|  |  |
| --- | --- |
| Контролируемые компетенции | Планируемый результат обучения |
| ОПК-6 - Способен разрабатывать алгоритмы и программы, пригодные для практического использования, применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов;  *ОПК-6.1* *-* *Знать:* *алгоритмические* *языки* *программирования,* *операционные* *системы* *и* *оболочки,* *современные* *среды* *разработки* *программного* *обеспечения* | Знать: возможности современных языков программирования по обработке структур данных; технологию создания линейных и нелинейных структур данных; методику оценки вычислительной сложности алгоритма; стратегии разработки алгоритмов; методику реализации структуры данных средствами языка программирования; алгоритмы поиска с линейной и логарифмической сложностью; применение иерархических структур данных в алгоритмах поиска; алгоритмы сжатия данных и методы сжатия; реализацию графовых структур и алгоритмы их обработки; особенности NP полных задач и подходы к их решению; |
| ОПК-6 - Способен разрабатывать алгоритмы и программы, пригодные для практического использования, применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов;  *ОПК-6.2* *-* *Уметь:* *составлять* *алгоритмы,* *писать* *и* *отлаживать* *коды* *на* *языке* *программирования,* *тестировать* *работоспособность* *программы* *,* *интегрировать* *программные* *модули* | Уметь: используя возможности среды разработки создавать собственные структуры данных; использовать предоставляемые средой разработки контейнеры и коллекции при разработке практического и учебного программного приложения; тестировать работоспособность программы, интегрировать программные модули при решении сложных производственных и учебных задач; проводить оценку сложности алгоритмов; выбирать, для разрабатываемого приложения, оптимальную по времени и затрачиваемой памяти структуру данных; составлять алгоритмы решаемых задач, используя существующие алгоритмы обработки; применять структуру граф и его алгоритмы в программах; |
| ОПК-6 - Способен разрабатывать алгоритмы и программы, пригодные для практического использования, применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов;  *ОПК-6.3* *-* *Владеть:* *языком* *программирования;* *навыками* *отладки* *и* *тестирования* *работоспособности* *программы* | Владеть: практическими навыками создания программного обеспечения, предназначенного для обработки структур данных, средствами современных языков программирования и сред программирования при решении производственных и учебных задач; навыками отладки и тестирования работоспособности программы; выбора средств языка высокого уровня для реализации структур данных и реализации алгоритма; методикой оценки вычислительной и объемной сложности алгоритма; навыками реализации задач |

**Компетенции по дисциплине «Структуры и алгоритмы обработки данных»:**

**Структура рабочей программы по дисциплине «Структуры и алгоритмы обработки данных»:**

|  |  |
| --- | --- |
| Количество разделов: | 10 (тем 16) |

Количество семестров: **2** **семестра**

**(16 часов лекций 1 семестр, 32 часа практики 1 семестр,**

**16 часов лекций 2 семестр, 32 часа практики 2 семестр)**

Темы лекций + описание к ним **1 семестр:**

**Раздел 1. Введение в дисциплину**

*Тема 1. Основные понятия* (*ОПК-6.1)*

Предмет, цели и задачи дисциплины. *Структуры данных* Типы и структуры данных. Уровни представления данных. Классификация структур данных. Понятие абстрактного типа данных (АТД). Поэтапная разработка программы с использованием АТД. *Алгоритмы:* Задача, алгоритм, программа. Понятие алгоритма. Свойства алгоритмов.

*Тема 2. Анализ вычислительной сложности алгоритмов* (*ОПК-6.1 ОПК-6.2)*

Вычислительная сложность алгоритмов. Асимптотический метод оценки вычислительной сложности алгоритмов. Сравнение наилучших, средних и наихудших оценок. O-, o-, ω- и θ-нотации. Стандартные классы сложности. Методы эмпирической оценки сложности алгоритмов. Анализ сложности итерационных и рекурсивных алгоритмов.

**Раздел 2. Алгоритмы сортировки**

*Тема 3. Рекурсивные алгоритмы (ОПК-6.1 ОПК-6.2)*

Рекуррентные зависимости и рекурсивные алгоритмы. Применение метода «разделяй и властвуй» в рекурсивных алгоритмах. Алгоритм Хоара быстрой сортировки. Рекурсивный бинарный поиск.

*Тема 4. Простые алгоритмы сортировки (ОПК-6.1 ОПК-6.2)*

Задача сортировки. Алгоритмы внутренней сортировки. Обменные сортировки (Exchange sort), сортировка выборкой (Selection sort), сортировки вставкой (Insertion sort). Алгоритмы сортировки слиянием (Merge sort). Простое слияние Естественное слияние. Многофазная сортировка. Корректность алгоритмов сортировки. Вычислительная сложность алгоритмов сортировки слиянием

*Тема 5. Нетривиальные алгоритмы сортировки (ОПК-6.1 ОПК-6.2)*

Алгоритм турнирной сортировки. Алгоритм пирамидальной сортировки. Алгоритм Хоара быстрой сортировки (Quick sort), вероятностная версия быстрой сортировки. Блуждающая сортировка (Stooge sort). Вычислительная сложность алгоритмов сортировки

**Раздел 3. Алгоритмы поиска**

*Тема 6. Алгоритмы поиска в массивах (ОПК-6.1 ОПК-6.2)*

Задача поиска. Классификация алгоритмов поиска (алгоритмы, использующие ключи и алгоритмы, основанные на цифровых свойствах ключей). Последовательный поиск. Бинарный поиск. Фибоначчиев поиск. Интерполяционный поиск. Использование бинарного дерева поиска. Вычислительная сложность алгоритмов поиска.

*Тема 7. Алгоритмы поиска в тексте (ОПК-6.1 ОПК-6.2)*

Алгоритм последовательного поиска. Алгоритм Кнута-Морриса-Пратта. Алгоритм Бойера-Мура. Регулярные выражения. Алгоритм Рабина-Карпа. Вычислительная сложность алгоритмов поиска в тексте.

**Раздел 4. Линейные структуры данных**

*Тема 8. Линейные списки (ОПК-6.1 ОПК-6.2 ОПК-6.3)*

Линейные списки. Виды и способы реализации линейных списков (массивы, связанные списки). Линейные односвязные списки (очередь, стек). Линейные двусвязные списки (дек). Кольцевые (циклические) списки. Применение очереди и стека в алгоритмах организации вычисления выражений (Польская постфиксная и префиксные записи выражений).

Темы лекций 2 семестр

**Раздел 5. Иерархические структуры данных**

*Тема 9. Деревья (ОПК-6.1 ОПК-6.2)*

Понятие дерева. АТД n–арное дерево. Способы представления в памяти. Бинарное дерево. Алгоритмы обхода бинарного дерева. Алгоритм вывода бинарного дерева. Дерево выражений. Сортировка с использованием бинарного дерева. Оценка эффективности.

Темы лекций 2-ой семестр

**Раздел 6. Оптимизация алгоритмов поиска в структурах данных**

*Тема 10. Бинарное дерево поиска. AVL-дерево (ОПК-6.1 ОПК-6.2 ОПК-6.3)*

Бинарное дерево поиска. Алгоритмы построения бинарного дерева поиска. Сбалансированное бинарное дерево поиска. AVL-дерево (АВЛ – Адельсона-Вельского и Ландиса). Алгоритмы вставки и удаления узлов в AVL-дерево. Красно-черные деревья. Способы представления в памяти. Вычислительная сложность алгоритмов поиска с использованием бинарного дерева поиска и AVL-дерева.

*Тема 11. Хеш-таблицы и алгоритмы хеширования (ОПК-6.1 ОПК-6.2 3)*

Понятие хеш-таблицы. Организация быстрого поиска с использованием хеш-таблицы. Хеш-функции. Коллизии. Способы разрешения коллизий: цепное хеширование, открытый адрес. Понятие кластера. Алгоритм двойного хеширования. Вычислительная сложность алгоритмов поиска с использованием цепного хеширования и хеширования с открытым адресом.

**Раздел 7. Внешние структуры данных**

*Тема 12.Файлы. Типовые алгоритмы работы с файлами (ОПК-6.1)*

Файл. Текстовый файл, двоичный файл. Доступ к данным файлов. Структура файловой системы ОС. Сортировка файлов. Файловые потоки в языках объектно-ориентированного программирования. Алгоритмы поиска в файле. Использование индексных файлов. В-дерево и поиск данных в файле.

**Раздел 8. Сложные структуры данных**

*Тема 13. Графы (ОПК-6.1 ОПК-6.2)*

Основные алгоритмы работы с графами Основные понятия графа. Направленные и ненаправленные графы. Сети. Способы представления графа в памяти (списки и матрицы). Алгоритмы обхода графа в ширину и в глубину. Транзитивное замыкание орграфа. Алгоритм Дейкстры поиска кратчайшего пути в графе между двумя заданными вершинами. Алгоритм Флойда-Уоршелла поиска кратчайших путей в графе между всеми вершинами в графе. Алгоритмы Прима и Крускала построения минимального покрывающего дерева. Топологическая сортировка.

**Раздел 9. Алгоритмы сжатия данных**

*Тема 14. Алгоритмы сжатия и кодирования данных (ОПК-6.1 ОПК-6.2)*

Алгоритмы сжатия данных. Характеристики. Словарные и статистические методы сжатия данных. сжатия информации. Алгоритмы: группового кодирования RLE, Лемпеля-Зива, Шеннона-Фано. Алгоритм Хаффмана. Алгоритм сжатия, применяемый в архиваторе Zip.

**Раздел 10. Методы разработки сложных алгоритмов для решения NP-полных задач (***ОПК-6.1 ОПК-6.2)*

*Тема 15. Стратегии и методы построения алгоритмов*

Полный перебор. Метод “разделяй и властвуй”. “Жадные” алгоритмы. Метод динамического программирования. Бэктрекинг (backtracking) – перебор с возвратом. Метод ветвей и границ. Эвристический поиск. Поиск по образцу. Алгоритмы аппроксимации числовых функций.

*Тема 16.Метод линейного программирования (ОПК-6.1)*

Стандартная и каноническая формы задач линейного программирования. Преобразование задач линейного программирования в стандартную (каноническую) форму. Разрешимость задачи линейного программирования. Задача целочисленного линейного программирования (ЦЛП). Метод ветвей и границ для решения задач ЦЛП.

Практические занятия + описания к ним **1 семестр:**

Практическая работа № 1 *(ОПК-6.1 ОПК-6.2 ОПК-6.3)*

Разработка и программная реализация простых алгоритмов внутренней сортировки массивов.

Практическая работа № 2 *(ОПК-6.1 ОПК-6.2 ОПК-6.3)*

Разработка и программная реализация усовершенствованных алгоритмов внутренней сортировки массивов.

Практическая работа № 3 *(ОПК-6.1 ОПК-6.2 ОПК-6.3)*

Разработка и программная реализация алгоритмов внешней сортировки массивов.

Практическая работа № 4 *(ОПК-6.1 ОПК-6.2 ОПК-6.3)*

Разработка и программная реализация алгоритмов поиска в неотсортированных массивах.

Практическая работа № 5*(ОПК-6.1 ОПК-6.2 ОПК-6.3)*

Разработка и программная реализация алгоритмов поиска в отсортированных массивах.

Практическая работа № 6 *(ОПК-6.1 ОПК-6.2 ОПК-6.3)*

Разработка и программная реализация алгоритмов поиска строк в тексте.

Практическое задание №7 *(ОПК-6.1 ОПК-6.2 ОПК-6.3)*

Разработка и программная реализация задач с применением линейных динамических списковых структур.

Практическое задание №8 *(ОПК-6.1 ОПК-6.2 ОПК-6.3)*

Разработка и программная реализация структуры данных – стек. Разработка программ, использующих созданную структуру стек и стек коллекции.

Практические 2 семестр

Практическое задание № 9 *(ОПК-6.1 ОПК-6.2 ОПК-6.3)*

Разработка и программная реализация структуры данных – очередь. Разработка программ, использующих созданную очередь и очередь коллекции.

Практическое задание №10 *(ОПК-6.1 ОПК-6.2 ОПК-6.3)*

Разработка и программная реализация структуры данных – Бинарное дерево поиска. Разработка программ, использующих созданную структуру.

Практическое задание №11 *(ОПК-6.1 ОПК-6.2 ОПК-6.3)*

Разработка и программная реализация задач поиска данных в таблицах с применением механизма хеширования.

Практическое задание №12 *(ОПК-6.1 ОПК-6.2 ОПК-6.3)*

Разработка и программная реализация задач, использующих внешние структуры данных - файлы (текстовые и двоичные) и применение для поиска в файлах поисковых структур: хеш-таблицы, бинарные поисковые деревья. С реализацией файлов на потоковых структурах языков программирования.

Практическая работа № 13 *(ОПК-6.1 ОПК-6.2 ОПК-6.3)*

Разработка и программная реализация класса Граф: создание, обходы, поиск кратчайшего пути. Реализация задач, требующих применение графа для хранения данных. Основные алгоритмы работы с графами.

Практическая работа №14 *(ОПК-6.1 ОПК-6.2 ОПК-6.3)*

Разработка и программная реализация алгоритмов сжатия и данных

Практическая работа № 15 *(ОПК-6.1 ОПК-6.2 ОПК-6.3)*

Разработка и реализация программ с применение метода динамического программирования, метода ветвей и границ.

Практическое занятие №16 *(ОПК-6.1 ОПК-6.2 ОПК-6.3)*

Программная реализация Симплекс метода.

**Типовые контрольные вопросы и задания для проведения текущего контроля и промежуточной аттестации по итогам освоения дисциплины** (достаточно 20-30 вопросов):

Целью текущего контроля знаний является установление подробной, реальной картины студенческих достижений и успешности усвоения ими учебной программы на данный момент времени. В условиях рейтинговой системы контроля результаты текущего оценивания студента используются как показатель его текущего рейтинга.

Текущий контроль успеваемости осуществляется в течение семестра, в ходе повседневной учебной работы по индивидуальной инициативе преподавателя. Данный вид контроля стимулирует у студентов стремление к систематической самостоятельной работе по изучению дисциплины.

