МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Ф. УТКИНА

Кафедра «Автоматики и информационных технологий в управлении»

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ДИСЦИПЛИНЫ

Объектно-ориентированное программирование

Направление 27.03.04
«Управление в технических системах»

ОПОП
«Управление в технических системах»

Квалификация выпускника – бакалавр

Формы обучения – очная

Рязань 2023 г.

Оценочные материалы – это совокупность учебно-методических материалов (контрольных вопросов, описаний форм и процедур), предназначенных для оценки качества освоения обучающимися данной дисциплины как части основной профессиональной образовательной программы.

Цель – оценить соответствие знаний, умений и уровня приобретенных компетенций обучающихся целям и требованиям основной профессиональной образовательной программы в ходе проведения текущего контроля и промежуточной аттестации.

Основная задача – обеспечить оценку уровня сформированности общекультурных, общепрофессиональных и профессиональных компетенций, приобретаемых обучающимся в соответствии с этими требованиями.

Контроль знаний обучающихся проводится в форме текущего контроля и промежуточной аттестации.

Текущий контроль успеваемости проводится с целью определения степени усвоения учебного материала, своевременного выявления и устранения недостатков в подготовке обучающихся и принятия необходимых мер по совершенствованию методики преподавания учебной дисциплины, организации работы обучающихся в ходе учебных занятий и оказания им индивидуальной помощи.

К контролю текущей успеваемости относятся проверка знаний, умений и навыков обучающихся: на занятиях; по результатам выполнения лабораторных и практических работ; по результатам выполнения обучающимися индивидуальных заданий.

При оценивании результатов освоения практических занятий и лабораторных работ применяется шкала оценки «зачтено – не зачтено». Количество лабораторных и практических работ и их тематика определена рабочей программой дисциплины, утвержденной заведующим кафедрой.

Результат выполнения каждого индивидуального задания должен соответствовать всем критериям оценки в соответствии с компетенциями, установленными для заданного раздела дисциплины.

Промежуточная аттестация по дисциплине осуществляется проведением теоретического зачета и экзамена.

Форма проведения теоретического зачета – устный ответ по утвержденным экзаменационным билетам. В экзаменационный билет включается два вопроса по темам курса.

Форма проведения экзамена – письменный ответ по утвержденным экзаменационным билетам. В экзаменационный билет включается два вопроса по темам курса и одна практическая задача. После выполнения обучаемым письменной работы производится ее оценка преподавателем и, при необходимости, проводится теоретическая беседа с обучаемым для уточнения экзаменационной оценки.

Паспорт оценочных материалов по дисциплине

№ п/п	Контролируемые разделы (темы) дисциплины	Код контролируемой компетенции (или её части)	Вид, метод, форма оценочного мероприятия
Раздел 1. Семестр 5			
1	Введение в объектно-ориентированное программирование	ОПК-6.1; ПК-1.1	Зачет, экзамен
2	Встроенные типы данных в языке C++	ОПК-6.1; ПК-1.1	Зачет, экзамен
3	Имена, стандартные операции и управляющие конструкции	ОПК-6.1; ПК-1.1	Зачет, экзамен
4	Указатели, массивы, ссылки, константы	ОПК-6.1; ПК-1.1	Зачет, экзамен
5	Типы данных, определяемые пользователем. Функции	ОПК-6.1; ПК-1.1	Зачет, экзамен
6	Приведение типов. Модель памяти программы	ОПК-6.1; ПК-1.1	Зачет, экзамен
7	Многофайловые программы. Директивы компилятора	ОПК-6.1; ПК-1.1	Зачет, экзамен
8	Основы работы с классами	ОПК-6.1; ПК-1.1	Зачет, экзамен
Раздел 2. Семестр 6			
1	Перегрузка операций, друзья класса	ОПК-6.1; ПК-1.1	Экзамен
2	Наследование классов		Экзамен
3	Обобщенное программирование, шаблоны	ОПК-6.1; ПК-1.1	Экзамен
4	Обработка исключительных ситуаций	ОПК-6.1; ПК-1.1	Экзамен
5	Принципы организации стандартной библиотеки языка C++	ОПК-6.1; ПК-1.1	Экзамен
6	Последовательные контейнеры	ОПК-6.1; ПК-1.1	Экзамен
7	Ассоциативные контейнеры	ОПК-6.1; ПК-1.1	Экзамен
8	Адаптеры последовательных контейнеров	ОПК-6.1; ПК-1.1	Экзамен
9	Итераторы, функциональные объекты	ОПК-6.1; ПК-1.1	Экзамен
10	Алгоритмы стандартной библиотеки	ОПК-6.1; ПК-1.1	Экзамен

Критерии оценивания компетенций (результатов)

Учитываются следующие показатели:

1) Уровень усвоения материала, предусмотренного программой дисциплины.

2) Умение анализировать материал, устанавливать причинно-следственные связи.

3) Ответы на вопросы: полнота, аргументированность, убежденность, умение.

4) Качество ответа: его общая композиция, логичность, убежденность, общая эрудиция.

5) Использование основной и дополнительной литературы при подготовке ответов.

6) Качество выполнения лабораторных и практических работ.

7) Правильность выполненной контрольной работы.

Критерии оценки результата:

1) Обучающийся должен продемонстрировать знание современных тенденций использования объектно-ориентированного подхода при разработке программного обеспечения.

2) Обучающийся должен продемонстрировать умение решать задачи обработки данных с использованием средств языка C++, объективно оценивать результаты исследований.

3) Обучающийся должен продемонстрировать знание базовых принципов объектно-ориентированного программирования в ходе защиты проекта программы.

4) Обучающийся должен продемонстрировать умение применять теорию и методы объектно-ориентированного программирования для решения задач компьютерной обработки информации с использованием средств языка C++.

5) Обучающийся должен обеспечить соответствие структуры и содержания выполненного задания объектно-ориентированному подходу при проектировании и разработке программ.

6) Обучающийся должен продемонстрировать владение навыками объектно-ориентированного программирования в своей профессиональной деятельности.

7) Обучающийся должен продемонстрировать умение создавать программы, устойчивые к возникновению исключительных ситуаций и аппаратных сбоев.

8) Обучающийся должен продемонстрировать знание принципов объектно-ориентированной разработки программного обеспечения для моделирования процессов и объектов автоматизации и управления.

Оценка сформированности компетенций при текущем контроле

В рамках текущего контроля на протяжении семестра в качестве оценочных средств используются устные и письменные ответы студентов на индивидуальные вопросы, письменное тестирование по теоретическим разделам курса, отчеты о выполнении практических заданий, отчеты о выполнении лабораторных работ и результаты их защиты.

Оценка степени формирования контролируемых компетенций у обучающихся на различных этапах их формирования проводится преподавателем во время лекций, практических занятий и лабораторных работ по шкале оценок «зачтено», «не зачтено».

Устанавливаются следующие уровни сформированности компетенций в рамках текущего контроля:

1) 0%-80% оценок «зачтено» соответствует неудовлетворительному уровню сформированности компетенций.

2) 81%-90% оценок «зачтено» соответствует пороговому уровню сформированности компетенций.

3) 91%-100% оценок «зачтено» соответствует продвинутому уровню сформированности компетенций.

Уровень сформированности компетенций, оцененный в рамках текущего контроля, учитывается при прохождении промежуточной аттестации по данной дисциплине. Студенты, имеющие уровень сформированности компетенций ниже продвинутого, могут исправить свои оценки в установленном порядке.

Оценка сформированности компетенций при промежуточной аттестации

Формами промежуточной аттестации по данной дисциплине являются теоретический зачет и экзамен.

Теоретический зачет организуется и осуществляется в форме устного собеседования. Средством, определяющим содержание собеседования студента с экзаменатором, является утвержденный билет, в который включается два вопроса по темам курса согласно настоящей рабочей программе. Оценке на заключительной стадии зачета подвергаются устный ответ студента на вопросы билета, ответы на дополнительные вопросы экзаменатора.

В процессе оценки знаний, умений и навыков студента, производимой на этапе промежуточной аттестации в форме теоретического зачета, используется оценочная шкала «зачтено», «не зачтено», что соответствует шкале «компетенции студента соответствуют требованиям ФГОС ВО», «компетенции студента не соответствуют требованиям ФГОС ВО»:

Оценка «зачтено» выставляется студенту, который:

– по результатам текущего контроля имеет уровень сформированности компетенций не ниже порогового;

– показал полные и твёрдые знания материала дисциплины;

– правильно и аргументировано ответил на все вопросы, с приведением примеров;

– владеет приемами рассуждения и сопоставляет материал из разных источников;

– продемонстрировал понимание сущности обсуждаемых вопросов и допустил несущественные ошибки в ответах на дополнительные вопросы.

Дополнительным условием получения оценки «зачтено» могут стать хорошие успехи при выполнении самостоятельных и лабораторных работ, систематическая активная работа на практических занятиях.

