

МИНОБРНАУКИ РОССИИ
Ярославский государственный университет им. П.Г. Демидова

Кафедра компьютерной безопасности и математических методов обработки информации

УТВЕРЖДАЮ

Декан математического факультета

Нестеров П.Н.

20 мая 2025 г.

Рабочая программа дисциплины

Методы программирования

Направление подготовки (специальности)
10.05.01 Компьютерная безопасность

Направленность (профиль)
«Математические методы защиты информации»

Форма обучения очная

Программа рассмотрена
на заседании кафедры
от 24.04.2025, протокол № 8

Программа одобрена НМК
математического факультета
протокол № 9 от 05.05.2025

1. Цели освоения дисциплины

Целью изучения дисциплины «Методы программирования» является теоретическая и практическая подготовка специалистов к деятельности, связанной с применением технологий программирования и с анализом вычислительной сложности алгоритмов, для обеспечения информационной безопасности.

Задачи дисциплины:

- изучение основных подходов к организации процесса разработки программного обеспечения;
- изучение базовых структур данных;
- изучение основных алгоритмов сортировки и поиска;
- изучение основных алгоритмов поиска подстрок;
- изучение основных алгоритмов на графах;
- изучение основных методов оценки вычислительной сложности алгоритмов.

2. Место дисциплины в структуре образовательной программы

Дисциплина «Методы программирования» относится к обязательной части образовательной программы. Для успешного усвоения данной дисциплины необходимо, чтобы студент владел знаниями, умениями и навыками, сформированными в процессе изучения дисциплин:

«Информатика» - работа с программными средствами общего назначения;

«Языки программирования» - знание одного из языков программирования высокого уровня;

«Математический анализ» - знание основных положений теории пределов функций, теории числовых и функциональных рядов;

«Теория вероятностей и математическая статистика» - основные понятия, виды распределений;

«Дискретная математика» - основные понятия и методы дискретной математики, включая дискретные функции, конечные автоматы, комбинаторный анализ и теорию графов.

Дисциплина «Методы программирования» является предшествующей для следующих базовых дисциплин: «Сети и системы передачи информации», «Криптографические методы защиты информации», «Компьютерные сети», «Системы управления базами данных», «Основы построения защищённых операционных систем», «Основы построения защищённых СУБД», «Основы построения защищённых сетей», «Защита программ и данных». Знания и практические навыки, полученные при изучении дисциплины «Методы программирования», используются студентами при разработке курсовых и дипломных работ.

3. Планируемые результаты обучения по дисциплине, соотнесенные с планируемыми результатами освоения образовательной программы

Процесс изучения дисциплины направлен на формирование следующих элементов компетенций в соответствии с ФГОС ВО, ОП ВО и приобретения следующих знаний, умений, навыков и (или) опыта деятельности:

| Формируемая компетенция (код и формулировка) | Индикатор достижения компетенции (код и формулировка) | Перечень планируемых результатов обучения |
|---|--|---|
| Универсальные компетенции | | |

| | | |
|---|--|---|
| УК-2 Способен управлять проектом на всех этапах его жизненного цикла | И-УК-2.1 Знать этапы жизненного цикла проекта, уметь грамотно формулировать цель проекта, уметь выстраивать этапы работы над проектом с учетом последовательности их реализации | Знать: - этапы жизненного цикла проекта; - показатели качества программного обеспечения; - принципы SOLID и основные паттерны проектирования; - основные структуры данных. Уметь: - составлять функциональные и нефункциональные требования к системе; - разрабатывать эффективные алгоритмы и программы; - планировать разработку сложного программного обеспечения |
| | И-УК-2.2 Решать конкретные задачи проекта исходя из действующих правовых норм и имеющихся ресурсов и ограничений, применяя современные методы и технологии; получать запланированный результат в установленные сроки и с заявленным качеством. | Уметь: - оценивать качество готового программного обеспечения. Владеть: - навыками разработки алгоритмов решения типовых профессиональных задач; - навыками документирования программного обеспечения. разбивать поставленные задачи на более мелкие; - давать описание поставленных задач; - определять средства и методы реализации поставленных задач; - навыками разработки проектов, согласно указанным к ним требованиям; - навыками создания технической документации к разрабатываемому проекту. |
| Общепрофессиональные компетенции | | |
| ОПК-7 Способен создавать программы на языках высокого и низкого уровня, применять методы и инструментальные средства программирования для решения профессиональных задач, осуществлять обоснованный выбор инструментария программирования и способов организации программ | И-ОПК-7.1 Способен формализовать задачу, разработать алгоритм ее решения и реализовать его на языках программирования, применяя методы и инструментальные средства программирования | Знать: - современные технологии программирования; Уметь: - разрабатывать эффективные алгоритмы и программы; Владеть: - навыками разработки алгоритмов решения типовых профессиональных задач |
| | И-ОПК-7.2 Знает языки программирования высокого и низкого уровней и современные среды разработки программ | Знать: - базовые структуры данных; - основные комбинаторные и теоретико-графовые алгоритмы, а также способы их эффективной реализации и оценки сложности. - язык программирования C#, его библиотеки и структуры данных; Уметь: - разрабатывать модульные тесты. |

| | | |
|--|--|--|
| | И-ОПК-7.3 Выбирает наиболее подходящие средства и методы программирования для решения профессиональных задач | Уметь: - выбирать подходящую структуру данных для решения конкретной задачи; Владеть: - навыками разработки алгоритмов решения типовых профессиональных задач |
| | И-ОПК-7.4 Владеет навыками кодирования, отладки, тестирования | Уметь: - проводить оценку сложности алгоритмов; Владеть: - навыками тестирования белого и черного ящика; - навыками работы с системой контроля версий. |

4. Объем, структура и содержание дисциплины

Общая трудоемкость дисциплины составляет **8** зачетных единиц, **288** акад. часов.

