МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждение
высшего образования

«Российский государственный гуманитарный университет»
(ФГАОУ ВО «РГГУ»)

ИНСТИТУТ СОЦИАЛЬНО-ЭКОНОМИЧЕСКИХ НАУК
ЭКОНОМИЧЕСКИЙ ФАКУЛЬТЕТ
Кафедра информационных технологий и систем

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

38.05.02 Таможенное дело
Код и наименование направления подготовки/специальности

Таможенное обеспечение внешнеэкономической деятельности
Наименование направленности (профиля)/ специализации

Уровень высшего образования: *Специалитет*

Форма обучения: *Очная. заочная*

РПД адаптирована для лиц
с ограниченными возможностями
здоровья и инвалидов

Москва 2026

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ
Рабочая программа дисциплины

Составитель:
Канд.филол.наук, доц. Муромцева А.В.

УТВЕРЖДЕНО:
Протокол заседания кафедры
№10 от 16.12 2025 года

ОГЛАВЛЕНИЕ

1 Пояснительная записка

1.1 Цель и задачи дисциплины

Цель дисциплины: формирование у обучающихся знаний основных принципов проектирования и анализа алгоритмов и структур данных, знаний основных типов алгоритмов, применяемых в современном программировании для обработки соответствующих структур данных, а также умений обоснования корректности алгоритмов, их практической реализации, теоретической и экспериментальной оценки их временной сложности, развитие необходимых практических навыков их применения в будущей профессиональной деятельности.

Задачи дисциплины:

- ознакомление с разнообразием структур данных и их реализациями в проектировании алгоритмов;
- изучение основных операций над структурами данных в современном программировании;
- овладение структурным подходом к разработке алгоритмов;
- формирование и развитие у обучаемых конкретных практических умений и навыков проектирования и анализа алгоритмов, и структур данных.

1.2 Перечень планируемых результатов обучения по дисциплине (модулю), соотнесенных с индикаторами достижения компетенций

Изучение дисциплины направлено на формирование у обучающихся следующих компетенций:

Компетенция (код и наименование)	**Индикаторы компетенций** (код и наименование)	**Результаты обучения**
ОПК-6 Способен понимать принципы работы современных информационных технологий и использовать их для решения задач профессиональной деятельности	ОПК-6.1 Способен понимать принципы работы и применения современных информационных технологий, в том числе технологий искусственного интеллекта	Знать: базовые принципы работы современных информационных технологий, используемых в алгоритмизации и программировании (включая технологии ИИ); основные алгоритмические языки программирования и их особенности; фундаментальные алгоритмы и структуры данных (сортировка, поиск, стеки, очереди, деревья, хеш-таблицы); принципы работы систем автоматизированного анализа кода и инструментов на основе ИИ; основы машинного обучения и нейронных сетей, их применение в программировании; понятия и технологии работы с большими данными (Big Data), их роль в разработке алгоритмов; принципы автономной отладки и тестирования программ, включая использование автоматизированных систем тестирования; архитектурные паттерны и принципы модульности в программировании, применяемые для создания масштабируемых решений. Уметь: анализировать задачи с точки зрения применимости современных ИТ-решений, включая технологии ИИ;

		выбирать подходящие алгоритмы и структуры данных для решения конкретных задач; использовать среды разработки (IDE) с поддержкой ИИ (например, Visual Studio Code, PyCharm) для повышения продуктивности разработки; применять базовые методы машинного обучения для автоматизации рутинных задач (например, классификация данных, предсказание результатов); работать с библиотеками и фреймворками для анализа данных; оценивать эффективность алгоритмов с учётом временных и пространственных сложностей (интегрировать готовые API и сервисы ИИ (например, для распознавания речи, обработки изображений) в собственные программы; формулировать требования к программному продукту с учётом современных ИТ-трендов. Владеть: навыками работы с современными средами разработки и инструментами отладки; приёмами оптимизации кода с использованием профилировщиков и анализаторов производительности; базовыми навыками работы с системами контроля версий (Git, GitHub); умениями применять технологии ИИ для автоматизации типовых задач разработки (генерация кода, поиск ошибок, оптимизация структуры программ); навыками работы с базами данных и SQL-запросами в контексте разработки алгоритмов; методами оценки надёжности и безопасности программных решений с учётом современных ИТ-угроз; практическими навыками разработки простых приложений с элементами ИИ (например, чат-бот, система рекомендаций).

	ОПК-6.2 Способен выбирать и применять современные информационные технологии для обработки и анализа данных, соответствующие содержанию профессиональных задач	Знать: методы сбора, хранения и обработки данных с использованием алгоритмических подходов; основные парадигмы программирования и области их применения; технологии работы с файлами и потоками данных, сериализацию/десериализацию; принципы работы с API и веб-сервисами для обмена данными (REST, JSON, XML); основы работы с облачными платформами и хранилищами данных; алгоритмы обработки строк, работы с графами, деревьями решений, используемые в анализе данных; подходы к визуализации данных с помощью программных средств; этические и правовые аспекты работы с данными. Уметь: формулировать задачу обработки данных в терминах алгоритмизации и выбирать подходящий язык программирования; разрабатывать алгоритмы для решения задач анализа данных; реализовывать программы для автоматизации сбора и предварительной обработки данных; использовать инструменты ETL (Extract, Transform, Load) для перемещения данных между хранилищами; применять статистические методы и алгоритмы машинного обучения для выявления закономерностей в данных; строить простые модели прогнозирования и классификации с использованием готовых библиотек; визуализировать результаты анализа данных, подбирать оптимальные типы графиков и диаграмм; оценивать корректность и эффективность разработанных алгоритмов обработки данных. Владеть: навыками написания эффективного кода для обработки больших объёмов данных с учётом ограничений по времени и памяти; умениями работать с JSON, XML, CSV и другими форматами хранения данных; приёмами интеграции данных из разных источников с использованием API и скриптов; навыками разработки модульных программ с разделением на функции/методы, соблюдения

		принципов DRY (Don’t Repeat Yourself) и KISS (Keep It Simple, Stupid); практическими навыками работы с системами управления базами данных (SQLite, PostgreSQL — на базовом уровне); методами отладки и тестирования программ, обрабатывающих реальные данные (юнит-тесты, интеграционные тесты); инструментами профилирования и оптимизации кода для задач анализа данных (cProfile, line_profiler и др.); навыками документирования алгоритмов и программ, составления технической документации по результатам разработки.

1.3 Место дисциплины в структуре образовательной программы

Дисциплина «Алгоритмы и структуры данных» относится к обязательной части блока дисциплин учебного плана.

В результате освоения дисциплины формируются знания, умения и владения, необходимые для изучения следующих дисциплин и прохождения практик: «Основы анализа данных на Python», «Базы данных», «Моделирование систем и методы оптимизации».

2 Структура дисциплины

Структура дисциплины для очной формы обучения

Общая трудоёмкость дисциплины составляет 3 з.е., 108 ч., в том числе контактная работа обучающихся с преподавателем 42 ч., самостоятельная работа обучающихся 66 ч.

Семестр	Тип учебных занятий	Количество часов
2	Лекция	18
2	Практическая работа	24
	Всего:	42

3 Содержание дисциплины

№	Наименование раздела дисциплины	Содержание
1.	Тема 1. Понятия алгоритма и структур данных	Место учебной дисциплины в системе подготовки по направлению. Понятие алгоритма и его основные свойства. Свойства алгоритма, определяемые математически и из потребностей экономики. Разработка и реализация алгоритма в виде программы для компьютера. Способы задания алгоритмов: словесный, формульно-словесный, блок-схемный, псевдокодом, структурными диаграммами и языками программирования. Основные элементы блок-схем и изображение в них типов циклов. Системы счисления: двоичная, шестнадцатеричная и десятичная. Алгоритмы перевода чисел из одной системы счисления в другую. Представление в компьютере целых чисел и чисел с плавающей запятой. Представление в компьютере

		текстов, изображений, звука и видео. Понятие структуры данных, её связь с обработкой данных. Уровни структур данных. Уровни данных в программировании. Классификация структур данных. Операции над структурами данных. Структурность данных и структурное программирование. Принцип модульного программирования и его применение.
2.	Тема 2. Анализ алгоритмов	Проверка правильности (верификация) алгоритма. Понятие сложности алгоритма и её анализ. Факторы, определяющие длительность выполнения алгоритма на компьютере. Сравнительные оценки алгоритмов. Классификация алгоритмов по виду функции трудоёмкости. Асимптотический анализ функций трудоёмкости. Трудоёмкость алгоритмов и временные оценки. Примеры анализа простых алгоритмов: суммирования элементов квадратной матрицы, поиска наибольшего элемента в массиве. Методики перехода к временным оценкам работы алгоритма. Теоретический предел трудоёмкости алгоритмов. Рекуррентные соотношения и их использование для оценивания времени работы алгоритмов.
3.	Тема 3. Базовые алгоритмы решений задач	Основные типы алгоритмов: описание. Базовые циклические алгоритмы: описание. Алгоритмические стратегии: описание вариантов и особенностей.
4.	Тема 4. Алгоритмы поиска и выборки	Алгоритмы последовательного поиска. Алгоритмы двоичного поиска. Алгоритмы Фибоначчиева поиска. Алгоритмы интерполяционного поиска. Алгоритмы поиска по бинарному дереву. Алгоритмы поиска по бору. Алгоритмы поиска хешированием. Алгоритмы поиска словесной информации. Алгоритмы выборки из списка.
5.	Тема 5. Алгоритмы сортировки	Понятия и цели сортировки. Сортировки массивов и сортировки файлов, т.е. внутренняя и внешняя сортировка. Терминология. Требования к методам сортировки массивов. Меры эффективности. Сортировка простыми включениями. Сортировка бинарными включениями. Сортировка простым выбором. Метод «пузырька». Шейкерсортировка. Особенности сортировки последовательных файлов. Сортировка последовательных файлов прямым слиянием. Понятие о сортировке естественным слиянием, многопутёвой и многофазной сортировках. Понятие усовершенствованных методов сортировки. Сортировка включениями с убывающим приращением (сортировка Шелла). Сортировка с помощью дерева (сортировка кучей). Пирамидальная сортировка. Сортировка с разделением (быстрая сортировка). Сравнение методов сортировки.
6.	Тема 6. Деревья сортировки и сбалансированные деревья	Определение дерева сортировки, приложения использования. Алгоритм поиска в дереве сортировки. Алгоритм вставки в дерево сортировки. Алгоритм удаления из дерева сортировки. Определение сбалансированного дерева. Балансировка деревьев. АВЛ-деревья, их балансировка, алгоритмы вставки и удаления в них. Красно-чёрные деревья, алгоритмы вставки и удаления в них.

