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

Институт информационных технологий (ИИТ)Кафедра инструментального и прикладного программного обеспечения (ИиППО)

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №5**

**по дисциплине**

«Инструментальное ПО разработки и проектирования ИС»

**на тему:**

«Создание калькулятора на Python»

Выполнил студент группы ИКМО-01-20 Попов К.С.

Фамилия И.О.

Принял(а) Куликов А.А.

Фамилия И.О.

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

(подпись студента)

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

(подпись руководителя)

Москва, 2021 г.

Оглавление

[Цель практической работы 3](#_Toc69901641)

[Ход выполнения работы 3](#_Toc69901642)

[Пример работы программы 8](#_Toc69901643)

[Листинг программы 9](#_Toc69901644)

[Модуль main: 9](#_Toc69901645)

[Модуль ГПИ calcUIRU: 12](#_Toc69901646)

[Заключение 17](#_Toc69901647)

# Цель практической работы

В данной практической работе требуется создать калькулятор на Python, который будет принимать выражение, введенное пользователем через графический интерфейс и выдавать его результат:

* Форма пользовательского интерфейса должна иметь следующие компоненты:
  + Цифровые кнопки от 0 до 9
  + Следующие арифметические операции:
    - Сложение
    - Вычитание
    - Умножение
    - Деление
    - Возведение в степень
  + Выделения приоритета операций при помощи скобок
  + Кнопки управления:
    - Очистка выражения
    - Удаление последнего символа выражения
    - Расчет введенного выражения
    - Выход из программы
  + Вывод результата выражения на форму
* Для реализации алгоритма разбора, введенного пользователем выражения, необходимо использовать алоритм обратной польской записи.

# Ход выполнения работы

Для разработки калькулятора была создана форма, представленная на рисунке 1.

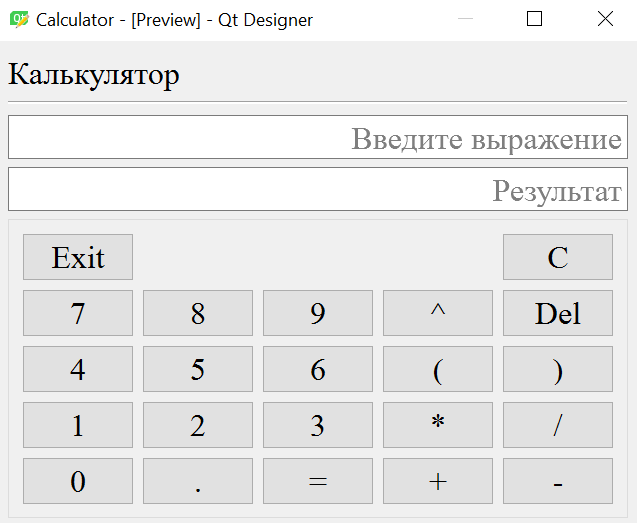


Рисунок 1 – Форма калькулятора в QtDesigner

Каждая кнопка на форме при помощи сигнала привязана к слоту лямбда-функции, осуществляющая действие, предназначенное для данной кнопки. Так, для ввода выражения в поле данных используются кнопки с цифрами и арифметическими операциями.

self.ZeroPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("0"))  
self.DotPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("."))  
self.EqualPushButton.clicked.connect(lambda: self.calculate(self.ExpressionLineEdit.text()))  
self.PlusPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("+"))  
self.MinusPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("-"))  
self.OnePushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("1"))  
self.TwoPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("2"))  
self.ThreePushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("3"))  
self.MultPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("\*"))  
self.DivPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("/"))  
self.FourPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("4"))  
self.FivePushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("5"))  
self.SixPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("6"))  
self.LeftBracketPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("("))  
self.RightBracketPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert(")"))  
self.SevenPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("7"))  
self.EightPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("8"))  
self.NinePushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("9"))  
self.ExpPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("^"))  
self.DelPushButton.clicked.connect(lambda: self.ExpressionLineEdit.backspace())  
self.ClearPushButton.clicked.connect(lambda: self.ExpressionLineEdit.clear())  
self.ExitPushButton.clicked.connect(self.close)

Также для удобства ввода выражения, к каждой кнопки привязана к клавише (сочетанию клавиш) на клавиатуре.

