ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
УФИМСКИЙ УНИВЕРСИТЕТ НАУКИ И ТЕХНОЛОГИЙ

Кафедра ВМиК

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

по предмету **«Объектно-Ориентированное Программирование»**

Выполнил: студент группы МО-204Б

Исламов Ильнур Фандасович.

Проверил:
доцент каф. ВМиК

Котельников В.А.

Уфа 2025г.

Цель лабораторной работы

В рамках лабораторной работы необходимо разобраться:

- каким образом создаётся интерфейс приложения из стандартных объектов;
- каким образом изменяются до и после запуска программы свойства объектов;
- каким образом одни объекты вызывают методы других объектов;
- в каких случаях срабатывают какие события из стандартного списка.

Задание

Создание простейшего desktop-приложения с графическим интерфейсом пользователя (GUI) в любой среде (Visual Studio - Windows Forms Application или WPF; C++ Builder - VCL Application; C++ и Qt Framework, Python + PyQt и т.д.).

- Создание простейшего приложения с GUI
  - форма, меню, пункты меню
  - добавлять на форму разнообразные стандартные объекты и изучать их свойства, влияющие на внешний вид и поведение
- Управление простейшими событиями (при происходящих событиях должна появляться какая-либо реакция приложения, позволяющая понять, что событие произошло)
  - нажатие/отпускание мыши, перемещение мыши, нажатие/отпускание клавиши
  - выбор пункта меню
  - перерисовка (как и для чего использовать событие Paint – обратите внимание, не надо делать программу Paint, надо разобраться, как использовать событие Paint: показать это

событие в списке событий, разобраться, когда оно происходит и как его использовать)

- таймер (как использовать событие таймера для организации задержек)
- изменение размера окна (как и для чего использовать событие SizeChanged)

- При использовании Qt/pyQt необходимо разобраться и научиться использовать два механизма: сигналы/слоты и события/обработчики.
- Управление событиями, влияние объектов друг на друга (при каком-либо событии, связанным с одним объектом, должно что-то меняться в другом объекте).
- Управление событиями и обработчиками событий: один обработчик на несколько событий разных объектов (например, три различные кнопки, один обработчик, нажатая кнопка меняет цвет), программный вызов обработчиков (вызов метода-обработчика в программе), программный вызов событий (вызов методов типа PerformClick).
- Управление свойствами объектов в design time и в runtime: выделить два главных свойства каждого объекта и научиться изменять их как до запуска программы, так и во время работы приложения.
- Изучить (знать, для каких целей используются и как работают) все стандартные элементы управления:
  - В случае Visual Studio - Windows Forms Application: все компоненты с закладок «Стандартные элементы управления», «Контейнеры», плюс меню и таймер.
  - В случае C++ Builder - VCL Application: все компоненты с закладок «Standard», «Additional», «Win32», «System».
  - В остальных случаях – аналогичный набор основных компонент
- Для всех стандартных компонентов из предыдущего пункта необходимо понять их назначение (для чего используется чаще всего), и выделить основное свойство (одно или два) или основной

метод (например, для текстового поля ввода – свойство, хранящее текст, для выпадающего списка – список элементов и номер выбранного и т.д.). Необходимо у каждого типового компонента выделить его главные свойства (одно-два) и научиться манипулировать ими до запуска программы и во время её работы.

- Динамическое создание и уничтожение объектов интерфейса:
    - создание элементов интерфейса как реакция на события: при нажатии мышкой на любое место на форме, должен создаваться объект любого типового класса (например, кнопка). Элемент обязательно должен иметь обработчик события (например, если создаются кнопки, то при нажатии на них должна появляться реакция)

