Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Новосибирский государственный технический университет»
Кафедра вычислительной техники

Паспорт
курсовой работы

по дисциплине «Программирование», 3 семестр

1. Методика оценки.

Выполнение курсовой работы (далее – КР) является обязательным видом самостоятельной работы студента по дисциплине, предусмотренным учебным планом.

Основной целью выполнения КР является формирование компетенций и соотнесенных с ними индикаторов по дисциплине «Программирование», 3 посредством закрепления, углубления и обобщения знаний, полученных студентами за время теоретического обучения и прохождения практик, а также выработка навыков самостоятельного применения знаний и навыков для творческого решения конкретных задач. Выполнение курсовой работы должно способствовать подготовке их к решению более сложной задачи - выполнению выпускной квалификационной работы.

Задачами выполнения курсовой работы является овладение студентами рациональными приёмами сбора, обработки, систематизации информации, применения компьютерных технологий в области разработки программ, применения нормативно-законодательной базы и умения оценивать эффективность реализуемых проектов и программ в региональной социально-экономической политике.

Тематика КР соответствует профилю (направленности) подготовки, формируются преподавателями в начале семестра и утверждаются распоряжением заведующего кафедрой. Количество тем КР достаточно для обеспечения, каждого обучающегося.

Выполнение студентами КР начинается с ознакомления с примерной тематикой. Закрепление тем КР за студентами и назначение научных руководителей производится распоряжением заведующего кафедрой.

Курсовая работа выполняется индивидуально

Структура курсовой работы:

1. Развернутое описание задания.
2. Структурное описание разработки. Описание используемых структур данных, форматов данных, сопровождаемое необходимыми графическими иллюстрациями (рисунками). Содержательное описание основных алгоритмов, их особенностей (4-5 стр.).
3. Функциональное описание. Содержательное описание типов данных, структур данных (шаблоны классов), фрагментов нетривиальных алгоритмов. Для описания использовать смешанное текстово-формальное представление программы с включением этих элементов в связный текст изложения материала.
4. Описание пользовательского интерфейса (программа выполняется в консоли).
5. Описание работы программы на контрольных примерах (например, на большом текстовом файле), временные характеристики программы на простых типах для одного из методов: сортировка, балансировка, поиск в неупорядоченной последовательности.
6. Выводы. Описание того, что было сделано. Ограничения (по памяти, по времени), ошибки, особенности проектирования.
7. Список используемой литературы.

8. Приложение: текст программы с комментариями по существу алгоритма и структур данных.

Список литературы оформляется в соответствии с библиографическими требованиями в алфавитном порядке и включает от 5 до15 источников (книг, статей разных авторов, интернет-источников, документов), которые были изучены при выполнении работы.

Требования к оформлению:

Объем КР до *25* страниц машинописного текста формата А4. Шрифт Times New Roman, 12. Нумерация страниц сквозная, в нижней части листа по центру арабскими цифрами. КР должна быть отредактирована, не содержать орфографических, синтаксических и стилевых ошибок.

Законченная курсовая работа предоставляется для проверки в электронном виде в срок, установленный преподавателем. Преподаватель оценивает качество КР с учетом теоретического и практического содержания, достижения ее целей и задач.

Курсовая работа проверяется руководителем работы, который дает письменное заключение по работе — рецензию.

Если при выполнении КР были допущены ошибки, то работа возвращается студенту для исправления выявленных недочетов и затем вновь предоставляется руководителю для проверки. При положительном результате оценивания студент распечатывает работу, передает на кафедру и защищает до сессии в назначенное преподавателем время.

Защита КР проходит публично перед группой студентов.

По результатам защиты студенту выставляется оценка в соответствии с критериями, приведенными в п. 2 настоящего Паспорта.

2. Уровни сформированности компетенций и критерии оценки

В соответствии с балльно-рейтинговой системой НГТУ курсовая работа дисциплине «Программирование», 3 имеет максимальную оценку 100 баллов.

