|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования  **«Российский технологический университет - МИРЭА»** |
| **РТУ МИРЭА** |

Институт информационных технологий (ИИТ)Кафедра вычислительной техники (ВТ)

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №6**

**по дисциплине**

«Инструментальное программное обеспечение разработки и проектирования информационных систем»

Выполнил студент группы ИКМО-01-20 Баранецкий Д.А.

Фамилия И.О.

Принял Куликов А.А.

Фамилия И.О.

Работы выполнены «\_\_»\_\_\_\_\_\_\_2021 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

«Зачтено» «\_\_»\_\_\_\_\_\_\_2021 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Москва, 2021 г.

Оглавление

[Задание 3](#_Toc71570186)

[Код программы 3](#_Toc71570187)

[Скриншоты 12](#_Toc71570188)

# Задание

**Цель практическое работы:**

Создать полноценный текстовый редактор на языке Python и разработать для него форму в QT Designer.

На форме должны присутствовать элементы: открыть файл, сохранить (QPushButton) и textBrowser(QTextBrowser). Так же элементы для стиля текста.

**Задачи:**

* Реализовать основной набор функций редактора:
  + Открыть файл
  + Сохранить файл
  + Сохранить файл как
* Реализовать набор функций редактора по стилизации:
  + Изменить шрифт
  + Изменить размер шрифта
  + Изменить начертание шрифта
  + Изменить положение текста:
    - По центру
    - По ширине
    - По левому краю
    - По правому краю
  + Создать нумерованный список
  + Создать маркированный список
  + Изменить цвет шрифта
* Реализовать форму добавления текста

# Код программы

TextEditorInterface.py – файл, формирующий QT-форму редактора

# -\*- coding: utf-8 -\*-  
  
# Form implementation generated from reading ui file 'TextEditorUI.ui'  
#  
# Created by: PyQt5 UI code generator 5.15.0  
#  
# WARNING: Any manual changes made to this file will be lost when pyuic5 is  
# run again. Do not edit this file unless you know what you are doing.  
  
  
from PyQt5 import QtCore, QtGui, QtWidgets  
  
  
class Ui\_TextEditor(object):  
 def setupUi(self, TextEditor):  
 TextEditor.setObjectName("TextEditor")  
 TextEditor.resize(460, 410)  
 icon = QtGui.QIcon()  
 icon.addPixmap(QtGui.QPixmap("icons/notepad-96.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)  
 TextEditor.setWindowIcon(icon)  
 self.centralwidget = QtWidgets.QWidget(TextEditor)  
 self.centralwidget.setObjectName("centralwidget")  
 self.verticalLayout = QtWidgets.QVBoxLayout(self.centralwidget)  
 self.verticalLayout.setObjectName("verticalLayout")  
 self.TextEdit = QtWidgets.QTextEdit(self.centralwidget)  
 self.TextEdit.setObjectName("TextEdit")  
 self.verticalLayout.addWidget(self.TextEdit)  
 TextEditor.setCentralWidget(self.centralwidget)  
 self.toolBar = QtWidgets.QToolBar(TextEditor)  
 self.toolBar.setObjectName("toolBar")  
 TextEditor.addToolBar(QtCore.Qt.TopToolBarArea, self.toolBar)  
 self.statusBar = QtWidgets.QStatusBar(TextEditor)  
 self.statusBar.setObjectName("statusBar")  
 TextEditor.setStatusBar(self.statusBar)  
  
 self.retranslateUi(TextEditor)  
 QtCore.QMetaObject.connectSlotsByName(TextEditor)  
  
 def retranslateUi(self, TextEditor):  
 \_translate = QtCore.QCoreApplication.translate  
 TextEditor.setWindowTitle(\_translate("TextEditor", "Текстовый редактор"))  
 self.toolBar.setWindowTitle(\_translate("TextEditor", "toolBar"))  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 import sys  
  
