ПРИЛОЖЕНИЕ 1

Исходный код программы

Файл int.py

```
# -*- coding: utf-8 -*-

from PyQt5 import QtCore, QtGui, QtWidgets
from cv2 import cv2
from PIL import Image
import os
import numpy as np

# ================================================================
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        font = QtGui.QFont()
        font.setFamily("Segoe UI")
        font.setPointSize(12)
        font.setBold(False)
        font.setItalic(False)
        font.setWeight(50)

        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(450, 650)
        MainWindow.setMinimumSize(QtCore.QSize(450, 650))
        MainWindow.setMaximumSize(QtCore.QSize(450, 650))
        MainWindow.setSizeIncrement(QtCore.QSize(0, 0))
        MainWindow.setWindowTitle("Find_Dubl")
        MainWindow.setFont(font)
        MainWindow.setAnimated(True)

        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        MainWindow.setCentralWidget(self.centralwidget)

        self.lineEdit = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit.setGeometry(QtCore.QRect(20, 20, 270, 30))
        font.setPointSize(10)
        self.lineEdit.setFont(font)
        font.setPointSize(12)
        self.lineEdit.setFocusPolicy(QtCore.Qt.ClickFocus)
```

```
self.lineEdit.setObjectName("lineEdit")
self.lineEdit.setPlaceholderText("Путь к каталогу")

self.lineEdit_2 = QtWidgets.QLineEdit(self.centralwidget)
self.lineEdit_2.setObjectName("lineEdit_2")
self.lineEdit_2.hide()

self.pushButton = QtWidgets.QPushButton(self.centralwidget)
self.pushButton.setGeometry(QtCore.QRect(300, 19, 130, 32))
self.pushButton.setFont(font)
self.pushButton.setObjectName("pushButton")
self.pushButton.clicked.connect(self.selectCatalog)
self.pushButton.setText("Выбрать каталог")

self.label = QtWidgets.QLabel(self.centralwidget)
self.label.setGeometry(QtCore.QRect(20, 70, 410, 480))
self.label.setFrameShape(QtWidgets.QFrame.Panel)
self.label.setAlignment(QtCore.Qt.AlignCenter)
self.label.setLineWidth(1)
self.label.setMidLineWidth(1)
self.label.setText("")
self.label.setObjectName("label")

self.pushButton_2 = QtWidgets.QPushButton(self.centralwidget)
self.pushButton_2.setGeometry(QtCore.QRect(20, 70, 410, 480))
self.pushButton_2.setFont(font)
self.pushButton_2.setAutoFillBackground(False)
self.pushButton_2.setInputMethodHints(QtCore.Qt.ImhNone)
self.pushButton_2.setCheckable(False)
self.pushButton_2.setAutoDefault(False)
self.pushButton_2.setFlat(True)
self.pushButton_2.setObjectName("pushButton_2")
self.pushButton_2.setText("Выбрать изображение")
self.pushButton_2.clicked.connect(self.selectImage)

self.pushButton_3 = QtWidgets.QPushButton(self.centralwidget)
self.pushButton_3.setGeometry(QtCore.QRect(175, 570, 100, 32))
self.pushButton_3.setFont(font)
self.pushButton_3.setObjectName("pushButton_3")
self.pushButton_3.setText("Найти")
self.pushButton_3.clicked.connect(self.find)

self.statusbar = QtWidgets.QStatusBar(MainWindow)
```

```
self.statusbar.setObjectName("statusbar")
MainWindow.setStatusBar(self.statusbar)

self.menuBar = QtWidgets.QMenuBar(MainWindow)
self.menuBar.setGeometry(QtCore.QRect(0, 0, 450, 21))
self.menuBar.setObjectName("menuBar")
MainWindow.setMenuBar(self.menuBar)

self.action = QtWidgets.QAction(MainWindow)
self.action.setObjectName("action")
self.action.setText("Выбрать изображение")
self.action.triggered.connect(self.selectImage)

self.action_2 = QtWidgets.QAction(MainWindow)
self.action_2.setObjectName("action_2")
self.action_2.setText("Выбрать каталог поиска")
self.action_2.triggered.connect(self.selectCatalog)

self.action_3 = QtWidgets.QAction(MainWindow)
self.action_3.setObjectName("action_3")
self.action_3.setText("Выход")
self.action_3.triggered.connect(self.exit)

self.action_4 = QtWidgets.QAction(MainWindow)
self.action_4.setObjectName("action_4")
self.action_4.setText("Справка")
self.action_4.triggered.connect(self.useDef1)

self.action_5 = QtWidgets.QAction(MainWindow)
self.action_5.setObjectName("action_5")
self.action_5.setText("О программе")
self.action_5.triggered.connect(self.useDef2)

self.action_6 = QtWidgets.QAction(MainWindow)
self.action_6.setObjectName("action_6")
self.action_6.setText("Очистить все")
self.action_6.triggered.connect(self.clearAll)

self.menu = QtWidgets.QMenu(self.menuBar)
self.menu.setObjectName("menu")
self.menu.setTitle("Файл")
self.menu.addAction(self.action) # Файл -> Выбрать изображение
self.menu.addAction(self.action_2) # Файл->Выбратькаталог поиска
```

```
            self.menu.addAction(self.action_6) # Файл -> Очистить все
            self.menu.addAction(self.action_3) # Файл -> Выход

            self.menu_2 = QtWidgets.QMenu(self.menuBar)
            self.menu_2.setObjectName("menu_2")
            self.menu_2.setTitle("Справка")
            self.menu_2.addAction(self.action_4) # Справка -> Справка
            self.menu_2.addAction(self.action_5) # Справка -> О программе
            self.menuBar.addAction(self.menu.menuAction()) # Файл
