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.А. Шанибов
«___» _____ 202

# ФОНД ОЦЕНОЧНЫХ СРЕДСТВ (ОЦЕНОЧНЫХ МАТЕРИАЛОВ) ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

# «ОБЕСПЕЧЕНИЕ КАЧЕСТВА РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С++»

**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<br>**Уметь:** соотносить разнородные явления и систематизировать их в рамках проблемной ситуации в профессиональной деятельности.<br>**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.<br>**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.<br><br>**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<br><br>**ПКС-1.1.** У-1. Умеет ставить задачи исследования и оптимизации сложных объектов на основе методов<br><br>математического моделирования,<br><br>**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<br><br>(п. 5.2.2);<br><br>типовые оценочные материалы к зачету<br><br>(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<br><br>области прикладной математики и информатики<br><br>**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* ***(ПКС-2 ):***

*Тема 1. Синтаксис языка программирования С++. Отличия его от других языков.*

1. Основы синтаксиса языка С++.
2. Типы данных.
3. Типы литералов.
4. Объявление переменных и их инициализация.

*Тема 2. Основные операторы языка программирования С++.*

1. Операторы языка С++.
2. Запись выражений.
3. Приоритеты операторов.
4. Основные алгоритмические конструкции.

*Тема 3. Реализация функций на С++.*

1. Создание функций на С++.
2. Перегрузка функций.
3. Перегрузка операторов.

*Тема 4. Особенности объектно-ориентированного программирования на С++.*

1. Создание классов на С++.

2. Наследование и полиморфизм.
3. Механизм RTTI.

*Тема 5. Классы стандартной библиотеки C++.*

1. Классы потоков стандартного ввода-вывода.
2. Перегрузка операторов вывода в потоки.
3. Потоки вывода в строки и файлы.

*Тема 6. Стандартная библиотека шаблонов.*

1. Понятие шаблона.
2. Коллекции в C++.
3. Итераторы.

### *3.2. Оценочные материалы для контрольной работы:*
### **контролируемая компетенция** УК-1**,** ПКС-1**:**

*1.* Отметьте свойства языка C++, которые могут быть источниками возможных ошибок программирования:

a) наличие встроенных типов данных;
b) возможность создания абстрактных классов;
c) наличие указателей;
d) возможность динамического распределения памяти;
e) возможность определения символических констант;
f) возможность преобразования типов.

2. Какое приведение типов используется в следующем выражении? int a = 0; float f = 3.4; f += (int)a;

a) неявное приведение типов;
b) явное приведение типов;
c) стандартное приведение типов.

- Класс B наследован от класса A. Отметьте верное для класса B:

a) объект класса B может использоваться как объект базового класса;
b) класс B должен быть определен с ключевым словом derived;
c) класс B может непосредственно обращаться к внутренним атрибутам базового класса;
d) класс B наследует все операторы базового класса.

3. Что нужно сделать для освобождения памяти после выполнения такого кода? char *a; a = new char[20];

a) delete []a;
b) delete a[];
c) delete a.

4. Какая из записей является правильной записью абстрактного класса?

a) abstract class A {virtual int f () = 0;};
b) class A {virtual int f () = 0;};
c) class A {virtual int f ();}.

5. В чем заключается принцип полиморфизма?

a) в наличии виртуальных методов;
b) в наличии множественного наследования;

c) в использовании виртуального наследования.

6. Какая операция позволяет получить значение, записанное по адресу, который содержится в указателе?
a) *;
b) ? ;
c) ^ ;
d) &.

7. Операция "." обозначает
a) что атрибут объекта, следующий за этим оператором, будет изменен;
b) обращение к атрибуту объекта, используемое в специальных случаях;
c) обращение к атрибуту объекта.

8. Прототип функции задает
a) тип функции, включая количество и тип аргументов и тип результата;
b) возможность выполнения этой функции из программ на других языках программирования;
c) имя функции и минимальное количество параметров.

9. Как называется функция, которая вызывает саму себя?
a) конструктором;
b) деструктором;
c) подставляемой;
d) рекурсивной.

10. В каких выражениях используются унарные арифметические операции?
a) c1 + d2;
b) s2 % d % 2;
c) –b – B

11. В программе на языке Си++ обязательно имеется функция:
a) head;
b) start;
c) prime;
d) main;
e) finish.

