Федеральное государственное бюджетное образовательное учреждение высшего образования
«РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА И ГОСУДАРСТВЕННОЙ СЛУЖБЫ ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»

Северо-Западный институт управления – филиал РАНХиГС

Кафедра бизнес-информатики
(наименование кафедры)

УТВЕРЖДЕНО
Директор СЗИУ РАНХиГС
А.Д.Хлутков

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ
Б1.О.11 Программирование

(индекс, наименование практики (научно-исследовательской работы), в соответствии с учебным планом)

38.03.05 Бизнес-информатика
(код,наименование направления подготовки)

«Бизнес-аналитика»
(профиль)

бакалавр
(квалификация)

очная
(форма обучения)

Год набора – 2024

Санкт-Петербург, 2024 г.

Автор–составитель:
Доцент кафедры бизнес информатики Рассказов Владимир Александрович

Директор образовательной программы «Бизнес-информатика»
к.т.н, доцент Борисова Елена Юрьевна

РПД по дисциплине Б1.О.11. Программирование одобрена на заседании кафедры бизнес-информатики. Протокол от 04.07.2022г. №9

В новой редакции РПД одобрена на заседании кафедры бизнес-информатики. Протокол от 27.06.2024 г. № 10

СОДЕРЖАНИЕ

1. Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения программы

1.1. Дисциплина «Программирование» обеспечивает овладение следующими компетенциями:

Код компетенции	Наименование компетенции	Код этапа освоения компетенции	Наименование этапа освоения компетенции
ОПК-3	Способность управлять процессами создания и использования продуктов и услуг в сфере информационно-коммуникационных технологий, в том числе разрабатывать алгоритмы и программы для их практической реализации	ОПК-3.1	Способность разрабатывать алгоритмы и программы, проектирует базы данных с целью использования на практике основных методов управления процессами создания продуктов и услуг ИКТ

В результате освоения дисциплины у студентов должны быть сформированы:

Таблица 1.1

ОТФ/ТФ (при наличии профстандарта)/ профессиональные действия	Код этапа освоения компетенции	Результаты обучения
Выполнение работ и управление работами по созданию (модификации) и сопровождению ИС, автоматизирующих задачи организационного управления и бизнес-процессы	ОПК-3.1	на уровне знаний: – основные понятия и методы программирования, классификацию языков программирования, парадигмы программирования; – основные свойства алгоритмов, формы записи алгоритмов, базовые алгоритмические структуры; – способы описания синтаксиса языков программирования, основные синтаксические конструкции, основные структурами данных и типовые методы обработки этих структур;
		на уровне умений: – разрабатывать алгоритмы; – реализовывать алгоритмы на языке высокого уровня; – описывать основные структуры данных; – работать в инструментальных средах программирования; – кодировать на языках программирования, тестировать результаты кодирования, выполнять отладку программ.

2. Объем и место дисциплины в структуре ОП ВО

Объем дисциплины

Общая трудоемкость дисциплины составляет 7 зачетных единицы /252 академ. часов.

Дисциплина реализуется частично с применением дистанционных образовательных технологий (далее – ДОТ).

Доступ к системе дистанционных образовательных технологий осуществляется каждым обучающимся самостоятельно с любого устройства на портале: https://lms.ranepa.ru/. Пароль и логин к личному кабинету / профилю предоставляется студенту в деканате.

Таблица 2

Вид работы	**Трудоемкость в акад. часах ауд./ЭО, ДОТ**	**Трудоемкость в астрон. часах ауд./ЭО, ДОТ**
Общая трудоемкость	**252**	**189**
Контактная работа с преподавателем	***78***	***59***
Лекции	32	24
Практические занятия	44	33
Лабораторные занятия		
Самостоятельная работа	**138**	**104**
Контроль	***36***	***27***
Формы текущего контроля		
Форма промежуточной аттестации	***Курсовая работа, экзамен***	

Место дисциплины в структуре ОП ВО

Дисциплина Б1.О.12 «Программирование» относится к числу обязательных дисциплин базовой части учебного плана по направлению «Бизнес-информатика» 38.03.05. Преподавание дисциплины «Программирование» основано на дисциплине Б1.О.11 «Основы информатики». В свою очередь она создаёт необходимые предпосылки для освоения программ таких дисциплин, как Б1.О.13 «Базы данных», Б1.О.15 «Объектно–ориентированный. анализ и программирование», Б1.В.ДВ.07.01 «Сетевые технологии» и ряда дисциплин по выбору студента.

Объем ЭК (в составе дисциплины): количество академических часов, выделенных на самостоятельную работу обучающихся: всего по ЭК - 138_а.ч., из них : 138- количество академических часов, выделенных на практикоориентированные задания и текущий контроль успеваемости : всего по ЭК – 138 а.ч. Количество академических часов, выделенных на самостоятельную работу обучающихся в рамках ЭК - 138 а.ч.

Дисциплина изучается во 1-м и 2-м семестрах 1-го курса.

Формой промежуточной аттестации в соответствии с учебным планом в первом семестре является курсовая работа, во втором семестре экзамен.

3. Содержание и структура дисциплины

Таблица 3

№ п/п	**Наименование тем**	**Объем дисциплины, час.**						**Форма текущего контроля успеваемости, промежуточной аттестации**
		Всего	**Контактная работа обучающихся с преподавателем по видам учебных занятий**			**СР (ЭК)**		
			Л	**ПЗ**	**КСР**	СРО	СП	
Тема 1	Введение в теорию алгоритмов. Основные	34	4	4		22	4	ПКЗ, ЗР, Т*

	модели структур данных.							
Тема 2	Основные конструкции языка программирования *C#.*	34	4	4		22	4	ПКЗ, ЗР, Т*
Тема 3	Основы объектно-ориентированного программирования на *C#.*	38	4	8		22	4	ПКЗ, ЗР, Т*
Тема 4	Введение в создание приложений с графическим пользовательским интерфейсом на *C#.*	38	4	8		22	4	ПКЗ, ЗР, Т*
Промежуточная аттестация (2 семестр)								Курсовая работа
Тема 4	Введение в создание приложений с графическим пользовательским интерфейсом на *C#.*	70	16	20		26	8	ПКЗ, ЗР, Т*
Контроль		36						
Промежуточная аттестация (2 семестр)		2			2*			Экзамен
Всего (акад./астр. часы):		252	32	44	2*	114	24	

2* - консультация

Т – тестирование;

ПКЗ – выполнение практического контрольного задания;

ЗР – защита выполненного задания.

СР – самостоятельная работа, осуществляемая без участия педагогических работников организации и (или) лиц, привлекаемых организацией к реализации образовательных программ на иных условиях;

СП – самопроверка;

СРО – самостоятельная работа обучающегося

Применяемые на занятиях формы интерактивной работы:

а) Лекция-визуализация - передача преподавателем информации студентам сопровождается показом различных рисунков, структурно-логических схем, диаграмм, использование среды разработки;

В процессе освоения данной учебной дисциплины используются следующие образовательные технологии:

Лекционные занятия:

- сопровождаются демонстрацией слайдов, подготовленных в среде *MS PowerPoint*;
- сопровождаются демонстрацией приёмов работы в изучаемых средах программирования;
- сопровождаются элементами дискуссии по рассматриваемым вопросам.

Практические занятия выполняются в компьютерных классах:

- направлены на закрепление полученных теоретических знаний;
- включают анализ полученных результатов и способов его достижения;
- сопровождаются элементами дискуссии;
- завершается занятие защитой работы.

Для лекционных и практических занятий используются мультимедийное обеспечение, современное компьютерное оснащение. В аудиториях наличие локальной вычислительной

сети института и глобальной сети Интернет, лицензионное программное обеспечение.

Содержание дисциплины

Тема 1. Введение в теорию алгоритмов. Основные модели структур данных.

Информационные процессы. Определение и свойства алгоритма. Вычислимые функции и частично-рекурсивные функции. Способы записи и типы алгоритмов. Технические средства реализации информационных систем. Тенденции развития ЭВМ, классификация ЭВМ. Программные средства реализации информационных процессов. Классификация программного обеспечения. Парадигмы программирования. Способы выполнение программ.

Вычислительная сложности алгоритмов. Оценка сложности алгоритмических конструкций. Асимптотический анализ трудоемкости алгоритмов. Типовые функции временной оценки сложности алгоритмов. Составные классы сложности. Анализ сложности алгоритмов на языке *C#.* Оценка времени выполнения алгоритмов вручную.

Концепция абстрактных типов данных. Абстрактный тип данных «Вектор». Абстрактный тип данных «Список». Абстрактный тип данных «стек». Абстрактный тип данных «Очередь». Абстрактный тип данных «Дек».

Тема 2. Основные конструкции языка программирования C#.

Введение в *.NET*. Версии *.NET Runtime* и *.NET SDK*. Общеязыковые исполняющие среды, библиотеки базовых классов и исполняющие среды. Краткая история языка *C#*.