 app = QtWidgets.QApplication(sys.argv)  
 TextEditor = QtWidgets.QMainWindow()  
 ui = Ui\_TextEditor()  
 ui.setupUi(TextEditor)  
 TextEditor.show()  
 sys.exit(app.exec\_())

main.py – файл, содержащий основную логику работы редактора с поясняющими комментариями

import codecs  
from os.path import splitext  
from PyQt5 import QtWidgets, QtGui, QtCore  
import sys  
import TextEditorInterface  
  
  
class TextEditor(QtWidgets.QMainWindow, TextEditorInterface.Ui\_TextEditor):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.setupUi(self)  
 self.selected\_file = None  
 self.html\_extensions = [".html", ".htm", ".php", ".aspx", ".asp", ".jsp"]  
 font\_sizes = ["8", "9", "10", "11", "12", "14", "16", "18", "20", "22", "24", "26", "28", "36", "48", "72"]  
 self.colors = {"Black": "#000000",  
 "Blue": "#0000ff",  
 "Green": "#008000",  
 "Red": "#ff0000",  
 }  
 self.statusBar.showMessage("Строка: 1 | Столбец: 1 | Позиция: 0 ")  
 self.title = self.windowTitle()  
 self.resize(800, 600)  
  
 self.TextEdit.setAutoFormatting(QtWidgets.QTextEdit.AutoAll)  
 self.TextEdit.cursorPositionChanged.connect(self.cursor\_position)  
 self.TextEdit.selectionChanged.connect(self.update\_selection)  
  
 #добавление кнопки "Открыть"  
 self.open\_file\_action = QtWidgets.QAction("Открыть")  
 self.open\_file\_action.triggered.connect(self.open\_file)  
 self.open\_file\_action.setShortcut("CTRL+O")  
 self.open\_file\_action.setIcon(QtGui.QIcon("./icons/denis-opened-folder-96.png"))  
  
 # добавление кнопки "Сохранить"  
 self.save\_file\_action = QtWidgets.QAction("Сохранить")  
 self.save\_file\_action.triggered.connect(self.save\_file)  
 self.save\_file\_action.setShortcut("CTRL+S")  
 self.save\_file\_action.setIcon(QtGui.QIcon("./icons/denis-save-96.png"))  
  
 # добавление кнопки "Сохранить как"  
 self.save\_as\_file\_action = QtWidgets.QAction("Сохранить как")  
 self.save\_as\_file\_action.triggered.connect(self.save\_as\_file)  
 self.save\_as\_file\_action.setIcon(QtGui.QIcon("./icons/denis-save-as-96.png"))  
  
 # добавление кнопки "Жирный"  
 self.bold\_action = QtWidgets.QAction("Жирный")  
 self.bold\_action.setIcon(QtGui.QIcon("./icons/denis-bold-96.png"))  
 self.bold\_action.setShortcut("CTRL+B")  
 self.bold\_action.setCheckable(True)  
 self.bold\_action.toggled.connect(  
 lambda x: self.TextEdit.setFontWeight(QtGui.QFont.Bold if x else QtGui.QFont.Normal))  
  
 # добавление кнопки "Курсив"  
 self.italic\_action = QtWidgets.QAction("Курсив")  
 self.italic\_action.setIcon(QtGui.QIcon("./icons/denis-italic-96.png"))  
 self.italic\_action.setShortcut("CTRL+I")  
 self.italic\_action.setCheckable(True)  
 self.italic\_action.toggled.connect(self.TextEdit.setFontItalic)  
  
 # добавление кнопки "Подчёркнутый"  
 self.under\_line\_action = QtWidgets.QAction("Подчёркнутый")  
 self.under\_line\_action.setIcon(QtGui.QIcon("./icons/denis-underline-96.png"))  
 self.under\_line\_action.setShortcut("CTRL+U")  
 self.under\_line\_action.setCheckable(True)  
 self.under\_line\_action.toggled.connect(self.TextEdit.setFontUnderline)  
  
