

# Пояснительная записка к программе

## «Simple text processor»

### Оглавление

Формулировка задания .....	1
Внешний вид приложения .....	3
Выбор необходимых классов .....	5
Архитектура приложения .....	5
Детали реализации .....	6
Заключение .....	7

### Формулировка задания

07.03.2019 15:01 от Андрея Соколова (sokol19945@gmail.com) по электронной почте поступило задание следующего содержания:

Тестовое задание

1. Вывести произвольное число в двоичной системе исчисления.

Описание.

Написать на языке C++ функцию, принимающую на вход произвольное целочисленное значение и возвращающую строку, содержащую его представление в двоичной системе исчисления. Запрещается использовать встроенные функции для перевода из одной системы счисления в другую.

Декларация.

```
void NumberAsBinary(  
char* _result, // строка с результатом  
unsigned int _number // произвольное число  
)
```

Ожидаемый результат.

Например, для числа 13 данная функция должна вернуть строку "1101";.

Проверка работы.

Решение должно демонстрировать работу функции на примере 10 случайных чисел в диапазоне 0 до 512.

2. На вход подаются сведения о ценах на бензин на автозаправочных станциях (АЗС). В первой строке содержится число N, каждая из последующих строк имеет формат:

<Цена 1 литра (в копейках)> <Улица> <Марка бензина> <Компания>

Имеется не более 20 различных компаний и не более 30 различных улиц; названия компаний и улиц не содержат пробелов. В качестве марки бензина указываются числа 92, 95 или 98. Цена задается целым числом в диапазоне от 2000 до 3000. Каждая компания имеет не более одной АЗС на каждой улице; цены на разных АЗС одной и той же компании могут различаться. Вывести данные обо всех АЗС, предлагавшим все три марки бензина (вначале выводится название улицы, затем выводится название компании). Сведения о каждой АЗС выводить на новой строке и упорядочивать по названиям улиц в

алфавитном порядке, а для одинаковых названий улиц – по названиям компаний (также в алфавитном порядке). Если ни одной требуемой АЗС не найдено, то вывести текст «Нет». Программу нужно написать на языке C++ с использованием контейнеров Map().

Исходные данные

1: «19»

«2570 ул. Айвазовского 95 Ойл-вест»

«2860 ул. Айвазовского 98 Ойл-вест»

«2590 ул. Петровская 95 Оптима»

«2240 ул. Березовская 95 Премьер-нефть»

...

3. На вход подаются сведения о задолженности по оплате коммунальных услуг жильцов 144-квартирного 9-этажного дома. В первой строке указывается общее число задолжников N. Каждая из последующих N строк содержит информацию об одном из задолжников в формате:

<Фамилия> <Номер квартиры> <Задолженность>

Задолженность указывается в виде дробного числа (целая часть – рубли, дробная часть – копейки). В каждом подъезде на каждом этаже располагаются по 4 квартиры. Для каждого из 9 этажей дома найти жильца с наименьшей задолженностью и вывести сведения о нем: задолженность (выводится с двумя дробными знаками), номер этажа, номер квартиры, фамилия жильца. Считать, что в наборе исходных данных все задолженности имеют различные значения. Сведения о каждом задолжнике выводить на отдельной строке и упорядочивать по убыванию размера задолженности. Если на каком-либо этаже задолжники отсутствуют, то данные об этом этаже не выводить.

Программу нужно написать на языке C++.

Исходные данные

1: «12»

«Сорокина 13 838.53»

«Марченко 104 1090.06»

«Леонидов 52 548.48»

«Юрьев 94 179.56»

...

Пример верного решения

1: «758.32 8 101 Иванова»

Срок выполнения задания – два дня. Если возникнут любые вопросы – обращайтесь.

На очной встрече было обговорено, что разработка ведется с применением фреймворка Qt, поэтому задание также было выполнено на этом фреймворке (версия 5.12).

