Приложение Guess Word и его вариации

**Глава 3. Приложение Guess Word и его вариации**

Техническое задание:

* Иметь графический (оконный) интерфейс пользователя.
* Получать от пользователя некоторое количество букв русского алфавита.
* Выводить в специальном окне русские слова длинной 4 или 5 символов, которые можно составить из введенных букв.
* Использовать каждую из введенных букв в одном слове столько раз. сколь она была введена.
* Давать возможность пользователю выбирать параметр длины 4 или 5 (определяет, какие слова искать).
* Использовать для поиска список слов русского языка из отдельного файла.
* Выводить количество проверенных комбинаций.
* Выводить время исполнения в секундах.
* Иметь индикатор хода процесса.

Для начала нужно построить графический интерфейс пользователя в Qt Designer. Главным виджетом, как обычно, будет виджет QMainWindow. Внутри него все остальные виджеты (его дети) будут скомпонованы по сетке (Grid Layout). Выбрать виджеты и разместить их на окне QLabel, QLineEdit, выпадающий список (QComboBox), QComboBox, QPushButton, многострочное текстовое поле QPlainTextEdit (см.рис.1).

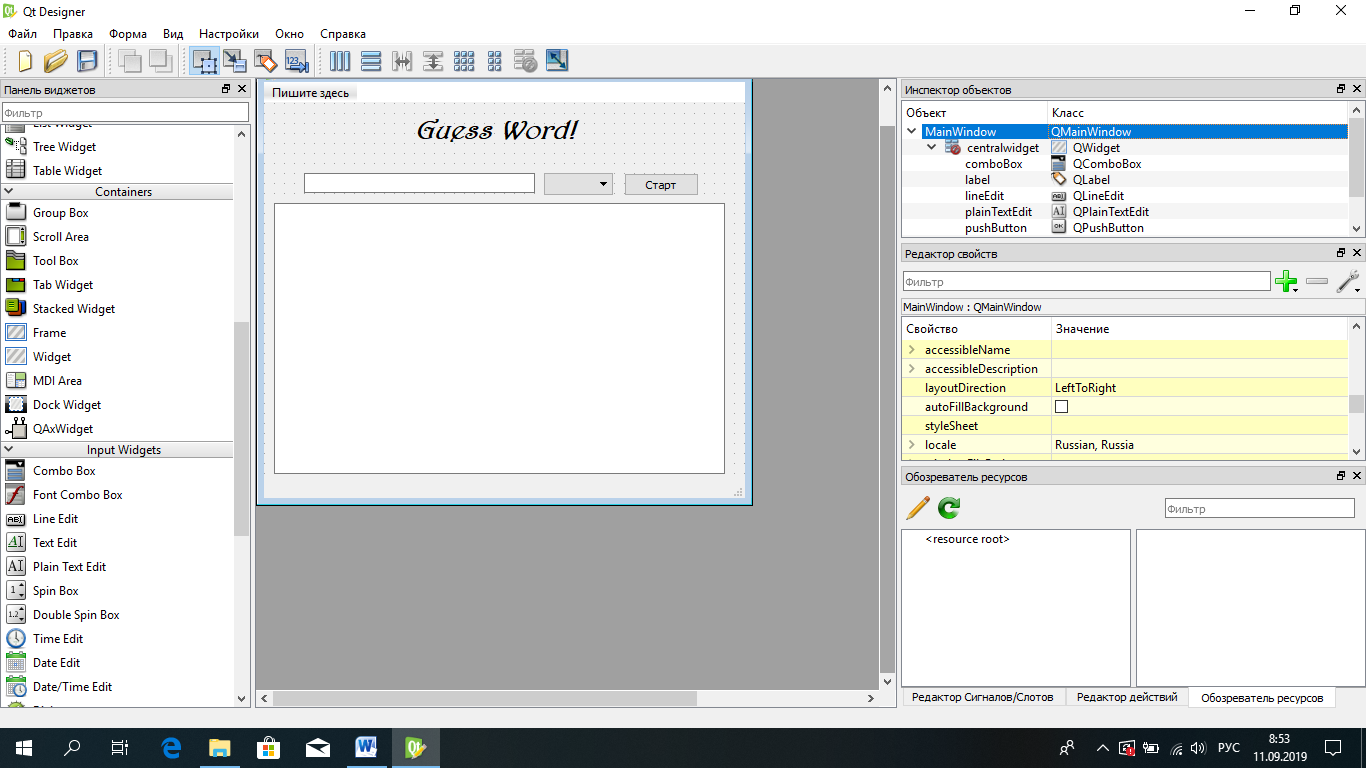


Рисунок 1. Интерфейс программы

Для того, чтобы при увеличении главного окна увеличивалось и все внутри него нажмите на объект MainWindow правой кнопкой мыши, в появившемся контекстном меню выберите пункт **Компоновка,** затем **Скомпоновать по сетке.** Выберите в меню Qt Designer **Форма -> Предпросмотр** или нажмите **Ctrl + R.** Попробуйте изменить размеры главного окна. Если Вы правильно выполнили все инструкции, то все виджеты будут пропорционально изменяться. Если Вам вдруг захочется отказаться от компоновки, то для этого есть команда **Компоновка —> Удалить компоновщик.**

Конвертируйте интерфейс с помощью командной строки в файл guess.ру. Далее создайте главный файл под названием guessmain.py, для этого воспользуйтесь кодом из файла myintmain.py, заменив в строке 2 импортируемый файл на guess. Интерфейс готов, программа запускается из файла guessmain.py, но конечно же, еще ничего не работает.

Для того, чтобы добавить значения в выпадающий список, можно воспользоваться методом **addltems()**. Параметром метода является массив (*список)* значений. Также полезно будет установить первое (т.е. нулевое!) значение списка как значение по умолчанию. Добавьте в функцию init () две строки:

self.ui.comboBox.setCurrentlndex(0)

self.ui.comboBox.addlterns('4 5'.split())

Обратите внимание, что для создания списка используется строка и метод **split ().**

Добавим очень полезный элемент программы, который еще называют строкой состояния. Ее особенность состоит в том, что при любых изменениях размера главного окна она всегда будет находится внизу интерфейса. Разместим в ней сведения об авторе программы, добавив функцию init () еще одну строку:

self.ui.statusbar.showMessage ('ФИО')

Последней строкой в функции \_\_ init () будет привязка функции к кнопке:

self.ui.pushButton.clicked.connect(self.start1)

Сама функция startl() еще не создана, поэтому на одном уровне с функцией init () мы создадим пустую функцию startl () (см. рис.2):

