ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Москва, ул. 2-ая Бауманская, д.5, стр.1
+7 (495) 187-85-85
do@bmstu.ru
do.bmstu.ru

Программа курса «Разработчик на C#»

Номер	Название темы	Количество часов	Описание темы
1	Введение в платформу .Net. Введение в язык программирования C#	5	Основные элементы платформы .Net. Использование интегрированной среды разработки Visual Studio. Определение понятия Сборка и её описание. Особенности языка программирования C#. Создание и работа со сборками в Visual Studio. Работа в командной строке. Языковые лексемы C#. Введение в систему типов языка C#. Пространства имён. Работа со структурными типами и константами. Операции языка C#. Тип данных Nullable. Разработка пользовательских структурных типов. Создание и работа со структурой на C#.
2	Методы и операторы C#	5	Создание и вызов методов. Перегрузка и методы с переменным числом аргументов. Параметры по умолчанию и именованные параметры. Операторы ветвления. Операторы циклов. Написание методов в Visual Studio.
3	Массивы и строки в C#	5	Одномерные массивы. Многомерные массивы класса String. Динамические строки. Регулярные выражения. Работа с аргументами командной строки в Visual Studio.
4	Разработка классов на C#	4	Обзор основных концепций ООП. Объявление класса. Члены класса и создание объектов класса. Модификаторы доступа. Свойства и автоматические свойства. Модификаторы const, readonly и static. Конструкторы и деструкторы в C#. Partial классы. Вложенные классы. Написание классов для приложения Геометрия.
5	Наследование и полиморфизм:Наследование как механизм повторного использования кода	4	Конструктор при наследовании. Преобразование типов и операция is(as). Виртуальные методы и позднее связывание. Абстрактные классы и методы. Модификатор new и сокрытие членов класса. Создание иерархии для приложения Геометрия.
6	Иерархия классов .Net Framework. Обработка ошибок в C#	5	Класс Object и переопределение его методов. Упаковка и распаковка объектов. Метаданные и рефлексия. Другие полезные классы .Net Framework. Концепция исключений. Использование операторов try, catch и finally. Создание своих классов исключений. Контроль за переполнением при целочисленных вычислениях.
7	Интерфейсы в C#. Потоки данных в .Net Framework	5	Концепция интерфейсов. Объявление интерфейса. Реализация интерфейса. Итераторы в .Net Framework. Разработка альтернативных итераторов для вывода массива объектов. Обзор классов потоков.

			Работа с байтовыми потоками. Работа с потоками символов. Сериализация объектов в C#.
8	Коллекции в .Net Framework	5	Обзор классов коллекций. Концепция параметризованных типов данных. Ключевое слово default и ограничения для параметризованных типов. Обзор параметризованных коллекций. Работа с данными в приложении Геометрия.
9	Перегрузка операций в C#	4	Перегрузка унарных операций. Перегрузка бинарных операций. Перегрузка операции индексации. Перегрузка операции преобразования типа. Расширяющие методы для классов и интерфейсов.
10	Делегаты и события в C#. Расширенные возможности C#	4	Создание и работа с делегатами. Классы Action<> и Func<>. Анонимные методы. Лямбда выражения. События. Асинхронный вызов метода. Атрибуты. Сборка мусора и освобождение ресурсов. Работа с динамическими типами. Асинхронное программирование. LINQ. Реализация асинхронного паттерна.
11	Шаблоны функций и классов	4	Шаблоны функций и классов.
12	Расчетно-графическая работа	5	Расчетно-графическая работа.
13	Итоговая аттестация	1	Зачет.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Первый проректор –
проректор по учебной работе
МГТУ им. Н.Э. Баумана
_____ Б.В. Падалкин
«16» апреля 2024 г.

Дополнительное профессиональное образование

ДОПОЛНИТЕЛЬНАЯ ПРОФЕССИОНАЛЬНАЯ ПРОГРАММА
ПРОГРАММА ПОВЫШЕНИЯ КВАЛИФИКАЦИИ
«Программирование на C# (базовый уровень)»

Регистрац. №05.22.13.03.34

Москва, 2024

АВТОРЫ ПРОГРАММЫ:

Преподаватель _____ С.Ю. Камянецкий

СОГЛАСОВАНО:

Начальник УСП _____ Т.А. Гузева

Директор
Центра дополнительного образования _____ М.В. Стоянова

#412

Оглавление

1. ОБЩАЯ ХАРАКТЕРИСТИКА ДПП

Программа подготовлена на основе:

- Федерального закона от 29 декабря 2012 года № 273-ФЗ «Об образовании в Российской Федерации»;
- требований Приказа Минобрнауки России от 01.07.2013 года № 499 «Об утверждении Порядка организации и осуществления образовательной деятельности по дополнительным профессиональным программам»;
- методических рекомендаций-разъяснений Минобрнауки России по разработке дополнительных профессиональных программ на основе профессиональных стандартов от 22 апреля 2015 года № ВК-1030/06.

Реализация программы ДПП направлена на получение новой компетенции, необходимой для профессиональной деятельности.

1.1. Цель ДПП

Сформировать у обучающихся знания, навыки и умения в области разработки, отладки, проверки работоспособности, модификации компьютерного программного обеспечения.

1.2. Планируемые результаты обучения

Планируемые результаты обучения по ДПП:

- освоение профессиональных компетенций в процессе изучения перечисленных тем в учебном плане;
- успешное освоение программы повышения квалификации;
- успешное прохождение итоговой аттестации (зачет).

Обучающимся, успешно прошедшим обучение, выполнившим текущие контрольные задания и выдержавшим предусмотренное учебным планом зачет, выдается удостоверение о повышении квалификации по ДПП «Программирование на C# (базовый уровень)».

1.3. Дополнительные характеристики ДПП

Характеристики новой квалификации определены в приказе Минтруда России от 20.07.2022 №424н «Об утверждении профессионального стандарта «Программист».

Вид профессиональной деятельности:

- Разработка компьютерного программного обеспечения (Код 06.001).

Трудовые функции:

- Проектирование компьютерного программного обеспечения (D/03.6).

1.4. Перечень профессиональных компетенций в рамках имеющейся квалификации, качественное изменение которых осуществляется в результате обучения

Получаемые компетенции базируются на основании Приказа Минобрнауки России от 19 сентября 2017 г. № 929 «Об утверждении федерального государственного образовательного стандарта высшего образования - бакалавриат по направлению подготовки 09.03.01 Информатика и вычислительная техника».

Перечень компетенций:

ОПК-8. Способен разрабатывать алгоритмы и программы, пригодные для практического применения.

1.5. Соответствие видов деятельности профессиональным компетенциям и их составляющих

Профессиональные компетенции	Практический опыт	Умения	Знания
Проектирование компьютерного программного обеспечения (D/03.6)			
ОПК-8. Способен разрабатывать алгоритмы и программы, пригодные для практического применения	Проектирование структур данных; Проектирование баз данных; Проектирование программных интерфейсов	Применять методы и средства проектирования компьютерного программного обеспечения, структур данных, баз данных, программных интерфейсов	Типовые решения, библиотеки программных модулей, шаблоны, классы объектов, используемые при разработке компьютерного программного обеспечения

2. УЧЕБНЫЙ ПЛАН ДПП

2.1. Категория слушателей ДПП

Имеющаяся квалификация (требования к слушателям) – к освоению ДПП допускаются лица, имеющие среднее профессиональное и/или высшее образование.

2.2. Общая трудоёмкость программы, аудиторная и самостоятельная работа

Общая трудоемкость программы 120 академических часов, из них 90 академических часов аудиторной работы, 28 академических часов самостоятельной работы и 2 академических часов итоговой аттестации.

2.3. Форма обучения

Форма обучения по ДПП – очная с применением дистанционных образовательных технологий.

2.4. Учебный план

ДПП «Программирование на C# (базовый уровень)» реализуется одним модулем.

