Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»
(ВлГУ)**

УТВЕРЖДАЮ
Директор КИТП
_____ Н.Е. Мишулина
«20» марта 2025 г.

ФОНД ОЦЕНОЧНЫХ МАТЕРИАЛОВ УЧЕБНОЙ ДИСЦИПЛИНЫ ПРОФЕССИОНАЛЬНОЙ ПОДГОТОВКИ

«ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ»

09.02.09 Веб-разработка
Разработчик веб приложений

Владимир, 2025

Фонд оценочных материалов учебной дисциплины профессиональной подготовки «Основы алгоритмизации и программирования» разработал старший преподаватель кафедры ИСПИ Шамышева О.Н.

Фонд оценочных материалов учебной дисциплины рассмотрен и одобрен на заседании УМК специальности 09.02.09 Веб-разработка
протокол № 1 от «10» марта 2025 г.

Председатель УМК специальности _____ И.Е. Жигалов

Фонд оценочных материалов учебной дисциплины рассмотрен и одобрен на заседании кафедры ИСПИ
протокол № 7а от «12» марта 2025 г.

Заведующий кафедрой _____ И.Е. Жигалов

Фонд оценочных материалов учебной дисциплины рассмотрен и одобрен на заседании УМК КИТП
протокол № 8 от «17» марта 2025 г.

1. ПЕРЕЧЕНЬ КОМПЕТЕНЦИЙ И ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

Формируемые компетенции	Результаты обучения по дисциплине	Наименование оценочного средства
ОК 02	**Знания**: – Основы программирования на процедурном языке; – Синтаксис выбранного процедурного языка программирования, операторы ветвления, циклов, объявления функций; **Умения**: – Применять выбранный язык программирования для написания простейших программ.	Лабораторные и самостоятельные работы, итоговый тест, экзамен
ОК 09	**Знания:** – Отраслевая нормативная техническая документация; **Умения:** – Пользоваться нормативно-технической документацией в области программного обеспечения.	Лабораторные и самостоятельные работы, итоговый тест, экзамен
ПК 4.2	**Знания:** – Современные объектно-ориентированные языки программирования; – Синтаксис и стандартные библиотеки выбранного (объектно-ориентированного) языка программирования, особенности программирования на этом языке; – Среды разработки выбранного (объектно-ориентированного) языка программирования и их особенности; – Технологии программирования; – Методы повышения читаемости программного кода; **Умения:** – Применять выбранные (объектно-ориентированный) язык программирования для написания программного кода; – Использовать выбранную среду программирования для разработки с использованием выбранного (объектно-ориентированный) языка программирования; **Практический опыт:** – Создания программного кода в соответствии с техническим заданием (готовыми спецификациями) на выбранном (объектно-ориентируемом) языке программирования; – Оптимизации программного кода, написанного на выбранном (интерпретируемом) языке программирования, с использованием специализированных программных средств;	Лабораторные и самостоятельные работы, итоговый тест, экзамен

ПК 4.3	**Знания:** – Методы и приемы отладки программного кода; – Типы и форматы сообщений об ошибках, предупреждений; – Современные компиляторы, отладчики и оптимизаторы программного кода. **Умения:** – Выявлять ошибки в программном коде; – Применять методы и приемы отладки программного кода; **Практический опыт:** – Анализа и проверки исходного программного кода; – Отладки программного кода на уровне программных модулей;	Лабораторные и самостоятельные работы, итоговый тест, экзамен

2. ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ ЗНАНИЙ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

Текущий контроль знаний в рамках изучения дисциплины «Основы алгоритмизации и программирования» предполагает оценивание выполнения лабораторных и самостоятельных работ.

В рамках освоения дисциплины, обучающиеся выполняют 24 лабораторные работы и 3 самостоятельные работы. Каждая работа относится к одному из 6-ти рейтинг-контролей и оценивается согласно установленным ниже правилам.

Распределение лабораторных и самостоятельных работ

Название работы	**Максимальное количество баллов**
Рейтинг-контроль №1	
Лабораторная работа №1 «Интерфейс среды разработки»	2
Лабораторная работа №2 «Вычисления»	2
Лабораторная работа №3 «Операторы ветвления»	2
Лабораторная работа №4 «Операторы цикла»	2
Итог за рейтинг-контроль №1	**8**
Рейтинг-контроль №2	
Лабораторная работа №5 «Методы в Java»	2
Лабораторная работа №6 «Рекурсивные вычисления»	2
Лабораторная работа №7 «Одномерные массивы»	2
Лабораторная работа №8 «Двумерные массивы»	2
Итог за рейтинг-контроль №2	**8**
Рейтинг-контроль №3	
Лабораторная работа №9 «Строки»	2
Лабораторная работа №10 «Классы и объекты»	2
Лабораторная работа №11 «Наследование»	2
Лабораторная работа №12 «Исключения»	2
Итог за рейтинг-контроль №3	**8**

