

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ
ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ТУЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

Институт прикладной математики и компьютерных наук

Кафедра «Информационная безопасность»

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ
ПО КОНТРОЛЬНО-РАСЧЕТНОМУ ЗАДАНИЮ**

по дисциплине

«ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ»

Уровень профессионального образования: *высшее образование –
бакалавриат*

Направление: *09.03.03 Прикладная информатика*

Профиль: *Прикладная информатика в экономике*

Квалификация выпускника: *бакалавр*

Форма обучения: *дистанционная*

Тула, 2018

ЦЕЛЬ И ЗАДАЧИ РАБОТЫ: создания приложения на языке программирования Python с графическим пользовательским интерфейсом с помощью библиотеки tkinter.

1. ОБЩИЕ ПОЛОЖЕНИЯ

В многообразии программ, которые пишут программисты, выделяют приложения с графическим пользовательским интерфейсом (GUI). При создании таких программ становятся важными не только алгоритмы обработки данных, но и разработка для пользователя программы удобного интерфейса, взаимодействуя с которым, он будет определять поведение приложения.

Современный пользователь в основном взаимодействует с программой с помощью различных кнопок, меню, значков, вводя информацию в специальные поля, выбирая определенные значения в списках и т. д. Эти "изображения" в определенном смысле и формируют GUI, в дальнейшем мы их будем называть виджетами (от англ. widget - "штучка").

Для языка программирования Python такие виджеты включены в специальную библиотеку — tkinter. Если ее импортировать в программу (скрипт), то можно пользоваться ее компонентами, создавая графический интерфейс.

Последовательность шагов при создании графического приложения имеет свои особенности. Программа должна выполнять свое основное назначение, быть удобной для пользователя, реагировать на его действия. Мы не будем вдаваться в подробности разработки, а рассмотрим какие этапы приблизительно нужно пройти при программировании, чтобы получить программу с GUI:

1. Импорт модуля tkinter

Как и любой модуль, tkinter в Python можно импортировать двумя способами: командами `import tkinter` или `from tkinter import *`. В дальнейшем мы будем пользоваться только вторым способом, т. к. это позволит не указывать каждый раз имя модуля при обращении к объектам, которые в нем содержатся. *Следует обратить внимание, что в версии Python 3 имя модуля пишется со строчной буквы (tkinter), хотя в более ранних версиях использовалась прописная (Tkinter).* Итак, первая строка программы должна выглядеть так:

```
from tkinter import *
```

или (в версиях Python <3.0):

```
from Tkinter import *
```

2. Создание главного окна

В современных операционных системах любое пользовательское приложение заключено в окно, которое можно назвать главным, т.к. в нем располагаются все остальные виджеты. Объект окна верхнего уровня создается при обращении к классу Tk модуля tkinter. Переменную связанную с объектом-окном принято называть root (хотя понятно, что можно назвать как угодно, но так уж принято). Вторая строка кода:

```
root = Tk()
```

3. Создание виджет

Допустим в окне будет располагаться всего одна кнопка. Кнопка создается при обращении к классу Button модуля tkinter. Объект кнопка связывается с какой-нибудь переменной. У класса Button (как и всех остальных классов, за исключением Tk) есть обязательный параметр — объект, которому кнопка принадлежит (кнопка не может "быть ничейной"). Пока у нас есть единственное окно (root), оно и будет аргументом, передаваемым в класс при создании объекта-кнопки:

```
but = Button(root)
```

4. Установка свойств виджет

У кнопки много свойств: размер, цвет фона и надписи и др. Например, установим всего одно свойство — текст надписи (text):

```
but["text"] = "Печать"
```

5-6. Определение событий и их обработчиков

Что же будет делать кнопка и в какой момент она это будет делать? Предположим, что задача кнопки вывести какое-нибудь сообщение в поток вывода, используя функцию print. Делать она это будет при нажатии на нее левой кнопкой мыши.