            self.menuBar.addAction(self.menu_2.menuAction()) # Справка

            QtCore.QMetaObject.connectSlotsByName(MainWindow)

        def selectCatalog(self): # Выбрать каталог

self.lineEdit.setText(str(QtWidgets.QFileDialog.getExistingDirectory()))

        def selectImage(self): # Выбрать изображение
            way = str(QtWidgets.QFileDialog.getOpenFileName()[0])
            img = QtGui.QPixmap(way)
            w, h = self.label.width(), self.label.height()
            w_img, h_img = img.width(), img.height()
            try:
                x = w / w_img if w_img > h_img else h / h_img
                self.lineEdit_2.setText(way)
                self.label.setPixmap(img.scaled(int(w_img*x), int(h_img*x)))
                self.pushButton_2.setText("")
            except:
                self.label.setPixmap(img)
                self.pushButton_2.setText("Выбрать изображение")

        def clearAll(self): # Очистить все
            img = QtGui.QPixmap("")
            self.label.setPixmap(img)
            self.pushButton_2.setText("Выбрать изображение")
            self.lineEdit_2.setText("")
            self.lineEdit.setText("")

        def exit(self): # Выход
            MainWindow.close()

        def useDef1(self):
            self.dialog = Def1(self)
```

```
            self.dialog.show()

        def useDef2(self):
            self.dialog = Def2(self)
            self.dialog.show()

        def errors(self, way, way_img): # Есть ли изображение и каталог
            msgBox = QtWidgets.QMessageBox()
            msgBox.setWindowTitle("Предупреждение")
            if way == "" and way_img == "":
                msgBox.setText("Выберите изображение и каталог поиска.")
                msgBox.exec()
                return False
            elif way_img == "":
                msgBox.setText("Выберите изображение.")
                msgBox.exec()
                return False
            elif way == "":
                msgBox.setText("Выберите каталог поиска.")
                msgBox.exec()
                return False
            return True

        def find(self): # Найти
            way = str(self.lineEdit.text())
            way_img = str(self.lineEdit_2.text())
            if self.errors(way, way_img):
                self.dialog = Find(self)
                self.dialog.show()

    # =================================================================
    class Find(QtWidgets.QDialog):
        def __init__(self, parent):
            super().__init__()
            font = QtGui.QFont()
            font.setFamily("Segoe UI")
            font.setPointSize(12)
            font.setBold(False)
            font.setItalic(False)
            font.setWeight(50)

            MainWindow.resize(360, 172)
            self.setMinimumSize(QtCore.QSize(360, 172))
```

```
self.setMaximumSize(QtCore.QSize(360, 172))
self.setSizeIncrement(QtCore.QSize(0, 0))
self.setFont(font)
self.setWindowFlags(QtCore.Qt.Window)
self.setWindowTitle("Find_Dubl")

self.centralwidget = QtWidgets.QWidget(self)
self.centralwidget.setObjectName("centralwidget")

self.label = QtWidgets.QLabel(self.centralwidget)
self.label.setGeometry(QtCore.QRect(10, 10, 340, 30))
self.label.setFont(font)
self.label.setObjectName("label")
self.label.setText("Проверяется файл 0/0")

self.label_2 = QtWidgets.QLabel(self.centralwidget)
self.label_2.setGeometry(QtCore.QRect(10, 40, 340, 30))
font.setPointSize(10)
self.label_2.setFont(font)
font.setPointSize(12)
self.label_2.setObjectName("label_2")
self.label_2.setText("")

self.pushButton = QtWidgets.QPushButton(self.centralwidget)
self.pushButton.setGeometry(QtCore.QRect(210, 120, 130, 32))
self.pushButton.setFont(font)
self.pushButton.setObjectName("pushButton")
self.pushButton.setText("Стоп")

self.pushButton_2 = QtWidgets.QPushButton(self.centralwidget)
self.pushButton_2.setGeometry(QtCore.QRect(210, 120, 130, 32))
self.pushButton_2.setFont(font)
self.pushButton_2.setObjectName("pushButton_2")
self.pushButton_2.setText("OK")
self.pushButton_2.hide()

self.progressBar = QtWidgets.QProgressBar(self.centralwidget)
self.progressBar.setGeometry(QtCore.QRect(10, 80, 340, 30))
font.setPointSize(1)
self.progressBar.setFont(font)
self.progressBar.setFormat("")
self.progressBar.setObjectName("progressBar")
self.progressBar.setMinimum(0)
```

```
            self.progressBar.setProperty("value", 0)

            self.way = str(parent.lineEdit.text())
            self.way_img = str(parent.lineEdit_2.text())
            self.catalog_i = self.catalog(self.way)
            self.progressBar.setMaximum(len(self.catalog_i))
            self.pushButton.clicked.connect(self.click)

            self.thread1 = Thread(self)

self.thread1.progressBar_signal.connect(self.progressBar.setValue)
            self.thread1.label_signal.connect(self.label.setText)
            self.thread1.label_2_signal.connect(self.label_2.setText)
            self.thread1.start()
