**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П.О.СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

специальности 6-05-0611-01 Информационные системы

и технологии (в игровой индустрии)

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине: «Высокоуровневые языки программирования»

на тему: **«Основные компоненты. Размещение компонентов в окнах»**

Выполнил: студент гр. ИТИ-21

Важная Я ИИ

Принял: ассистент

Карась О.В.

Гомель 2024

# **Цель работы:** изучить основные компоненты, определенные в модуле QtWidgets; изучить способы размещения компонентов в окнах.

**Задание:** Создать приложение, которое при запуске выводит окно с надписью (см. ниже). Поменять значок в заголовке на свой. В заголовке окна указать свои имя, фамилию и группу.

Под надписью должно находится 3 кнопки. При наведении мыши на кнопку должна выводится подсказка.

Кнопки

1 – Открыть окно с анкетой (на выбранную самостоятельно тему). Обязательно использовать типы размещения компонентов из таблицы 1 и типы компонентов из таблицы 2. После нажатия на кнопку «отправить» все ответы пользователя должны выводится в консоль.

2 – Открыть окно, в котором по введенному значению аргумента, вычисляется значение функции, заданной в виде графика (см. приложение 1). В окне должен быть организован ввод пользователем аргумента Х. И вывод значение функции Y. На форме вывести график в виде рисунка. Необходимо добавить кнопки: решить, очистить, выход.

3 – Открыть окно, в котором по введенному значению аргумента, определяется попадает ли точка с заданными координатами в область, закрашенную серым цветом (см. приложение 2). В окне должен быть организован ввод пользователем аргумента Х и радиуса R. И вывод – попадает или не попадает точка в область. На форме исходные данные вывести в виде рисунка. Необходимо добавить кнопки: определить, очистить, выход.

Надпись в окне

**Лабораторная работа №3**

**Основные компоненты. Размещение компонентов в окнах**

Выполнил студент группы ИТИ-21

Фамилия Имя Отчество

Таблица 1 – Тип размещения

|  |  |
| --- | --- |
| Тип размещения | № Варианта |
| Контейнер-сетка | 2, 5, 8 |

Таблица 2 – Тип компонента

|  |  |
| --- | --- |
| Тип компонента | № варианты |
| Календарь | Не четные |

**Ход работы:**

1. Импорт необходимых модулей

Все необходимые модули, нужные для создания окна представлены на рисунке 1.

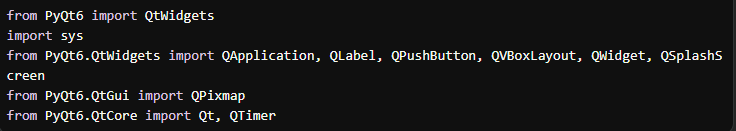


Рисунок 1 – Импорт необходимых модулей

* *PyQt6.QtWidgets*: Содержит все виджеты для создания графического интерфейса.
* *sys*: Используется для работы с системными аргументами и выхода из приложения.
* *QApplication*: Основной класс для управления потоком приложения.
* *QLabel*: Виджет для отображения текстовой информации.
* *QPushButton*: Виджет кнопки.
* *QVBoxLayout*: Макет для вертикального размещения виджетов.
* *QWidget*: Базовый класс для всех виджетов.
* *QPixmap*: Класс для работы с изображениями.

1. Функции для создания главного окна

На рисунке 2 представлен фрагмент кода для создания главного окна.

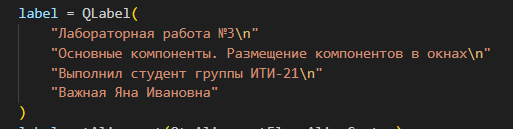


Рисунок 2 – Фрагмент кода для создания главного окна

* Создание окна: main\_window = *QWidget*() создает экземпляр главного окна.
* Установка заголовка: *setWindowTitle*("Имя Фамилия ИТИ-21") задает заголовок окна.

1. Создание макета и добавление виджетов.

На рисунке 3 представлен фрагмент кода для создания макета и добавления виджетов.

