Приложение 4

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Нижегородский государственный университет
им. Н.И. Лобачевского»

Арзамасский филиал ННГУ - Факультет естественных и математических наук

УТВЕРЖДЕНО

решением президиума Ученого совета ННГУ

протокол № 1 от 16.01.2024 г.

Рабочая программа дисциплины

Разработка интерфейса с использованием языка C#

Уровень высшего образования
Бакалавриат

Направление подготовки / специальность
09.03.03 - Прикладная информатика

Направленность образовательной программы
Системное и прикладное программирование

Форма обучения
очная, очно-заочная

г. Арзамас

2024 год начала подготовки

1. Место дисциплины в структуре ОПОП

Дисциплина Б1.В.ДВ.01.02 Разработка интерфейса с использованием языка C# относится к части, формируемой участниками образовательных отношений образовательной программы.

2. Планируемые результаты обучения по дисциплине, соотнесенные с планируемыми результатами освоения образовательной программы (компетенциями и индикаторами достижения компетенций)

Формируемые компетенции (код, содержание компетенции)	**Планируемые результаты обучения по дисциплине (модулю), в соответствии с индикатором достижения компетенции**		**Наименование оценочного средства**	
	Индикатор достижения компетенции (код, содержание индикатора)	**Результаты обучения по дисциплине**	**Для текущего контроля успеваемости**	**Для промежуточной аттестации**
ПК-11: Способен осуществлять модульное и интеграционное тестирование ИС (ИИС), устранять (по мере возможности) обнаруженные несоответствия	*ПК-11.1: Демонстрирует знание методологических основ модульного и интеграционного тестирования ИС (ИИС).* *ПК-11.2: Демонстрирует умение осуществлять модульное и интеграционное тестирование ИС (ИИС) и устранять (по мере возможности) обнаруженные несоответствия.* *ПК-11.3: Имеет практический опыт модульного и интеграционного тестирования конкретной ИС (ИИС).*	*ПК-11.1:* *Знать технологии разработки алгоритмов и программ, методы отладки и решения задач на ЭВМ в различных режимах, основы объектно-ориентированного подхода к программированию, системы программирования на языке высокого уровня, технологии процесса подготовки и решения задач на ПЭВМ* *Уметь создавать консольные и оконные (GUI) приложения на C#, работать с базами данных, используя C#, работать с файлами и каталогами, разрабатывать и отлаживать апплеты для web-страниц реализую вопросы формализации решения прикладных задач* *Владеть навыками создавать консольные и оконные (GUI) приложения на C#, работать с базами данных, используя C#, работать с файлами и каталогами* *ПК-11.2:* *Знать основные приемы алгоритмизации и программирования на языке высокого уровня, принципы разработки программ, принципы автономной*	*Задания* *Контрольная работа* *Практическое задание* *Реферат* *Тест*	*Зачёт:* *Контрольные вопросы*

		отладки программ *Уметь создавать web-сервисы и J2EE-приложения; интегрировать web-приложения с внешними системами; конструировать интерактивные порталы для доступа к данным, процессам и приложениям на основе использования системного подхода в формализации решения прикладных задач.* *Владеть навыками разрабатывать и отлаживать апплеты для web-страниц реализую вопросы формализации решения прикладных задач, создавать web-сервисы и J2EE-приложения* *ПК-11.3:* *Знать основные приемы алгоритмизации и программирования на языке высокого уровня, принципы разработки программ, принципы автономной отладки программ* *Уметь создавать web-сервисы и J2EE-приложения; интегрировать web-приложения с внешними системами; конструировать интерактивные порталы для доступа к данным, процессам и приложениям на основе использования системного подхода в формализации решения прикладных задач.* *Владеть навыками разрабатывать и отлаживать апплеты для web-страниц реализую вопросы формализации решения прикладных задач, создавать web-сервисы и J2EE-приложения*		
ПК-8: Способен разрабатывать лингвистическое, информационное и программное	*ПК-8.1: Демонстрирует знание современных языков и систем программирования, формализмов описания*	*ПК-8.1:* *Знать современное состояние и принципиальные возможности языка*	*Задания* *Контрольная работа* *Практическое*	*Зачёт: Контрольные вопросы*

обеспечение ИС (ИИС) и сопровождающую его документацию	*знаний на концептуальном и инфологическом уровнях, требований к технической документации на все виды обеспечения ИС (ИИС).* *ПК-8.2: Применяет современные языки и системы программирования, формализмы описания знаний на концептуальном и инфологическом уровнях при разработке лингвистического, информационного и программного обеспечения ИИС и сопровождающей его документации.* *ПК-8.3: Имеет практический опыт разработки лингвистического, информационного и программного обеспечения конкретной ИС (ИИС) и сопровождающей ее документации.*	*программирования C# и использующих его систем программирования;* *Уметь ставить задачи и разрабатывать алгоритм их решения, используя C#, разрабатывать основные программные документы; работать с современными системами программирования, включая объектно-ориентированные.* *Владеть навыками разработки и отладки программ на C#, основными шаблонами проектирования программных систем с использованием технологии C#,* *ПК-8.2:* *Знать возможности языка программирования C# для проведения анализа социально-экономических задач и процессов с применением методов системного анализа и математического моделирования.* *Уметь инсталлировать, тестировать, испытывать и использовать программные средства C#,* *Владеть приемами разработки прикладных программ на языке C#.* *ПК-8.3:* *Знать особенности осуществления разработки лингвистического, информационного и программного обеспечения конкретной ИС* *Уметь разрабатывать программное обеспечение ИС и сопровождающую его документацию* *Владеть способностью осуществлять разработку лингвистического, информационного и программного обеспечения*	*задание* *Реферат* *Тест*	

		конкретной ИС (ИИС) и сопровождающей его документации.		

3. Структура и содержание дисциплины

3.1 Трудоемкость дисциплины

	очная	**очно-заочная**
Общая трудоемкость, з.е.	**3**	**3**
Часов по учебному плану	**108**	**108**
в том числе		
аудиторные занятия (контактная работа):		
- занятия лекционного типа	**0**	**0**
- занятия семинарского типа (практические занятия / лабораторные работы)	**36**	**8**
- КСР	**1**	**1**
самостоятельная работа	**71**	**99**
Промежуточная аттестация	**0** **Зачёт**	**0** **Зачёт**

3.2. Содержание дисциплины

(структурированное по темам (разделам) с указанием отведенного на них количества академических часов и виды учебных занятий)

Наименование разделов и тем дисциплины	Всего (часы)		в том числе							
			Контактная работа (работа во взаимодействии с преподавателем), часы из них						Самостоятельная работа обучающегося, часы	
			Занятия лекционного типа		Занятия семинарского типа (практические занятия/лабораторные работы), часы		Всего			
	очно	очно-заочно	очно	очно-заочно	очно	очно-заочно	очно	очно-заочно	очно	очно-заочно
Тема 1. Основы программирования на языке C#. Синтаксис языка C#	17	18			6	1	6	1	11	17
Тема 2. Графические интерфейсы пользователя	18	18			6	2	6	2	12	16
Тема 3. Введение в .NET Framework	18	18			6	1	6	1	12	17
Тема 4. Классы и структуры. Методы.	18	18			6	2	6	2	12	16
Тема 5. Принципы разработки пользовательского интерфейса	18	17			6	1	6	1	12	16
Тема 6. Обзор передовых технологий языка C#	18	18			6	1	6	1	12	17
Аттестация	0	0								