            self.pushButton_2.clicked.connect(self.thread1.click_2)

        def click(self):
            if self.thread1.running is True:
                self.thread1.running = False
                print("Stop")

        def catalog(self, way):
            tree = os.walk(way)
            cat = []
            for d, _, files in tree:
                for f in files:
                    if f.endswith(".jpg") or f.endswith(".png") or
f.endswith(".bmp"):
                        p = (d + "/" + f).replace("\\", "/")
                        cat.append(p)
            return cat

    # ================================================================
    class Thread(QtCore.QThread):
        progressBar_signal = QtCore.pyqtSignal(int)
        label_signal = QtCore.pyqtSignal(str)
        label_2_signal = QtCore.pyqtSignal(str)
        def __init__(self, parent = None):
            super(Thread, self).__init__(parent)
            self.way = parent.way
            self.way_img = parent.way_img
            self.catalog = parent.catalog_i
            self.images = []
```

```
            self.parent = parent

        def click_2(self):
            self.parent.close()
            self.dialog = Result(self)
            self.dialog.show()

        def find(self, way, way_img): # Найти
            p = 70 # Порог
            N = 1000 # Количество ключевых точек
            D, K = 20, 100
            orb = cv2.ORB_create(nfeatures=N)

            img = np.array(Image.open(way_img))
            img1 = cv2.cvtColor(cv2.cvtColor(img, cv2.COLOR_RGB2BGR),
cv2.COLOR_BGR2GRAY)
            _, des1 = orb.detectAndCompute(img1, None)

            kf = 0 # Количество файлов прошло проверку
            for i in self.catalog:
                if self.running is True:
                    kf += 1
                    if i != way_img:
                        img = np.array(Image.open(i))
                        img2 = cv2.cvtColor(cv2.cvtColor(img,
cv2.COLOR_RGB2BGR), cv2.COLOR_BGR2GRAY)
                        _, des2 = orb.detectAndCompute(img2, None)

                        bf = cv2.BFMatcher(cv2.NORM_HAMMING,
crossCheck=True)
                        matches = bf.match(des1, des2)
                        matches = sorted(matches, key=lambda x: x.distance)
                        per = 0
                        for point in matches[:K]:
                            if point.distance < D:
                                per += 1
                        if per > p:
                            self.images.append(i)
                        #print(i, " -> ", per, "% ", ("Совпадение!" if per >
p else ""), EXIF.get_GPS(i), " ", EXIF.get_TIME(i), sep="")

                        self.progressBar_signal.emit(kf)
```

```
                        self.label_signal.emit("Проверяется файл "+ str(kf)
+ "/" + str(len(self.catalog)))
                        self.label_2_signal.emit(i)
                else:
                    self.label_signal.emit("Поиск остановлен")
                    break
            self.parent.pushButton.hide()
            self.parent.pushButton_2.show()
            self.label_2_signal.emit("")
            if len(self.catalog) == kf:
                self.label_signal.emit("Все файлы проверены")

        def run(self):
            self.running = True
            self.find(self.way, self.way_img)

    # ======================================================================
    class Result(QtWidgets.QDialog):
        def __init__(self, parent):
            super().__init__()
            font = QtGui.QFont()
            font.setFamily("Segoe UI")
            font.setPointSize(12)
            font.setBold(False)
            font.setItalic(False)
            font.setWeight(50)

            self.resize(450, 650)
            self.setMinimumSize(QtCore.QSize(450, 650))
            self.setMaximumSize(QtCore.QSize(450, 650))
            self.setSizeIncrement(QtCore.QSize(0, 0))
            self.setFont(font)
            self.setWindowFlags(QtCore.Qt.Window)
            self.setWindowTitle("Find_Dubl")

            self.centralwidget = QtWidgets.QWidget(self)
            self.centralwidget.setObjectName("centralwidget")

            self.pushButton_3 = QtWidgets.QPushButton(self.centralwidget)
            self.pushButton_3.setGeometry(QtCore.QRect(175, 590, 100, 32))
            self.pushButton_3.setFont(font)
            self.pushButton_3.setObjectName("pushButton_3")
            self.pushButton_3.setText("OK")
```

```
            self.pushButton_3.clicked.connect(self.exit)

            self.label_2 = QtWidgets.QLabel(self.centralwidget)
            self.label_2.setGeometry(QtCore.QRect(20, 19, 410, 30))
            font.setPointSize(20)
            self.label_2.setFont(font)
            self.label_2.setAlignment(QtCore.Qt.AlignCenter)
            self.label_2.setObjectName("label_2")
            if len(parent.images) != 0:
                self.label_2.setText("Найдено совпадение!")
