Лабораторная работа №7: «GUI, классы, модуль Tkinter»

Цель работы:

Дать студентам практический навык в написании программ, использующих графический интерфейс пользователя. Ознакомить с такими понятиями, как класс, виджеты, массивы записей, а также с атрибутами класса и методами передачи данных между классами.

Можно не ограничиваться выбором модуля и способов решения заданий этой лабораторной работы. В качестве средства для написания можно использовать и другие модули и библиотеки.

Постановка задачи

1. Описать запись с именем Zodiac, содержащую следующие поля:
    - фамилия, имя;
    - знак Зодиака;
    - день рождения (массив из трех чисел).
2. Написать программу, выполняющую следующие действия:
    - ввод данных с клавиатуры в массив, состоящий из восьми элементов типа Zodiac; записи должны быть упорядочены по датам дней рождения.
    - вывод на экран информации о человеке, чья фамилия введена с клавиатуры; если такого нет, вывести на экран соответствующее сообщение;
    - запись массива в файл под заданным с клавиатуры именем.

Теоретическое введение

Для выполнения такой работы нам необходимы следующие знания:

- классы, атрибуты классов (свойства, методы), организация передачи данных; типы данных - записи, кортежи, словари, массивы записей;
- форматы фалов, формат CSV; работа с файлами: чтение, запись;
- виджеты модуля Tkinter - конфигурирование и создание методов.

Начнем с формата CSV.

CSV - формат

Формат текстового файла, который используется для представления табличных данных, и в котором в качестве разделителя используется запятая, принято называть CSV - форматом (Coma-Separated Values).

Следующая таблица будет далее использована для демонстрации того, как оформляется CSV файл. В ней указаны параметры издания: автор, наименование, год издания количество экземпляров и примечание, поясняющее содержание издания.

Автор (author)	Наименование (title)	Год (year)	Экз-ов (copy)	Примечание (note)

Доля П.Г.	Введение в научный Python	2016	2	Классы, оконные приложения
Travis E.	Guide to NumPy		5	Руководство
МусинД.	Самоучитель Pvthon	2016	3	Язык Pvthon

В существующей спецификации рассматриваются следующие моменты:

- Каждая строка файла - это одна строка таблицы. Строка заканчивается символом CRLF. Последняя строка может не содержать символа CRLF;
- Разделителем (delimiter) колонок является запятая;
- Значения, содержащие зарезервированные символы, обрамляются двойными кавычками. Если в самом значении встречается двойная кавычка, то эта кавычка дублируется.
- Допускается наличие строки заголовка. Формат строки такой же, как обыч ные строки файла. Этот заголовок может содержать имена, соответствующие полям в файле и должен содержать такое же количество полей.

Часто в качестве разделителя используются и другие символы, например, знак табуляции (формат TSV), точка с запятой. Одно из объяснений такого нарушения стандарта связано с тем обстоятельством, что в русской локации запятая используется как десятичный разделитель.

Существует более общий формат файла - DSV (Delimiter-Separated Values - значения, разделённые разделителем). В этом формате допускаются любые символы в качестве разделителя. Вместе с тем разработчики программного обеспечения не сильно озабочены и часто путают форматы, не смотря на наличие документов, например, RFC-4180.

Текстовый файл для приведенной выше таблицы, может иметь следующий вид:

author, title, year, copy, note

Доля П.Г., Введение в научный Python, 2016, 2,"Классы, оконные приложения"

Travis E., Guide to NumPy, , 5, Руководство

Мусин Д., Самоучитель Python, 2016, 3, Язык Python

Обратите внимание на то, что в первой строке имена полей начинаются со строчной буквы. Начинать имена полей с заглавной буквы не следует, поскольку при чтении заголовок (первая строка) помещается в форматную строку с преобразованием к строчным буквам. Так что при последующем обращении через имя поля, это имя следует вводить со строчной буквы.

В следующей строке последняя запись обрамлена двойными кавычками, поскольку в тексте используется запятая.

В третьей строке отсутствует информация о годе издания, и эта позиция отделяется запятой - остается пустое поле. При чтении такое поле заменяется значением -1.

