ПРИЛОЖЕНИЕ

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Рязанский государственный радиотехнический университет имени В.Ф. Уткина»

КАФЕДРА «ЭЛЕКТРОННЫЕ ВЫЧИСЛИТЕЛЬНЫЕ МАШИНЫ»

МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

«МЕТОДЫ ПРОМЫШЛЕННОГО ПРОГРАММИРОВАНИЯ»

Специальность

27.05.01 Специальные организационно-технические системы

Специализация

Информационные технологии и программное обеспечение в специальных организационно-технических системах

Квалификация (степень) выпускника — инженер-системотехник

Форма обучения — очная, очно-заочная

1. МЕТОДИЧЕСКИЕ УКАЗАНИЯ К КУРСОВОЙ РАБОТЕ

1. Цель курсового проектирования и исходные данные

В курсовой работе ставится задача разработки, отладки и тестирования приложения Windows в системе программирования Delphi (или C++), реализующего алгоритмы обработки данных согласно заданному варианту.

Исходными данными для работы являются:

1. Тексты заданий согласно варианту (см. п. 2);
2. Операционная среда программирования: персональный компьютер, операционная система Windows (любой модификации), система программирования на языке Object Pascal (или C++);
3. Метод проектирования: модульное объектно-ориентированное программирование.

2. Варианты заданий

Описания вариантов заданий приведены в приложении. Номер варианта соответствует номеру студента в журнале группы или выбирается из предлагаемого множества вариантов случайным образом. Возможны инициативные задания, которые утверждаются по согласованию с преподавателем.

3. Основные технические условия и требования

По заданному варианту для каждой его части следует выполнить разработку приложения Windows по следующим пунктам:

1. Анализ требований и уточнение спецификаций
   - 1.1. Выполнить анализ задачи с целью определения основных состояний интерфейсной части программы.
   - 1.2. Для каждого состояния определить внешнее представление (экранную форму и требуемые визуальные компоненты), т. е. вид окна приложения в каждом случае.
   - 1.3. Выполнить объектную декомпозицию интерфейсной и предметной частей программы.
2. Проектирование программы
   - 2.1. Логическое проектирование интерфейсной части программы (разработка структуры классов).
     - 2.1.1. Открыть новый проект и создать окно главной формы.
     - 2.1.2. С помощью инспектора объектов настроить параметры формы и используемых компонент.
     - 2.1.3. Исходя из возможных вариантов (списка событий задействованных компонент) определить события, по которым будет происходить смена состояний интерфейсной части. Одновременно определить, какие обработчики событий должны выполнять основные операции при смене этих состояний.
   - 2.2. Физическое проектирование программы (разбивка программы на модули).

     Выполняется системой программирования Delphi (или C++) автоматически.
3. Разработка алгоритмической и программной части
   - 3.1. На основании результатов п.2.1.3 разработать алгоритмы обработчиков всех событий и представить их в виде блок-схем с соответствующими текстовыми пояснениями.
   - 3.2. Подготовить исходных тексты обработчиков всех событий и текст модуля программы в целом.
   - 3.3. Выполнить компиляцию и отладку программы.
   - 3.4. Разработать справочную подсистему и программу инсталляции.
4. Экспериментальная часть
   - 4.1. Осуществить инсталляцию программ на жестком диске компьютера и их запуск средствами ОС из основного меню системы.
   - 4.2. Выполнить экспериментальную проверку работоспособности программ каждого задания на нескольких контрольных наборах (примерах) исходных данных.

4. Содержание пояснительной записки к курсовой работе

Записка должка быть оформлена как программный документ по ГОСТам ЕСПД, и содержать следующие разделы:

1. Введение
2. Техническое задание

5 Дополнительные требования к содержанию пояснительной записки

1. Раздел «Техническое задание» оформляется по ГОСТ 19.201-78.

2. Раздел «Описание разработанной программы» подготавливается согласно ГОСТ 19.402-78.