КСР	1	1					1	1		
Итого	108	108	0	0	36	8	37	9	71	99

Содержание разделов и тем дисциплины

Тема 1. Основы программирования на языке C#. Синтаксис языка C#
История создания и развития языка C#. Основные понятия и синтаксические конструкции языка: переменные, типы данных, операторы, управляющие конструкции, классы и методы. Основы работы с интегрированной средой разработки Visual Studio и создание простых консольных приложений. Компиляция и выполнение программ на языке C#.
Тема 2. Графические интерфейсы пользователя
Что такое графический интерфейс пользователя (GUI)? Какие элементы управления есть в графических интерфейсах пользователя? Как создать графический интерфейс в C#? Модель-представление-контроллер (MVC) и как она используется в разработке графических интерфейсов. Технологии Windows Forms.
Тема 3. Введение в .NET Framework
История создания .NET Framework. Основные концепции и принципы работы .NET Framework. Версии .NET Framework и их особенности. Компоненты .NET Framework: библиотеки классов, среда исполнения, службы. Создание проекта C#. Структура программы C#. Базовые типы данных в C#. Переменные в C#. Операторы в C#. Управляющие конструкции в C#.
Тема 4. Классы и структуры. Методы.
Определение классов и структур в C#. Члены класса и их доступность (public, private, protected, internal). Статические члены класса. Конструкторы и деструкторы класса. Свойства и индексаторы класса. Методы класса (включая статические и виртуальные). Наследование классов и интерфейсов (одиночное и множественное наследование). Полиморфизм в C#. Абстрактные классы и интерфейсы. Ограниченная видимость членов класса (private, protected). Приведение типов и проверка на null. Инициализация объектов. Копирование объектов. Структуры в C#: определение, использование, отличия от классов. Перечисления в C#.
Тема 5. Принципы разработки пользовательского интерфейса
Введение в паттерн проектирования Model-View-Controller (MVC). Разделение представления и бизнес-логики с использованием MVC. Реализация Model в MVC. Реализация View в MVC. Реализация Controller в MVC. Преимущества использования паттерна MVC для разработки приложений. Недостатки использования паттерна MVC и способы их устранения. Пример использования паттерна MVC в реальном проекте. Рекомендации по использованию паттерна MVC при разработке приложений. Отличия паттерна MVC от других паттернов проектирования.
Тема 6. Обзор передовых технологий языка C#
Что такое LINQ и как он используется в C#?
Какие основные методы и операторы доступны в LINQ?
Как использовать параллельное программирование в C# для повышения производительности?
Какие существуют подходы к организации параллельных вычислений в C#?
Как обрабатывать исключения в параллельных вычислениях в C#?
Как синхронизировать доступ к общим ресурсам в параллельных программах на C#

Практические занятия /лабораторные работы организуются, в том числе, в форме практической подготовки, которая предусматривает участие обучающихся в выполнении отдельных элементов работ, связанных с будущей профессиональной деятельностью.
На проведение практических занятий / лабораторных работ в форме практической подготовки отводится: очная форма обучения - 6 ч., очно-заочная форма обучения - 6 ч.

4. Учебно-методическое обеспечение самостоятельной работы обучающихся

Самостоятельная работа обучающихся включает в себя подготовку к контрольным вопросам и заданиям для текущего контроля и промежуточной аттестации по итогам освоения дисциплины приведенным в п. 5.

Учебно-методические документы, регламентирующие самостоятельную работу
адреса доступа к документам:
https://arz.unn.ru/sveden/document/
https://arz.unn.ru/pdf/Metod_all_all.pdf

5. Фонд оценочных средств для текущего контроля успеваемости и промежуточной аттестации по дисциплине (модулю)

5.1 Типовые задания, необходимые для оценки результатов обучения при проведении текущего контроля успеваемости с указанием критериев их оценивания:

5.1.1 Типовые задания (оценочное средство - Задания) для оценки сформированности компетенции ПК-11:

1. Создание пользовательского интерфейса.
2. Принципы разработки пользовательского интерфейса.
3. Работа с формами.
4. Применение элементов управления и компонентов.
5. Меню.
6. Проверка данных, вводимых пользователем.
7. Приложение Virtual Doughnut Factory.

5.1.2 Типовые задания (оценочное средство - Задания) для оценки сформированности компетенции ПК-8:

Приведите описание основных понятий, утверждений (с доказательствами), моделей и формул следующих разделов дисциплины

1. Инфраструктура .NET Framework и общеязыковая исполняющая среда.
2. Библиотека базовых классов .NET.
3. Классы и структуры.
4. Методы.
5. Область видимости и уровни доступа.
6. Сбор мусора.
7. Применение классов и демонстрация сбора мусора.

Критерии оценивания (оценочное средство - Задания)

Оценка	Критерии оценивания
отлично	Ответ полный и правильный на основании изученной теории; материал изложен в необходимой логической последовательности, грамотный научный язык; ответ самостоятельный.
хорошо	Ответ полный и правильный на основании изученной теории; материал изложен в необходимой логической последовательности при этом допущены

Оценка	Критерии оценивания
	две-три несущественные ошибки, исправленные по требованию преподавателя.
удовлетворительно	Ответ полный, но при этом допущена существенная ошибка или неполный, несвязный ответ.
неудовлетворительно	Ответ обнаруживает непонимание студентом основного содержания учебного материала или допущены существенные ошибки, которые не могут быть исправлены при наводящих вопросах преподавателя.

5.1.3 Типовые задания (оценочное средство - Контрольная работа) для оценки сформированности компетенции ПК-11:

Задание №4

Дана строка, состоящая из английских слов, разделенных одним или несколькими пробелами. В начале и в конце строки также могут быть пробелы

1. Вывести все слова - каждое на отдельной строке

```
var r = new Regex(@"\w+");
  foreach (Match s in r.Matches(" jh kj hjk k "))
Console.WriteLine(s);
```

2. Вывести все слова, начинающиеся и заканчивающиеся на 'a'

3. Вывести все слова, содержащие в начале от 2 до 4 согласных букв

4. Вывести все числа с десятичной точкой и вычислить их сумму

5. Вывести все полные имена файлов (и только их). Полные имена файлов задаются в формате Диск:\папка1\папка2\имя.расширение. Диск и папки могут отсутствовать.

6. Считать многострочную строку из файла слов (слова могут разделяться одним или несколькими пробелами и символами перехода на новую строку), используя вызов

```
var s = System.IO.File.ReadAllText('Text.txt');
```

Создать регулярное выражение с опцией RegexOptions.Multiline.

а) Вывести все слова, стоящие в начале строки.

б) Вывести все слова, стоящие в конце строки.

в) Вывести все слова стоящие первыми в строке (перед словами могут быть один или несколько пробелов)

г) Вывести все строки, начинающиеся с пробела

д) Вывести номера символов, с которых начинается каждая строка

Задание №5

Для выполнения заданий следует подключить к основной сборке следующие сборки: System.Runtime.Serialization System.Runtime.Serialization.Formatters.Soap

1. Используя BinaryFormatter и атрибут [Serializable], сериализовать на диск в бинарном формате объекты классов Student и Teacher, после чего десериализовать. Убедиться, что сериализация работает корректно (для этого десериализовать и вывести на экран повторно). Обратить особое внимание, что класс Teacher содержит поле List<Student>. Попробовать

провести сериализацию, используя SoapFormatter. Что не работает? Как это можно исправить (какое поле убрать/не сериализовать)?