Оценка «не зачтено» выставляется студенту, который:

– по результатам текущего контроля имеет неудовлетворительный уровень сформированности компетенций;

– не ответил на один из вопросов билета;

– допустил существенные ошибки в ответах на вопросы;

– продемонстрировал отсутствие знаний значительной части материала дисциплины;

– допустил существенные и грубые ошибки в ответах на дополнительные вопросы, предложенные преподавателем;

– продемонстрировал отсутствие целостного представления о взаимосвязях элементов курса и использования предметной терминологии.

Экзамен заключается в письменном ответе студента по утвержденному экзаменационному билету. В процессе оценки знаний, умений и навыков студента, производимой на этапе промежуточной аттестации в форме экзамена, используется следующая оценочная шкала: «отлично», «хорошо», «удовлетворительно» и «неудовлетворительно», что соответствует шкале «компетенции студента полностью соответствуют требованиям ФГОС ВО», «компетенции студента соответствуют требованиям ФГОС ВО», «компетенции студента в основном соответствуют требованиям ФГОС ВО» и «компетенции студента не соответствуют требованиям ФГОС ВО»:

Оценка «Отлично» выставляется студенту, который:

– по результатам текущего контроля имеет уровень сформированности компетенций не ниже порогового;

– продемонстрировал всестороннее, систематическое и глубокое знание учебно-программного материала дисциплины, умение успешно выполнять задания, предусмотренные программой;

– усвоил основную и ознакомился с дополнительной литературой, рекомендованной программой.

Оценка «отлично» выставляется студентам, усвоившим взаимосвязь основных понятий дисциплины в их значении для приобретаемой профессии; способным исчерпывающе, последовательно, грамотно и логически стройно изложить теоретический материал, безупречно ответить на дополнительные вопросы в рамках рабочей программы дисциплины.

Оценка «Хорошо» выставляется студенту, который:

– по результатам текущего контроля имеет уровень сформированности компетенций не ниже порогового;

– продемонстрировал полное знание учебно-программного материала дисциплины, умение успешно выполнять предусмотренные программой задания;

– усвоил основную литературу, рекомендованную в программе.

Оценка «хорошо» выставляется студентам, показавшим систематический характер знаний по дисциплине и способным к их самостоятельному пополнению и обновлению в ходе дальнейшей профессиональной деятельности; продемонстрировавшим знание всех основных теоретических понятий.

Оценка «Удовлетворительно» выставляется студенту, который:

– по результатам текущего контроля имеет уровень сформированности компетенций не ниже порогового;

– продемонстрировал общее знание основного учебно-программного материала дисциплины в объеме, необходимом для дальнейшей учебы и предстоящей работы по специальности;

– справился с выполнением заданий, предусмотренных программой;

– ознакомился с основной литературой, рекомендованной программой.

Оценка «удовлетворительно» выставляется студентам, допустившим ошибки в ответе на экзамене, но обладающим необходимыми знаниями для их устранения под руководством преподавателя, либо способным ответить на дополнительные вопросы того же раздела дисциплины.

Оценка «Неудовлетворительно» выставляется студенту, который:

– по результатам текущего контроля имеет неудовлетворительный уровень сформированности компетенций;

– продемонстрировал незнание значительной части основного учебно-программного материала дисциплины;

– допустил принципиальные ошибки в выполнении предусмотренных программой заданий;

– показал отсутствие навыков в обосновании выдвигаемых предложений;

– допустил существенные ошибки при изложении учебного материала.

Оценка «неудовлетворительно» выставляется студентам, которые не могут продолжить обучение по данной образовательной программе или приступить к профессиональной деятельности без дополнительных занятий по соответствующей дисциплине, а также, если студент после начала экзамена отказался его сдавать или нарушил правила защиты (не самостоятельно работал, обманом пытался получить более высокую оценку и т.д.).

Типовые контрольные задания или иные материалы

Контрольные вопросы к лабораторным работам по дисциплине

1) Какие фундаментальные типы данных имеются в языке C++?
2) Какие существуют особенности объявления имен?
3) Что такое пространство имен?
4) Какие операторы используются для ветвления?
5) Какие операторы используются для организации циклов?
6) Какие существуют арифметические и логические операции?
7) Что представляет собой значение указателя?
8) Какие переменные называются динамическими?
9) Как можно получить адрес объекта?
10) Что такое ссылка на переменную?
11) Каким образом можно получить значение по указателю на него?
12) Что такое константный указатель и указатель на константные данные?
13) Что такое объявление и определение функции?
14) Какими способами могут передаваться аргументы в функцию?
15) Как можно осуществить досрочный выход из функции?
16) Зачем применяется директива #define?
17) Чем обычно отличаются заголовочные файлы от фалов с исходным кодом?
18) Зачем применяется директива #include?
19) Какие существуют способы защиты от множественного включения заголовочных файлов?

20) В чем заключаются базовые принципы объектно-ориентированного программирования?
21) Что такое класс? Что такое объект?
22) Как связаны между собой классы и объекты в программе?
23) Что такое поля и методы класса?
24) Какие существую спецификаторы доступа в классе?
25) Что такое конструктор и деструктор? Для чего они применяются?
26) Чем отличаются константные и статические методы от обычных методов?
27) Что такое перегрузка операций?
28) Зачем реализуются конструктор копии и оператор присваивания?
29) Что такое наследование? В каких случаях используется этот механизм?
30) Как описывается наследование классов в программе?
31) Как влияют спецификаторы доступа при наследовании?
32) Какие существуют правила использования базовых и производных классов?
33) В чем заключаются особенности работы с конструкторами базового и производного классов?
34) Какая последовательность вызова конструкторов и деструкторов базового и производного классов?
35) Каким образом обеспечивается доступ к закрытым, защищенным и открытым компонентам класса в зависимости от типа наследования?
36) Что такое статическое и динамическое связывание?
37) Что такое виртуальная функция? В чем ее отличие от обычной функции?
38) Для чего используются виртуальные функции?
39) Какие функции не могут быть виртуальными?
40) Какие существуют правила использования виртуальных функций?
41) В чем назначение виртуального деструктора?
42) Может ли конструктор быть виртуальным?
43) Что такое абстрактный класс? В чем его отличие от обычного класса?
44) Почему нельзя создать экземпляр абстрактного класса?
45) Каким образом абстрактные классы используются в программах?
46) Что такое множественное наследование? Как оно описывается в программе?
47) Как объявляется шаблонная функция? Шаблонный класс?
48) Что такое инстанцирование шаблона?
49) Что может являться параметрами шаблона?
50) Какие предъявляются требования к элементам контейнеров?
51) Какие особенности имеет контейнер vector?
52) Какие особенности имеет контейнер deque?
53) Какие особенности имеет контейнер list?
54) Какие существуют итераторы?
55) Какие алгоритмы стандартной библиотеки вы знаете?
56) Какие особенности имеет контейнер set?
57) Какие особенности имеет контейнер multiset?
58) Какие особенности имеет контейнер map?
59) Какие особенности имеет контейнер multimap?

Типовые задания для практической и самостоятельной работы

1) Разработать класс, реализующий операции с комплексными числами.
2) Разработать класс, реализующий операции с дробями.
3) Разработать класс, реализующий операции с векторами.
4) Разработать класс, реализующий операции с матрицами.
5) Разработать класс, реализующий операции с полиномами.
6) Разработать класс для хранения и обработки информации о студенте.

7) Разработать класс для хранения и обработки информации о транспортном средстве.

8) Разработать проект программы для получения, хранения и обработки результатов эксперимента.

9) Разработать класс-телефонный справочник.

10) Разработать класс для хранения и обработки информации о книге.

11) Разработать иерархию классов Учащийся: Школьник, Студент.

12) Разработать иерархию классов Домашнее животное: Собака, Кошка, Птица.

13) Разработать иерархию классов Тригонометрическая функция: Синус, Косинус, Тангенс, Котангенс.

14) Разработать иерархию классов Фигура: Круг, Прямоугольник, Треугольник.

15) Разработать иерархию классов Работник фирмы: Экономист, Администратор, Программист.

16) Разработать проект программы для обработки результатов научных исследований в заданной области.

17) Разработать программу для автоматизации расчетов по лабораторной или курсовой работе другой дисциплины.

18) Разработать программу, реализующую базу данных книг, марок или монет.

19) Разработать программу для учета и планирования расходов.

20) Разработать программу для работы с расписанием занятий.

Вопросы к теоретическому зачету по дисциплине

1) Основные понятия и сферы применения объектно-ориентированного программирования. Преимущества объектно-ориентированного подхода по отношению к процедурному подходу к разработке программ.

2) Предпосылки и история создания языка C++. Достоинства и недостатки языка C++. Сферы применения языка C++.

3) Компиляторы языка C++. Средства разработки программ на языке C++.

4) Классификация типов данных в языке C++. Целые типы данных. Символьные типы данных. Типы для представления чисел с плавающей точкой. Логический тип данных. Тип void.

5) Оператор sizeof(). Гарантии стандарта по соотношению размеров встроенных типов. Получение информации о встроенных типах данных для конкретной платформы.

6) Классификация типов данных в языке C++. Литералы встроенных типов.

7) Структура и особенности объявления имен в языке C++. Имена (идентификаторы). Область видимости. Вложенные области видимости.