3. Подраздел 3.3 должен содержать блок-схемы алгоритмов модулей программы с **подробными** текстовыми описаниями. Элементы блок-схемы не должны включать обозначения идентификаторов программы.

4. Раздел 4 «Программа и методика испытаний» должен соответствовать ГОСТу 19.301-79. В подразделе 4.6 должны быть приведены описания используемых методов отладки и тестирования, все подготовленные тесты и контрольные примеры, а также результаты выполнения всех тестов с соответствующими выводами и результаты контрольных примеров.

5. Раздел «Руководство программиста» оформляется в соответствии с ГОСТом 19.504-79, а «Руководство оператора» – по ГОСТ 19.505-79.

6. Оформление основного текста выполняется в соответствии с ГОСТ 7.32-2017 «Отчет о научно-исследовательской работе. Структура и правила оформления».

7. Оформление библиографического списка в соответствии с ГОСТ 7.1-2003 «Библиографическая запись».

8. В приложении приводятся исходные тексты разработанных программ с подробными комментариями.

6 Порядок сдачи и защиты курсовой работы

Нормативный **срок сдачи** КР – предзачетная неделя семестра.

Студент допускается к защите после проверки пояснительной записки преподавателем. При обнаружении серьезных ошибок и упущений записка может быть возвращена на доработку.

Защита проекта включает:

1. Краткий доклад студента по результатам выполненного проектирования.
2. Демонстрацию работающих программ на компьютере с пояснениями.
3. Ответы на вопросы преподавателя.

Результаты курсового проектирования оцениваются с учетом:

a) наличия работающих программ и соответствия их п.3;
b) качества выполнения пояснительной записки и соответствия ее п.4;
c) уровня ответов студента.

ЛИТЕРАТУРА

а) основная

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

2. Ахо А., Хопкрофт Дж., Ульман Дж. Структуры данных и алгоритмы. М.: Вильяме, 2001. 384 с.

3. Бентли Д. Жемчужины творчества программистов. М.: Радио и связь, 1990.

4. Вирт Н. Алгоритмы + структуры данных = программы. М.: Мир, 1985.

5. Вирт Н. Алгоритмы и структуры данных. М: Мир, 1989. 360 с.

6. Грин Д., Кнут Д. Математические методы анализа алгоритмов. М: Мир, 1987.

7. Гудман С, Хидетниеми С. Введение в разработку и анализ алгоритмов. М.: Мир, 1981.

8. Дейкстра Э. Дисциплина программирования. М: Мир, 1978.

9. Кнут Д. Е. Искусство программирования: В 3 т. М.: Вильяме, 2000.

10. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: Построение и анализ. М.: МЦНМО, 2001.

7 Варианты заданий

Вариант выполнения курсовой работы включает три части. Содержанием каждой части является одно из заданий. Описания вариантов приведены в таблице вариантов, описания заданий представлены ниже.

Таблица вариантов

Номер варианта	Номера заданий		
	Часть 1	Часть 2	Часть 3
1	15	1	2
2	14	15	3
3	13	2	4
4	12	14	15
5	11	3	14
6	10	13	13
7	9	4	5
8	8	12	6
9	7	5	7
10	6	11	12
11	5	6	11
12	4	10	10
13	3	7	1
14	2	9	8
15	1	8	9
16	15	14	9
17	10	11	8
18	8	7	13
19	6	9	12
20	14	13	6

Часть 1. Задачи на графах

1. Дан ориентированный граф с N вершинами (N<50). Вершины и дуги окрашены в цвета с номерами от 1 до M (M<6). Указаны две вершины, в которых находятся фишки игрока и конечная вершина. Правило перемещения фишек: игрок может передвигать фишку по дуге, если ее цвет совпадает с цветом вершины, в которой находится другая фишка; ходы можно делать только в направлении дуг графа; поочередность ходов необязательна. Игра заканчивается, если одна из фишек достигает конечной вершины. Написать программу поиска кратчайшего пути до конечной вершины, если он существует.

