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

Факультет естественных и математических наук

УТВЕРЖДЕНО
решением Ученого совета ННГУ
протокол № 6 от 31.05.2023 г.

Рабочая программа дисциплины

Разработка интерфейса с использованием языка C#

(наименование дисциплины)

Уровень высшего образования
бакалавриат
(бакалавриат / магистратура / специалитет)

Направление подготовки / специальность
09.03.03 Прикладная информатика
(указывается код и наименование направления подготовки / специальности)

Направленность образовательной программы
Системное и прикладное программирование
(указывается профиль / магистерская программа / специализация)

Форма обучения
Очная/очно-заочная/заочная
(очная / очно-заочная / заочная)

Год начала подготовки 2021
Арзамас
2023 год

1. Место дисциплины (модуля) в структуре ООП

Дисциплина Б1.В.ДВ.01.02 «Разработка интерфейса с использованием языка C#» относится к части, формируемой участниками образовательных отношений, образовательной программы направления подготовки 09.03.03 Прикладная информатика, направленность (профиль) Системное и прикладное программирование.

Дисциплина предназначена для освоения студентами очной/очно-заочной/заочной формы обучения в 6 семестре/6 семестре/8 семестре.

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 (дескрипторы компетенции) ****	
ПК-8. Способен разрабатывать лингвистическое, информационное и программное обеспечение ИС (ИИС) и сопровождающую его документацию	ПК-8.1. Демонстрирует знание современных языков и систем программирования, формализмов описания знаний на концептуальном и инфологическом уровнях, требований к технической документации на все виды обеспечения ИС (ИИС).	*Знать* современное состояние и принципиальные возможности языка 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#,	*Тест*
	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 ее документации.	*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#.	*Учебно-исследовательские реферативные работы*
	ПК-8.3. Имеет практический опыт разработки лингвистического, информационного и программного обеспечения конкретной ИС (ИИС) и сопровождающей ее документации.	*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,	*Контрольные задания по теоретическим основам дисциплины*

		информационного и программного обеспечения конкретной ИС (ИИС) и сопровождающей его документации.	
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).	*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	*Тест* *Контрольные задания по теоретическим основам дисциплины*
	ПК-11.2. Демонстрирует умение осуществлять модульное и интеграционное тестирование ИС (ИИС) и устранять (по мере возможности) обнаруженные несоответствия.	*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	*Учебно-исследовательские реферативные работы*
	ПК-11.3. Имеет практический опыт модульного и интеграционного тестирования конкретной ИС (ИИС).	*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 форма-	*Тест* *практические задания*

		лизации решения прикладных задач, создавать web-сервисы и J2EE-приложения	

3. Структура и содержание дисциплины

3.1.Структура дисциплины

Трудоемкость	очная форма обучения	очно-заочная форма обучения	заочная форма обучения
Общая трудоемкость	3 з.е.		
часов по учебному плану, из них	108		
Контактная работа, в том числе: аудиторные занятия:			
– занятия лекционного типа	-	-	-
– занятия семинарского типа	36	36	6
контроль самостоятельной работы	1	1	1
Промежуточная аттестация зачет			4
Самостоятельная работа	71	71	97