12. Ключевое слово void обозначает что функция...
a) возвращает число с плавающей запятой;
b) возвращает целое число;
c) ничего не возвращает;
d) является главной.

13. Укажите зарезервированное ключевое слово для динамического выделения памяти
a) malloc;
b) new;
c) create;
d)value.

14. Выберите наиболее подходящее определение класса:
a) тип, содержащий набор функций;

b) тип, который отображает состояние некоторого объекта;
c) тип, описывающий поведение некоторой сущности;
d) тип, описывающий характеристики и поведение объекта.

15. Как называется способность объекта скрывать свои данные и реализацию от других объектов системы?
a) полиморфизм;
b) инкапсуляция;
c) абстракция;
d) наследование.

16. Каково преимущество использования ключевого слова const вместо директивы #define?
a) константу, определенную с помощью const, можно изменять во время работы
b) к константе, определенной с помощью const, можно применить операции инкремента и декремента;
c) константа, определенная с помощью const, доступна в других модулях программы;
d) константа, определенная с помощью const, имеет тип, и компилятор может проследить за ее использованием в соответствии с объявленным типом.

17. Конструктор класса - это метод, который вызывается при создании объекта для...
a) выделения памяти под динамические атрибуты класса;
b) выделения памяти под статические атрибуты класса;
c) инициализации атрибутов объекта;
d)загрузки методов класса в память.

18. Какими по умолчанию объявляются методы класса?
a) private;
b) public;
c) protected;
d) по умолчанию не объявляются.

19. Класс - это:
- любой тип данных, определяемый пользователем
+ тип данных, определяемый пользователем и сочетающий в себе данные и функции их обработки
- структура, для которой в программе имеются функции работы с нею

20. Членами класса могут быть
+как переменные, так и функции, могут быть объявлены как private и как public
- только переменные, объявленные как private
- только функции, объявленные как private
- только переменные и функции, объявленные как private
- только переменные и функции, объявленные как public

21. Что называется конструктором?
+ метод, имя которого совпадает с именем класса и который вызывается автоматически при создании объекта класса
- метод, имя которого совпадает с именем класса и который вызывается автоматически при объявлении класса (до создания объекта класса)

- метод, имя которого необязательно совпадает с именем класса и который вызывается при создании объекта класса
- метод, имя которого совпадает с именем класса и который необходимо явно вызывать из головной программы при объявлении объекта класса

22. Объект - это
- переменная, содержащая указатель на класс
+ экземпляр класса
- класс, который содержит в себе данные и методы их обработки

23. Отметьте правильные утверждения
+ конструкторы класса не наследуются
- конструкторов класса может быть несколько, их синтаксис определяется программистом
+ конструкторов класса может быть несколько, но их синтаксис должен подчиняться правилам перегрузки функций
- конструктор возвращает указатель на объект
+ конструктор не возвращает значение

24. Что называется деструктором?
- метод, который уничтожает объект
- метод, который удаляет объект
+ метод, который освобождает память, занимаемую объектом
- системная функция, которая освобождает память, занимаемую объектом

25. Выберите правильные утверждения
+ у конструктора могут быть параметры
- конструктор наследуется, но должен быть перегружен
- конструктор должен явно вызываться всегда перед объявлением объекта
+ конструктор вызывается автоматически при объявлении объекта
- объявление каждого класса должно содержать свой конструктор
+ если конструктор не создан, компилятор создаст его автоматически

26. Выберите правильные утверждения
- деструктор - это метод класса, применяемый для удаления объекта
+ деструктор - это метод класса, применяемый для освобождения памяти, занимаемой объектом
- деструктор - это отдельная функция головной программы, применяемая для освобождения памяти, занимаемой объектом
+деструктор не наследуется
- деструктор наследуется, но должен быть перегружен

27. Что называется наследованием?
+ это механизм, посредством которого производный класс получает элементы родительского и может дополнять либо изменять их свойства и методы
- это механизм переопределения методов базового класса
- это механизм, посредством которого производный класс получает все поля базового класса
- это механизм, посредством которого производный класс получает элементы родительского, может их дополнить, но не может переопределить

28. Выберите правильное объявление производного класса
- class MoreDetails:: Details;

- class MoreDetails: public class Details;
+ class MoreDetails: public Details;
- class MoreDetails: class(Details);