2. Задан неориентированный граф. При прохождении по некоторым ребрам отдельные (определенные заранее) ребра могут исчезать или появляться. Написать программу поиска кратчайшего пути из вершины с номером q в вершину с номером w.

3. Заданы два числа N и M (20<M<N<150), где N — количество точек на плоскости. Написать программу построения дерева из M точек так, чтобы оно было оптимальным.

Примечание. Дерево называется оптимальным, если сумма всех его ребер минимальна. Все ребра — это расстояния между вершинами, заданными координатами точек на плоскости.

4. Даны два числа N и M. Написать программу построения графа из N вершин и M ребер. Каждой вершине ставится в соответствие число ребер, входящих в нее. Граф должен быть таким, чтобы сумма квадратов этих чисел была минимальна.

5. Задан ориентированный граф с N вершинами, каждому ребру которого приписан неотрицательный вес. Написать программу поиска простого цикла, для которого среднее геометрическое весов его ребер было бы минимально.

Примечание. Цикл называется простым, если через каждую вершину он проходит не более одного раза, петли в графе отсутствуют.

6. Внутрь квадрата с координатами левого нижнего угла (0,0) и координатами верхнего правого угла (100, 100) поместили N $(1<N<30)$ квадратиков. Написать программу поиска кратчайшего пути из точки (0,0) в точку (100,100), который бы не пересекал ни одного из этих квадратиков. Ограничения:

- длина стороны каждого квадратика равна 5;
- стороны квадратиков параллельны осям координат;
- координаты всех углов квадратиков — целочисленные;
- квадратики не имеют общих точек.

Примечание. Строится граф из начальной, конечной точек и вершин квадратиков (ребра не должны пересекать квадраты). Затем ищется кратчайший путь, например, с помощью алгоритма Дейкстры.

7. Задан неориентированный граф с N вершинами, пронумерованными целыми числами от 1 до N. Написать программу, которая последовательно решает следующие задачи:

- выясняет количество компонент связности графа;
- находит и выдает все такие ребра, что удаление любого из них ведет к увеличению числа компонент связности;
- определяет, можно ли ориентировать все ребра графа таким образом, чтобы получившийся граф оказался сильно связным;
- ориентирует максимальное количество ребер, чтобы получившийся граф оказался сильно связным;
- определяет минимальное количество ребер, которые следует добавить в граф, чтобы ответ на третий пункт был утвердительным.

8. Задан ориентированный граф с N (1<N<30) вершинами, пронумерованными целыми числами от 1 до N. Разработать программу, которая подсчитывает количество различных путей между всеми парами вершин графа.

9. Задан ориентированный граф с N вершинами, пронумерованными целыми числами от 1 до N. Разработать программу, которая подсчитывает количество различных путей между всеми парами вершин графа.

Входные данные. Входной файл $input.txt$ содержит количество вершин графа N (1<N<30) и список дуг графа, заданных номерами начальной и конечной вершин.

Выходные данные. В выходной файл *output.txt* вывести матрицу $N \times N$, элемент (i,j) которой равен числу различных путей, ведущих из вершины i в вершину j или -1, если существует бесконечно много таких путей.

10. Дан ориентированный граф. Вершинам приписаны веса. Начальная и конечная вершины заданы. Требуется найти путь между этими вершинами с максимально возможным суммарным весом. Путь должен удовлетворять следующим условиям:

- по каждой дуге разрешается пройти один раз;
- если в вершину попадаем по входящей дуге, то к суммарному весу вес этой вершины прибавляется;
- если в вершину попадаем по выходящей дуге, то из суммарного веса вес этой вершины вычитается.

Входные данные. Входной файл *input.txt* содержит данные в следующей последовательности:

- $N\, x_1\, x_2 \ldots x_N$ — количество вершин графа и их веса ($1 \le N \le 30$, $1 < x_i < 30000$);
- *b,e* — номера начальной и конечной вершин;
- M — количество дуг;
- *i1, j1* — номера вершин, соединяемых первой дугой;
- i_2, j_2 — номера вершин, соединяемых второй дугой;
- i_M, j_M — номера вершин, соединяемых дугой с номером M.

