса
d) Свойство класса
Ответ: a
5. **Что такое наследование?**
a) Возможность одного класса использовать методы и поля другого
b) Скрытие данных
c) Создание нескольких методов с одним именем
d) Определение интерфейса
Ответ: a
6. **Какой принцип позволяет объектам одного интерфейса вести себя по-разному?**
a) Инкапсуляция

- b) Наследование
- c) Полиморфизм
- d) Агрегация

Ответ: c

7. Что такое интерфейс в ООП?

- a) Абстрактный класс с только абстрактными методами
- b) Конкретный класс
- c) Структура данных
- d) Перечисление

Ответ: a

8. Какой тип коллекции хранит пары «ключ-значение»?

- a) List
- b) Array
- c) Dictionary
- d) Queue

Ответ: c

9. Что такое обобщения (generics)?

- a) Позволяют создавать классы и методы с параметрами типа
- b) Способ перегрузки операторов
- c) Механизм обработки исключений
- d) Тип данных для работы с датами

Ответ: a

10. Что означает ключевое слово base в C#?

- a) Ссылка на текущий объект
- b) Ссылка на базовый класс
- c) Ссылка на интерфейс
- d) Ссылка на статический метод

Ответ: b

Лабораторные работы:

1. Работа с классами: создание класса, полей, методов, конструкторов.
2. Перегрузка методов и конструкторов.
3. Определение операций в классе (перегрузка операторов).
4. Создание наследованных классов (базовый и производный).
5. Работа с объектами через интерфейсы.
6. Использование стандартных интерфейсов (Comparable, Disposable).
7. Работа с типом данных «структура» (struct).
8. Коллекции. Параметризованные классы (List<T>, Dictionary<K,V>).
9. Использование регулярных выражений (Regex).
10. Операции со списками (LINQ).

Тема 1.1.4. Паттерны проектирования

Вопросы для устного опроса:

1. Что такое паттерн проектирования? Зачем они нужны?
2. Какие группы паттернов выделяют (порождающие, структурные, поведенческие)?
3. Приведите пример порождающего паттерна (Singleton, Factory).
4. Для чего используется паттерн «Одиночка» (Singleton)?
5. Что такое паттерн «Фабричный метод»?
6. Приведите пример структурного паттерна (Adapter, Decorator).
7. Что делает паттерн «Адаптер»?
8. Назовите поведенческие паттерны (Strategy, Observer, Command).
9. Для чего используется паттерн «Наблюдатель» (Observer)?
10. Что такое анти-паттерн? Пример.

Тестовые задания:

1. **Какой паттерн гарантирует, что класс имеет только один экземпляр?**
a) Factory
b) Singleton
c) Prototype
d) Builder
Ответ: b
2. **Какая группа паттернов отвечает за создание объектов?**
a) Порождающие
b) Структурные
c) Поведенческие
d) Архитектурные
Ответ: a
3. **Какой паттерн позволяет объектам изменять своё поведение в зависимости от состояния?**
a) Strategy
b) State
c) Command
d) Observer
Ответ: b
4. **Какой паттерн используется для организации взаимодействия объектов «один-ко-многим»?**
a) Mediator
b) Observer
c) Chain of Responsibility
d) Visitor
Ответ: b
5. **Какой паттерн позволяет подключать новые функции к объекту без изменения его кода?**
a) Adapter
b) Decorator
c) Proxy
d) Composite
Ответ: b
6. **Какой паттерн предоставляет интерфейс для создания семейств взаимосвязанных объектов?**
a) Abstract Factory
b) Factory Method
c) Builder
d) Prototype
Ответ: a
7. **Какой паттерн позволяет инкапсулировать запрос как объект?**
a) Strategy
b) Command
c) Template Method
d) Visitor
Ответ: b
8. **Какой паттерн используется для замены алгоритма на лету?**
a) Strategy
b) State
c) Template Method
d) Chain of Responsibility

Ответ: а

9. **Какой паттерн позволяет работать с группой объектов как с одним?**

- a) Composite
- b) Facade
- c) Flyweight
- d) Bridge

Ответ: а

10. **Что такое «анти-паттерн»?**

- a) Часто встречающееся неправильное решение
- b) Лучший способ решения задачи
- c) Паттерн для тестирования
- d) Паттерн для работы с базами данных

Ответ: а

Лабораторные работы:

- 1. Использование основных шаблонов (например, Singleton).
- 2. Использование порождающих шаблонов (Factory Method, Abstract Factory).
- 3. Использование структурных шаблонов (Adapter, Decorator).
- 4. Использование поведенческих шаблонов (Observer, Strategy, Command).