7.	Тема 7. Динамические структуры данных	Линейные связанные списки: однонаправленные и двунаправленные. Очередь, стек, дек – их реализации в виде массива и списка. Циклические связанные списки. Просмотр связанного списка. Общий алгоритм добавления и исключения в списках, очередях, стеках и деках. Рекурсивная обработка списков.
8.	Тема 8. Итеративные и рекурсивные алгоритмы	Итеративный алгоритм. Рекурсивный алгоритм. Рекурсивные структуры данных. Виды обхода бинарных деревьев.
9.	Тема 9. Граф как структура данных. Деревья как частные случаи графов	Граф как структура данных. Основные определения теории графов. Представления графов в программах с помощью матриц. Приложения, использующие графы как структуры данных. Алгоритмы обхода графов: поиск в глубину и поиск в ширину. Алгоритмы поиска кратчайших путей в графе: алгоритм Флойда и алгоритм Дейкстры. Построение кратчайших остовов графа: алгоритм Краскала. Определения ориентированного, упорядоченного, бинарного дерева. Представление деревьев в программе. Код Прюфера для графа, алгоритмы его формирования и восстановления графа по нему. Представление упорядоченных ориентированных деревьев. Представление бинарных деревьев. Определение B-дерева. Алгоритмы поиска в B-дереве. Алгоритм вставки в Вдерево. Алгоритм удаления из B-дерева.

4 Образовательные технологии

В период временного приостановления посещения обучающимися помещений и территории РГГУ для организации учебного процесса с применением электронного обучения и дистанционных образовательных технологий могут быть использованы следующие образовательные технологии:

– видео-лекции**;**

– онлайн-лекции в режиме реального времени;

– электронные учебники, учебные пособия, научные издания в электронном виде и доступ к иным электронным образовательным ресурсам;

– системы для электронного тестирования;

– консультации с использованием телекоммуникационных средств.

5 Оценка планируемых результатов обучения

5.1 Система оценивания

Форма контроля	**Макс. количество баллов**	
	За одну работу	**Всего**
Текущий контроль:		
- защита практических работ	*10 баллов*	*60 баллов*
Промежуточная аттестация *зачет*		*40 баллов*
Итого за семестр *(дисциплину)*		*100 баллов*

Полученный совокупный результат конвертируется в традиционную шкалу оценок и в шкалу оценок Европейской системы переноса и накопления кредитов (European Credit Transfer System; далее – ECTS) в соответствии с таблицей:

1 0 0 - б а л л ь н а я шкала	Традиционная шкала		Шкала ECTS
95 – 100	Отлично	зачтено	A
83 – 94			B
68 – 82	Хорошо		C
56 –67	удовлетворительно		D
50 –55			E
20 – 49	неудовлетворительно	не зачтено	FX
0 – 19			F

5.2 Критерии выставления оценки по дисциплине

Баллы/ Шкала ECTS	**Оценка по дисциплине**	**Критерии оценки результатов обучения по дисциплине**
100-83/ A, B	«отлично»/ «зачтено (отлично)»/ «зачтено»	Выставляется обучающемуся, если он глубоко и прочно усвоил теоретический и практический материал, может продемонстрировать это на занятиях и в ходе промежуточной аттестации. Обучающийся исчерпывающе и логически стройно излагает учебный материал, умеет увязывать теорию с практикой, справляется с решением задач профессиональной направленности высокого уровня сложности, правильно обосновывает принятые решения. Свободно ориентируется в учебной и профессиональной литературе. Оценка по дисциплине выставляются обучающемуся с учётом результатов текущей и промежуточной аттестации. Компетенции, закреплённые за дисциплиной, сформированы на уровне – «высокий».
82-68/ C	«хорошо»/ «зачтено (хорошо)»/ «зачтено»	Выставляется обучающемуся, если он знает теоретический и практический материал, грамотно и по существу излагает его на занятиях и в ходе промежуточной аттестации, не допуская существенных неточностей. Обучающийся правильно применяет теоретические положения при решении практических задач профессиональной направленности разного уровня сложности, владеет необходимыми для этого навыками и приёмами. Достаточно хорошо ориентируется в учебной и профессиональной литературе. Оценка по дисциплине выставляются обучающемуся с учётом результатов текущей и промежуточной аттестации. Компетенции, закреплённые за дисциплиной, сформированы на уровне – «хороший».

Баллы/ Шкала ECTS	**Оценка по дисциплине**	**Критерии оценки результатов обучения по дисциплине**
67-50/ D, E	«удовлетворительно»/ «зачтено (удовлетвори-тельно)»/ «зачтено»	Выставляется обучающемуся, если он знает на базовом уровне теоретический и практический материал, допускает отдельные ошибки при его изложении на занятиях и в ходе промежуточной аттестации. Обучающийся испытывает определённые затруднения в применении теоретических положений при решении практических задач профессиональной направленности стандартного уровня сложности, владеет необходимыми для этого базовыми навыками и приёмами. Демонстрирует достаточный уровень знания учебной литературы по дисциплине. Оценка по дисциплине выставляются обучающемуся с учётом результатов текущей и промежуточной аттестации. Компетенции, закреплённые за дисциплиной, сформированы на уровне – «достаточный».
49-0/ F, FX	«неудовлетвори-тельно»/ не зачтено	Выставляется обучающемуся, если он не знает на базовом уровне теоретический и практический материал, допускает грубые ошибки при его изложении на занятиях и в ходе промежуточной аттестации. Обучающийся испытывает серьёзные затруднения в применении теоретических положений при решении практических задач профессиональной направленности стандартного уровня сложности, не владеет необходимыми для этого навыками и приёмами. Демонстрирует фрагментарные знания учебной литературы по дисциплине. Оценка по дисциплине выставляются обучающемуся с учётом результатов текущей и промежуточной аттестации. Компетенции на уровне «достаточный», закреплённые за дисциплиной, не сформированы.

Критерии оценивания практических работ:

Критерии оценивания / Уровень требований к обучающемуся	Макс. кол-во баллов
Текущий контроль, всего в т.ч.:	**60**
Практическая работа	**10**
Задания выполнены не полностью и (или) допущены две и более ошибки или три и более недочета	1-5
Задания выполнены полностью, но допущены два-три недочета, в т. ч. при ответе на контрольные вопросы	6-7
Задания выполнены полностью, возможна одна неточность, ответы на контрольные вопросы правильные	8-10

5.3 Оценочные средства (материалы) для текущего контроля успеваемости, промежуточной аттестации обучающихся по дисциплине (модулю)

Практические задания приведены в разделе 9.1

Примеры вопросов к экзамену

1. Место учебной дисциплины в системе подготовки по направлению.
2. Понятие алгоритма и его основные свойства.
3. Свойства алгоритма, определяемые математически и из потребностей экономики.
4. Разработка и реализация алгоритма в виде программы для компьютера.
5. Способы задания алгоритмов: словесный, формульно-словесный, блоксхемный, псевдокодом, структурными диаграммами и языками программирования. Основные элементы блок-схем и изображение в них типов циклов.
6. Системы счисления: двоичная, шестнадцатеричная и десятичная. Алгоритмы перевода чисел из одной системы счисления в другую.
7. Представление в компьютере целых чисел и чисел с плавающей запятой.
8. Представление в компьютере текстов, изображений, звука и видео.
9. Понятие структуры данных, её связь с обработкой данных. Уровни структур данных.
10. Уровни данных в программировании. Классификация структур данных.
11. Операции над структурами данных.
12. Структурность данных и структурное программирование.
13. Принцип модульного программирования и его применение.
14. Проверка правильности (верификация) алгоритма. Понятие сложности алгоритма и её анализ. Факторы, определяющие длительность выполнения алгоритма на компьютере.
15. Сравнительные оценки алгоритмов. Классификация алгоритмов по виду функции трудоёмкости.
16. Асимптотический анализ функций трудоёмкости. Трудоёмкость алгоритмов и временные оценки.
17. Примеры анализа простых алгоритмов: суммирования элементов квадратной матрицы, поиска наибольшего элемента в массиве.
18. Методики перехода к временным оценкам работы алгоритма: пооперационный анализ, метод Гиббсона, метод прямого определения среднего времени.
19. Теоретический предел трудоёмкости алгоритмов. Рекуррентные соотношения и их использование для оценивания времени работы алгоритмов.
20. Основные типы алгоритмов: линейный, разветвляющийся с полным и неполным ветвлением, циклический с предусловием и постусловием.
21. Базовые циклические алгоритмы: табулирование функций; организация счетчика.
22. Базовые циклические алгоритмы: накопление суммы или произведения.
23. Базовые циклические алгоритмы: поиск минимального или максимального члена последовательности, поиск минимального или максимального элемента двумерной матрицы.
24. Базовые циклические алгоритмы: сортировка элементов одномерного массива.
25. Алгоритмические стратегии: методы «грубой силы» (перебор всех вариантов); жадные алгоритмы (локально оптимальные).
26. Алгоритмические стратегии: алгоритмы типа «разделяй и властвуй» (декомпозиции); эвристические алгоритмы.
27. Алгоритмические стратегии: алгоритмы поиска с возвратом; поиска методом проб и ошибок.
28. Алгоритмические стратегии: алгоритмы случайного поиска, муравьиные алгоритмы.
29. Алгоритмические стратегии: генетические алгоритмы; эволюционные алгоритмы, алгоритмы численных приближений; алгоритмы сравнения с образцом.
30. Алгоритмы последовательного поиска.
31. Алгоритмы двоичного поиска.
32. Алгоритмы Фибоначчиева поиска.
33. Алгоритмы интерполяционного поиска.

