

Введение

В пособии приведены задания к лабораторным работам и требования к их оформлению. Задания предназначены для выполнения практических работ, которые студенты выполняют в процессе прохождения материала по теме "Программирование на языке высокого уровня. Python".

Лабораторные работы снабжены методическими указаниями по их решению и примерами программного кода с комментариями.

В теоретической части каждой лабораторной работы рассматриваются основы применения языка Python, а так же алгоритмы, использованные в решениях.

В приложениях к пособию, дан справочный материал по установке Python и настройке переменных среды ОС и ярлыка для запуска IDLE Python, основным функциям среды IDLE, приведены обозначения, используемые для построения блок-схем алгоритмов, а так же описания основных модулей Python, которые используются в процессе написания и отладки программ.

Пособие предназначено для студентов, обучающихся по направлению "Информатика и вычислительная техника", и может быть использовано преподавателем при подготовке к лекциям, при выполнении практических занятий и в лабораторных работах.

Порядок выполнения работы

1. Внимательно прочитать и уяснить условие задачи, которую предстоит решить.
2. Ознакомиться с необходимым теоретическим материалом, используя литературу, указанную в пособии и к лабораторным работам. В качестве основного источника теоретического и практического материала рекомендуется использовать книги [1] и [2] (см. список литературы на стр.6).
3. Разработать алгоритм решения задачи. Уточнить последовательность выполнения его пунктов.
4. Изучить примеры, приведенные в литературе, в том числе и в этом пособии. При необходимости выполнить их на компьютере, а в дальнейшем использовать фрагменты для написания собственного решения.
5. Подготовить свой вариант решения и отладить его с помощью компьютера.
6. ***Подготовить отчёт.***

Форма отчёта

Каждый отчёт оформляется в виде пояснительной записки и должен содержать следующие элементы:

- титульный лист;
- текст пояснительной записки в машинописном или рукописном виде;
- список использованной литературы или сайтов Интернет;
- листинг программы на языке Python, результат работы программы - в виде приложения. Допускается приводить результат работы программы в виде фрагмента (не полное решение).

С примером оформления отчёта по лабораторной работе можно познакомиться в материале для лабораторной работы №2, задание 1 (см. стр. 14)

Требования к оформлению

Данные требования относятся к машинописному варианту оформления отчёта текстовым процессором Word. При оформлении другими программными средствами следует использовать режимы, которые в максимальной степени приближают оформление к настоящим требованиям. Рукописный вариант должен быть оформлен аккуратно и читаемо.

Графические изображения (рисунки, блок-схемы) можно оформлять карандашом. При наличии навыков, графические изображения могут быть оформлены в Word'e или с использованием возможностей электронного табличного процессора Excel, графического редактора Paint, офисного приложения Visio.

Отчёт должен быть оформлен на бумаге формата **A4**. Основной текст, кроме титульного листа, выполняется шрифтом **Times New Roman** размером **12** с одиночным или полуторным интервалом.

В параметрах страницы следует использовать поля следующих размеров:
Отступ сверху - **1.5**; Отступ снизу - **2.5**; Отступ слева - **2.5**; Отступ справа - **1.0**.

Листинг программы оформляется равношириным шрифтом **Courier New** размером **10:-12**. При выборе размера следует ориентироваться на то, что строка программы должна размещаться на строке текста без переносов. Длинные инструкции можно разбивать на более короткие¹.

Содержание пояснительной записи

1. Постановка задачи.
2. Краткие теоретические сведения об особенностях применяемых операторов и методов (теоретическое введение).
3. Описание программы:
 - общие сведения (язык программирования, операционная система, тип процессора);
 - описание логической структуры программы;
 - описание алгоритма решения задачи (в виде блок-схемы);
 - описание входных и выходных данных программы;
 - описание подпрограмм;
 - тестовые примеры, перечень аномальных и допустимых значений входных данных.

Перечень лабораторных работ

В процессе прохождения материала по теме "Программирование на языке высокого уровня. Python" студенты выполняют следующие лабораторные работы:

Тема: Структурное программирование

1. Линейные программы.
2. Разветвляющиеся вычислительные процессы: Задание 1, Задание 2.
3. Организация циклов: Задание 1, Задание 2, Задание 3.
4. Одномерные массивы.
5. Двумерные массивы и подпрограммы.

¹ Длинную инструкцию можно перенести на следующую строку так:

- в конце строки пометить символ обратного слэш - \. За символом должен следовать символ перевода строки (Enter)

- поместить выражение в круглые скобки или, при определении списка или словаря – квадратные и фигурные скобки. Подробности можно найти в примерах к лабораторным работам.

6. Строки, записи, модуль Crt.
7. Программирование в графическом режиме.
8. Динамические структуры данных.

Тема: Объектно-ориентированное программирование

9. Объекты.
10. Наследование.
11. Использование стандартных объектов.

Данное пособие разбито на две части в соответствии с изучаемыми темами.

Выбор варианта

Лабораторные работы могут иметь несколько заданий, как, например, лабораторные работы № 2 и №3, и множество вариантов. Студент выполняет, для каждой лабораторной работы (задания), свой вариант, который связан с номером зачетки:

- две последние цифры зачетки **nm** целочисленно делить на число вариантов в задании **z**.

$$N = nm \% z + 1,$$

где **%** - получение остатка при целочисленном делении, а **N** – номер варианта задания.

Пример: В лабораторной работе предложено 24 варианта. Последние номера в зачетках студентов А и Б соответственно 30 и 47.

Студент А выполняет задание $30 \% 24 + 1 = 7$, а студент Б: $47 \% 24 + 1 = 24$.

Задания к лабораторным работам могут изменяться преподавателем. Актуальные задания в электронном виде можно получить на кафедре или у преподавателя. Объем электронной версии лабораторных работ составляет около 3-х Мб.

Рекомендуемая литература

1. Автостопом по Python, 2017, Kenneth Reitz, Tanya Schlusse
2. Автоматизация рутинных задач с помощью Python, 2016, Albert Sweigart
3. Простой Python. Современный стиль программирования, 2016, Bill Lubanovic
4. Data Structures and Algorithms in Python, 2013, Michael Goodrich, Roberto Tamassia, Michael Goldwasser
5. Hacking Secret Chiphers with Python, 2013, Al Sweigart
6. Python на примерах. Практический курс по программированию., 2016, Алексей Васильев
7. Изучаем Python, 4-е издание, 2011, Mark Lutz
8. Программирование на Python для начинающих, 2015, Mike McGrath
9. Python. К вершинам мастерства, 2015, Luciano Ramalho
10. Python и анализ данных, 2015, Wes McKinney
11. Программируем на Python, 2014, Michael Dawson
12. Beginning Game Development with Python and Pygame, 2007, Will McGugan
13. Python 3 и PyQt. Разработка приложений, 2012, Николай Прохоренок
14. Python Algorithms: Mastering Basic Algorithms in the Py..., 2010, Magnus Lie Hetland
15. Pro Python System Administration, 2010, Rytis Sileika
16. Python на практике, 2014, Mark Summerfield
17. Python for Data Analysis, 2012, Wes McKinney
18. Python. Карманный справочник, 5-е издание, 2014, Mark Lutz
19. Gray Hat Python: Programming for hackers and reverse en..., 2009, Justin Seitz
20. Программируем коллективный разум, 2008, Toby Segaran
21. Программирование на Python 3. Подробное руководство, 2009, Mark Summerfield
22. Практикум по алгоритмизации и программированию на Python, 2010, И. А. Хахаев
23. Изучаем Python, 3-е издание, 2009, Mark Lutz
24. Django. Разработка веб-приложений на Python, 2009, Jeff Forcier, Paul Bissex, Wesley Chun
25. Django. Подробное руководство, 2-е издание., 2010, Adrian Holovaty, Jacob Kaplan-Moss
26. Структура и интерпретация компьютерных программ, 2006, Harold Abelson, Gerald Jay Sussman
27. Программист-прагматик. Путь от подмастерья к мастеру, 2007, Andrew Hunt, David Thomas
28. Совершенный код, 2010, Steve McConnell,3
29. Python. Подробный справочник, 2010, David Beazley
30. Violent Python: A Cookbook for Hackers, Forensic Analys..., 2013, T.J. O'Connor
31. Foundations of Python Network Programming, 2010, Brandon Rhodes, John Goerze,5
32. Программирование на Python, 4-е издание (2 тома), 2011, Mark Lutz
33. Practical Django Projects, 2008, James Bennett
34. Python Cookbook, 3rd Edition, 2013, David Beazley, Brian K. Jones

Лабораторная работа №1: "Линейные программы"

Цель работы

Дать студентам практический навык в подготовке простой программы и в записи математических выражений на языке программирования Python.

Постановка задачи

Напишите программу для расчета по формулам. Предварительно подготовьте тестовые примеры с помощью калькулятора или электронной таблицы Excel.

$$1) \quad y = \operatorname{tg}^2\left(\frac{x^2}{2} - 1\right) + \frac{2\cos(x - \pi/6)}{1/2 + \sin^2 \alpha};$$
$$2) \quad y = \frac{\log_{(3+\sin(x))}(3-\cos(\pi/4+2x))}{1+\operatorname{tg}^2(2x/\pi)}.$$

Теоретическое введение

При вычислении подобных выражений необходимо анализировать область допустимых значений аргументов, которые используются в выражении. Так, например, знаменатель дроби может получить нулевое значение и программа прервётся по ошибке деления на ноль. Необходимо учитывать и допустимый диапазон аргументов используемых функций. Так, основание логарифма должно быть больше нуля и не равняться единице, а логарифмируемая функция должна быть больше нуля.

Внимательно следует относиться к выражению, в котором, например, выполняется извлечение квадратного корня или, в общем случае, возведение в степень, показатель которой является не целым числом. В этом случае для вычисления используется логарифмирование, и для отрицательного основания степени возникнет ошибка, которая так же приведёт к прерыванию работы программы.

В наших примерах знаменатели $1/2 + \sin^2 \alpha$ и $1 + \operatorname{tg}^2(2x/\pi)$ всегда неравны нулю. Кроме этого, во втором примере, основание логарифма $3 + \sin(x)$ не отрицательно и не равно единице, а логарифмируемая функция $3 - \cos(\pi/4 + 2x)$ всегда больше нуля.

Для математических вычислений в Python имеются как встроенные, так и дополнительные функции и методы. Для применения дополнительных математических функций необходимо использовать модуль `math`, который подключается с помощью инструкции:

```
import math
```

либо

```
from math import *
```

В первом случае функции рассматриваются как методы объекта `math` и должны записываться так:

```
import math
print(math.sin(math.pi/4))
print(math.sqrt(2)/2)
```

Во втором случае вызов функции может быть сделан в более привычной для нас форме:

```
from math import *
print(sin(pi/4))
```

```
print(sqrt(2)/2)
```

Вместе с тем, такой способ импорта может нарушить пространство имен программы, поскольку может возникнуть конфликт между именами переменных, которые использует программист и именами импортируемых функций.

При импорте можно ограничиться только необходимыми функциями, например:

```
from math import (pi, sin, cos,
                  tan, log)
```

В этом примере демонстрируется способ импорта необходимых функций, и способ размещения инструкции на нескольких строках. Такие функции так же можно использовать в привычной для нас манере.

Набор функций и методов, реализованных в Python 3, приводится в Таблице 1, см. Приложение 1. Больше информации о функциях модуля math можно получить из документации или в сети Интернет.

В тех случаях, когда в языке программирования нужная функция отсутствует, ее можно написать, либо вычислить, используя известные формулы. Например,

$$\operatorname{ctg}(x) = 1/\tan(x) = \cos(x)/\sin(x).$$

Имена переменных следует выбирать тщательнее и использовать либо принятые в математике или физике символы, либо фразы, отражающие назначение переменной.

Например, символ α можно заменить символом a , который соответствует требованиям языка. Значение цвета можно хранить в переменной Colour, а объема в переменной Volume или Capacity.

Обращайте внимание на цветовую раскраску переменной. Она должна быть черной.

Для решения задания потребуется вводить и выводить данные. В нашем случае это числа целого или вещественного типа.

Ввод данных

Ввод данных можно выполнить с клавиатуры функцией `input()`:

```
m = input([str])
```

При этом на экран будет выведена строка `str`, а переменная `m` получит значение строкового типа, введеное пользователем. Строковый тип может быть преобразован, например, к типу `int` или `float`, если введеное значение - число.

Для ввода нескольких значений можно воспользоваться методом `split()`, который позволяет разбить строку на подстроки. Например, для ввода значений параметра `a` и переменной `x` можно поступить так:

```
a, x = input('Введите данные (a, x): ').split()
a = float(a)
x = float(x)
```

Используемый разделитель указывается в качестве параметра метода `split()`. Если разделитель не указан, то им будет пробел. При вводе десятичного числа целая часть отделяется от дроби точкой.

Если пользователь не ввел данные (просто нажал Enter) или вместо цифр и точки ввел недопустимые символы, например буквы, программа завершится аварийно. Исключительная ситуация, которая при этом возникает, может быть обработана с помощью инструкции `try`. Более подробно об этом следует прочитать, например, в [1].

В следующем примере пользователь должен ввести первое число целого типа, а второе – вещественного. Если ввод будет неправильным, то возникнет исключительная ситуация и управление программой будет передано в блок `except`. В этом блоке можно

предусмотреть возможные ситуации и принять необходимое решение, например, заставить пользователя правильно ввести данные.

```
import sys, traceback

while True:
    try:
        a = x = ""
        a, x = input('Ввод данных (a: int, x: float): ') \
            .split()
        a = int(a)
        x = float(x)
    except ValueError:
        if a == "" and x == "":
            a = x = 0
            break
        print("a - int, x - float. Пример: 3 4.5")
    print(a, x)
```

В теле "вечного" цикла, в блоке `try`, инициализируются две переменные, а затем следует инструкция для ввода данных. При ошибочном вводе числа, например, вместо целого – вещественное, или вместо цифры – буква, возникнет исключительная ситуация `ValueError`. Управление будет передано в модуль `except`, где, в условном операторе, проверяется, были ли введены данные. Если был нажат Enter, то переменные получат нулевое значение и управление будет передано инструкции, которая следует за циклом. Если ввод данных был сделан, то исключительная ситуация возникла при преобразовании типов данных. В этом случае выдается предупреждающее сообщение, и управление передается в начало цикла (ввод должен быть повторен).

Замечание: Если не выполнить инициализацию переменных перед инструкцией `input()`, то при возникновении исключительной ситуации (при вводе нажат только Enter), управление будет передано в модуль `except`, где возникнет новая исключительная ситуация в условном операторе `if`: переменная `a` неопределена.

Вывод данных

Вывод данных на экран монитора может быть выполнен функцией `print()`. Эта функция позволяет выполнять вывод как в Си-подобном формате, так и с использованием форматной строки.

Следующие примеры демонстрируют, как можно форматировать вывод.

```
for x in range(1,11):
    print('%2d %3d %.2f' % (x, x*x, x*x*x))

print("{0:.2f} {1:.2f} {2:.4f}".format(a, x, y))
```

Буква в формате числа определяет тип выводимого числа. Так, `d` – это целый тип, `f` – вещественное число. Число в формате означает то число позиций, которое будет использовано для вывода числа. Для вещественного числа указывается, после точки, количество выводимых десятичных знаков.

В первом примере использован Си-подобный формат, в котором формат числа начинается с процента "%", а во втором примере используется форматная строка, в которой формат числа задается в фигурных скобках.

Обратите внимание на то, что сами форматные строки начинаются и завершаются одиночной или двойной кавычкой. В Python допускаются оба вида кавычек для выделения строки. Важно только что бы начало и конец были одинаковыми.

Так же следует понимать, что в промежутках между символами форматирования могут находиться и другие символы или стока:

```
print('x=%2d x^2=%3d x^3=%7.2f' % (x, x*x, x*x*x))
print("a={0:.2f} x={1:.2f} y={2:.4f}".format(a, x, y))
```

Использование форматных строк позволяет делать вывод данных более внятным. За более подробной информацией обращайтесь к учебникам или Интернет.

Решение задания

Вернемся к нашим примерам и запишем их, используя правила языка Python:

```
1. y = tan(x**2/2-1)**2+(2*cos(x-pi/6))/(1/2+sin(a)**2)
```

Второе выражение представим в виде двух:

```
2. tmp = log(3-cos(pi/4+2*x), 3+sin(x))/(1+tan(2*x/pi)**2)
y = pow(2, tmp)
```

Описание алгоритма

Для вычислений необходимо обеспечить ввод двух переменных x и a . Поскольку по условиям задачи их тип и точность представления не заданы, выберем для них вещественный тип (*float*). Для оптимизации записи выражения используем промежуточную переменную *tmp*.

1. Ввести значения a и x , преобразовать к типу *float*.
2. Вычислить выражение 1.
3. Вывести результат вычисления.
4. Вычислить значение переменной *tmp*;
5. Вычислить выражение 2.
6. Вывести результат вычисления.

Листинг программы

```
# -*- coding: cp1251 -*-
from math import *
a = float(input('Введите параметр a: '))
x = float(input('Введите значение переменной x: '))
y=tan(x**2/2-1)**2+(2*cos(x-pi/6))/(1/2+sin(a)**2)
print("{0:.2f} {1:.2f} {2:.4f}".format(a, x, y))
tmp=log(3-cos(pi/4+2*x), 3+sin(x))/(1+tan(2*x/pi)**2)
y=pow(2, tmp)
print("{0:.2f} {1:.4f}".format(x, y))
```

Результаты тестирования программы

a	x	Первое выражение		Второе выражение
		Калькулятор	Программа	
-2	-2	1.196954	1.1970	1.1184
0	-2	-0.834654	-0.8347	1.1184
0	0	5.889618	5.8896	1.6880
2	0	3.730931	3.7309	1.6880
1.5	0.5	2.771242	2.7712	1.7955
4	3	-1.326566	-1.3266	1.0517

Примечание: Эта таблица оформлялась в текстовом редакторе вручную.

Задания к лабораторной работе №1 "Линейные программы"

Напишите программу для расчета по двум формулам. Подготовьте не менее пяти тестовых примеров. Предварительно выполните вычисления с использованием калькулятора или Excel (результаты вычисления по обеим формулам должны совпадать). Используйте не менее пяти значений переменных.

Отсутствующие в языке функции выражите через имеющиеся.

$$1. \quad z_1 = 2\sin^2(3\pi - 2\alpha) \cdot \cos^2(5\pi + 2\alpha);$$

$$z_2 = \frac{1}{4} - \frac{1}{4}\sin\left(\frac{5}{2}\pi - 8\alpha\right).$$

$$2. \quad z_1 = \cos\alpha + \sin\alpha + \cos 3\alpha + \sin 3\alpha;$$

$$z_2 = 2\sqrt{2}\cos\alpha \cdot \sin\left(\frac{\pi}{4} + 2\alpha\right).$$

$$3. \quad z_1 = \frac{\sin 2\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha + 1 - 2\sin^2 2\alpha}; \quad z_2 = 2\sin\alpha.$$

$$4. \quad z_1 = \frac{2 \cdot \cos\alpha \cdot \sin 2\alpha - \sin\alpha}{\cos\alpha - 2 \cdot \sin\alpha \cdot \sin 2\alpha}; \quad z_2 = \operatorname{tg} 3\alpha.$$

$$5. \quad z_1 = 1 - \frac{1}{4}\sin^2 2\alpha + \cos 2\alpha; \quad z_2 = \cos^2\alpha + \cos^4\alpha.$$

$$6. \quad z_1 = \cos\alpha + \cos 2\alpha + \cos 6\alpha + \cos 7\alpha;$$

$$z_2 = 4\cos\frac{\alpha}{2} \cdot \cos\frac{5}{2}\alpha \cdot \cos 4\alpha.$$

$$7. \quad z_1 = \cos^2\left(\frac{3}{8}\pi - \frac{\alpha}{4}\right) - \cos^2\left(\frac{11}{8}\pi + \frac{\alpha}{4}\right); \quad z_2 = \frac{\sqrt{2}}{2}\sin\frac{\alpha}{2}.$$

$$8. \quad z_1 = \cos^4 x + \sin^2 y + \frac{1}{4}\sin^2 2x - 1;$$

$$z_2 = \sin(y+x) \cdot \sin(y-x).$$

$$9. \quad z_1 = (\cos\alpha - \cos\beta)^2 - (\sin\alpha - \sin\beta)^2;$$

$$z_2 = -4\sin^2\frac{\alpha - \beta}{2} \cdot \cos(\alpha + \beta).$$

$$10. \quad z_1 = \frac{\sin\left(\frac{\pi}{2} + 3\alpha\right)}{1 - \sin(3\alpha - \pi)}; \quad z_2 = \operatorname{ctg}\left(\frac{5}{4}\pi + \frac{3}{2}\alpha\right).$$

$$11. \quad z_1 = \frac{1 - 2\sin^2\alpha}{1 + \sin 2\alpha}; \quad z_2 = \frac{1 - \operatorname{tg}\alpha}{1 + \operatorname{tg}\alpha}.$$

$$12. \quad z_1 = \frac{\sin 4\alpha}{1 + \cos 4\alpha} \cdot \frac{\cos 2\alpha}{1 + \cos 2\alpha}; \quad z_2 = \operatorname{ctg}\left(\frac{3}{2}\pi - \alpha\right).$$

$$13. \quad z_1 = \frac{\sin\alpha + \cos(2\beta - \alpha)}{\cos\alpha - \sin(2\beta - \alpha)}; \quad z_2 = \frac{1 + \sin 2\beta}{\cos 2\beta}.$$

$$14. \quad z_1 = \frac{\cos \alpha + \sin \alpha}{\cos \alpha - \sin \alpha}; \quad z_2 = \operatorname{tg} 2\alpha + \sec 2\alpha.$$

$$15. \quad z_1 = \frac{\sqrt{2b+2\sqrt{b^2-4}}}{\sqrt{b^2-4+b+2}}; \quad z_2 = \frac{1}{\sqrt{b+2}}.$$

$$16. \quad z_1 = \frac{x^2+2x-3+(x+1)\cdot\sqrt{x^2-9}}{x^2-2x-3+(x-1)\cdot\sqrt{x^2-9}}; \quad z_2 = \sqrt{\frac{x+3}{x-3}}.$$

$$17. \quad z_1 = \frac{\sqrt{(3m+2)^2-24m}}{3\sqrt{m}-\frac{2}{\sqrt{m}}}; \quad z_2 = \sqrt{m}.$$

$$18. \quad z_2 = \left(\frac{a+2}{\sqrt{2a}} - \frac{a}{\sqrt{2a}+2} + \frac{2}{a-\sqrt{2a}} \right) \cdot \frac{\sqrt{a}-\sqrt{2}}{a+2}; \quad z_2 = \frac{1}{\sqrt{a}+\sqrt{2}}.$$

$$19. \quad z_1 = \left(\frac{1+a+a^2}{2a+a^2} + 2 - \frac{1-a+a^2}{2a-a^2} \right) \cdot (5-2a^2); \quad z_2 = \frac{4-a^2}{2}.$$

$$20. \quad z_1 = \frac{(m-1)\sqrt{m}-(n-1)\sqrt{n}}{\sqrt{m^3n}+nm+m^2-m}; \quad z_2 = \frac{\sqrt{m}-\sqrt{n}}{m}.$$

$$21. \quad z_1 = \frac{1+\sin^4(-a)-\cos^4(-a)}{\cos^2 a}; \quad z_2 = 2\operatorname{tg}^2 a.$$

$$22. \quad z_1 = \sqrt{\frac{1-\sin(\frac{\pi}{2}-a)}{2}} + \sqrt{\frac{1+\cos(2\pi-a)}{2}}; \quad \pi < a < \frac{3\pi}{2}.$$

$$z_2 = \sin \frac{a}{2} - \cos \frac{a}{2}; \quad \pi < a < \frac{3\pi}{2}.$$

$$23. \quad z_1 = \left(\frac{1+6ac}{a^3-8c^3} - \frac{1}{a-2c} \right) : \left(\frac{1}{a^3-8c^3} - \frac{1}{a^2+2ac+4c^2} \right);$$

$$z_2 = 1-2c+a.$$

$$24. \quad z_1 = \frac{\frac{1}{a}-\frac{1}{b+c}}{\frac{1}{a}+\frac{1}{b+c}} \cdot \left(1 + \frac{b^2+c^2-a^2}{2bc} \right) : \frac{a-b-c}{abc}; \quad z_2 = \frac{a-b-c}{2} \cdot a.$$

$$25. \quad z_1 = \frac{\sin^2(\pi+a)+\sin^2(\frac{\pi}{2}+a)}{\cos(\frac{3\pi}{2}+a)} \operatorname{ctg}(1.5\pi-a); \quad z_2 = \frac{1}{\cos a}.$$

$$26. \quad z_1 = \frac{\operatorname{tg}(x-\frac{\pi}{2})\cos(\frac{3\pi}{2}+x)-\sin^3(3.5\pi-x)}{\cos(x-0.5\pi)\operatorname{tg}(1.5\pi+x)}; \quad z_2 = \sin^2 x.$$

$$27. \quad z_1 = a^{\sqrt{\log_a b}} - b^{\sqrt{\log_b a}} + \operatorname{tg}(ab+3\pi/2); \quad z_2 = \operatorname{tg}(ab+\frac{3}{2}\pi);$$

$$28. \quad z_1 = \frac{\sin^2 a - \operatorname{tg}^2 a}{\cos^2 a - \operatorname{ctg}^2 a}; \quad z_2 = \operatorname{tg}^6 a;$$

$$29. \quad z_1 = \frac{1 - 2 \sin^2 a}{2 \operatorname{ctg}(\frac{\pi}{4} + a) \cos^2(\frac{\pi}{4} - a)} + e^a; \quad z_2 = 1 + e^a;$$

$$30. \quad z_1 = \frac{\sin^4 a + 2 \sin a \cos a - \cos^4}{\operatorname{tg} 2a - 1}; \quad z_2 = \cos 2a;$$

Лабораторная работа №2: "Разветвляющиеся вычислительные процессы". Задание 1. Пример оформления.

Цель работы

Дать студентам практический навык в использовании условных операторов ветвления на языке программирования Python. Работа состоит из двух заданий.

Следующий материал представляет пример оформления лабораторной работы.

ФГБОУ ВО "МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ"

Лабораторная работа №2

Разветвляющиеся вычислительные процессы

Задание 1

Вариант № 1

по дисциплине:

Программирование

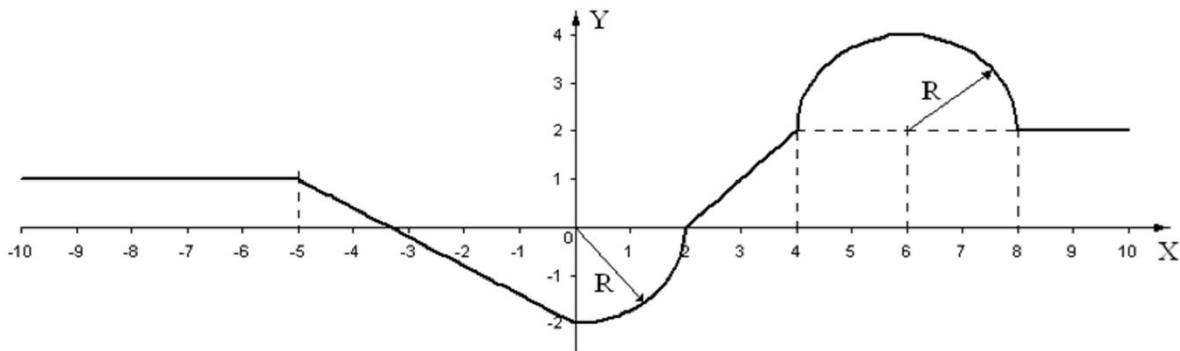
Выполнил
студент 1 курса
группы 171-361
Иванов И.И.

Проверил
_____ Никишиа И.Н.

МОСКВА 2018

Постановка задачи

Написать программу, которая по введённому значению аргумента вычисляет значение функции, заданной в виде графика.



Теоретическая часть

Для решения задачи использован оператор ветвления, который в языке Python имеет следующий вид:

```
if <Логическое выражение>:  
    <Блок – выполняется, если условие истинно>  
[elif <Логическое выражение>:  
    <Блок – выполняется, если условие истинно>  
]  
[else:  
    <Блок – выполняется, если все условия ложны>  
]  
<Блок> – это набор инструкций, которые выделяются одинаковым количеством пробелов (обычно четырьмя).
```

Для ввода данных используется инструкция `input()`, которая возвращает строку. Введенные значения должны быть преобразованы к числовому формату перед использованием в арифметических выражениях.

Вывод данных выполняется инструкцией `print()`, в которой можно использовать форматирование выводимых данных.

Функция представлена фрагментами прямых линий, описываемых уравнением $y = kx + b$ и дугами кругов, обобщенное уравнение которых: $(x - a)^2 + (y - b)^2 = R^2$. Неизвестные параметры, угол наклона и смещение прямой, а также координаты центра дуг, определим, используя данные из графика. В итоге функция примет вид:

$$y = \begin{cases} 1 & x < -5 \\ -\frac{3}{5}x - 2 & -5 \leq x < 0 \\ -\sqrt{4 - x^2} & 0 \leq x < 2 \\ \frac{x - 2}{2 + \sqrt{4 - (x - 6)^2}} & 2 \leq x < 4 \\ 2 & 4 \leq x < 8 \\ 2 & x \geq 8 \end{cases}$$

Функция определена на всём диапазоне $x \in (-\infty; +\infty)$. При этом, особых точек у неё нет.

Описание программы

Программа написана на алгоритмическом языке Python 3.6, реализована в среде ОС Windows 10 и состоит из частей, отвечающих за ввод данных, вычисление и представление данных на экране монитора.

Описание алгоритма

1. Ввести значение аргумента x и преобразовать его к типу float.
2. Определить, к какому интервалу из области определения функции оно принадлежит, и вычислить значение функции y по соответствующей формуле.
3. Вывести значение x и y .

Описание входных и выходных данных

Входные данные поступают с клавиатуры, а выходные - выводятся на монитор для просмотра. Входные и выходные данные имеют тип float.

Листинг программы (вариант 1)

```
# -*- coding: cp1251 -*-
from math import * # теперь можно так:
                  # print(sin(pi/4))
x = float(input('Введите значение x='))
if x < -5: y = 1
if x >=-5 and x<0: y = -(3/5)*x-2
if x >= 0 and x<2: y = -sqrt(4-x**2)
if x >= 2 and x<4: y = x-2
if x >= 4 and x<8: y = 2+sqrt(4-(x-6)**2)
if x >= 8: y = 2
print("X={0:.2f}      Y={1:.2f}".format(x, y))
```

Блок-схема алгоритма этого решения приведена в Приложении 1 (рис.1) к лабораторной работе.

Следует отметить, что в такой записи алгоритма проверка выполняется для всех условных операторов, в том числе и тех, которые следуют за вычислением. Так, например, если x равно -3, то выполнится второй оператор, но и во всех последующих операторах операция сравнения будет проведена. Число проверок можно сократить, если написать программу с использованием вложенных условных операторов.

Листинг программы (вариант 2)

```
# -*- coding: cp1251 -*-
from math import * # теперь можно так:
                  # print sin(pi/4)
x = float(input('Введите значение x='))
if x < -5:
    y = 1
elif x >=-5 and x<0:
    y = -(3/5)*x-2
elif x >= 0 and x<2:
    y = -sqrt(4-x**2)
elif x >= 2 and x<4:
    y = x-2
elif x >= 4 and x<8:
    y = 2+sqrt(4-(x-6)**2)
else: y = 2
```

```
print("X={0:.2f}      Y={1:.2f}".format(x, y))
```

Блок-схема алгоритма решения приведена в Приложении 2 (рис.2) к лабораторной работе.

Результат работы программы

```
Ведите значение аргумента: -6
X= -6.00      Y= 1
Ведите значение аргумента: -3.33
X= -3.33      Y= -0.00
Ведите значение аргумента: 6
X= 6.00       Y= 4.00
```

Список используемой литературы

1. Н.А. Прохоренок, В.А. Дронов, Python 3 и PyQt 5. Разработка приложений: СПб.: БХВ-Петербург, 2017
2. В.П. Рядченко, Методическое пособие по выполнению лабораторных работ.