Ход выполнения лабораторной работы

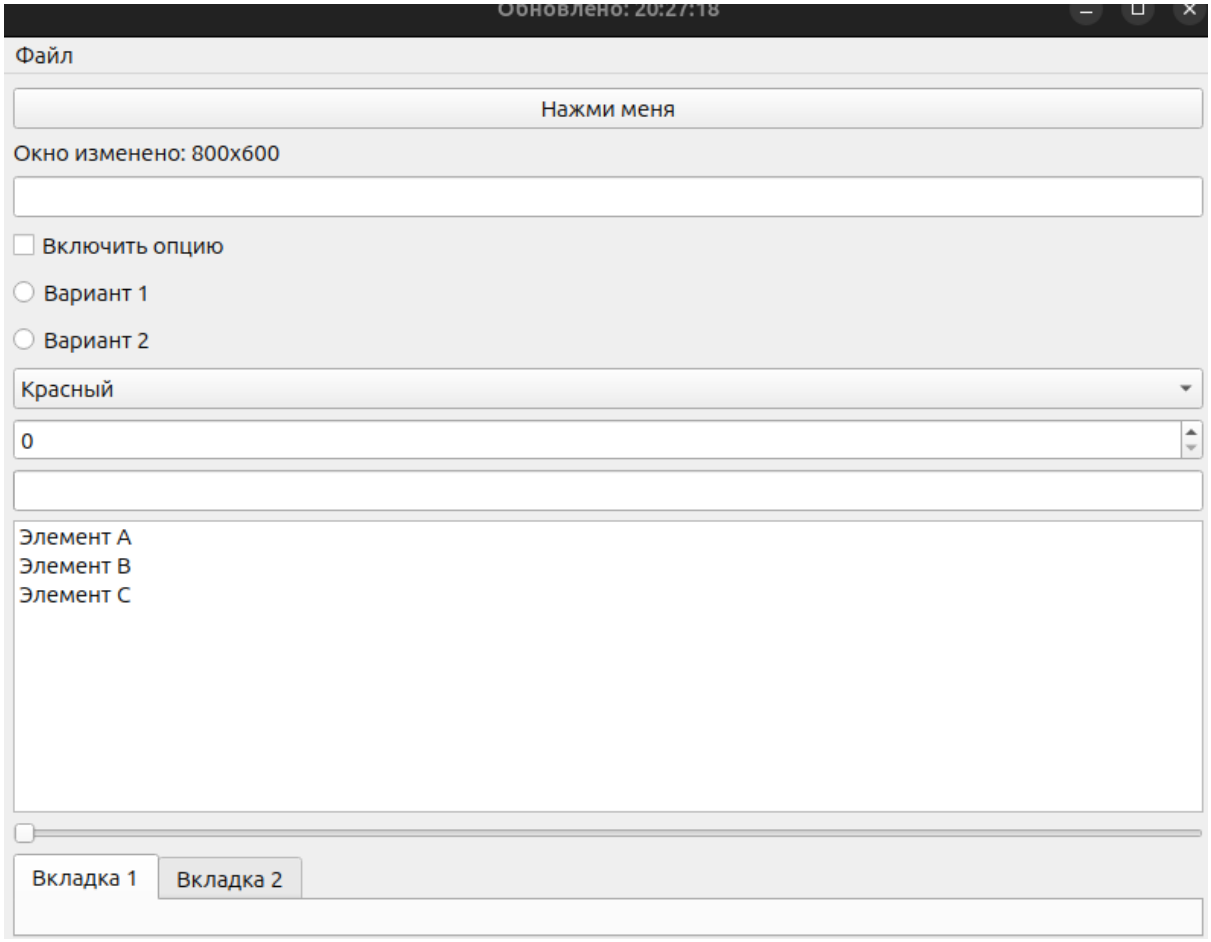

Рис .1 Создание окна GUI

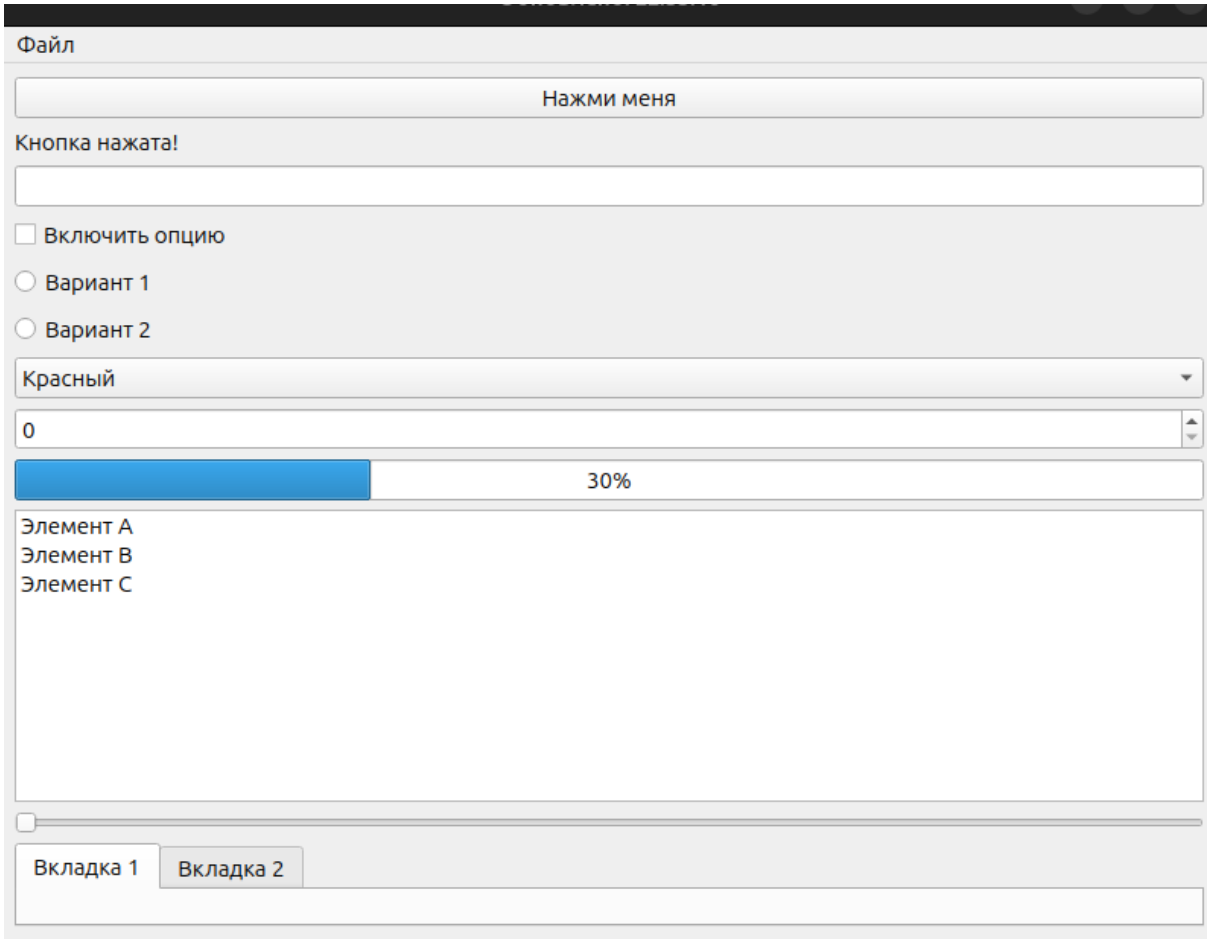

Рис .2 При нажатии вызывает слот onButtonClicked(), который меняет текст QLabel и двигает прогресс-бар на 10%.

Рис .3, Рис .4, Рис .5 Показывает текстовое состояние программы (например, результат нажатий, размер окна, выбранный цвет и т.д.).