В Python имеется модуль CSV, функции и методы которого позволяют читать файлы в формате CSV и записывать в них данные. Кроме этого имеются и другие модули, например, pylab, pandas. С сожалением следует отметить, что реализация записи в формате CSV в некоторых модулях для Python 3 вызывает ошибку. Эта ошибка вызвана ошибками в самих модулях. Для чтения и записи наши х данных мы воспользуемся функциями модуля **pandas**: read_csv() и to _csv(). Эти функции позволяют прочитать файл формата CSV и сохранить данные в этом же формате.

Pandas считается одной из наиболее динамично развивающихся библиотек для анализа данных на Python.

Запись и массив записей

Кроме стандартных типов данных существуют типы данных, которые пользователь может конструировать самостоятельно. Одним из таких типов является запись (`record` - Паскаль) или структура (`struct` - Си).

Запись представляет собой логическое объединение данных разного типа. Пример структуры данных такого типа представлен в таблице выше. В одной строке, в связке, находятся как данные строкового типа, так и числового.

Поскольку в нашей задаче необходимо обрабатывать строковые поля (фамилия, название подразделения), поле с датой (дата рождения), числовое поле (зарплата), а таких данных достаточно много (массив), нам надо научиться работать с записями. В Python эта работа реализована в модуле NumPy. Методы и функции этого модуля позволяют создавать массивы и форматировать структуру элемента массива под нужды пользователя.

Для создания массива записей необходимо сформировать формат такой записи, указав название поля и его тип. Так, например, для примера с таблицей можно

подготовить следующий формат:

```
dt = np.dtype([('author', 'S15'), ('title', 'S30'),
               ('year', 'int16'), ('copy', 'int16'),
               ('note', 'S3')])
```

Обратите внимание, что формат состоит из списка кортежей, а в каждом кортеже задано имя поля и его размер.

Для формирования массива можно использовать функцию empty() модуля NumPy, которой в качестве параметра можно передать количество элементов массива и тип элемента:

```
mas = np.empty(20, dtype = dt)
```

С записью в таком массиве можно работать как через имя поля, так и через номер элемента в массиве:

```
print(mas[n-1])
```

Этот пример напечатает значения всех полей n-1-ой записи в массиве. Вывод записей можно форматировать. Например, так:

for row in mas:

```
print('{ : ^15} {:<30} {:<5} {:<30}'.format(*row))
```

Тут, в цикле, выводятся значения полей всех записей. Символ "<" означает, что при выводе значение будет выравниваться по левому краю поля, а "^" - по центру поля. Форматный вывод позволяет представлять результат в аккуратной форме, что улучшает его восприятие.

В следующем листинге представлен текст программы, которая читает текстовый файл в формате CSV, выводит тестовые сообщения и сохраняет информацию в новом файле.

Модуль Tkinter

В Python реализовано достаточно много модулей и библиотек, позволяющих создавать графический интерфейс пользователя (GUI). Вот небольшой перечень: Tkinter, PyQt5, PyGTK, wxPython, Pythonwin, и др.

Стандартным модулем, на котором, кстати, разработана IDLE, является модуль Tkinter. Этот модуль устанавливается вместе с Python. Сравнивая разработку, например, на PyQt и Tkinter можно сделать вывод, что Tkinter не является универсальной библиотекой. Его следует использовать только в программах, где требуются лишь основные компоненты, где накладывается мало ограничений на ввод пользователя. Другими словами, Tkinter подходит для простых программ с графическим интерфейсом и т.к. имеет более понятный и ясный синтаксис подходит для обучения.

Далее будем рассматривать решение нашей задачи, поставленной в начале этой лабораторной работы, в рамках возможностей модуля Tkinter.

Первый шаг - это структура программы: организация программы, управляемой событиями.

Событийно-управляемое программирование

Под событием в операционной системе понимается любое действие пользователя при его взаимодействии с программным интерфейсом: нажатие клавиш клавиатуры, щелчки кнопками мыши, перемещение мыши. Кроме этих событий имеются события, вызываемые как самой операционной системой, так и события, вызываемые другими приложениями: работа системного таймера, информационный обмен между процессами, например, получение сообщения от сервера и т.д.