Тема 1.1.5. Событийно-управляемое программирование

Вопросы для устного опроса:

- 1. Что такое событийно-управляемое программирование?
- 2. Что такое событие (event)? Как объявить событие в C#?
- 3. Что такое обработчик события (event handler)?
- 4. Как подписаться на событие и отписаться?
- 5. Что такое делегат в C#? Как он связан с событиями?
- 6. Какие стандартные элементы управления Windows Forms вы знаете?
- 7. Что такое диалоговое окно? Примеры (OpenFileDialog, SaveFileDialog).
- 8. Как создать модальное и немодальное окно?
- 9. Как обработать событие нажатия кнопки?
- 10. Как рисовать графику на форме (GDI+)?

Тестовые задания:

1. **Что такое событие в C#?**

- a) Делегат с ключевым словом event
- b) Переменная
- c) Класс
- d) Интерфейс

Ответ: а

2. **Какой делегат используется для стандартных событий в Windows Forms?**

- a) Action
- b) Func
- c) EventHandler
- d) Predicate

Ответ: с

3. **Какой элемент управления используется для ввода текста?**

- a) Label
- b) TextBox
- c) Button
- d) ComboBox

Ответ: b

4. **Что делает метод Application.Run()?**

- a) Запускает цикл обработки сообщений

- b) Завершает приложение
- c) Создаёт форму
- d) Показывает диалоговое окно

Ответ: a

5. Какой элемент управления предназначен для выбора файла?

- a) OpenFileDialog
- b) SaveFileDialog
- c) FolderBrowserDialog
- d) ColorDialog

Ответ: a

6. Как создать обработчик события для кнопки в Windows Forms?

- a) `button1.Click += new EventHandler(button1_Click);`
- b) `button1.Click = button1_Click;`
- c) `button1.OnClick(button1_Click);`
- d) `button1.AddClick(button1_Click);`

Ответ: a

7. Что такое «делегат» в C#?

- a) Тип, представляющий ссылку на метод
- b) Класс
- c) Структура
- d) Перечисление

Ответ: a

8. Какой метод рисует линию на форме?

- a) DrawLine
- b) DrawRectangle
- c) DrawEllipse
- d) DrawString

Ответ: a

9. Что такое модальное окно?

- a) Окно, которое блокирует родительское окно
- b) Окно, которое не блокирует
- c) Окно без рамки
- d) Окно с прокруткой

Ответ: a

10. Какой объект используется для рисования в Windows Forms?

- a) Graphics
- b) Canvas
- c) Painter
- d) Drawing

Ответ: a

Лабораторные работы:

- 1. Разработка приложения с текстовыми компонентами (TextBox, Label, Button).
- 2. Разработка приложения с несколькими формами (открытие, закрытие).
- 3. Разработка приложения с невидимыми компонентами (Timer, FileSystemWatcher).
- 4. Разработка игрового приложения (движение объекта по таймеру).
- 5. Разработка приложения с анимацией (GDI+).

Тема 1.1.6. Оптимизация и рефакторинг кода

Вопросы для устного опроса:

- 1. Что такое рефакторинг? Назовите цели рефакторинга.
- 2. Какие признаки «плохого кода» (code smells) вы знаете?
- 3. Что такое «дублирование кода»? Как его устранить?

4. Что такое «длинный метод»? Как его рефакторить?
5. Какие методы оптимизации производительности кода вы знаете?
6. Что такое «кэширование»? Приведите пример.
7. Как избежать преждевременной оптимизации?
8. Какие инструменты для статического анализа кода вы знаете?
9. Что такое «профилирование» кода?
10. Как рефакторинг влияет на тестирование?