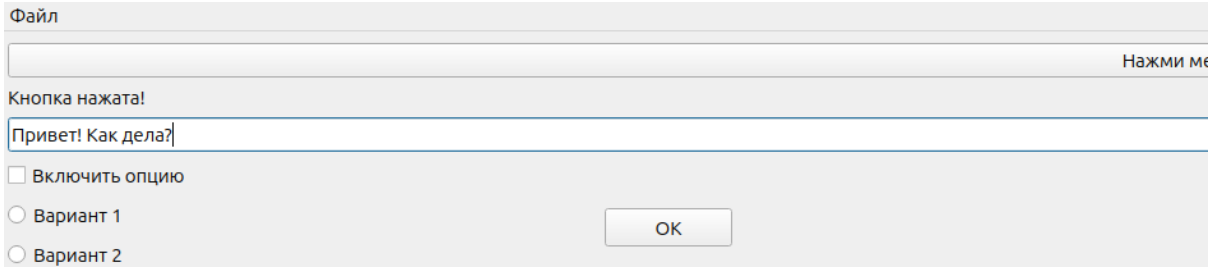

Рис .6 Текст вводится пользователем. Можно сказать: “используется для ввода данных пользователем”.

Рис .7 Демонстрируют выбор одного варианта из двух (“вариант 1” и “вариант 2”).

Рис .8 Имеет 2 вкладки: “Вкладка 1” и “Вкладка 2”. Это пример контейнера для группировки элементов.