Действия (алгоритм), которые происходят при том или ином событии, могут быть достаточно сложным. Поэтому часто их оформляют в виде функции, а затем вызывают, когда они понадобятся. Пусть у нас печать на экран будет оформлена в виде функции printer:

```
def printer(event):  
    print ("Как всегда очередной 'Hello World!'")
```

Не забывайте, что функцию желательно (почти обязательно) размещать в начале кода. Параметр event — это какое-либо событие.

Событие нажатия левой кнопкой мыши выглядит так: <Button-1>. Требуется связать это событие с обработчиком (функцией printer). Для связи предназначен метод bind. Синтаксис связывания события с обработчиком выглядит так:

```
but.bind("<Button-1>",printer)
```

7. Размещение виджет

Если вы заметите, то в любом приложении виджеты не разбросаны по окну как попало, а хорошо организованы, интерфейс продуман до мелочей и обычно подчинен определенным стандартам. До стандартов нам далеко, нужно просто кнопку как-то отобразить в окне. Самый простой способ — это использование метода pack.

```
but.pack()
```

Если не вставить эту строчку кода, то кнопка в окне так и не появится, хотя она есть в программе.

8. Отображение главного окна

Ну и наконец, главное окно тоже не появится, пока не будет вызван специальный метод.mainloop:

```
root.mainloop()
```

Данная строчка кода должна быть всегда в конце скрипта!
В итоге, код программы может выглядеть таким образом:

```
from tkinter import *

def printer(event):
    print ("Как всегда очередной 'Hello World!'")

root = Tk()
but = Button(root)
but["text"] = "Печать"
but.bind("<Button-1>",printer)

but.pack()
root.mainloop()
```

При программировании графического интерфейса пользователя более эффективным оказывается объектно-ориентированный подход. Поэтому многие «вещи» оформляются в виде классов. В нашем примере также можно использовать класс:

```
from tkinter import *

class But_print:
    def __init__(self):
        self.but = Button(root)
        self.but["text"] = "Печать"
        self.but.bind("<Button-1>",self.printer)
        self.but.pack()
    def printer(self,event):
        print ("Как всегда очередной 'Hello World!'")

root = Tk()
obj = But_print()
root.mainloop()
```

9. Кнопки

Объект-кнопка создается вызовом класса Button модуля tkinter. При этом обязательным аргументом является лишь родительский виджет (например, окно верхнего уровня). Другие свойства могут указываться при создании кнопки или задаваться (изменяться) позже. Синтаксис:

переменная = Button (родит_виджет, [свойство=значение,])

У кнопки много свойств, в примере ниже указаны лишь некоторые из них.

```
from tkinter import *

root = Tk()

but = Button(root,
    text="Это кнопка", #надпись на кнопке
    width=30,height=5, #ширина и высота
```

```
bg="white",fg="blue") #цвет фона и надписи
```

```
but.pack()  
root.mainloop()
```

bg и fg – это сокращения от background (фон) и foreground (передний план). Ширина и высота измеряются в знаках (количество символов).

10. Метки

Метки (или надписи) — это достаточно простые виджеты, содержащие строку (или несколько строк) текста и служащие в основном для информирования пользователя.

```
lab = Label(root, text="Это метка! \n Из двух строк.", font="Arial 18")
```

11. Однострочное текстовое поле

Такое поле создается вызовом класса Entry модуля tkinter. В него пользователь может ввести только одну строку текста.

```
ent = Entry(root,width=20,bd=3)
```

bd – это сокращение от borderwidth (ширина границы).

12. Многострочное текстовое поле

Text предназначен для предоставления пользователю возможности ввода не одной строки текста, а существенно больше.

```
tex = Text(root,width=40,  
font="Verdana 12",  
wrap=WORD)
```

Последнее свойство (wrap) в зависимости от своего значения позволяет переносить текст, вводимый пользователем либо по символам, либо по словам, либо вообще не переносить, пока пользователь не нажмет Enter.

