|  |  |
| --- | --- |
| C:\Documents and Settings\admin\Рабочий стол\МИРЭА_ЭМБЛЕМА_приказ.JPG | |
| МИНОБРАНАУКИ РОССИИ | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  «**РОССИЙСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»**  **МИРЭА** | |
| Институт информационных технологий  Кафедра прикладной математики | |
| **Отчет по лабораторной работе №7**  **Тема работы: “GUI, классы, модуль Tkinter”** | |
| Выполнил: |  |
| Студент группы ИВБО-12-17  Отчет принял:  Преподаватель | Кузнецов Н.А.  Митина Ольга Алексеевна |

Москва 2018

**Цель работы:** Разработать программу, содержащую меню и ввод-вывод данных в окна на экране. Необходимо предусмотреть контроль ошибок пользователя при вводе данных.

При разработке программы применить технологию объктно-ориентированного программирования и минимизировать использование глобальных переменных.

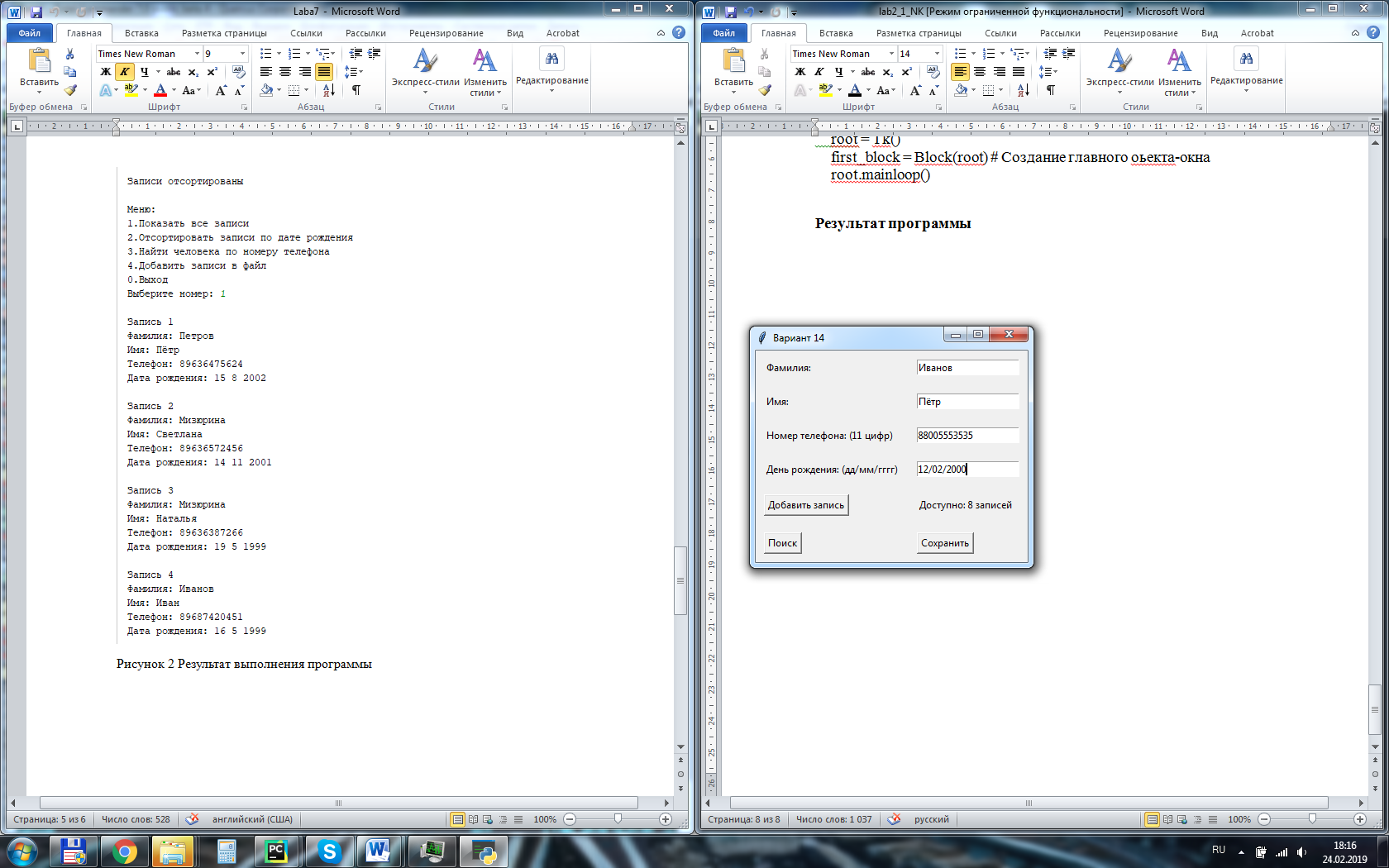
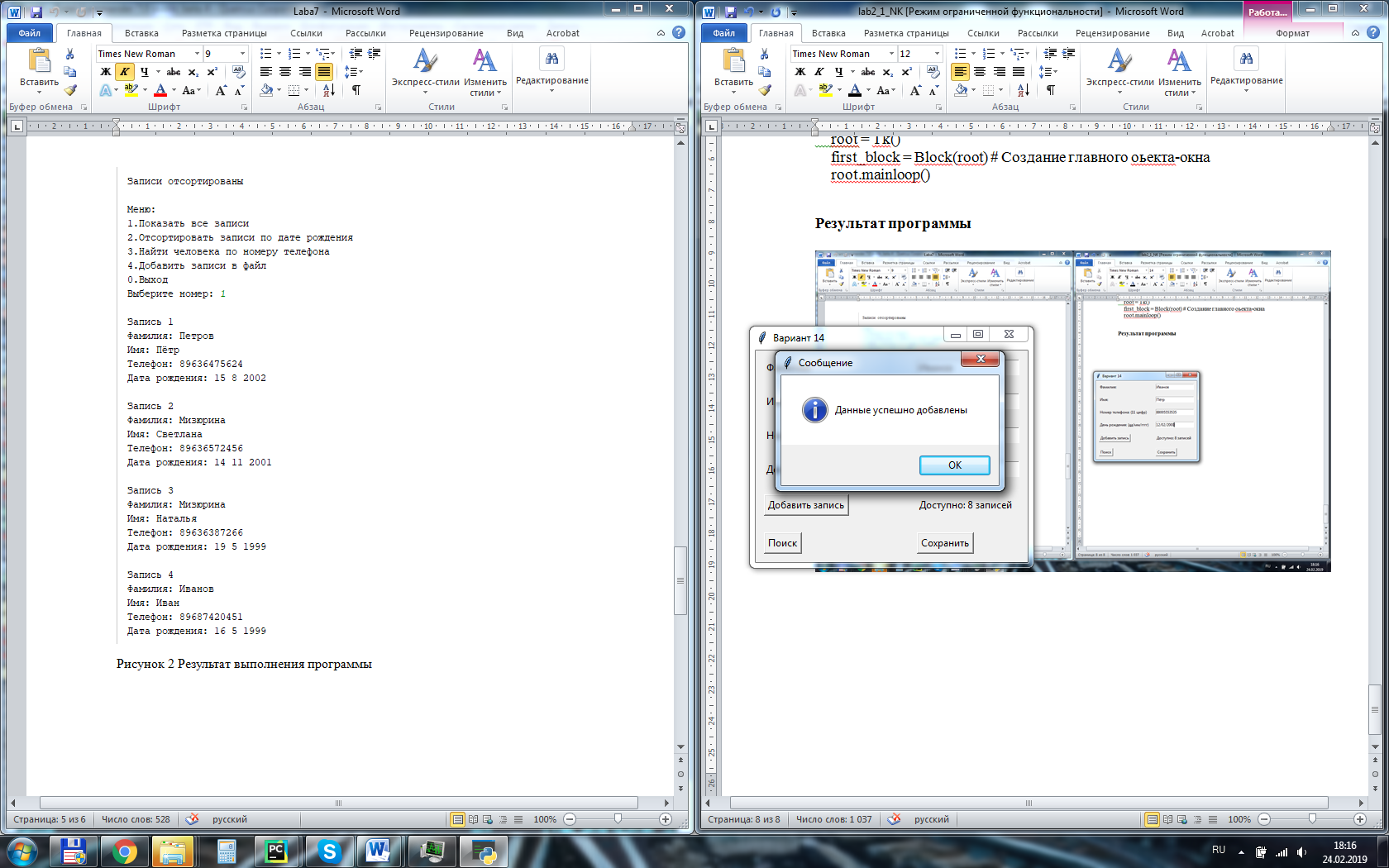
**Задание 1:** Описать запись с именем Note, содержащую следующие поля: фамилия, имя; номер телефона; день рождения (массив из 3х чисел).