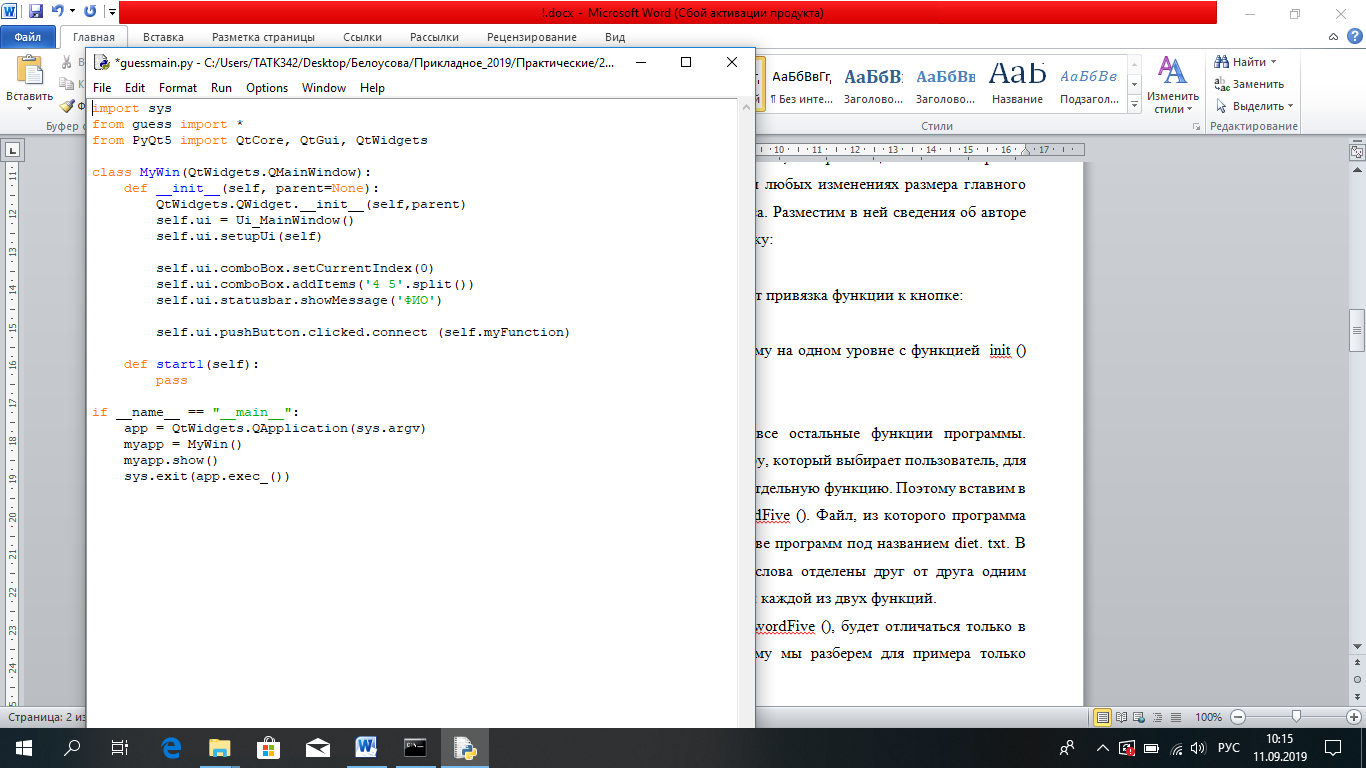


Рисунок 2. Код программы «Guessmain.py»

Вставим в код еще две пустые функции: wordFour () и wordFive ().

Создать файл dict.txt, в котором прописать русские слова через 1 пробел, из него программа будет брать слова для проверки.

Алгоритм, используемый в wordFour () и wordFive (), будет отличаться только в количестве вложенных циклов проверки, поэтому мы разберем для примера только функцию wordFour (). Вот ее основные блоки:

1. Проводится замер текущего времени и помещение полученной величины в переменную (для этого импортируется модуль time).
2. Происходит считывание содержимого файла dict.txt в список (массив).
3. С помощью четырех вложенных циклов for по очереди перебираются все возможные комбинации длинной 4 символа из введенных пользователем букв.
4. Внутри последнего вложенного цикла очередная составленная комбинация сравнивается с каждым словом из файла dict.txt. Если есть совпадение, то комбинация помещается в список (массив) результата.
5. Также внутри последнего вложенного цикла производится счет комбинаций.
6. Еще раз происходит замер текущего времени, полученная величина вычитается из первоначальной, разница дает время выполнения функции.
7. Производится вывод списка результата, количества проверенных комбинаций и времени выполнения.

В качестве аргумента функция будет получать текст, введенный пользователем.

Внутри функции startl() будет проводиться проверка текущего значения выпадающего списка. В зависимости от результата будет запускаться функция wordFour () или wordFive ().

Для того, чтобы элементы списка результата выводились через запятую, а после последнего стояла точка, создадим специальную функцию arrOutput (), которая будет перерабатывать список в одну строку.

Итак, на текущем этапе получаем следующий код (см. рис.3,4,5).