2. Используя SoapFormatter и атрибут [Serializable], сериализовать на диск в XML формате связный список из нескольких

```
class Node<T>
 {
   public T data;
   private int i;
   public Node<T> next;
// конструктор
 }
```

Затем десериализовать его и убедиться, что все данные сохранились. Посмотреть содержимое XML-файла и понять, за счет чего десериализуется связный список (как восстанавливаются ссылки). Можно ли в этом случае использовать BinaryFormatter?

4. Используя класс XMLSerializer, сериализовать-десериализовать объекты класса Student в XML-формате. Использовать атрибуты [XmlAttribute] для полей, которые необходимо сериализовать. Чем отличается XML-представление от SoapFormatter? Понять ограничения этого сериализатора.

5. Используя SoapFormatter, создать структуру, поддерживающую граф объектов и сериализовать-десериализовать граф. Подумать над представлением графа в виде списка инцидентных вершин, не используя List<T>.

6. Используя DataContractSerializer, его методы ReadObject и WriteObject и атрибуты [DataContract] для класса и [DataMember] для открытых полей или свойств, сериализовать на диск в XML-формате объекты класса Student. Вызвав конструктор DataContractSerializer в виде new DataContractSerializer (typeof (Student), new Type[ ]{typeof (SeniorStudent)}), сериализовать-десериализовать также потомков Student типа SeniorStudent.

Задание №6

1. Продемонстрировать асинхронную работу методов с использованием пула потоков. Операции для пула потоков должны быть достаточно длительными.

2) Продемонстрировать работу именованного семафора для синхронизации действий между несколькими процессами.

3) Продемонстрировать работу именованного мьютекса для синхронизации действий между несколькими процессами.

4) Продемонстрировать работу ManualResetEvent в ситуации когда несколько потоков дожидаются возникновения события.

5. Продемонстрировать выполнение асинхронных вычислительных операций с использованием методов BeginInvoke и EndInvoke для делегатов.

6. Продемонстрировать выполнение асинхронных операций ввода-вывода для задачи чтения из большого файла, используя:

а) модель ожидания

б) модель опроса

в) модель с обратным вызовом

5.1.4 Типовые задания (оценочное средство - Контрольная работа) для оценки сформированности компетенции ПК-8:

Задание №1

1. Проверить, как работает явная и неявная реализация методов интерфейса. Для этого **явно** реализовать в классах Person и Student интерфейс

```
interface IPrintable {
  void Print();
}
```

и вызвать этот метод, используя переменную типа интерфейс. Заметим, что старый метод Print также можно оставить.

2. В классе Student реализовать интерфейс IComparable<Student>. Воспользовавшись Array.Sort, отсортировать массив студентов.

3. Реализовать интерфейс IComparer<Student> в классе StudentComparer, вложенном в Student. Параметром его конструктора должен выступать критерий сортировки. Реализовать в классе Student несколько статических свойств по количеству критериев сортировки (например, Student.SortByGroup). Воспользовавшись второй формой Array.Sort, отсортировать массив студентов по разным критериям.

4. Реализовать в классах Person-Student-Teacher интерфейс ICloneable и проиллюстрировать его использование.

5. Реализовать в классе Person интерфейс IDisposable и убедиться в корректности его работы в операторе using

Интерфейс IDisposable имеетвид:

```
interface IDisposable {
void Dispose();
}
```

Он используется для детерминированного освобождения ресурсов в стиле C++, метод Dispose играет роль деструктора. Если класс My поддерживает интерфейс IDisposable, то его можно использовать в операторе using:

```
using (My m = new My())
{
} // в конце гарантированно вызовется Dispose
```

В методе Dispose() класса Person достаточно выдавать диагностическое сообщение о том, что метод Dispose() вызван.

6. Создать контейнер Persons, который можно было бы использовать в foreach. Для этого поместить в него поле List<Person> list и метод Add(Person p), а также реализовать интерфейс IEnumerable<Person>, используя в методе GetEnumerator конструкцию yield return.

Примерреализации:

```
  public class CustomContainer
  {
     public IEnumerator<int> GetEnumerator()
     {
        for (int i = 0; i < 10; i++)
yield return i;
     }
  }
```

Заполнить контейнер Persons персонами и студентами, используя его метод Add. После этого воспользоваться методом foreach по данному контейнеру, выдавая всех персон и студентов в контейнере. Перед каждой персоной или студентом должен выводиться порядковый номер во внутреннем списке.

7. Реализовать обобщенную функцию MinIndex, возвращающую пару (индекс, МинимальныйЭлемент) в массиве элементов типа T. Для этого наложить в секции where ограничений на параметры обобщения условие where T: IComparable<T>. Результат возвращать в виде Tuple<int,T>. Если в массиве нет элементов, то индекс минимального равен -1, а минимальный равен default(T) - значению по умолчанию для типа T.

Задание № 2

1. Используя System.Globalization.NumberFormatInfo, переопределить разделитель в вещественном числе с запятой на точку (в русских версях windows по умолчанию - запятая). Для этого создать экземпляр NumberFormatInfo и переопределить в нем свойство NumberDecimalSeparator на '.'. Для преобразования вещественного в строку воспользоваться d.ToString(nfi), где nfi - переменная типа NumberFormatInfo. Для считывания вещественного с другим разделителем воспользоваться преобразованием строки в вещественное
double.Parse(s,nfi);
Реализовать на основе этого суммирование всех вещественных, записанных в строке в виде
3.14 2.8 4.19 5.3
и сформировать строку вида
3.14 + 2.8 + 4.19 + 5.3 = ...
Для формирования строки воспользоваться объектом класса StringBuilder, добавляя строку в конец.
2. Считать содержащиеся в строке шестнадцатеричные числа
AAFF BCDF 1FF4 123D9 CC11D3
и просуммировать их. Результат выдать в восьмеричной системе счисления.

Задание №3

1. Сохранить в текстовых файлах русско-английский тест в различных кодировках:
а) в однобайтовых: MS DOS, Koi-8, Windows
б) в Unicode: Utf-8, Utf-16
2. В текстовом файле записаны вещественные числа (на каждой строчке - несколько, разделены несколькими пробелами, в формате 3.14 2.597) и другие лексемы (не числа). Найти сумму чисел, игнорируя неверные лексемы.
3. Создать типизированный файл целых, затем модифицировать его, возведя все элементы в квадрат.
4. Для данной папки рекурсивно выдать список её файлов и подпапок.