№ п/п	Наименование темы, модуля	Форма контроля	Всего, час	В том числе			
				Лекции	Практ. занятия	Самост. работа	Итоговая аттестация
1.	Знакомство, введение в C# и платформу .NET, настройка окружения	Практ. задание	10	4	2	4	-
2.	Основные концепции языка C#: примитивные типы данных и работа с ними	Практ. задание	10	4	4	2	-
3.	Управление потоком выполнения	Практ. задание	10	4	4	2	-
4.	Методы в C#	Практ. задание	10	4	4	2	-
5.	Работа с массивами	Практ. задание	14	6	4	4	-
6.	Работа с коллекциями	Практ. задание	12	6	4	2	-
7.	Непримитивные типы: кортежи, структуры, записи: наполнение и деконструкция	Практ. задание	16	6	6	4	-
8.	Работа с файлами и исключениями	Практ. задание	12	6	4	2	-
9.	Тонкости работы со строками и датами	Практ. задание	8	2	4	2	-

10.	Ссылочные и значимые типы: введение в классы	Практ. задание	10	4	4	2	-
11.	Отладка приложений	Практ. задание	6	2	2	2	-
12.	Итоговая аттестация	Зачет	2	-	-	-	2
	ИТОГО	-	120	48	42	28	2

3. КАЛЕНДАРНЫЙ УЧЕБНЫЙ ГРАФИК

№ п/п	Наименование темы, модуля	1 день	2 день	3 день	4 день	5 день	6 день	7 день	8 день
1	Знакомство, введение в C# и платформу .NET, настройка окружения								
2	Основные концепции языка C#: примитивные типы данных и работа с ними								
3	Управление потоком выполнения								
4	Методы в C#								
5	Работа с массивами								
6	Работа с коллекциями								
7	Непримитивные типы: кортежи, структуры, записи: наполнение и деконструкция								
8	Работа с файлами и исключениями								
9	Тонкости работы со строками и датами								
10	Ссылочные и значимые типы: введение в классы								
11	Отладка приложений								
12	Итоговая аттестация								

№ п/п	Наименование темы, модуля	9 день	10 день	11 день	12 день	13 день	14 день	15 день
1	Знакомство, введение в C# и платформу .NET, настройка окружения							
2	Основные концепции языка C#: примитивные типы данных и работа с ними							
3	Управление потоком выполнения							
4	Методы в C#							
5	Работа с массивами							
6	Работа с коллекциями							
7	Непримитивные типы: кортежи, структуры, записи: наполнение и деконструкция							
8	Работа с файлами и исключениями							
9	Тонкости работы со строками и датами							
10	Ссылочные и значимые типы: введение в классы							
11	Отладка приложений							
12	Итоговая аттестация							Зачет

Минимальный срок освоения ДПП – 15 дней.

4. РАБОЧАЯ ПРОГРАММА ДПП

4.1. Рабочая программа модуля «Программирование на C# (базовый уровень)»

4.1.1. Цель изучения модуля: сформировать у обучающихся знания, навыки и умения в области разработки, отладки, проверки работоспособности, модификации компьютерного программного обеспечения.

4.1.2. Задачи изучения модуля:

1. Освоение основных концепций программирования: учащиеся познакомятся с фундаментальными принципами программирования, такими как переменные, условия, циклы и функции, что обеспечит им уверенное понимание базовых концепций языка C#.

2. Разработка навыков написания кода: курс сфокусирован на практическом опыте написания программ на C#, что позволит научиться структурировать код, следовать соглашениям о стиле и писать эффективные программы.

3. Работа с основными инструментами и средами разработки: учащиеся изучат основные инструменты и среды разработки, научатся создавать, отлаживать и выполнять программы, что позволит им эффективно использовать средства разработки.

4. Понимание основ работы с данными и структурами: курс включает в себя изучение базовых структур данных и работы с данными в C#, что даст ключевые навыки для эффективной обработки информации в своих программах.

5. Применение знаний на практике через проектную работу: завершающая часть курса предусматривает разработку простого проекта, где учащиеся смогут применить полученные знания, закрепив свои навыки программирования и улучшив понимание процесса разработки приложений на языке C#.

4.1.3. Планируемые результаты обучения

Процесс изучения раздела направлен на формирование следующих компетенций

Код компетенции	Перечень планируемых результатов обучения по модулю	Формы и методы обучения, способствующие формированию и развитию компетенции
ОПК-8	**Знать:** Типовые решения, библиотеки программных модулей, шаблоны, классы объектов, используемые при разработке компьютерного программного обеспечения. **Уметь:** Применять методы и средства проектирования компьютерного программного обеспечения,	Формы обучения: Фронтальная. Методы обучения: Лекция; Практическая работа; Самостоятельная работа.

	структур данных, баз данных, программных интерфейсов. **Владеть:** Проектирование структур данных; Проектирование баз данных; Проектирование программных интерфейсов.	

4.1.4 Содержание курса

Тема 1. Знакомство, введение в C# и платформу .NET, настройка окружения (10 часов)

Лекции (4 часа). В данном разделе рассматривается несколько ключевых вопросов, часто возникающих у новичков в программировании. Различия между C# и .NET, расскажем о среде выполнения приложений .NET и ее отличиях от приложений на языках C и C++. Кроме того, происходит знакомство с базовыми элементами приложений .NET, такими как классы, пространства имен, сборки, а также предоставим информацию о средах разработки, которые можно скачать.

Для того чтобы более глубоко понять язык программирования C#, начинается курс с написания первой программы на этом языке. Такой подход поможет ощутить особенности языка и начать его использование на практике. Также рассматриваются полезные инструменты, такие как ReSharper и другие дополнения, которые упрощают процесс написания кода, делая его более быстрым и эффективным.

В конце раздела проводится краткий обзор основных полученных знаний и предстоящие темы для изучения.

Практические занятия (2 часа). В практической части применяются полученные знания, создается первый проект на C# с использованием выбранной среды разработки. Разбор основных концепций и начало применения C# на практике. Ответы на вопросы.

Самостоятельная работа (4 часа).

Наименование темы	Дидактические единицы, вынесенные на самостоятельное изучение	Формы самостоятельной работы	Учебно-методическое обеспечение	Форма контроля
Знакомство, введение в C# и платформу .NET, настройка окружения	Платформа .NET	Проработка дополнительной литературы	Троелсен Э. C# и платформа. NET. Библиотека программиста: пер. с англ. / Троелсен Э.; пер. Михеев Р. – СПб.: Питер, 2002. – 795 с.	Практ. задание

Тема 2. Основные концепции языка C#: примитивные типы данных и работа с ними (10 часов)

Лекции (4 часа). Изучение основных концепций языка C# и примитивных типов данных является ключевым этапом для тех, кто стремится освоить программирование на данном языке. Примитивные типы данных, такие как целочисленные, вещественные и символьные, являются фундаментальными строительными блоками любой программы. Понимание их особенностей и возможностей позволяет эффективно оперировать данными, создавать более эффективные алгоритмы и обеспечивать надежную работу приложений. Этот раздел не только расширит знания в области программирования, но также предоставит прочный фундамент для более сложных тем, которые будут изучены в дальнейшем. Владение основами C# – важный шаг на пути к разработке качественного и масштабируемого программного обеспечения.

Практические занятия (4 часа). Оптимизация существующего кода, замена объектов на примитивные типы данных и измерение изменения производительности. Такая задача поможет осознать влияние выбора типов данных на эффективность программы.

В завершение, создание программы с форматированным выводом данных или работа с форматированным выводом в консоли подчеркнет важность визуализации и структурирования данных при программировании на C#. Это позволит развить не только навыки работы с типами данных, но и умение представлять информацию в удобном и понятном виде.

Самостоятельная работа (2 часа).