3.2. Содержание дисциплины

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

Наименование разделов (Р) или тем (Т) дисциплины (модуля), Форма(ы) промежуточной аттестации по дисциплине	Всего (часы)			Контактная работа (работа во взаимодействии с преподавателем), часы, из них												Самостоятельная работа обучающегося, часы, в период					
				Занятия лекционного типа			Занятия семинарского типа (в т.ч. текущий контроль успеваемости)						Контроль самостоятельной работы			промежуточной аттестации (контроля)			теоретического обучения		
							семинары, практические занятия			лабораторные работы											
	Очная	Очно-заочная	Заочная	Очная	Очно-заочная	Заочная	Очная	Очно-заочная	Заочная	Очная	Очно-заочная	Заочная	Очная	Очно-заочная	Заочная	Очная	Очно-заочная	Заочная	Очная	Очно-заочная	Заочная
Тема 1. Основы программирования на языке C#. Синтаксис языка C#	**17**	**17**	**17**				6	6	1										11	11	16
Тема 2. Графические интерфейсы пользователя	**18**	**18**	**17**				6	6	1										12	12	16
Тема 3. Введение в .NET Framework	**18**	**18**	**17**				6	6	1										12	12	16
Тема 4. Классы и структуры. Методы.	**18**	**18**	**17**				6	6	1										12	12	16
Тема 5. Принципы разработки пользовательского интерфейса	**18**	**18**	**17**				6	6	1										12	12	16
Тема 6. Обзор передовых технологий языка C#	**18**	**18**	**18**				6	6	1										12	12	17

В том числе текущий контроль	**1**	**1**	**1**										**1**	**1**	**1**						
Зачет			**4**															**4**			
ИТОГО	**108**	**108**	**108**				**36**	**36**	**10**				**1**	**1**	**1**			**4**	**71**	**71**	**93**

Практические занятия (семинарские занятия) организуются, в том числе в форме практической подготовки, которая предусматривает участие обучающихся в выполнении отдельных элементов работ, связанных с будущей профессиональной деятельностью.

Практическая подготовка предусматривает выполнение заданий разработке приложений на языке программирования Java.

На проведение практических занятий (семинарских занятий) в форме практической подготовки отводится 6 часов.

Практическая подготовка направлена на формирование и развитие:
- практических навыков в соответствии с профилем ОП:
  - разработка лингвистического, информационного и программного обеспечения ИС (ИИС)
  - осуществление модульного и интеграционного тестирования ИС (ИИС),
- компетенций ПК-8, ПК-11.

Текущий контроль успеваемости реализуется в рамках занятий семинарского типа.

4. Учебно-методическое обеспечение самостоятельной работы студентов

Самостоятельная работа является важнейшей составной частью учебного процесса и обязанностью каждого студента.

Для обеспечения самостоятельной работы обучающихся используется электронный управляемый курс «Разработка интерфейса с использованием языка C#» (https://e-learning.unn.ru/course/view.php?id=10494) созданный в системе электронного обучения ННГУ https://e-learning.unn.ru/.

Самостоятельная работа студентов по дисциплине «Разработка интерфейса с использованием языка C#» осуществляется в следующих видах: работа с основной и дополнительной литературой, выполнение заданий различных типов, составления тезисов литературных источников, подготовки рефератов, разработка проектных работ, подготовка презентаций.

Контрольные вопросы и задания для проведения текущего контроля и промежуточной аттестации по итогам освоения дисциплины приведены в п. 5.3.

Методические рекомендации к самостоятельной работе

Методические рекомендации по подготовке к занятиям семинарского типа

Подготовка к занятиям семинарского типа (практическим занятиям) – традиционная форма самостоятельной работы обучающихся, включает отработку лекционного материала, изучение рекомендованной литературы, конспектирование предложенных источников.

Подготовка к опросу, проводимому в рамках практического занятия, требует уяснения вопросов, вынесенных на конкретное занятие, подготовки выступлений, повторения основных терминов, запоминания формул и алгоритмов.

На практических занятиях рассматриваются наиболее важные, существенные, сложные вопросы, которые, как свидетельствует преподавательская практика, наиболее трудно усваиваются студентами. Готовиться к практическим занятиям необходимо заблаговременно.

Подготовка к семинарским (практическим) занятиям включает в себя:

- обязательное ознакомление с планом практического занятия, в котором содержатся основные вопросы, выносимые на обсуждение;
- изучение конспектов лекций, соответствующих разделов учебника, учебного пособия, содержания рекомендованных нормативных правовых актов;

- изучение дополнительной литературы по теме практического занятия с обязательным конспектированием материала, который понадобится при обсуждении на семинаре.