Критерии оценивания (оценочное средство - Контрольная работа)

Оценка	Критерии оценивания
отлично	выставляется студенту, если представленная контрольная работа выполнена полностью без ошибок и недочетов
хорошо	выставляется студенту, если представленная контрольная работа выполнена полностью, но при наличии в ней не более одной негрубой ошибки и одного недочета, не более трех недочетов
удовлетворительно	выставляется студенту, если представленная им контрольная работа выполнена правильно не менее чем на 2/3 всей работы или в работе допущены не более одной грубой ошибки и двух недочетов, не более одной грубой и одной негрубой ошибки, не более трех негрубых ошибок, одной негрубой

Оценка	Критерии оценивания
	ошибки и трех недочетов, при наличии четырех-пяти недочетов
неудовлетворительно	выставляется студенту, если число ошибок и недочетов в работе превысило норму для оценки 3 или правильно выполнено менее 2/3 всей работы

5.1.5 Типовые задания (оценочное средство - Практическое задание) для оценки сформированности компетенции ПК-11:

Задание 7 (на файлы и потоки)

7.1. Сохранить в текстовых файлах русско-английский тест в различных кодировках:

а) в однобайтовых: MS DOS, Koi-8, Windows

б) в Unicode: Utf-8, Utf-16

7.2. В текстовом файле записаны вещественные числа (на каждой строчке - несколько, разделены несколькими пробелами, в формате 3.14 2.597) и другие лексемы (не числа). Найти сумму чисел, игнорируя неверные лексемы.

7.3. Создать типизированный файл целых, затем модифицировать его, возведя все элементы в квадрат.

7.4. Для данной папки рекурсивно выдать список её файлов и подпапок.

Задание 8 (на регулярные выражения)

Дана строка, состоящая из английских слов, разделенных одним или несколькими пробелами. В начале и в конце строки также могут быть пробелы

8.1. Вывести все слова - каждое на отдельной строке

```
var r = new Regex(@"\w+");
  foreach (Match s in r.Matches(" jh kj hjk k "))
Console.WriteLine(s);
```

8.2. Вывести все слова, начинающиеся и заканчивающиеся на 'a'

8.3. Вывести все слова, содержащие в начале от 2 до 4 согласных букв

8.4. Вывести все числа с десятичной точкой и вычислить их сумму

8.5. Вывести все полные имена файлов (и только их). Полные имена файлов задаются в формате Диск:\папка1\папка2\имя.расширение. Диск и папки могут отсутствовать.

8.6. Считать многострочную строку из файла слов (слова могут разделяться одним или несколькими пробелами и символами перехода на новую строку), используя вызов

```
var s = System.IO.File.ReadAllText('Text.txt');
```

Создать регулярное выражение с опцией RegexOptions.Multiline.

а) Вывести все слова, стоящие в начале строки.

б) Вывести все слова, стоящие в конце строки.

в) Вывести все слова стоящие первыми в строке (перед словами могут быть один или несколько пробелов)

г) Вывести все строки, начинающиеся с пробела

д) Вывести номера символов, с которых начинается каждая строка

Задание 9 (на сериализацию)

Для выполнения заданий следует подключить к основной сборке следующие сборки: System.Runtime.Serialization System.Runtime.Serialization.Formatters.Soap

9.1. Используя BinaryFormatter и атрибут [Serializable], сериализовать на диск в бинарном формате объекты классов Student и Teacher, после чего десериализовать. Убедиться, что сериализация работает корректно (для этого десериализовать и вывести на экран повторно). Обратить особое внимание, что класс Teacher содержит поле List<Student>. Попробовать провести сериализацию, используя SoapFormatter. Что не работает? Как это можно исправить (какое поле убрать/не сериализовать)?

9.2. Используя SoapFormatter и атрибут [Serializable], сериализовать на диск в XML формате связный список из нескольких

```
class Node<T>
 {
   public T data;
   private int i;
   public Node<T> next;
// конструктор
 }
```

Затем десериализовать его и убедиться, что все данные сохранились. Посмотреть содержимое XML-файла и понять, за счет чего десериализуется связный список (как восстанавливаются ссылки). Можно ли в этом случае использовать BinaryFormatter?

9.4. Используя класс XMLSerializer, сериализовать-десериализовать объекты класса Student в XML-формате. Использовать атрибуты [XmlAttribute] для полей, которые необходимо сериализовать. Чем отличается XML-представление от SoapFormatter? Понять ограничения этого сериализатора.

9.5. Используя SoapFormatter, создать структуру, поддерживающую граф объектов и сериализовать-десериализовать граф. Подумать над представлением графа в виде списка инцидентных вершин, не используя List<T>.

9.6. Используя DataContractSerializer, его методы ReadObject и WriteObject и атрибуты [DataContract] для класса и [DataMember] для открытых полей или свойств, сериализовать на диск в XML-формате объекты класса Student. Вызвав конструктор DataContractSerializer в виде new DataContractSerializer (typeof (Student), new Type[ ]{typeof (SeniorStudent)}), сериализовать-десериализовать также потомков Student типа SeniorStudent.

Задание 10 (на LINQ)

Как в дисплейном классе настроить папку для решения заданий LinqBegin из электронного задачника Programming Taskbook

1. Создать на своем диске папку LINQ
2. Установить PTForLinq
3. Скопировать в свою папку следующие файлы:

Load.lnk

results.dat

4. Запустить ярлык Load.lnk, правой мышью выбрать среду Visual Studio 2010 или 2012 или 2013
5. Набрать имя выполняемого задания LinqBegin1, LinqBegin2 и т.д., нажать Enter и проигнорировать вопрос, появляющийся при открытии проекта с заданием в VS.

Пример решения LinqBegin1

```
public static void Solve()
{
  Task("LinqBegin1");
```

```
  var seq = GetEnumerableInt();
  Put(seq.First(x => x > 0));
  Put(seq.Last(x => x < 0));
}
```

Задание 11 (на рефлексию)

11.1. Используя механизм отражения, "расшифровать" все классы, содержащиеся в dll. Создать объект класса, вызвать его метод, свойство (на чтение и запись).

11.2. Для данного типа вывести цепочку всех его предков и интерфейсов, ими реализуемых.

11.3. Создать программу автоподключения плагинов к программе. Плагин представляет собой dll, содержащую класс, удовлетворяющий некоторому интерфейсу. Программа должна анализировать все классы в dll, отбирать те, которые реализуют интерфейс, создавать по одному объекту каждого такого класса и вызывать методы интерфейса для этого объекта.

Задание 12 (на потоки (Threads))

12.1. Создать несколько потоков, выводящих на консоль в цикле свой hashcode. Посмотреть порядок вывода. Добавить Sleep(1), Sleep(10), Sleep(0). Посмотреть изменения в порядке вывода. Закоментировать Sleep() и установить приоритеты потокам. Посмотреть изменения в порядке вывода.

12.2. Написать программу, демонстрирующую работу Join(): основной поток должен дожидаться окончания работы (не менее) 2-х потоков, потом продолжать работу. Конструктивно: дополнительные потоки должны предоставлять данные, необходимые для дальнейшей работы основного потока.

12.3. Продемонстрировать работу критических секций на примере банкомата.

12.4. Используя критические секции, реализовать потокобезопасный класс Стек. Операции Push и Pop должны блокироваться одним объектом. Проиллюстрировать корректность работы стека, выполняя операции Push и Pop случайным образом в разных потоках. Продемонстрировать, что обычный класс стека не является потокобезопасным.

12.5. Продемонстрировать ситуацию с возникновением DeadLock.

12.6. Продемонстрировать работу Monitor'ов (Monitor.Wait, Monitor.Pulse, Monitor.PulseAll). Продемонстрировать отличия Monitor.Pulse и Monitor.PulseAll. Проверить работу Monitor.Wait, Monitor.Pulse для работы с очередью посетителей (один поток генерирует клиентов через разные промежутки времени, второй их обслуживает)

Задание 13 (на асинхронное программирование)

13.1. Продемонстрировать асинхронную работу методов с использованием пула потоков. Операции для пула потоков должны быть достаточно длительными.