## Внешний вид приложения

Внешний вид разработанного приложения (рис. 1):

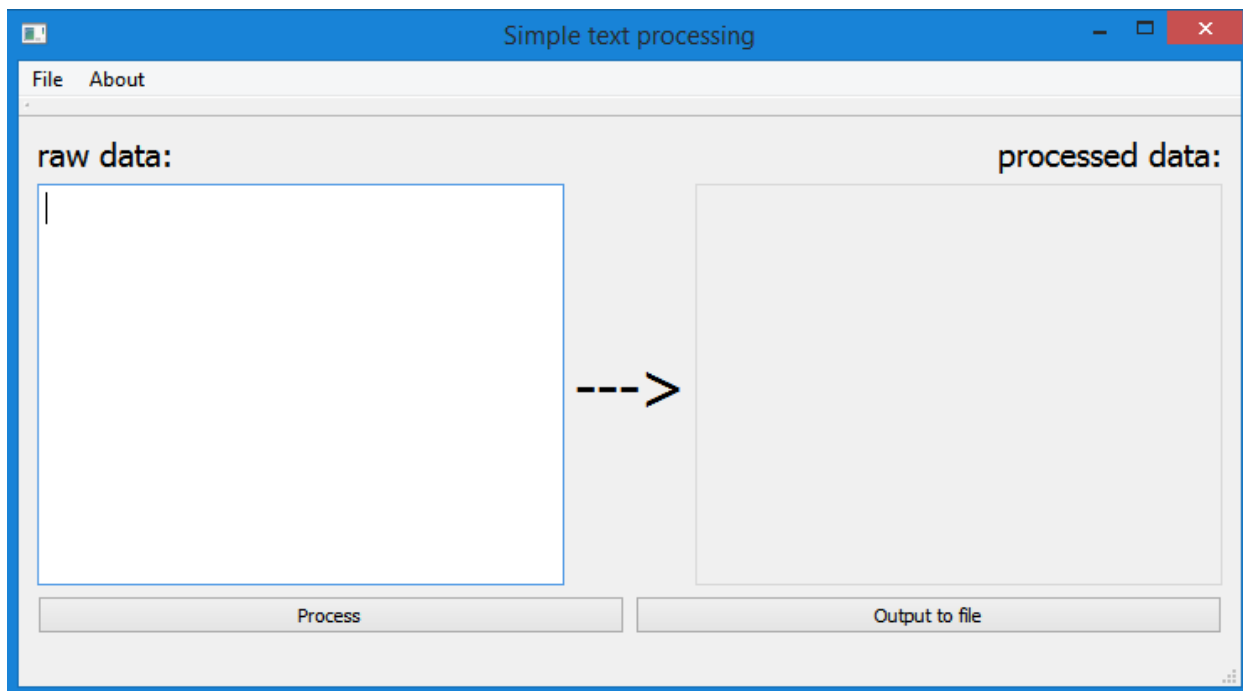


Рис. 1. Внешний вид приложения

В левом окне (raw data) предполагается введение данных, которое затем после нажатия кнопки «Process» будут обработаны и выведены на правое окно (processed data). Под правым окном, как видно, есть кнопка «Output to file», после нажатия на которую, данные правого окна через Проводник будут сохранены в файл с введенным пользователем именем.