Помните, что необходимо:

- выписать основные термины и запомнить их дефиниции;
- записывать возникшие во время самостоятельной работы с учебниками и научной литературы вопросы, чтобы затем на семинаре получить на них ответы;
- иметь продуманные и аргументировано обоснованные формулировки собственной позиции по каждому вопросу плана практического занятия;
- обращаться за консультацией к преподавателю при возникновении затруднений в освоении материала практической работы.

Выступление на практических занятиях должно удовлетворять следующим требованиям: в выступлении излагаются теоретические подходы к рассматриваемому вопросу, дается анализ принципов, законов, понятий и категорий; теоретические положения подкрепляются фактами, примерами, выступление должно быть аргументированным. Для более углубленного изучения вопросов рекомендуется конспектирование основной и дополнительной литературы.

Большую помощь при подготовке к занятиям может оказать изучение публикаций в научных журналах, а также специальные Интернет-ресурсы по тематике дисциплины, указанные п. 6 настоящей рабочей программы дисциплины

Рекомендации для работы с основной и дополнительной литературой

Работа с литературой должна сопровождаться записями в форме конспекта, плана, тезисов. При этом важно не только привлечь более широкий круг литературы, но и суметь на ее основе разобраться в степени изученности темы. Стоит выявить дискуссионные вопросы, нерешенные проблемы, попытаться высказать свое отношение к ним. Привести и аргументировать свою точку зрения или отметить, какой из имеющихся в литературе точек зрения по данной проблематике придерживаетесь и почему.

По завершении изучения рекомендуемой литературы полезно проверить уровень своих знаний с помощью контрольных вопросов для самопроверки. Необходимо вести систематическую работу над литературными источниками. Необходимо изучать не только литературу, рекомендуемую в данных учебно-методических материалах, но и новые, важные издания по курсу, вышедшие в свет после публикации. При этом следует выделять неясные, сложные для восприятия вопросы. В целях прояснения последних нужно обращаться к преподавателю.

Рекомендации для написания
учебно-исследовательской реферативной работы

Учебно-исследовательская реферативная работа – изложение в письменном виде содержания научного труда (трудов), литературы по теме. Цель написания учебно-исследовательской реферативной работы – овладение навыками анализа и краткого изложения изученных материалов в соответствии с требованиями, предъявляемыми к таковым работам. Это самостоятельная работа студента, где раскрывается суть исследуемой проблемы, приводятся различные точки зрения, собственные взгляды на нее. Содержание работы должно быть логическим, изложение материала носит проблемно-тематический характер.

Примерный алгоритм действий при написании реферата:

1. Подберите и изучите основные источники по теме (как правило, при разработке реферата или доклада используется не менее 8-15 различных источников).
2. Составьте библиографию.
3. Разработайте план реферата или доклада исходя из имеющейся информации.
4. Обработайте и систематизируйте подобранную информацию по теме.
5. Отредактируйте текст реферата или доклад с использованием компьютерных технологий.
6. Подготовьте публичное выступление по материалам реферата или доклада, желательно подготовить презентацию, иллюстрирующую основные положения работы.

Критерии результатов работы для самопроверки:

- актуальность темы исследования;
- соответствие содержания теме;
- глубина проработки материала;
- правильность и полнота использования источников;
- соответствие оформления реферата или доклада предъявляемым требованиям.

Самостоятельное изучение отдельных тем (вопросов) в соответствии со структурой дисциплины по учебной и специальной литературе

Активизация учебной деятельности и индивидуализация обучения предполагает вынесение для самостоятельного изучения отдельных тем или вопросов. Выбор тем (вопросов) для самостоятельного изучения – одна из ключевых проблем педагога в организации эффективной работы обучающихся по овладению учебным материалом.

Особую роль самостоятельное изучение отдельных тем (вопросов) дисциплины играет для студентов заочной формы обучения.