 # добавление кнопки "Перечисляемый список"  
 self.bullet\_list\_action = QtWidgets.QAction("Перечисляемый список")  
 self.bullet\_list\_action.setShortcut("Ctrl+Shift+B")  
 self.bullet\_list\_action.setIcon(QtGui.QIcon("./icons/denis-bulleted-list-96.png"))  
 self.bullet\_list\_action.setCheckable(True)  
 self.bullet\_list\_action.triggered.connect(self.set\_bullet\_list)  
  
 # добавление кнопки "Вставить нумерованный список"  
 self.numbered\_list\_action = QtWidgets.QAction("Вставить нумерованный список")  
 self.numbered\_list\_action.setIcon(QtGui.QIcon("./icons/denis-numbered-list-96.png"))  
 self.numbered\_list\_action.setShortcut("Ctrl+Shift+L")  
 self.numbered\_list\_action.triggered.connect(self.set\_number\_list)  
 self.numbered\_list\_action.setCheckable(True)  
  
  
 self.list\_group\_action = QtWidgets.QActionGroup(self)  
 self.list\_group\_action.setExclusive(True)  
 self.list\_group\_action.addAction(self.bullet\_list\_action)  
 self.list\_group\_action.addAction(self.numbered\_list\_action)  
  
 # добавление кнопки "Выровнять по левому краю"  
 self.left\_alignment\_action = QtWidgets.QAction("Выровнять по левому краю")  
 self.left\_alignment\_action.triggered.connect(  
 lambda: self.TextEdit.setAlignment(QtCore.Qt.AlignmentFlag.AlignLeft))  
 self.left\_alignment\_action.setIcon(QtGui.QIcon("./icons/denis-align-left-96.png"))  
 self.left\_alignment\_action.setShortcut("CTRL+L")  
 self.left\_alignment\_action.setCheckable(True)  
  
 # добавление кнопки "Выровнять по центру"  
 self.center\_alignment\_action = QtWidgets.QAction("Выровнять по центру")  
 self.center\_alignment\_action.triggered.connect(  
 lambda: self.TextEdit.setAlignment(QtCore.Qt.AlignmentFlag.AlignCenter))  
 self.center\_alignment\_action.setIcon(QtGui.QIcon("./icons/denis-align-center-96.png"))  
 self.center\_alignment\_action.setShortcut("CTRL+E")  
 self.center\_alignment\_action.setCheckable(True)  
  
 # добавление кнопки "Выровнять по правому краю"  
 self.right\_alignment\_action = QtWidgets.QAction("Выровнять по правому краю")  
 self.right\_alignment\_action.triggered.connect(  
 lambda: self.TextEdit.setAlignment(QtCore.Qt.AlignmentFlag.AlignRight))  
 self.right\_alignment\_action.setIcon(QtGui.QIcon("./icons/denis-align-right-96.png"))  
 self.right\_alignment\_action.setShortcut("CTRL+R")  
 self.right\_alignment\_action.setCheckable(True)  
  
 # добавление кнопки "Выровнять по ширине"  
 self.justify\_alignment\_action = QtWidgets.QAction("Выровнять по ширине")  
 self.justify\_alignment\_action.triggered.connect(  
 lambda: self.TextEdit.setAlignment(QtCore.Qt.AlignmentFlag.AlignJustify))  
 self.justify\_alignment\_action.setIcon(QtGui.QIcon("./icons/denis-align-justify-96.png"))  
 self.justify\_alignment\_action.setShortcut("CTRL+J")  
 self.justify\_alignment\_action.setCheckable(True)  
  
  
 self.alignment\_group\_action = QtWidgets.QActionGroup(self)  
 self.alignment\_group\_action.setExclusive(True)  
 self.alignment\_group\_action.addAction(self.left\_alignment\_action)  
 self.alignment\_group\_action.addAction(self.right\_alignment\_action)  
 self.alignment\_group\_action.addAction(self.center\_alignment\_action)  
 self.alignment\_group\_action.addAction(self.justify\_alignment\_action)  
  