| № п/ п | Темы (разделы) дисциплины, их содержание | Семестр | Виды учебных занятий, включая самостоятельную работу студентов, и их трудоемкость (в академических часах) | | | | | | Формы текущего контроля успеваемости Форма промежуточной аттестации (по семестрам) |
|--------------|---|---------|---|--------------|--------------|--------------|-----------------------------|---------------------------|--|
| | | | Контактная работа | | | | | | |
| | | | лекции | практические | лабораторные | консультации | аттестационные испытания | самостоятельная работа | |
| 1 | Вводная лекция. Жизненный цикл программного обеспечения | 5 | 2 | 2 | | | | 2 | |
| 2 | Качество программных систем | 5 | 2 | 2 | 2 | 1 | | 8 | Лаб.работа1 |
| 3 | Анализ и разработка требований | 5 | 2 | 2 | 4 | 1 | | 10 | Проект "Разработка приложения" |
| 4 | Проектирование архитектуры программных систем | 5 | 6 | 6 | 8 | 1 | | 20 | |
| 5 | Аттестация и верификация | 5 | 2 | 2 | 2 | 1 | | 8 | |
| 6 | Управление проектами | 5 | 2 | 2 | | 1 | | 2 | |
| | | | | | | | 0,3 | 4,7 | Зачет |
| | Всего за 5 семестр 108 акад. часов | | 16 | 16 | 16 | 5 | 0,3 | 54,7 | |
| 7 | Структуры данных | 6 | 10 | 2 | 6 | 2 | | 30 | Лаб.работа 1, 2, 3, 4, контрольная работа |

| | | | | | | | | | |
|----|---|---|----|----|----|----|-----|-------|---------------------------------|
| 8 | Алгоритмы | 6 | 2 | 4 | | 2 | | 2 | |
| 9 | Сортировки | 6 | 6 | 4 | 4 | 1 | | 15 | Лаб.работа 5 |
| 10 | Алгоритмы поиска подстрок | 6 | 6 | 4 | 2 | 2 | | 15 | Лаб.работа 6 |
| 11 | Алгоритмы на графах | 6 | 8 | 2 | 4 | 1 | | 10 | Контест по алгоритмам на графах |
| | | | | | | 2 | 0,5 | 33,5 | экзамен |
| | Всего за 6 семестр 180 акад. часов | | 32 | 16 | 16 | 10 | 0,5 | 105,5 | |
| | ИТОГО | | 48 | 32 | 32 | 15 | 0,8 | 160,2 | |

Содержание разделов дисциплины:

Раздел 1. Технология программирования

Тема 1. Вводная лекция. Жизненный цикл программного обеспечения

Программные системы. Основные проблемы разработки сложных программных систем. Жизненный цикл ПО. Стандартизация жизненного цикла в системе государственных стандартов ЕСПД и ГОСТ Р ИСО/МЭК 12207. Модели жизненного цикла (каскадная, спиральная, формальные и др.). Гибкие методологии разработки (XP, Scrum и др.).

Тема 2. Качество программных систем

Критерии оценки качества программных систем, характеристики качества и метрики качества. Методы контроля качества. Модель процесса оценивания качества. ГОСТ Р ИСО 9000 и ГОСТ Р ИСО 9126. Инструментальные системы оценки качества программных систем.

Тема 3. Анализ и разработка требований

Функциональные и нефункциональные требования. Разработка требований. Спецификация требований к программному обеспечению, техническое задание (ГОСТ 19.201-78, ГОСТ 34.602-89). Методы первичного сбора требований. Модели системы. Языки спецификации требований (сценарии, PDL, потоковые диаграммы и др.). Оценка требований (ГОСТ Р ИСО/МЭК 12207). Методы аттестации требований (обзор, прототипирование, тестовые сценарии). Управление требованиями. Инструментальные средства отслеживания и контроля требований.

Тема 4. Проектирование архитектуры программных систем

Архитектурное и детальное проектирование. Основные принципы проектирования. Сквозная функциональность. Сцепление и связность. Архитектурные стили (парадигмы: объектно-ориентированная, компонентная, многослойная, DDD, клиент-сервер, многоуровневая, SOA и др.). Принципы объектно-ориентированного проектирования (SOLID). Паттерны проектирования. Унифицированный язык моделирования (UML). Документирование архитектуры.

Тема 5. Аттестация и верификация

Понятие аттестации и верификации. Методы инспектирования. Инструментальные средства инспектирования и оценки качества. Тестирование ПО. Неразрешимость проблемы тестирования. Виды и уровни тестирования. Стратегии восходящего и нисходящего тестирования. Методы «белого» и «черного» ящика. Автоматизированное и ручное тестирование. Разработка через тестирование (TDD). Непрерывная интеграция. Покрываемость кода тестами. Методы отладки программ. Инструментальные средства поддержки тестирования и отладки.

Тема 6. Управление проектами

Группы процессов и процедуры управления проектом. План управления проектом. Инструментальные средства управления проектами. Управление персоналом. Права и обязанности членов коллектива. Мотивация. Организация совместной работы коллектива.

Управление рисками. Модель зрелости возможностей. Инструментальные средства планирования и управления процессом разработки.

Раздел 2. Структуры данных и прикладные алгоритмы

Тема 7. Структуры данных

Массив, стек, очередь, список. Реализация, методы работы с этими структурами данных. Бинарные деревья. Применение деревьев для поиска.

Самобалансирующиеся деревья. АВЛ-деревья. Б-деревья, их применение для внешней сортировки. Список с пропусками. Пирамида (двоичная куча). Ее реализация на основе массива. Хеш-таблицы. Разрешение коллизий методом цепочек. Разрешение коллизий с помощью открытой адресации: применение линейного, квадратичного метода и метода двойного хеширования.

Тема 8. Алгоритмы

Модель вычислений для сравнения алгоритмов. Наихудшее и среднее время работы. Обозначения O и Θ .

Тема 9. Сортировки

Алгоритмы внутренней сортировки: сортировки сравнениями: вставками, выбором и обменами. Оценка сложности работы алгоритмов внутренней сортировки. Пирамидальная сортировка. Время ее работы. Быстрая сортировка. Время ее работы. Основная теорема для оценки работы рекурсивных алгоритмов (без доказательства).

Тема 10. Алгоритмы поиска подстрок

Алгоритм простого поиска подстрок. Алгоритм Рабина-Карпа. Алгоритм Кнута-Морриса-Пратта. Алгоритм Бойера- Мура.

Тема 11. Алгоритмы на графах

Способы задания графов. Алгоритмы обхода графов в ширину и глубину. Алгоритм нахождения двусвязных компонент графа. Остовные деревья минимальной стоимости и алгоритмы их построения. Алгоритмы нахождения кратчайших расстояний между вершинами (Дейкстры, Флойда, Беллмана-Форда). Задача о максимальном потоке (алгоритм Эдмонса-Карпа).

5. Образовательные технологии, в том числе технологии электронного обучения и дистанционные образовательные технологии, используемые при осуществлении образовательного процесса по дисциплине

В процессе обучения используются следующие образовательные технологии:

Вводная лекция – дает первое целостное представление о дисциплине и ориентирует студента в системе изучения данной дисциплины. Студенты знакомятся с назначением и задачами курса, его ролью и местом в системе учебных дисциплин и в системе подготовки в целом. Дается краткий обзор курса, история развития науки и практики, достижения в этой сфере, имена известных ученых, излагаются перспективные направления исследований. На этой лекции высказываются методические и организационные особенности работы в рамках данной дисциплины, а также дается анализ рекомендуемой учебно-методической литературы.