Тестовые задания:

1. **Что такое рефакторинг?**
 - a) Исправление ошибок
 - b) Изменение внутренней структуры кода без изменения его поведения
 - c) Оптимизация производительности
 - d) Добавление новой функциональности

Ответ: b
2. **Какой признак «плохого кода» означает, что метод слишком длинный?**
 - a) Duplicate code
 - b) Long method
 - c) Large class
 - d) Feature envy

Ответ: b
3. **Какой приём рефакторинга применяется для устранения дублирования кода?**
 - a) Extract Method
 - b) Inline Method
 - c) Move Method
 - d) Rename Method

Ответ: a
4. **Что такое «кэширование»?**
 - a) Сохранение результатов дорогих вычислений для повторного использования
 - b) Удаление неиспользуемых переменных
 - c) Сжатие данных
 - d) Шифрование данных

Ответ: a
5. **Какой инструмент используется для профилирования кода в Visual Studio?**
 - a) Debugger
 - b) Profiler (Performance Profiler)
 - c) IntelliSense
 - d) NuGet

Ответ: b
6. **Что означает «преждевременная оптимизация»?**
 - a) Оптимизация кода до того, как выявлены узкие места
 - b) Оптимизация после тестирования
 - c) Оптимизация компилятором
 - d) Оптимизация базы данных

Ответ: a
7. **Какой метод рефакторинга заменяет временную переменную вызовом метода?**
 - a) Inline Temp
 - b) Replace Temp with Query
 - c) Split Temporary Variable
 - d) Remove Assignments to Parameters

Ответ: b
8. **Что такое «Code Smell»?**
 - a) Ошибка в коде

- b) Поверхностный признак возможной проблемы в коде
- c) Комментарий
- d) Стил ь кода

Ответ: b

9. **Какой метод оптимизации позволяет ускорить цикл?**

- a) Вынесение инвариантных вычислений за цикл
- b) Добавление лишних проверок
- c) Увеличение количества итераций
- d) Использование рекурсии

Ответ: a

10. **Что такое «рефакторинг базы данных»?**

- a) Изменение схемы БД без потери данных
- b) Оптимизация запросов
- c) Индексация
- d) Резервное копирование

Ответ: a

Лабораторные работы:

1. Оптимизация и рефакторинг кода: выявление «плохих запахов», применение Extract Method, Inline Method, Rename Variable и др.

Тема 1.1.7. Разработка пользовательского интерфейса

Вопросы для устного опроса:

1. Какие основные принципы проектирования пользовательского интерфейса?
2. Что такое «юзабилити»? Какие критерии оценки?
3. Какие виды интерфейсов вы знаете (GUI, CLI, TUI)?
4. Что такое «модальное окно»? Когда его использовать?
5. Какие элементы управления для ввода данных вы знаете?
6. Что такое «меню»? Типы меню.
7. Как обеспечить удобство интерфейса для людей с ограниченными возможностями?
8. Что такое «прототипирование» интерфейса?
9. Какие инструменты для дизайна интерфейсов существуют (Figma, Sketch)?
10. Что такое «адаптивный дизайн»?

Тестовые задания:

1. **Какой принцип означает, что интерфейс должен быть простым и понятным?**

- a) Гибкость
- b) Простота (Simplicity)
- c) Эффективность
- d) Безопасность

Ответ: b

2. **Что такое «юзабилити»?**

- a) Удобство и простота использования
- b) Скорость работы
- c) Надёжность
- d) Безопасность

Ответ: a

3. **Какой элемент управления используется для выбора одного варианта из нескольких?**

- a) CheckBox
- b) RadioButton
- c) ComboBox
- d) ListBox

Ответ: b

4. **Что такое «модальное окно»?**

- a) Окно, которое блокирует родительское окно
- b) Окно, которое всегда сверху
- c) Окно без заголовка
- d) Окно с прокруткой

Ответ: a

5. **Какой элемент управления позволяет выбрать файл?**

- a) FolderBrowserDialog
- b) OpenFileDialog
- c) SaveFileDialog
- d) ColorDialog

Ответ: b

6. **Что такое «GUI»?**

- a) Graphical User Interface
- b) Command Line Interface
- c) Text User Interface
- d) Application Programming Interface