Приложение 1
к лабораторной работе №2, задание 1
И.Х. Джанкёзова

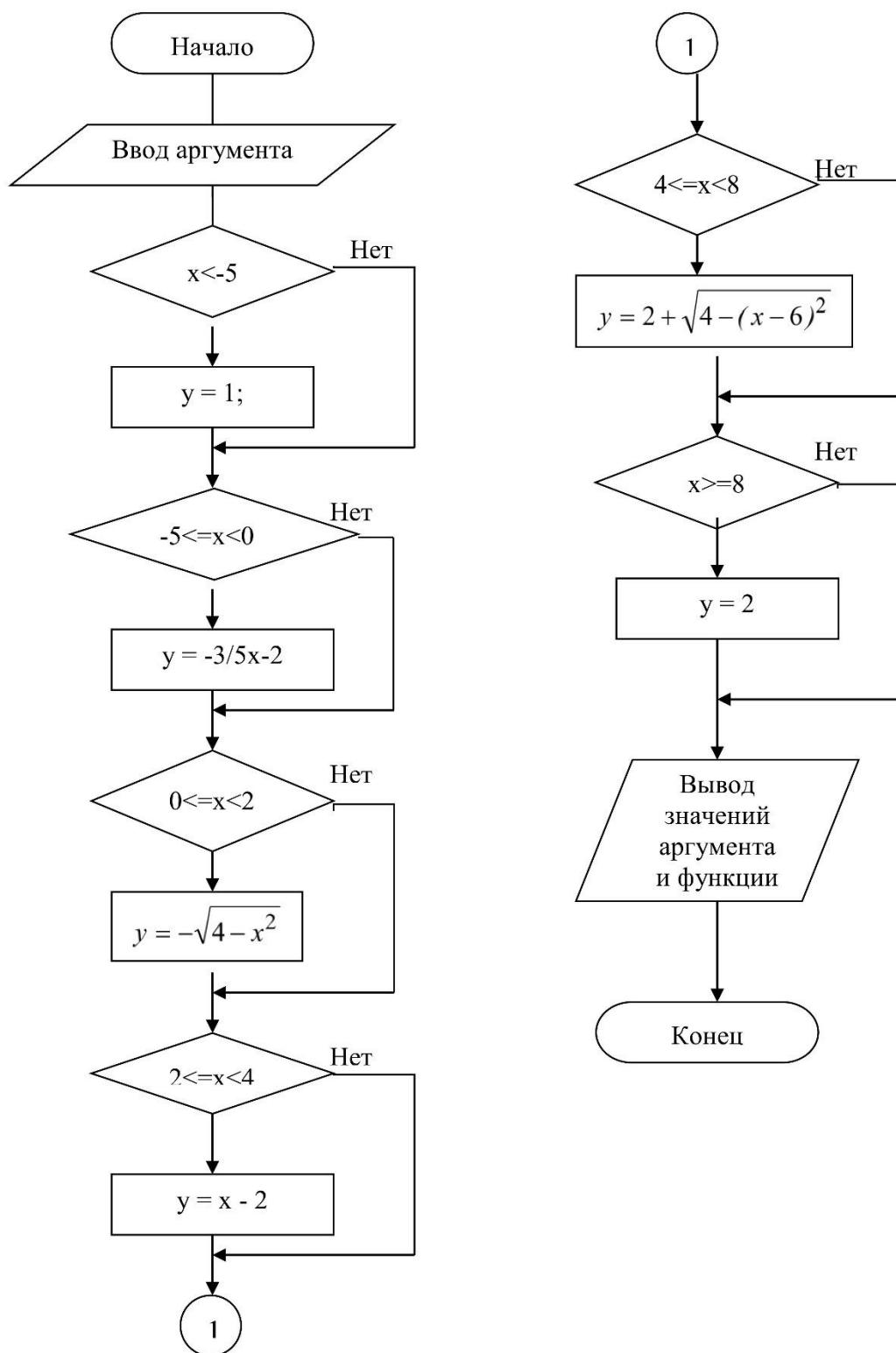


Рис.1 Блок-схема алгоритма программы Lab2_1a

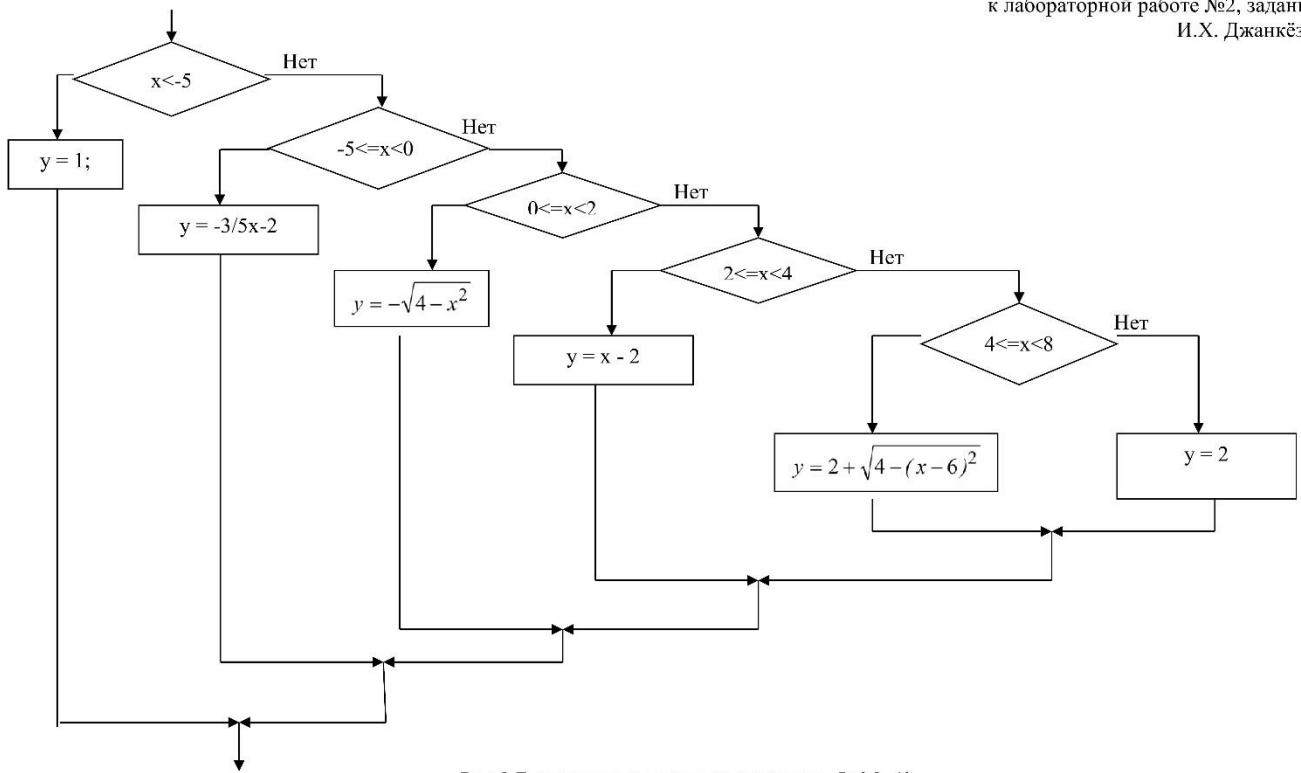
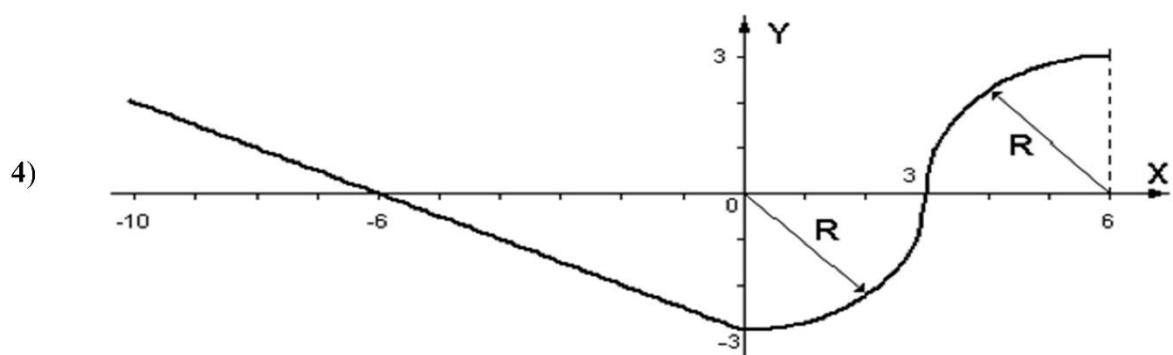
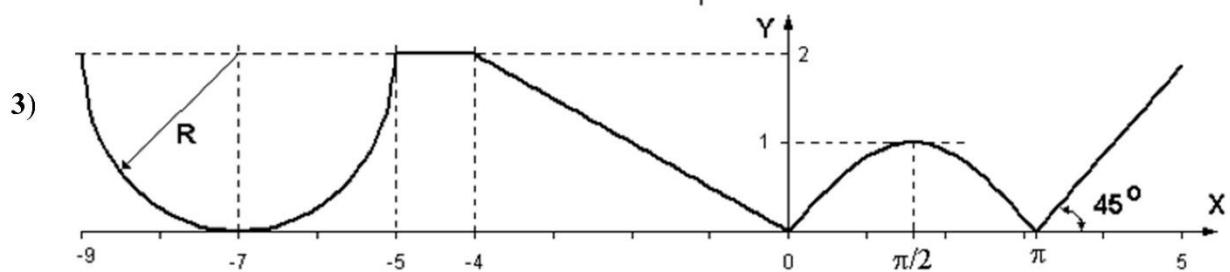
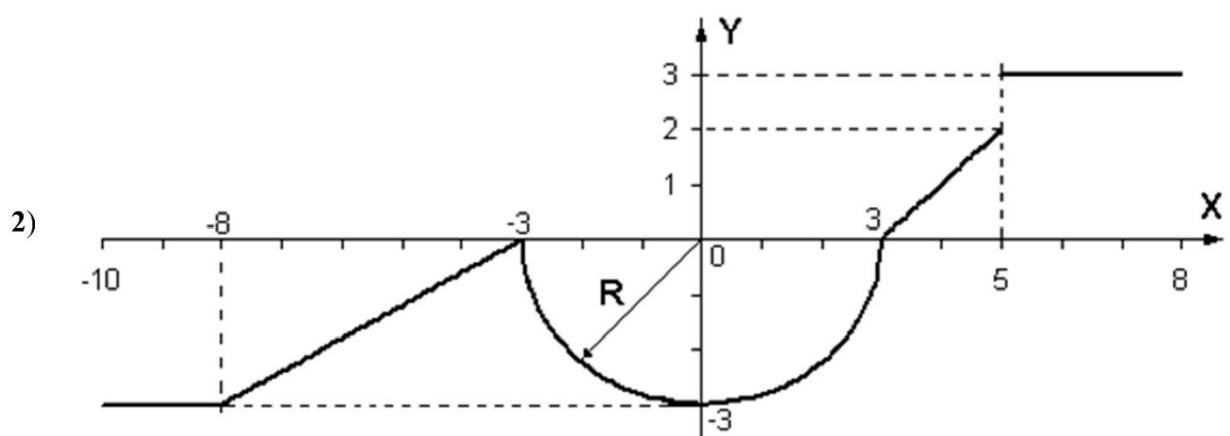
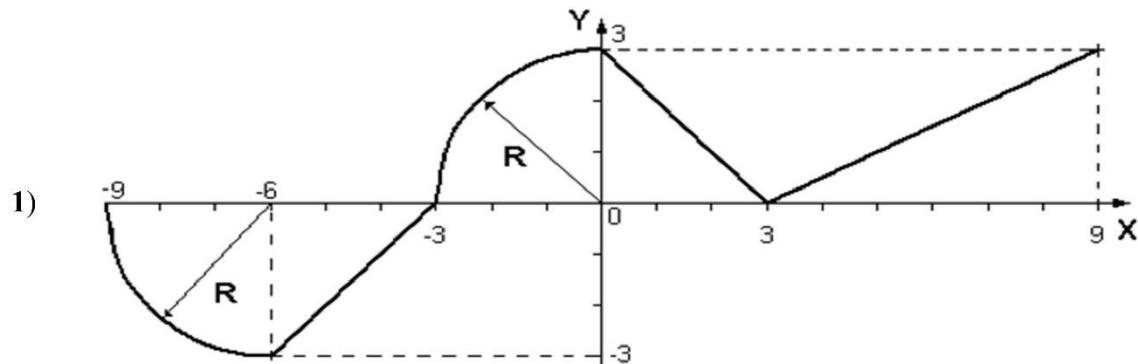
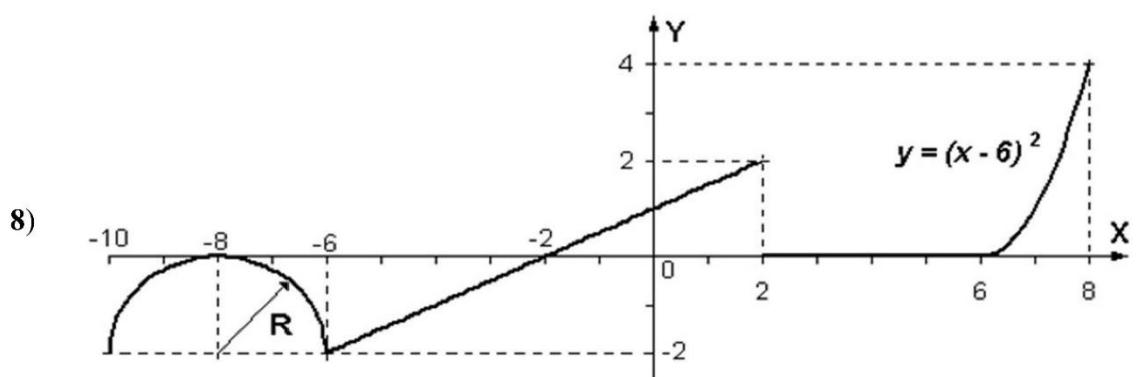
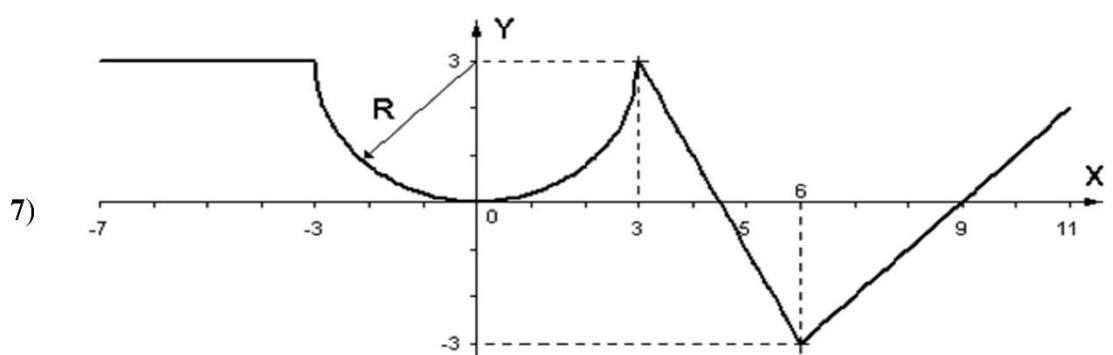
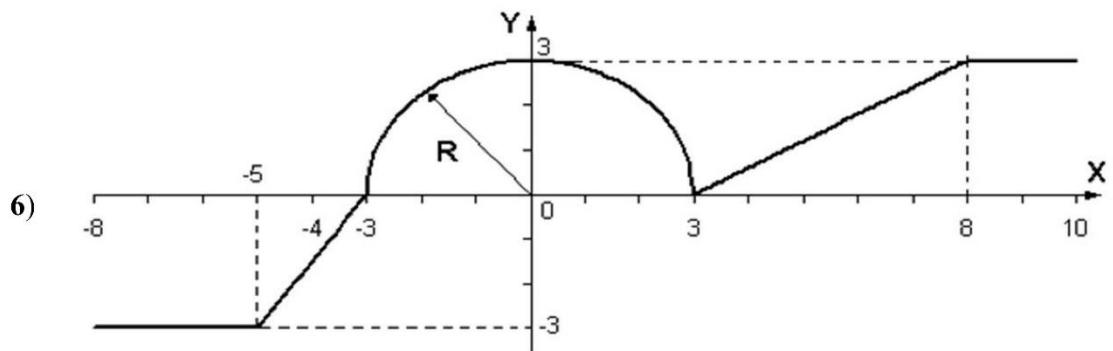
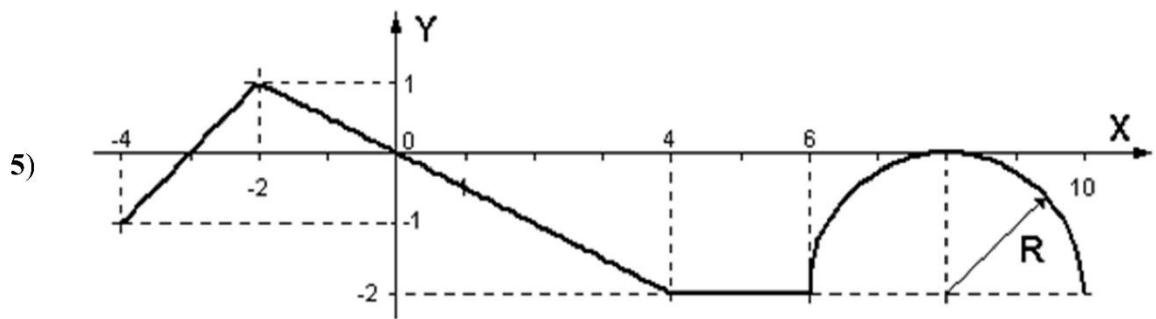


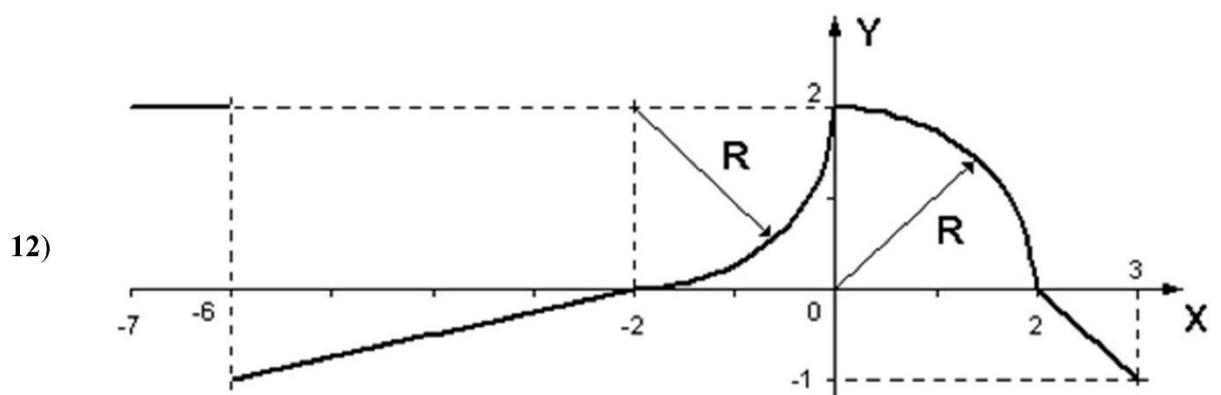
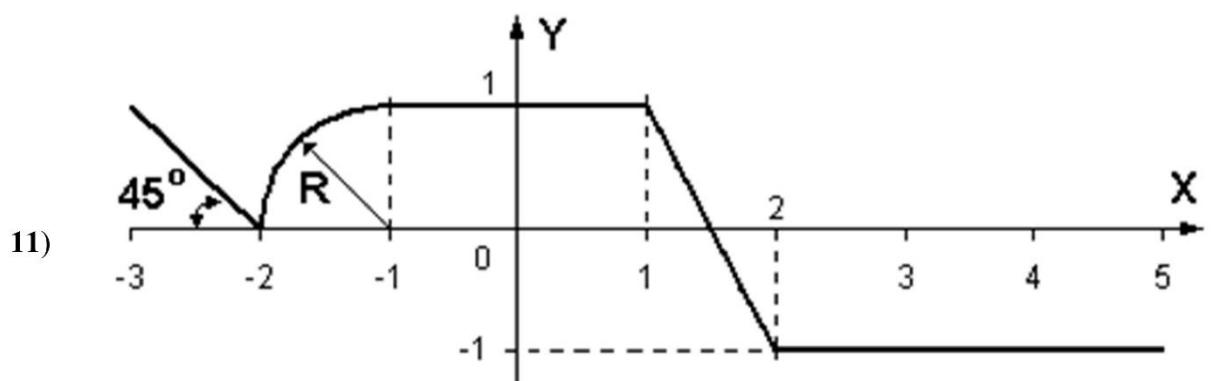
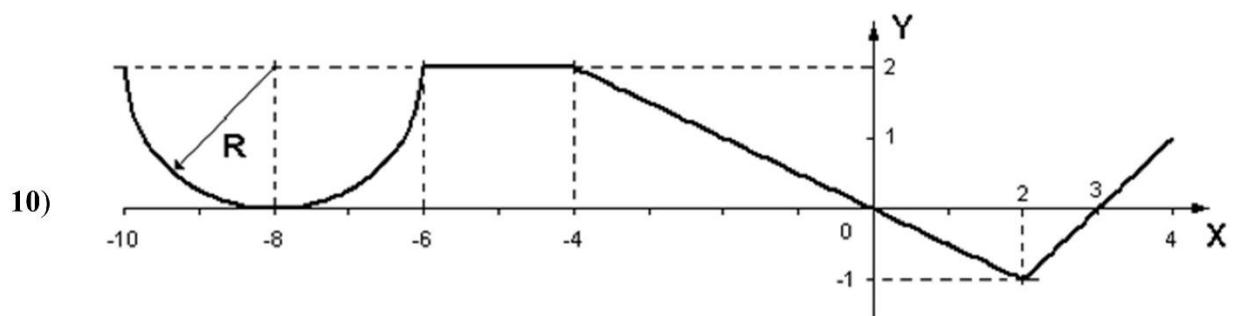
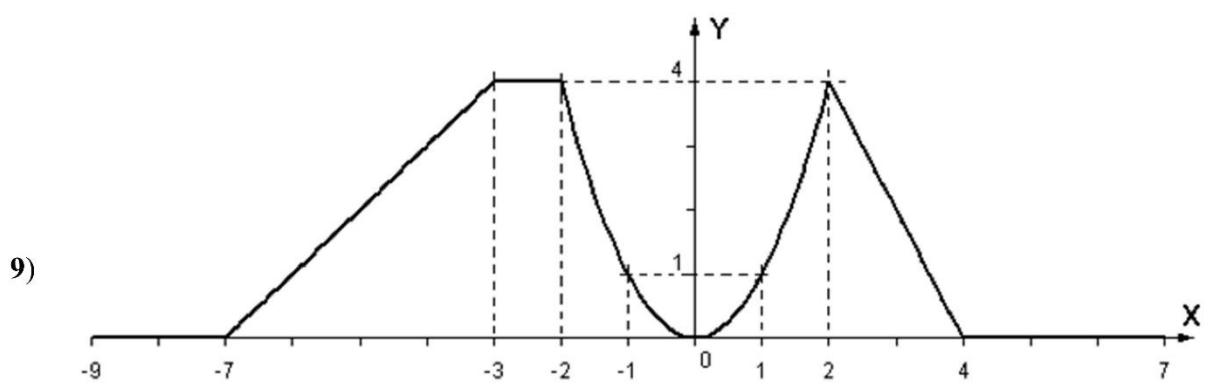
Рис.2 Блок-схема алгоритма программы Lab2_1b

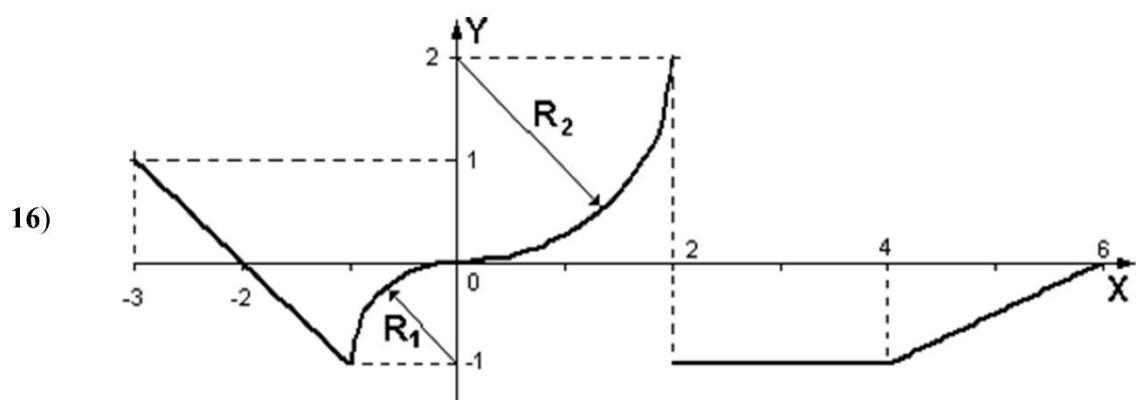
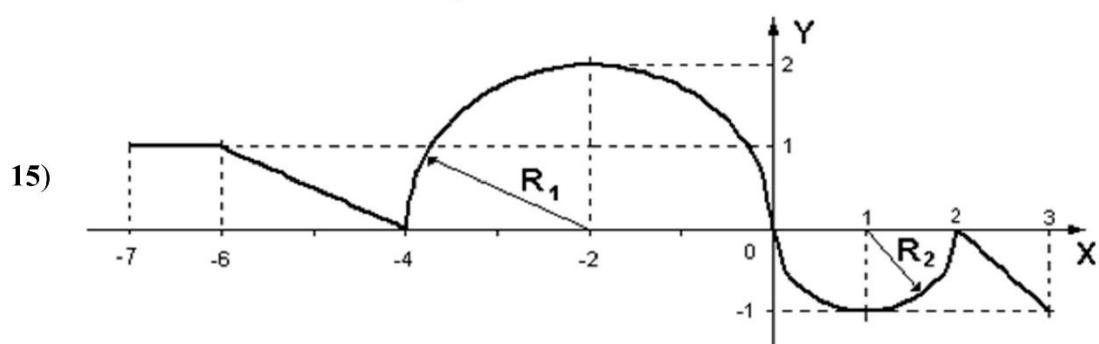
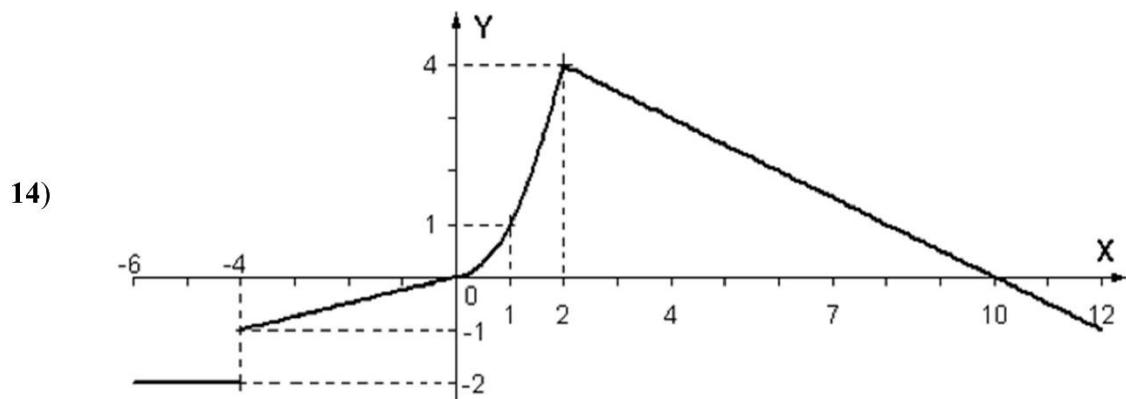
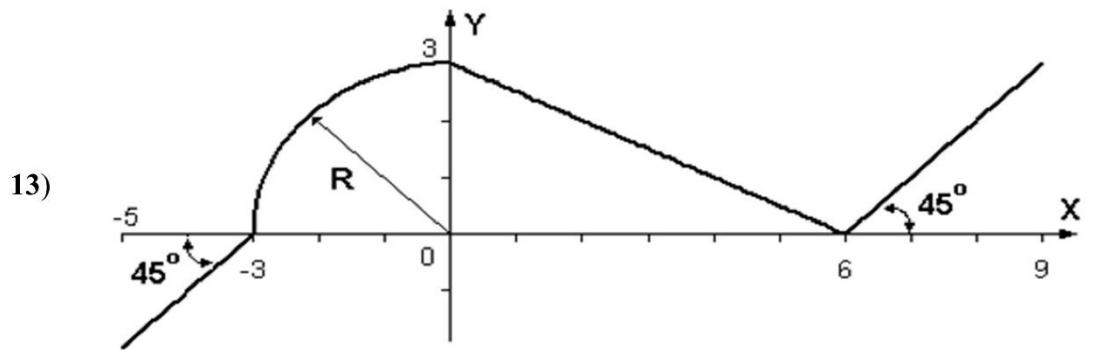
Задание к лабораторной работе №2 "Разветвляющиеся вычислительные процессы". Задание 1

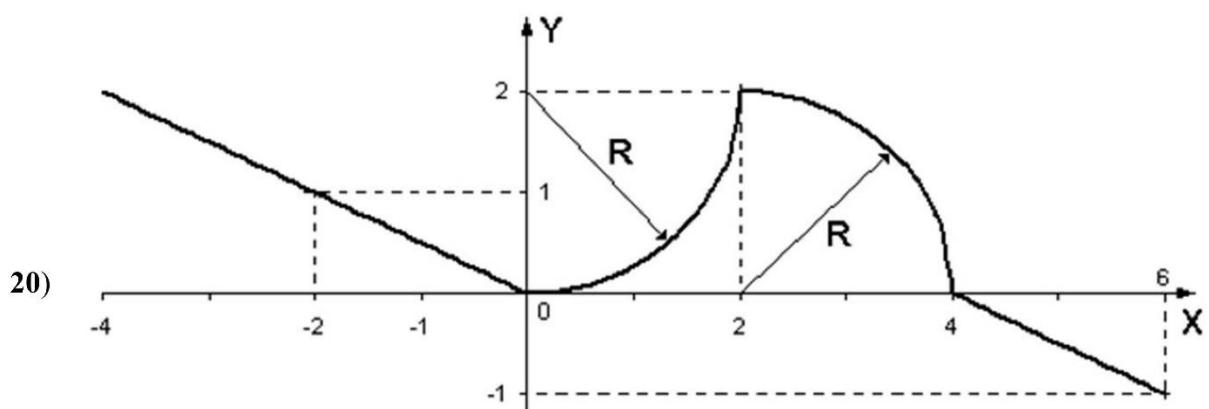
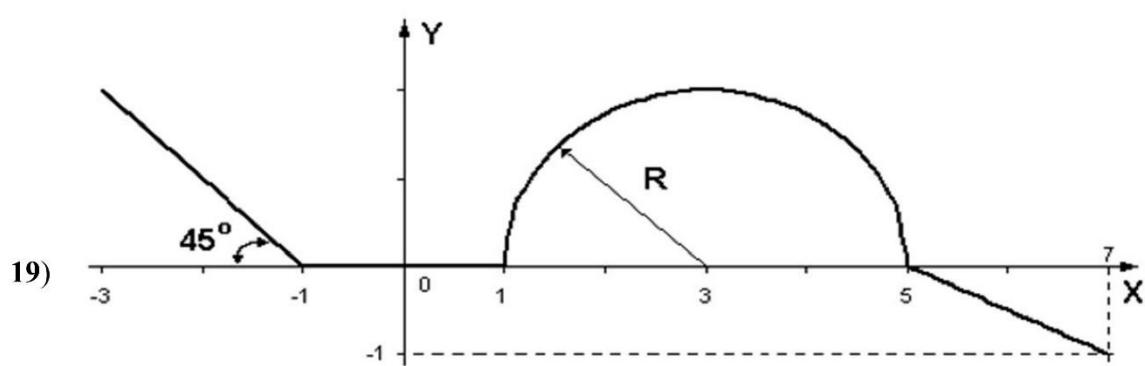
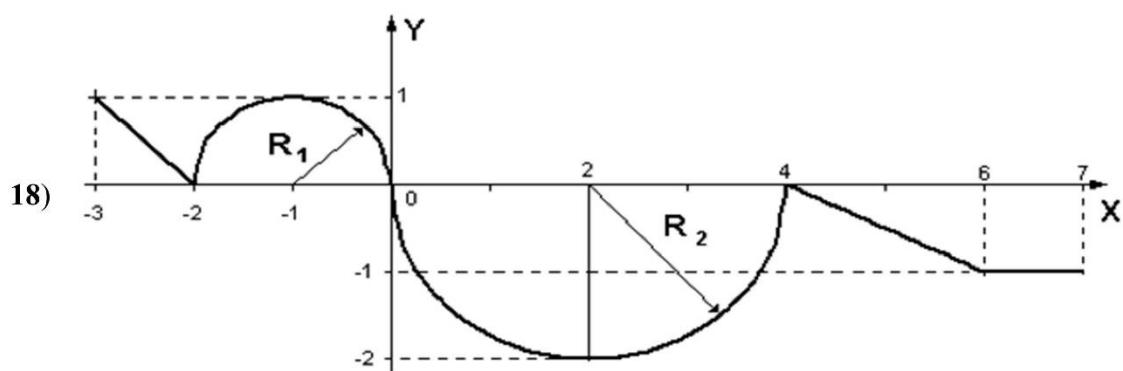
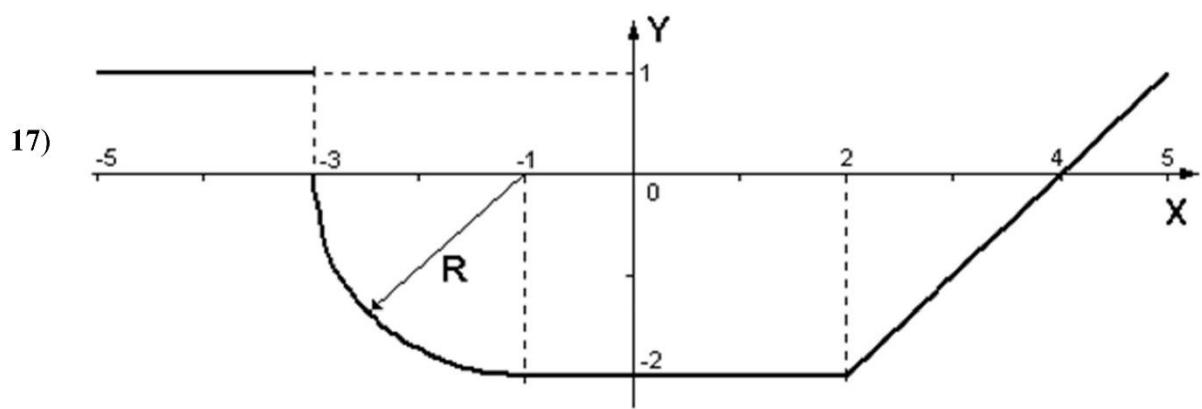
Написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графика. Параметры, необходимые для решения задания следует получить из графика и определить в программе.

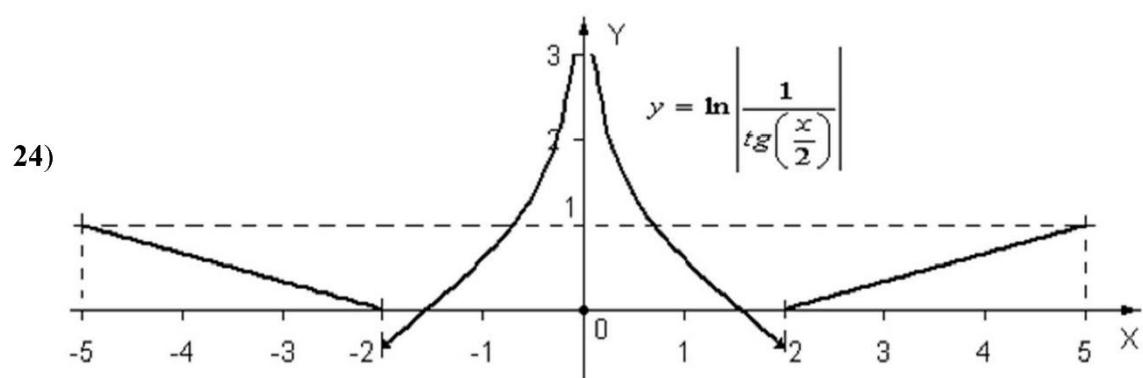
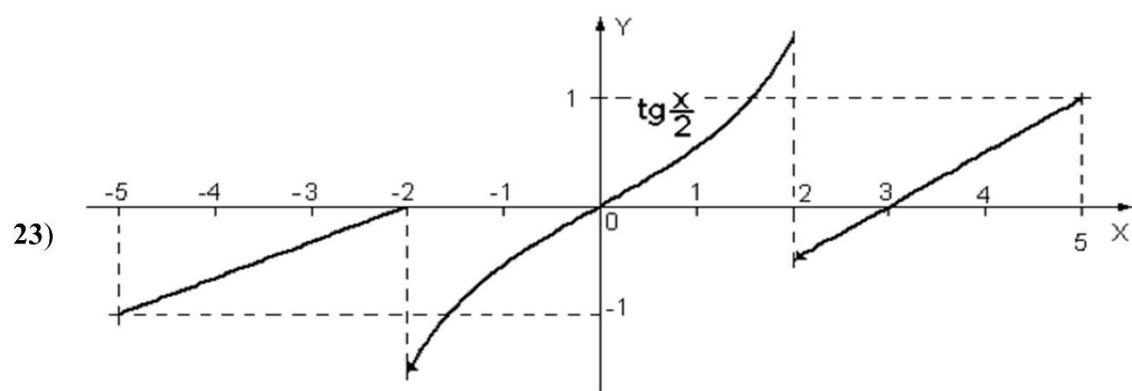
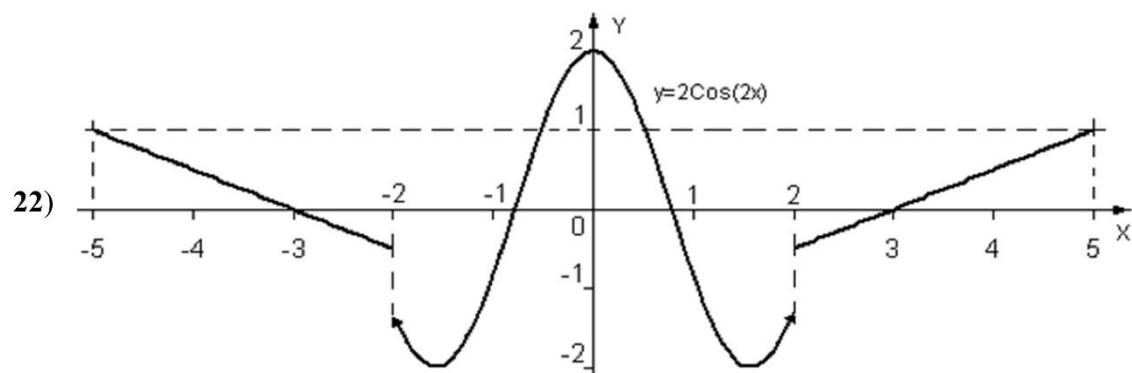
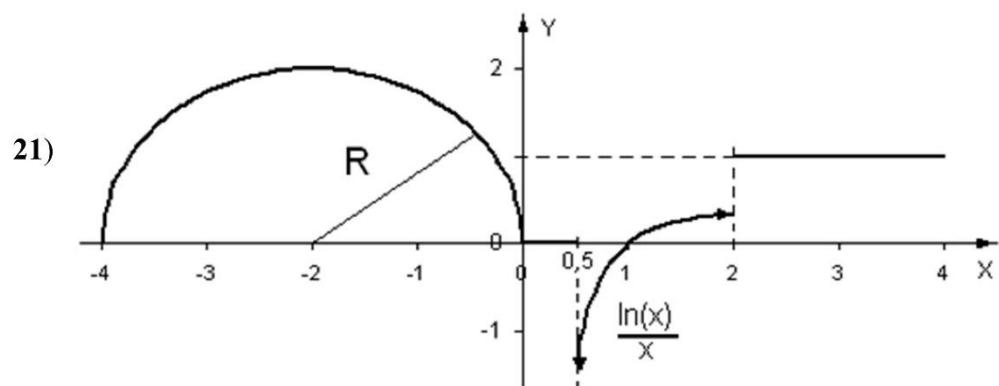


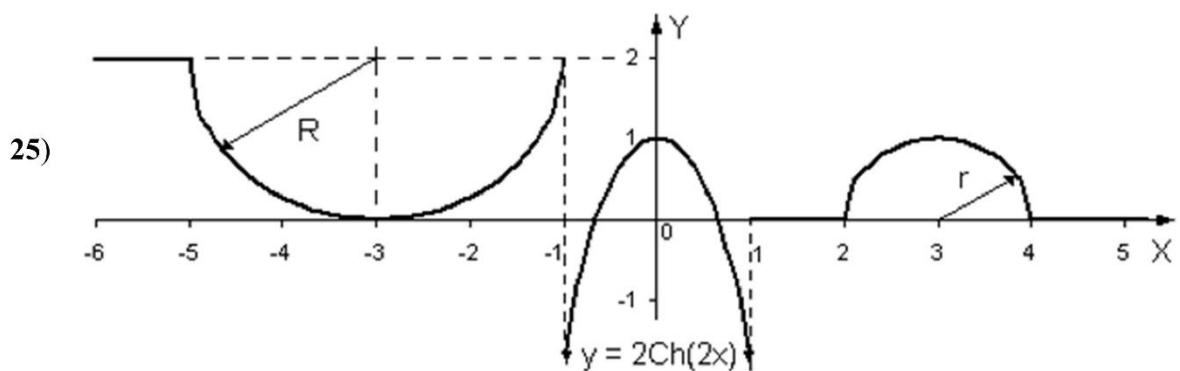




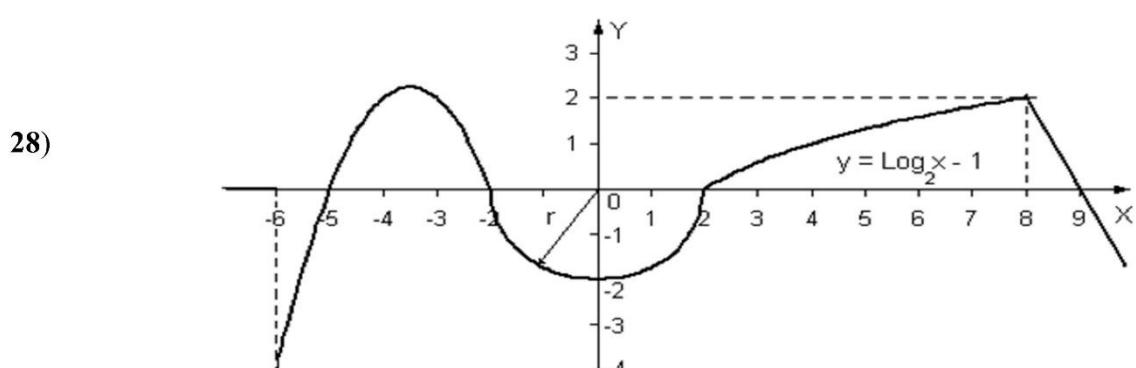
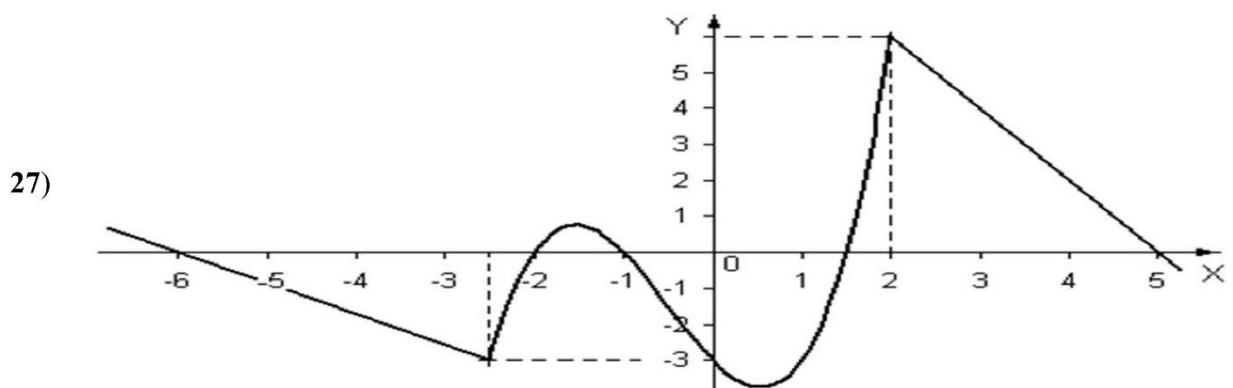
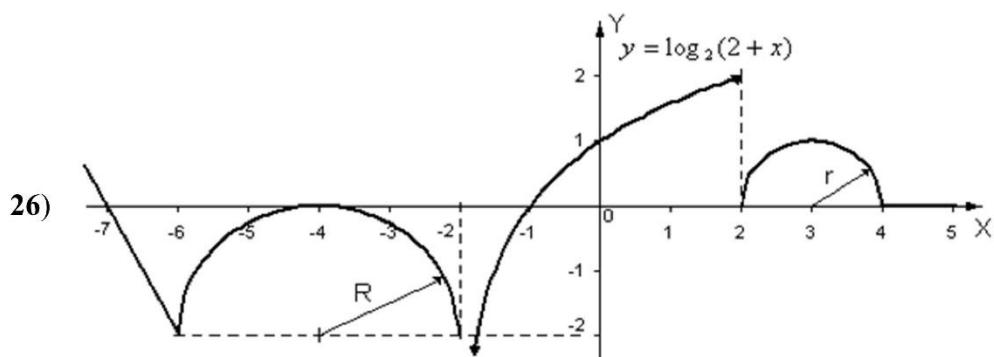




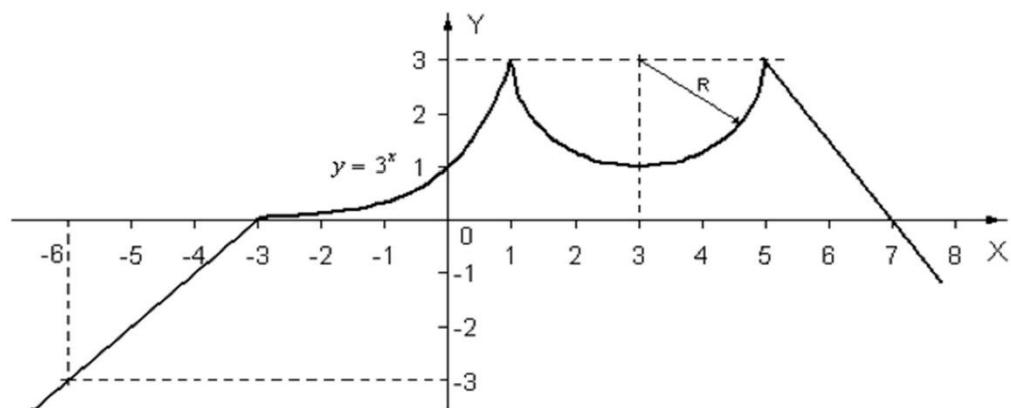




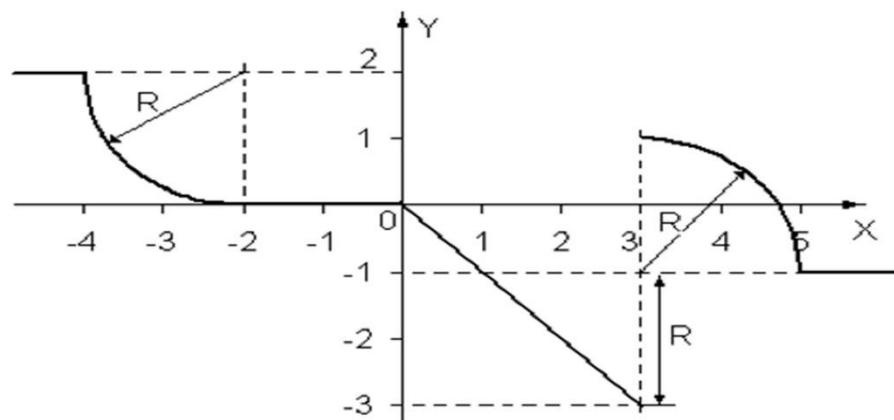
Гиперболический косинус может быть вычислен по формуле: $\text{Ch}(x) = \frac{1}{2}(e^x + e^{-x})$.



29)



30)

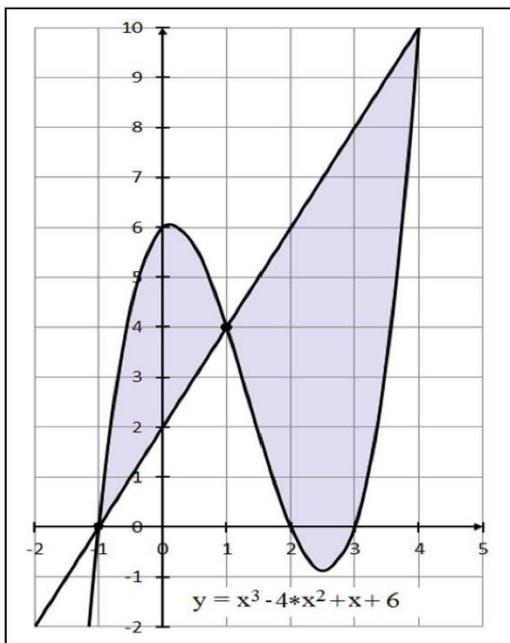


Дополнительное требование к заданию №30: Программа должна быть написана так, что бы решения получались при различных значениях R , вводимых с клавиатуры. Центр положения левой четверти окружности изменяется в соответствии с введённым радиусом, а правой - остаётся постоянным при $X = 3, Y = -1$.

Лабораторная работа №2: "Разветвляющиеся вычислительные процессы". Задание 2

Постановка задачи

Написать программу, которая определяет, попадает ли точка с заданными координатами в заштрихованную область. Точки на границе принадлежат области. Необходимые параметры получить из рисунка. Результат работы программы вывести в виде текстового сообщения: Попадает, Не попадает.



Теоретическое введение

Для решения задачи воспользуемся условным оператором:

```
if <Логическое выражение>:  
    <Блок – выполняется, если условие истинно>  
elif <Логическое выражение>:  
    <Блок – выполняется, если условие истинно>  
]  
else:  
    <Блок – выполняется, если все условия ложны>  
]  
<Блок> – это набор инструкций, которые выделяются одинаковым количеством пробелов (обычно четырьмя).
```

Нам важно правильно составить логическое выражение, параметрами которого будут значение координат точки (x, y) и уравнения линий.

Обмен с консолью выполняется стандартными функциями ввода/вывода: `input()` и `print()`.

Для решения задачи требуется уравнение прямой (уравнение кривой приведено на рисунке). Из рисунка получаем координаты двух точек, через которые проходит прямая линия (-1, 0) и (1, 4). Уравнение прямой линии в общем виде: $y = kx + b$. Подставим значения выбранных точек и вычислим параметры k и b . В результате вычислений найдем уравнение прямой: $y = 2x + 2$.

Описание алгоритма

1. Ввести координаты точки (x, y) и привести значения к типу float.
2. Выполнить проверку на попадание точки в заданную область.
3. Вывести результат в виде: "Точка x, y попадает в область." и "Точка x, y не попадает в область."