Рейтинг-контроль №4	
Лабораторная работа №13 «Использование классов и интерфейсов»	2
Лабораторная работа №14 «Паттерны проектирования»	2
Лабораторная работа №15 «Работа с коллекциями. ArrayList»	2
Лабораторная работа №16 «Работа с коллекциями. LinkedList»	2
Самостоятельная работа №1 «Алгоритмы динамического программирования»	4
Итог за рейтинг-контроль №4	**12**
Рейтинг-контроль №5	
Лабораторная работа №17 «Работа с коллекциями. HashMap»	2
Лабораторная работа №18 «Алгоритмы сортировки»	2
Лабораторная работа №19 «Алгоритмы поиска»	2
Лабораторная работа №20 «Линейные структуры данных»	2
Самостоятельная работа №2 «Создание собственных подклассов исключений. Использование исключений. Блоки перехвата исключения. Оператор throw. Порядок обработки исключений. Понятие потока thread и общих принципов многопоточного программирования. Класс Thread и интерфейс Runnable. Синхронизация потоков, оператор synchronized. Взаимная блокировка потоков. 10. Основы разработки многопоточных программ»	4
Итог за рейтинг-контроль №5	**12**
Рейтинг-контроль №6	
Лабораторная работа №21 «Куча. Пирамидальная сортировка»	2
Лабораторная работа №22 «Деревья»	2
Лабораторная работа №23 «Обходы графов»	2
Лабораторная работа №24 «Алгоритмы на графах»	2
Самостоятельная работа №3 «Сортировка и настройка коллекций. Внутренние классы. Введение в JDBC. JDBC SQL программирование. Дополнительные возможности JDBC. Регулярные выражения. Дополнение – Основные классы коллекции. Основы Git Basics. Ветвление Git. Git на сервере. GitLab в деталях»	4
Итог за рейтинг-контроль №6	**12**

Шкала оценивания лабораторных и самостоятельных работ

Оценка выполнения заданий	**Критерий оценки**
Шкала оценивания лабораторных работ	
2 балла	методические указания к лабораторной работе выполнены правильно и в полном объеме, обучающийся правильно ответил на контрольные вопросы
1 балл	методические указания к лабораторной работе выполнены правильно, но не в полном объеме, или допущены ошибки, или обучающийся неправильно ответил на некоторые контрольные вопросы
0 баллов	методические указания к лабораторной работе выполнены неправильно, или обучающийся неправильно ответил на все контрольные вопросы, или работа отсутствует
Шкала оценивания самостоятельных работ	
3-4 балла	методические указания к самостоятельной работе выполнены правильно и в полном объеме, обучающийся правильно ответил на контрольные вопросы

1-2 балла	методические указания к самостоятельной работе выполнены правильно, но не в полном объеме, или допущены ошибки, или обучающийся неправильно ответил на некоторые контрольные вопросы
0 баллов	методические указания к самостоятельной работе выполнены неправильно, или обучающийся неправильно ответил на все контрольные вопросы, или работа отсутствует

Оценочные средства лабораторных и самостоятельных работ студентов проверяются на занятиях в соответствии с учебным планом, методические рекомендации содержатся в документах «Методические указания к лабораторным работам по дисциплине «Основы алгоритмизации и программирования» для студентов СПО», «Методические рекомендации к самостоятельным работам студента по дисциплине «Основы алгоритмизации и программирования» для студентов СПО» и доводится до сведения обучающихся исключительно в ходе применения этих оценочных средств в процессе обучения. Общее распределение баллов текущего контроля по видам учебных работ представлено в таблице ниже.

Распределение баллов

п/п	**Наименование занятий**	**Максимальное количество баллов**
1	Рейтинг-контроль №1	8
2	Рейтинг-контроль №2	8
3	Рейтинг-контроль №3	8
4	Рейтинг-контроль №4	12
5	Рейтинг-контроль №5	12
6	Рейтинг-контроль №6	12
	Всего по дисциплине	**60**

3. ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

Промежуточная аттестация проводится в форме экзамена.

К сдаче экзамена по дисциплине «Основы алгоритмизации и программирования» допускаются обучающиеся, полностью выполнившие программу дисциплины: получившие 1 или более баллов за каждую лабораторную и самостоятельную работу.

Экзамен по дисциплине «основы алгоритмизации и программирования» оценивается одной из следующих оценок: «отлично», «хорошо», «удовлетворительно» и «неудовлетворительно».

Оценка за экзамен складывается из 2-х компонентов: текущая работа обучающегося в течение семестра (не более 60 баллов), ответы на теоретические вопросы и решение практических задач в рамках промежуточной аттестации (не более 40 баллов).

Порядок формирования оценки за текущую работу в течение семестра представлено в разделе 2 «Оценочные средства для текущего контроля знаний по учебной дисциплине».

Теоретические вопросы и примеры практических заданий, которые используются в рамках промежуточной аттестации, представлены в приложении 1.

Шкала оценивания ответов на теоретические вопросы и выполнения практических заданий в рамках промежуточной аттестации представлена в таблице ниже.