34. Алгоритмы поиска по бинарному дереву.
35. Алгоритмы поиска по бору.
36. Алгоритмы поиска хешированием.
37. Алгоритмы поиска словесной информации.
38. Алгоритмы выборки из списка.
39. Понятия и цели сортировки. Сортировки массивов и сортировки файлов, т.е.
внутренняя и внешняя сортировка. Терминология. Требования к методам сортировки массивов. Меры эффективности.
40. Сортировка простыми включениями.
41. Сортировка бинарными включениями.
42. Сортировка простым выбором.
43. Сортировка методом «пузырька».
44. Шейкер-сортировка.
45. Особенности сортировки последовательных файлов. Сортировка последовательных файлов прямым слиянием. Понятие о сортировке естественным слиянием, многопутёвой и многофазной сортировках.
46. Сортировка включениями с убывающим приращением (сортировка Шелла).
47. Сортировка с помощью дерева (сортировка кучей).
48. Пирамидальная сортировка.
49. Сортировка с разделением (быстрая сортировка).
50. Сравнение методов сортировки.
51. Определение дерева сортировки, приложения использования. Алгоритм поиска в дереве сортировки.
52. Алгоритм вставки в дерево сортировки. Алгоритм удаления из дерева сортировки.
53. Определение сбалансированного дерева. Балансировка деревьев. АВЛ-деревья, их балансировка, алгоритмы вставки и удаления в них. Красно-чёрные деревья, алгоритмы вставки и удаления в них.
54. Линейные связанные списки: однонаправленные и двунаправленные. Очередь, стек, дек – их реализации в виде массива и списка. Примеры приложений, использующих списки, стеки и очереди.
55. Связанные списки. Просмотр связанного списка. Очереди. Общий алгоритм добавления и исключения. Рекурсивная обработка списков. Двусвязные кольца.
56. Итеративный алгоритм. Рекурсивный алгоритм. Рекурсивные структуры данных. Виды обхода бинарных деревьев.
57. Граф как структура данных. Основные определения теории графов. Приложения, использующие графы как структуры данных. Представления графов в программах с помощью матриц.
58. Алгоритмы обхода графов: поиск в глубину и поиск в ширину.
59. Алгоритмы поиска кратчайших путей в графе: алгоритм Флойда 60. Алгоритмы поиска кратчайших путей в графе: алгоритм Дейкстры.
61. Построение кратчайших остовов графа: алгоритм Краскала.
62. Определения ориентированного, упорядоченного, бинарного дерева. Представление деревьев в программе.
63. Код Прюфера для графа, алгоритмы его формирования и восстановления графа по нему.
64. Представление упорядоченных ориентированных деревьев. Представление бинарных деревьев.
65. Определение B-дерева. Алгоритмы поиска в B-дереве.
66. Алгоритм вставки в B-дерево. Алгоритм удаления из B-дерева.

6 Учебно-методическое и информационное обеспечение дисциплины

6.1 Список источников и литературы

Источники

Федеральный закон "Об информации, информационных технологиях и о защите информации" от 27.07.2006 N 149-ФЗ (редакция от 31.07.2023) www.consultant.ru

Литература

Основная

Белов, В. В. Алгоритмы и структуры данных : учебник / В. В. Белов, В. И. Чистякова. - Москва : КУРС : ИНФРА-М, 2020. - 240 с. - (Бакалавриат). - ISBN 978-5-906818-25-6. - Текст : электронный. - URL: https://znanium.com/catalog/product/1057212

Варфоломеева, Т. Н. Структуры данных и основные алгоритмы их обработки : учебное пособие / Т. Н. Варфоломеева. - 2-е изд., стер. - Москва : ФЛИНТА, 2023. - 159 с. - ISBN 978-5-9765-3691-3. - Текст : электронный. - URL: https://znanium.com/catalog/product/2091302

Колдаев, В. Д. Структуры и алгоритмы обработки данных : учебное пособие / В.Д. Колдаев. — Москва : РИОР : ИНФРА-М, 2021. — 296 с. — (Высшее образование: Бакалавриат). — www.dx.doi.org/10.12737/2833. - ISBN 978-5-369-01264-2. - Текст : электронный. - URL: https://znanium.com/catalog/product/1230215

Дополнительная

Григорьев, А. А. Методы и алгоритмы обработки данных : учебное пособие / А.А. Григорьев, Е.А. Исаев. — 2-е изд., перераб. и доп. — Москва : ИНФРА-М, 2022. — 383 с. + Доп. материалы [Электронный ресурс]. — (Высшее образование: Бакалавриат). — DOI 10.12737/1032305. - ISBN 978-5-16-015581-4. - Текст : электронный. - URL: https://znanium.com/catalog/product/1862852

Круз, Р. Л. Структуры данных и проектирование программ : учебное пособие / Р. Л. Круз. - 4-е изд. - Москва : Лаборатория знаний, 2021. - 768 с. - (Программисту). - ISBN 978-5-93208-560-8. - Текст : электронный. - URL: https://znanium.com/catalog/product/1987456

6.2 Перечень ресурсов информационно-телекоммуникационной сети «Интернет»

1. Форум программистов и сисадминов Киберфорум http://www.cyberforum.ru/
2. Клуб программистов https://programmersforum.ru/
3. Форум программистов https://programmersforum.ru/

6.3 Профессиональные базы данных и информационно-справочные системы

Доступ к профессиональным базам данных: https://liber.rsuh.ru/ru/bases

Информационные справочные системы:

1. Консультант Плюс
2. Гарант

7 Материально-техническое обеспечение дисциплины

Для материально-технического обеспечения дисциплины необходимы:

- для лекций:

- учебная аудитория,
- доска,
- проектор (стационарный или переносной),
- компьютер или ноутбук,
- программное обеспечение (ПО).

- для практических занятий:

- лаборатория,
- доска,
- проектор (стационарный или переносной),
- компьютер или ноутбук для преподавателя,
- компьютеры для обучающихся,
- выход в Интернет,
- программное обеспечение (ПО).

Перечень программного обеспечения (ПО)

- для лекций:

№п/п	Наименование ПО	Способ распространения
1	Microsoft Office 2010 Pro	лицензионное
2	Windows 10	лицензионное
3	Kaspersky Endpoint Security	лицензионное

- для практических занятий:

Наименование ПО	Способ распространения
Windows 10	лицензионное
Microsoft Office 2010 Pro	лицензионное
Mozilla Firefox	свободно распространяемое
Kaspersky Endpoint Security	лицензионное
Microsoft SQL Server 2008	лицензионное
Microsoft Visual Professional 2019	лицензионное

8 Обеспечение образовательного процесса для лиц с ограниченными возможностями здоровья и инвалидов

В ходе реализации дисциплины используются следующие дополнительные методы обучения, текущего контроля успеваемости и промежуточной аттестации обучающихся в зависимости от их индивидуальных особенностей:

- для слепых и слабовидящих:

- лекции оформляются в виде электронного документа, доступного с помощью компьютера со специализированным программным обеспечением;

- письменные задания выполняются на компьютере со специализированным программным обеспечением, или могут быть заменены устным ответом;

- обеспечивается индивидуальное равномерное освещение не менее 300 люкс;

- для выполнения задания при необходимости предоставляется увеличивающее устройство; возможно также использование собственных увеличивающих устройств;

- письменные задания оформляются увеличенным шрифтом;

- экзамен и зачёт проводятся в устной форме или выполняются в письменной форме на компьютере.

- для глухих и слабослышащих:

- лекции оформляются в виде электронного документа, либо предоставляется звукоусиливающая аппаратура индивидуального пользования;

- письменные задания выполняются на компьютере в письменной форме;

- экзамен и зачёт проводятся в письменной форме на компьютере; возможно проведение в форме тестирования.

- для лиц с нарушениями опорно-двигательного аппарата:

- лекции оформляются в виде электронного документа, доступного с помощью компьютера со специализированным программным обеспечением;

- письменные задания выполняются на компьютере со специализированным программным обеспечением;

- экзамен и зачёт проводятся в устной форме или выполняются в письменной форме на компьютере.

При необходимости предусматривается увеличение времени для подготовки ответа.

Процедура проведения промежуточной аттестации для обучающихся устанавливается с учётом их индивидуальных психофизических особенностей. Промежуточная аттестация может проводиться в несколько этапов.

При проведении процедуры оценивания результатов обучения предусматривается использование технических средств, необходимых в связи с индивидуальными особенностями обучающихся. Эти средства могут быть предоставлены университетом, или могут использоваться собственные технические средства.

Проведение процедуры оценивания результатов обучения допускается с использованием дистанционных образовательных технологий.

Обеспечивается доступ к информационным и библиографическим ресурсам в сети Интернет для каждого обучающегося в формах, адаптированных к ограничениям их здоровья и восприятия информации:

- для слепых и слабовидящих:
- в печатной форме увеличенным шрифтом;
- в форме электронного документа;
- в форме аудиофайла.
- для глухих и слабослышащих:
- в печатной форме;
- в форме электронного документа.
- для обучающихся с нарушениями опорно-двигательного аппарата:
- в печатной форме;
- в форме электронного документа;
- в форме аудиофайла.

Учебные аудитории для всех видов контактной и самостоятельной работы, научная библиотека и иные помещения для обучения оснащены специальным оборудованием и учебными местами с техническими средствами обучения:

- для слепых и слабовидящих:
  - устройством для сканирования и чтения с камерой SARA CE;
  - дисплеем Брайля PAC Mate 20;
  - принтером Брайля EmBraille ViewPlus;
- для глухих и слабослышащих:
  - автоматизированным рабочим местом для людей с нарушением слуха и слабослышащих;
  - акустический усилитель и колонки;
- для обучающихся с нарушениями опорно-двигательного аппарата:
  - передвижными, регулируемыми эргономическими партами СИ-1;
  - компьютерной техникой со специальным программным обеспечением.

9 Методические материалы

9.1 Планы практических работ

Тема 1. Понятия алгоритма и структур данных

Практическое занятие № 1. МЕТОДЫ ХРАНЕНИЯ И ОБРАБОТКИ РАЗРЯЖЁННЫХ МАТРИЦ

Цель занятия: изучить предложенные методы хранения и обработки разреженных матриц; получить навыки использования изученных методов при решении практических задач; на основе навыков полученных в курсе «Программирование на языках высокого уровня» по работе с массивами, разработать программу с использованием одного из методов; на основе численных экспериментов сделать выводы об эффективности метода.

Методические указания к практическому занятию

Математические модели многих практических задач приводят к системе линейных алгебраических уравнений с большими и разреженными матрицами коэффициентов. Типичный пример - решение уравнений с частными производными методами конечных разностей или конечных элементов; аналитико-численные методы решения задач теории массового обслуживания /1/.