Выходные данные. В выходной файл *output.txt* требуется вывести максимальный вес пути и путь, на котором он достигается. В случае, если решение не существует, выходной файл должен содержать единственную строку *«решения нет».*

11. Дан взвешенный (ребрам приписаны веса) связный граф *G=(V,E).*

Обозначим через *D{υ,u}* минимальное расстояние между вершинами $\upsilon, u \in V$. Величина *D(G) = Max(D(υ,u))* называется диаметром графа. Величина *R(υ) = Max(D(υ,u))* максимальным удалением в графе от вершины *υ.* Величину *R(G) = Min(R(υ))* называют радиусом графа. Любая вершина *t€V,* такая, что *R(t)=R(G*), называется центром графа.

Разработать программу поиска диаметра, радиуса и центров графа.

Примечание. Первый шаг решения заключается в формировании матрицы минимальных расстояний между всеми парами вершин графа.

12. Согласно теореме Д. Кенига (1936 г.) для двудольности графа необходимо и достаточно, чтобы он не содержал циклов нечетной длины. Определить, является ли заданный граф двудольным.

Примечание. При поиске в ширину вершины графа помечаются метками 0, 1, 2 и т. д. Первой вершине, с которой начинается просмотр, приписывается метка 0, вершинам, связанным с ней, — метка 1 и т. д. Разобьем граф после просмотра на два подграфа: подграф X включает вершины с четными метками, подграф Y — с нечетными. Если оба подграфа пусты — исходный граф является двудольным.

- Задача Штейнера на графах. В связном взвешенном графе *G* с выделенным подмножеством вершин *U (U € V)* требуется найти связный подграф *T*, удовлетворяющий двум условиям:
  - множество вершин подграфа *T* содержит заданное подмножество *U*;
  - граф *T* имеет минимальный вес среди всех подграфов, удовлетворяющих первому условию.

Примечание. Очевидно, что искомый подграф является деревом, оно называется деревом Штейнера. Задача сводится к нахождению дерева минимального веса в подграфах графа *G,* множество вершин которых содержит *U.* Эффективных алгоритмов решения задачи неизвестно, поэтому наиболее простой состоит в переборе вариантов при небольших значениях числа вершин.

13. Задача о пяти ферзях. На шахматной доске 8x8 расставить наименьшее число ферзей так, чтобы каждая клетка доски была под боем.

Примечание. Построить граф для данной доски с учетом правил перемещения ферзя. Всякой искомой расстановке соответствует наименьшее доминирующее множество в графе.

15. Для заданного ориентированного графа с N вершинами разработать:

- структуру представления графа;
- процедуру заполнения графа из N узлов;
- процедуру поиска всех путей от заданной вершины ко всем остальным методом в глубину;

– процедуру поиска кратчайших путей от заданной вершины ко всем остальным методом в ширину;
– процедуру построения остового дерева графа.

Часть 2. Реализация списковых и древовидных структур данных

1. Для кольцевого односвязного неупорядоченного списка реализовать операции:
   – включение нового узла вслед за *k*-м;
   – вывод списка на экран.
2. Для кольцевого односвязного неупорядоченного списка реализовать операции:
   – поиск узла с заданным значением;
   – исключение *k*-го узла.
3. Для кольцевого односвязного неупорядоченного списка реализовать операции:
   – сохранение содержимого списка в типизированном файле;
   – чтение содержимого списка из типизированного файла.
4. Для кольцевого односвязного неупорядоченного списка реализовать операции:
   – очистка списка;
   – сортировка списка.

5. Реализовать построение дерева с произвольным числом потомков каждого узла. Разработать процедуру подсчета количества узлов дерева и количества потомков заданного узла.

6. Реализовать построение дерева с произвольным числом потомков каждого узла. Разработать процедуру отображения поддерева с наибольшим числом узлов.

7. Реализовать построение дерева с произвольным числом потомков каждого узла. Разработать процедуру отображения поддерева с наименьшим числом узлов.