Оценка в баллах	**Критерии оценивания**
30-40	Обучающийся глубоко и прочно усвоил программный материал, исчерпывающе, последовательно, четко и логически стройно его излагает, умеет тесно увязывать теорию с практикой, свободно справляется с задачами, вопросами и другими видами применения знаний, причем не затрудняется с ответом при видоизменении заданий, использует в ответе материал монографической литературы, правильно обосновывает принятое решение, владеет разносторонними навыками и приемами выполнения практических задач, подтверждает полное освоение требований, предусмотренных программой экзамена
20-39	Обучающийся показывает твердое знания материала, грамотно и по существу излагает его, не допуская существенных неточностей в ответе на вопрос, правильно применяет теоретические положения при решении практических вопросов и задач, владеет необходимыми навыками и приемами их выполнения, допуская некоторые неточности; демонстрирует хороший уровень освоения материала, информационной и коммуникативной культуры и в целом подтверждает освоение требований, предусмотренных программой экзамена
10-19	Обучающийся показывает знания только основного материала, но не усвоил его деталей, допускает неточности, недостаточно правильные формулировки, в целом, не препятствует усвоению последующего программного материала, нарушения логической последовательности в изложении программного материала, испытывает затруднения при выполнении практических работ, подтверждает освоение требований, предусмотренных программой экзамена на минимально допустимом уровне
Менее 10	Обучающийся не знает значительной части программного материала (менее 50% правильно выполненных заданий от общего объема работы), допускает существенные ошибки, неуверенно, с большими

	затруднениями выполняет практические работы, не подтверждает освоение требований, предусмотренных программой экзамена

Общая шкала оценивания результатов освоения обучающимся дисциплины и порядок перевода итоговых баллов в оценку представлена в таблице ниже.

Оценка в баллах	**Обоснование**	**Уровень сформированности требований**
91 -100 «Отлично»	Теоретическое содержание курса освоено полностью, без пробелов необходимые практические навыки работы с освоенным материалом сформированы, все предусмотренные программой обучения учебные задания выполнены, качество их выполнения оценено числом баллов, близким к максимальному	**Высокий уровень**
74-90 «Хорошо»	Теоретическое содержание курса освоено полностью, без пробелов, некоторые практические навыки работы с освоенным материалом сформированы недостаточно, все предусмотренные программой обучения учебные задания выполнены, качество выполнения ни одного из них не оценено минимальным числом баллов, некоторые виды заданий выполнены с ошибками	**Продвинутый уровень**
61-73 «Удовлетвори-тельно»	Теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые практические навыки работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий, возможно, содержат ошибки	**Пороговый уровень**
Менее 60 «Неудовлетво-рительно»	Теоретическое содержание курса не освоено, необходимые практические навыки работы не сформированы, выполненные учебные задания содержат грубые ошибки	**Требования не сформированы**

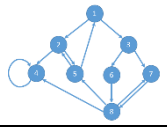
4. ИТОГОВЫЕ ТЕСТОВЫЕ ЗАДАНИЯ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

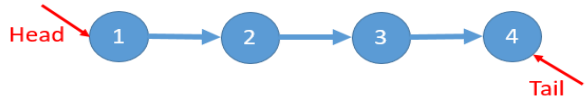
Итоговые тестовые задания применяются для контроля освоения дисциплины. Тест состоит из 50 вопросов. Каждый вопрос оценивается в 1 балл. При полном правильном ответе на вопрос обучающемуся ставится 1 балл за вопрос, иначе ставится 0 баллов.

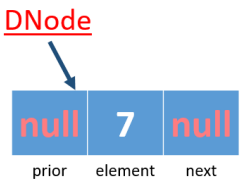
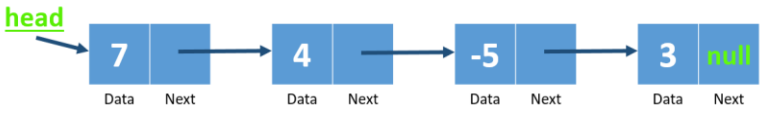
Максимальное количество баллов, которое можно набрать за тест, — 50 баллов. Уровень освоения дисциплины определяется согласно таблице ниже.

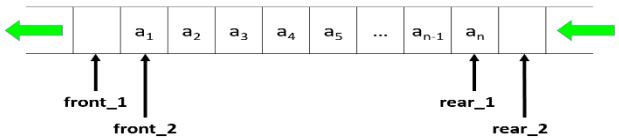
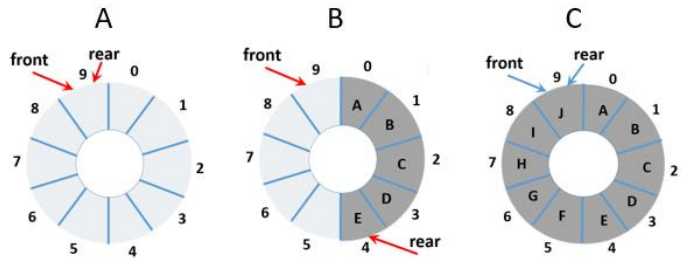
Уровень освоения дисциплины	**Количество набранных баллов**
Высокий уровень	43-50
Продвинутый уровень	34-42
Пороговый уровень	25-33
Неудовлетворительный уровень	Менее 25

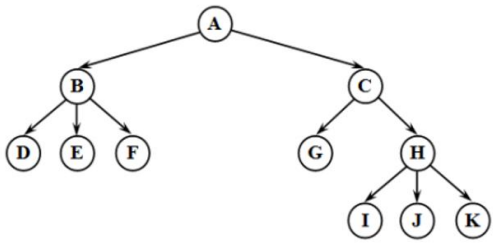
Тестовые задания разработаны по всем темам дисциплины и контролируют формирование всех компетенций. Тестовые задания представлены в таблице ниже.