Описание входных и выходных данных

Входные данные - координаты точки, введенные пользователем. Тип данных и точность представления в задаче не заданы. Установим вещественный тип (float).

Выходные данные - сообщения, в текстовом виде, о попадании или непопадании точки в заданную область.

Тестовые примеры

X	Y	Результат
-1	0	Попадает
-0.5	-1	Не попадает
0	3	Попадает
1	4	Попадает
1	5	Не попадает
1.5	1	Не попадает
2	3	Попадает
2.5	-1	Не попадает
2.5	-0.3	Попадает
3.5	10	Не попадает

Листинг программы

```
# -*- coding: cp1251 -*-
from math import *
flag = 0
print('Введите координаты точки')
x = float(input('X='))
y = float(input('Y='))
if (x < -1) or (x > 4):
    flag = 0          #False
if (x>=-1) and (x<1) and (y>=2*x+2) \
    and (y<=x**3-4*x**2+x+6) or \
    (x>=1) and (x<=4) and (y>=x**3-4*x**2+x+6) \
    and (y<=2*x+2):
    flag = 1
else:
    flag = 0
print("Точка X={0: 6.2f} Y={1: 6.2f}"\
      .format(x, y), end=" ")
if flag:
    print("попадает в область")
else:
    print("не попадает в область")
```

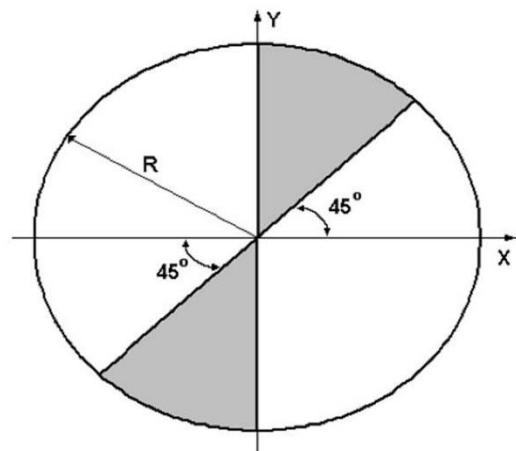
Замечание к тексту программы

Для переноса длинных строк в Python используется обратный слэш (\) за которым сразу следует возврат каретки (Enter)

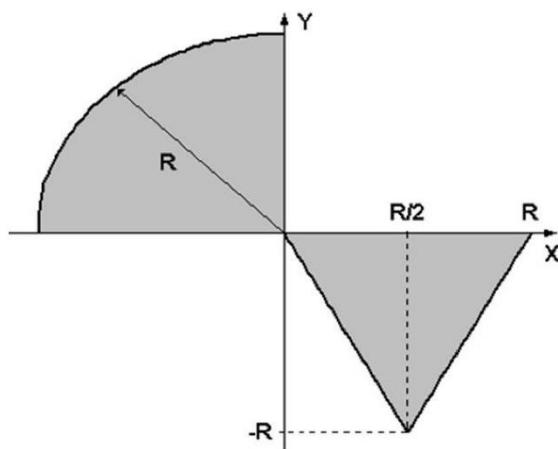
Задание к лабораторной работе №2 "Разветвляющиеся вычислительные процессы". Задание 2

Написать программу, которая определяет, попадает ли точка с заданными координатами X , Y в область, закрашенную на рисунке серым цветом. Результат работы программы вывести в виде текстового сообщения. Параметр R вводится с клавиатуры.

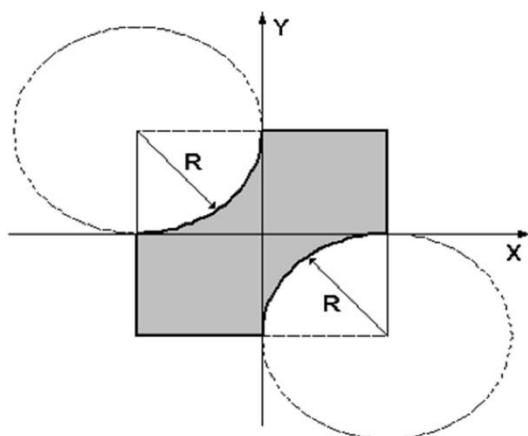
1)



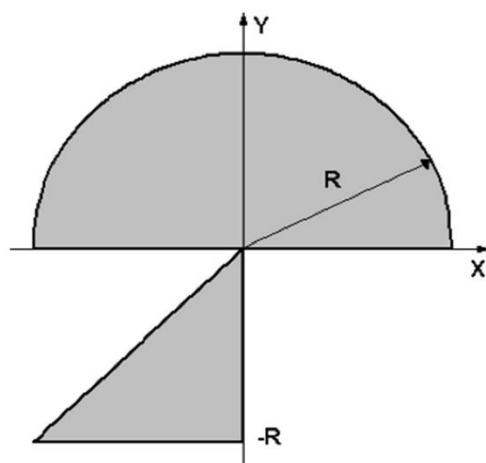
2)



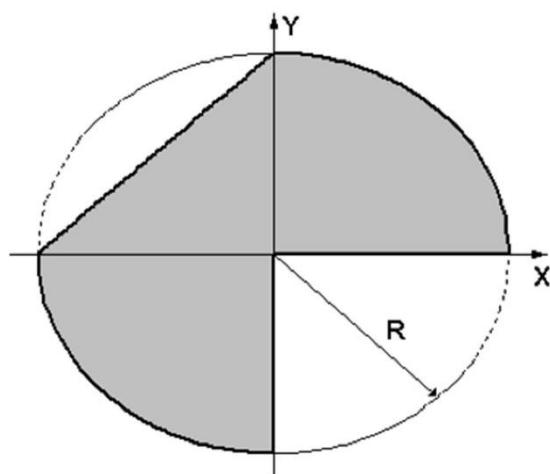
3)



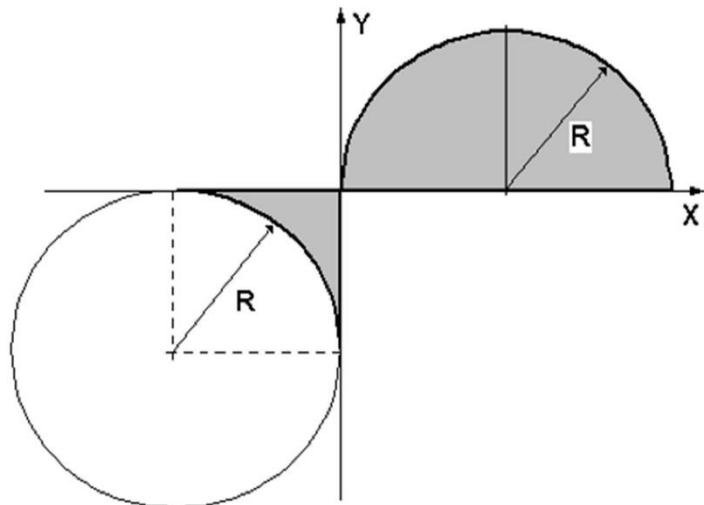
4)



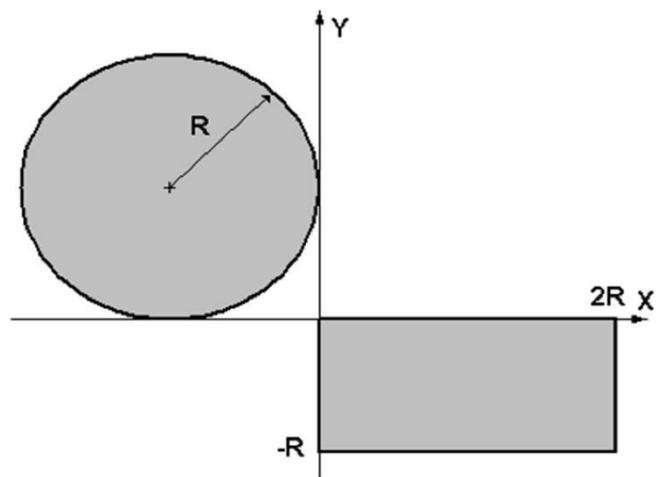
5)



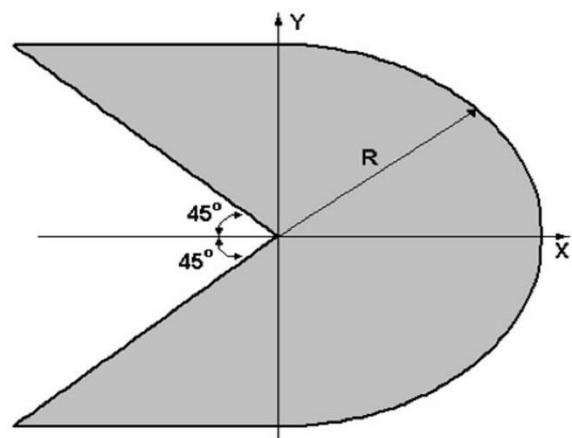
6)



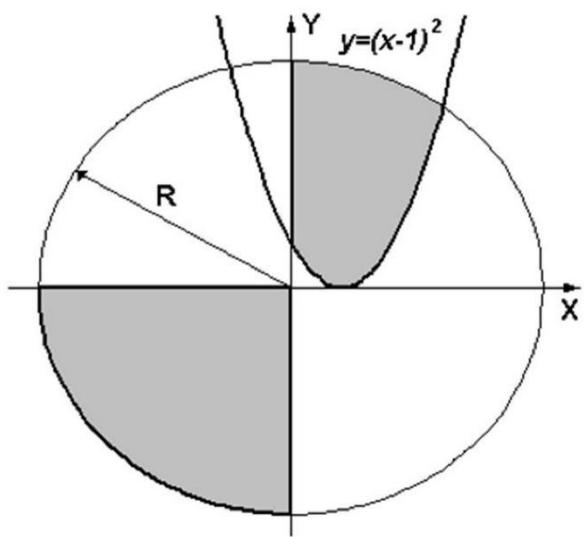
7)



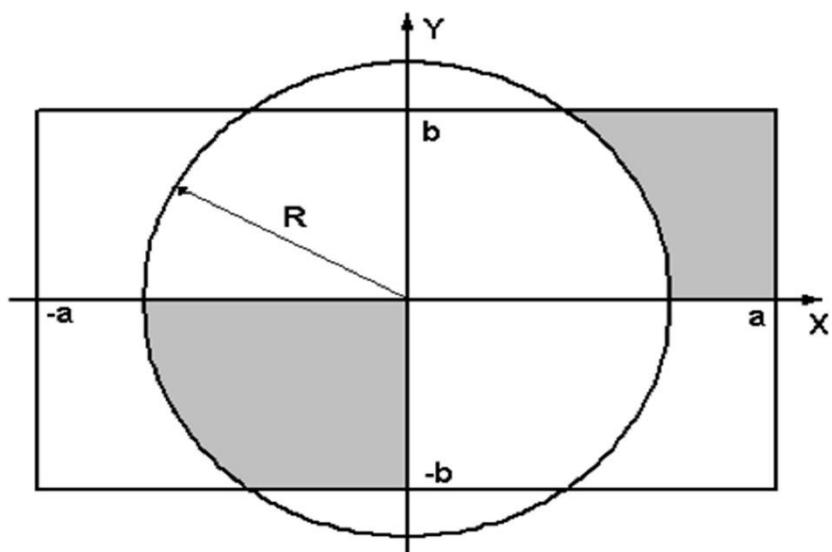
8)



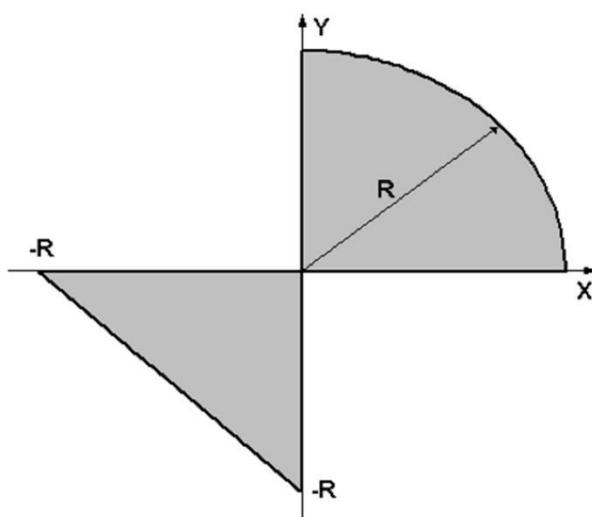
9)



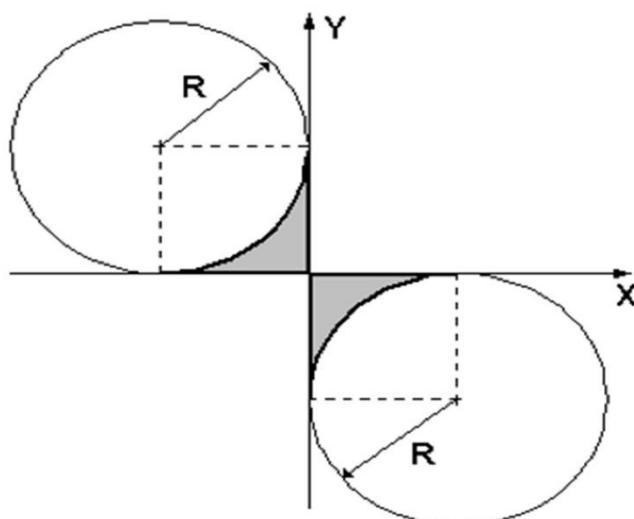
10)



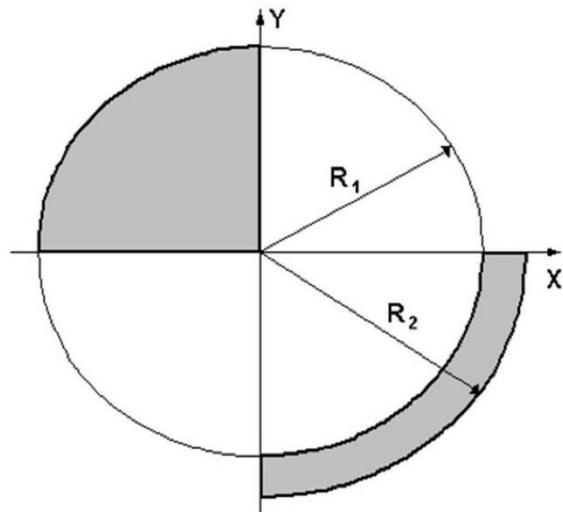
11)



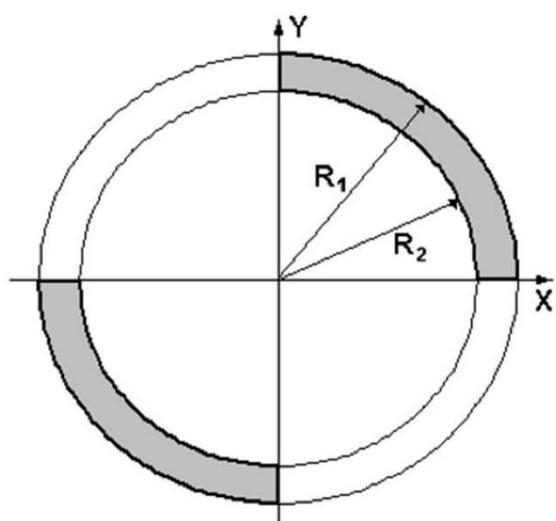
12)



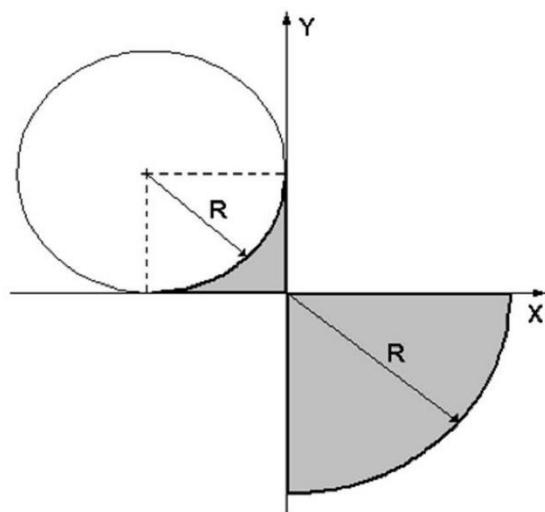
13)



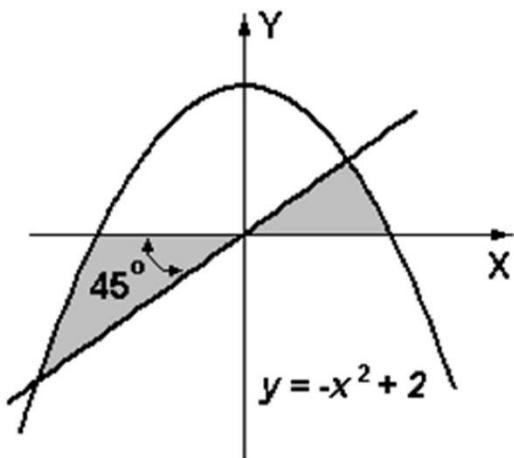
14)



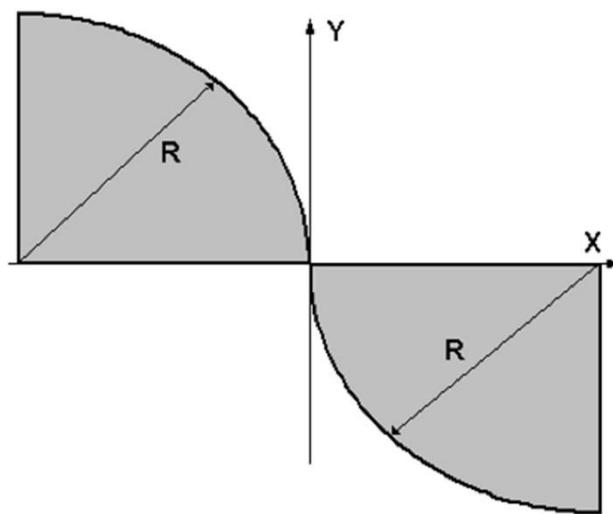
15)



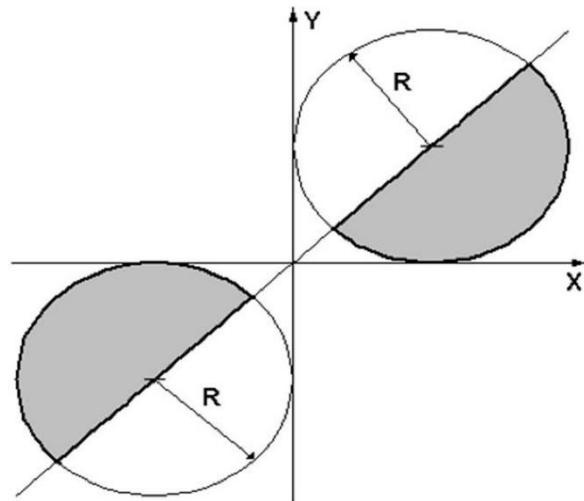
16)

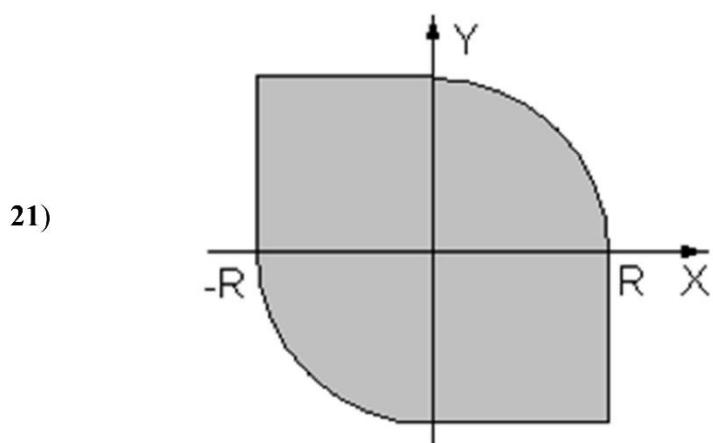
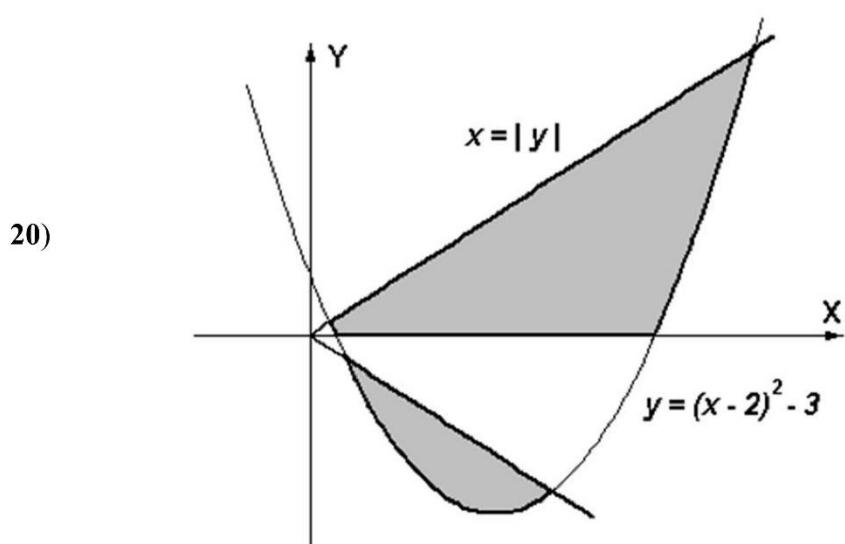
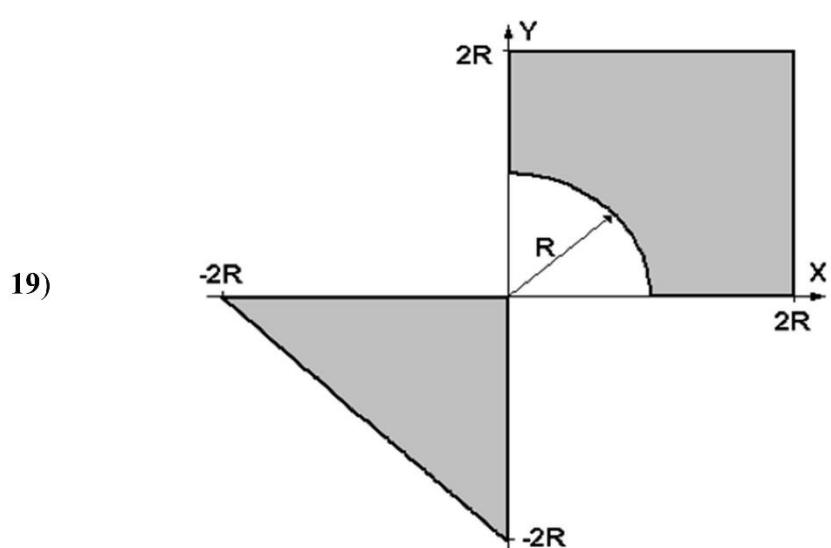


17)

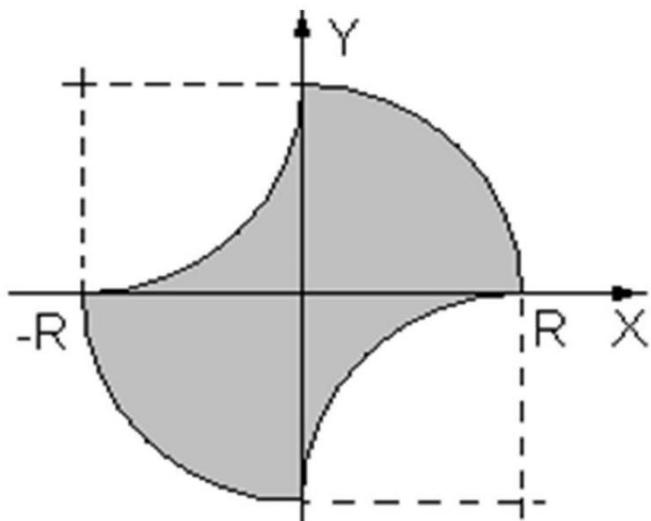


18)

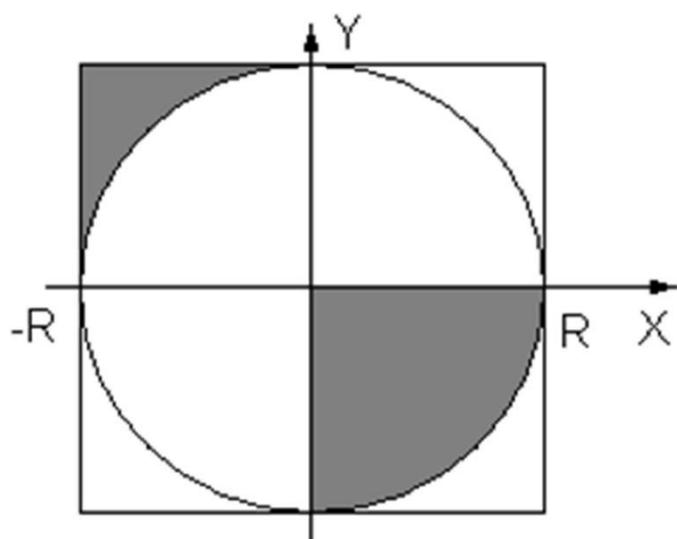




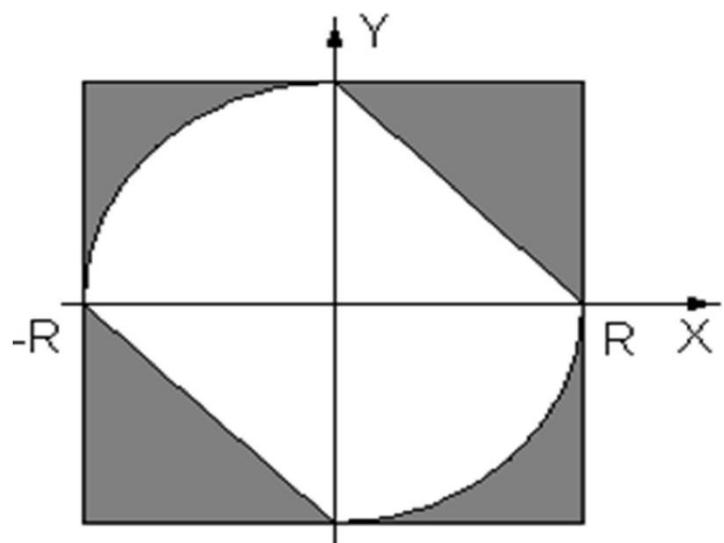
22)



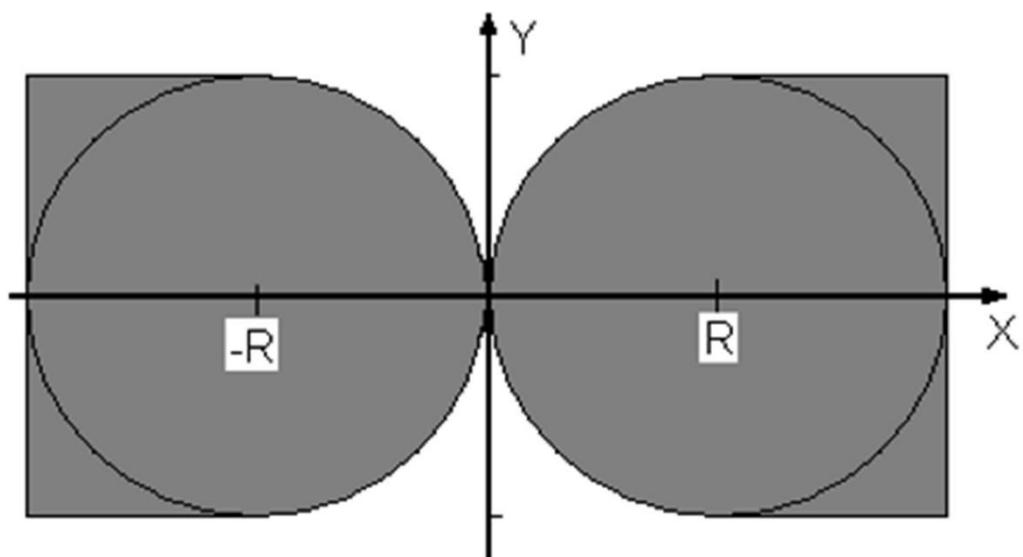
23)



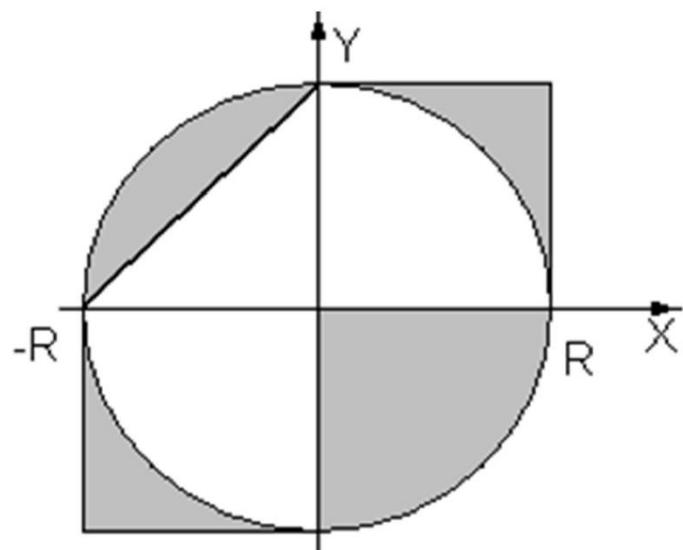
24)



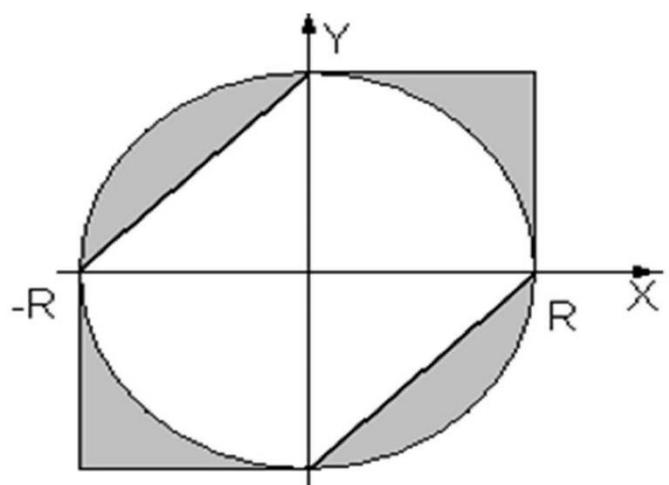
25)

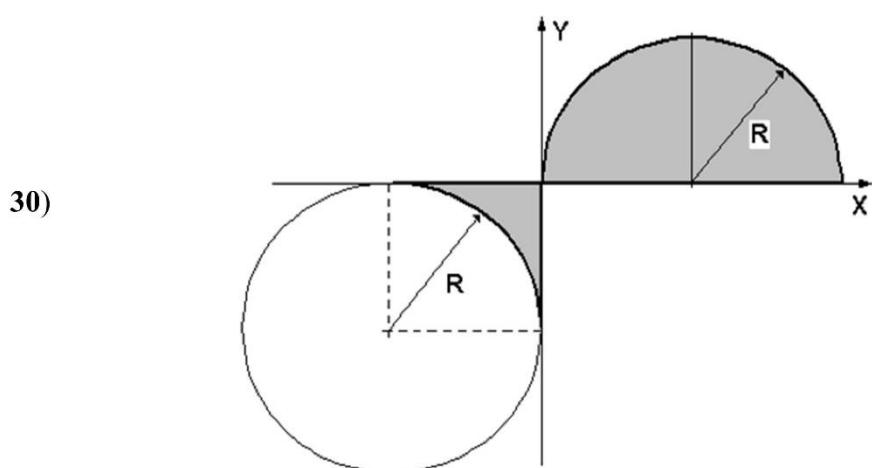
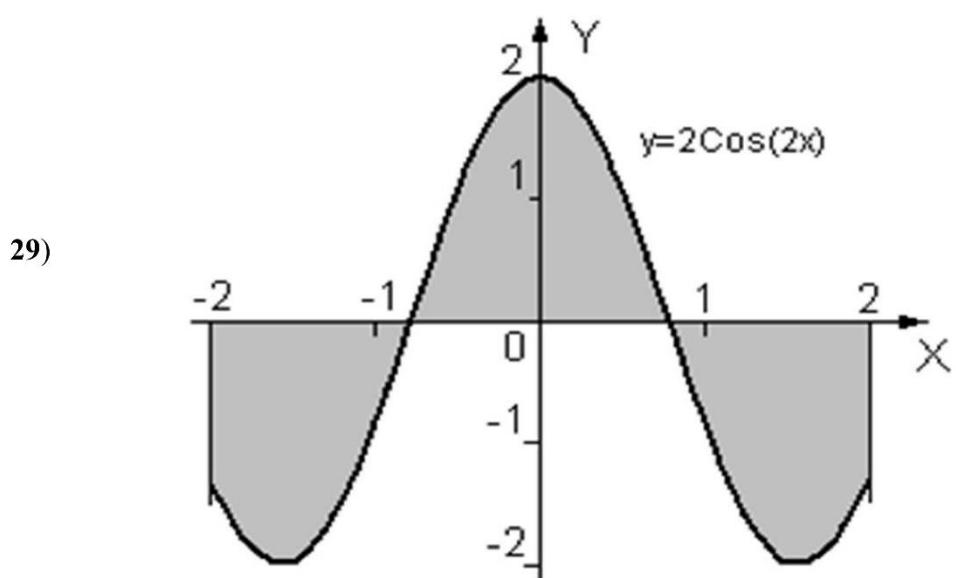
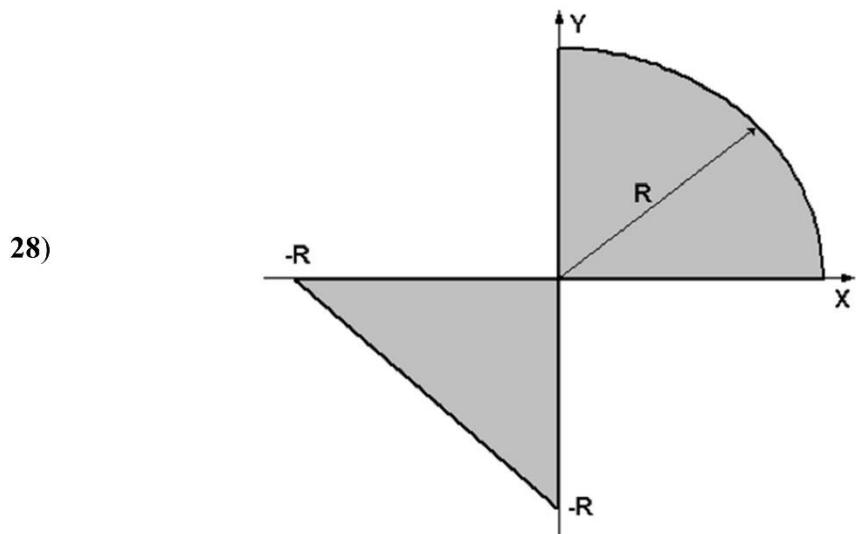


26)



27)





Лабораторная работа №3: "Организация циклов". Задание 1

Цель работы

Дать студентам практический навык в использовании базовых конструкций структурного программирования - операторов цикла. Работа составлена из трёх заданий.

Постановка задачи

Вычислить и вывести на экран или в файл в виде таблицы значения функции, заданной графически (см. лабораторная работа № 2, задание 1), на интервале от $X_{нач}$ до $X_{кон}$ с шагом dx . Интервал и шаг задать таким образом, чтобы проверить все ветви программы. Таблицу снабдить заголовком и шапкой.

Теоретическое введение

Для решения задачи использована программа, подготовленная в лабораторной работе №2, задание 1 и оператор цикла с предусловием:

```
<Начальное значение>
while <Условие>:
    <Инструкции>
    <Приращение>
[else:
    <Блок, выполняемый, если неиспользовался break>
]
```

Для обмена с консолью (вывод сообщений и ввод начальных данных) использованы стандартные процедуры `print()` и `input()`. Результаты работы программы записываются в текстовый файл.

Описание алгоритма

1. Ввести значения переменных X_{beg} , X_{end} , Dx .
2. Присвоить текущему значению Xt начальное значение: $Xt = X_{нач}$.
3. Вычислить значение функции и вывести в виде строки таблицы.
4. Вычислить новое значение аргумента $Xt = Xt + Dx$.
5. Если значение аргумента меньше X_{end} , то перейти к пункту 3.
6. Завершить рисование таблицы и работу программы.

Описание входных и выходных данных

В предшествующей работе был принят вещественный тип данных (*real*). В этой работе тип данных сохранён. Для упрощения последующего контроля работы программы в выходной текстовый файл записываются и начальные данные.

Листинг программы

```
# -*- coding: cp1251 -*-
from math import *
print('Введите Xbeg, Xend и Dx')
xb = float(input('Xbeg='))
xe = float(input('Xend='))
dx = float(input('Dx='))
print("Xbeg={0: 7.2f} Xend={1: 7.2f}".format(xb, xe))
print(" Dx={0: 7.2f}".format(dx))
xt = xb
print("-----+-----+")
print("I      X      I      Y      I")
```

```

print("-----+")
while xt <= xe:
    if xt < -5:
        y = 1
    elif xt >=-5 and xt<0:
        y = -(3/5)*xt-2
    elif xt >= 0 and xt<2:
        y = -sqrt(4-xt**2)
    elif xt >= 2 and xt<4:
        y = xt-2
    elif xt >= 4 and xt<8:
        y = 2+sqrt(4-(xt-6)**2)
    else: y = 2
    print("I{0: 7.2f} I{1: 7.2f} I".format(xt, y))
    xt += dx
print("-----+")

```

Результат работы программы

Xbeg	Xend	Dx		
-10.00	10.00	1.00		
-----+				
I	X	I	Y	I
-----+				
I	-10.00	I	1.00	I
I	-9.00	I	1.00	I
I	-8.00	I	1.00	I
I	-7.00	I	1.00	I
I	-6.00	I	1.00	I
I	-5.00	I	1.00	I
I	-4.00	I	0.40	I
I	-3.00	I	-0.20	I
I	-2.00	I	-0.80	I
I	-1.00	I	-1.40	I
I	0.00	I	-2.00	I
I	1.00	I	-1.73	I
I	2.00	I	0.00	I
I	3.00	I	1.00	I
I	4.00	I	2.00	I
I	5.00	I	3.73	I
I	6.00	I	4.00	I
I	7.00	I	3.73	I
I	8.00	I	2.00	I
I	9.00	I	2.00	I
I	10.00	I	2.00	I
-----+				

Задание к лабораторной работе №3 "Организация циклов". Задание 1

Вычислить и вывести на экран в виде таблицы значения функции, заданной графически (см. задание 1 лабораторной работы № 2, стр. 21), на интервале от $X_{нач}$ до $X_{кон}$ с шагом dx . Интервал и шаг задать таким образом, чтобы проверить все ветви программы. Таблицу снабдить заголовком и шапкой.

Лабораторная работа №3: "Организация циклов". Задание 2

Постановка задачи

Для десяти выстрелов, координаты которых задаются генератором случайных чисел, вывести текстовые сообщения о попадании в мишень (см. лабораторная работа №2, задание 2).

Теоретическое введение

Для решения задачи использована программа, подготовленная в лабораторной работе №2, задание 2 (см. стр. 29) и оператор цикла с параметром:

```
for <Текущий элемент> in <Последовательность>:  
    <Инструкции внутри цикла>  
[else:  
    <Блок, выполняемый, если неиспользовался break>  
]
```

Вывод сообщения выполняется стандартной функцией `print()`.

Для формирования координат точки используется модуль генератора случайных чисел, который подключается инструкцией:

```
import random
```

или

```
from random import *
```

Для формирования случайного вещественного числа воспользуемся функцией
`uniform(<Начало>, <Конец>)`

В нашей задаче значения X формируются в диапазоне (-1, 4), а для Y – (-1, 10).

Описание алгоритма

1. Вывести "шапку".
2. В цикле от 1 до 10.
3. Сформировать координаты точки X, Y.
4. Определить попадание точки в заданную область. Если есть попадание, то переменная flag получает значение 1, а иначе – 0.
5. Вывести координаты точки и маркер оставить на строке сообщения.
6. Вывести результат Yes или No в соответствии со значением переменной flag.
7. Изменить параметр цикла и проверить условие завершения.
8. Если условие `false`, то перейти к п. 3.
9. Завершить работу программы.

Описание входных и выходных данных

Типы переменных, использованные в предыдущей работе, не изменились. Для организации цикла введена новая переменная целого типа (`int`).

Листинг программы

```
# -*- coding: cp1251 -*-
from math import *
from random import *
flag = 0
print("    X      Y   Res")
print("-----")
for n in range(10):
    x = uniform(-1, 4)
    y = uniform(-1, 10)
    if (x < -1) or (x > 4):
        flag = 0          #False
    if (((x>=-1) and (x< 1) and (y>=2*x+2)
         and (y<= x**3-4*x**2+x+6))
        or
        ((x>= 1) and (x<=4) and (y>=x**3-4*x**2+x+6)
         and (y<= 2*x+2))):
        flag = 1
    else:
        flag = 0
    print("{0: 7.2f} {1: 7.2f}".format(x, y), end=" ")
    if flag:
        print("Yes")
    else:
        print("No")
```

Результат тестирования программы

X	Y	Res

-0.68	7.15	No
3.53	3.44	No
-0.05	4.02	Yes
-0.24	0.01	No
2.48	5.58	Yes
0.41	4.77	Yes
0.09	0.89	No
0.68	0.98	No
-0.03	5.46	Yes
-0.73	0.73	Yes

Задание к лабораторной работе №3 "Организация циклов". Задание 2

Для десяти выстрелов, координаты которых задаются генератором случайных чисел, вывести текстовые сообщения о попадании в мишень (см. лабораторная работа № 2, задание 2, стр.31).