Академическая лекция с элементами лекции-беседы – последовательное изложение материала, осуществляемое преимущественно в виде монолога преподавателя. Элементы лекции-беседы обеспечивают контакт преподавателя с аудиторией, что позволяет привлекать внимание студентов к наиболее важным темам дисциплины, активно вовлекать их в учебный процесс, контролировать темп изложения учебного материала в зависимости от уровня его восприятия.

Практическое занятие – занятие, посвященное освоению конкретных умений и навыков по закреплению полученных на лекции знаний.

Консультации – вид учебных занятий, являющийся одной из форм контроля самостоятельной работы студентов. На консультациях по просьбе студентов рассматриваются наиболее сложные моменты при освоении материала дисциплины, преподаватель отвечает на вопросы студентов, которые возникают у них в процессе самостоятельной работы.

Лабораторная работа – организация учебной работы с реальными материальными и информационными объектами, экспериментальная работа с аналоговыми моделями реальных объектов.

6. Перечень лицензионного и (или) свободно распространяемого программного обеспечения, используемого при осуществлении образовательного процесса по дисциплине

В процессе осуществления образовательного процесса по дисциплине используются:

для формирования материалов для текущего контроля успеваемости и проведения промежуточной аттестации, для формирования методических материалов по дисциплине:

- программы Microsoft Office;
- издательская система LaTeX;
- Adobe Acrobat Reader;

для проведения лабораторных занятий:

- MS Windows 7,10, Microsoft Visual Studio 2017/2019;
- система контроля версий Git.

7. Перечень современных профессиональных баз данных и информационных справочных систем, используемых при осуществлении образовательного процесса по дисциплине (при необходимости)

В процессе осуществления образовательного процесса по дисциплине используются:

- Автоматизированная библиотечно-информационная система «БУКИ-NEXT»
http://www.lib.uniyar.ac.ru/opac/bk_cat_find.php
- Электронная библиотечная система «Лань» <https://e.lanbook.com>
- Электронная библиотечная система «Юрайт» <https://urait.ru>
- Электронная библиотечная система «Консультант студента»
<https://www.studentlibrary.ru>

8. Перечень основной и дополнительной учебной литературы, ресурсов информационно-телекоммуникационной сети «Интернет» (при необходимости), рекомендуемых для освоения дисциплины

а) основная литература

1. Алгоритмы: построение и анализ. / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн; пер. с англ. И. В. Красикова, Н. А. Ореховой, В. Н. Романова - 2-е изд. - М.: Вильямс, 2012. - 1290 с.
2. Кулямин В. В. Технологии программирования. Компонентный подход: учеб. пособие для вузов. / В. В. Кулямин - М.: Интернет-Ун-т Информационных Технологий : БИНОМ. Лаборатория знаний, 2007. - 463 с.
3. Чернышев С. А. Принципы, паттерны и методологии разработки программного

обеспечения: учебное пособие для вузов — Москва: Издательство Юрайт, 2021.
<https://urait.ru/viewer/principy-patterny-i-metodologii-razrabotki-programmnogo-obespecheniya-477495>

4. Якимова О. П., Дольников В. Л. Основные алгоритмы на графах. Текст лекций. - Яр-ль, Изд. ЯрГУ, 2011. <http://www.lib.uni Yar.ac.ru/edocs/iuni/20110210.pdf>

б) дополнительная литература

1. Кнут Дональд Э. Искусство программирования для ЭВМ. Т.1, Основные алгоритмы. - М.: Мир, 1976.

2. Кнут Дональд Э. Искусство программирования для ЭВМ. Т.2, Получисленные алгоритмы. - М.: Мир, 1977.

3. Кнут Дональд Э. Искусство программирования: Учеб.пособие.. Т.3, Сортировка и поиск. / Дональд Э.Кнут; Пер.с англ. Под общ. ред. Ю.В. Козаченко - 2-е изд.,испр.и доп. - М.-СПб.-Киев: Вильямс, 2000. - 822с.

4. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. М.: Мир, 1979.

<https://djvu.online/file/SIt3J8EdmWsy6?ysclid=ljzpbxmg9e828086546>

в) ресурсы сети «Интернет»

1. Проектирование на C# <https://stepik.org/course/3944>

2. Постановка задачи на разработку ПО <https://stepik.org/course/1128>

3. Основы тестирования программного обеспечения:

<https://universarium.org/course/715>

9. Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине

Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине включает в свой состав специальные помещения:

- учебные аудитории для проведения занятий лекционного типа и практических занятий (семинаров);

- учебные аудитории для проведения лабораторных работ (лаборатория информационных технологий);

- учебные аудитории для проведения групповых и индивидуальных консультаций,

- учебные аудитории для проведения текущего контроля и промежуточной аттестации;

- помещения для самостоятельной работы;

- помещения для хранения и профилактического обслуживания технических средств обучения.

Помещения для самостоятельной работы обучающихся оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа в электронную информационно-образовательную среду организации.

Компьютерные классы, оборудованные ПЭВМ класса не ниже Intel Pentium IV, 1 Gb RAM, 100G HDD с установленным программным обеспечением: MS Windows 7/10, Microsoft Visual Studio 2017/2019. Из расчета одна ПЭВМ на одного человека.

Инструментальные средства разработки и управления проектами: Git, Resharper.

Автор:

доцент кафедры компьютерной безопасности и
математических методов обработки информации, к. ф.-м. н.,

Якимова О.П.

**Приложение №1 к рабочей программе дисциплины
«Методы программирования»**

**Фонд оценочных средств
для проведения текущей и промежуточной аттестации студентов
по дисциплине**

**1. Типовые контрольные задания или иные материалы,
используемые в процессе текущей аттестации**

Лабораторный практикум 5 семестр.

Лабораторная работа №1. "Методы программирования". 5 семестр.