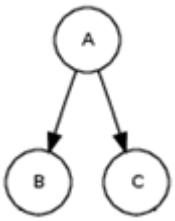
№ п/п	**Тестовые задания**
1	Определите верно ли суждение или нет? Ответ: Да/Нет Трансляторы реализуются в виде компиляторов или интерпретаторов.
2	Определите верно ли суждение или нет? Ответ: Да/Нет Структура данных — это способ организации информации для более эффективного использования. В программировании структурой обычно называют набор данных, связанных определённым образом.
3	Определите верно ли суждение или нет? Ответ: Да/Нет Качество алгоритма во многом зависит от того, является ли выбранная структура данных оптимальной для реализации конкретной задачи.
4	Определите верно ли суждение или нет? Ответ: Да/Нет Различают логическую, физическую и гибридную структуры данных.
5	Определите верно ли суждение или нет? Ответ: Да/Нет Логическая структура данных заключается в наблюдении за данными и анализом данных с точки зрения логики, что не имеет отношения с местоположением хранения данных.
6	Определите верно ли суждение или нет? Ответ: Да/Нет На рисунке представлена логическая структура - дерево. 
7	Определите верно ли суждение или нет? Ответ: Да/Нет В памяти компьютера данные могут иметь последовательное или связанное представление. Связная структура данных: мы сохраняем элемент данных в непрерывном блоке памяти, используем относительное положение между

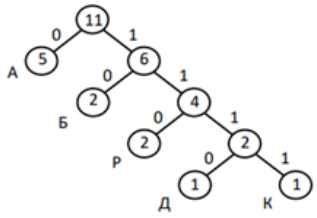
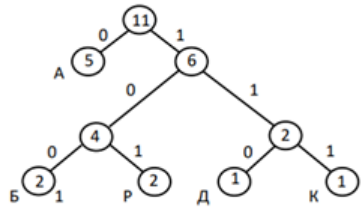
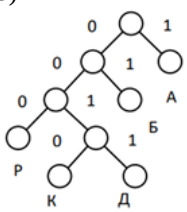
	элементами, что выражает их логическую связь.
8	Определите верно ли суждение или нет? Ответ: Да/Нет Сложность алгоритма. Верно ли: $O(1) < O(\log n) < O(n) < O(n^2) < O(n^3)$
9	Определите верно ли суждение или нет? Ответ: Да/Нет Верно ли, что сложность алгоритма метода main() будет равна O(1)? `public static void main(String[] args){` `  int n = 100, sum=0;` `  for (int i = 1; i<=n; i++)` `  sum = sum + 1 ;` `  System.out.println(sum);` `}`
10	Определите верно ли суждение или нет? Ответ: Да/Нет Верно ли, что сложность алгоритма метода main() будет равна O(log n)? `public static void main(String[] args){` `  int n = 16, count=0;` `  for (int i = 1; i<=n; i*=2) {` `  count += 1 ;` `  }` `  System.out.println(count);`
11	Заполните пустое место словами Конструкция, позволяющая описать свойства класса, не описывая их реализацию - ___?___.
12	Заполните пустое место словами Чтобы указать, что класс реализует интерфейс(ы) используется ключевое слово - ___?___.
13	Заполните пустое место словами Реализация линейной таблицы при последовательной структуре хранения `public class SeqList<T> implements__?__{` `  ...` `}`
14	Заполните пустое место словами Отношения между элементами линейного списка на рисунке R={(1,2), ___?___} 
15	Заполните пустое место словами Класс узла Node – `public class Node<T> {` `  Object element;` `  Node<T> next;` `  Node(Node<T> nextval){` `  next = ___?___;` `  }` `  Node(Object obj, Node<T> nextval){` `  element = ___?___;` `  next = nextval;` `  }` `}`
16	Заполните пустое место словами Каждый элемент списка хранит значение и минимум одну ___?___. Ссылка на первый элемент списка обычно называется ___?___.