Курсовая работа выполнена **на продвинутом** уровне, если:

- она выполнена в полном соответствии с заданием, отличается глубиной проработки всех разделов содержательной части, текстовая часть оформлена с соблюдением установленных правил;
- руководитель характеризует деятельность студента положительно (в частности, отмечает его инициативу, самостоятельность, систематичность работы на всех этапах выполнения работы);
- в докладе исчерпывающе, последовательно, четко и логически правильно изложена суть работы и ее основные результаты;
- студент свободно владеет теоретическим материалом, безошибочно применяет его при решении задач, сформулированных в задании;
- в докладе суть работы и ее основные результаты представлены исчерпывающе, последовательно, четко и логически правильно; на все вопросы студент дал обстоятельные и аргументированные ответы, убедительно защищал свою точку зрения;
- компетенции и соотнесенные с ними индикаторы, закрепленные за дисциплиной, по которым выполняется курсовая работа сформированы в полном объеме.

Оценка за выполнение КР составляет *100-87 баллов.*

Курсовая работа выполнена на **базовом** уровне, если:

- соответствует заданию, отличается глубиной проработки всех разделов содержательной части, текстовая часть оформлена с соблюдением установленных правил;
- руководитель характеризует деятельность студента положительно, но с незначительными замечаниями;
- в докладе правильно изложена суть работы и ее основные результаты;
- студент достаточно твердо усвоил теоретический материал и может самостоятельно его применять;
- в докладе суть работы и ее основные результаты представлены полно; на все вопросы студент дал ответы, но их полнота и аргументированность недостаточны;
- компетенции и соотнесенные с ними индикаторы, закрепленные за дисциплиной, по которым выполняется курсовая работа сформированы с небольшими пробелами и соответствуют базовому уровню.

Оценка за выполнение КР составляет *86-73 балла*.

Курсовая работа выполнена **на пороговом** уровне, если:
- выполнена в основном правильно, но без необходимой проработки некоторых разделов;
- в докладе упущены некоторые принципиальные моменты содержательной части работы;
- в докладе представлены суть работы и ее основные результаты; ответы на вопросы вызвали существенные затруднения;
- компетенции и соотнесенные с ними индикаторы, закрепленные за дисциплиной, по которым выполняется курсовая работа сформированы с пробелами и соответствуют пороговому уровню.

Оценка за выполнение КР составляет *72-50 баллов.*

Курсовая работа считается **не выполненной,** если студентом не проработаны важные разделы исследования, допущены принципиальные ошибки, не исправленные после замечаний руководителя курсовой КР . Студент не допущен к защите курсовой работы компетенции и соотнесенные с ними индикаторы, закрепленные за дисциплиной, по которым выполняется курсовая работа не сформированы.
Оценка составляет *менее 49 баллов*.

3. Шкала оценки.

В общей оценке по дисциплине баллы за работы учитываются в соответствии с правилами балльно-рейтинговой системы, приведенными в рабочей программе дисциплины.

Курсовая работа по дисциплине считается успешно выполненной, если сумма полученных баллов составляет от 100 до 50 баллов включительно.

4. Перечень тем курсовой работы

1. Простые структуры данных в памяти

Задан объект со списком полей. Реализуемая структура данных по заданию должна быть полностью динамической и также реализована в виде класса. Обязательные операции – добавление, удаление, включение и извлечение по логическому номеру, сортировка, включение с сохранением порядка, загрузка и сохранение объектов в бинарном файле, поиск по различным критериям. Предполагается, что операции сравнения хранимых объектов переопределены стандартным образом (в виде операций <,> и т.д.).

Программа должна реализовывать указанные выше действия.

Протестировать структуру данных на простом типе данных (например, int, double) и сложном, из выбранного задания. Программа тестирования должна содержать **меню**, обеспечивающее выбор операций.

Содержание объекта данных.