В языке C# список(List<T>) реализован на основе массива. Необходимо написать класс, который будет иметь такую же функциональность, как и List<T> , используя технологию Test- Driven Development(TDD). Исходный код:

```
using System;
using System.Collections.Generic;
namespace ListTDD
{
    public class ArrayList<T>
    {
        public int Count { get; private set; }
        public int Capacity { get; private set; }
        private T[] _list;
        public ArrayList()
        {
            Capacity = 10;
            _list = new T[Capacity];
            Count = 0;
        }

        public void Add(T element)
        {
            if (Count >= Capacity)
            {
                IncreaseListCapacity();
            }
            _list[Count] = element;
            Count++;
        }

        private void IncreaseListCapacity()
        {
            var tempArray = new T[Capacity];
            _list.CopyTo(tempArray, 0);
            Capacity = Capacity*2;
            _list = new T[Capacity ];
            tempArray.CopyTo(_list, 0);
        }
    }
}
```

}

1. Используя TDD, добавьте в проект тесты, а затем методы, которые реализуют функциональность, RemoveAt(int position), Contains(T item), и индексатор.
2. Примите приглашение соседа по совместной работе. Клонировите репозиторий.
3. В клонированный проект добавьте новый файл. В этот файл добавьте свою функциональность из проекта. У имени класса в двух файлах напишите слово partial. Сделайте commit. Поместите изменения в удаленное хранилище.

Проект "Разработка приложения" по дисциплине "Методы программирования". 5 семестр.

1. Разработать и документировать требования к программному продукту(по вариантам. Варианты по книге: Большакова Елена Игоревна Задания практикума по объектно-ориентированному программированию: Учебно-методическое пособие. – М.: Издательский отдел факультета ВМК МГУ (лицензия ИД № 05899 от 24.09.2001), 2010 – 48 с) <http://diss.seluk.ru/m-raznoe/771957-1-e-bolshakova-zadaniya-praktikuma-obektno-orientirovannomu-programmirovaniyu-uchebno-metodicheskoe-posobie-moskva-2011-udk-bbk-recen.php>

2. Разработать архитектуру программного продукта по вариантам.
3. Разработать прототип интерфейса и собрать отзывы фокус-группы
4. Осуществить кодирование согласно п. 1-3 с параллельным созданием unit- тестов. Обязательно использовать систему контроля версий.
5. Протестировать работу команды сокурсников, написать отчет о тестировании, оформить баг-репорты.

Задание на разработку может быть предложено самими студентами исходя из их интересов или практических задач.

Контрольная работа по теме "Паттерны проектирования". Образец задания.

1. Вы разрабатываете библиотеку для построения графических интерфейсов. Элементу управления TextView можно добавить новое состояние (рамку) или поведение (возможность прокрутки). По умолчанию элемент управления TextView не имеет полос прокрутки, поскольку они не всегда нужны и не имеет рамки, но также возможно одновременно использовать и полосы прокрутки, и рамку. Спроектируйте необходимые классы. Какой паттерн Вы применили?

Ответ: паттерн Декоратор.

2. Надо написать программу, которая обрабатывает изображения. Пользователь может загрузить изображение, а затем отредактировать его, например, применить фильтр, который может изменить оттенок изображения, или Гауссовский фильтр для его затемнения, чтобы оно выглядело так, как будто оно не в фокусе. И даже применить некое 2D преобразование к нему, чтобы оно не выглядело плоским на поверхности. Спроектируйте набор классов для серии фильтрующих изображения преобразований. Какой паттерн Вы использовали?

Ответ: паттерн Декоратор.

3. Надо написать приложение-чат. Аналог "беседы" вконтакте. Пользователи, кто имеет право доступа к "беседе", 1) получают все сообщения, написанные там; 2) получают уведомления о том, что такой-то участник покинул чат или присоединился к нему. Спроектируйте набор классов, который будет решать эту задачу. Какой паттерн Вы использовали? Где применяется этот паттерн в программировании на C#?

Ответ: паттерн Наблюдатель. Используется при обработке событий.

4. Существует веб-сервис, который отслеживает курсы валют на текущий момент. На основе данных сервиса строятся графики, которые визуализируют изменения курса валют (курс для разных валют и соотношения между ними). После того, как сервис

установит курс валют на текущее время, должны измениться все графики. Спроектируйте набор классов, который будет решать эту задачу. Какой паттерн Вы использовали? Где применяется этот паттерн в программировании на C#?

Ответ: паттерн Наблюдатель. Используется при обработке событий.

5. Хорошо известно, что небольшие объемы данных лучше сортировать с помощью простых алгоритмов сортировки, например, сортировкой вставками, а гигантские - пирамидальной или быстрой сортировкой. Причем быстрая сортировка быстрее работает, но требует больше памяти из-за рекурсии. Разработайте диаграмму классов для приложения. На вход поступает массив объектов, которые реализуют интерфейс `IComparable<T>`. Приложение оценивает объем поступившего массива и сортирует подходящим алгоритмом. Какой паттерн Вы использовали?

Ответ: паттерн Стратегия.

6. Разработайте диаграмму классов для приложения Калькулятор. Калькулятор умеет складывать, вычитать, отменять операции и вновь выполнять их (undo, redo) над целыми числами. Какой паттерн Вы использовали?

Ответ: паттерн Команда.

Контрольный тест по теме "Тестирование" (ЭУК LMS Moodle)

Вопрос 1. К основным преимуществам автоматизированных тестов можно отнести

1. Оперативное информирование о состоянии качества продукта
2. Удобство их запуска из консоли
3. Исключение человеческого фактора при проверке
4. Гибкость составления тестовых наборов

Вопрос 2. В качестве дополнительной информации к сообщению об ошибке можно приложить

- a) Снимок экрана с дефектом
- b) Логи во время возникновения дефекта
- c) Видео воспроизведения дефекта
- d) Тестовые данные необходимые для воспроизведения дефекта

Вопрос 3. Тестирование методом белого, серого и черного ящиков - это тестирование, которое классифицировано

- a) По знанию тестируемого объекта
- b) По степени подготовленности к тестированию
- c) По моменту проведения
- d) По характеру сценариев

Вопрос 4. Обеспечения качества - это процесс или результат формирования и поддержки требуемых характеристик и свойств продукции на этапах:

- a) создания
- b) хранения
- c) транспортирования
- d) эксплуатации

Вопрос 5. Что из перечисленного относится к внутренним процедурам обеспечения качества?