Ответ: a

7. **Какой элемент управления отображает статический текст?**

- a) TextBox
- b) Label
- c) Button
- d) Panel

Ответ: b

8. **Что такое «прототипирование» интерфейса?**

- a) Создание интерактивного макета будущего интерфейса
- b) Написание кода
- c) Тестирование интерфейса
- d) Оптимизация

Ответ: a

9. **Какой принцип означает, что интерфейс должен предсказуемо реагировать на действия пользователя?**

- a) Согласованность
- b) Обратная связь
- c) Простота
- d) Эффективность

Ответ: a

10. **Что такое «адаптивный дизайн»?**

- a) Интерфейс, подстраивающийся под размер экрана
- b) Интерфейс для мобильных устройств
- c) Интерфейс с анимацией
- d) Интерфейс с тёмной темой

Ответ: a

Практическое занятие:

- 1. Разработка пользовательского интерфейса (создание формы с элементами управления, настройка свойств, обработка событий).

Тема 1.1.8. Основы [ADO.Net](#)

Вопросы для устного опроса:

- 1. Что такое [ADO.Net](#)? Назначение.
- 2. Какие объекты [ADO.Net](#) используются для подключения к БД (Connection)?
- 3. Что такое Command в [ADO.Net](#)?

4. Как выполнить SQL-запрос на выборку данных (SELECT)?
5. Как выполнить запрос на изменение данных (INSERT, UPDATE, DELETE)?
6. Что такое DataReader? Для чего используется?
7. Что такое DataSet и DataTable? Чем отличается от DataReader?
8. Что такое DataAdapter? Как он связан с DataSet?
9. Как создать хранимую процедуру и вызвать её из C#?
10. Что такое параметризованный запрос? Зачем он нужен?

Тестовые задания:

1. **Какой объект [ADO.Net](#) представляет соединение с базой данных?**
 - a) SqlCommand
 - b) SqlConnection
 - c) SqlDataReader
 - d) SqlDataAdapter

Ответ: b
2. **Какой метод открывает соединение с БД?**
 - a) Open()
 - b) Connect()
 - c) Start()
 - d) Begin()

Ответ: a
3. **Какой объект используется для выполнения SQL-запроса?**
 - a) SqlConnection
 - b) SqlCommand
 - c) SqlDataReader
 - d) SqlDataAdapter

Ответ: b
4. **Какой метод выполняет SQL-запрос и возвращает количество затронутых строк?**
 - a) ExecuteReader()
 - b) ExecuteNonQuery()
 - c) ExecuteScalar()
 - d) ExecuteQuery()

Ответ: b
5. **Что такое DataSet?**
 - a) Набор данных в памяти, включающий таблицы, связи, ограничения
 - b) Односторонний поток данных
 - c) Соединение с БД
 - d) Команда SQL

Ответ: a
6. **Какой объект заполняет DataSet данными из БД?**
 - a) SqlDataReader
 - b) SqlDataAdapter
 - c) SqlCommand
 - d) SqlConnection

Ответ: b
7. **Какой метод SqlDataAdapter заполняет DataSet?**
 - a) Fill()
 - b) Update()
 - c) Select()
 - d) Load()

Ответ: a
8. **Для чего нужны параметризованные запросы?**

- a) Для защиты от SQL-инъекций
- b) Для ускорения запросов
- c) Для упрощения синтаксиса
- d) Для работы с хранимыми процедурами

Ответ: a

9. Какой объект используется для чтения данных в режиме «только вперед»?

- a) SqlDataReader
- b) SqlDataAdapter
- c) DataTable
- d) DataSet

Ответ: a

10. Какой метод SqlCommand используется для получения одного значения?

- a) ExecuteReader()
- b) ExecuteNonQuery()
- c) ExecuteScalar()
- d) ExecuteXmlReader()

Ответ: c

Лабораторные работы:

1. Создание приложения с подключением к БД (SQL Server, SQLite).
2. Создание запросов к БД (SELECT, INSERT, UPDATE, DELETE) с использованием SqlCommand.
3. Создание хранимых процедур и вызов их из приложения.