13.2а) Продемонстрировать работу именованного семафора для синхронизации действий между несколькими процессами.

13.2б) Продемонстрировать работу именованного мьютекса для синхронизации действий между несколькими процессами.

13.2в) Продемонстрировать работу ManualResetEvent в ситуации когда несколько потоков дожидаются возникновения события.

13.3. Продемонстрировать выполнение асинхронных вычислительных операций с использованием методов BeginInvoke и EndInvoke для делегатов.

13.4. Продемонстрировать выполнение асинхронных операций ввода-вывода для задачи чтения из большого файла, используя:

а) модель ожидания

б) модель опроса

в) модель с обратным вызовом

5.1.6 Типовые задания (оценочное средство - Практическое задание) для оценки сформированности компетенции ПК-8:

Задание 1

1.1. Откомпилировать простейшую библиотеку .dll и простейшую программу .exe, вызывающую методы библиотеки, с помощью csc.exe. Записать размер полученных файлов.

1.2. Просмотреть метаданные в сборках .dll и .exe с помощью ILDasm

1.2a. Создать простейшее оконное приложение и записать его размер.

1.3. Написать dll на PascalABC.NET, содержащую функцию add, складывающую 2 числа. Просмотреть метаданные с помощью ildasm или ILSpy (скачать ILSpy из Интернета) и вызвать функцию add из PascalABC.NET-dll в программе на C#

1.4. Вызвать метод из dll, написанной на C#, в программе на PascalABC.NET

Задание 1а

1.5. Откомпилировать сборки exe и dll с помощью VisualStudioврежимахDebugиRelease. Сравнить размеры полученных файлов.

1.6. Аналогично обеспечить межъязыковое взаимодействие VB.NET <--> Managed C++

Задание 1б

1.7. Сравнить скорость работы вычислительного алгоритма на языках C#, C++, *C#,* PascalABC.NET, Free Pascal, Python

а) Сумма(i=1..n)(1/(i*j)), n - достаточно большое

б) Сумма(i=1..n)(1/(a[i]*a[j])), n - достаточно большое

в) Произведение квадратных матриц n x n, n - достаточно большое

Для замера времени в PascalABC.NET воспользоваться функцией Milliseconds, возвращающей время с начала работы программы в секундах. В настройках отключить режим Debug и запускать программу по Shift-F9.

Для замера времени в FP - установить режим Release и:

```
uses Windows;

{$apptype console}

var  tt: Cardinal;
begin
  tt := GetTickCount;
 ...
  writeln(GetTickCount-tt);
```

Для PascalABC.NET в настройках опций компиляции отключить "Генерировать отладочную информацию" и "Удалять exe после выполнения". Для компиляции программы воспользоваться командой Компилировать (Ctrl-F9) и запускать exe файл вне среды, либо запускать по Shift-F9 в режиме без связи с оболочкой.

Занести все данные по скорости в таблицу

В пунктах б) и в) заполнить массив a и матрицы случайными числами в диапазоне от 1 до 1.1. Для генерации случайных чисел в .NET пользоваться классом Random:

```
Random r = new Random();
r.NextDouble();
```

Сравнивать скорость в Debug и Release - конфигурациях (для FP включать все возможные оптимизации).
Для .NET-языков сравнить IL-код для циклов с помощью ildasm и выявить неэффективность генерации кода.

Задание 2 (на наследование и полиморфизм)

2.1. Создать иерархию классов Person-Student-Teacher. Каждый класс – в своей сборке. В каждом классе должны быть свойства, а также виртуальная функция Print и переопределенная функция ToString(). Основная программа создает массив объектов Person или их наследников, после чего выдает его на экран. У каждого Teacher должен быть список Students, которыми он руководит, у каждого Student - Teacher, который им руководит.

Замечание 1. В процессе реализации возникнет такая ошибка как циклическая зависимость сборок: сборка Student зависит от сборки Teacher и наоборот. Для устранения этой ошибки рекомендуется создать класс Student без поля Teacher, после чего создать производный класс StudentWithAdvisor с полем Teacher в отдельной сборке.

2.2. Для классов Person-Student-Teacher реализовать и оттестировать ToString(), Equals(), GetHashCode().

2.3. Для классов Person-Student-Teacher реализовать статические методы RandomPerson, RandomStudent, RandomTeacher, которые возвращают случайного из некоторого статического массива.

2.4. С помощью is, as, GetType определить, сколько в массиве персон, студентов и преподавателей и перевести всех студентов на следующий курс.

2.5. Для классов Person-Student-Teacher реализовать глубокое клонирование, определив виртуальный метод Clone(). Клон должен возвращать точную копию по значению и типу. Проиллюстрировать Clone на примере контейнера персон - должны создаваться клоны объекты ровно тех типов, которые содержатся в исходном контейнере.

2.6. Используя метод GetType() класса Student и метод BaseType() класса Type, вывести всех предков класса Student (написать общий метод)

Замечание 2. Свойства ("умные" поля) определяются так:

```
private int age;
public int Age {
  get { return age; }
  set { if (value<0) value = 0; age = value; }
}
```

Здесь value - переменная, неявно объявленная в каждом сеттере. Свойства отличаются от полей тем, что при доступе на чтение и запись можно совершать дополнительные действия. Обычная практика - проверка в сеттере значения на допустимость и его исправление или генерация исключения.

Замечание 3. В конструкторе потомка следует вызывать конструктор предка в списке инициализации:

```
Student(...): base(...) {}
```

Замечание 4. Виртуальные функции следует объявлять с ключевым словом virtual в предке и с ключевым словом override в потомках. Виртуальную функцию следует вызывать через переменную базового класса:

```
Person p = new Student(...);
p.Print();
```

Замечание 5. Функции ToString() и Equals() определены в базовом классе Object как виртуальные.

Замечание 6. p is Student возвращает True если в p - студент или производный класс. p as Student преобразует тип p к Student, а если это невозможно, возвращает null.

Замечание 7. Для сравнения на точное совпадение типа используется GetType: if (p.GetType()==typeof(Student))

Задание 3 (на перегрузку операторов)

3.0. Для классов Person-Student-Teacher реализовать ==, !=

3.1. Создать структуру Complex с перегруженными операциями, а также с возможностью приведения типа double->complex. Должны быть реализованы также ToString(), Equals(), ==, !=. Сравнить производительность в случае реализации Complex как класса и как структуры.

3.2. Создать класс Frac с перегруженными операциями + - * / , а также с возможностью приведения типа Frac->double. Должны быть реализованы также ToString(), Equals(), ==, !=. Вычислить значение полинома в точке. Все коэффициенты и x должны иметь тип Frac. Сравнить производительность в случае реализации Frac как класса и как структуры.

3.3. Используя класс Frac, реализовать метод Гаусса для решения системы линейных уравнений из n уравнений с n неизвестными и с рациональными коэффициентами, заданными типом Frac. Найти точное решение, представляющее собой List<Frac>.

Замечание 1. Операции реализуются как статические методы:

public static bool operator==(Person p1, Person p2) {return p1.Equals(p2);}

Замечание 2. Операции приведения типа определяются так:

public static explicit operator double(Frac f) {...}

explicit означает явное приведение типа, вместо него может стоять implicit - неявное приведение типа.