self.ZeroPushButton.setShortcut("0")  
self.DotPushButton.setShortcut(".")  
self.EqualPushButton.setShortcut("=")  
self.PlusPushButton.setShortcut("+")  
self.MinusPushButton.setShortcut("-")  
self.OnePushButton.setShortcut("1")  
self.TwoPushButton.setShortcut("2")  
self.ThreePushButton.setShortcut("3")  
self.MultPushButton.setShortcut("\*")  
self.DivPushButton.setShortcut("/")  
self.FourPushButton.setShortcut("4")  
self.FivePushButton.setShortcut("5")  
self.SixPushButton.setShortcut("6")  
self.LeftBracketPushButton.setShortcut("(")  
self.RightBracketPushButton.setShortcut(")")  
self.SevenPushButton.setShortcut("7")  
self.EightPushButton.setShortcut("8")  
self.NinePushButton.setShortcut("9")  
self.ExpPushButton.setShortcut("^")  
self.DelPushButton.setShortcut("Backspace")  
self.ClearPushButton.setShortcut("CTRL+C")  
self.ExitPushButton.setShortcut("ESC")

Для построения алгоритма обратной польской записи необходимо добавить следующие структуры:

* Словарь с операциями и их приоритетами
* Словарь с функциями для выполнения операций
* Стек для заполнения числами
* Стек для заполнения операциями

Данные структуры представлены в коде ниже

self.stack\_of\_numbers = []  
self.stack\_of\_operations = []  
self.operations\_priority = {"+": 1, "-": 1, "\*": 2, "/": 2, "^": 3, "(": 0, ")": 0}  
self.operations = {"+": self.sum, "-": self.sub, "\*": self.mul, "/": self.div, "^": self.exp}

Функции выполнения операций представлены ниже

def sum(self) -> None:  
 b = self.stack\_of\_numbers.pop()  
 a = self.stack\_of\_numbers.pop()  
 self.stack\_of\_numbers.append(float(a) + float(b))  
 if self.get\_top\_of\_the\_stack(self.stack\_of\_operations) == "+":  
 self.stack\_of\_operations.pop()  
  
def sub(self) -> None:  
 b = self.stack\_of\_numbers.pop()  
 a = self.stack\_of\_numbers.pop()  
 self.stack\_of\_numbers.append(float(a) - float(b))  
 if self.get\_top\_of\_the\_stack(self.stack\_of\_operations) == "-":  
 self.stack\_of\_operations.pop()  
  
def mul(self) -> None:  
 b = self.stack\_of\_numbers.pop()  
 a = self.stack\_of\_numbers.pop()  
 self.stack\_of\_numbers.append(float(a) \* float(b))  
 if self.get\_top\_of\_the\_stack(self.stack\_of\_operations) == "\*":  
 self.stack\_of\_operations.pop()  
  
def div(self) -> None:  
 b = self.stack\_of\_numbers.pop()  
 a = self.stack\_of\_numbers.pop()  
 self.stack\_of\_numbers.append(float(a) / float(b))  
 if self.get\_top\_of\_the\_stack(self.stack\_of\_operations) == "/":  
 self.stack\_of\_operations.pop()  
  
def exp(self) -> None:  
 b = self.stack\_of\_numbers.pop()  
 a = self.stack\_of\_numbers.pop()  
 self.stack\_of\_numbers.append(float(a) \*\* float(b))  
 if self.get\_top\_of\_the\_stack(self.stack\_of\_operations) == "^":  
 self.stack\_of\_operations.pop()

Для получения элемента на вершине выбранного стека необходима следующая статическая функция

@staticmethod  
def get\_top\_of\_the\_stack(stack: list) -> str:  
 return stack[len(stack) - 1]

Перед выполнением алгоритма, необходимо разобрать введенное выражение на токены и отфильтровать его, для этого используется статическая функция, представленная ниже

@staticmethod  
def get\_tokens(expression: str) -> list:  
 raw\_list = re.split(r"(\D)", expression)  
 tokens\_list = []  
 for i in raw\_list:  
 if i != " " and i != "":  
 if i == ".":  
 tokens\_list[len(tokens\_list) - 1] += i  
 tokens\_list.append(i)  
 elif len(tokens\_list) > 0 and tokens\_list[len(tokens\_list) - 1] == ".":  
 tokens\_list.pop()  
 tokens\_list[len(tokens\_list) - 1] += i  
 else:  
 tokens\_list.append(i)  
 return tokens\_list

