1.

Класс "Двунаправленный список отрезков прямой с возможностью сортировки".

Реализация класса должна поддерживать следующие возможности:

создать пустой список, создать список из одного элемента, создать список, элементами которого являются заданные концы отрезов (хранящиеся в массиве; хранящиеся в файле);

вставить элемент в конец (начало) списка;

вставить элемент справа (слева) от текущего;

добавить в конец (начало) списка все элементы другого списка;

получить индекс (порядковый номер) текущего элемента;

удалить текущий элемент; элемент, который находится справа (слева) от текущего, элемент, имеющий заданный индекс;

получить количество элементов в списке;

удалить все элементы из списка;

сортировать элементы списка методом слияния без рекурсии по указанной функции сравнения отрезков. В процессе сортировки нельзя копировать элементы списка (использовать вставку и удаление элементов).

Формальное определение интерфейса не задано и должно быть разработано студентом. Тесты должны включать проверку работы всех реализованных методов в различных стандартных и нестандартных ситуациях.

2.

Класс "Расширенное битовое множество". Требуется реализовать хранение и работу с подмножествами целых чисел, принадлежащих заданному интервалу. Идея реализации: заводим массив целых чисел (типа unsigned long int), в котором каждому возможному элементу множества сопоставляется один бит. Таким образом, работа сводится к установке или проверке значения отдельного бита в некотором элементе целочисленном массиве. При размещении элемента, выходящего из заданного диапазона, диапазон должен автоматически расширяться.

Реализация класса должна поддерживать следующие возможности:

создание пустого множества; создание множества, элементами которого являются заданные числа (хранящиеся в целочисленном массиве, в файле, записанные в виде последовательности 0 и 1);

очистка всего множества;

сравнение двух подмножеств на равенство;

добавление/ удаление заданного элемента;

проверка принадлежности заданного элемента подмножеству;

получение максимального и минимального элемента данного подмножества;

итератор по множеству (возможность последовательного перебора всех элементов множества);

пересечение, объединение, дополнение, исключающее объединение множеств;

сдвиг вправо (уменьшение всех чисел на 1) и влево (увеличение на 1);

получение границ текущего диапазона множества;

Формальное определение интерфейса не задано и должно быть разработано студентом. Тесты должны включать проверку работы всех реализованных методов в различных стандартных и нестандартных ситуациях.

3.

Класс "Двунаправленный список строк (указателей char *) неограниченной длины с возможностью сортировки". Для размещения строки в памяти использовать функцию new и полученные указатели сохраняются в списке.

Реализация класса должна поддерживать следующие возможности:

создать пустой список, список из одного элемента, список строк, хранящихся в файле;

вставить элемент в конец (начало) списка;

вставить элемент справа (слева) от текущего;

вставить элемент так, чтобы он имел указанный индекс;

добавить в конец (начало) списка все элементы другого списка;

получить индекс (порядковый номер) текущего элемента;

получить указатель на текущий (следующий/предыдущий) элемент списка (перемещение текущей позиции и итератор)

удалить текущий элемент; элемент, который находится справа (слева) от текущего, элемент, имеющий заданный индекс;

получить количество элементов в списке;

удалить все элементы из списка;

сортировать элементы списка методом быстрой[1] сортировки по заданной функции сравнения строк. В процессе сортировки нельзя копировать строки (использовать вставку и удаление элементов).

Формальное определение интерфейса не задано и должно быть разработано студентом. Тесты должны включать проверку работы всех реализованных методов в различных стандартных и нестандартных ситуациях.

4.

Класс "Динамический массив чисел с возможностью сортировки". Требуется реализовать динамический массив чисел (значений double) с возможностью сортировки и быстрого (бинарного) поиска. Идея реализации состоит в том, что сначала для хранения выделяется небольшой массив фиксированной длины, а по мере добавления элементов выделяются дополнительные такие же массивы, которые связываются в список.