Задание 4(на индексаторы)

4.1. Создать класс, реализующий битовый массив на основе обычного, используя индексные свойства.

4.2. Создать класс ассоциативного массива, используя два списка List - список ключей и список значений. Основная операция: x = d[K] (на чтение) и d[K] = V (на запись)

Индексные свойства создаются следующим образом:

```
private Dictionary<string,int> a;
public int this[string s] {
  get { return a[s]; }
  set { a[s] = value; }
}
```

Задание 5 (на интерфейсы)

5.1. Проверить, как работает явная и неявная реализация методов интерфейса. Для этого **явно** реализовать в классах Person и Student интерфейс

```
interface IPrintable {
  void Print();
}
```

и вызвать этот метод, используя переменную типа интерфейс. Заметим, что старый метод Print также можно оставить.

5.2. В классе Student реализовать интерфейс IComparable<Student>. Воспользовавшись Array.Sort, отсортировать массив студентов.

5.3. Реализовать интерфейс IComparer<Student> в классе StudentComparer, вложенном в Student. Параметром его конструктора должен выступать критерий сортировки. Реализовать в классе Student несколько статических свойств по количеству критериев сортировки (например, Student.SortByGroup). Воспользовавшись второй формой Array.Sort, отсортировать массив студентов по разным критериям.
5.4. Реализовать в классах Person-Student-Teacher интерфейс ICloneable и проиллюстрировать его использование.
5.5. Реализовать в классе Person интерфейс IDisposable и убедиться в корректности его работы в операторе using
Интерфейс IDisposable имеет вид:

```
interface IDisposable {
void Dispose();
}
```

Он используется для детерминированного освобождения ресурсов в стиле C++, метод Dispose играет роль деструктора. Если класс My поддерживает интерфейс IDisposable, то его можно использовать в операторе using:

```
using (My m = new My())
{
} // в конце гарантированно вызовется Dispose
```

В методе Dispose() класса Person достаточно выдавать диагностическое сообщение о том, что метод Dispose() вызван.
5.6. Создать контейнер Persons, который можно было бы использовать в foreach. Для этого поместить в него поле List<Person> list и метод Add(Person p), а также реализовать интерфейс IEnumerable<Person>, используя в методе GetEnumerator конструкцию yield return.
Пример реализации:

```
  public class CustomContainer
  {
    public IEnumerator<int> GetEnumerator()
    {
      for (int i = 0; i < 10; i++)
yield return i;
    }
  }
```

Заполнить контейнер Persons персонами и студентами, используя его метод Add. После этого воспользоваться методом foreach по данному контейнеру, выдавая всех персон и студентов в контейнере. Перед каждой персоной или студентом должен выводиться порядковый номер во внутреннем списке.
5.7. Реализовать обобщенную функцию MinIndex, возвращающую пару (индекс, Минимальный Элемент) в массиве элементов типа T. Для этого наложить в секции where ограничений на параметры обобщения условие where T: IComparable<T>. Результат возвращать в виде Tuple<int,T>. Если в массиве нет элементов, то индекс минимального равен -1, а минимальный равен default(T) - значению по умолчанию для типа T.

Задание 6 (на строки)

6.1. Используя System.Globalization.NumberFormatInfo, переопределить разделитель в вещественном числе с запятой на точку (в русских версях windows по умолчанию - запятая). Для этого создать экземпляр NumberFormatInfo и переопределить в нем свойство

NumberDecimalSeparator на '.'. Для преобразования вещественного в строку воспользоваться d.ToString(nfi), где nfi - переменная типа NumberFormatInfo. Для считывания вещественного с другим разделителем воспользоваться преобразованием строки в вещественное
double.Parse(s,nfi);
Реализовать на основе этого суммирование всех вещественных, записанных в строке в виде
3.14 2.8 4.19 5.3
и сформировать строку вида
3.14 + 2.8 + 4.19 + 5.3 = ...
Для формирования строки воспользоваться объектом класса StringBuilder, добавляя строку в конец.
6.2. Считать содержащиеся в строке шестнадцатеричные числа
AAFF BCDF 1FF4 123D9 CC11D3
и просуммировать их. Результат выдать в восьмеричной системе счисления

Критерии оценивания (оценочное средство - Практическое задание)

Оценка	Критерии оценивания
зачтено	Ответ полный и правильный на основании изученной теории; теоретический материал и решение поставленных задач изложены в необходимой логической последовательности, грамотный научный язык; ответ самостоятельный. Могут быть допущены две-три несущественные ошибки, исправленные по требованию преподавателя.
не зачтено	Ответ обнаруживает непонимание студентом основного содержания учебного материала или допущены существенные ошибки, которые не могут быть исправлены при наводящих вопросах преподавателя.

5.1.7 Типовые задания (оценочное средство - Реферат) для оценки сформированности компетенции ПК-11:

1. Графический интерфейс языка C#
2. Создание графического интерфейса при помощи классов пакета JFC Swing
3. Объектно-ориентированное программирование

5.1.8 Типовые задания (оценочное средство - Реферат) для оценки сформированности компетенции ПК-8:

1. Создание графического интерфейса при помощи классов пакета AWT
2. Введение в C#: классы
3. Особенности языка программирования C#

Критерии оценивания (оценочное средство - Реферат)

Оценка	Критерии оценивания
отлично	Реферативная работа полностью раскрывает основные вопросы теоретического материала. Студент приводит информацию из первоисточников и изданий периодической печати, приводит практические примеры, отвечает на дополнительные вопросы преподавателя и студентов (в

Оценка	Критерии оценивания
	процессе выступления с докладом).
хорошо	Реферативная работа частично раскрывает основные вопросы теоретического материала. Студент приводит информацию из первоисточников, отвечает на дополнительные вопросы преподавателя и студентов (в процессе выступления с докладом), но при этом дает не четкие ответы, без достаточно их аргументации.
удовлетворительно	Реферативная работа в общих чертах раскрывает основные вопросы теоретического материала. Студент приводит информацию только из учебников. При ответах на дополнительные вопросы (в процессе выступления с докладом) путается в ответах, не может дать понятный и аргументированный ответ.
неудовлетворительно	ставится за рефераты, в которых нет информации о проблематике работы и ее месте в контексте других работ по исследуемой теме.

5.1.9 Типовые задания (оценочное средство - Тест) для оценки сформированности компетенции ПК-11:

Тест № 3

1. Какой из перечисленных методов позволит нарисовать квадрат, закрашенный определенным цветом?

A. Graphics.DrawLines
B. Graphics.DrawRectangle
C. Graphics.DrawPolygon
D. Graphics.FillRectangle
E. Graphics.FillEllipse

2. Какие из перечисленных классов необходимы, чтобы нарисовать окружность без заливки? (Укажите все верные ответы)

A. System.Drawing.Graphics
B. System.Drawing.Pen
C. System.Drawing.Brush
D. System.Drawing.Bitmap

3. Какую из перечисленных кистей следует использовать для рисования прямоугольника с градиентной заливкой от красного до белого цвета?

A. System.Drawing.Drawing2D.HatchBrush
B. System.Drawing.Drawing2D.LinearGradientBrush
C. System.Drawing.Drawing2D.PathGradientBrush
D. System.Drawing.Drawing2D.SolidBrush

4. Какие из перечисленных классов можно использовать для вывода на форму JPEG-изображения из существующего файла? (Укажите все верные ответы)

A. System.Drawing.Image
B. System.Drawing.Bitmap

C. System.Drawing.Imaging.Metafile
D. System.Windows.Forms.PictureBox

5. Какой из форматов следует выбрать для сохранения фото, предназначенного для открытия в разных приложениях?

