Разработать указанные классы.

- **Запрещается использовать контейнеры и алгоритмы STL.**
- **Реализовать необходимые конструкторы.**
- **Реализовать деструктор, конструкторы копирования и перемещения, операторы копирующего и перемещающего присваивания.**
- **Память должна корректно выделяться и освобождаться.**
- **Реализовать методы, позволяющие осуществлять работу с объектами. (Необходимо продумать, как объекты будут использоваться)**
- **Некоторые методы можно реализовать в виде перегруженных операторов (если подходят по смыслу).**
- **Реализовать перегруженный оператор << для вывода информации в поток.**
- **Объявление класса должно быть в заголовочном файле (.h), а определения методов в файле реализации (.cpp). Шаблон класса полностью описывается в заголовочном файле.**
- **В main продемонстрировать работу всех(!) методов, операторов и конструкторов.**

Варианты (брать по кругу 1, 2, 3, 4, 5, 1, 2 и т.д.):

1. Класс для работы с матрицами. Методы для сложения, умножения, транспонирования матриц. Методы для получения и задания произвольных элементов матрицы (с произвольными индексами). Подсказка: внутри объекта массив не обязательно должен быть двумерным.
2. Класс массива динамической дины (наподобие класса vector из STL). Массив должен автоматически увеличивать свой размер при добавлении элементов, увеличение должно производиться с некоторым запасом. Методы для добавления и удаления элементов массива, для обращения к элементам, а также для сортировки массива. (В идеале класс должен быть шаблонным, но можно сделать только для типа int)
3. Класс для работы с односвязным списком. Методы для добавления, извлечения и просмотра элементов с обоих концов списка (кроме удаления из конца списка), метод для удаления элемента с указанным значением, метод для переворачивания списка в обратном порядке. (В идеале класс должен быть шаблонным, но можно сделать только для типа int)
4. Класс для работы с двусвязным списком. Методы для добавления, извлечения и просмотра элементов с обоих концов списка. Методы для перемещения по списку в обе стороны: один метод должен возвращать значение элемента, который в данный момент считается текущим (изначально первый), другие перемещать текущий элемент вправо, влево, в начало и в конец списка (имеется ввиду не перемещение самого элемента, а изменение того, какой элемент считается текущим). (В идеале класс должен быть шаблонным, но можно сделать только для типа int)
5. Класс для работы с односвязным отсортированным списком уникальных элементов. При добавлении элемента он должен вставляться в нужное место списка и только в том случае, если элемента с таким значением ещё нет. Написать методы для добавления элемента, проверки наличия элемента в списке, метод для получения нового списка путём объединения двух существующих (оба списка должны просматриваться только по одному разу). (В идеале класс должен быть шаблонным, но можно сделать только для типа int)