Событие воспринимается операционной системой и преобразуется в сообщение - запись, содержащую необходимую информацию о событии. Например, это может быть код клавиши, была нажата или отпущена клавиша, клик мыши и координаты маркера мыши в момент клика, ...

Сообщения поступают в общую очередь, откуда распределяются по очередям приложений. У каждого приложения имеется цикл обработки сообщений, который выбирает сообщение из своей очереди и через операционную систему вызывает подпрограмму, предназначенную для обработки этого сообщения.

Можно сделать следующее заключение - программа состоит из главной части, которая выполняет инициализацию и завершение приложения, цикла обработки сообщений, и набора обработчиков событий (подпрограмм).

Следующий шаг - виджеты модуля Tkinter.

Виджеты

Виджет (англ. widget) - **элемент интерфейса** - примитив графического интерфейса пользователя (GUI), имеющий стандартный внешний вид и выполняющий стандартные действия. Иногда можно встретить и другое название контрол (англ. control). Вот неполный перечень виджетов Tkinter:

- меню (Menu) - используется для организации меню, в том числе и каскадного;
- кнопка (Button) - используется для запуска команды, или, например, для выбора следующего действия. Так, кнопка "Сохранить" запускает процесс сохранения текущих данных, а кнопка "Удалить" удаляет выбранный элемент;
- радиокнопка (Radiobutton) - используется в группе. Позволяет выбрать только одно значение из нескольких;
- флажок (Checkbutton) - позволяет выбрать несколько не взаимоисключающих значений;
- список (Listbox) - этот элемент пре дставляет собой перечень элементов. Настройка конфигурации позволяет пользователю выбрать либо только один элемент, либо несколько.
- многострочное текстовое поле (Text) - этот виджет позволяет вывести или получить от пользователя многострочное текстовое сообщение;
- однострочное текстовое поле (Entry) - этот виджет предназначен для вывода или ввода только одной, как правило, небольшой строки, например, название предмета или фамилию;
- метка (Label) - используется, как правило, для информирования пользователя. Например, в качестве метки выступает надпись, информирующая пользователя о назначении однострочного текстового поля;
- раскрывающийся список (Combobox) - этот виджет используется для выбора одного из элементов, хранящихся в его списке;
- счетчик (Spinbox) - такой виджет используется, например, в качестве счетчика записей, выводимых на экран;
- полоса прокрутки (Scrollbar) - используется для просмотра части текста или списка, которые выходят за рамки, наприм ер, текстового поля;
- ползунок (Slider) – позволяет выбирать значения из набора, задаваемого программистом. Может быть использован, например, как регулятор громкости;
- окна сообщений (message box) - это диалоговые окна, которые позволяют выводить сообщения, предупреждения или ошибки при взаимодействии с пользователем. Кроме этого имеются диалоговые окна для работы с файлами: окна выбора файла при открытии или сохранении файла

Все виджеты имеют два набора конфигурационных параметров. Один набор параметров является общим для всех виджетов: размеры, положение на форме.

Второй набор параметров определяется индиви дуальными характеристикам и виджета: состояние счетчика виджета Spinbox, или индекс значения, выбранного пользователем в виджете Combobox.

События, связанные с виджетом (клик мыши, нажатие клавиш клавиатуры) могут быть привязаны к методу, который будет обрабатывать событие. Например, при вводе текста в однострочном роле (Entry) и нажатии клавиши Enter (используется для завершения ввода) может быть вызван метод, который прочитает введенный текст.

Важно отметить, что все виджеты будут показаны на форме только после их обработки упаковщиком.

Существуют три метода упаковки виджетов:

Метод pack

В этом методе указывается, как виджет будет заполнять пространство формы, будет ли он растягиваться при изменении размеров формы, как виджет будет размещаться на форме.

Метод place

Этот метод позволяет программисту контролировать положение виджета через задание координат, и его размеры через задание ширины и высоты виджета. Координаты положения задаются относительно верхнего левого угла окна, к которому привязан виджет. При этом, если окно является вложенным и его координаты меняются, то относительные координаты виджета не меняются.