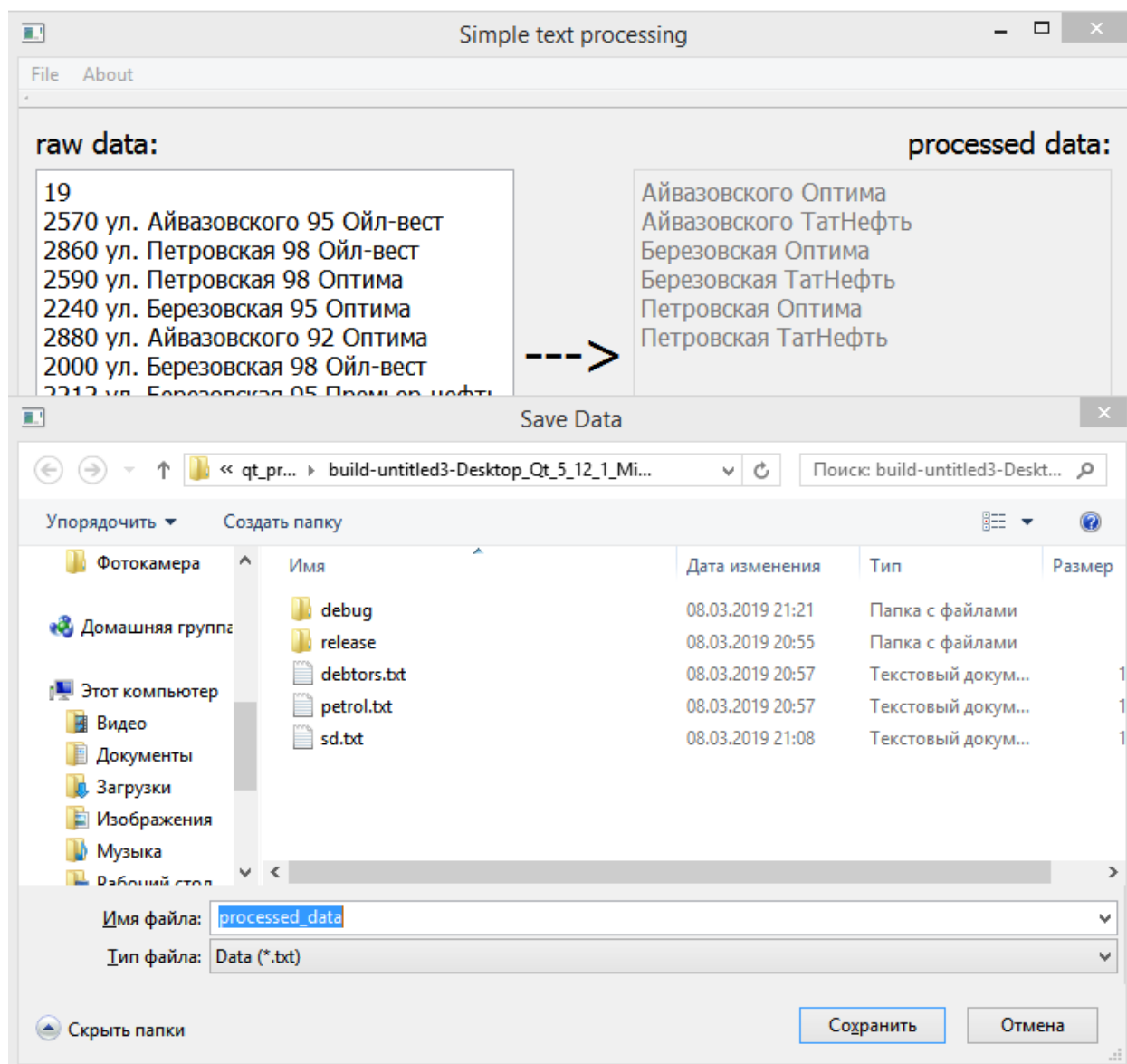


Рис. 2. Демонстрация функционала сохранения в файл

Также дополнительно реализовано считывание данных из файла. Считать данные можно кликнув на Меню: File→Read from file, сохранить данные в файл можно также через Меню: File→Save to file или воспользоваться горячими клавишами ALT+V или ALT+C, соответственно.

## Выбор необходимых классов

Для достижения поставленной цели, нам понадобятся следующие классы библиотеки Qt:

- графический интерфейс: QApplication, QMainWindow, QWidget, QDialog, QMessageBox, QLineEdit, QPushButton, QLabel, QAction;
- файловая система: QFileDialog,
- парсинг данных: QStringList, QRegExp;
- передача данных: QTextStream, QFile, QString;
- контейнеры: QMap, QVector,

А также библиотека cmath.

## Архитектура приложения

Для обеспечения требуемого функционала было принято решение разбить приложение на два класса:

- Processor (унаследованный от QWidget). Отвечает, как ясно из названия, за обработку данных, введенных пользователем;
- MainWindow (унаследованного от QMainWindow). Этот класс отвечает за отрисовку окна приложения.

На рисунке 3 покажем иерархию включения файлов и библиотек. Итоговую диаграмму классов приведем на рисунке 4.