Наименование темы	Дидактические единицы, вынесенные на самостоятельное изучение	Формы самостоятельной работы	Учебно-методическое обеспечение	Форма контроля
Основные концепции языка C#: примитивные типы данных и работа с ними	Типы данных	Проработка дополнительной литературы	Троелсен Э. C# и платформа. NET. Библиотека программиста: пер. с англ. / Троелсен Э.; пер. Михеев Р. – СПб.: Питер, 2002. – 795 с.	Практ. задание

Тема 3. Управление потоком выполнения (10 часов)

Лекции (4 часа). Углубление в изучение структурных элементов программирования, таких как циклы и условные операторы, предоставляя необходимые инструменты для эффективного управления ходом выполнения кода.

Циклы предоставляют мощные возможности для многократного выполнения операций. В этой теме рассматриваются разные виды циклов, такие как цикл for, идеальный для итераций по числовому диапазону, и цикл while, который позволяет гибко управлять выполнением кода в зависимости от условия.

Операторы ветвления, такие как if/else и switch/case, будут подробно исследованы, предоставляя возможность принимать решения на основе различных условий. Эти конструкции помогут эффективно адаптировать код, делая его более гибким и податливым.

Примеры практического использования циклов и условных операторов, такие как обработка данных с повторяющимися операциями и динамичное управление ходом выполнения в зависимости от действий пользователя, помогут лучше усвоить эти ключевые концепции на практике.

Практические занятия (4 часа). В рамках практической работы будет реализована разработка небольшого программного проекта, используя язык C# и углубившись в тему управления потоком управления. Например, создание приложения для учета личных финансов, где можно реализовать циклы для ввода и обработки транзакций, а также условные операторы для анализа и предоставления статистики. Это позволит не только закрепить теоретические знания, но и применить их на практике, разрабатывая полезное и функциональное программное решение.

Самостоятельная работа (2 часа).

Наименование темы	Дидактические единицы, вынесенные на самостоятельное изучение	Формы самостоятельной работы	Учебно-методическое обеспечение	Форма контроля
Управление потоком выполнения	Поток выполнения	Проработка дополнительной литературы	Троелсен Э. C# и платформа. NET. Библиотека программиста: пер. с англ. / Троелсен Э.; пер. Михеев Р. – СПб.: Питер, 2002. – 795 с.	Практ. задание

Тема 4. Методы в C# (10 часов)

Лекции (4 часа). Глубокое погружение в концепцию методологии программирования, осваивая ключевые аспекты создания, вызова и оптимизации методов. Детально рассматриваются ключевые слова, такие как «void», «ref», «out» и др, которые расширяют функциональность методов. Ключевое слово «void» указывает на отсутствие

возвращаемого значения, обеспечивая гибкость в использовании методов для выполнения операций без необходимости возвращать конкретный результат.

Особое внимание уделяется ключевым словам «ref», «out», «in», которые позволяют передавать аргументы методам по ссылке, что открывает новые возможности для модификации значений переменных внутри метода и обмена информацией между методами и вызывающим кодом.

Практические занятия (4 часа). Анализ кодового проекта, с фокусом на проблемах разбиения кода на методы и ключевых ошибках проектирования методов в языке C#. Применяя принципы модульности и читаемости кода, можно оптимизировать структуру программы, выделяя логически связанные блоки кода в отдельные методы. Затем, осуществляя рефакторинг, происходит рассмотрение и устранение ошибок, таких как избыточная сложность методов, неправильное использование ключевых слов, или недостаточная документация. Такой подход позволит не только понять принципы хорошего проектирования, но и научиться избегать распространенных ошибок при создании и оптимизации методов в программировании на C#.

Самостоятельная работа (2 часа).

Наименование темы	Дидактические единицы, вынесенные на самостоятельное изучение	Формы самостоятельной работы	Учебно-методическое обеспечение	Форма контроля
Методы в C#	Методы	Проработка дополнительной литературы	Троелсен Э. C# и платформа. NET. Библиотека программиста: пер. с англ. / Троелсен Э.; пер. Михеев Р. – СПб.: Питер, 2002. – 795 с.	Практ. задание

Тема 5. Работа с массивами (14 часов)

Лекции (6 часов). Массивы представляют собой фундаментальную структуру данных, и в данной теме изучаются различные способы создания, инициализации и манипуляции массивами.

Вначале подробно рассматривается процесс создания массивов с использованием ключевого слова «new». Погружение в синтаксис, который позволяет динамически выделять память под массив во время выполнения программы. Отдельное внимание уделяется различным типам массивов и их инициализации, что позволяет создавать более гибкие и адаптивные структуры данных.

Далее, изучаются основные методы класса Array, который предоставляет широкий спектр функциональности для работы с массивами. Рассматриваются методы для копирования, сортировки и поиска элементов в массиве. Это предоставляет инструменты для эффективного управления данными в массивах и создания более производительного кода. Тема завершается практическими примерами, где слушатели применяют полученные знания для решения конкретных задач с использованием массивов в языке C#.

Практические занятия (4 часа). Во время практической работы слушатель может заняться анализом кодового проекта, фокусируясь на проблемах разбиения кода на методы и ключевых ошибках проектирования методов в языке C#. Применяя принципы модульности и читаемости кода, слушатель может оптимизировать структуру программы, выделяя логически связанные блоки кода в отдельные методы. Затем, осуществляя рефакторинг, слушатель может рассмотреть и устранить ошибки, такие как избыточная сложность методов, неправильное использование ключевых слов, или недостаточная документация. Такой подход позволит не только понять принципы хорошего проектирования, но и научиться избегать распространенных ошибок при создании и оптимизации методов в программировании на C#.

Самостоятельная работа (4 часа).

Наименование темы	Дидактические единицы, вынесенные на самостоятельное изучение	Формы самостоятельной работы	Учебно-методическое обеспечение	Форма контроля
Работа с массивами	Массивы	Проработка дополнительной литературы	Троелсен Э. C# и платформа. NET. Библиотека программиста: пер. с англ. / Троелсен Э.; пер. Михеев Р. – СПб.: Питер, 2002. – 795 с.	Практ. задание

Тема 6. Работа с коллекциями (12 часов)

Лекции (6 часов). Тема включает в себя изучение ключевых коллекций List, Dictionary и HashSet в языке C#, предоставляя более широкий инструментарий для эффективного управления данными. Слушатели углубляют свои знания, изучая, как задавать и использовать словари (Dictionary) для хранения пар «ключ-значение». Работа с хэш-таблицей в основе Dictionary позволяет эффективно осуществлять поиск и доступ к данным.

Дополнительно, тема касается использования HashSet, предоставляя инструмент для работы с уникальными наборами данных. Сравнение HashSet с другими коллекциями,

такими как List, подчеркивает преимущества уникальности элементов в HashSet и его применение в сценариях, где необходимо поддерживать уникальность данных.

Это обширное изучение различных коллекций в теме дает не только понимание базовых принципов работы с List, но и расширенные знания о словарях и множествах, обеспечивая более глубокий взгляд на применение коллекций в реальных проектах.

Практические занятия (4 часа). На практической работе слушатель может сосредоточиться на разработке проекта, в котором используются коллекции в языке C#. Примером может быть система управления библиотекой, где List применяется для хранения списка книг, Dictionary – для отслеживания информации о читателях, и HashSet – для уникальных жанров. Этот проект не только даст возможность углубить практические навыки работы с коллекциями, но и потребует анализа и решения возможных ключевых ошибок, таких как неправильное обращение к элементам коллекций, управление пустыми коллекциями, или некорректное использование методов. Разбор этих ошибок поможет не только избежать потенциальных проблем, но и улучшит общую качественную работу с коллекциями в будущем.

Самостоятельная работа (4 часа).

