МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования «Кабардино-Балкарский государственный университет им. Х.М. Бербекова» (КБГУ)

ИНСТИТУТ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА И ЦИФРОВЫХ ТЕХНОЛОГИЙ

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

СОГЛАСОВАНО

Руководитель образовательной программы _____ **М.М. Лафишева**

«12» 04 **2023г.**

УТВЕРЖДАЮ

Директор института _____ **А.Х. Шапсигов**

«12» 04 **2023г.**

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ (ОЦЕНОЧНЫХ МАТЕРИАЛОВ ПО ДИСЦИПЛИНЕ (МОДУЛЮ) «ПРОФЕССИОНАЛЬНАЯ РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ C#»

02.03.02 Фундаментальная информатика и информационные технологии
(код и наименование направления подготовки)

«Проектирование систем искусственного интеллекта»
(наименование профиля подготовки)

Бакалавр
Квалификация (степень) выпускника

Очная
Форма обучения

Нальчик - 2023

СОДЕРЖАНИЕ

1. Перечень компетенций и этапы их формирования

Процесс изучения дисциплины в соответствии с ФГОС ВО и ОПОП ВО по данному направлению подготовки направлен на формирование элементов следующих компетенций:

универсальных (УК):

Коды	Содержание компетенций
УК-1	Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач

профессиональных (ПКС):

Коды	Содержание компетенций
ПКС-1	Способен понимать, совершенствовать и применять современный математический аппарат

Общая характеристика компетенции

Тип компетенции: общепрофессиональная компетенция выпускника образовательной программы по направлению подготовки высшего образования 02.03.02 Фундаментальная информатика и информационные технологии, уровень ВО бакалавр.

1.1. Этапы формирования компетенций и средства оценивания

Результаты обучения (компетенции)	Индикаторы достижения компетенций	Основные показатели оценки результатов обучения	Виды оценочного материала, обеспечивающий формирование компетенций
УК- 1. Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	**УК-1.1.** Способен применять системный подход и методы анализа и синтеза в научно-познавательной деятельности	**Знать:** Принципы сбора, отбора, обобщения и систематизации информации, вероятные стратегии действий **Уметь:** соотносить разнородные явления и систематизировать их в рамках проблемной ситуации в профессиональной деятельности. **Владеть:** Опытом работы с информационными источниками, выработки стратегий действия	Типовые оценочные материалы для устного опроса (п. 5.1.1); типовые оценочные материалы для контрольной работы (п. 5.2.2); типовые оценочные материалы к зачету (п. 5.2.3.)
	УК-1.2. Способен осуществлять поиск алгоритмов решения проблемной ситуации на основе доступных источников информации с	**Знать:** Принципы и методы системного подхода. **Уметь:** отличать факты от мнений, интерпретаций, оценок и т.д. в рассуждениях других участников деятельности;	

	применением современных информационных и коммуникационных средств и технологий	применять принципы и методы системного подхода для решения поставленных задач. **Владеть:** Практическими навыками выбора оптимальных способов решения задач, исходя из действующих правовых норм, имеющихся ресурсов и ограничений.	
ПКС-1. Способен понимать, совершенствовать и применять современный математический аппарат	**ПКС-1.1.** Способен владеть знаниями в области математических методов и методы исследования математических моделей объектов различной природы	**ПКС-1.1.** З-1. Знает основные принципы построения математических моделей сложных комплексных объектов и процессов и методики исследования этих моделей; современные технологии математического моделирования и вычислительного эксперимента **ПКС-1.1.** У-1. Умеет ставить задачи исследования и оптимизации сложных объектов на основе методов математического моделирования, **ПКС-1.1.** В-1. Владеет навыками выявлять общие закономерности исследуемых объектов, выбирать методы исследования математических моделей	Типовые оценочные материалы для устного опроса (п. 5.1.1); типовые оценочные материалы для контрольной работы (п. 5.2.2); типовые оценочные материалы к зачету (п. 5.2.3.)
	ПКС-1.2. Способен использовать методы проектирования и производства программного продукта, принципы построения, структуры и приемы работы с инструментальными средствами, поддерживающими создание программного продукта	**ПКС-1.2.** З-1. Знает методы и средства планирования и организации исследований и разработок; методы проведения экспериментов и наблюдений, обобщения и обработки информации в области прикладной математики и информатики **ПКС-1.2**. У-1. Умеет исследовать научные и технические проблемы с применением современных технологий математического моделирования и вычислительного эксперимента систематизировать результаты научно-исследовательских и опытно-конструкторских работ; применять методы анализа научно-технической информации	

		ПКС-1.2. В-1. Владеет навыками применения методов анализа научно-технической информации	

1.2. Критерии формирования оценок на различных этапах их формирования

Текущий и рубежный контроль

Этап (уровень)	**Первый этап (уровень)**	**Второй этап (уровень)**	**Третий этап (уровень)**
Баллы	36-50 баллов	51-60 баллов	61-70 баллов
Характеристика	Полное или частичное посещение аудиторных занятий. Частичное выполнение домашнего задания. Частичное выполнение заданий контрольных работ, тестовых заданий на оценку «удовлетворительно».	Полное или частичное посещение аудиторных занятий. Полное выполнение домашнего задания. Выполнение заданий на коллоквиуме на оценку «хорошо».	Полное посещение аудиторных занятий. Полное выполнение домашнего задания, заданий контрольных работ. Выполнение заданий на коллоквиуме на оценку «отлично».

На первом (начальном) этапе формирования компетенции формируются знания, умения и навыки, составляющие базовую основу компетенции, без которой невозможно ее дальнейшее развитие. Обучающийся воспроизводит термины, факты, методы, понятия, принципы и правила; решает учебные задачи по образцу.

На втором (основном) этапе формирования компетенции приобретается опыт деятельности, когда отдельные компоненты компетенции начинают «работать» в комплексе и происходит выработка индивидуального алгоритма продуктивных действий, направленных на достижение поставленной цели. На этом этапе обучающийся осваивает аналитические действия с предметными знаниями по конкретной дисциплине, способен самостоятельно решать учебные задачи, внося коррективы в алгоритм действий, осуществляя координирование хода работы, переносит знания и умения на новые условия.

Третий (завершающий) этап – это овладение компетенцией. Обучающийся способен использовать знания, умения, навыки при решении задач повышенной сложности и в нестандартных условиях. По результатам этого этапа обучающийся демонстрирует итоговый уровень сформированности компетенции.

Промежуточная аттестация – зачет(2, 3, 4 семестры)

Семестр	**Шкала оценивания**	
	Незачтено (36-60)	**Зачтено (61-70)**
2	Студент имеет 36-60 баллов по итогам текущего и рубежного контроля, на зачёте не	Студент имеет 36-45 баллов по итогам текущего и рубежного контроля, на зачете представил полный ответ на один вопроси частично (полностью) ответил на второй. Студент имеет 46-60 баллов по итогам текущего и рубежного контроля, на зачете дал полный ответ на один вопросили частично ответил на оба вопроса.

	ответил ни на один вопрос.	Студенту, имеющему 61-70 баллов по итогам текущего и рубежного контроля, выставляется отметка «зачтено» без сдачи зачёта.