1. Что определяет структура данных? (*ОПК-6.1)*
2. Приведите классификацию структур данных. *(ОПК-6.1 ОПК-6.2)*
3. Какую вы будете использовать структуру для задачи поиска кратчайшего пути от одной станции московского метро до друго1? *(ОПК-6.1 ОПК-6.2)*
4. Какие способы хранения структур данных используют современные компьютеры? *(ОПК-6.1)*
5. Как можно определить структуру ОЗУ на логическом уровне? *(ОПК-6.1)*
6. Дайте определение ячейке ОЗУ, ее адресу *(ОПК-6.1)*.
7. Какие структуры называют динамическими? *(ОПК-6.1)*
8. Что определяет функция в асимптотической сложности?
9. Сформулируйте правило суммы для расчета сложности алгоритма по символике O. *(ОПК-6.1 ОПК-6.2)*
10. Сложность в худшем случае. *(ОПК-6.1)*
11. Сложность в среднем и лучшем случае. *(ОПК-6.1)*
12. Какие функции в оценке сложности называют доминирующими? *(ОПК-6.1 ОПК-6.2)*
13. Охарактеризуйте класс P – задач. *(ОПК-6.1)*
14. В чем суть парадигмы динамического программирования. *(ОПК-6.1)*
15. Как определить, что алгоритм имеет рекурсивную природу? *(ОПК-6.1)*
16. Какой алгоритм называется рекурсивным? *(ОПК-6.1)*
17. Что определяет шаг в рекурсии? *(ОПК-6.1)*
18. Что определяет глубина рекурсии? *(ОПК-6.1)*
19. В чем особенность *Линейной рекурсии? (ОПК-6.1)*
20. Как используется системный стек при выполнении рекурсивных вызовов и что он хранит? *(ОПК-6.1)*
21. Какой алгоритм называют Жадным?. Свойство жадного выбора. *(ОПК-6.1)*
22. Преобразование решения динамического программирования в жадное решение. *(ОПК-6.2)*
23. Каково быстродействие алгоритма сортировки прямого обмена по сравнению с алгоритмом простыми вставками? *(ОПК-6.1 ОПК-6.2)*
24. В чем заключаются усовершенствования алгоритма простого обмена и каковы его достоинства? *(ОПК-6.1)*
25. Какую идею реализует алгоритм сортировки слиянием? *(ОПК-6.1)*
26. Почему алгоритм быстрой сортировки Хоара называют обменной сортировки с разделением? *(ОПК-6.1)*
27. Сформируйте бинарное дерево поиска, каждая вершинами которого представляет собой структуру вида: Страна | Город | Количество жителей | Метка. Выведите содержимое дерево всеми известными Вам методами обхода. *(ОПК-6.1 ОПК-6.2)*
28. На чем основана методика оценки вычислительной сложности алгоритмов поиска? *(ОПК-6.1)*
29. В чем состоят особенности задач внешнего поиска и сортировки? *(ОПК-6.1)*
30. Какой критерий является основным для алгоритмов внешней сортировки? *(ОПК-6.1)*
31. Для решения каких задач используется структура данных бинарное дерево поиска? *(ОПК-6.1)*
32. Опишите организацию узла динамического однонаправленного списка и алгоритм включения нового узла в список. *(ОПК-6.1 ОПК-6.2)*
33. Расскажите, используя графическое представление динамического однонаправленного списка, как выполняется операция удаления узла в любой позиции. *(ОПК-6.1 ОПК-6.2)*
34. Дана последовательность целых чисел. Сформируйте линейный однонаправленный список по правилу: справа от вершины все отрицательные числа последовательности, а слева положительные и нули. Реализуйте функцию. *(ОПК-6.1 ОПК-6.2 ОПК-6.3)*
35. Сравните алгоритмы Кнута-Морриса-Пратта и последовательного поиска. *(ОПК-6.1 ОПК-6.2)*
36. Приведите определение структуры данных Очередь. *(ОПК-6.1)*
37. Расскажите о способах реализации очереди. *(ОПК-6.1)*
38. Как организация очереди на однонаправленном динамическом списке с головным элементом может уменьшить сложность операции добавления узла до O(1)? *(ОПК-6.1 ОПК-6.2)*

**Вопросы для устного/письменного опроса по разделам**: (зависит от того сколько вы указали разделов, если 2 и более, то необходимо заполнить все разделы разными вопросам, если 1, то заполняете один).

Опрос проводится в устной или письменной форме с целью контроля самостоятельной работы студента в ходе подготовки к занятиям. Для опроса подбираются базовые вопросы, которые должен знать студент по данной теме.

**КОНТРОЛЬНЫЕ ВОПРОСЫ ПО ТЕМАМ**

**Основные понятия курса**

1. Охарактеризуйте каждый из трех уровней представления данных.
2. Что определяет типа данных в языках программирования?
3. Дайте определение структуре данных.
4. Как тип связан с понятием структура данных?
5. Определите структуру оперативной памяти на логическом уровне. Ячейка памяти.
6. Расскажите о способе агрегирование ячеек средствами С++.
7. Расскажите о структурах хранения данных, предоставляемых компьютерными технологиями.
8. Дайте определение линейной структуре данных.
9. Как реализуется прямой доступ к элементу в линейных структурах данных, в нелинейных структурах данных.
10. Понятие абстракции и ее назначение в разработке сложных программ.

**Анализ вычислительной сложности алгоритмов**

1. Приведите критерии эффективности программы.
2. Дайте определение вычислительной сложности алгоритма.
3. Как определить вычислительную сложность алгоритма поиска в структуре данных?
4. Какой критерий используется при выборе структуры данных при реализации задачи?
5. Объясните правило асимптотической оценки сложности алгоритма –нотация О.
6. Что определяет функция в асимптотической сложности?
7. Сформулируйте правило суммы для расчета сложности алгоритма по символике O.
8. Сложность в худшем случае.
9. Сложность в среднем и лучшем случае.
10. Приведите пример алгоритма и оцените его сложность алгоритма: в лучшем, среднем и худшем случае.
11. Какую вы выберите реализацию линейного списка, если предполагается часто вставлять новые элементы надо часто и вставлять в произвольную позицию?
12. Какие функции в оценке сложности называют доминирующими?
13. Охарактеризуйте класс P – задач.
14. Охарактеризуйте класс NP задач.
15. Классы функций, которые используются для оценки порядка роста времени выполнения.

**Стратегии и методы построения алгоритмов**

1. Парадигма динамического программирования.
2. Понятие оптимальной подструктуры.
3. Построение оптимального решения.
4. Жадные алгоритмы. Свойство жадного выбора.
5. Преобразование решения динамического программирования в жадное решение.
6. Рекурсивный жадный алгоритм. Итерационный жадный алгоритм.
7. Коды Хаффмана (Huffman codes).

**Рекурсивные алгоритмы**

1. Дайте определение рекурсивного объекта
2. Опишите структуру рекурсивного алгоритма
3. Как определить, что алгоритм имеет рекурсивную природу
4. Определение рекурсивного алгоритма
5. Что определяет шаг в рекурсии
6. Что определяет глубина рекурсии
7. Приведите рекуррентное соотношение для возведения числа х в степень n.
8. Что означает понятие – *линейная рекурсия*, *каскадная рекурсия* ?
9. Если в рекурсивной функции не предусмотреть условие завершения, то что произойдет с программой?
10. Разработайте рекурсивную функцию, которая считывает последовательность из n целых чисел и выводит сначала отрицательные числа, а затем положительные. Определите глубину и шаг рекурсии.
11. Расскажите о заполнении стека при рекурсивных вызовах.

**Алгоритмы внутренней сортировки**

1. Какая стратегия сортировки называется обменной сортировкой?
2. Какой метод является наиболее очевидным способом обменной сортировки?
3. Каково быстродействие алгоритма прямого обмена по сравнению с алгоритмом простыми вставками?
4. В чем заключаются усовершенствования алгоритма простого обмена и каковы его достоинства?
5. Какую идею реализует алгоритм сортировки слиянием?
6. Почему алгоритм быстрой сортировки Хоара называют обменной сортировки с разделением?
7. Какие из алгоритмов сортировки подходят для параллельных вычислений, а какие – для последовательных вычислений?
8. Каким образом представляются ключи в методе обменной поразрядной сортировки?
9. Какова оценка вычислительной сложности алгоритма R обменной поразрядной сортировки?
10. На чем основано семейство методов сортировки, называемое сортировкой посредством выбора?
11. Какие усовершенствования сортировки простым выбором Вам знакомы?
12. Выполните упорядочивание последовательности 1,7, 3, 2, 0, 5, 0, 8 с помощью алгоритмов простого обмена, простых вставок и простого выбора.
13. На чем основан алгоритм сортировки, называемой сортировкой простыми вставками?
14. Какова роль сортировки простыми вставками в сортировке бинарными вставками?
15. Какой механизм сортировки реализует алгоритм Шелла с убывающим шагом?
16. Как отличается эффективность алгоритма Шелла от сортировки простыми вставками в различных условиях?
17. Выполните упорядочивание последовательности 22, 36, 6, 79, 26, 45, 75, 13, 31, 62, 27, 76, 33, 16 используя пирамидальную сортировку.
18. Выполните упорядочивание последовательности 22, 36, 6, 79, 26, 45, 75, 13, 31, 62, 27, 76, 33, 16 используя быструю сортировку и сортировку вставками.
19. Какие пути усовершенствования алгоритма сортировки простыми вставками, основанные на тщательном анализе структур данных, Вам знакомы?
20. Каковы общие особенности турнирной и пирамидальной сортировки?
21. Что такое сортировка с прохождением бинарного дерева в симметричном порядке и сортировка методом турнира с выбыванием?
22. Какая сортировка называется пирамидальной?
23. В чем состоит сущность и реализация алгоритма пирамидальной сортировки?
24. Какие из перечисленных алгоритмов наиболее эффективны на почти отсортированных массивах: пирамидальная сортировка, сортировка слиянием, сортировка Шелла и сортировка Хоара? За счет чего происходит выигрыш?
25. Почему алгоритм быстрой сортировки Хоара называют обменной сортировки с разделением?
26. Принцип работы алгоритма быстрой сортировки Хоара
27. Быстрая сортировка Хоара с рекурсией и сортировка простым обменом (пузырьковая)
28. Быстрая сортировка Хоара с рекурсией и шейкерная сортировка.
29. Быстрая сортировка Хоара с рекурсией и выбором опорного элемента (средний, медиана из 3 элементов: первый, средний и последний).
30. Сортировка Хоара (быстрая) с рекурсией и модификация алгоритма, устраняющая одну ветвь рекурсии: вместо того, чтобы после разделения массива вызывать рекурсивно процедуру разделения для обоих найденных подмассивов, рекурсивный вызов делается только для меньшего подмассива, а больший обрабатывается в цикле в пределах этого же вызова процедуры.
31. Реализуйте быструю сортировку с использованием случайной выборки из 2k – 1 элементов. Сначала отсортируйте выборку, затем рекурсивной программой разбейте файл по медиане выборки, а оставшиеся половины выборки поместите в каждый подфайл так, чтобы они использовались в этих подфайлах без дальнейшей сортировки.
32. Определите эмпирическим путем наилучший размер выборки для сортировки методом случайной выборки для N = 103, 104, 105 и 106. Имеет ли значение, какой вид сортировки используется для упорядочения самой выборки: быстрая сортировка или сортировка методом случайной выборки?
33. Продемонстрируйте процесс блуждающей сортировки в порядке возрастания с помощью массива А = (5, 3, 17, 10, 84, 19, 6, 22, 9). Проведите анализ вычислительной сложности алгоритма.
34. Сущность и реализация алгоритма топологической сортировки.
35. Определите общую характеристику простых (прямых) методов сортировки: включения, выбора, обмена.
36. Проведите сравнительную оценку вычислительной сложности прямых алгоритмов сортировки по числу операций сравнения и числу операций перемещений элементов.
37. Укажите характеристику улучшенных методов сортировки, приведите оценки их вычислительной сложности.
38. Какую роль выполняет опорный элемент (ключ) в алгоритме быстрой сортировки Хоара?
39. В чем суть условия Айверсона и в каком алгоритме сортировки оно применяется?
40. Парадигма динамического программирования.
41. Какова вычислительная сложность алгоритма сортировки простым выбором.
42. Определите понятие устойчивости алгоритма сортировки. Какой из простых алгоритмов сортировки является устойчивым?
43. Какой критерий является основным для алгоритмов внешней сортировки?

**Алгоритмы поиска в массивах**

1. Сущность и реализация алгоритма последовательного поиска с барьером.
2. Опишите алгоритм упорядочивания элементов одномерного массива методом бинарного поиска.
3. Что такое Фибоначчиев поиск?
4. Почему нельзя использовать бинарный и интерполяционный алгоритмы поиска для списков?
5. Укажите виды обхода бинарных деревьев. Какой из них используется в алгоритме поиска?
6. Сформируйте бинарное дерево поиска, каждая вершинами которого представляет собой структуру вида: Страна | Город | Количество жителей | Метка. Выведите содержимое дерево всеми известными Вам методами обхода.
7. Определите особенности поиска по бинарному дереву.
8. В чем состоит методика оценки вычислительной сложности алгоритмов поиска?
9. В чем состоят особенности задач внешнего поиска и сортировки?
10. Какой критерий является основным для алгоритмов внешней сортировки?
11. Для решения каких задач используется структура данных бинарное дерево?
12. Какими свойствами обладает бинарное дерево как дерево поиска?
13. Как оценивается эффективность использования бинарных деревьев для задач поиска?
14. Какие шаги включает алгоритм поиска в бинарном дереве?

**Алгоритмы поиска в тексте**

1. К чему сводятся особенность поиска по бору?
2. В чем состоит методика оценки вычислительной сложности алгоритмов поиска?
3. В чем состоят особенности задач внешнего поиска и сортировки?
4. Сравните алгоритмы Кнута-Морриса-Пратта и последовательного поиска.
5. Сравните алгоритмы Бойера-Мура и последовательного поиска.
6. Сущность алгоритма Рабина-Карпа?
7. Что такое вычислительная сложность алгоритма?
8. Понятие вычислительной сложности. Основные функции оценки вычислительной сложности.

**Алгоритмы внешней сортировки**

1. Сравните между собой различные методы сжатия данных без потерь. Укажите основные направления развития методов сжатия данных без потерь.
2. Сферы применения и проблемы алгоритмов сжатия с потерями.
3. Чем обусловлено использование алгоритмов внешних сортировок?
4. Как расходуется ОЗУ при использовании различных алгоритмов внешних сортировок?
5. Каким слиянием, простым или естественным, эффективнее объединять два упорядоченных по общему ключу файла? Ответ обоснуйте.
6. Какие еще факторы, кроме числа фаз и путей, следует учитывать при анализе эффективности алгоритмов внешних сортировок?
7. Как определить, какому алгоритму внешних сортировок отдать предпочтение при решении задачи?
8. Чем можно объяснить многообразие алгоритмов сортировок?
9. Почему на данный момент не существует универсального алгоритма сортировки?
10. Как соблюдение свойств устойчивости и естественности влияет на трудоемкость алгоритма сортировки?
11. За счет чего в алгоритмах быстрых сортировок происходит выигрыш при выполнении операций сравнения и перестановок?
12. Какие из перечисленных алгоритмов наиболее эффективны на почти отсортированных массивах: бинарная, пирамидальная сортировка, сортировка слиянием, сортировка Шелла и сортировка Хоара? За счет чего происходит выигрыш?
13. Почему алгоритмы быстрых сортировок не дают большого выигрыша при малых размерах массивов?
14. В чем преимущества и недостатки по отношению друг к другу следующих алгоритмов сортировок: бинарная, пирамидальная сортировка, сортировка слиянием, сортировка Шелла и сортировка Хоара?
15. Как определить, какому алгоритму сортировки отдать предпочтение при решении задачи?
16. Чем обусловлено использование алгоритмов внешних сортировок?
17. Как расходуется ОЗУ при использовании различных алгоритмов внешних сортировок?
18. Каким слиянием, простым или естественным, эффективнее объединять два упорядоченных по общему ключу файла? Ответ обоснуйте.
19. Какие еще факторы, кроме числа фаз и путей, следует учитывать при анализе эффективности алгоритмов внешних сортировок?
20. Как определить, какому алгоритму внешних сортировок отдать предпочтение при решении задачи?
21. Что нужно сделать для корректности работы алгоритма?
22. Приведите несколько примеров алгоритмов сортировки? Какова вычислительная сложность приведенных алгоритмов?

