Задания к работе №2 по алгоритмам и структурам данных.

Все задания реализуются на языке программирования C++ (стандарт C++14 и выше). Реализованные в заданиях приложения не должны завершаться аварийно; все возникающие исключительные ситуации должны быть перехвачены и обработаны.

Во всех заданиях запрещено пользоваться функциями, позволяющими завершить выполнение приложения из произвольной точки выполнения.

Во всех заданиях при реализации необходимо разделять контексты работы с данными (поиск, сортировка, добавление/удаление, модификация и т. п.) и отправка данных в поток вывода / выгрузка данных из потока ввода.

Во всех заданиях все вводимые (с консоли, файла, командной строки) пользователем данные должны (если не сказано обратное) быть подвергнуты валидации в соответствии с типом валидируемых данных.

Во всех заданиях необходимо контролировать ситуации с невозможностью [пере]выделения памяти; во всех заданиях необходимо корректно освобождать всю выделенную динамическую память.

Все ошибки, связанные с операциями открытия системных ресурсов уровня ОС (файлы, средства синхронизации, etc.), должны быть обработаны; все открытые системные ресурсы должны быть возвращены ОС.

Реализованные компоненты должны зависеть от абстракций, а не от конкретных реализаций абстракций. Для реализованных компонентов должны быть переопределены (либо перекрыты - при обосновании) следующие механизмы классов C++: конструктор копирования, деструктор, оператор присваивания, конструктор перемещения, присваивание перемещением.

1. Опишите родовой контракт (интерфейс, абстрактный класс), предназначенный для выполнения алгоритма сортировки значений в коллекции типа *std::vector<T>*. Реализуйте контракт в виде классов, реализующих следующие алгоритмы сортировки в виде объектов-стратегий:
   - подсчётом;
   - вставками;
   - выбором;
   - методом Шелла;
   - быстрая;
   - пирамидальная;
   - поразрядная (при ограничении типа родового параметра как целочисленного).

   Реализуйте контекст сортировки, в методы которого можно подавать коллекции данных с возможностью конфигурации компаратора отношения порядка (используйте контракт *std::function*) на пространстве значений родового типа элементов, а также дополнительные параметры сортировок (при необходимости).