8) Пространства имен. Создание псевдонимов для типов данных, ключевое слово typedef.

9) Инициализация сущностей в языке C++. Имена (идентификаторы). Область видимости.

10) Правила инициализации переменных в разных областях видимости при отсутствии инициализатора.

11) Стандартные операции в языке C++. Арифметические операции. Преобразование типов в арифметических выражениях. Приоритет операций.

12) Логические операции. Побитовые логические операции. Операции присваивания. Операция запятая. Приоритет операций.

13) Управляющие конструкции в языке C++. Оператор ветвления if. Тернарная операция. Оператор выбора вариантов switch.

14) Операторы циклов while и do-while. Оператор цикла for. Операторы прерывания циклов break и continue.

15) Определение и основы работы с указателями и массивами в языке C++. Многомерные массивы. Связь между указателями и массивами, индексация с помощью указателей.

16) Константы. Константные указатели и указатели на константу. Ссылки. Константные ссылки.

17) Перечисления. Структуры. Битовые поля. Объединения. Опережающие объявления. Эквивалентность типов.

18) Объявление и определение функции в языке C++. Передача параметров в функцию по значению. Возврат результатов работы функции.

19) Передача параметров по ссылке и константной ссылке. Передача параметров в функцию через указатель. Возврат результатов работы функции.

20) Передача массивов в качестве параметров функции. Многомерные массивы в качестве аргументов функции.

21) Использование внутри функции переменной из охватывающей области видимости. Статические переменные в функциях. Встраиваемые (inline) функции. Перегрузка функций.

22) Перегрузка функций. Аргументы по умолчанию. Функции с переменным числом аргументов. Указатели на функции.

23) Общие сведения о приведении типов. Приведение типов в стиле языка C. Приведение типов в функциональном стиле. Операторы приведения типов в языке C++.

24) Структура памяти программы. Модель памяти с точки зрения программиста на языке C++. Создание динамических объектов (размещение объектов в куче).

25) Динамические массивы. Многомерные динамические массивы.

26) Организация многофайловых программ в языке C++. Спецификатор extern. Директива #include. Защита от множественного включения заголовочных файлов.

27) Понятие о препроцессоре, директивы препроцессора. Директива #define, макросы. Макросы с параметрами. Директивы условной компиляции.

28) Понятие класса и экземпляра класса (объекта). Поля класса.

29) Объявление класса. Спецификаторы доступа к полям и методам.

30) Определение методов класса внутри и снаружи объявления класса.

31) Отличие методов класса от обычных функций, указатель this. Константные методы.

32) Конструкторы и деструкторы. Конструктор по умолчанию. Список инициализации конструктора.

33) Конструктор копии и оператор присваивания. Защита объектов от копирования.

Вопросы к экзамену по дисциплине

1) Понятие класса и экземпляра класса (объекта). Объявление класса. Спецификаторы доступа к полям и методам. Поля класса.

2) Определение методов класса внутри и снаружи объявления класса. Отличие методов класса от обычных функций, указатель this. Константные методы.

3) Конструкторы. Конструктор по умолчанию. Список инициализации конструктора. Конструктор копии и оператор присваивания. Защита объектов от копирования.

4) Статические поля. Статические константы в классе. Статические методы.

5) Владение ресурсами, конструкторы и деструктор, идиома RAII.

6) Перегрузка операций. Перегрузка операций operator[] и operator(). Операторы приведения типов.

7) Наследование классов. Терминология, применяемая для описания иерархий наследования. Доступ к полям и методам базового класса при наследовании. Спецификаторы доступа при наследовании.

8) Переопределение методов базового класса. Реализация конструкторов и деструкторов при наследовании. Статические элементы при наследовании. Наследование и вложенные классы.

9) Проблема динамической идентификации типов. Виртуальные функции. Статическое и динамическое связывание. Виртуальный деструктор. Чистые виртуальные функции и абстрактные классы.

10) Множественное наследование. Виртуальные базовые классы.

11) Механизм RTTI. Применение операции dynamic_cast. Операция typeid.

12) Шаблоны функций и классов. Инстанцирование и работа с шаблонными объектами. Параметры шаблонов. Эквивалентность типов.

13) Шаблоны функций. Инстанцирование шаблонных функций. Аргументы шаблонных функций.

14) Параметры шаблонов по умолчанию. Полная и частичная специализации шаблонов. Наследование шаблонов.

15) Обработка исключений при помощи кодов ошибок и недостатки данного подхода.

16) Механизм исключений. Генерация исключений. Перехват исключений, блок try-catch. Повторная генерация исключений.

17) Механизм исключений. Передача исключений из вложенного блока. Необработанные исключения. Спецификация исключений.

18) Организация стандартной библиотеки языка C++. Взаимодействие контейнеров, алгоритмов и итераторов.

19) Стандартная библиотека языка C++. Классификация контейнеров. Отличия последовательных и ассоциативных контейнеров.

20) Стандартная библиотека языка C++. Функциональные объекты. Сравнение функциональных объектов и обычных функций.

21) Стандартная библиотека языка C++. Контейнеры стандартной библиотеки. Требования к элементам контейнера. Общие операции контейнеров. Сравнение контейнеров.

22) Динамический массив (vector). Устройство контейнера. Сложность операций. Размер и емкость. Сравнение с массивами в стиле языка C.

23) Двунаправленная очередь. Устройство контейнера. Сложность операций. Управление памятью и целостность итераторов.

24) Список. Устройство контейнера. Сравнение с контейнером vector. Специфические операции со списками.

25) Множество и мультимножество. Внутреннее устройство. Требования к критерию сортировки. Вычислительная сложность операций с множествами, функции поиска.

26) Отображение и мультиотображение. Внутреннее устройство. Сравнение с множествами. Сложность операций. Отображения в качестве ассоциативных массивов.

27) Адаптеры последовательных контейнеров. Стек. Основной интерфейс стека.

28) Адаптеры последовательных контейнеров. Очередь. Основной интерфейс очереди.

29) Адаптеры последовательных контейнеров. Очередь с приоритетом. Интерфейс очереди с приоритетом.

30) Битовые поля. Операции с битовыми полями. Методы класса bitset.

31) Итераторы стандартной библиотеки. Категории итераторов. Итераторы ввода. Итераторы вывода.

32) Итераторы стандартной библиотеки. Категории итераторов. Прямые, двунаправленные итераторы. Итераторы произвольного доступа.

33) Итераторы стандартной библиотеки. Категории итераторов. Функции advance(), distance(), iter_swap().

34) Итераторы стандартной библиотеки. Обратные итераторы.

35) Итераторы вставки. Разновидности итераторов вставки.

36) Потоковые итераторы. Итераторы входного и выходного потоков.

37) Строки STL (класс string). Обращение к элементам строки. Размер и емкость строки.

38) Строки STL (класс string). Операции присваивания, вставки, замены и удаления.

39) Строки STL (класс string). Подстроки и конкатенация. Поиск символов и строк. Строки в качестве контейнеров. Строки и итераторы.

40) Ввод/вывод с помощью потоковых объектов. Иерархия потоковых классов. Назначение потоковых буферов. Глобальные потоковые объекты. Общие сведения о манипуляторах.

41) Ввод/вывод с помощью потоковых объектов. Флаги состояния потоков. Функции для работы с состоянием потоков. Потоковые классы и условные операторы.

42) Ввод/вывод с помощью потоковых объектов. Перегрузка потоковых операторов. Функции ввода и вывода.

43) Ввод/вывод с помощью потоковых объектов. Флаги форматирования. Функции для работы с форматными флагами.

44) Файловый ввод/вывод с помощью потоковых объектов. Режимы открытия фалов. Чтение и запись в произвольной позиции курсора.

Вопросы для контрольного тестирования

Часть 1

Блок 1. Принципы объектно-ориентированного программирования (ОПК-6.1)

1 Вопрос. Инкапсуляция – свойство системы, позволяющее …
Дать определение

2 Вопрос. Наследование – свойство системы, позволяющее …
Дать определение

3 Вопрос. Полиморфизм – свойство системы, позволяющее …
Дать определение

Блок 2. Базовые понятия (ОПК-6.1)

4 Вопрос. Объектно-ориентированное программирование – методология программирования, …
Дать определение

5 Вопрос. Класс – …
Дать определение

6 Вопрос. Объект (экземпляр) – …
Дать определение

Блок 3. Объектно-ориентированное программирование, классы и объекты (ОПК-6.1)

7 Вопрос. Принципы объектно-ориентированного программирования включают в себя:
- виртуализацию, полиморфизм, наследование
- композиционность, инкапсуляцию, наследование
- инкапсуляцию, наследование, полиморфизм
- наследование, виртуализацию, абстрагирование
- инкапсуляцию, обобщенность, полиморфизм

8 Вопрос. Состояние объекта определяется …
- методами класса
- значениями всех полей
- интерфейсом пользователя
- объемом оперативной памяти

9 Вопрос. Поведение объекта определяется …
- полиморфизмом
- значениями всех полей
- методами класса
- связностью всех частей программы

10 Вопрос. Класс, как правило, состоит из …
- полей и методов
- множества объектов, имеющих общие свойства и методы
- наследования и полиморфизма
- контейнеров и шаблонов