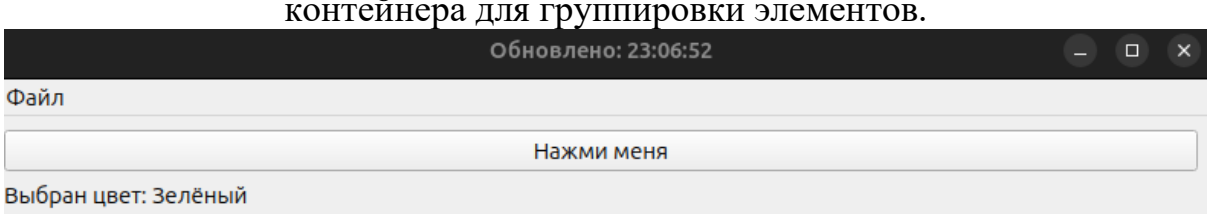

Рис .9 Показывает пример фонового события — обновляет время в заголовке.

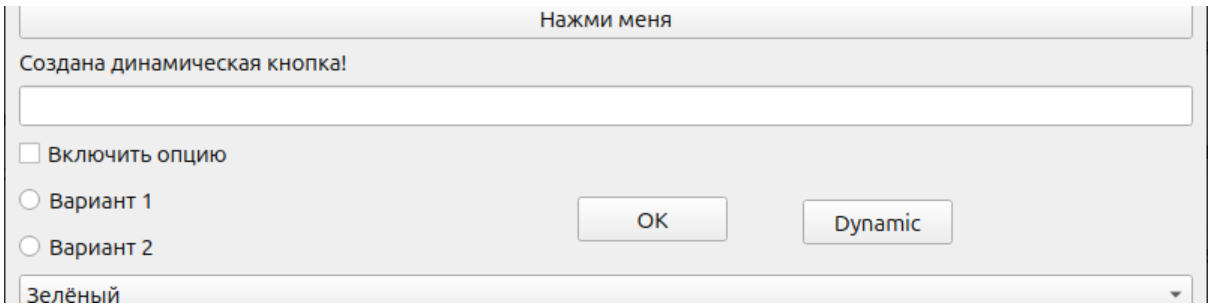

Рис .10 При клике по пустому месту на форме создаётся новая кнопка (“Dynamic”), которая реагирует на нажатие.

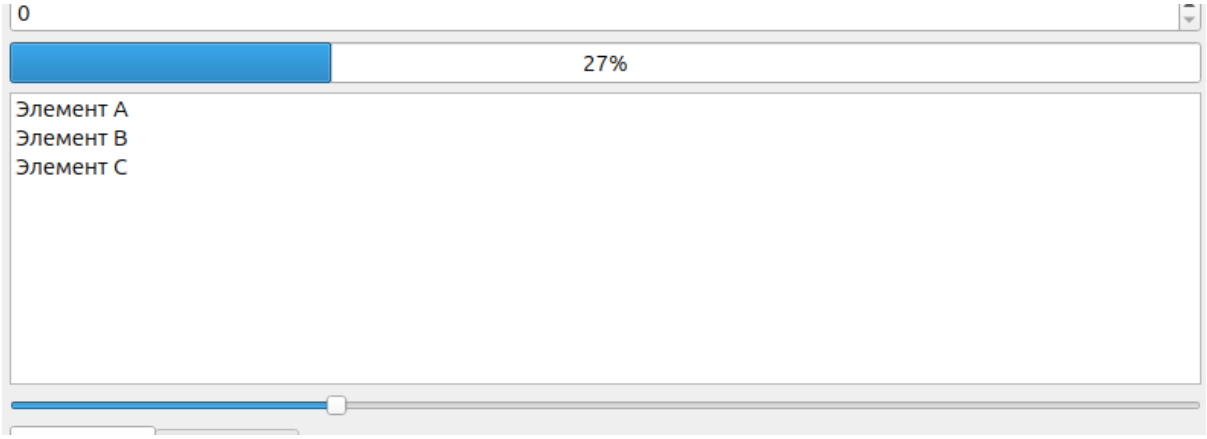

Рис .11 При движении ползунка изменяет значение уровень заполнения.\

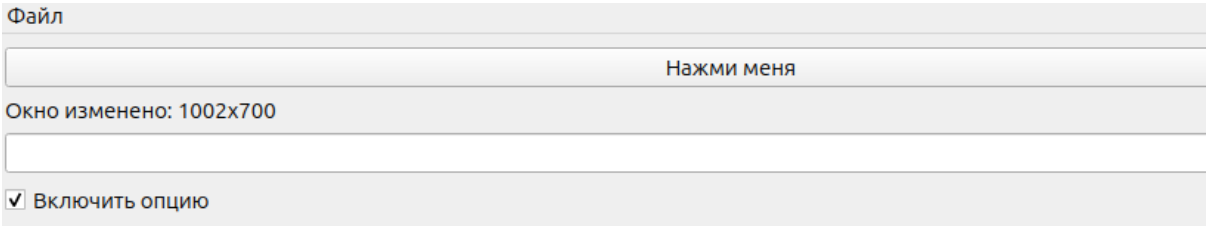

Рис .12 Пример булевого свойства объекта (включен/выключен).