Лабораторная работа №3: "Организация циклов". Задание 3

Постановка задачи

Вычислить и вывести на экран в виде таблицы значения функции интегрального синуса, заданной с помощью степенного ряда, на интервале от $X_{\text{нач}}$ до $X_{\text{кон}}$ с шагом dx с точностью ε .

Таблицу снабдить заголовком и шапкой. Каждая строка таблицы должна содержать значение аргумента, значение функции и количество просуммированных членов ряда.

$$Si(x) = \int_0^x \frac{\sin(t)}{t} dt = \sum_{n=0}^{\infty} (-1)^n \cdot \frac{x^{2n+1}}{(2n+1)!} = x - \frac{x^3}{3 \cdot 1} + \frac{x^5}{5 \cdot 3} - \dots + \quad |x| < \infty$$

Теоретическое введение

При решении задач, в которых дана общая формула вычисления элемента, очень полезно получить рекуррентную формулу. Такая формула позволяет упростить процесс программирования и ускоряет работу программы, так как получение следующего результата основывается на предыдущем. Особое внимание следует обратить на сходимость ряда. Если ряд расходится или сходится слабо, то программа может формировать сообщения о переполнении значений переменных или выводить неверный результат.

1. Будем искать рекуррентное соотношение в виде выражения: $a_{n+1} = k \cdot a_n$.

Получим выражение для k :

$$k = \frac{(-1)^{n+1} \cdot x^{2(n+1)+1} \cdot (2n+1)! \cdot (2n+1)}{(2(n+1)+1)! \cdot (2(n+1)+1) \cdot (-1)^n \cdot x^{2n+1}} = -\frac{x^2 \cdot (2n+1)}{(2n+2) \cdot (2n+3)^2}$$

2. Для решения задачи нам потребуется два цикла. Первый цикл `While` (с предусловием), будет обеспечивать изменение значения переменной X от $X_{\text{нач}}$ до $X_{\text{кон}}$ с шагом dx . Второй цикл – это цикл с постусловием. Он будет обеспечивать итерационные вычисления элементов ряда, которые удовлетворяют условию $|a_n| < \varepsilon$.

В языке Python цикл с постусловием отсутствует. Для организации такого цикла воспользуемся циклом с предусловием в следующей форме:

```
while True:  
    <тело цикла>  
    if not <условие>:  
        break;
```

Для обмена с консолью (ввод/вывод) использованы стандартные функции `input()` и `print()`.

Описание алгоритма

1. Ввести значения переменных $X_{\text{нач}}$, $X_{\text{кон}}$, dx и параметр точности ε .
2. Вывести "шапку" таблицы.
3. Инициализировать X_t начальным значением ($X_{\text{нач}}$).
3. В цикле по X_t .
4. Инициализировать переменную для подсчёта суммы членов ряда и переменную, которая отвечает за номер члена ряда (n).
5. В цикле по a_n .
6. Вычислить k , элемент ряда a_n , сумму элементов ряда и номер элемента.
7. Если модуль элемента ряда меньше ε , то прервать цикл (`break`) по a_n , иначе перейти к п.6.

6. Вывести строки таблицы: значение Xt , вычисленное значение функции и количество просуммированных членов ряда
7. Вычислить новое значение переменной $Xt = Xt + dx$.
8. Если значение аргумента меньше $Xkon$, то перейти к пункту 4.
9. Завершить рисование таблицы, и работу программы.

Описание входных и выходных данных

Поскольку тип переменных и точность представления не ограничены условием задачи, то входные переменные ($Xnach$, $Xkon$, dx и параметр точности ε) и вычисляемые значения аргумента и функции представляются переменными вещественного типа (*float*). Количество членов ряда подсчитывается переменной целого типа (*int*).

Листинг программы

```
# -*- coding: cp1251 -*-
from math import *
print('Введите Xbeg, Xend, Dx и Eps')
xb = float(input('Xbeg='))
xe = float(input('Xend='))
dx = float(input('Dx='))
eps = float(input('Eps='))
print("-----+-----+-----+")
print("I   X   I   Y   I   N I")
print("-----+-----+-----+")
xt = xb
while xt <= xe:
    an = xt
    n = 0
    y = an
    while True:
        k = -(xt**2) * (2*n+1) / ((2*n+2) * (2*n+3)**2)
        an = an*k
        y = y + an
        n = n + 1
        if abs(an) < eps:
            break
    print("I{0: 7.2f} I{1: 7.3f} I{2: 4} I".format(xt,y,n))
    xt = xt + dx
print("-----+-----+-----+")
```

Результат работы программы

```
Xnach= -4.00  Xkon=  6.00
Dx=  2.00    Eps=  0.00003
-----+-----+-----+
I   X   I   Y   I   N I
-----+-----+-----+
I  -4.00 I -1.758 I   8 I
I  -2.00 I -1.605 I   5 I
I   0.00 I  0.000 I   1 I
I   2.00 I  1.605 I   5 I
I   4.00 I  1.758 I   8 I
I   6.00 I  1.425 I  10 I
-----+-----+-----+
```

Задание к лабораторной работе №3 "Организация циклов". Задание 3

Вычислить и вывести на экран в виде таблицы значения функции, заданной с помощью ряда Тейлора, на интервале от $X_{\text{нач}}$ до $X_{\text{кон}}$ с шагом dx с точностью ϵ .

Таблицу снабдить заголовком и шапкой. Каждая строка таблицы должна содержать значение аргумента, значение функции и количество просуммированных членов ряда.

$$1. \ln \frac{x+1}{x-1} = 2 \cdot \sum_{n=0}^{\infty} \frac{1}{(2 \cdot n + 1) \cdot x^{2 \cdot n + 1}} = 2 \cdot \left(\frac{1}{x} + \frac{1}{3 \cdot x^3} + \frac{1}{5 \cdot x^5} + \dots \right), \quad |x| > 1.$$

$$2. e^{-x} = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^n}{n!} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \dots, \quad |x| < \infty.$$

$$3. e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots, \quad |x| < \infty.$$

$$4. \ln(x+1) = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{n+1}}{n+1} = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \dots, \quad -1 < x \leq 1.$$

$$5. \ln \frac{1+x}{1-x} = 2 \cdot \sum_{n=0}^{\infty} \frac{x^{2 \cdot n + 1}}{2 \cdot n + 1} = 2 \cdot \left(x + \frac{x^3}{3} + \frac{x^5}{5} + \dots \right), \quad |x| < 1.$$

$$6. \ln(1-x) = -\sum_{n=1}^{\infty} \frac{x^n}{n} = -\left(x + \frac{x^2}{2} + \frac{x^3}{3} + \dots \right), \quad -1 \leq x < 1.$$

$$7. \operatorname{arcctg} x = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1} \cdot x^{2 \cdot n + 1}}{2 \cdot n + 1} = \frac{\pi}{2} - x + \frac{x^3}{3} - \frac{x^5}{5} + \dots, \quad x \leq 1.$$

$$8. \operatorname{arctg} x = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2 \cdot n + 1) \cdot x^{2 \cdot n + 1}} = \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3 \cdot x^3} - \frac{1}{5 \cdot x^5} + \dots, \quad x > 1.$$

$$9. \operatorname{arctg} x = \sum_{n=0}^{\infty} \frac{(-1)^{n+1} \cdot x^{2 \cdot n + 1}}{(2 \cdot n + 1)} = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots, \quad |x| \leq 1.$$

$$10. \operatorname{Arth} x = \sum_{n=0}^{\infty} \frac{x^{2 \cdot n + 1}}{2 \cdot n + 1} = x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + \dots, \quad |x| < 1.$$

$$11. \operatorname{Arth} = \sum_{n=0}^{\infty} \frac{1}{(2 \cdot n + 1) \cdot x^{2 \cdot n + 1}} = \frac{1}{x} + \frac{1}{3 \cdot x^3} + \frac{1}{5 \cdot x^5} + \dots, \quad |x| > 1.$$

$$12. \operatorname{arctg} x = -\frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2 \cdot n + 1) \cdot x^{2 \cdot n + 1}} = -\frac{\pi}{2} - \frac{1}{x} + \frac{1}{3 \cdot x^3} - \frac{1}{5 \cdot x^5} + \dots, \quad x < -1.$$

$$13. e^{-x^2} = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2 \cdot n}}{n!} = 1 - x^2 + \frac{x^4}{2!} - \frac{x^6}{3!} + \frac{x^8}{4!} - \dots, \quad |x| < \infty.$$

$$14. \cos x = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2 \cdot n}}{(2 \cdot n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots, \quad |x| < \infty.$$

$$15. \frac{\sin x}{x} = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2 \cdot n}}{(2 \cdot n + 1)!} = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \dots, \quad |x| < \infty.$$

- 16.** $\ln x = 2 \cdot \sum_{n=0}^{\infty} \frac{(x-1)^{2n+1}}{(2 \cdot n + 1) \cdot (x+1)^{2n+1}} = 2 \cdot \left(\frac{x-1}{x+1} + \frac{(x-1)^3}{3 \cdot (x+1)^3} + \frac{(x-1)^5}{5 \cdot (x+1)^5} + \dots \right), \quad x > 0.$
- 17.** $\ln x = \sum_{n=0}^{\infty} \frac{(x-1)^{n+1}}{(n+1) \cdot (x+1)^{n+1}} = \frac{x-1}{x} + \frac{(x-1)^2}{2 \cdot x^2} + \frac{(x-1)^3}{3 \cdot x^3} + \dots, \quad x > \frac{1}{2}.$
- 18.** $\ln x = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot (x-1)^{n+1}}{(n+1)} = (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \dots, \quad 0 < x \leq 2.$
- 19.** $\arcsin x = x + \sum_{n=1}^{\infty} \frac{1 \cdot 3 \cdot \dots \cdot (2 \cdot n - 1) \cdot x^{2n+1}}{2 \cdot 4 \cdot \dots \cdot 2 \cdot n \cdot (2 \cdot n + 1)} = x + \frac{x^3}{2 \cdot 3} + \frac{1 \cdot 3 \cdot x^5}{2 \cdot 4 \cdot 5} + \frac{1 \cdot 3 \cdot 5 \cdot x^7}{2 \cdot 4 \cdot 6 \cdot 7} + \frac{1 \cdot 3 \cdot 5 \cdot 7 \cdot x^9}{2 \cdot 4 \cdot 6 \cdot 8 \cdot 9} + \dots, \quad |x| < 1.$
- 20.** $\arccos x = \frac{\pi}{2} - \left(x + \sum_{n=1}^{\infty} \frac{1 \cdot 3 \cdot \dots \cdot (2 \cdot n - 1) \cdot x^{2n+1}}{2 \cdot 4 \cdot \dots \cdot 2 \cdot n \cdot (2 \cdot n + 1)} \right) = \frac{\pi}{2} - \left(x + \frac{x^3}{2 \cdot 3} + \frac{1 \cdot 3 \cdot x^5}{2 \cdot 4 \cdot 5} + \frac{1 \cdot 3 \cdot 5 \cdot x^7}{2 \cdot 4 \cdot 6 \cdot 7} + \frac{1 \cdot 3 \cdot 5 \cdot 7 \cdot x^9}{2 \cdot 4 \cdot 6 \cdot 8 \cdot 9} + \dots \right), \quad |x| < 1.$
- 21.** $shx = \frac{1}{2} \cdot (e^x - e^{-x}) = \sum_{n=0}^{\infty} \frac{x^{2n+1}}{(2 \cdot n + 1)!} = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \dots, \quad [x^2 < \infty].$
- 22.** $chx = \frac{1}{2} \cdot (e^x + e^{-x}) = \sum_{n=0}^{\infty} \frac{x^{2n}}{(2 \cdot n)!} = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \dots, \quad [x^2 < \infty].$
- 23.** $\ln\left(\frac{x+1}{x}\right) = 2 \cdot \left[\frac{1}{2x+1} + \frac{1}{3 \cdot (2x+1)^3} + \frac{1}{5 \cdot (2x+1)^5} + \dots \right] \quad (2x+1)^2 > 1.$
- 24.** $(1+x)^{\frac{1}{4}} = 1 + \frac{1}{4} \cdot x - \frac{1}{4} \cdot \frac{3}{8} \cdot x^2 + \frac{1}{4} \cdot \frac{3}{8} \cdot \frac{7}{12} \cdot x^3 - \frac{1}{4} \cdot \frac{3}{8} \cdot \frac{7}{12} \cdot \frac{11}{16} \cdot x^4 + \dots, \quad |x| \leq 1.$
- 25.** $(1-x)^{\frac{1}{2}} = 1 - \frac{1}{2} \cdot x - \frac{1}{2} \cdot \frac{1}{4} \cdot x^2 - \frac{1}{2} \cdot \frac{1}{4} \cdot \frac{3}{6} \cdot x^3 - \frac{1}{2} \cdot \frac{1}{4} \cdot \frac{3}{6} \cdot \frac{5}{8} \cdot x^4 - \dots, \quad |x| \leq 1.$
- 26.** $(1+x)^{-\frac{1}{3}} = 1 - \frac{1}{3} \cdot x + \frac{1}{3} \cdot \frac{4}{6} \cdot x^2 - \frac{1}{3} \cdot \frac{4}{6} \cdot \frac{7}{9} \cdot x^3 + \frac{1}{3} \cdot \frac{4}{6} \cdot \frac{7}{9} \cdot \frac{10}{12} \cdot x^4 - \dots, \quad |x| \leq 1$
- 27.** $(1+x)^{-3} = 1 - \frac{1}{1 \cdot 2} (2 \cdot 3 \cdot x - 3 \cdot 4 \cdot x^2 + 4 \cdot 5 \cdot x^3 - 5 \cdot 6 \cdot x^4 + \dots) \quad |x| \leq 1.$
- 28.** $(1-x)^{-4} = \frac{1}{1 \cdot 2 \cdot 3} (2 \cdot 3 \cdot 4 \cdot x + 3 \cdot 4 \cdot 5 \cdot x^2 + 4 \cdot 5 \cdot 6 \cdot x^3 + 5 \cdot 6 \cdot 7 \cdot x^4 + \dots) \quad |x| \leq 1.$
- 29.** $(1+x)^{-5} = \frac{1}{1 \cdot 2 \cdot 3 \cdot 4} (2 \cdot 3 \cdot 4 \cdot 5 \cdot x - 3 \cdot 4 \cdot 5 \cdot 6 \cdot x^2 + 4 \cdot 5 \cdot 6 \cdot 7 \cdot x^3 - 5 \cdot 6 \cdot 7 \cdot 8 \cdot x^4 + \dots) \quad |x| \leq 1.$
- 30.** $\sin x = x \cdot \prod_{n=1}^{\infty} \left(1 - \frac{x^2}{n^2 \pi^2} \right) = x \cdot \left(1 - \frac{x^2}{\pi^2} \right) \cdot \left(1 - \frac{x^2}{2^2 \pi^2} \right) \cdot \left(1 - \frac{x^2}{3^2 \pi^2} \right) \cdot \dots \quad |x| < \infty.$

Лабораторная работа №4: "Одномерные массивы"

Цель работы

Дать студентам практический навык в написании программ обработки одномерных массивов: поиск максимумов и минимумов, сортировка².

Постановка задачи

Сформировать одномерный список, состоящий из N вещественных чисел, полученных генератором случайных чисел. Количество элементов списка (N) запрашивается у пользователя, но не превышает 30. Диапазон значений элементов от -5.0 до 5.0. Вычислить:

1. Первый и второй максимальные по модулю элементы списка.
2. Сумму элементов, модуль которых меньше единицы.
3. Все элементы, модуль которых превышает A_{max} обнулить.
3. Отсортировать список, сохраняя порядок ненулевых элементов. Равные нулю элементы разместить в конце списка.

Теоретическое введение

Для работы с одномерными массивами и матрицами (многомерными массивами) в Python имеются специализированные модули и библиотеки (array, numpy, ...).

Массив – это конечная именованная последовательность однотипных величин.

Для организации такой структуры в Python можно использовать такие структуры данных, как списки, кортежи, множества и диапазоны, которые представляют нумерованные наборы объектов. Каждый элемент набора содержит ссылку на объект, который, в свою очередь, может представлять объект произвольного типа данных и иметь неограниченную степень вложенности.

В решении этого задания для хранения однотипных данных (массивов данных) предлагается использовать структуру данных, которая называется **список**.

Список представляет собой последовательность элементов, пронумерованных от **0 (нуля)**. Элементы списка могут иметь различные типы. Список можно задать перечислением элементов списка в квадратных скобках, например, список можно задать так:

```
Preshe = [1, 3, 5, 7, 11, 13]
PColor = ['Red', 'Orange', 'Yellow', 'Green', 'Blue']
```

или так:

```
Trest = [1, 3, 'Old', 7, 'Or', 13]
```

В списке Preshe — 6 элементов, а именно, Preshe[0] == 1, Preshe[1] == 3, Preshe[2] == 5, Preshe[3] == 7, Preshe[4] == 11, Preshe[5] == 13.

Список PColor состоит из 5 элементов, каждый из которых является строкой, а вот список Trest состоит из шести элементов, часть которых – число, а часть – строка.

При обращении к элементам списка можно использовать как положительные индексы, так и отрицательные. Если индекс положительный, то счет ведется от нуля до максимального элемента, слева направо. Если индекс отрицательный, то счет ведется справа налево: Preshe[-1] == 13, Preshe[-6] == 1.

² Структуры данных Python, могут использоваться для организации массивов. В этих структурах могут быть методы и функции поиска максимума, минимума или сортировки. В этом руководстве предлагается использовать алгоритмы поиска решений без использования готовых методов и функций.

Количество элементов в списке (длину списка), можно получить при помощи функции `len`, например, `len(Preshe) == 6`.

Существует несколько способов работы со списком. Разработчики предлагают различные варианты от специальных модулей до библиотек. Стандартные решения языка для нашего примера достаточны.

Рассмотрим один из способов создания списка и его наполнения:

- Запросить размер списка у пользователя. Пусть этот размер не должен быть меньше 5 элементов и не превышать 30;
- Создать пустой список (не содержащий элементов, длины 0);
- Наполним список случайными числами, применяя метод `append`.

```
n = int(input("Элементов в списке (N<=30) N: "))  
if n > 30: n = 30  
elif n < 5: n = 5  
mas = [] # Создаем пустой список  
for i in range(n): # Инициализация  
    mas.append(uniform(-5, 5))
```

Если использовать, например, модуль `array`, то изменения в программе будут минимальными:

```
# создание массива нулевой длины  
# с элементами вещественного типа  
mas = array('f')  
for i in range(n): # Инициализация  
    mas.append(uniform(-5, 5))
```

В решении задачи используется цикл с параметром, что позволяет получать доступ к элементам массива:

```
for <Текущий элемент> in <Последовательность>:  
    <Инструкции внутри цикла>
```

Для обмена с консолью (ввод/вывод) использованы стандартные функции `input()` и `print()`.

После формирования списка, в следующем цикле подсчитывается сумма элементов, модуль которых не превышает единицу. В этом же цикле при обнаружении в массиве элемента, значение которого превышает пороговое значение, выполняется обнуление элемента.

Поиск первого и второго максимальных элементов построен по принципу однопроходного алгоритма.

Модуль элемента $A[i]$ сравнивается с первым максимальным элементом $Max1$.

Если $\text{abs}(A[i]) > Max1$, то значение $Max1$ сдвигается на позицию второго максимального элемента ($Max2$), а $Max1 = \text{abs}(A[i])$. Иначе модуль элемента $A[i]$ сравнивается со вторым максимальным элементом $Max2$.

Для сжатия массива по заданному принципу (нулевые элементы размещаются в конце массива при сохранении порядка ненулевых элементов), используется следующий алгоритм: используется дополнительный индекс j . Массив просматривается с первого элемента. При обнаружении не нулевого элемента он копируется в элемент с индексом j и индекс j инкрементируется. После просмотра массива все элементы, начиная с элемента с индексом j обнуляются.

Описание алгоритма

1. Запросить количество элементов N и пороговое значение $Amax$.
2. Инициализировать массив случайными данными и вывести начальное состояние.

3. В цикле от 0 до N-1. Найти сумму элементов, модуль которых меньше 1, и обнулить элементы, значение которых превысило установленный порог A_{max} .
4. Инициализировать $Max1$ и $Max2$ модулем значения нулевого элемента массива.
5. В цикле от 1 до N-1. Если модуль элемента массива больше $Max1$, то $Max1$ сохранить в $Max2$, а модуль элемента массива в $Max1$. Иначе, если модуль элемента массива больше $Max2$, то модуль элемента массива сохранить в $Max2$.
6. Инициализировать переменную j .
7. В цикле от 0 до N-1. Если значение элемента больше нуля, то копировать в элемент с индексом j . Увеличить j на 1.
8. В цикле от j до N-1. Все элементы приравнять нулю.
9. Вывести полученный массив и значения $Max1$, $Max2$ и суммы.

Описание входных и выходных данных

Поскольку тип элементов массива задан как вещественный, то тип переменных, используемых в подсчётах, также определим как вещественный (**float**).

Листинг программы

```
# -*- coding: cp1251 -*-
from math import *
from random import *
n = int(input("Элементов в массиве (N<=30) N: "))
if n > 30: n = 30
elif n < 5: n = 5
amax = float(input("Пороговое значение A: "))
# Генерация массива и вывод
print("Начальное состояние")
mas = []
for i in range(n):
    mas.append(uniform(-5, 5))
    print("{0: 7.3f}".format(mas[i]), end=" ")
print()
# Нахождение суммы
# Обнуление элементов превысивших порог
asum = 0.0
for i in range(n): # от 0 до n-1
    if abs(mas[i]) < 1.0:
        asum = asum + mas[i]
    if abs(mas[i]) > amax:
        mas[i] = 0.0
# Поиск максимального элемента
max1 = abs(mas[0])
max2 = abs(mas[0])
for i in range(1,n):
    if max1 < abs(mas[i]):
        max2 = max1
        max1 = abs(mas[i])
    else:
        if max2 < abs(mas[i]):
            max2 = abs(mas[i])
# Сортировка массива
j = 0
for i in range(n):
```

```

if abs(mas[i]) > 0.00001:
    mas[j] = mas[i]
    j = j + 1
for i in range(j,n): # от j до n-1
    mas[i] = 0.0
# Массив после сортировки
print("Конечное состояние")
for i in range(n):
    print("{0: 7.3f}".format(mas[i]), end=" ")
print()
print("max1={0: 7.3f} max2={1: 7.3f} sum={2: 7.3f}"\
      .format(max1, max2, asum))

```

Результат работы программы

Элементов в массиве (N<=30) N: 7
Пороговое значение A: 2.8
Начальное состояние
-2.655 2.194 1.034 2.804 -1.490 -0.474 -0.817
Конечное состояние
-2.655 2.194 1.034 -1.490 -0.474 -0.817 0.000
max1= 2.655 max2= 2.655 sum= -1.291

Задание к лабораторной работе №4 "Одномерные массивы"

Вариант 1

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Сумму отрицательных элементов.
2. Произведение элементов, расположенных между максимальным и минимальным элементами.

Упорядочить элементы массива по возрастанию.

Вариант 2

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Сумму положительных элементов.
2. Произведение элементов, расположенных между максимальным по модулю и минимальным по модулю элементами.

Упорядочить элементы массива по убыванию.

Вариант 3

В одномерном массиве, состоящем из n целочисленных элементов, вычислить:

1. Произведение элементов с четными номерами.
2. Сумму элементов, расположенных между первым и последним нулевыми элементами.
Преобразовать массив таким образом, чтобы сначала располагались все положительные элементы, а потом - все отрицательные (элементы, равные нулю, считать положительными).

Вариант 4

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Сумму элементов с нечетными номерами.
2. Сумму элементов, расположенных между первым и последним отрицательными элементами. Сжать массив, удалив из него все элементы, модуль которых не превышает единицу. Освободившиеся в конце массива элементы заполнить нулями.

Вариант 5

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Максимальный элемент массива.
2. Сумму элементов, расположенных до последнего положительного элемента.
Сжать массив, удалив из него все элементы, модуль которых находится в интервале $[a, b]$.
Освободившиеся в конце массива элементы заполнить нулями.

Вариант 6

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Минимальный элемент массива.
2. Сумму элементов, расположенных между первым и последним положительными элементами.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, равные нулю, а потом - все остальные.

Вариант 7

В одномерном массиве, состоящем из n целочисленных элементов, вычислить:

1. Номер максимального элемента массива.
2. Произведение элементов массива, расположенных между первым и вторым нулевыми элементами.

Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй половине - элементы, стоявшие в четных позициях.

Вариант 8

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Номер минимального элемента.
2. Сумму элементов, расположенных между первым и вторым отрицательными элементами.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, модуль которых не превышает единицу, а потом — все остальные.

Вариант 9

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Максимальный по модулю элемент.
2. Сумму элементов, расположенных между первым и вторым положительными элементами. Преобразовать массив таким образом, чтобы элементы, равные нулю, располагались после всех остальных.

Вариант 10

В одномерном массиве, состоящем из n целочисленных элементов, вычислить:

1. Минимальный по модулю элемент.
2. Сумму модулей элементов, расположенных после первого элемента, равного нулю. Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в четных позициях, а во второй половине — элементы, стоявшие в нечетных позициях.

Вариант 11

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Номер минимального по модулю элемента.
2. Сумму модулей элементов, расположенных после первого отрицательного элемента. Сжать массив, удалив из него все элементы, величина которых находится в интервале $[a, b]$. Освободившиеся в конце массива элементы заполнить нулями.

Вариант 12

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Номер максимального по модулю элемента.
2. Сумму элементов, расположенных после первого положительного элемента. Преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых лежит в интервале $[a, b]$, а потом — все остальные.

Вариант 13

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Количество элементов массива, лежащих в диапазоне от A до B .
2. Сумму элементов, расположенных после максимального элемента. Упорядочить элементы массива по убыванию модулей.

Вариант 14

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Количество элементов массива, равных нулю.
2. Сумму элементов, расположенных после минимального элемента. Упорядочить элементы массива по возрастанию модулей.

Вариант 15

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Количество элементов массива, больших C .
2. Произведение элементов, расположенных после максимального по модулю элемента. Преобразовать массив таким образом, чтобы сначала располагались все отрицательные элементы, а потом — все положительные (элементы, равные нулю, считать положительными).

Вариант 16

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Количество отрицательных элементов.
 2. Сумму модулей элементов, расположенных после минимального по модулю элемента.
- Заменить все отрицательные элементы массива их квадратами и упорядочить элементы массива по возрастанию.

Вариант 17

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Количество положительных элементов.
 2. Сумму элементов, расположенных после последнего элемента, равного нулю.
- Преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых не превышает единицу, а потом - все остальные.

Вариант 18

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Количество элементов массива, меньших С.
2. Сумму целых частей элементов массива, расположенных после последнего отрицательного элемента.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, отличающиеся от максимального не более чем на 20 %, а потом - все остальные:

Вариант 19

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Произведение отрицательных элементов.
 2. Сумму положительных элементов, расположенных до максимального элемента.
- Изменить порядок следования элементов в массиве на обратный.

Вариант 20

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Произведение положительных элементов.
 2. Сумму элементов, расположенных до минимального элемента.
- Упорядочить по возрастанию отдельно элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах.

Вариант 21

В одномерном массиве, состоящем из n целых элементов, вычислить:

1. Максимальный элемент массива.
 2. Сумму остатков получаемых от деления элементов массива на максимальный элемент.
- Упорядочить элементы массива по возрастанию остатков, полученных от их деления на максимальный элемент.

Вариант 22

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Среднее значение, как отношение суммы всех элементов на их количество $\bar{a} = \frac{1}{n} \cdot \sum_{i=1}^n a_i$.
2. Сумму квадратов разностей между элементами массива и полученным средним значением $S = \sum (a_i - \bar{a})^2$.

Упорядочить элементы массива по убыванию их разности со средним значением ($a_i - \bar{a}$): вначале расположить элементы с положительной максимальной разностью, а затем остальные по убыванию разности.

Вариант 23

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Второй по величине элемент.
2. Сумму элементов, расположенных между максимальным и вторым по величине элементами.

Упорядочить элементы массива по возрастанию остатков, полученных от их деления на второй по величине элемент.

Вариант 24

В одномерном массиве, состоящем из n вещественных элементов:

1. Вычислить максимальный и минимальный элементы (Max и Min).
2. Выполнить вставку нового значения в элемент, который расположен в середине между максимальным и минимальным элементами. Вставку выполнить следующим образом: запросить новое значение, которое должно быть в диапазоне ($Min \leq a \leq Max$); переместить вправо на одну позицию элементы массива от точки вставки (последний элемент теряется); освободившемуся элементу присвоить новое значение.

Упорядочить по возрастанию модулей разности элементов массива и нового значения:

$$|ai - a|$$

Вариант 25

С использованием модуля *Random* сформировать одномерный массив, состоящий из n вещественных элементов в котором элементы принимают случайное значение, и выполняется условие: $a1 < a2 < a3 < \dots < an$.

1. Для заданного числа y , такого, что $a1 < y < an$, определить такой k , чтобы $ak < y < ak+1$.
2. Вычислить сумму элементов массива, расположенных до элемента ak .

Упорядочить по убыванию элементы, стоящие после элемента ak .

Вариант 26

С использованием модуля *Random* сформировать одномерный массив, состоящий из n вещественных элементов в котором элементы принимают случайное значение, и выполняется условие: $a1 > a2 > a3 > \dots > an$.

1. Для заданного числа y , такого, что $a1 < y < an$, определить такой k , чтобы $ak < y < ak+1$.
2. Вычислить сумму элементов массива, расположенных после элемента ak .

Упорядочить по убыванию элементы, стоящие до элемента ak .

Вариант 27

С использованием модуля *Random* сформировать одномерный массив, состоящий из n вещественных элементов в котором элементы случайным образом принимают положительный или отрицательный знак и значение от -5 до 5. Для заданного числа y , такого, что $amin < y < amax$, вычислить:

1. Сумму элементов массива, значения модуля которых меньше y .
2. Произведение остальных элементов.

Вариант 28

С использованием модуля *Random* сформировать одномерный массив, состоящий из n вещественных элементов в котором элементы случайным образом принимают положительный или отрицательный знак и значение от -10 до 10. Для заданного числа y , такого, что $amin < y < amax$, вычислить:

1. Произведение элементов массива, значения модуля которых больше y .
2. Сумму модулей остальных элементов.

Вариант 29

Координаты n точек заданы как элементы одномерного массива. Нечётные элементы - значения ординат, а чётные - абсцисс. Вычислить максимальное расстояние между ближайшими точками.

Вывести координаты всех точек, которые попадают в круг с координатами центра X_0 , Y_0 и радиусом R .

Упорядочить положение точек по возрастанию их ординат.

Вариант 30

Координаты n точек заданы как элементы одномерного массива. Нечётные элементы - значения ординат, а чётные - абсцисс. Вычислить минимальное расстояние между ближайшими точками.

Вывести координаты всех точек, которые попадают в кольцо с координатами центра X_0 , Y_0 , внутренним радиусом $R1$ и внешним радиусом $R2$.

Упорядочить положение точек по возрастанию их абсцисс.

Лабораторная работа №5: "Двумерные массивы и функции"

Цель работы

Дать студентам практический навык в написании программ обработки двумерных массивов с использованием функций.

Постановка задачи

Дан двумерный массив вещественных чисел.

1. Инициализировать элементы случайными числами в диапазоне (-10 :- 10).
2. Вычислить среднее и дисперсию (D) по всем элементам массива.
3. Заменить, на среднее значение, те элементы массива, отклонение которых от среднего превышает σ , где $\sigma = \sqrt{D}$.
4. Пункты задания оформить в виде функций.

Теоретическое введение

Организация двумерных массивов

В Python реализовать массив можно через вложенные списки. В следующем листинге программы показан прием формирования двумерного списка и способы инициализации его элементов. В этой программе следует обратить внимание на то, что элемент списка, ранее инициализированный числом, получает значение строкового типа и преобразование делается поумолчанию.

Листинг программы

```
from random import *
n=int(input("Введите размер матрицы (NxN): "))
A=[] # Пустой список
B=[]
for i in range(n):
    A.append([0]*n) # Вложенный список
    # инициализированный 0
    B.append([3]*n) # Вложенный список
    # инициализированный 3
print(A) # Вывод списка
print(B)
print() # Просто "Enter"
for Row in range(n): # По строкам и
    for Col in range(n): # по столбцам
        A[Row][Col]=randint(0,99) # псевдослучайное число
for Row in range(n): # Вывод результата
    for Col in range(n):
        print("{0:02}".format(A[Row][Col]), end=" ")
    print()
print()
#
A[1][2] = 'AB' # A это "сюрприз"
for i in range(n): # Вывод результата
    for j in range(n):
        if ((i==1) and (j==2)):
            print(A[1][2], end=" ")
        else:
            print("{0:02}".format(A[i][j]), end=" ")
```

```
print()
```

Результат работы программы

```
Ведите размер матрицы (NxN) : 4
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
[[3, 3, 3, 3], [3, 3, 3, 3], [3, 3, 3, 3], [3, 3, 3, 3]]

58 75 20 06
17 99 45 72
26 49 73 46
74 20 35 03

58 75 20 06
17 99 AB 72
26 49 73 46
74 20 35 03
```

Отметим следующие моменты, касающиеся такой реализации кода:

- элементом списка может быть объект любого типа, это снижает эффективность кода;
- в случае обработки массивов (совокупность однотипных элементов) следует использовать более эффективный код.

В научных вычислениях массивы используются часто. В качестве примеров можно рассмотреть и временные ряды метеонаблюдений, и матрицы, используемые при решении систем уравнений.

Как уже отмечалось, Python является объектно-ориентированным языком. В этом языке массивы – это объекты со своими атрибутами и методами.

Так, для получения размерности массива можно воспользоваться атрибутом `ndim`:

```
dim = Mymas.ndim
```

Количество элементов по каждой из координат возвращает атрибут `shape`:

```
a, b = Mymas.shape
```

Атрибут `size` возвращает количество элементов в массиве:

```
n = Mymas.size
```

Функция `len()` позволяет получить количество элементов в строке:

```
m = len(Mymas)
```

С целью повышения эффективности кода и реализации дополнительных методов и функций, применяемых в обработке массивов, в Python разработаны различные модули. Например, большие возможности для работы с многомерными массивами предоставляет модуль **NumPy** (`numpy`), который и предлагается использовать для решения задач нашего пособия.

Следующий листинг программы демонстрирует способы генерации матриц и вывода матриц на экран с использованием модуля `numpy`.

Листинг программы

```
import numpy as np

n=int(input("Введите размер матрицы (NxN) : "))
# Нулевая матрица
Z=np.zeros((n, n), int) # Row, Colum, type
print(Z)
print()
# Диагональная единичная матрица
```

```

E=np.eye(n)
print(E)
print()
# Матрица со случайным набором значений
A = 20*np.random.random(size=(n,n)) - 10
for Row in range(n):
    for Col in range(n):
        print("{0:>5.2f}".format(A[Row][Col]), end=" ")
    print()
print()
# Инициализация матрицы
for Row in range(n):
    for Col in range(n):
        A[Row][Col]=Row*10+Col
for Row in range(n):
    for Col in range(n):
        print("{0:>05.2f}".format(A[Row][Col]), end=" ")
    print()

```

Результат работы программы

```

Введите размер матрицы (NxN): 4
[[0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]

[[ 1.   0.   0.   0.]
 [ 0.   1.   0.   0.]
 [ 0.   0.   1.   0.]
 [ 0.   0.   0.   1.]]


-0.09 -1.68  5.82  0.47
-5.72 -5.65 -5.96 -2.20
 6.20  1.72 -7.72 -5.56
 8.60  0.18  8.39  4.11

00.00 01.00 02.00 03.00
10.00 11.00 12.00 13.00
20.00 21.00 22.00 23.00
30.00 31.00 32.00 33.00

```

В решении заданий следует автоматизировать процесс формирования значений для двумерных массивов, а не вводить их руками. Не следует использовать готовые методы и функции, которые скрывают алгоритм решения, например, такие, как поиск максимума или минимума.

Для решения нашего задания инициализацию элементов матрицы выполним функцией, которая генерирует псевдослучайные числа вещественного типа из состава модуля NumPy.

Некоторые функции модуля NumPy, формируют псевдослучайное число в диапазоне $[0, 1)$ (включая ноль и исключая 1), например, `random()`, `ranf()`, `sample()`. В случае, когда генерацию псевдослучайных чисел необходимо выполнить в диапазоне $[a, b)$, где $b > a$, следует использовать следующее выражение:

$$y = a + (b - a) * \text{Rand},$$

где Rand – псевдослучайное число, генерируемое одной из перечисленных выше функций. Пример представлен в программе, приведенной выше.

Для расчёта среднего, дисперсии и среднеквадратичного отклонения воспользуемся следующими формулами:

$$\text{Среднее: } A = \frac{\sum_{i=1}^n \sum_{j=1}^n a_{ij}}{n^2}; \quad \text{Дисперсия: } D = \frac{\sum_{i=1}^n \sum_{j=1}^n (a_{ij} - A)^2}{n^2 - 1}; \quad \sigma = \sqrt{D}.$$

Инициализацию матрицы, вычисление среднего и дисперсии, а также корректировку матрицы оформим в виде функций.

Функции

В программе часто повторяющийся фрагмент кода может быть оформлен в виде отдельной именованной структуры. В терминах информатики такую структуру принято называть подпрограммой.

Существуют два вида подпрограмм – это процедуры и функции. Функции, в отличие от процедур, могут участвовать в выражениях. Функция возвращает результат через собственное имя. Кроме этого часть результата может быть возвращена через один из параметров.

В некоторых языках это разделение существует явно, а в некоторых – неявно. Так, например, в языке Паскаль при описании подпрограмм используются слова `procedure` и `function`, в Basic – `Sub` (от `subroutine`) и `function`.

Пример из Паскаля:

```
readln(x, y);  
y := 2*y+sin(x);
```

В этом примере `readln()` – процедура, а `sin(x)` – функция, с аргументом `x`.

В языке Си и Python такое разделение сделано неявно. В этих языках функция может невозвращать или возвращать значение.

В языке Си подпрограммам предшествует описатель типа. Для невозвращающих значение (подпрограмм), используется тип `void`. Для возвращающих значение (функций) – тип, который соответствует типу возвращаемого значения, например, `int`, `float` или `bool`.

В Python используется только понятие "функция".

В тех случаях, когда используется инструкция `return`, функция возвращает объект через свое имя, которое указан за инструкцией `return`. Такая функция может участвовать в правой части выражений. Если инструкция `return` не используется, то функция возвращает значение `None`, которое так же может быть проанализировано в условном выражении.

ЗАМЕЧАНИЕ: Переменные, используемые при описании функции, называют параметрами, а переменные, которые используются при вызове процедуры – аргументами.

Существует несколько стилей оформления программы с подпрограммами. В одних из них описание подпрограмм предшествует описанию самой программы, в других – описание подпрограмм следует за описание программы (перед программой помещается только заголовок подпрограммы), а в третьих – подпрограммы помещаются в отдельные модули, которые подключаются к программе с помощью специальных инструкций: `uses` (Паскаль), `include` (Си), `import` (Python). Во всех стилях принимаются меры к тому, чтобы к моменту вызова подпрограммы в программе, ее имя (имя подпрограммы) было известно.

С этой целью в Python, рекомендуется все функции описывать в голове программы. Описание функции выполняется по следующей схеме:

```

def <Имя_функции> ( [Параметры] ) :
    '''Строка документирования'''
    <Тело_функции>
        [return <Возвращаемый_результат>]

```

Тут <Имя_функции> - это имя, которое будет использовано при вызове функции в программе. После имени функции, в круглых скобках, могут присутствовать параметры. По своей сути, описание параметров – это описание переменных, которые используются в теле функции для выполнения различных инструкций с их участием. С другой стороны, параметры – это переменные, через которые функция получает данные из программы или возвращает новые данные в программу.

Имя функции должно быть уникальным и соответствовать требованиям к именованию переменных.

Инструкция `return` необязательна, но может быть использована для возвращения данных из функции через имя этой функции.

Опишем алгоритм решения нашей задачи.

Описание алгоритма

1. Запросить размер массива N. Поскольку форма массива неопределена, решим задачу для квадратной матрицы (NxN).
2. Изготовить массив – инициализировать набором псевдослучайных вещественных чисел (функция `MakeMatr()`).
3. Вывести полученную матрицу (функция `PrintMatr ()`)
4. Вычислить среднее значение и дисперсию (функция `MidlDisp()`).
5. Выполнить корректировку элементов (функция `CorrectMatr ()`).
6. Вывести откорректированную матрицу

Описание входных и выходных данных

Программа запрашивает размер массива. Результат работы программы визуализируется на экране монитора. Тип элементов матрицы задан как вещественный (**float**) в соответствии с заданием.

Описание подпрограмм

Функция `MakeMatr(n, a, b)`

Функция инициализирует квадратную матрицу размером $n \times n$ псевдослучайными числами в диапазоне $[a, b]$.

Возвращается объект типа `array`.

Функция `MidlDisp(Matr)`

Функция вычисляет среднее значение по элементам массива `Matr` и дисперсию.

Функция возвращает объект типа кортеж, в котором первый элемент – среднее значение, второй - дисперсия.

Функция `CorrectMatr (Matr)`

Функция выполняет корректировку массива `Matr`, заменяя значения элементов, средним, если модуль значения в массиве превышает среднеквадратичное отклонение.

Функция возвращает значение типа `array`.

Функция `PrintMatr (Matr)`

Служит для вывода массива на экран монитора.