29. Выберите правильные утверждения:
- если элементы класса объявлены как private, то они доступны только наследникам класса, но не внешним функциям
+ если элементы класса объявлены как private, то они недоступны ни наследникам класса, ни внешним функциям
- если элементы объявлены как public, то они доступны наследникам класса, но не внешним функциям
+ если элементы объявлены как public, то они доступны и наследникам класса, и внешним функциям

30. Возможность и способ обращения производного класса к элементам базового определяется
- ключами доступа: private, public, protected в теле производного класса
- только ключом доступа protected в заголовке объявления производного класса
+ ключами доступа: private, public, protected в заголовке объявления производного класса
- ключами доступа: private, public, protected в теле базового класса

31. Выберите правильные соответствия между спецификатором базового класса, ключом доступа в объявлении производного класса и правами доступа прозводного класса к элементам базового
- ключ доступа - public; в базовом классе: private; права доступа в производном классе - protected
+ ключ доступа - любой; в базовом классе: private; права доступа в производном классе - нет прав
+ ключ доступа - protected или public ; в базовом классе: protected; права доступа в производном классе - protected
- ключ доступа - private; в базовом классе: public; права доступа в производном классе - public
+ ключ доступа – любой; в базовом классе: public; права доступа в производном классе – такие же, как ключ доступа

32. Дружественная функция - это
- функция другого класса, среди аргументов которой есть элементы данного класса
+ функция, объявленная в классе с атрибутом friend, но не являющаяся членом класса;
- функция, являющаяся членом класса и объявленная с атрибутом friend;
- функция, которая в другом классе объявлена как дружественная данному

33. Выберите правильные утверждения:
+ одна функция может быть дружественной нескольким классам
- дружественная функция не может быть обычной функцией, а только методом другого класса
+ дружественная функция объявляется внутри класса, к элементам которого ей нужен доступ
- дружественная функция не может быть методом другого класса

34. Шаблон функции - это...
+ определение функции, в которой типу обрабатываемых данных присвоено условное обозначение

- прототип функции, в котором вместо имен параметров указан условный тип
- определение функции, в котором указаны возможные варианты типов обрабатываемых параметров
- определение функции, в котором в прототипе указан условный тип, а в определении указаны варианты типов обрабатываемых параметров

35. Переопределение операций имеет вид:
- имя_класса, ключевое слово operation, символ операции
+ имя_класса, ключевое слово operator, символ операции, в круглых скобках могут быть указаны аргументы
- имя_класса, ключевое слово operator, список аргументов
- имя_класса, два двоеточия, ключевое слово operator, символ операции

36. Для доступа к элементам объекта используются:
+ при обращении через имя объекта – точка, при обращении через указатель – операция «->»
- при обращении через имя объекта – два двоеточия, при обращении через указатель – операция «точка»
- при обращении через имя объекта – точка, при обращении через указатель – два двоеточия
- при обращении через имя объекта – два двоеточия, при обращении через указатель – операция «->»

37. Полиморфизм – это :
+ средство, позволяющее использовать одно имя для обозначения действий, общих для родственных классов
- средство, позволяющее в одном классе использовать методы с одинаковыми именами;
- средство, позволяющее в одном классе использовать методы с разными именами для выполнения одинаковых действий
- средство, позволяющее перегружать функции для работы с разными типами или разным количеством аргументов.

38. Полиморфизм реализован через механизмы:
+ перегрузки функций, виртуальных функций, шаблонов
- перегрузки функций, наследования методов, шаблонов;
- наследования методов, виртуальных функций, шаблонов
- перегрузки функций, наследования, виртуальных функций.

39. Виртуальными называются функции:
+ функции базового класса, которые могут быть переопределены в производном классе
- функции базового класса, которые не используются в производном классе;
- функции базового класса, которые не могут быть переопределены в базовом классе;
- функции производного класса, переопределенные относительно базового класса

40. Выберите правильный вариант выделения динамической памяти под переменную X типа float:
+ float *ptr = new float; X = *ptr;
- float & ptr = new float; X = & ptr;
- float * ptr = &X; X = new float;

41. Полиморфизм в объектно-ориентированном программировании реализуется:
+ через механизмы перегрузки (функций и операций), виртуальные функции и шаблоны

- через механизмы перегрузки (функций и операций) и шаблоны;
- через виртуальные функции и шаблоны;
- через механизмы перегрузки (функций и операций) и виртуальные функции