Алгоритм обратной польской записи разбирает выражение на два стека: числа кладутся в стек для чисел, операции в стек для операций. И на основе приоритетов считанного токена операции и токена на вершине стека, строится логика выполнения алгоритма, представленная функцией ниже:

def calculate(self, expression: str) -> None:  
 self.stack\_of\_numbers.clear()  
 self.stack\_of\_operations.clear()  
 tokens = self.get\_tokens(expression)  
 try:  
 for token in tokens:  
 if len(self.stack\_of\_numbers) == 0 and token in self.operations\_priority and \  
 (token != '(' and token != ')'):  
 self.stack\_of\_numbers.append(float(0))  
 if tokens.index(token) == len(tokens) - 1 and token in self.operations\_priority and \  
 (token != '(' and token != ')'):  
 self.stack\_of\_numbers.append(float(0))  
 if token not in self.operations\_priority:  
 self.stack\_of\_numbers.append(float(token))  
 elif len(self.stack\_of\_operations) > 0 and (token != '(' and token != ')'):  
 while (len(self.stack\_of\_operations) != 0 and self.operations\_priority[token] <=  
 self.operations\_priority[self.get\_top\_of\_the\_stack(self.stack\_of\_operations)]):  
 self.operations[self.get\_top\_of\_the\_stack(self.stack\_of\_operations)]()  
 self.stack\_of\_operations.append(token)  
 elif token == ')':  
 self.operations[self.get\_top\_of\_the\_stack(self.stack\_of\_operations)]()  
 self.stack\_of\_operations.reverse()  
 for i in self.stack\_of\_operations:  
 if i == '(':  
 self.stack\_of\_operations.remove(i)  
 break  
 self.stack\_of\_operations.reverse()  
 else:  
 self.stack\_of\_operations.append(token)  
 while len(self.stack\_of\_operations) != 0:

self.operations[self.get\_top\_of\_the\_stack(self.stack\_of\_operations)]()  
 except (KeyError, IndexError, ValueError, ZeroDivisionError

):  
 self.msg\_box.setText("Ошибка выполнения выражения")  
 self.msg\_box.exec()  
 else:  
 if len(self.stack\_of\_numbers) != 0:

self.ResultLineEdit.setText(str(self.stack\_of\_numbers[0]))

# Пример работы программы

Примеры работы программы представлены на рисунках 2 и 3.