13. Радиокнопки (переключатели)

Объект-радиокнопка никогда не используется по одному. Их используют группами, при этом в одной группе может быть «включена» лишь одна кнопка.

```
var=IntVar()  
var.set(1)  
rad0 = Radiobutton(root,text="Первая",  
variable=var,value=0)  
rad1 = Radiobutton(root,text="Вторая",  
variable=var,value=1)  
rad2 = Radiobutton(root,text="Третья",  
variable=var,value=2)
```

Одна группа определяет значение одной переменной, т. е. если в примере будет выбрана радиокнопка rad2, то значение переменной будет var будет 2. Изначально также требуется установить значение переменной (выражение var.set(1) задает значение переменной var равное 1).

14. Флажки

Объект `checkboxbutton` предназначен для выбора не взаимоисключающих пунктов в окне (в группе можно активировать один, два или более флажков или не один). В отличие от радиокнопок, значение каждого флажка привязывается к своей переменной, значение которой определяется опциями `onvalue` (включено) и `offvalue` (выключено) в описании флажка.

```
c1 = IntVar()
c2 = IntVar()
che1 = Checkbutton(root, text="Первый флажок",
    variable=c1, onvalue=1, offvalue=0)
che2 = Checkbutton(root, text="Второй флажок",
    variable=c2, onvalue=2, offvalue=0)
```

15. Списки

Вызов класса `Listbox` создает объект, в котором пользователь может выбрать один или несколько пунктов в зависимости от значения опции `selectmode`. В примере ниже значение `SINGLE` позволяет выбирать лишь один пункт из списка.

```
r = ['Linux', 'Python', 'Tk', 'Tkinter']
lis = Listbox(root, selectmode=SINGLE, height=4)
for i in r:
    lis.insert(END, i)
```

Изначально список (`Listbox`) пуст. С помощью цикла `for` в него добавляются пункты из списка (тип данных) `r`. Добавление происходит с помощью специального метода класса `Listbox` — `insert`. Данный метод принимает два параметра: куда добавить и что добавить.

16. Frame (рамка)

Как выяснится позже, рамки (фреймы) хороший инструмент для организации остальных виджетов в группы внутри окна, а также оформления.

```
from tkinter import *

root = Tk()

fra1 = Frame(root, width=500, height=100, bg="darkred")
fra2 = Frame(root, width=300, height=200, bg="green", bd=20)
fra3 = Frame(root, width=500, height=150, bg="darkblue")

fra1.pack()
fra2.pack()
fra3.pack()

root.mainloop()
```

Данный скрипт создает три фрейма разного размера. Свойство `bd` (сокращение от `boderwidth`) определяет расстояния от края рамки до заключенных в нее виджетов (если они есть).

На фреймах также можно размещать виджеты как на основном окне (`root`). Здесь текстовое поле находится на рамке `fra2`.

```
ent1 = Entry(fra2,width=20)
ent1.pack()
```

17. Scale (шкала)

Назначение шкалы — это предоставление пользователю выбора какого-то значения из определенного диапазона. Внешне шкала представляет собой горизонтальную или вертикальную полосу с разметкой, по которой пользователь может передвигать движок, осуществляя тем самым выбор значения.

```
sca1 = Scale(fra3,orient=HORIZONTAL,length=300,
            from_=0,to=100,tickinterval=10,resolution=5)
sca2 = Scale(root,orient=VERTICAL,length=400,
            from_=1,to=2,tickinterval=0.1,resolution=0.1)
```

Свойства:

- `orient` определяет направление шкалы;
- `length` – длина шкалы в пикселях;
- `from_` и `to` – с какого значения шкала начинается и каким заканчивается (т. е. диапазон значений);
- `tickinterval` – интервал, через который отображаются метки для шкалы;
- `resolution` - минимальная длина отрезка, на которую пользователь может передвинуть движок.

18. Scrollbar (полоса прокрутки)

Данный виджет позволяет прокручивать содержимое другого виджета (например, текстового поля или списка). Прокрутка может быть как по горизонтали, так и по вертикали.

```
from tkinter import *

root = Tk()

tx = Text(root,width=40,height=3,font='14')
scr = Scrollbar(root,command=tx.yview)
tx.configure(yscrollcommand=scr.set)

tx.grid(row=0,column=0)
scr.grid(row=0,column=1)
root.mainloop()
```

В примере сначала создается текстовое поле (`tx`), затем полоса прокрутки (`scr`), которая привязывается с помощью опции `command` к полю `tx` по вертикальной оси (`yview`). Далее поле `tx` изменяется (конфигурируется) с помощью метода `configure`: устанавливается значение опции `yscrollcommand`.

