**Московский Технический Университет Связи и Информатики**

**Кафедра информатики**



*Лабораторная работа №8 по теме:*

«Создание приложений Windows для обработки строк»

Выполнил: Гоголев Георгий Михайлович

Группа ЗБИК2051

Выполненные варианты: №28 и №3

Проверил: Саркисова Ирина Олеговна

Оглавление

[1. Общее задание 3](#_Toc87469236)

[2. Разработка приложения 4](#_Toc87469237)

[2.1. Разработка графического интерфейса пользователя 4](#_Toc87469238)

[2.2. Описание функций, используемых в проекте 6](#_Toc87469239)

[2.3. Установка свойств объектов 6](#_Toc87469240)

[2.4. Разработка схем функциональных алгоритмов поставленной задачи 7](#_Toc87469241)

[2.5. Программный код 10](#_Toc87469242)

[3. Результаты работы проекта 17](#_Toc87469243)

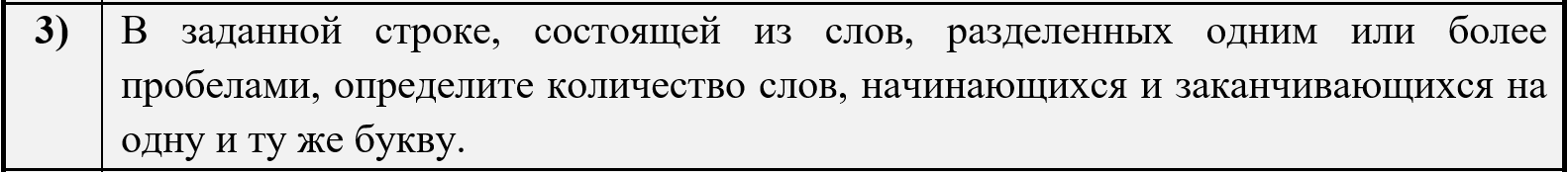
# 1. Общее задание

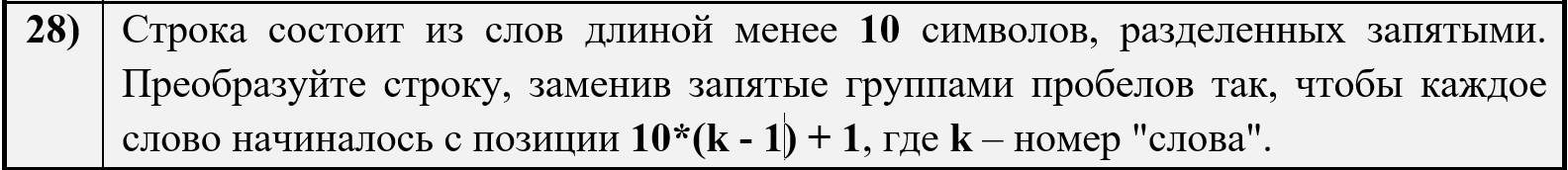
1. ***Изучите*** *вопросы программирование алгоритмов обработки символьных и строковых данных.*
2. ***Выберите вариант задания*** *из табл. 8.3-1.*
3. ***Проведите формализацию***поставленной задачи.
4. ***Разработайте графический интерфейс*** *пользователя.*
5. ***Разработайте схемы алгоритмов процедур*** *пользователя в соответствии с индивидуальным заданием.*
6. ***Разработайте программные коды*** *событийных процедур и пользовательских функций в соответствии с разработанными схемами алгоритмов.*
7. ***Разработайте проект приложения Windows Form****, содержащий:*

* ***графический интерфейс;***
* ***файл (файлы)*** *с раздельно откомпилированными программными кодами необходимых для решения задачи функций:*
* ***событийные процедуры,*** *которые должна содержать только операторы вызова пользовательских функций и контроля корректности исходных данных.*

1. ***Выполните*** приложение ***и получите*** результаты.
2. ***Докажите*** *правильность полученных результатов.*

В качестве разделителя выступают запятые. Фраза с пробелами и прочими символами считается за целое «слово» и воспринимается как одно «слово» (ограничение – 10 символов). Программа выполнена в среде Qt (Qt 5.12.3, MSVC 2017 x32).