Целью изложенных ниже методов является решение основ- ной проблемы: хранение в памяти ЭВМ при решении практических задач, только значащих элементов обрабатываемых матриц. Методы представления и обработки разреженных матриц зависят от типа матрицы. Можно выделить три основных типа разреженных матриц: треугольные, квазидиагональные и разреженные матрицы произвольной структуры. В зависимости от типа матрицы ниже рассмотрены методы их представления в памяти, основанные на различных подходах представления двумерного массива с малым количеством значащих элементов в форме другого массива (или группы массивов) содержащего только значащие элементы. Основной проблемой использования данных методов является не преобразования собственно массивов, а преобразования, которые необходимо произвести с известными алгоритмами обработки матриц для работы с матрицами в новом представлении, и оценке их эффективности с точки зрения выполняемых операций. Естественно, что существует проблема компромиссного выбора: «что выгоднее?» - экономия памяти или производительность выполняемых операций.

Треугольные матрицы

Алгоритм. Пусть задана верхняя треугольная матрица A, т.е. матрица в которой $a_{ij} = 0$ при $j < i$ (рис.1.1).

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} & \ldots\ldots & a_{1n} \\ 0 & a_{22} & a_{23} & \ldots\ldots & a_{2n} \\ 0 & 0 & a_{33} & \ldots\ldots & a_{3n} \\ \ldots\ldots & & & & \\ 0 & 0 & 0 & \ldots\ldots & a_{nn} \end{vmatrix}$$

Рис 1.1. Верхняя треугольная матрица *A*

При обработке матрицы A можно хранить только элементы с $j >= i$, расположенные выше главной диагонали и на ней. Для решения этой задачи матрица $A(n,n)$ (квадратная) может храниться в памяти как вектор (одномерный массив) L размерностью m ($L(m)$), где $\boldsymbol{m = n(n+1)/2}$, при этом возникает следующее соответствие:

$$\begin{array}{ccccccc} L_1 & & L_2 & & L_3 & L_4 \ldots . & L_m \\ | & & | & & | & | & | \\ a_{11} & & a_{12} & & a_{22} & a_{13} \ldots . & a_{nn} \end{array}$$

Каждый элемент вектора (массива) $\boldsymbol{L(k)}$ может быть определен через индексы элемента матрицы $\boldsymbol{A}$ следующими образом**:**

$$\boldsymbol{k = j*(j-1)/2+i.}$$

Квазидиагональные матрицы

Алгоритм 1. Пусть задана квазидиагональная матрица (напри- мер, состоящая из трех диагоналей), т.е. матрица, в которой отличны от нуля только те элементы $\boldsymbol{a_{ij}}$, для которых $\boldsymbol{i = j}$ или $\boldsymbol{j = i + 1}$, $\boldsymbol{j = i - 1}$ (рис.1.2).

$$\begin{vmatrix} a_{11} & a_{12} & 0 & 0 & 0 & \dots \\ a_{21} & a_{22} & a_{23} & 0 & 0 & \dots \\ 0 & a_{32} & a_{33} & a_{34} & 0 & \dots \\ 0 & 0 & a_{43} & a_{44} & a_{45} & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & a_{n,n-1} & a_{n,n} \end{vmatrix}$$

Рис 1.2. Квазидиагональная матрица *A*

Квазидиагональная матрица ***A*(*n*,*n*),** может храниться как век- тор ***R*(*m*),** где $m = 3 \cdot n - 2$, например:

$$\begin{array}{cccccc} R_1 & R_2 & R_3 & R_4 & R_5 & \dots R_m \\ | & | & | & | & | & | \\ a_{11} & a_{21} & a_{12} & a_{22} & a_{32} & \dots a_{nn} \end{array}$$

Между элементами матрицы ***A*(*i*,*j*)** и элементами вектора ***R*(*k*)**

устанавливается следующая взаимосвязь: $k = 2 \cdot j + i - 2$.

Алгоритм 2. В качестве способа хранения квазидиагональной матрицы можно использовать ее представление в виде набора векторов (для каждой диагонали). Например, для матрицы (рис. 1.2)

$$\begin{array}{lccccc} b_1 & [a_{12} & a_{23} & a_{34} & \dots\dots & a_{n-1,n}] \\ b_2 & [a_{11} & a_{22} & a_{33} & \dots\dots & a_{n,n}\ ] \\ b_3 & [a_{21} & a_{32} & a_{43} & \dots\dots & a_{n,n-1}] \end{array}$$

Разреженные матрицы произвольной структуры

Алгоритм 1. Любая матрица, в том числе и разреженная матрица произвольной структуры, может быть, представлена в виде графа (рис. 1.3).

$$A = \begin{vmatrix} 4 & 3 & 0 & 0 & 0 \\ 2 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 8 & 0 & 0 & -2 & 0 \\ 7 & 0 & 1 & 3 & 2 \end{vmatrix}$$

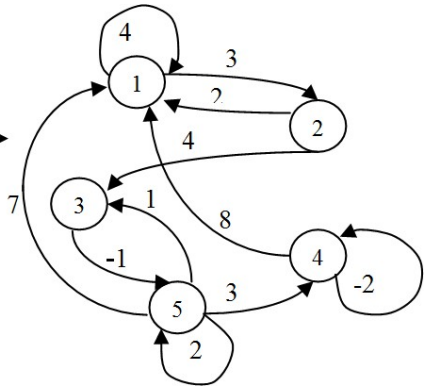


Рис. 1.3. Граф матрицы произвольной структуры

Полученный граф задается матрицей ***Q***, в которой два первых элемента любого столбца определяют некоторую дугу, третий элемент представляет вес, приданный этой дуге (т.е. столбец со- держит индексы и значение ненулевых элементов матрицы ***A***):

$$Q = \begin{vmatrix} 1 & 1 & 2 & 2 & 3 & 4 & 4 & 5 & 5 & 5 & 5 \\ 1 & 2 & 1 & 3 & 5 & 1 & 4 & 1 & 3 & 4 & 5 \\ 4 & 3 & 2 & 4 & -1 & 8 & -2 & 7 & 1 & 3 & 2 \end{vmatrix}$$

При таком способе хранения матрицы ***A*(*n*,*m*)** экономия в использовании памяти достигается при выполнении условия $3k < n \cdot m$, где ***k*-** число ненулевых элементов в матрице ***A***. Однако представление разреженной матрицы в виде матрицы ***Q*** неудобно при выполнении операции умножения (т.к. элементы, принадлежащие одному столбцу в матрице ***A*,** разбросаны по матрице ***Q***). Алгоритмы, связанные с обработкой матриц, представленных подобным образом, являются достаточно громоздкими и мало эффективными.

Алгоритм 2. Рассмотрим ориентированный граф, в котором существует вершина для каждого ненулевого элемента разреженной матрицы и упорядоченная тройка {*номер строки, номер столбца, значение*} присоединена к каждой вершине как метка. Все ненулевые элементы, принадлежащие к одному столбцу или строке, соединены дугами (рис.1.4).

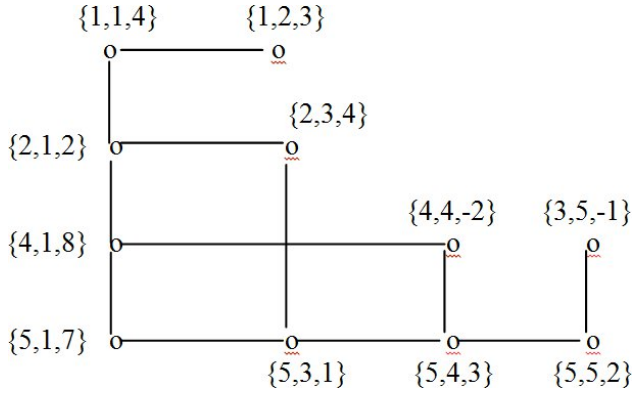


Рис. 1.4. Ориентированный граф

Подобный граф можно представить с помощью матрицы **QQ,** позволяющей более эффективно выполнять вычислительные операции над матрицами. Данная матрица отличается от матрицы **Q** добавлением четвертой строки, реализующей указатели на элементы, находящиеся в одном столбце:

$$QQ = \begin{vmatrix} 1 & 1 & 2 & 2 & 3 & 4 & 4 & 5 & 5 & 5 & 5 \\ 1 & 2 & 1 & 3 & 5 & 1 & 4 & 1 & 3 & 4 & 5 \\ 4 & 3 & 2 & 4 & -1 & 8 & -2 & 7 & 1 & 3 & 2 \\ 3 & 0 & 6 & 9 & 11 & 8 & 10 & 0 & 0 & 0 & 0 \end{vmatrix}$$

Нулевой элемент в четвертой строке матрицы ***QQ*** говорит о том, что находящийся в данном столбце матрицы ***QQ*** значащий элемент является последним значащим элементом в соответствующем столбце матрицы ***A***.

Варианты заданий к практическому занятию

В качестве вариантов заданий предлагается разработать алгоритмы выполнения стандартных матричных операций для заданного типа разреженных матриц и способа их представления в памяти. Основные матричные операции: транспонирование матрицы; сложение двух матриц; умножение двух матриц.

1. Непосредственная реализация программ требует предварительного ознакомления с методами хранения и обработки разреженных матриц и составления алгоритма функционирования программы.

2. Разработка и отладка программ по заданию преподавателя, в которых используется обычный и один из предложенных методов обработки и хранения разреженных матриц, обязательным является разработка программы на языке Pascal (по желанию студента программа может быть реализована средствами языка Си).

3. Проведение численных экспериментов по обработке матриц различной размерности с использованием обычного и специальных методов.

4. Формулировка результатов экспериментов, выводов об эффективности методов, оформление отчета.

Требования к программной реализации

1. Первая программа должна реализовывать названные операции для обычного представления разреженных матриц;

2. Вторая программа реализует те же операции, но с матрицами, представленными в сжатом виде, при этом должна иметься возможность проверить правильность получаемых результатов (т.е. должна быть предусмотрена возможность преобразования полученных результатов в обычный формат представления);

3. В обеих программах должны быть предусмотрены средства оценки времени выполнения операций, для результирующей оценки эффективности алгоритмов;

4. При реализации алгоритмов могут быть использованы в качестве базовых типов не только массивы и их свойства, но и например возможности работы с элементами массивов через указатели, или специальные структурированные типы данных;

5. Программная реализация алгоритмов должна позволять осуществлять численные эксперименты с матрицами различной размерности.