Здесь используется незнакомый нам пока еще метод `grid`, представляющий собой другой способ расположения виджет на окне.

19. Toplevel (окно верхнего уровня)

С помощью класс `Toplevel` создаются дочерние окна, на которых также могут располагаться виджеты. Следует отметить, что при закрытии главного окна (или

родительского), окно Toplevel также закрывается. С другой стороны, закрытие дочернего окна не приводит к закрытию главного.

```
win = Toplevel(root,relief=SUNKEN,bd=10,bg="lightblue")
win.title("Дочернее окно")
win.minsize(width=400,height=200)
```

Метод title определяет заголовок окна. Метод minsize конфигурирует минимальный размер окна (есть метод maxsize, определяющий максимальный размер окна). Если значение аргументов minsize будет таким же как у maxsize, то пользователь не сможет менять размеры окна.

Обычно, чтобы графическое приложение что-то сделало, должно случиться какое-нибудь событие, т. е. воздействие на GUI из вне.

20. Типы событий

Можно выделить три основных типа событий: производимые мышью, нажатиями клавиш на клавиатуре, а также события, возникающие в результате изменения других графических объектов.



21. Способ записи

При вызове метода bind событие передается в качестве первого аргумента.

↓
widget.bind(**event**,function)

Название события заключается в кавычки, а также в знаки < и >. Событие описывается с помощью зарезервированных последовательностей ключевых слов.

22. События, производимые мышью

- <Button-1> - щелчок левой кнопкой мыши
- <Button-2> - щелчок средней кнопкой мыши
- <Button-3> - щелчок правой кнопкой мыши
- <Double-Button-1> - двойной клик левой кнопкой мыши
- <Motion> - движение мыши
- и т. д.

```
from tkinter import *
def b1(event):
```

```

    root.title("Левая кнопка мыши")
def b3(event):
    root.title("Правая кнопка мыши")
def move(event):
    root.title("Движение мышью")

root = Tk()
root.minsize(width = 500, height=400)

root.bind('<Button-1>',b1)
root.bind('<Button-3>',b3)
root.bind('<Motion>',move)

```

```

root.mainloop()

```

В этой программе меняется надпись в заголовке главного окна в зависимости от того двигается мышь, щелкают левой или правой кнопкой мыши.

23. События, производимые с помощью клавиатуры

- Буквенные клавиши можно записывать без угловых скобок (например, 'L').
- Для неалфавитных клавиш существуют специальные зарезервированные слова
 - * <Return> - нажатие клавиши Enter;
 - * <space>- пробел;
 - * и т. д.
- Сочетания клавиш пишутся через тире. Например:
 - * <Control-Shift> - одновременное нажатие клавиш Ctrl и Shift.

```

from tkinter import *

def exit_(event):
    root.destroy()
def caption(event):
    t = ent.get()
    lbl.configure(text = t)

root = Tk()

ent = Entry(root, width = 40)
lbl = Label(root, width = 80)

ent.pack()
lbl.pack()

ent.bind('<Return>',caption)
root.bind('<Control-z>',exit_)

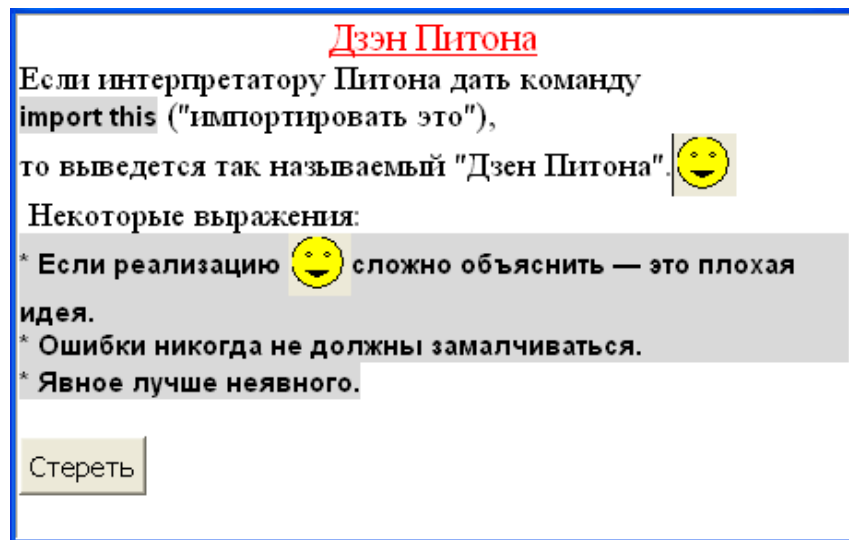
root.mainloop()