42. Дано определение класса

```
class monstr {
int health, armo;
monstr(int he, int arm);
public:
monstr(int he=50, int arm=10);
int color;
}
```

Укажите свойства и методы, доступные внешним функциям
- health, armo
monstr(int he, int arm);
monstr(int he=50, int arm=10);
+ int color;
monstr(int he=50, int arm=10);
- health, armo, color
monstr(int he=50, int arm=10);
- int color;
monstr(int he, int arm);

43. Функция вычисляет произведение двух чисел. Исходные данные вводятся с клавиатуры. Какие проверки целесообразно ввести в программе:
+ проверка, что исходные данные являются числами
- проверки не нужны, все возможные ошибки отловит компилятор
- проверка исходных данных на равенство нулю

44. Для чего предназначен оператор namespace:
- для использования классов, переменных и функций из других модулей программы без использования заголовочных файлов
+ для заключения в группу объявлений классов, переменных и функций в отдельный контекст со своим именем
- для заключения в группу объявлений классов, переменных и функций для использования только в текущем модуле

45. Если определена операция вычитания для двух объектов класса A, а операция преобразования к int не определена, что будет вызвано при: A a1,a2,a3=5; a3 = a1 – a2;
- только операция вычитания
+ произойдет ошибка
- преобразование к целому

46. Какой из наборов перечисляемых значений записан правильно:
- enum { a, b = 3, c = 4, 3 };
- enum { a, b, 3, 4 };
+ enum {a, b = 3, c, d };

47. В чем различие использования следующих выражений #include <…> и #include «…»:
+ различие заключается в методе поиска препроцессором включаемого файла
- в различии использования заголовочных и исходных файлов

- нет различий

48. Чему будет равен результат вычисления выражения: int d=5; bool b = true, c; c = (!b||(d>3)):
- Ошибка компилятора
- false
+ true

49. Если в арифметическом выражении участвуют целый и вещественный операнды, то:
- ошибка компиляции
+ целый тип приводится к вещественному
- вещественный тип приводится к целому

50. Укажите в каком выражении произойдет потеря точности:
+ int i; float x = 134, y = 14; i = x/y;
- short i = 0x3; float x = 7, v; v = i + x;
- float M = 2 2; double Z = 3; Z *= M;

51. Если после выражения стоит точка с запятой, то:
- выражение вычисляется, а его значение запоминается в специальной переменной, которую можно использовать в следующем операторе
+ это оператор-выражение, действие которого заключается в вычислении выражения
- выражение вычисляется только если первой стоит операция присваивания

52. Что из себя представляет динамическое выделение памяти:
+ память под объект (переменную) может выделяться не сразу, а в процессе работы программы, освобождение памяти производится вручную
- память под объект (переменную) может выделяться не сразу, а в процессе работы программы, освобождение памяти производится автоматически после завершения программы
- память под объект (переменную) выделяется каждый раз при обращении к переменной

53. Отметьте истинное высказывание:
- переменная инициализируется, потом объявляется
- переменная объявляется, потом инициализируется и изменяется
+ переменная объявляется, потом изменяется

54. Какие операции поддаются перегрузке:
+ унарные и бинарные
- только бинарные
- только унарные

55. Переменная типа signed char может принимать значения:
- только символов английского алфавита, цифр и символа подчеркивания
+ из первой половины кодовой таблицы
+ только из алфавита языка C+

56. Переменная типа signed char может принимать значения: [
+ только из алфавита языка C+
- только символов английского алфавита, цифр и символа подчеркивания
+ от -128 до 127

57. В переменной типа unsigned char можно хранить число:
-213
+ 213
- 1213

58. Чему равно числовое значение выражения e/2*a-abs(e)*1e0 при e = 4, a = 2:
- 3
+ 0
- 1

59. Чему равно значение выражения (a && ! b || c), где a, b и c -величины типа bool, имеющие значения false, true и true соответственно:
- false
- yes
+ true

60. Какое выражение не содержат синтаксических ошибок:
- a*exp(t)(2t)
- sin(abs(0. 6(e*3))
+ 0XCC00*. 34E-4/_do/k-2