При этом, как правило, основанием выбора является наилучшая обеспеченность литературой и учебно-методическими материалами по данной теме, ее обобщающий характер, сформированный на аудиторных занятиях алгоритм изучения. Обязательным условием результативности самостоятельного освоения темы (вопроса) является контроль выполнения задания.

Вопросы для самостоятельного изучения тем (вопросов) указаны в рабочей программе дисциплины (модуля)».

Результаты самостоятельного изучения вопросов, будут проверены преподавателем в форме: опросов, конспектов, рефератов, ответов на экзаменах.

Самостоятельное выполнение расчетных заданий

1. Внимательно прочитайте теоретический материал – конспект, составленный на лекционном занятии, материал учебника, пособия. Выпишите формулы из конспекта по изучаемой теме.
2. Обратите внимание, как использовались данные формулы при решении задач на занятии.
3. Решите предложенную задачу, используя выписанныеформулы.
4. В случае необходимости воспользуйтесь справочными данными.
5. Проанализируйте полученный результат (проверьте размерности величин, правильность подстановки в формулы численных значений, правильность расчетов, правильность вывода неизвестной величины из формулы).
6. Решение задач должно сопровождаться необходимыми пояснениями. Расчётные формулы приводите на отдельной строке, выделяя из текста, с указанием размерности величин. Формулы записывайте сначала в общем виде (буквенное выражение), затем подставляйте числовые значения без указания размерностей, после чего приведите конечный результат расчётной величины.

Показатели результатов работы для самопроверки:

- грамотная запись условия задачи и ее решения;
- грамотное использование формул;
- грамотное использование справочной литературы;
- точность и правильность расчетов;
- обоснование решения задачи.

Подготовка к промежуточной аттестации: подготовка к зачету

Методические рекомендации

по подготовке к зачету

Зачет проводится в традиционной форме (ответ на вопросы экзаменационного билета, контрольная работа, тестирование) и/или в иных формах (с учетом оценок за коллоквиум, кейс, деловая или ролевая игра, презентация проекта и др.)

Подготовка к зачету начинается с первого занятия по дисциплине. При этом важно с самого начала планомерно осваивать материал, руководствуясь требованиями, конспектировать важные для решения учебных задач источники, обращаться к преподавателю за консультацией по неусвоенным вопросам.

Для подготовки к сдаче зачета необходимо первоначально прочитать лекционный материал, а также соответствующие разделы рекомендуемых изданий. Лучшим вариантом является тот, при котором при подготовке используется несколько источников информации. Это способствует разностороннему восприятию каждой конкретной темы дисциплины.

В обобщённом варианте подготовка к сдаче зачета включает в себя:

- просмотр программы учебной дисциплины, перечня вопросов к зачету;
- подбор рекомендованных преподавателем источников (учебников, нормативных правовых актов, дополнительной литературы и т.д.),
- использование конспектов лекций, материалов занятий и их изучение;
- консультирование у преподавателя.

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 промежуточной аттестации по дисциплине

5.1. Описание шкал оценивания результатов обучения по дисциплине

В ходе промежуточной аттестации по дисциплине осуществляется оценка сформированности компонентов компетенций (полнота знаний/ наличие умений/ навыков), т.е. результатов обучения, указанных в таблице п.2 настоящей рабочей программы, на основе оценки усвоения содержания дисциплины.

Обобщенная оценка сформированности компонентного состава компетенции в ходе промежуточной аттестации по дисциплине проводится на основе учета текущей успеваемости в ходе освоения дисциплины и учета результата сдачи промежуточной аттестации.

Выявленные признаки несформированности компонентов (индикаторов) хотя бы одной компетенции не позволяют выставить интегрированную положительную оценку сформированности компетенций и освоения дисциплины на данном этапе обучения.

Обобщенная оценка сформированности компонентного состава компетенций на промежуточной аттестации, которая вносится в зачетно-экзаменационную ведомость по дисциплине и зачетную книжку студента, осуществляется по следующей оценочной шкале.