Наименование темы	Дидактические единицы, вынесенные на самостоятельное изучение	Формы самостоятельной работы	Учебно-методическое обеспечение	Форма контроля
Работа с коллекциями	Коллекции	Проработка дополнительной литературы	Троелсен Э. C# и платформа. NET. Библиотека программиста: пер. с англ. / Троелсен Э.; пер. Михеев Р. – СПб.: Питер, 2002. – 795 с.	Практ. задание

Тема 7. Непримитивные типы: кортежи, структуры, записи: наполнение и деконструкция (16 часов)

Лекции (6 часов). Тема включает в себя изучение ключевых коллекций List, Dictionary и HashSet в языке C#, предоставляя более широкий инструментарий для эффективного управления данными. Слушатели углубляют свои знания, изучая, как задавать и использовать словари (Dictionary) для хранения пар «ключ-значение». Работа с хэш-таблицей в основе Dictionary позволяет эффективно осуществлять поиск и доступ к данным.

Дополнительно, тема касается использования HashSet, предоставляя инструмент для работы с уникальными наборами данных. Сравнение HashSet с другими коллекциями, такими как List, подчеркивает преимущества уникальности элементов в HashSet и его применение в сценариях, где необходимо поддерживать уникальность данных.

Это обширное изучение различных коллекций в теме дает не только понимание базовых принципов работы с List, но и расширенные знания о словарях и множествах, обеспечивая более глубокий взгляд на применение коллекций в реальных проектах.

Практические занятия (6 часов). На практической работе слушатель может сосредоточиться на разработке проекта, в котором используются коллекции в языке C#. Примером может быть система управления библиотекой, где List применяется для хранения списка книг, Dictionary – для отслеживания информации о читателях, и HashSet – для уникальных жанров. Этот проект не только даст возможность углубить практические навыки работы с коллекциями, но и потребует анализа и решения возможных ключевых ошибок, таких как неправильное обращение к элементам коллекций, управление пустыми коллекциями, или некорректное использование методов. Разбор этих ошибок поможет не только избежать потенциальных проблем, но и улучшит общую качественную работу с коллекциями в будущем.

Самостоятельная работа (4 часа).

Наименование темы	Дидактические единицы, вынесенные на самостоятельное изучение	Формы самостоятельной работы	Учебно-методическое обеспечение	Форма контроля
Непримитивные типы: кортежи, структуры, записи: наполнение и деконструкция	Непримитивные типы	Проработка дополнительной литературы	Прайс Марк Дж. C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов. 3-е изд. / Прайс Марк Дж. - Санкт-Петербург: Питер, 2018. - 640 с. - ISBN 978-5-4461-0516-8	Практ. задание

Тема 8. Работа с файлами и исключениями (12 часов)

Лекции (6 часов). Тема файлов и исключений в C# является важной частью программирования, гарантирующей надежность и безопасность работы с внешними ресурсами, в частности, файлами. Для обработки исключений используются ключевые

слова try, catch и finally, которые обеспечивают устойчивость программы к возможным ошибкам.

Блок try содержит код, который может вызвать исключение, например, при попытке чтения данных из файла. Если в блоке try возникает исключение, управление передается блоку catch, где можно предусмотреть логику обработки ошибок. Каждый блок catch предназначен для обработки определенного типа исключения, например, FileNotFoundException, возникающего при попытке открыть несуществующий файл, или IOException, связанного с ошибками ввода/вывода.

Блок finally выполняется независимо от того, произошло исключение или нет. Этот блок обычно используется для освобождения ресурсов, таких как закрытие файла, что делает его полезным даже при возникновении исключений.

При работе с файлами важно адекватно обрабатывать возможные ошибки, такие как отсутствие файла или неверный формат данных. Это обеспечивает стабильность работы приложения и улучшает взаимодействие с пользователем.

Также при работе с массивами ключевыми моментами являются обработка исключений, связанных с выходом за пределы массива (IndexOutOfRangeException) или ошибками, возникающими при доступе к элементу массива. Правильная обработка исключений улучшает устойчивость программы и предотвращает возможные сбои в работе.

Практические занятия (4 часа). Во время практической работы слушатель может заняться разработкой проекта, например, системы управления задачами с использованием непримитивных типов данных в C#. Подход может включать в себя создание структур для представления задач, использование кортежей для описания их статусов, а также записей для удобного хранения информации о приоритетах. Однако, важной частью работы будет разбор ключевых ошибок, таких как неправильное использование кортежей, изменение значений в структурах и некорректная обработка исключений при работе с данными задач. Этот подход позволит не только углубить знания по непримитивным типам данных, но и развивать навыки обработки ошибок для создания более надежных приложений.

Самостоятельная работа (2 часа).

Наименование темы	Дидактические единицы, вынесенные на самостоятельное изучение	Формы самостоятельной работы	Учебно-методическое обеспечение	Форма контроля
Работа с файлами и исключениями	Файлы и исключения	Проработка дополнительной литературы	Прайс Марк Дж. C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов. 3-е изд. / Прайс Марк Дж. - Санкт-Петербург: Питер, 2018. - 640 с. - ISBN 978-5-4461-0516-8	Практ. задание

Тема 9. Тонкости работы со строками и датами (8 часов)

Лекции (2 часа). Данная тема в языке программирования C# включает в себя ряд важных аспектов, необходимых для эффективной обработки текстовой и временной информации.

В области работы со строками, различие между экземплярными и статическими методами класса string становится ключевым. Экземплярные методы, такие как `Substring()` или `Replace()`, применяются к конкретному объекту строки, в то время как статические методы, такие как `String.Format()` или `String.Concat()`, вызываются через сам класс string без создания объекта. Понимание этой разницы помогает более эффективно манипулировать текстовой информацией.

В контексте работы с датами и временем, важным является использование UTC (Coordinated Universal Time) для обеспечения корректности временных операций в различных временных зонах. Для этого используются методы класса `DateTimeOffset`, которые позволяют явно указывать временную зону и обеспечивают более точную и надежную работу с временем в приложении.

Таким образом, освоение тонкостей работы со строками и датами в C# включает в себя умение выбирать и применять подходящие методы для манипуляций с текстовой информацией, а также использование UTC для правильной работы с датами и временем в различных сценариях разработки.

Практические занятия (4 часа). В рамках практической работы слушатель может взяться за разработку проекта, например, системы учета персональных задач с акцентом на эффективное использование строк и дат в языке C#. Подходящими задачами могут быть создание функций для форматирования текста задач, использование различных методов работы со строками для фильтрации или поиска по ключевым словам. При этом важно

предусмотреть обработку ключевых ошибок, таких как попытка доступа к несуществующей задаче или ввод даты в неверном формате. Разбор и обработка этих ошибок помогут улучшить стабильность приложения и повысить качество взаимодействия пользователя с системой управления задачами.

Самостоятельная работа (2 часа).

Наименование темы	Дидактические единицы, вынесенные на самостоятельное изучение	Формы самостоятельной работы	Учебно-методическое обеспечение	Форма контроля
Тонкости работы со строками и датами	Строки и даты	Проработка дополнительной литературы	Прайс Марк Дж. C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов. 3-е изд. / Прайс Марк Дж. - Санкт-Петербург: Питер, 2018. - 640 с. - ISBN 978-5-4461-0516-8	Практ. задание

Тема 10. Ссылочные и значимые типы: введение в классы (10 часов)

Лекции (4 часа). Разбор проблемы разделения бизнес-логики и работы с данными на уровне отдельного приложения, которую решает широко известный архитектурный шаблон Data Access Layer (DAL). Для того, чтобы этот шаблон можно было масштабировать до уровня всего предприятия, необходимо дополнить его рядом архитектурных принципов, которые рассматриваются в данной лекции.

Практические занятия (4 часа). Проектирование слоя доступа к данным (Data access layer). Реализуем паттерн Repository. Знакомство с технологией ORM (entity framework и dapper).

Самостоятельная работа (2 часа).