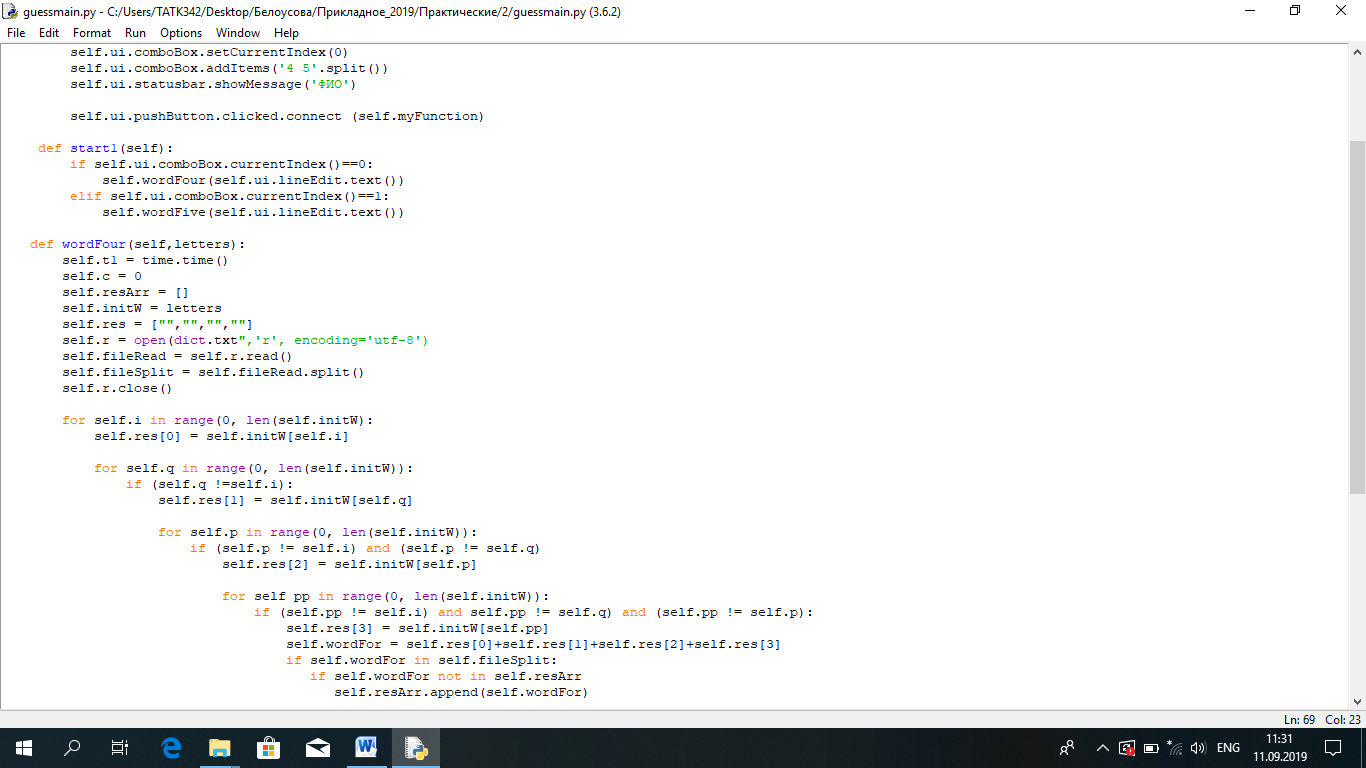


Рисунок 3. Процедура «start1»