8. Реализовать построение дерева с произвольным числом потомков каждого узла. Разработать процедуру отображения поддерева с задаваемым числом узлов.

9. Реализовать построение бинарного дерева поиска и его отображение с помощью обхода в ширину. Разработать процедуру добавления нового узла.

10. Реализовать построение бинарного дерева поиска и его отображение с помощью алгоритма прямого обхода. Разработать процедуру удаления узла с минимальным значением.

11. Реализовать построение бинарного дерева поиска и его отображение с помощью алгоритма обхода в ширину. Разработать процедуру удаления узла с максимальным значением.

12. Реализовать построение бинарного дерева поиска и его отображение с помощью алгоритма прямого обхода. Разработать процедуру поиска узла с максимальным значением.

13. Реализовать построение бинарного дерева поиска и его отображение с помощью алгоритма обхода в ширину. Разработать процедуру поиска узла с минимальным значением.

14. Реализовать построение бинарного дерева поиска и его отображение с помощью алгоритма прямого обхода. Разработать процедуру определения суммы значений, находящихся в листьях дерева.

15. Реализовать построение бинарного дерева поиска и его отображение с помощью алгоритма обхода в ширину. Разработать процедуру определения суммы значений, находящихся в *n* задаваемых узлах дерева.

Часть 3. Разработка хеш-таблиц и функций хеширования

1. Разработать приложение, которое использует хеш-таблицу для организации прямого доступа к элементам массива данных. Разработать процедуру поиска задаваемого элемента.

Для предотвращения коллизий использовать метод цепочек.

2. Разработать приложение, которое использует хеш-таблицу для организации прямого доступа к элементам массива данных. Разработать процедуру вставки задаваемого элемента. Для предотвращения коллизий использовать метод линейного апробирования.

3. Разработать приложение, которое использует хеш-таблицу для организации прямого доступа к элементам массива данных. Разработать процедуру удаления задаваемого элемента. Для предотвращения коллизий использовать метод квадратичного апробирования.

4. Разработать приложение, которое использует хеш-таблицу для организации прямого доступа к элементам массива данных. Разработать процедуру удаления задаваемого элемента. Для предотвращения коллизий использовать метод двойного хеширования.

5. Разработать приложение, которое использует хеш-таблицу для организации прямого доступа к элементам массива данных. Разработать процедуру удаления задаваемого элемента. Для предотвращения коллизий использовать метод цепочек.

6. Разработать приложение, которое использует хеш-таблицу для организации прямого доступа к элементам массива данных. Разработать процедуру вставки задаваемого элемента. Для предотвращения коллизий использовать метод цепочек.

7. Разработать приложение, которое использует хеш-таблицу для организации прямого доступа к элементам массива данных. Разработать процедуру поиска задаваемого элемента. Для предотвращения коллизий использовать метод двойного хеширования.

8. Разработать приложение, которое использует хеш-таблицу для организации прямого доступа к элементам массива данных. Разработать процедуру поиска задаваемого элемента. Для предотвращения коллизий использовать метод линейного апробирования.

9. Разработать приложение, которое использует хеш-таблицу для организации прямого доступа к элементам массива данных. Разработать процедуру поиска задаваемого элемента. Для предотвращения коллизий использовать метод квадратичного апробирования.

10. Разработать приложение, которое использует хеш-таблицу для организации прямого доступа к элементам массива данных. Разработать процедуру вставки задаваемого элемента. Для предотвращения коллизий использовать метод цепочек.

11. Разработать приложение, которое использует хеш-таблицу для организации прямого доступа к элементам массива данных. Разработать процедуру вставки задаваемого элемента. Для предотвращения коллизий использовать метод квадратичного апробирования.

12. Разработать приложение, которое использует хеш-таблицу для организации прямого доступа к элементам массива данных. Разработать процедуру вставки задаваемого элемента. Для предотвращения коллизий использовать метод двойного хеширования.