Шкала оценки сформированности компонентного состава компетенций на промежуточной аттестации

Оценка		Уровень подготовки
Зачтено	Отлично	сформированность компонентного состава (индикаторов) компетенций соответствует требованиям компетентностной модели будущего выпускника на данном этапе обучения, основанным на требованиях ОС ННГУ по направлению подготовки, студент готов самостоятельно решать стандартные и нестандартные профессиональные задачи в предметной области дисциплины в соответствии с типами задач профессиональной деятельности осваиваемой образовательной программы
	Хорошо	сформированность компонентного состава (индикаторов) компетенций соответствует требованиям компетентностной модели будущего выпускника на данном

		этапе обучения, основанным на требованиях ОС ННГУ по направлению подготовки, но студент готов самостоятельно решать только различные стандартные профессиональные задачи в предметной области дисциплины в соответствии с типами задач профессиональной деятельности осваиваемой образовательной программы
	Удовлетворительно	сформированность компонентного состава (индикаторов) компетенций соответствует в целом требованиям компетентностной модели будущего выпускника на данном этапе обучения, основанным на требованиях ОС ННГУ по направлению подготовки, но студент способен решать лишь минимум стандартных профессиональных задач в предметной области дисциплины в соответствии с типами задач профессиональной деятельности осваиваемой образовательной программы
Не зачтено	Неудовлетворительно	сформированность компонентного состава (индикаторов) компетенций не соответствует требованиям компетентностной модели будущего выпускника на данном этапе обучения, основанным на требованиях ОС ННГУ по направлению подготовки, студент не готов решать профессиональные задачи в предметной области дисциплины в соответствии с типами задач профессиональной деятельности осваиваемой образовательной программы

Шкала оценивания сформированности компетенции

Уровень сформированности компетенции (индикатора достижения компетенции)				
	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 требованиям программы 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.

5.2 Критерии и процедуры оценивания результатов обучения по дисциплине

Критерии оценки устного опроса

Оценка **«отлично»** - Ответ полный и правильный, на основании изученной теории; материал изложен в определенной логической последовательности, грамотный научный язык; ответ самостоятельный.

Оценка **«хорошо» -** Ответ полный и правильный, на основании изученной теории; материал изложен в определенной логической последовательности при этом допущены две-три несущественные ошибки, исправленные по требованию преподавателя.

Оценка **«удовлетворительно»** - Ответ полный, но при этом допущена существенная ошибка или неполный, несвязный ответ.

Оценка **«неудовлетворительно»** - Ответ обнаруживает непонимание студентом основного содержания учебного материала или допущены существенные ошибки, которые не могут быть исправлены при наводящих вопросах преподавателя.

Критерии оценивания письменных контрольных работ

*оценка «**отлично**»* выставляется студенту, если представленная контрольная работа выполнена полностью без ошибок и недочетов;

*оценка «**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;

*оценка «**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 ошибки и трех недочетов, при наличии четырех-пяти недочетов;

*оценка «**неудовлетворительно**»* выставляется студенту, если число ошибок и недочетов в работе превысило норму для оценки 3 или правильно выполнено менее 2/3 всей работы.

Критерии оценки тестирования

Оценка "**отлично"** - 85-100% правильных ответов;

Оценка **"хорошо"**66-84 % правильных ответов;

Оценка **"удовлетворительно"**– 50-65 % правильных ответов;

Оценка "**неудовлетворительно"** - меньше 50 %.

Критерии оценки письменной учебно-исследовательской реферативной работы

Оценка **"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 процессе выступления с докладом).

Оценка **"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.

Оценка **"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.

Оценка **«неудовлетворительно»** ставится за рефераты, в которых нет информации о проблематике работы и ее месте в контексте других работ по исследуемой теме.