**Задание 2:**  Написать программу, выполняющую следующие действия: ввод данных с клавиатуры в массив, состоящий из восьми элементов типа Note; записи упорядочить по первым трём цифрам номера телефона; вывод на экран информации о человеке, чья фамилия была введена с клавиатуры, если такой нет, то вывести соответствующее сообщение; Запись массива в файл под заданным с клавиатуры именем.

**Код программы:**

import datetime  
import tkinter as tk  
from tkinter import ttk  
from tkinter import messagebox  
from tkinter import \*  
  
# Класс из задания 1  
class Note:  
 name = 'Имя'  
 surname = 'Фамилия'  
 phone = '88005553535'  
 birthday = ['', '', '']  
  
 # Конструтор  
 def \_\_init\_\_(self, n='', sn='', p='', bd=''):  
 self.name = n  
 self.surname = sn  
 self.phone = p  
 self.birthday = bd  
  
 # Метод используется при добавлении данных в обьект после ввода  
 def appendNote(self, name, surname, phone, bd):  
 self.name = name  
 self.surname = surname  
 self.phone = phone  
 self.birthday = bd  
  
 # Статический метод (чтобы вызывать без обьекта) для сортировки обьектов по их номеру телефона через функцию  
 @staticmethod  
 def sortPhone(noteList):  
 return sorted(noteList, key=byPhone\_key)  
  
# Возвращает первые три числа из номера телефона  
def byPhone\_key(note):  
 return note.phone[:3]  
  
  
# Главное окно  
class Block(tk.Frame):  
 counts = 0 # Счётчик уже введённых обьетков  
 labelCountText = ''  
  
 # Через конструктор выводяться все элементы окна  
 def \_\_init\_\_(self, master):  
 master.title("Вариант 14")  
  
 # Создание списка из ровно 8 элементов класса Note  
 self.noteList = []  
 for i in range(8):  
 self.noteList.append(Note())  
  
 # Для вывода изменяемого текста  
 self.labelCountText = StringVar()  
 self.labelCountText.set("Доступно: 8 записей")  
  
 # Поле для воода фамилии  
 Label(text="Фамилия:").grid(row=0, column=0, sticky=W, pady=10, padx=10)  
 self.note\_surname = Entry()  
 self.note\_surname.grid(row=0, column=1, columnspan=2, sticky=W + E, padx=10)  
  
 # Поле для ввода имени  
 Label(text="Имя:").grid(row=1, column=0, columnspan=2, sticky=W, padx=10, pady=10)  
 self.note\_name = Entry()  
 self.note\_name.grid(row=1, column=1, sticky=W + E, padx=10)  
  
 # Поле для ввода номера телефона  
 Label(text="Номер телефона: (11 цифр)").grid(row=2, column=0, columnspan=2, sticky=W, pady=10, padx=10)  
 self.note\_phone = Entry()  
 self.note\_phone.grid(row=2, column=1, sticky=W + E, padx=10)  
  
 # Поле для ввода дня рождения  
 Label(text="День рождения: (дд/мм/гггг)").grid(row=3, column=0, sticky=W + E, pady=10, padx=10)  
 self.note\_bd\_sting = Entry()  
 self.note\_bd\_sting.grid(row=3, column=1, padx=1, pady=1)  
  
 # Кнопка для добаления данных  
 Button(text="Добавить запись", command=self.inputCheck).grid(row=4, sticky=W, column=0, pady=10, padx=10)  
 self.labelCounts = Label(master, textvariable=self.labelCountText).grid(row=4, column=1, sticky=W, pady=10, padx=10)  
  
 # Кнопки поиска по фамилии и сохранения  
 Button(text="Поиск", command=AddPurchase).grid(row=5, sticky=W, column=0, pady=10, padx=10)  
 Button(text="Сохранить", command=SaveNote).grid(row=5, sticky=W, column=1, pady=10, padx=10)  
  
 # Метод для подсчёта сколько ввели записей и сколько осталось  
 def countNum(self):  
 if self.counts < 8:  
 self.labelCountText.set("Доступно: " + str(8 - self.counts) + " записей")  
 else:  
 # Всплывающее сообщение  
 messagebox.showinfo("Предупреждение", "Список полностью заполнен")  
  
 # Вывод текста в поле счётчика  
 self.labelCountText.set("Список полностью заполнен")  
  
 # После окончания ввода все записи сортируются по номеру телефона (выдно после сохранения)  
 self.noteList = Note.sortPhone(self.noteList)  
  
