Отчёт

А.А.Федяшов, Д.Е.Теплюков

13.07.2017

1 Постановка задачи

Реализовать оконное приложение - редактор таблиц, поддерживающий следующие функции:
1)открытие таблицы из бинарного файла
2)сохранение таблицы в бинарный файл
3)создание пустой таблицы для последующего заполнения
4)редактирование данных в ячейках
5)вставка пустой строки в произвольную позицию
6)удаление произвольной строки
Требования к реализации:
1)Реализовать свой наследник класса QAbstractTableModel для отображения данных из файла
2)Переопределить метод setData для редактирования данных в файле
3)Добавить методы removeRow, insertRow для добавления и удаления записей из файла
4)Использовать поток QDataStream для чтения/записи данных
5)Использовать таблицу для отображения структур типа:

(QString name, uint course, uint group)

2 Содержание проекта

1)Заголовочные файлы(.h)

redactor.h-заголовочный файл диалогового окна. Содержит конструктор окна, заголовки функций - слотов, привязанных к определенным кнопкам в окне

studenttable.h-заголовочный файл таблицы - наследника QAbstractTableModel. Содержит заголовки переопределенных функций для модели таблицы. Имеет определение вложенной структуры Person

2)Файлы реализации(.cpp)

redactor.cpp - содержит реализацию методов основного диалогового окна

studenttable.cpp - содержит реализацию методов модели таблицы, а также перегрузки операторов « и » для ввода/вывода структур Person через поток QDataStream

main.cpp - точка входа в программу, вызывает конструктор окна Redactor и осуществляет его показ мотодом show

3)Файлы интерфейса(.ui)

redactor.ui - интерфейс главного диалогового окна

3 Описание классов Redactor и StudentTable

StudentTable

Класс StudentTable унаследован от класса QAbstractTableModel и содержит все его невиртуальные методы. В StudentTable находится вложенная структура Person(QString,uint,uint)

1)**Конструктор, возвращает модель данных(таблицу):**

*StudentTable(QObject*parent=0)*

2)**Вставка строки в таблицу:**

bool insertRows(int position, int rows, const QModelIndex &index)

Возвращает true, если вставка прошла успешно.

3)**Удаление строки в таблице:**

bool removeRows(int position, int rows, const QModelIndex &index)

Возвращает true, если вставка прошла успешно.

4) **Количество строк в таблице:**

int rowCount(const QModelIndex &parent = QModelIndex()) const

Возвращает число строк – размер контейнера для хранения структур.

5) **Количество столбцов в таблице:**

int columnCount(const QModelIndex &parent = QModelIndex()) const

Возвращает число столбцов – фиксировано 3.

6)**Отрисовка данных в таблице. Вызывается как автоматически моделью, так и явно:**

QVariant data(const QModelIndex &index, int role = Qt::DisplayRole) const

Возвращает шаблонный тип данных QVariant, содержащий данные из одной ячейки.

7)**Запись данных в таблицу. Позволяет вручную внести данные в ячейку таблицу:**

bool setData(const QModelIndex &index, const QVariant &value, int role = Qt::EditRole)

Возвращает true, если запись прошла успешно.

8)**Отрисовка данных в заголовках столбцов. Автоматически вызывается моделью:**

QVariant headerData(int section, Qt::Orientation orientation, int role) const

В текущей реализации возвращает пустое значение.

9)**Получение состояния текущей ячейки в таблице. Автоматически вызывается моделью:**

Qt::ItemFlags flags (const QModelIndex &index) const

Возвращает тип данных флаг: ItemFlags.

Данные класса StudentTable:

QVector <Person> students - контейнер для хранения структур из файла.

StudentTable содержит перегрузки вывода структур Person в поток QDataStream. Принцип сериализации структуры состоит в ее разбиении на составные типы данных.

В файле studenttable.h содержится структура Person, которая хранит в себе основные строки таблицы

QString name - имя студента

uint course - номер курса
uint group - номер группы

Redactor

Класс Redactor унаследован от класса QMainWindow.

1)**Конструктор, создает экземпляр диалогового окна:**

*Redactor(QWidget *parent = 0)*

Создает модель типа StudentTable, затем присваивает ее виджету для таблиц QTableWidget.

2)**Слот открытия таблицы:**

void on_actionOpen_triggered()

Создает диалоговое окна выбора файла, затем открывает поток по выбранному имени. Данные из файла переносятся в новую модель таблицы, которая затем присваивается виджету.

3)**Слот сохранения таблицы:**
void on_actionSave_triggered()

Создает диалоговое окно создания файла, затем открывает поток по введенному имени файла. Данные из таблицы построчно записываются в файл.

4)**Слот создания новой таблицы:**

void on_actionCreate_new_triggered()

Создает новую таблицу из трех пустых строк.

5)**Слот открытия диалогового окна About:**

void on_actionAbout_triggered()

Создает новое окно с помощью QMessageBox::about и заполняет его данными о разработчиках.

6)**Слот закрытия программы:**

void on_actionQuit_triggered()

Вызывает метод QApplication::quit().

7)**Слот кнопки для добавления новой строки в таблицу:**

void on_addButton_clicked()

Вызывает метод StudentTable::InsertRows и получает индекс выбранной ячейки в таблице.

8)**Слот кнопки для удаления выбранной строки в таблице:**

void on_deleteButton_clicked()

Вызывает метод StudentTable::RemoveRows и получает индекс выбранной ячейки в таблице.

4 Особенности реализации программы

Программа является полноценным редактором таблиц, реализующим связку модель данных <-> виджет.

Для улучшения внешнего вида приложения была добавлена иконка, которая отображается в заголовке окна и в диалоговом окне About.

Для невозможности растяжения окна после запуска приложения были установлены максимальные и минимальные размеры окна, с соответствующей блокировкой кнопки “На весь экран”.

Кнопки Save, Create New, Open, About и Quit были внесены в закладку File, расположенную в верхней части окна. Для наглядности, некоторые действия

были разграничены линиями.

В программе производится обработка ошибочных ситуаций, таких как попытка удаления последней строки в файле. При ошибке открытия/сохранения файла выводится диалоговое окно QMessageBox::critical.