Наименование темы	Дидактические единицы, вынесенные на самостоятельное изучение	Формы самостоятельной работы	Учебно-методическое обеспечение	Форма контроля
Ссылочные и значимые типы: введение в классы	Классы	Проработка дополнительной литературы	Прайс Марк Дж. C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов. 3-е изд. / Прайс Марк Дж. - Санкт-Петербург: Питер, 2018. - 640 с. - ISBN 978-5-4461-0516-8	Практ. задание

Тема 11. Отладка приложений (6 часов)

Лекции (2 часа). Отладка приложений в языке программирования C# является важным этапом разработки, обеспечивающим выявление и исправление ошибок. При использовании отладчика, программист может устанавливать точки останова (breakpoints), при достижении которых выполнение программы приостанавливается, что позволяет анализировать текущее состояние приложения. Шагом с заходом (step into) и шагом с обходом (step over) позволяют последовательно перемещаться по коду, отслеживая его выполнение.

Отладка рекурсивных функций может быть более сложной задачей, но отладчик предоставляет инструменты для анализа каждого шага в рекурсивном вызове. Это включает в себя просмотр стека вызовов, изменение значений переменных и использование точек останова внутри рекурсивной функции для более детального анализа и исправления ошибок. Пример, работы со стеков вызова.

Практические занятия (2 часа). На практической работе слушатель может заняться созданием консольного приложения для учета личных финансов, используя язык программирования C#. Идеей проекта может быть разработка системы, которая позволяет вводить и отслеживать расходы и доходы, а затем анализировать бюджет. Отладка в этом контексте включает в себя использование точек останова для детального анализа переменных и состояния программы. Разбор ключевых ошибок может включать в себя обработку сценариев, когда вводятся некорректные финансовые данные, например, отрицательные суммы или неверные форматы. Такой проект поможет не только отточить навыки отладки, но и создать полезное консольное приложение для учета финансов.

Самостоятельная работа (2 часа).

Наименование темы	Дидактические единицы, вынесенные на самостоятельное изучение	Формы самостоятельной работы	Учебно-методическое обеспечение	Форма контроля
Отладка приложений	Отладка приложений	Проработка дополнительной литературы	Прайс Марк Дж. C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов. 3-е изд. / Прайс Марк Дж. - Санкт-Петербург: Питер, 2018. - 640 с. - ISBN 978-5-4461-0516-8	Практ. задание

4.1.5. Оценочное средство для текущего контроля (формулировка практических заданий):

Тема 1. Формулировка практического задания:

1. Установите среду разработки Visual Studio (или другую на ваш выбор).
2. Создайте новый проект на C#.
3. Напишите программу для вывода приветствия и запроса имени пользователя.
4. Программа должна ответить приветствием с учетом введенного имени.
5. Убедитесь в успешной компиляции и запуске программы.

Эта задача позволит проверить понимание базовых шагов по созданию и запуску простой программы на C#, а также их умение работать с выводом данных.

Критерии оценивания:

1. Установка среды разработки: Слушатель успешно устанавливает среду разработки, такую как Visual Studio, и готовит ее к созданию проекта на C#.

2. Создание нового проекта: Слушатель успешно создает новый проект на языке C# в выбранной среде разработки.

3. Написание программы: Слушатель разрабатывает программу, используя конструкции C#, которая выводит приветствие и запрашивает имя пользователя.

4. Обработка ввода и вывода: Программа успешно обрабатывает введенные данные и выводит приветствие с учетом введенного имени.

5. Компиляция и запуск программы: Слушатель убеждается в успешной компиляции и запуске программы, проверяя, что программа работает корректно в соответствии с поставленной задачей.

Таким образом, успешное выполнение задачи определяется наличием рабочего окружения, корректной установкой, созданием функционального проекта, написанием работающей программы и успешной компиляцией с последующим запуском.

Тема 2. Формулировка практического задания:

1. Объявите переменные следующих типов данных: int, double, char, bool.

2. Инициализируйте переменные значениями, представляющими различные данные (например, целое число, дробное число, символ, логическое значение).

3. Выполните математические операции, используя объявленные переменные (+, -, *, /) и сохраните результат в новых переменных.

4. Выведите результаты операций на консоль, используя метод Console.WriteLine().

5. Продемонстрируйте использование операторов сравнения (<, >, <=, >=, ==, !=) с примитивными типами данных и выведите результаты на консоль.

6. Работая с bool, используйте логические операторы (&&, ||, !) и также выведите результаты на консоль.

Эта задача позволит проверить знания по объявлению, инициализации и использованию примитивных типов данных в языке C#, а также применить базовые операции и операторы сравнения в практическом контексте.

Критерии оценивания:

1. Объявление переменных: Проверка наличия корректного объявления переменных каждого из указанных примитивных типов данных.

2. Инициализация переменных: Оценка правильности присвоения значений переменным, соответствующим их типам данных.

3. Математические операции: Проверка правильности выполнения математических операций и корректности сохранения результатов.

4. Вывод на консоль: Оценка использования метода `Console.WriteLine()` для вывода результатов на консоль.

5. Операторы сравнения: Проверка корректности применения операторов сравнения и вывод результатов сравнения на консоль.

6. Логические операторы: Оценка правильности использования логических операторов и вывод результатов на консоль.

Оценивание проводится на основе выполнения каждого из указанных шагов задачи. Все шаги должны быть выполнены корректно для положительной оценки.

Тема 3. Формулировка практического задания:

1. Запросите у пользователя ввод числа.

2. Используя условный оператор (if), определите, является ли введенное число положительным, отрицательным или нулем.

3. Если число положительное, выведите сообщение «Число положительное» на консоль.

4. Если число отрицательное, выведите сообщение «Число отрицательное» на консоль.

5. Если число равно нулю, выведите сообщение «Число равно нулю» на консоль.

6. Дополнительно, используя цикл (for, while или do-while), выведите все числа от 1 до введенного числа включительно.

Эта задача поможет понять основные концепции управления потоком выполнения в C#, включая условные операторы и циклы, а также проверит их умение применять эти концепции на практике.

Критерии оценивания:

1. Правильность ввода данных: Успешный ввод числа пользователем.

2. Корректное использование условного оператора (`if`): Правильная проверка на положительное, отрицательное или равное нулю число.

3. Вывод сообщений на консоль: Правильное отображение соответствующего сообщения в зависимости от введенного числа.

4. Использование цикла: Дополнительная проверка на правильное использование цикла для вывода чисел от 1 до введенного числа.

5. Правильность вывода чисел: Если использован цикл, то правильный вывод всех чисел от 1 до введенного числа включительно.

Оценка проводится на основе корректности решения каждого из перечисленных критериев. Все шаги задачи должны быть выполнены правильно для положительной оценки.

Тема 4. Формулировка практического задания:

1. Создайте метод, который принимает два целых числа и возвращает их сумму.

2. Создайте метод, который принимает строку и число, преобразует число в строку и возвращает объединенную строку.

3. Напишите метод, принимающий массив целых чисел и возвращающий среднее арифметическое значений массива.

4. Создайте метод, принимающий три параметра: имя, фамилию и возраст, и возвращающий строку с полной информацией о человеке.

5. Напишите метод, который принимает два числа и возвращает результат деления первого на второе. Предусмотрите обработку ошибки деления на ноль.

Эта задача позволит применить знания о создании и использовании методов в C#, а также проверит их умение решать практические задачи, используя методологию функционального программирования.

Критерии оценивания:

1. Правильность создания методов: Успешное создание всех пяти методов с корректными сигнатурами и возвращаемыми значениями.

2. Корректность логики методов: Проверка на правильность реализации логики каждого метода в соответствии с поставленной задачей.

3. Правильность возврата значений: Оценка правильности возвращаемых значений методов, соответствующих ожидаемым результатам.

4. Обработка ошибок: Проверка наличия и корректности обработки ошибки деления на ноль в соответствующем методе.

5. Тестирование методов: Вызов методов с различными значениями аргументов для проверки их корректной работы.

Оценка проводится на основе выполнения каждого из указанных критериев. Все методы должны быть реализованы правильно и успешно пройти тестирование для положительной оценки.

Тема 5. Формулировка практического задания:

1. Создайте массив целых чисел и заполните его значениями.