Идентификаторы и ключевые слова, литералы, знаки пунктуации. Основы типов, классификация типов. Отображение вывода пользователю и получение пользовательского ввода в консольных приложениях.

Переменные. Определенное присваивание и стандартные значения. Числовые типы, числовые преобразования, арифметические операции. Булевский тип, булевские преобразования, булевы операции. Строки и символьные типы, основные символьные преобразования и строковые операции. Операции для работы со значениями *null*. Приведение и преобразование типов. Явное и неявное приведение типов, преобразование с помощью *System.Convert*. Объявление неявно типизированных переменных.

Массивы, стандартная инициализация элементов, индексы и диапазоны. Многомерные и ступенчатые массивы.

Директива *using*, Директива *using static*. Правила внутри пространства имен. Назначение псевдонимов типам и пространствам имен. Дополнительные возможности пространств имен.

Управляющие операторы. Оператор *if*, вложенные операторы *if*, конструкция *if-else-if*. Оператор *switch*, вложенные операторы *switch*. Оператор цикла *for*, оператор цикла *while*, оператор цикла *do-while*, оператор цикла *foreach*. Оператора *break, continue, return, goto*.

Тема 3. Основы объектно-ориентированного программирования на C#.

Общая форма определения класса Создание объектов. Переменные ссылочного типа и присваивание.

Методы. Возврат из метода. Использование параметров. Конструкторы и деструкторы, ключевое слово *this*. Модификаторы доступа. Организация закрытого и открытого доступа. Применение ключевого слова *static*. Статические классы и статические конструкторы.

Перегрузка методов. Перегрузка конструкторов. Инициализаторы объектов. Необязательные аргументы и именованные аргументы.

Перегрузка операторов. Перегрузка бинарных операторов. Перегрузка унарных операторов. Перегрузка операторов отношения. Перегрузка логических операторов. Индексаторы и перегрузка индексаторов.

Свойства. Автоматически реализуемые свойства. Применение инициализаторов объектов в свойствах. Применение модификаторов доступа в аксессорах. Применение индексаторов и свойств.

Применение абстрактных классов. Упаковка и распаковка. Интерфейсы, структуры и перечисления. Реализация интерфейсов и применение интерфейсных ссылок.

Основы наследования и полиморфизма. Доступ к членам базового класса. Наследование и сокрытие имен, применение ключевого слова *base* для доступа к скрытому имени. Создание многоуровневой иерархии классов и порядок вызова конструкторов. Ссылки на базовый класс и объекты производных классов. Предотвращение наследования с помощью ключевого слова *sealed*. Класс *object* как универсальный тип данных. Виртуальные методы и их переопределение.

Обобщения и обобщенные типы. Обобщенные методы. Объявление параметров типа/ Операция *typeof* и несвязанные обобщенные типы. Стандартное значение для параметра обобщенного типа. Ограничения обобщений. Создание подклассов для обобщенных типов. Ковариантность и контравариантность.

Тема 4. Введение в создание приложений с графическим пользовательским интерфейсом.

Основы *WPF*. Язык разметки *XAML*. Процесс компоновки пользовательского интерфейса. Простая компоновка с помощью *StackPanel*. Элемент *Border*. Панели *WrapPanel* и *DockPanel*. Вложение контейнеров компоновки. Панель *Grid*. Разделенные окна. Координатная компоновка с помощью *Canvas*. Контейнер *InkCanvas*.

Общие сведения о элементах управления, базовый класс *Control*. Элементы управления содержимым *ContentControl*: Метки, Кнопки (Класс *Button, RepeatButton, ToggleButton*, класс *CheckBox, RadioButton*), всплывающие подсказки, контекстное меню (Класс *Popup*), специализированные контейнеры. Элементы управления содержимым с заголовками. Текстовые элементы управления. Элементы управления списками *ItemsControl*. Элементы управления, основанные на диапазонах значений. Элементы управления датами.

Общие сведения о ресурсах. Коллекция ресурсов. Статические и динамические ресурсы. Доступ к ресурсам в коде. Ресурсы приложения. Создание и использование словаря ресурсов.

Общие сведения о командах. Модель команд *WPF*. Выполнение команд. Источники команд. Привязки команд. Использование одной и той же команды в разных местах. Отслеживание и отмена команд.

Основные сведения о стилях. Создание объекта стиля, установка свойств, присоединение обработчиков событий. Автоматическое применение стилей по типу. Триггеры, простой триггер, триггер события. Модель поведений, создание поведения.

Фигуры, кисти и трансформации. Классы фигур. Масштабирование фигур. Кисти, *SolidColorBrush, LinearGradientBrush, RadialGradientBrush, ImageBrush*.

Трансформация фигур и элементов. Основы анимации. Анимация на основе таймера. Анимация на основе свойств. Классы анимации.

Шаблоны элементов управления. Логические и визуальные деревья. Создание шаблонов элементов управления. Пользовательские элементы управления.

Привязка данных. Привязка пользовательских объектов к базе данных. Привязка к коллекции объектов.

4. Материалы текущего контроля успеваемости обучающихся и фонд оценочных средств промежуточной аттестации по дисциплине

4.1. Формы и методы текущего контроля успеваемости обучающихся и промежуточной аттестации.

4.1.1. В ходе реализации дисциплины «Программирование» используются следующие методы текущего контроля успеваемости обучающихся

Таблица 4.1

Тема (раздел)	Формы (методы) текущего контроля успеваемости
Тема 1. Введение в теорию алгоритмов. Основные модели структур данных.	Защита задания, тестирование
Тема 2. Основные конструкции языка программирования C#. Создание и отладки программ с использованием инструментальных сред программирования.	Защита задания, тестирование
Тема 3. Основы объектно-ориентированного программирования на *C#*.	Защита задания, тестирование
Тема 4. Введение в создание приложений с графическим пользовательским интерфейсом.	Защита задания, защита курсового проекта, тестирование

4.1.2. **Зачет и экзамен проводятся с применением следующих методов (средств):**

Зачет и экзамен проводится в компьютерном классе. Во время экзамена проверяются этапы освоения компетенций ОПК ОС-4.1 и ОПК-3.1.

Во время проверки сформированности этапа компетенции ОПК ОС-4.1 оценка правильности ответов на поставленные вопросы, степени их полноты и обоснованности.

Во время проверки сформированности этапа ОПК-3.1 оценка правильности ответов на поставленные вопросы, степени их полноты и обоснованности, презентация модели модуля информационной системы.

Промежуточная аттестация может проводиться устно в ДОТ/письменно с прокторингом/ тестирование с прокторингом. Для успешного освоения курса учащемуся рекомендуется ознакомиться с литературой, размещенной в разделе 6, и материалами, выложенными в ДОТ.

4. 2. **Материалы текущего контроля успеваемости обучающихся.**

4.2.1. Типовые оценочные материалы по теме 1

4.2.1.1. Примеры типовых заданий для практических работ

Тема 1. Введение в теорию алгоритмов. Основные модели структур данных.

Практическое контрольное задание 1. Основы алгоритмизации

Рассматривать в деталях способы описания алгоритма. Сделать обзор стандартов описания блок-схем. Разработать блок схемы линейного алгоритма. Разработать блок схемы разветвляющего алгоритма. Выполнить трассировку алгоритма. Разработать блок схему цикла с параметром. Выполнить трассировку алгоритма. Разработать блок схемы итерационного цикла. Выполнить трассировку алгоритма.

Тема 2. Основные конструкции языка программирования *C#*

Практическое контрольное задание 2. Знакомство с Visual Studio, Visual Studio Code. Управление потоком исполнения, преобразование типов.

Рассматривать в деталях Visual Studio. Управление несколькими проектами в Visual Studio. Папки и файлы, созданные компилятором. Написание, компиляция и запуск кода, основные сочетания клавиш Visual Studio. Отладка в процессе разработки. Преднамеренное добавление ошибок в код. Установка и настройка точек останова. Пошаговое выполнение кода. Разработать консольное приложение, содержащее методы. реализующее арифметические, логические и строковые операции. Показать работу разработанных методов на примере.