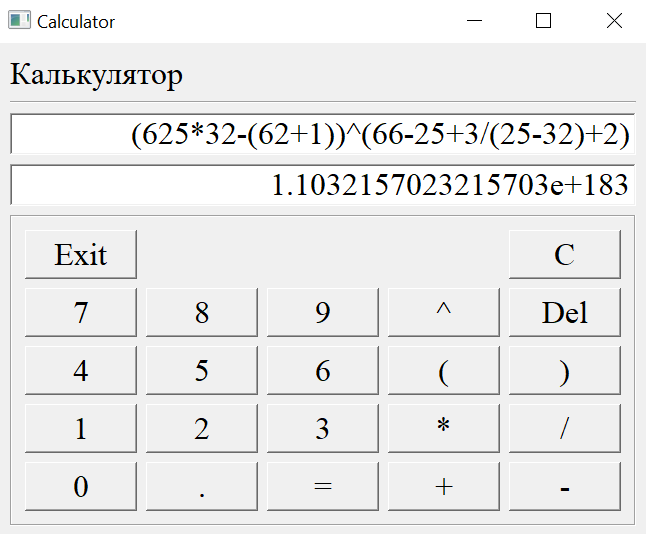


Рисунок 2 – Пример 1

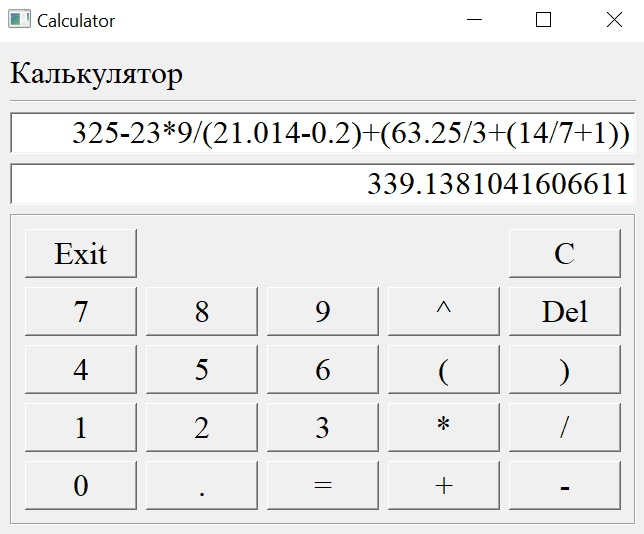


Рисунок 3 -Пример 2

# Листинг программы

### Модуль main:

from PyQt5 import QtWidgets  
import sys  
import calcUIRU  
import re  
  
  
class Calc(QtWidgets.QMainWindow, calcUIRU.Ui\_Calculator):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.setupUi(self)  
 self.setFocus()  
 self.msg\_box = QtWidgets.QMessageBox()  
 self.msg\_box.setWindowTitle(self.windowTitle())  
 self.stack\_of\_numbers = []  
 self.stack\_of\_operations = []  
 self.operations\_priority = {"+": 1, "-": 1, "\*": 2, "/": 2, "^": 3, "(": 0, ")": 0}  
 self.operations = {"+": self.sum, "-": self.sub, "\*": self.mul, "/": self.div, "^": self.exp}  
 self.ZeroPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("0"))  
 self.DotPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("."))  
 self.EqualPushButton.clicked.connect(lambda: self.calculate(self.ExpressionLineEdit.text()))  
 self.PlusPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("+"))  
 self.MinusPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("-"))  
 self.OnePushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("1"))  
 self.TwoPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("2"))  
 self.ThreePushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("3"))  
 self.MultPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("\*"))  
 self.DivPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("/"))  
 self.FourPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("4"))  
 self.FivePushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("5"))  
 self.SixPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("6"))  
 self.LeftBracketPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("("))  
 self.RightBracketPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert(")"))  
 self.SevenPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("7"))  
 self.EightPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("8"))  
 self.NinePushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("9"))  
 self.ExpPushButton.clicked.connect(lambda: self.ExpressionLineEdit.insert("^"))  
 self.DelPushButton.clicked.connect(lambda: self.ExpressionLineEdit.backspace())  
 self.ClearPushButton.clicked.connect(lambda: self.ExpressionLineEdit.clear())  
 self.ExitPushButton.clicked.connect(self.close)  
 self.ZeroPushButton.setShortcut("0")  
 self.DotPushButton.setShortcut(".")  
 self.EqualPushButton.setShortcut("=")  
 self.PlusPushButton.setShortcut("+")  
 self.MinusPushButton.setShortcut("-")  
 self.OnePushButton.setShortcut("1")  
 self.TwoPushButton.setShortcut("2")  
 self.ThreePushButton.setShortcut("3")  
 self.MultPushButton.setShortcut("\*")  
 self.DivPushButton.setShortcut("/")  
 self.FourPushButton.setShortcut("4")  
 self.FivePushButton.setShortcut("5")  
 self.SixPushButton.setShortcut("6")  
 self.LeftBracketPushButton.setShortcut("(")  
 self.RightBracketPushButton.setShortcut(")")  
 self.SevenPushButton.setShortcut("7")  
 self.EightPushButton.setShortcut("8")  
 self.NinePushButton.setShortcut("9")  
 self.ExpPushButton.setShortcut("^")  
 self.DelPushButton.setShortcut("Backspace")  
 self.ClearPushButton.setShortcut("CTRL+C")  
 self.ExitPushButton.setShortcut("ESC")  
  
 def sum(self) -> None:  
 b = self.stack\_of\_numbers.pop()  
 a = self.stack\_of\_numbers.pop()  
 self.stack\_of\_numbers.append(float(a) + float(b))  
 if self.get\_top\_of\_the\_stack(self.stack\_of\_operations) == "+":  
 self.stack\_of\_operations.pop()  
  
 def sub(self) -> None:  
 b = self.stack\_of\_numbers.pop()  
 a = self.stack\_of\_numbers.pop()  
 self.stack\_of\_numbers.append(float(a) - float(b))  
 if self.get\_top\_of\_the\_stack(self.stack\_of\_operations) == "-":  
 self.stack\_of\_operations.pop()  
  