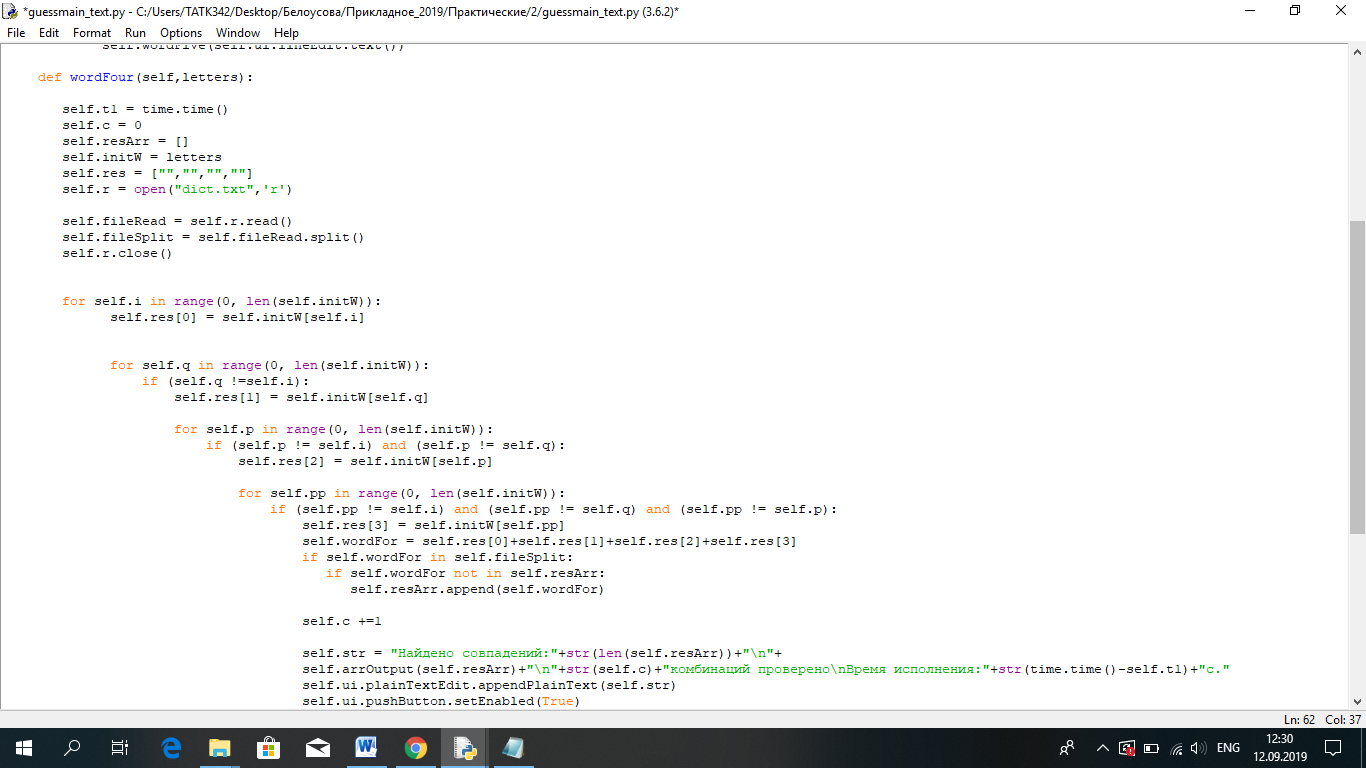


Рисунок 2. Функция wordFor

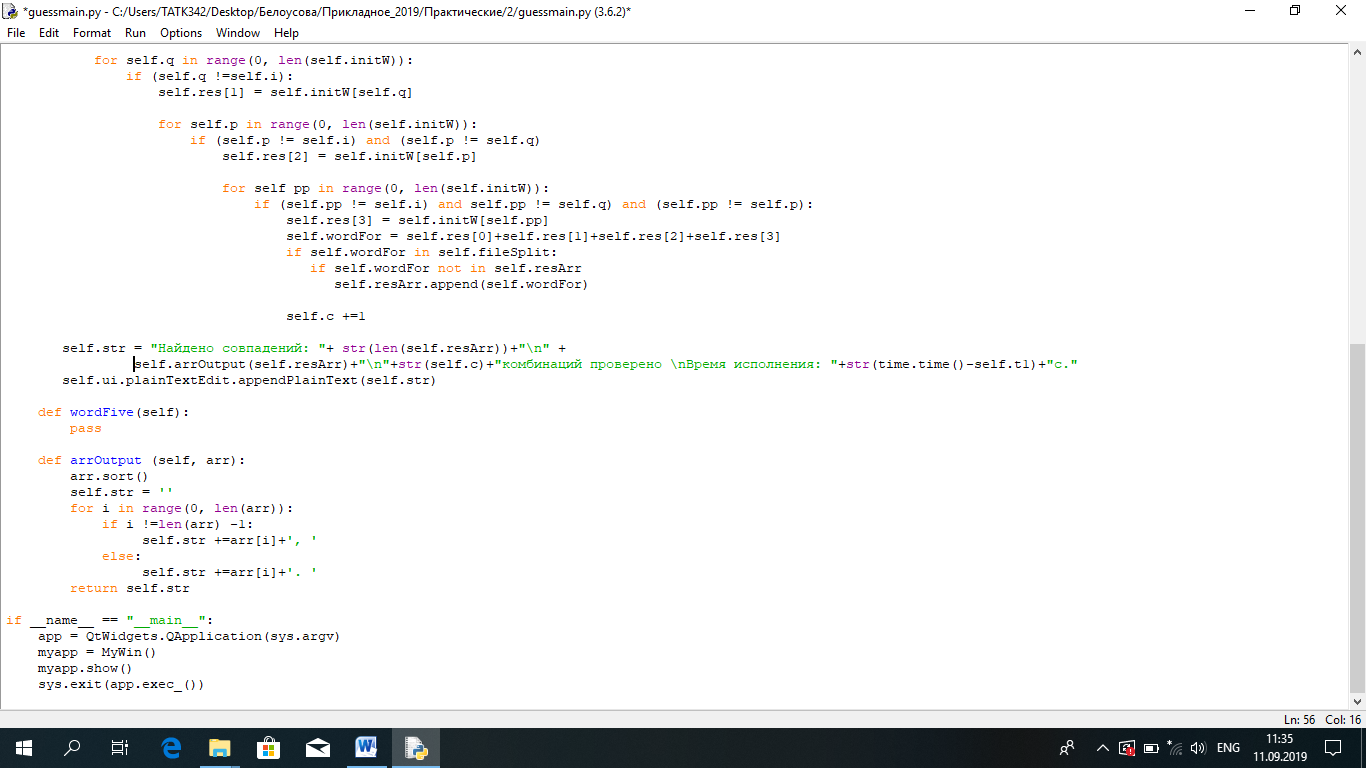


Рисунок 3. Функция arrOutput

Самостоятельно:

1. Аналогично построить функцию wordFive () (добавить еще один цикл for).
2. Добавьте к интерфейсу программы Guess Word кнопку, которая бы позволяла сохранить содержимое многострочного текстового поля в файл text.txt.