Реализация класса должна поддерживать следующие возможности:

создать массив заданной начальной длины;

добавить элемент в конец массива (удлинить массив);

вставить элемент в указанное место (по известному индексу) с удлинением массива;

получить указатель (или ссылку) на элемент с заданным индексом для прямого доступа к его значению;

удалить элемент по указанному индексу (сократить массив);
получить количество элементов в массиве;
установить новую длину массива;
искать заданное значение в массиве;

сортировать массив методом HeapSort(пирамидальная);
удалить все элементы из массива.

Формальное определение интерфейса не задано и должно быть разработано студентом. Тест должен содержать заполнение массива элементы из файла и проверку работы всех реализованных методов в различных корректных и некорректных ситуациях.

5.

Класс "Токенайзер текстовой строки".

Требуется разбить данную строку на токены (слова) по заданному набору символов-разделителей и обеспечить работу с этими токенами. Наряду с исходной строкой в реализации создается список, хранящий позиции и длины, выделенных токенов.

Реализация класса должна поддерживать следующие возможности:

Инициализировать токенайзер данным набором символов-разделителей (взятых из файла);

Разбить данную строк на токены;

Получить общее количество найденных токенов;

Возможность последовательного просмотра токенов (итератор);

Получение токена с заданным порядковым номером;

Возможность изменить набор разделителей для данной строки;

Формальное определение интерфейса не задано и должно быть разработано студентом. Тест должен содержать разбиение различных строк по различным разделителям и проверку работы всех реализованных методов в различных корректных и некорректных ситуациях.

6.

Класс "Динамический массив чисел с возможностью сортировки". Требуется реализовать динамический массив чисел (значений double) с возможностью сортировки и быстрого (бинарного) поиска. Идея реализации состоит в том, что сначала для хранения выделяется небольшой массив фиксированной длины, а по мере добавления элементов выделяются дополнительные такие же массивы, которые связываются в список.

Реализация класса должна поддерживать следующие возможности:

создать массив заданной начальной длины;

добавить элемент в конец массива (удлинить массив);

вставить элемент в указанное место (по известному индексу) с удлинением массива;

получить указатель (или ссылку) на элемент с заданным индексом для прямого доступа к его значению;

удалить элемент по указанному индексу (сократить массив);
получить количество элементов в массиве;
установить новую длину массива;
искать заданное значение в массиве;

сортировать массив методом бинарных вставок;
удалить все элементы из массива.

Формальное определение интерфейса не задано и должно быть разработано студентом. Тест должен содержать заполнение массива элементы из файла и проверку работы всех реализованных методов в различных корректных и некорректных ситуациях.

7.

Класс "Двумерное битовое множество". Требуется реализовать хранение и работу с подмножествами пар (x,y) целых чисел, принадлежащих заданным диапазонам ($[x_{min}, x_{max}], \times [y_{min}, y_{max}]$. Идея реализации: сначала паре (x,y) сопоставляем линейный номер (как при размещении матрицы по строкам в одномерном массиве), затем заводим массив целых чисел (типа unsigned long int), в котором каждому возможному линейному номеру сопоставляется один бит. Таким образом, работа сводится к установке или проверке значения отдельного бита в некотором элементе целочисленного массива. Значения, выходящие из заданного диапазона, не могут храниться в множестве (сообщение об ошибке), но диапазон можно изменить отдельной функцией.

Реализация класса должна поддерживать следующие возможности:
создание пустого множества для заданного диапазона чисел;
очистка всего множества;
сравнение двух множеств на равенство;

добавление/ удаление заданного значения;
проверка принадлежности заданного значения множеству;

пересечение, объединение, дополнение, исключающее объединение множеств;

получение max и min значений, содержащихся в множестве (ограничивающий прямоугольник);

итератор по множеству (возможность последовательного перебора всех значений множества)

изменение диапазона допустимых значений;
получение границ текущего диапазона.