 def mul(self) -> None:  
 b = self.stack\_of\_numbers.pop()  
 a = self.stack\_of\_numbers.pop()  
 self.stack\_of\_numbers.append(float(a) \* float(b))  
 if self.get\_top\_of\_the\_stack(self.stack\_of\_operations) == "\*":  
 self.stack\_of\_operations.pop()  
  
 def div(self) -> None:  
 b = self.stack\_of\_numbers.pop()  
 a = self.stack\_of\_numbers.pop()  
 self.stack\_of\_numbers.append(float(a) / float(b))  
 if self.get\_top\_of\_the\_stack(self.stack\_of\_operations) == "/":  
 self.stack\_of\_operations.pop()  
  
 def exp(self) -> None:  
 b = self.stack\_of\_numbers.pop()  
 a = self.stack\_of\_numbers.pop()  
 self.stack\_of\_numbers.append(float(a) \*\* float(b))  
 if self.get\_top\_of\_the\_stack(self.stack\_of\_operations) == "^":  
 self.stack\_of\_operations.pop()  
  
 @staticmethod  
 def get\_top\_of\_the\_stack(stack: list) -> str:  
 return stack[len(stack) - 1]  
  
 @staticmethod  
 def get\_tokens(expression: str) -> list:  
 raw\_list = re.split(r"(\D)", expression)  
 tokens\_list = []  
 for i in raw\_list:  
 if i != " " and i != "":  
 if i == ".":  
 tokens\_list[len(tokens\_list) - 1] += i  
 tokens\_list.append(i)  
 elif len(tokens\_list) > 0 and tokens\_list[len(tokens\_list) - 1] == ".":  
 tokens\_list.pop()  
 tokens\_list[len(tokens\_list) - 1] += i  
 else:  
 tokens\_list.append(i)  
 return tokens\_list

def calculate(self, expression: str) -> None:  
 self.stack\_of\_numbers.clear()  
 self.stack\_of\_operations.clear()  
 tokens = self.get\_tokens(expression)  
 try:  
 for token in tokens:  
 if len(self.stack\_of\_numbers) == 0 and token in self.operations\_priority and \  
 (token != '(' and token != ')'):  
 self.stack\_of\_numbers.append(float(0))  
 if tokens.index(token) == len(tokens) - 1 and token in self.operations\_priority and \  
 (token != '(' and token != ')'):  
 self.stack\_of\_numbers.append(float(0))  
 if token not in self.operations\_priority:  
 self.stack\_of\_numbers.append(float(token))  
 elif len(self.stack\_of\_operations) > 0 and (token != '(' and token != ')'):  
 while (len(self.stack\_of\_operations) != 0 and self.operations\_priority[token] <=  
 self.operations\_priority[self.get\_top\_of\_the\_stack(self.stack\_of\_operations)]):  
 self.operations[self.get\_top\_of\_the\_stack(self.stack\_of\_operations)]()  
 self.stack\_of\_operations.append(token)  
 elif token == ')':  
 self.operations[self.get\_top\_of\_the\_stack(self.stack\_of\_operations)]()  
 self.stack\_of\_operations.reverse()  
 for i in self.stack\_of\_operations:  
 if i == '(':  
 self.stack\_of\_operations.remove(i)  
 break  
 self.stack\_of\_operations.reverse()  
 else:  
 self.stack\_of\_operations.append(token)  
 while len(self.stack\_of\_operations) != 0:

self.operations[self.get\_top\_of\_the\_stack(self.stack\_of\_operations)]()  
 except (KeyError, IndexError, ValueError, ZeroDivisionError

):  
 self.msg\_box.setText("Ошибка выполнения выражения")  
 self.msg\_box.exec()  
 else:  
 if len(self.stack\_of\_numbers) != 0:  
 self.ResultLineEdit.setText(str(self.stack\_of\_numbers[0]))

def main() -> None:  
 app = QtWidgets.QApplication(sys.argv)  
 window = Calc()  
 window.show()  
 sys.exit(app.exec\_())  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