- a) тестирование спецификации
- b) интеграционное тестирование
- c) юнит-тесты
- d) обработка заявок в техническую поддержку

Вопрос 6. Позволяет экономить время на генерации тестовых данных

- a) Нагрузочное тестирование
- b) Объемное тестирование

- c) Автоматизированное тестирование
- d) Фокус-тестирование

Вопрос 7. В описание ошибки необходимо указывать

- a) Последовательность действий для воспроизведения
- b) Ожидаемый результат
- c) ФИО потенциально виновного разработчика
- d) Фактический результат

Вопрос 8. Тестирование на уровне кода, когда проверяется работа конкретной функции или метода- это

- a) Модульное тестирование
- b) Тестирование методом свободного поиска
- c) Функциональное тестирование
- d) Интеграционное тестирование

Вопрос 9. Тестирование спецификации включает проверки на:

- a) грамотность, политкорректность
- b) логику, полноту
- c) функциональность, однозначность
- d) понятность, удобство

Вопрос 10. Что из перечисленного НЕ входит в основные этапы обеспечения качества?

- a) оценка уровня качества имеющихся на рынке аналогичных изделий, анализ требований покупателей
- b) пооперационный контроль в процессе производства
- c) исправление критических дефектов
- d) контроль качества изделия в условиях эксплуатации (после продажи)

Ключ к тесту

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|-----|---------|---|---------|------|---|-------|---|-----|----|
| a,c | a,b,c,d | a | a,b,c,d | a, c | c | a,b,d | a | b,c | c |

Лабораторная работа №1. "Методы программирования". 6 семестр.

Применение стандартных контейнеров данных для решения задач.

Пример задания: необходимо найти первые 10 часто встречающихся слов из текста объемом в 3-5 Мб. Напишите три метода, решающих эту задачу, причем в одном методе в качестве контейнера для данных используется SortedDictionary<TKey, TValue>, в другом -- Dictionary<TKey, TValue>, в третьем -- SortedList<TKey, TValue>. Сравните время работы для каждого из методов. Почему так происходит?

Лабораторная работа №2. "Методы программирования". 6 семестр.

I. Напишите класс AVLTree<T>, который содержит методы удаления элемента из дерева, поиска элемента в дереве, вставки элемента, балансировки.

Напишите консольное приложение, которое

- генерирует массив из случайных 10000 чисел;
- создает АВЛ-дерево из чисел этого массива;
- удаляет из дерева числа, которые находились в массиве на месте с 5000 до 7000;
- выполняет поиск каждого элемента массива и замеряет общее время, затраченное на поиск;
- создайте SortedDictionary<int, int> на основе чисел массива. Также удалите из этого контейнера числа, которые находились в массиве на месте с 5000 до 7000 и замерьте время, затраченное на поиск всех элементов массива в SortedDictionary.

Сравните время работы двух коллекций.

Лабораторная работа №3. "Методы программирования". 6 семестр.

1. Напишите класс `HashTable<TKey, TValue>`, который содержит открытые методы: удаления элемента, поиска элемента по ключу, вставки элемента, реализует интерфейс `IEnumerable <KeyValuePair< TKey, TValue>>` и необходимые внутренние методы (например, увеличение таблицы).

Разрешение коллизий: методом цепочек либо открытая адресация (линейное, квадратичное или двойное исследование).

2. Напишите юнит-тесты, которые проверяют работу методов класса хеш-таблицы.

3. Напишите консольное приложение, которое замеряет время работы класса `Dictionary` и написанного Вами. Для этого нужно написать два метода.

На входе для каждого метода - массив слов, полученных из файла `WarAndWorld.txt`

В методе создается частотный словарь для слов данного файла. Из него получить выборку слов длины 7. И каждое из слов выборки удалить из основного словаря.

Сравните время работы двух коллекций.

Лабораторная работа № 4. "Методы программирования". 6 семестр.

Реализация и использование списка с пропусками.

Пример задания: допишите метод удаления элемента (поиска элемента и т.п. -- по вариантам) в проект, созданный на лекции. Сравните время работы по поиску, вставке, удалению элементов разных типов двух коллекций --- стандартного `SortedList` и списка с пропусками на не менее чем 10000 элементов.

Исходный код:

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace SkipListLib
{
    internal class Node<TKey, TValue>
    {
        public TKey Key { get; set; }
        public TValue Value { get; set; }
        public Node<TKey, TValue> Right,
            Up,
            Down;
        public Node()
        { }
        public Node(TKey key, TValue value)
        {
            Key = key;
            Value = value;
            Right = null;
            Up = null;
            Down = null;
        }
    }

    public class SkipList<TKey, TValue> where TKey : IComparable<TKey>
    {
        public int Count { get; private set; }
        int _maxLevel;
        int _curLevel;
        double _probability;
```

```

Random _rd;

Node<TKey, TValue>[] _head;
Node<TKey, TValue> _tail;

public SkipList(int maxLevels = 10, double p = 0.5)
{
    _maxLevel = maxLevels;
    _probability = p;
    _curLevel = 0;
    _rd = new Random();
    _head = new Node<TKey, TValue>[_maxLevel];
    _tail = new Node<TKey, TValue>();

    for (int i = 0; i < _maxLevel; i++)
    {
        _head[i] = new Node<TKey, TValue>();
        _head[i].Right = _tail;
        if (i == 0) continue;
        _head[i].Down = _head[i - 1];
        _head[i - 1].Up = _head[i];
    }
}

public void Add(TKey key, TValue value)
{
    var previousItems = new Node<TKey, TValue>[_maxLevel + 1];
    var current = _head[_curLevel];
    for (int i = _curLevel; i >= 0; i--)
    {
        while (current.Right != _tail && current.Right.Key.CompareTo(key) < 0)
        {
            current = current.Right;
        }
        if (current.Right.Key.CompareTo(key) == 0)
            throw new ArgumentException("Key must be unique");
        previousItems[i] = current;
        current = current.Down;
    }
    int height = 0;
    while (_rd.NextDouble() < _probability && height < _maxLevel)
    {
        height++;
    }
    if (height > _curLevel)
    {
        for (int i = _curLevel + 1; i <= height; i++)
        {
            previousItems[i] = _head[i];
        }
        _curLevel = height;
    }
    for (int i = 0; i <= height; i++)

```