 # добавление кнопки "Шрифт"  
 self.font\_combobox = QtWidgets.QFontComboBox()  
 self.font\_combobox.setToolTip("Шрифт")  
 self.font\_combobox.setEditable(False)  
 self.font\_combobox.setCurrentFont(self.TextEdit.currentFont())  
 self.font\_combobox.currentFontChanged.connect(self.set\_font)  
  
 # добавление кнопки "Размер шрифта"  
 self.font\_size\_combobox = QtWidgets.QComboBox()  
 self.font\_size\_combobox.setToolTip("Размер шрифта")  
 self.font\_size\_combobox.setEditable(False)  
 self.font\_size\_combobox.addItems(font\_sizes)  
 self.font\_size\_combobox.currentTextChanged.connect(self.set\_font\_size)  
  
 # добавление кнопки "Цвет текста"  
 self.text\_color\_combobox = QtWidgets.QComboBox()  
 self.text\_color\_combobox.setToolTip("Цвет текста")  
 self.text\_color\_combobox.setEditable(False)  
 for color in self.colors:  
 self.text\_color\_combobox.addItem(QtGui.QIcon("./icons/text-color-48.png"), color)  
 self.text\_color\_combobox.currentTextChanged.connect(self.paint\_text)  
  
 self.toolBar.addAction(self.open\_file\_action)  
 self.toolBar.addAction(self.save\_file\_action)  
 self.toolBar.addAction(self.save\_as\_file\_action)  
 self.toolBar.addSeparator()  
  
 self.toolBar.addWidget(self.font\_combobox)  
 self.toolBar.addWidget(self.font\_size\_combobox)  
 self.toolBar.addSeparator()  
  
 self.toolBar.addAction(self.bold\_action)  
 self.toolBar.addAction(self.italic\_action)  
 self.toolBar.addAction(self.under\_line\_action)  
 self.toolBar.addSeparator()  
  
 self.toolBar.addAction(self.bullet\_list\_action)  
 self.toolBar.addAction(self.numbered\_list\_action)  
 self.toolBar.addSeparator()  
  
 self.toolBar.addAction(self.left\_alignment\_action)  
 self.toolBar.addAction(self.center\_alignment\_action)  
 self.toolBar.addAction(self.right\_alignment\_action)  
 self.toolBar.addAction(self.justify\_alignment\_action)  
 self.toolBar.addSeparator()  
  
 self.toolBar.addWidget(self.text\_color\_combobox)  
 self.format\_actions = [  
 self.font\_combobox,  
 self.font\_size\_combobox,  
 self.text\_color\_combobox,  
 self.bold\_action,  
 self.italic\_action,  
 self.under\_line\_action,  
 ]  
 self.msg\_box = QtWidgets.QMessageBox()  
 self.msg\_box.setWindowTitle(self.windowTitle())  
 self.msg\_box.setWindowIcon(self.windowIcon())  
  
 def set\_win\_title(self):  
 self.setWindowTitle(self.selected\_file + "-" + self.title)  
  
 def set\_bullet\_list(self) -> None:  
 cursor = self.TextEdit.textCursor()  
 cursor.insertList(QtGui.QTextListFormat.ListDisc)  
  
 def set\_number\_list(self) -> None:  
 cursor = self.TextEdit.textCursor()  
 cursor.insertList(QtGui.QTextListFormat.ListDecimal)  
  
 def cursor\_position(self) -> None:  
 cursor = self.TextEdit.textCursor()  
 line = cursor.blockNumber() + 1  
 column = cursor.columnNumber() + 1  
 position = cursor.position()  
 self.statusBar.showMessage(f"Строка: {line} | Столбец: {column} | Позиция: {position}")  
  
 def show\_msg(self, msg: str) -> None:  
 self.msg\_box.setText(msg)  
 self.msg\_box.exec()  
  
 @staticmethod  
 def block\_signals(objects: QtWidgets, block: bool) -> None:  
 for obj in objects:  
 obj.blockSignals(block)  
  