1. Список студентов факультета. Основная таблица: Фамилия, дата рождения, дата поступления, дата отчисления, адрес, группа. Редактирование с выбором по группе и шаблону фамилии.
2. Доска объявлений. Категория объявления: куплю-продам, рубрика: транспорт, недвижимость, материалы и т.д. Текст объявления – строка (абзац) переменной длины, дата подачи, контактный телефон. Поиск по заданной рубрике и по шаблону искомого слова в тексте объявления. Сортировка по дате подачи.
3. Движение поездов. Номер поезда, дни недели отправления, время отправления, время в пути (часов, минут). Станция отравления, назначения, промежуточные станции. Промежуточные станции запоминаются в динамическом массиве номеров станций. Поиск всех поездов, следующих до заданной станции.
4. Учет посещаемости индивидуальных и коллективных занятий. Фамилия, группа, динамический массив дат посещения занятий. Операции добавления даты посещения для выбранного слушателя и для выбранной группы слушателей. Вывод списка студентов, посещавших занятия по заданной дате. Добавление и удаление групп. Сортировка по количеству посещений и по фамилии.
5. Справочник склада. Наименование товара, категория товара, количество, дата поступления, цена и процент торговой надбавки. Сортировка по всем параметрам. Составление фактуры: выбор нескольких товаров, количества (с уменьшением его на складе), подсчет общей суммы и торговой надбавки.
6. База данных междугородных переговоров. Город, код города, тариф, дата, время переговоров, продолжительность в минутах, телефон абонента, дата оплаты. Сортировка по дате переговоров. Вывод списка неоплаченных переговоров и суммы оплаты (дата оплаты пуста).
7. База данных пункта проката. Предмет проката, стоимость проката, дата, время получения и возврата предмета, фамилия. Несданные предметы – пустая дата возврата. Подсчет суммы оплаты за сданный предмет, подсчет дохода пункта за заданный год/месяц.

Вид структуры данных:

1. Односвязный список структур**.**
2. Двусвязный циклический список структур**.**
3. Динамический массив структур с уплотнением при удалении и с расширением при переполнении.
4. Дерево, вершина которого содержит два указателя на поддеревья, счетчик количества вершин и структурированную переменную **(начальный балл 4).**

2. Последовательный двоичный файл

Двоичный файл содержит записи переменной длины (см. **простые структуры данных**). Формат записи предполагает ее переменную размерность. Реализовать набор операций над записями без загрузки одновременно всей структуры данных в память (поэлементная загрузка СД): добавление строки, извлечение, удаление, вставка по логическому номеру и редактирование (обновление) строки, вставка с сохранением

порядка, сортировка, постраничный просмотр, сжатие файла. При изменении размерности записи она переписывается в конец файла.

Программа должна реализовывать указанные выше действия. Протестировать структуру данных на простом типе данных (например, int, double) и сложном, из выбранного по заданию. Программа тестирования должна содержать **меню**, обеспечивающее выбор операций.

Содержание объекта данных.

8. Список студентов факультета. Основная таблица: Фамилия, дата рождения, дата поступления, дата отчисления, адрес, группа. Редактирование с выбором по группе и шаблону фамилии.
9. Доска объявлений. Категория объявления: куплю-продам, рубрика: транспорт, недвижимость, материалы и т.д. Текст объявления – строка (абзац) переменной длины, дата подачи, контактный телефон. Поиск по заданной рубрике и по шаблону искомого слова в тексте объявления. Сортировка по дате подачи.
10. Движение поездов. Номер поезда, дни недели отправления, время отправления, время в пути (часов, минут). Станция отравления, назначения, промежуточные станции. Промежуточные станции запоминаются в динамическом массиве номеров станций. Поиск всех поездов, следующих до заданной станции.
11. Учет посещаемости индивидуальных и коллективных занятий. Фамилия, группа, динамический массив дат посещения занятий. Операции добавления даты посещения для выбранного слушателя и для выбранной группы слушателей. Вывод списка студентов, посещавших занятия по заданной дате. Добавление и удаление групп. Сортировка по количеству посещений и по фамилии.
12. Справочник склада. Наименование товара, категория товара, количество, дата поступления, цена и процент торговой надбавки. Сортировка по всем параметрам. Составление фактуры: выбор нескольких товаров, количества (с уменьшением его на складе), подсчет общей суммы и торговой надбавки.
13. База данных междугородных переговоров. Город, код города, тариф, дата, время переговоров, продолжительность в минутах, телефон абонента, дата оплаты. Сортировка по дате переговоров. Вывод списка неоплаченных переговоров и суммы оплаты (дата оплаты пуста).
14. База данных пункта проката. Предмет проката, стоимость проката, дата, время получения и возврата предмета, фамилия. Несданные предметы – пустая дата возврата. Подсчет суммы оплаты за сданный предмет, подсчет дохода пункта за заданный год/месяц.