Листинг программы

```

import numpy as np
from math import *

```

```

def MakeMatr(n, a, b):
    """ Инициализация квадратной матрицы
    размером NxN псевдослучайными величинами
    в диапазоне [a,b] """
    Matr = (b-a)*np.random.random(size=(n,n)) + a
    return Matr

def MidlDisp(Matr):
    """ Вычисление среднего значения
    Вычисление дисперсии """
    sum = 0
    (nRow, nCol) = Matr.shape # Размеры матрицы
    for Row in range(nRow):
        for Col in range(nCol):
            sum += Matr[Row][Col]
    Midl = sum / Matr.size # Среднее
    Disp = 0
    for Row in range(nRow):
        for Col in range(nCol):
            Disp += (Matr[Row][Col] - Midl)**2
    return (Midl, Disp / (Matr.size - 1)) # Кортеж

def CorrectMatr(Matr, aver, sigma):
    """ Замена "отскочивших" значений """
    (nRow, nCol) = Matr.shape # Размеры матрицы
    for Row in range(nRow):
        for Col in range(nCol):
            if abs(abs(Matr[Row][Col]) - aver) > sigma:
                Matr[Row][Col] = aver
    return (Matr)

def PrintMatr(Matr):
    """ Печать матрицы """
    (nRow, nCol) = Matr.shape
    for Row in range(nRow):
        for Col in range(nCol):
            print("{0: 7.3f}".format(Matr[Row][Col]), end=" ")
        print()
    print()

n=int(input("Введите размер матрицы (NxN) : "))
MyMatr=MakeMatr(n, -5, 5) # Изготовить матрицу
PrintMatr(MyMatr)

(mMidl, mDisp) = MidlDisp(MyMatr) # Среднее и дисперсия
print("Midl = {0:7.3f}".format(mMidl))
print("Disp = {0:7.3f} Sig = {1:7.3f}\n"
      .format(mDisp, sqrt(mDisp)))

NMatr = CorrectMatr(MyMatr, mMidl, sqrt(mDisp)) # Корректировка
PrintMatr(NMatr)

```

Результат работы программы

Введите размер матрицы (NxN) : 7

-3.775	-3.782	-1.127	4.921	-3.295	1.567	-0.084
-1.091	-4.529	3.789	-4.245	3.748	4.199	-0.593
-4.977	4.779	1.184	-0.448	-1.605	3.030	-4.608
0.076	-1.509	-3.713	-3.791	-4.139	4.570	3.234
2.997	-3.789	1.384	-2.880	-2.572	-2.550	-3.275
-2.468	-1.418	-0.306	2.019	4.532	2.518	1.570
-2.188	4.122	-0.271	2.883	-3.256	4.610	-1.648

Mid1 = -0.249

Disp = 9.902 Sig = 3.147

-0.249	-0.249	-1.127	-0.249	-0.249	1.567	-0.084
-1.091	-0.249	-0.249	-0.249	-0.249	-0.249	-0.593
-0.249	-0.249	1.184	-0.448	-1.605	-0.249	-0.249
0.076	-1.509	-0.249	-0.249	-0.249	-0.249	-0.249
-0.249	-0.249	1.384	-2.880	-2.572	-2.550	-0.249
-2.468	-1.418	-0.306	2.019	-0.249	2.518	1.570
-2.188	-0.249	-0.271	2.883	-0.249	-0.249	-1.648

Задание к лабораторной работе №5 "Двумерные массивы и функции"

Размерности двумерных массивов следует запрашивать у пользователя. Все необходимые данные должны передаваться в функции в качестве параметров; Все переменные, используемые только внутри функции, должны быть описаны как локальные.

Использование глобальных переменных в функциях не допускается. Обеспечить вывод, как исходного массива, так и массива, полученного в результате работы программы, там, где это возможно по условию задачи.

Пункты задания оформить в виде функций.

Вариант 1

Дана целочисленная прямоугольная матрица. Определить:

1. Количество строк, не содержащих ни одного нулевого элемента.
2. Максимальное значение из чисел, встречающихся в заданной матрице более одного раза.

Вариант 2

Дана целочисленная прямоугольная матрица.

1. Определить количество столбцов, не содержащих ни одного нулевого элемента.
2. Переставляя строки заданной матрицы, расположить их в соответствии с ростом характеристик.

ПРИМЕЧАНИЕ: Характеристикой строки целочисленной матрицы назовем сумму ее положительных четных элементов.

Вариант 3

Дана целочисленная прямоугольная матрица. Определить:

1. Количество столбцов, содержащих хотя бы один нулевой элемент.
2. Номер строки, в которой находится самая длинная серия одинаковых элементов.

Вариант 4

Дана целочисленная квадратная матрица. Определить:

1. Произведение элементов в тех строках, которые не содержат отрицательных элементов.
2. Максимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы.

Вариант 5

Дана целочисленная квадратная матрица. Определить:

1. Сумму элементов в тех столбцах, которые не содержат отрицательных элементов.
2. Минимум среди сумм модулей элементов диагоналей, параллельных побочной диагонали матрицы.

Вариант 6

Дана целочисленная прямоугольная матрица. Определить:

1. Сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент.
2. Номера строк и столбцов всех седловых точек матрицы.

ПРИМЕЧАНИЕ: Матрица A имеет седловую точку A_{ij} , если A_{ij} является минимальным элементом в i-й строке и максимальным в j-м столбце.

Вариант 7

Для заданной матрицы размером 8 x 8 найти такие k, что элементы k-й строки матрицы совпадают с элементами k-ого столбца.

Найти сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент.

Вариант 8

Переставляя столбцы заданной матрицы, расположить их в соответствии с ростом характеристик.

Найти сумму элементов в тех столбцах, которые содержат хотя бы один отрицательный элемент.

ПРИМЕЧАНИЕ: Характеристикой столбца целочисленной матрицы назовем сумму модулей его отрицательных нечетных элементов.

Вариант 9

Соседями элемента A_{ij} в матрице назовем элементы A_{kl} , где $i-1 \leq l \leq k \leq i+1$, $j-1 \leq l \leq j+1$, $(k, l) \neq (i, j)$.

$$\left(\begin{array}{cccccc} a_{1,1} & a_{1,2} & \dots & a_{1,j-1} & a_{1,j} & a_{1,j+1} & \dots \\ a_{2,1} & a_{2,2} & \dots & a_{2,j-1} & a_{2,j} & a_{2,j+1} & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{i-1,1} & a_{i-1,2} & \dots & a_{i-1,j-1} & a_{i-1,j} & a_{i-1,j+1} & \dots \\ a_{i,1} & a_{i,2} & \dots & a_{i,j-1} & a_{i,j} & a_{i,j+1} & \dots \\ a_{i+1,1} & a_{i+1,2} & \dots & a_{i+1,j-1} & a_{i+1,j} & a_{i+1,j+1} & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{array} \right)$$

В этой матрице соседи углового элемента a_{11} и элемента a_{ij} выделены полужирным шрифтом.

Операция сглаживания матрицы дает новую матрицу того же размера, каждый элемент которой получается как среднее арифметическое имеющихся соседей соответствующего элемента исходной матрицы.

Построить результат сглаживания заданной вещественной матрицы размером 10×10 .

В сглаженной матрице найти сумму модулей элементов, расположенных ниже главной диагонали.

Вариант 10

Элемент матрицы называется локальным минимумом, если он строго меньше всех имеющихся у него соседей.

Соседями элемента A_{ij} в матрице назовем элементы A_{kl} , где $i-1 \leq l \leq k \leq i+1$, $j-1 < l < j+1$, $(k, l) \neq (i, j)$.

$$\left(\begin{array}{cccccc} a_{1,1} & a_{1,2} & \dots & a_{1,j-1} & a_{1,j} & a_{1,j+1} & \dots \\ a_{2,1} & a_{2,2} & \dots & a_{2,j-1} & a_{2,j} & a_{2,j+1} & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{i-1,1} & a_{i-1,2} & \dots & a_{i-1,j-1} & a_{i-1,j} & a_{i-1,j+1} & \dots \\ a_{i,1} & a_{i,2} & \dots & a_{i,j-1} & a_{i,j} & a_{i,j+1} & \dots \\ a_{i+1,1} & a_{i+1,2} & \dots & a_{i+1,j-1} & a_{i+1,j} & a_{i+1,j+1} & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{array} \right)$$

В этой матрице соседи углового элемента a_{11} и элемента a_{ij} выделены полужирным шрифтом.

Подсчитать количество локальных минимумов заданной матрицы размером 10×10 .

Найти сумму модулей элементов, расположенных выше главной диагонали.

Вариант 11

Коэффициенты системы линейных уравнений заданы в виде прямоугольной матрицы. С помощью допустимых преобразований привести систему к треугольному виду.

Найти количество строк, среднее арифметическое элементов которых меньше заданной величины.

Вариант 12

Уплотнить заданную матрицу, удаляя из нее строки и столбцы, заполненные нулями.

Найти номер первой из строк, содержащих хотя бы один положительный элемент.

Вариант 13

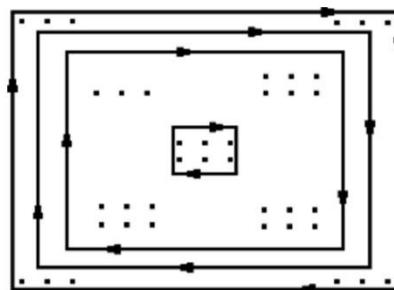
Осуществить циклический сдвиг элементов прямоугольной матрицы на n элементов вправо или вниз (в зависимости от введенного режима), n может быть больше количества элементов в строке или столбце.

ПРИМЕЧАНИЕ: Под циклическим сдвигом элементов понимается перестановка элементов строки или столбца по следующему правилу:

1, 2, 3, ..., n	->	n, 1, 2, 3, ..., n-1	->	n-1, n, 1, ..., n-2	->	n-2, n-1, n, 1, 2, ..., n-3
Исходное		Циклический		на 2 элем.		на 3 элем.
состояние		сдвиг вправо на				
		1 элем.				

Вариант 14

Осуществить циклический сдвиг элементов матрицы размером $M \times N$ (M строк $\times N$ столбцов). Сдвиг выполнить вправо на k элементов таким образом: элементы первой строки сдвигаются в последний столбец сверху вниз, из него - в последнюю строку справа налево, из нее - в первый столбец снизу вверх, из него - в первую строку; для остальных элементов - аналогично.



Вариант 15

Дана целочисленная прямоугольная матрица. Определить номер первого из столбцов, содержащих хотя бы один нулевой элемент.

Переставляя строки заданной матрицы, расположить их в соответствии с убыванием характеристик.

ПРИМЕЧАНИЕ: Характеристикой строки целочисленной матрицы назовем сумму ее отрицательных четных элементов.

Вариант 16

Упорядочить строки целочисленной прямоугольной матрицы по возрастанию количества одинаковых элементов в каждой строке.

Найти номер первого из столбцов, не содержащих ни одного отрицательного элемента.

Вариант 17

Путем перестановки элементов квадратной вещественной матрицы добиться того, чтобы ее максимальный элемент находился в левом верхнем углу (1,1), следующий по величине - в позиции (2, 2), следующий по величине - в позиции (3, 3) и т. д., заполнив, таким образом, всю главную диагональ.

Найти номер первой из строк, не содержащих ни одного положительного элемента.

Вариант 18

Дана целочисленная прямоугольная матрица. Определить:

1. Количество строк, содержащих хотя бы один нулевой элемент.
2. Номер столбца, в котором находится самая длинная серия одинаковых элементов.

Вариант 19

Дана целочисленная квадратная матрица. Определить:

1. Сумму элементов в тех строках, которые не содержат отрицательных элементов.
2. Минимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы.

Вариант 20

Дана целочисленная прямоугольная матрица. Определить:

1. Количество отрицательных элементов в тех строках, которые содержат хотя бы один нулевой элемент.
2. Номера строк и столбцов всех седловых точек матрицы.

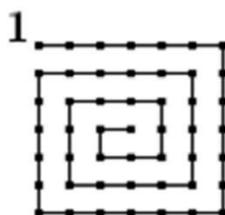
ПРИМЕЧАНИЕ:

Матрица А имеет седловую точку A_{ij} , если A_{ij} является минимальным элементом в i -й строке и максимальным в j -м столбце.

Вариант 21

Напишите программу, формирующую квадратную матрицу, элементы которой являются натуральными числами, расположенными в порядке возрастания от 1 до n^2 (n - порядок матрицы) согласно схеме, приведённой на рисунке.

Вычислить сумму элементов, расположенных на главной диагонали полученной матрицы.

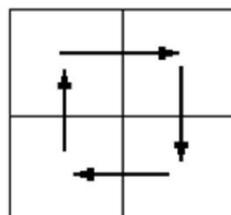


Вариант 22

Написать программу, которая меняет местами столбцы квадратной матрицы, содержащие наибольший и наименьший элементы и вычисляет сумму элементов главной диагонали.

Вариант 23

Квадратная матрица порядка $2n$ состоит из 4-х блоков. Написать программу, которая формирует новую матрицу, переставляя блоки исходной матрицы согласно схеме, и печатает сумму элементов каждого блока.



Вариант 24

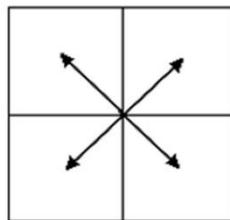
Имеется квадратная матрица с целочисленными элементами. Написать программу, которая столбцы заданной матрицы делает строками новой матрицы.

Для каждого значения элемента матрицы подсчитать количество элементов, которые принимают такое же значение. Подсчёт проводить лишь для тех значений, которые представлены в матрице.

Вариант 25

Квадратная матрица порядка 2^*n состоит из 4-х блоков. Написать программу, которая:

- формирует новую матрицу, переставляя блоки исходной матрицы согласно схеме;
- печатает произведение элементов каждого блока.



Вариант 26

Написать программу, которая заполняет матрицу размерности 4×13 числами от 1 до 52 случайным образом: повторение чисел не допускается.

Полученный результат вывести на экран монитора.

Вариант 27

Даны две матрицы одного порядка $M \times N$ (M строк $\times N$ столбцов).
Написать программу сложения, вычитания и транспонирования матриц.

1. Сложение и вычитание: $c_{ij} = a_{ij} \pm b_{ij}$

2. Транспонирование $b_{ij} = a_{ji}$

Вариант 28

Произведением двух матриц A_{mn} на B_{nl} называется такая матрица C_{ml} , для которой:

$$c_{ik} = a_{i1} \cdot b_{1k} + a_{i2} \cdot b_{2k} + \dots + a_{in} \cdot b_{nk} = \sum_{j=1}^n a_{ij} \cdot b_{jk}.$$

Т.е. элемент c_{ik} матрицы C равен сумме произведений элементов i -й строки матрицы A на соответствующие элементы k -го столбца матрицы B .

Написать программу вычисления произведения двух матриц.

Программа должна по заданным размерностям матриц сообщать о возможности получения такого произведения.

Вариант 29

Экспонента от (квадратной) матрицы A может быть определена следующим образом:

$$e^{\mathbf{A}} = \mathbf{I} + \mathbf{A} + \frac{1}{2!} \mathbf{A}^2 + \frac{1}{3!} \mathbf{A}^3 + \dots,$$

где \mathbf{I} - единичная матрица, у которой элементы главной диагонали равны единице, а остальные - нулю: $i_{mn} = \begin{cases} 1 & m = n \\ 0 & m \neq n \end{cases}$, $\mathbf{A}^2 = \mathbf{A} \times \mathbf{A}$.

Произведением двух матриц $A_{m,n}$ на $B_{n,l}$ называется такая матрица $C_{m,l}$, для которой:

$$c_{ik} = a_{i1} \cdot b_{1k} + a_{i2} \cdot b_{2k} + \dots + a_{in} \cdot b_{nk} = \sum_{j=1}^n a_{ij} \cdot b_{jk}.$$

Т.е. элемент c_{ik} матрицы C равен сумме произведений элементов i -й строки матрицы A на соответствующие элементы k -го столбца матрицы B .

Написать программу для вычисления матричной суммы первых n элементов ряда. Примеры представления единичной матрицы и матрицы размерностью 2×2 и вычисления квадрата матрицы:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \quad A^2 = \begin{pmatrix} 1 \cdot 1 + 2 \cdot 3 & 1 \cdot 2 + 2 \cdot 4 \\ 3 \cdot 1 + 4 \cdot 3 & 3 \cdot 2 + 4 \cdot 4 \end{pmatrix} = \begin{pmatrix} 7 & 10 \\ 15 & 22 \end{pmatrix}$$

Вариант 30

Имеется набор символов 0..9 и A..Z (всего 36 символов). Написать программу, которая:

- случайным образом заполняет элементы двухмерной матрицы размерности 6×6 . Повторение символов не допускается.

- выводит предложение, которое содержит не более 10 слов, составленных по следующему принципу:

- случайным образом задаётся число, которое является номером столбца или строки;
- элементы строки (столбца) формируют слово.

Лабораторная работа №6: "Файлы"

Цель работы

Дать студентам практический навык в написании программ, в которых выполняются операции с текстовыми файлами - чтение, запись.

Постановка задачи

В предыдущих заданиях необходимые для программы данные, вводились с клавиатуры, а результат выводился на экран монитора. Очевидно, что и при отладке программ, и при вводе большого объема данных в программу такой подход трудозатратен и потому непригоден. Наиболее подходящее решение – это ввод данных из файла и вывод результатов работы в файл. При этом входные данные подготавливаются однократно и с должным многообразием, а результаты работы можно анализировать многократно.

Напишем несколько программ, которые будут считывать входные данные из заранее подготовленного текстового файла, и выводить результат в текстовый файл. Для этой цели воспользуемся программами, написанными в предыдущих заданиях.

Теоретическое введение

Файловые типы данных введены в языках программирования для работы с внешними устройствами – файлами на диске, портами, принтерами и т.д. Файловые типы подразделяются на стандартные типы и типы, определяемые программистом. К стандартному типу относится текстовый тип файлов, с которым мы и будем работать.

Доступ к файлам может быть последовательным или прямым. При последовательном доступе каждый следующий элемент может быть прочитан только после выполнения аналогичной операции с предыдущим элементом. При прямом доступе операция чтения (записи) может быть выполнена для произвольного элемента с заданным адресом.

Текстовые файлы относятся к файлам с последовательным доступом. Они предназначены для хранения информации строкового типа. При этом ввод и вывод информации сопровождается преобразованием типов данных. При выводе в текстовый файл данные преобразуются из внутреннего представления в символы, а при вводе выполняется обратное преобразование.

Например, если в памяти некоторая переменная целого типа хранит в двух байтах значение 731_{10} , то в двоичном представлении это будет: 0000 0010 1101 1011₂, а в текстовом: 37 33 31_{string}.

Общий подход к работе с файлами, для всех языков программирования, строится на следующих представлениях:

- в операционной системе имеется файловая подсистема, обеспечивающая надежную работу с файлами в многопользовательском и многозадачном режимах при работе с различными внешними носителями: жесткие диски, флэш, DVD, ...;
- операционная система предоставляет функциональный набор (API-функции), который позволяет работать с файлами на логическом уровне. Программист не занимается управлением внешних устройств, контролем записей о файлах в папках, контролем качества записи, и т.д.

Для организации работы с файлами, при программировании на языке высокого уровня, выполняются, как правило, четыре шага:

- создается объект файла. Для этого используются подпрограммы, которые связывают имя файла, задаваемое пользователем, с переменной, которая хранит ссылку на специально создаваемую операционной системой структуру. Эта структура содержит информацию о

файле, о буфере данных, через который будет проходить обмен между программой и файлом и о текущем состоянии процесса обмена данными.

- задается режим обмена, в котором будет происходить работа с файлом: будет ли это режим чтения, записи, добавления или какой либо совмещенный режим, например, чтение и запись. Этот шаг реализуется либо после создания объекта файла, либо в процессе выполнения первого шага.

- производится запись или чтение данных. Процесс обмена данными между программой и файлом состоит в обмене данными между программой и буфером данных под управлением файловой подсистемы. При записи данных в буфер, файловая подсистема контролирует процесс записи и при заполнении буфера до некоторого уровня выполняет запись данных в файл, а буфер очищается, разрешая программе продолжать запись. При чтении данных из буфера файловая подсистема контролирует объем данных в буфере и, при необходимости, выполняет подкачку свежих данных из файла.

- выполняется операция закрытия файла. При этом остаток данных, находящийся в буфере записывается в файл и файл закрывается.

Операция закрытия файла обязательно должна выполняться, если файл был открыт на запись. Если файл не закрыть, то при завершении программы, ресурсы, выделенные операционной системой, будут закрыты. В этом случае может возникнуть состояние, когда файл окажется пуст (чаще всего) либо будет содержать не всю информацию, которую в него пытались записать. Это зависит от размера буфера, который в современных ОС может быть достаточно большим.

Файлы, открытые на чтение, так же необходимо закрывать. Для этого есть две причины:

- открытый на чтение файл блокируется и другие приложения не получают к нему доступа;
- и в программе, и в операционной системе есть ограничение на число открытых файлов.

И хотя, по вашему мнению, доступ к файлу из других приложений маловероятен, а число открытых файлов достаточно большое, не стоит испытывать судьбу, только для того, что бы получить сообщение об ошибке, а затем долго и мучительно искать ее причину.

В объектно-ориентированном языке программирования, как это, например, реализовано в Python, часть описанных шагов может быть реализована в скрытой форме.

Например, при создании файлового объекта первый и второй шаги выполняются в одной инструкции:

```
fh = open(<Имя_файла> [, mode = <mod>])
```

где fh – переменная, хранящая ссылку на файловый объект, <Имя_файла> – абсолютный или относительный путь и имя файла, mode=<mod> – режим в котором открывается файл: запись, чтение, добавление, ...

Язык Python поддерживает протокол менеджеров контекста. Этот протокол гарантирует правильное закрытие файла в независимости от того, произошло исключение внутри блока кода или нет. Например, следующий код открывает файл на запись, записывает в файл строки, закрывает файл, а затем вновь открывает его, выводит текст на экран и закрывает файл. Обратите внимание на то, что операция закрытия файла в явном виде в коде отсутствует:

```
with open(r"lab6.txt", "w", encoding="cp1251") as fh:  
    fh.write("Меркурий\n") # Запись строк в файл  
    fh.write("Венера\n")  
    fh.write("Земля\n")  
# В этом месте файл fh закрыт автоматически
```

```

with open(r"lab.6.txt", "r", encoding="cp1251") as fh:
    print(fh.read())
    # В этом месте файл fh так же закрыт

```

При создании объекта файла, необходимо указывать путь и имя существующего, либо будущего файла. Путь к файлу можно задавать как относительно текущей рабочей папки, так и абсолютно. При этом под относительным путем понимается путь относительно текущей рабочей папки, а под текущей рабочей папкой понимается папка, в которой находится пользователь в момент запуска файла (запускаемый файл может находиться в другой папке).

Для правильного понимания того, как может формироваться путь, следует обратиться к литературе, например [1] и провести самостоятельные эксперименты.

Далее рассмотрены модификации программ, написанных к лабораторным работам №1 и №5. В этих модификациях приведены способы работы с файлами.

Дополнение к лабораторной работе №1

В этой работе мы учились записывать выражения на языке Python. Внесем следующее изменение в нашу программу:

- подготовим текстовый файл с исходными данными
- используя инструкции для работы с текстовым файлом, прочитаем записанные строки
- результат запишем в текстовый файл в виде таблицы.

Напоминание: При написании программы длинную строку инструкции можно разместить на нескольких строках. Для этого используется символ обратного слэша, за которым, сразу, следует Enter или, например, круглые или квадратные скобки.

Вычисляемые выражения оформим в виде функций:

```

def f1(a,x):
    y = tan(x**2/2-1)**2+(2*cos(x-pi/6)) \
        / (1/2+sin(a)**2)
    return y

def f2(x):
    y = pow(2, log(3-cos(pi/4+2*x),3+sin(x)) \
        / (1+tan(2*x/pi)**2))
    return y

```

Текстовый файл

Установим следующий формат текстового файла:

– две строки – это шапка, в которой указано назначение столбцов. Эти строки снабдим символом комментария, который используется в Python: '#';

– два столбца, в которых записаны значения, для которых будут проводиться вычисления.

Пример:

```

# a      x
#-----
-2      -2
0       -2
...

```

Используем упрощенную схему работы с текстовым файлом.

Создать файловый объект:

```
fh = open(<Имя_файла> [, mode = <mod>]),
```

где `fh` – переменная, хранящая ссылку на создаваемый файловый объект, `<Имя_файла>` – абсолютный или относительный путь и имя файла, `mode=<mod>` – режим в котором открывается файл. Вместо `<mod>` необходимо подставить один из символов:

mod	Режим	Примечание
'r'	чтение файла (read)	файл должен существовать, если файл не существует, то возбуждается исключение: <code>FileNotFoundException</code>
'w'	запись в файл (write)	если файла несуществует, то он создается
'a'	добавление в файл (append)	если файла несуществует, то он создается, запись выполняется в конец файла
'r+'	чтение и запись	файл должен существовать, если файл не существует, то возбуждается исключение: <code>FileNotFoundException</code>
'w+'	чтение и запись	если файла несуществует, то он создается, существующий файл перезаписывается
'a+'	чтение и запись	если файла несуществует, то он создается, запись выполняется в конец файла
'x'	создать файл для записи	если файл существует, то возбуждается исключение: <code>FileExistsError</code>
'x+'	создать файл для чтения и записи	если файл существует, то возбуждается исключение: <code>FileExistsError</code>

Вместе с указанием режима может следовать модификатор, определяющий режим открытия файла: текстовый или бинарный.

Для решения нашей задачи нам необходимо открыть два файла. Один файл будет содержать информацию для расчета выражений и будет открыт на чтение, а второй – для вывода результатов расчета – будет открыт для записи:

```
fi = open("lab1_pb_in.txt", mode = "rt")
fo = open("lab1_pb_ou.txt", mode = "wt")
```

Читать файл:

Чтение файла можно организовать по-разному, например, считывать по строкам (метод `readline()`) или считать весь файл в буфер (метод `readlines()`) и затем обрабатывать строки. Обратим внимание на то, что строки заканчиваются символом конца строки (`'\n'`). Метод `readline()` читает строку, включая и символ конца строки.

При чтении по строкам итерацию (чтение следующей строки) можно выполнять через цикл `for`. Рассмотрим два примера:

1) `while True:`

```
        line = fi.readline()    # чтение строки
        if not line:            # строка пустая
            break               # конец файла и обработка
        elif line=="\n":         # конец строки
            continue            # продолжим чтение строк
        (b, c) = line.split()   # разделить строку
```

2) `for line in fi:`

```
        if line=="\n":          # конец строки
            continue            # продолжим чтение строк
        (b, c) = line.split("\t") # разделить строку
```

В первом примере итерации выполняются в цикле `while` при выполнении инструкции чтения строки. Считанное значение сохраняется в переменной `line`, и

проверяется на то, что получено не пустое значение. Если инструкция `fi.readline()` вернет пустую строку, то это значит, что прочитан конец файла (EOF) и дальнейшую обработку можно прекратить (`break`). Кроме этого проверяется, что строка содержит информацию. Если строка не содержит информации, то в ней будет только символ конца строки. В этом случае обработку следует продолжить с чтения следующей строки (`continue`).

Замечание: Если строка не содержит информацию (в строке нет символов, которые можно было бы визуализировать), то в ней есть символ конца строки. Если строка пустая, то в ней никаких символов нет.

Во втором примере итерации (чтение строк) выполняются циклом `for`. Строки по очередичитываются в переменную `line`. В этом случае нет необходимости контролировать конец файла (EOF).

Дальнейшая обработка считанной информации выполняется в соответствии с форматом записи данных.

В нашем примере мы поместили в начале файла две строки, описывающие данные, которые следуют за ними, а сами данные разместили по строкам в форме двух столбцов. Разделителем между столбцами может выступать знак табуляции или несколько пробелов.

При чтении такого файла поступим следующим образом:

- прочитаем две строки без обработки (пропустим эти строки);
- в цикле читаем строку, и расщеплять ее для получения данных.

Формат записи метода расщепления (разделения) следующий:

```
str.split(sep=None, maxsplit=-1),
```

где `str` – строка символов, `sep` – разделитель, а `maxsplit` – количество групп, на которые делится строка.

Если указан разделитель (`sep`) и количество групп `maxsplit`, то строка будет разделена на `maxsplit + 1` части. Если `maxsplit` не указан или равен `-1`, то число частей, на которые будет поделена строка неограничено. Пример разделения строки:

```
'1,2,3'.split(',',maxsplit=1) ==> ['1', '2,3']
```

Мы можем не указывать параметры в методе `split()`, поскольку при расщеплении будет формироваться список из двух значений (в строке два столбца), а разделителем мы выбрали пробелы или знак табуляции.

В левой части инструкции мы укажем две переменные, которые примут значения, полученные при расщеплении строки.

```
b, c = line.split()
```

Полученные при расщеплении значения будут строкового типа и для дальнейшего их использования необходимо выполнить преобразование к вещественному типу (`float`).

Писать в файл:

Для записи в файл будем использовать метод `write(<Данные>)`. Под данными тут выступает строка, в том числе и форматная строка, содержащая знаки форматирования. При записи строки в файл метод `write()` не добавляет символа конца строки. Для того, что бы следующие данные записывались с новой строки, необходимо, что бы текущая строка завершалась символом конца строки ('`\n`'). Пример использования вы найдете в листинге программы ниже.

Далее представлены два варианта программы и результаты ее работы. Во втором варианте для загрузки данных из файла использована функция модуля NumPy, а данные загружаются в двумерный массив (строка выделена жирным шрифтом).

Листинг программы

Первый вариант:

```
# -*- coding: cp1251 -*-
from math import *
def f1(a, x):
    y = tan(x**2/2-1)**2+(2*cos(x-pi/6)) \
        / (1/2+sin(a)**2)
    return y

def f2(x):
    y = pow(2, log(3-cos(pi/4+2*x), 3+sin(x)) \
            / (1+tan(2*x/pi)**2))
    return y

fi = open(r"lab1_pb_in.txt", "rt")      # читать файл
fo = open(r"lab1_pb_ou.txt", "wt")      # писать в файл
line = fi.readline()                  # Пропустить строки
line = fi.readline()                  # заголовка в файле
# Вывести шапку таблицы в файл
fo.write("=====+=====+=====+=====+\n")
fo.write("I    A    I    X    I    F1    I    F2    I\n")
fo.write("=====+=====+=====+=====+=====+\n")
for line in fi:                      # для всех строк файла
    if line=="\n":
        continue
    (b, c) = line.split()    # расщепить
    a = float(b)           # привести к вещественному типу
    x = float(c)
    # Вывод в файл
    fo.write("I {0: .2f} I {1: .2f} I {2: 6.4f} I" \
             .format(a, x, f1(a, x)))
    fo.write("{0: 6.4f} I\n".format(f2(x)))
    fo.write("-----+-----+-----+-----+\n")
# закроем файлы
fi.close()
fo.close()
```

Второй вариант:

```
# -*- coding: cp1251 -*-
from math import *
import numpy as np
def f1(a, x):
    y = tan(x**2/2-1)**2+(2*cos(x-pi/6)) \
        / (1/2+sin(a)**2)
    return y
def f2(x):
    y = pow(2, log(3-cos(pi/4+2*x), 3+sin(x)) \
            / (1+tan(2*x/pi)**2))
    return y

fi = open(r"lab1_pc_in.txt", "rt")      # читать файл
fo = open(r"lab1_pc_ou.txt", "wt")      # писать в файл
# Вывести шапку таблицы в файл
```

```

fo.write("=====+\n")
fo.write("I A I X I F1 I F2 I\n")
fo.write("=====+\n")
# Загрузим данные в массив
ax = np.loadtxt(fi,dtype=np.float, ndmin = 1)
nRow, nCol = ax.shape # Строк и колонок в массиве
for Row in range(nRow):      # для всех строк
    a = ax[Row][0]
    x = ax[Row][1]
    fo.write("I {0: .2f} I {1: .2f} I {2: 6.4f} I"\n
              .format(a, x, f1(a, x)))
    fo.write("{0: 6.4f} I\n".format(f2(x)))
    fo.write("=====+\n")
# закроем файлы
fi.close()
fo.close()

```

Результат работы программы (текстовый файл lab1_pb_out.txt)

```

=====+
I A I X I F1 I F2 I
=====+
I -2.00 I -2.00 I 1.1970 I 1.1184 I
-----+
I 0.00 I -2.00 I -0.8347 I 1.1184 I
-----+
I 0.00 I 0.00 I 5.8896 I 1.6880 I
-----+
I 2.00 I 0.00 I 3.7309 I 1.6880 I
-----+
I 1.50 I 0.50 I 2.7712 I 1.7955 I
-----+
I 4.00 I 3.00 I -1.3266 I 1.0517 I
-----+

```

Дополнение к лабораторной работе №5

Эту работу выполним в два шага. На первом шаге мы сформируем массив с использованием функции `random()` и сохраним его в текстовом формате. Формирование массива, его запись в файл и чтение из файла выполним с использованием функций модуля NumPy.

Для инициализации массива, вычисления среднего значения и дисперсии, а так же корректировки массива, воспользуемся ранее написанными функциями.

Небольшие изменения внесем в функцию генерации массива с тем, что бы можно было получать не только квадратный, но и прямоугольный массив.

Листинг программы

```

import numpy as np
from math import *

def MakeMatr(n, m, a = -5, b = 5):
    """ Инициализация прямоугольной матрицы
    размером nxm псевдослучайными величинами
    в диапазоне [a, b]"""
    Matr = a + (b - a) * np.random.random(size = (n,m))

```

```

        return Matr

def MidlDisp(Matr):
    """ Вычисление среднего значения
    Вычисление дисперсии """
    sum = 0
    (nRow, nCol) = Matr.shape # Размеры матрицы
    for Row in range(nRow):
        for Col in range(nCol):
            sum += Matr[Row][Col]
    Midl = sum / Matr.size      # Среднее
    Disp = 0
    for Row in range(nRow):
        for Col in range(nCol):
            Disp += (Matr[Row][Col] - Midl)**2
    return (Midl, Disp / (Matr.size - 1))

def MakeCorrect(Matr, aver, sigma):
    """ Замена "отскочивших" значений """
    (nRow, nCol) = Matr.shape # Размеры матрицы
    for Row in range(nRow):
        for Col in range(nCol):
            if abs(abs(Matr[Row][Col]) - aver) > sigma:
                Matr[Row][Col] = aver
    return (Matr)

n, m = input("Введите размеры матрицы (N M): ").split()
n = int(n)
m = int(m)
tstMatr=MakeMatr(n, m, -7, 7)      # приготовим матрицу
str_o = 'Resalt after initiation:'
fh_i = open(r"lab5_pd_in.txt", "wb") # для записи
np.savetxt(fh_i, tstMatr, fmt = ' %8.4f', \
          header = str_o)             # выгрузим ее в файл
fh_i.close()                      # закроем файл

# Выполняем задание
# Загрузим данные в массив
fh_i = open(r"lab5_pd_in.txt", "rt")
MyMatr = np.loadtxt(fh_i,dtype=np.float, ndmin = 1)
# Откроем файл для результатов работы программы
fh_o = open(r"lab5_pd_ou.txt", "wb") # файл результата
# Сохраним исходный массив
str_o = 'Before correction:'
np.savetxt(fh_o, MyMatr, fmt = ' %8.4f', header = str_o)

# Вычисляем среднее и дисперсию
mMidl, mDisp = MidlDisp(MyMatr)
# подготовим строки для записи
str_o = '\nAfter correction:'
str_o1 = '\nMidl = {0:5.4f} Disp = {1:6.4f}\n' \
         'Sigm = {2:6.4f}' \

```

```

        .format(mMidl, mDisp, sqrt(mDisp))
# Корректируем массив
NMatr = MakeCorrect(MyMatr, mMidl, sqrt(mDisp))
# Сохраняем результат работы в файле в формате:
# Заголовок
# Скорректированный массив
# Завершающая часть
np.savetxt(fh_o, NMatr, fmt = ' %8.4f', \
           header = str_o, footer = str_o1)
fh_o.close()                                # закроем файл

```

Результат работы программы (текстовый файл lab1_pd_out.txt)

```

# Before correction:
  3.3232   -1.7425   -5.2076
  6.2726   -0.8202    6.1146
  4.6182    0.2746   -2.9510
  2.6626   -3.9889   -5.1610
#
# After correction:
  3.3232   -1.7425    0.2829
  0.2829   -0.8202    0.2829
  0.2829    0.2746   -2.9510
  2.6626   -3.9889    0.2829
#
# Midl = 0.2829 Disp = 17.9449   Sigm = 4.2361

```

Задание к лабораторной работе №6 "Файлы"

Выполнить корректировку программ, написанных для лабораторных работ №1, №4 и №5, с таким условием, что бы ввод данных и вывод результатов работы осуществлялся с использованием файлов.

Лабораторная работа №7: "GUI, классы, модуль Tkinter"

Цель работы

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

ЗАМЕЧАНИЕ: Студенты не ограничиваются выбором модуля и способов решения заданий этой лабораторной работы. В качестве средства для написания GUI можно использовать и другие модули и библиотеки, например PyQt 5.

Постановка задачи

В текстовом файле workers.csv хранится база отдела кадров предприятия. На предприятии не более 100 сотрудников. Каждая строка содержит запись об одном сотруднике. Форма записи: фамилия и инициалы (30 позиций), фамилия должна начинаться с первой позиции), код подразделения (4 позиции), номер телефона (4 позиции), дата рождения (10 позиций), код должности (5 позиций), оклад (10 позиций). Наименование подразделения хранится в текстовом файле workers.subd. Форма записи: код подразделения (4 позиции), полное наименование подразделения (30 позиций). Перечень должностей хранится в текстовом файле workers.post. Формат: код должности (5 позиций), полное наименование должности (15 позиций).

Все файлы имеют формат CSV – Comma-Separated Values.

Написать программу, которая:

- позволяет вводить данные о сотруднике и корректировать их;
- контролирует ввод данных: проверяются поля, которые обязательны для заполнения. Например, поля для ввода фамилии, кода подразделения, кода должности, зарплаты;
- сохраняет данные в текстовых файлах в формате CSV;
- позволяет вводить данные и редактировать справочники с перечнем подразделений и должностей.
- позволяет просматривать данные о сотруднике;
- по заданному наименованию или части наименования подразделения выводит полное наименование подразделения, перечень сотрудников и подсчитывает средний оклад сотрудников этого подразделения;
- по системной дате выводит фамилии сотрудников, у которых на текущую дату и следующий день выпадает день рождения.

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 файлов, 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 (Comma-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
Мусин Д.	Самоучитель Python	2016	3	Язык Python

В существующей спецификации рассматриваются следующие моменты:

- Каждая строка файла – это одна строка таблицы. Стока 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 обычные 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 году 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. 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 наших 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', 'S30')])
```

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.

Листинг программы

```
import pandas as pd
import numpy as np
from datetime import datetime

# Принудительное расширение окна вывода
pd.set_option('expand_frame_repr', False)
```

```

dt = np.dtype([('avtor', 'S15'), ('title', 'S30'),
               ('year', 'int16'), ('copy', 'int16'),
               ('note', 'S30')])
mas = np.empty(20, dtype = dt)
# Читаем данные в DataFrame структуру
df = pd.read_csv('booksN.csv', sep=',',
                  encoding = 'cp1251')
# Формируем массив записей
mas = df.to_records(index = False) # False - столбец
# индексов
# не переносится
# в массив
print(mas)
now = datetime.now()
print('Издание: \n', mas[2], '\n',
      now.year - mas[2]['year'], 'лет давности.')
# Заменяем автора
tmp = mas[0]['avtor']
mas[0]['avtor'] = mas[2]['avtor']
mas[2]['avtor'] = tmp
# Переводим массив записей в структуру DataFrame
df1 = pd.DataFrame.from_records(mas)
# Пишем данные в файл
df1.to_csv('booksNa.csv', sep=',', index = False)
# index = False или None - столбец индексов не попадает
# в файл
#      = True - столбец индексов пишется в файл,
#      но без заголовка
print(df1)

```

Следует отметить, что при формировании массива записей указывать структуру и тип данных необязательно. Встроенные компоненты модуля `pymr` выполняют парсинг (разборку строки) и самостоятельно определяют типы считанных данных.

Для проверки выше описанного кода выполните следующие действия:

- создайте файл с программой (текст можно скопировать, если у вас электронный образ пособия);
- в этой же папке разместите файл `booksN.csv` со следующим содержимым:

```

avtor,title,year,copy(note
Мусин Д.,Введение в научный Python,2014.0,2,"Классы, оконные приложения"
Travis E.,Guide to NumPy,,5,Руководство
Доля П.Г.,Самоучитель Python,2016.0,3,Язык Python

```

- запустите IDLE Python 3, откройте и запустите программу на выполнение.
- ознакомьтесь с выведенной информацией и сформировавшимся новым файлом.

Обратите внимание на то, как можно работать с датой, а так же на замену автора издания.

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.

На следующем рисунке представлена структура программы, управляемой событиями.

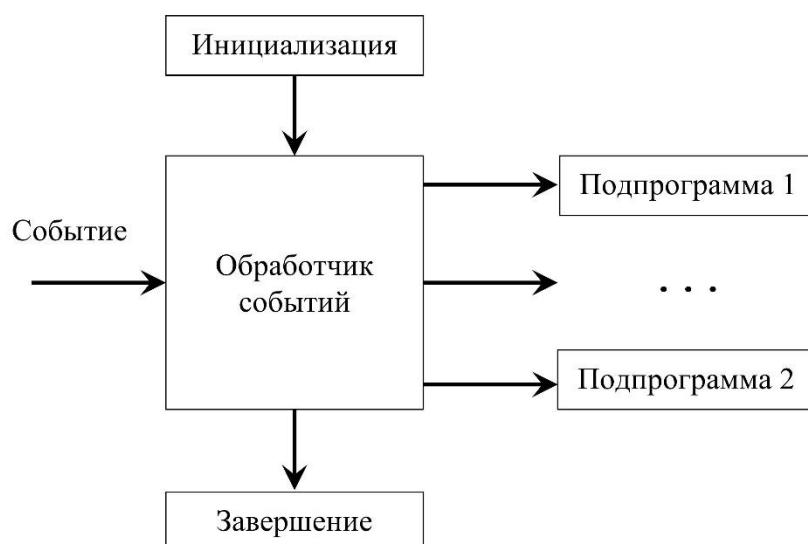


Рис.1. Структура программы, управляемой событиями

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 **контроль** (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 представляет 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;
- односторочное 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 односторочного 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, например, 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 индивидуальными характеристиками 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 одностороннем поле (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.

Больше информации о виджетах и методах их упаковки можно получить в соответствующей литературе или в сети Интернет. Часть информации размещена в приложении к этому пособию.

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

Классы

Далее предполагается, что студент получил необходимые знания о классах и объектах и у него есть представление о том, что такое класс, объект, методы и свойства класса, а также, что представляет собой понятие "передача сообщений". Кратко.

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 квантаций (ответов).

Рассмотрим примеры использования классов при написании GUI. Отметим, что применение классов не является панацеей и, в простых случаях, можно написать GUI без применения классов.

Программирование GUI с использованием классов рассмотрим на примере статьи Микки Нардо³. Первый пример - создание окна в Tkinter.

³ Перевод на русский язык: Филипп Занько, 2016 год.

Создайте файл window_01.py:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

# Импорт модулей python
from tkinter import *

# Создание окна
root = Tk()
root.title('myWindow')
root.geometry('200x150+300+225')

# Вывод окна на экран
root.mainloop()
```

Поясним приведенный пример.