**Линейные динамические списки**

1. Что в математике считается линейным списком?
2. Определите линейный список с позиций структуры данных.
3. Определите отношения в линейных структурах данных.
4. Какая структура считается однородной, неоднородной?
5. Как вы понимаете операцию – произвольный доступ к элементу в структуре; последовательный доступ?
6. Определите список базовых операций над линейным списком как структурой данных.
7. Приведите структуры языка программирования для реализации линейного списка.
8. Опишите динамический однонаправленный список.
9. Расскажите, используя графическое представление динамического однонаправленного списка, как выполняется операция вставки в любую позицию.
10. Расскажите, используя графическое представление динамического однонаправленного списка, как выполняется операция удаления узла в любой позиции.
11. Дана последовательность целых чисел. Сформируйте линейный однонаправленный список по правилу: справа от вершины все отрицательные числа последовательности, а слева положительные и нули. Реализуйте функцию.
12. Определите через нотацию O(f(n)) сложность выполнения операции вставки элемента в линейный список реализованный на массиве и на динамическом однонаправленном списке.
13. Реализуйте функцию, которая возвращает указатель на последний узел динамического однонаправленного списка.
14. Используя средства языка программирования опишите структуру узла двунаправленного списка.
15. Имеется ссылка на узел двунаправленного списка, который надо удалить. Приведите, используя необходимые переменные, последовательность операторов, которая обеспечит удаление узла.
16. Какие списки называют циклическими?
17. Какую операцию над линейным однонаправленным списком может оптимизировать реализация динамического списка на структуре с головным элементом?
18. Как определить начало в циклическом однонаправленном списке?
19. Определите структуру узла динамического двунаправленного списка для решения следующей задачи: требуется хранить сведения о книгах, взятых пользователями библиотеки. Каждый пользователь может иметь на руках несколько книг, причем у разных пользователей это число может быть различным. Реализуйте функцию, для поиска читателя с наибольшим количеством книг.
20. Приведите функцию вставки узлов в упорядоченный двунаправленный список. Этот список с самого начала формируется как упорядоченный. Список упорядочивается по значению ключа. Информационная часть узла любого типа.

**Линейный список стек**

1. Дайте определение структуре данных стек.
2. Каков алгоритм обслуживания стека?
3. Чем стек отличается от линейного списка?
4. Приведите определение АТД Стек.
5. Объясните на примере применение стека для вычисления значения без скобочного арифметического выражения, представленного в строковом формате.
6. Дано арифметическое выражение, содержащее круглые скобки. Напишите функцию, которая в выражении проверяет баланс скобок.
7. Приведите постфиксную форму следующего выражения:

5+7-4\*(7\*2+8/4)/3

1. Приведите префиксную форму следующего выражения: 5+7-4\*(7\*2+8/4)/3
2. Какие структуры удобно использовать для реализации алгоритма вычисления значения выражения, представленного в постфиксной форме?
3. Можно ли в одном массиве можно реализовать два стека?
4. Предложите структуру для реализации нескольких стеков в одном массиве. Какие проблемы при использовании этой структуры могут возникнуть.
5. Пусть имеется класс Stek с полным набором операций: Push, Pop, Peek, isEmpty, Реализуйте функцию сложения двух целых чисе любой разрядности, вводимых как последовательность цифр.

**Линейный список Очередь**

1. Приведите определение структуры данных Очередь?
2. Расскажите о способах реализации очереди.
3. Как организация очереди на однонаправленном динамическом списке с головным элементом может уменьшить сложность операции добавления узла до O(1)?
4. Опишите АТД Очередь с базовым набором операций.
5. Расскажите особенность реализации круговой очереди.
6. Как определить конец круговой очереди?
7. Объясните алгоритмы вставки и удаления в очереди с приоритетом.
8. Можно ли 10 очередей произвольного размера реализовать на одномерном массиве?
9. Объясните алгоритм вычисления значения выражения – операнды однозначные числа:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| + | + | 5 | 4 | / | \* | 3 | 8 | 6 |

Расскажите об особенностях структуры данных Дек.

**Деревья**

1. Дайте определение двоичного дерева.
2. Что такое высота дерева? Средняя высота?
3. Что такое обход дерева?
4. Назовите основные алгоритмы обходы дерева.
5. Дайте определение двоичного дерева поиска.
6. Что такое идеально сбалансированное дерево поиска?
7. Дайте определение Б-дерева порядка *m*.
8. Назовите трудоемкость поиска вершины в Б-дереве порядка *m*.
9. Какова высота Б-дерева порядка *m*.
10. Каким образом добавляется новый элемент в Б-дерева порядка *m.*
11. Что такое двоичное Б-дерево?
12. Какова высота двоичного Б-дерева?
13. Какова трудоемкость добавления вершины в двоичное Б-дерево?
14. Дайте определение АВЛ-дерева.
15. Является ли АВЛ-дерево деревом поиска?
16. Какова высота АВЛ-дерева?
17. Для чего нужны повороты АВЛ-дерева?
18. Какова трудоемкость включения и исключения вершины АВЛ-дерева
19. Что такое показатель баланса вершины?
20. Красно-черные деревья. Способы представления в памяти.
21. Какое дерево называется красно-черным деревом?
22. Чему равно наибольшее возможное число внутренних узлов в красно-черном дереве с черной высотой h? А наименьшее возможное число?
23. Приведите пример пошагового построения красно-черного дерева.

**Основные алгоритмы работы с графами**

1. Что такое граф, подграф, орграф?  Какие графы бывают?
2. Что такое направленные и ненаправленные графы?
3. Какие бывают способы представления графа в памяти?
4. Как от вида или представления графа зависит вычислительная сложность алгоритмов поиска в глубину и в ширину?
5. Реализуйте алгоритм нахождения матрицы достижимости на основе использования стратегии поиска в глубину или ширину и сравните время работы этого алгоритма с временем работы алгоритма нахождения транзитивного замыкания.
6. Что такое алгоритм Дейкстры поиска кратчайшего пути в графе между двумя заданными вершинами?
7. Что такое алгоритм Флойда-Уоршелла поиска кратчайших путей в графе между всеми вершинами в графе?
8. Укажите отличия алгоритмов Прима и Крускала построения минимального покрывающего дерева?

**Хеш таблицы и алгоритмы хеширования**

1. Что такое хеширование?
2. Структура записи таблицы, к которой применяется механизм хеширования?
3. Какое назначение имеет хеш – функция?
4. Какие способы выбора хеш- функций вы знаете?
5. Как определяется размер хэш-таблицы?
6. Что такое коллизии?
7. Какие способы разрешения коллизий вы знаете?
8. Расскажите о проблемах, возникающих при использовании цепного хеширования?
9. Какой процесс называют рехеширование?
10. Как определить, что пора выполнить рехеширование, если используется цепное хеширование?
11. В чем суть принципа однородности в процессе хеширования?
12. Правило Дейкстры о назначении размера таблицы.
13. Объясните алгоритм выполнения операции вставки в таблицу с открытым адресом.
14. Объясните термин «Открытый адрес» в таблице.
15. Расскажите, как реализуется коллизия в таблице с открытым адресом в самом простом варианте?
16. Если способ подбора свободного адреса порождает цикличность поиска, то какой алгоритм подбора можно применить?
17. Объясните, какая проблема может возникнуть, если удалить из таблицы: значение с ключом 304002, а потом надо найти данные по ключу 305002 и как ее устранить.

Хеш таблица с данными

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | ….. | 100 |
| 111001 | 111002 | 112001 | 212001 | 303002 | 304002 | 111007 | 305002 |  | 303010 |  | ….. |  |

1. Расскажите о проблемах хеширования с открытым адресом.

**Внешняя структура данных – файл**

1. Дайте определение файла на физическом и логическом уровне.
2. Дайте определение текстовому и двоичному файлам.
3. Приведите способы создания текстовых файлов-данных
4. Как можно создать двоичный файл данных?
5. Объясните процесс открытия файла.
6. Объясните процесс закрытия файла.
7. Расскажите, какую операцию выполняет метод open(…) над потоковыми файлами.
8. Как вы понимаете смысл выражения – *буфер потока?*
9. В текстовом файле подготовлены данные по строкам

*С++ Методики программирования Шилдта*

*Г. Шилдт*

*2017*

*Вильямс*

*475*

для заполнения объекта Book(Название, Автор, год выпуска, Издательство, Количество страниц)/

1. Разработайте функцию, которая будет считывать эти данные и записывать в соответствующие поля объекта. Структуру объекта определите через struct. Типы полей должны соответствовать типам значений.
2. Как вы понимаете смысл операции прямой доступ к записи в двоичном файле?
3. Если в файле хранятся значения матрицы размером 5х5, то как прочитать значения целой строки, например, с номером 5?
4. Какова задача метода flush в файловых потоках?

**Алгоритмы сжатия и кодирования данных**

1. Порядок работы алгоритма Хаффмана.

2. Построение оптимального кодового дерева.

3. Средняя длина кода и ее расчет.

4. Дисперсия кода и ее расчет.

5. Какой алгоритм сжатия применяется в архиваторе Zip?

1. При кодировании каких данных можно использовать сжатие данных с потерями? Ответ обоснуйте.
2. В чем преимущества и недостатки статических методов и словарного сжатия?
3. Каким образом кодирование по алгоритму Хаффмана через префиксный код гарантирует минимальную длину кода?
4. За счет чего в методе Хаффмана поддерживается однозначность соответствия кода кодируемому символу?
5. Почему алгоритм Хаффмана малоэффективен для файлов маленьких размеров?
6. Выполните кодирование по методу Хаффмана через префиксный код символов, которые встречаются с вероятностями 0,3; 0,2; 0,1; 0,1; 0,1; 0,05; 0,05; 0,04; 0,03; 0,03. Сравните полученный результат с данными программной реализации.
7. Докажите, что метод Хаффмана кодирует информацию без потерь.
8. Какие параметры надо определить, прежде чем сравнивать два алгоритма сжатия?
9. Почему некорректно сравнивать временные параметры реализаций алгоритмов сжатия, оптимально реализованных на разных компьютерах? Приведите примеры ситуаций, когда архитектура компьютера дает преимущества тому или иному алгоритму.
10. Назовите основные требования приложений к алгоритмам компрессии.
11. Задача сжатия информации. Сжатие без потерь, сжатие с потерями, их сферы применения.
12. Задача сжатия информации. Сжатие без потерь. Алгоритмы RLE, LZW.
13. Приведите примеры статистических и словарных методов сжатия данных без потерь.
14. Статистическое сжатие данных. RLE кодирование. Арифметическое кодирование.
15. Коды Хаффмана и их применение для сжатия.
16. Словарные методы сжатия данных без потерь. Метод Лемпеля–Зива и его производные.
17. Какие структуры данных используются в методе Лемпеля-Зива. Проблемы практической реализации алгоритмов словарного сжатия.

**Основы линейного программирования**

1. Линейное программирование. Стандартная и каноническая формы задач линейного программирования.
2. Преобразование задач линейного программирования в стандартную форму.
3. Преобразование задач линейного программирования в каноническую форму.
4. Начальное базисное допустимое решение. Поиск начального решения.
5. Операция замещения. Поиск оптимального решения.
6. Двойственность (дуальность) задач линейного программи­рования.
7. Задача целочисленного линейного программирования (ЦЛП).
8. Метод ветвей и границ для решения задач ЦЛП.

**Задания для практических занятий** (заданий можно сделать больше чем занятий)

Практические задания выдаются студентам с целью применения полученных знаний на практике. Практические задания могут быть представлены в виде решения задач, проблемных заданий, подготовки доклада и презентации, тренингов и иных видах, направленных на получение практических знаний.

# Практическая работа № 1

**Простые алгоритмы внутренней сортировки массивов**

***Постановка задачи***

Составить программу сортировки одномерного целочисленного массива A[n], используя алгоритм согласно варианту индивидуального задания. Провести тестирование программы на исходном массиве, сформированном вводом с клавиатуры. Рабочий массив A сформировать с использованием генератора псевдослучайных чисел. Провести контрольные прогоны программы для размеров массива n = 100, 1000, 10000, 100000 и 1000000 элементов с вычислением времени выполнения T(n). Полученные результаты свести в сводную таблицу. Построить график зависимости времени выполнения программы от размера массива.

Провести эмпирическую (практическую) оценку вычислительной сложности алгоритма, для чего предусмотреть в программе подсчет фактического количества операций сравнения Сф и количества операций перемещения Мф. Полученные результаты свести в сводную таблицу. Построить в одной координатной плоскости графики зависимости теоретической О(n)=f(С+М) и эмпирической (Сф+Мф) вычислительной сложности алгоритма от размера массива n.

Провести анализ полученных результатов. Сделать выводы о проделанной работе, основанные на полученных результатах.

***Сводная таблица результатов***

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **T** | **f(C+M)** | **Cф+Mф** |
| 100 |  |  |  |
| 1000 |  |  |  |
| 10000 |  |  |  |
| 100000 |  |  |  |
| 1000000 |  |  |  |

***Варианты индивидуальных заданий***

|  |  |
| --- | --- |
| **№** | **Алгоритм** |
| 1.1 | Простой вставки (***Insertion sort***) |
| 1.2 | Простой вставки (***Exchange sort***) |
| 1.3 | Простого выбора ***(Selection sort)*** |

***Дополнения к индивидуальным заданиям повышенной сложности***

1. Провести дополнительные прогоны программы на рабочих массивах, отсортированных строго в убывающем и возрастающем порядке значений элементов. Провести анализ зависимости (или независимости) алгоритма сортировки от исходной упорядоченности массива***.***

2. Провести программную реализацию нескольких алгоритмов с последующим сравнительным анализом полученных результатов контрольных прогонов и построением соответствующих графиков.