Рассматривать в деталях организацию ветвления в программе. Разработать методы, использующие полное и неполное ветвление. Разработать программу, использующую конструкцию выбор *switch …case*. Исследовать структуры ветвления и выбора с точки зрения повышения эффективности работы алгоритма. Разработать самостоятельно программу с использованием конструкций выбора по вариантам в таблице 4.2

Таблица 4.2

Номер варианта	Функция
1.	$Z=\begin{cases}10{,}5x^2+1, \text{если } x<0;\\ 5(x+1)^3, \text{если } 0\le x<1\\ \sqrt{x^2+1}, \text{если } x\ge 1\end{cases}$
2.	$Z=\begin{cases}\cos(x-1), \text{если } -2\pi<x\le 0;\\ \cos x+1, \text{если } 0<x<\pi;\\ \sin(x+0{,}5), \text{если } \pi\le x\le 2\pi\end{cases}$
3.	$Z=\begin{cases}x^4-12, \text{если} x\le -1;\\ (2x+1)^3, \text{если } -1<x<1;\\ \sqrt{\dfrac{x+11}{x}}, \text{если } x\ge 1\end{cases}$
4.	$Z=\begin{cases}x^2+5x, \text{если } x<1;\\ (x+3x)^2, \text{если } x=1\\ (x-2x^2), \text{если} x>1\end{cases}$
5.	$Z=\begin{cases}0{,}5x^3+1, \text{если } x\le -10;\\ (x+1)^2, \text{если } -10<x<10;\\ \sqrt{x+10}, \text{если } x\ge 10\end{cases}$
6.	$Z=\begin{cases}x^4-12, \text{если } x\le -1;\\ (2x+1)^3, \text{если } -1<x<1;\\ \sqrt{\dfrac{x+11}{x}}, \text{если} x\ge 1\end{cases}$
7.	$Z=\begin{cases}15x+1, \text{если } x<0;\\ 5(x+1)^2, \text{если } 0<x<1;\\ x^3-1, \text{если} x-1\end{cases}$
8.	$Z=\begin{cases}x2+1, \text{если} x<1;\\ (x+2)^2 \text{ если}, x-1;\\ (x^3+2x), \text{если} x>1\end{cases}$
9.	$Z=\begin{cases}sin3x, \text{если } -\pi<x<0\\ cos(x+1), \text{если } 0\le x\le \pi/2\\ sin\,2x, \text{если } \pi/2<x\le\pi\end{cases}$
10.	$Z=\begin{cases}0{,}5x^3+1, \text{если } x\le -10\\ (x+1)^2, \text{если } -10<x<10\\ \sqrt{x+10}, \text{если } x\ge 10\end{cases}$
11.	$Z=\begin{cases}\dfrac{1}{x+1}, \text{если } x<0;\\ -x^2, \text{если } x\ge 0\end{cases}$

12.	$Z = \begin{cases} \frac{1}{x^2}, \text{если } x < -1; \\ x^2, \text{если } -1 \le x \le 2; \\ 4, \text{если } x > 2 \end{cases}$
13.	$Z = \begin{cases} x^2, \text{если } x < 0; \\ -x^4, \text{если } x > 0 \end{cases}$
14.	$Z = \begin{cases} -x - 1, \text{если } x < -1; \\ x + 1, \text{если } -1 \le x \le 0; \\ -x + 1, \text{если } 0 < x \le 1; \\ x - 1, \text{если } x > 1 \end{cases}$
15.	$Z = \begin{cases} -3x - 1, \text{если } x \le -2; \\ -x + 6, \text{если } -2 < x < 1; \\ x + 4, \text{если } 1 \le x \le 3; \\ 3x - 2, \text{если } x > 3 \end{cases}$
16.	$Z = \begin{cases} -1, \text{если } x < -1; \\ x, \text{если } -1 \le x \le 1; \\ 1, \text{если } x > 1 \end{cases}$
17.	$Z = \begin{cases} 1 - x^2, \text{если } x < 0; \\ 2 - x^3, \text{если } x \ge 0 \end{cases}$
18.	$Z = \begin{cases} -x, \text{если } x < -1; \\ 1, \text{если } -1 \le x \le 1; \\ x, \text{если } x > 1 \end{cases}$
19.	$Z = \begin{cases} -\frac{1}{x^2} + 1, \text{если } x < -1; \\ x^2 + 1, \text{если } -1 \le x \le 2; \\ 5, \text{если } x > 2 \end{cases}$
20.	$Z = \begin{cases} x + 1, \text{если} x < -1; \\ -x - 1, \text{если } -1 \le x \le 0; \\ x - 1, \text{если } 0 < x \le 1; \\ -x + 1, \text{если } x > 1 \end{cases}$

Практическое контрольное задание 3. Циклические конструкции и массивы.

Сделать обзор циклических операторов. Рассмотреть в деталях использование арифметического цикла for. Разработать программу, использующую цикл *for*.

Сделать обзор итерационных циклов. Разработать программу с использованием цикла *while*. Исследовать использование цикла *do…while*. Разработать программы с использованием цикла *while*, используя постусловие и предусловие по вариантам в таблице 4.3:

Таблица 4.3

Номер варианта	Функция	Номер варианта	Функция
1	$A = \sum_{k=1}^{5} \sqrt[4]{1000k + x^4}$	3	$A = \prod_{k=1}^{5} sin(x + 3k)$
2	$A = \prod_{k=1}^{5} (x^2 - k)$	4	$A = \sum_{k=1}^{5} \sqrt{(k^2 + 2x)^3}$
5	$A = \prod_{k=1}^{5} \frac{\cos x}{2k}$	13	$A = \prod_{k=1}^{5} \sqrt{(xk + 2)^3}$

6	$A = \sum_{k=1}^{5}(x^4 + 4k)$	14	$A = \sum_{k=1}^{5}\frac{x^k}{k}$
7	$\mathrm{A} = \prod_{k=1}^{5} e^x + kx$	15	$A = \sum_{k=1}^{5} tg(x+k)$
8	$A = \sum_{k=1}^{5}\frac{(kx-k)}{k^2}$	16	$A = \sum_{i=1}^{5}\frac{1+2^{\mathrm{k}}}{x+1}$
9	$A = \sum_{k=1}^{5}\frac{x+1}{k^3}$	17	$A = \prod_{k=1}^{5}(xk + 1)^2$
10	$A = \prod_{k=1}^{5}\frac{\sin x}{2k+1}$	18	$A = \sum_{k=1}^{5}\frac{x(-1)^{k+1}}{k(k+1)}$
11	$A = \sum_{k=1}^{5}\frac{1}{(2k+1)^2}$	19	$A = \sum_{k=1}^{5}\frac{x(-1)^k}{k(2k+1)}$
12	$A = \sum_{k=1}^{5}\frac{x(-1)^{k+1}}{(k+1)(k+2)}$	20	$A = \sum_{k=1}^{5}\frac{x(-1)^{k+1}}{(2k-5)}$

Дать определение массива. Сделать обзор различных видов массива. Исследовать возможности массивов. Разработать программу, выполняющую заполнение одномерного массива, вычисляющую сумму элементов массива.

Практическое контрольное задание 4. Обработка ошибок и исключительных ситуаций.

Составить программу деления вещественных чисел. Программа должна выполнять обработку ошибок и исключений с использованием конструкции *try ... catch*, и выдавать следующие сообщения о характере ошибки:

- не введено число (с помощью оператора условия);
- введено слишком длинное число (с помощью оператора условия);
- деление на ноль;
- ошибка преобразования.

Составить программу, содержащую массив и метод возвращающий элемент массива по его номеру. Программа должна содержать обработчик исключения *IndexOutOfRangeException,* выводящий сообщение пользователю «Нарушение границ массива».

Тема 3. Основы объектно-ориентированного программирования на *C#.*

Практическое контрольное задание 5. Разработка простого объектно-ориентированного приложения на *C#.*

Разработать консольное приложение, содержащее класс *Dice,* имеющий открытое свойство целого типа *Score* (число очков от 1 до 6), метод *roll* не имеющий параметров и изменяющий свойство *Score* на случайное значение в диапазоне от 1 до 6. Создать класс *Player* имеющий открытое свойство строкового типа *Name*, закрытое свойство *Cube* типа *Dice,* открытое свойство статического целого типа *TourId*, открытое свойство *Scores* типа массив целых чисел, метод *sum* – возвращающий сумму чисел массива *Scores*, метод *doStep* – добавляющий число очков очередного хода к массиву *Scores* и увеличивающий счётчик туров игры *TourId*. Главный класс приложения *Main* должен содержать объект типа *Player*, цикл десяти ходов игрока, вывод в консоль суммы очков после десяти ходов.

Практическое контрольное задание 6. Использование абстрактных классов и интерфейсов в объектно-ориентированных приложениях на *C#.*

Разработать консольное приложение, содержащее иерархию классов геометрических фигур: треугольник, квадрат, прямоугольник, многоугольник. Базовый классов фигу является

абстрактный класс *Section* - содержащий открытое свойство *Side* типа массив чисел с плавающей запятой, метод *perim* – возвращающий сумму элементов массива *Side*. Дополнительно в классах квадрат и прямоугольник реализовать метод *squre*, возвращающий площадь данных фигур.

Создайте интерфейс *Human*, включающий декларацию метода *Name* (возвращает строку). Создайте два класса на основе этого интерфейса: *Student* и *Teacher*. Конкретизируйте в классе *Student* метод *Name* для возвращения имени студента, а в классе *Teacher* – фамилии преподавателя.