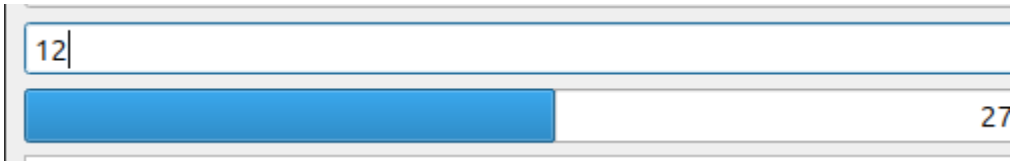

Рис .13 Позволяет изменять числовое значение.

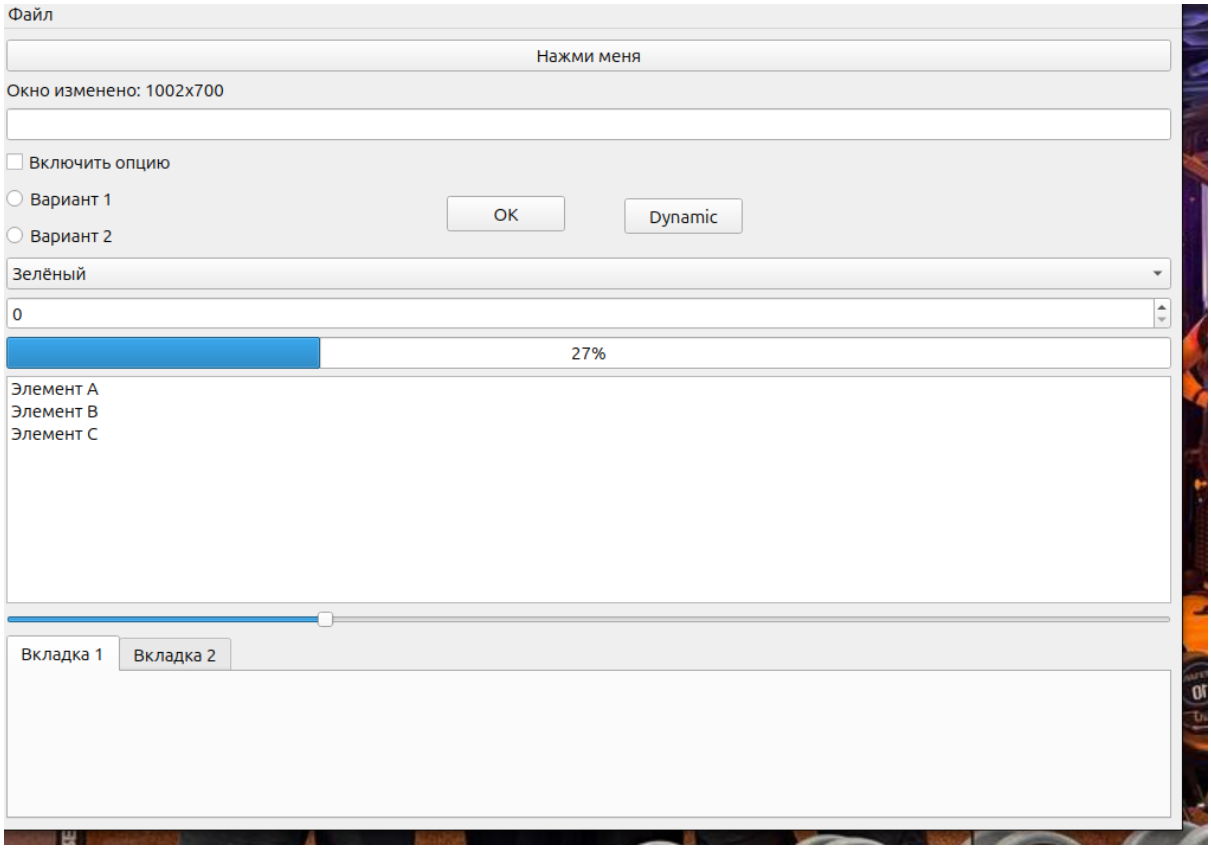

Рис .14 При изменении размеров окна в метке отображаются новые размеры.