2. Напишите метод, который принимает массив и выводит на консоль сумму всех его элементов.

3. Создайте второй массив с числами и объедините его с первым массивом, создав третий массив, содержащий элементы обоих.

4. Напишите метод, который принимает массив и возвращает наибольший элемент.

5. Используя цикл, найдите и выведите на консоль индексы всех четных элементов в созданном третьем массиве.

Эта задача поможет закрепить знания о работе с массивами в C#, использовать методы для обработки массивов и проверить умение применять циклы для выполнения различных задач с массивами.

Критерии оценивания:

1. Создание и заполнение массивов: Успешное создание и заполнение двух массивов целых чисел.

2. Правильность создания методов: Корректное создание двух методов: одного для вывода суммы элементов массива и второго для нахождения наибольшего элемента.

3. Объединение массивов: Правильное создание третьего массива, содержащего элементы обоих массивов.

4. Вывод результатов: Корректный вывод суммы элементов и наибольшего элемента на консоль.

5. Нахождение индексов четных элементов: Использование цикла для нахождения и вывода индексов всех четных элементов в третьем массиве.

Оценка проводится на основе успешного выполнения каждого из указанных критериев. Все шаги задачи должны быть выполнены правильно для положительной оценки.

Тема 6. Формулировка практического задания:

1. Создайте коллекцию List целых чисел и заполните ее значениями.

2. Напишите метод, который принимает коллекцию и выводит на консоль сумму всех ее элементов.

3. Создайте вторую коллекцию List с числами и объедините ее с первой коллекцией, создав третью коллекцию, содержащую элементы обоих списков.

4. Напишите метод, который принимает коллекцию и возвращает наибольший элемент.

5. Используя цикл, найдите и выведите на консоль индексы всех четных элементов в созданной третьей коллекции.

Эта задача поможет закрепить знания о работе с коллекциями в C#, использовать методы для обработки коллекций и проверить умение применять циклы для выполнения различных задач с коллекциями.

Критерии оценивания:

1. Создание и заполнение коллекций: Успешное создание и заполнение двух коллекций List целых чисел.

2. Правильность создания методов: Корректное создание двух методов: одного для вывода суммы элементов коллекции и второго для нахождения наибольшего элемента.

3. Объединение коллекций: Правильное создание третьей коллекции, содержащей элементы обоих списков.

4. Вывод результатов: Корректный вывод суммы элементов и наибольшего элемента на консоль.

5. Нахождение индексов четных элементов: Использование цикла для нахождения и вывода индексов всех четных элементов в третьей коллекции.

Оценка проводится на основе успешного выполнения каждого из указанных критериев. Все шаги задачи должны быть выполнены правильно для положительной оценки.

Тема 7. Формулировка практического задания:

1. Создайте структуру Person, представляющую информацию о человеке: имя (string), возраст (int), и адрес (string).

2. Создайте метод, который принимает на вход объект типа Person и выводит его информацию на консоль.

3. Создайте кортеж, содержащий информацию о трех различных людях.

4. Напишите метод, который принимает кортеж и возвращает средний возраст всех людей из кортежа.

5. Используя деконструкцию, разделите кортеж на отдельные переменные и выведите каждого человека на консоль отдельно.

Эта задача поможет закрепить знания о непримитивных типах данных в C#, включая структуры, кортежи и деконструкцию, а также проверит их умение применять эти концепции на практике.

Критерии оценивания:

1. Создание и заполнение коллекций: Успешное создание и заполнение двух коллекций List целых чисел.

2. Правильность создания методов: Корректное создание двух методов: одного для вывода суммы элементов коллекции и второго для нахождения наибольшего элемента.

3. Объединение коллекций: Правильное создание третьей коллекции, содержащей элементы обоих списков.

4. Вывод результатов: Корректный вывод суммы элементов и наибольшего элемента на консоль.

5. Нахождение индексов четных элементов: Использование цикла для нахождения и вывода индексов всех четных элементов в третьей коллекции.

Оценка проводится на основе успешного выполнения каждого из указанных критериев. Все шаги задачи должны быть выполнены правильно для положительной оценки.

Тема 8. Формулировка практического задания:

1. Создайте текстовый файл и запишите в него несколько строк текста.

2. Напишите метод, который принимает путь к файлу, читает его содержимое и выводит на консоль.

3. Внесите изменения в метод, чтобы он обрабатывал возможные исключения, такие как отсутствие файла или ошибки при чтении.

4. Добавьте функциональность по дописыванию новых строк текста в существующий файл.

5. Реализуйте метод, который принимает путь к несуществующему файлу и создает его, записывая в него заданный текст.

Эта задача позволит применить знания о работе с файлами и исключениями в C#, проверит их умение обрабатывать возможные ошибки при взаимодействии с файловой системой.

Критерии оценивания:

1. Создание и запись в файл: Успешное создание текстового файла и запись в него нескольких строк текста.

2. Правильность метода для чтения и вывода: Корректное создание метода, который принимает путь к файлу, читает его содержимое и выводит на консоль.

3. Обработка исключений при чтении файла: Правильное обращение с исключениями, такими как отсутствие файла или ошибки при чтении.

4. Дописывание новых строк: Внесение изменений в метод для возможности дописывания новых строк в существующий файл.

5. Создание и запись в новый файл: Реализация метода, который принимает путь к несуществующему файлу, создает его и записывает в него заданный текст.

Оценка проводится на основе успешного выполнения каждого из указанных критериев. Все шаги задачи должны быть выполнены правильно для положительной оценки.

Тема 9. Формулировка практического задания:

1. Создайте строку, содержащую ваше имя и фамилию.

2. Напишите метод, который принимает вашу строку и возвращает ее в верхнем регистре.

3. Используя методы работы со строками, разделите ваше имя и фамилию на две отдельные строки.

4. Создайте текущую дату и выведите ее на консоль в формате «ГГГГ-ММ-ДД».

5. Напишите метод, который принимает дату в виде строки и возвращает объект DateTime. Обработайте возможные ошибки ввода.

Эта задача поможет закрепить навыки работы со строками и датами в C#, а также проверит их умение обрабатывать ошибки при преобразовании данных.

Критерии оценивания:

1. Создание строки с именем и фамилией: Успешное создание строки, содержащей имя и фамилию.

2. Правильность метода для преобразования строки в верхний регистр: Корректное создание метода, который принимает строку и возвращает ее в верхнем регистре.

3. Разделение строки на отдельные компоненты: Использование методов работы со строками для разделения строки с именем и фамилией на две отдельные строки.

4. Форматирование текущей даты: Создание текущей даты и вывод ее на консоль в формате «ГГГГ-ММ-ДД».

5. Правильность метода для преобразования строки в DateTime: Корректное создание метода, который принимает дату в виде строки и возвращает объект DateTime. Обработка возможных ошибок ввода.

Оценка проводится на основе успешного выполнения каждого из указанных критериев. Все шаги задачи должны быть выполнены правильно для положительной оценки.

Тема 10. Формулировка практического задания:

1. Создайте класс Book, представляющий книгу, с полями: название, автор и год издания.

2. Напишите конструктор класса, который будет принимать значения для каждого из полей при создании экземпляра.

3. Создайте метод в классе Book, который будет выводить информацию о книге на консоль.

4. Создайте несколько экземпляров класса Book с разными значениями.

5. Используя методы класса, выведите информацию о каждой созданной книге на консоль.

Эта задача поможет закрепить основы работы с классами в C# и оценить их умение создавать и использовать объекты классов.

Критерии оценивания:

1. Создание класса Book: Успешное создание класса Book с тремя полями: название, автор и год издания.

2. Правильность конструктора: Корректное написание конструктора класса, который принимает значения для каждого из полей при создании экземпляра.

3. Метод вывода информации: Создание метода в классе Book, который корректно выводит информацию о книге на консоль.

4. Создание экземпляров класса Book: Успешное создание нескольких экземпляров класса Book с разными значениями.