***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 С++»

| | Вопрос | Код компетенции (согласно РПД) |
|---|---|---|
| *2.* | Структура программы на языке С/С++. | УК-1; ПКС-1 |
| *3.* | Концепции восходящего и нисходящего программирования. | УК-1; ПКС-1 |
| *4.* | Понятия объекта и класса в объектно-ориентированном программировании. | УК-1; ПКС-1 |
| *5.* | Принцип инкапсуляции в объектно-ориентированном программировании. | УК-1; ПКС-1 |
| *6.* | Наследование в объектно-ориентированном программировании. | УК-1; ПКС-1 |
| *7.* | Понятие виртуальных методов в объектно-ориентированном программировании. | УК-1; ПКС-1 |

| | | |
|---|---|---|
| *8.* | Понятие компонента и принцип компонентного программирования. | УК-1; ПКС-1 |
| *9.* | Что называется, свойством компонента? Понятие события компонента. | УК-1; ПКС-1 |
| *10.* | Классы в языке C++. Инкапсуляция. Описание класса. Рекомендации по составу класса. | УК-1; ПКС-1 |
| *11.* | Конструкторы и деструкторы классов в языке C++. | УК-1; ПКС-1 |
| *12.* | Структуры, объединения и классы в языке C++. Общее и отличия. Примеры применения. | УК-1; ПКС-1 |
| *13.* | Конструктор копирования в языке C++. Поверхностное и глубинное копирование. Конструкторы и присваивание строк. Примеры применения. | УК-1; ПКС-1 |
| *14.* | Указатели на элементы классов в языке C++. Указатель *this*. | УК-1; ПКС-1 |
| *15.* | Константные поля и методы класса в языке C++. Примеры применения. | УК-1; ПКС-1 |
| *16.* | Статические элементы класса в языке C++. Статические поля и статические методы. Примеры применения. | УК-1; ПКС-1 |
| *17.* | Полиморфизм в языке C++. Виртуальные методы классов. Примеры применения. | УК-1; ПКС-1 |
| *18.* | Иерархия наследования классов в языке C++. Доступ к членам базовых классов. Ключи доступа. Примеры применения. | УК-1; ПКС-1 |
| *19.* | Простое и множественное наследование в языке C++. Примеры применения. | УК-1; ПКС-1 |
| *20.* | Виртуальные базовые классы в языке C++. Примеры применения. | УК-1; ПКС-1 |
| *21.* | Виртуальные деструкторы в языке C++. Примеры применения. | УК-1; ПКС-1 |
| *22.* | Друзья класса в языке C++. Дружественные функции и поля. Дружественный класс. Примеры применения. | УК-1; ПКС-1 |
| *23.* | Механизм позднего связывания в языке C++. Примеры применения. | УК-1; ПКС-1 |
| *24.* | Абстрактные классы в языке C++. Абстрактные методы и классы. Примеры применения. | УК-1; ПКС-1 |
| *25.* | Шаблоны функций в языке C++. Объявление, определение, параметры, возвращаемые значения. Достоинства и недостатки. Примеры применения. | УК-1; ПКС-1 |
| *26.* | Шаблоны структур и объединений в языке C++. Определение и инициализация структур-переменных. Примеры применения. | УК-1; ПКС-1 |
| *27.* | Шаблоны классов в языке C++. Создание и использование. Специализация шаблонов классов. Достоинства и недостатки шаблонов классов. Примеры применения. | УК-1; ПКС-1 |
| *28.* | Перегрузка унарных и бинарных операций в языке C++. Примеры применения. | УК-1; ПКС-1 |
| *29.* | Перегрузка операций присваивания в языке C++. Примеры применения. | УК-1; ПКС-1 |
| *30.* | Перегрузка операций new и delete в языке C++. Примеры применения. | УК-1; ПКС-1 |
| *31.* | Перегрузка операции вызова функции в языке C++. Примеры применения. | УК-1; ПКС-1 |
| *32.* | Перегрузка операции индексирования в языке C++. Примеры применения. | УК-1; ПКС-1 |
| *33.* | Перегрузка операций приведения типа в языке C++. Примеры применения. | УК-1; ПКС-1 |