Метод grid

Наиболее часто используемый упаковщик. В этом методе окно делится условной сеткой на ячейки (grid). Программист задает ячейку, в которой будет размещаться виджет, через номер столбца и колонки. При этом можно указать, сколько строк и столбцов будет занимать виджет.

Последний шаг - это представление о том, как можно писать программу с GUI используя такие понятия, как объектно-ориентированное программирование, класс, объект.

Классы

Свойства класса - это набор параметров (значений), которыми оперирует класс: цвет, вес, скорость, ...

Методы класса - это набор функций, реализующих поведение класса. Метод - это то, что класс делает. Например, методы, которые рисуют объект или форму, методы, которые реализует перемещение объекта или контролируют заполнение полей формы. Все методы класса доступны в другом классе. При вызове следует указать имя класса и вызываемый метод, через точку: <имя_класса>.<метод>.

Класс - это структура, объединяющая свойства и методы, а объект - это реализация класса. Класс может порождать объекты с различными свойствами.

Понятие "обмен сообщениями" - это процесс обмена данными между классами. Слово "сообщение" тут используется потому, что в процессе обмена передаются не только данные, но и команды. Информационный поток между классами может быть достаточно

сложным. Как правило, он подчиняется некоторым, разработанным программистом, правилам и подразумевает не только передачу сообщений, но и получение квитанций (ответов).

Дочернее окно

В программах с GUI возникает необходимость создавать не одно окно, а несколько. Первое окно - главное, а последующие окна - дочерние (верхнего уровня - top level).

Создание дочернего окна можно проследить на следующем коде:

```
# -*- coding: utf-8 -*-
# импорт модулей python
from tkinter import *
#создание главного окна
root = Tk() root.title('parent')
root.geometry('200x150+200+150')
#создание дочернего окна
child = Toplevel(root) child.title('child')
child.geometry('200x150+400+300')
# запуск окна
root.mainloop()
```

Toplevel() - это класс Tkinter, с помощью которого можно создавать любые окна, кроме главного. Т.о. переменная `child` - это объект, созданный на основе класса Toplevel() и привязанный к главному окну `root`. При закрытии главного окна закрывается и дочернее окно, но не наоборот.

Работа с классами

Python - это объектно-ориентированный язык программирования. Tk() и Toplevel() являются классами модуля Tkinter, принимающими форму объектов для создания на экране графических окон. Программирование на Tkinter подразумевает комбинирование и преобразование встроенных классов Tkinter в новые классы с индивидуальными свойствами и методами.

Обратим внимание на следующие моменты:

- наличие метода __init__(self, master) является обязательным. Этот метод отрабатывается всегда, когда выполняется создание объекта на основе класса. С помощью этого метода объект получает индивидуальные особенности;
- self - это параметр (метка экземпляра), необходимый для привязки переменных класса к данному объекту. Вместо self можно писать имя класса, которому принадлежит это свойство или метод;
- master - это параметр, посредством которого, при вызове класса, значение передается в метод. Так, например, вызов `main(root)` создает объект, в котором запись:

```
self.master.title('myWindow')
```

переводится в

```
self.root.title('myWindow')
```

Задавая различные параметры, в методе инициализации, можно создавать окна с различными характеристиками: цвет, размер, положение на экране монитора и т.д.;

Модальное дочернее окно

В программах часто возникает необходимость создать такое дочернее окно, которое захватывает фокус на себя. При этом дочернее окно перехватывает все события и организует цикл ожидания на закрытие. Цикл ожидания закрытия дочернего окна не влияет на работу основного цикла. Такое дочернее окно называют модальным.

Основная цель организации модального окна состоит в том, чтобы пользователь выполнил все работы в этом окне и мог продолжить работу только после его закрытия.

Например, при вводе данных о работнике пользователь должен ввести некоторый контролируемый набор данных (заполнить важные поля), прежде, чем данные будут зарегистрированы в базе данных.