Создайте класс с двумя методами с одинаковой сигнатурой – через аргументы передаётся массив с Фамилией, Именем и Отчеством сотрудника, а возвращается строка – в первом методе «Фамилия И.О.», во втором – «Фамилия Имя». Создайте делегат для ссылки на эти методы и продемонстрируйте их работоспособность.

Практическое контрольное задание 7. Запись в потоки, сериализация и десириализация объектов, графов объектов.

Создайте класс *Note*, содержащий открытое свойство *Id* – идентификатор записки, открытое свойства *Text* (текст записки), *CreateDate* (дата создания записки), *FinalDate* (конечная дата исполнения), *IsDelay* (признак просрочки). Создайте класс *NoteBook* содержащий свойство *Notes* - коллекция объектов класса *Note*, свойство *CurrentId* – идентификатор текущей записки, методы *add* и *remove* – добавление новой и удаление текущей записки, метод *edit* – корректура текущей записки, метод *Show* – вывести в консоль свойства текущей записки, метод *save* – сохранение (сериализация: *XML*-сериализация или *JSON*-сериализация) коллекции записок *Notes* в файл (базу данных), метод *open* – чтение (десериализация: *XML*-десериализация или *JSON*-десериализация) коллекции записок *Notes* из файла (базы данных). Задание выполнить используя шаблон консольного приложения.

Тема 4. Введение в создание приложений с графическим пользовательским интерфейсом на C#.

Практическое контрольное задание 8. Разработка простого приложения с графическим пользовательским интерфейсом.

Используя разработку контрольного задания 7, создать приложение с графическим пользовательским интерфейсом, отображающее свойства выбранной пользователем записки. Дополнительно для класса *NoteBook* реализовать методы *GetNextNote* – отобразить следующую записку, *GetPreviousNote* – отобразить предыдущую записку.

Практическое контрольное задание 9. Разработка простого текстового редактора.

На основе элемента управления *TexBox*, меню, панелей элементов создать приложение - простой текстовый редактор. Редактор должен выполнять простые команды *Save, New, Copy, Paste, Cut*, иметь возможность отмены выполненных команд.

Практическое контрольное задание 10. Простое бизнес-приложение, журнал отдела кадров.

Разрабатываемое приложение должно обеспечивать хранение и обработку следующих данных по сотрудникам компании: фамилия; имя; отчество; должность; дата рождения; телефон; адрес электронной почты.

Функции приложения:

– просмотр данных по сотрудникам;

– ввод данных по новому сотруднику;

– редактирование данных по сотруднику;

– удаление данных по сотруднику;

– поиск данных по сотруднику.

При реализации приложения требуется использовать шаблон приложения *MVVM* и привязку данных.

Практическое контрольное задание 10. Разработка приложения с графическим пользовательским интерфейсом и привязкой к коллекции данных.

Создать приложение с графическим пользовательским интерфейсом *Store* (Магазин). Приложение из сохраненной в файл коллекции (базы данных) выводит на экран пользователю список товаров (Имя товара), позволяет получить детальное описание каждого товара (номер модели, название модели, стоимость, описание, имеемое количество). Приложение позволяет добавлять новый вид товара, удалять существующие товары и корректировать данные существующих товарах.

4.2.1.2.Типовые вопросы для устного опроса

Тема 1. Введение в теорию алгоритмов. Основные модели структур данных.

1. Сделать обзор технологий программирования.
2. Сформулировать принципы структурного программирования.
3. Сформулировать принципы объектно-ориентированного программирования.
4. Сделать обзор языков программирования.
5. Сделать обзор области применения языков программирования.
6. Дать определение алгоритму.
7. Сделать обзор свойств алгоритма.
8. Перечислить способы описания алгоритма.
9. Перечислить основные алгоритмические структуры.

Тема 2. Основные конструкции языка программирования *C#*

1. Дать определение интегрированной среды разработки
2. Сделать обзор структуры ИСР *Visual Studio*.
3. Рассмотреть в деталях структуру оператора *if*.., оператора *switch*…*case*.
4. Сделать обзор типов ошибок.
5. Дать определение синтактическим ошибкам.
6. Дать определение логическим или алгоритмическим ошибкам.
7. Дать определение ошибкам времени выполнения.
8. Сделать обзор основных операторов, используемых при обработке исключительных ситуаций.
9. Перечислить различные виды циклов.
10. В чем отличие между *break* и *continue*.
11. Дать определение массива, коллекции.
12. Что обозначает ключевое слово *var*.
13. Что делает блок операторов *try-catch*.
14. Какие бывают массивы?
15. Что сделает программа, выполнив следующий код: *Console.WriteLine(«Hello, World!»)*?

Тема 3. Основы объектно-ориентированного программирования

1. Дать определение объекту.
2. Дать определение методу.
3. Дать определение свойству.
4. Что такое поле, свойство, методов класса? В чем различие свойства и поля?
5. В чем разница между автоматическими и полными свойствами?
6. В чем заключается принцип инкапсуляции?
7. В чем заключается принцип наследования?
8. Объясните порядок вызова конструкторов в иерархии наследования.
9. В чем заключается принцип полиморфизма?
10. Что произойдет, если в классе наследнике определить переопределить не виртуальный метод?
11. Как в пользовательских классах производится переопределение операций?

Тема 4. Введение в создание приложений с графическим пользовательским интерфейсом.

1. Для чего нужен язык *XAML*? Возможно ли создание только с помощью *XAML*?
2. Каким образом производится компоновка пользовательского интерфейса в *WPF*.

3. Как устанавливаются размеры элементов управления в *WPF*.
4. Какие контейнеры компоновки наиболее часто используют?
5. Как производит компоновку элементов контейнер *Grid?*
6. Как производит компоновку элементов контейнер *StackPanel?*
7. Что такое присоединенные свойства?
8. Что такое маршрутизируемые события и свойства зависимости?
9. Что такое привязка данных?
10. Какие существуют элементы управления содержимым?

4.2.1.3.Примеры тестовых заданий

Тема 1. Введение в теорию алгоритмов. Основные модели структур данных.

1. Определить принцип структурного программирования «разделяй и властвуй» , как

а) открытость программы для быстрых модификаций, поэтому она должна быть понятна и хорошо прокомментирована
б) рассмотрение всей программной системы как многоуровневой системы
в) реализацию некого алгоритма, который построен на определенной математической модели решения задачи
г) решение трудной задачи путем разделения ее на множество мелких, легко решаемых подзадач

2. Интерпретировать свойство алгоритма, подходящее под следующее утверждение: «Каждый шаг алгоритма должен быть четким»

а) Дискретность
б) Понятность
в) Определенность
г) Массовость

3. Интерпретировать компонент интегрированной системы программирования, предназначенный для перевода исходного текста программы в машинный код.

а) редактор связей
б) переводчик
в) построитель кода
г) транслятор

4. Определить понятие алгоритма, как

а) Последовательность действий, выполняемых пользователем
б) Точное и понятное предписание исполнителю совершить последовательность действий, направленных на решение поставленной задачи
в) Последовательное выполнение команд в процессоре
г) Метод в математике

5. Определить первый этап решения задачи на компьютере, как

а) Постановка задачи
б) Разработка алгоритма
в) Тестирование
г) Отладка

6. Определить каким способом представлен алгоритм?

```
uses crt,graph;
var dr,dv:integer;
p:pointer;
size:word;
BEGIN
dr:=detect;
initgraph(dr,dv,'d:\tp71\bgi');
setColor(7);
setfillstyle(1,13);
fillEllipse(100,100,50,50);
floodfill(100,100,7);
SetTextStyle(0,0,3);
setColor(10);
outtextXy(73,90,'MIR');
readkey;
End.
```

а) Графическим
б) Словесным
в) Псевдокодом
г) Программным

7. Перечислить, что входит в интегрированные системы программирования?

а) редактор, транслятор, компилятор, компоновщик
б) редактор, компилятор, компоновщик
в) редактор, транслятор, компоновщик
г) редактор, транслятор

8. Определить конечное значение переменной *F* , если начальное значении равно 5

```
если F>=0 то  F:=F*F
иначе F:=-F*3;
вывод  F;
```

будет выведено …

а) -25
б) -15
в) 15
г) 25

9. Определить количество итераций цикла

```
a := 24
b := 32
нц пока  b >= a
|      b := b - a
кц
```

тело цикла выполнится _____ раз(а).

а) 1
б) 2
в) 3
г) 4

10. Выбрать в чем заключается принципиальное отличие компилятора от интерпретатора. Компилятор …

а) осуществляет п оиск синтаксических ошибок в исходной программе
б) создает объектный модуль (код)
в) делает пошаговый анализ команд и выполнение исходной программы
г) осуществляет поиск семантических ошибок в исходной программе

11. Определить конечное значение переменной Z

```
x := -1; y := 1; z := 0
если (y - x) > 0
   то
     если z <> 0
        то z := z + 1
     все
     если y > 0
        то z := x + 1
     иначе z := 2 * y
     все
вывод z
```

а) 0
б) 1
в) 2
г) 3

12. Определить, что выводится при помощи переменной f

Дан массив целых чисел $\{X_i\}$, i=1,2,…N, N=10.