 # обновление выделенной области  
 def update\_selection(self):  
 self.block\_signals(self.format\_actions, True)  
 self.font\_combobox.setCurrentFont(self.TextEdit.currentFont())  
 self.font\_size\_combobox.setCurrentText(str(int(self.TextEdit.fontPointSize())))  
 self.italic\_action.setChecked(self.TextEdit.fontItalic())  
 self.under\_line\_action.setChecked(self.TextEdit.fontUnderline())  
 self.bold\_action.setChecked(self.TextEdit.fontWeight() == QtGui.QFont.Bold)  
  
 self.left\_alignment\_action.setChecked(  
 self.TextEdit.alignment().\_\_index\_\_() == QtCore.Qt.AlignmentFlag.AlignLeft.\_\_index\_\_())  
 self.right\_alignment\_action.setChecked(  
 self.TextEdit.alignment().\_\_index\_\_() == QtCore.Qt.AlignmentFlag.AlignRight.\_\_index\_\_())  
 self.center\_alignment\_action.setChecked(  
 self.TextEdit.alignment().\_\_index\_\_() == QtCore.Qt.AlignmentFlag.AlignCenter.\_\_index\_\_())  
 self.justify\_alignment\_action.setChecked(  
 self.TextEdit.alignment().\_\_index\_\_() == QtCore.Qt.AlignmentFlag.AlignJustify.\_\_index\_\_())  
  
 for k, v in self.colors.items():  
 if v == self.TextEdit.textColor().name():  
 self.text\_color\_combobox.setCurrentText(k)  
 cursor = self.TextEdit.textCursor()  
 if not cursor.currentList():  
 self.bullet\_list\_action.setChecked(False)  
 self.numbered\_list\_action.setChecked(False)  
 self.block\_signals(self.format\_actions, False)  
  
 def set\_font(self, font: QtGui.QFont) -> None:  
 self.TextEdit.setCurrentFont(font)  
 self.TextEdit.setFocus()  
  
 def set\_font\_size(self, size: str) -> None:  
 self.TextEdit.setFontPointSize(float(size))  
 self.TextEdit.setFocus()  
  
 def paint\_text(self, color: str) -> None:  
 self.TextEdit.setTextColor(QtGui.QColor(color))  
 self.TextEdit.setFocus()  
  
 def paint\_back(self, color: str) -> None:  
 self.TextEdit.setTextBackgroundColor(QtGui.QColor(color))  
 self.TextEdit.setFocus()  
  
 def open\_file(self) -> None:  
 file\_path, \_ = QtWidgets.QFileDialog.getOpenFileName(parent=self,  
 caption=u'Открыть файл',  
 directory='E:/',  
 filter='HTML documents (\*.html);')  
 if file\_path:  
 try:  
 with codecs.open(filename=file\_path, mode="r", encoding="utf-8", errors="ignore") as file:  
 data = file.read()  
 except Exception as e:  
 self.show\_msg("Не удалось открыть файл:\n" + str(e))  
 else:  
 self.selected\_file = file\_path  
 self.TextEdit.setText(data)  
 self.set\_win\_title()  
  
 def save\_file(self):  
 if self.selected\_file is None:  
 return self.save\_as\_file()  
 text = self.TextEdit.toHtml() if splitext(self.selected\_file)[  
 1] in self.html\_extensions else self.TextEdit.toPlainText()  
 try:  
 with open(file=self.selected\_file, mode='w', encoding="utf-8") as file:  
 file.write(text)  
 except Exception as e:  
 self.show\_msg("Не удалось сохранить файл:\n" + str(e))  
  
 def save\_as\_file(self):  
 file\_path, \_ = QtWidgets.QFileDialog.getSaveFileName(self, "Save file", "",  
 "HTML documents (\*.html)")  
 if not file\_path:  
 return  
 text = self.TextEdit.toHtml() if splitext(file\_path)[  
 1] in self.html\_extensions else self.TextEdit.toPlainText()  
  
 try:  
 with open(file=file\_path, mode='w', encoding="utf-8") as file:  
 file.write(text)  
 except Exception as e:  
 self.show\_msg("Не удалось сохранить файл:\n" + str(e))  
 else:  
 self.selected\_file = file\_path  
 self.set\_win\_title()  
  
  
def main():  
 app = QtWidgets.QApplication(sys.argv)  
 win = TextEditor()  
 win.show()  
 sys.exit(app.exec\_())  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