```

При нажатии клавиши Enter в пределах текстовой строки (ent) вызывается функция caption, которая помещает символы из текстовой строки (ent) в метку (lbl). Нажатие комбинации клавиш Ctrl + z приводит к закрытию главного окна.

24. Особенности работы с виджетом Text модуля Tkinter.

Графический элемент Text предоставляет большие возможности для работы с текстовой информацией. Помимо разнообразных операций с текстом и его форматированием в экземпляр объекта Text можно вставлять другие виджеты (следует отметить, что такая же возможность существует и для Canvas). В данном примере рассматриваются лишь некоторые возможности виджета Text на примере создания окна с текстовым полем, содержащим форматированный текст, кнопку и возможность добавления экземпляров холста.



1. Для начала создадим текстовое поле, установив при этом некоторые из его свойств:

```
#текстовое поле и его первоначальные настройки
tx = Text(font=('times',12),width=50,height=15,wrap=WORD)
tx.pack(expand=YES,fill=BOTH)
```

2. Теперь допустим нам нужно добавить какой-нибудь текст. Сделать это можно с помощью метода insert, передав ему два обязательных аргумента: место, куда вставить, и объект, который следует вставить. Объектом может быть строка, переменная, ссылающаяся на строку или какой-либо другой объект. Место вставки может указываться несколькими способами. Один из них — это индексы. Они записываются в виде 'х.у', где х — это строка, а у — столбец. При этом нумерация строк начинается с единицы, а столбцов с нуля. Например, первый символ в первой строке имеет индекс '1.0', а десятый символ в пятой строке — '5.9'.

```
tx.insert(1.0,'Дзен Питона\n\
Если интерпретатору Питона дать команду\n\
import this ("импортировать это"),\n\
то выведется так называемый "Дзен Питона".\n\
Некоторые выражения:\n\
* Если реализацию сложно объяснить — это плохая идея.\n\
* Ошибки никогда не должны замалчиваться.\n\
* Явное лучше неявного.\n\n')
```

Комбинация символов '\n' создает новую строку (т.е. при интерпретации последующий текст начнется с новой строки). Одиночный символ '\n' никак не влияет на отображение текста при выполнении кода, его следует вставлять при переносе текста при написании скрипта.

Когда содержимого текстового поля нет вообще, то единственный доступный индекс — '1.0'. В заполненном текстовом поле вставлять можно в любое место (где есть содержимое).

Если выполнить скрипт, содержащий только данный код (+ импорт модуля Tkinter, + создание главного окна, + `mainloop()` в конце), то мы увидим текстовое поле с восемью строками текста. Текст не оформлен.

3. Теперь отформатируем разные области текста по-разному. Для этого сначала зададим теги для нужных нам областей, а затем для каждого тега установим настройки шрифта и др.

```
#установка тегов для областей текста
tx.tag_add('title','1.0','1.end')
tx.tag_add('special','6.0','8.end')
tx.tag_add('special','3.0','3.11')

#конфигурирование тегов
tx.tag_config('title',foreground='red',
              font=('times',14,'underline'),justify=CENTER)
tx.tag_config('special',background='grey85',font=('Dejavu',10,'bold'))
```

Добавление тега осуществляется с помощью метода `tag_add`. Первый атрибут — имя тега (произвольное), далее с помощью индексов указывается к какой области текстового поля он прикрепляется (начальный символ и конечный). Вариант записи как '1.end' говорит о том, что нужно взять текст до конца указанной строки. Разные области текста могут быть помечены одинаковым тегом.