Вид структуры данных:

1. Односвязный список структур**.**
2. Двусвязный циклический список структур**.**
3. Динамический массив структур с уплотнением при удалении и с расширением при переполнении.
4. Дерево, вершина которого содержит два указателя на поддеревья, счетчик количества вершин и структурированную переменную

Шаблон иерархической структуры данных в памяти

Для заданной двухуровневой структуры данных, содержащей указатели на объекты (или сами объекты) - параметры шаблона, разработать полный набор операций (добавление, включение и извлечение по логическому номеру, сортировка, включение с сохранением порядка, загрузка и сохранение строк в бинарном файле, балансировка –

выравнивание размерностей структур данных нижнего уровня). Предполагается, что операции сравнения хранимых объектов переопределены стандартным образом (в виде операций <,> и т.д.). Программа должна использовать шаблонный класс с объектами-строками и реализовывать указанные выше действия над текстом любого объема, загружаемого из файла.

Программа должна реализовывать указанные выше действия. Протестировать структуру данных. Программа тестирования должна содержать **меню**, обеспечивающее выбор операций.

1. Шаблон структуры данных – двухуровневый массив указателей на объекты. Массив верхнего уровня – статический, массивы нижнего уровня – динамические, размерность – параметр конструктора, последовательность указателей в каждом массиве ограничена NULL. Если после включения указателя массив заполняется полностью, то создается еще один массив указателей, в который переписывается половина указателей из старого.

2. Шаблон структуры данных – двухуровневый массив указателей на объекты с типом – параметром шаблона. Массив верхнего уровня – статический. Массивы нижнего уровня – динамические, размерность каждого следующего в 2 раза больше размерности предыдущего, последовательность указателей в каждом массиве ограничена NULL.

3. Шаблон структуры данных – односвязный список, каждый элемент которого содержит статический массив указателей на объекты. Последовательность указателей в каждом массиве ограничена NULL. При переполнении текущего массива указателей создается новый элемент списка, в который переписывается половина указателей из текущего.

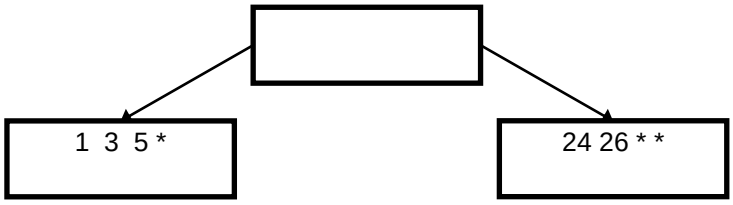
4. Шаблон структуры данных – односвязный список, каждый элемент которого содержит динамический массив указателей на объекты. Размерность массива указателей в каждом последующем элементе списка в 2 раза больше, чем в предыдущем. Последовательность указателей в каждом массиве ограничена NULL. При переполнении текущего массива указателей последний указатель переносится в следующий элемент списка.

5. Шаблон структуры данных – двусвязный циклический список, каждый элемент которого содержит статический массив указателей на объекты. Последовательность указателей в каждом массиве ограничена NULL. При переполнении текущего массива указателей создается новый элемент списка, в который переписывается половина указателей из текущего.

6. Шаблон структуры данных – односвязный список, каждый элемент является заголовком односвязного списка. Элемент списка второго уровня содержит указатель на объект. (При включении элемента последним в список предусмотреть ограничение длины текущего списка и переход к следующему).

7. Шаблон структуры данных – массив указателей на заголовки списков. Элемент списка содержит указатель на объект. (При включении элемента последним в список предусмотреть ограничение длины текущего списка и переход к следующему).