# Практическая работа № 2

**Усовершенствованные алгоритмы внутренней сортировки массивов**

***Постановка задачи***

Составить программу сортировки одномерного целочисленного массива A[n], используя алгоритм согласно варианту индивидуального задания. Провести тестирование программы на исходном массиве, сформированном вводом с клавиатуры. Рабочий массив A сформировать с использованием генератора псевдослучайных чисел. Провести контрольные прогоны программы для размеров массива n = 100, 1000, 10000, 100000 и 1000000 элементов с вычислением времени выполнения T(n). Полученные результаты свести в сводную таблицу. Построить график зависимости времени выполнения программы от размера массива.

Провести эмпирическую (практическую) оценку вычислительной сложности алгоритма, для чего предусмотреть в программе подсчет фактического количества операций сравнения Сф и количества операций перемещения Мф. Полученные результаты свести в сводную таблицу. Построить в одной координатной плоскости графики зависимости теоретической О(n)=f(С+М) и эмпирической (Сф+Мф) вычислительной сложности алгоритма от размера массива n.

Провести анализ полученных результатов. Сделать выводы о проделанной работе, основанные на полученных результатах.

***Сводная таблица результатов***

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **T** | **f(C+M)** | **Cф+Mф** |
| 100 |  |  |  |
| 1000 |  |  |  |
| 10000 |  |  |  |
| 100000 |  |  |  |
| 1000000 |  |  |  |

***Варианты индивидуальных заданий***

|  |  |
| --- | --- |
| **№** | **Алгоритм** |
| 2.1 | Шейкерной ***(Shaker sort)*** или челночной сортировки |
| 2.2 | Шелла ***(Shell sort)*** |
| 2.3 | Простого выбора ***(Selection sort)*** |
| 2.4 | Быстрой сортировки (***Quick-sort*** или сортировки ***Хоара***) |
| 2.5 | Турнирной сортировки |
| 2.6 | Пирамидальной сортировки (***Heapsort)*** |

***Дополнения к индивидуальным заданиям повышенной сложности***

1. Провести дополнительные прогоны программы на рабочих массивах, отсортированных строго в убывающем и возрастающем порядке значений элементов. Провести анализ зависимости (или независимости) алгоритма сортировки от исходной упорядоченности массива***.***

2. Провести программную реализацию нескольких алгоритмов (как простых, так и усовершенствованных) с последующим сравнительным анализом полученных результатов контрольных прогонов и построением соответствующих графиков.

# Практическая работа № 3

**Алгоритмы внешней сортировки массивов**

***Постановка задачи***

Составить программу сортировки одномерного целочисленного массива A[n] методом слияния (merge-sort). Провести тестирование программы на исходном массиве, сформированном вводом с клавиатуры. Рабочий массив A сформировать с использованием генератора псевдослучайных чисел. Провести контрольные прогоны программы для размеров массива n = 100, 1000, 10000, 100000 и 1000000 элементов с вычислением времени выполнения T(n). Полученные результаты свести в сводную таблицу. Построить график зависимости времени выполнения программы от размера массива.

Провести эмпирическую (практическую) оценку вычислительной сложности алгоритма, для чего предусмотреть в программе подсчет фактического количества операций сравнения Сф и количества операций перемещения Мф. Полученные результаты свести в сводную таблицу. Построить в одной координатной плоскости графики зависимости теоретической О(n)=f(С+М) и эмпирической (Сф+Мф) вычислительной сложности алгоритма от размера массива n.

Провести анализ полученных результатов. Сделать выводы о проделанной работе, основанные на полученных результатах.

***Сводная таблица результатов***

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **T** | **f(C+M)** | **Cф+Mф** |
| 100 |  |  |  |
| 1000 |  |  |  |
| 10000 |  |  |  |
| 100000 |  |  |  |
| 1000000 |  |  |  |

***Варианты индивидуальных заданий***

|  |  |
| --- | --- |
| **№** | **Метод** |
| 3.1 | Простое слияние |
| 3.2 | Естественное слияние |
| 3.3 | Многофазная сортировка |

***Дополнение к индивидуальным заданиям повышенной сложности***

Провести дополнительные прогоны программы на рабочих массивах, отсортированных строго в убывающем и возрастающем порядке значений элементов. Провести анализ зависимости (или независимости) алгоритма сортировки от исходной упорядоченности массива***.***

# Практическая работа № 4

**Алгоритм поиска в неотсортированных массивах**

***Постановка задачи***

Составить программу последовательного (с барьером) поиска заданного элемента по ключу в одномерном целочисленном массиве A[n]. Провести тестирование программы на исходном массиве, сформированном вводом с клавиатуры. Рабочий массив A сформировать с использованием генератора псевдослучайных чисел. Провести контрольные прогоны программы для размеров массива n = 100, 1000, 10000, 100000 и 1000000 элементов в трех режимах: на массивах, строго убывающих, строго возрастающих и случайных чисел (в наилучшем, наихудшем и среднем случаях) и сделать вывод о зависимости (устойчивости) алгоритма от исходной упорядоченности массива.

Провести эмпирическую (практическую) оценку вычислительной сложности алгоритма, для чего предусмотреть в программе подсчет фактического количества операций сравнения Сф. Полученные результаты свести в сводную таблицу. Построить в одной координатной плоскости графики зависимости теоретической О(n)=f(С(n)) и эмпирической (Сф(n)) вычислительной сложности алгоритма от количества элементов в массиве n.

Провести анализ полученных результатов. Сделать выводы о проделанной работе, основанные на полученных результатах.

***Сводная таблица результатов***

|  |  |  |
| --- | --- | --- |
| **n** | **f(C)** | **Cф** |
| 100 |  |  |
| 1000 |  |  |
| 10000 |  |  |
| 100000 |  |  |
| 1000000 |  |  |

# Практическая работа № 5

**Алгоритм поиска в отсортированных массивах**

***Постановка задачи***

Составить программу поиска заданного элемента по ключу в одномерном целочисленном массиве A[n], используя алгоритм согласно варианту индивидуального задания. Провести тестирование программы на исходном массиве, сформированном вводом с клавиатуры. Рабочий массив A сформировать с использованием генератора псевдослучайных чисел. Провести контрольные прогоны программы для размеров массива n = 100, 1000, 10000, 100000 и 1000000 элементов в трех режимах: на массивах, строго убывающих, строго возрастающих и случайных чисел и сделать вывод о зависимости (устойчивости) алгоритма от исходной упорядоченности массива.

Провести эмпирическую (практическую) оценку вычислительной сложности алгоритма, для чего предусмотреть в программе подсчет фактического количества операций сравнения Сф.

Полученные результаты свести в сводную таблицу. Построить в одной координатной плоскости графики зависимости теоретической О(n)=f(С(n)) и эмпирической (Сф(n)) вычислительной сложности алгоритма от количества элементов в массиве n.

Сравнить вычислительную сложность алгоритма с вычислительной сложностью алгоритма последовательного поиска. Экспериментально оценить долю случаев, когда последовательный поиск выполняется быстрее, чем быстрый поиск.

Провести анализ полученных результатов. Сделать выводы о проделанной работе, основанные на полученных результатах.

***Сводная таблица результатов***

|  |  |  |
| --- | --- | --- |
| **n** | **f(C)** | **Cф** |
| 100 |  |  |
| 1000 |  |  |
| 10000 |  |  |
| 100000 |  |  |
| 1000000 |  |  |

***Варианты индивидуальных заданий***

|  |  |
| --- | --- |
| **№** | **Алгоритм** |
| 5.1 | Двоичного поиска |
| 5.2 | С использованием бинарного дерева поиска |
| 5.3 | Фибоначчиего поиска |
| 5.4 | Поиска хэшированием |
| 5.5 | Поиска по бору |
| 5.6 | Поиска Рабина-Карпа |

# Практическая работа № 6

**Алгоритмы поиска строк в тексте**

***Постановка задачи***

Составить программу поиска первого вхождения заданной строки P длиной m символов в тексте S, размером n символов, используя алгоритм согласно варианту индивидуального задания. Уточнение: настоящая задача поиска сводится к нахождению в тексте (массиве) S индекса, начиная с которого строка P полностью совпадает с фрагментом текста S. В частном случае заданная строка может отсутствовать в тексте.

Провести тестирование программы на исходном массиве, сформированном вводом с клавиатуры.

Рабочий текст (массив) сформировать из произвольного текстового файла, например, романа Л.Н. Толстого «Война и мир». Провести контрольные прогоны программы как минимум на трех текстовых файлах различной длины.

Провести эмпирическую (практическую) оценку вычислительной сложности алгоритма, для чего предусмотреть в программе подсчет фактического количества операций посимвольного сравнения Сф и сдвигов подстроки Мф относительно текста.

Полученные результаты свести в сводную таблицу. Построить в одной координатной плоскости графики зависимости теоретической О(n)=f(С+М) и эмпирической (Сф+Мф) вычислительной сложности алгоритма от размера текста (количества элементов в массиве) n.

Сравнить вычислительную сложность алгоритма с вычислительной сложностью алгоритма прямого поиска строки.

Провести анализ полученных результатов. Сделать выводы о проделанной работе, основанные на полученных результатах.

***Сводная таблица результатов***

|  |  |  |
| --- | --- | --- |
| **n** | **f(C+М)** | **Cф+Мф** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

***Варианты индивидуальных заданий***

|  |  |
| --- | --- |
| **№** | **Алгоритм** |
| 6.1 | Прямого поиска строки |
| 6.2 | Кнута-Морриса-Пратта |
| 6.3 | Бойера-Мура |
| 6.4 | Поиска хэшированием |

# Практическое задание №7

**Линейные динамические списки**

1. ***Однонаправленные списки***

***Постановка задачи***

Определите список операции над списками варианта, включая операцию добавления узла в начало списка, вывода списка. Разработайте для каждой операции функцию с параметрами. Информационная часть узла имеет тип int.

Реализуйте программу решения задачи варианта.

***Вариант 1***

Даны два линейных однонаправленных списка L1 и L2. Разработать процедуру, которая формирует список L, включив в него по одному разу элементы, значения которых входят хотя бы в один из списков L1 и L2.

***Вариант 2***

Даны два линейных однонаправленных списка L1 и L2. Разработать процедуру, которая формирует список L, включив в него по одному разу элементы, значения которых входят одновременно в оба списка L1 и L2.

***Вариант 3***

Даны два линейных однонаправленных списка L1 и L2. Разработать процедуру, которая формирует список L, включив в него по одному разу элементы, значения которых входят в список L1 и не входят в список L2.

***Вариант 4***

Даны два линейных однонаправленных списка L1 и L2. Разработать процедуру, которая формирует список L, включив в него по одному разу элементы, значения которых входят в один из списков L1 и L2 и в не входят в другой.

1. ***Двунаправленные списки***

***Постановка задачи***

Разработать многомодульную программу, которая демонстрирует выполнение всех операций, определенных вариантом, над линейным двунаправленным динамическим списком.

Требования к разработке.

1. Разработать структуру узла списка, структура информационной части узла определена вариантом. Для определения структуры узла списка, используйте тип struct. Сохраните определение структуры узла в заголовочном файле.
2. Разработайте функции для выполнения операции над линейным динамическим списком:

* вывод списка в двух направлениях
* поиск узла с заданным значением (операция должна возвращать указатель на узел с заданным значением).

1. Дополнительные операции над списком, указанные вариантом оформите в виде функций и включите в отдельный файл с расширением cpp. Подключите к этому файлу заголовочный файл с определением структуры узла.
2. Разработайте программу, управляемую текстовым меню, и включите в меню демонстрацию выполнения всех операций задания и варианта.
3. Проведите тестирование операций.
4. Оцените сложность алгоритма первой дополнительной операции для реализации линейного списка:

* на линейном динамическом списке
* на одномерном массиве.

Примечание. В определении информационной части узла варианта, подчеркнутое поле считать полем ключа.

***Варианты***

|  |  |  |
| --- | --- | --- |
| Вариант | Тип информационной части узла списка | Дополнительные Операции |
| 1 | Номер зач. книжки, Номер группы, Оценка. | Вставить новый узел перед первым узлом с таким же ключом, если такого узла еще нет, то вставить перед первым узлом, у которого ключ больше.  Удалить узлы с указанным номером группы.  Сформировать новый список из исходного, включив в него узлы с оценкой неуд, исключив их при этом из исходного списка. |
| 2 | Номер телефона (из 7 цифр), время разговора (целое число), номер телефона вызываемого абонента. | Добавить новый узел в список, упорядочивая узлы по первым четырем цифрам телефона в порядке возрастания.  Удалить последний узел с заданным значением телефона.  Подсчитать суммарное время разговора с заданного телефона. |
| 3 | Номер абонемента, Название книги, дата выдачи, дата возврата, дата фактического возврата. | Вставить новый узел в список после последнего узла с таким же номером абонента(дата фактического возврата еще не заполнена).  Изменить значение поля фактической даты возврата по указанной книге, указанного абонемента.  Удалить узлы, в которых дата возврата и дата фактического возврата совпадают.  Определить количество книг, заданного абонемента. |
| 4 | Номер мед. полиса, Дата обращения, Код диагноза (число). | Вставка нового узла перед первым узлом с заданным значением Мед. полиса, если такого нет, то узел вставить в конец списка.  Удаление из списка всех узлов с заданным значением Кода диагноза.  Переместить все узлы с одинаковым мед. полисом в новый список.  Определить количество обращений в одну и туже дату с одним и тем же диагнозом. |
| 5 | Номер счета в банке, дата, вид операции (приход или расход), сумма вклада. | Вставка нового узла перед первым узлом.  Удаление сведений по счету (всех узлов), у которого общая сумма вклада равна нулю ( сумма по приходу, минус сумма по расходу).  Создать новый список из исходного, которого будет содержать остаток по всем видам операций одного счета, указав вид операции – приход, и текущую дату. |
| 6 | Номер автобусного маршрута, время отправления (целое число), номер автобуса, стоимость одной поездки, дата отправления. | Вставить новый узел после последнего узла с заданным номером автобуса.  Удалить все узлы заданного автобуса.  Подсчитать, сколько раз автобус выходил на маршрут в течении заданного дня. |

# Практическое задание №8

**Линейный список стек и его применение**

***Постановка задачи***

Реализовать операции над стеком: втолкнуть элемент в стек; вытолкнуть элемент из стека; получить значение узла стека; проверить не пуст ли стек; не полон ли стек. Информационная часть узла стека определяется задачей варианта. Каждую операцию оформить функцией с параметрами.

Реализовать хранение стека двумя способами: на динамическом массиве и на однонаправленном динамическом списке. Для каждого способа операции реализовать отдельно.

Операции и определение узла представить отдельным модулем программы.

Разработать программу по задаче варианта.

***Вариант 1***

Разработать программу сложения двух больших целых чисел (не попадающих в диапазон стандартных типов), вводимых с клавиатуры, как последовательность символов.

***Вариант 2***

Дан массив из n элементов целого типа. Предполагая, что элементы массива – это высоты некоторых сооружений, необходимо определить, сколько таких сооружений, можно увидеть из начальной точки 0. Все остальные точки (от 1 до n) – это элементы массива.