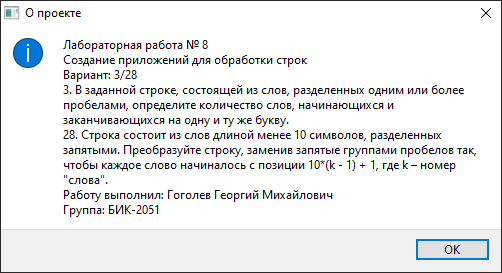




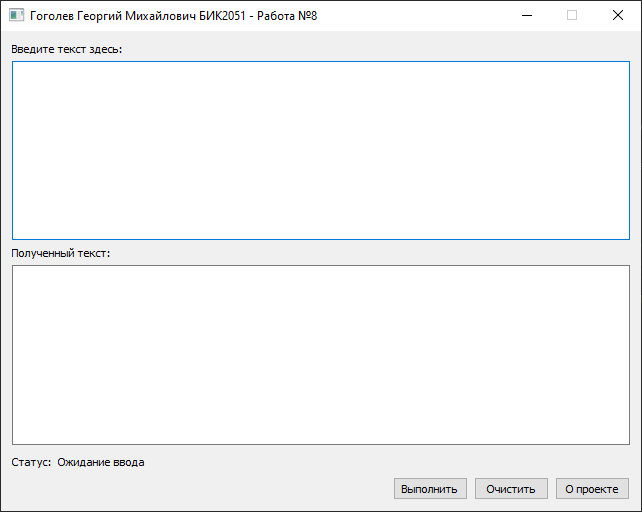
*рис. 1. Скриншоты выполненных в программе заданий*

# 2. Разработка приложения

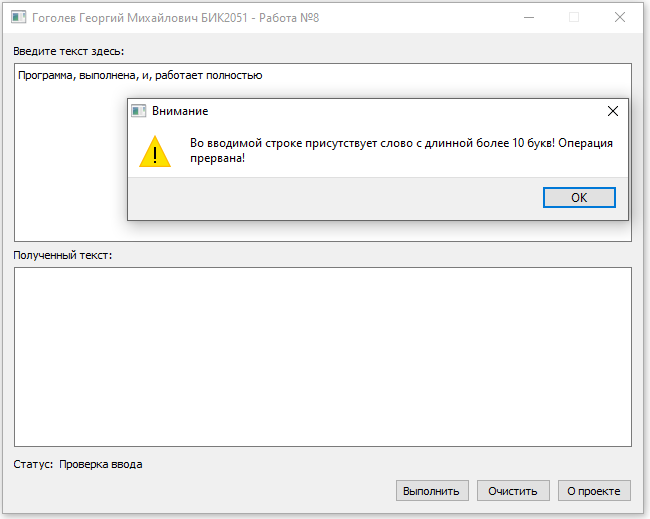
## 2.1. Разработка графического интерфейса пользователя



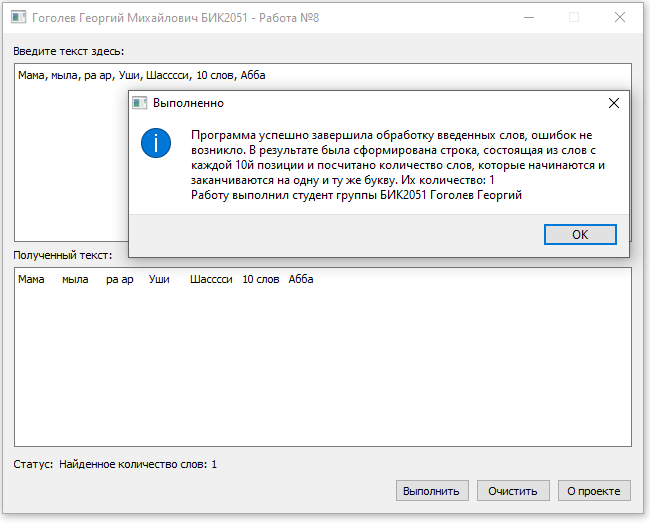
*рис. 2. Приветственное окно (кнопка «О проекте»)*



*рис. 3. Графический интерфейс пользователя. Окно развернуто до максимального размера*

**

*рис. 4. Ошибка ввода*

**

*рис. 5. Вывод результатов*

## 2.2. Описание функций, используемых в проекте

В проекте используются классы Core::cWord и наследуемый от него QcWord. Значимые функции этих классов:

bool Core::cWord::isStartAndEnd() const // Возвращает, является ли слово начинающимся и заканчивающимся на один символ