Первые две строки необязательны для Python 3 под ОС Windows, но для ОС Linux они должны быть вставлены. Интерпретатор команд Linux первые строки воспринимает иначе. Первая строка указывает, какое приложение должно быть запущено для обработки следующего далее текста и где это приложение размещается в системе. Вторая строка указывает, какая кодировка символов использована в тексте.

В третьей строке задана команда, которая загружает модуль Tkinter⁴ целиком.

В следующих трех строках создается окно, и задаются его свойства (параметры): определяется надпись в заголовке окна и его геометрия.

Строка кода `root = Tk()` создает объект `root`, используя класс `Tk()`. Имя объекта выбирается нами, но принято главное (корневое) окно называть `root` (рус. корень). В следующих двух строках вызываются методы класса `Tk()`, которые теперь являются методами объекта `root`.

Задается заголовок окна `myWindow`, его размеры и положение на экране монитора. Размеры окна устанавливаются в 200x150 пикселов, а координаты верхнего левого угла – `x = 300, y = 250`.

Начало отсчета системы координат находится в левом верхнем углу экрана монитора. Заметим, что координаты могут принимать и отрицательные значения. В этом случае координаты будут указывать положение правого нижнего угла окна относительно правого нижнего угла экрана монитора. Закрепите это практическим примером. Например, установите следующие значения:

```
root.geometry('200x150+30+20')
root.geometry('200x150-30+20')
```

Подумайте, не нарушается ли наше правило во второй версии задания геометрии окна?

В последней строке производится вызов метода `mainloop()`. Этот метод держит окно открытым до тех пор, пока оно не будет закрыть кнопкой окна "Закрыть" ([X]) или не будет вызван метод `destroy()`.

Обратитесь к рис.1. Три строки, создающие окно нашей программы – это стадия "Инициализация", а последний шаг – это переход в стадию "Обработчик событий".

Web-адрес перевода: <http://www.russianlutheran.org/python/python.html>

⁴ В Python 2 модуль Tkinter обязательно начинается с заглавной буквы. В Python 3 имя этого модуля начинается с прописной буквы - `tkinter`. Далее, в примерах используется имя с маленькой буквы, т.к. скрипты написаны для Python 3. В тексте пособия имя пишется с большой буквы – как имя собственное.

Закрытие окна через кнопку "Закрыть" или вызов метода `destroy()` переводит программу в стадию "Завершение".

Добавим в нашу программу три виджета: поле `Entry`, метку `Label` и кнопку `Button`. Связем их методами, которые будут отрабатывать при нажатии клавиши `Enter` в поле `Entry` и кнопки `Button`. Состояние метки будет отображать результат наших действий.

Создайте файл `window_01a.py`:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

# Импорт модулей python
from tkinter import *

clk = 0
def btn_click():
    global clk
    lba.config(text = "Нажата кнопка!",
               bg = '#660066', fg = 'white')
    clk += 1
    root.title("myWindow: {}".format(clk))

def ent_enter(event):
    global clk
    txt = ent.get()
    lba.config(text = "Текст: " + txt,
               bg = 'yellow', fg = 'blue')
    clk -= 1
    root.title("myWindow: {}".format(clk))

# Создание окна
root = Tk()
root.title('myWindow:')
root.geometry('200x150+50+50')
#
# поле Entry
ent = Entry(root)
ent.config(fg = "blue", font = 'Courier_New 10')
ent.insert(0, 'ABCD')
ent.bind('<Return>', ent_enter)
ent.pack(side = TOP)
#
# метка Label
lba = Label(root)
lba.config(text = 'Метка')
lba.pack(side = TOP)
#
# кнопка Button
btn = Button(root)
btn.config(text = 'Нажми меня!',
           command = btn_click)
btn.pack(side = TOP)

# Вывод окна на экран
```

```
root.mainloop()
```

Обратите внимание на следующие моменты:

- виджеты привязываются к окну, в котором они размещаются, см., например, строки вида:
 ent = Entry(root)
- для конфигурирования виджетов используется метод config(), в котором определяются характеристики виджета;
- упаковка виджетов выполнена методом pack().

После ввода текста в поле Entry и нажатии клавиши "Enter" возникает событие '<Return>', которое связано с методом ent_enter. В этом методе считывается введенный текст, а затем выводится в поле метки с установленным цветом фона и символов. Дополнительно, выполняется декрементация счетчика, и вывод нового значения в заголовке формы (при необходимости растяните форму по горизонтали).

При нажатии на кнопку 'Нажми меня' управление передается методу btn_click. В этом методе выполняется вывод текста "Нажата кнопка!" с новыми значениями цвета фона и текста. Кроме этого выполняется инкремент счетчика и вывод нового значения в заголовке формы.

И так, мы познакомились с виджетами, способами их настройки, а также с методами и способами изменения конфигурации виджетов.

Вновь сопоставьте рис.1 с текстом программы. Блоки подпрограмм, изображенные на рисунке – это методы ent_enter и btn_click, которые обрабатывают события.

Следующий шаг – это создание дочернего окна.

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:

```
#!/usr/bin/python
# -*- 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**.

Попробуем объектно-ориентированными средствами описать последний пример: создание главного и дочернего окон. Код программы может выглядеть, например, так:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

# импорт модулей python
from tkinter import *

# класс родительских окон
class main:
    def __init__(self, master):
        self.master = master
        self.master.title('myWindow')
        self.master.geometry('200x150+200+150')
        child(self.master)
        self.master.mainloop()

# класс дочерних окон
class child:
    def __init__(self, master):
        self.slave = Toplevel(master)
        self.slave.title('child')
        self.slave.geometry('200x150+400+300')

# создание окна
root = Tk()

# запуск окна
main(root)
```

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.;

- строка `child(self.master)` – это вызов другого класса, определенного в текущей программе (модуле). В классе `child()` написан свой метод инициализации класса.

Для лучшего понимания программы проведите несколько экспериментов, например, определяя положение и цвет окна. В следующем примере показан способ задания цвета фона окон.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

# импорт модулей python
from tkinter import *

# класс родительских окон
class main:
    def __init__(self, master, main_color, child_color):
        self.master = master
        self.master.title('myWindow')
    # Примеры цвета: 'white', 'black', 'red', 'green',
    # 'blue', 'cyan', 'yellow', and 'magenta'
        self.master.configure(bg = main_color)

        self.master.geometry('200x150+200+150')
        child(self.master, child_color)
        self.master.mainloop()

# класс дочерних окон
class child:
    def __init__(self, master, child_color):
        self.slave = Toplevel(master)
        self.slave.title('child')
        self.slave.configure(bg = child_color)
        self.slave.geometry('200x150+400+300')

# создание окна
root = Tk()

# запуск окна
main(root, 'green', 'yellow')
```

На следующем шаге добавим в классе `main` кнопку, нажатие на которую (клик мыши на кнопке) будет открывать дочернее окно. Для этого опишем виджет `Button` и метод `openDialog()`, в котором будет открываться дочернее окно.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

# импорт модулей python
from tkinter import *

# класс главного окна
class main:
```

```

def __init__(self, master):
    self.master = master
    self.master.title('parent')
    self.master.geometry('200x150+300+225')
    self.button = Button(self.master,
                         text = 'myButton',
                         command = self.openDialog)
    self.button.pack(side = BOTTOM)
    self.master.mainloop()

def openDialog(self):
    child(self.master)

# класс дочерних окон
class child:
    def __init__(self, master):
        self.slave = Toplevel(master)
        self.slave.title('child')
        self.slave.geometry('200x150+500+375')
        self.slave.grab_set()
        self.slave.focus_set()
        self.slave.wait_window()

# создание окна
root = Tk()

# запуск окна
main(root)

```

Событие "нажатие кнопки" свяжем с методом `openDialog()`. Таким образом, в классе `main` мы реализуем два метода: метод инициализации окна:

```
__init__(self, master)
```

и метод, который создает экземпляр объекта `child`:

```
openDialog(self).
```

Для передачи сообщений в дочернее окно можно воспользоваться методом инициализации, добавляя параметры, как это сделано в примере выше с передачей значения цвета.

А как передавать сообщения от дочернего окна к родительскому окну? Для ответа на этот вопрос нам потребуется понятие модального дочернего окна.

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, что бы 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()`:

<code>self.slave.grab_set()</code>	child перехватывает все события, происходящие в приложении;
<code>self.slave.focus_set()</code>	child захватывает фокус;
<code>self.slave.wait_window()</code>	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.

Рассмотрим пример, в котором реализован обмен сообщениями в обоих направлениях: от основного окна к дочернему окну и наоборот. Создайте файл со следующим текстом и запустите программу.

```

#!/usr/bin/python
# -*- coding: utf-8 -*-

# импортирование модулей python
from tkinter import *

# класс главного окна
class main:
    def __init__(self, master):
        self.master = master
        self.master.title('parent')
        self.master.geometry('200x150+300+225')
        self.button = Button(self.master,
                             text = 'dialog',
                             command = self.openDialog)
        self.button.pack(side = BOTTOM)
        self.text = Text(self.master,
                        background = 'white')
        self.text.pack(side = TOP,
                      fill = BOTH,
                      expand = YES)
        self.master.mainloop()

    def openDialog(self):
        self.dialog = child(self.master)
        self.sendValue = self.text.get('0.0', END)
        self.returnValue = self.dialog.go(self.sendValue)
        if self.returnValue:
            self.text.delete('0.0', END)
            self.text.insert('0.0', self.returnValue)

# класс дочернего окна
class child:
    def __init__(self, master):
        self.slave = Toplevel(master)
        self.slave.title('child')

```

```

        self.slave.geometry('200x150+500+375')
        self.button = Button(self.slave,
                             text = 'accept',
                             command = self.accept)
        self.button.pack(side = BOTTOM)
        self.text = Text(self.slave,
                         background = 'white')
        self.text.pack(side = TOP,
                      fill = BOTH,
                      expand = YES)

    def go(self, myText = ''):
        self.text.insert('0.0', myText)
        self.newValue = None
        self.slave.grab_set()
        self.slave.focus_set()
        self.slave.wait_window()
        return self.newValue

    def accept(self):
        self.newValue = self.text.get('0.0', END)
        self.slave.destroy()

# создание окна
root = Tk()

# запуск окна
main(root)

```

Как эта программа работает:

Команда `main(root)` создает объект главного окна. В этом окне создаются два виджета `Text` и `Button`. Виджет `Text` позволяет вводить любой текст, а виджет `Button` – создает объект `dialog` (дочернее окно) и вызывает метод `go()` этого объекта:

```
self.returnValue = self.dialog.go(self.sendValue)
```

В дочернем окне так же создаются два виджета: `Text` и `Button`. Текстовое поле используется для ввода строки, а кнопка – для закрытия дочернего окна: метод `accept()`.

Метод `go()` дочернего окна принимает сообщение и вставляет его в виджет `Text`, а затем резервирует переменную, через которую сообщение будет передано от дочернего окна к родительскому окну. При этом дочернее окно преобразуется в модальное, что позволяет перехватить событие закрытия дочернего окна и вернуть новый текст главному окну:

```

...
self.slave.grab_set()
self.slave.focus_set()
self.slave.wait_window()
return self.newValue

```

После открытия главного окна пользователь вводит текст и нажимает кнопку '`dialog`'. Событие, связанное с нажатием кнопки вызывает метод `openDialog()`. В этом методе создается дочернее окно, копируется текст, введенный пользователем и вызывается метод `go()` с параметрами.

Метод `go()` дочернего окна вставляет текст, полученный через параметр `myText`, резервирует переменную для возврата результата, переводит дочернее окно в режим модального. При этом создается цикл ожидания закрытия дочернего окна:

```
self.slave.wait_window()).
```

Далее пользователь вводит текст в дочернем окне и нажимает кнопку 'accept'. Нажатие этой кнопки вызывает соответствующий метод `accept()`, в котором происходит сохранение текста в переменной `newValue`, ранее зарегистрированной в методе `go()`, и закрытие окна методом `destroy()`. Событие закрытия модального окна перехватывается внутренним циклом ожидания, созданным в методе `go()`.

Таким образом, процесс работы дочернего окна возвращается в метод `go()` и завершается передачей нового сообщения в главное окно:

```
return self.newValue
```

Попробуйте самостоятельно разобраться в описании работы программы с использованием рис.1.

Подсказка: Рассмотрите два рисунка, первый – главное окно, а второй – дочернее.

Ответьте на вопросы:

Какие процессы можно отнести к блоку "Инициализации", а какие к блоку "Завершение" в случае рассмотрения рисунка для дочернего окна.

Для рассмотрения листинга программы, реализующей поставленную вначале задачу этого достаточно. Более подробную информацию и ответы придется искать в сети Интернет и на форумах.

Листинг программы

```
# Подгружаемые модули
import os
import pylab as pl
import numpy as np
import pandas as pd
#
from tkinter import *
from tkinter import font
from tkinter.ttk import *
from tkinter.messagebox import *
from tkinter.filedialog import *
from datetime import datetime
#
# класс главного окна: реализация методов и свойств
class main:
#
# Описание констант
    ftypes = [ ("CSV-формат", ".csv"),
               ("Справочник", ".dic"),
               ("Все файлы", ".*")]
    defext = ".csv"      # тип файла по умолчанию
    Workers = np.empty(20) # данные о работниках
    Subd = np.empty(20)   # справочник "Подразделения"
    Post = np.empty(20)   # справочник "Должность"
    Fname = "workers.csv" # имя файла данных
    Fload = False         # загрузка не выполнена
    Fsave = False         # сохранение не выполнено
    Nfind = 0             # номер записи при поиске
#
# Инициализация класса
    def __init__(self, master = None):
```

```

        self.master = master
# размер окна: Width x Height + X + Y
# X >= 0 - от левой границы экрана до левой границы окна
# X < 0 - от правой границы экрана до правой границы окна
# Y >= 0 - от верхней границы экрана до верхней границы окна
# Y < 0 - от нижней границы экрана до нижней границы окна
        self.master.geometry("400x210+300+250")
# заголовок окна
        self.master.title("Lab7 - Ver: 0 - 2017")
# создание каскадного меню
        self.myMenu()
# создание окна (фрейма)
        self.myFrame()
# закрытие окна через кнопку управления окном - X - "Закрыть",
# событие закрытия вызывает метод close_win
        self.master.protocol('WM_DELETE_WINDOW',
                            self.close_win)
# Запрет на изменение размеров окна
        self.master.resizable(width = False, height = False)
# Настройка параметров текста для всех виджетов: фонт и размер
        newfont = font.Font(family="Courier New", size = 10)
        self.master.option_add('*Font', newfont)
# главный цикл ожидания событий
        self.master.mainloop()
#
# ===== Реализация методов главной формы =====
# Этот метод создает каскадное меню
def myMenu(self):
    m = Menu(self.master)          # объект Меню на главном окне
    self.master.config(menu = m)   # конфигурация окна с меню
#
# Каскадное меню "Файл" с доп. пунктами
# Команды меню размещаются на основном меню (self.m)
    self.fm = Menu(m)
    m.add_cascade(label = "Файл", menu = self.fm)
# список команд
    self.fm.add_command(label = "Открыть...",
                        command = self.open_f)
    self.fm.add_command(label = "Сохранить",
                        command = self.save_f)
    self.fm.add_command(label = "Сохранить как...",
                        command = self.saveas_f)
    self.fm.add_command(label = "Выход",
                        command = self.close_win)
#
# Каскадное меню "Редактировать" с доп. пунктами
    self.em = Menu(m)
    m.add_cascade(label = "Редактировать", menu = self.em)
    self.em.add_command(label = "Подразделение",
                        command = self.edit_sd)
    self.em.add_command(label = "Должность",
                        command = self.edit_p)
    self.em.add_command(label = "Работник",
                        command = self.edit_worker)
#
# Каскадное меню "Справка" с доп. пунктами
    self.hm = Menu(m)
    m.add_cascade(label = "Справка", menu = self.hm)

```

```

        self.hm.add_command(label = "Помощь",
                             command = self.about_h)
        self.hm.add_command(label = "О программе",
                             command = self.inf_h)
        self.hm.add_command(label = "О работнике",
                             command = self.Worker_h)
        self.hm.add_command(label = "О подразделении",
                             command = self.subd_h)
        self.hm.add_command(label = "День рождения",
                             command = self.bday_h)

#
# ===== Методы Меню =====
# Меню "Файл". Команда - "Открыть..."
def open_f(self):
    """
    Чтение данных.
    Заполнение массива персонала и справочников
    """

# Подгружаем данные о работниках
    if self.Fload:                      # Загрузка была сделана ранее
        if not self.Fsave:                # а сохранение не выполнено
            msg = "Сохранить текущие данные и\n\
                     продолжить загрузку?"
            self.dialog = askyesno("Загрузка",msg,
                                    icon = QUESTION)
            if self.dialog:
                self.save_f() # Сохраним
# открыть окно для выбора файла данных (Fname)
    fn = askopenfilename(defaultextension = self.defext,
                          filetypes = self.ftypes,
                          initialfile = self.Fname)
    if not fn:                           # Файл не выбран
        if not self.Fload:               # загрузки не было
            showerror("Ошибка", "Данные не загружены!")
            return
    else:                                # файл выбран
        tfn, ext = fn.split('. ',1)
        if ("." + ext).lower() != self.defext: # не .csv
            showerror("Ошибка", "Загрузка только csv-файлов!")
            return

#
# Загружаем данные
    self.Fname = fn          # Fname - актуальное имя файла
# df - данные в формате DataFrame
    df = pd.read_csv(self.Fname, sep=',',
                     encoding = 'cp1251',
                     comment = '#')
# перенесем данные в массив NumPy (а надо ли это делать?)
# индексацию из DataFrame не используем
    self.Workers = df.to_records(index = False)
#
# Подгружаем справочники подразделений и должностей
# получим имена файлов справочников и загрузим данные
    tfn, ext = self.Fname.split('. ',1)
    fdic = tfn + ".subd"
    df = pd.read_csv(fdic, sep = ',',
                     encoding = 'cp1251',
                     comment = '#')

```

```

        self.Subd = df.to_records(index = False)
#
        fdic = tfn + ".post"
        df = pd.read_csv(fdic, sep = ',',
                          encoding = 'cp1251',
                          comment = '#')
        self.Post = df.to_records(index = False)
#
        self.Fload = True      # данные загружены,
        self.Fsave = False     # но не сохранены
#
# Команда - "Сохранить"
def save_f(self):
    """
    Сохранение данных в файле Fname
    """
    if not self.Fload:
        showerror("Error", "Данные не загружены!")
        return
#
# Сохраняем персонал
    df = pd.DataFrame.from_records(self.Workers)
    df.to_csv(self.Fname, sep=',', index = False)
# файлы справочников
    tfn, ext = self.Fname.split('. ',1)
    fdic = tfn + ".subd"                      # подразделения
    df = pd.DataFrame.from_records(self.Subd)
    df.to_csv(fdic, sep=',', index = False)
    fdic = tfn + ".post"                      # должности
    df = pd.DataFrame.from_records(self.Post)
    df.to_csv(fdic, sep=',', index = False)
#
    self.Fsave = True      # сохранили данные
#
# Команда - "Сохранить как ..."
def saveas_f(self):
    """
    Сохранение данных в новом файле
    Fname заменяется на новое
    """
    if not self.Fload:
        showerror("Error", "Данные не загружены!")
        return
#
# Сохраняем данные в новом файле
# имя файла данных по умолчанию - текущее имя файла
    fini = self.Fname
    fn = asksaveasfilename(defaultextension = self.defext,
                           filetypes = self.ftypes,
                           initialfile = fini)
    if not fn:
        showerror("Error", "Файл не выбран!")
        return
    tfn, ext = fn.split('. ',1)
    self.Fname = tfn + self.defext  # получили имя нового файла
    self.save_f()                  # сохранили данные
#
# Команда - "Выход"
def close_win(self):

```

```

    """
    Завершить работу
    При необходимости сохранить данные
    """
    msg1 = ("Завершить работу - Да\n"
            "Продолжить работу - Отмена")
    msg2 = ("Завершить работу и СОХРАНИТЬ данные - Да\n"
            "Завершить работу не сохраняя данные - Нет\n"
            "Продолжить работу - Отмена")
# Когда данные загружены и не сохранены
    if (self.Fload and (not self.Fsave)):
        ans = askyesnocancel("Выход", msg2)
        if ans:
            self.saveas_f()          # сохранить с выбором файла
        elif ans is None:
            return
# Когда данные сохранены
    else:
        if not askokcancel("Выход", msg1):
            return
# Закрываем окно и выгружаем программу
    root.destroy()
#
#-----
-
# Меню "Редактировать": Команда - "Подразделение"
def edit_sd(self):
    if not self.ControlLoaded("Подразделение"):
        return
    self.newD = Edit_dic(self.master, "Подразделение")
    self.Subd = self.newD.Change(True, self.Workers, self.Subd)
#
# Меню "Редактировать": Команда - "Должность"
def edit_p(self):
    if not self.ControlLoaded("Должность"):
        return
    self.newD = Edit_dic(self.master, "Должность")
    self.Post = self.newD.Change(False, self.Workers, self.Post)
#
# Редактируем справочник Работники
def edit_worker(self):
    if not self.ControlLoaded("Работник"):
        return
    self.newWk = Worker3(self.master, "Работник")
    self.Workers = self.newWk.Change(self.Workers,
                                    self.Subd,
                                    self.Post)
#
# ----- Меню "Справка" -----
# Меню "Справка": Команда - "Помощь"
def about_h(self):
    msg = 'В этом методе можно реализовать' \
          '\n' \
          'открытие pdf-файла,\n' \
          'содержащего необходимые руководства.\n' \
          'Закройте окно и посмотрите результат.'
    showinfo("Помощь", msg, icon = INFO)
# Указание абсолютного пути. r - модификатор для спец. символов
# Если модификатор не указать, то символ '\' надо экранировать - '\\'

```

```

# fname = r'E:\University\Metodichka\Metod_Python\Combobox.pdf'
# Пример указания относительного пути.
# Файл программы и pdf - файл должны находиться в одной папке.
    fname = 'Combobox.pdf'
    os.startfile(fname)
#
# Меню "Справка": Команда - "О программе"
def inf_h(self):
# Создается форма с меткой
    win = Toplevel(root)
    win.resizable(width = False, height = False)
    txt = ('Это демонстрационная программа.\n'
            'В этой программе используются виджеты '
            'модуля tkinter.\n'
            'Демонстрируются:\n'
            '- способы использования виджетов;\n'
            '- формирование классов;\n'
            '- организация обмена данными между ними.\n'
            '- работа с файлами;\n'
            '- работа с массивами записей.\n\n'
            'Версия: 1.0 - 2017')
    lab = Label(win, text = txt, justify = 'left' )
    lab.pack()
#
# Меню "Справка": Команда - "О работнике"
def Worker_h(self): # Заполнение полей о работнике
    if not self.ControlLoaded("О работнике"):
        return
# для вывода данных используем форму и методы класса Worker2
    self.WorkerH = Worker2(self.master, "О работнике")
# классы Worker2 передаем необходимые данные
    self.WorkerH.Change(self.Workers, self.Subd, self.Post)
#
# Меню "Справка": Команда - "О подразделении"
def subd_h(self):
    """
    Данные о подразделении:
    - список работников
    - средняя зарплата
    """
#
    def butSubd1_clk(): # поиск назад
        Subd_data(-1)
#
    def butSubd2_clk(): # поиск вперед
        Subd_data(1)
#
    def Subd_data(stp): # формирование списка работников
# Имя или часть имени подразделения
        mystr = ent_Subd2.get().lower()
        if not mystr: # пустое поле
            msg = 'Не заполнено поле "Найти"!'
            showinfo('О подразделении', msg)
            return
# Поиск имени или его части в списке подразделений
        CSubd = 0 # код подразделения
        NSubd = '' # найденное имя
#
# Поиск со следующей позиции

```

```

        i = self.Nfind
        while True:
# i принимает значения в диапазоне [0 :-: self.Subd.size - 1]
            i += stp
            if i >= self.Subd.size:
                i = 0
            elif i < 0:
                i = self.Subd.size - 1
# Если совпадение, то сохраним значения и выйдем из цикла
            if
str(self.Subd[i]['name']).lower().startswith(mystr):
                CSubd = self.Subd[i]['cod']
                NSubd = self.Subd[i]['name']
                self.Nfind = i      # текущая запись
                break
            if i == self.Nfind:    # прошлись по кругу
                break              # завершим поиск
#
# Подготовим список работников и значение для ср. зарплаты
            ent_Subd1.delete(0, END)    # очистим поле
            ent_Subd1.insert(0, NSubd)  # выводим имя
#
            msg = ''                  # список работников
            mSum = 0                   # средняя зарплата
            k = 0                      # работников в подразделении
            for i in range(self.Workers.size):
                if self.Workers[i]['cod_sd'] == CSubd:
                    msg += '{0:15s}\n'.format(self.Workers[i]['fio'])
                    k +=1
                    mSum += self.Workers[i]['salary']
                if not k:               # нет данных
                    mSum = 0
                    msg = 'Нет данных.'
                else:                  # есть данные
                    mSum = mSum / k
            msg = msg + '\nСредняя зарплата: {0:<10.2f}' \
                  .format(mSum)
            lab_Subd3['text'] = msg
#
            if not self.ControlLoaded("О подразделении"):
                return
# Откроем форму верхнего уровня: форма для вывода данных
            win_Subd = Toplevel(root)
            win_Subd.title('Поиск подразделения')
            win_Subd.geometry("355x240")
            win_Subd.resizable(width = False, height = False)
# Создадим окно в форме с виджетами
            frm_Subd = Frame(win_Subd, bg = "Lightgrey", bd = 5)
            frm_Subd.place(width = 355, height = 240,
                           x = 0, y = 0)
# Поле для вывода имени подразделения
            ent_Subd1 = Entry(frm_Subd, bg = 'white', fg = 'blue',
                               font = 'Courier_New 12', justify = 'left')
            ent_Subd1.place(width = 160,height = 25,
                           x = 10, y = 35)
# Поле для ввода имени или части имени подразделения
            ent_Subd2 = Entry(frm_Subd, bg = 'white', fg = 'black',

```

```

        font = 'Courier_New 12', justify = 'left')
ent_Subd2.place(width = 160, height = 20,
                 x = 175, y = 20)
ent_Subd2.delete(0,END)
# кнопки поиска
but_Subd1 = Button(frm_Subd, text = 'Назад',
                     command = butSubd1_clk)
but_Subd1.place(width = 80, height = 20,
                 x = 175, y = 40)
but_Subd2 = Button(frm_Subd, text = 'Далее',
                     command = butSubd2_clk)
but_Subd2.place(width = 80, height = 20,
                 x = 255, y = 40)
# Метки
lab_Subd1 =Label(frm_Subd, text = 'Подразделение',
                  font = 'Courier_New 12', justify = 'center',
                  bg = "Lightgrey")
lab_Subd1.place(width = 160, height = 25,
                 x = 10, y = 5)
lab_Subd2 =Label(frm_Subd, text = 'Найти:')
lab_Subd2.place(width = 160, height = 20,
                 x = 175, y = 0)
lab_Subd3 =Label(frm_Subd, text = 'О подразделении')
lab_Subd3.place(width = 325, height = 170,
                 x = 10, y = 65)
#
# Меню "Справка": Команда - "День рождения"
def bday_h(self):
    if not self.ControlLoaded("День рождения"):
        return
    self.win_bd = Toplevel(root)
    self.win_bd.title('День рождения')
    self.win_bd.geometry("400x110")
# Запрет на изменение размеров окна
    self.win_bd.resizable(width = False, height = False)
# текущая дата и время
    now = datetime.now()
    msg = 'Сегодня: {0: 2d}.{1: 2d}.{2:4d}\n'.format(now.day,
                                                     now.month,
                                                     now.year)
    msg = msg + ('-----\n'
                 'Ф.И.О.          Лет          Д.Р.          Телефон')
    msgf = []
    for count in range(self.Workers.size):
        dd, mm, yy = self.Workers[count]['birthday'].split('.')
        bday = datetime(now.year, int(mm), int(dd))
        period = bday - now
        let = now.year - int(yy)
        if ((period.days >= -1) and (period.days <= 1)):
            msgf.append('{0:<17s} {1:>2d} {2:<10s} {3:<7s}'\
                         .format(self.Workers[count]['fio'],
                                 now.year - int(yy),
                                 self.Workers[count]['birthday'],
                                 self.Workers[count]['phon']))
    if not any(msgf):
        msgf = ['Список пуст']
# Информационная метка
    self.lab_bd = Label(self.win_bd, text = msg,

```

```

                font = '"Courier New" 10')
        self.lab_bd.place(width = 380, height = 75,
                           x = 10, y = 5)
# Виджет Combobox (с выпадающим списком) для вывода данных
# о днях рождения работников: ФИО - Дата рождения - Телефон
        self.com_bd = Combobox(self.win_bd, values = msgf,
                               state='readonly', height = 4)
        self.com_bd.configure(font = '"Courier New" 10')
        self.com_bd.current(0) # начальное значение
        self.com_bd.place(width = 380, height = 25,
                           x = 10, y = 70)
#
# Метод для контроля загрузки данных
def ControlLoaded(self, stitle):
    if not self.Fload:
        showinfo(stitle, 'Данные не загружены!')
        return(False)
    else:
        return(True)
#
#*****#
# Класс для реализации команд меню "Работник" и "О работнике".
# В классе реализованы основные поля для вывода данных о работнике
# и методы для выбора работников и заполнения полей.
# Worker1 - базовый класс, на основе которого создаются два класса:
#           Worker2 - класс для просмотра данных о работнике.
#           Worker3 - класс для редактирования данных о работнике.
#*****#
class Worker1:
#
    Wcount = 0 # этот элемент - свойство класса: счетчик для виджета
               # Spinbox
#
    def __init__(self, master, str_title):
        """
        Построение формы с метками и полями
        для вывода данных о работниках.
        Поля "Подразделение" и "Должность"
        отсутствуют.
        """
        self.wWorker = Toplevel(master)
        self.wWorker.title(str_title)
        self.wWorker.geometry("400x210")
        self.wWorker.resizable(width = False, height = False)
# управление событием закрытия окна, через кнопку управления
# окном - X - "Закрыть", передаем методу btl_exit
        self.wWorker.protocol('WM_DELETE_WINDOW',
                             self.btl_exit)
        self.fWorker = Frame(self.wWorker, bg = "Lightgrey", bd = 5)
        self.fWorker.place(width = 400, height = 210,
                           x = 0, y = 0)
# Метки - наименования полей вывода
        self.lb1 = Label(self.fWorker)
        self.lb1.config(bg = "Lightgrey", text = "Подразделение",
                       fg = "blue", font = 'Courier_New 10')
        self.lb1.place(width = 120, height = 25,
                       x = 10, y = 5)
        self.lb2 = Label(self.fWorker)

```

```

self.lb2.config(bg = "Lightgrey", text = "Должность",
               fg = "blue", font = 'Courier_New 10')
self.lb2.place(width = 100, height = 25,
               x = 150, y = 5)
self.lb3 = Label(self.fWorker)
self.lb3.config(bg = "Lightgrey", text = "Телефон",
               fg = "blue", font = 'Courier_New 10')
self.lb3.place(width = 100, height = 25,
               x = 260, y = 5)
self.lb4 = Label(self.fWorker)
self.lb4.config(bg = "Lightgrey", text = "Ф.И.О.",
               fg = "blue", font = 'Courier_New 10')
self.lb4.place(width = 100, height = 25,
               x = 10, y = 65)
self.lb5 = Label(self.fWorker)
self.lb5.config(bg = "Lightgrey", text = "Оклад",
               fg = "blue", font = 'Courier_New 10')
self.lb5.place(width = 100, height = 25,
               x = 150, y = 65)
self.lb6 = Label(self.fWorker)
self.lb6.config(bg = "Lightgrey", text = "Дата рождения",
               fg = "blue", font = 'Courier_New 10')
self.lb6.place(width = 100, height = 25,
               x = 260, y = 65)

# Поля вывода данных
self.entPhon = Entry(self.fWorker)
self.entPhon.config(bg = "white", fg = "blue",
                    justify = "left")
self.entPhon.place(width = 100, height = 25,
                   x = 260, y = 35)
self.entFIO = Entry(self.fWorker)
self.entFIO.config(bg = "white", fg = "blue",
                   justify = "left")
self.entFIO.place(width = 120, height = 25,
                  x = 10, y = 95)
self.entSal = Entry(self.fWorker)
self.entSal.config(bg = "white", fg = "blue",
                   justify = "left")
self.entSal.place(width = 100, height = 25,
                  x = 150, y = 95)
self.entBday = Entry(self.fWorker)
self.entBday.config(bg = "white", fg = "blue",
                    justify = "left")
self.entBday.place(width = 100, height = 25,
                   x = 260, y = 95)

# Виджеты для выбора записи: метка + Spinbox
self.lbl1 = Label(self.fWorker)
self.lbl1.config(bg = "Lightgrey", text = "Запись:",
                 fg = "blue")
self.lbl1.place(width = 100, height = 25,
                x = 225, y = 135)
self.sb = Spinbox(self.fWorker)
self.sb.config(justify = "right",
               state = "readonly", wrap = True,
               command = self.sb_click)
# self.sb.config(from_ = 0, to=self.masW.size - 1)
self.sb.place(width = 45, height = 25,
              x = 250, y = 160)

```

```

        self.bt1 = Button(self.fWorker)
        self.bt1.config(bg = "Lightgrey", text = "Выход",
                       fg = "blue", command = self.bt1_exit)
        self.bt1.place(width = 60, height = 25,
                       x = 320, y = 160)
#
def sb_click(self):
    """
    Обработка событий виджета Spinbox
    """
    if self.masW.size < 2:
        return
    self.Wcount = int(self.sb.get())
    self.v_Worker(self.masW)

def v_Worker(self, masW):
    """
    Вывод данных в форму
    """

# очистим поля
    self.entPhon.delete(0, END)
    self.entFIO.delete(0, END)
    self.entSal.delete(0, END)
    self.entBday.delete(0, END)
#
    self.entPhon.insert(0, masW[self.Wcount]['phon'])
    self.entFIO.insert(0, masW[self.Wcount]['fio'])
    self.entSal.insert(0, masW[self.Wcount]['salary'])
    self.entBday.insert(0, masW[self.Wcount]['birthday'])
    self.v_Subd_Post()
#
def v_Subd_Post(self):
    """
    Заглушка. Этот метод модифицируется в следующих классах
    """
    pass
#
def bt1_exit(self):
    self.new_masW = self.masW      # используется в классе Worker3
    self.wWorker.destroy()         # Закрытие текущего окна
#
#*****
# Класс для меню "Справка": Команда - "О работнике"
# Worker2 - класс для просмотра данных о работнике. В этом классе
# добавлены поля для вывода наименования подразделения и
# должности, изменен (дополнен) метод вывода данных в
# добавленные поля.
# Добавлен метод Change() для получения данных о работниках.
#*****
class Worker2(Worker1):
    def __init__(self, master, str_title):
        """
        Используем свойства и методы класса Worker1 и
        добавляем свои свойства (поля) и методы
        """
        Worker1.__init__(self, master, str_title)
#
        self.entSDiv = Entry(self.fWorker)

```

```

        self.entSDiv.config(bg = "white", fg = "blue",
                           justify = "left")
        self.entSDiv.place(width = 120, height = 25,
                           x = 10, y = 35)
    self.entPost = Entry(self.fWorker)
    self.entPost.config(bg = "white", fg = "blue",
                        justify = "left")
    self.entPost.place(width = 100, height = 25,
                       x = 150, y = 35)
#
def Change(self, masWk, masSd, masPs):
    self.masW = masWk           # данные о работниках
    self.mass = masSd           # справочник "Подразделения"
    self.masP = masPs           # справочник "Должности"
    self.Wcount = 0
    self.sb.config(from_= 0, to=self.masW.size - 1)
    self.v_Worker(self.masW)
#
def v_Subd_Post(self):
    """
    Этот метод - модифицированный метод класса Worker1
    Он применен для заполнения данными виджетов Entry
    """
    nSubd = ''
    nPost = ''
    for i in range(self.mass.size):  # код подразделения
        if self.masW[self.Wcount]['cod_sd'] ==\
            self.mass[i]['cod']:
            nSubd = self.mass[i]['name']
            break
    self.entSDiv.delete(0, END)
    self.entSDiv.insert(0, nSubd)    # подразделение
#
    for i in range(self.masP.size):  # ищем должность по коду
        if self.masW[self.Wcount]['cod_p'] ==\
            self.masP[i]['cod']:
            nPost = self.masP[i]['name']
            break
    self.entPost.delete(0, END)
    self.entPost.insert(0, nPost)    # должность
#*****
# Класс для меню "Редактировать". Команда - "Работник".
# Worker3 - класс для редактирования данных о работнике.
# В этом классе добавлены поля Combobox: выбор подразделения
# и должности работника. Изменен (дополнен) метод вывода
# данных в добавленные поля.
# Добавлен метод Change() для получения данных о работниках,
# возвращаем измененные данные в основную часть программы.
# Дополнительно добавлены кнопки редактирования и методы
# обработки событий, вызываемых при нажатии на эти кнопки.
#*****
class Worker3(Worker1):
    def __init__(self, master, str_title):
        """
        Используем свойства и методы класса Worker1 и
        добавляем свои свойства (поля) и методы
        """
        Worker1.__init__(self, master, str_title)

```

```

# Дополнительные поля: Подразделение, Должность
self.msgS = StringVar()
self.cbSDiv = Combobox(self.fWorker)
self.cbSDiv.config(state='readonly', height = 4,
                     textvariable = self.msgS)
self.cbSDiv.place(width = 120, height = 25,
                   x = 10, y = 35)
self.msgP = StringVar()
self.cbPost = Combobox(self.fWorker)
self.cbPost.config(state='readonly', height = 4,
                     textvariable = self.msgP)
self.cbPost.place(width = 100, height = 25,
                   x = 150, y = 35)
self.butNRec = Button(self.fWorker)
self.butNRec.config(bg = "Lightgrey", fg = "blue",
                     text = 'Новая',
                     command = self.butNew_rec)
self.butNRec.place(width = 60, height = 25,
                   x = 10, y = 160)
self.butSave = Button(self.fWorker)
self.butSave.config(bg = "Lightgrey", fg = "blue",
                     text = 'Сохранить',
                     command = self.butSave_rec)
self.butSave.place(width = 80, height = 25,
                   x = 75, y = 160)
self.butDel = Button(self.fWorker)
self.butDel.config(bg = "Lightgrey", fg = "blue",
                     text = 'Удалить',
                     command = self.butDel_rec)
self.butDel.place(width = 70, height = 25,
                   x = 160, y = 160)
self.FNewRec = False # кнопка "Новая" не нажата
# True - кнопка "Новая" нажата

#
# Этот метод обеспечивает обмен данными между классами main и Worker3
# Новые данные возвращаются при закрытии окна редактирования
def Change(self, masWk, masSd, masPs):
    self.masW = masWk # данные о работниках
    self.mass = masSd # справочник "Подразделения"
    self.masP = masPs # справочник "Должности"
    self.sb.config(from_= 0, to=self.masW.size - 1)
    self.Wcount = 0
# перечень подразделений
    self.val = []
    for i in range(self.mass.size): # список значений
        if self.mass[i]['cod'] < 100: # только актуальные
            # данные
            self.val.append(self.mass[i]['name'])
    self.cbSDiv.config(values = self.val)
# перечень должностей
    self.val = []
    for i in range(self.masP.size): # список значений
        if self.masP[i]['cod'] < 100: # только актуальные
            # данные
            self.val.append(self.masP[i]['name'])
    self.cbPost.config(values = self.val)
#
    self.v_Worker(self.masW) # начальное заполнение полей

```

```

#
    self.new_masW = None
    self.wWorker.wait_window() # Ожидание события закрытия
                                # текущего окна
    return self.new_masW
#
def v_Subd_Post(self):
    """
    Этот метод - модифицированный метод класса Worker1
    Он применен для заполнения данными виджетов Combobox
    """
    if self.FNewRec:           # сброс состояния "Новая запись"
        self.FNewRec = False   # смотрим следующую запись
# наименование подразделения
    self.msgS.set('')
    for i in range(self.masS.size):
        if self.masW[self.Wcount]['cod_sd'] == \
            self.masS[i]['cod']:
            self.cbSDiv.current(i)
            break
# наименование должности
    self.msgP.set('')
    for i in range(self.masP.size):
        if self.masW[self.Wcount]['cod_p'] == \
            self.masP[i]['cod']:
            self.cbPost.current(i)
            break

def butNew_rec(self):
    """
    Подготовка полей для новой записи
    """
    self.msgS.set('')
    self.msgP.set('')
    self.entPhon.delete(0, END)
    self.entFIO.delete(0, END)
    self.entSal.delete(0, END)
    self.entBday.delete(0, END)
    self.FNewRec = True     # нажата кнопка "Новая"

def butSave_rec(self):
    """
    Сохранение измененных или новых данных
    """
    stf = self.entFIO.get()
    cod = self.cbSDiv.current()
    cop = self.cbPost.current()
    nsa = self.entSal.get()
    if (stf == '') or (cod < 0) or (cop < 0) or (nsa == ''):
        showerror("Ошибка", "Следующие поля:\n"
                  "- ФИО\n"
                  "- Подразделение\n"
                  "- Должность\n"
                  "- Оклад\n"
                  "обязательны для заполнения.")
        return
    if self.FNewRec:          # сохраняем новую запись
        self.Wcount = self.masW.size

```