2. Методические материалы и типовые контрольные задания, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций в процессе освоения образовательной программы

Перечень оценочных средств

№	Наименование оценочного средства	Краткая характеристика оценочного средства	Представление оценочного средства в фонде
1.	Коллоквиум	Средство контроля усвоения учебного материала темы, раздела или разделов дисциплины, организованное как учебное занятие в виде собеседования преподавателя с обучающимися.	Вопросы по темам/разделам дисциплины
2.	Тест	Система стандартизированных заданий, позволяющая автоматизировать процедуру измерения уровня знаний и умений обучающегося.	Фонд тестовых заданий

1. Перечень контрольных заданий и иных материалов, необходимых для оценки знаний, умений, навыков и опыта деятельности

3.1. Задания для коллоквиумов (*вопросы для оценки компетенции* ***УК-1, ПКС-1):***

Тема 1. История языка C#. Платформа .Net.Framework.

1. Язык Java.
2. C# вместо Java.
3. Платформа.Net.Framework.
4. Виртуальная машина.

Тема 2. Основы языка C#. Операторы. Структура программы.

1. Алфавит.
2. Идентификаторы.
3. Ключевые слова.
4. Операторы.
5. Структура программы.
6. Пространства имен.
7. Метод MAIN.

Тема 3. Типы данных. Массивы и коллекции.

1. Простые типы.
2. Ссылочные типы.
3. Массивы.

Коллекции

Тема 4. Операторы.
1. Арифметические.
2. Логические.
3. Отношения.

Тема 5. Инструкции. Выражения и разделители.
1. Типы инструкций.
2. Выражения.
3. Разделители.

Тема 6. Решения и ветвления. Циклы.
1. Решение и проекты.
2. Ветвления.
3. Вложенные ветвления.
4. Циклы.
5. Выбор.
6. Инструкция Break.
7. Инструкция Continue.

Тема 7. Обработка ошибок и исключений.
1. Обнаружение ошибок.
2. Окно ошибок в ИСР.
3. Типы исключений.
4. Сборщик мусора.
5. Запрет множественного наследования

Тема 8. Работа со строками.
1. Правила задания символов
2. Правила задания строк.
3. Инструкции работы со строками.

Тема 9. Классы и структуры. Интерфейсы. Делегаты
1. Определения класса.
2. Поля класса.
3. Методы класса.
4. Свойства класса.
5. Использование интерфейсов.
6. Делегаты.

Тема 10. Графика.
1. Приложение Windows Form.
2. Рисование графиков функций.
3. Рисование примитивных фигур.
4. Списки.
5. Таблицы.
6. Сжатие изображений.

3.2. Оценочные материалы для контрольной работы:

контролируемая компетенция УК-1**,** ПКС-1**:**

1. Когда вызываются статические конструкторы классов в C#?

+: один раз при первом создании экземпляра класса или при первом обращении к статическим членам класса;
-: статических конструкторов в C# нет;
-: строгий порядок вызова не определён;
-: после каждого обращения к статическим полям, методам и свойства.

2. Каким образом можно перехватить добавление и удаление делегата из события?

-: такая возможность не предусмотрена;
+: для этого существуют специальные ключевые слова add и remove;
-: использовать ключевые слова get и set;
-: переопределить операторы + и – для делегата.

3. Что произойдёт при исполнении следующего кода? int i=5; object o=I; long j=(long)o;

-: ошибка не произойдёт. Переменная j будет иметь значение 5;
+: средой исполнения будет вызвано исключение InvalidGastException;
-: произойдёт ошибка времени компиляции;
-: значение переменной j предсказать нельзя.

4. Выберите средства, которые предоставляет C# для условной компиляции.

+: директива #if;
+: директива #endif;
+: директива #else;
-: директива #typedef;
+: директива #define;
-: директива #elseif;
+: атрибут Conditional;

5. Реализация какого паттерна (шаблона проектирования) являются события в C# ?

-: детектор (Decorator);
-: Посетитель (Visitor)
+: Издатель-подписчик (Publicher-Subscriber);
-: Шаблонный метод (Template Method).

6. Определяемый программистом тип может быть

- стеком
- списком
- очередью
+ классом
+ интерфейсом

7. Из приведенных ниже высказываний укажите истинное высказывание:

- если метод описывается вне класса, то в классе должен быть указан его прототип

- если метод не возвращает значение, то он должен иметь модификатор (спецификатор) virtual
- для вызова метода можно не создавать экземпляр (объект) класса только в том случае, если метод вызывается внутри класса

+ метод может иметь пустое тело
- метод должен возвращать какое-либо значение

8. Из приведенных ниже высказываний укажите все истинные высказывания:

- если переменная описывается вне класса, то она является глобальной

- если метод не возвращает значение, то он должен иметь модификатор (спецификатор) abstract
+ для вызова метода необходимо создавать экземпляр (объект) класса, если метод не является статическим или не вызывается внутри своего класса
+ метод не обязан возвращать управление оператору, следующему после оператора вызова
- из тела метода можно выйти с помощью оператора goto

9. Из приведенных ниже высказываний укажите все истинные высказывания:

- если метод описывается вне класса, то этот метод становится доступным из любого кода сборки

- члены класса с модификатором (спецификатором) private недоступны методам этого же класса
+ для вызова метода можно не создавать экземпляр (объект) класса только в том случае, если метод является статическим или метод вызывается внутри класса
+ метод, не возвращающий значение, должен вызываться отдельным оператором (не входящим в состав выражения)
- метод, возвращающий значение, должен входить в состав выражения

10. В заголовке определяемого не вложенного типа программист может указать доступность типа с помощью модификатора:

+ internal
- protected
- private
- static
- abstract

11. В заголовке определяемого не вложенного типа программист может указать доступность типа

- с помощью модификатора override
- с помощью модификатора protected
- с помощью модификатора private
- с помощью модификатора new
+ по умолчанию

12. Программист определил тип как class MyOut {}. Тип MyOut будет доступен

- из любого класса любого компоновочного блока

+ из любого класса внутри компоновочного блока, в котором MyOut определен
- только из классов с модификатором public
- только из классов с модификатором internal
- только из классов, у которых доступность указана по умолчанию

13. Сборка Assembly2.cs, приведенная ниже:

```
 class Test3
{   static void Main()
   {   ClassUser user = new ClassUser();
       int  count = user.count;  System.Console.Write("count=" + count);
   }
}
```

ссылается на подключенную сборку Assembly1.cs, которая имеет следующий код:

```
internal class ClassUser
{   public int  count = 10; }
```

14. Укажите результат вывода на консоль после попытки запустить программу на компиляцию и выполнение:

- программа не выполнится, так как сборка Assembly2.cs создает объект user, в то время как класс ClassUser не принадлежит сборке Assembly2.cs
- count=0
- count=10
+ будет выдано сообщение об ошибке компиляции, так как класс ClassUser недоступен в соответствии с его модификатором доступа