Блок 4. Многофайловые программы (ОПК-6.1)

11 Вопрос. Файл с расширением h или hpp называют …
- объектным файлом
- файлом с исполняемым кодом
- заголовочным файлом

- исполняемым файлом
12 Вопрос. Заголовочные файлы подключаются с помощью директивы …
- #define
- #include
- #pragma
- #using
13 Вопрос. Файл с расширением с или cpp называют …
- объектным файлом
- файлом с исполняемым кодом
- заголовочным файлом
- исполняемым файлом
14 Вопрос. Страж включения заголовочного файла используется для …
- сокрытия деталей реализации классов и функций
- множественных включений одного файла
- остановки процесса компиляции
- избежания множественных включений одного файла

Блок 5. Типы данных (ОПК-6.1)
15 Вопрос. Типы bool, int, char являются …
- фундаментальными типами
- перечислениями
- структурами
- типами, определяемыми пользователем
16 Вопрос. Типы float, double являются …
- интегральными типами
- типами с плавающей точкой
- классами
- типами, определяемыми пользователем
17 Вопрос. Типы bool, int, char являются …
- интегральными типами
- типами с плавающей точкой
- объединениями
- типами, определяемыми пользователем
18 Вопрос. Классы и структуры являются …
- арифметическими типами
- интегральными типами
- символьными типами
- типами, определяемыми пользователем
19 Вопрос. Типы double, int, char являются …
- типами с плавающей точкой
- символьными типами
- арифметическими типами
- интегральными типами
20 Вопрос. Перечисления и объединения являются …
- арифметическими типами
- интегральными типами
- символьными типами
- типами, определяемыми пользователем

Блок 6. Создание переменных (ПК-1.1)
21 Вопрос. Создайте переменную с произвольным именем, являющуюся указателем на константные данные типа с плавающей точкой.
22 Вопрос. Создайте переменную целого типа с произвольным именем.
Создайте переменную, являющуюся константной ссылкой на первую переменную.
23 Вопрос. Создайте переменную с произвольным именем, являющуюся статическим массивом символов (размер произвольный).
24 Вопрос. Создайте переменную логического типа с произвольным именем.
Создайте вторую переменную, хранящую адрес первой переменной.

25 Вопрос. Создайте переменную с произвольным именем и интегральным типом. Создайте вторую переменную, являющуюся ссылкой на первую переменную.
26 Вопрос. Создайте переменную с произвольным именем, являющуюся статическим двумерным массивом целых чисел (размеры произвольные).
27 Вопрос. Создайте переменную с произвольным именем, являющуюся константным указателем на константные данные целого типа.
28 Вопрос. Создайте переменную логического типа с произвольным именем. Создайте вторую переменную, являющуюся ссылкой на первую переменную.

Блок 7. Размеры типов (ПК-1.1)

29 Вопрос. Оператор, позволяющий узнать размер типа на используемой платформе:
- typedef
- sizeof
- тернарный оператор
- бинарный оператор

30 Вопрос. Согласно стандарту выполняется следующее соотношение:
- размер(char) = размер(short) = размер(int) = размер(long)
- размер(char) ≤ размер(short) ≤ размер(int) ≤ размер(long)
- размер(char) ≥ размер(short) ≥ размер(int) ≥ размер(long)

31 Вопрос. Согласно стандарту выполняется следующее соотношение:
- размер(float) = размер(double) = размер(long double)
- размер(float) ≤ размер(double) ≤ размер(long double)
- размер(float) ≥ размер(double) ≥ размер(long double)

32 Вопрос. Все размеры в C++ кратны размеру типа …
- int
- char
- float
- short

33 Вопрос. Согласно стандарту выполняется следующее соотношение:
- размер(char) = размер(wchar_t) = размер(long)
- размер(char) ≤ размер(wchar_t) ≤ размер(long)
- размер(char) ≥ размер(wchar_t) ≥ размер(long)

34 Вопрос. Согласно стандарту выполняется следующее соотношение:
- размер(T) = размер(signed T) = размер(unsigned T)
- размер(T) ≤ размер(signed T) ≤ размер(unsigned T)
- размер(T) ≥ размер(signed T) ≥ размер(unsigned T)

Блок 8. Объявление функций (ПК-1.1)

35 Вопрос. Объявите функцию без параметров, возвращающую указатель на данные целого типа. Имя функции произвольное.
36 Вопрос. Объявите функцию с одним аргументом типа string, передаваемым по ссылке. Возвращаемый тип – логический. Имя функции произвольное.
37 Вопрос. Объявите не возвращающую ничего функцию с двумя аргументами целого типа, передаваемыми по значению. Имя функции произвольное.
38 Вопрос. Объявите функцию без параметров, возвращающую константную ссылку на значение типа с плавающей точкой. Имя функции произвольное.
39 Вопрос. Объявите функцию с одним аргументом типа double, передаваемым через указатель. Возвращаемый тип – целое число. Имя функции произвольное.
40 Вопрос. Объявите не возвращающую ничего функцию с двумя аргументами целого типа, передаваемыми по константной ссылке. Имя функции произвольное.

Блок 9. Циклы и условный оператор (ПК-1.1)

41 Вопрос. Какой результат будет выведен в консоль при выполнении функции main?

```
void main() {
  int i=10;
  for (i=i+1; i<12; ++i)
    cout << i;
}
```

- ничего
- 10
- 11
- 1011

42 Вопрос. Какой результат будет выведен в консоль при выполнении функции main?

```
void main() {
  int i = 10;
  if (i = 11)
    cout << i;
}
```

- ничего
- 10
- 11
- 0

43 Вопрос. Какой результат будет выведен в консоль при выполнении функции main?

```
void main() {
  int i=10;
  for (i=i+1; i<11; ++i)
    cout << i;
}
```

- ничего
- 10
- 11
- 1011

44 Вопрос. Какой результат будет выведен в консоль при выполнении функции main?

```
void main() {
  int i=10;
  if (i += 1)
    cout << i;
}
```

- ничего
- 10
- 11
- 0

45 Вопрос. Какой результат будет выведен в консоль при выполнении функции main?

```
void main() {
  int i=11;
  for (i=i–1; i<12; ++i)
    cout << i;
}
```

- ничего
- 10
- 11
- 1011

46 Вопрос. Какой результат будет выведен в консоль при выполнении функции main?

```
void main() {
  int i=10;
  if (i –= 10)
    cout << i;
}
```

- ничего
- 10
- 20
- 0

Блок 10. Передача аргументов в функцию (ПК-1.1)

47 Вопрос. Какой результат будет выведен в консоль при выполнении функции main?

```
void calc(int x)
```

```
{ x = x+5; }

void main() {
  int x=5;
  calc(x);
  cout << x;
}
```
- точно определить невозможно
- 0
- 5
- 10

48 Вопрос. Какой результат будет выведен в консоль при выполнении функции main?

```
void calc(int* p)
{ *p –= 5; }

void main() {
  int x=10;
  calc(&x);
  cout << x–5;
}
```
- точно определить невозможно
- 0
- 5
- 10

49 Вопрос. Какой результат будет выведен в консоль при выполнении функции main?

```
int calc(int& x)
{ return x+5; }

void main() {
  int x=0;
  calc(x);
  cout << calc(x);
}
```
- точно определить невозможно
- 0
- 5
- 10

50 Вопрос. Какой результат будет выведен в консоль при выполнении функции main?

```
int calc(int x)
{ return x–5; }

void main() {
  int x=10;
  x = calc(5);
  cout << x+5;
}
```
- точно определить невозможно
- 0
- 5
- 10

51 Вопрос. Какой результат будет выведен в консоль при выполнении функции main?

```
void calc(int* p)
{ p = 0; }

void main() {
  int x=10;
  calc(&x);
  cout << x;
```

```
}
```
- точно определить невозможно
- 0
- 5
- 10

52 Вопрос. Какой результат будет выведен в консоль при выполнении функции main?

```
void calc(int& y)
{ y += 5; }

void main() {
  int x=5;
  calc(x);
  cout << x;
}
```
- точно определить невозможно
- 0
- 5
- 10

Блок 11. Арифметические операции и присваивание (ПК-1.1)

53 Вопрос. Чему будет равна переменная X?

```
int X(6);
X –= X + 3 * 2;
```
- -6
- 18
- 12
- 6

54 Вопрос. Чему будет равна переменная X?

```
double Y(5.2);
int X = (Y > 5) ? 6 : 5;
```
- 5
- 5.2
- 6
- 6.2

55 Вопрос. Чему будет равна переменная X?

```
int Y(5);
int X = Y – 3 * 2;
```
- 4
- 0
- -1
- 1

56 Вопрос. Чему будет равна переменная X?

```
bool Y(true);
int X = (!Y) ? 3+Y : 5+Y;
```
- 3
- 4
- 5
- 6

57 Вопрос. Чему будет равна переменная X?

```
double X(5.5);
X += 5.5–1;
```
- 5.5
- 11
- 10
- 4.5

Блок 12. Инкремент и декремент (ПК-1.1)

58 Вопрос. Чему будет равна переменная X?