```

    {
        var newItem = new Node<TKey, TValue>(key, value);
        newItem.Right = previousItems[i].Right;
        previousItems[i].Right = newItem;
        if (i == 0) continue;
        newItem.Down = previousItems[i - 1].Right;
        previousItems[i - 1].Right.Up = newItem;
    }
}
}
}

```

**Лабораторная работа № 5. "Методы программирования". 6 семестр.
Использование очереди с приоритетами(пирамиды, кучи) для решения
различных задач**

I. Напишите класс `BinaryHeap<T>`, который содержит открытые методы вставки элемента, удаления наибольшего(наименьшего) элемента, поиска наибольшего(наименьшего) элемента и необходимые закрытые методы(например, метод восстановления свойств пирамиды).

II. Напишите приложение, решающее задачу(по вариантам) и использующее описанный Вами класс. Пример задачи: Дано k отсортированных списков с общим количеством n элементов. Надо их соединить в один отсортированный список за время $O(n \log k)$, используя пирамиду.

Лабораторная работа №6. "Методы программирования". 6 семестр.

1. Опишите интерфейс для алгоритма поиска подстроки. Метод интерфейса должен возвращать список позиций, начиная с которых подстрока входит в текст.

Напишите два класса, реализующие этот интерфейс, для алгоритмов, указанных в Вашем варианте.(Алгоритм простого поиска подстрок. Алгоритм Рабина-Карпа,

Напишите юнит-тесты для каждого из алгоритмов.

2. Напишите приложение, которое ищет все вхождения подстроки, заданной пользователем в большом текстовом файле("anna.txt").

Сравните время работы двух различных алгоритмов.

Лабораторная работа № 7. "Методы программирования". 6 семестр.

1. Опишите класс граф. Граф считывается из файла. В первой строке файла указывается количество вершин в графе. Во второй строке - номера вершин, смежных с нулевой, в третьей - номера вершин смежных с первой и т.д.

2. Напишите реализацию алгоритма согласно варианту. (Алгоритм Беллмана-Форда, Алгоритм поиска максимального потока Алгоритм Краскала (список ребер хранится в куче) Алгоритм Дейкстры (список ребер хранится в куче) Алгоритм выделения двусвязных компонент Алгоритм Флойда

Лабораторную работу № 7 можно сдать дистанционно решив контест

<https://official.contest.yandex.ru/contest/8169/enter/>

Контрольная работа по теме "бинарные деревья". Пример задания.

1. Построить AVL-дерево по последовательности ключей: 7, 11, 44, 27, 29, 16, 13, 15, 18, 3
2. Удалить из построенного AVL-дерева последовательность ключей: 15, 7, 29, 11

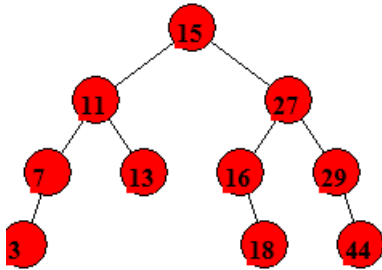
3. Написать метод, который возвращает максимальный элемент в AVL-дереве.

```
public T GetMaxTreeItem()
```

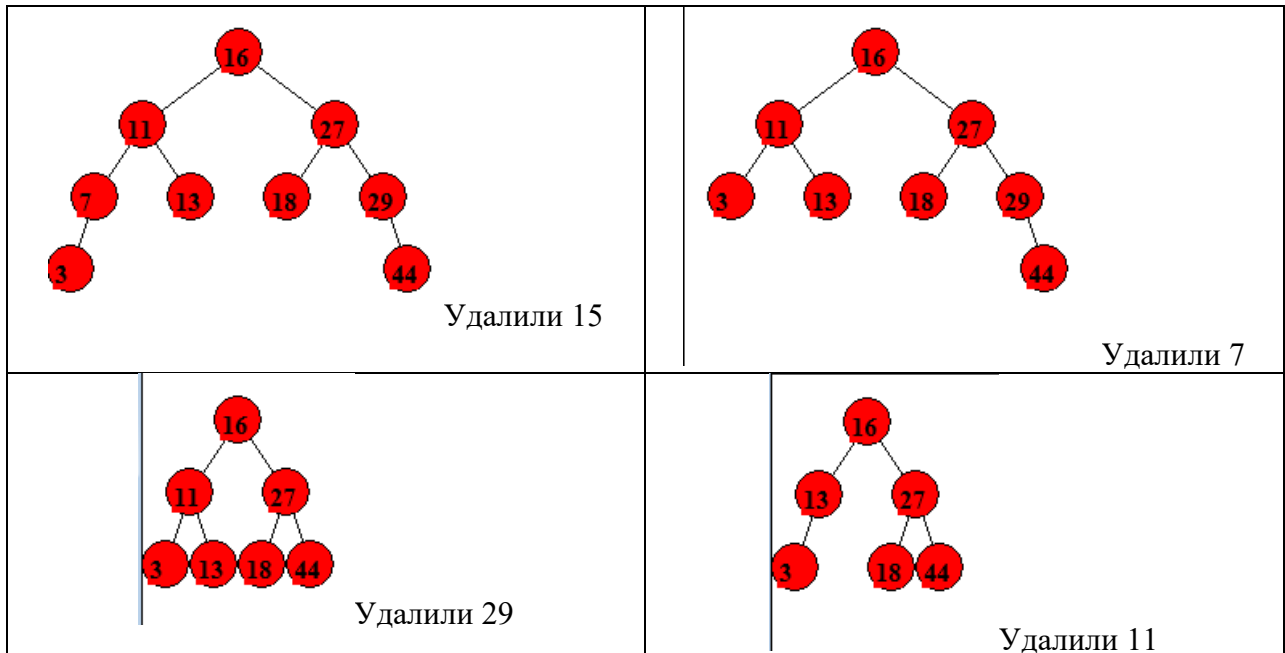
4. Написать метод обхода бинарного дерева

Ответ:

1.



2.



3. Например, код может быть таким:

```

public T GetMaxTreeItem()
{
    if( _root== null)
    { throw new ArgumentNullException();}
    if(_root.Right == null)
        return _root.Value;
    var current = _root.Right;
    while( current.Right != null)
        current = current.Right;
    return current.Value;
}
  
```

4. Например, код может быть таким:

```

public class AVLTreeTraverse
{
    public IEnumerable<AVLTreeNode<T>> Traverse<T>(AVLTreeNode <T> node)
    {
        var nodeWalkOrder = new List<AVLTreeNode<T>>();
        if (node != null)
  
```