A. ImageFormat.Bmp
B. ImageFormat.Gif
C. ImageFormat.Jpeg
D. ImageFormat.Png

6. Как добавить текст на изображение?

A. Создать объекты Graphics и string. После этого вызвать string.Draw.
B. Создать объекты Graphics, Font и Brush. После этого вызвать Graphics.DrawString.
C. Создать объекты Graphics, Font и Pen. После этого вызвать Graphics.DrawString.
D. Создать объекты Bitmap, Font и Brush. После этого вызвать Bitmap.DrawString.

5.1.10 Типовые задания (оценочное средство - Тест) для оценки сформированности компетенции ПК-8:

Тест № 1

1. Какие из перечисленных типов являются ссылочными? (Укажите все верные ответы)

A. Double
B. StringBuilder
C. Exception
D. Все типы, порожденные от System.Object

2. Укажите ошибочное утверждение.

A. В языке C# переменные могут быть описаны внутри блока.
B. В языке C# локальные переменные по умолчанию инициализируются нулевыми значениями.
C. В языке C# допускается явно инициализировать локальные переменные в момент их описания.
D. В языке C# переменная цикла for может быть описана в его заголовке.

3. Укажите выражение, позволяющее получить из вещественного числа x его целочисленное округленное значение (к ближайшему целому)

A. Math.Round(x)
B. (int)Math.Round(x)
C. Math.Round((int)x)
D. (int)x

4. Что произойдет при обработке следующего оператора:
Console.WriteLine(Math.Sqrt(-1));

A. При компиляции будет выведено сообщение об ошибке.
B. При выполнении будет возбуждена исключительная ситуация.
C. При выполнении будет выведен текст "NaN".
D. При выполнении будет выведен текст "I".

5. Укажите оператор, позволяющий в консольном приложении ввести с клавиатуры вещественное число и записать его в вещественную переменную x.

A. Console.Read(x);
B. x = Console.ReadDouble();
C. x = (double)Console.ReadLine();
D. x = double.Parse(Console.ReadLine());

6. Укажите оператор, обеспечивающий вывод вещественного числа x с двумя дробными знаками после запятой.

A. Console.WriteLine("{0:f2}", x);

B. Console.WriteLine("{f2}", x);

C. Console.WriteLine("{0,f2}", x);

D. Console.WriteLine("{0,2}", x);

Тест № 2

1. Укажите, какое из перечисленных утверждений является верным.

A. С каждым событием необходимо связать ровно один обработчик.

B. С каждым событием можно связать не более одного обработчика.

C. С событием можно связать несколько обработчиков, причем в любой момент времени можно определить, сколько обработчиков связано с данным событием.

D. С событием можно связать несколько обработчиков, однако определить их точное количество нельзя.

2. Скрытие каких из перечисленных элементов заголовка формы выполняется автоматически в случае, если для формы установлен стиль границы FixedDialog? (Укажите все верные ответы)

A. Значок в левой части заголовка

B. Текст заголовка

C. Кнопка минимизации формы

D. Кнопка максимизации формы

3. Укажите верный вариант завершения следующего утверждения: «При закрытии подчиненной формы…

A. … всегда происходит разрушение этой формы»

B. … никогда не происходит разрушения этой формы»

C. … форма, отображенная в модальном режиме, разрушается, а форма, отображенная в немодальном режиме, — нет»

D. … форма, отображенная в немодальном режиме, разрушается, а форма, отображенная в модальном режиме, — нет»

4. Нажатие на какие из перечисленных клавиш не будет перехвачено обработчиком события KeyPress? (Укажите все верные ответы)

A. [Esc]

B. [Enter]

C. [PgUp]

D. [F1]

5. В каких обработчиках событий от мыши в свойстве e.Button содержится информация обо всех кнопках мыши, нажатых в момент срабатывания обработчика?

A. MouseDown

B. MouseMove

C. MouseDown и MouseMove

D. Ни в одном из указанных обработчиков данная информация не содержится; ее можно получить только с помощью свойства Control.MouseButtons

6. Какое из перечисленных событий, связанных с режимом Drag & Drop, возникает в ситуации, когда перетаскивание завершается над недоступным приемником?

A. DragEnter

B. DragOver
C. DragDrop
D. DragLeave

Критерии оценивания (оценочное средство - Тест)

Оценка	Критерии оценивания
отлично	85-100% правильных ответов
хорошо	66-84 % правильных ответов
удовлетворительно	50-65 % правильных ответов
неудовлетворительно	меньше 50 % правильных ответов

5.2. Описание шкал оценивания результатов обучения по дисциплине при промежуточной аттестации

Шкала оценивания сформированности компетенций

Уровень сформированности компетенций (индикатора достижения компетенций)	**неудовлетворительно**	**удовлетворительно**	**хорошо**	**отлично**
	не зачтено	**зачтено**		
Знания	Уровень знаний ниже минимальных требований. Имели место грубые ошибки	Минимально допустимый уровень знаний. Допущено много негрубых ошибок	Уровень знаний в объеме, соответствующем программе подготовки. Допущено несколько негрубых ошибок	Уровень знаний в объеме, соответствующем программе подготовки, без ошибок
Умения	При решении стандартных задач не продемонстрированы основные умения. Имели место грубые ошибки	Продемонстрированы основные умения. Решены типовые задачи с негрубыми ошибками. Выполнены все задания, но не в полном объеме	Продемонстрированы все основные умения. Решены все основные задачи с негрубыми ошибками. Выполнены все задания в полном объеме, но некоторые с недочетами	Продемонстрированы все основные умения. Решены все основные задачи с отдельными несущественными недочетами, выполнены все задания в полном объеме
Навыки	При решении стандартных задач не продемонстрированы базовые навыки. Имели место грубые ошибки	Имеется минимальный набор навыков для решения стандартных задач с некоторыми недочетами	Продемонстрированы базовые навыки при решении стандартных задач с некоторыми недочетами	Продемонстрированы навыки при решении нестандартных задач без ошибок и недочетов

Шкала оценивания при промежуточной аттестации

Оценка		**Уровень подготовки**
зачтено	**отлично**	Все компетенции (части компетенций), на формирование которых направлена дисциплина, сформированы на уровне не ниже «отлично», при этом хотя бы одна компетенция сформирована на уровне «отлично»

	хорошо	Все компетенции (части компетенций), на формирование которых направлена дисциплина, сформированы на уровне не ниже «хорошо», при этом хотя бы одна компетенция сформирована на уровне «хорошо»
	удовлетворительно	Все компетенции (части компетенций), на формирование которых направлена дисциплина, сформированы на уровне не ниже «удовлетворительно», при этом хотя бы одна компетенция сформирована на уровне «удовлетворительно»
не зачтено	**неудовлетворительно**	Хотя бы одна компетенция сформирована на уровне «неудовлетворительно».