double Y(5.2);
int X = Y++;
- 5
- 5.2
- 6
- 6.2

59 Вопрос. Чему будет равна переменная X?
double Y(6.2);
double X = --Y;
- 5
- 5.2
- 6
- 6.2

60 Вопрос. Чему будет равна переменная X?
double Y(5.2);
int X = ++Y;
- 5
- 5.2
- 6
- 6.2

61 Вопрос. Чему будет равна переменная X?
double Y(6.2);
int X = Y--;
- 5
- 5.2
- 6
- 6.2

62 Вопрос. Чему будет равна переменная X?
double Y(5.2);
double X = ++Y;
- 5
- 5.2
- 6
- 6.2

Блок 13. Выделение и освобождение памяти (ПК-1.1)

63 Вопрос. В приведенном фрагменте кода происходит …
string *pt = new string;
delete pt;
- выделение и освобождение памяти под объект типа string
- выделение и некорректное освобождение памяти под объект типа string
- выделение и освобождение памяти под массив объектов типа string
- выделение и некорректное освобождение памяти под массив объектов типа string

64 Вопрос. В приведенном фрагменте кода происходит …
double *Ptr = new double;
delete[] Ptr;
- выделение и освобождение памяти под значение типа double
- выделение и некорректное освобождение памяти под значение типа double
- выделение и освобождение памяти под массив значений типа double
- выделение и некорректное освобождение памяти под массив значений типа double

65 Вопрос. В приведенном фрагменте кода происходит …
string *pt = new string[7];
delete[] pt;
- выделение и освобождение памяти под объект типа string
- выделение и некорректное освобождение памяти под объект типа string
- выделение и освобождение памяти под массив объектов типа string
- выделение и некорректное освобождение памяти под массив объектов типа string

66 Вопрос. В приведенном фрагменте кода происходит …

```
double *Ptr = new double[10];
delete Ptr;
```
- выделение и освобождение памяти под значение типа double
- выделение и некорректное освобождение памяти под значение типа double
- выделение и освобождение памяти под массив значений типа double
- выделение и некорректное освобождение памяти под массив значений типа double

Часть 2

Блок 1. Конструкторы и деструкторы (ОПК-6.1)

67 Вопрос. Метод класса, который вызывается автоматически при создании объекта, называется …
- дружественным
- конструктором
- константным
- деструктором

68 Вопрос. Метод класса, который вызывается автоматически при уничтожении объекта, называется …
- дружественным
- конструктором
- константным
- деструктором

69 Вопрос. Конструктор класса может быть …
- константным методом
- виртуальным методом
- и константным, и виртуальным
- никаким из перечисленных

70 Вопрос. Деструктор класса может быть …
- константным методом
- виртуальным методом
- и константным, и виртуальным
- никаким из перечисленных

Блок 2. Аргументы и возвращаемый тип конструкторов и деструкторов (ОПК-6.1)

71 Вопрос. Сколько аргументов может принимать деструктор класса?
- >= 0
- всегда 1
- > 0
- всегда 0

72 Вопрос. Что является возвращаемым типом конструктора класса?
- тип первого поля класса
- тип, указанный при объявлении конструктора
- сам класс
- конструктор ничего не возвращает

73 Вопрос. Сколько аргументов может принимать конструктор класса?
- >= 0
- всегда 1
- всегда 0

74 Вопрос. Что является возвращаемым типом деструктора класса?
- тип первого поля класса
- тип, указанный при объявлении деструктора
- сам класс
- деструктор ничего не возвращает

Блок 3. Операторы доступа к содержимому класса (ОПК-6.1, ПК-1.1)

75 Вопрос. Какой оператор используется для доступа к открытому содержимому класса при работе с объектом класса?
- .
- ->

- *
- никакой из перечисленных

76 Вопрос. Какой оператор используется для доступа к закрытому содержимому класса при работе с указателем на объект класса?
- .
- ->
- *
- никакой из перечисленных

77 Вопрос. Какой оператор используется для доступа к закрытому содержимому класса при работе с объектом класса?
- .
- ->
- *
- никакой из перечисленных

78 Вопрос. Какой оператор используется для доступа к открытому содержимому класса при работе с указателем на объект класса?
- .
- ->
- *
- никакой из перечисленных

Блок 4. Спецификаторы доступа (ОПК-6.1, ПК-1.1)

79 Вопрос. Защищенные поля и методы класса находятся после спецификатора …
- public
- friend
- private
- protected

80 Вопрос. Спецификатор private разрешает доступ к содержимому класса …
- только из методов этого класса
- из методов этого класса и методов его наследников
- из любого места программы

81 Вопрос. По умолчанию в классе используется спецификатор доступа …
- public
- protected
- private
- спецификатор доступа не определён

82 Вопрос. Спецификатор public разрешает доступ к содержимому класса …
- только из методов этого класса
- из методов этого класса и методов его наследников
- из любого места программы

83 Вопрос. Закрытые поля и методы класса находятся после спецификатора …
- public
- friend
- private
- protected

84 Вопрос. Спецификатор protected разрешает доступ к содержимому класса …
- только из методов этого класса
- из методов этого класса и методов его наследников
- из любого места программы

85 Вопрос. Открытые поля и методы класса находятся после спецификатора …
- public
- friend
- private
- protected

Блок 5. Объявление конструкторов и деструкторов (ПК-1.1)

86 Вопрос. Как будет выглядеть объявление конструктора без параметров для приведенного класса?

```
class Car {
  string name;
public:
  string getName() const;
};
```
- void Car();
- Constructor();
- Car(string s);
- Car();
- string Car();

87 Вопрос. Как будет выглядеть объявление деструктора для приведенного класса?

```
class Cat {
  string name;
public:
  string getName() const;
};
```
- void Cat();
- string ~Cat();
- Destructor();
- ~Cat();
- ~Cat(string s);

88 Вопрос. Как будет выглядеть объявление конструктора без параметров для приведенного класса?

```
class Human {
  string name;
public:
  string getName() const;
};
```
- Constructor();
- Constructor(string s);
- ~Human();
- Human();
- void Human();

89 Вопрос. Как будет выглядеть объявление деструктора для приведенного класса?

```
class House {
  string name;
public:
  string getName() const;
};
```
- Destructor();
- ~House();
- ~Destructor();
- void ~House();
- string ~House();

90 Вопрос. Как будет выглядеть объявление конструктора с параметрами для приведенного класса?

```
class Base {
  string name;
public:
  string getName() const;
};
```
- Base();
- ~Base();
- Base(string s);
- ~Base(string s);
- void Base();

91 Вопрос. Как будет выглядеть объявление конструктора без параметров для приведенного класса?

```
class Boss {
  string name;
public:
  string getName() const;
};
```
- Boss();
- ~Boss();
- Boss(string s);
- ~Boss(string s);
- void Boss();

Блок 6. Доступ к содержимому класса (ОПК-6.1)

92 Вопрос. Дружественным функциям разрешен доступ …
- ко всему содержимому класса
- ко всему содержимому класса и содержимому классов-наследников
- к открытому и защищенному содержимому класса
- к открытому содержимому класса
- все перечисленные варианты не верны

93 Вопрос. Вложенному (объявленному в другом классе) классу разрешен доступ …
- ко всему содержимому основного класса
- ко всему содержимому основного класса и содержимому его наследников
- к открытому и защищенному содержимому основного класса
- к открытому содержимому основного класса
- все перечисленные варианты не верны

94 Вопрос. Методы класса могут обращаться …
- ко всему содержимому класса
- к открытому и защищенному содержимому класса
- к открытому содержимому класса
- все перечисленные варианты не верны

95 Вопрос. Методы класса-наследника могут обращаться …
- ко всему содержимому базового класса
- к открытому и защищенному содержимому базового класса
- к открытому содержимому базового класса
- все перечисленные варианты не верны

Блок 7. Константные методы класса (ОПК-6.1)

96 Вопрос. Внутри константных методов класса можно …
- изменять значения полей
- получать значения полей
- изменять и получать значения полей
- ничего из перечисленного

97 Вопрос. Внутри константных методов класса можно …
- вызывать константные методы класса
- вызывать неконстантные методы класса
- вызывать константные и неконстантные методы класса
- ничего из перечисленного

98 Вопрос. Внутри константных методов класса можно …
- получать значения полей и вызывать неконстантные методы класса
- изменять значения полей и вызывать любые методы класса
- получать значения полей и вызывать константные методы класса
- ничего из перечисленного

Блок 8. Состояние объектов (ПК-1.1)

99 Вопрос. В функции func происходит ...

```
class Student {
  string Name;
public:
  void setName(string s)
```

```
  { Name = s; }
  string getName() const
  { return Name; }
};
void func(Student &A) {
  A.setName("Ivanov");
}
```
- изменение состояния объекта A
- ошибка при компиляции
- получение состояния объекта A
- точно определить невозможно

100 Вопрос. В функции func происходит ...
```
class Calc {
  int val;
public:
  void setVal(string s);
  int getResult();
};
void func(const Calc& A) {
  int r = A.getResult();
}
```
- изменение состояния объекта A
- ошибка при компиляции
- получение состояния объекта A
- точно определить невозможно