8. Шаблон структуры данных – дерево. Конечная вершина дерева содержит статический массив указателей на объекты (ограниченный NULL-указателем), промежуточная вершина содержит счетчик вершин дерева и два указателя на правое и левое поддерево. Значения в дереве не упорядочены. Естественная нумерация значений производится путем обхода дерева по принципу “левое поддерево – вершина – правое поддерево”. Если при включении указателя в найденный массив последний переполняется, он становится промежуточной вершиной. **(начальный балл 4,5).**



9. Шаблон структуры данных – двусвязный список (нециклический), каждый элемент списка содержит указатель на объект. Для ускорения процедуры обхода структуры данные имеется динамический массив указателей на каждый 10-ый элемент списка (0,10,20 и т.д.).

4. Класс - структура данных в двоичном файле

Класс двоичного файла, производный от fstream. Двоичный файл содержит заданную структуру данных. Программа должна представлять собой простой текстовый редактор, использующий структуру данных для промежуточного хранения редактируемого файла. Должны быть реализованы операции создания и заполнения двоичного файла из заданного текстового файла и сохранения содержимого двоичного файла в текстовый файл, а также операции добавления, удаления, вставки и редактирования строк.

Программа должна реализовывать указанные выше действия. Протестировать структуру данных. Программа тестирования должна содержать **меню**, обеспечивающее выбор операций.

1. Класс – двоичный файл, производный от fstream. Файл содержит массив указателей на строки, представленные записями переменной длины: целый счетчик и последовательность символов строки (без 0-ограничителя). Формат файла: в начале – файла, размерность массива указателей, текущее количество указателей, адрес (смещение) массива указателей в файле.
2. Класс – двоичный файл, производный от fstream. Файл содержит односвязный список строк в формате записей переменной длины со счетчиком. Формат файла: в начале файла – заголовок списка. Элемент списка содержит файловый указатель на следующий и саму строку в виде записи переменной длины.
3. Класс – двоичный файл, производный от fstream. Файл содержит двусвязный циклический список строк в формате записей переменной длины со счетчиком. Формат файла: в начале файла – заголовок списка. Элемент списка содержит файловые указатели на следующий и предыдущий и саму строку в виде записи переменной длины.
4. Класс – двоичный файл, производный от fstream. Файл содержит дерево, конечная вершина которого содержит строку, представленную записью переменной длины: целый счетчик и последовательность символов строки (без 0-ограничителя). Промежуточная вершина содержит указатели на правое и левое поддерево и число вершин в поддереве. Формат файла: в начале – файла, указатель на корневую вершину.

5. Шаблон - структура данных в двоичном файле

Класс двоичного файла, производный от fstream. Двоичный файл содержит заданную структуру данных с типом хранимых объектов - параметром шаблона. Программа должна выполнять операции создания файла, просмотра, добавления, удаления, обновления и сортировки объектов заданного при генерации типа.

Реализовать две версии программы для различных хранимых объектов (например, текстовых строк и объектов класса из лабораторной работы). Предполагается, что операции сравнения хранимых объектов переопределены стандартным образом (в виде операций <,> и т.д.). Для позиционирования в потоке можно использовать методы seekg(long,mode) и long tellg(), для хранимого объекта переопределен метод fstream << object который записывает объект с текущей позиции в открытый двоичный поток и метод fstream >> object, который читает содержимое объекта с текущей позиции открытого потока.

Программа должна реализовывать указанные выше действия. Протестировать структуру данных на простом типе данных (например, int, double и т.д.) и сложном. Программа тестирования должна содержать **меню**, обеспечивающее выбор операций.