5.3 Типовые контрольные задания или иные материалы, необходимые для оценки результатов обучения на промежуточной аттестации с указанием критериев их оценивания:

5.3.1 Типовые задания (оценочное средство - Контрольные вопросы) для оценки сформированности компетенции ПК-11

1. Инфраструктура .NET Framework и общеязыковая исполняющая среда.
2. Классы и структуры.
3. Область видимости и уровни доступа.
4. Применение классов и демонстрация сбора мусора.
5. Принципы разработки пользовательского интерфейса.
6. Применение элементов управления и компонентов.
7. Проверка данных, вводимых пользователем.

5.3.2 Типовые задания (оценочное средство - Контрольные вопросы) для оценки сформированности компетенции ПК-8

1. История создания и развития .NET Framework.
2. Библиотека базовых классов .NET.
3. Методы.
4. Сбор мусора.
5. Создание пользовательского интерфейса.
6. Работа с формами.
7. Меню.
8. Приложение Virtual Doughnut Factory.

Критерии оценивания (оценочное средство - Контрольные вопросы)

Оценка	Критерии оценивания
зачтено	Ответ полный и правильный на основании изученной теории; теоретический материал и решение поставленных задач изложены в необходимой логической последовательности, грамотный научный язык; ответ самостоятельный. Могут быть допущены две-три несущественные ошибки, исправленные по требованию преподавателя.
не зачтено	Ответ обнаруживает непонимание студентом основного содержания учебного материала или допущены существенные ошибки, которые не могут быть исправлены при наводящих вопросах преподавателя.

6. Учебно-методическое и информационное обеспечение дисциплины (модуля)

Основная литература:

1. Подбельский В. В. Программирование. Базовый курс C# : учебник / В. В. Подбельский. - Москва : Юрайт, 2022. - 369 с. - (Высшее образование). - URL: https://urait.ru/bcode/469616 (дата обращения: 14.08.2022). - ISBN 978-5-534-10616-9 : 1439.00. - Текст : электронный // ЭБС "Юрайт"., https://e-lib.unn.ru/MegaPro/UserEntry?Action=FindDocs&ids=817197&idb=0.
2. Зыков С. В. Программирование. Объектно-ориентированный подход : учебник и практикум / С. В. Зыков. - Москва : Юрайт, 2022. - 155 с. - (Высшее образование). - URL: https://urait.ru/bcode/490423 (дата обращения: 14.08.2022). - ISBN 978-5-534-00850-0 : 709.00. - Текст : электронный // ЭБС "Юрайт"., https://e-lib.unn.ru/MegaPro/UserEntry?Action=FindDocs&ids=822044&idb=0.
3. Казанский А. А. Программирование на Visual C# : учебное пособие / А. А. Казанский. - 2-е изд. ; пер. и доп. - Москва : Юрайт, 2022. - 192 с. - (Высшее образование). - URL: https://urait.ru/bcode/470261 (дата обращения: 14.08.2022). - ISBN 978-5-534-12338-8 : 829.00. - Текст : электронный // ЭБС "Юрайт"., https://e-lib.unn.ru/MegaPro/UserEntry?Action=FindDocs&ids=819622&idb=0.
4. Тузовский А. Ф. Объектно-ориентированное программирование / Тузовский А. Ф. - Москва : Юрайт, 2022. - 206 с. - (Высшее образование). - URL: https://urait.ru/bcode/490369 (дата обращения: 05.01.2022). - ISBN 978-5-534-00849-4 : 699.00. - Текст : электронный // ЭБС "Юрайт"., https://e-lib.unn.ru/MegaPro/UserEntry?Action=FindDocs&ids=785008&idb=0.

Дополнительная литература:

1. Гуриков Сергей Ростиславович. Введение в программирование на языке Visual C# : Учебное пособие / Московский технический университет связи и информатики. - Москва : Издательство "ФОРУМ", 2020. - 447 с. - ВО - Бакалавриат. - ISBN 978-5-00091-458-8. - ISBN 978-5-16-105882-4. - ISBN 978-5-16-013100-9., https://e-lib.unn.ru/MegaPro/UserEntry?Action=FindDocs&ids=633122&idb=0.
2. Язык C#. Базовый курс / Подбельский В.В. - Москва : Финансы и статистика, 2015., https://e-lib.unn.ru/MegaPro/UserEntry?Action=FindDocs&ids=644571&idb=0.

Программное обеспечение и Интернет-ресурсы (в соответствии с содержанием дисциплины):

Лицензионное программное обеспечение: Операционная система Windows.
Лицензионное программное обеспечение: Microsoft Office.

Профессиональные базы данных и информационные справочные системы

Российский индекс научного цитирования (РИНЦ), платформа Elibrary: национальная информационно-аналитическая система. Адрес доступа: http://elibrary.ru/project_risc.asp

ГАРАНТ. Информационно-правовой портал [Электронный ресурс].– Адрес доступа: http://www.garant.ru

Свободно распространяемое программное обеспечение:

программное обеспечение LibreOffice;
программное обеспечение Yandex Browser;
программное обеспечение Paint.NET;

программное обеспечение 1С:
* "Бухгалтерия предприятия", редакция 3.0, см. http://v8.1c.ru/buhv8/ ,
* "Управление торговлей", редакция 11.1, см. http://v8.1c.ru/trade/ ,
* "Зарплата и управление персоналом", редакция 3.0, см. http://v8.1c.ru/hrm/ ,
* "Управление небольшой фирмой", редакция 1.5, см. http://v8.1c.ru/small.biz/ ,
* "ERP Управление предприятием 2.0", см. http://v8.1c.ru/erp/ .
* "Бухгалтерия государственного учреждения", редакция 1.0, см. http://v8.1c.ru/stateacc/ ,
* "Зарплата и кадры государственного учреждения", редакция 1.0, http://v8.1c.ru/statehrm/ .

программное обеспечение PascalABC.NET

Электронные библиотечные системы и библиотеки:
Электронная библиотечная система "Лань" https://e.lanbook.com/
Электронная библиотечная система "Консультант студента" http://www.studentlibrary.ru/
Электронная библиотечная система "Юрайт"http://www.urait.ru/ebs
Электронная библиотечная система "Znanium" http://znanium.com/
Электронно-библиотечная система Университетская библиотека ONLINE http://biblioclub.ru/

Фундаментальная библиотека ННГУ www.lib.unn.ru/
Сайт библиотеки Арзамасского филиала ННГУ. – Адрес доступа: lib.arz.unn.ru

Ресурс «Массовые открытые онлайн-курсы Нижегородского университета им. Н.И. Лобачевского» https://mooc.unn.ru/
Портал «Современная цифровая образовательная среда Российской Федерации» https://online.edu.ru/public/promo

7. Материально-техническое обеспечение дисциплины (модуля)

Учебные аудитории для проведения учебных занятий, предусмотренных образовательной программой, оснащены мультимедийным оборудованием (проектор, экран), техническими средствами обучения, компьютерами.

Помещения для самостоятельной работы обучающихся оснащены компьютерной техникой с возможностью подключения к сети "Интернет" и обеспечены доступом в электронную информационно-образовательную среду.

Программа составлена в соответствии с требованиями ОС ННГУ по направлению подготовки/специальности 09.03.03 - Прикладная информатика.

Автор(ы): Сазанов Александр Анатольевич.

Рецензент(ы): Ямпурин Николай Петрович, доктор технических наук.

Заведующий кафедрой: Нестерова Лариса Юрьевна, кандидат педагогических наук.

Программа одобрена на заседании методической комиссии от 10.01.2024 г., протокол № 1.