 # Проверка ввода даты рождения  
 def checkBd(self, data):  
 bd = data.split('/')  
  
 if len(bd) == 3:  
 # Проверка на то, что дата из чисел  
 try:  
 for i in range(len(bd)):  
 bd[i] = int(bd[i])  
 except ValueError:  
 return False  
  
 # Проверка на то, что дата существует  
 try:  
 datetime.date(bd[2], bd[1], bd[0])  
 return True  
 except (TypeError, ValueError):  
 return False  
 else:  
 return False  
  
 # Проверка ввода в поля  
 def inputCheck(self):  
 err = 0  
 data = {} # Словарь для врменного хранения. Очищается после каждого ввода  
 data['surname'] = self.note\_surname.get()  
 data['name'] = self.note\_name.get()  
 data['phone'] = self.note\_phone.get()  
 data['bd'] = self.note\_bd\_sting.get()  
  
 # Проверка на пустое поле  
 for i in 'surname', 'name', 'phone', 'bd':  
 if len(data[i]) == 0:  
 err = 1  
  
 if err == 1:  
 messagebox.showinfo("Ошибка", "Пожалуйста, заполните все поля")  
 elif not self.checkBd(data['bd']):  
 messagebox.showinfo("Ошибка", "Неправильно введена дата")  
 elif not((data['name']).isalpha() and (data['surname']).isalpha()):  
 # Состоит ли строка только из букв без пробелов  
 messagebox.showinfo("Ошибка", "Неправильно введены фамилия или имя")  
 elif not((data['phone']).isdigit() and len(data['phone']) == 11):  
 messagebox.showinfo("Ошибка", "Неправильно введён номер телефона")  
 elif self.counts < 8:  
 # К обьекту из списка (по индексу) применяется метод для добавления данных  
 self.noteList[self.counts].appendNote(data['name'], data['surname'], data['phone'], data['bd'].split('/'))  
 self.counts += 1  
 messagebox.showinfo("Сообщение", "Данные успешно добавлены")  
 self.countNum() # Изменить зн-е счётчика  
 else:  
 messagebox.showinfo("Ошибка", "Список полностью заполнен! Выберете другое действие")  
  
  
# Окно для поиска по фамилии. Удаляется после выполнения  
class AddPurchase(tk.Toplevel):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_(root)  
 self.init\_child()  
  
 # Рисование кнопок и полей  
 def init\_child(self):  
 self.title("Поиск")  
  
 label\_lname = ttk.Label(self, text="Фамилия:")  
 label\_lname.grid(row=0, column=0, sticky=W, pady=10, padx=10)  
  
 self.entry\_lname = ttk.Entry(self)  
 self.entry\_lname.grid(row=0, column=1, sticky=W+E, pady=10, padx=10)  
  
 btn\_add = ttk.Button(self, text="Найти", command=self.searchSurname)  
 btn\_add.grid(row=1, column=0, sticky=W+E, pady=10, padx=10)  
  
 # метод поиска по фамилии  
 def searchSurname(self):  
 dataSearch = {}  
 dataSearch['surname'] = self.entry\_lname.get()  
 k = 0  
  
 # Проверка на пустоту поля  
 if len(dataSearch['surname']) == 0:  
 pass # Ничего не делать  
 elif not (dataSearch['surname'].isalpha()):  
 messagebox.showinfo("Ошибка", "Неправильно введены данные")  
 else:  
 for i in range(8):  
 if first\_block.noteList[i].surname == dataSearch['surname']:  
 # messagebox.showinfo("Сообщение", "Пользователь найден")  
 message = ("Пользователь найден \n" +  
 "Фамилия: " + first\_block.noteList[i].surname + "\n" +  
 "Имя: " + first\_block.noteList[i].name + "\n" +  
 "Номер телефона: " + first\_block.noteList[i].phone + "\n" +  
 "День рождения: " + '/'.join(first\_block.noteList[i].birthday))  
 messagebox.showinfo("Сообщение", message)  
 k = 1  
 self.destroy() # Удаляет окно  
 if k == 0:  
 messagebox.showinfo("Ошибка", "Пользователь не найден")  
 self.destroy()  
  