1. Шаблон – двоичный файл, содержащий двусвязный циклический список объектов. Тип хранимого в файле объекта – параметр шаблона. В начале файла расположен файловый указатель на первый элемент списка. Элемент списка содержит 2 файловых указателя на следующий и предыдущий элементы, за которыми следует объект.
2. Шаблон – двоичный файл, содержащий односвязный список объектов. Тип хранимого в файле объекта – параметр шаблона. В начале файла расположен файловый указатель на первый элемент списка. Элемент списка содержит файловый указатель на следующий элемент, за которым следует объект.
3. Шаблон – двоичный файл, содержащий массив указателей на объекты. Тип хранимого в файле объекта – параметр шаблона. В начале файла расположены: размерность массива указателей (int), текущее количество указателей (int) и смещение (адрес) массива указателей. Если происходит заполнение массива указателей, то его размерность увеличивается в 2 раза и он переписывается в конец файла.
4. Шаблон – двоичный файл, содержащий двоичное дерево объектов. Тип хранимого в файле объекта – параметр шаблона. В начале файла расположен указатель на корневую вершину. Вершина содержит два файловых указателя на правое и левое поддерево и сам объект, который непосредственно следует за указателями.
5. Шаблон – двоичный файл, содержащий двоичное дерево объектов. Тип хранимого в файле объекта – параметр шаблона. В начале файла расположен указатель на корневую вершину. Вершина содержит два файловых указателя на правое и левое поддерево, счетчик количества объектов (int) и статический массив указателей на объекты. Структура данных упорядочена (дерево и массив объектов).

6. Коллекция объектов произвольных типов в памяти

Необходимо разработать интерфейс для объединения в структуру данных множества объектов различных классов - абстрактный базовый класс объектов object, для которого предусмотреть виртуальные методы: загрузки объекта из текстовой строки, выгрузки объекта в текстовую строку в динамической памяти, добавления объекта в последовательный двоичный файл, чтения объекта из последовательного двоичного файла, возврата уникального идентификатора класса, возврата указателя на строку с именем класса, сравнения двух объектов, “сложения “(объединения) двух объектов, создание динамической копии объекта. Сделать классы хранимых объектов производными от абстрактного базового класса object (например, класс Float, String и класс из лабораторной работы).

Коллекция представляет собой структуру данных, хранящую указатели на объекты произвольного типа - объекты базового класса. Программа должна обеспечивать просмотр коллекции, добавление объекта выбранного типа, редактирование и удаление объекта,

сложение и сравнение объектов при условии совпадения типов с помещением результата в ту же коллекцию, сохранение и загрузку объектов из текстового (или двоичного) файла.

Программа должна реализовывать указанные выше действия. Протестировать структуру данных на простом типе (например, Int или String). Программа тестирования должна содержать **меню**, обеспечивающее выбор операций.

Вид структуры данных

1. Двусвязный список.
2. Двусвязный циклический список.
3. Список с организацией стека.
4. Список с организацией очереди.
5. Дерево, вершина которого содержит два указателя на поддеревья, счетчик количества вершин и указатель на объект

7. Разное. (начальный балл от 4,5)

Полная функциональность класса – это все операции по просмотру, добавлению, вставке по номеру, удалению, обновлению и сортировке объектов заданной структуры.

1. Создать шаблон – двоичный файл, содержащий односвязный список объектов. Тип объекта – параметр шаблона. В начале файла расположен файловый указатель на первый элемент списка. Элемент списка содержит файловый указатель на следующий элемент списка. Разработать полную функциональность класса. Проверить его на простых типах и сложных типах. Протестировать шаблон. Программа тестирования должна содержать меню, обеспечивающее выбор операций.
2. Создать шаблон – двоичный файл, содержащий двусвязный список объектов. Тип объекта – параметр шаблона. В начале файла расположены файловые указатели на первый и последний элементы списка. Элемент списка содержит файловые указатели на следующий и предыдущий элементы списка. Разработать полную функциональность класса. Проверить его на простых типах и сложных типах. Протестировать шаблон. Программа тестирования должна содержать меню, обеспечивающее выбор операций.
3. Создать класс, производный от fstream. Файл содержит массив указателей на упорядоченные по алфавиту строки, представленные записями переменной длины: целый счетчик и последовательность символов. Формат файла: в начале его размер массива указателей, текущее количество указателей, адрес массива указателей в файле. Обеспечить полную функциональность класса. Протестировать его. Программа тестирования должна содержать меню, обеспечивающее выбор операций.
4. Создать шаблон – двоичный файл, содержащий бинарное дерево. Каждая вершина дерева содержит указатели на поддеревья и объект, тип которого задается параметром шаблона. Разработать полную функциональность класса. Проверить его на простых типах и сложных типах. Протестировать шаблон. Программа тестирования должна содержать меню, обеспечивающее выбор операций. (балл 5)
5. Создать класс, производный от fstream. Файл содержит бинарное дерево. Каждая вершина дерева содержит динамический массив целых переменных. Целые в дереве и массивах упорядочены. Формат файла: в начале его указатель на корневую вершину. Вершина содержит файловые указатели на левое и правое поддеревья, количество целых значений в вершине, размерность массива, сам