### Модуль ГПИ calcUIRU:

# -\*- coding: utf-8 -\*-  
  
# Form implementation generated from reading ui file 'calcUIRU.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\_Calculator(object):  
 def setupUi(self, Calculator):  
 Calculator.setObjectName("Calculator")  
 Calculator.resize(472, 316)  
 self.centralwidget = QtWidgets.QWidget(Calculator)  
 self.centralwidget.setObjectName("centralwidget")  
 self.verticalLayout\_5 = QtWidgets.QVBoxLayout(self.centralwidget)  
 self.verticalLayout\_5.setObjectName("verticalLayout\_5")  
 self.verticalLayout\_4 = QtWidgets.QVBoxLayout()  
 self.verticalLayout\_4.setObjectName("verticalLayout\_4")  
 self.verticalLayout\_3 = QtWidgets.QVBoxLayout()  
 self.verticalLayout\_3.setObjectName("verticalLayout\_3")  
 self.horizontalLayout = QtWidgets.QHBoxLayout()  
 self.horizontalLayout.setObjectName("horizontalLayout")  
 self.TitleLabel = QtWidgets.QLabel(self.centralwidget)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.TitleLabel.setFont(font)  
 self.TitleLabel.setObjectName("TitleLabel")  
 self.horizontalLayout.addWidget(self.TitleLabel)  
 spacerItem = QtWidgets.QSpacerItem(138, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)  
 self.horizontalLayout.addItem(spacerItem)  
 self.verticalLayout\_3.addLayout(self.horizontalLayout)  
 self.line = QtWidgets.QFrame(self.centralwidget)  
 self.line.setFrameShape(QtWidgets.QFrame.HLine)  
 self.line.setFrameShadow(QtWidgets.QFrame.Sunken)  
 self.line.setObjectName("line")  
 self.verticalLayout\_3.addWidget(self.line)  
 self.verticalLayout\_4.addLayout(self.verticalLayout\_3)  
 self.verticalLayout\_2 = QtWidgets.QVBoxLayout()  
 self.verticalLayout\_2.setObjectName("verticalLayout\_2")  
 self.ExpressionLineEdit = QtWidgets.QLineEdit(self.centralwidget)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.ExpressionLineEdit.setFont(font)  
 self.ExpressionLineEdit.setText("")  
 self.ExpressionLineEdit.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)  
 self.ExpressionLineEdit.setReadOnly(True)  
 self.ExpressionLineEdit.setObjectName("ExpressionLineEdit")  
 self.verticalLayout\_2.addWidget(self.ExpressionLineEdit)  
 self.ResultLineEdit = QtWidgets.QLineEdit(self.centralwidget)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.ResultLineEdit.setFont(font)  
 self.ResultLineEdit.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)  
 self.ResultLineEdit.setReadOnly(True)  
 self.ResultLineEdit.setObjectName("ResultLineEdit")  
 self.verticalLayout\_2.addWidget(self.ResultLineEdit)  
 self.ButtonsGroupBox = QtWidgets.QGroupBox(self.centralwidget)  
 self.ButtonsGroupBox.setTitle("")  
 self.ButtonsGroupBox.setObjectName("ButtonsGroupBox")  
 self.verticalLayout = QtWidgets.QVBoxLayout(self.ButtonsGroupBox)  
 self.verticalLayout.setObjectName("verticalLayout")  
 self.ButtonsGridLayout = QtWidgets.QGridLayout()  
 self.ButtonsGridLayout.setObjectName("ButtonsGridLayout")  
 self.ExpPushButton = QtWidgets.QPushButton(self.ButtonsGroupBox)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.ExpPushButton.setFont(font)  
 self.ExpPushButton.setObjectName("ExpPushButton")  
 self.ButtonsGridLayout.addWidget(self.ExpPushButton, 1, 3, 1, 1)  
 self.DelPushButton = QtWidgets.QPushButton(self.ButtonsGroupBox)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.DelPushButton.setFont(font)  
 self.DelPushButton.setObjectName("DelPushButton")  
 self.ButtonsGridLayout.addWidget(self.DelPushButton, 1, 4, 1, 1)  
 self.NinePushButton = QtWidgets.QPushButton(self.ButtonsGroupBox)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.NinePushButton.setFont(font)  
 self.NinePushButton.setObjectName("NinePushButton")  
 self.ButtonsGridLayout.addWidget(self.NinePushButton, 1, 2, 1, 1)  
 self.MultPushButton = QtWidgets.QPushButton(self.ButtonsGroupBox)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.MultPushButton.setFont(font)  
 self.MultPushButton.setObjectName("MultPushButton")  
 self.ButtonsGridLayout.addWidget(self.MultPushButton, 3, 3, 1, 1)  
 self.ZeroPushButton = QtWidgets.QPushButton(self.ButtonsGroupBox)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.ZeroPushButton.setFont(font)  
 self.ZeroPushButton.setObjectName("ZeroPushButton")  
 self.ButtonsGridLayout.addWidget(self.ZeroPushButton, 4, 0, 1, 1)  
 self.RightBracketPushButton = QtWidgets.QPushButton(self.ButtonsGroupBox)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.RightBracketPushButton.setFont(font)  
 self.RightBracketPushButton.setObjectName("RightBracketPushButton")  
 self.ButtonsGridLayout.addWidget(self.RightBracketPushButton, 2, 4, 1, 1)  