Тема 2. Анализ алгоритмов

Практическое занятие № 2.

СТРУКТУРИРОВАННЫЕ ТИПЫ ДАННЫХ. СПОСОБЫ ОБРАБОТКИ И ПРИНЦИПЫ ИСПОЛЬЗОВАНИЯ

Цель занятия: изучить и освоить методы использования и обработки структурированных типов данных в языках программирования; получить навыки использования структурированных типов данных (запись, множество, объединение, файл) при решении практических задач; разработать серию программ реализующих базовую задачу с использованием различных сочетаний структурированных типов данных в языках программирования C, C++, Pascal; на основе численных экспериментов сделать выводы об эффективности использования структурированных типов данных и принципах информационного описания предметных областей.

Методические указания к практическому занятию

Содержание прикладных задач предназначенных в качестве заданий к практическому занятию ориентированно на последовательное выполнение следующих операций:

- ввод данных сложной структурированной формы и создание на их основе последовательного файла или файлов;
- обработка данных хранящихся в файлах в соответствии с заданным алгоритмом обработки и создание результирующего файла данных;
- алгоритм обработки должен быть реализован двумя способами: с использованием в качестве базовой структуры типа данных - запись; с использованием типа данных множество;
- исходные файлы могут оставаться общими для обоих вариантов программы;
- алгоритмы обработки, которые, по существу, представляют собой выборки данных по заданному критерию, в случае использования типа данных “запись” используются характерные конструкции “ *IF _ THEN* “, в случае использования типа данных “множество”, должны использоваться основные операции над множествами: пересечение, объединение, принадлежность.

На рисунке 2.1 изображена технологическая схема обработки данных, которая должна быть реализована в обеих вариантах реализации программы.

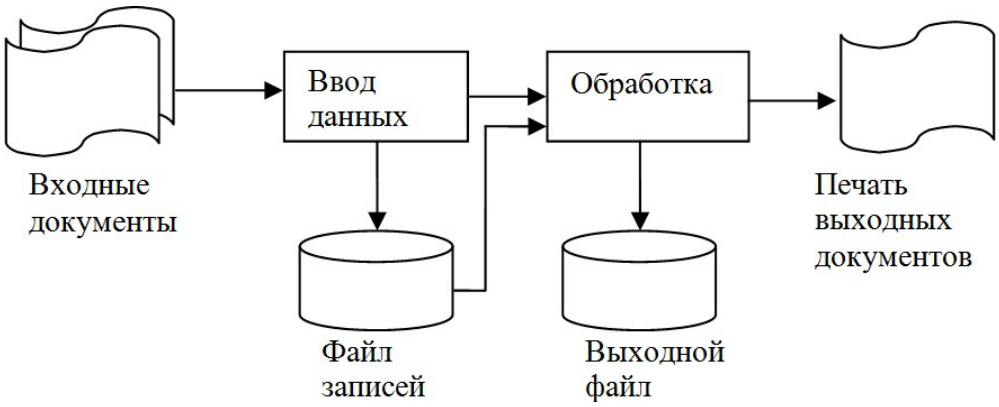


Рис. 2.1. Технологическая схема обработки данных

На рисунке 2.2 отражен основной принцип формирования критериев обработки входных данных в случаях использования различной типизации данных и используемых операций обработки, характерных для записей и множеств. Критерий выборки данных, для случая, использующего тип данных “множество”, формируется как некоторое “эталонное” множество, с которым сравниваются порождаемые на основе входных данных “рабочие” множества аргументов.

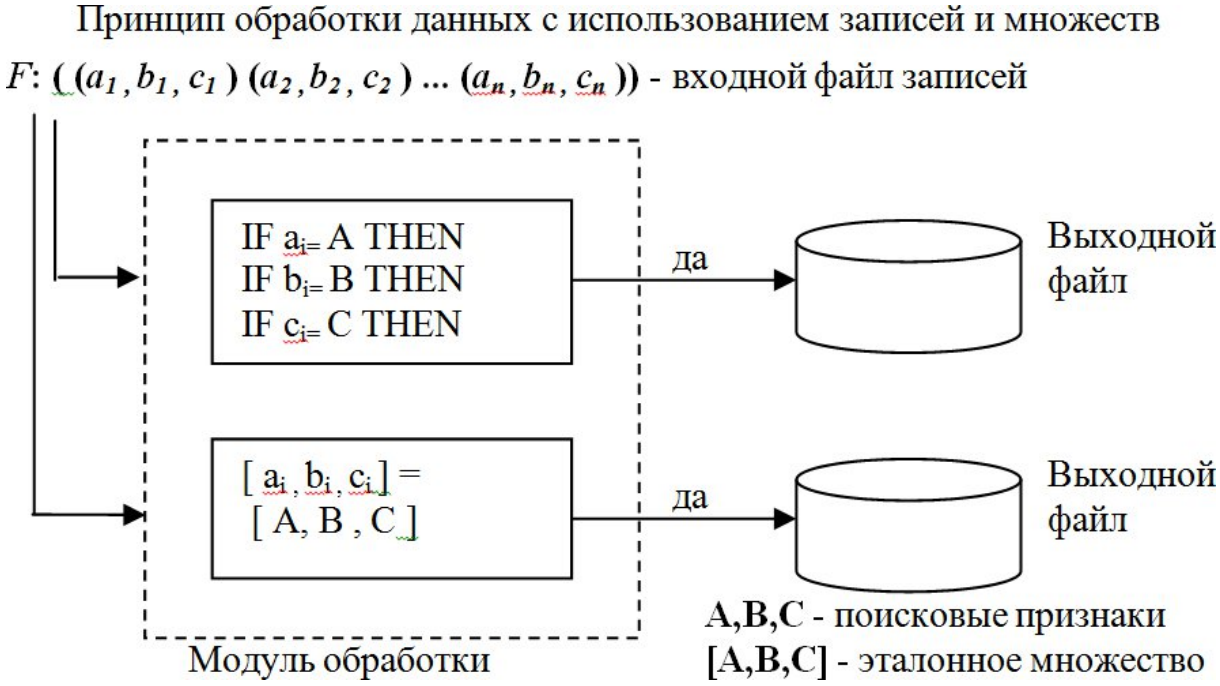


Рис. 2.2. Основной принцип формирования критериев обработки входных данных

При реализации алгоритмов обработки могут быть использованы типы данных: запись с вариантами в языке Pascal; объединение в языке С; аналог типа данных “множество” в языке С - битовая строка.

Варианты заданий к практическому занятию

Варианты заданий определяются заданной формой входных документов или структурой обрабатываемой информации, а также алгоритмом ее обработки и формой представления результатов.

Требования к программной реализации

1. Во всех вариантах программной реализации задачи должен быть реализован режим ведения исходных данных во входных файлах (ввод, корректировка, удаление);
2. Программная реализация алгоритма обработки основанного на использовании типа данных “запись” (структура) осуществляется на языках Pascal и C;
3. Программная реализация алгоритма обработки основанного на использовании типа данных “множество” осуществляется на языке Pascal.

Тема 3. Базовые алгоритмы решений задач
Практическое занятие № 3.
МЕТОДЫ СОРТИРОВКИ ДАННЫХ. ИЗУЧЕНИЕ И АНАЛИЗ

Цель занятия: изучение методов сортировки данных (сортировка массивов, файлов), получение практических навыков по использованию методов сортировки данных при решении прикладных задач, исследование характеристик эффективности алгоритмов и способов их реализации.

Методические указания к практическому занятию

Изучение методов сортировки данных в курсе “Алгоритмы и структуры данных” обусловлено необходимостью освоения двух основных навыков в алгоритмизации и программировании задач обработки данных:

практическое уяснение зависимости алгоритмов обработки от основных типов данных, используемых в программе;

оценка эффективности алгоритмов на примере алгоритмов сортировки.

В общем случае под сортировкой понимается перестановка элементов $A_1, A_2, A_3, \ldots A_n$, в таком порядке $A\ k_1, A\ k_2, A\ k_3, \ldots A\ k_n$, что при заданной функции упорядочения **F** справедливо отношение:

$$F(A\ k_1) \boxtimes F(A\ k_2) \boxtimes F(A\ k_3) \boxtimes \ldots \boxtimes F(A\ k_n).$$

Обычно функция упорядочения не вычисляется по какому-то специальному правилу, а содержится в каждом сортируемом элементе в виде явной компоненты (ключа).
Алгоритмы сортировки могут быть классифицированы следующим образом (рис. 3.1).

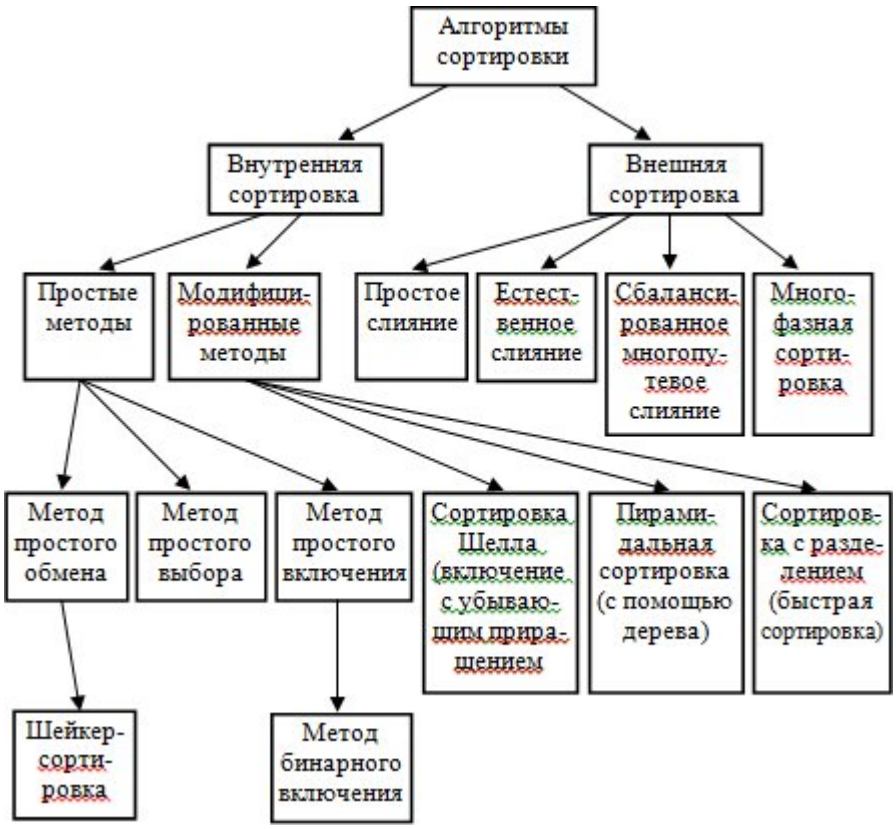


Рис. 3.1. Классификация методов сортировки

Разделение алгоритмов сортировки на внешние и внутренние, обусловлено разницей свойств типов данных, сортируемых последовательностей.
Методы внутренней сортировки ориентированы на сортировку массивов. При этом используются свойства типа данных “массив”, позволяющие работать со всеми его элементами, находящимися в оперативной памяти.
Разделение методов внутренней сортировки на сложные (модифицированные) и простые, связано с оценкой эффективности алгоритмов по двум основным параметрам:

- число сравнений ключей элементов (***C***);
- число перестановок элементов (***M***).