```

    {
        nodeWalkOrder.Add(node);
        nodeWalkOrder.AddRange(Traverse(node.LeftNode));
        nodeWalkOrder.AddRange(Traverse(node.RightNode));
    }
    return nodeWalkOrder;
}
}

```

Таблица соответствия контрольных мероприятий, компетенций и индикаторов их достижения

| Контрольное мероприятие | Индикатор освоения компетенции |
|--|---------------------------------------|
| <i>Работа на практических занятиях</i> | И-УК-2.1, И-ОПК-7.2, И-ОПК-7.1 |
| <i>Сдача проекта</i> | И-УК-2.1, И-ОПК-7.1 |
| <i>Лабораторные работы 1-6</i> | И-ОПК-7.1 |
| <i>Тест по оценке качества ПО</i> | И-ОПК-7.2 |
| <i>Контрольные работы</i> | И-ОПК-7.1 |

2. Список вопросов и (или) заданий для проведения промежуточной аттестации

Вопросы к зачету по дисциплине «Методы программирования» V семестр

1. Программные системы. Основные проблемы разработки сложных программных систем. Принципы работы со сложными системами
2. Жизненный цикл ПО. Стандартизация жизненного цикла в системе государственных стандартов ЕСПД и ГОСТ Р ИСО/МЭК 12207.
3. Модели жизненного цикла (каскадная, спиральная, формальные и др.).
4. Гибкие методологии разработки (XP, Scrum, RUP и др.).
5. Анализ предметной области и требования к ПО. Функциональные и нефункциональные требования. Разработка требований. Спецификация требований к программному обеспечению, техническое задание.
6. Архитектурное и детальное проектирование. Основные принципы проектирования(структурирование системы, моделирование управления, модульная декомпозиция).
7. Архитектура распределенных систем. Характеристики распределенных систем. Архитектура клиент-сервер(модели тонкого и толстого клиента). Архитектура распределенных объектов.
8. Объектно-ориентированное проектирование. SOLID. Процесс объектно-ориентированного проектирования: окружение системы и модели ее использования, проектирование архитектуры, определение объектов, модели архитектуры, специфицирование интерфейсов объектов.
9. Проектирование с повторным использованием компонентов. Достоинства и недостатки. Покомпонентная разработка. Семейства приложений. Проектные паттерны(фасад, одиночка, адаптер, фабрика, команда, стратегия, шаблонный метод, наблюдатель, декоратор, итератор).
10. Проектирование интерфейса пользователя. Принципы проектирования интерфейсов пользователя. Взаимодействие с пользователем. Представление информации. Средства поддержки пользователя. Оценивание интерфейса.
11. Качество ПО и методы его контроля. Качество программного обеспечения. Методы контроля качества. Тестирование. Проверка на моделях. Виды и уровни тестирования. Стратегии восходящего и нисходящего тестирования. Методы «белого» и «черного» ящика. Автоматизированное и ручное тестирование. Разработка через

тестирование (TDD). Покрытие кода тестами.

12. Управление проектом. Инструментальные средства управления проектами. Управление персоналом. Права и обязанности членов коллектива. Мотивация. Организация совместной работы коллектива.

Промежуточная аттестация по дисциплине проводится в форме зачета.

Зачет проводится в форме собеседования. К моменту зачета у студента должен быть сдан проект, сделана лабораторная работа, пройден тест. Согласно случайному выбору студент дает ответ на один вопрос, вынесенный на зачет.

Вопросы к экзамену по дисциплине «Методы программирования» VI семестр

I. Структуры данных.

1. Массив, стек, очередь, список. Словарь. Реализация, методы работы с этими структурами данных.
2. Бинарные деревья. Применение деревьев для поиска.
3. Самобалансирующиеся деревья. AVL-деревья.
4. Б-деревья, их применение для внешней сортировки.
5. Список с пропусками.
6. Пирамида (двоичная куча). Ее реализация на основе массива.
7. Хеш-таблицы. Разрешение коллизий методом цепочек. Разрешение коллизий с помощью открытой адресации: применение линейного, квадратичного метода и метода двойного хеширования.

II. Алгоритмы. Модель вычислений для сравнения алгоритмов. Наихудшее и среднее время работы. Обозначения O и Θ .

III. Сортировки.

1. Алгоритмы внутренней сортировки: сортировки сравнениями: вставками, выбором и обменами. Оценка сложности работы алгоритмов внутренней сортировки.
2. Пирамидальная сортировка. Время ее работы.
3. Быстрая сортировка. Время ее работы.
4. Основная теорема для оценки работы рекурсивных алгоритмов (без доказательства).

IV. Алгоритмы поиска подстрок.

1. Алгоритм простого поиска подстрок.
2. Алгоритм Рабина-Карпа.
3. Алгоритм Кнута-Морриса-Пратта.
4. Алгоритм Бойера-Мура.

V. Алгоритмы на графах.

1. Способы задания графов.
2. Алгоритмы обхода графов в ширину и глубину.
3. Алгоритм нахождения двусвязных компонент графа.
4. Остовные деревья минимальной стоимости и алгоритмы их построения.
5. Алгоритмы нахождения кратчайших расстояний между вершинами (Дейкстра, Флойд, Беллмана-Форда).
6. Задача о максимальном потоке (алгоритм Эдмонса-Карпа).

Экзаменационный ответ оценивается по 4-х бальной системе, в соответствие с которой выставляются оценки «отлично», «хорошо», «удовлетворительно», «неудовлетворительно».

Правила выставления оценки:

оценка «отлично» выставляется студенту, если он владеет материалом дисциплины; четко и определенно отвечает на вопросы, легко сравнивает различные части, разбирает

новые и сложные предлагаемые ему случаи, видит и понимает взаимосвязь понятий, легко справляется с практическими заданиями.

оценка **«хорошо»** выставляется студенту, если он твердо знает и понимает материал, грамотно и по существу излагает его, не допускает существенных неточностей в ответе на поставленные вопросы, правильно применяет теоретические положения при решении практических вопросов и задач, владеет необходимыми навыками и приемами их выполнения.

оценка **«удовлетворительно»** выставляется студенту, если он имеет знания основного материала, но не усвоил его деталей, может ответить на вопросы по определениям основных понятий, но приходит в замешательство от заданий по объяснению взаимосвязи или доказательству положений дисциплины; испытывает затруднения при выполнении практических работ.

оценка **«неудовлетворительно»** выставляется студенту, который не знает значительной части программного материала, допускает существенные ошибки, неуверенно, с большими затруднениями и ошибками выполняет практические работы.

3. Методические рекомендации преподавателю по процедуре оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Целью процедуры оценивания является определение степени овладения студентом ожидаемыми результатами обучения (знаниями, умениями, навыками и (или) опытом деятельности).

Процедура оценивания степени овладения студентом ожидаемыми результатами обучения осуществляется с помощью методических материалов, представленных в разделе «Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций»

3.1 Критерии оценивания степени овладения знаниями, умениями, навыками и (или) опытом деятельности, определяющие уровни сформированности компетенций

Пороговый уровень (общие характеристики):

- владение основным объемом знаний по программе дисциплины;
- знание основной терминологии данной области знаний, стилистически грамотное, логически правильное изложение ответа на вопросы без существенных ошибок;