101 Вопрос. В функции func происходит ...
```
class Student {
  string Name;
public:
  void setName(string s)
  { Name = s; }
  string getName() const
  { return Name; }
};
void func(Student &A) {
  string s = A.getName();
}
```
- изменение состояния объекта A
- ошибка при компиляции
- получение состояния объекта A
- точно определить невозможно

102 Вопрос. В функции func происходит ...
```
class Calc {
  int val;
public:
  void setVal(string s);
  int getResult();
};
void func(Calc &A) {
  int r = A.getResult();
}
```
- изменение состояния объекта A
- получение состояния объекта A
- точно определить невозможно

103 Вопрос. В функции func происходит ...
```
class Student {
  string Name;
public:
```

```
  void setName(string s)
  { Name = s; }
  string getName() const
  { return Name; }
};
void func(const Student &A) {
  A.setName("Ivanov");
}
```
- изменение состояния объекта A
- ошибка при компиляции
- получение состояния объекта A
- точно определить невозможно

104 Вопрос. В функции func состояние объекта A ...
```
class Calc {
 int val;
public:
 void setVal(string s);
 int getResult() const;
};
void func(Calc &A) {
  int r = A.getResult();
}
```
- изменяется
- не изменяется
- точно определить невозможно

Блок 9. Конструктор копии и оператор присваивания (ПК-1.1)

105 Вопрос. В какой строке приведенного фрагмента кода вызывается конструктор копии класса Student?
```
1  Student A;
2  Student B("Ivanov");
3  Student C(B);
4  A = Student("Petrov");
```
- 1
- 2
- 3
- 4

106 Вопрос. Как будет выглядеть объявление конструктора копии для класса Comp?
- Comp(const Comp& x);
- Copy(const Comp& x);
- Comp& Comp();
- Comp Comp(const Comp& x);

107 Вопрос. Какая строка будет использоваться вместо многоточия для защиты от самоприсваивания?
```
Calc& Calc::operator=(const Calc& a) {
  …
  val = a.val;
  return *this;
}
```
- if (this == &a)
- if (this != &a)
- if (this == a)
- if (this != a)

108 Вопрос. В какой строке приведенного фрагмента кода вызывается оператор присваивания класса Student?
```
1  Student X("Ivanov");
2  string s = X.getName();
3  Student Y(X);
```

4 Y = Student("Petrov");
- 1
- 2
- 3
- 4

109 Вопрос. Какая строка будет использоваться вместо многоточия для защиты от самоприсваивания?

```
Cat& Cat::operator=(const Cat& c) {
  …
  name = c.name;
  return *this;
}
```

- if (*this != &c)
- if (*this == &c)
- if (this != &c)
- if (this == &c)

110 Вопрос. В какой строке приведенного фрагмента кода вызывается конструктор копии класса Car?

```
  void func1(Car z) {  }
  void func2(Car &z) {  }
  …
1  Car A("Car");
2  Car B;
3  func1(A);
4  B = A;
5  func2(B);
```

- 1
- 2
- 3
- 4
- 5

Блок 10. Перегрузка операторов (ПК-1.1)

111 Вопрос. Как будет выглядеть объявление перегруженного префиксного оператора инкремента для класса Value?

- void operator++(Value);
- Value& operator++();
- Value& operator++(int);
- Value& ++operator();

112 Вопрос. Как будет выглядеть объявление перегруженного оператора вычитания для класса Value?

- void operator-(const Value&);
- Value -(const Value&);
- Value -operator(const Value&);
- Value operator-(const Value&);

113 Вопрос. Как будет выглядеть объявление перегруженного постфиксного оператора инкремента для класса Value?

- void operator++(Value);
- Value operator++(int);
- Value operator++();
- Value ++operator();

114 Вопрос. Как будет выглядеть объявление перегруженного унарного оператора минус для класса Value?

- void operator-();
- Value -();
- Value operator-();
- Value -operator();

115 Вопрос. Как будет выглядеть объявление перегруженного префиксного оператора декремента для класса Value?
- Value& operator--(int);
- Value& --operator();
- void operator--(Value);
- Value& operator--();

116 Вопрос. Как будет выглядеть объявление перегруженного оператора сложения для класса Value?
- void operator+(const Value&);
- Value operator+(const Value&);
- Value +(const Value&);
- Value +operator(const Value&);

117 Вопрос. Как будет выглядеть объявление перегруженного постфиксного оператора декремента для класса Value?
- Value operator--();
- Value --operator();
- void operator--(Value);
- Value operator--(int);

Блок 11. Доступ при наследовании (ПК-1.1)

118 Вопрос. В функции main при работе с объектом X будут доступны …

```
class Transport {
public:
  double weight;
protected:
  double speed;
  int calc();
};
class Car : private Transport {
public:
  int calc();
};
void main {
  Car X;
}
```

- Car::calc()
- weight, Car::calc()
- weight, speed, Car::calc()
- weight, Transport::calc(), Car::calc()
- weight, speed, Transport::calc(), Car::calc()

119 Вопрос. В методе calc класса Car будут доступны …

```
class Transport {
private:
  int num;
public:
  double weight;
protected:
  double speed;
  int calc();
};
class Car : public Transport {
private:
  int calc();
};
```

- num, weight, speed, Transport::calc()
- num, weight, speed
- weight, speed, Transport::calc()
- weight, speed

- weight

120 Вопрос. В функции main при работе с объектом X будут доступны …

```
class Transport {
public:
  double weight;
protected:
  double speed;
  int calc();
};
class Car : public Transport {
public:
  int calc();
};
void main {
  Car X;
}
```

- weight, speed, Transport::calc(), Car::calc()
- weight, Transport::calc(), Car::calc()
- weight, Car::calc()
- weight, speed, Car::calc()
- Car::calc()

121 Вопрос. В методе calc класса Car будут доступны …

```
class Transport {
public:
  int num;
private:
  double weight;
protected:
  double speed;
  int calc();
};
class Car : private Transport {
public:
  int calc();
};
```

- num, speed, Transport::calc()
- num, weight, speed
- num, weight, speed, Transport::calc()
- num, speed
- num

122 Вопрос. В функции main при работе с объектом X будут доступны …

```
class Transport {
public:
  double weight;
  int calc();
protected:
  double speed;
};
class Car : public Transport {
private:
  int calc();
};
void main {
  Car X;
}
```

- Car::calc()
- weight, Car::calc()

- weight, Transport::calc()
- weight, Transport::calc(), Car::calc()
- weight, speed, Transport::calc(), Car::calc()

123 Вопрос. В методе calc класса Car будут доступны …

```
class Transport {
public:
  int num;
protected:
  double weight;
private:
  int calc();
  double speed;
};
class Car : protected Transport {
private:
  int calc();
};
```

- num
- weight, num
- weight, num, speed, Transport::calc()
- weight, num, speed
- weight, num, Transport::calc()

Блок 12. Переопределение методов при наследовании (ПК-1.1)

124 Вопрос. Что произойдет в функции main?

```
class Parent {
public:
  void out()
  { std::cout << "Parent"; }
};
class Child : public Parent {
public:
  void out()
  { std::cout << "Child"; };
};
int main() {
  Child A;
  Parent B = A;
  B.Parent::out();
  return 0;
}
```

- Вывод Parent в консоль
- Вывод Child в консоль
- Ошибка при компиляции
- Ничего из перечисленного

125 Вопрос. Что произойдет в функции main?

```
class Parent {
public:
  void out()
  { std::cout << "Parent"; }
};
class Child : public Parent {
public:
  void out()
  { std::cout << "Child"; };
};
int main() {
  Child A;
```

```
  Parent &B = A;
  B.out();
  return 0;
}
```
- Вывод Parent в консоль
- Вывод Child в консоль
- Ошибка при компиляции
- Ничего из перечисленного

126 Вопрос. Что произойдет в функции main?
```
class Parent {
public:
  void out()
  { std::cout << "Parent"; }
};
class Child : public Parent {
public:
  void out()
  { std::cout << "Child"; };
};
int main() {
  Child A;
  Parent B = A;
  B.Child::out();
  return 0;
}
```
- Вывод Parent в консоль
- Вывод Child в консоль
- Ошибка при компиляции
- Ничего из перечисленного

127 Вопрос. Что произойдет в функции main?
```
class Parent {
public:
  virtual void out()
  { std::cout << "Parent"; }
};
class Child : public Parent {
public:
  void out()
  { std::cout << "Child"; };
};
int main() {
  Child A;
  Child *B = &A;
  B->out();
  return 0;
}
```
- Вывод Parent в консоль
- Вывод Child в консоль
- Ошибка при компиляции
- Ничего из перечисленного

128 Вопрос. Что произойдет в функции main?
```
class Parent {
public:
  virtual void out()
  { std::cout << "Parent"; }
};
class Child : public Parent {
public:
```

```
  void out()
  { std::cout << "Child"; };
};
int main() {
  Child A;
  Parent B = A;
  B.out();
  return 0;
}
```

- Вывод Parent в консоль
- Вывод Child в консоль
- Ошибка при компиляции
- Ничего из перечисленного