            else:
                self.label_2.setText("Совпадений нет")

            self.scrollArea = QtWidgets.QScrollArea(self.centralwidget)
            self.scrollArea.setGeometry(QtCore.QRect(20, 70, 410, 501))
            self.scrollArea.setWidgetResizable(True)
            self.scrollArea.setObjectName("scrollArea")
            self.scrollAreaWidgetContents = QtWidgets.QWidget()
            self.scrollAreaWidgetContents.setGeometry(QtCore.QRect(0, 0,
408, 479))

self.scrollAreaWidgetContents.setObjectName("scrollAreaWidgetContents")
            self.scrollArea.setWidget(self.scrollAreaWidgetContents)

            for i in range(len(parent.images)):
                self.lineEdit_3 =
QtWidgets.QLineEdit(self.scrollAreaWidgetContents)
                font.setPointSize(10)
                self.lineEdit_3.setFont(font)
                font.setPointSize(12)
                self.new_img(parent.images[i], i)

        def exit(self): # Выход (ОК)
            self.destroy()

        def new_img(self, way_img, numb):
            p = 0 if numb == 0 else 10

            self.lineEdit_3.setGeometry(QtCore.QRect(10, p + 10 + numb*232,
390, 30))
            self.lineEdit_3.setText(way_img)

            self.label_5 = QtWidgets.QLabel(self.scrollAreaWidgetContents)
```

```
            self.label_5.setGeometry(QtCore.QRect(10, p + 50 + numb*232,
164, 192))
            self.label_5.setFrameShape(QtWidgets.QFrame.Panel)
            self.label_5.setText("")
            self.label_5.setAlignment(QtCore.Qt.AlignCenter)

            img = QtGui.QPixmap(way_img)
            w, h = self.label_5.width(), self.label_5.height()
            w_img, h_img = img.width(), img.height()
            x = w / w_img if w_img > h_img else h / h_img
            self.label_5.setPixmap(img.scaled(int(w_img*x), int(h_img*x)))

    # ================================================================
    class Def1(QtWidgets.QDialog):
        def __init__(self, parent):
            super().__init__()
            font = QtGui.QFont()
            font.setFamily("Segoe UI")
            font.setPointSize(12)
            font.setBold(False)
            font.setItalic(False)
            font.setWeight(50)

            self.resize(450, 650)
            self.setMinimumSize(QtCore.QSize(450, 650))
            self.setMaximumSize(QtCore.QSize(450, 650))
            self.setSizeIncrement(QtCore.QSize(0, 0))
            self.setFont(font)
            self.setWindowFlags(QtCore.Qt.Window)
            self.setWindowTitle("Find_Dubl")

            self.centralwidget = QtWidgets.QWidget(self)
            self.centralwidget.setObjectName("centralwidget")

    # ================================================================
    class Def2(QtWidgets.QDialog):
        def __init__(self, parent):
            super().__init__()
            font = QtGui.QFont()
            font.setFamily("Segoe UI")
            font.setPointSize(12)
            font.setBold(False)
            font.setItalic(False)
```

```
        font.setWeight(50)

        self.resize(450, 650)
        self.setMinimumSize(QtCore.QSize(450, 650))
        self.setMaximumSize(QtCore.QSize(450, 650))
        self.setSizeIncrement(QtCore.QSize(0, 0))
        self.setFont(font)
        self.setWindowFlags(QtCore.Qt.Window)
        self.setWindowTitle("Find_Dubl")

        self.centralwidget = QtWidgets.QWidget(self)
        self.centralwidget.setObjectName("centralwidget")

# ======================================================================
if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```