15. Исходный модуль содержит код:

```
protected internal class A
{ public int x = 123; }

class B : A
{
   static void Main()
   {
      A a = new A();
      int i = a.x = 10;
      B b = new B();
      int j = b.x = 20;
      System.Console.Write("a.x={0} b.x={1}", i, j);
   }
}
```

16. Укажите результат вывода на консоль после попытки запустить программу на компиляцию и выполнение:

- a.x=20 b.x=10
- a.x=20 b.x=20
- a.x=10 b.x=20
- a.x=10 b.x=10
+ программа не выполнится, так как класс A не может иметь модификатор protected internal

17. Исходный модуль содержит код:

```
protected class A
{ int x = 123; }

class B : A
{  static void Main()
   {  A a = new A();
      int i = a.x = 10;
      B b = new B();
      int j = b.x = 20;
      System.Console.Write("a.x={0} b.x={1}", i, j);
   }
}
```

Укажите результат вывода на консоль после попытки запустить программу на компиляцию и выполнение:
- программа завершится аварийно
- a.x=20 b.x=20
- a.x=10 b.x=20
- a.x=10 b.x=10
+ программа не выполнится, так как класс A не может иметь модификатор protected

18. Исходный модуль содержит код:

```
internal class Point
 {
    public double x,y;  //координаты точки
    public Point(double x, double y) { this.x = x; this.y = y; }
    public void Move(double d){ x+=d; y+=d;}
 }
class Program
 {
    static void Main()
    {
       Point p = new Point();
       p.Move(2);
       System.Console.Write("{0} {1}",p.x, p.y);
    }
 }
```

Метод Main разрабатывался для перемещения точки по обеим координатам на 2. Трансляция программы завершена неудачно. Из приведенных высказываний укажите высказывание, которое раскрывает причину некорректности программы:
- значение полей x и y не определено
- статический метод Main не может обращаться к полям объекта
- тип Point не доступен из класса Program
- поля x и y не доступны из класса Program
+ в классе Point не определен конструктор без параметров

19. Исходный модуль содержит код:

```
internal class Point
  {
     public double x,y;
     public Point(double x, double y) { this.x = x; this.y = y; }
     public static void Move(double d){ x+=d; y+=d;}
  }
class Program
  {
     static void Main()
     {
        Point p = new Point(1,1);
        Point.Move(2);
     }
  }
```

Метод Main разрабатывался для перемещения точки по обеим координатам на 2. Трансляция программы завершена неудачно. Из приведенных высказываний укажите высказывание, которое раскрывает причину некорректности программы:
- значение полей x и y не определено
+ метод Move не может обращаться к полям объекта
- тип Point не доступен из класса Program
- метод Move должен вызываться как метод объекта, а не метод класса
- в классе Point не определен конструктор без параметров

20. Исходный модуль содержит код:

```
internal class Point
{
   public double x,y;
   public Point(double x, double y) { this.x = x; y = y; }
   public void Move(double d){ x+=d; y+=d;}
}
class Program
{
   static void Main()
   {
      Point p = new Point(1,1);
      p.Move(1);
      System.Console.Write("{0} {1}", p.x, p.y);
   }
}
```

Метод Main разрабатывался для перемещения точки по обеим координатам на 1. При запуске программы выведены значения 2 1 вместо ожидаемых значений 2 2. Из приведенных высказываний укажите высказывание, которое раскрывает причину некорректности программы:
- значение полей x и y не определено
- значение поля y может быть изменено только конструктором
+ локальная переменная y скрывает поле y
- метод Move должен вызываться как метод объекта, а не метод класса
- в классе Point не определен конструктор без параметров

21. Исходный модуль содержит код:

```
internal class Rectangle
{  public double width, hight;
   public void Zoom(double d)
   { width += d; hight += d; }
   public void Rectangle(double width, double hight)
   { this.width = width; this.hight = hight; }
}
class Program
{  static void Main()
   { Rectangle p = new Rectangle(10, 20);
      p.Zoom(-5);
      System.Console.Write("{0} {1}", p.width, p.hight);
}   }
```

Метод Main разрабатывался для изменения размера прямоугольника. Трансляция программы завершена неудачно. Из приведенных высказываний укажите высказывание, которое раскрывает причину некорректности программы:
- значение полей width и hight не определено
- статический метод Main не может обращаться к полям объекта
- тип Rectangle не доступен из класса Program
+ в конструкторе не должно быть ключевого слова void
- в классе Rectangle не определен конструктор без параметров

22. Исходный модуль содержит код:

```
internal class Rectangle
{  public double width, hight;
   public void Zoom(double d)
   { width += d; hight += d; }
   public int Rectangle(double width, double hight)
   { this.width = width; this.hight = hight; return width * hight;}
}
class Program
{  static void Main()
   { Rectangle p = new Rectangle(10, 20);
      p.Zoom(-5);
      System.Console.Write("{0} {1}", p.width, p.hight);
}   }
```

Метод Main разрабатывался для изменения размера прямоугольника. Трансляция программы завершена неудачно. Из приведенных высказываний укажите высказывание, которое раскрывает причину некорректности программы:
- метод Zoom должен вызываться как метод класса, а не метод объекта
- статический метод Main не может обращаться к полям объекта
+ конструктор не может возвращать значение
- тип значения, возвращаемого методом Rectangle, должен быть double
- конструктор не может следовать за методами класса

23. Исходный модуль содержит код:

```
class Rectangle
{  public double width, hight;
   public Rectangle()  { width = 40; hight = 80; }
   static Rectangle()   { width = 100; hight = 200; }
   public void Zoom(double d)  { width += d; hight += d; }
}
class Program
{   static void Main()
   {  Rectangle p;
      p = new Rectangle();
      p = new Rectangle();
      p.Zoom(15);
      System.Console.Write("{0} {1}", p.width, p.hight);
   }
}
```

Метод Main разрабатывался для изменения размера прямоугольника. Трансляция программы завершена неудачно. Из приведенных высказываний укажите высказывание, которое раскрывает причину некорректности программы:
+ статический конструктор в классе Rectangle не может использовать нестатические поля
- в классе Rectangle не могут быть одновременно определены статический и нестатический конструкторы с одинаковой сигнатурой
- у нестатического конструктора в классе Rectangle отсутствует модификатор (спецификатор) доступа public
- тип значения, возвращаемого конструктором Rectangle, должен быть void
- в методе Main предпринята попытка создать объект дважды

24. Исходный модуль содержит код классов MyMessage и Program.

```
class MyMessage
{
   private static string msg = "Назад!";
   public MyMessage(string s) { msg = s; }
   public MyMessage() { }
   public string Msg() { return msg; }
}
```

В классе Program определен метод Main, в теле которого находится код:

```
MyMessage m1 = new MyMessage("Ура!");
MyMessage m2 = new MyMessage("Вперед!");
MyMessage m3 = new MyMessage();
System.Console.Write(m1.Msg() + m2.Msg() + m3.Msg());
```

Укажите результат вывода на консоль после выполнения метода Main():
- Ура!Вперед!Назад!
- Ура!Вперед!
+ Вперед! Вперед! Вперед!
- Назад! Назад! Назад!