5. Правильное использование методов: Использование методов класса для вывода информации о каждой созданной книге на консоль.

Оценка проводится на основе успешного выполнения каждого из указанных критериев. Все шаги задачи должны быть выполнены правильно для положительной оценки.

Тема 11. Формулировка практического задания:

1. Напишите простое консольное приложение, в котором есть ошибки, такие как неправильные операции, отсутствующие переменные или некорректная логика.

2. Запустите отладчик, найдите и исправьте ошибки в коде, используя точки останова, шаг с заходом и шаг с обходом.

3. Добавьте в код несколько точек останова и проследите за ходом выполнения программы.

4. Решите проблемы с переменными, чтобы приложение успешно компилировалось и выполнялось без ошибок.

5. Добавьте отладочные выводы, чтобы отслеживать значения переменных в ходе выполнения программы.

Эта задача позволит практиковаться в отладке приложений, использовать инструменты отладки и разбираться с обнаружением и устранением ошибок.

Критерии оценивания:

1. Создание приложения с ошибками: Успешное создание консольного приложения с различными ошибками в коде.

2. Использование отладчика: Корректное использование отладчика для поиска и исправления ошибок, используя точки останова, шаг с заходом и шаг с обходом.

3. Добавление точек останова: Успешное добавление нескольких точек останова в коде и правильное отслеживание хода выполнения программы.

4. Решение проблем с переменными: Исправление проблем с переменными, чтобы приложение успешно компилировалось и выполнялось без ошибок.

5. Отладочные выводы: Добавление отладочных выводов для отслеживания значений переменных в ходе выполнения программы.

Оценка проводится на основе успешного выполнения каждого из указанных критериев. Все шаги задачи должны быть выполнены правильно для положительной оценки.

5. УСЛОВИЯ РЕАЛИЗАЦИИ ДПП

5.1. Организационные условия реализации ДПП

Наименование аудитории	Вид занятия	Наименование оборудования, программного обеспечения
Компьютерный класс/вебинар	Лекции	Windows: - Операционная система: Windows 10 и выше. (можно брать более ранние версии, но с ПО могут быть проблемы) - Программное обеспечение: VS Code + SDK или Visual Studio (Community) – рекомендуемая среда разработки для C# на Windows macOS: - Операционная система: macOS 10.14 и выше. - Программное обеспечение: Visual Studio Code + SDK. Linux: - Операционная система: Различные дистрибутивы Linux, такие как Ubuntu. - Программное обеспечение: Visual Studio Code + SDK
Компьютерный класс/вебинар	Практические занятия	Windows: - Операционная система: Windows 10 и выше. (можно брать более ранние версии, но с ПО могут быть проблемы) - Программное обеспечение: VS Code + SDK или Visual Studio (Community) – рекомендуемая среда разработки для C# на Windows macOS: - Операционная система: macOS 10.14 и выше. - Программное обеспечение: Visual Studio Code + SDK. Linux: - Операционная система: Различные дистрибутивы Linux, такие как Ubuntu. - Программное обеспечение: Visual Studio Code + SDK
Компьютерный класс/вебинар	Самостоятельная работа	Windows: - Операционная система: Windows 10 и выше. (можно брать более ранние версии, но с ПО могут быть проблемы) - Программное обеспечение: VS Code + SDK или Visual Studio (Community) – рекомендуемая среда разработки для C# на Windows macOS: - Операционная система: macOS 10.14 и выше.

		- Программное обеспечение: Visual Studio Code + SDK. Linux: - Операционная система: Различные дистрибутивы Linux, такие как Ubuntu. - Программное обеспечение: Visual Studio Code + SDK
Компьютерный класс/вебинар	Итоговая аттестация	Windows: - Операционная система: Windows 10 и выше. (можно брать более ранние версии, но с ПО могут быть проблемы) - Программное обеспечение: VS Code + SDK или Visual Studio (Community) – рекомендуемая среда разработки для C# на Windows macOS: - Операционная система: macOS 10.14 и выше. - Программное обеспечение: Visual Studio Code + SDK. Linux: - Операционная система: Различные дистрибутивы Linux, такие как Ubuntu. - Программное обеспечение: Visual Studio Code + SDK

5.2. Педагогические условия реализации ДПП

Реализация программы обеспечивается преподавательским составом, удовлетворяющим следующим условиям:

- наличие высшего профессионального образования, соответствующее профилю программы, из числа штатных преподавателей, или привлеченных на условиях почасовой оплаты труда;

- значительный опыт практической деятельности в соответствующей сфере из числа штатных преподавателей или привлеченных на условиях почасовой оплаты труда

5.3. Учебно-методическое обеспечение ДПП

Основная литература:

1. Троелсен Э. C# и платформа. NET. Библиотека программиста: пер. с англ. / Троелсен Э.; пер. Михеев Р. – СПб.: Питер, 2002. – 795 с.
2. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. / Рихтер Дж. – Санкт-Петербург: Питер, 2018. – 896 с.
3. Прайс Марк Дж. C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов. 3-е изд. / Прайс Марк Дж. - Санкт-Петербург: Питер, 2018. - 640 с. - ISBN 978-5-4461-0516-8.

Дополнительная литература:

1. Язык программирования C# 7 и платформы .NET и .NET Core; Автор: Эндрю Троелсен, Филипп Джепикс ; Год: 2019. — 1333 с.

Интернет-источники:

1. https://learn.microsoft.com/ru-ru/dotnet/csharp/.
2. https://github.com/iksergey/dotnet-csharp-basic.
3. https://github.com/orgs/dotnet/repositories.
4. http://referencesource.microsoft.com.
5. http://sharplab.io.
6. https://dotnet.microsoft.com/en-us/download.
7. https://code.visualstudio.com.
8. https://github.com/dotnet/roslyn.

5.4. Методические рекомендации

ДПП построена по тематическому принципу, каждый раздел представляет собой логически завершенный материал.

Преподавание программы основано на личностно-ориентированной технологии образования, сочетающей два равноправных аспекта этого процесса: обучение и учение. Личностно-ориентированный подход развивается при участии слушателей в активной работе на практических занятиях. Личностно-ориентированный подход направлен, в первую очередь, на развитие индивидуальных способностей обучающихся, создание условий для развития творческой активности слушателя и разработке инновационных идей, а также на развитие самостоятельности мышления при решении учебных задач разными способами, нахождение рационального варианта решения, сравнения и оценки нескольких вариантов их решения и т.п. Это способствует формированию приемов умственной деятельности по восприятию новой информации, ее запоминанию и осознанию, созданию образов для сложных понятий и процессов, приобретению навыков поиска решений в условиях неопределенности.

Практические занятия проводятся для приобретения навыков решения практических задач в предметной области модуля. Задания, выполняемые на практических занятиях, выполняются с использованием активных и интерактивных методов обучения.

Самостоятельная работа слушателей предназначена для проработки дополнительной литературы. Результаты практических заданий слушателей учитываются на итоговой аттестации.

При изучении курса предусмотрены следующие методы организации и осуществления учебно-познавательной деятельности:

- объяснительно-иллюстративный метод;

- репродуктивный метод;
- частично-поисковый метод.

6. ФОРМЫ ИТОГОВОЙ АТТЕСТАЦИИ ДПП

Итоговая аттестация проводится в форме зачета для проверки сформированности компетенций, полученных в рамках ДПП.

Зачет проводится в формате тестирования и ответов на вопросы билета. Результатом зачета служат правильные ответы на вопросы билета.

По результатам итоговой аттестации обучающемуся выставляется оценка «ЗАЧТЕНО/НЕ ЗАЧТЕНО»:

Оценка «ЗАЧТЕНО» выставляется обучающемуся, который:

- правильно ответил не менее, чем на 60% вопросов теста;
- правильно ответил на вопрос билета;
- продемонстрировал необходимые систематизированные знания и достаточную степень владения принципами предметной области программы, понимание их особенностей и взаимосвязь между ними в течение всего срока обучения по ДПП.