Формальное определение интерфейса не задано и должно быть разработано студентом. Тест должен содержать заполнение множества некоторым набором значений и проверку работы всех реализованных методов в различных корректных и некорректных ситуациях.

8.

Класс "Кольцевая очередь строк".

Требуется реализовать работу очереди строк на базе одного отрезка памяти, причем при полном заполнении этого отрезка новые добавленные строки затирают (целиком) строки из начала очереди. Все строки размещаются в кольцевом буфере памяти без промежутков (при этом одна из строк может оказаться физически разорванной). Указатели на строки очереди можно дополнительно хранить в отдельном списке.

Реализация класса должна поддерживать следующие возможности:

создать пустую очередь на заданное количество байт;

добавить строку (в конец);
получить длину строки в начале очереди;
копировать строку из начала очереди по заданному указателю;
удалить начало очереди;
получить количество строк в очереди;

удалить все элементы очереди (очистить);
получить количество потерянных строк в начале очереди;

Формальное определение интерфейса не задано и должно быть разработано студентом. Тест должен содержать заполнение очереди строками из файла и проверку работы всех реализованных методов в различных корректных и некорректных ситуациях.

9.

Класс "Контейнер строк".

Для хранения текстовых строк выделяется блок памяти, в котором последовательно размещаются поступающие строки. Если очередная строка не помещается в остатке блока, то вызывается следующий блок. Все блоки завязываются в список для осуществления операции очистки контейнера.

Реализация класса должна поддерживать следующие возможности:

создание пустого контейнера, контейнера, в котором хранится одна строка;

очистка всего контейнера;

добавление строки в контейнер (возвращает указатель на место размещения строки);

получение количества строк, хранящихся в контейнере.

Формальное определение интерфейса не задано и должно быть разработано студентом. Тест должен содержать заполнение контейнера некоторым набором строк из файла и проверку работы всех реализованных методов в различных корректных и некорректных ситуациях.

10.
Класс "Строковый буфер".

Требуется реализовать возможность работы со строками произвольной длины. Строка хранится отдельными кусками фиксированной длины, которые связываются в список и добавляются и удаляются по мере необходимости.

Реализация класса должна поддерживать следующие возможности:

создание пустого буфера, буфера, инициализированного данной строкой;

очистка строкового буфера;

добавление строки в конец буфера;

вставка строки в заданную позицию буфера;

получение длины строк, накопленной в буфере;

копирование строки, хранящейся в буфере, в заданный символьный массив;

копирование подстроки из буфера, в заданный символьный массив;

получить/изменить символ в заданной позиции;

искать заданный символ/подстроку в буфере;

заменить одну подстроку буфера на другую;

обрезать строку буфера до указанной длины;

Формальное определение интерфейса не задано и должно быть разработано студентом. Тест должен содержать заполнение буфера некоторыми строками и проверку работы всех реализованных методов в различных корректных и некорректных ситуациях.

11.
Класс "Буфер чтения строк".

Требуется реализовать возможность чтения строк произвольной длины из текстового файла. Строка считывается отдельными блоками ограниченной длины, которые связываются в список.

Реализация класса должна поддерживать следующие возможности:

начать работу с заданным файлом (по имени или указателю FILE *);

прочитать очередную строку;

получить длину прочитанной строки;

скопировать прочитанную строку, всю или частично, в заданный символьный массив;

копирование подстроки из буфера, в заданный символьный массив;

искать заданный символ/подстроку в прочитанной строке;

Формальное определение интерфейса не задано и должно быть разработано студентом. Тест должен содержать чтение текста из файла и проверку работы всех реализованных методов в различных корректных и некорректных ситуациях.

12.
Класс "Разреженная матрица переменного размера".

Требуется реализовать числовую матрицу, в которой количество ненулевых элементов значительно меньше общего количества элементов. Для этого каждая строка матрицы с номером i представляется списком, хранящим ненулевые значения элементов вместе с их j-м индексом. Такие списки сами собраны в список, упорядоченный по номеру строки, т.е. i-й элемент этого списка хранит указатель на список-строку и соответствующий ей список индексов.