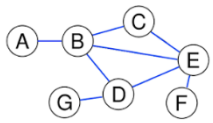
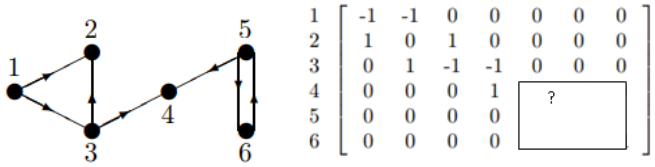
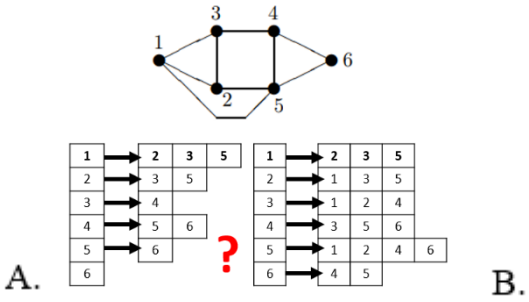
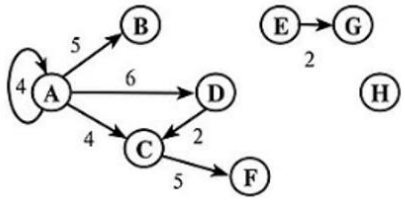
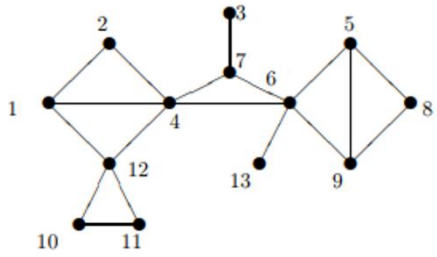
17	Заполните пустое место словами Дан код программы. Что будет выведено на экран? `public static void main(String[] args) throws Exception {` `  LinList<Integer> ll = new LinList<Integer>();` `  ll.insert(0, 0);` `  ll.insert(1, 1);` `  ll.insert(2, 2);` `  ll.insert(3, 3);` `  ll.insert(4, 4);` `  System.out.println(ll.size);  #___?___` `  System.out.println(ll.getData(2));  #___?___` `  System.out.println(ll.delete(3));  #___?___` `  System.out.println(ll.size);  #___?___` `}`
18	 Заполните пустое место словами Дан элемент двунаправленного списка. Закончите описание атрибутов класса DNode. `public class DNode<T> {` `  private T ___?___;` `  private DNode<T> ___?___;` `  private DNode<T> ___?___;` `  ...` `}`
19	Заполните пустое место словами В стеке: элемент добавляется в ___?___ стека методом ___?___. В стеке: элемент извлекается из ___?___ стека методом ___?___.
20	Заполните пустое место словами В очереди: элемент добавляется в ___?___ очереди методом ___?___. В очереди: элемент извлекается из ___?___ очереди методом ___?___.
21	выберите правильный вариант Дан код программы. Что будет выведено на экран? `public class tribonach {` `  public static int trib(int n) {` `  if (n <= 3)` `  return 1;` `  else` `  return trib(n-1) + trib(n-2) + trib(n-3);` `  }` `  public static void main(String[] args) {` `  System.out.println(trib( 5 ));` `  }` `}` A. 3 B. 5 C. 15 D. 35
22	выберите правильный вариант Список просматриваем методом `recurs_viewList(head.getNext())`. Что будет выведено на экран? 

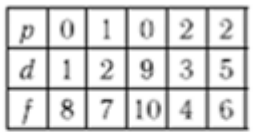
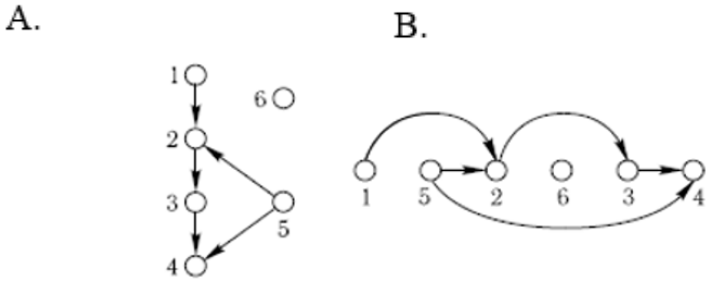
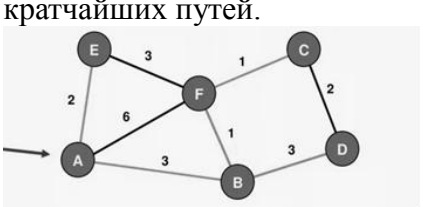
	`` public void recurs_viewList(Node<T> node) { if (node == null) System.out.println(); else { recurs_viewList(node.getNext()); System.out.print(node.getElement() + " "); //recurs_viewList(node.getNext()); } } ``` A. 7 4 -5 3 B. 3 -5 4 7 C. head 7 4 -5 3 D. 7 4 -5 3 null
23	выберите правильный вариант Стеки и очереди могут быть реализованы с помощью (): A. массивов и связанных списков B. массивов C. связных списков D. матриц
24	выберите правильный вариант Принцип входа и выхода в стек составляет (). A. FIFO B. LIFO C. вход при пустом стеке D. выход при наполненном стеке
25	выберите правильный вариант На рисунке очередь из n элементов. ()  A. Начало очереди front_1, конец очереди rear_1. B. Начало очереди front_2, конец очереди rear_2. C. Начало очереди front_2, конец очереди rear_1. D. Начало очереди front_1, конец очереди rear_2.
26	выберите правильный вариант Кольцевая очередь является: заполненной (___?___), пустой (___?___), не пустой и не заполненной (___?___) . 
27	выберите правильный вариант Определить значение front после извлечения элемента из кольцевой очереди можно