Для простых методов сортировки эти параметры имеют в среднем порядок n^2, а для сложных (модифицированных) - $n \cdot \log n$
, где ***n***- количество сортируемых ключей /2, 6/.

Методы внешней сортировки ориентированы на сортировку файлов и предполагают, что доступ к элементам файла осуществляется последовательно. Основой всех алгоритмов внешней сортировки является обязательное выполнение следующих фаз обработки входного файла:

- фаза разбиения входной последовательности;
- фаза слияния серий элементов.

Разнообразие методов внешней сортировки обусловлено различием в способах реализации этих фаз. Базовые алгоритмы внешней сортировки могут быть модифицированы в части формирования начальных серий (отрезков) элементов. Для этой цели вместо естественного или простого формирования начальных отрезков используются специальные процедуры их формирования (рис. 3.2). Для реализации данных процедур могут быть использованы алгоритмы внутренней сортировки /2,6,7/.

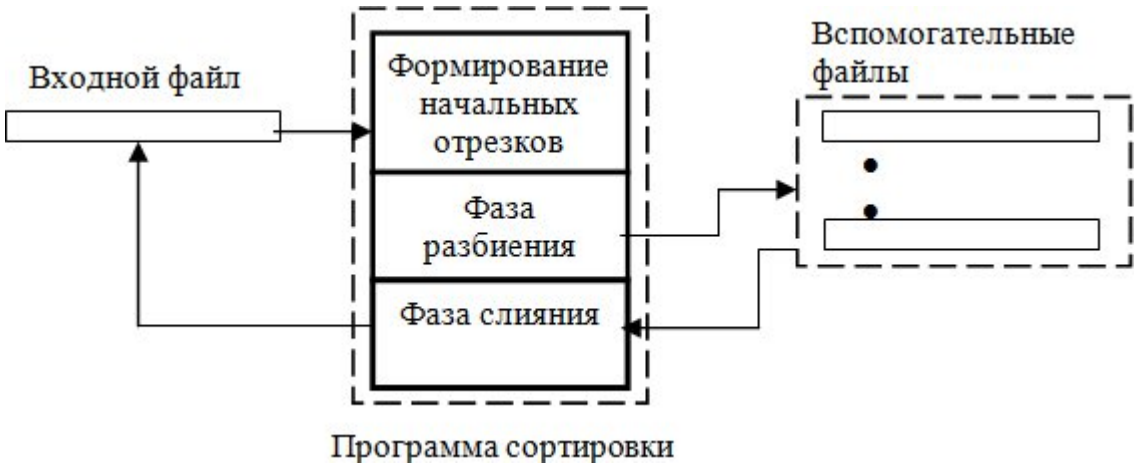


Рис. 3.2. Основные компоненты программы сортировки файлов

Варианты заданий к практическому занятию

Вариант задания формируется на основе задания сочетаний используемых алгоритмов внутренней и внешней сортировки с оговоренными ограничениями на используемый метод. Алгоритмы внутренней сортировки используются для формирования начальных отрезков первой реализации фазы разбиения, при этом предполагается, что длина отрезка является задаваемой величиной. В качестве регулируемого параметра для алгоритмов внешней сортировки, используемых в программе, выступает количество файлов задействованных в алгоритме. Этап анализа базового алгоритма предполагает выполнение численных экспериментов

по оценке эффективности и корректности алгоритма с точки зрения использования различных (по упорядоченности) входных последовательностей и раз-личных значений изменяемых параметров алгоритма. Варианты сочетаний алгоритмов внутренней и внешней сортировки представлены в таблице 3.1.

В качестве вариантов построения входных последовательностей могут быть использованы:

- последовательность, упорядоченная в порядке обратном от требуемого;
- упорядоченная последовательность, за исключением одного или нескольких элементов;
- полностью упорядоченная последовательность;
- случайная последовательность.

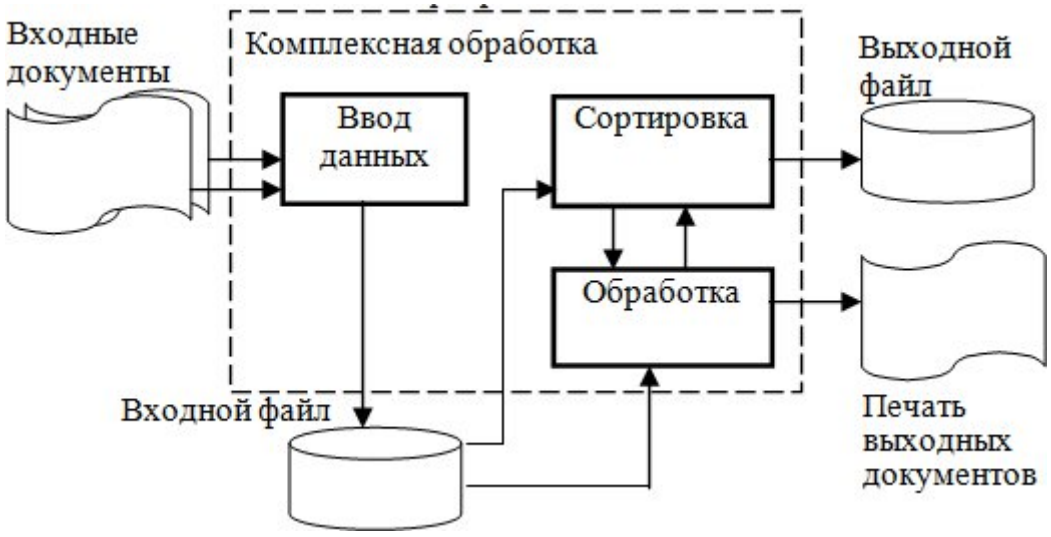
В приложении 1 приведены тексты программ на языке С, реализующих ряд алгоритмов внутренней и внешней сортировки, которые могут быть использованы для формирования базового комплексного алгоритма сортировки.

Таблица 3.1 **Варианты сочетаний алгоритмов внешней и внутренней сортировки**

Номер	**Алгоритм подготовки начальных отрезков**	**Алгоритм внешней сортировки**
1.	Метод простого включения	Сбалансированное многопутевое слияние
2.	Метод бинарного включения	Сбалансированное многопутевое слияние
3.	Пирамидальная сортировка	Каскадная сортировка
4.	Шейкер - сортировка	Естественное слияние
5.	Сортировка с разделением	Многофазная сортировка
6.	Сортировка Шелла	Естественное слияние
7.	Метод простого включения	Многофазная сортировка
8.	Метод простого выбора	Сбалансированное многопутевое слияние
9.	Простое слияние (для массива)	Каскадная сортировка
10.	Сортировка с разделением	Естественное слияние

После проведения численных экспериментов на модельных последовательностях, программа должна быть модифицирована для выполнения функций сортировки входных последовательностей (или результирующих файлов) сформированных в лабораторной работе № 2 (рис. 3.3).

Рис. 3.3. Структура комплексной программы обработки с использованием сортировки данных



Требования к программной реализации

В процессе выполнения практического задания рекомендуется следующий порядок разработки и исследования программ:

- поэтапная разработка и отладка процедур формирования начальных отрезков и собственно программы внешней сортировки;
- интеграция и совместная отладка комплексной программы сортировки;
- разработка программы формирования модельных последовательностей различной упорядоченности;
- проведение численных экспериментов по оценке эффективности разработанного алгоритма с учетом варьирования пара- метров: длина начального отрезка; количество вспомогательных файлов;
- модификация и отладка программы предназначенной для сортировки входных последовательностей лабораторной работы № 1;
- в качестве языков программирования используются языки C и Pascal.

Тема 7. Динамические структуры данных

Практическое занятие №4.

ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ. ИЗУЧЕНИЕ, АНАЛИЗ И ИСПОЛЬЗОВАНИЕ

Цель занятия: изучение возможностей языков программирования C, Pascal по работе с указателями; изучение принципов создания и обработки динамических структур данных; получение практических навыков использования динамических структур данных для решения прикладных задач.

Методические указания к практическому занятию

В практическом программировании существует достаточно большое количество задач, в которых необходимо использование данных, для которых характерно, что в процессе вычисления изменяется не только значение переменных, но и их структура. Такие переменные называются данными с динамической структурой.

Свойства динамических структур данных позволяют достаточно просто реализовывать функции эффективного и многокритериального поиска. Можно выделить следующий перечень динамических объектов, которые могут быть использованы для организации поиска и обработки данных сложной структуры:

- упорядоченные линейные и двунаправленные списки;
- многосвязные списки;
- бинарные деревья поиска;
- сильно ветвящиеся деревья поиска.

Для использования каждого из перечисленных объектов необходимо реализовать набор функций, позволяющих обрабатывать элементы объектов в соответствии с заданным алгоритмом:

- поиск заданного элемента;
- включение нового элемента;
- удаление заданного элемента;
- модификация элемента.

Упорядочение элементов в динамической структуре осуществляется в соответствии с заданной функцией упорядочения для ключевого поля структурированного типа.