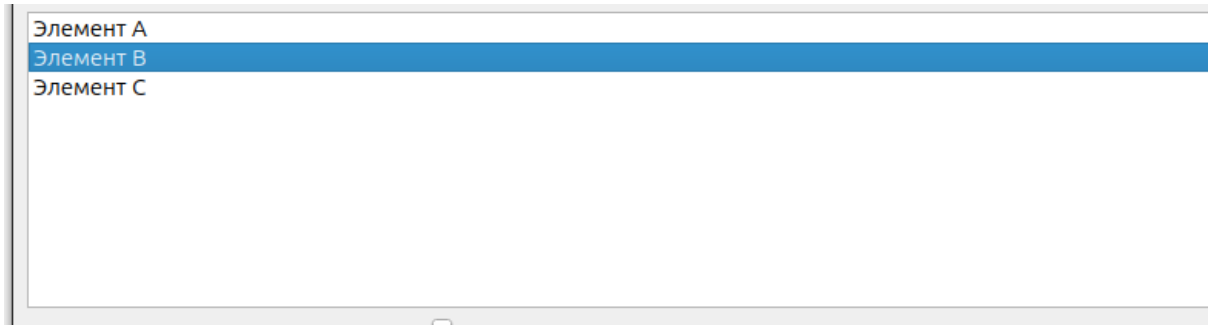

Рис .15 Содержит текстовые элементы ("Элемент A", "B", "C").

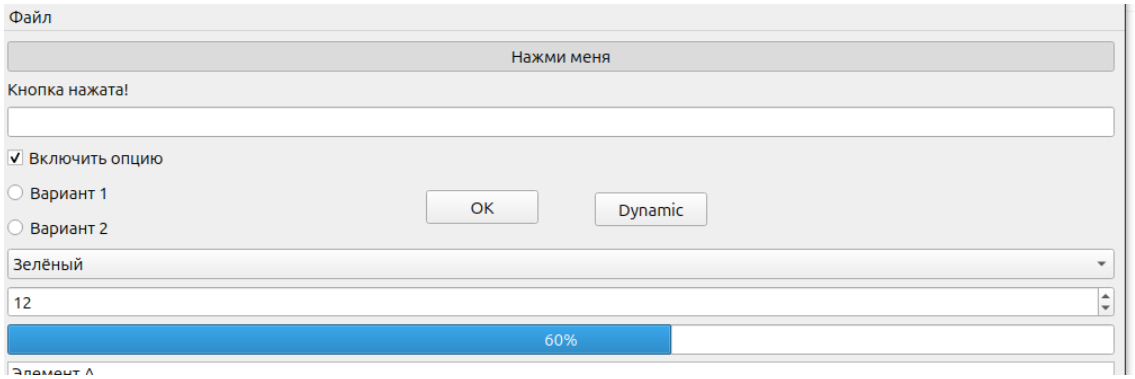

Рис .16 При нажатии любой клавише будет видно её название.