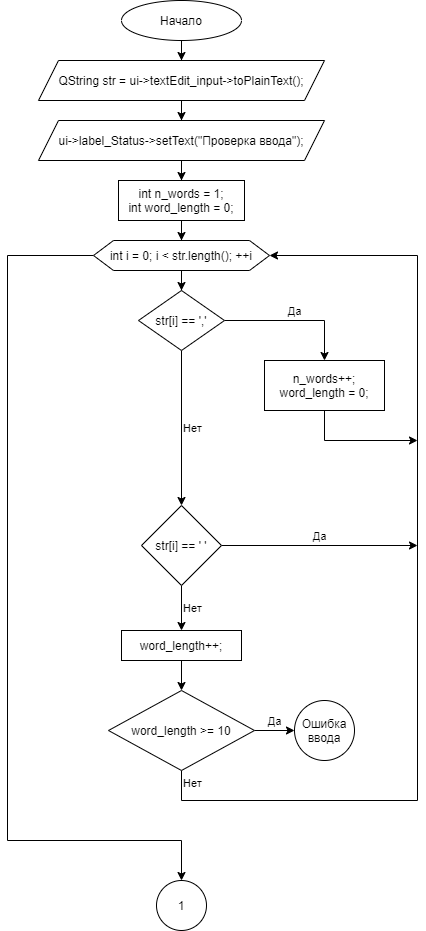
char\* Core::cWord::getFullWord(); // Возвращает слово, до 10 символа заполненное пробелами

## 2.3. Установка свойств объектов

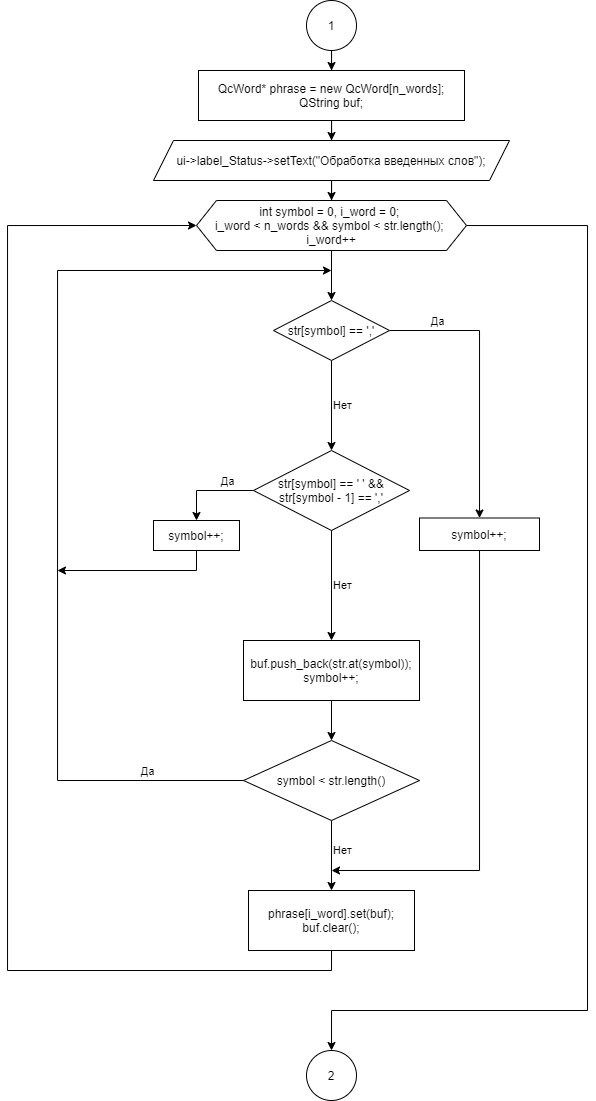
Таблица используемых объектов на MainForm и их свойств:

| **Объект** | **Имя объекта** | **Свойство** | **Значение свойства** |
| --- | --- | --- | --- |
| QWidget | centralWidget | Смысловая нагрузка | Контейнер, который организует объекты внутри себя |
| layoutName | verticalLayout |
| QHBoxLayout | horizontalLayout | Смысловая нагрузка | Контейнер, который формирует строку статуса программы |
| Spacer | horizontalSpacer | orientation | Horizontal |
| Смысловая нагрузка | Ужимает объекты на строке статуса |
| QLabel | label\_4 | text | Статус: |
| label\_Status | text | Ожидание ввода |
| QHBoxLayout | horizontalLayout\_2 | Смысловая нагрузка | Контейнер, который формирует строку управления программой |
| Spacer | horizontalSpacer\_2 | orientation | Horizontal |
| Смысловая нагрузка | Ужимает объекты на строке управления программой |
| QPushButton | pushButton\_Clear | text | Очистить |
| Смысловая нагрузка | Кнопка очистки данных |
| pushButton\_Work | text | Выполнить |
| Смысловая нагрузка | Кнопка выполнения программы |
| pushButton\_about | text | О проекте |
| Смысловая нагрузка | Кнопка вывода информации о программе |
| QLabel | label | text | Введите текст здесь: |
| label\_2 | text | Полученный текст: |
| QTextEdit | textEdit\_input | Смысловая нагрузка | Сюда происходит ввод слов |
| horizontalScrollBarPolicy | ScrollBarAlwaysOff |
| textEdit\_input | Смысловая нагрузка | Сюда происходит вывод |
| horizontalScrollBarPolicy | ScrollBarAlwaysOff |
| readOnly | True |

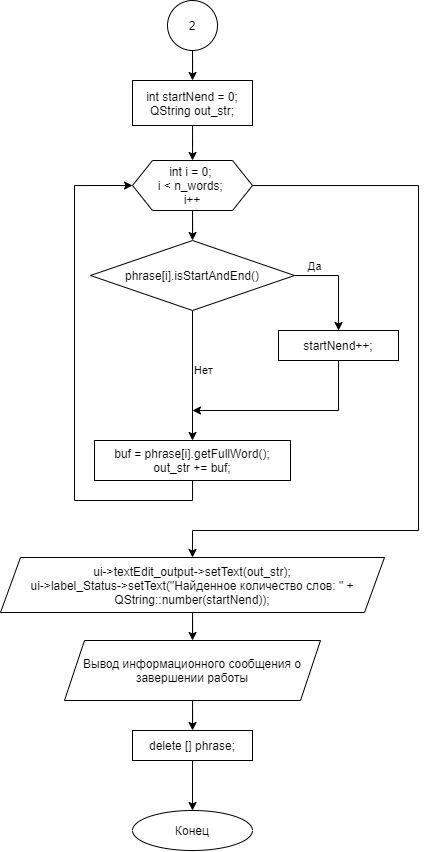
## 2.4. Разработка схем функциональных алгоритмов поставленной задачи



*рис. 6. Основной алгоритм. Подготовительная часть.*



*рис. 7. Основной алгоритм. Первая рабочая часть – разбиение «Фразы» на «Слова».*



*рис. 8. Основной алгоритм. Вторая рабочая часть – подсчет слов, которые начинаются и заканчиваются на один и тот же символ, а так же дополнение «Слов» пробелами до 10 знаков. Вывод результатов.*



*рис. 9. Основной алгоритм. Вывод ошибки ввода*

## 2.5. Программный код

**Файл main.cpp**

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(*argc*, argv);

MainWindow w;

w.show();

*return* a.exec();

}

**Файл Core.h**

#pragma once

#define LAB\_NUM "8"

#define LAB\_TEXT "Создание приложений для обработки строк"

#define VAR "3/28"

#define VAR\_TEXT "3. В заданной строке, состоящей из слов, разделенных одним или более пробелами, определите количество слов, начинающихся и заканчивающихся на одну и ту же букву.\n28. Строка состоит из слов длиной менее 10 символов, разделенных запятыми. Преобразуйте строку, заменив запятые группами пробелов так, чтобы каждое слово начиналось с позиции 10\*(k - 1) + 1, где k – номер \"слова\"."

#define FIO "Гоголев Георгий Михайлович"

#define GROUP "БИК-2051"

*namespace* **Core** {

int **GetRandom**(int min = -15000, int max = 15000);

*class* **cWord**{

*public*:

**cWord**();

**cWord**(char\* word);

~**cWord**();

*private*: char\* Value;

*protected*:

int Length;

bool IsWord;

bool IsStartAndEnd;

void **UpdateWord**();

*public*:

void **set**(char \*new\_word);

char\* **get**() *const*;

void **get**(char\* get\_word) *const*;

int **getLenght**() *const*;

bool **isWord**() *const*;

bool **isStartAndEnd**() *const*;

char\* **getFullWord**();

};

}

**Файл Core.cpp**

#include "Core.h"

#include <cstdlib>

#include <cstring>

int Core::**GetRandom**(int min, int max)

{

*return* min + rand() % max;

}

Core::cWord::**cWord**()

