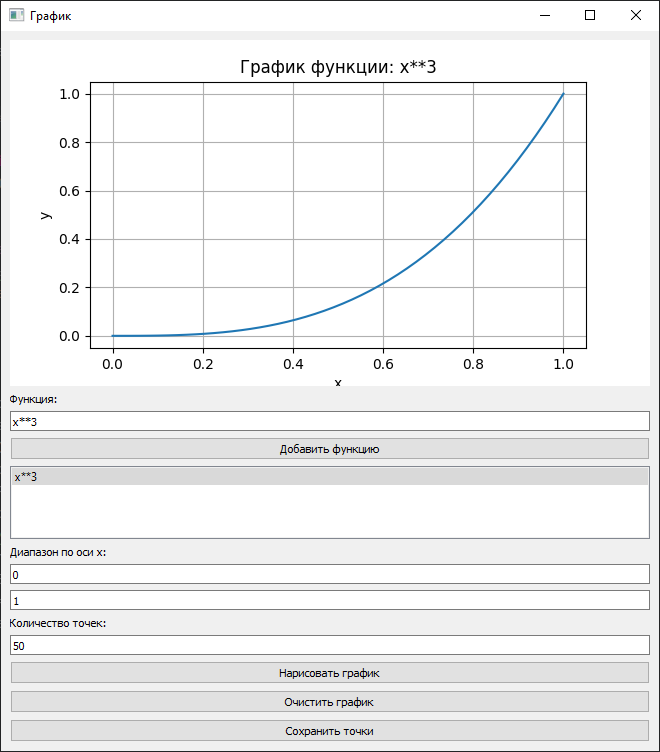
Министерство науки и высшего образования Российской Федерации  
 Федеральное государственное автономное образовательное учреждение  
 высшего образования  
 **Пермский национальный исследовательский  
 политехнический университет**  
 Факультет прикладной математики и механики  
 Кафедра «Вычислительная математика, механика и биомеханика»  
 направление подготовки: 15.03.03 Прикладная механика  
 профиль: «Вычислительная механика и компьютерный инжиниринг»

**Отчёт по практическому заданию,  
 «Графический интерфейс»  
 по дисциплине «Информатика»**

Выполнил студент гр. ПМ-24-2б  
Поплаухин Пётр Михайлович

Проверил ст. пр. каф. ВММБ,   
Ильиных Глеб Валентинович

from PyQt5.QtWidgets import (  
 QApplication, QLabel, QLineEdit, QMainWindow, QPushButton,  
 QVBoxLayout, QWidget, QMessageBox, QListWidget, QListWidgetItem,  
 QFileDialog  
)  
import numpy as np  
import matplotlib.pyplot as plt  
from matplotlib.backends.backend\_qt5agg import FigureCanvasQTAgg as FigureCanvas  
  
  
class MainWindow(QMainWindow):  
 def \_\_init\_\_(self, parent=None):  
 super(MainWindow, self).\_\_init\_\_(parent)  
 self.setWindowTitle("График")  
  
 self.figure = plt.figure()  
 self.canvas = FigureCanvas(self.figure)  
 self.functions = {} # Словарь для хранения функций {название: функция}  
 self.selected\_function = None # Выбранная функция  
 self.x = None  
 self.y = None  
  
 # Виджеты  
 self.plot\_button = QPushButton("Нарисовать график")  
 self.plot\_clear\_button = QPushButton("Очистить график")  
 self.function\_label = QLabel("Функция:")  
 self.function\_input = QLineEdit('x\*\*3') # Поле ввода названия функции  
 self.add\_function\_button = QPushButton("Добавить функцию")  
 self.function\_list = QListWidget()  
 self.range\_label = QLabel("Диапазон по оси x:")  
 self.range\_start\_input = QLineEdit('0')  
 self.range\_end\_input = QLineEdit('1')  
 self.plot\_point\_count\_label = QLabel("Количество точек:")  
 self.num\_points\_input = QLineEdit('50')  
 self.save\_plot\_points\_button = QPushButton("Сохранить точки")  
  
  
 # Макет  
 central\_widget = QWidget()  
 layout = QVBoxLayout()  
 central\_widget.setLayout(layout)  
  
 # Добавление виджетов на макет  
 layout.addWidget(self.canvas)  
 layout.addWidget(self.function\_label)  
 layout.addWidget(self.function\_input)  
 layout.addWidget(self.add\_function\_button)  
 layout.addWidget(self.function\_list)  
 layout.addWidget(self.range\_label)  
 layout.addWidget(self.range\_start\_input)  
 layout.addWidget(self.range\_end\_input)  
 layout.addWidget(self.plot\_point\_count\_label)  
 layout.addWidget(self.num\_points\_input)  
 layout.addWidget(self.plot\_button)  
 layout.addWidget(self.plot\_clear\_button)  
 layout.addWidget(self.save\_plot\_points\_button)  
  
  
 self.setCentralWidget(central\_widget)  
  