Данная программа

```
F:=0;
нц для i:=1 до n-1
   если x[i]=x[i+1] то
      f:=f+1;
   все
кц;
вывод   f
```

выводит …

а) Количество пар соседних элементов, расположенных не по возрастанию
б) Количество пар соседних элементов с одинаковыми значениями
в) Количество пар с неравными значениями
г) Все элементы с одинаковыми значениями

Тема 2. Основные конструкции языка программирования C#.

1. Какие циклы существуют в языке C#:
   а) `for, while`
   б) `for, while, do while, foreach`
   в) `for, while, do while`

2. Что обозначает ключевое слово var:
   а) Устраивает «войну» между программами
   б) Обозначает что переменная имеет явный тип данных
   в) Обозначает что переменная без явного типа данных

3. Какие типы переменных существуют:
   а) `int, char, bool, float, double`
   б) `int, char, bool, string`
   в) Оба варианта верны
   г) Нет верного ответа

4. Где правильно создана переменная:
   а) `$x = 10;`
   б) `char symbol = ‘A’;`
   в) `x = 0;`

5. Какой оператор возвращает значение из метода:
   а) end
   б) out
   в) return

6. В чем отличие между break и continue:
   а) Continue пропускает итерацию, break выходит из цикла
   б) Break используется в Switch case, a continue в циклах
   в) Continue работает только в циклах, break дополнительно в методах

7. Какие бывают циклы:
   а) Цикл, Форич, Двойной цикл, Многократный
   б) Большие и маленькие
   в) `for, while, do-while, foreach`

8. Что делает *try-catch*:
   а) Работает с файлами
   б) Работает с исключениями
   в) Работает с классами

9. Что такое цикл и для чего они нужны:
   а) Циклы нужны для многократного выполнения кода
   б) Циклы нужны для многократного запуска программы
   в) Циклы нужны для многократного размещения данных

10. Для чего можно использовать язык *C#*:
   а) Для создания веб сайтов
   б) Для создания программ под ПК
   в) Оба варианта верны
   г) Нет верного ответа

11. Какие бывают массивы:
   а) Одномерные и многомерные
   б) Резиновые и статичные
   в) Сложные и простые

12. Какой тип переменной используется в коде: int a = 5:
   а) Знаковое 64-бит целое
   б) Знаковое 8-бит целое
   в) Знаковое 32-бит целое

13. Что делает оператор «%»:
   а) Возвращает процент от суммы
   б) Возвращает остаток от деления
   в) Возвращает тригонометрическую функцию

14. Как называется оператор «?:»:
   а) Прямой оператор

б) Вопросительный
в) Тернарный оператор

15. Что сделает программа выполнив следующий код: Console.WriteLine(«Hello, World!»);
а) Напишет Hello, World!
б) Напишет на новой строчке Hello, World!
в) Удалит все значения с Hello, World!

16. Для чего нужны условные операторы:
а) Чтобы устанавливать условия пользователю
б) Для оптимизации программы
в) Для ветвления программы

17. Как сделать инкрементацию числа:
а) !=
б) ++
в) -

21. Чему равен d, если `int a = 0; int b = a++; int c = 0; int d = a+ b + c + 3;`
а) 4
б) False
в) 3

18. Как сделать декрементацию числа:
а) !=
б) --
в) %%

19. Чему будет равен c, если int a = 0; int c = —a;
а) Null
б) 1
в) -1

20. Как найти квадратный корень из числа x:
а) Sqrt(x)
б) Math.Sqrt(x)
в) Arifmetic.sqrt(x)

Тема 3. Основы объектно-ориентированного программирования.

1. Ключевое слово `sealed` можно применить только к ...
а) Полям
б) Интерфейсам
в) Методам
г) Локальным переменным
д) Классам

2. Какое ключевое слово используется в производном классе для вызова конструктора класса-предка?
а) class
б) base
в) inherited
г) this

3. Что будет напечатано в консоли?

```
class A
    {
        static A()
        {
            Console.WriteLine("Static Hello from A");
        }
        public A()
        {
            Console.WriteLine("Hello from A");
        }
    }
    class B
    {
        public static string x = "Hello";
        static B()
        {
            Console.WriteLine("Static Hello from B");
        }
        public B()
        {
            Console.WriteLine("Hello from B");
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            A a = new A();
            Console.WriteLine(B.x);
        }
    }
```

Варианты ответов:

а) Hello from A Static Hello from A Hello
б) Hello from A Hello
в) Static Hello from A Hello from A Static Hello from B Hello
г) Static Hello from A Hello from A Static Hello from B Hello from B Hello
д) Ошибка компиляции

4. Что верно относительно следующего фрагмента кода?

```
class A
{
   public virtual void m1() {   }
}
class B : A
{
   public override void m1() {   }
}
class C : B
{
   public override void m1()
   {
      /* программный код */
```

```
    }
}
```

Выберите все подходящие варианты:

а) Из класса `C` невозможно обратиться к методу `m1()` класса `A` для одного и того же объекта
б) Из класса `C` можно получить доступ к методу `m1()` класса `B` используя вызов `base.m1()`
в) Из класса `C` можно получить доступ к методу `m1()` класса `A` с помощью вызова `((A) this).m1()`
г) Из класса `C` можно получить доступ к методу `m1()` класса `A` с помощью вызова `base.base.m1()`
д) Ничего из вышеперечисленного

5. Какие утверждения об интерфейсах справедливы?

а) Интерфейсы поддерживают множественное наследование
б) Интерфейсы могут содержать поля
в) Интерфейсы могут содержать статические конструкторы
г) Интерфейсы унаследованы от *System.Object.*

6. Скомпилируется ли следующий код?

```
namespace A
{
    private class B
    {
        void f()
        { }
    }

    class Program
    {
        static void Main(string[] args)
        {
            B b = new B();
            b.f();
        }
    }
}
```

Да
Нет

Тема 4. Введение в создание приложений с графическим пользовательским интерфейсом.

1. Какой элемент используется для объединения ячеек в панели Grid?

а) `Grid.CombineCells`
б) `Merge`
в) `Grid.RowSpan и Grid.ColumnSpan`
г) `Combine`

2. Что представляет собой язык разметки XAML в WPF?

а) Расширяемый язык разметки приложений
б) Графический дизайнер
в) Язык программирования

3. Каким образом можно задать цвет фона кнопки в XAML?

а) <Button.BackgroundColor="#f0f0f0" />
б) <Button Background="#f0f0f0" />
в) <Button.Background>#f0f0f0 </Button.Background>

4. Как указать линейную градиентную заливку для фона кнопки в XAML?

а) <Button.GradientBrush>...</Button.GradientBrush>

б) <Button.Background.LinearGradient>
</Button.Background.LinearGradient>

в) <Button.Background><LinearGradientBrush>...
</LinearGradientBrush></Button.Background>

5. Какой из перечисленных диспетчеров компоновки поддерживает абсолютное позиционирование содержимого?

а) Canvas
б) StackPanel
в) DockPanel
г) WrapPanel

6. Какой из диспетчеров компоновки пристыковывает дочерние элементы к различным сторонам панели?

а) Grid
б) WrapPanel
в) DockPanel
г) StackPanel

7. Какой из диспетчеров компоновки располагает элементы в табличной сетке?

а) DockPanel
б) Grid
в) WrapPanel
г) StackPanel

4.3. Оценочные средства для промежуточной аттестации.

Таблица 4.2

Код компетенции	Наименование компетенции	Код этапа освоения компетенции	Наименование этапа освоения компетенции
ОПК-3	Способность управлять процессами создания и использования продуктов и услуг в сфере информационно-коммуникационных технологий, в том числе разрабатывать алгоритмы и программы для их практической реализации	ОПК-3.1	Способность разрабатывать алгоритмы и программы, проектирует базы данных с целью использования на практике основных методов управления процессами создания продуктов и услуг ИК

Таблица 4.3

Этап освоения компетенции	Показатель оценивания	Критерий оценивания
ОПК -3.1	1. Демонстрирует знание технологий программирования. 2. Показывает умение использовать знания по алгоритмизации при решении прикладных задач. 3. Демонстрирует знание основных синтактических конструкций языка программирования 4. Показывает умение разрабатывать приложения в *MS Visual Studio, VisualStudio Code.*	1. Продемонстрировано знание основных алгоритмических конструкций. 2. Продемонстрировано знание основных синтактических конструкций языка программирования. Правильно решены задачи. 3. Продемонстрировано умение разрабатывать приложения для заданной прикладной области в среде *MS Visual Studio, VisualStudio Code*.