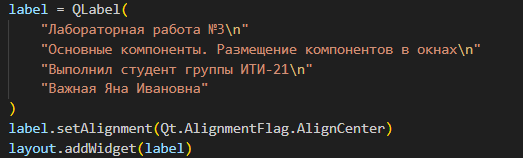


Рисунок 3 – Фрагмент кода для создания макета и добавления виджетов

* Создание макета: layout = *QVBoxLayout*() создает вертикальный макет для размещения виджетов.
* Создание метки: label = *QLabel*(...) создает текстовую метку.
* Добавление метки в макет: *layout.addWidget(label)* добавляет метку в макет.

1. Создание кнопки и добавление её в макет

На рисунке 4 представлен результат выполнения программы для четвертого задания.

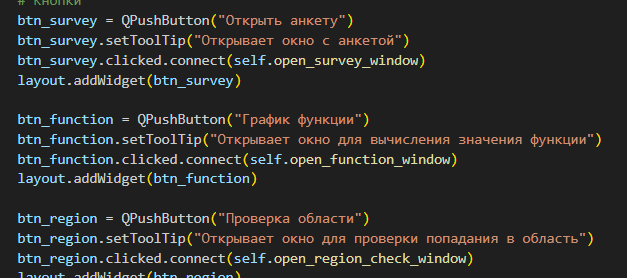


Рисунок 4 – Результат выполнения программы для четвертого задания

* Создание кнопок: *QPushButton* создает кнопки с текстом.
* Установка подсказок: *setToolTip* добавляет всплывающие подсказки.
* Соединение сигналов с методами: *clicked.connect*связывает нажатие кнопки с функцией.
* Добавление кнопок в макет: *addWidget* добавляет кнопки в макет.

1. Результат работы программы

На рисунках 8, 9, 10 представлен результат работы программы.

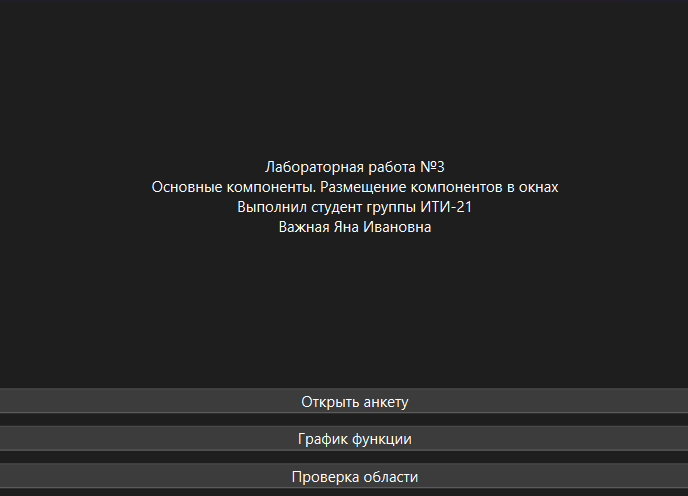


Рисунок 8 – Главное окно приложения

После запуска приложения сразу же появляется главное окно, показанное на рисунке 8. На рисунке 9 показано окно с анкетой и вывод полученных данных их анкетирования в консоль.

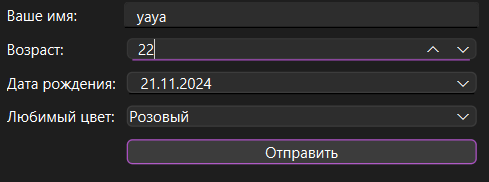


Рисунок 9 – Анкета и вывод результатов анкеты в консоль

На рисунке 10 показано окно расчета значения графика функции.

Рисунок 10 – Окно расчета значения графика функции

На рисунке 11 показан результат вывода графики функции после вычисления значения *Y*.

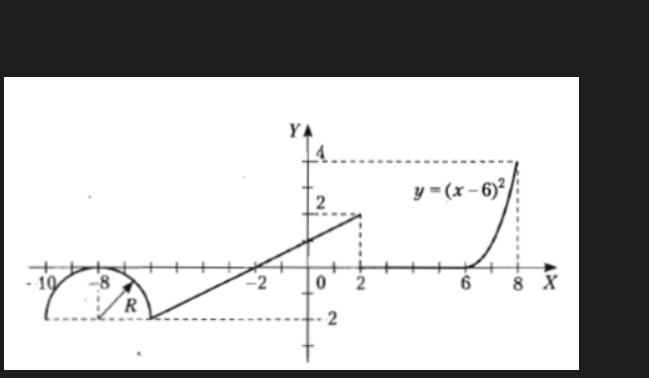


Рисунок 11 – Вывод графика функции

На рисунке 12 показано окно проверки области функции. В окне расположены кнопки «Определить», «Очистить» и «Выход». А также поля для ввода значения аргумента *X* и радиуса заданной окружности *R*.