Каждый из перечисленных динамических объектов обладает соответствующими характеристиками эффективности обработки. Основными характеристиками являются:

~ средняя длина пути поиска элемента в объекте;

затраты (количество операций), связанные с ведением объекта.

~ Основой для построения динамического объекта сложной структуры служат элементы структурированного типа, компонентами которых являются ключевые и информационные компоненты, а также элементы типа «указатель».

На рис. 4.1 представлен пример построения динамических объектов на основе линейного и многосвязного списка. Использование многосвязных списков позволяет реализовывать функции поиска для нескольких атрибутов.

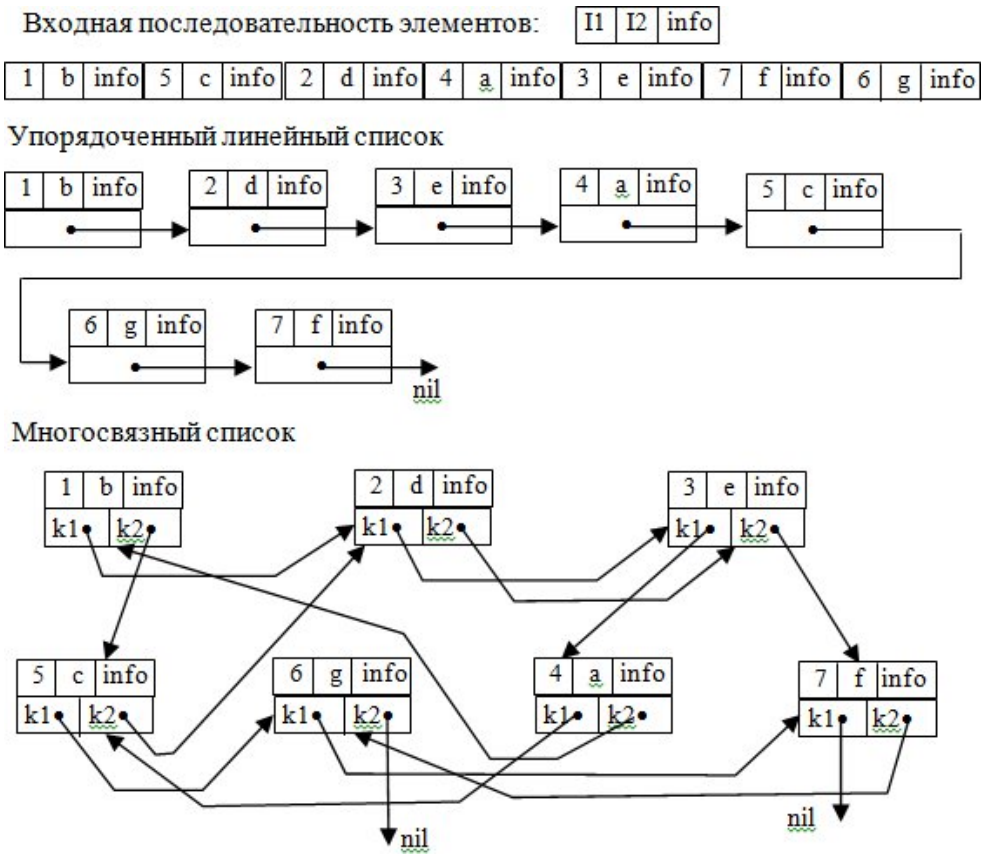


Рис. 4.1. Принципы формирования линейного и многосвязного списков

Более эффективным динамическим объектом для реализации функций поиска является дерево поиска. Классификация деревьев поиска представлена на рис. 4.2.

Основной проблемой использования деревьев поиска является решение задачи выбора между эффективностью поиска и затратами на балансировку дерева. Затраты на балансировку дерева сильно зависят от характера входной последовательности.

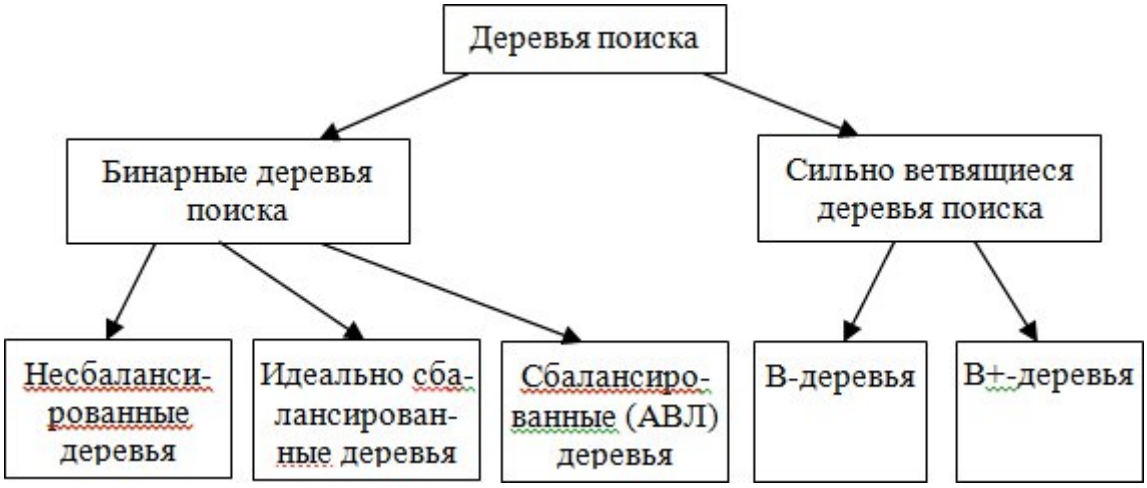


Рис. 4.2. Классификация деревьев поиска

На рис. 4.3 представлен пример построения сбалансированного и несбалансированного деревьев поиска для одной входной последовательности. Наиболее эффективными деревьями поиска по совокупному критерию «время доступа + затраты на ведение» являются АВЛ-деревья и В-деревья /2/. Отличительной особенностью сильно ветвящихся деревьев является организация узлов де- рева как списка значений ключей и списка указателей, что уменьшает необходимость балансировок /2,6,7/.

Для повышения эффективности обработки данных, хранящихся во внешней памяти, динамические объекты могут быть использованы в соответствии содержит следующие данные: схемой обработки, представленной на рис. 4.4. Представленная схема позволяет повысить эффективность поиска и обработки данных за счет использования свойств динамических объектов.

Варианты заданий к практическому занятию

Вариант задания определяется типом и особенностью использования динамического объекта, заданного для реализации функций обработки и информационных запросов. Варианты заданий представлены в табл. 4.1.

Выполнение задания разбивается на три этапа:

1. Разработка тестовой программы, реализующей основные функции обработки, заданного динамического объекта;
2. Проведение численных экспериментов на модельных данных для оценки эффективности и адекватности реализованного алгоритма;
3. Интеграция разработанного модуля (и его модификация) с программой и исходными данными лабораторных работ №№2,3.