Для оценки сформированности компетенций, знаний и умений, соответствующих данным компетенциям, используются контрольные вопросы, а также выполнение практических заданий.

Темы курсовых работ

1. Разработка модуля ИС для гостиницы
2. Разработка модуля ИС для общежития
3. Разработка модуля ИС для библиотеки
4. Разработка модуля ИС для агентства недвижимости
5. Разработка модуля ИС для туристического агентства
6. Разработка модуля ИС для страховой компании
7. Разработка модуля ИС для мебельного магазина
8. Разработка модуля ИС для страховой компании
9. Разработка модуля ИС для строительной компании
10. Разработка модуля ИС для определения сферы деятельности
11. Разработка модуля ИС для кафе
12. Разработка модуля ИС для автовокзала
13. Разработка модуля ИС для экскурсионного бюро
14. Разработка модуля ИС для расчета налогов индивидуальных налогоплательщиков
15. Разработка модуля ИС для спортивного клуба
16. Разработка модуля ИС «Расчет зарплаты»
17. Разработка модуля ИС «Расчет пенсии»
18. Разработка модуля ИС «Расчет военной пенсии»
19. Разработка модуля ИС «Страхование недвижимости»
20. Разработка модуля ИС «Страхование автомобиля»
21. Разработка модуля ИС «Кадровое агентство»
22. Разработка модуля ИС «Психологический тест»
23. Разработка модуля ИС «Тестирование знаний по иностранному языку»
24. Разработка теста остаточных знаний
25. Разработка модуля ИС «Кастинг на должность»
26. Разработка модуля ИС «Регистрация на самолет»
27. Разработка модуля ИС «Успеваемость»
28. Разработка модуля ИС «Управление заказами»

29. Разработка модуля ИС «Управление человеческими ресурсами»
30. Разработка модуля ИС «Планирование бюджета»
31. Разработка модуля ИС «Электронный переводчик»
32. Разработка модуля ИС “Управление портфелем заказов”

Темы можно модифицировать, можно предлагать свои. В группе тема не должна повторяться.

Типовые вопросы, выносимые на экзамен:

1. Что такое среда выполнения *Common Language Runtime (CLR)*?
2. Что такое наследование? Поддерживает ли *C #* множественное наследование? Привести примеры
3. В чем разница между структурой и классом в C #? Привести примеры.
4. Что такое свойства в *C #*? Что такое `enum` в *C #*?
5. Что такое массивы? Что такое неровные массивы? Привести примеры.
6. Что такое пространства имен в *C #*? Привести примеры использования.
7. Что такое индексаторы в *C # .NET*? Привести примеры использования.
8. Что такое классы *System. String* и *System.Text.StringBuilder*? Где используются, основные операции?
9. Рассказать об использовании оператора *if …else, switch .. case* и привести примеры.
10. Сформулировать назначение обработчика ошибок и рассмотреть обзор типовой структуры обработчика ошибок.
11. Дать определение цикла, перечислить различные типы циклов, привести примеры блок схем.
12. Дать определение массиву, перечислить различные виды массивов.
13. Перечислить принципы объектно ориентированного программирования.
14. Сделать обзор объектно-ориентированных языков программирования.
15. Дать определение объекту, привести примеры объектов.
16. Дать определение понятиям свойство, событие, метод.
17. Дать обзор типов классов в *C #*?
18. Сделать обзор абстрактных классов и интерфейсов. В чем разница между абстрактным классом *C #* и интерфейсом?
19. Что такое частичные классы и частичные методы в *C #*? Приведите примеры.
20. Что такое *XAML*? Где применяется этот язык. Приведите простой пример.
21. Как происходит компоновка пользовательского интерфейса в приложениях *WPF*? Какие контейнеры компоновки наиболее часто используются?
22. Какие элементы управления содержимым существуют? Определите их основные свойства.
23. Что такое свойства зависимости, прикрепляемые свойства, маршрутизируемые события? Приведите примеры.
24. Что такое *WPF*? Каковы его преимущества по сравнению с *WinForms*?
25. Что такое ресурсы, и какой способ определения ресурсов является наилучшим?
26. Что такое *MVVM*? Может ли использование *MVVM* принести пользу разработчику? Что такое привязка данных?
27. Что такое шаблон элемента управления? Как можно использовать шаблон элемента управления в *WPF*?
28. Что такое кисти? Какие кисти поддерживает *WPF*?
29. Что такое анимация в приложениях *WPF*? Какая бывает анимация, как она реализуется?
30. Что такое команды в приложениях *WPF*? Как они используются в приложениях с развитым графическим интерфейсом?

Типовые задания, выносимые на экзамен:

1. Создайте класс *Condition*, реализующий метод *Calc* вычисления выражения, представленного ниже. Метод *Calc* должен принимать параметр x и возвращать

значение y. Задание выполнить используя шаблон консольного приложения *VS* на языке *C#*.

$$y = \begin{cases} x + 1, x \le 0 \\ x^2 + 3x, 10 \ge x > 0 \\ \sqrt{x^3 + 2x}, x > 10 \end{cases}$$

2. Создайте класс *Condition*, реализующий метод *Calc* вычисления выражения, представленного ниже. Метод *Calc* должен принимать параметр x и возвращать значение y. Задание выполнить используя шаблон консольного приложения *VS* на языке *C#*.

$$y = \begin{cases} x^3 + 1, x \le -1 \\ x^2 + \sin x\,, 13 \ge x > -1 \\ \sqrt{x^2 + 2x}, x > 13 \end{cases}$$

3. Создайте класс *Condition*, реализующий метод *Calc* вычисления выражения, представленного ниже. Метод *Calc* должен принимать параметр x и возвращать значение y. Задание выполнить используя шаблон консольного приложения *VS* на языке *C#*.

$$y = \begin{cases} |x - 5|, x \le 4 \\ x^3 - x^2, 21 \ge x > 4 \\ -x^3(1 + 2x + x^2), x > 21 \end{cases}$$

4. Создайте класс *Condition*, реализующий метод *Calc* вычисления выражения, представленного ниже. Метод *Calc* должен принимать параметр x и возвращать значение y. Задание выполнить используя шаблон консольного приложения *VS* на языке *C#*.

$$y = \begin{cases} x + 1, x \le 0 \\ x^2, 10 \ge x > 0 \\ \sqrt{x + 2x}, x > 10 \end{cases}$$

5. Создайте класс *IterFunc*, реализующий метод *CalcFunc* вычисления выражения, представленного ниже. Метод *CalcFunc* должен принимать параметр x и возвращать значение y. Задание выполнить используя шаблон консольного приложения *VS* на языке *C#*.

б) $y = \sum_{i=1}^{10} \frac{x^2 \sin x}{5}$

6. Создайте класс *IterFunc*, реализующий метод *CalcFunc* вычисления выражения, представленного ниже. Метод *CalcFunc* должен принимать параметр x и возвращать значение y. Задание выполнить используя шаблон консольного приложения *VS* на языке *C#*.

$$y = \prod_{i=1}^{10} \frac{3x^2 \cos x}{6}$$

7. Создайте класс *IterFunc*, реализующий метод *CalcFunc* вычисления выражения, представленного ниже. Метод *CalcFunc* должен принимать параметр x и возвращать значение y. Задание выполнить используя шаблон консольного приложения *VS* на языке *C#*.

$$y = \sum_{i=1}^{10} x^2 + 40(\sin x)^2$$

8. Создайте класс *IterFunc*, реализующий метод *CalcFunc* вычисления выражения, представленного ниже. Метод *CalcFunc* должен принимать параметр x и возвращать значение y. Задание выполнить используя шаблон консольного приложения *VS* на языке *C#*.