5.2. Критерии оценивания

5.2.1. Критерии оценивания устного ответа

Оценка	Характеристика ответа
«5»	Ответ правильный, полный, логически выстроен, литературным языком.
«4»	Ответ правильный, полный, но есть отдельные затруднения в формулировке выводов.
«3»	Ответ в основном правильный, но схематичный или с нарушениями последовательности, неполный.
«2»	Непонимание основного содержания, грубые ошибки, отсутствие логики и обобщений.

5.2.2. Критерии оценивания тестовых заданий

Оценка	Процент выполнения
«5»	90–100 %
«4»	70–89,9 %
«3»	50–69,9 %
«2»	0–49,9 %

5.2.3. Критерии оценивания выполнения заданий на лабораторных и практических занятиях

Оценка	Характеристика выполнения
«5»	Работа выполнена полностью и правильно, сделаны верные выводы.
«4»	Работа выполнена правильно с 1–2 несущественными ошибками, исправленными по требованию преподавателя.
«3»	Работа выполнена не менее чем наполовину или допущены 3–4 существенные ошибки.

«2»	Допущено 5 и более существенных ошибок, которые обучающийся не может исправить.
-----	---

5.2.4. Общая классификация ошибок

Грубые ошибки:

- незнание основных понятий, законов;
- неумение выделить главное, обобщить;
- неумение применить знания для решения задач;
- неумение пользоваться справочной литературой;
- нарушение техники безопасности.

Негрубые ошибки:

- неточность формулировок, определений;
- недостаточно продуманный план ответа;
- нерациональные методы работы с литературой.

6. ОЦЕНОЧНЫЕ СРЕДСТВА ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО МДК

Примерные вопросы к экзамену по МДК.01.01

1. Понятие жизненного цикла (ЖЦ) программного обеспечения. Основные этапы ЖЦ.
2. Модели ЖЦ ПО: каскадная (водопадная), её преимущества и недостатки.
3. Итеративные и инкрементные модели ЖЦ. Спиральная модель.
4. Гибкие методологии разработки (Agile). Принципы Agile-манифеста.
5. Роль технического задания (ТЗ) в процессе разработки программных модулей.
6. Основные принципы структурного программирования. Управляющие конструкции (следование, ветвление, цикл).
7. Блок-схемы алгоритмов: основные символы, правила построения.
8. Понятие сложности алгоритма. Временная и ёмкостная сложность.
9. Асимптотическая нотация «О-большое» (Big O). Примеры: $O(1)$, $O(n)$, $O(n^2)$, $O(\log n)$.
10. Оценка сложности алгоритмов сортировки (пузырьковая, быстрая сортировка).
11. Оценка сложности алгоритмов поиска (линейный, бинарный).
12. Рекурсивные алгоритмы: принцип работы, оценка сложности. Примеры (факториал, числа Фибоначчи).
13. Эвристические алгоритмы: определение, область применения.
14. Основные принципы ООП: инкапсуляция, наследование, полиморфизм (определение, примеры).
15. Классы и объекты. Поля, свойства, методы. Конструкторы (по умолчанию, с параметрами, копирования).
16. Модификаторы доступа (public, private, protected, internal). Их назначение.
17. Перегрузка методов и конструкторов. Правила перегрузки.
18. Наследование: базовый и производный классы. Ключевые слова base и this.
19. Полиморфизм: переопределение методов (virtual/override), скрытие методов.
20. Абстрактные классы и абстрактные методы. Интерфейсы. Отличия интерфейса от абстрактного класса.
21. Структуры (struct) и классы (class): сходства и различия.
22. Делегаты в C#: определение, назначение, пример использования.
23. Регулярные выражения: назначение, основные метасимволы, примеры.
24. Коллекции: массивы, списки ($List<T>$), словари ($Dictionary<K, V>$), очереди ($Queue<T>$), стеки ($Stack<T>$).
25. Обобщения (generics): параметризованные классы и методы. Преимущества.
26. Понятие паттерна проектирования. Классификация паттернов (порождающие, структурные, поведенческие).
27. Порождающие паттерны: Singleton (одиночка) – назначение, реализация, недостатки.
28. Порождающие паттерны: Factory Method (фабричный метод) и Abstract Factory.
29. Структурные паттерны: Adapter (адаптер), Decorator (декоратор), Proxy (заместитель).
30. Поведенческие паттерны: Observer (наблюдатель), Strategy (стратегия), Command (команда).
31. Анти-паттерны: определение, примеры (God Object, Spaghetti Code).
32. Понятие событийно-управляемого программирования. Цикл обработки сообщений.
33. События (event) в C#: объявление, подписка, отписка. Связь с делегатами.
34. Стандартный делегат EventHandler. Аргументы событий (EventArgs).
35. Элементы управления Windows Forms: Label, TextBox, Button, ComboBox, ListBox, CheckBox, RadioButton.