Дочернее окно можно превратить в модальное окно с помощью трех методов класса `Toplevel()`:

`self.slave.grab.set()`	`child` перехватывает все события, происходящие в приложении;
`self.slave.focusset()`	`child` захватывает фокус;
`self.slave.wait_window()`	`child` ждет, когда будет уничтожен текущий объект, не возобновляя работы (но и не оказывая влияния на основной цикл).

Важной составляющей метода wait_window() является то, что этот метод перехватывает событие закрытия и передает управление следующей за ним команде. Этой следующей командой может быть команда, которая передаст сообщение от дочернего окна к родительскому окну.

Листинг программы

```
# -*- coding: UTF-8 -*-
from tkinter import *



class Zodiac:

    def __init__(self, name, surname, zodiac, dob):
        self.name = name
        self.surname = surname
```

```
        self.zodiac = zodiac
        self.dob = dob.split('.')


class Application(Frame):
    n = -1

    def __init__(self, master):
        super(Application, self).__init__(master)
        self.grid()

        self.name_label = Label(text="Введите имя:")
        self.surname_label = Label(text="Введите фамилию:")
        self.zodiac_label = Label(text="Введите зодиак:")
        self.dob_lable = Label(text="Введите дату рождения\
дд.мм.гггг:")

        self.name = StringVar()
        self.surname = StringVar()
        self.zodiac = StringVar()
        self.dob = StringVar()
        self.nameoutp = StringVar()
        self.filename = StringVar()

        self.name_outp_lable = Label(text="Введите имя:")
        self.name_outp_entry = Entry(textvariable=self.nameoutp)
        self.name_outp_button = Button(text="Output",
command=self.outp)
        self.info_outp_lable = Label(text="")

        self.blank_lable = Label(text="")
        self.file_entry = Entry(textvariable=self.filename)
        self.save_button = Button(text="Save",
command=self.save)

        self.name_entry = Entry(textvariable=self.name)
```

```
        self.surname_entry = Entry(textvariable=self.surname)
        self.zodiac_entry = Entry(textvariable=self.zodiac)
        self.dob_entry = Entry(textvariable=self.dob)
        self.input_button = Button(text="Input",\
command=self.increase)

        self.increase()

    def increase(self):

        global zod

        if self.n == -1:
            self.name_label.grid(row=0, column=0, sticky="w")
            self.surname_label.grid(row=0, column=1, sticky="w")
            self.zodiac_label.grid(row=0, column=2, sticky="w")
            self.dob_lable.grid(row=0, column=3, sticky="w")

            self.name_entry.grid(row=1, column=0, padx=5,
pady=5)
            self.surname_entry.grid(row=1, column=1, padx=5,
pady=5)
            self.zodiac_entry.grid(row=1, column=2, padx=5,
pady=5)
            self.dob_entry.grid(row=1, column=3, padx=5, pady=5)

            self.input_button.grid(row=2, column=3, padx=5,
pady=5,\ sticky="e")

        elif self.n < 7:
            self.name_label.grid(row=0, column=0, sticky="w")
            self.surname_label.grid(row=0, column=1, sticky="w")
            self.zodiac_label.grid(row=0, column=2, sticky="w")
            self.dob_lable.grid(row=0, column=3, sticky="w")

            self.name_entry.grid(row=1, column=0, padx=5,
pady=5)
```

```
            self.surname_entry.grid(row=1, column=1, padx=5,