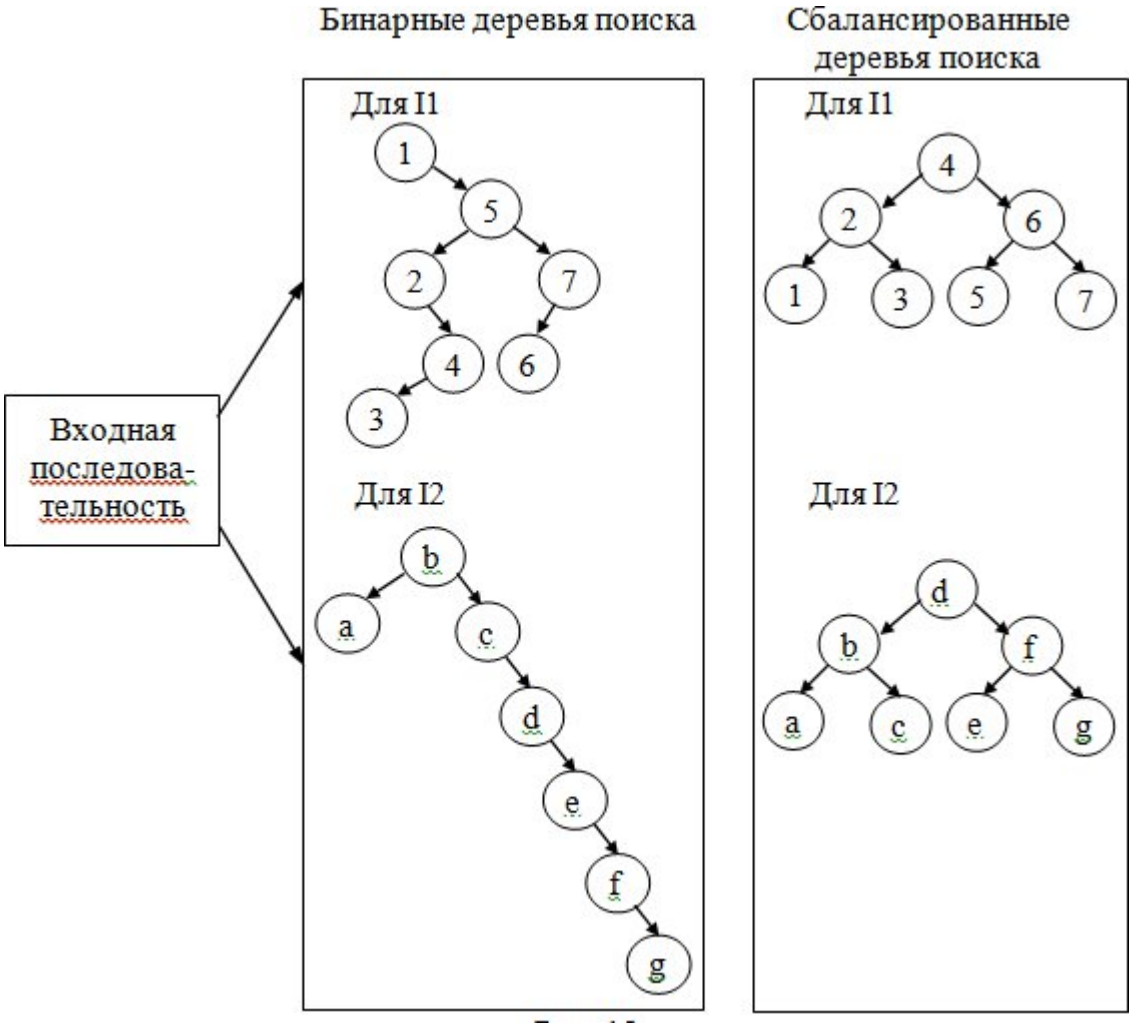


Рис. 4.3. Принцип формирования деревьев поиска

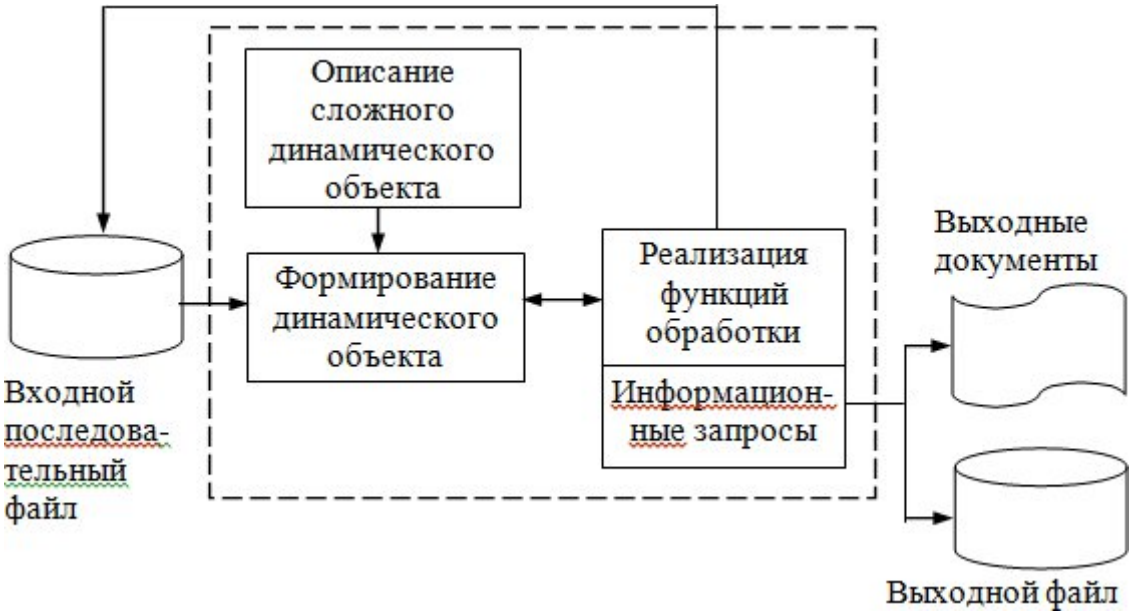


Рис. 4.4. Схема обработки данных с использованием динамических объектов

В приложении 2 приведены тексты программ на языке С, иллюстрирующих способы описания и методы обработки некоторых динамических объектов.

Требования к программной реализации

В процессе выполнения работы рекомендуется следующий порядок разработки и исследования программ:

- поэтапная разработка процедур обработки динамического объекта;
- создание модельных последовательностей для проведения численных экспериментов по оценке эффективности;
- проведение численных экспериментов для определения оценок времени поиска и затрат на модификацию;
- модификация и интеграция с программой, разработанной при выполнении лабораторных работ №№ 1,2;
- в качестве языка программирования используется С, С++.

Таблица 4.1 **Тип и особенности использования динамического объекта**

№	Тип динамического объекта	Дополнительные требования
1.	Линейный список	Использование нескольких списков для входной группы файлов
2.	Двунаправленный линейный список	Использование нескольких списков для входной группы файлов
3.	Многосвязный список	Для различных атрибутов поиска одного входного файла
4.	Многосвязный список	Для атрибутов поиска нескольких входных файлов
5.	Бинарное дерево поиска (без балансировки)	Для группы атрибутов одного входного файла
6.	Сбалансированное дерево поиска (АВЛ-дерево)	Для группы атрибутов одного входного файла
7.	B-дерево	С заданным порядком дерева
8.	B-дерево	С произвольным порядком (возможность настройки)
9.	B+-дерево	Для заданного атрибута входного файла

9.2 Методические рекомендации по подготовке письменных работ

Письменными работами по данной дисциплине являются отчеты о практических работах, которые обучающиеся выполняют и оформляют в соответствии с требованиями.

В среде приложения MS Office Word набирается текст по следующему шаблону. По центру пишется «Отчет о практической работе №__ «Наименование Практической работы» (в соответствии с наименованием, указанным в методических указаниях), указывается фамилия, имя и отчество студента, а также шифр его группы.

Затем в отчете указывается наименование раздела «1. Понятия алгоритма и структур данных» и под наименованием раздела формулируется цель соответствующей работы. Точно также выполняется и оформляется раздел «2. Методика работы». Разделы «3.1-3.4» выполняются и оформляются в отчете в соответствии с составом и содержанием выполненной практической работы. В тексте соответствующего раздела, например, см. раздел 3.3, необходимо поместить скриншоты результатов практической работы, например, условие задачи, исходные данные, листинг программ, таблиц и др. Скриншоты снабжаются наименованиями, например, «Рис.1. Алгоритм решения задачи», пишется наименование под скриншотом (рисунком). При наличии в отчете таблицы над ней пишется наименование, например, «Таблица 1. Дефекты программы». Шрифт текста отчета Times New Roman, размер шрифта – 14, абзацный отступ - 1,25 см., выравнивание «по ширине», интервал между строк – 1,5. Наименование разделов выделять жирным шрифтом. Данный раздел 3.5. в отчете не оформляется. Он рассматривается как методические рекомендации для данной и всех последующих практических работ.

В конце каждого отчета оформляется раздел «Выводы». По каждому из этапов приводятся краткие выводы (резюме) по методике, средствам, ресурсам, которые можно использовать в решении рассматриваемой задачи.

Приложение 1

АННОТАЦИЯ ДИСЦИПЛИНЫ

Цель дисциплины: формирование у обучающихся знаний основных принципов проектирования и анализа алгоритмов и структур данных, знаний основных типов алгоритмов, применяемых в современном программировании для обработки соответствующих структур данных, а также умений обоснования корректности алгоритмов, их практической реализации, теоретической и экспериментальной оценки их временной сложности, развитие необходимых практических навыков их применения в будущей профессиональной деятельности.

Задачи дисциплины:

- ознакомление с разнообразием структур данных и их реализациями в проектировании алгоритмов;
- изучение основных операций над структурами данных в современном программировании;
- овладение структурным подходом к разработке алгоритмов;
- формирование и развитие у обучаемых конкретных практических умений и навыков проектирования и анализа алгоритмов, и структур данных.

Дисциплина направлена на формирование следующих компетенций:

ОПК-6 Способен понимать принципы работы современных информационных технологий и использовать их для решения задач профессиональной деятельности

Фундаментальные структуры данных (массивы, списки, деревья, хеш-таблицы), лежащие в основе организации файловых систем, буферного пула СУБД и индексов .

Алгоритмы управления памятью (страничная организация, алгоритмы вытеснения страниц: LRU, FIFO), используемые при администрировании ОС и настройке СУБД .

Алгоритмы поиска и сортировки, определяющие производительность выполнения SQL-запросов и работы компонентов ИС .

Стандарты и протоколы информационного взаимодействия (например, T-SQL, PL/SQL) с точки зрения алгоритмов обработки запросов на стороне сервера .

Принципы организации транзакций и журналирования (WAL — Write-Ahead Logging) как алгоритмической основы восстановления данных после сбоев .

Уметь:

Анализировать планы выполнения запросов (Execution Plans) в СУБД и определять, какие алгоритмы (Nested Loops, Hash Join, Merge Join) используются, настраивать параметры для выбора оптимального алгоритма соединения таблиц .

Выбирать тип индекса (B-Tree, Hash, Bitmap) в зависимости от структуры данных и алгоритмов поиска, оптимальных для конкретного вида нагрузки (OLTP или OLAP) .

Настраивать параметры буферного кэша и алгоритмы кэширования для повышения производительности дисковой подсистемы.

Подбирать структуры данных (например, выбор между LSM Tree и B-Tree) на этапе параметрической настройки СУБД под характер входящих данных (вставка vs чтение) .

Владеть:

Навыками развертывания СУБД (инсталляции) с пониманием того, как алгоритмы сжатия данных или шифрования влияют на выбор аппаратной конфигурации (CPU, RAM)

Навыками конфигурирования хранилищ данных (RAID-массивов, файловых систем) с учетом алгоритмов чтения/записи, используемых конкретной ИС.

Навыками установки и первичной настройки компонентов, реализующих алгоритмы распределенного взаимодействия (шардирование, репликация) для обеспечения отказоустойчивости системы .

Знать:

Роль структур данных (B+ деревья, битовые карты, динамические хеш-таблицы) в методологии физического проектирования баз данных .

Алгоритмы нормализации (замыкание атрибутов, поиск ключей) как основу

логического проектирования для устранения избыточности .

Алгоритмы оптимизации запросов (эвристические и стоимостные) и их влияние на создаваемое информационное обеспечение .

Методы сжатия данных (дельта-кодирование, префиксное сжатие), применяемые при организации хранения в современных СУБД .

Уметь:

Разрабатывать схему базы данных (ER-диаграммы), выбирая оптимальные типы данных и структуры для хранения атрибутов с учетом алгоритмов последующей обработки

Применять алгоритмы декомпозиции отношений для приведения схемы базы данных к требуемым нормальным формам (3NF, BCNF) .

Создавать хранимые процедуры и триггеры, реализуя в них эффективные алгоритмы обработки данных на стороне сервера .

Проектировать индексы, основываясь на знании алгоритмов поиска и частоты выполняемых операций (SELECT/INSERT/UPDATE) .

Владеть:

Навыками реализации компонентов СУБД или структур данных для хранения (например, реализация B-дерева, хеш-таблицы или простого key-value хранилища на языке C++/Java/Python), что дает глубинное понимание работы БД .

Опытом рефакторинга (оптимизации) структуры базы данных путем применения алгоритмов денормализации для повышения производительности.

Навыками анализа и оптимизации "медленных" запросов (slow queries) через изменение алгоритмов доступа к данным (замена вложенных запросов на соединения, использование оконных функций) .

Опытом ведения базы данных, включая мониторинг фрагментации индексов и применение алгоритмов их реорганизации/перестроения.

Общая трудоемкость освоения дисциплины составляет 3 зачетные единицы.