129 Вопрос. Что произойдет в функции main?

```
class Parent {
public:
  virtual void out()
  { std::cout << "Parent"; }
};
class Child : public Parent {
public:
  void out()
  { std::cout << "Child"; };
};
int main() {
  Child A;
  Parent &B = A;
  B.out();
  return 0;
}
```

- Вывод Parent в консоль
- Вывод Child в консоль
- Ошибка при компиляции
- Ничего из перечисленного

130 Вопрос. Что произойдет в функции main?

```
class Parent {
public:
  void out()
  { std::cout << "Parent"; }
};
class Child : public Parent {
public:
  void out()
  { std::cout << "Child"; };
};
int main() {
  Parent A;
  Child &B = A;
  B.out();
  return 0;
}
```

- Вывод Parent в консоль
- Вывод Child в консоль
- Ошибка при компиляции
- Ничего из перечисленного

131 Вопрос. Что произойдет в функции main?

```
class Parent {
public:
```

```
  virtual void out()
  { std::cout << "Parent"; }
};
class Child : public Parent {
public:
  void out()
  { std::cout << "Child"; };
};
int main() {
  Child A;
  Parent *B = &A;
  B->Parent::out();
  return 0;
}
```

- Вывод Parent в консоль
- Вывод Child в консоль
- Ошибка при компиляции
- Ничего из перечисленного

Блок 13. Виртуальные функции (ОПК-6.1, ПК-1.1)

132 Вопрос. Функция, которая является чистой виртуальной:
- virtual void out();
- virtual int result() = 0;
- virtual int calc(int x, int y);
- virtual void find(int x = 0);

133 Вопрос. Следующее утверждение верно:
- деструкторы могут быть виртуальными
- конструкторы могут быть виртуальными
- статические методы могут быть виртуальными
- виртуальная функция может быть не только методом класса

134 Вопрос. Какой класс будет называться интерфейсом?
- Объявленный с ключевым словом virtual
- Объявленный с ключевым словом interface
- В котором есть хотя бы одна чистая виртуальная функция
- В котором все методы являются чистыми виртуальными функциями

135 Вопрос. Следующее утверждение верно:
- деструкторы не могут быть виртуальными
- конструкторы не могут быть виртуальными
- статические методы могут быть виртуальными
- виртуальная функция может быть не только методом класса

136 Вопрос. Какой класс будет называться абстрактным?
- Объявленный с ключевым словом virtual
- Объявленный с ключевым словом abstract
- В котором есть хотя бы одна чистая виртуальная функция
- В котором все методы являются чистыми виртуальными функциями

137 Вопрос. Правильно объявленная виртуальная функция в классе Base:
- virtual Base();
- void calc() = 0;
- static virtual void calc();
- virtual ~Base();

Блок 14. Вызов конструкторов и деструкторов при наследовании (ПК-1.1)

138 Вопрос. Объект Z будет иметь следующее состояние:

```
class C1 {
public:
  int x;
  C1() { x = 1; };
  C1(int v) { x = v; };
```

```
};

class C2 : public C1 {
public:
  int y;
  C2() : C1(2), y(2) { x = 3; };
};

C2  Z;
```

- x == 1, y == 2
- x == 1, y == 0
- x == 2, y == 2
- x == 2, y == 0
- x == 3, y == 2
- x == 3, y == 0

139 Вопрос. При уничтожении объекта класса D2 в консоль будет выведено:

```
class D1 {
public:
  D1() { };
  ~D1() { std::cout << 'A'; };
};
class D2 : public D1 {
public:
  D2() { };
  ~D2() { std::cout << 'B'; };
};
```

- A
- B
- AB
- BA
- ничего

140 Вопрос. Объект Z будет иметь следующее состояние:

```
class C1 {
public:
  int x;
  C1() { x = 3; };
  C1(int v) : x(1) { x = v; };
};
class C2 : public C1 {
public:
  int y;
  C2() : y(2) { x = 2; };
};

C2  Z;
```

- x == 1, y == 2
- x == 1, y == 0
- x == 2, y == 2
- x == 2, y == 0
- x == 3, y == 2
- x == 3, y == 0

141 Вопрос. При создании объекта класса C2 в консоль будет выведено:

```
class C1 {
public:
  int x;
  C1() : x(1) { std::cout << 'B'; };
};
class C2 : public C1 {
```

```
public:
  int y;
  C2() : y(2) { std::cout << 'A'; };
};
```
- A
- B
- AB
- BA
- ничего

142 Вопрос. При уничтожении объекта класса D2 в консоль будет выведено:
```
class D1 {
public:
  D1() { };
  ~D1() { std::cout << '0'; };
};

class D2 : public D1 {
public:
  D2() { };
  ~D2() { std::cout << '1'; };
};
```
- 0
- 1
- 01
- 10
- ничего

Часть 3

Блок 1. Параметры шаблонов (ОПК-6.1, ПК-1.1)

143 Вопрос. Параметрами шаблона могут быть …
- только типы и значения
- только значения и шаблоны
- только шаблоны и типы
- типы, значения, шаблоны

144 Вопрос. Параметры-типы шаблона предваряются словом …
- type или class
- type или typename
- typename или class
- type или typename, или class

145 Вопрос. В качестве параметров шаблона нельзя передавать …
- типы
- неконстантные значения
- классы
- шаблоны

Блок 2. Шаблоны: общие вопросы (ОПК-6.1)

146 Вопрос. Могут ли параметры шаблона иметь значения по умолчанию?
- Да
- Нет
- Только для шаблона функции
- Только для шаблона класса

147 Вопрос. При вызове шаблонной функции параметры шаблона определяются …
- исходя из типа возвращаемого значения функции
- исходя из имени функции
- исходя из числа аргументов функции
- исходя из типов аргументов функции

148 Вопрос. Явное задание параметров шаблона допускается:
- только при вызове шаблонной функции

- только при создании объекта шаблонного класса
- оба варианта (1 и 2) верны
- ни в каком из перечисленных случаев

149 Вопрос. Что не может быть шаблоном?
- Класс
- Функция
- Метод класса
- Пространство имен

Блок 3.Шаблоны функций (ПК-1.1)

150 Вопрос. Правильный вызов шаблонной функции template<class X,int Y> void calc(X z):
- calc<12>(5.3);
- calc<double,12>(5.3);
- calc<double,int>(5.3);
- calc(5.3);

151 Вопрос. Правильный вызов шаблонной функции template<class T1,class T2> T1 f(T2 x):
- double y = f<double,10>();
- double y = f<double>(10);
- double y = f<>(10);
- double y = f(10);

152 Вопрос. Правильное объявление шаблонной функции:
- template<T = int> T min(T a, T b);
- T template<T> T min(T a, T b);
- T template<class T = int> min(T a, T b);
- template<class T> T min(T a, T b);

153 Вопрос. Выберите правильный вызов функции func с заданными аргументами a и b.
```
template<class V> void func(V i, V j) { }
int a(10);
float b(5.0);
```
- func(a, b);
- func<float>(a, b);
- func<float,int>(a, b);
- func<int, float>(a, b);

Блок 4. Шаблоны классов (ПК-1.1)

154 Вопрос. Выберите правильное определение метода get вне тела класса Ex.
```
template<class V> class Ex {
   void get(V &v);
};
```
- template<class V> void Ex<V>::get(V &v) { }
- template<class V> void Ex::get(V &v) { }
- template<V> void Ex<class V>::get(V &v) { }
- template<V> void Ex<V>::get(V &v) { }

155 Вопрос. Выберите правильное создание объекта класса Temp.
```
template<class T> class Temp {
   T val;
public:
   Temp(T v) : val(v) { }
};
```
- Temp a(7);
- Temp<7> a;
- Temp<int> a(7);
- Temp<int> a;

156 Вопрос. Выберите правильное определение метода calc вне тела класса vec.
```
template<class T,class X> class vec {
   T calc(X x);
};
```
- template<T,X> T vec<T,X>::calc(X x) { return x+x; }

- template<class T,class X> T vec<T,X>::calc(X x) { return x+x; }
- template<T,X> T vec<class T,class X>::calc(X x) { return x+x; }
- template<class T,class X> T vec::calc(X x) { return x+x; }

157 Вопрос. Какой вариант создания объекта класса Buf вызовет ошибку при компиляции?

```
template<class T, int i> class Buf {
   T v[i];
};
const int s1(1000);
int s2(300);
```

- Buf<float, s2> a;
- Buf<char, 127> b;
- Buf<int, s1> c;
- Buf<std::string, 1> d;

Блок 5. Исключения (ОПК-6.1)

158 Вопрос. Как сгенерировать исключение std::exception?
- try std::exception();
- catch std::exception();
- throw std::exception();
- terminate std::exception();

159 Вопрос. Что произойдет, если какое-то исключение не будет обработано?
- Аварийное завершение программы
- Ошибка во время компиляции программы
- Ошибка во время компоновки программы

160 Вопрос. В каком блоке осуществляется обработка любых исключений?
- terminate(…) { }
- throw(…) { }
- catch(…) { }
- try(…) { }