# Скриншоты

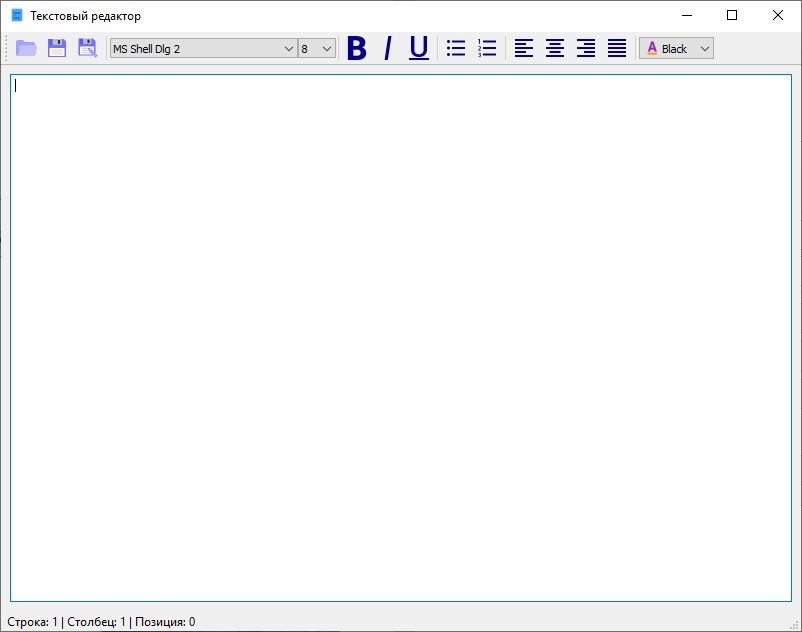


Рис.1. – Чистая форма редактора текста с основным интерфейсом

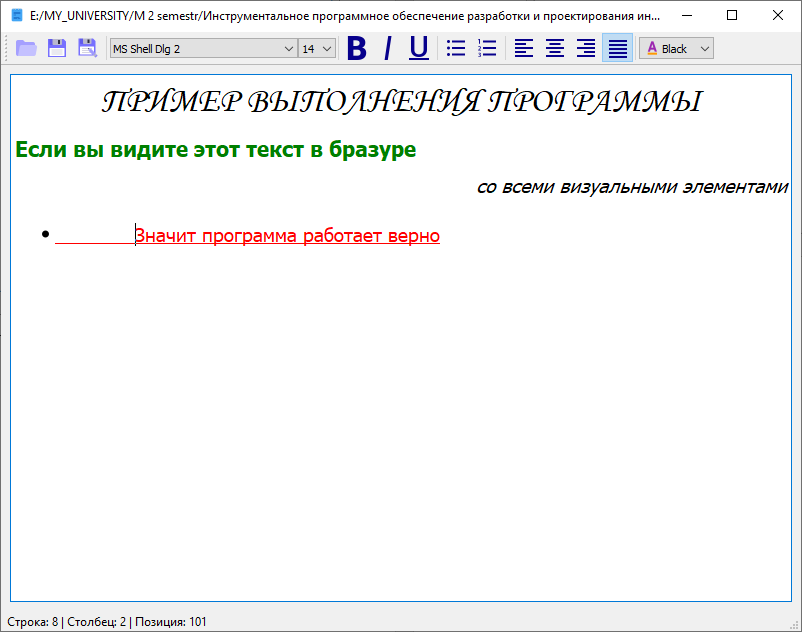