 #  
 self.plot\_button.clicked.connect(self.plot\_data)  
 self.plot\_clear\_button.clicked.connect(self.clear\_plot)  
 self.add\_function\_button.clicked.connect(self.add\_function)  
 self.function\_list.itemSelectionChanged.connect(self.update\_selected\_function)  
 self.save\_plot\_points\_button.clicked.connect(self.save\_plot\_points)  
  
 self.add\_function()  
  
  
 def add\_function(self):  
 try:  
 expression = self.function\_input.text()  
 # Создаем лямбда-функцию в безопасном окружении проверить, можно ли вычислить функцию с каким-либо значением,  
 # и выбросить исключение до того, как функция будет добавлена  
 func = eval(f"lambda x: {expression}", {"\_\_builtins\_\_": None}, {"x": None, "np": np, "sin": np.sin, "cos": np.cos, "tan": np.tan, \  
 "exp": np.exp, "log": np.log, "sqrt": np.sqrt, "pi": np.pi}) # add more functions as needed  
  
 # Тестируем функцию, чтобы выявить ошибки  
 test\_value = 0  
 func(test\_value) # Вызываем функцию для проверки  
  
 self.functions[expression] = func  
 item = QListWidgetItem(expression)  
 self.function\_list.addItem(item)  
 self.function\_list.setCurrentItem(item) # Выбираем добавленную функцию  
  
 except (SyntaxError, NameError, TypeError, ValueError, ZeroDivisionError, OverflowError) as e:  
 QMessageBox.warning(self, "Ошибка", f"Функция введена не верно: {e}")  
 except Exception as e: # Ловим другие потенциальные ошибки  
 QMessageBox.warning(self, "Ошибка", f"Функция не может быть вычислена: {e}")  
  
  
 def update\_selected\_function(self):  
 selected\_items = self.function\_list.selectedItems()  
 if selected\_items:  
 self.selected\_function = self.functions[selected\_items[0].text()]  
 else:  
 self.selected\_function = None  
  
 if self.selected\_function is None:  
 QMessageBox.warning(self, "Ошибка", "Выберите функцию")  
 return  
 range\_start = float(self.range\_start\_input.text())  
 range\_end = float(self.range\_end\_input.text())  
 num\_point = int(self.num\_points\_input.text())  
 x = np.linspace(range\_start, range\_end, num\_point)  
 y = self.selected\_function(x)  
 self.x = x  
 self.y = y  
  
  
 def plot\_data(self):  
 self.figure.clear()  
 ax = self.figure.add\_subplot(111)  
 ax.plot(self.x, self.y)  
 ax.grid(True)  
 ax.set\_xlabel('x')  
 ax.set\_ylabel('y')  
 ax.set\_title(f'График функции: {self.function\_input.text()}')  
 self.canvas.draw()  
  
  
 def clear\_plot(self):  
 self.figure.clear()  
 self.canvas.draw()  
  
  
 def save\_plot\_points(self):  
 # Открываем диалоговое окно "Сохранить как..."  
 options = QFileDialog.Options()  
 filename, \_ = QFileDialog.getSaveFileName(self, "Сохранить точки графика в файл", "plot\_data.txt")  
  
 # Если пользователь нажал "Отмена", filename будет пустой строкой  
 if not filename:  
 return  
  
 # Добавляем расширение .txt, если пользователь его не указал  
 if not filename.endswith(".txt"):  
 filename += ".txt"  
  
 with open(filename, "w") as f:  
 for i in range(len(self.x)):  
 f.write(f"{self.x[i]:.6f}\t{self.y[i]:.6f}\n")  
 QMessageBox.information(self, "Успешно", f"Данные графика успешно сохранены в файл {filename}")  
  
  
# Создать приложение QApplication  
app = QApplication([])  
# Создать окно приложения  
main\_window = MainWindow()  
main\_window.show()  
# Запустить приложение  
app.exec\_()



**Цель работы:**

 Разработка графического интерфейса программы для построения графиков функций, задаваемых пользователем.

**Выполненные задания:**

1. Установка необходимых пакетов: *PyQt5, Matplotlib, NumPy*
2. Создан интерфейс с использованием *PyQt5*, позволяющий пользователю вводить функцию, диапазон значений по оси x и количество точек.
3. Реализована функция построения графика заданной функции с использованием *Matplotlib*.
4. Добавлена возможность выбора количества точек для построения графика.
5. Используя *try-except* реализована обработка ошибок неверного ввода функции и диапазона значений.
6. Добавлена кнопка "Очистить график", которая очищает область построения.
7. Реализована функция сохранения точек графика в CSV файл, с возможностью выбора пути сохранения файла. Добавлены проверки на существование данных перед сохранением.
8. Добавлена возможность добавления функций в список и выбора функции для построения из списка.

**Вывод:**

В ходе выполнения практического задания был разработан графический интерфейс программы для построения графиков функций. Все поставленные задачи успешно выполнены. Программа позволяет пользователю удобно вводить функцию, задавать параметры построения и сохранять результаты. Обработка ошибок обеспечивает стабильную работу программы.