| | | |
|---|---|---|
| *34.* | Динамическое определение типа и преобразование типов. Повышающие и понижающие преобразования. Примеры применения. | УК-1; ПКС-1 |
| *35.* | Преобразование ссылок в языке C++. Перекрестное преобразование. Примеры применения. | УК-1; ПКС-1 |
| *36.* | Преобразование типов в языке C++: операции const_cast, static_cast, dynamic_cast, reinterpret_cast. Примеры применения. | УК-1; ПКС-1 |
| *37.* | Обработка исключительных ситуаций в языке C++. Иерархии исключений. Общий механизм обработки исключений. Примеры применения. | УК-1; ПКС-1 |
| *38.* | Синтаксис исключений в языке C++. Список исключений функции. Примеры применения. | УК-1; ПКС-1 |
| *39.* | Исключения в конструкторах и деструкторах языка C++. Перехват исключений. Примеры применения. | УК-1; ПКС-1 |
| *40.* | Потоковые классы в языке C++. Стандартные потоки. Форматирование данных. Флаги и форматирующие методы манипуляторы. Примеры применения. | УК-1; ПКС-1 |
| *41.* | Строковые потоки в языке C++. Примеры применения. | УК-1; ПКС-1 |
| *42.* | Ввод-вывод встроенных (стандартных) типов в языке C++. Примеры применения. | УК-1; ПКС-1 |
| *43.* | Состояния предопределенных объектов (потоков) в языке C++. Ошибки потоков. Примеры применения. | УК-1; ПКС-1 |
| *44.* | Потоки и типы, определенные пользователем в языке C++. Ввод-вывод типов, определенных пользователем. Примеры применения. | УК-1; ПКС-1 |
| *45.* | Форматированный ввод-вывод в языке C++. Манипуляторы. Примеры применения. | УК-1; ПКС-1 |
| *46.* | Файловый ввод-вывод в языке C++. Файловые потоки. Примеры применения. | УК-1; ПКС-1 |
| *47.* | Контейнерные классы в языке C++. Примеры применения. | УК-1; ПКС-1 |
| *48.* | Последовательные контейнеры в языке C++. Вектор (*vector*). Вектор логических значений *vector<bool>*. Примеры применения. | УК-1; ПКС-1 |
| *49.* | Последовательные контейнеры в языке C++. Двусторонняя очередь (*deque*). Примеры применения. | УК-1; ПКС-1 |
| *50.* | Последовательные контейнеры в языке C++. Список (*list*). Примеры применения. | УК-1; ПКС-1 |
| *51.* | Адаптеры последовательных контейнеров в языке C++. Стек (*stack*). Примеры применения. | УК-1; ПКС-1 |
| *52.* | Адаптеры последовательных контейнеров в языке C++. Очередь *FIFO (queue)*. Примеры применения. | УК-1; ПКС-1 |
| *53.* | Адаптеры последовательных контейнеров в языке C++. Очередь с приоритетами (*prioriy_queue*). Примеры применения. | УК-1; ПКС-1 |
| *54.* | Ассоциативные контейнеры в языке C++. Словари (*map*), словари с дубликатами (*multimap*). Примеры применения. | УК-1; ПКС-1 |
| *55.* | Асоциативные контейнеры в языке C++. Множества (*set*), множества с дубликатами (*multiset*), битовые множества (*bitset*). Примеры применения. | УК-1; ПКС-1 |
| *56.* | Итераторы в языке C++. Обратные итераторы. Итераторы вставки. Потоковые итераторы. Примеры применения. | УК-1; ПКС-1 |
| *57.* | Адаптеры указателей на функции языка C++. Адаптеры методов. Примеры применения. | УК-1; ПКС-1 |