Критерии оценки выполнения контрольных заданий по теоретическим основам дисциплины

Оценка **«отлично»** - Ответ полный и правильный на основании изученной теории; материал изложен в необходимой логической последовательности, грамотный научный язык; ответ самостоятельный.

Оценка **«хорошо»** - Ответ полный и правильный на основании изученной теории; материал изложен в необходимой логической последовательности при этом допущены две-три несущественные ошибки, исправленные по требованию преподавателя.

Оценка **«удовлетворительно»** - Ответ полный, но при этом допущена существенная ошибка или неполный, несвязный ответ.

Оценка **«неудовлетворительно»** - Ответ обнаруживает непонимание студентом основного содержания учебного материала или допущены существенные ошибки, которые не могут быть исправлены при наводящих вопросах преподавателя.

Критерии оценки выполнения практических контрольных заданий

Оценка **«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.

Оценка **«не зачтено» -** Ответ обнаруживает непонимание студентом основного содержания учебного материала или допущены существенные ошибки, которые не могут быть исправлены при наводящих вопросах преподавателя.

Критерии устного ответа студента при опросе на зачете

Оценка «отлично» выставляется, когда студент глубоко и прочно усвоил весь программный материал, исчерпывающе, последовательно, грамотно и логически стройно его излагает, не затрудняется с ответом при видоизменении задания, свободно справляется с ситуационными заданиями, правильно обосновывает принятые решения, умеет самостоятельно обобщать и излагать материал, не допуская ошибок.

Оценка «хорошо» выставляется, если студент твердо знает программный материал, грамотно и по существу излагает его, не допускает существенных неточностей в ответе на вопрос, может правильно применять теоретические положения и владеет необходимыми умениями и навыками при анализе информации.

Оценка «удовлетворительно» выставляется в том случае, при котором студент освоил только основной материал, но не знает отдельных деталей, допускает неточности, недостаточно правильные формулировки, нарушает последовательность в изложении программного материала и испытывает затруднения в выполнении анализа информации.

Оценка «неудовлетворительно» выставляется студенту, в ответе которого обнаружились существенные пробелы в знании основного содержания учебной программы дисциплины и / или неумение использовать полученные знания.

5.3 Типовые контрольные задания или иные материалы, необходимые для оценки результатов обучения и для контроля формирования компетенции

Примерные контрольные задания по теоретическим основам дисциплины 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.

для оценки сформированности компетенции ПК 11

8. Создание пользовательского интерфейса.
9. Принципы разработки пользовательского интерфейса.
10. Работа с формами.
11. Применение элементов управления и компонентов.
12. Меню.
13. Проверка данных, вводимых пользователем.
14. Приложение Virtual Doughnut Factory.

Примерные практические контрольные задания по дисциплине 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. Создатьиерархиюклассов 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. Дляклассов Person-Student-Teacher реализоватьиоттестировать 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 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.

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.

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.

5.7. 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 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.

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.

Примеррешения 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

Примерная тематика учебно-исследовательских реферативных работ 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#

для оценки сформированности компетенции ПК 11

4. Графический интерфейс языка C#
5. Создание графического интерфейса при помощи классов пакета JFC Swing
6. Объектно-ориентированное программирование

Примерные тестовые задания для оценки сформированности компетенции ПК 8

Теста № 1

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

A. Double

B. StringBuilder

C. Exception

D. Всетипы, порожденныеот 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);

Теста № 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

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.

Примерные контрольные работы
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.