- владение инструментарием дисциплины, умение его использовать в решении стандартных (типовых) задач;
- способность самостоятельно применять типовые решения в рамках рабочей программы дисциплины;
- усвоение основной литературы, рекомендованной рабочей программой дисциплины;
- знание базовых теорий, концепций и направлений по изучаемой дисциплине;
- самостоятельная работа на практических и лабораторных занятиях, периодическое участие в групповых обсуждениях, достаточный уровень культуры исполнения заданий.

Продвинутый уровень (общие характеристики):

- достаточно полные и систематизированные знания в объеме программы дисциплины;
- использование основной терминологии данной области знаний, стилистически грамотное, логически правильное изложение ответа на вопросы, умение делать выводы;

- владение инструментарием дисциплины, умение его использовать в решении учебных и профессиональных задач;
- способность самостоятельно решать сложные задачи (проблемы) в рамках рабочей программы дисциплины;
- усвоение основной и дополнительной литературы, рекомендованной рабочей программой дисциплины;
- умение ориентироваться в базовых теориях, концепциях и направлениях по изучаемой дисциплине и давать им сравнительную оценку;
- самостоятельная работа на практических и лабораторных занятиях, участие в групповых обсуждениях, высокий уровень культуры исполнения заданий.

Высокий уровень (общие характеристики):

- систематизированные, глубокие и полные знания по всем разделам дисциплины;
- точное использование терминологии данной области знаний, стилистически грамотное, логически правильное изложение ответа на вопросы, умение делать обоснованные выводы;
- безупречное владение инструментарием дисциплины, умение его использовать в постановке и решении научных и профессиональных задач;
- способность самостоятельно и творчески решать сложные задачи (проблемы) в рамках рабочей программы дисциплины;
- полное и глубокое усвоение основной и дополнительной литературы, рекомендованной рабочей программой дисциплины;
- умение ориентироваться в основных теориях, концепциях и направлениях по изучаемой дисциплине и давать им критическую оценку;
- активная самостоятельная работа на практических и лабораторных занятиях, творческое участие в групповых обсуждениях, высокий уровень культуры исполнения заданий.

3.2 Описание процедуры выставления оценки

В зависимости от уровня сформированности каждой компетенции по окончании освоения дисциплины студенту выставляется оценка. Для дисциплин, изучаемых в течение нескольких семестров, оценка может выставляться не только по окончании ее освоения, но и в промежуточных семестрах. Вид оценки («отлично», «хорошо», «удовлетворительно», «неудовлетворительно», «зачтено», «незачтено») определяется рабочей программой дисциплины в соответствии с учебным планом.

Оценка «отлично» выставляется студенту, у которого каждая компетенция (полностью или частично формируемая данной дисциплиной) сформирована на высоком уровне.

Оценка «хорошо» выставляется студенту, у которого каждая компетенция (полностью или частично формируемая данной дисциплиной) сформирована не ниже, чем на продвинутом уровне.

Оценка «удовлетворительно» выставляется студенту, у которого каждая компетенция (полностью или частично формируемая данной дисциплиной) сформирована не ниже, чем на пороговом уровне.

Оценка «неудовлетворительно» выставляется студенту, у которого хотя бы одна компетенция (полностью или частично формируемая данной дисциплиной) сформирована ниже, чем на пороговом уровне.

Оценка «зачет» выставляется студенту, у которого каждая компетенция (полностью или частично формируемая данной дисциплиной) сформирована не ниже, чем на пороговом уровне.

Оценка «незачтено» выставляется студенту, у которого хотя бы одна компетенция (полностью или частично формируемая данной дисциплиной) сформирована ниже, чем на пороговом уровне.

Приложение №2 к рабочей программе дисциплины «Методы программирования»

Методические указания для студентов по освоению дисциплины

Основной формой изложения учебного материала по дисциплине «Методы программирования» являются лекции, причем в форме лекции-беседы или мастер-класса. По большинству тем предусмотрены лабораторные занятия, на которых происходит закрепление лекционного материала путем применения его к конкретным задачам и отработка навыков работы по разработке алгоритмов и различных структур данных, или работы с различным программным обеспечением(например, с ПО по контролю версий). В пятом семестре для закрепления материала по темам "Анализ и разработка требований", " Проектирование архитектуры программных систем", " Аттестация и верификация ПО" необходимо выполнить проект: спроектировать, разработать и протестировать приложение, решающее реальную задачу. Проект выполняется командой из двух человек, при этом обязательно использовать систему контроля версий и общий репозиторий.

Для успешного освоения дисциплины очень важно решение достаточно большого количества задач, требующих разработки алгоритма и написания программы, как в аудитории, так и самостоятельно в качестве домашних заданий. Примеры решения задач разбираются на лекциях и практических занятиях, при необходимости по наиболее трудным темам проводятся дополнительные консультации. Для решения всех задач необходимо знать и понимать лекционный материал. Поэтому в процессе изучения дисциплины рекомендуется регулярное повторение пройденного лекционного материала. Материал, законспектированный на лекциях, необходимо дома еще раз прорабатывать и при необходимости дополнять информацией, полученной на консультациях, практических занятиях или из учебной литературы.

Большое внимание должно быть уделено выполнению домашней работы. В качестве заданий для самостоятельной работы дома студентам предлагаются задачи, аналогичные разобранным на лекциях и практических занятиях или немного более сложные, которые являются результатом объединения нескольких базовых задач.

Для проверки и контроля усвоения материала в течение обучения при сдаче лабораторных работ преподаватель задает вопросы позволяющие выяснить понимание материала. Также проводятся консультации (при необходимости) по разбору заданий для самостоятельной работы, которые вызвали затруднения.

В конце 5 семестра студенты сдают зачет, а в конце бго - экзамен. Экзамен принимается по экзаменационным билетам, каждый из которых включает в себя один теоретический вопрос и три задачи. На самостоятельную подготовку к экзамену выделяется 3 дня, во время подготовки к экзамену предусмотрена групповая консультация.

1. Написать метод, который возвращает максимальный элемент в АВЛ-дереве(в бинарном дереве поиска).
2. Написать метод, который возвращает минимальный элемент в АВЛ-дереве(в бинарном дереве поиска).
3. Написать метод обхода АВЛ-дерева(бинарного дерева поиска).
4. Написать метод вставки элемента в бинарное дерево поиска.
5. Написать реализацию любого алгоритма сортировки целочисленного массива
6. Написать реализацию алгоритма поиска в глубину на графе
7. Написать реализацию алгоритма поиска в ширину на графе
8. Написать реализацию метода, который возвращает число компонент связности графа