Метод `tag_config` применяет те или иные свойства к тегу, указанному в качестве первого аргумента.

4. В многострочное текстовое поле можно добавлять не только текст, но и другие объекты. Например, вставим в поле кнопку (ну и функцию заодно).

```
def erase():
    tx.delete('1.0',END)
...
#добавление кнопки
bt = Button(tx,text='Стереть',command=erase)
tx.window_create(END>window=bt)
```

Кнопка — это виджет. Виджеты добавляются в текстовое поле с помощью метода `window_create`, где в качестве первой опции указывается место добавления, а второй (`window`) — в качестве значения присваивается переменная, связанная с объектом.

При щелчке ЛКМ (левой кнопкой мыши) по кнопке будет вызываться функция `erase`, в которой с помощью метода `delete` удаляется все содержимое поля (от '1.0' до `END`).

5. А вот более интересный пример добавления виджета в поле `Text`:

```
def smiley(event):
    cv = Canvas(height=30,width=30)
    cv.create_oval(1,1,29,29,fill="yellow")
    cv.create_oval(9,10,12,12)
    cv.create_oval(19,10,22,12)
```

```
cv.create_polygon(9,20,15,24,22,20)
tx.window_create(CURRENT,window=cv)
...
#ЛКМ -> смайлик
tx.bind('<Button-1>',smiley)
```

Здесь при щелчке ЛКМ в любом месте текстового поля будет вызываться функция smiley. В теле данной функции создается объект холста, который в конце с помощью метода window_create добавляется на объект tx. Место вставки указано как CURRENT, т. е. "текущее" - это там, где был произведен щелчок мышью.

2. ОБОРУДОВАНИЕ

ПЭВМ IBM PC, SVGA монитор с разрешением не менее 800*600 пикселей; клавиатура; мышь, ОС Linux/Windows, язык программирования Python, специальная библиотека tkinter, среда Geany.

3. ПОДГОТОВКА К РАБОТЕ

1. Изучить теоретическую часть.

4. ПРОГРАММА РАБОТЫ

1. Разобрать примеры, приведенные в общих положениях данной лабораторной работы.
2. Написать и отладить программу, соответствующему индивидуальному заданию (приложение А) с использованием библиотеки tkinter (предусмотреть разработку графического интерфейса программы с режимами ввода, корректировки и отображения необходимой информации, организованными с помощью меню)

Номер индивидуального задания соответствует последней цифре номера зачетки.

3. Результаты работы программ показать преподавателю.
4. Оформить отчет.
5. Защитить работу перед преподавателем.

5. СОДЕРЖАНИЕ ОТЧЕТА

1. Название работы
2. Цель и задачи
3. Тексты программ и результаты выполнения.
4. Результаты и выводы по работе

Приложение А

№0

Ввести и сохранить в файле данные следующей структуры: наименование фирмы, адрес фирмы, наименование товара, розничная цена, оптовая цена. Организовать просмотр исходных данных и вывести список фирм, производящих оптовую продажу товара определенного наименования (по выбору), отсортированных по возрастанию цены (метод пузырька), с указанием адреса и цены.

Ввод и вывод данных организовать в виде таблиц. Отладку программы производить на примере файла, состоящего не менее чем из 15 записей.

№1

Ввести и сохранить в файле данные следующей структуры: наименование банка, наименование обмениваемых валют, курс покупки – продажи валюты. Организовать просмотр исходных данных и вывести список банков, отсортированный по убыванию курса продажи валюты (метод слияния), с указанием наименования валюты и курса.

Ввод и вывод данных организовать в виде таблиц. Отладку программы производить на примере файла, состоящего не менее чем из 15 записей.