Рис.2. – Пример заполнения формы со стилистическими улучшениями

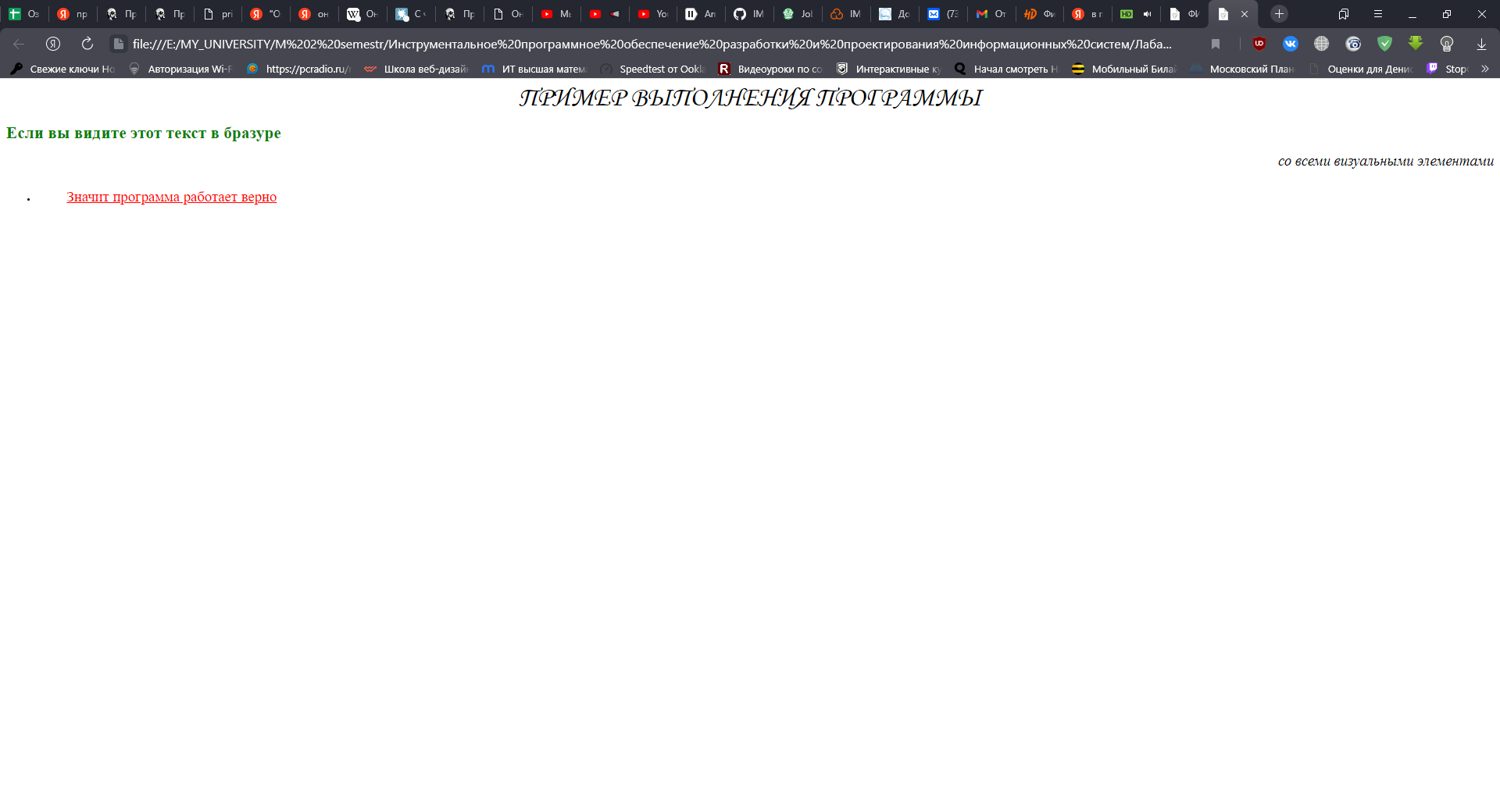


Рис.3. – Результат в html файле, отображённом в браузере