  
# Окно для созрания данных  
class SaveNote(tk.Toplevel):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_(root)  
 self.init\_child()  
  
 def init\_child(self):  
 self.title("Сохранение")  
  
 title\_lname = ttk.Label(self, text="Для сохранения файла введите название. Внимание, если файл существует, он будет перезаписан")  
 title\_lname.grid(row=0, column=0, sticky=W, pady=10, padx=10, columnspan=2)  
  
 label\_lname = ttk.Label(self, text="Название файла:")  
 label\_lname.grid(row=1, column=0, sticky=W, pady=10, padx=10)  
  
 self.entry\_lname = ttk.Entry(self)  
 self.entry\_lname.grid(row=1, column=1, sticky=W + E, pady=10, padx=10)  
  
 btn\_add = ttk.Button(self, text="Сохранить", command=self.saveFile)  
 btn\_add.grid(row=2, column=0, sticky=W + E, pady=10, padx=10)  
  
 # Метод сохранения данных  
 def saveFile(self):  
 dataSave = {}  
 dataSave['name'] = self.entry\_lname.get()  
 k = 0  
 if len(dataSave['name']) == 0:  
 pass  
 else:  
 f = open(dataSave['name'] + ".txt", 'w') # Открывает файл с заданным именем на запись  
 # Перебор всех обьектов из списка и запись данных из их полей  
 for obj in first\_block.noteList:  
 message = ("Фамилия: " + obj.surname + "\n" +  
 "Имя: " + obj.name + "\n" +  
 "Номер телефона: " + obj.phone + "\n" +  
 "День рождения: " + '/'.join(obj.birthday) + "\n")  
 f.write(message + '\n')  
 f.close()  
 messagebox.showinfo("Сообщение", "Запись завершена")  
 self.destroy()  
  
  
# Определение начала выполнения  
if \_\_name\_\_ == '\_\_main\_\_':  
 root = Tk()  
 first\_block = Block(root) # Создание главного оьекта-окна  
 root.mainloop()

**Результат** **программы**



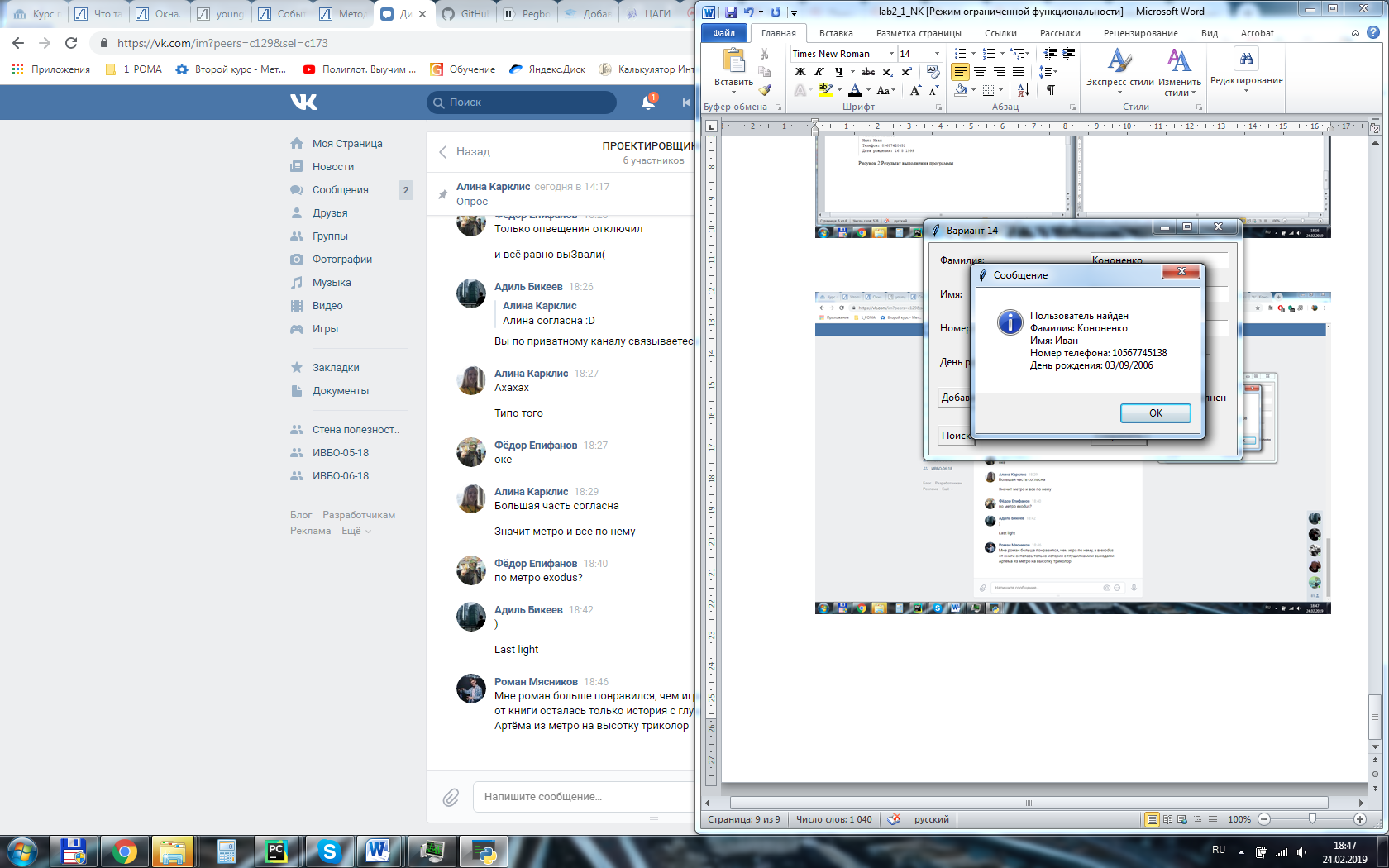
Рисунок 1 Результат выполнения программы