pady=5)
            self.zodiac_entry.grid(row=1, column=2, padx=5,
pady=5)
            self.dob_entry.grid(row=1, column=3, padx=5, pady=5)

            self.input_button.grid(row=2, column=3, padx=5,
pady=5,\ sticky="e")
            self.write_down()

        else:
            self.write_down()
            self.name_label.grid_forget()
            self.surname_label.grid_forget()
            self.zodiac_label.grid_forget()
            self.dob_lable.grid_forget()

            self.name_entry.grid_forget()
            self.surname_entry.grid_forget()
            self.zodiac_entry.grid_forget()
            self.dob_entry.grid_forget()

            self.input_button.grid_forget()

            self.name_outp_lable.grid(row=3, column=0,
sticky="w")
            self.name_outp_entry.grid(row=3, column=1, padx=5,\
pady=5)
            self.name_outp_button.grid(row=3, column=2, padx=5,\
pady=5, sticky="w")
            self.info_outp_lable.grid(row=4, column=1,
sticky="w")

            self.blank_lable.grid(row=4, column=0, sticky="w")
            self.file_entry.grid(row=5, column=0, padx=5,
pady=5)
```

```
            self.save_button.grid(row=5, column=1, padx=5,
pady=5,\ sticky="w")

        if self.n == 7:
            self.sort()

        self.n += 1

    def write_down(self):
        global zod
        z = Zodiac(self.name.get(), self.surname.get(),\
self.zodiac.get(), self.dob.get())
        zod.append(z)

    def sort(self):
        global zod
        zn = []

        for i in range(8):
            zn.append(int(zod[i].dob[0])+ int(zod[i].dob[1]) *
30 +\ int(zod[i].dob[2]) * 365)
        for i in range(7):
            for j in range(i, 8):
                if zn[i] > zn[j]:
                    promzn = zn[i]
                    promzod = zod[i]
                    zn[i] = zn[j]
                    zod[i] = zod[j]
                    zn[j] = promzn
                    zod[j] = promzod

    def outp(self):
        ind = 1

        try:
```

```
            (name, surname) = str(self.nameoutp.get()).split('
')
        except:
            self.info_outp_lable.grid_forget()
            self.info_outp_lable = Label(text='Не найдено')
            self.info_outp_lable.grid(row=4, column=1,
sticky="w")

        for i in range(8):
            if zod[i].name == name and zod[i].surname ==
surname:
                self.info_outp_lable.grid_forget()
                self.info_outp_lable = Label(text='{0} {1} {2}
{3}.{4}.{5}'.format(zod[i].name, zod[i].surname,

zod[i].zodiac, zod[i].dob[0],

zod[i].dob[1], zod[i].dob[2]))
                self.info_outp_lable.grid(row=4, column=1,\
sticky="w")
                ind = 0
        if ind:
            self.info_outp_lable.grid_forget()
            self.info_outp_lable = Label(text='Не найдено')
            self.info_outp_lable.grid(row=4, column=1,
sticky="w")


    def save(self):
        name = self.filename.get()
        f = open(name+'.csv', 'w')
        out = ''
        out += 'Имя;Фамилия;Зодиак;Дата рождения;\n'
        for i in range(8):
            out +=
'{0};{1};{2};{3}.{4}.{5}\n'.format(zod[i].name,\
zod[i].surname,\

zod[i].zodiac,\ zod[i].dob[0],\
```

```
zod[i].dob[1],\ zod[i].dob[2])
        f.write(out)
        f.close()


# main
zod = []

root = Tk()
root.title("GUI")
app = Application(root)
root.mainloop()
```