{

Value = *new* char[10];

Length = 0;

IsWord = *false*;

IsStartAndEnd = *false*;

}

Core::cWord::**cWord**(char\* word)

{

Value = *new* char[10];

*for* (int i = 0; i < 10 && word[i] != '\0'; i++)

{

Value[i] = word[i];

}

UpdateWord();

}

Core::cWord::~**cWord**()

{

*delete*[] Value;

}

void Core::cWord::**UpdateWord**()

{

Length = 0;

IsWord = *true*;

IsStartAndEnd = *false*;

*for* (int i = 0; Value[i] != '\0' && i < 10; i++)

{

Length++;

*if* (int(Value[i]) < 65

|| (int(Value[i]) > 90 && int(Value[i]) < 97)

|| (int(Value[i]) > 122 && int(Value[i]) < 128)

|| (int(Value[i]) > 175 && int(Value[i]) < 224)

|| int(Value[i]) > 241)

{

IsWord = *false*;

}

}

*if* (Value[0] == Value[Length - 1]) IsStartAndEnd = *true*;

*return*;

}

void Core::cWord::**set**(char \*new\_word)

{

*for* (int i = 0; i < 10 && new\_word[i] != '\0'; i++)

{

Value[i] = new\_word[i];

}

UpdateWord();

*return*;

}

char\* Core::cWord::**get**() *const*

{

*return* Value;

}

void Core::cWord::**get**(char\* get\_word) *const*

{

std::strcpy(get\_word, Value);

*return*;

}

int Core::cWord::**getLenght**() *const*

{

*return* Length;

}

bool Core::cWord::**isWord**() *const*

{

*return* IsWord;

}

bool Core::cWord::**isStartAndEnd**() *const*

{

*return* IsStartAndEnd;

}

char\* Core::cWord::**getFullWord**()

{

char\* out = *new* char[10];

*for* (int i = 0; i < 10; i++)

{

*if* (i < Length) out[i] = Value[i];

*else* out[i] = 32;

}

*return* out;

}

**Файл qcword.h**

#ifndef QCWORD\_H

#define QCWORD\_H

#include "Core.h"

#include "QString"

*class* **QcWord** : *public* Core::cWord

{

*public*:

**QcWord**();

**QcWord**(QString word);

~**QcWord**();

*private*:

QString QValue;

void **UpdateQWord**();

*public*:

void **set**(QString new\_word);

QString **getFullWord**();

};

#endif *//* *QCWORD\_H*

**Файл qcword.cpp**

#include "qcword.h"

QcWord::**QcWord**()

{

QValue.clear();

Length = 0;

IsWord = *false*;

IsStartAndEnd = *false*;

}

QcWord::**QcWord**(QString word)

{

QValue.clear();

QValue.push\_back(word);

UpdateQWord();

}

QcWord::~**QcWord**()

{

QValue.~QString();

}

void QcWord::**UpdateQWord**()

{

Length = QValue.length();

*//IsWord* *=* *true;*

*if* (QValue.front() == QValue.back()) IsStartAndEnd = *true*;

*else* IsStartAndEnd = *false*;

*return*;

}

void QcWord::**set**(QString new\_word)

{

QValue.clear();

QValue.push\_back(new\_word);

UpdateQWord();

*return*;

}

QString QcWord::**getFullWord**()

{

QString out;

out.push\_back(QValue);

*for* (int i = out.length(); i < 10; i++)

{

out.push\_back(" ");

}

*return* out;

}

**Файл mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

*namespace* **Ui** {

*class* **MainWindow**;

}

*class* **MainWindow** : *public* QMainWindow

{

Q\_OBJECT

*public*:

*explicit* **MainWindow**(QWidget \*parent = *nullptr*);

~***MainWindow***();

*private* slots:

void **on\_pushButton\_Work\_clicked**();

void **on\_pushButton\_Clear\_clicked**();

void **on\_pushButton\_about\_clicked**();

*private*:

Ui::MainWindow \*ui;

};

#endif *//* *MAINWINDOW\_H*

**Файл mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "QMessageBox"

#include "qcword.h"

MainWindow::**MainWindow**(QWidget \*parent) :

QMainWindow(parent),

ui(*new* Ui::MainWindow)

{

ui->setupUi(*this*);

on\_pushButton\_about\_clicked();

}

MainWindow::~***MainWindow***()

{

*delete* ui;

}

void MainWindow::**on\_pushButton\_Work\_clicked**()