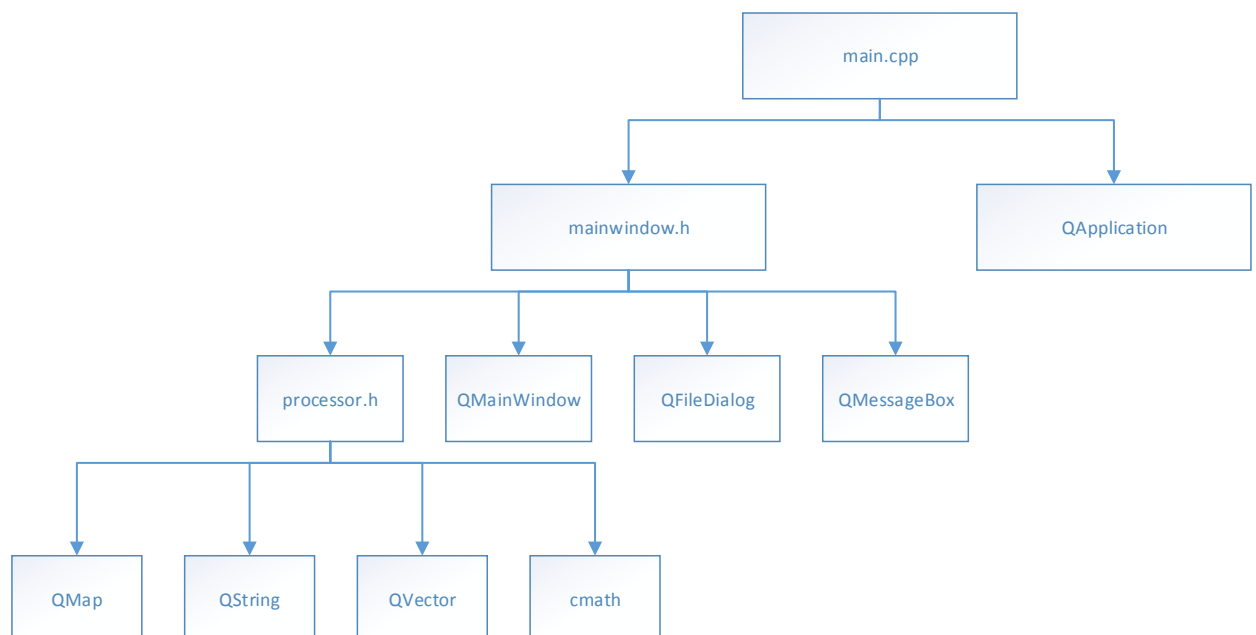


Рис. 3. Иерархия включения файлов и библиотек для main.cpp

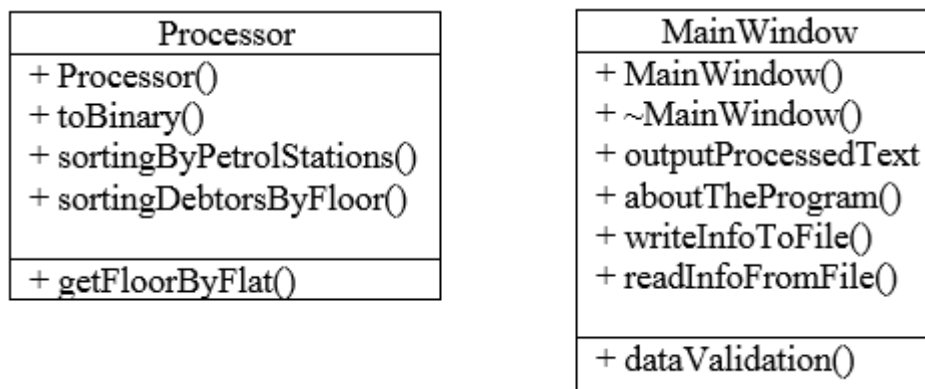


Рис. 4. Диаграмма классов

## Детали реализации

В первом задании, как видно из прототипа функции,  
`void NumberAsBinary(char* _result, // строка с результатом`  
`unsignedint _number // произвольное число`  
`)`

нужно передать в функцию указатель на массив типа `char`, при этом функция не возвращает никакого значения. Было решено, с учетом реализации GUI, заменить эту функцию на функцию `toBinary`, которая принимает число `int` и возвращает объект `QString`.

Второе задание было выполнено, но не хватило времени на то, чтобы реализовать дополнительную сортировку по компаниям, при условии, если одинаковые улицы. Функция обработки – `sortingByPetrolStations`.

Третье задание было выполнено полностью. Функция обработки – `sortingDebtorsByFloor`.

Ввиду крайней ограниченности времени не была сделана оптимизация алгоритмов и адекватная входных значений на валидность, хотя примитивная проверка данных на консистентность все-таки есть, а именно функция `dataValidation` проверяет введенные в левом окне данные и если они удовлетворяют какому-то из форматов задания, то приложение само определяет, какая функция отработки должна вызываться и выводит соответствующий результат в правое окно. В задании номер 3 не осуществляется округление задолженности до двух знаков после запятой. Также, несмотря на то, что Qt – кроссплатформенный фреймворк, не была проверена обратная совместимость с семейством ОС Linux, не сделан установочный пакет на базе WiX.

## Заключение

Таким образом, поставленное задание было выполнено. Репозиторий с историей изменений можно найти по адресу:

[https://bitbucket.org/ivanwolodin/test\\_v1336/src/master/](https://bitbucket.org/ivanwolodin/test_v1336/src/master/). Сама пояснительная записка находится по адресу:

[https://bitbucket.org/ivanwolodin/test\\_v1336/src/master/explanatory\\_note.pdf](https://bitbucket.org/ivanwolodin/test_v1336/src/master/explanatory_note.pdf)