```

        self.masW = np.resize(self.masW, self.Wcount + 1)
#
        self.masW[self.Wcount]['fio'] = self.entFIO.get()
        self.masW[self.Wcount]['cod_sd'] = self.cbSDiv.current()
        self.masW[self.Wcount]['phon'] = self.entPhon.get()
        self.masW[self.Wcount]['birthday'] = self.entBday.get()
        self.masW[self.Wcount]['cod_p'] = self.cbPost.current()
        self.masW[self.Wcount]['salary'] = self.entSal.get()
#
# изменим параметры Spinbox
        self.sb.config(from_= 0, to=self.masW.size - 1)
        self.FNewRec = False # кнопка "Новая" не нажата

    def butDel_rec(self):
        """
        Удаление записей. Массив записей сжимается.
        """
        if self.masW.size < 2:
            showerror('Error', 'Нельзя удалить!\n'
                           'Одна запись должна оставаться!\n'
                           'См. "Руководство пользователя"\n')
            return
        if self.FNewRec: # сброс состояния "Новая запись"
            self.FNewRec = False # т.к. нажата кнопка "Удалить"
        else:
            i = self.Wcount
            while i < self.masW.size - 1:
                self.masW[i]['fio'] = self.masW[i + 1]['fio']
                self.masW[i]['cod_sd'] = \
                self.masW[i + 1]['cod_sd']
                self.masW[i]['phon'] = self.masW[i + 1]['phon']
                self.masW[i]['birthday'] = \
                self.masW[i + 1]['birthday']
                self.masW[i]['cod_p'] = self.masW[i + 1]['cod_p']
                self.masW[i]['salary'] = \
                self.masW[i + 1]['salary']
                i += 1
            self.masW = np.resize(self.masW, self.masW.size-1)
#
# изменим параметры Spinbox
        self.sb.config(from_= 0, to=self.masW.size - 1)
        self.v_Worker(self.masW)
#
# ===== Меню "Редактировать" =====
#*****
# Класс меню "Редактировать". Команды - "Подразделение", "Должность"
# Edit_dic - класс для редактирования данных в справочниках.
# В этом классе реализованы поля и методы, которые
# позволяют изменять данные в справочниках "Подразделение"
# и "Должность".
# Поскольку структура справочников одинаковая, то и поля
# для данных и методы также одинаковые.
# Для передачи данных между этим классом и основным классом
# используется метод Change().
#*****
**
```

class Edit_dic:

```

    def __init__(self, master, str_title):
        """
        Подготовка формы для редактирования справочников

```

```

"""
    self.winD = Toplevel(master)
    self.winD.title(str_title)
    self.winD.geometry("280x140")
# Запрет на изменение размеров
    self.winD.resizable(width = False, height = False)
# закрытие окна через кнопку управления окном - X - "Закрыть"
    self.winD.protocol('WM_DELETE_WINDOW',
                       self.butD_exit)
#
    self.frmD = Frame(self.winD, bg = "Lightgrey", bd = 5)
    self.frmD.place(width = 280, height = 140,
                    x = 0, y = 0)
#
    self.cbD = Combobox(self.frmD, height = 5)
    self.cbD.place(width = 160, height = 25,
                   x = 10, y = 5)
    self.butD1 = Button(self.frmD, text = 'Добавить',
                        command = self.butD_Add)
    self.butD1.place(width = 80, height = 25,
                      x = 180, y = 5)
    self.butD2 = Button(self.frmD, text = 'Удалить',
                        command = self.butD_Del)
    self.butD2.place(width = 80, height = 25,
                      x = 180, y = 35)
    self.butD3 = Button(self.frmD, text = 'Выход',
                        command = self.butD_exit)
    self.butD3.place(width = 80, height = 25,
                      x = 180, y = 65)
#
# Методы класса
def butD_Add(self):
    """
    Добавление записи. Новая запись вставляется
    либо вместо ранее удаленной, либо в конец массива
    """
    mystr = self.cbD.get().strip() # получим значение
    if not mystr: # пусто
        return
#
    flg = False # в списке значения нет
    for s in self.val: # проверим список
        if str(s).lower() == mystr.lower():
            flg = True # значение есть
            break
    if flg: # значение есть
        return
    self.val.append(mystr) # добавим в список
    self.cbD.config(values = self.val)
    flg = False # Значение не в массиве
    for i in range(self.masDn): # ищем место для записи
        if self.masD[i]['cod'] > 100: # запись удалена ранее
            self.masD[i]['name'] = mystr
            self.masD[i]['cod'] = self.masD[i]['cod'] - 100
            flg = True # Новая запись добавлена в массив
            break
    if not flg: # запись все еще не добавлена
        # Новую запись добавим в хвост массива
        # изменим число записей в справочнике

```

```

        self.masDn += 1
        # изменим размер массива
        self.masD = np.resize(self.masD, self.masDn)
        # добавим новые данные
        self.masD[self.masDn - 1]['cod'] = self.masDn
        self.masD[self.masDn - 1]['name'] = mystr

def butD_Del(self):
    """
        Удаление записи. Запись помечается как удаленная.
        Удаляются только не используемые записи.
    """
    if self.masD.size < 2:
        showerror('Error', 'Нельзя удалить!\n'
                  'Одна запись должна оставаться!\n'
                  'См. "Руководство пользователя"\n')
        return
    mystr = self.cbD.get().strip()
    if not mystr:      # пусто
        return
    # получим код удаляемого значения
    for i in range(self.masDn):      # ищем запись в массиве
        mystr0 = str(self.masD[i]['name']).lower()
        if mystr0 == mystr.lower():  # нашли
            delcod = self.masD[i]['cod']
            break
    # Используется ли этот код в записях о работниках?
    for i in range(self.masW.size):
        flg_1 = False
        if self.flag:      # "Подразделение"
            if self.masW[i]['cod_sd'] == delcod:
                msg = 'Код подразделения используется.\n'
                flg_1 = True
        else:             # "Должность"
            if self.masW[i]['cod_p'] == delcod:
                msg = 'Код должности используется.\n'
                flg_1 = True
        if flg_1:
            showerror('Error', 'Нельзя удалить!\n',
                      msg, 'См. "Руководство пользователя"\n')
            return
    self.val.remove(mystr)
    self.cbD.config(values = self.val)
    for i in range(self.masDn):      # ищем в массиве
        if self.masD[i]['cod'] == delcod: # нашли
            self.masD[i]['cod'] += 100     # удалим
            self.masD[i]['name'] = ''
            break
    self.cbD.set(self.masD[0]['name'])
#
# метод для обмена данными между классами main и Edit_dic
# Новые данные возвращаются при закрытии окна редактирования
def Change(self, flg, mas_rec, mas_D):
    """
        Получение данных из класса main.
        Возвращение измененных данных в класс main
    """
    self.flag = flg

```

```

        self.masW = mas_rec
        self.masD = mas_D
        self.masDn = self.masD.size
        self.val = []
        for i in range(self.masDn):           # список значений
            if self.masD[i]['cod'] < 100:      # актуальные данные
                self.val.append(self.masD[i]['name'])
        self.cbD.config( values = self.val)
        self.cbD.set(self.masD[0]['name'])
        self.new_mas_rec = None
        self.winD.wait_window()             # Ожидание события
                                            # закрытия окна
                                            # возвращаем данные
        return self.new_mas_rec
    #
    # Этот метод сохраняет данные в новой переменной и закрывает окно
    def butD_exit(self):
        self.new_mas_rec = self.masD
        self.winD.destroy()                 # Закрытие окна
    #
    # Запуск программы с созданием главного окна
    if __name__ == '__main__':
        root = Tk()
        main(root)

```

Пояснения к программе

Листинг программы содержит достаточно большое число строк, к которым даны подробные комментарии.

Класс `main()` реализует главное окно (метод `__init__()`) с набором каскадных меню, создаваемых методом `myMenu()` и набором методов, реализующих команды этого меню:

Меню "Файл"

Для реализации команд чтения, сохранения и завершения работы программы используются методы:

- `open_f()` – команда "Открыть". Загружаются данные о работниках. Файлы, содержащие информацию о работниках, должны быть с расширение *.csv. В процессе загрузки данных о работниках подгружаются и данные из двух справочников: "Справочник подразделений" и "Справочник должностей". Внутренний формат этих файлов – CSV. Расширение файлов задано как: *.subd – для "Справочник подразделений", и *.post – для "Справочник должностей";

- `save_f()` и `saveas_f()` – команды "Сохранить" и "Сохранить как ...". По команде "Сохранить" выполняется перезапись последнего загруженного файла, а по команде "Сохранить как ..." можно задать новое имя файла для сохраняемых данных. При этом новое имя файла становится текущим. По обоим командам выполняется и сохранение справочников.

- `close_win()` – команда "Выход". В этом методе выполняется контроль текущего состояния задачи и перед закрытием программы предлагается выполнить сохранение текущих данных. Этот же метод вызывается при попытке закрыть главное окно командой "Закрыть" - [X].

Меню "Редактировать"

В этом меню реализованы команды для редактирования справочников и данных о работниках. Для редактирования справочников вызываются методы:

- `edit_sd()` – команда "Подразделение". По этой команде открывается окно для редактирования наименования подразделений организации;

- **edit_p()** – команда "Должность". По этой команде открывается окно для редактирования перечня должностей.

Справочники имеют одинаковый формат записей:

<Код> <Наименование>

и для их редактирования построен один класс `Edit_dic()`, при вызове которого передается информация о соответствующем справочнике. Описание класса приводится ниже.

В процессе редактирования справочников выполняется контроль введенного значения. Если такая информация уже присутствует, то добавления в справочник не выполняется. При удалении записи из справочника выполняется проверка на ее использование в данных о работнике. Если запись используется, то удаление не выполняется. Для удаления записей, используемых в данных о работниках, необходимо выполнить корректировку данных по работникам, переводя их в некоторое условное подразделение или на некоторую условную должность, а затем удалить запись из справочника. При необходимости повторить корректировку данных о работниках.

- **edit_worker()** – команда "Работник". Окно, открываемое при выборе этой команды, имеет набор полей, в которые вводятся данные о новых работниках или редактируется информация об уже зарегистрированных работниках. Эта форма имеет набор методов и полей, совпадающих с методами и полями формы, в которой просматриваются данные о работниках. Отличие состоит в том, что поля для задания имени подразделения и выбора должности в первом случае заполняются путем выбора данных из справочников, а во втором случае – только показываются. В связи с этим разработан промежуточный класс `Worker1()` свойства и методы которого наследуются в классах `Worker2()` и `Worker3()`. Описание этих классов приводится далее.

Меню "Справка"

Меню "Справка" содержит набор команд, в которых реализованы следующие функции:

about_h() – команда "Помощь". По этой команде открывается файл формата PDF, в котором может содержаться описание программы в стиле руководства с гиперссылками.

inf_h() – команда "О программе". По этой команде открывается окно, содержащее сведения о назначении программы и ее версии.

worker_h() – команда "О работнике". По этой команде открывается окно с полями, которые предназначены только для просмотра информации о работниках предприятия. Для реализации этой команды разработан класс `Worker2()`, наследующий поля и методы класса `Worker1()`.

subd_h() – команда "О подразделении". Данная команда предназначена для получения списка работников заданного подразделения и вывода информации о средней заработной плате по этому подразделению.

bday_h() – команда "День рождения". Эта команда используется для получения информации о сотрудниках, чей день рождения приходится на день запроса и на следующий день.

Класс `Edit_dic`

В этом классе реализовано дочернее окно с полем для вывода или ввода информации, а также кнопки для добавления, удаления и завершения работы этого окна. Обмен данными между родительским и дочерним окнами (передача информации о справочнике) реализован в методе `Change()`, который построен по принципу, рассмотренному в теоретической части этой лабораторной работы.

Класс Worker1

Этот класс создан как промежуточный класс, используемый при создании классов Worker2() – просмотр информации о работниках, и Worker3() – редактирование информации о работниках. В классе отсутствуют поля "Подразделение" и "Должность". Это связано с тем обстоятельством, что данные поля играют разную роль при редактировании данных или их просмотре. В классе реализованы метод работы с виджетом Spinbox, метод вывода информации в поля формы: v_Worker() и метод bt1_exit(), который закрывает форму.

Для заполнения полей "Подразделение" и "Должность", в зависимости от их назначения, реализован виртуальный метод v_Subd_Post(). этот метод вызывается при заполнении полей формы, но в данном классе он ничего не делает. Необходимый, соответствующий функциям полей, метод добавляется в классе наследнике.

Класс Worker2

Этот класс является наследником класса Worker1(). Кроме наследуемых свойств и методов в классе добавлены новые поля Entry и методы. Метод v_Subd_Post() реализует заполнение добавленных полей "Подразделение" и "Должность", а метод Change() – обмен данными между главным и дочерним окнами по ранее описанной технологии.

Класс Worker3

Этот класс является наследником класса Worker1(). Кроме наследуемых свойств и методов в классе добавлены виджеты типа Combobox. Этот тип виджетов представляет собой выпадающий список, в котором можно задать количество строк в открывающемся окне. В случае, когда список виджета больше количества видимых записей, можно выполнять скроллинг и таким образом получать доступ к любой записи списка.

По аналогии с классом Worker2() добавлены методы: v_Subd_Post() и Change().

Поскольку класс предназначен для редактирования данных о работнике, добавлены кнопки и соответствующие им методы, реализующие удаление и сохранение записи, а также добавление новой записи: выполняется очистка полей и возводится флаг FNewRec.

Значение флага проверяется при сохранении данных и если он установлен в True, то новая запись добавляется в массив записей о работниках.

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 выдает 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.

Вариант 11

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 маршрутах, которые начинаются или кончаются в пункте, название которого введено с клавиатуры; если таких маршрутов нет, вывести на экран соответствующее сообщение;
- запись массива в файл под заданным с клавиатуры именем.

Вариант 12

1. Описать запись с именем Note, содержащую следующие поля:

- фамилия, имя;
- номер телефона;
- день рождения (массив из трех чисел).

2. Написать программу, выполняющую следующие действия:

- ввод данных с клавиатуры в массив, состоящий из восьми элементов типа Note; записи должны быть упорядочены по датам дней рождения;
- вывод на экран информации о человеке, номер телефона которого введен с клавиатуры; если такого нет, вывести на экран соответствующее сообщение;
- запись массива в файл под заданным с клавиатуры именем.

Вариант 13

1. Описать запись с именем Note, содержащую следующие поля:

фамилия, имя;

- номер телефона;
- день рождения (массив из трех чисел).

2. Написать программу, выполняющую следующие действия:

- ввод данных с клавиатуры в массив, состоящий из восьми элементов типа Note; записи должны быть размещены по алфавиту;
- вывод на экран информации о людях, чьи дни рождения приходятся на месяц, значение которого введено с клавиатуры; если таких нет, вывести на экран соответствующее сообщение;
- запись массива в файл под заданным с клавиатуры именем.

Вариант 14

1. Описать запись с именем Note, содержащую следующие поля:

- фамилия, имя;
- номер телефона;
- день рождения (массив из трех чисел).

2. Написать программу, выполняющую следующие действия:

- ввод данных с клавиатуры в массив, состоящий из восьми элементов типа Note; записи должны быть упорядочены по трем первым цифрам номера телефона;
- 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.

Вариант 15

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.

Вариант 16

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 людях, родившихся под знаком, наименование которого введено с клавиатуры; если таких нет, вывести на экран соответствующее сообщение;
 - запись массива в файл под заданным с клавиатуры именем.

Вариант 17

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 знакам Зодиака;
 - вывод на экран информации о людях, родившихся в месяц, значение которого введено с клавиатуры; если таких нет, вывести на экран соответствующее сообщение;
 - запись массива в файл под заданным с клавиатуры именем.

Вариант 18

1. Описать запись с именем Price, содержащую следующие поля:
 - название товара;
 - название магазина, в котором продается товар;
 - стоимость товара в рублях.
2. Написать программу, выполняющую следующие действия:
 - ввод данных с клавиатуры в массив, состоящий из восьми элементов типа Price; записи должны быть размещены в алфавитном порядке по названиям товаров;
 - вывод на экран информации о товаре, название которого введено с клавиатуры; если таких товаров нет, вывести на экран соответствующее сообщение;
 - запись массива в файл под заданным с клавиатуры именем.

Вариант 19

1. Описать запись с именем Price, содержащую следующие поля:
 - название товара;

- название магазина, в котором продается товар;
- стоимость товара в рублях.

2. Написать программу, выполняющую следующие действия:

- ввод данных с клавиатуры в массив, состоящий из восьми элементов типа Price; записи должны быть размещены в алфавитном порядке по названиям магазинов;
- вывод на экран информации о товарах, продающихся в магазине, название которого введено с клавиатуры; если такого магазина нет, вывести на экран соответствующее сообщение;
- запись массива в файл под заданным с клавиатуры именем.

Вариант 20

1. Описать запись с именем Bill, содержащую следующие поля:

- расчетный счет плательщика;
- расчетный счет получателя;
- перечисляемая сумма в рублях.

2. Написать программу, выполняющую следующие действия:

- ввод данных с клавиатуры в массив, состоящий из восьми элементов типа Bill; записи должны быть размещены в алфавитном порядке по расчетным счетам плательщиков;
- вывод на экран информации о сумме, снятой с расчетного счета плательщика, введенного с клавиатуры; если такого расчетного счета нет, вывести на экран соответствующее сообщение;
- запись массива в файл под заданным с клавиатуры именем.

Задания к лабораторной работе №7 "GUI, классы, модуль Tkinter"

Эти задания можно использовать вместо и/или в дополнение к выше приведенным.

Вариант 1

Составить программу, которая содержит динамическую информацию о наличии автобусов в автобусном парке.

Сведения о каждом автобусе содержат:

- номер автобуса;
- фамилию и инициалы водителя;
- номер маршрута.

Программа должна обеспечивать следующие функциональные возможности:

- начальное формирование данных обо всех автобусах в парке в виде списка;
- при выезде каждого автобуса из парка вводится номер автобуса, и программа удаляет данные об этом автобусе из списка автобусов, находящихся в парке, и записывает эти данные в список автобусов, находящихся на маршруте;
- при въезде каждого автобуса в парк вводится номер автобуса, и программа удаляет данные об этом автобусе из списка автобусов, находящихся на маршруте, и записывает эти данные в список автобусов, находящихся в парке;
- по запросу выдаются сведения об автобусах, находящихся в парке, или об автобусах, находящихся на маршруте.

Вариант 2

Составить программу, которая содержит текущую информацию о книгах в библиотеке.

Сведения о книгах содержат:

- номер УДК;
- фамилию и инициалы автора;
- название;
- год издания;
- количество экземпляров данной книги в библиотеке.

Программа должна обеспечивать следующие функциональные возможности:

- начальное формирование данных обо всех книгах в библиотеке в виде двоичного дерева;
- добавление данных о книгах, вновь поступающих в библиотеку;
- удаление данных о списываемых книгах;
- по запросу выдаются сведения о наличии книг в библиотеке, упорядоченные по годам издания.

Вариант 3

Составить программу, которая содержит текущую информацию о заявках на авиабилеты.

Каждая заявка содержит:

- пункт назначения;
- номер рейса;
- фамилию и инициалы пассажира;
- желаемую дату вылета.

Программа должна обеспечивать:

- хранение всех заявок в виде списка;
- добавление заявок в список;
- удаление заявок;
- вывод заявок по заданному номеру рейса и дате вылета;
- вывод всех заявок.
-

Вариант 4

Составить программу, которая содержит текущую информацию о заявках на авиабилеты. Каждая заявка содержит:

- пункт назначения;
- номер рейса;
- фамилию и инициалы пассажира;
- желаемую дату вылета;

Программа должна обеспечивать:

- хранение всех заявок в виде двоичного дерева;
- добавление и удаление заявок;
- по заданному номеру рейса и дате вылета вывод заявок с их последующим удалением;
- вывод всех заявок.

Вариант 5

Составить программу, которая содержит текущую информацию о книгах в библиотеке.

Сведения о книгах содержат:

- номер УДК;
- фамилию и инициалы автора;
- название;
- год издания;
- количество экземпляров данной книги в библиотеке.

Программа должна обеспечивать следующие функциональные возможности:

- начальное формирование данных обо всех книгах в библиотеке в виде списка;
- при взятии каждой книги вводится номер УДК, и программа уменьшает значение количества книг на единицу или выдает сообщение о том, что требуемой книги в библиотеке нет или она находится на руках;
- при возвращении каждой книги вводится номер УДК, и программа увеличивает значение количества книг на единицу;
- по запросу выдаются сведения о наличии книг в библиотеке.

Вариант 6

Составить программу, которая содержит динамическую информацию о наличии автобусов в автобусном парке. Сведения о каждом автобусе содержат:

- номер автобуса;
- фамилию и инициалы водителя;
- номер маршрута;
- признак того, где находится автобус - на маршруте или в парке.

Программа должна обеспечивать следующие функциональные возможности:

- начальное формирование данных о всех автобусах в виде списка;
- при выезде каждого автобуса из парка вводится номер автобуса, и программа устанавливает значение признака «автобус на маршруте»;
- при въезде каждого автобуса в парк вводится номер автобуса, и программа устанавливает значение признака «автобус в парке»;
- по запросу выдаются сведения об автобусах, находящихся в парке, или об автобусах, находящихся на маршруте.

Вариант 7

Составить программу, отыскивающую проход по лабиринту.

Лабиринт представляется в виде матрицы, состоящей из квадратов. Каждый квадрат либо открыт, либо закрыт. Вход в закрытый квадрат запрещен. Если квадрат открыт, то вход в

него возможен со стороны, но не с угла. Каждый квадрат определяется его координатами в матрице.

Программа находит проход через лабиринт, двигаясь от заданного входа. После отыскания прохода программа выводит найденный путь в виде координат квадратов. Для хранения пути использовать стек.

Вариант 8

Гаражная стоянка имеет одну стояночную полосу, причем единственный въезд и единственный выезд находятся в одном конце, полосы. Если владелец автомашины приходит забрать свой автомобиль, который не является ближайшим к выходу, то все автомашины, загораживающие проезд, удаляются, и машина данного владельца выводится со стоянки, после чего другие машины возвращаются на стоянку в исходном порядке.

Написать программу, которая моделирует процесс прибытия и отъезда машин. Прибытие или отъезд автомашины задается командной строкой, которая содержит признак прибытия или отъезда и номер машины. Программа должна выводить сообщение при прибытии или выезде любой машины. При выезде автомашины со стоянки сообщение должно содержать число раз, которое машина удалялась со стоянки для обеспечения выезда других автомобилей.

Вариант 9

Составить программу, моделирующую заполнение гибкого магнитного диска. Общий объем памяти на диске 360 Кбайт. Файлы имеют произвольную длину от 18 байт до 32 Кбайт. В процессе работы файлы либо записываются на диск, либо удаляются с него.

В начале работы файлы записываются подряд друг за другом. После удаления файла на диске образуется свободный участок памяти, и вновь записываемый файл либо размещается на свободном участке, либо, если файл не помещается в свободный участок, размещается после последнего записанного файла.

В случае, когда файл превосходит длину самого большого свободного участка, выдается аварийное сообщение. Требование на запись или удаление файла задается в командной строке, которая содержит имя файла, его длину в байтах, признак записи или удаления.

Программа должна выдавать по запросу сведения о занятых и свободных участках памяти на диске.

СОВЕТ: Следует создать список занятых участков и список свободных участков памяти на диске.

Вариант 10

В файловой системе каталог файлов организован как линейный список. Для каждого файла в каталоге содержатся следующие сведения:

- имя файла;
- дата создания;
- количество обращений к файлу.

Составить программу, которая обеспечивает:

- начальное формирование каталога файлов;
- вывод каталога файлов;
- удаление файлов, дата создания которых меньше заданной;
- выборку файла с наибольшим количеством обращений.

Вариант 11

Предметный указатель организован как линейный список. Каждая компонента указателя содержит слово и номера страниц, на которых это слово встречается. Количество номеров страниц, относящихся к одному слову, может лежать в интервале от одного до десяти.

Составить программу, которая обеспечивает:

- начальное формирование предметного указателя;
- вывод предметного указателя;
- вывод номеров страниц для заданного слова.

Вариант 12

Текст помощи для некоторой программы организован как линейный список. Каждая компонента текста помощи содержит термин (слово) и текст пояснения к этому термину. Допустимое количество строк текста, относящихся к одному термину, - от одной до пяти. Составить программу, которая обеспечивает:

- начальное формирование текста помощи;
- вывод текста помощи;
- вывод поясняющего текста для заданного термина.

Вариант 13

Картотека в бюро обмена квартир организована как линейный список. Сведения о каждой квартире содержат:

- количество комнат;
- этаж;
- площадь;
- адрес.

Составить программу, которая обеспечивает:

- начальное формирование картотеки;
- ввод заявки на обмен;
- поиск в картотеке подходящего варианта: при равенстве количества комнат и этажа и различии площадей в пределах 10 % выводится соответствующая карточка, которая затем удаляется из списка; в противном случае поступившая заявка включается в список;
- вывод всего списка.

Вариант 14

Англо-русский словарь построен как двоичное дерево. Каждая компонента содержит английское слово, соответствующее ему русское слово и счетчик количества обращений к данной компоненте.

Первоначально дерево формируется согласно английскому алфавиту. В процессе эксплуатации словаря при каждом обращении к компоненте в счетчик обращений добавляется единица.

Составить программу, которая:

- обеспечивает начальный ввод словаря с конкретными значениями счетчиков обращений;
- формирует новое представление словаря в виде двоичного дерева по следующему алгоритму:
 - 1) в старом словаре ищется компонента с наибольшим значением счетчика обращений;
 - 2) найденная компонента заносится в новый словарь и удаляется из старого;
 - 3) переход к п. 1 до исчерпания исходного словаря;
- производит вывод исходного и нового словарей.

Вариант 15

Анкета для опроса населения содержит две группы вопросов. Первая группа содержит сведения о респонденте:

- возраст;

- пол;
- образование (начальное, среднее, высшее).

Вторая группа содержит собственно вопрос анкеты, ответом на который может быть либо «Да», либо «Нет».

Составить программу, которая:

- обеспечивает начальный ввод анкет и формирует из них линейный список;
- на основе анализа анкет выдает ответы на следующие вопросы:
 - 1) сколько мужчин старше 40 лет, имеющих высшее образование, ответили «Да» на вопрос анкеты;
 - 2) сколько женщин моложе 30 лет, имеющих среднее образование, ответили «Нет» на вопрос анкеты;
 - 3) сколько мужчин моложе 25 лет, имеющих начальное образование, ответили «Да» на вопрос анкеты;
- производит вывод всех анкет и ответов на вопросы.

Вариант 16

Составить программу, которая содержит текущую информацию о книгах в библиотеке.

Сведения о книгах содержат:

- номер УДК;
- фамилию и инициалы автора;
- название;
- год издания;
- количество экземпляров данной книги в библиотеке.

Программа должна обеспечивать следующие функциональные возможности:

- начальное формирование данных обо всех книгах в библиотеке в виде списка;
- добавление данных о книгах, вновь поступающих в библиотеку;
- удаление данных о списываемых книгах;
- по запросу выдаются сведения о наличии книг в библиотеке, упорядоченные по годам издания.

Вариант 17

На междугородной телефонной станции картотека абонентов, содержащая сведения о телефонах и их владельцах, организована как линейный список.

Составить программу, которая:

- обеспечивает начальное формирование картотеки в виде линейного списка;
- производит вывод всей картотеки;
- получает номер телефона и время разговора;
- выводит извещение на оплату телефонного разговора.

Вариант 18

На междугородной телефонной станции картотека абонентов, содержащая сведения о телефонах и их владельцах, организована как двоичное дерево.

Составить программу, которая:

- обеспечивает начальное формирование картотеки в виде двоичного дерева;
- производит вывод всей картотеки;
- получает номер телефона и время разговора;
- выводит извещение на оплату телефонного разговора.

Вариант 19

Автоматизированная информационная система на железнодорожном вокзале содержит сведения об отправлении поездов дальнего следования.

Для каждого поезда указывается:

- номер поезда;
- станция назначения;
- время отправления.

Данные в информационной системе организованы в виде линейного списка. Составить программу, которая:

- обеспечивает первоначальный ввод данных в информационную систему и формирование линейного списка;
- производит вывод всего списка;
- получает номер поезда и выводит все данные об этом поезде;
- получает название станции назначения и выводит данные обо всех поездах, следующих до этой станции.

Вариант 20

Автоматизированная информационная система на железнодорожном вокзале содержит сведения об отправлении поездов дальнего следования.

Для каждого поезда указывается:

- номер поезда;
- станция назначения;
- время отправления.

Данные в информационной системе организованы в виде двоичного дерева. Составить программу, которая:

- обеспечивает первоначальный ввод данных в информационную систему и формирование двоичного дерева;
- производит вывод всего дерева;
- получает номер поезда и выводит все данные об этом поезде;
- получает название станции назначения и выводит данные обо всех поездах, следующих до этой станции.

Лабораторная работа №8: "Программирование в графическом режиме"

Цель лабораторной работы

Познакомить студентом на практике с написанием программ для формирования графических изображений с использованием модулей Python.

Постановка задачи

Используя программный код из лабораторной работы №2, Задание 1 и лабораторной работы №2, Задание 2, а так же один из графических модулей Python, написать программы, которые иллюстрируют работу кода в графическом виде:

- Задание 1 – Решить обратную задачу – построить график функции, используя программный код, написанный к этому заданию;
- Задание 2 – построить графическое изображение, которое получается в результате генерации точек со случайными координатами. Использовать до 10000 точек. Оценить площадь заштрихованной фигуры методом Монте-Карло и сравнить с реальной. Оценку вывести в процентах по отношению к реальной площади.

Теоретическое введение

В Python разработано несколько модулей, обеспечивающих работу с графикой. Два графических модуля являются частью стандартной библиотеки Python:

- `turtle` (черепашка) – простой графический пакет, который так же может быть использован для создания несложного пользовательского графического интерфейса – Graphical User Interface (GUI);
- `tkinter` – разработан непосредственно для создания графического интерфейса пользователя GUI. Так, интерфейс IDLE построен с использованием `tkinter`.

Существуют и другие модули, из которых мы рассмотрим модуль `matplotlib`, который, по большей части, используется в научной среде и позволяет строить двумерную (2D) и трехмерную (3D) графику. Исходно этот пакет разрабатывался в подражание MATLAB, но является независимым от него проектом.

Другой путь написания графических приложений – это использование кроссплатформенных графических библиотек. Одной из кроссплатформенных графических библиотек является Qt. Эта библиотека используется с такими языками, как C++, Java, Ruby, Delphi, Lazarus, и др. У нее имеется "привязка" и к Python – PyQt.

Компьютерная графика – это довольно обширная и сложная область знания. Она включает знания о технических средствах, позволяющих отображать изображение: растровые и векторные дисплеи, плоттеры и принтеры. Знание о способах формирования цветного изображения, которое воспринимается либо как свечение отдельных точек дисплея, либо как отражение, например, от листа бумаги: аддитивная цветовая модель RGB или субтрактивная - CMY. Обширная область математических и физических знаний помогает решать вопросы перемещения, поворота объекта, формирования второго и первого планов изображения (сцены), учета рефлексов (вторичных отражений) и еще много чего.

В этом пособии рассмотрен функционал графических модулей, который позволяет рисовать графики функций.

При построении графиков нам потребуются знания о функциях и методах графической системы, которые позволяют:

- формировать окно, в котором будет строиться график (размер, система координат);
- строить графические примитивы (точка, линия, ...);
- задавать ширину и цвет линий, цвет фона, выполнять заливку изображения или его части заданным цветом;

- управлять маркером (указателем), который используется для рисования
- наносить текст.

Модуль turtle

Модуль `turtle` входит в стандартную поставку библиотеки Python. Исходно этот модуль позиционируется как средство для обучения компьютерной графике детей.

Модель этого модуля следующая. Имеется прямоугольная поверхность, по которой ползает черепашка. Черепашка может перемещаться на заданное расстояние прямо, назад, под углом или по заданным координатам. Черепашку можно клонировать, создавая группу черепашек. При этом каждая черепашка живет своей жизнью. Для рисования черепашка использует цветовое перо (карандаш), которое может быть поднято или опущено. Если перо опущено, то остается след. Можно изменять цвет и толщину линии.

Черепашка понимает команды, с помощью которых можно нарисовать окружность заданного радиуса и цвета, дугу с заданным углом, залить фигуру определенным цветом, получить текущее состояние настроек или изменить их. Форма черепашки может быть изменена пользователем и использована как штамп, после которого на холсте остается рисунок.

В модуле `turtle` реализованы и интерактивные способы взаимодействия с черепашкой. События, связанные с кликом кнопки мыши или нажатия / отпускания клавиши клавиатуры, могут обрабатываться пользовательскими функциями, привязанными к этим событиям.

ВНИМАНИЕ:

1. Язык Python и его модули находятся в стадии постоянного развития и модификации. При использовании модулей следует контролировать их обновления и появление обновленной документации.
2. С электронной версией пособия распространяется файл `Turtle_3-6-1.pdf`, в котором приводится перечень функций и методов модуля `turtle` на русском языке.

Знакомство с функциями модуля `turtle` начнем с решения конкретной задачи: построим график функции, которая задавалась в графическом виде в лабораторной работе №2 Задание 1 (решим обратную задачу).

Решение первой задачи

На первом шаге оформим программу, написанную в лабораторной работе №2, Задание 1 в виде функции, а затем добавим графическую часть.

При построении графика функции мы будем перебирать значения аргумента функции в том диапазоне, который задан рисунком. Поскольку процесс получения аргумента программный, возможны ситуации, в общем случае, когда для заданного аргумента значение функции не может быть вычислено и работа программы завершиться с ошибкой. Ошибка может быть вызвана, например, делением на ноль или получением квадратного корня из отрицательного числа.

Для исключения подобных ситуаций необходимо исследовать функцию и те точки, в которых она не может быть вычислена, следует исключить условными операторами. Примем, что функция для таких точек возвращает значение `None`.

У нашей функции особых точек нет, но, тем не менее, в листинге функции, приведенном ниже, вставлен дополнительный код. Этот код приведен для примера и закомментирован, но в последующем вы сможете проверить его работу.

```
def MyFun(x):
    #      if (x >= 5) and (x <= 7):
    #          return(None)
    if x < -5:
```

```

y = 1
elif x >=-5 and x<0:
    y = -(3/5)*x-2
elif x >= 0 and x<2:
    y = -sqrt(4-x**2)
elif x >= 2 and x<4:
    y = x-2
elif x >= 4 and x<8:
    y = 2+sqrt(4-(x-6)**2)
else: y = 2
return(y)

```

Замечание: Параметры, необходимые для решения задания следует получить из графика и определить в программе, например, радиус дуги.

Графическая часть

При построении графика нам необходимо выполнить ряд условий и провести подготовительную работу. Прежде, чем перейти к написанию графической части программы следует разобраться с графической системой. Необходимо ответить на вопрос: "Как это работает?"

Замечание: Под монитором тут понимается внешнее устройство, которое подключается к видеокарте системного блока. Дисплей – устройство, с помощью которого формируется изображение. Дисплей является составной частью монитора. Изображение составляется из квадратных элементов - пикселов (pixel) дисплея, а каждый пиксель состоит из трех прямоугольных элементов, позволяющих формировать цвет пикселя в формате RGB. Одной из технических характеристик дисплея является разрешение: число точек по горизонтали и по высоте. Например, 1680x1050.

Графическое изображение может создаваться на дисплее монитора, принтере или плоттере. Каждое из этих устройств обладает определенными функциональными возможностями. С целью построения инвариантной графической системы (device-independent graphics) используют определенные положения. Рассмотрим некоторые из них для плоского рисунка.

1. В многозадачной системе каждое приложение может иметь свое графическое окно, которое может занимать всю поверхность дисплея или его часть. Такое окно назовем главным окном (main window). Размеры окна и его положение задаются в качестве аргументов функции `setup()`:

```
turtle.setup(Dw, Dh, Xc, Yc),
```

где Dw и Dh размеры окна по горизонтали и вертикали, соответственно, а Xc и Yc – координаты окна на дисплее. Если значения координат положительные, то они определяют положение верхнего левого угла главного окна, а если отрицательные – нижнего правого угла. Начало системы координат будет находиться в центре окна. Единицей измерения является пиксель. Размеры окна можно менять путем перетаскивания его границ. Направление осей системы координат зависит от режима, см. функцию `mode()`. Дополнительную информацию можно получить из документации на модуль `turtle`. Обратите внимание и на конфигурационный файл, в котором могут быть определены параметры главного окна и не только.

2. Для рисования графическая система Python предоставляет пользователю холст (полотно, канва – canvas). Размеры холста можно задать через функцию `screensize()`:

```
turtle.screensize(Cw, Ch, bg),
```

где Cw, Ch – размеры холста по горизонтали и вертикали, а bg – цвет фона.

Замечание: Если воспользоваться функцией screensize() для изменения только цвета фона;

```
  turtle.screensize(bg="yellow"),
```

то размеры холста будут установлены как размеры по умолчанию.

Другой способ определить размеры холста, а заодно и положение мировой системы координат (world coordinate system) – это использовать функцию setworldcoordinates():

```
  turtle.setworldcoordinates(Xlc, Ylc, Xrc, Yrc),
```

где и X_{rc} , Y_{rc} – координаты левого нижнего и правого верхнего углов холста, соответственно. При этом в графической системе устанавливается режим "world". Для задания цвета фона может использоваться функция bgcolor(). Если размеры холста больше размеров главного окна, то появляются полосы прокрутки – виджеты холста.

И главное окно, и холст инициализируются при импорте модуля `turtle`. Значения, для инициализации, берутся из конфигурационного файла, а при его отсутствии устанавливаются по умолчанию. Кроме этого можно, используя функционал модуля, изменять размеры и положение главного окна, а так же параметры холста по своему усмотрению.

Если размеры главного окна и его координаты задаются в пикселях, то размеры холста – пользовательские единицы. Например, вы рисуете на листе бумаги геометрические фигуры, а размеры задаете в миллиметрах. Если ваш лист формата А4, то размеры холста можно определить как 210x297, а координаты некоторой точки на этом холсте можно задать как (34.7; 12.0). Если единицей измерения являются сантиметры, то размеры холста можно задать как 21x29.7, а координаты этой же точки будут – (3.47; 1.2). И наконец, если вы рисуете Солнечную систему, то с таким же успехом можно определить размеры холста в астрономических единицах и указывать положения изображаемых тел в тех же единицах и их долях.

В каких отношениях находятся введенные понятия: главное окно, холст, мировая система координат? Следующий рисунок поясняет это.

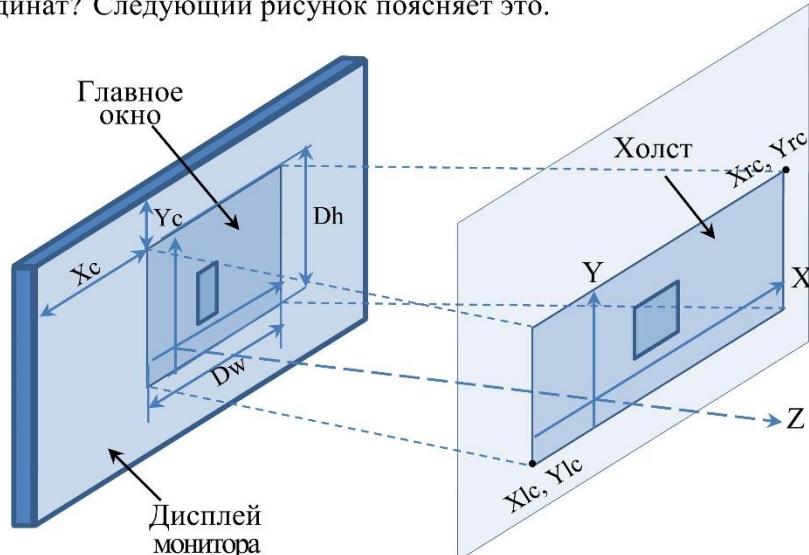


Рис.1. Связь дисплея, главного окна, холста и мировой системы координат

Холст размещается в плоскости проектирования ($Z=0$). Координаты X_{lc} и Y_{lc} определяют положение левого нижнего угла холста, а X_{rc} , Y_{rc} – правого. Размеры холста составляют: $C_w = X_{rc} - X_{lc}$ и $C_h = Y_{rc} - Y_{lc}$. Графическая система проектирует холст на главное окно так, что угловые точки главного окна и холста совпадают, и устанавливается соответствие между единицами измерения в мировой системе координат и пикселями главного окна – масштаб. Масштабы, установленные по осям X и Y, могут быть разными.

Из рисунка очевиден следующий момент: если отношение ширины холста к его высоте и отношение ширины главного окна к его высоте (соотношение сторон – aspect ratio) неодинаковые, то возникает искажение. Искажаются углы и отношения между элементами изображения. Как показано на рис.1, квадрат преобразуется в прямоугольник, поскольку ширина холста больше ширины главного окна при их равной высоте. Этот эффект может быть использован, например, для изменения размеров изображения или преднамеренного их искажения. Кстати, аналогичный эффект можно наблюдать с графиками в Excel.

Замечание: На рисунке координаты левого нижнего угла холста отрицательные и начало системы координат находится на холсте. Если координаты угла определить как положительные числа, то начало координат будет находиться вне холста. Фигуры, размещаемые на холсте, могут находиться за размерами холста и, соответственно проектироваться вне главного окна. Для обнаружения таких "убежавших" фигур можно изменять размеры путем перетаскивания границ мышкой.

Прежде, чем строить график, надо принять решение о том, в каких диапазонах будут изменяться аргумент и значение функции. Примем следующие обозначения: X_{\min} , X_{\max} – диапазон изменения аргумента, а Y_{\min} , Y_{\max} – диапазон значений функции. Тогда параметры холста можно задать так:

```
turtle.setworldcoordinates(Xmin, Ymin, Xmax, Ymax)
```

Следует учитывать и то, что бордюр окна перекрывает часть изображения, поэтому параметры холста и соответствующие параметры главного окна можно задавать с небольшим запасом. Это небольшая проблема и она может быть решена в процессе отладки программы.

Если считать, что ширина и высота главного окна, в которое будет спроектирован холст, составляют D_w и D_h , соответственно, то для исключения искажений углов и элементов графика следует удовлетворить следующее условие:

$$k = \frac{D_w}{D_h} = \frac{X_{\max} - X_{\min}}{Y_{\max} - Y_{\min}} \text{ или, например, } D_h = D_w/k$$

Теперь можно перейти к решению наших задач.

Первый шаг:

Определим границы области, в которой будем рисовать наш график: границы изменения аргумента (x) и границы изменения значений функции (y). В общем случае, когда нам неизвестно поведение функции на заданном интервале аргумента, можно предварительно выполнить вычисления функции и найти минимальное и максимальное значения. Эти значения помогут задать границы области построения. К тому же такие вычисления можно сделать один раз, поместив результаты вычислений либо в массив, либо в список, который затем можно использовать при построении графика.

Вновь обратимся к лабораторной работе №2, Задание 1, и примем обозначения переменных, которые мы будем использовать в нашей программе.

а) Импорт модулей. Нам потребуются математические функции и функции модуля `turtle`:

```
from math import sqrt
import turtle as tr
```

б) Размеры по горизонтали и вертикали составляют (-10, 10) и (-2, 4) условных единиц соответственно. Увеличим эти размеры на 20% (по 10% с каждой стороны) с тем, что бы график был в границах главного окна (выбор процентов – это субъективная вещь). Мы получим:

```
aX = [-12, 12] # левая и правая
aY = [-3, 5] # нижняя и верхняя
```

в) Определим размер главного окна по горизонтали (Dx) в 800 пикселов. Тогда, при заданных параметрах графика, высота главного окна, для исключения искажений, составит:

```
Dx = 800  
Dy = Dx / ((aX[1] - aX[0]) / (aY[1] - aY[0]))  
# создаем главное окно  
tr.setup(Dx, Dy)
```

г) Установим число точек для получения качественного рисунка:

```
# число точек рисования  
Nmax = 1000
```

д) Установим мировую систему координат

```
tr.setworldcoordinates(aX[0], aY[0], aX[1], aY[1])
```

Второй шаг

На этом шаге нарисуем оси, нанесем деления и надписи, нарисуем стрелки. Для этого используем такие функции модуля `turtle`, как `up()`, `down()`, `goto()`, `write()`, `begin_fill()`, `end_fill()`.

Рисование осей, меток, стрелок и надписей к осям, оформлено в виде функций.
Алгоритм рисования оси прост:

- поднять перо
- перейти к левой (нижней) границе
- опустить перо
- перейти к правой (верхней) границе

Алгоритм нанесения делений и надписей

- поднять перо
- в цикле от левой (нижней) границы к правой (верхней) границе, с заданным шагом
- перейти к точке на оси
- опустить перо
- перейти к точке ниже (правее) оси. Расстояние от оси (длина штриха) подбирается опытным путем
- поднять перо
- перейти к точке нанесения надписи (ниже / правее оси): определяем опытным путем
- выводим число, предварительно преобразовав его в строку.
- если цикл не завершился, то перейти к следующей итерации.

Алгоритм рисования стрелки:

- подготавливаем два списка с координатами характерных точек рисунка а и б. Пер первую точку с координатами (0, 0) не указываем, см. рис 2
- в цикле строим многоугольник и регистрируем его как новую форму черепашки.
- назначаем черепашке новую форму – наш многоугольник и вытягиваем стрелку
- устанавливаем черепашку в то место, где должна быть стрелка, разворачиваем ее под нужным углом и создаем штамп
- надписываем ось

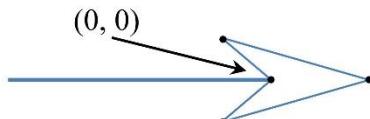


Рис.2. Характерные точки стрелки, использованные в программе

Третий шаг

Переходим к рисованию функции. На этом шаге выберем другой цвет и толщину линии. По размеру области значений аргумента и по числу точек Nmax определяем приращение по оси X:

$$dx = (ax[1] - ax[0]) / Nmax.$$

Алгоритм рисования следующий:

- получить начальную координату, вычислить значение функции
- установить черепашку в начальную позицию
- в цикле while, от начального значения аргумента до конечного значения
- получить текущее значение аргумента, вычислить значение функции
- если функция не вычислена (None), перо поднять, а иначе перейти к точке и опустить перо
- продолжить цикл пока не получено значение, превышающее значение для правой точки аргумента

Замечание: Следует обратить внимание на то, что в теле цикла проводится проверка на допустимость значений для Y. Так как мы приняли, что для особых точек функция возвращает значение None, то при обнаружении такой ситуации процесс рисования пропускается (перо поднимается).

Собственно все. Листинг программы и результат работы приводится.

PS: Удалите символы комментария в функции и посмотрите на результат.

Листинг программы

```
# -*- coding: cp1251 -*-
from math import sqrt
import turtle as tr
#
def Fun1(x):
    """
    Кривая из лаб. 2 Задание 1
    """
    #if (x >= 5) and (x <= 7):
    #    return(None)
    if x < -5:
        y = 1
    elif x >= -5 and x < 0:
        y = -(3/5) * x - 2
    elif x >= 0 and x < 2:
        y = -sqrt(4 - x**2)
    elif x >= 2 and x < 4:
        y = x - 2
    elif x >= 4 and x < 8:
        y = 2 + sqrt(4 - (x - 6)**2)
    else: y = 2
    return(y)

def Axis(txy, ax = 'X'):
    """
    Рисование оси.
    txy - список: [Xmin, Xmax]
                  или      [Ymin, Ymax]
    ax - 'X' или 'Y'
    """

```

```

a = txy[0]
b = txy[1]
tr.up()
if (ax == 'X'):
    pb = [a, 0]
    pe = [b, 0]
else:
    pb = [0, a]
    pe = [0, b]
tr.goto(pb)
tr.down()
tr.goto(pe)

def Mark(txy, ax = 'X'):
    """
    Маркировка оси.
    txy - список: [Xmin, Xmax]
        или      [Ymin, Ymax]
    ax - 'X' или 'Y'
    """
    a = txy[0]
    b = txy[1]
    tr.up()
    for t in range(a, b):
        if (ax =='X'):
            pb = [t, 0]
            pe = [t, 0.2]
            pw = [t, -0.5]
        else:
            pb = [0, t]
            pe = [0.2, t]
            pw = [0.2, t]
        tr.goto(pb)
        tr.down()
        tr.goto(pe)
        tr.up()
        tr.goto(pw)
        tr.write(str(t))

def Arrow(txy, ax = 'X'):
    """
    Рисование стрелки.
    txy - список: [Xmin, Xmax]
        или      [Ymin, Ymax]
    ax - 'X' или 'Y'
    """
    # Параметры многоугольника
    a = [0.1, 0, -0.1]
    b = [-0.1, 0.3, -0.1]
    tr.up()
    tr.goto(0, 0)                      # в начало
    tr.begin_poly()                    # начинаем запись вершин
    for i in range(2):                # для всех вершин

```

```

        tr.goto(a[i], b[i])    # многоугольника
tr.end_poly()                      # останавливаем запись
p = tr.get_poly()                  # ссылка на многоугольник
tr.register_shape("myArrow", p)    # регистрируем
tr.resizemode("myArrow")          # черепашке новую форму
tr.shapesize(1,2,1)                # растягиваем (это для
примера)
if (ax == 'X'):                   # для оси X
    tr.tiltangle(0)               # угол для формы
    tr.goto(txy[1] + 0.2,0)       # к месту стрелки
    pw = [int(txy[1]), -1.0]     # место для надписи
else:                            # для оси Y
    tr.tiltangle(90)             # новый угол для формы
    tr.goto(0, txy[1] + 0.2)      # к месту стрелки
    pw = [0.2, int(txy[1])]     # место для надписи
    tr.stamp()                  # оставить штамп - стрелка
# надпишем ось
    tr.goto(pw)                  # к месту надписи
    tr.write(ax, font=("Arial",14,"bold"))
#
#
def main():
# Начальные параметры
# *****
# Границы графика: [Xmin, Xmax] и [Ymin, Ymax]
aX = [-12, 12] # левая и правая
aY = [-3, 5]   # нижняя и верхняя
# Главное окно
Dx = 800
Dy = Dx / ((aX[1] - aX[0]) / (aY[1] - aY[0]))
tr.setup(Dx, Dy)
tr.reset()
# Число точек рисования
Nmax = 1000
#
# Установка мировой системы координат
tr.setworldcoordinates(aX[0], aY[0], aX[1], aY[1])
#
tr.title("Lab_8_2_1")           # заголовок
tr.width(2)                     # толщина линии
tr.color("blue", "blue")         # цвет и заливка
# *****
tr.ht()                         # невидимая
tr.tracer(0,0)                  # нет задержек
#
# X - ось, метки, стрелка
Axis(aX, 'X')
Mark(aX, 'X')
Arrow(aX, 'X')
# Y - ось, метки, стрелка
Axis(aY, 'Y')
Mark(aY, 'Y')
Arrow(aY, 'Y')

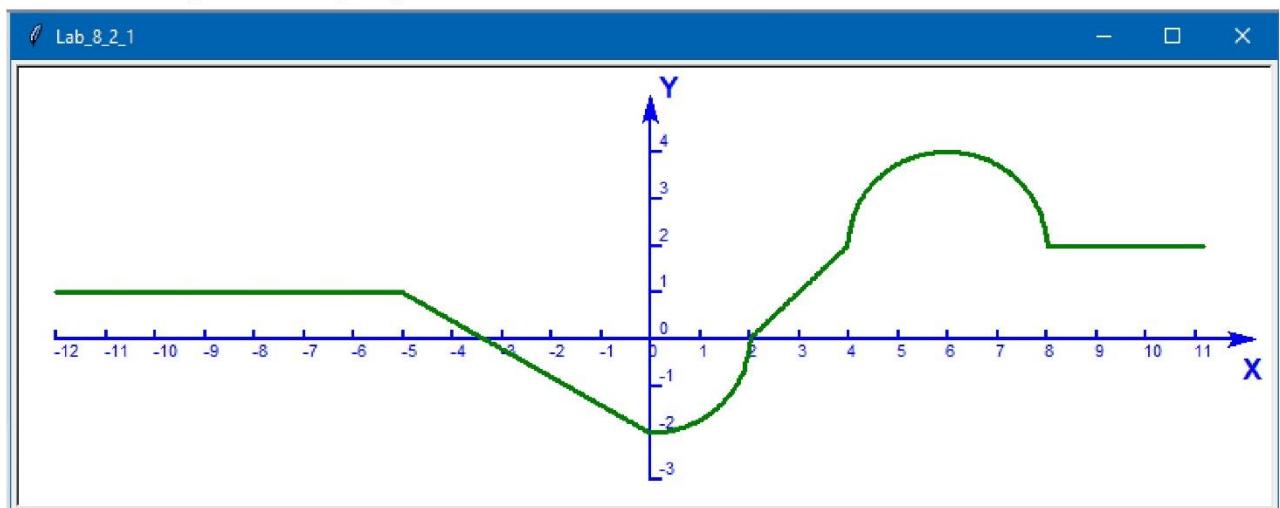
```

```

#
# Функция
    tr.color("green")          # цвет линии
    tr.width(3)                # и толщина
    dx = (aX[1]-aX[0])/Nmax   # шаг
#
# в начало
    x = aX[0]
    y = Fun1(x)
    if (y is None):
        tr.up()
        tr.goto(x, 0)
    else:
        tr.goto(x, y)
        tr.down()
#
# рисуем
    while x <= aX[1]:
        x = x + dx
        y = Fun1(x)
        if (y is None):
            tr.up()
            continue
        else:
            tr.goto(x, y)
            tr.down()
#
if __name__ == "__main__":
    main()
#
# Комментировать при работе в IDLE
# tr.mainloop()

```

Результат работы программы



Вторая задача

Для решения второй задачи (лабораторная работа №2, Задание 2), нам потребуется понимание того, что такое метод Монте-Карло и генератор случайных чисел.

Напомним, что в задании требовалось написать программу, которая определяет по введенным пользователем координатам точки, попадает ли эта точка в заштрихованную область.

Теоретическое введение

Метод Монте-Карло – численный метод решения математических задач при помощи моделирования случайных величин (метод статистических испытаний).

Одним из простейших механизмов генерации случайных величин является ... рулетка – основной атрибут игорных домов. Европейский город, знаменитый игорными домами – Монте-Карло, столица княжества Монако. От имени этого города и пошло название метода.

Одной из задач, решаемых методом Монте-Карло, является задача вычисления площади или объема сложной фигуры. Суть метода в следующем. Если фигура изображена на плоскости, то около нее можно описать другую фигуру, площадь которой мы можем вычислить точно. Например, круг или правильный многоугольник. Генерируя N случайных точек, координаты которых будут равномерно распределены по поверхности описанной фигуры, мы можем подсчитать число точек, которые попали в фигуру с неизвестной площадью – N_f . Зная площадь описывающей фигуры S , можно вычислить площадь искомой фигуры с определенной точностью.

$$S_f = S \cdot \frac{N_f}{N}$$

Точность будет тем выше, чем больше точек будет сгенерировано и чем равномернее они будут разбросаны по всей описывающей фигуре. Точность пропорциональна $\sqrt{\frac{D}{N}}$, где D – некоторая постоянная, а N – число испытаний (число точек).

Обратите внимание на то, что для повышения точности на порядок (в 10 раз), потребуется увеличить число испытаний в 100 раз. Применение метода стало возможным с развитием вычислительной техники.

Больше информации о методе Монте-Карло можно найти в Интернете.

Решение второй задачи

Для нашей задачи необходим генератор, который формирует случайные числа с вероятностью, равномерно распределенной в заданном интервале. Для этой цели используем функцию `uniform()` модуля `random`:

```
from random import uniform
```

Используя листинг программы, написанной в лабораторной работе №2, Задание 2 и знания, полученные при решении предыдущей задачи, нам несложно написать программу, в которой координаты точек будут генерироваться случайным образом, и результат попадания будет отображаться графически.

Определение точной площади фигуры

В задачах, которые описаны в лабораторной работе 2 Задание 2, площадь большинства заштрихованных фигур находится просто.

Поскольку наша фигура образована пересечением кубической параболы и прямой, точное определение площади возможно с использованием интегрального исчисления.

В общем случае площадь фигуры, образованной двумя линиями, можно найти из выражения: $S_f = \int_a^b (f_2(x) - f_1(x)) \cdot dx$, где $f_2(x)$ и $f_1(x)$ – функции, которыми ограничена фигура, a и b – границы фигуры слева и справа, соответственно. При этом $f_2(x) \geq f_1(x)$. Уравнения для наших линий следующие:

$$f_1(x) = 2 \cdot x + 2 \quad \text{и} \quad f_2(x) = x^3 - 4 \cdot x^2 + x + 6$$

Разобьем нашу фигуру на две области. Одна область будет в диапазоне аргумента $[-1, 1]$, а вторая – $[1, 4]$. Тогда получим, что площадь равна сумме двух интегралов:

$$S_f = \int_{-1}^1 (f_2(x) - f_1(x)) \cdot dx + \int_1^4 (f_1(x) - f_2(x)) \cdot dx$$

Подробного решения не приводим. Вычисление дает: $S_f = \frac{253}{12} \approx 21.0833$

Для вычисления площади фигуры методом Монте-Карло опишем около нее прямоугольник с основанием 7 и высотой 13 условных единиц (диапазон $aX = [-2, 5]$ и $aY = [-2, 11]$). Площадь прямоугольника составляет $S = 91$.

В программе нет каких-либо особенностей. Следует обратить внимание лишь на то, что рисование большого числа точек требует много времени.

Для ускорения вычислений можно не рисовать точки, которые не попадают в заштрихованную область, т.е. оставить только первую часть условного оператора, который находится в теле цикла генерации координат точек.

Листинг программы

```
# -*- coding: cp1251 -*-
import turtle as tr
from random import uniform
#
def fun2_2(x, y):
    if (x < -1) or (x > 4):
        flag = 0          #False
    if (x>=-1) and (x<1) and (y>=2*x+2) \
       and (y<=x**3-4*x**2+x+6) or (x>=1) \
       and (x<=4) and (y>=x**3-4*x**2+x+6) \
       and (y<=2*x+2):
        flag = 1
    else:
        flag = 0
    return(flag)

# Инициализация turtle
# *****
# Границы графика
aX = [-2, 5]      # левая и правая
aY = [-2, 11]     # нижняя и верхняя
Dx = 300
Dy = Dx/((aX[1] - aX[0])/(aY[1] - aY[0]))
tr.setup(Dx, Dy, 200, 200)
tr.reset()
# число точек рисования
Nmax = 10000
#
# Установка мировой системы координат
tr.setworldcoordinates(aX[0], aY[0], aX[1], aY[1])
#
tr.title("Lab_8_2_2")      # заголовок
tr.width(2)                # толщина линии

tr.ht()                    # невидимая
tr.tracer(0,0)  # 0 - задержка между обновлениями
# *****
```

```

#
# рисование функции
tr.up()
mfun = 0           # точек попало в
                   # заштрихованную область
for n in range(Nmax):
    # генерируем координаты точки
    x = uniform(aX[0],aX[1])
    y = uniform(aY[0],aY[1])
    tr.goto(x,y)
    if fun2_2(x,y) != 0:   # попала
        tr.dot(3,"green")
        mfun += 1
    #     else:                 # не попала
    #         tr.dot(3, "#ffccff")
    #
    tr.color("blue", "blue") # цвет и заливка
    #
    # Рисуем оси
    # Ось X
    tr.up()
    tr.goto(aX[0], 0)
    tr.down()
    tr.goto(aX[1], 0)
    # Ось Y
    tr.up()
    tr.goto(0, aY[1])
    tr.down()
    tr.goto(0, aY[0])
    #
    # координатные метки
    # и надписи на оси X
    tr.up()
    for x in range(aX[0], aX[1]):
        tr.goto(x, 0.1)
        tr.down()
        tr.goto(x, 0)
        tr.up()
        tr.sety(-0.4)
        coords = str(x)
        tr.write(coords)
    #
    # на оси Y
    for y in range(aY[0], aY[1]):
        tr.goto(0, y)
        tr.down()
        tr.goto(0.1, y)
        tr.up()
        tr.setx(0.2)
        coords = str(y)
        tr.write(coords)
    #
    # Рисуем стрелки

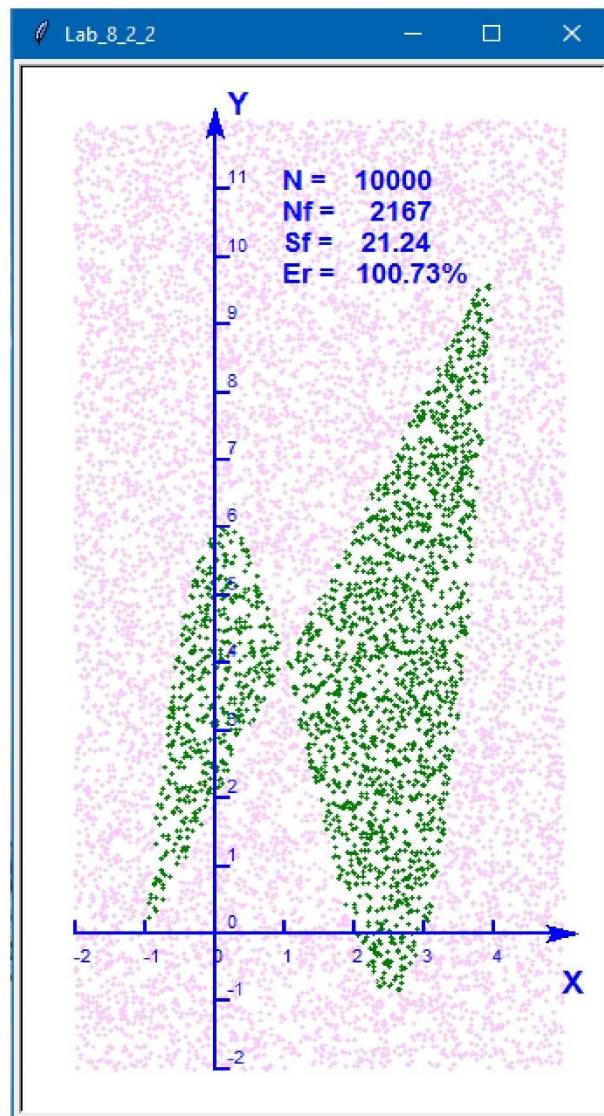
```

```

# на оси X
poli = [0, 0.1, 0, -0.1, 0]
Arrbeg = int(aX[1])
Xpoli = [Arrbeg, Arrbeg - 0.1, Arrbeg+0.3,
          Arrbeg - 0.1, Arrbeg]
tr.goto(Xpoli[0], poli[0])
tr.begin_fill()
tr.down()
for i in range(1,5):
    tr.goto(Xpoli[i], poli[i])
tr.end_fill()      # заливаем стрелку
# Надпишем ось X
tr.up()
tr.goto(Arrbeg, -0.7)
tr.write("X", font=("Arial",14,"bold"))
#
# на оси Y
Arrbeg = int(aY[1])
Ypoli = [Arrbeg, Arrbeg - 0.1, Arrbeg + 0.3,
          Arrbeg - 0.1, Arrbeg]
tr.up()
tr.goto(poli[0], Ypoli[0])
tr.begin_fill()
tr.down()
for i in range(1,5):
    tr.goto(poli[i], Ypoli[i])
tr.end_fill()
# Надпишем ось Y
tr.up()
tr.goto(0.2, Arrbeg)
tr.write("Y", font=("Arial",14,"bold"))
Sf = (aX[1] - aX[0]) * (aY[1] - aY[0]) * mfun/Nmax
tr.goto(1, 9)
meseg = "N = {0:8d}\nNf = {1:8d}\nSf = {2:8.2f}" \
        .format(Nmax, mfun, Sf)
tr.write(meseg, font=("Arial",12,"bold"))
print(meseg)
#
# Комментировать при работе в IDLE
# tr.done()

```

Результат работы программы



Литература

1. Прохоренок Н.А., В.А. Дронов, Python 3 и PyQt 5. Разработка приложений, БХВ-Петербург, 2017
2. Эйнджел Э., Интерактивная компьютерная графика. Вводный курс на базе OpenGL, 2 изд., и.д. "Вильямс", 2001
3. Хахаев И.А., Практикум по алгоритмизации и программированию на Python, Альт Линукс, 2011
4. Ермаков С.М., Метод Монте-Карло в вычислительной математике (вводный курс), СПб., 2009
5. Новожилов Б.В., Метод Монте-Карло. М.: Знание, 1966.

Задание к лабораторной работе №8 "Программирование в графическом режиме"

Замечание: Эти задания лучше выполнить с модулем `Matplotlib`, подготовив отдельную лабораторную работу.

С модулем `Turtle`, повидимому, лучше выполнить лабораторную работу 3 (Задания 1, 2, 3). При этом, при выполнении Задания 3 рисовать два графика: график с прямым вычислением значений функции и график функции, значения которой получаются путем вычисления рядов.

Изображение должно занимать большую часть экрана, сопровождаться заголовком, содержать наименования и градации осей и масштабироваться в зависимости от исходных данных. При любых допустимых значениях исходных данных изображение должно полностью помещаться на экране. Программа не должна опираться на конкретные значения разрешения экрана.

Вывести на экран в графическом режиме графики двух функций на интервале от $x_{\text{нач}}$ до $x_{\text{кон}}$ с шагом dx . Первая функция задана с помощью ряда Тейлора, ее вычисление должно выполняться с точностью ε . Значение параметра b для второй функции вводится с клавиатуры. Графики должны быть плавными и различаться цветами.

Вариант 1

$$y(x) = 2 \cdot \sum_{n=0}^{\infty} \frac{1}{(2 \cdot n + 1) \cdot x^{2 \cdot n + 1}} = 2 \cdot \left(\frac{1}{x} + \frac{1}{3 \cdot x^3} + \frac{1}{5 \cdot x^5} + \dots \right), \quad |x| > 1;$$
$$z(x) = \ln \frac{x+1}{x-1} + b.$$

Вариант 2

$$y(x) = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^n}{n!} = \left(1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \dots \right), \quad |x| < \infty;$$
$$z(x) = e^{-x} + b;$$

Вариант 3

$$y(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!} = \left(1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots \right), \quad |x| < \infty$$
$$z(x) = e^x + b.$$

Вариант 4

$$y(x) = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2 \cdot n + 1) \cdot x^{2 \cdot n + 1}} = \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3 \cdot x^3} - \frac{1}{5 \cdot x^5} + \dots, \quad x > 1;$$
$$z(x) = \arctan x + b.$$

Вариант 5

$$y(x) = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2 \cdot n + 1}}{(2 \cdot n + 1)} = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots, \quad |x| \leq 1$$
$$z(x) = \arctan x + b.$$

Вариант 6

$$y(x) = 2 \cdot \sum_{n=0}^{\infty} \frac{1}{(2 \cdot n + 1) \cdot x^{2 \cdot n + 1}} = 2 \cdot \left(\frac{1}{x} + \frac{1}{3 \cdot x^3} + \frac{1}{5 \cdot x^5} + \dots \right), \quad |x| > 1;$$
$$z(x) = \operatorname{Arth} x + b.$$

Вариант 7

$$y(x) = -\frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2 \cdot n + 1) \cdot x^{2 \cdot n + 1}} = -\frac{\pi}{2} - \frac{1}{x} + \frac{1}{3 \cdot x^3} - \frac{1}{5 \cdot x^5} + \dots, \quad x < -1$$

$$z(x) = \arctgx + b.$$

Вариант 8

$$y(x) = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2 \cdot n}}{n!} = 1 - x^2 + \frac{x^4}{2!} - \frac{x^6}{3!} + \frac{x^8}{4!} - \dots, \quad |x| < \infty$$

$$z(x) = e^{-x^2} + b.$$

Вариант 9

$$y(x) = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^n}{(2 \cdot n)!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6} + \dots, \quad |x| < \infty$$

$$z(x) = \cos x + b.$$

Вариант 10

$$y(x) = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2 \cdot n}}{(2 \cdot n + 1)!} = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \dots, \quad |x| < \infty$$

$$z(x) = \frac{\sin x}{x} + b.$$

Вариант 11

$$y(x) = 2 \cdot \sum_{n=0}^{\infty} \frac{(x-1)^{2 \cdot n+1}}{(2 \cdot n + 1) \cdot (x+1)^{2 \cdot n+1}} = 2 \cdot \left(\frac{x-1}{x+1} + \frac{(x-1)^3}{3 \cdot (x+1)^3} + \frac{(x-1)^5}{5 \cdot (x+1)^5} + \dots \right), \quad x > 0;$$

$$z(x) = \ln x + b.$$

Вариант 12

Написать программу, которая выводит на экран секторную диаграмму. Диаграмму снабдить заголовком и наименованием для каждого сектора. Исходные данные сформировать в текстовом файле. Количество секторов задавать в программе в виде именованной константы.

Построение секторной диаграммы оформить в виде процедуры. Параметры процедуры: координаты центра диаграммы; радиус; количество секторов; массив процентов; массив наименований. Пример исходных данных см. Таблица 1.

Вариант 13

Написать программу, которая выводит на экран две секторные диаграммы, расположив их рядом. Диаграмму снабдить заголовком и наименованием для каждого сектора. Исходные данные сформировать в текстовом файле. Количество секторов задавать в программе в виде именованной константы.

Построение секторной диаграммы оформить в виде процедуры. Параметры процедуры: координаты центра диаграммы; радиус; количество секторов; массив процентов; массив наименований. Пример исходных данных см. Таблица 1.

Вариант 14

Написать программу, которая выводит на экран две столбиковые диаграммы. На экране диаграммы расположить рядом, каждую в своих координатных осях. Каждую диаграмму снабдить заголовком и наименованием единиц измерений по осям X и Y. Исходные данные сформировать в текстовом файле. Количество столбцов задавать в программе в виде именованной константы. Построение диаграммы оформить в виде процедуры. Пример исходных данных см. Таблица 1.

Вариант 15

Написать программу, которая выводит на экран две столбиковые диаграммы в одной координатной плоскости. Диаграмму снабдить градацией осей и заголовком. Исходные данные сформировать в текстовом файле. Количество столбцов задавать в программе в виде именованной константы. Построение диаграммы оформить в виде процедуры. Пример исходных данных см. Таблица 1.

Вариант 16

Написать программу, которая выводит на экран трехмерную столбиковую диаграмму. Диаграмму снабдить градацией осей и заголовком. Исходные данные сформировать в текстовом файле. Количество столбцов задавать в программе в виде именованной константы. Построение диаграммы оформить в виде процедуры. Пример исходных данных см. Таблица 1.

Вариант 17

Написать программу, которая выводит на экран столбиковую диаграмму, представляющую оптовые и розничные цены на различные наименования кофе. Исходные данные сформировать в текстовом файле.

Построение диаграммы оформить в виде процедуры. Параметры процедуры: количество наименований; массив значений оптовых цен; массив значений розничных цен; массив наименований. Наименования товаров разместить вертикально под осью абсцисс.

Вариант 18

Написать программу, которая выводит на экран столбиковую диаграмму, представляющую максимальную и среднюю норму прибыли при реализации различных сортов шоколада. Исходные данные сформировать в текстовом файле самостоятельно.

Построение диаграммы оформить в виде процедуры. Параметры процедуры: количество наименований; массив значений оптовых цен; массив значений розничных цен; массив наименований. Наименования товаров разместить вертикально под осью абсцисс.

Вариант 19

Написать программу, которая выводит на экран графики динамики изменения максимального, минимального и среднего курса доллара за заданное количество дней. Исходные данные сформировать в текстовом файле самостоятельно.

Построение графика оформить в виде процедуры. Параметры процедуры: массив дат; количество дней; массивы максимальных, минимальных и средних значений.

Вариант 20

Написать программу, которая выводит на экран трехмерную столбиковую диаграмму курса немецкой марки по отношению к рублю за заданное количество дней. Исходные данные сформировать в текстовом файле самостоятельно.

Построение диаграммы оформить в виде процедуры. Параметры процедуры: массив дат; количество дней; массив значений по оси Y; код заполнителя.

Пример исходных данных для вариантов 12 - 16

Таблица 1. Лидеры мирового рынка ПК

Рейтинг 1996 г.	Поставщик	Объем	Доля	Объем	Доля	Рост 95/96, %
		Продаж 1996 г. тыс. шт.	Рынка 1996 г. , %	Продаж 1995 г. тыс. шт.	Рынка 1995 г. , %	
1	Compaq	7 036	10,3	5 757	9,8	22
2	IBM	6 081	8,9	4 785	8,1	27
3	Packard Bell, NEC	4 247 3 587	6,2 5,2	4 392 4 627	7,5 7,9	-3 22
4	Apple	2 995	4,4	2 023	3,4	48
5	HP	44 459	65,0	37 221	63,3	19
6	Другие					

Примечание: Попробуйте обновить информацию, получив необходимые данные из сети Интернет.

Заключение

Изложенный в этом пособии материал содержит большой объём программного кода. В процессе подготовки к изданию все программы пособия были проверены и отлажены с использованием интерпретатора Python 3.6.1.

Предполагается, что читатель внимательно отнесётся к представленной информации и сможет самостоятельно разобраться в возможных ошибках, которые часто возникают в процессе подготовки издания. Замечу, что работа над ошибками позволяет более глубоко понять излагаемый материал, а поэтому настоятельно рекомендуется проверять работоспособность приведённых программ.

Алгоритмы решений, использованные в пособии, не являются идеальными и более того, они написаны так, что бы не очень подготовленный читатель смог понять основную нить предлагаемого решения. По желанию, все программы можно доработать или модифицировать для получения более приемлемого, с точки зрения читателя, решения.

ПРИЛОЖЕНИЕ 1

Таблица 1. Функции и методы в Python 3

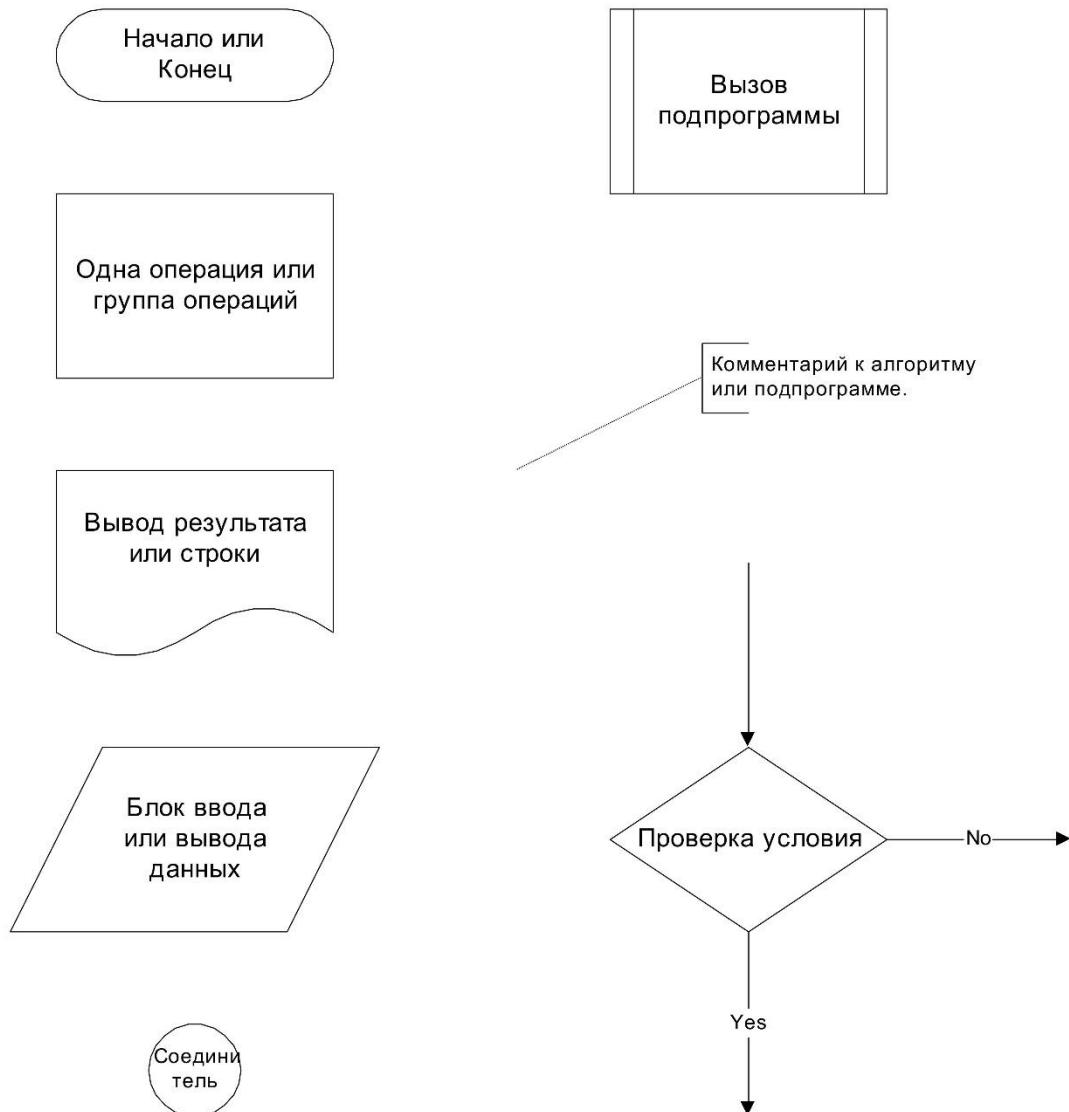
Константы (Math)	Описание и пример использования
pi	Возвращает число π . <code>math.pi => 3.141592653589793</code>
e	Возвращает число e . <code>math.e => 2.718281828459045</code>
Функции (Sys)	
<code>int([<Объект>[,<Система счисления>]])</code>	Преобразует объект в целое число. Второй параметр указывает систему счисления <u>преобразуемого числа</u> . По умолчанию используется десятичная система счисления. <code>>>>int(7.5), int('71',10), int('0o71',8)⁵</code> <code>(7, 71, 57)</code> <code>>>>int("0xA",16)</code> <code>10</code> <code>>>>int(), int("0b11111111",2)</code> <code>(0, 255)</code>
<code>float([<Число или строка>])</code>	Преобразует целое число или строку в вещественное число. <code>>>>float(7), float("7.1"), float("12.")</code> <code>(7.0, 7.1, 12.0)</code> <code>>>>float("inf"), float("-inf"), float("nan")</code> <code>(inf, inf, nan)</code>
<code>bin(<Число>)</code>	Преобразует десятичное число в двоичное. Возвращает строковое представление числа. <code>>>>bin(255), bin(1), bin(-45)</code> <code>('0b11111111', '0b1', '-0b101101')</code>
<code>oct(<Число>)</code>	Преобразует десятичное число в восьмеричное. Возвращает строковое представление числа. <code>>>>oct(7), oct(8), oct(64)</code> <code>('0o7', '0o10', '0o100')</code>
<code>hex(<Число>)</code>	Преобразует десятичное число в шестнадцатеричное. Возвращает строковое представление числа. <code>>>>hex(10), hex(16), hex(255)</code> <code>('0xa', '0x10', '0xff')</code>
<code>round(<Число>[, Кол-во знаков после точки])</code>	Вещественное число округляется до целого по следующему правилу: дробная часть меньше 0.5 – округление вниз; равна 0.5 – округление до четного числа; больше 0.5 – округление вверх. <code>>>>round(1.49), round(1.50), round(1.51)</code> <code>(1, 2, 2)</code> <code>>>>round(2.49), round(2.50), round(2.51)</code> <code>(2, 2, 3)</code> Второй параметр определяет число цифр после запятой. В этом случае округление выполняется по правилу: отбрасываемая часть меньше 0.5 – округление вниз; больше или равно 0.5 – округление вверх.

⁵ В Python можно использовать одиночные и двойные прямые кавычки. Важно, что бы начальная и конечная кавычки были одинаковыми. При наборе программы в Word необходимо настроить соответствующие пункты автозамены: Файл → Параметры → Правописание → Параметры автозамены (для Office 2010)

	<code>>>>round(2.449,1), round(2.450,1), round(2.451,1)</code> <code>(2.49, 2.5, 2.5)</code>
<code>abs(<Число>)</code>	Возвращает абсолютное значение.
<code>pow(<Число>, <Степень>, [<Делитель>])</code>	Возвращает число, введенное в степень. Степень может быть вещественного типа. Если указан третий параметр, то возвращается остаток от деления результата на значение этого параметра. <code>>>>pow(3, 3), pow(3, 3, 2), pow(3, 3.25)</code> <code>(27, 1, 35.533998349717294)</code>
<code>max(<Список чисел через запятую>)</code>	Возвращается максимальное число из списка.
<code>min(<Список чисел через запятую>)</code>	Возвращается минимальное число из списка.
<code>sum(<Последовательность> [, <Начальное значение>])</code>	Возвращает сумму значений элементов последовательности (списка, кортежа) плюс начальное значение. <code>>>>sum([1, 2, 3], 7), sum((1, 2, 3))</code> <code>(13, 6)</code>
<code>divmod(x, y)</code>	Возвращает кортеж из двух значений: целая часть и остаток от деления: $x // y$ и $x \% y$.
Функции (Math)	
<code>sin(x), cos(x), tan(x)</code>	Стандартные тригонометрические функции. Угол задается в радианах.
<code>asin(x), acos(x), atan(x)</code>	Обратные тригонометрические функции
<code>degrees(x)</code>	Преобразование радиан в градусы
<code>radians(x)</code>	Преобразование градусов в радианы
<code>exp(x)</code>	Экспонента
<code>log(<Число>[, <База>])</code>	Логарифм числа по основанию <База>. Если база не указана, то вычисляется натуральный логарифм.
<code>log10(x), log2(x)</code>	Десятичный и двоичный логарифмы (по базе 10 и 2 соответственно)
<code>sqrt(x)</code>	Квадратный корень
<code>ceil(x)</code>	Округление до ближайшего большего целого
<code>floor(x)</code>	Округление до ближайшего меньшего целого
<code>fabs(x)</code>	Возвращает абсолютное значение. тоже, что и <code>abs(x)</code>
<code>fmod(x, y)</code>	Остаток от деления. Тоже, что и $x \% y$.
<code>factorial(n)</code>	Факториал числа.
<code>fsum(<список чисел>)</code>	Возвращает точную сумму чисел из заданного списка. <code>>>>sum([.1, .1, .1, .1, .1, .1, .1, .1, .1])</code> <code>0.9999999999999999</code> <code>>>>fsum([.1, .1, .1, .1, .1, .1, .1, .1, .1])</code> <code>1.0</code>

ПРИЛОЖЕНИЕ 1

Символы, используемые при составлении блок-схем алгоритмов (ГОСТ 19.003-80)



При составлении программы в Word эти символы можно получить через меню:
"Вставка" – "Фигуры" – "Блок-схема".

Для добавления текста необходимо на выбранном объекте кликнуть правой кнопкой мыши, и в появившемся контекстном меню выбрать: "Добавить текст". Через пункт контекстного меню "Параметры фигуры..." можно настроить цвет, заливку, толщину линий...