Реализация должна поддерживать следующие возможности:

создание пустой матрицы,

получить текущие размеры матрицы (размер строки определяется по максимальному индексу j);

получить или изменить значение элемента матрицы с заданными индексами (размеры матрицы при этом могут измениться);

обменять местами указанные строки матрицы;

выполнить линейную комбинацию одной строки матрицы с другой;

вычислить сумму элементов i-строки для $j_1 \le j \le j_2$.

получить подматрицу по заданному диапазону или множеству индексов строк и столбцов.

Формальное определение интерфейса не задано и должно быть разработано студентом. Тест должен включать заполнение матрицы некоторыми значениями и проверку работы всех реализованных методов в различных корректных и некорректных ситуациях. Одним из тестов должно быть решение системы линейных однородных уравнений методом Гаусса..

13.
Класс "Разреженная матрица фиксированного размера".

Требуется реализовать числовую $m \times n$ матрицу, в которой количество ненулевых элементов значительно меньше mn. Каждая строка матрицы с номером i представляется списком, хранящим ненулевые значения элементов вместе с их j-м индексом. Такие списки собраны в массив, т.е. i-й элемент этого массива представляет собой список i-й строки матрицы.

Реализация должна поддерживать следующие возможности:

создание нулевой матрицы заданного размера, матрицы, у которой ненулевой является строка с заданным номером;
получить или изменить значение элемента матрицы с заданными индексами;
обменять местами указанные строки матрицы;
умножить строку матрицы на заданное число;
получить вектор столбец (строку) матрицы по указанному индексу.

Формальное определение интерфейса не задано и должно быть разработано студентом. Тест должен включать заполнение матрицы некоторыми значениями и проверку работы всех реализованных методов в различных корректных и некорректных ситуациях. Одним из тестов должно быть моделирование электронной таблицы, содержащей столбец с суммой столбцов и строку с суммой строк.

14.
Класс "Стек строк с файловым буфером".

Требуется реализовать работу со строками произвольной длины по стековому принципу с ограниченным использованием оперативной памяти. Идея реализации: выделяется блок памяти и строки укладываются в него одна за другой. Если для добавления новой строки не хватает места, то заполненный блок записывается в файл, а освободившееся место используется для новых добавлений. Если при извлечении строки буфер окажется пустым, то он заполняется чтением ранее сохраненной в файле информацией. Таким образом, в реализации есть два стека: Стек строк в памяти и стек блоков записей в файле.

Реализация класса должна поддерживать следующие возможности:

создать пустой стек;
добавить строку;
получить длину строки на вершине стека;
копировать строку на вершине стока по заданному указателю;
удалить вершину стека;
получить количество строк в стеке;
удалить все элементы стека (очистить);
сохранить состояние стека в файле /загрузить из файла;

Формальное определение интерфейса не задано и должно быть разработано студентом. Тест должен содержать заполнение стека строками и проверку работы всех реализованных методов в различных корректных и некорректных ситуациях.

15.
Класс "Очередь строк с отметкой времени".

Требуется реализовать работу очереди строк, причем каждый элемент очереди дополнительно хранит отметку времени добавления этого элемента в очередь. Очередь реализуется на базе списка и тем самым не имеет ограничений на количество элементов. Строки размещаются в памяти независимо, а указатели на них сохраняются в очереди.

Реализация класса должна поддерживать следующие возможности:
создать пустую очередь;
добавить строку в конец очереди;
получить длину строки в начале очереди;
копировать строку в начале очереди по заданному указателю;
удалить начало очереди;
получить количество строк в очереди;
удалить все элементы из очереди (очистить);
получить метку времени в начале очереди;

Формальное определение интерфейса не задано и должно быть разработано студентом. Тест должен содержать заполнение очереди строками из файла и проверку работы всех реализованных методов в различных корректных и некорректных ситуациях.