№2

Ввести и сохранить в файле данные следующей структуры: наименование фирмы, наименование товара, стоимость товара, размер скидки при покупке более 10 единиц товара, стоимость перевозки, количество товара. Организовать просмотр исходных данных и вывести список фирм, предоставляющих скидку более заданной (по выбору), отсортированный по стоимости товара (убывание метод вставки), с указанием наименования товара, стоимости товара, размера скидки, стоимости с учетом скидки и количества.

Ввод и вывод данных организовать в виде таблиц. Отладку программы производить на примере файла, состоящего не менее чем из 15 записей.

№3

Ввести и сохранить в файле данные следующей структуры: наименование фирмы, наименование товара, стоимость единицы товара, стоимость перевозки, количество товара. Организовать просмотр исходных данных и вывести список фирм, отсортированный по стоимости затрат на приобретение и перевозку товара (возрастание метод пузырька), с указанием наименования товара, количества и общих затрат.

Ввод и вывод данных организовать в виде таблиц. Отладку программы производить на примере файла, состоящего не менее чем из 15 записей.

№4

Ввести и сохранить в файле данные следующей структуры: наименование фирмы, марка компьютера, стоимость компьютера, гарантийный срок обслуживания, устанавливаемая система при продаже (MS-DOS, Windows). Организовать просмотр исходных данных и вывести список всех фирм, отсортированный по возрастанию стоимости (метод Шелла) и имеющих гарантию более 1 года, с указанием марки, стоимости и срока гарантии.

Ввод и вывод данных организовать в виде таблиц. Отладку программы производить на примере файла, состоящего не менее чем из 15 записей.

№5

Ввести и сохранить в файле данные следующей структуры: номер рейса, пункт отправления, пункт назначения, время отправления, продолжительность полета,

стоимость билета. Организовать просмотр исходных данных и вывести список самолетов данного направления (по выбору), отсортированный по возрастанию продолжительности полета (метод пузырька), с указанием номера рейса, времени отправления и стоимости билета.

Ввод и вывод данных организовать в виде таблиц. Отладку программы производить на примере файла, состоящего не менее чем из 15 записей.

№6

Ввести и сохранить в файле данные следующей структуры: ф.,и.,о. водителя, номер и марка автомашины (5-7 марок), год приобретения, количество нарушений. Организовать просмотр исходных данных и вывести список нарушителей (сортировка по количеству нарушений по возрастанию метод Шелла), с указанием ф.,и.,о. водителя, номера машины и количества нарушений.

Ввод и вывод данных организовать в виде таблиц. Отладку программы производить на примере файла, состоящего не менее чем из 15 записей.

№7

Ввести и сохранить в файле данные следующей структуры: фамилия лечащего врача, фамилия пациента, возраст, заболевание, количество обращений к врачам. Организовать просмотр исходных данных и вывести список пациентов, отсортированный по количеству обращений к врачам (по убыванию метод вставки), с указанием возраста, заболеваний, количества обращений.

Ввод и вывод данных организовать в виде таблиц. Отладку программы производить на примере файла, состоящего не менее чем из 15 записей.

№8

Ввести и сохранить в файле данные следующей структуры: название фильма, режиссер, цена, продолжительность фильма, количество проданных кассет. Организовать просмотр исходных данных и вывести список фильмов, отсортированный с учетом популярности (по убыванию метод пузырька), с указанием режиссера, цены, количества проданных кассет.

Ввод и вывод данных организовать в виде таблиц. Отладку программы производить на примере файла, состоящего не менее чем из 15 записей.

№9

Ввести и сохранить в файле данные следующей структуры: номер заказа, тип телевизора, количество комплектующих деталей, фамилия мастера (4 мастера), срок исполнения. Организовать просмотр исходных данных и вывести список заказов, отсортированный по количеству необходимых комплектующих деталей (по возрастанию метод Шелла), с указанием фамилии мастера, срока исполнения.

Ввод и вывод данных организовать в виде таблиц. Отладку программы производить на примере файла, состоящего не менее чем из 15 записей.