3.1 Задание

Необходимо разработать программу для сортировки составных данных, удовлетворяющую следующим минимальным требованиям.

1. Программа должна работать в трех режимах: генерация тестовых данных, сортировка данных, печать данных на экран.
2. Режим работы программы выбирается пользователем с помощью аргументов командной строки.
3. Программа, запущенная с флагом `--generate N` или `-g N`, генерирует случайные данные в соответствии со схемой данных (см. вариант) в количестве N строк.

   Если указан флаг `--out="output.csv"` или `-o output.csv`, вывод осуществляется в соответствующий файл а формате значений, разделенных запятой (CSV). Иначе вывод осуществляется на стандартный поток вывода в соответствующем формате.
4. Программа, запущенная со флагом `--sort` или `-s`, считывает данные в соответствии со схемой данных в структуру-контейнер (см. вариант), сортирует их и выводит отсортированные.

   Если указан флаг `--out="output.csv"` или `-o output.csv`, вывод осуществляется в соответствующий файл а формате значений, разделенных запятой (CSV). Иначе вывод осуществляется на стандартный поток вывода в соответствующем формате.

   Если указан флаг `--in="data.csv"`или `-i data.csv`, ввод осуществляется из соответствующего файла. Иначе ввод осуществляется со стандартного потока ввода в соответствующем формате.

   По-умолчанию сортировка осуществляется по возрастанию. Также по возрастанию следует сортировать, если указан флаг `--type=asc` или `-t A`. Если указан флаг `--type=desc` или `-t D`, сортировка выполняется в порядке убывания.
5. Программа, запущенная с флагом `-print` или `-P`, считывает данные и выводит их в формате таблицы с фиксированной шириной.

   Если указан флаг `--in="data.csv"`или `-i data.csv`, ввод осуществляется из соответствующего файла. Иначе со стандартного потока ввода

считывается имя файла, содержащее данные для считывания.

Если указан флаг `--out="output.txt"` или `-o output.txt`, вывод осуществляется в соответствующий файл. Иначе вывод осуществляется на стандартный поток вывода.

Логические значения при выводе заменить на `YES/NO`. Вещественные значения выводить с точностью до двух знаков после десятичного разделителя.

Минимальные требования, предъявляемые к исходному коду:

1. Проект должен состоять из нескольких логически разделенных файлов:
   - структура-запись, созданная по схеме данных, соответствующая одной строке;
   - структура-контейнер и реализация методов для работы с ней;
   - обработка аргументов командной строки;
   - алгоритм сортировки;
   - точки входа (`main`) и функций для работы с вводом-выводом.
2. Структура-контейнер должна поддерживать следующие методы:
   - инициализация;
   - получение текущего размера (количества хранимых элементов);
   - получение произвольного элемента по индексу;
   - получение указателей на начало и конец;
   - получение следующего и предыдущего элемента от указателя;
   - добавление элемента в начало, конец и произвольное место;
   - удаление элемента из начала, конца и произвольного места;
   - замена пары элементов друг на друга;
   - преобразование массива в структуру-контейнер и обратно;
   - очистка.
3. Алгоритм сортировки «не должен ничего знать» о деталях реализации структуры-контейнера: все взаимодействие с ней должно осуществляться через описанные выше функции.
4. Алгоритм сортировки также не должен явно использовать поля структуры-

записи: сравнение записей должно осуществляться через компаратор – функцию, передаваемую в подпрограмму сортировки по указателю.

5. Для сборки проекта необходимо использовать любой из рекомендованных подходов к автоматизации: `Make`, `CMake`, `bash/batch`-скрипт, `setup.py`-скрипт.
6. Разработка проекта должна осуществляться с применением системы контроля версий `git`:
   - работу с git следует осуществлять через интерфейс командной строки, а не графические интерфейсы интегрированных сред разработки;
   - последовательность фиксаций изменений в системе контроля версий (commit) должна отражать хронологию разработки задачи. Единовременная фиксация всего проекта в репозитории недопустима;
   - локальный репозиторий должен быть синхронизирован с удаленным (на github.com, gitlab.com, bitbucket.org) через интерфейс командной строки, а не графические интерфейсы интегрированных сред разработки или веб-интерфейс сервиса;

Для получения более высокого балла следует реализовать один из методов продвинутой сортировки, получить данные о зависимости времени выполнения сортировки от количества записей для него и метода из своего варианта, построить графики этих зависимостей.

3.1.1 Варианты сортировок

№ варианта	Алгоритм сортировки
1-2	Пузырьком
3-4	Перемешиванием
5-6	Расческой
7-8	Вставками
9-10	Выбором
11-12	Гномья
*	Быстрая (Хоара)
*	Слиянием
*	Пирамидальная (Кучей)

3.1.2 Варианты контейнеров

№ варианта	Структура-контейнер
1, 5, 9	Вектор
2, 6, 10	Двусвязный список
3, 7, 11	Двухсторонняя очередь (дек)
4, 8, 12	Стек

3.1.3 Варианты схем данных

№ варианта	**Схема данных (структура-запись)**
Четный	***Научная публикация*** **Название публикации**: *строка* **Фамилия первого автора**: *строка* **Инициалы первого автора**: *строка* **Название журнала**: *строка* **Год публикации**: четырехзначное *число* **Том журнала**: *целое число* **Входит в РИНЦ**: *логическое* значение **Количество страниц**: *целое число* **Цитирований**: *целое число*
Нечетный	***Многоквартирный дом*** **Название застройщика**: *строка* **Название микрорайона**: *строка* **Тип**: *перечисление* (панельный, кирпичный, монолитный) **Год постройки**: четырехзначное *число* **Наличие лифта**: *логическое* значение **Наличие мусоропровода**: *логическое* значение **Количество квартир**: *целое число* **Количество этажей**: *целое число* **Средняя площадь квартиры**: *вещественное число*