| | | |
|---|---|---|
| *58.* | Модифицирующие операции с последовательностями в языке C++. Примеры применения. | УК-1; ПКС-1 |
| *59.* | Средства для численных расчетов и работы с комплексными числами в языке C++. Примеры применения. | УК-1; ПКС-1 |
| *60.* | Средства поддержки языка и локализации. Примеры применения. | УК-1; ПКС-1 |
| *61.* | Текстовые и бинарные файлы в языке C++. Связывание файловых переменных с внешней средой. Примеры применения. | УК-1; ПКС-1 |
| *62.* | Типовые действия с файлами в языке C++: создание, открытие, закрытие, чтение и изменение. Примеры применения. | УК-1; ПКС-1 |
| *63.* | Последовательный и произвольный доступ к файлу в языке C++. Примеры применения. | УК-1; ПКС-1 |
| *64.* | Основы ООП. Понятие инкапсуляции, наследования и полиморфизма. | УК-1; ПКС-1 |
| *65.* | Класс в ООП и его основные компоненты. | УК-1; ПКС-1 |
| *66.* | Перегрузка функций. | УК-1; ПКС-1 |
| *67.* | Уровни доступа к элементам класса. | УК-1; ПКС-1 |
| *68.* | Область видимости объектов, скрытие имен. | УК-1; ПКС-1 |
| *69.* | Динамическая память, функции работы с памятью. | УК-1; ПКС-1 |
| *70.* | Динамическая память, операции работы с памятью. | УК-1; ПКС-1 |
| *71.* | Ссылки в C++. Отличие ссылок от переменных-указателей. | УК-1; ПКС-1 |
| *72.* | Передача аргументов в функцию по умолчанию. | УК-1; ПКС-1 |
| *73.* | Понятие класса, общая структура. | УК-1; ПКС-1 |
| *74.* | Характеристика элементов-данных класса. | УК-1; ПКС-1 |
| *75.* | Характеристика методов класса. Использование операции привязки «::» | УК-1; ПКС-1 |
| *76.* | Указатель «this». Пример явного использования. | УК-1; ПКС-1 |
| *77.* | Функции-друзья класса. | УК-1; ПКС-1 |
| *78.* | Функции-конструкторы. Явный и косвенный вызов конструктора. | УК-1; ПКС-1 |
| *79.* | Функции-деструкторы. | УК-1; ПКС-1 |
| *80.* | Методы класса с атрибутом «const». | УК-1; ПКС-1 |
| *81.* | Статические методы и данные. Атрибут «static» | УК-1; ПКС-1 |
| *82.* | Указатели на компоненты класса | УК-1; ПКС-1 |
| *83.* | Наследование. Базовый и производный классы. | УК-1; ПКС-1 |
| *84.* | Инициализация объектов при наследовании. | УК-1; ПКС-1 |
| *85.* | Указатели на производный и базовый классы. Формат явного преобразования указателей на базовый класс. | УК-1; ПКС-1 |
| *86.* | Виртуальный базовый класс. | УК-1; ПКС-1 |
| *87.* | Конструктор во множественном наследовании. | УК-1; ПКС-1 |
| *88.* | Виртуальные функции. Переопределение виртуальных функций. | УК-1; ПКС-1 |
| *89.* | Понятие абстрактного класса. | УК-1; ПКС-1 |
| *90.* | Перегрузка операций. | УК-1; ПКС-1 |
| *91.* | Особенности перегрузки операций при помощи методов класса и функций-друзей. | УК-1; ПКС-1 |
| *92.* | Перегрузка методами класса. | УК-1; ПКС-1 |
| *93.* | Использование ссылок при перегрузке унарных операций. | УК-1; ПКС-1 |
| *94.* | Стандартная библиотека. Общая характеристика. | УК-1; ПКС-1 |
| *95.* | Строковый класс стандартной библиотеки. | УК-1; ПКС-1 |
| *96.* | Контейнерные классы. | УК-1; ПКС-1 |
| *97.* | Итераторы. | УК-1; ПКС-1 |
| *98.* | Алгоритмы. | УК-1; ПКС-1 |

| | |
|---|---|
| *99.* Потоковые классы. | УК-1; ПКС-1 |
| *100.* Управление выводом. Манипуляторы и флажки. | УК-1; ПКС-1 |
| *101.* Файловые потоки. | УК-1; ПКС-1 |
| *102.* Шаблоны функций | УК-1; ПКС-1 |
| *103.* Шаблоны классов. | УК-1; ПКС-1 |
| *104.* Обработка исключений. Общая характеристика. | УК-1; ПКС-1 |
| *105.* Вложенные классы. | УК-1; ПКС-1 |
| *106.* Классы и указатели при наследовании. | УК-1; ПКС-1 |
| *107.* Многократная перегрузка операций. | УК-1; ПКС-1 |
| *108.* Многоточие в качестве параметра функции. | УК-1; ПКС-1 |
| *109.* Указатель типа «void». | УК-1; ПКС-1 |
| *110.* Адрес в качестве возвращаемого значения функции | УК-1; ПКС-1 |
| *111.* Операция «typeid». | УК-1; ПКС-1 |
| *112.* Виды обработчиков исключительных операций. | УК-1; ПКС-1 |
| *113.* Отличие вызова функций от вызова обработчика исключительной ситуации. | УК-1; ПКС-1 |
| *114.* Создание собственного завершающего кода при перехвате исключительной ситуации. | УК-1; ПКС-1 |
| *115.* Форма конструктора со списком инициализации. | УК-1; ПКС-1 |