13. Разработать приложение, в котором создаётся таблица имен лексического анализатора программы. Таблица содержит характеристики каждого имени: номер, тип, число символов, указатель начала цепочки символов и др. Разработать процедуру поиска задаваемого имени. Для предотвращения коллизий использовать метод линейного апробирования.

14. Разработать приложение, в котором создаётся таблица имен лексического анализатора программы. Таблица содержит характеристики каждого имени: номер, тип, число символов, указатель начала цепочки символов и др. Разработать процедуру вставки нового имени. Для предотвращения коллизий использовать метод квадратичного апробирования.

15. Разработать приложение, в котором создаётся таблица имен лексического анализатора программы. Таблица содержит характеристики каждого имени: номер, тип, число символов, указатель начала цепочки символов и др. Разработать процедуру удаления задаваемого имени. Для предотвращения коллизий использовать метод двойного хеширования.

Примечание. Для заданий 1 – 12 рекомендуется использовать исходный массив, содержащий сведения, аналогичные списку физических лиц, например: порядковый номер, фамилия, имя, отчество, адрес и т.д.

Список вопросов по курсовой работе и практическим занятиям

1. Сформулируйте понятие линейного списка.
2. Приведите логическую структуру односвязного линейного списка.
3. Поясните содержание основных операций односвязными списками.
4. В чем состоит особенность структуры двусвязных списков.
5. Поясните алгоритмы выполнения операций с двусвязными списками.
6. В чем состоит особенность структуры кольцевых списков.
7. Поясните алгоритмы выполнения операций с кольцевыми списками.
8. Поясните логическую структуру и принцип работы стека.
9. Приведите содержание основных операций со стеком.
10. Поясните логическую структуру и принцип работы очереди.
11. Опишите основные операции с очередями.
12. Поясните логическую структуру и принцип работы дека.
13. Приведите содержание основных операций с деком.
14. Поясните логическую структуру и основные элементы древовидных структур.

15. Что такое степень вершин дерева и степень дерева.
16. Поясните основные способы обхода деревьев.
17. Поясните логическую структуру бинарных деревьев.
18. Чем отличается сбалансированное бинарное дерево от почти сбалансированного.
19. В чем состоят основные операции, осуществляемые с бинарными деревьями
20. Объясните основные свойства красно-черных деревьев.
21. Поясните основные типы и понятия внешней сортировки данных.
22. Объясните на примере алгоритм внешней сортировки простым слиянием.
23. В чем заключаются особенности алгоритма внешней сортировки естественным слиянием.
24. Сформулируйте в общем виде алгоритм поиска данных.
25. Поясните алгоритм последовательного линейного поиска.
26. В чем особенности поиска с барьером.
27. Поясните алгоритм бинарного поиска.
28. Что такое двоичное упорядоченное дерево?
29. Сформулируйте основные операции с упорядоченными деревьями.
30. Что такое случайные деревья поиска?
31. Поясните понятие идеально сбалансированного дерева.
32. Сформулируйте понятие АВЛ-дерева поиска.
33. Поясните операцию малого правого (левого) вращения дерева.
34. Поясните операцию большого правого (левого) вращения дерева.
35. Поясните операцию добавления нового элемента в сбалансированное по АВЛ дерево.
36. Поясните операцию удаления элемента из сбалансированного по АВЛ дерева.
37. Сформулируйте понятие графа.
38. Поясните способы представления графов в памяти ЭВМ.
39. Поясните алгоритм поиска в глубину.
40. Поясните алгоритм поиска в ширину.
41. Сформулируйте алгоритм Дейкстры поиска кратчайшего пути в графе.
42. Сформулируйте алгоритм Флойда поиска кратчайшего пути в графе.
43. Сформулируйте переборный алгоритм поиска кратчайшего пути в графе.
44. Сформулируйте понятия хеширования данных, хеш-функции и хеш-таблицы.
45. Поясните основные требования к хеш-функциям.
46. Какими свойствами должны быть наделены хеш-таблицы?
47. Поясните основные способы получения хеш-функций.
48. Что такое коллизии?
49. Сформулируйте известные способы устранения влияния коллизий.