36. Диалоговые окна: OpenFileDialog, SaveFileDialog, ColorDialog, FontDialog. Их использование.
37. Обработчики событий: создание, привязка к событию.
38. Графика в Windows Forms: класс Graphics, рисование линий, фигур, текста.
39. Рефакторинг: определение, цели, признаки необходимости (code smells).
40. Основные приёмы рефакторинга: Extract Method, Inline Method, Rename Variable, Move Method, Replace Temp with Query.
41. Оптимизация программного кода: профилирование, выявление узких мест.
42. Кэширование как метод оптимизации. Примеры использования.
43. Инструменты статического анализа кода (StyleCop, SonarQube). Назначение.
44. Принципы проектирования пользовательского интерфейса (простота, согласованность, обратная связь, гибкость).
45. Понятие юзабилити (usability). Критерии оценки удобства интерфейса.
46. Модальные и немодальные окна. Примеры использования.
47. Прототипирование интерфейса: цели, инструменты (Figma, Balsamiq).
48. Адаптивный и отзывчивый дизайн (для веб-интерфейсов).
49. Назначение [ADO.NET](#). Объектная модель: Connection, Command, DataReader, DataAdapter, DataSet.
50. Установка соединения с базой данных (SqlConnection). Строки подключения.
51. Выполнение SQL-команд (SqlCommand): ExecuteNonQuery, ExecuteReader, ExecuteScalar.
52. Параметризованные запросы: защита от SQL-инъекций, синтаксис.
53. DataReader: особенности, чтение данных в режиме «только вперёд».
54. DataSet и DataTable: работа с данными в памяти, заполнение из БД с помощью DataAdapter.
55. Хранимые процедуры: создание в СУБД, вызов из приложения C# с параметрами.
56. Основные этапы разработки программного модуля (от анализа ТЗ до сдачи заказчику).
57. Способы оформления документации на программный модуль (ГОСТ [19.XXX](#), комментарии в коде).
58. Инструментальные средства разработки: IDE (Visual Studio, Rider), системы контроля версий (Git).
59. Роль тестирования в разработке модулей. Виды тестирования (модульное, интеграционное).
60. Профессиональные стандарты и этика программиста

Критерии:

- | | |
|---------------------------|---|
| «5» (отлично) | - Теоретический ответ полный, без ошибок. Практическое задание выполнено полностью, код корректен, программа работает. |
| «4» (хорошо) | - Теоретический ответ с незначительными неточностями. Практическое задание выполнено, но есть небольшие ошибки, не влияющие на общую работоспособность. |
| «3» (удовлетворительно) | - Теоретический ответ неполный, есть ошибки. Практическое задание выполнено частично или с существенными ошибками. |
| «2» (неудовлетворительно) | - Теоретический ответ отсутствует или полностью неверен. Практическое задание не выполнено. |

6.2. Курсовой проект

Примерная тематика курсовых проектов:

1. Разработка программного модуля «Управление библиотекой» (учёт книг, читателей, выдачи/возврата с использованием ООП и [ADO.NET](#)).