Пример входных данных:

6

8 2 3 11 11 10

5

12 20 39 45 89

Выходные данные:

2

5

Пояснение:

В первом случае, нам видна точка 8, за ней следует 11 (2 и 3 не видно, так как они меньше 8), последние два элемента (11 и 10) также не видны так как 11 находится на том же уровне, что и 4 элемент (тоже 11), а 10 – еще ниже. Таким образом, видны только два сооружения – 8 и 11 и ответ будет 2.¶

Во втором случае, будут видны все элементы, так как они расположены по возрастанию и ответом будет 5.

Решите задачу без использования стека и с использованием стека.

***Вариант 3***

Можно хранить два стека в одном массиве, если один располагается в начале массива и растетк концу массива, а торой располагается в конце и растет к началу. Реализуйте операцию Push(x,S) –втолкнуть элемент х в стек S, где S один или другой стек. Включите все необходимые проверки в эту процедуру.

***Вариант 4***

Дан текст, сформированный по правилу:

<текст>::=<пусто>|<элемент><текст>

<элемент>::=<буква>|(<текст>).

Требуется для каждой пары соответствующих открывающей и закрывающей скобок вывести номера их позиций в тексте, упорядочив пары номеров в порядке возрастания номеров позиций:

а) закрывающих скобок

б) открывающих скобок.

Например, для текста А+(45-А(Х)\*(В-С)) должно быть выведено:

а) 8 10; 12 16; 3 17;

б) 3 17; 8 10; 12 16.

***Вариант 5***

Вычислить значение арифметического без скобочного выражения, введенного в виде строки, предварительно преобразовав его в постфиксную форму. Операнды операций – это целые числа. Перед преобразованием выражения следует проверить его на корректность записи операций и операндов.

***Вариант 6***

Преобразовать арифметическое без скобочное выражение, введенное в виде строки, в префиксную форму. Операнды операций – это целые числа. Перед преобразованием выражения следует проверить на корректность записи операций и операндов.

***Вариант 7***

Разработайте калькулятор, который вычисляет значение арифметического выражения, заданного в постфиксной форме и содержащего как бинарные операции, так и унарный минус.

***Вариант 8***

Вычислить значение выражения, представленного в виде формулы вида:

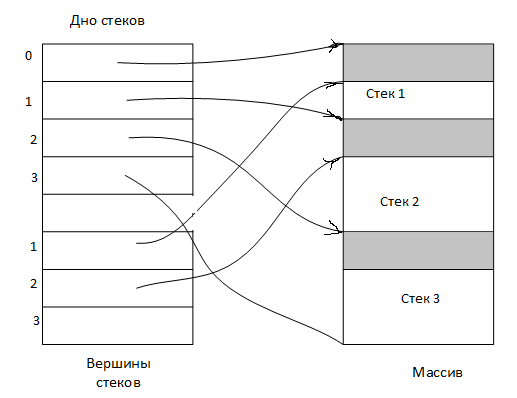
<формула>::=<цифра>| **М**(<формула>,<формула>)|**m**(<формула>,<формула>)|

<цифра>::=0|1|2|3|4|5|6|7|8|9/

Где М обозначает функцию вычисления максимального, а m функцию вычисления минимального. Например, М(5, m(6,8)), то результат равен 6.

***Вариант 9***

Реализуйте хранение ***k*** стеков одном массиве, используя следующую конструкцию структуры хранения:

******

На рисунке представлена структура для хранения трех стеков в одном массиве. Для этих стеков должны выполняться все операции над стеком. Если при вставке в i-ый стек вершина i-ого стека Top(i) совпадает с «дном» предыдущего стека Bottom(i-1), то надо переместить стеки так, чтобы между карой парой смежных стеков были пустые ячейки. Количество пустых ячеек между стеками можно сделать одинаковым или пропорциональным длине соседнего с образованной пустотой стека. Разработайте функции реорганизации массива при возникновении конфликта между стеками и функцию, которая определяет новые вершины, так чтобы пространство перед стеком было пропорционально его длине. Примечание. Стек i может перемещаться как вниз, так и вверх по массиву.

# Практическое задание № 9

**Линейный список Очередь**

***Постановка задачи***

Реализовать операции над очередью: вставить элемент в очередь; удалить элемент из очереди; получить значение элемента очереди; проверить не пуста ли очередь; не полна ли очередь. Информационная часть узла стека определяется задачей варианта. Каждую операцию оформить функцией с параметрами.

Реализовать хранение очереди двумя способами: на динамическом массиве и на однонаправленном динамическом списке. Операции реализовать отдельно для каждого способа.

Операции и определение узла представить отдельным модулем программы.

Разработать программу по задаче варианта.

Оценить сложность выполнения задачи при различных реализациях хранения, используя небольшое и большое количество элементов.

***Вариант 1***

Выполнить сортировку массива целых чисел, используя алгоритм поразрядной сортировки. Используйте свободный двумерный массив для хранения десяти очередей, причем длина очереди зависит от количества элементов в очереди.

Усовершенствуйте алгоритм, используя для хранения десяти очередей один одномерный массив.

***Вариант 2***

Используя очередь с приоритетом решите следующую задачу. В компании имеется несколько категорий сотрудников, которые могут подавать заявки на выполнение заданий: менеджер (М), контролер(К), рабочий(Р). Прием заявок автоматизирован. О каждой заявке хранится информация: категория сотрудника, его номер, время выполнения заявки в минутах. Заявки вводятся в очередь приоритетов с указанием приоритета сотрудника. Приоритет устанавливается так: у М – высший(1), у К – 2, у Р –3. Заявки извлекаются из очереди приоритетов и выполняются (раз извлечена из очереди, то считается что выполнена). Вывести время, затраченное на обслуживание каждой категории сотрудников и информацию о каждом задании.

***Вариант 3***

Разработать префиксный калькулятор, который вычисляет значение арифметического выражения, представленного в префиксной форме. Выражение поступает в программу в инфиксной форме в виде строки и может содержать скобки. Калькулятор преобразует выражение в префиксную форму и вычисляет. Операнды выражения целые числа, операции - +, -, \*,/.

***Вариант 4***

Очередь в теории массового обслуживания. Применение очереди в модели продвижения автомобилей на пункте мойки машин.

Фиксируя время прибытия автомашину на мойку, рассчитать среднее время ожидания для каждой машины.

Условие.

Станция мойки имеет один пункт обслуживания. Время мойки каждой машины составляет ровно 10 минут. В любой момент в очереди на мойку может стоять не более 5 машин. Если в процессе мойки пребывает еще одна машина, а в очереди уже имеется 5 машин, эта машина не допускается на мойку и наступает «переполнение очереди». Среднее время ожидания определяется суммированием времени ожидания каждой машиной и делением полученной суммы на количество машин. В качестве метки конца ввода данных о времени прибытия машины можно использовать некоторое значение, например, число 999. Для прибытия и убытия машин действуют следующие правила:

1. Если прибытие и убытие имеют место в течение одной и той же минуты, то первым учитывается убытие.
2. Если машины прибывают, когда очередь пуста, и никакая из машин в данный момент не обслуживается, то машина поступает на мойку немедленно, и в очередь она не ставится.
3. Машина покидает очередь и завершает свое ожидание, когда начинается 10 минутный цикл мойки.

Теста 1

Входные данные

Время поступления машин: 1, 3, 5, 8, 15, 999

время убытия =11,

Результат

Время убытия машин соответственно: 21, 31, 41, 51, 61

Среднее время ожидания в минутах: 19,33

Тест 2

Входные данные

Время поступления машин:

5, 5, 7, 12, 12, 13, 14, переполнение, 18, время убытия =15, 19 переполнение, 25, время убытия=25, 999

время убытия =11,

Результат

Время убытия машин соответственно: 35, 45 55, 65, 75, 85

Среднее время ожидания в минутах: 27,875

# Практическое задание №10

**Бинарное дерево поиска. AVL дерево**

1. Написать программу создания двоичного дерева поиска по заданной последовательности целых чисел.
2. Написать процедуры вставки и удаления узлов дерева.
3. Проанализировать время работы алгоритмов вставки и удаления узлов дерева.
4. Приведите оценку эффективности построения бинарного дерева поиска для сортировки массива.
5. Вывести упорядоченный массив из бинарного дерева поиска.
6. Построить диаграмму активности (или блок-схему) всех вариантов взаимного расположения и цветов узлов, возможных при вставке нового узла в красно-черном дереве (а также действий, выполняемых в каждой ситуации для вставки узла).
7. Скачать приложение RBTree Workshop и Воспроизведите в нем все ситуации, описанные в задании 1, и выполните инструкции по вставке.
8. Выполните достаточное количество вставок и убедитесь, что при точном соблюдении красно-черных правил 1, 2 и 3 правило 4 выполняется автоматически.

Примечание. Правила построения красно-черного дерева

1. Каждый узел окрашен в красный или черный цвет.

2. Корень всегда окрашен в черный цвет.

3. Если узел красный, то его потомки должны быть черными (хотя обратное не всегда истинно).

4. Все пути от корня к узлу или пустому потомку должны содержать одинаковое количество черных узлов.

# Практическое задание №11

**Хеширование - прямой доступ к данным**

***Постановка задачи***

1. Разработайте приложение, которое использует хеш таблицу для организации прямого доступа к элементам множества, реализованного на массиве, структура элементов которого приведена в варианте.
2. Множество реализуйте через класс - шаблон с операциями вставки, удаления, поиска, вывода и включите в него хеш таблицу.
3. . Хеш функцию подберите самостоятельно, используя правила выбора функции.
4. Разработайте такие тесты, чтобы возникли коллизии.
5. Выведите список индексов, которые формируются при вставке элементов в таблицу.
6. Проводите рехеширование, когда этого требуется, в соответствии с типом реализации коллизий.

Примечание. Ключи записей в вариантах подчеркнуты.

***Варианты***

|  |  |  |
| --- | --- | --- |
| № | Метод хеширования (способ реализации коллизий) | Структура элемента множества |
| 1 | С открытой адресацией (увеличение на 1) | Читательский абонемент: номер читательского - целое пятизначное число, ФИО, Адрес |
| 2 | С открытой адресацией (увеличение на номер выполняемого подбора) | Счет в банке: номер счета 7 разрядное число, ФИО, Адрес |
| 3 | С открытой адресацией (двойное хеширование) | Владелец телефона: номер телефона – последовательность символов, адрес |
| 4 | Цепное хеширование | Товар: название, код – шестиразрядное число |
| 5 | Цепное хеширование | Специализация вуза: код специальности, название вуза |
| 6 | Открытый адрес (двойное хеширование) | Книга: ISBN – двенадцатизначное число, Автор, Название |
| 7 | Цепное хеширование | Страховой полис: номер, компания, фамилия владельца |
| 8 | Открытый адрес (увеличение на 1) | Проверка проавописания по словарю правильных слов: слово (в тексте найти слова с орфографическими ошибками) |

# Практическое задание №12

**Внешние структуры данных - файлы и алгоритмы их обработки**

***Постановка задачи***

Создать программные модули с операциями над двоичными и текстовыми файлами для выполнения задания варианта. Двоичный файл состоит из записей определенной структуры. Записи имеют ключ, уникальный в пределах файла, создается он как счетчик: количество записей + 1. Для выполнения варианта задания использовать потоковые файлы С++.

1. Подготовьте тестовые данные в текстовом файле с кодировкой ASCII.
2. Разработайте структуру записи двоичного файла согласно варианту задания.
3. Разработайте функции для операций:

* преобразование тестовых данных в двоичный файл
* отображение всех записей двоичного файла
* манипулирование записями в двоичном файле: согласно операций, определенных в варианте

1. Удаление записи производить путем замены на последнюю запись.
2. Сохраните функции в новом модуле.
3. Разработать приложение, демонстрирующее выполнение всех операций, подключив к нему модуль с функциями.
4. Дополнительное задание. Реализуйте ускоренный доступ к данным по ключу, используя рассмотренные в курсе структуры: бинарное дерево поиска и хеш таблицу. Проведите анализ поиска бинарным деревом поиска и через хеш таблицу.

***Варианты***

|  |  |  |
| --- | --- | --- |
| № |  | |
| 1 | Структура записи | Читательский абонемент: номер читательского - целое пятизначное число, ФИО, Адрес |
| Доп. операция | 1. Поиск записи с заданным значением ключа. 2. Удаление найденного значения. |
| 2 | Структура записи | Счет в банке: номер счета 7 разрядное число, ФИО, Адрес |
| Доп. операция | * 1. Поиск записи с заданным значением ключа.   2. Обновить значение одного поля. |
| 3 | Структура записи | Владелец телефона: номер телефона – последовательность символов, адрес, ФИО |
| Доп. операция | 1. Сформировать текстовый файл из фамилий владельцев, чьи номера начинаются с введенных первых трех цифр(например, 434) 2. Удалить сведения о владельцах телефонов, которые начинаются с заданной цифры. |
| 4 | Структура записи | Владельцев автомобилей. номер машины, марка, сведения о владельце, сведения об угоне (логического типа). |
| Доп. операция | 1. Список автомобилей, числящихся в угоне. 2. Установить факт угона автомобиля с заданным номером. |
| 5 | Структура записи | Пациент поликлиники: номер карточки, код хронического заболевания, Фамилия лечащего врача |
| Доп. операция | * + - 1. Сформировать двоичный файл с записями о пациентах с заданным кодом заболевания.       2. Заменить фамилию, имя, отчество врача у указанных пациентов (список пациентов – это массив номеров карточек) |

# Практическая работа № 13

**Основные алгоритмы работы с графами**

***Вариант 1***

***Постановка задачи***

Составить программу реализации алгоритма Крускала построения остовного дерева минимального веса. Выбрать и реализовать способ представления графа в памяти.

Предусмотреть ввод с клавиатуры произвольного графа.

Разработать доступный способ (форму) вывода результирующего дерева на экран монитора.

Провести тестовый прогон программы для заданного графа в соответствии с индивидуальным заданием

***Таблица индивидуальных заданий***

|  |  |
| --- | --- |
| **№** | **Граф** |
| 14.1.1 |  |
| 14.1.2 |  |
| 14.1.3 |  |
| 14.1.4 |  |

***Вариант 2***

***Постановка задачи***

Составить программу реализации алгоритма Прима построения остовного дерева минимального веса.

Выбрать и реализовать способ представления графа в памяти.

Предусмотреть ввод с клавиатуры произвольного графа.

Разработать доступный способ (форму) вывода результирующего дерева на экран монитора.

Провести тестовый прогон программы для заданного графа в соответствии с индивидуальным заданием

***Таблица индивидуальных заданий***

|  |  |
| --- | --- |
| **№** | **Граф** |
| 14.2.1 |  |
| 14.2.2 |  |
| 14.2.3 |  |
| 14.2.4 |  |

***Вариант 3***

***Постановка задачи***

Составить программу нахождения кратчайших путей в графе заданным методом.