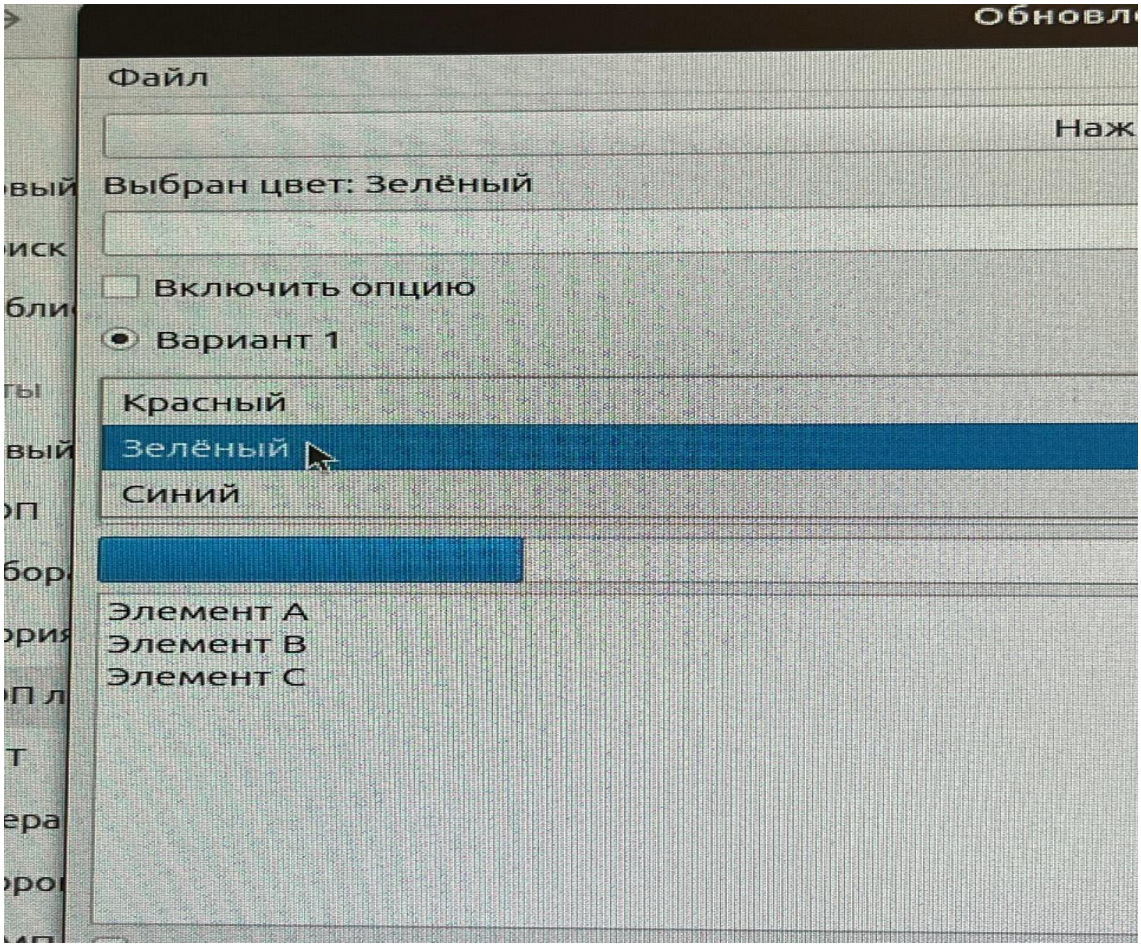

Рис .17 Позволяет выбрать цвет (“Красный”, “Зелёный”, “Синий”).

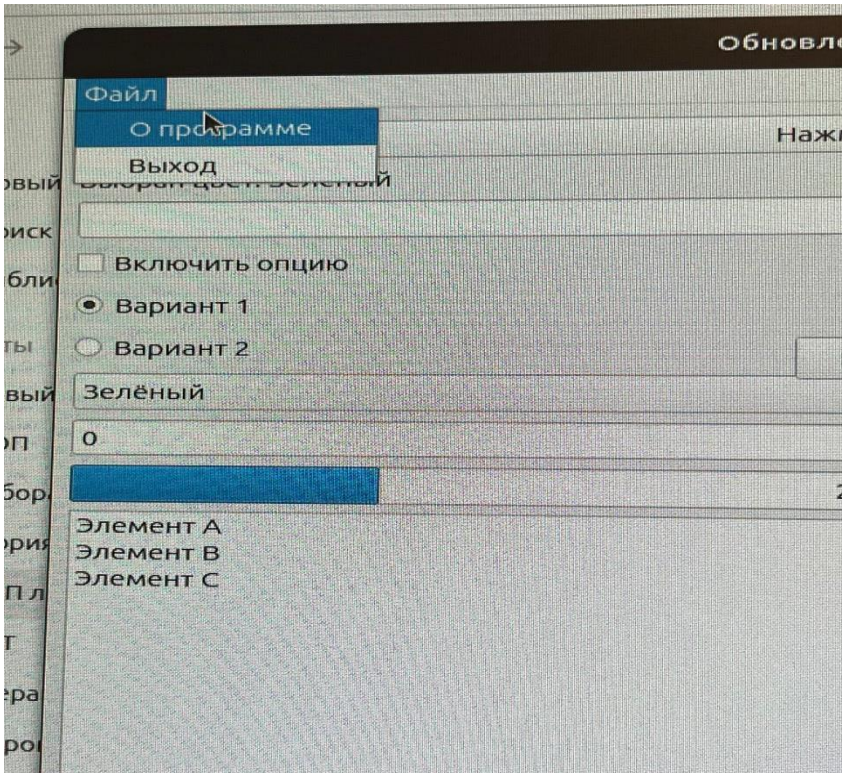

Рис .18 “О программе” открывает второе окно (AboutWindow), “Выход” закрывает приложение.