161 Вопрос. Чтобы перехватить исключение, код необходимо поместить в блок …
- terminate { }
- throw { }
- catch { }
- try { }

162 Вопрос. Перехват и обработка исключений std::exception:
- throw { } terminate(const std::exception &e) { }
- catch { } terminate(const std::exception &e) { }
- catch { } try(const std::exception &e) { }
- try { } catch(const std::exception &e) { }

Блок 6. STL: общие вопросы (ОПК-6.1, ПК-1.1)

163 Вопрос. STL – это …
- простая библиотека шаблонов (Simple Template Library)
- стандартная библиотека шаблонов (Standard Template Library)
- библиотека умных типов (Smart Type Library)
- библиотека статических типов (Static Type Library)

164 Вопрос. Что не является частью STL?
- Алгоритмы
- Драйверы
- Контейнеры
- Итераторы

165 Вопрос. Какой из заголовочных файлов не относится к контейнерам?
- <queue>
- <list>
- <numeric>
- <map>

166 Вопрос. В каком заголовочном файле объявлены основные алгоритмы STL?
- <stack>

- <algorithm>
- <functional>
- <iterator>

167 Вопрос. Функциональные объекты STL объявлены в заголовочном файле …
- <stack>
- <algorithm>
- <functional>
- <iterator>

Блок 7. Классификация контейнеров (ОПК-6.1, ПК-1.1)

168 Вопрос. Какой контейнер не является базовым последовательным контейнером?
- vector
- deque
- list
- queue

169 Вопрос. Какой контейнер не является ассоциативным?
- list
- multiset
- map
- set

170 Вопрос. Какой контейнер не является последовательным?
- vector
- map
- deque
- list

Блок 8. Особенности контейнеров (ОПК-6.1, ПК-1.1)

171 Вопрос. Структура какого контейнера представлена двоичным деревом?
- set
- list
- deque
- array

172 Вопрос. В чем разница между контейнерами set и multiset?
- копировать объекты set нельзя, копий объекта multiset может быть много
- multiset – это всегда массив контейнеров set
- set содержит уникальные ключи, multiset допускает дублирование ключей
- set автоматически сортирует содержимое, а multiset – нет

173 Вопрос. Элементы какого контейнера хранятся в непрерывном блоке памяти?
- set
- list
- map
- vector

174 Вопрос. В чем разница между контейнерами map и multimap?
- map содержит уникальные ключи, multimap допускает дублирование ключей
- multimap – это всегда массив контейнеров map
- map содержит уникальные значения, multimap допускает дублирование значений
- копировать объекты map нельзя, копий объекта multimap может быть много

Блок 9. Вычислительная сложность операций с контейнерами (ОПК-6.1, ПК-1.1)

175 Вопрос. Вставки и удаления в начале и конце выполняются быстрее для …
- vector
- deque
- set
- map

176 Вопрос. Вставки и удаления в произвольной позиции выполняются быстрее для …
- list
- multiset
- multimap

- deque
177 Вопрос. Для какого контейнера операция поиска имеет логарифмическую сложность?
- vector
- list
- deque
- set
178 Вопрос. Какой контейнер предоставляет быстрый доступ к произвольному элементу?
- vector
- list
- multiset
- map
179 Вопрос. Вставки и удаления имеют логарифмическую сложность для контейнера …
- deque
- list
- vector
- map

Блок 10. Итераторы контейнеров (ПК-1.1)
180 Вопрос. Методы контейнеров, возвращающие обратные итераторы:
- begin(), end()
- cbegin(), cend()
- rbegin(), rend()
- crbegin(), crend()
181 Вопрос. Методы контейнеров, возвращающие константные обычные итераторы:
- begin(), end()
- cbegin(), cend()
- rbegin(), rend()
- crbegin(), crend()
182 Вопрос. Методы контейнеров, возвращающие константные обратные итераторы:
- begin(), end()
- cbegin(), cend()
- rbegin(), rend()
- crbegin(), crend()
183 Вопрос. Методы контейнеров, возвращающие обычные итераторы:
- begin(), end()
- cbegin(), cend()
- rbegin(), rend()
- crbegin(), crend()

Блок 11. Алгоритмы и контейнеры (ПК-1.1)
184 Вопрос. Что будет выведено в консоль?
```
vector<char> v = { 'c', 'z', 'a' };
reverse(v.begin(), v.end());
copy(v.begin(), v.end(), ostream_iterator<char>(cout, " "));
```
- c z a
- a z c
- a c z
- z c a
185 Вопрос. Что будет выведено в консоль?
```
deque<int> d = { 4, 4, 5, 6 };
d.push_front(2);
d.push_back(1);
copy(d.begin(), d.end(), ostream_iterator<int>(cout, " "));
```
- 4 4 5 6
- 4 4 5 6 2 1
- 2 4 4 5 6 1
- 1 4 4 5 6 2
186 Вопрос. Что будет выведено в консоль?

```
multiset<int> m = { 6, 3, 6, 1 };
m.erase(m.find(1));
copy(m.begin(), m.end(), ostream_iterator<int>(cout, " "));
```
- 6 3 6
- 3 6 6
- 1 3 6
- 3 6

187 Вопрос. Что будет выведено в консоль?
```
vector<short> c = { 9, 7, 5, 3 };
sort(c.begin(), c.begin() + 2);
copy(c.begin(), c.end(), ostream_iterator<short>(cout, " "));
```
- 9 7 5 3
- 7 9 5 3
- 3 5 7 9
- 9 7 3 5

Блок 12. Особенности контейнера «множество» (ПК-1.1)

188 Вопрос. Что будет выведено в консоль?
```
set<int, less<int>> s = { 2,4,3,5,1 };
copy(s.begin(), s.end(),    ostream_iterator<int>(cout, " "));
```
- 2 4 3 5 1
- 1 2 3 4 5
- 1 5 3 4 2
- 5 4 3 2 1

189 Вопрос. Что будет выведено в консоль?
```
set<int, greater<int>> s = { 2,4,3,5,1 };
copy(s.rbegin(), s.rend(),    ostream_iterator<int>(cout, " "));
```
- 2 4 3 5 1
- 1 2 3 4 5
- 1 5 3 4 2
- 5 4 3 2 1

190 Вопрос. Что будет выведено в консоль?
```
set<int, less<int>> s = { 2,4,3,5,1 };
copy(s.rbegin(), s.rend(),     ostream_iterator<int>(cout, " "));
```
- 2 4 3 5 1
- 1 2 3 4 5
- 1 5 3 4 2
- 5 4 3 2 1

191 Вопрос. Что будет выведено в консоль?
```
set<int, greater<int>> s = { 2,4,3,5,1 };
copy(s.begin(), s.end(),     ostream_iterator<int>(cout, " "));
```
- 2 4 3 5 1
- 1 2 3 4 5
- 1 5 3 4 2
- 5 4 3 2 1

Блок 13. Функторы (ПК-1.1)

192 Вопрос. Может ли объект класса Add1 использоваться как объект-функция?
```
class Add1 {
public:
   void operator() (int& elem) const
   { ++elem; }
};
```
- Да
- Нет

193 Вопрос. В каком алгоритме может использоваться объект класса Add2 в качестве объекта-функции?
```
class Add2 {
```

```
public:
   bool operator() (int v1, int v2) const
   { return v1 < v2; }
};
vector<int> vec = { 5,3,6 };
```
- for_each(vec.begin(), vec.end(), Add2());
- sort(vec.begin(), vec.end(), Add2());
- Оба варианта (1 и 2) верны
- Ни в каком из перечисленных

194 Вопрос. Может ли объект класса Add1 использоваться как объект-функция?
```
class Add1 {
public:
   void operator+ (int& elem) const
   { ++elem; }
};
```
- Да
- Нет

195 Вопрос. В каком алгоритме может использоваться объект класса Add2 в качестве объекта-функции?
```
class Add2 {
public:
   void operator() (int &v1) const
   { v1 += (v1 > 0); }
};
vector<int> vec = { 5,3,6 };
```
- for_each(vec.begin(), vec.end(), Add2());
- sort(vec.begin(), vec.end(), Add2());
- Оба варианта (1 и 2) верны
- Ни в каком из перечисленных

Блок 14. Размер контейнеров (ПК-1.1)

196 Вопрос. Что будет выведено в консоль?
```
set<short> s = { 9, 2, 2, 6, 3, 4, 4 };
cout << s.size();
```
- 7
- 6
- 5
- 2

197 Вопрос. Что будет выведено в консоль?
```
vector<double> v = { 1.23, 2.4, 2.1, 5, 4 };
cout << (v.size() > v.capacity() ? "Size" : "Capacity");
```
- Size
- Capacity
- Ничего
- Точно определить нельзя

198 Вопрос. Что будет выведено в консоль?
```
list<char> q = {'a', 'b', 'c'};
q.push_front('f');
q.clear();
cout << q.size();
```
- 3
- 4
- 1
- 0

199 Вопрос. Что будет выведено в консоль?
```
multiset<int> m = { 5, 5, 7, 3, 1, 2, 5 };
cout << m.size();
```
- 7

- 0
- 5
- 4