Рисунок 2 Результат выполнения программы

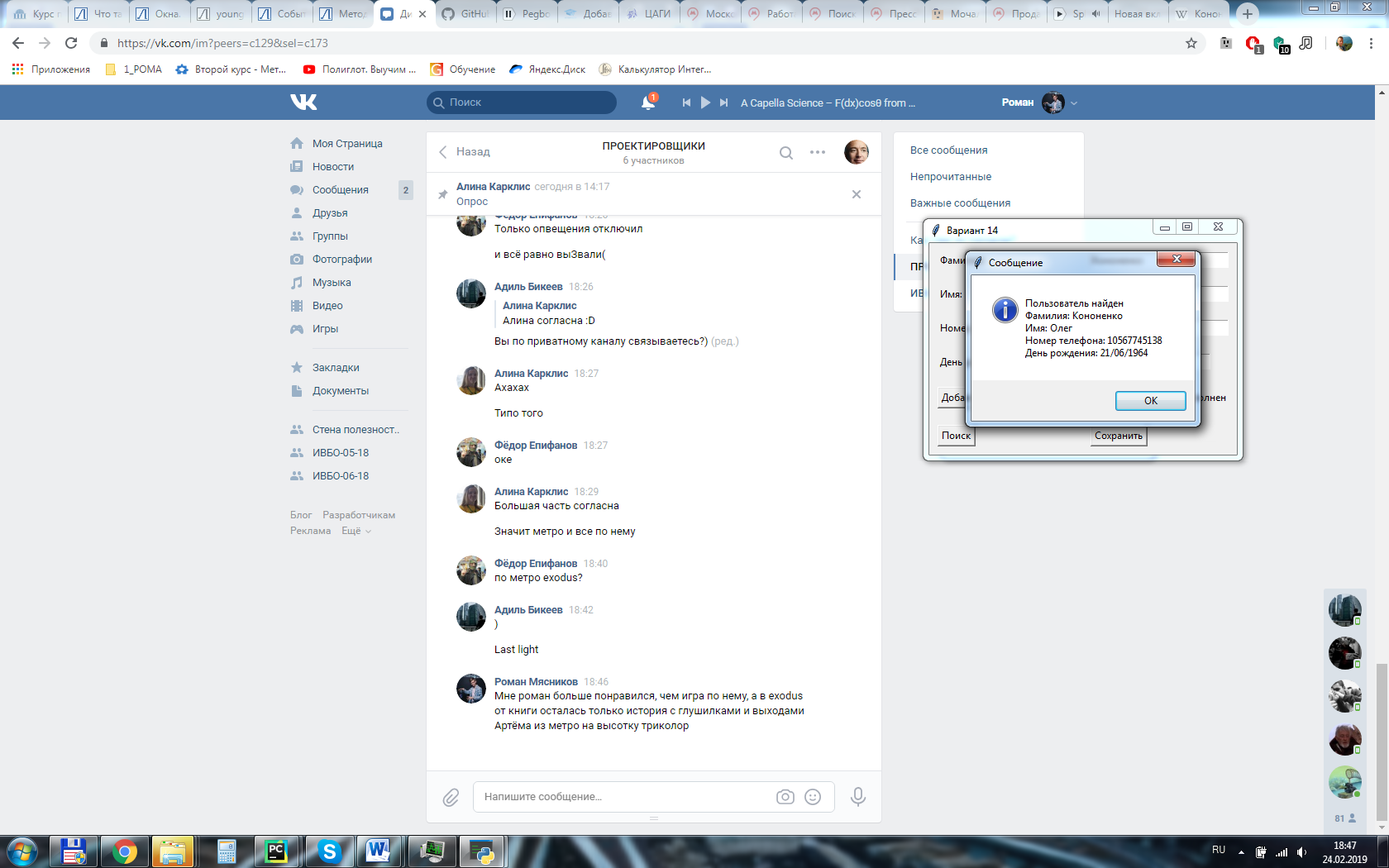


Рисунок 3 Результат выполнения программы