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

Контрольные вопросы для промежуточной аттестации (к зачету)

Вопрос	Код компетенции
1. История создания и развития .NET Framework.	ПК-8
2. Инфраструктура .NET Framework и общеязыковая исполняющая среда.	ПК-11
3. Библиотека базовых классов .NET.	ПК-8
4. Классы и структуры.	ПК-11
5. Методы.	ПК-8
6. Область видимости и уровни доступа.	ПК-11
7. Сбор мусора.	ПК-8

8. Применение классов и демонстрация сбора мусора.	ПК-11
9. Создание пользовательского интерфейса.	ПК-8
10. Принципы разработки пользовательского интерфейса.	ПК-11
11. Работа с формами.	ПК-8
12. Применение элементов управления и компонентов.	ПК-11
13. Меню.	ПК-8
14. Проверка данных, вводимых пользователем.	ПК-11
15. Приложение Virtual Doughnut Factory.	ПК-8

6. Учебно-методическое и информационное обеспечение дисциплины

а) основная литература

1. Подбельский, В. В. Программирование. Базовый курс C# : учебник для бакалавриата и специалитета / В. В. Подбельский. — Москва : Издательство Юрайт, 2019. — 369 с. — (Бакалавр и специалист). — ISBN 978-5-534-10616-9. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/book/programmirovanie-bazovyy-kurs-s-439068

2. Зыков, С. В. Программирование. Объектно-ориентированный подход : учебник и практикум для академического бакалавриата / С. В. Зыков. — Москва : Издательство Юрайт, 2019. — 155 с. — (Бакалавр. Академический курс). — ISBN 978-5-534-00850-0. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/book/programmirovanie-obektno-orientirovannyy-podhod-434106

3. Казанский, А. А. Программирование на visual C# 2013 : учебное пособие для прикладного бакалавриата / А. А. Казанский. — Москва : Издательство Юрайт, 2019. — 191 с. — (Бакалавр. Прикладной курс). — ISBN 978-5-534-00592-9. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/book/programmirovanie-na-visual-c-2013-434085

4. Тузовский, А. Ф. Объектно-ориентированное программирование : учебное пособие для прикладного бакалавриата / А. Ф. Тузовский. — Москва : Издательство Юрайт, 2019. — 206 с. — (Университеты России). — ISBN 978-5-534-00849-4. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/book/obektno-orientirovannoe-programmirovanie-434045

б) дополнительная литература

1. **Гуриков С.Р.** Введение в программирование на языке Visual C# : учеб. пособие / С.Р. Гуриков. — М. : ФОРУМ : ИНФРА-М, 2017. — 447 с. – ЭБС Znanium.com: [Электронный ресурс]. – Адрес доступа: http://znanium.com/catalog.php?bookinfo=752394

2. **Подбельский В.В.** Язык C# Базовый курс [Электронный ресурс] : учеб. пособие / В.В. Подбельский. - М. : Финансы и статистика, 2011. – ЭБС «Консультант студента»: [Электронный ресурс]. – Адрес доступа: http://www.studentlibrary.ru/book/ISBN9785279034970.html

в) программное обеспечение и Интернет-ресурсы:

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

MathSciNet: информационно-библиографическая и реферативная база данных по математике, в т.ч. прикладной математике и статистике. Электронная версия Mathematical Reviews. Адрес доступа: http://www.ams.org/mathscinet

Math-Net.Ru: Общероссийский математический портал. Адрес доступа: http://www.mathnet.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)

Помещения представляют собой учебные аудитории для проведения учебных занятий, предусмотренных программой, оснащенные оборудованием и техническими средствами обучения: ноутбук, проектор, экран.

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

Программа дисциплины **Разработка интерфейса с использованием языка C#** составлена в соответствии с образовательным стандартом высшего образования (ОС ННГУ) по направлению подготовки 09.03.03 Прикладная информатика (уровень бакалавриата) (приказ ННГУ от 17.05.2023 года № 06.49-04-0214/23)

Автор(ы):

Старший преподаватель Сазанов А.А.

Рецензент (ы):

д.т.н., профессор Ямпурин Н.П.

Кафедра математики, физики и информатики

д.п.н., доцент Фролов И.В.

Программа одобрена на заседании методической комиссии от 24.05.2023 года, протокол № 5

Председатель МК факультета естественных и математических наук
к.п.н., доцент Володин А.М.

П.6. а) СОГЛАСОВАНО:

Заведующий библиотекой Федосеева Т.А.