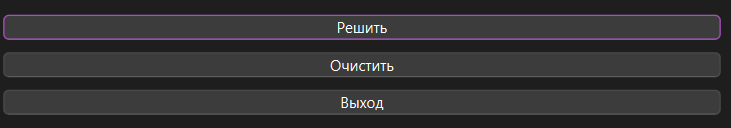


Рисунок 12 – Окно проверки области

На рисунке 13 показан результат вывода области функции после нажатия на кнопку «Определить».

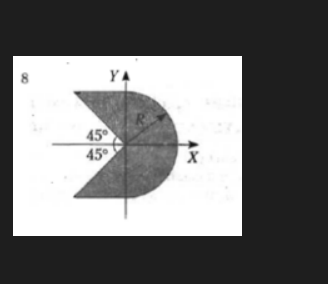


Рисунок 13 – Вывод области функции

На рисунке 14 показан результат нажатия на кнопку «Очистить».

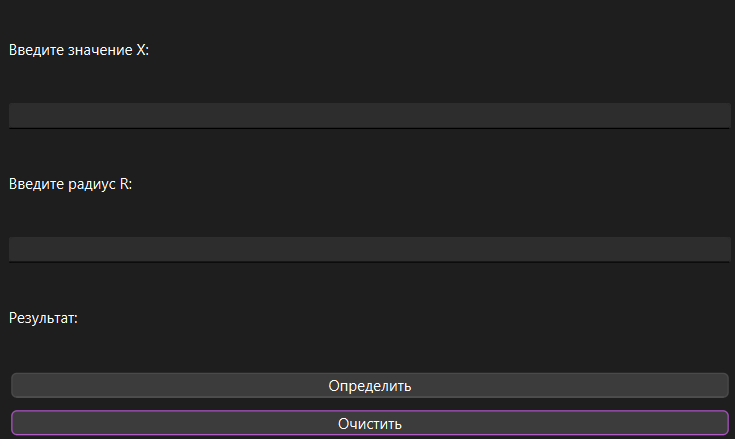


Рисунок 14 – Результат нажатия на кнопку «Очистить»

Листинг кода представлен в приложении А.

**Вывод:** в ходе выполнения лабораторной работы были изучены основные компоненты, определенные в модуле *QtWidgets*; изучены способы размещения компонентов в окнах.

**ПРИЛОЖЕНИЕ А  
(обязательное)  
Листинг кода**

*lab3\_script\_pyqt\_var8.py*

import sys

from PyQt6.QtWidgets import (

    QApplication, QMainWindow, QLabel, QPushButton, QVBoxLayout, QWidget,

    QLineEdit, QDialog, QFormLayout, QSpinBox, QComboBox, QDateEdit

)

from PyQt6.QtGui import QIcon, QPixmap

from PyQt6.QtCore import Qt, QDate

class MainApp(QMainWindow):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        # Настройки главного окна

        self.setWindowTitle("Спиридонов Дмитрий Алексеевич, ИТИ-21")

        self.setGeometry(200, 200, 600, 400)

        # Главный виджет и компоновка

        main\_widget = QWidget()

        self.setCentralWidget(main\_widget)

        layout = QVBoxLayout()

        # Текст в главном окне

        label = QLabel(

            "Лабораторная работа №3\n"

            "Основные компоненты. Размещение компонентов в окнах\n"

            "Выполнил студент группы ИТИ-21\n"

            "Спиридонов Дмитрий Алексеевич"

        )

        label.setAlignment(Qt.AlignmentFlag.AlignCenter)

        layout.addWidget(label)