$$y = \sum_{i=1}^{10} x^3 \cos x - x^2 \sin x$$

9. Создайте класс *Person*, содержащий закрытые поля *name*, *surname* (фамилия), массив *salary* (зарплата за каждый месяц в году), *extraSalary* (премия за каждый месяц в году). Доступ к закрытым полям обеспечивается через открытые аксессоры соответствующих свойств. Кроме того, класс *Person* должен реализовывать метод *TotalSalary*, возвращающий общий доход за год.
10. Создайте класс *Rectangles* содержащий, поля *Description* (описание набора прямоугольников), массивы: *hight* (высота) и *width* (ширина) для 10 прямоугольников. Доступ к закрытым полям обеспечивается через открытые аксессоры соответствующих свойств. В классе должен реализован метод *TotalSqure,* возвращающий общую площадь всех прямоугольников.
11. Создайте класс *Year*, содержащий закрытые поля: массив *months* (название месяцев), *days* (число дней в году), а также методы *GetDays* и *GetMonth*. Доступ к закрытым полям обеспечивается через открытые аксессоры соответствующих свойств. Метод *GetDays* принимает единственный параметр номер месяца, возвращает число дней в месяце. Метод *GetMonth* принимает в качестве параметра номер месяца, возвращает строку название месяца.
12. Создайте класс *Student*, содержащий закрытые поля *name* (имя), *surname* (фамилия), массив *grade_1* (12 оценок по учебной дисциплине 1), *grade_2* (12 оценок по учебной дисциплине 2), а также метод *AverageGrade* возвращающий средний балл по двум учебным дисциплинам. Доступ к закрытым полям обеспечивается через открытые аксессоры соответствующих свойств.
13. Создайте класс *Rectangle*, содержащий закрытые свойства *Width* (ширина), *Height* (высота) и метод *Square* возвращающий площадь прямоугольника. Класс *Rectangle* должен иметь конструктор, принимающий два параметра, значениями которых инициализируются закрытые свойства *Width* (ширина) и *Height* (высота). Создайте класс *Triangular* (прямоугольный треугольник), унаследованный от класса *Rectangle,* имеющий дополнительное свойство *Hyp,* возвращающее длину гипотенузы прямоугольного треугольника. Переопределите метод *Square* базового класса в классе *Triangular* для корректного расчета площади прямоугольного треугольника.
14. Создайте класс *Triangle*, содержащий закрытые свойства (или свойства) *Side1* (сторона 1), *Side2* (сторона 2), *Side3* (сторона 3), а также метод *SideSum*, возвращающий периметр треугольника. Класс *Triangle* должен иметь конструктор, принимающий три параметра, значениями которых инициализируются закрытые свойств. Создайте класс *Rectangle* унаследованный от класса *Triangle* имеющий дополнительное открытое свойство *Side4* (сторона 4). Переопределите метод *SumSide* базового класса в классе *Rectangle* для корректного расчета периметра прямоугольника (половина площади прямоугольника).
15. Создайте класс *Product*, содержащий закрытые поля (или свойства) *Name* (наименование), *Price* (цена), *RProcent* (скидка или повышающий коэффициент, %), метод *CurrentPrice* возвращающий цену товара со скидкой. Класс *Product* должен иметь конструктор, принимающий три параметра, значениями которых инициализируются закрытые поля (или свойства). Создайте класс *ProductSt* унаследованный от класса *Product,* имеющий дополнительное поле *HProcent* и переопределяющий метод *CurrentPrice* и возвращающий цену товара с учетом повышения цены на *Procent* процентов.
16. Создайте класс *PointP*, содержащий закрытые поля (или свойства) x_1, y_1 (координаты точек) и метод *GetDist*, возвращающий расстояние от начала координат до данной точки (расчет расстояния производится по формуле $D = \sqrt{x_1^2 + y_1^2}$). Класс *Point* должен иметь конструктор, принимающий три параметра, значениями которых инициализируются закрытые поля (или свойства). Создайте класс *LineP* унаследованный от класса *PointP,* имеющий дополнительные свойства x_2, y_2 и переопределяющий метод *GetDist*, который теперь должен возвращать расстояние

между точками (расчет расстояния производится по формуле $D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$).

17. Реализуйте операцию сложения для объектов класса *A* (перегрузка операторов)

```
public class A { int i }
```

18. Реализуйте операцию вычитания для объектов класса *A* (перегрузка операторов)

```
public class A { int i }
```

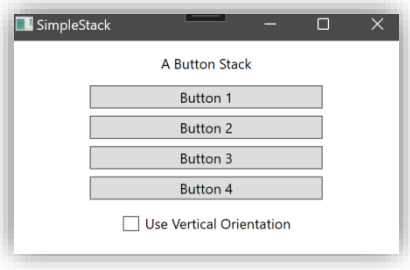
19. Реализуйте операцию умнжения для объектов класса *A* (перегрузка операторов)

```
public class A { int i }
```

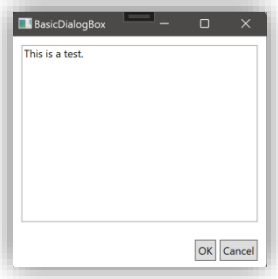
20. Реализуйте операцию деления для объектов класса *A* (перегрузка операторов)

```
public class A { int i }
```

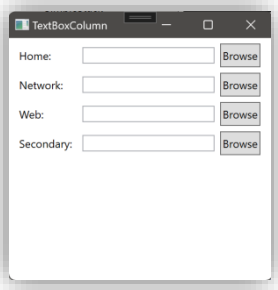
21. Создайте приложение с простым графическим интерфейсом, показанным ниже.



17. Создайте приложение с простым графическим интерфейсом, показанным ниже.



18. Создайте приложение с простым графическим интерфейсом, показанным ниже.



Описание системы оценивания

Оценочные средства (формы текущего и промежуточного контроля)	Показатели оценки	Критерии оценки
Опрос	Корректность и полнота ответов	**Сложный вопрос:** полный, развернутый, обоснованный ответ – 4 балла Правильный, но не аргументированный ответ – 2 балла Неверный ответ – 0 баллов **Обычный вопрос:** полный, развернутый, обоснованный ответ – 4 балла Правильный, но не аргументированный ответ – 2 балла Неверный ответ – 0 баллов. **Простой вопрос:** Правильный ответ – 2 балла; Неправильный ответ – 0 баллов
Тест	Корректность ответов	Максимальное количество баллов за один тес составляет 5
Задание	1) полнота выполнения задания; 2) корректность выводов**;** 3)обоснованность решений.	Выполнена обязательная и самостоятельная часть, даны развернутые ответы на вопросы – **5 баллов** Выполнена обязательная часть, даны развернутые ответы на вопросы – **4 балла** В обязательной части допущены ошибки, формальные ответы на вопросы - **3 балла** В обязательной части допущены ошибки, нет ответов на контрольные вопросы - **2 балла** Имеются множественные ошибки и нет ответов на контрольные вопросы -**1 балл** Работа, представленная для защиты позже установленного срока, оценивается **с понижением баллов**. Просроченные работы и представленные на последнем практическом занятии оцениваются максимум на **1 балл**.

Оценивание студентов на экзамене по дисциплине «Проектирование и разработка web-приложений»

Баллы	Критерии
100-85 «отлично»	Оценка «отлично» на экзамене выставляется обучающемуся, если он глубоко и прочно усвоил программный материал, исчерпывающе, последовательно, четко и логически стройно его излагает, умеет тесно увязывать теорию с практикой, свободно справляется с задачами, вопросами и другими видами применения знаний, причем не затрудняется

	с ответом при видоизменении заданий, использует в ответе материал монографической литературы, правильно обосновывает принятое решение.
84-70 «хорошо»	Оценка «хорошо» выставляется обучающемуся, если он твердо знает материал, грамотно и по существу излагает его, не допуская существенных неточностей в ответе на вопрос, правильно применяет теоретические положения, допускает неточности в увязывании теории с практикой.
69-55 «удовлетворительно»	Оценка «удовлетворительно» выставляется обучающемуся, если он имеет знания только основного материала, но не усвоил его деталей, допускает неточности, недостаточно правильные формулировки, нарушения логической последовательности в изложении программного материала, испытывает затруднения при установлении связи теории и практики.
Менее 55 «неудовлетворительно»	Оценка «неудовлетворительно» выставляется обучающемуся, который не знает значительной части программного материала, допускает существенные ошибки, неуверенно, с большими затруднениями устанавливает связь теории и практики.

Шкала оценивания.

Оценка результатов производится на основе балльно-рейтинговой системы (БРС). Использование БРС осуществляется в соответствии с приказом от 06 сентября 2019 г. №306 «О применении балльно-рейтинговой системы оценки знаний обучающихся».

Схема расчетов сформирована в соответствии с учебным планом направления, согласована с руководителем научно-образовательного направления, утверждена деканом факультета.

Схема расчетов доводится до сведения студентов на первом занятии по данной дисциплине, является составной частью рабочей программы дисциплины и содержит информацию по изучению дисциплины, указанную в Положении о балльно-рейтинговой системе оценки знаний обучающихся в РАНХиГС.

В случае если студент в течение семестра не набирает минимальное число баллов, необходимое для сдачи промежуточной аттестации, то он может заработать дополнительные баллы, отработав соответствующие разделы дисциплины, получив от преподавателя компенсирующие задания.

В случае получения на промежуточной аттестации неудовлетворительной оценки студенту предоставляется право повторной аттестации в срок, установленный для ликвидации академической задолженности по итогам соответствующей сессии.

Обучающийся, набравший в ходе текущего контроля в семестре от 51 до 70 баллов, по его желанию может быть освобожден от промежуточной аттестации.