Выбрать и реализовать способ представления графа в памяти.

Предусмотреть ввод с клавиатуры произвольного графа.

Разработать доступный способ (форму) вывода результирующего дерева на экран монитора.

Провести тестовый прогон программы для заданного графа в соответствии с индивидуальным заданием

***Методы нахождения кратчайших путей***

|  |  |
| --- | --- |
| **№** | **Метод** |
| 14.3.1 | Построения дерева решений |
| 14.3.2 | Естественное слияние |

***Таблица индивидуальных заданий***

|  |  |
| --- | --- |
| **№** | **Граф** |
| 14.3.3 |  |
| 14.3.2 |  |
| 14.3.3 |  |

***Вариант 4***

***Постановка задачи***

Составить программу нахождения кратчайших путей в графе заданным методом.

Выбрать и реализовать способ представления графа в памяти.

Предусмотреть ввод с клавиатуры произвольного графа.

Разработать доступный способ (форму) вывода результирующего дерева на экран монитора.

Провести тестовый прогон программы для заданного графа в соответствии с индивидуальным заданием

***Методы нахождения кратчайших путей***

|  |  |
| --- | --- |
| **№** | **Метод** |
| 14.4.1 | Дейкстра |
| 14.4.2 | Флойда |
| 14.4.3 | Йена |
| 14.4.4 | Беллмана-Форда |

***Таблица индивидуальных заданий***

|  |  |
| --- | --- |
| **№** | **Граф** |
| 14.4.5 |  |
| 14.4.6 |  |

# Практическая работа №14

**Алгоритмы сжатия и кодирования данных**

***Цели работы***

Изучение алгоритма оптимального префиксного кодирования Хаффмана

Практическое применение алгоритма Хаффмана для сжатия данных

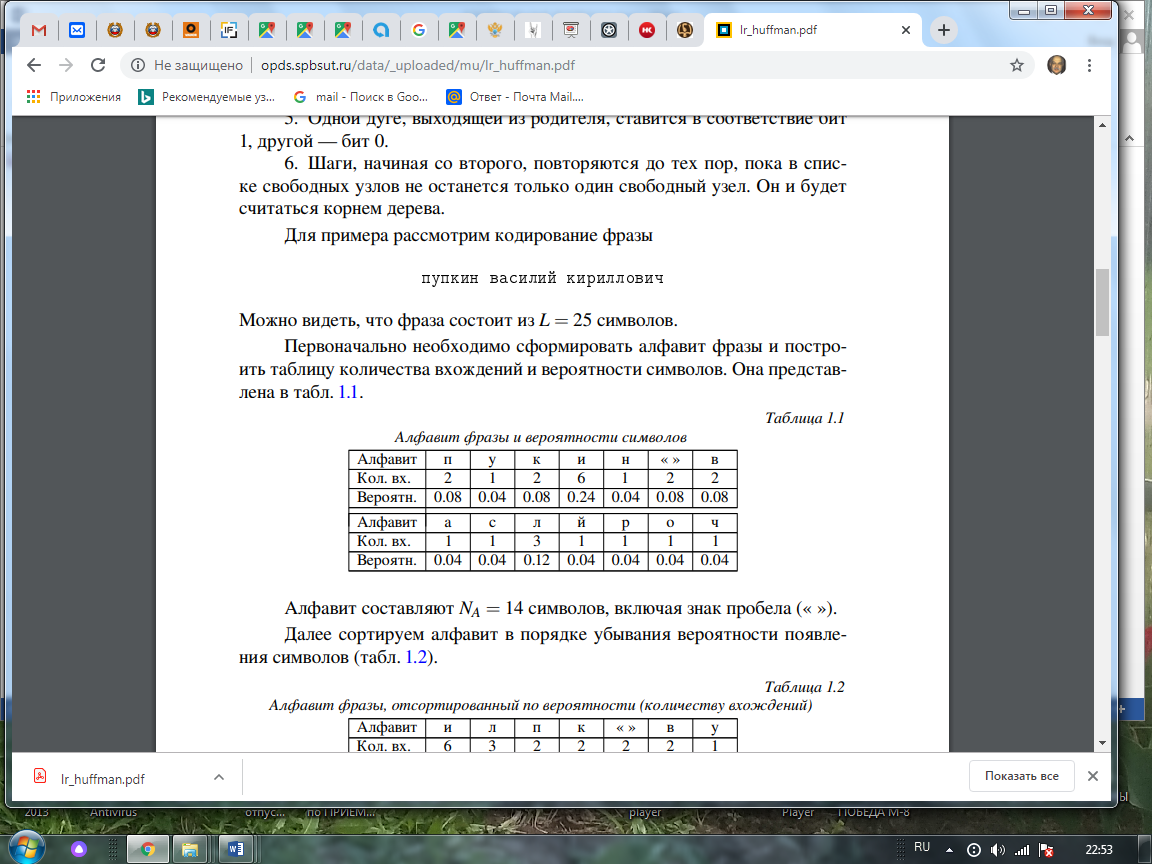
***Постановка задачи***

Провести кодирование исходной строки символов «Фамилия Имя Отчество» с использованием алгоритма Хаффмана. Исходная строка символов, таким образом, определяет индивидуальный вариант задания для каждого студента.

Для выполнения работы необходимо выполнить следующие действия:

1. Построить таблицу частот встречаемости символов в исходной строке символов для чего сформировать алфавит исходной строки и посчитать количество вхождений (частот) символов и их вероятности появления, например, для строки **пупкин василий кириллович** такая таблица будет иметь вид:

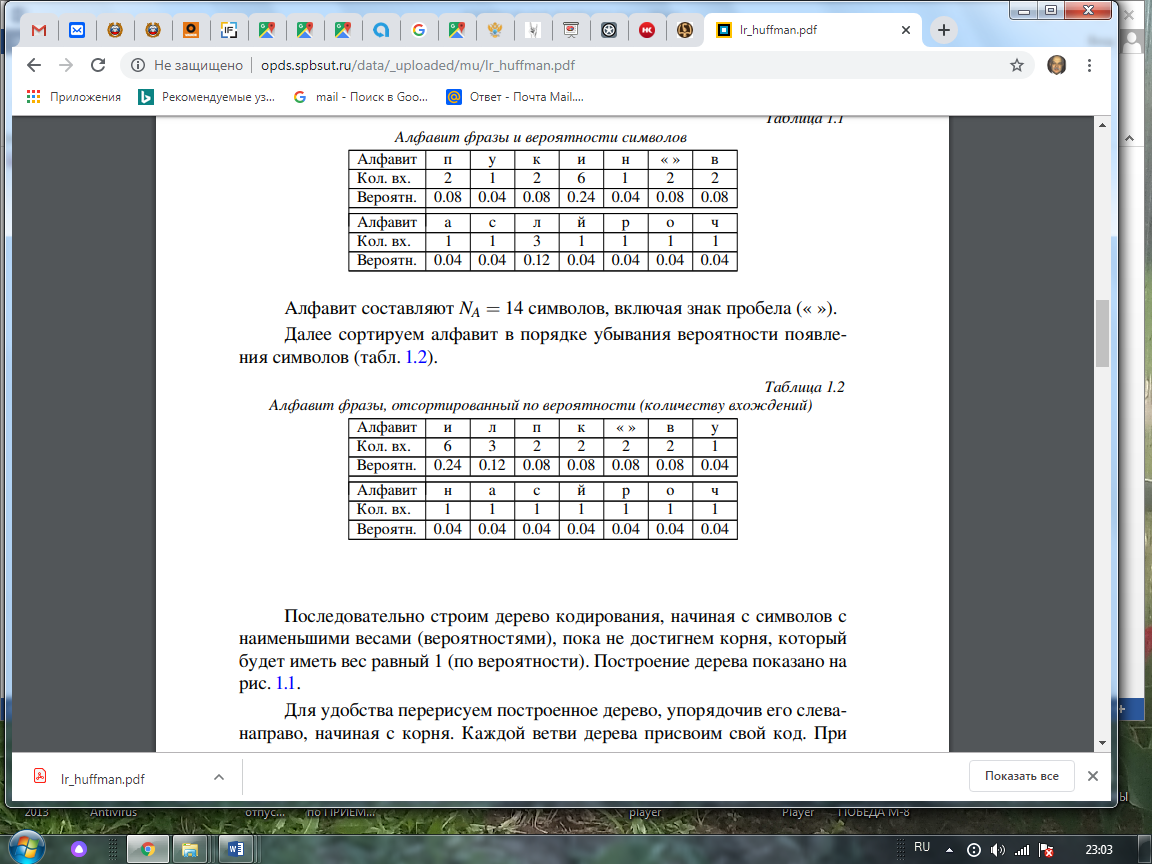
Таблица частот



(скобки < > обозначают пробел в исходной строке)

1. Отсортировать алфавит в порядке убывания частот появления символов по аналогии как показано ниже

Таблица отсортированных частот



1. Построить дерево кодирования Хаффмана, в данном примере оно имеет вид:

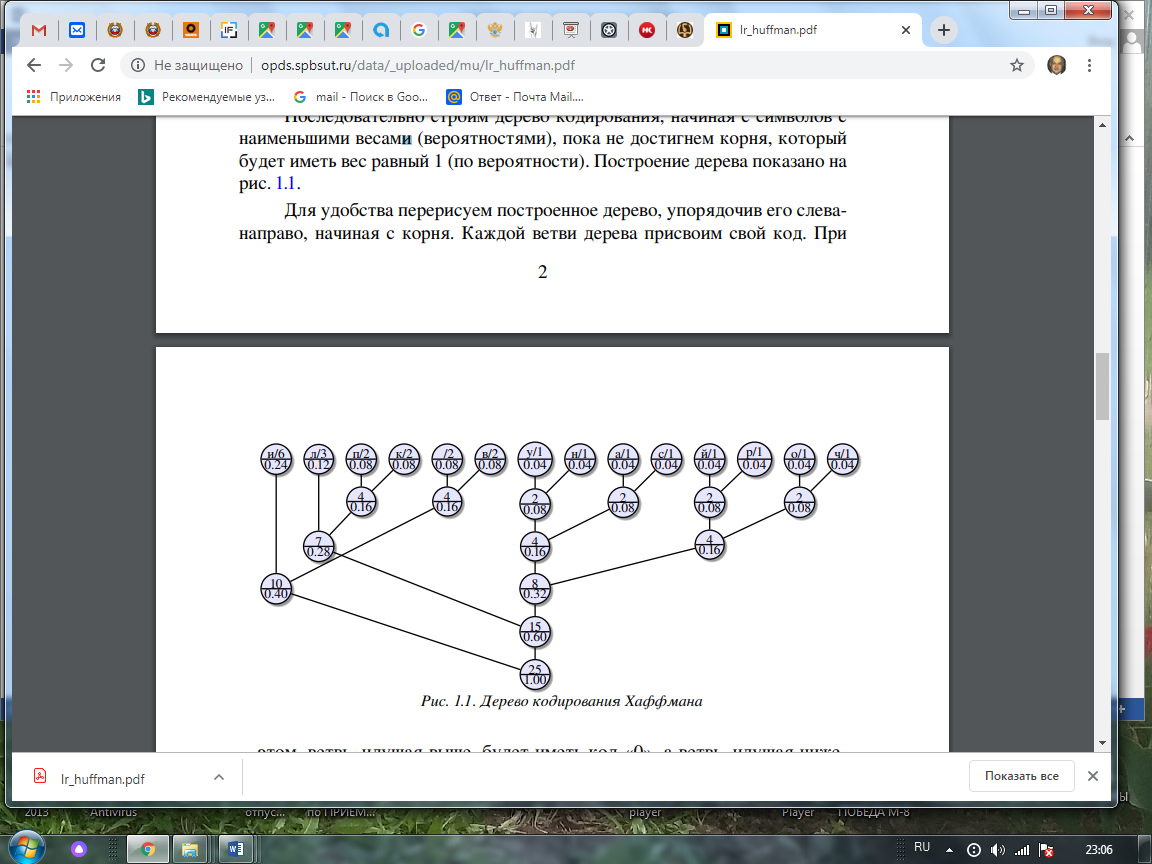


Рис.1 Дерево кодирования Хаффмана

1. Упорядочить построенное дерево слева-направо (при необходимости). Присвоить ветвям коды. Определить коды символов:

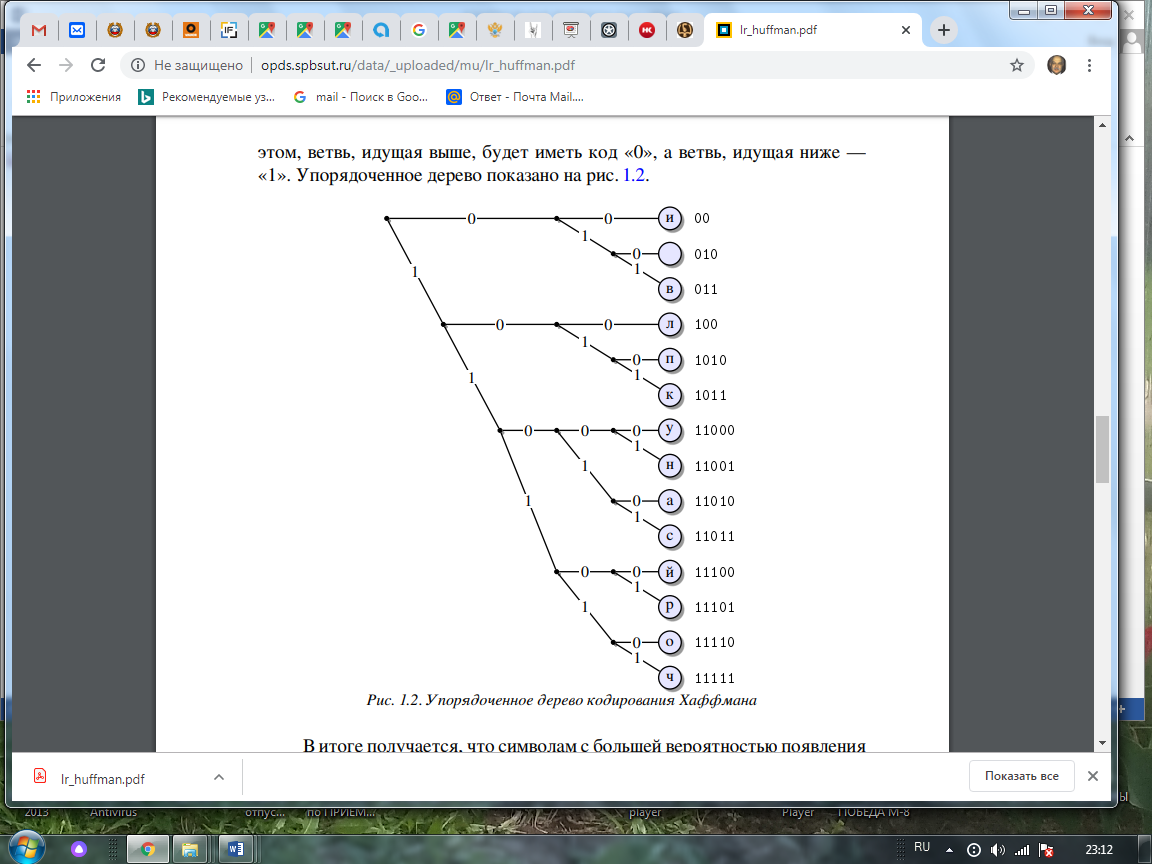
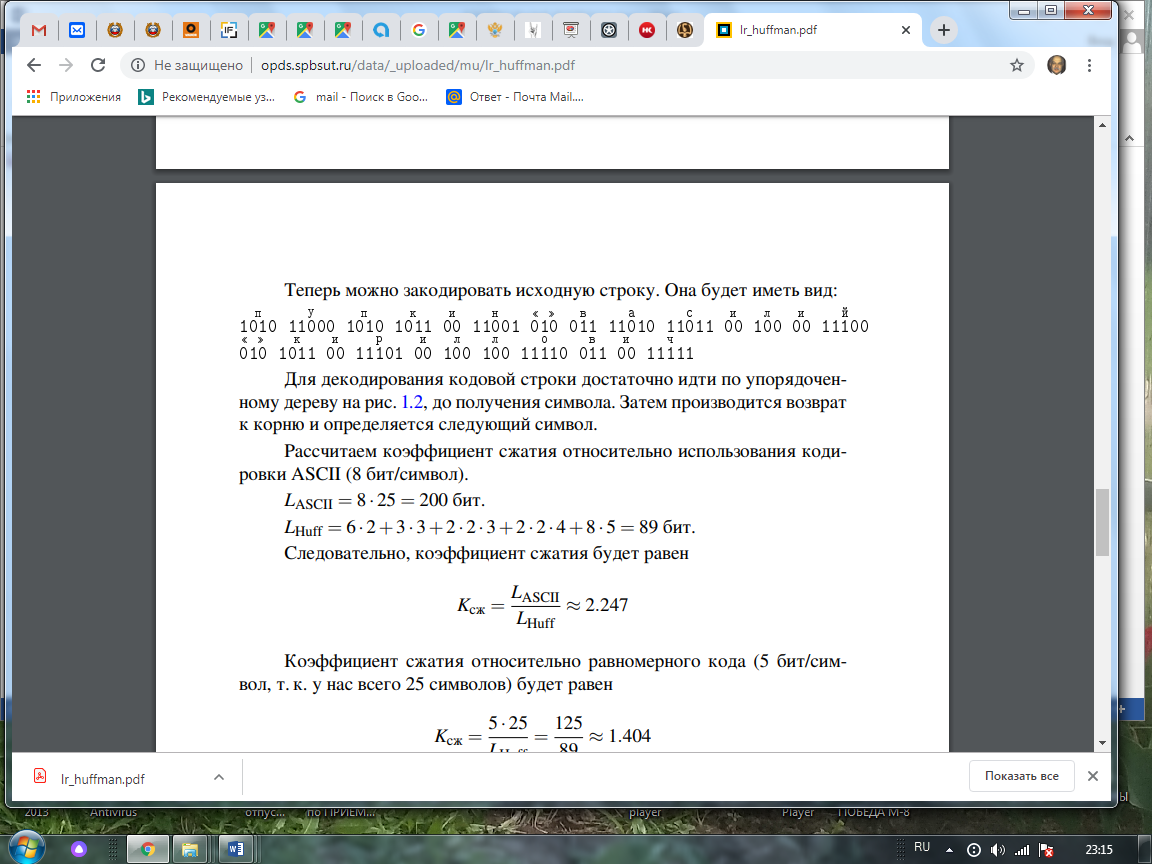


Рис.2 Упорядоченное дерево кодирования Хаффмана

1. Провести кодирование исходной строки по аналогии с примером:



Рассчитать коэффициенты сжатия относительно кодировки ASCII и относительно равномерного кода.

1. Рассчитать среднюю длину полученного кода и его дисперсию.По результатам выполненной работы сделать выводы и сформировать отчет.

# Практическая работа № 15

**Стратегии и методы построения алгоритмов**

***Постановка задачи***

Выбрать метод построения алгоритма задачи варианта. Реализовать программу. Разработать тесты. Провести тестирование. Провести эмпирическую (практическую) оценку вычислительной сложности алгоритма.

***Вариант 1***

Разработать процедуру оптимальной сборки изделия для случая *n* конвейеров, имеющих *m* рабочих мест каждый.

***Вариант 2***

Разработать процедуру оптимального способа расстановки скобок в произведении последовательности матриц, размеры которых равны (5,10,3,12,5,50,6), чтобы количество скалярных умножений стало минимальным (максимальным).

***Вариант 3***

Разработать процедуру решения задачи (на основе парадигмы жадного программирования) о выборе подмножества взаимно совместимых процессов, образующих множество максимального размера.

**Вариант 4**

Разработать процедуру сжатия данных на основе жадного алгоритма Хаффмана.

# Практическое занятие №16

**Метод линейного программирования**

**Постановка задачи**

Примените метод линейного программирования для решения задачи варианта. Разработайте программу реализации задачи варианта. Выполните тестирование. Провести эмпирическую (практическую) оценку вычислительной сложности алгоритма.

***Вариант 1.***

Приведите следующую задачу линейного программирования к стандартной форме.

Минимизировать ***2х***1 ***+*** 7*x*2 при условиях:

1. ***х***1 = 7; 2)3***х***1 ***+ х2* ≥**24 ; 3) ***х***1 ≥ 0; 4) ***х2* ≤** 0

***Вариант 2***

Преобразуйте следующую задачу линейного программирования в каноническую форму.

Максимизировать2***х***1 ***-*** 6*x*3  при условиях

***1)х***1 + *x*2 - *x*3 ≤ 7

2)3*x*1 - *x*2  ≥ 8

*3)-x*1 + 2 ***х2*** + 2х3 ≥ 0

4) *x*1 ,  ***х2*** , х3 ≥ 0

Какие переменные являются базисными, а какие небазисными?

***Вариант 3***

Покажите, что следующая задача линейного программирования является неразрешимой:

Максимизировать 3*x*1 - 2*x*2 при условиях

***1) х***1 + *x*2 ≤ 2

2) - 2*x*1 - 2х2 ≤ -10

3) *x*1 ,  ***х2*** ≥ 0

***Вариант 4***

Реализовать симплекс-метод для решения задачи о коммивояжере.

***Вариант 5***

Реализовать задачу о коммивояжере методом ветвей и границ.

***Вариант 6***

Реализовать задачу о рюкзаке методом ветвей и границ.

.

**Вопросы к зачету 1 семестр** (Вопросов может быть сколько угодно)

**Основные понятия курса**

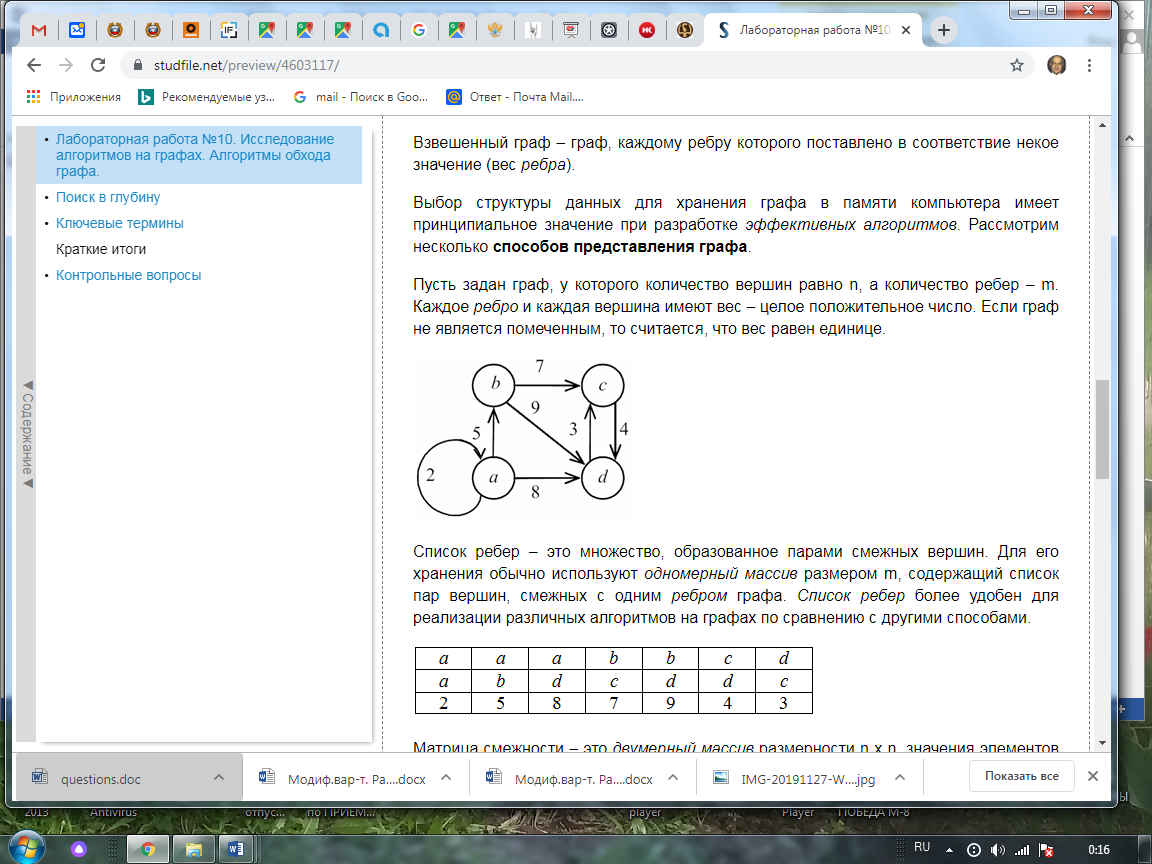
1. Охарактеризуйте каждый из трех уровней представления данных.
2. Что определяет типа данных в языках программирования?
3. Дайте определение структуре данных.
4. Как тип связан с понятием структура данных?
5. Определите структуру оперативной памяти на логическом уровне. Ячейка памяти.
6. Расскажите о структурах хранения данных, предоставляемых компьютерными технологиями.
7. Дайте определение линейной структуре данных.
8. данных, в нелинейных структурах данных.
9. Понятие абстракции и ее назначение в разработке сложных программ.
10. Дайте определение вычислительной сложности алгоритма.
11. Как определить вычислительную сложность алгоритма поиска в структуре данных?
12. Какой критерий используется при выборе структуры данных при реализации задачи?
13. Объясните правило асимптотической оценки сложности алгоритма –нотация О.
14. Что определяет функция в асимптотической сложности?
15. Сформулируйте правило суммы для расчета сложности алгоритма по символике O.
16. Сложность в худшем случае.
17. Сложность в среднем и лучшем случае.
18. Приведите пример алгоритма и оцените его сложность алгоритма: в лучшем, среднем и худшем случае.
19. Какие функции в оценке сложности называют доминирующими?
20. Охарактеризуйте класс P – задач.
21. Охарактеризуйте класс NP задач.
22. Классы функций, которые используются для оценки порядка роста времени выполнения.
23. Дайте определение рекурсивного объекта
24. Опишите структуру рекурсивного алгоритма
25. Как определить, что алгоритм имеет рекурсивную природу
26. Определение рекурсивного алгоритма
27. Что определяет шаг в рекурсии
28. Что определяет глубина рекурсии
29. Приведите рекуррентное соотношение для возведения числа х в степень n.
30. Разработайте рекурсивную функцию, которая считывает последовательность из n целых чисел и выводит сначала отрицательные числа, а затем положительные. Определите глубину и шаг рекурсии.
31. Какая стратегия сортировки называется обменной сортировкой?
32. В чем заключаются усовершенствования алгоритма простого обмена и каковы его достоинства?
33. Какую идею реализует алгоритм сортировки слиянием?
34. Почему алгоритм быстрой сортировки Хоара называют обменной сортировки с разделением?
35. На чем основано семейство методов сортировки, называемое сортировкой посредством выбора?
36. Какие усовершенствования сортировки простым выбором Вам знакомы?
37. Какова вычислительная сложность алгоритма сортировки простым выбором.
38. Определите понятие устойчивости алгоритма сортировки. Какой из простых алгоритмов сортировки является устойчивым?
39. Какой критерий является основным для алгоритмов внешней сортировки?
40. Алгоритмы поиска в массивах
41. Сущность и реализация алгоритма последовательного поиска с барьером.
42. Опишите алгоритм упорядочивания элементов одномерного массива методом бинарного поиска.
43. Почему нельзя использовать бинарный и интерполяционный алгоритмы поиска для списков?
44. Укажите виды обхода бинарных деревьев. Какой из них используется в алгоритме поиска?
45. В чем состоит методика оценки вычислительной сложности алгоритмов поиска?
46. В чем состоят особенности задач внешнего поиска и сортировки?
47. Сравните алгоритмы Кнута-Морриса-Пратта и последовательного поиска.
48. Алгоритмы внешней сортировки
49. Что в математике считается линейным списком?
50. Определите линейный список с позиций структуры данных.
51. Определите отношения в линейных структурах данных.
52. Какая структура считается однородной, неоднородной?
53. Определите список базовых операций над линейным списком как структурой данных.
54. Приведите структуры языка программирования для реализации линейного списка.
55. Опишите динамический однонаправленный список.
56. Расскажите, используя графическое представление динамического однонаправленного списка, как выполняется операция вставки в любую позицию.
57. Расскажите, используя графическое представление динамического однонаправленного списка, как выполняется операция удаления узла в любой позиции.
58. Дана последовательность целых чисел. Сформируйте линейный однонаправленный список по правилу: справа от вершины все отрицательные числа последовательности, а слева положительные и нули. Реализуйте функцию.
59. Определите через нотацию O(f(n)) сложность выполнения операции вставки элемента в линейный список реализованный на массиве и на динамическом однонаправленном списке.
60. Какие списки называют циклическими?
61. Какую операцию над линейным однонаправленным списком может оптимизировать реализация динамического списка на структуре с головным элементом?
62. Как определить начало в циклическом однонаправленном списке?
63. Приведите функцию вставки узлов в упорядоченный двунаправленный список. Этот список с самого начала формируется как упорядоченный. Список упорядочивается по значению ключа. Информационная часть узла любого типа.
64. Дайте определение структуре данных стек.
65. Чем стек отличается от линейного списка?
66. Объясните на примере применение стека для вычисления значения без скобочного арифметического выражения, представленного в строковом формате.
67. Дано арифметическое выражение, содержащее круглые скобки. Напишите функцию, которая в выражении проверяет баланс скобок.
68. Приведите постфиксную форму следующего выражения:
69. 5+7-4\*(7\*2+8/4)/3
70. Можно ли в одном массиве можно реализовать два стека?
71. Предложите структуру для реализации нескольких стеков в одном массиве. Какие проблемы при использовании этой структуры могут возникнуть?
72. Приведите определение структуры данных Очередь?
73. Расскажите о способах реализации очереди.
74. Как организация очереди на однонаправленном динамическом списке с головным элементом может уменьшить сложность операции добавления узла до O(1)?
75. Опишите АТД Очередь с базовым набором операций.
76. Расскажите особенность реализации круговой очереди.
77. Как определить конец круговой очереди?
78. Объясните алгоритмы вставки и удаления в очереди с приоритетом.
79. Можно ли 10 очередей произвольного размера реализовать на одномерном массиве?
80. Объясните алгоритм вычисления значения выражения – операнды однозначные числа:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| + | + | 5 | 4 | / | \* | 3 | 8 | 6 |

