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.

Задание на курсовую работу выдается каждому студенту индивидуально. Курсовая работа должна быть выполнена с использованием технологии объектно-ориентированного программирования (классы, наследование, виртуальные функции, шаблоны) и реализована в виде набора классов. Программа демонстрирует выполнение всех своих функций в любой последовательности через систему меню.

Структура:
1. Проектирование и реализация системы классов.
2. Проверка на тестовых данных.
3. Пояснительная записка.

Этапы выполнения и защиты:
1. Проектирование структур данных.
2. Реализация функциональности классов.
3. Тестирование.
4. Написание пояснительной записки к курсовой работе.
5. Защита курсовой работы.

Оцениваемые позиции:
1. Реализация в программе заданной функциональности.
2. Ответы на поставленные вопросы по работе.
3. Качество оформления пояснительной записки.

Правила оформления пояснительной записки к курсовой работе:

Пояснительная записка должна иметь объем не менее 20 стр. текста (без приложения, шрифт 12), выполнена в соответствии с ГОСТом и содержать следующие разделы.

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.

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.

2. **Критерии оценки.**

- работа считается **не выполненной,** если реализован не весь функционал программы, указанный в задании, при защите студент не может объяснить алгоритмы работы программы, пояснительная записка выполнена небрежно и содержит ошибки, оценка составляет 0-49 баллов.
- работа считается выполненной **на пороговом** уровне, если реализован весь функционал программы, указанный в задании, при защите студент объясняет алгоритмы работы программы с неточностями, есть небольшие недочеты в программе, а также пояснительная записка содержит некоторое количество ошибок, оценка составляет 50 - 72 баллов.
- работа считается выполненной **на базовом** уровне, если реализован весь функционал программы, указанный в задании, при защите студент грамотно объясняет алгоритмы работы программы, отвечает на поставленные вопросы, пояснительная записка выполнена согласно требованиям по оформлению, оценка составляет 73-86 баллов.
- работа считается выполненной **на продвинутом** уровне, если реализован весь функционал программы, указанный в задании, студент в разработке применил оригинальные подходы и алгоритмы, при защите студент отвечает на все поставленные вопросы полно, пояснительная записка выполнена согласно требованиям по оформлению, оценка составляет 87-100 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 оценивается по 100 бальной системе в соответствии с правилами балльно-рейтинговой системы.

4. **Примерный перечень тем курсового проекта (работы).**

1. 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.

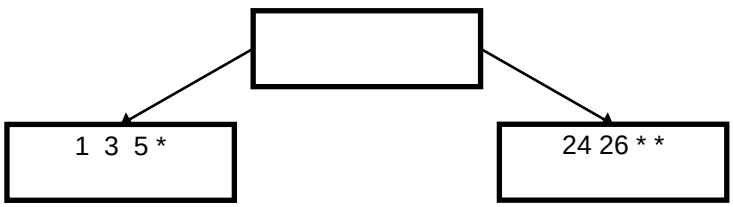
4. Шаблон структуры данных – односвязный список, каждый элемент которого содержит динамический массив указателей на объекты. Размерность массива указателей в каждом последующем элементе списка в 2 раза больше, чем в предыдущем. Последовательность указателей в каждом массиве ограничена NULL. При переполнении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



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.).

2. 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 и сохранения содержимого двоичного файла в текстовый.

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.

3. 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).

4. 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.

Дополнение: добавить в коллекцию вложенный класс, реализующий функции итератора **(дополнительный балл 0,5)**.

5. Реализовать 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. Перечень вопросов к защите курсового проекта (работы).

При защите курсовой работы вопросы студент отвечает на вопросы по реализуемым структурам данных и алгоритмам.