Рисунок 4 Результат выполнения программы

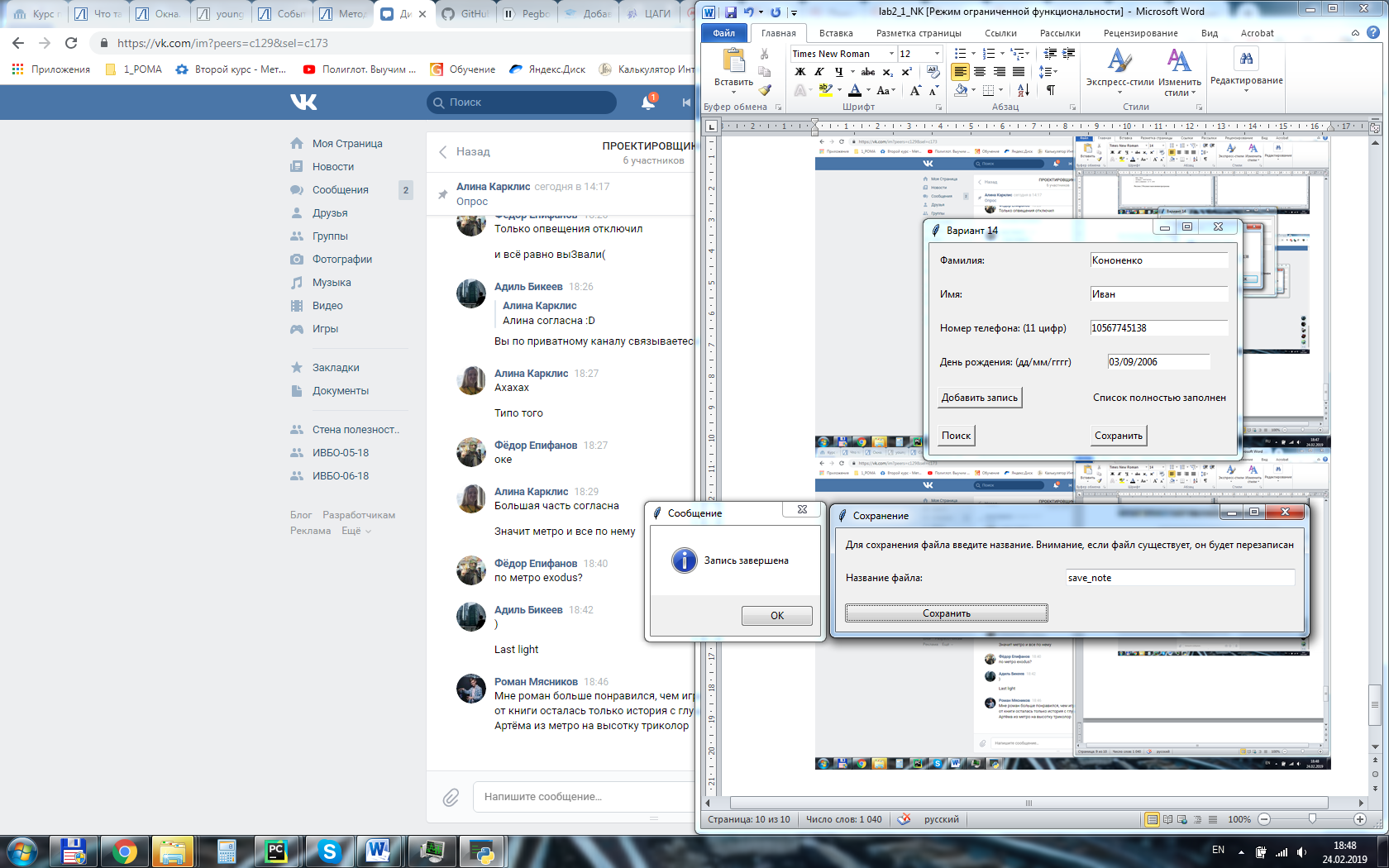