Описание подпрограмм

Функция __init__(self)

Конструктор класса.

Функция increase(self)

Отвечает за ввод данных пользователем n раз.

Функция write_down(self)

Отвечает за создание списка объектов

Функция sort(self)

Отвечает за сортировку списка объектов

Функция outp(self)

Отвечает за вывод данных из списка по запросу

Функция save(self)

Отвечает за сохранение списка в файл

Задание к лабораторной работе **№7 "GUI, классы, модуль Tkinter"**

Программа должна содержать меню и ввод-вывод в окна на экране. Необходимо предусмотреть контроль ошибок пользователя при вводе данных.

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

Вариант 1

Сводная ведомость результатов экзаменационной сессии студенческой группы находится в файле на диске и для каждого студента содержит фамилию, инициалы и оценки по пяти предметам. Количество студентов в группе не превышает 20 человек. Составить программу, с помощью которой можно корректировать и дополнять список и получать:

- список студентов;
- список студентов, сдавших экзамены только на «5»;
- список студентов, имеющих тройки;
- список студентов, имеющих двойки. При этом студент, имеющий более чем одну двойку, исключается из списка.

Вариант 2

Предприятие имеет местную телефонную станцию на 20 номеров. Телефонный справочник данного предприятия для каждого номера телефона содержит номер помещения и список служащих, сидящих в данном помещении. Составить программу, которая:

- корректирует базу;
- по номеру телефона выдаст номер помещения и список сидящих в нем людей;
- по номеру помещения выдает номер телефона;
- по фамилии выдает номер телефона и номер помещения.
- Номер телефона - двузначный. В одном помещении может находиться от одного до четырех служащих.

Вариант 3

В гостинице имеется 15 номеров, из них 5 одноместных и 10 двухместных.
Составить программу, которая заполняет и (или) корректирует данные о жильцах и по фамилии определяет номер, где проживает жилец. Программа запрашивает фамилию жильца.

- Если жильца с такой фамилией нет, об этом выдается сообщение.
- Если жилец с такой фамилией в гостинице единственный, программа выдает фамилию жильца и номер проживания.
- Если в гостинице проживает два или более жильцов с такой фамилией, программа дополнительно запрашивает инициалы.

Вариант 4

В текстовом файле хранится список служащих. Для каждого служащего указаны фамилия и инициалы, название занимаемой должности, год поступления на работу и оклад. Написать программу, выполняющую следующие действия:

- корректировку или дополнение списка с клавиатуры;
- сортировку по фамилии, окладу или году поступления;
- вывод на экран информации о служащем, фамилия которого введена с клавиатуры;
- запись списка в файл под тем же или новым именем.

Вариант 5

Расписание электричек хранится в текстовом файле на диске. Каждая запись содержит название пункта назначения, пометки типа «СВ», «ПВ», «КСВ» и время отправления. Написать программу, выполняющую следующие действия:

- корректировку или дополнение расписания с клавиатуры;
- сортировку по станции назначения или по времени отправления;
- вывод на экран информации о поездах, отходящих после введенного времени;
- запись расписания в файл под тем же или новым именем.

Вариант 6

В текстовом файле хранится список товаров. Для каждого товара указаны его название, стоимость единицы товара в тыс. руб., количество и единица измерения (например, 100 шт., 20 кг). Написать программу, выполняющую следующие действия:

- корректировку или дополнение списка с клавиатуры;
- сортировку по названию товара или по общей стоимости;
- вывод на экран информации о товаре, название которого введено с клавиатуры;
- запись списка в файл под тем же или новым именем.

Вариант 7

В текстовом файле хранится список товаров. Для каждого товара указаны его название, название магазина, в котором продается товар, стоимость товара в тыс. руб. и его количество с указанием единицы измерения (например, 100 шт., 20 кг). Написать программу, выполняющую следующие действия:

- корректировку или дополнение списка с клавиатуры;
- сортировку по названию товара или по названию магазина;
- вывод на экран информации о товаре, название которого введено с клавиатуры;
- запись списка в файл под тем же или новым именем.

Вариант 8

Список студенческой группы записан на диске как текстовый файл. Каждая строка списка содержит фамилию студента и три экзаменационные оценки, причем список никак не упорядочен. Составить программу, которая корректирует список и сортирует его либо по среднему баллу, либо по алфавиту, либо по оценкам по заданному предмету. Список записывается в файл либо под старым, либо под новым именем.

Вариант 9

В текстовом файле хранится список товаров. Для каждого товара указаны его название, название магазина, в котором продается товар, стоимость товара в тыс. руб. и его количество с указанием единицы измерения (например, 100 шт., 20 кг). Написать программу, выполняющую следующие действия:

- корректировку или дополнение списка с клавиатуры;
- сортировку по названию товара или по названию магазина;
- вывод на экран информации о товаре, название которого введено с клавиатуры;
- запись списка в файл под тем же или новым именем.

Вариант 10

1. Описать запись с именем Route, содержащую следующие поля:
   - название начального пункта маршрута;
   - название конечного пункта маршрута;
   - номер маршрута.
2. Написать программу, выполняющую следующие действия:
   - ввод данных с клавиатуры в массив, состоящий из восьми элементов типа Route; записи должны быть упорядочены по номерам маршрутов;
   - вывод на экран информации о маршруте, номер которого введен с клавиатуры; если таких маршрутов нет, вывести на экран соответствующее сообщение;

- запись массива в файл под заданным с клавиатуры именем.