 self.DivPushButton = QtWidgets.QPushButton(self.ButtonsGroupBox)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.DivPushButton.setFont(font)  
 self.DivPushButton.setObjectName("DivPushButton")  
 self.ButtonsGridLayout.addWidget(self.DivPushButton, 3, 4, 1, 1)  
 self.FourPushButton = QtWidgets.QPushButton(self.ButtonsGroupBox)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.FourPushButton.setFont(font)  
 self.FourPushButton.setObjectName("FourPushButton")  
 self.ButtonsGridLayout.addWidget(self.FourPushButton, 2, 0, 1, 1)  
 self.ThreePushButton = QtWidgets.QPushButton(self.ButtonsGroupBox)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.ThreePushButton.setFont(font)  
 self.ThreePushButton.setObjectName("ThreePushButton")  
 self.ButtonsGridLayout.addWidget(self.ThreePushButton, 3, 2, 1, 1)  
 self.MinusPushButton = QtWidgets.QPushButton(self.ButtonsGroupBox)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.MinusPushButton.setFont(font)  
 self.MinusPushButton.setObjectName("MinusPushButton")  
 self.ButtonsGridLayout.addWidget(self.MinusPushButton, 4, 4, 1, 1)  
 self.OnePushButton = QtWidgets.QPushButton(self.ButtonsGroupBox)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.OnePushButton.setFont(font)  
 self.OnePushButton.setObjectName("OnePushButton")  
 self.ButtonsGridLayout.addWidget(self.OnePushButton, 3, 0, 1, 1)  
 self.TwoPushButton = QtWidgets.QPushButton(self.ButtonsGroupBox)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.TwoPushButton.setFont(font)  
 self.TwoPushButton.setObjectName("TwoPushButton")  
 self.ButtonsGridLayout.addWidget(self.TwoPushButton, 3, 1, 1, 1)  
 self.EightPushButton = QtWidgets.QPushButton(self.ButtonsGroupBox)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.EightPushButton.setFont(font)  
 self.EightPushButton.setObjectName("EightPushButton")  
 self.ButtonsGridLayout.addWidget(self.EightPushButton, 1, 1, 1, 1)  
 self.SevenPushButton = QtWidgets.QPushButton(self.ButtonsGroupBox)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.SevenPushButton.setFont(font)  
 self.SevenPushButton.setObjectName("SevenPushButton")  
 self.ButtonsGridLayout.addWidget(self.SevenPushButton, 1, 0, 1, 1)  
 self.DotPushButton = QtWidgets.QPushButton(self.ButtonsGroupBox)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.DotPushButton.setFont(font)  
 self.DotPushButton.setObjectName("DotPushButton")  
 self.ButtonsGridLayout.addWidget(self.DotPushButton, 4, 1, 1, 1)  
 self.PlusPushButton = QtWidgets.QPushButton(self.ButtonsGroupBox)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.PlusPushButton.setFont(font)  
 self.PlusPushButton.setObjectName("PlusPushButton")  
 self.ButtonsGridLayout.addWidget(self.PlusPushButton, 4, 3, 1, 1)  
 self.EqualPushButton = QtWidgets.QPushButton(self.ButtonsGroupBox)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.EqualPushButton.setFont(font)  
 self.EqualPushButton.setObjectName("EqualPushButton")  
 self.ButtonsGridLayout.addWidget(self.EqualPushButton, 4, 2, 1, 1)  
 self.FivePushButton = QtWidgets.QPushButton(self.ButtonsGroupBox)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.FivePushButton.setFont(font)  
 self.FivePushButton.setObjectName("FivePushButton")  
 self.ButtonsGridLayout.addWidget(self.FivePushButton, 2, 1, 1, 1)  
 self.SixPushButton = QtWidgets.QPushButton(self.ButtonsGroupBox)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.SixPushButton.setFont(font)  
 self.SixPushButton.setObjectName("SixPushButton")  
 self.ButtonsGridLayout.addWidget(self.SixPushButton, 2, 2, 1, 1)  
 self.LeftBracketPushButton = QtWidgets.QPushButton(self.ButtonsGroupBox)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.LeftBracketPushButton.setFont(font)  
 self.LeftBracketPushButton.setObjectName("LeftBracketPushButton")  
 self.ButtonsGridLayout.addWidget(self.LeftBracketPushButton, 2, 3, 1, 1)  
 self.ExitPushButton = QtWidgets.QPushButton(self.ButtonsGroupBox)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.ExitPushButton.setFont(font)  
 self.ExitPushButton.setObjectName("ExitPushButton")  
 self.ButtonsGridLayout.addWidget(self.ExitPushButton, 0, 0, 1, 1)  
 self.ClearPushButton = QtWidgets.QPushButton(self.ButtonsGroupBox)  
 font = QtGui.QFont()  
 font.setFamily("Times New Roman")  
 font.setPointSize(16)  
 self.ClearPushButton.setFont(font)  
 self.ClearPushButton.setObjectName("ClearPushButton")  
 self.ButtonsGridLayout.addWidget(self.ClearPushButton, 0, 4, 1, 1)  
 self.verticalLayout.addLayout(self.ButtonsGridLayout)  
 self.verticalLayout\_2.addWidget(self.ButtonsGroupBox)  
 self.verticalLayout\_4.addLayout(self.verticalLayout\_2)  
 self.verticalLayout\_5.addLayout(self.verticalLayout\_4)  
 Calculator.setCentralWidget(self.centralwidget)  
  