        # Кнопки

        btn\_survey = QPushButton("Открыть анкету")

        btn\_survey.setToolTip("Открывает окно с анкетой")

        btn\_survey.clicked.connect(self.open\_survey\_window)

        layout.addWidget(btn\_survey)

        btn\_function = QPushButton("График функции")

        btn\_function.setToolTip("Открывает окно для вычисления значения функции")

        btn\_function.clicked.connect(self.open\_function\_window)

        layout.addWidget(btn\_function)

        btn\_region = QPushButton("Проверка области")

        btn\_region.setToolTip("Открывает окно для проверки попадания в область")

        btn\_region.clicked.connect(self.open\_region\_check\_window)

        layout.addWidget(btn\_region)

        # Установка компоновки

        main\_widget.setLayout(layout)

    # Открытие окна с анкетой

    def open\_survey\_window(self):

        survey\_dialog = SurveyWindow(self)

        survey\_dialog.exec()

    # Открытие окна с графиком функции

    def open\_function\_window(self):

        function\_dialog = FunctionWindow(self)

        function\_dialog.exec()

    # Открытие окна с проверкой области

    def open\_region\_check\_window(self):

        region\_dialog = RegionCheckWindow(self)

        region\_dialog.exec()

class SurveyWindow(QDialog):

    def \_\_init\_\_(self, parent=None):

        super().\_\_init\_\_(parent)

        self.setWindowTitle("Анкета")

        self.setGeometry(300, 300, 400, 300)

        # Создание формы

        form\_layout = QFormLayout()

        self.name\_input = QLineEdit()

        form\_layout.addRow("Ваше имя:", self.name\_input)

        self.age\_input = QSpinBox()

        self.age\_input.setRange(1, 100)

        form\_layout.addRow("Возраст:", self.age\_input)

        self.birth\_date = QDateEdit()

        self.birth\_date.setCalendarPopup(True)

        self.birth\_date.setDate(QDate.currentDate())

        form\_layout.addRow("Дата рождения:", self.birth\_date)

        self.color\_input = QComboBox()

        self.color\_input.addItems(["Красный", "Синий", "Зеленый"])

        form\_layout.addRow("Любимый цвет:", self.color\_input)

        # Кнопка отправки

        submit\_button = QPushButton("Отправить")

        submit\_button.clicked.connect(self.submit\_form)

        form\_layout.addWidget(submit\_button)

        self.setLayout(form\_layout)

    def submit\_form(self):

        print(f"Имя: {self.name\_input.text()}")

        print(f"Возраст: {self.age\_input.value()}")

        print(f"Дата рождения: {self.birth\_date.date().toString()}")

        print(f"Любимый цвет: {self.color\_input.currentText()}")

        self.accept()

class FunctionWindow(QDialog):

    def \_\_init\_\_(self, parent=None):

        super().\_\_init\_\_(parent)

        self.setWindowTitle("График функции")

        self.setGeometry(300, 300, 600, 200)

        layout = QVBoxLayout()

        # Поля ввода

        self.x\_input = QLineEdit()

        layout.addWidget(QLabel("Введите значение X:"))

        layout.addWidget(self.x\_input)

        self.result\_label = QLabel("Значение Y:")

        layout.addWidget(self.result\_label)

        # Кнопки

        solve\_button = QPushButton("Решить")

        solve\_button.clicked.connect(self.solve\_function)

        layout.addWidget(solve\_button)

        clear\_button = QPushButton("Очистить")

        clear\_button.clicked.connect(self.clear\_inputs)

        layout.addWidget(clear\_button)

        exit\_button = QPushButton("Выход")

        exit\_button.clicked.connect(self.close)

        layout.addWidget(exit\_button)

        self.setLayout(layout)

    def solve\_function(self):

        try:

            x = float(self.x\_input.text())

            y = self.calculate\_function\_y(x)

            self.result\_label.setText(f"Значение Y: {y}")

            self.show\_graph()  # Отображаем график в отдельном окне

        except ValueError:

            self.result\_label.setText("Ошибка: некорректный ввод")

    def clear\_inputs(self):

        self.x\_input.clear()

        self.result\_label.setText("Значение Y:")

    def calculate\_function\_y(self, x):

        if -8 <= x <= 0:

            return -2

        elif 0 < x <= 6:

            return x - 2

        elif x > 6:

            return (x - 6) \*\* 2

        return 0

    def show\_graph(self):

        graph\_window = GraphWindow()

        graph\_window.exec()  # Открываем как модальное окно

class GraphWindow(QDialog):

    def \_\_init\_\_(self, parent=None):

        super().\_\_init\_\_(parent)

        self.setWindowTitle("График функции")

        self.setGeometry(300, 300, 600, 400)

        layout = QVBoxLayout()

        self.graph\_label = QLabel()

        self.graph\_pixmap = QPixmap("1.png")  # Укажите путь к изображению

        self.graph\_label.setPixmap(self.graph\_pixmap)

        layout.addWidget(self.graph\_label)

        self.setLayout(layout)

class RegionCheckWindow(QDialog):

    def \_\_init\_\_(self, parent=None):

        super().\_\_init\_\_(parent)

        self.setWindowTitle("Проверка области")

        self.setGeometry(300, 300, 600, 400)

        layout = QVBoxLayout()

        # Поля ввода

        self.x\_input = QLineEdit()

        layout.addWidget(QLabel("Введите значение X:"))

        layout.addWidget(self.x\_input)

        self.r\_input = QLineEdit()

        layout.addWidget(QLabel("Введите радиус R:"))

        layout.addWidget(self.r\_input)

        self.result\_label = QLabel("Результат:")

        layout.addWidget(self.result\_label)

        # Кнопки

        check\_button = QPushButton("Определить")

        check\_button.clicked.connect(self.check\_region)

        layout.addWidget(check\_button)

        clear\_button = QPushButton("Очистить")

        clear\_button.clicked.connect(self.clear\_inputs)

        layout.addWidget(clear\_button)

        exit\_button = QPushButton("Выход")

        exit\_button.clicked.connect(self.close)

        layout.addWidget(exit\_button)

        self.setLayout(layout)

        # Для графика

        self.point\_x = None

        self.point\_r = None

    def check\_region(self):

        try:

            self.point\_x = float(self.x\_input.text())

            self.point\_r = float(self.r\_input.text())

            if self.is\_point\_in\_region(self.point\_x, self.point\_r):

                self.result\_label.setText("Точка попадает в область")

            else:

                self.result\_label.setText("Точка не попадает в область")

            self.show\_graph()  # Отображаем график области

        except ValueError:

            self.result\_label.setText("Ошибка: некорректный ввод")

    def clear\_inputs(self):

        self.x\_input.clear()

        self.r\_input.clear()

        self.result\_label.setText("Результат:")

        self.point\_x = None

        self.point\_r = None

    def is\_point\_in\_region(self, x, r):

        if -r <= x <= 0 and -r <= r:

            return True

        elif 0 < x <= r and 0 <= x <= r:

            return True

        elif x > r and (x - r) \*\* 2 + r \*\* 2 <= r \*\* 2:

            return True

        else:

            return False

    def show\_graph(self):

        graph\_window = GraphRegionWindow()

        graph\_window.exec()  # Открываем как модальное окно

class GraphRegionWindow(QDialog):

    def \_\_init\_\_(self, parent=None):

        super().\_\_init\_\_(parent)

        self.setWindowTitle("График области")

        self.setGeometry(300, 300, 600, 400)

        layout = QVBoxLayout()

        self.graph\_label = QLabel()

        self.graph\_pixmap = QPixmap("2.png")  # Укажите путь к изображению

        self.graph\_label.setPixmap(self.graph\_pixmap)

        layout.addWidget(self.graph\_label)

        self.setLayout(layout)

if \_\_name\_\_ == '\_\_main\_\_':

    app = QApplication(sys.argv)

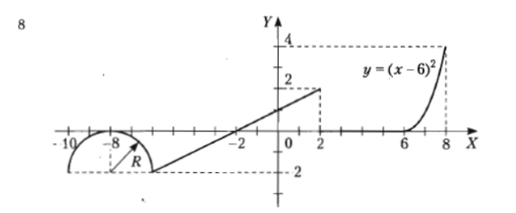
    main\_window = MainApp()

    main\_window.show()

    sys.exit(app.exec())

**Приложение 1**

**Графики функции**



**Приложение 2**

**Области функции**