Оценка «НЕ ЗАЧТЕНО» ставятся обучающемуся, который:

- правильно ответил менее, чем на 60% вопросов теста;
- неправильно ответил на вопрос билета;
- имеет крайне слабые теоретические и практические знания, обнаруживает неспособность к построению самостоятельных заключений.

7. ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ИТОГОВОЙ АТТЕСТАЦИИ

7.1. Паспорт комплекта оценочных средств

Предметы оценивания	Объекты оценивания	Показатели оценки
ОПК-8. Способен разрабатывать алгоритмы и программы, пригодные для практического применения	Ответы на вопросы	Правильные ответы на вопросы теста

7.2. Комплект оценочных средств

7.2.1. Темы для подготовки к зачету:

1. C#, .NET, платформа, окружение.
2. Примитивные типы данных, переменные, константы, операторы, преобразование типов.
3. Условные операторы, циклы, ветвления, операторы break и continue.
4. Переменные и выражения.
5. Методы, параметры, возвращаемые значения, перегрузка методов.
6. Массивы, индексы, инициализация массивов, многомерные массивы.
7. Коллекции, List, ArrayList, различия между List и ArrayList.
8. Кортежи, структуры, записи, деконструкция.
9. Работа с файлами, чтение, запись, исключения, ключевые слова try, catch, finally.
10. Работа со строками, класс string, методы форматирования строк, работа с датами.
11. Ссылочные типы, значимые типы, классы, объекты, конструкторы, методы, свойства.
12. Отладка, отладчик, точки останова, шаг с заходом, шаг с обходом.

7.2.2. Вопросы теста для проведения зачета:

1. Какой компонент .NET отвечает за выполнение CIL-кода?

a. Just-In-Time Compiler (JIT)

b. Common Language Runtime (CLR)

c. Intermediate Language Compiler (ILC)

d. .NET Compiler

2. Какие из перечисленных шагов необходимы для создания простой программы на C#?

a. Отладка кода

b. Написание кода

c. Компиляция кода

d. Все вышеперечисленное

3. Какие операционные системы поддерживают разработку на C# и использование .NET?

a. Windows

b. Linux

c. MacOS

d. Все вышеперечисленные

4. Какой из перечисленных типов данных является примитивным в C#?

a. array

b. string

c. int

d. class

5. Какие из следующих операторов используются для сравнения значений в C#?

a. + и -

b. == и !=

c. * и /

d. && и ||

6. Что такое переменная в контексте программирования на C#?

a. Элемент массива

b. Объект класса

c. Именованное хранилище данных

d. Метка ветвления

7. Какие из перечисленных конструкций используются для управления потоком выполнения в C#?

a. for и foreach

b. switch и case

c. if и if-else

d. все вышеперечисленные

8. Какой оператор позволяет выполнить блок кода, если условие истинно, иначе выполнить другой блок кода?

a. if

b. switch

c. else

d. for

9. Какой из операторов предоставляет более компактный синтаксис для условного оператора, возвращая одно из двух значений в зависимости от условия?

a. while

b. for

c. ternary (?:)

d. do-while

10. Какова основная цель использования методов в C#?

a. Обеспечение визуального интерфейса

b. Группировка данных в массивы

c. Повторное использование кода

d. Создание новых переменных

11. Что такое сигнатура метода?

a. Набор аргументов метода

b. Имя метода

c. Тип возвращаемого значения

d. Тело метода

12. Вопрос: Какие из перечисленных операторов используются для создания метода в C#?

a. def

b. function

c. void

d. return

13. Как создать массив в C#?

a. int array = new int[];

b. int array = [1, 2, 3];

c. int[] array = new int[3];

d. array<int> = {1, 2, 3};

14. Как получить длину массива в C#?

a. array.Length

b. array.Size

c. array.Count

d. array.Index

15. Как выполнить инициализацию массива значениями при его создании?

a. int[] arr = new int[3];

b. int[] arr = {1, 2, 3};

c. arr[] = {1, 2, 3};

d. int arr = [1, 2, 3];

16. Какое ключевое слово используется для объявления списка в C#?

a. List

b. Array

c. Collection

d. Set

17. Что обеспечивает использование коллекций в C#?

a. Эффективность работы с данными

b. Только удобство синтаксиса

c. Возможность создания графических интерфейсов

d. Работу с базами данных

18. Как удалить элемент из списка в C#?

a. list.DeleteAt(index);

b. list.Delete(index);

c. list.RemoveAt(index);

d. list.Erase(index);

19. Какой из следующих типов в C# является структурой?

a. Array

b. List

c. Tuple

d. struct

20. Как создать кортеж в C#?

a. tuple<int, string> myTuple = (1, "Hello");

b. tuple myTuple = (1, "Hello");

c. (int, string) myTuple = (1, "Hello");

d. Tuple<int, string> myTuple = (1, "Hello");

21. Что такое деконструкция кортежа в C#?

a. Преобразование кортежа в массив

b. Извлечение значений из кортежа в отдельные переменные

c. Удаление элементов из кортежа

d. Создание кортежа из двух других кортежей

22. Какой из следующих операторов используется для обработки исключений в C#?

a. try

b. handle

c. catch

d. exception

23. Как открыть файл для чтения в C#?

a. FileStream file = new FileStream("file.txt", FileMode.Open);

b. StreamReader reader = new StreamReader("file.txt");

c. File.OpenRead("file.txt");

d. File.Read("file.txt");

24. Что представляет собой блок finally в конструкции try-catch-finally?

a. Блок, выполняемый в случае успешного выполнения try

b. Блок, выполняемый при возникновении исключения

c. Блок, который всегда выполняется, независимо от исключений

d. Блок, выполняемый после завершения программы

25. Какая из следующих операций выполняет конкатенацию строк в C#?

a. str1 + str2

b. str1 .concat(str2)

c. concatenate(str1, str2)

d. concat(str1, str2)

26. Как представить текущую дату и время в C#?

a. DateTime.Now()

b. System.Time.Now

c. DateTime.Current()

d. DateTime.Now

27. Какие из следующих методов используются для форматирования строки, содержащей дату и время?

a. ToString()

b. Format()

c. DateToString()

d. TimeFormat()

28. Что такое ссылочный тип в C#?

a. Тип, значение которого хранится непосредственно в памяти

b. Тип, значение которого хранится в стеке

c. Тип, который ссылается на объект в куче

d. Тип, используемый только для числовых значений

29. Какие из следующих элементов используются при работе с классами в C#?

a. Конструкторы

b. Индексы

c. Статические методы

d. Все вышеперечисленные

30. Что такое конструктор в C#?

a. Метод, возвращающий значение

b. Метод, выполняющий деконструкцию объекта

c. Специальный метод, вызываемый при создании объекта

d. Метод, используемый для сравнения объектов

31. Какой из следующих инструментов отладки используется для временной остановки выполнения программы в определенной точке?

a. Breakpoint

b. Stop

c. Halt

d. Pause

32. Какие из следующих операций поддерживаются в процессе отладки в C#?

a. Точки останова (breakpoints)

b. Шаг с заходом (step into)

c. Шаг с обходом (step over)

d. Все вышеперечисленные

33. Какой инструмент отладки позволяет просматривать стек вызовов в момент времени?

a. Stack Viewer

b. Call Stack

c. Debug Watch

d. Stack Inspector

7.2.3. Вопросы для проведения зачета:

1. Каким образом можно объединить несколько строк в C# без использования оператора конкатенации `+`?
2. Как создать и инициализировать одномерный массив в C#?
3. Какой оператор в C# используется для принятия решений на основе условий?
4. Как в C# обработать исключение, возникающее при чтении файла, и предотвратить завершение программы из-за ошибки?
5. В чем основное различие между ссылочными и значимыми типами в C#?
6. Какие методы класса String в C# можно использовать для преобразования строки в верхний и нижний регистры?