массив, Разработать полную функциональность класса. Протестировать его. Программа тестирования должна содержать меню, обеспечивающее выбор операций. (балл 5)

8. Реализовать абстрактный тип данных (АТД) «Файловая структура данных» в соответствии с вариантом.

Операции АТД:

Insert – вставка записи в файл,
Delete – удаление записи из файла,
Search - поиск записи в файле.

1. Элементом данных, хранящихся в файле, является запись, имеющая уникальное ключевое значение.
2. Запись в файле представлена индексом, т.е. парой (k,p), где k – ключевое значение, p – файловый указатель на начало записи в файле.
3. Для файла данных поддерживается файловая структура индексов, указанная в варианте задания.
4. Файл индексов имеет страничную структуру. Страницы содержат индексы записей и имеют фиксированный размер.
5. Чтение и запись в файл индексов ведется постранично.
6. Тестирование файловой структуры ведется для различных значений параметров:
   N – число записей в файле данных, $N = 10^3, 10^4, 10^5, 10^6$,
   M – число индексов на странице файла индексов, M = 10,100, 1000.
7. Число обращений к блокам файла индексов в процессе выполнения операций должно соответствовать:
   – для хешированного файла - $N/(M*K)$, где K – число сегментов хеш – таблицы,
   – для индексированного файла – $2 + \log(N/M)$,
   – для B- дерева файла – $2 + \log_t (N/M)$, где t – степень B – дерева.

Варианты заданий.

1. АТД «Плотный индекс файла». Записи не закрепленные. Ключ записи – строка символов переменной длины.
2. АТД «Плотный индекс файла». Записи закрепленные. Ключ записи – строка символов переменной длины.
3. АТД «Плотный индекс файла». Записи не закрепленные. Ключ записи – вещественное число.
4. АТД «Плотный индекс файла». Записи закрепленные. Ключ записи - вещественное число.
5. АТД «Разреженный индекс файла». Записи не закрепленные. Ключ записи – строка символов переменной длины.
6. АТД «Разреженный индекс файла». Записи не закрепленные. Ключ записи – вещественное число.
7. АТД «Разреженный индекс файла». Записи не закрепленные Ключ записи – вещественное число. Блок файла индексов имеет $K = M/10$ для возможности дальнейшей вставки записей без выделения новых страниц индексов.
8. АТД «B – дерево файла» с нисходящим разделением полных узлов при вставке записей. Записи не закрепленные. Ключ записи – целое число.
9. АТД «B – дерево файла» с нисходящим разделением полных узлов при вставке записей. Записи закрепленные. Ключ записи – целое число.
10. АТД «B – дерево файла» с восходящим разделением полных узлов при вставке записей. Записи не закрепленные. Ключ записи – целое число.

11. АТД «B – дерево файла» с восходящим разделением полных узлов при вставке записей. Записи закрепленные. Ключ записи – целое число.
12. АТД «Хешированный файл». Метод хеширования – метод цепочек. Записи не закрепленные. Ключ записи – строка символов переменной длины.
13. АТД «Хешированный файл». Метод хеширования – метод цепочек. Записи закрепленные. Ключ записи – строка символов переменной длины.

5. Примерный перечень вопросов к защите курсовой работы.

1. Оцените трудоемкость основных алгоритмов проекта
2. Опишите процессы преобразования структур данных при выполнении базовых операций
3. Расшифруйте дамп двоичного файла, формируемого программой
4. Прокомментируйте результаты оценки производительности выполнения прграммы