Общие требования

- Реализовать базу данных для хранения данных в файле. Реализовать операции «создать» - создать новую базу данных, «открыть...» - загрузить данные из файла, «сохранить» - сохранить данные в текущий файл, «сохранить как...» - сохранить данные в новый файл. При выходе из программы или загрузке другого файла предлагать сохранять текущие данные.
- Заменить массив данных на контейнер. Вид контейнера определяется индивидуальным заданием.
- Каждой записи присвоить уникальный идентификатор, который не изменяется в течение всей жизни программы (идентификатор задается как дополнительное поле класса записи - Id). Поместить идентификатор в браузер и использовать его для поиска записи в контейнере.
- Создать собственный класс модели данных (например, BookDatabase), содержащий в себе контейнер записей и поддерживающий следующие операции над данными:
  - получить кол-во записей:

    ```
    int count() const
    ```

  - добавить запись в базу данных; возвращается позиция записи в соответствии с порядком сортировки:

    ```
    int append(T &record)
    ```

    где T — класс, соответствующий вашей записи;

    a) для тех, у кого создается запись «по умолчанию», метод дополнительно возвращает копию созданной записи «по умолчанию» через параметр record (для созданной записи функция генерирует уникальный идентификатор и он записывается в поле Id параметра record).

    b) для тех, у кого не создается запись «по умолчанию», метод генерирует уникальный идентификатор для указанной записи и он записывается в поле Id параметра record.

  - `void remove(unsigned int id)` — удалить из базы данных запись с заданным идентификатором;

- изменить запись в базе данных; возвращается новая позиция записи в соответствии с порядком сортировки:

  a) для тех, у кого содержимое поля записи изменяется по окончанию его редактирования

  ```
  int update(unsigned int id, const QString & name, const QString & value)
  ```

  где id — идентификатор редактируемой записи, name — название редактируемого поля, value — новое значение редактируемого поля (в текстовом формате);

  b) для тех, у кого содержимое записи изменяется по кнопке «Сохранить»

  ```
  int update(const T &record)
  ```

  где T — класс, соответствующий вашей записи. В поле Id параметра record содержится идентификатор редактируемой записи;

- `void record(unsigned int id, T &record) const` — возвращает запись (только для чтения) по заданному идентификатору;
- `const QVector<S> records() const` - возвратить вектор записей, которые должны отображается в браузере с учетом сортировки записей, где S — структура, соответствующая строке браузера (поля структуры совпадают с колонками барузера, также структура содержат идентификатор записи);
- `bool save(QString filename) const` — сохранить данные в заданный файл, возвращает `false`, если сохранить данные не удалось;
- `bool load(QString filename)` — загрузить данные из заданного файла; при этом предыдущие данные уничтожаются, возвращает `false`, если сохранить данные не удалось;
- `void clear()` - уничтожает все данные.
- `bool isModified() const` - показывает, имеются ли изменения БД после ее загрузки/сохранения - ДОБАВИТЬ

- ЗАПРЕЩАЕТСЯ ИЗМЕНЯТЬ ИНТЕРФЕЙС КЛАССА МОДЕЛИ: ДОБАВЛЯТЬ[1] И УДАЛЯТЬ МЕТОДЫ ОТКРЫТЫЕ (PUBLIC), ИЗМЕНЯТЬ СОСТАВ ПАРАМЕТРОВ, ОТКРЫВАТЬ СВОЙСТВА (ПЕРЕМЕННЫЕ) КЛАССА. РАЗРЕШАЕТСЯ ДОБАВЛЯТЬ МЕТОДЫ В ЗАКРЫТУЮ (PRIVATE) ЧАСТЬ КЛАССА, ЕСЛИ ИХ ИСПОЛЬЗОВАНИЕ УЛУЧШАЕТ СТРУКТУРУ ПРОГРАММЫ.
- Любой перебор элементов в контейнере выполнять только через итераторы. Использовать константные итераторы везде, где это возможно.
- Запрещается каждый раз переупорядочивать весь контейнер при добавлении и изменении записей в базе данных.
- Запрещается вызывать метод records() при каждом действии с записями базы данных (используйте вместо этого позиции записей, возвращаемые методами append() и update()), этот метод должен вызываться только при открытии нового файла.
- Для задания имени файла использовать стандартный диалог для работы с файлами (класс QFileDialog); диалог должен работать только с файлами вашего расширения.
- Для сохранения и считывания данных из файла использовать класс QDataStream.

Требования к протоколу

- словесные алгоритмы методов класса модели данных.

[1] При решении задач повышенной сложности (варианты 29,30) разрешается добавлять открытые методы в класс модели, для проверки ограничений на связи между записями и их поддержки по согласованию с преподавателем.