	так ( ): capacity - это максимальная длина очереди, front это указатель на голову, а rear это указатель на хвост. A. front = rear B. front = (front+1) % capacity C. front = (rear+1) % capacity D. front = (rear - front + capacity) % capacity
28	выберите правильный вариант Какая из следующих структур данных является нелинейной? A. очередь B. стек C. бинарное дерево D. линейный список
29	выберите правильный вариант Бинарное дерево описывает (___?___) между данными. A. Связные отношения B. Иерархические отношения C. Сеть-подобные отношения D. Случайные отношения
30	выберите правильный вариант  Сколько листьев содержит дерево на рисунке: A. 1 B. 3 C. 7 D. 11
31	выберите правильный вариант В полном бинарном дереве с глубиной 4: __?__ узлов , __?__ ребер. A. 4 узла 16 ребер B. 4 узла 3 ребра C. 7 узлов 6 ребер Д. 15 узлов 14 ребер
32	выберите правильный вариант С помощью метода insert() будет построено дерево: public void insert(Tree tree, int data) { if (tree.root == null) { tree.root = new Node(data); } else if (Math.random() <= 0.5) { insert (tree.root.left, data); } else { insert (tree.root.right, data); } } A. иерархическое B. случайное C. поиска (сортированное) Д. дерево Хаффмана

33	выберите правильный вариант С помощью метода insert() будет построено дерево: `public void insert(Tree tree, int data) {` `  if (tree.root == null) {` `  tree.root = new Node(data);` `  } else if (data <= tree.root.data) {` `  insert(tree.root.left, data);` `  } else {` `  insert(tree.root.right, data);` `  }` `}` A. иерархическое B. случайное C. поиска (сортированное) Д. дерево Хаффмана
34	выберите правильный вариант Правило обхода для каждой вершины: 1) просмотрим левое поддерево, 2) просмотрим текущую вершину, 3) просмотрим правое поддерево. A. preOrder B. inOrder C. postOrder D. randomOrder
35	выберите правильный вариант `public void preOrder(BinaryNode<T> p) {` `  if ( p!=null ) {` `  System.out.print( p.data.toString() + "(" );` `  preOrder(p.lchild);` `  System.out.print(",");` `  preOrder(p.rchild);` `  System.out.print(")");` `  }` `}`  A. A(B,C) B. A(B(null,null),C(null,null)) C. A(,) B(,) C(,) D. A(B(,),C(,))
36	выберите правильный вариант Код метода find() в дереве поиска(сортированном) выполняет (?). Дерево построено по правилу: в левые поддеревья добавляются меньшие элементы, чем текущая вершина. `public int find(){` `  if (this.left == null) {` `  return root.Data;` `  }` `  return this.left.find();` `}` A. ищет максимальный элемент B. ищет минимальный элемент C. ищет элемент, равный ключу D. удаляет элемент
37	выберите правильный вариант Дерево – это граф без __?__. A. ребер B. вершин C. циклов D. корня

38	выберите правильный вариант Бинарный граф 1 перестроили в граф 2, объявленный как класс TreeNode: `public class TreeNode<T> {` `  public T data;` `  public TreeNode<T> child;` `  public TreeNode<T> sibling;` `  public TreeNode(T data, TreeNode<T> child, TreeNode<T> sibling) {` `  this.data = data;` `  this.child = child;` `  this.sibling = sibling;` `  }` `  …` `  }` Каких ссылок не хватает на рисунке 2? A. BC, DE, FG B. AC, BE, CG C. AE, AC, AG D. все ссылки есть
39	выберите правильный вариант Алгоритм Хаффмана применяется для (__?__) данных. A. хранения B. передачи C. сжатия D. копирования
40	выберите правильный вариант Для слова “АБРАКАДАБРА” построили деревья для кодирования символов. Деревья равноценны, т.к. суммарная длина кодового сообщения равна 23 бита. Какое из этих деревьев построен по алгоритму Хаффмана? 1)  2)  3)  A.1 B.2 C.3 D. нет таких
41	Ответьте на вопрос Какой это граф?

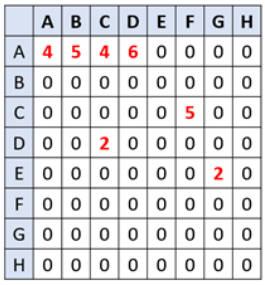
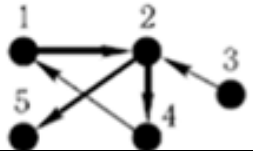
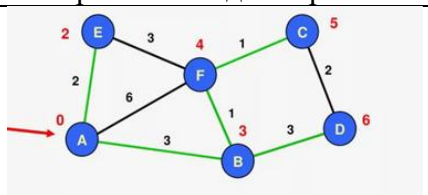
	 A. связный(+) ? не связный(-) B. ориентированный(+) ? не ориентированный(-) C. взвешенный(+) ? не взвешенный(-) A____B____C____
42	Ответьте на вопрос Матрица инциденций графа – 
43	Ответьте на вопрос Списки соседей. Когда выбираем способ хранения A, когда выбираем B? 
44	Запишите матрицу весов для графа – 
45	Ответьте на вопрос A.Укажите порядок открытия вершин при обходе графа методом DFS. B. Укажите порядок открытия вершин при обходе графа методом DFS. 
46	Ответьте на вопрос Метод обхода DFS: Добавим список предшественников (предков) – ***P.*** В $d[v]$ будем записывать «время» первого попадания в вершину v. В $f[v]$ — «время» окончания обработки всех исходящих из v рёбер. Получили результат:

	  Нарисуйте граф, который был на выходе метода DFS.
47	Ответьте на вопрос На входе программы граф A. На выходе получился граф B.  Какой алгоритм был выполнен?
48	Ответьте на вопрос Перечислите алгоритмы поиска оставного дерева.
49	Ответьте на вопрос Перечислите методы поиска минимального пути во взвешенном графе?
50	Ответьте на вопрос Решите задачу алгоритмом Дейкстры для графа. Необходимо отметить дерево кратчайших путей. 

Ключи к тесту представлены в таблице ниже

№ п/п	**Ответ**
1	Да
2	Да
3	Да
4	Нет
5	Да
6	Нет
7	Нет
8	Да
9	Нет
10	Да

11	интерфейс
12	implements
13	List<T>
14	(2,3),(3,4)
15	nextval obj
16	ссылку head
17	5 2 3 4
18	element prior next
19	начало push() начала pop()
20	конец / enqueuer() или append() или add() / начала / dequeue() или pop()
21	B. 5
22	B. 3 -5 4 7
23	A. массивов и связанных списков
24	B. LIFO
25	A. Начало очереди front_1, конец очереди rear_1.
26	C, A, B
27	B
28	C
29	B
30	C.7
31	Д. 15 узлов 14 ребер
32	B. случайное
33	C. поиска (сортированное)
34	B. inOrder
35	D. A(B(,),C(,))
36	B. ищет минимальный элемент
37	C. циклов
38	A. BC, DE, FG
39	C. сжатия
40	C.3
41	A + B – C–
42	1 0 0 -1 -1 1 0 1 -1
43	A. если важна память B. если важна скорость

44	
45	A. 1 2 4 6 5 8 9 7 3 13 12 10 11 B. 1 2 4 12 6 7 10 11 5 9 13 3 8
46	
47	Алгоритм топологической сортировки.
48	Алгоритм Крускала Алгоритм Прима
49	Алгоритм Дейкстры Алгоритм Форда-Белмана Алгоритм Флойдп-Уоршелла
50	

Приложение 1
к фонду оценочных материалов
учебной дисциплины «Основы
алгоритмизации и
программирования»

Перечень теоретических вопросов и примеры практических заданий, которые используются в рамках промежуточной аттестации

Перечень теоретических вопросов

1. Технология Java, история появления языка
2. Язык Java: типы данных.
3. Массивы.
4. Управляющие структуры языка (операторы циклов и ветвлений).
5. Классы, интерфейсы, абстрактные классы, объекты.
6. Пакеты.
7. Основы ООП.
8. Класс Object, классы-оболочки, перечисления.
9. Каркас коллекций.
10. Обработка исключений.
11. Паттерны проектирования.
12. Алгоритмы сортировки O(n2)
13. Алгоритм быстрой сортировки.
14. Алгоритмы поиска.
15. Классификация структур данных.
16. Линейные списки. Операции над списками, способы представления.
17. Линейные списки. Связное представление.
18. Линейные списки. Последовательное представление.
19. Линейные списки специального вида (циклические, двунаправленные).
20. Стеки.
21. Очереди.
22. Многомерные структуры переменного размера. Графы, структуры, деревья.
23. Деревья: определения, формы представления.
24. Бинарные деревья: порядок обхода.
25. Бинарные деревья: определение подобия и эквивалентности.
26. Куча. Пирамидальная сортировка.
27. Сбалансированные деревья.
28. Графы основные понятия. Обходы графов.
29. Методы поиска кратчайшего пути в графе.
30. Алгоритмы нахождения остовного дерева.

Примеры практических заданий

1. Реализовать классы Rectangle и Point. Класс Point определяется двумя координатами точки, класс Rectangle определяется точками верхнего левого угла и правого нижнего угла.

Необходимо реализовать приложение, работающее с отрезками. Первый вспомогательный класс – точка – определяется параметрами x и y- координатами на плоскости. Второй вспомогательный класс – отрезок – определяется параметрами $p_1$1и p_2 – точками начала и концы отрезка. Помимо этого, у отрезка должен быть метод **length**, возвращающий длину

этого отрезка. Основной класс должен запросить у пользователя координаты первой и второй точки, создать на их основе отрезок, после чего вывести пользователю его длину.

2. Реализовать классы Rectangle и Point. Класс Point определяется двумя координатами точки, класс Rectangle определяется точками верхнего левого угла и правого нижнего угла.
Необходимо реализовать приложение, работающее с кругами. Первый вспомогательный класс – точка – определяется параметрами x и y- координатами на плоскости. Второй вспомогательный класс – круг – определяется параметрами p и R – точкой центра круга и радиусом. Помимо этого, у отрезка должен быть метод **inside**, который в качестве аргумента принимает точку. Он должен возвращать **true**, если точка находится внутри круга, и **false** в противном случае Основной класс должен запросить у пользователя координаты центра круга и его радиус, после чего создать объект круга. После этого он должен запросить у пользователя координаты точки и вывести ответ – находится ли данная точка внутри круга.