1. Расскажите об особенностях структуры данных Дек.
2. Даны две последовательности чисел a1<=a2<=...<=an и b1<=b2<=...<=bm. Образовать из них новую последовательность чисел так, чтобы она тоже была неубывающей. Дополнительный массив не использовать.
3. Пусть a1,a2,...,ai - упорядоченная последовательность, т.е. a1<=a2<=...<=ai. Берётся следующее число ai+1 и вставляется в последовательность так, чтобы новая последовательность была также возрастающей. Процесс производится до тех пор, пока все элементы от i+1 до n не будут перебраны. Определите, как называется этот алгоритм сортировки.
4. Составьте процедуры, реализующие алгоритм сортировки массивов данных методом простого обмена (Exchange sort) и методом Шелла. Рассмотрите по шагам работу этих процедур на примере конкретного массива из 8 элементов.
5. Составьте процедуры, реализующие алгоритм сортировки массивов данных методом простого обмена (Exchange sort) и шейкерным методом. Рассмотрите по шагам работу этих процедур на примере конкретного массива из 8 элементов. Проведите оценку вычислительной сложности алгоритма для обоих вариантов.
6. Составьте процедуру, реализующую алгоритм Хоара быстрой сортировки (Quick sort). Рассмотрите по шагам работу этой процедуры на примере конкретного массива из 8 элементов. Проведите оценку вычислительной сложности алгоритма.
7. Составьте процедуру, реализующую алгоритм турнирной сортировки. Рассмотрите по шагам работу этой процедуры на примере конкретного массива из 8 элементов. Проведите оценку вычислительной сложности алгоритма.
8. Составьте процедуру, реализующую алгоритм пирамидальной сортировки. Рассмотрите по шагам работу этой процедуры на примере конкретного массива из 8 элементов. Проведите оценку вычислительной сложности алгоритма.
9. Составьте процедуру поиска заданного элемента в исходном массиве (используя любой алгоритм поиска). Часть массива до этого элемента отсортировать по возрастанию, используя алгоритм Шелла. Часть массива после найденного элемента отсортировать по убывания, используя алгоритм Хоара быстрой сортировки.

**Вопросы к экзамену 2 семестр** (Вопросов может быть сколько угодно)

1. Асимптотический анализ вычислительной сложности алгоритмов. Анализ вычислительной сложности в худшем, среднем и наилучшем случаях. Асимптотические обозначения O, Θ, Ω.
2. Связные списки. Виды. Основные операции, вычислительная сложность операций.
3. Стек. Основные операции. Способы реализации. Реализовать Функции вставки для каждого способа.
4. Очередь. Основные операции. Реализация на основе циклического массива.
5. Очередь. Способы реализации. АТД очереди с набором основных операций. Разработать функцию по доказательству балансировки круглых скобок в арифметическом выражении.
6. Очередь. Определить АТД Очередь. Разработать функцию, вычисления значения выражения, представленного в префиксной форме.
7. Словари. Хеш-таблицы. Методы разрешения коллизий.
8. Хеш-таблица. Реализация цепного хеширования. Проблемы и пути их устранения. Оценка сложности операций вставки, удаления, поиска.
9. Хеш-таблица. Реализация на основе открытого адреса. Алгоритм двойного хеширования. Понятие Кластера.
10. Хеш-таблица. Реализация на основе открытого адреса. Проблема поиска ключа после операции удаления в кластере ключа поиска и алгоритм устранения проблемы.
11. Очереди с приоритетами. Бинарные двоичные кучи.
12. Определение линейного списка и его отличие от стека и очереди. Способы реализации. Реализовать функцию удаления узла из списка.
13. Линейный список и его отличие от стека и очереди. Способы реализации. Анализ вычислительной сложности реализаций. Реализовать функцию удаления узла из списка.
14. Двунаправленные списки. Их применение для реализации списков. Функция удаления узла из линейного списка, реализованного на двунаправленном списке.
15. Определите реализацию структуры данных Дек на языке высокого уровня: структура узла, операции вставки, удаления, вывода.
16. Циклический линейный список. Способы реализации. Реализуйте операцию вставки узла в конец списка.
17. Файловые структуры данных. Определение файла на физическом и логическом уровнях. Сортировка файлов.
18. В – дерево для поиска в файле. Алгоритм включения ключей в дерево.
19. Индексы для файлов. Реализация на хеш таблице. Привести пример.
20. Рекурсивные алгоритмы. Рекуррентные соотношения на примере ханойской башни. Виды рекурсивных алгоритмов.
21. Рекурсивные алгоритмы. Шаг, глубина рекурсии. Основное свойство алгоритма, к который можно реализовать рекурсивно.
22. 1. Проведите на конкретном примере n=10 сравнительный анализ алгоритма бинарного поиска в массиве A(n) с алгоритмом интерполяционного поиска.
23. 1. Напишите две процедуры поиска заданного элемента в массиве, используя алгоритм бинарного поиска и алгоритм интерполяционного поиска. Рассмотрите работу этих алгоритмов на конкретном примере массива из 10 элементов.
24. 2. Напишите процедуру поиска заданного элемента с использованием хэш-таблицы. Рассмотрите работу этого алгоритма на конкретном примере массива из 10 элементов.
25. 3. Напишите процедуру поиска заданного элемента с использованием бинарного дерева поиска. Рассмотрите работу этого алгоритма на конкретном примере массива из 10 элементов.
26. Алгоритм Дейкстры поиска кратчайшего пути в графе. Вычислительная сложность алгоритма.
27. Назовите все возможные формы представления графа в памяти. Опишите нижеприведенный граф, используя все формы его представления в памяти



1. Алгоритм Крускала построения минимального остовного дерева. Вычислительная сложность алгоритма.
2. Алгоритм Флойда-Уоршелла поиска кратчайших путей в графе между всеми вершинами в графе.
3. Задана последовательность вершин v0,v1,...,vp, которая определяет путь в графе G(V,E). Напишите процедуру определения длина этого пути.
4. Напишите процедуры, реализующие алгоритмы обхода графа в глубину и в ширину.
5. Какая последовательность называется последовательностью Фибоначчи. Напишите процедуру генерации N чисел последовательности Фибоначчи. Составьте процедуру поиска элемента в массиве с использованием чисел Фибоначчи.
6. Составьте процедуру интерполяционного поиска заданного элемента в массиве. Подробно рассмотрите работу алгоритма на примере массива из 10 элементов. Проведите оценку вычислительной сложности алгоритма.
7. Напишите две процедуры поиска заданного элемента в массиве, используя алгоритм бинарного и интерполяционного поиска. Подробно рассмотрите работу этих алгоритмов на примере массива из 10 элементов. Проведите оценку вычислительной сложности алгоритмов.
8. Составьте процедуру, реализующую алгоритм поиска с использованием бинарного дерева поиска. Подробно на конкретном примере массива из 10 элементов рассмотрите принцип построения бинарного дерева поиска и принцип работы алгоритма. Проведите оценку вычислительной сложности алгоритма.
9. Алгоритм Кнута–Морриса–Пратта поиска строки в тексте. Принцип работы. Вычислительная сложность в наилучшем и наихудшем случаях. Проведите оценку вычислительной сложности алгоритма. Сравните с алгоритмом последовательного поиска. Напишите процедуру, реализующую алгоритм Кнута–Морриса–Пратта поиска строки в тексте.
10. Составьте процедуры, реализующие алгоритм алгоритм последовательного поиска строки в тексте и алгоритм Бойера-Мура. Проведите сравнительный анализ алгоритмов принципу работы и по вычислительной сложности.
11. Алгоритм шейкерной сортировки массива из n элементов по возрастанию их значений. Условие Айверсона. Вычислительная сложность алгоритма. Сравните этот алгоритм с алгоритмом простого обмена. В чем заключается улучшение алгоритма шейкерной сортировки по сравнению с алгоритмом простого обмена.
12. Алгоритмы сортировки массивов данных методом простого обмена (Exchange sort) и алгоритм Шелла. Вычислительная сложность алгоритмов. Сравните эти алгоритмы по вычислительной сложности. В чем заключается улучшение алгоритма Шелла по сравнению с алгоритмом простого обмена.
13. Алгоритмы сортировки массивов данных методом простого обмена (Exchange sort) и быстрой сортировки Хоара (Quick sort). Вычислительная сложность этих алгоритмов.
14. Алгоритмы сортировки массивов данных методом простых вставок (Insertion sort) и методом слияния (Merge sort). Вычислительная сложность этих алгоритмов.
15. Алгоритм Хоара быстрой сортировки (Quick sort). Укажите оценки вычислительной сложности алгоритма в наилучшем, среднем и наихудшем случаях. Приведите примеры последовательностей для каждого из этих случаев.
16. Алгоритм турнирной сортировки. Последовательность построения бинарного дерева выбора. Принцип исключения элементов из бинарного дерева выбора. Вычислительная и емкостная сложности алгоритма.
17. Алгоритм пирамидальной сортировки (Heap sort). Свойства пирамидального дерева. Понятие триады. Возможные варианты преобразования триад при выполнении алгоритма.
18. Алгоритмы турнирной и пирамидальной сортировки. Отличия в принципах использования двоичного дерева в этих алгоритмах. Вычислительная сложность алгоритмов. Преимущества и недостатки алгоритмов.
19. Задан массив, состоящий из n случайных элементов. Переставить в обратном порядке элементы массива, расположенные между его минимальным и максимальным элементами и отсортировать полученный массив по убыванию. Исходный, рабочий (промежуточный) и результирующий массивы вывести на экран.
20. Составьте процедуру упорядочивания массива из n элементов по возрастанию их значений, используя алгоритм Хоара быстрой сортировки (Quick sort).
21. Составьте процедуру упорядочивания массива из n элементов по убыванию их значений, используя алгоритм пирамидальной сортировки.
22. Составьте процедуру упорядочивания массива из n элементов по возрастанию их значений, используя алгоритм турнирной сортировки.

**АННОТАЦИЯ**

**к рабочей программе дисциплины**

**Б1.О.18 «Структуры и алгоритмы обработки данных»**

Направление подготовки **09.03.04«Программная инженерия»**

Профиль подготовки **«Системная и программная инженерия»**

1. **Цель освоения дисциплины**

Дисциплина «Структуры и алгоритмы обработки данных» имеет своей целью способствовать формированию у обучающихся общепрофессиональных компетенций: ОПК-6.1, ОПК-6.2, ОПК-6.1, в соответствии с требованиями ФГОС ВО по направлению подготовки бакалавров 09.03.04 «Программная инженерия» с учетом специфики профиля подготовки – «Системная и программная инженерия».

1. **Место дисциплины в структуре основной профессиональной образовательной программы бакалавриата**

Дисциплина «Структуры и алгоритмы обработки данных» является обязательной дисциплиной базовой части блока «Дисциплины (модули)» учебного плана направления подготовки бакалавров 09.03.04 «Программная инженерия» с профилем подготовки – «Системная и программная инженерия».

Поддержка упражнений практических занятий, примеров реализации алгоритмов осуществляется средствами объектно-ориентированных языков программирования, поддерживающих указатели, библиотеки коллекций и классы–контейнеры.

В результате изучения дисциплины обучающийся должен:

***Знать:***

* технологию разработки программного обеспечения с использованием абстрактных типов данных;
* три уровня представления данных: внешний, логический, физический;
* структуры хранения данных в оперативной памяти, предлагаемые архитектурой компьютерных систем;
* особенности организации структур данных: линейный список, стек, очередь, дек и основные алгоритмы их создания и обработки;
* базовые структуры данных языка С++ и технологии создания нестандартных структур хранения данных, используемых для хранения, оптимизации поиска в больших наборах данных: линейные списки, бинарные деревья, сбалансированное бинарное дерево поиска, AVL-деревья, хеш-таблица, В-дерево, граф, файл;
* алгоритмы поиска данных, алгоритмы модификации структур данных;
* алгоритмы сортировки массивов данных, хранящихся в оперативной памяти и файлах;
* возможности стандартной библиотеки шаблонов (STL) языка С++ по применению шаблонов структур данных и алгоритмов;
* методику реализации рекурсивных алгоритмов: стратегию разработки «Разделяй и властвуй», метод ветвей и границ, метод динамического программирования;
* алгоритмы обхода иерархических структур данных, сетей;
* методы сжатия данных, в частности, алгоритмы Фано и Хаффмана;
* знать методику асимптотической оценки сложности алгоритмов ***O***-, ***Ω***- и ***Θ****-*нотации.

***Уметь:***

* проводить анализ данных задачи и выбор структуры (базовой или разработанной пользователем) данных для оптимизации алгоритмов решения задач, связанных с хранением, поиском данных;
* выполнять проектирование абстрактного типа данных для каждой задачи приложения, требующей использование структуры данных;
* применять методику рекурсивных алгоритмов, методы «разделяй и властвуй», метод ветвей и границ, метод динамического программирования при разработке программного обеспечения;

***Владеть:***

* навыками определения структур данных внешнего (пользовательского уровня), выбора оптимальной структуры реализации в программе;
* навыками структуризации сложной программы на основе объектно–ориентированного языка программировании;
* практическими навыками применения основных базовых типов данных объектно–ориентированных языков программирования и аппарата объектно–ориентированного языка для создания пользовательского типа данных – реализации абстрактного типа;
* навыками практической реализации структур данных в задачах обработки больших наборов данных средствами языка объектно-ориентированного программирования;
* навыками применения классов-контейнеров и классов коллекций объектно–ориентированных языков программирования в задачах обработки больших наборов данных.
* алгоритмами поиска данных и методиками оптимизации алгоритма поиска во внешних структурах данных;
* навыками расчета времени выполнения алгоритма, программы.

1. **Общая трудоемкость дисциплины** составляет 6 зачетных единиц (216 академических часа). Дисциплина читается два семестра: во втором и третьем. Форма промежуточной аттестации во втором семестре – зачет, в третьем – экзамен.