25. Исходный модуль содержит код классов MyMessage и Program.

```
class MyMessage
{
   public string msg = "Назад!";
   public MyMessage(string s) { msg = s; }
   public MyMessage() { }
   public string Msg() { return msg; }
}
```

В классе Program определен метод Main, в теле которого находится код:

```
MyMessage m1 = new MyMessage("Вперед!");
MyMessage m2 = new MyMessage("Ура!");
MyMessage m3 = m1;
m1.msg = "Ура!";
System.Console.Write(m1.Msg() + m2.Msg() + m3.Msg());
```

Укажите результат вывода на консоль после выполнения метода Main():

+ Ура!Ура!Ура!
- Ура! Ура!Вперед!
- Вперед!Ура!Ура!
- Назад!Ура!Назад!

26. Исходный модуль содержит код классов MyMessage и Program.

```
class MyMessage
{
   private readonly string msg = "Ура!";
   public MyMessage(string s) { msg = s; }
   public MyMessage() { msg = "Вперед"; }
   public string Msg() { return msg; }
}
```

В классе Program определен метод Main, в теле которого находится код:

```
MyMessage m1 = new MyMessage("Назад!");
MyMessage m2 = new MyMessage();
MyMessage m3 = new MyMessage("Назад!");
System.Console.Write(m1.Msg() + m2.Msg() + m3.Msg());
```

Укажите результат вывода на консоль после выполнения метода Main():

- программа не корректна: попытка изменить переменную, доступную только по чтению
- Назад! Ура!Вперед!
+ Назад!Вперед!Назад!
- Назад!Ура!Назад!

27. Исходный модуль содержит код:

```
class Rectangle
{  double width=15, hight=10, s;
   string t = "прямоугольник";
```

```
   public Rectangle(double w, double h) { width = w; hight = h; s = w * h; }
   public Rectangle(int w, int h) { width *= w; hight*=h; s = width * hight; }
   public string RectForm() { return string.Format("{0}: площадь={1}.  ", t, s); }
}
class Program
{  static void Main()
   { Rectangle k = new Rectangle(2, 3); Rectangle t = new Rectangle(3, 4.0);
     System.Console.Write(k.RectForm() + t.RectForm());
   }
}
```

Укажите результат вывода на консоль после попытки запустить программу на компиляцию и выполнение:

+ прямоугольник: площадь=900. прямоугольник: площадь=12
- прямоугольник: площадь=900. прямоугольник: площадь=1800
- прямоугольник: площадь=6. прямоугольник: площадь=1800
-прямоугольник: площадь=6. прямоугольник: площадь=12
-программа завершится аварийно, так как конструкторы имеют одинаковое количество параметров

28. Исходный модуль содержит код:

```
class Rectangle
{  double width=15, hight=10, s;
   string t = "прямоугольник";
   public Rectangle(double w, double h) { width = w; hight = h; s = w * h; }
   public Rectangle(int k) { width *= k; hight *= k; s = width * hight; }
   public Rectangle(double side) : this(side, side) { t = "квадрат"; }
   public string RectForm() { return string.Format("{0}: площадь={1}.  ", t, s); }
}
class Program
{  static void Main()
   {    Rectangle k = new Rectangle(2); Rectangle t = new Rectangle(20.0);
        System.Console.Write(k.RectForm() + t.RectForm());
   }
}
```

Укажите результат вывода на консоль после попытки запустить программу на компиляцию и выполнение:
- прямоугольник: площадь=4. квадрат: площадь=400
- прямоугольник: площадь=600. прямоугольник: площадь=6000
+ прямоугольник: площадь=600. квадрат: площадь=400
- программа завершится аварийно, так как третий конструктор не вычисляет площадь

29. Исходный модуль содержит код:

```
class Rectangle
{  double width=0.0, hight=0.0, s=0.0;
   string t = "Прямоугольник";
   public Rectangle(double w, double h) { width = w; hight = h; s = w * h; }
   public Rectangle(double side) : this(side, side) { t = "Квадрат"; }
```

```
    public Rectangle() : this(1.0, 1.0) { t = "Точка"; }
    public string RectForm() { return string.Format("{0}: площадь={1}  ", t, s); }
}
class Program
{    static void Main()
    {  Rectangle k = new Rectangle(10);    Rectangle t = new Rectangle();
       System.Console.Write(k.RectForm() + t.RectForm());
    }
}
```

Укажите результат вывода на консоль после попытки запустить программу на компиляцию и выполнение:
- квадрат: площадь=0; точка: площадь=1
- квадрат: площадь=0; точка: площадь=0
+ квадрат: площадь=100; точка: площадь=1
- программа завершится аварийно, так как вызываемые конструкторы не вычисляют площадь

30. Программа предназначена для вывода на консоль строки СИДОРОВ:

```
namespace Message
{
  class MyPrint{ public static string View(){return "КОЗЛОВ";}}
}
namespace OutPut
{
  class MyPrint { public static string View(){return "СИДОРОВ";}}
}
class Program
{
   static void Main(string[] args)
   {     System.Console.Write(  _____________  );   }
}
```

Среди приведенных ниже кодов укажите код, которым необходимо заменить знаки подчеркивания для получения корректной программы, решающей поставленную задачу:
- OutPut.View()
- MyPrint.View()::OutPut
+ OutPut.MyPrint.View()
- OutPut::MyPrint.View()
- this.MyPrint.View()

31. При выполнении метода Main необходимо создать объект класса Car со следующими атрибутами: заводской номер - 143155, марка - ГАЗ-3102.

```
 class Car
 {
    int nom;         //Заводской номер
    string mark;   //Марка
    public Car(int nom) { this.nom = nom; }
    public Car(int nom,string mark):_________
       { this.mark = mark; }
```

```
 }
class Program
{
   static void Main()
   {
      Car car = new Car(143155,"ГАЗ-3102");
   }
}
```

Среди приведенных ниже кодов укажите код, которым необходимо заменить знаки подчеркивания для получения корректной программы, решающей поставленную задачу:
- base(nom)
- base()
- Car(nom)
+ this(nom)
- nom

32. При выполнении метода Main необходимо создать объект класса Clock со следующими атрибутами: заводской номер - 3355, марка - ПОЛЕТ.

```
class Clock
 {
    int nom;                  //Заводской номер
    static string mark;   //Марка
    public Clock(int nom) { this.nom = nom; }
    public Clock() { nom = 3355; }
    static Clock() { mark = "ПОЛЕТ"; }
 }
class Program
{
   static void Main()
   {       Clock c = _____________;      }
}
```

Среди приведенных ниже кодов укажите все коды, подстановка каждого из которых вместо знаков подчеркивания позволяет получить корректную программу, решающую поставленную задачу:
+ new Clock(3355)
- Clock(3355)
- Clock(3355,"ПОЛЕТ")
- new Clock(3355,"ПОЛЕТ")
+ new Clock();

33. Укажите модификатор метода, который позволяет для его вызова из других классов не создавать объект:
- virtual

- protected
- public
- partial
+ static

34. Из приведенных ниже высказываний укажите все истинные высказывания:

+ инкапсуляция предполагает сокрытие полей класса

+ свойство - это одна или две специальные функции, доступные для чтения и/или записи
- свойство - это набор параметров, характеризующих состав класса
- свойство не может иметь модификатор private, так как оно не будет доступно извне
+ свойство может быть статическим

35. Из приведенных ниже высказываний укажите все истинные высказывания:

- индексатор не может быть статическим

- индекс, используемый в качестве параметра индексатора, должен быть числом
- индексатор может быть создан только в случае наличия у класса скрытого массива
- индексатор не может иметь модификатор private, так как оно не будет доступно извне
+ индексатор - это одна или две специальные функции, доступные для чтения и/или записи и получающие в качестве параметров один или несколько индексов

36. Доступность метода из других классов может быть указана с помощью модификатора:

- sealed
+ protected
+ private
- partial
- abstract

37. Доступность метода из других классов может быть указана:

+ по умолчанию
+ с помощью модификатора public
- с помощью модификатора static
+ с помощью модификатора protected
+ с помощью модификатора override

38. Доступность поля из других классов может быть указана:
+ по умолчанию
- с помощью модификатора new
- с помощью модификатора out
+ с помощью модификатора protected
+ с помощью модификатора private

---

39. Исходный модуль содержит код:

```
class A
{  private int b;
   public  static int B
   {  get { return b*b; }    set { b = value; }  }
}
```

```
public class Test
{   public static void Main()
   {       A.B = 5;
   System.Console.Write(A.B);
   }
}
```

Метод Main разрабатывался для вывода квадрата числа 5. Трансляция программы завершена неудачно. Из приведенных высказываний укажите высказывание, которое раскрывает причину некорректности программы:
- свойство B не может быть статическим
+ статическое свойство B использует нестатическое поле
- для использования свойства B должен быть создан объект класса A
- значение поля b не определено
- поле b недоступно для использования в свойстве

40. Исходный модуль содержит код:

```
class A
{
   private static double a;
   public static double this[double i]
   {  get { return System.Math.Pow(2.0, i); }
      set { a = value; }
   }
}
public class Test
{   public static void Main()
   {
      A[3] = 4;  double c = A[3];
      System.Console.Write(c);
   }
}
```

Метод Main разрабатывался для вывода степени числа 2. Трансляция программы завершена неудачно. Из приведенных высказываний укажите высказывание, которое раскрывает причину некорректности программы:
- класс A не может состоять только из статических членов
- для использования индексатора должен быть создан объект класса A
+ индексатор не может быть статическим
- в индексаторе не используется массив
- класс A должен содержать массив для его использования в индексаторе

41. Исходный модуль содержит код:

```
class A
{
   private static int b;
   public static int B
   { get { return b * b; } set { b = value; } }
}
public class Test
```

```
{
   public static void Main()
   {
      A a = new A();
      a.B = 5;
      System.Console.Write(a.B);
   }
}
```

Метод Main разрабатывался для вывода квадрата числа 5. Трансляция программы завершена неудачно. Из приведенных высказываний укажите высказывание, которое раскрывает причину некорректности программы:
- свойство B не может быть статическим
- класс A не может состоять только из статических членов
- значение поля b не определено
- для создания объекта класс A должен содержать конструктор
+ свойство B не может быть доступно через ссылку на объект класса

42. Исходный модуль содержит код:

```
class MyOut
{
  public  int A = 555;
          int B;
  public  int View() { return A + B; }
}
class Program
{
   static void Main()
   {
      MyOut mout = new MyOut();
      System.Console.Write(mout.A + mout.B);
   }
}
```

Метод Main разрабатывался для получения суммы полей A и B объекта класса MyOut. Трансляция программы завершена неудачно. Из приведенных высказываний укажите высказывание, которое раскрывает причину некорректности программы:
- значение поля B не определено
- статический метод Main не может обращаться к полям объекта
+ поле B закрыто для доступа из других классов
- тип MyOut не доступен из класса Program
- в классе MyOut не определен конструктор без параметров

43. Исходный модуль содержит код:

```
class MyOut
{
  int A,B=777;
  int View() { return A + B; }
}
class Program
```

```
{
   static void Main()
   {
      MyOut mout = new MyOut();
      System.Console.Write(mout.View());
   }
}
```

Метод Main разрабатывался для получения суммы полей A и B объекта класса MyOut. Трансляция программы завершена неудачно. Из приведенных высказываний укажите высказывание, которое раскрывает причину некорректности программы:
- поля A и B закрыты для доступа из других классов
- значение поля A не определено
- вызов метода View() должен содержать не имя объекта, а имя класса MyOut
- в классе MyOut не определен конструктор без параметров
+ метод View() закрыт для доступа из других классов

44. Исходный модуль содержит код:

```
class MyOut
{
  protected static int A = 999;
  protected int B;
  public static int View() { return A; }
}
class Program
{
   static void Main()
   {
      MyOut mout = new MyOut();
      System.Console.Write(mout.View() + mout.B);
   }
}
```

Метод Main разрабатывался для получения суммы полей A и B объекта класса MyOut. Трансляция программы завершена неудачно. Из приведенных высказываний укажите высказывание, которое раскрывает причину некорректности программы:
- поле A недоступно из класса Program
- в классе MyOut не определен конструктор без параметров
- тип MyOut не доступен из класса Program
+ поле B недоступно из класса Program
- значение поля B не определено

45. Исходный модуль содержит код:

```
class Сотрудник
{
   private int id;
   private float зарплата;
   public Сотрудник(int i, float z) { id = i; зарплата = z; }
   public float Зарплата(int номер)  {   return номер < id ? зарплата : -1; }
   public void Зарплата(int номер, ref float зарпл)
```

```
   {   if (номер < id)
       {  float зрпл = зарплата;
          зарплата = зарпл;   зарпл = зрпл;
 } }  }
class Program
{   static void Main()
    {  Сотрудник Иванов = new Сотрудник(12, 40000.0f);
       float z = 45000.0f; Иванов.Зарплата(10, ref z);
       System.Console.Write("Зарплата Иванова = " + z);
}  }
```

Укажите результат вывода на консоль после попытки запустить программу на компиляцию и выполнение:

+ 40000
- 45000
- 20000
- -1
- программа не выполняется, так как обнаружена одна или несколько синтаксических ошибок

46. Исходный модуль содержит код:

```
class Сотрудник
{  private int id;
   private float зарплата = 20000;
   public Сотрудник(int i, float z) { id = i; зарплата = z; }
   public float Зарплата(int номер)  {   return номер < id ? зарплата : -1; }
   public void Зарплата(int номер, ref float зарпл)
   {   if (номер < id)
       {  float зрпл = зарплата;
          зарплата = зарпл;   зарпл = зрпл;
 } }  }
class Program
{   static void Main()
    {  Сотрудник Иванов = new Сотрудник(12, 40000.0f);
       float z = 45000.0f; Иванов.Зарплата(10, ref z);
       System.Console.Write("Зарплата Иванова = " + Иванов.Зарплата(8));
}  }
```

Укажите результат вывода на консоль после попытки запустить программу на компиляцию и выполнение:

- 40000
- 20000
+ 45000
--1
- программа не выполняется, так как обнаружена одна или несколько синтаксических ошибок

47. Сборка Assembly2.cs, приведенная ниже:

```
public class Test3
{   static void Main()
```

```
   {  ClassUser user = new ClassUser ();
      int  count = ClassUser. count = 112;
      System.Console.Write ("count=" + count);
   }
}
```

ссылается на подключенную сборку Assembly1.cs, которая имеет следующий код:

```
public class ClassUser
{   internal static int  count = 10; }
```

Укажите результат вывода на консоль после попытки запустить программу на компиляцию и выполнение:
- count=10
- программа не выполнится, так как сборка Assembly2.cs создает объект user, в то время как класс ClassUser не принадлежит сборке Assembly2.cs
+ будет выдано сообщение об ошибке компиляции, так как нельзя работать с недоступным в соответствии с модификатором полем count
- count=112
- будет выдано сообщение об ошибке компиляции, так как нельзя создать объект класса ClassUser, содержащего недоступное в соответствии с модификатором поле count

48. Исходный модуль содержит код:

```
class Сотрудник
{
   private int id;
   private float зарплата;
   public Сотрудник(int i, float z) { id = i; зарплата = z; }
   public float Зарплата(int номер)  {   return номер < id ? зарплата : -1; }
   public void Зарплата(int номер, ref float зарпл)
   {   if (номер < id)
       {  float зрпл = зарплата;
          зарплата = зарпл;   зарпл = зрпл;
 } }  }
class Program
{   static void Main()
   {  Сотрудник Иванов = new Сотрудник(12, 40000.0f);
      float z = 45000.0f; Иванов.Зарплата(10, ref z);
      System.Console.Write("Зарплата Иванова = " + Иванов.Зарплата(12));
}  }
```

Укажите результат вывода на консоль после попытки запустить программу на компиляцию и выполнение:
- 40000
- 45000
- 20000
+ -1
- программа не выполняется, так как обнаружена одна или несколько синтаксических ошибок

49. Исходный модуль содержит код:

```
class Room
   {
      private int nom1;
      public  int nom2;
      public Room(int a, int b){ nom1 = a; nom2 = b; }
      public int Nom
      {
         get { return nom1 * 100 + nom2; }
         set { if (value<5555) {nom1 = value/100; nom2 = value%100; }}
      }
   }
   class Program
   {
      static void Main()
      {
         Room r = new Room(65,98);
         r.Nom = 6712;
         System.Console.Write("{0}  ",  r.Nom);
      }
   }
```

Укажите результат вывода на консоль после выполнения метода Main():

+ 6598
- 6712
- 6512
- 798

50. Исходный модуль содержит код:

```
class Room
   {
      private int nom1;
      public  int nom2;
      public Room(int a, int b){ nom1 = a; nom2 = b; }
      public int Nom
      {
         get { return nom1 * 100 + nom2; }
         set { if (value<5555) {nom1 = value/100; nom2 = value%100; }}
      }
   }
   class Program
   {
      static void Main()
      {
         Room r = new Room(12,45);
         r.Nom = 4431;
         System.Console.Write("{0}  ",  r.Nom);
      }
   }
```

Укажите результат вывода на консоль после выполнения метода Main():

- 1231
- 1245
+ 4431
- 4445

51. Исходный модуль содержит код:

```
class Room
   {
      public int nom1;
               int nom2;
      public Room(int a, int b){ nom1 = a; nom2 = b; }
      public int Nom
      {
         get { return nom1 * 100 + nom2; }
         set { if (value<5555) {nom1 = value/100; nom2 = value%100; }}
      }
   }
   class Program
   {
      static void Main()
      {
         Room r = new Room(21,56);
         r.nom1 = 77;
         System.Console.Write("{0}  ", r.Nom);
      }
   }
```

Укажите результат вывода на консоль после выполнения метода Main():
- 2156
-2177
- 7721
+ 7756

52. Программа предназначена для присвоения переменной r класса B значения 4 :

```
class A
{  public B b = null;
   public A()
   {   B b = new B();  }
   public class B
   {  static public int r;
      public int t;
}   }
class Test
{    public static void Main()
   {  A a = new A();
      ___________
   }
}
```

Среди приведенных ниже кодов укажите код, которым необходимо заменить знаки подчеркивания для получения корректной программы, решающей поставленную задачу:
- a.B.r = 4;
+ A.B.r = 4;
- a.b.r = 4;
- A.b.r = 4;
- B.r = 4;

53. Программа предназначена для присвоения переменной t класса B значения 8 :

```
class A
{  public B b = null;
   public A()
   {   B b = new B();  }
   public class B
   {  static public int r;
      public int t;
}   }
class Test
{    public static void Main()
   {  A a = new A();
      _____________
   }
}
```

Среди приведенных ниже кодов укажите код, которым необходимо заменить знаки подчеркивания для получения корректной программы, решающей поставленную задачу:
- a.B.t = 8;
- A.B.t = 8;
+ a.b.t = 8;
- A.b.t = 8;
- b.t = 8;

54. Программа предназначена для присвоения переменной t класса B значения 8 :

```
class A
{  static public B b = null;
   public A()
   {   B b = new B();  }
   public class B
   {  static public int r;
      public int t;
}   }
class Test
{    public static void Main()
   {  A a = new A();
      _____________
   }
}
```

Среди приведенных ниже кодов укажите код, которым необходимо заменить знаки подчеркивания для получения корректной программы, решающей поставленную задачу:

- a.B.t = 8;
- A.B.t = 8;
- a.b.t = 8;
+ A.b.t = 4;
- B.t = 8;

55. Исходный модуль содержит код:

```
class ЧЕЛОВЕК
   {
      class ПАСПОРТ {}
      class ДОЛЖНОСТЬ {}
   }
   class Program
   {
      static void Main() { ______  P; }
   }
```

В методе Main требуется объявить ссылку P на тип ДОЛЖНОСТЬ. Среди приведенных ниже вариантов ответов укажите истинный вариант:
- вместо знаков подчеркивания укажите ЧЕЛОВЕК
- вместо знаков подчеркивания укажите ЧЕЛОВЕК.ДОЛЖНОСТЬ
- вместо знаков подчеркивания укажите ДОЛЖНОСТЬ
- вместо знаков подчеркивания укажите ЧЕЛОВЕК.ПАСПОРТ.ДОЛЖНОСТЬ
+ объявить требуемую ссылку в методе Main невозможно

56. Исходный модуль содержит код:

```
class ЧЕЛОВЕК
   {
      class ПАСПОРТ {}
      public class ДОЛЖНОСТЬ {}
   }
   class Program
   {
      static void Main() { ______  P; }
   }
```

В методе Main требуется объявить ссылку P на тип ДОЛЖНОСТЬ. Среди приведенных ниже вариантов ответов укажите истинный вариант:
- вместо знаков подчеркивания укажите ЧЕЛОВЕК
+ вместо знаков подчеркивания укажите ЧЕЛОВЕК.ДОЛЖНОСТЬ
- вместо знаков подчеркивания укажите ДОЛЖНОСТЬ
- вместо знаков подчеркивания укажите ПАСПОРТ.ДОЛЖНОСТЬ
- объявить требуемую ссылку в методе Main невозможно