{

*//* *Считываем* *строку* *из* *формы*

QString str = ui->textEdit\_input->toPlainText();

*//* *Проверяем* *ввобимые* *слова* *и* *попутно* *находим* *количество* *слов*

ui->label\_Status->setText("Проверка ввода");

int n\_words = 1;

int word\_length = 0;

*for* (int i = 0; i < str.length(); ++i) {

*if* (str[i] == ',') {

n\_words++;

word\_length = 0;

*continue*;

}

*if* (str[i] == ' ') *continue*;

word\_length++;

*if* (word\_length >= 10) {

QMessageBox::warning(*this*, "Внимание", "Во вводимой строке присутствует слово с длинной более 10 букв! Операция прервана!");

ui->label\_Status->setText("Ошибка ввода.");

*return*;

}

}

*//* *Формируем* *"фразу"* *из* *"слов"*

QcWord\* phrase = *new* QcWord[n\_words];

QString buf;

ui->label\_Status->setText("Обработка введенных слов");

*for* (int symbol = 0, i\_word = 0; i\_word < n\_words && symbol < str.length(); ++i\_word) {

*do*{

*if* (str[symbol] == ',') {

symbol++;

*break*;

}

*if* (str[symbol] == ' ' && str[symbol - 1] == ',') {

symbol++;

*continue*;

}

buf.push\_back(str.at(symbol));

symbol++;

} *while* (symbol < str.length());

phrase[i\_word].set(buf);

buf.clear();

}

*//* *Теперь* *найдем* *количество* *слов* *с* *одинаковой* *первой* *и* *последней* *буквой* *и* *сформируем* *новую* *строку* *для* *вывода*

int startNend = 0;

QString out\_str;

*for* (int i = 0; i < n\_words; ++i) {

*if* (phrase[i].isStartAndEnd()) startNend++;

buf = phrase[i].getFullWord();

out\_str += buf;

}

*//* *Вывод*

ui->textEdit\_output->setText(out\_str);

ui->label\_Status->setText("Найденное количество слов: " + QString::number(startNend));

QMessageBox::information(*this*, "Выполненно", "Программа успешно завершила обработку введенных слов, ошибок не возникло. В результате была сформирована строка,

состоящая из слов с каждой 10й позиции и посчитано количество слов, которые начинаются и заканчиваются на одну и ту же букву. Их количество: " + QString::number(startNend) + "\nРаботу выполнил студент группы БИК2051 Гоголев Георгий");

*//* *Очистка* *памяти* *и* *выход*

*delete* [] phrase;

*return*;

}

void MainWindow::**on\_pushButton\_Clear\_clicked**()

{

ui->textEdit\_input->clear();

ui->textEdit\_output->clear();

ui->label\_Status->setText("Готово к вводу");

*return*;

}

void MainWindow::**on\_pushButton\_about\_clicked**()

{

QString about\_text;

about\_text.push\_back("Лабораторная работа № "); about\_text.push\_back(LAB\_NUM); about\_text += "\n";

about\_text.push\_back(LAB\_TEXT); about\_text += "\n";

about\_text.push\_back("Вариант: "); about\_text.push\_back(VAR); about\_text += "\n";

about\_text.push\_back(VAR\_TEXT); about\_text += "\n";

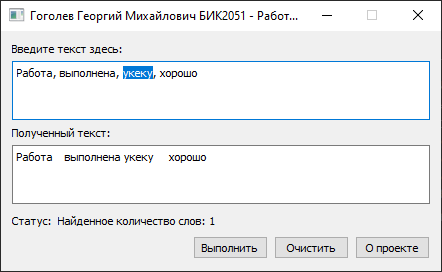
about\_text.push\_back("Работу выполнил: "); about\_text.push\_back(FIO); about\_text += "\n";

about\_text.push\_back("Группа: "); about\_text.push\_back(GROUP);

QMessageBox::information(*this*, "О проекте", about\_text);

}

# 3. Результаты работы проекта



*рис. 10. Результаты работы программы*

Правильность работы программы подтверждается приведенными выше результатами выполнения приложения. Было введена «Фраза» из четырех «Слов», среди которых – «укеку» начинается и заканчивается на одну и ту же букву, и «выполнена» – состоит из 9 букв («Символов»). По этим двум словам видно, что программа корректно добавляет пробелы до необходимого количества, а также определяет количество слов, у которых первый и последний символ одинаковый.