Рисунок 5 Результат выполнения программы

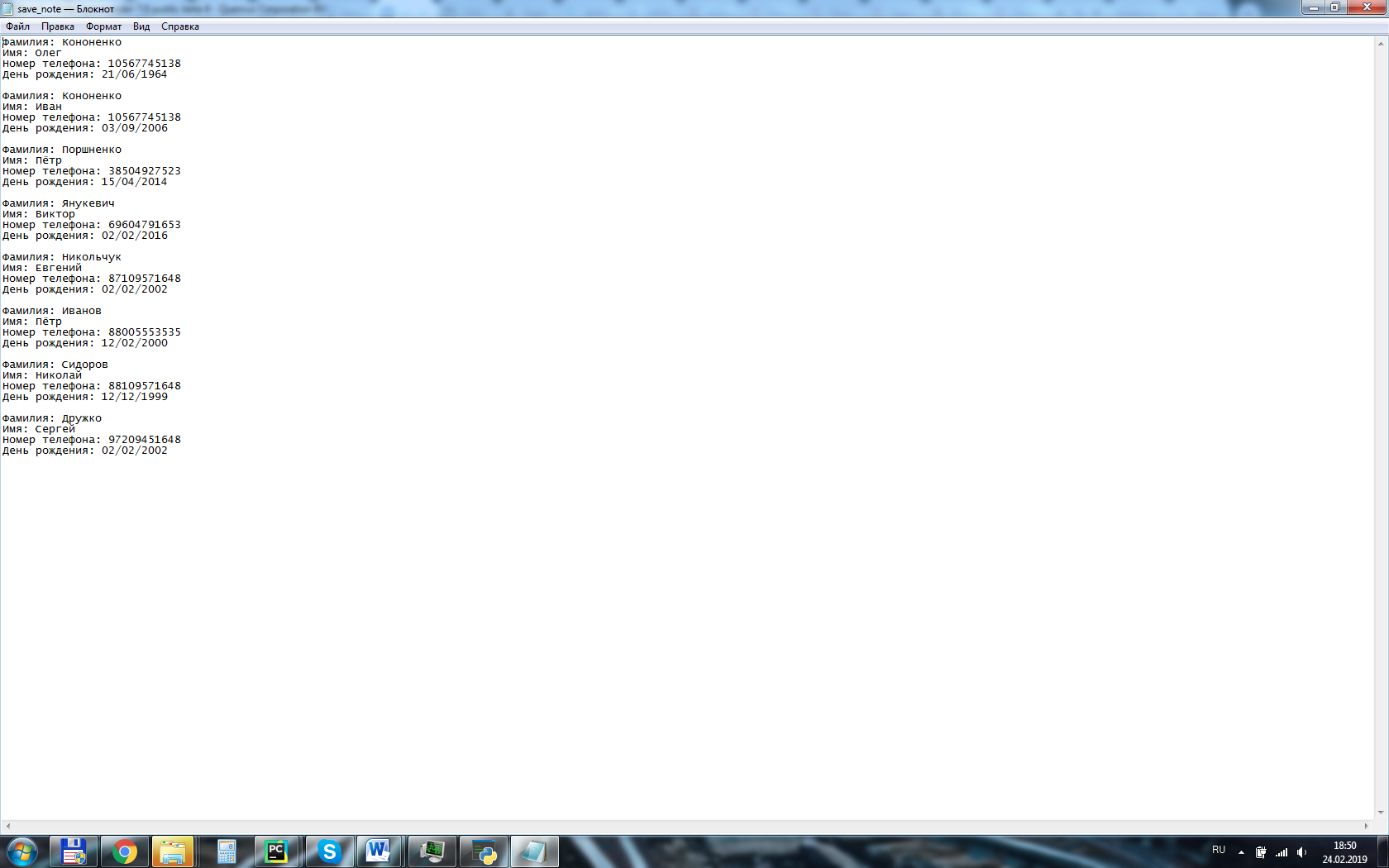


Рисунок 3 Результат выполнения программы