57. Исходный модуль содержит код:

```
class ЧЕЛОВЕК
   {
      public class ПАСПОРТ {}
      public class ДОЛЖНОСТЬ {}
```

```
}
class Program
{
   static void Main() { _______  P; }
}
```

В методе Main требуется объявить ссылку P на тип ПАСПОРТ. Среди приведенных ниже вариантов ответов укажите истинный вариант:
- вместо знаков подчеркивания укажите ЧЕЛОВЕК
- вместо знаков подчеркивания укажите ПАСПОРТ
+ вместо знаков подчеркивания укажите ЧЕЛОВЕК.ПАСПОРТ
- вместо знаков подчеркивания укажите ЧЕЛОВЕК.ДОЛЖНОСТЬ.ПАСПОРТ
- объявить требуемую ссылку в методе Main невозможно

58. Язык C# допускает

- наследование классом нескольких классов
+ наследование классом нескольких интерфейсов
- наследование классом делегата
- наследование классом структуры
- наследование структурой класса

59. Класс, в заголовке которого указан модификатор sealed:

+ допускает наследование другого класса
+ не может быть унаследован другим классом
- может быть унаследован другим классом в текущей сборке
- может быть унаследован структурой
- может быть унаследован другим классом во внешней сборке

60. Из приведенных ниже высказываний укажите все истинные высказывания:
- System.Object - это статический объект класса System
+ System.Object - это класс, являющийся базовым для всех остальных классов
+ базовый класс должен иметь конструктор без параметров, если в производном классе определен конструктор с параметрами и без ключевого слова base
- производный класс не накладывает каких-либо требований на состав конструкторов базового класса

+ если в производном классе нет явно определенных конструкторов, то в базовом классе должен быть конструктор без параметров

Критерии формирования оценок по тестовым заданиям:

По итогам выполнения тестовых заданий оценка производится по пятибалльной шкале. При правильных ответах на:

- 89-100% заданий – «5» (баллов);
- 70-88% заданий – «4» баллов);
- 50-69% заданий – «3» (балла);
- 30-49% заданий – «2» (балла);
- 10-29% заданий – «1» (балл);

- менее 10% заданий – «0» (баллов).

4. Вопросы к зачету (7 семестр) по дисциплине «Профессиональная разработка программного обеспечения C#»

Вопрос		Код компетенции (согласно РПД)
1.	Структура программы на языке C/C++.	УК-1; ПКС-1
2.	Концепции восходящего и нисходящего программирования.	УК-1; ПКС-1
3.	Понятия объекта и класса в объектно-ориентированном программировании.	УК-1; ПКС-1
4.	Принцип инкапсуляции в объектно-ориентированном программировании.	УК-1; ПКС-1
5.	Наследование в объектно-ориентированном программировании.	УК-1; ПКС-1
6.	Понятие виртуальных методов в объектно-ориентированном программировании.	УК-1; ПКС-1
7.	Понятие компонента и принцип компонентного программирования.	УК-1; ПКС-1
8.	Что называется, свойством компонента? Понятие события компонента.	УК-1; ПКС-1
9.	Классы в языке C++. Инкапсуляция. Описание класса. Рекомендации по составу класса.	УК-1; ПКС-1
10.	Конструкторы и деструкторы классов в языке C++.	УК-1; ПКС-1
11.	Структуры, объединения и классы в языке C++. Общее и отличия. Примеры применения.	УК-1; ПКС-1
12.	Конструктор копирования в языке C++. Поверхностное и глубинное копирование. Конструкторы и присваивание строк. Примеры применения.	УК-1; ПКС-1
13.	Указатели на элементы классов в языке C++. Указатель *this*.	УК-1; ПКС-1
14.	Константные поля и методы класса в языке C++. Примеры применения.	УК-1; ПКС-1
15.	Статические элементы класса в языке C++. Статические поля и статические методы. Примеры применения.	УК-1; ПКС-1
16.	Полиморфизм в языке C++. Виртуальные методы классов. Примеры применения.	УК-1; ПКС-1
17.	Иерархия наследования классов в языке C++. Доступ к членам базовых классов. Ключи доступа. Примеры применения.	УК-1; ПКС-1
18.	Простое и множественное наследование в языке C++. Примеры применения.	УК-1; ПКС-1
19.	Виртуальные базовые классы в языке C++. Примеры применения.	УК-1; ПКС-1
20.	Виртуальные деструкторы в языке C++. Примеры применения.	УК-1; ПКС-1
21.	Друзья класса в языке C++. Дружественные функции и поля. Дружественный класс. Примеры применения.	УК-1; ПКС-1
22.	Механизм позднего связывания в языке C++. Примеры применения.	УК-1; ПКС-1
23.	Абстрактные классы в языке C++. Абстрактные методы и классы. Примеры применения.	УК-1; ПКС-1

24.	Шаблоны функций в языке C++. Объявление, определение, параметры, возвращаемые значения. Достоинства и недостатки. Примеры применения.	УК-1; ПКС-1
25.	Шаблоны структур и объединений в языке C++. Определение и инициализация структур-переменных. Примеры применения.	УК-1; ПКС-1
26.	Шаблоны классов в языке C++. Создание и использование. Специализация шаблонов классов. Достоинства и недостатки шаблонов классов. Примеры применения.	УК-1; ПКС-1
27.	Перегрузка унарных и бинарных операций в языке C++. Примеры применения.	УК-1; ПКС-1
28.	Перегрузка операций присваивания в языке C++. Примеры применения.	УК-1; ПКС-1
29.	Перегрузка операций new и delete в языке C++. Примеры применения.	УК-1; ПКС-1
30.	Перегрузка операции вызова функции в языке C++. Примеры применения.	УК-1; ПКС-1
31.	Перегрузка операции индексирования в языке C++. Примеры применения.	УК-1; ПКС-1
32.	Перегрузка операций приведения типа в языке C++. Примеры применения.	УК-1; ПКС-1
33.	Динамическое определение типа и преобразование типов. Повышающие и понижающие преобразования. Примеры применения.	УК-1; ПКС-1
34.	Преобразование ссылок в языке C++. Перекрестное преобразование. Примеры применения.	УК-1; ПКС-1
35.	Преобразование типов в языке C++: операции const_cast, static_cast, dynamic_cast, reinterpret_cast. Примеры применения.	УК-1; ПКС-1
36.	Обработка исключительных ситуаций в языке C++. Иерархии исключений. Общий механизм обработки исключений. Примеры применения.	УК-1; ПКС-1
37.	Синтаксис исключений в языке C++. Список исключений функции. Примеры применения.	УК-1; ПКС-1
38.	Исключения в конструкторах и деструкторах языка C++. Перехват исключений. Примеры применения.	УК-1; ПКС-1
39.	Потоковые классы в языке C++. Стандартные потоки. Форматирование данных. Флаги и форматирующие методы манипуляторы. Примеры применения.	УК-1; ПКС-1
40.	Строковые потоки в языке C++. Примеры применения.	УК-1; ПКС-1
41.	Ввод-вывод встроенных (стандартных) типов в языке C++. Примеры применения.	УК-1; ПКС-1
42.	Состояния предопределенных объектов (потоков) в языке C++. Ошибки потоков. Примеры применения.	УК-1; ПКС-1
43.	Потоки и типы, определенные пользователем в языке C++. Ввод-вывод типов, определенных пользователем. Примеры применения.	УК-1; ПКС-1
44.	Форматированный ввод-вывод в языке C++. Манипуляторы. Примеры применения.	УК-1; ПКС-1
45.	Файловый ввод-вывод в языке C++. Файловые потоки. Примеры применения.	УК-1; ПКС-1
46.	Контейнерные классы в языке C++. Примеры применения.	УК-1; ПКС-1