 self.retranslateUi(Calculator)  
 QtCore.QMetaObject.connectSlotsByName(Calculator)  
  
 def retranslateUi(self, Calculator):  
 \_translate = QtCore.QCoreApplication.translate  
 Calculator.setWindowTitle(\_translate("Calculator", "Calculator"))  
 self.TitleLabel.setText(\_translate("Calculator", "Калькулятор"))  
 self.ExpressionLineEdit.setPlaceholderText(\_translate("Calculator", "Введите выражение"))  
 self.ResultLineEdit.setPlaceholderText(\_translate("Calculator", "Результат"))  
 self.ExpPushButton.setText(\_translate("Calculator", "^"))  
 self.DelPushButton.setText(\_translate("Calculator", "Del"))  
 self.NinePushButton.setText(\_translate("Calculator", "9"))  
 self.MultPushButton.setText(\_translate("Calculator", "\*"))  
 self.ZeroPushButton.setText(\_translate("Calculator", "0"))  
 self.RightBracketPushButton.setText(\_translate("Calculator", ")"))  
 self.DivPushButton.setText(\_translate("Calculator", "/"))  
 self.FourPushButton.setText(\_translate("Calculator", "4"))  
 self.ThreePushButton.setText(\_translate("Calculator", "3"))  
 self.MinusPushButton.setText(\_translate("Calculator", "-"))  
 self.OnePushButton.setText(\_translate("Calculator", "1"))  
 self.TwoPushButton.setText(\_translate("Calculator", "2"))  
 self.EightPushButton.setText(\_translate("Calculator", "8"))  
 self.SevenPushButton.setText(\_translate("Calculator", "7"))  
 self.DotPushButton.setText(\_translate("Calculator", "."))  
 self.PlusPushButton.setText(\_translate("Calculator", "+"))  
 self.EqualPushButton.setText(\_translate("Calculator", "="))  
 self.FivePushButton.setText(\_translate("Calculator", "5"))  
 self.SixPushButton.setText(\_translate("Calculator", "6"))  
 self.LeftBracketPushButton.setText(\_translate("Calculator", "("))  
 self.ExitPushButton.setText(\_translate("Calculator", "Exit"))  
 self.ClearPushButton.setText(\_translate("Calculator", "C"))  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 import sys  
 app = QtWidgets.QApplication(sys.argv)  
 Calculator = QtWidgets.QMainWindow()  
 ui = Ui\_Calculator()  
 ui.setupUi(Calculator)  
 Calculator.show()  
 sys.exit(app.exec\_())

# Заключение

В ходе практической работы были закреплены знания языка Python, а также был разработан калькулятор с использованием алгоритма обратной польской записи.