Количество баллов	**Оценка**	
	прописью	**буквой**
96-100	отлично	A
86-95	отлично	B
71-85	хорошо	C
61-70	хорошо	D
51-60	удовлетворительно	E

Шкала перевода оценки из многобалльной в систему «зачтено»/«не зачтено»:

от 0 по 50 баллов	«не зачтено»
от 51 по 100 баллов	«зачтено»

Перевод балльных оценок в академические отметки «отлично», «хорошо»,

«удовлетворительно»

- «Отлично» (A) - от 96 по 100 баллов – теоретическое содержание курса освоено полностью, без пробелов необходимые практические навыки работы с освоенным материалом сформированы, все предусмотренные программой обучения учебные задания выполнены, качество их выполнения оценено максимальным числом баллов.
- «Отлично» (B) - от 86 по 95 баллов – теоретическое содержание курса освоено полностью, без пробелов необходимые практические навыки работы с освоенным материалом сформированы, все предусмотренные программой обучения учебные задания выполнены, качество их выполнения оценено числом баллов, близким к максимальному.
- «Хорошо» (C) - от 71 по 85 баллов – теоретическое содержание курса освоено полностью, без пробелов, некоторые практические навыки работы с освоенным материалом сформированы недостаточно, все предусмотренные программой обучения учебные задания выполнены, качество выполнения ни одного из них не оценено минимальным числом баллов, некоторые виды заданий выполнены с ошибками.
- «Хорошо» (D) - от 61 по 70 баллов – теоретическое содержание курса освоено полностью, без пробелов, некоторые практические навыки работы с освоенным материалом сформированы недостаточно, большинство предусмотренных программой обучения учебных заданий выполнены, качество выполнения ни одного из них не оценено минимальным числом баллов, некоторые виды заданий выполнены с ошибками.
- «Удовлетворительно» (E) - от 51 по 60 баллов – теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые практические навыки работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий выполнены с ошибками.

4.4. Методические материалы по освоению дисциплины

Рабочей программой дисциплины предусмотрены следующие виды аудиторных занятий: лекции, практические занятия. На лекциях рассматриваются наиболее сложный материал дисциплины. Для развития у магистрантов креативного мышления и логики в каждой теме учебной дисциплины предусмотрены теоретические положения, инструментальные средства, а также примеры их использования при решении задач предиктивной аналитики. Кроме того, часть теоретического материала предоставляется на самостоятельное изучение по рекомендованным источникам для формирования навыка самообучения.

Практические занятия предназначены для самостоятельной работы по решению конкретных задач. Каждое практическое занятие сопровождается заданиями, выдаваемыми магистрантам для решения во внеаудиторное время.

Для работы с печатными и электронными ресурсами СЗИУ имеется возможность доступа к электронным ресурсам. Организация работы магистрантов с электронной библиотекой указана на сайте института (странице сайта – «Научная библиотека»).

5. Методические указания для обучающихся по освоению дисциплины

Рабочей программой дисциплины предусмотрены следующие виды аудиторных занятий: лекции, практические занятия, лабораторные работы. На лекциях рассматриваются наиболее сложный материал дисциплины. Лекция сопровождается презентациями, компьютерными текстами лекции, что позволяет студенту самостоятельно работать над повторением и закреплением лекционного материала. Для этого студенту должно быть предоставлено право самостоятельно работать в компьютерных классах в сети Интернет.

Практические занятия предназначены для углубленного изучения дисциплины. На этих занятиях идет осмысление теоретического материала, приобретаются навыки программирования.

Лабораторные работы позволяют объединить теоретико-методологические знания и практические навыки учащихся в процессе научно-исследовательской деятельности.

Все практические и лабораторные работы проводятся в компьютерных классах с использованием *MS Visual Studio*. Каждая работа должна быть защищена, т.е. студент должен ответить на вопросы преподавателя о ходе выполнения работы, а также на вопросы теоретического характера.

С целью контроля сформированности компетенций разработан фонд контрольных заданий. Его использование позволяет реализовать балльно-рейтинговую оценку, определенную приказом от 28 августа 2014 г. №168 «О применении балльно-рейтинговой системы оценки знаний студентов».

С целью активизации самостоятельной работы студентов в системе дистанцтонного обучения Moodle разработан учебный курс «Базы Данных», включающий набор файлов с текстами лекций, заданиями для выполнения практических и лабораторных работ.

Для активизации работы студентов во время контактной работы с преподавателем отдельные занятия проводятся в интерактивной форме. В основном, интерактивная форма занятий обеспечивается при проведении занятий в компьютерном классе. Интерактивная форма обеспечивается наличием разработанных файлом с заданиями, наличием контрольных вопросов, возможностью доступа к системе дистанционного обучения, а также к тестеру.

Для работы с печатными и электронными ресурсами СЗИУ имеется возможность доступа к электронным ресурсам. Организация работы студентов с электронной библиотекой указана на сайте института (странице сайта – «Научная библиотека»).

6. Учебная литература и ресурсы информационно-телекоммуникационной сети "Интернет", включая перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплинее

6.1. Основная литература.

1. Белоусова, С. И. Основные принципы и концепции программирования на языке VBA в Excel : учеб. пособие / С.И. Белоусова, И.А. Бессонова. - 4-е изд. - Москва : ИНТУИТ [и др.], 2020. - 191 с. - Текст : электронный. - URL: http://www.iprbookshop.ru/97558.html (дата обращения: 03.09.2020)

2. Гниденко, Ирина Геннадиевна. Технологии и методы программирования : учебное пособие для вузов / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. - Москва : Юрайт, 2020. - 235 с. - (Высшее образование) . - Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/450999 (дата обращения: 24.09.2020). - ISBN 978-5-534-02816-4

3. Кудрина, Еыгения Вячеславовна. Основы алгоритмизации и программирования на языке C#. Учебное пособие для бакалавриата и специалитета – Москва: Издательство Юрайт, 2020, ISBN 978-5-534-09796-2

4. Окулов, Станислав Михайлович. Программирование в алгоритмах : учебное пособие / С.М. Окулов. - 6-е изд. (электрон.). - Москва : Лаборатория знаний, 2017. - 384 с. : ил. - Текст : электронный. - URL: https://e.lanbook.com/book/94140 (дата обращения: 25.12.2020).

Все источники основной литературы взаимозаменяемы.

6.2.Дополнительная литература.

1. Гниденко, Ирина Геннадиевна. Технологии и методы программирования : учебное пособие для вузов / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. - Москва : Юрайт, 2020. - 235 с. - (Высшее образование) . - Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/450999 (дата обращения: 24.09.2020). - ISBN 978-5-534-02816-4

2. Казанский, Александр Анатольевич. Программирование на Visual C#: А. А. Казанский. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2024. — 192 с. — (Высшее образование). — ISBN 978-5-534-12338-8. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/537364 (дата обращения: 30.06.2024).
3. Окулов, Станислав Михайлович. Основы программирования : учебное пособие / С.М. Окулов. - 10-е изд., электрон. - Москва : Лаборатория знаний, 2020. - 337 с. : ил. - Текст : электронный. - URL: https://e.lanbook.com/book/135560 (дата обращения: 24.12.2020).

6.3. Учебно-методическое обеспечение самостоятельной работы.

1. Положение об организации самостоятельной работы студентов федерального государственного бюджетного образовательного учреждения высшего образования «Российская академия народного хозяйства и государственной службы при Президенте Российской Федерации» (в ред. приказа РАНХиГС от 11.05.2016 г. № 01-2211);
2. Положение о курсовой работе (проекте), выполняемой студентами федерального государственного бюджетного образовательного учреждения высшего образования «Российская академия народного хозяйства и государственной службы при Президенте Российской Федерации» (в ред. приказа РАНХиГС от 11.05.2016 г. № 01-2211)

6.4. Нормативные правовые документы.

Не используются

6.5. Интернет-ресурсы.

СЗИУ располагает доступом через сайт научной библиотеки http://nwapa.spb.ru/ к следующим подписным электронным ресурсам:

Русскоязычные ресурсы

1. Электронные учебники электронно - библиотечной системы (ЭБС) «Айбукс»
2. Электронные учебники электронно – библиотечной системы (ЭБС) «Лань»
3. Рекомендуется использовать следующий интернет-ресурсы
4. http://serg.fedosin.ru/ts.htm
5. http://window.edu.ru/resource/188/64188/files/chernyshov.pdf

6.6. Иные источники.

Не используются.

7. Материально-техническая база, информационные технологии, программное обеспечение и информационные справочные системы

Все практические занятия проводятся в компьютерном классе. Учебная дисциплина включает использование программного обеспечения *MS Visual Studio, MS Visual Studio Code*.

Методы обучения с использованием информационных технологий (компьютерное тестирование, демонстрация мультимедийных материалов).

Интернет-сервисы и электронные ресурсы (поисковые системы, электронная почта, профессиональные тематические чаты и форумы, системы аудио и видео конференций, онлайн энциклопедии, справочники, библиотеки, электронные учебные и учебно-методические материалы).

Система дистанционного обучения *Moodle*.