2. Разработка модуля учёта заказов для интернет-магазина (классы «Товар», «Клиент», «Заказ», хранение данных в БД SQLite).
3. Разработка приложения «Справочник сотрудников» (добавление, редактирование, удаление, поиск с графическим интерфейсом Windows Forms).
4. Разработка игрового модуля «Змейка» (Windows Forms, обработка событий клавиатуры, таймер, коллекции для сегментов змей).
5. Разработка игрового модуля «Тетрис» (Windows Forms, матричное представление фигур, логика вращения и удаления линий).
6. Разработка модуля «Калькулятор с историей вычислений» (Windows Forms, сохранение истории в файл JSON или XML).
7. Разработка модуля «Личный органайзер» (заметки, напоминания, планировщик задач с использованием таймера и уведомлений).
8. Разработка модуля «Тестирование знаний» (чтение вопросов из файла/БД, подсчёт баллов, вывод результата).
9. Разработка модуля «Ведомость успеваемости студентов» (группы, студенты, оценки, расчёт среднего балла, сортировка с использованием интерфейса IComparable).
10. Разработка модуля «Управление складом» (приход/расход товаров, отчёт о остатках, работа с БД MS SQL Server через [ADO.NET](#)).
11. Разработка модуля «Автоматизация работы автосервиса» (учёт клиентов, заказов, запчастей, формирование счета).
12. Разработка модуля «Библиотека паттернов проектирования» (демонстрация работы Singleton, Factory, Observer, Strategy на простых примерах с GUI).
13. Разработка модуля «Конвертер величин» (длина, масса, валюта с использованием паттерна Strategy для разных алгоритмов конвертации).
14. Разработка модуля «Графический редактор» (рисование линий, прямоугольников, кругов, сохранение изображения в файл с использованием GDI+).
15. Разработка модуля «Чат» (клиент-серверное приложение на сокетах для обмена текстовыми сообщениями в локальной сети).
16. Разработка модуля «Планировщик задач» (фоновое приложение с таймером, выполнение действий по расписанию, логирование).
17. Разработка модуля «Анализатор кода» (статический анализ C#-файлов: подсчёт строк, количество классов/методов, поиск ключевых слов).
18. Разработка модуля «Учёт личных финансов» (категории доходов/расходов, отчёты, диаграммы с использованием Windows Forms и библиотеки Chart).
19. Разработка модуля «Рефакторинг и оптимизация» (взять «плохой» код, применить рефакторинг, измерить производительность до/после, оформить отчёт).
20. Разработка модуля «Многопоточный поиск файлов» (поиск по маске в каталогах с использованием потоков (Thread/Task) и отображением прогресса).

Требования к курсовому проекту:

- Наличие пояснительной записки (введение, постановка задачи, проектирование, описание алгоритмов, листинг кода, тестирование, заключение).
- Реализация программного модуля в соответствии с техническим заданием.
- Использование принципов структурного/объектно-ориентированного программирования.
- Применение паттернов проектирования (по возможности).
- Проведение тестирования и отладки.

- Оформление документации.

Критерии оценивания курсового проекта:

Критерий	Макс. балл
Актуальность и полнота постановки задачи	10
Корректность разработанного алгоритма и его сложность	15
Качество кода (стиль, модульность, использование ООП)	20
Работоспособность программы, отсутствие критических ошибок	20
Оформление пояснительной записки (структура, грамотность, соответствие ГОСТ)	15
Защита проекта (доклад, ответы на вопросы)	20
Итого	100

Шкала перевода в оценку:

- 86–100 баллов – «отлично»
- 71–85 баллов – «хорошо»
- 56–70 баллов – «удовлетворительно»
- 0–55 баллов – «неудовлетворительно»

Студенты, не имеющие положительной оценки за курсовой проект, к экзамену не допускаются.

7. ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ

Основные печатные издания

1. Федорова Г.Н. Разработка модулей программного обеспечения для компьютерных систем: учебник. – М.: Академия, 2020. – 384 с.

Основные электронные издания

1. Федорова Г.Н. Разработка модулей программного обеспечения для компьютерных систем: электронный учебно-методический комплекс. – М.: Академия, 2021. – URL: <https://www.academia-moscow.ru/catalogue/5411/478674/>

Дополнительные источники

1. Гниденко, И. Г. Технология разработки программного обеспечения: учебное пособие для СПО. – М.: Издательство Юрайт, 2021. – 235 с.
2. Белугина С.В. Разработка программных модулей программного обеспечения для компьютерных систем. – СПб: Лань, 2021. – 312 с.

Интернет-ресурсы

- <https://metanit.com> – учебник по C#, .NET
- <https://learn.microsoft.com/ru-ru/dotnet/csharp/> – официальная документация C#
- <https://refactoring.guru/ru> – паттерны и рефакторинг