Выводы по лабораторной работе

В результате выполнения лабораторной работы были освоены основы создания desktop-приложений с графическим интерфейсом. Были изучены принципы работы со свойствами и событиями стандартных компонентов, а также механизмы их взаимодействия. Также было создано приложение, демонстрирующее разнообразие элементов управления и их реакцию на действия пользователя, что заложило базовые навыки разработки GUI.

Приложение №1

```
#include "mainwindow.h"
```

```
#include <QVBoxLayout>

#include <QMouseEvent>

#include <QKeyEvent>

#include <QMessageBox>

#include <QTime>

MainWindow::MainWindow(QWidget *parent) : QMainWindow(parent) {
resize(800, 600); setWindowTitle("Лабораторная работа №1 — Кнопки и
Формы");

// Главное меню
QMenu *menuFile = menuBar()->addMenu("Файл");
QAction *aboutAction = menuFile->addAction("О программе");
QAction *exitAction = menuFile->addAction("Выход");
connect(exitAction, &QAction::triggered, this, &QMainWindow::close);
connect(aboutAction, &QAction::triggered, this,
&MainWindow::openAboutWindow);

QWidget *central = new QWidget(this);
setCentralWidget(central);

QVBoxLayout *mainLayout = new QVBoxLayout(central);

mainButton = new QPushButton("Нажми меня");
infoLabel = new QLabel("Информация появится здесь");
inputField = new QLineEdit();
checkBox = new QCheckBox("Включить опцию");
radio1 = new QRadioButton("Вариант 1");
radio2 = new QRadioButton("Вариант 2");
comboBox = new QComboBox();
comboBox->addItems({"Красный", "Зелёный", "Синий"});
spinBox = new QSpinBox();
progressBar = new QProgressBar();
progressBar->setRange(0, 100);
listWidget = new QListWidget();
```

```
listWidget->addItems({"Элемент A", "Элемент B", "Элемент C"});
slider = new QSlider(Qt::Horizontal);
slider->setRange(0, 100);

tabs = new QTabWidget();
QWidget *tab1 = new QWidget();
QWidget *tab2 = new QWidget();
tabs->addTab(tab1, "Вкладка 1");
tabs->addTab(tab2, "Вкладка 2");

mainLayout->addWidget(mainButton);
mainLayout->addWidget(infoLabel);
mainLayout->addWidget(inputField);
mainLayout->addWidget(checkBox);
mainLayout->addWidget(radio1);
mainLayout->addWidget(radio2);
mainLayout->addWidget(comboBox);
mainLayout->addWidget(spinBox);
mainLayout->addWidget(progressBar);
mainLayout->addWidget(listWidget);
mainLayout->addWidget(slider);
mainLayout->addWidget(tabs);

// Таймер
timer = new QTimer(this);
connect(timer, &QTimer::timeout, this, &MainWindow::onTimerTick);
timer->start(1000);

// События
connect(mainButton, &QPushButton::clicked, this,
&MainWindow::onButtonClicked);
connect(slider, &QSlider::valueChanged, progressBar,
&QProgressBar::setValue);
connect(comboBox, &QComboBox::currentTextChanged, [this](const QString
&color){
    infoLabel->setText("Выбран цвет: " + color);
```

```
});


}

MainWindow::~MainWindow() = default;

void MainWindow::onButtonClicked() { infoLabel->setText("Кнопка нажата!"); progressValue += 10; if (progressValue > 100) progressValue = 0; progressBar->setValue(progressValue); }

void MainWindow::onTimerTick() { setWindowTitle("Обновлено: " + QTime::currentTime().toString("hh:mm:ss")); }

void MainWindow::mousePressEvent(QMouseEvent *event) { QPushButton *newButton = new QPushButton("Dynamic", this); newButton->move(event->pos()); newButton->show(); connect(newButton, &QPushButton::clicked, this, newButton{ infoLabel->setText("Создана динамическая кнопка!"); newButton->setText("OK"); }); }

void MainWindow::keyPressEvent(QKeyEvent *event) { QString keyName = QKeySequence(event->key()).toString(); infoLabel->setText("Нажата клавиша: " + keyName);

if (event->key() == Qt::Key_Space)
    infoLabel->setText("Нажата клавиша ПРОБЕЛ");


}

void MainWindow::resizeEvent(QResizeEvent *event) { infoLabel->setText("Окно изменено: " + QString::number(event->size().width()) + "x" + QString::number(event->size().height())); }

void MainWindow::openAboutWindow() { aboutWin = new AboutWindow(this); aboutWin->show(); }
```