47.	Последовательные контейнеры в языке C++. Вектор (*vector*). Вектор логических значений *vector<bool>*. Примеры применения.	УК-1; ПКС-1
48.	Последовательные контейнеры в языке C++. Двусторонняя очередь (*deque*). Примеры применения.	УК-1; ПКС-1
49.	Последовательные контейнеры в языке C++. Список (*list*). Примеры применения.	УК-1; ПКС-1
50.	Адаптеры последовательных контейнеров в языке C++. Стек (*stack*). Примеры применения.	УК-1; ПКС-1
51.	Адаптеры последовательных контейнеров в языке C++. Очередь *FIFO (queue)*. Примеры применения.	УК-1; ПКС-1
52.	Адаптеры последовательных контейнеров в языке C++. Очередь с приоритетами (*prioriy_queue*). Примеры применения.	УК-1; ПКС-1
53.	Ассоциативные контейнеры в языке C++. Словари (*map*), словари с дубликатами (*multimap*). Примеры применения.	УК-1; ПКС-1
54.	Асоциативные контейнеры в языке C++. Множества (*set*), множества с дубликатами (*multiset*), битовые множества (*bitset*). Примеры применения.	УК-1; ПКС-1
55.	Итераторы в языке C++. Обратные итераторы. Итераторы вставки. Потоковые итераторы. Примеры применения.	УК-1; ПКС-1
56.	Адаптеры указателей на функции языка C++. Адаптеры методов. Примеры применения.	УК-1; ПКС-1
57.	Модифицирующие операции с последовательностями в языке C++. Примеры применения.	УК-1; ПКС-1
58.	Средства для численных расчетов и работы с комплексными числами в языке C++. Примеры применения.	УК-1; ПКС-1
59.	Средства поддержки языка и локализации. Примеры применения.	УК-1; ПКС-1
60.	Текстовые и бинарные файлы в языке C++. Связывание файловых переменных с внешней средой. Примеры применения.	УК-1; ПКС-1
61.	Типовые действия с файлами в языке C++: создание, открытие, закрытие, чтение и изменение. Примеры применения.	УК-1; ПКС-1
62.	Последовательный и произвольный доступ к файлу в языке C++. Примеры применения.	УК-1; ПКС-1
63.	Основы ООП. Понятие инкапсуляции, наследования и полиморфизма.	УК-1; ПКС-1
64.	Класс в ООП и его основные компоненты.	УК-1; ПКС-1
65.	Перегрузка функций.	УК-1; ПКС-1
66.	Уровни доступа к элементам класса.	УК-1; ПКС-1
67.	Область видимости объектов, скрытие имен.	УК-1; ПКС-1
68.	Динамическая память, функции работы с памятью.	УК-1; ПКС-1
69.	Динамическая память, операции работы с памятью.	УК-1; ПКС-1
70.	Ссылки в C++. Отличие ссылок от переменных-указателей.	УК-1; ПКС-1
71.	Передача аргументов в функцию по умолчанию.	УК-1; ПКС-1
72.	Понятие класса, общая структура.	УК-1; ПКС-1
73.	Характеристика элементов-данных класса.	УК-1; ПКС-1
74.	Характеристика методов класса. Использование операции привязки «::»	УК-1; ПКС-1
75.	Указатель «this». Пример явного использования.	УК-1; ПКС-1
76.	Функции-друзья класса.	УК-1; ПКС-1
77.	Функции-конструкторы. Явный и косвенный вызов конструктора.	УК-1; ПКС-1
78.	Функции-деструкторы.	УК-1; ПКС-1
79.	Методы класса с атрибутом «const».	УК-1; ПКС-1

80.	Статические методы и данные. Атрибут «static»	УК-1; ПКС-1
81.	Указатели на компоненты класса	УК-1; ПКС-1
82.	Наследование. Базовый и производный классы.	УК-1; ПКС-1
83.	Инициализация объектов при наследовании.	УК-1; ПКС-1
84.	Указатели на производный и базовый классы. Формат явного преобразования указателей на базовый класс.	УК-1; ПКС-1
85.	Виртуальный базовый класс.	УК-1; ПКС-1
86.	Конструктор во множественном наследовании.	УК-1; ПКС-1
87.	Виртуальные функции. Переопределение виртуальных функций.	УК-1; ПКС-1
88.	Понятие абстрактного класса.	УК-1; ПКС-1
89.	Перегрузка операций.	УК-1; ПКС-1
90.	Особенности перегрузки операций при помощи методов класса и функций-друзей.	УК-1; ПКС-1
91.	Перегрузка методами класса.	УК-1; ПКС-1
92.	Использование ссылок при перегрузке унарных операций.	УК-1; ПКС-1
93.	Стандартная библиотека. Общая характеристика.	УК-1; ПКС-1
94.	Строковый класс стандартной библиотеки.	УК-1; ПКС-1
95.	Контейнерные классы.	УК-1; ПКС-1
96.	Итераторы.	УК-1; ПКС-1
97.	Алгоритмы.	УК-1; ПКС-1
98.	Потоковые классы.	УК-1; ПКС-1
99.	Управление выводом. Манипуляторы и флажки.	УК-1; ПКС-1
100.	Файловые потоки.	УК-1; ПКС-1
101.	Шаблоны функций	УК-1; ПКС-1
102.	Шаблоны классов.	УК-1; ПКС-1
103.	Обработка исключений. Общая характеристика.	УК-1; ПКС-1
104.	Вложенные классы.	УК-1; ПКС-1
105.	Классы и указатели при наследовании.	УК-1; ПКС-1
106.	Многократная перегрузка операций.	УК-1; ПКС-1
107.	Многоточие в качестве параметра функции.	УК-1; ПКС-1
108.	Указатель типа «void».	УК-1; ПКС-1
109.	Адрес в качестве возвращаемого значения функции	УК-1; ПКС-1
110.	Операция «typeid».	УК-1; ПКС-1
111.	Виды обработчиков исключительных операций.	УК-1; ПКС-1
112.	Отличие вызова функций от вызова обработчика исключительной ситуации.	УК-1; ПКС-1
113.	Создание собственного завершающего кода при перехвате исключительной ситуации.	УК-1; ПКС-1
114.	Форма конструктора со списком инициализации.	УК-1; ПКС-1