3. Реализовать классы Rectangle и Point. Класс Point определяется двумя координатами точки, класс Rectangle определяется точками верхнего левого угла и правого нижнего угла.
Необходимо реализовать приложение, работающее с отрезками. Первый вспомогательный класс – точка – определяется параметрами x, y и z - координатами точки в трехмерном пространстве. Второй вспомогательный класс – отрезок – определяется параметрами p_1 и p_2 – точками начала и конца отрезка. Помимо этого, у отрезка должен быть метод **length**, возвращающий длину этого отрезка. Основной класс должен запросить у пользователя координаты первой и второй точки, создать на их основе отрезок, после чего вывести пользователю его длину.

4. Реализовать классы Rectangle и Point. Класс Point определяется двумя координатами точки, класс Rectangle определяется точками верхнего левого угла и правого нижнего угла.
Необходимо реализовать приложение, работающее с шарами. Первый вспомогательный класс – точка – определяется параметрами x, y и z - координатами точки в трехмерном пространстве. Второй вспомогательный класс – шар – определяется параметрами p и R – точкой центра шара и радиусом. Помимо этого, у отрезка должен быть метод **inside**, который в качестве аргумента принимает точку. Он должен возвращать **true**, если точка находится внутри шара и **false** в противном случае Основной класс должен запросить у пользователя координаты центра шара и его радиус, после чего создать объект шара. После этого он должен запросить у пользователя координаты точки и вывести ответ – находится ли данная точка внутри шара.

5. Необходимо реализовать приложение, работающее с математическими операциями. Первый вспомогательный класс – сумматор – определяется параметрами x и y – числами, которые надо сложить. Второй вспомогательный класс – субтрактор – определяется тоже двумя параметрами x и y – числами, второе из которого надо вычесть их первого. Оба класса имеют дополнительный метод **calc**, возвращающий результат выполнения операции. Основной класс должен запросить у пользователя два числа и создать на их

основе сумматор и субтрактор, после чего вывести пользователю на экран результаты выполнения операций этими двумя элементами.

6. Необходимо реализовать приложение, работающее с математическими операциями. Первый вспомогательный класс- множитель – определяется параметрами x и y – числами, которые надо перемножить. Второй вспомогательный класс делитель – определяется тоже двумя параметрами x и y – числами, первое из которых надо поделить на второе. Оба класса имеют дополнительный **calc**, возвращающий результат выполнения операции. Основной класс должен запросить у пользователя два числа и создать на их основе множитель и делитель, после чего вывести пользователю на экран результаты выполнения операций этими двумя элементами.

7. Необходимо реализовать приложение, работающее с логическими операциями. Первый вспомогательный класс – конъюнктор, определяется параметрами x и y – двумя булевыми переменными, которые надо сложить с помощью операции 'AND'. Второй вспомогательный класс – дизъюнктор – также определяется двумя параметрами x и y – булевыми переменными, которые надо сложить с помощью операции 'OR'. Оба класса имеют дополнительный метод **calc**, возвращающий результат выполнения операции (также булево значение). Основной класс должен для всех возможных пар значений x и y создать экземпляры этих двух элементов и вывести на экран результат их выполнения.

8. Необходимо реализовать приложение, работающее с логическими операциями. Первый вспомогательный класс – исключитель, определяется параметрами x и y – двумя булевыми переменными, которые надо сложить с помощью операции 'XOR'. Второй вспомогательный класс – компаратор – также определяется двумя параметрами x и y – булевыми переменными, которые надо проверить на равенство. Оба класса имеют дополнительный метод **calc**, возвращающий результат выполнения операции (также булево значение). Основной класс должен для всех возможных пар значений x и y создать экземпляры этих двух элементов и вывести на экран результат их выполнения.

9. Необходимо реализовать приложение, работающее со строками. Первый вспомогательный класс – повышатель, определяется параметром **str** – строкой, все символы которой надо преображать в верхний регистр. Второй вспомогательный класс – понижатель – определяется параметром **str** – строкой, все символы которой надо в нижний регистр. Оба класса имеют дополнительный метод **out put**, который возвращает результат преобразования строки. Основной класс должен запросить у пользователя строку, создать на ее основе экземпляры этих двух классов и вывести на экран результат их выполнения.

10. Необходимо реализовать приложение, работающее со строками. Первый вспомогательный класс – инвертор, определяется параметром **str** – строкой, символы которой надо вывести в обратном порядке. Второй вспомогательный класс – компрессор – определяется параметром **str** – строкой, из которой надо удалить все пробелы. Оба класса имеют дополнительный метод **out put**, который возвращает результат преобразования строки. Основной класс должен запросить у пользователя строку, создать на ее основе экземпляры этих двух классов и вывести на экран результат их выполнения.