МИНИСТЕРСТВО ОБРАЗОВАНИЯ СТАВРОПОЛЬСКОГО КРАЯ

Государственное бюджетное образовательное учреждение высшего образования «Ставропольский государственный педагогический институт»

Шаяхметов Олег Хазиакрамович

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ «PYTHON»

Методические указания для выполнения лабораторных работ по дисциплине «Программирование» для бакалавров, обучающихся по направлению подготовки 44.03.05 Педагогическое образование (с двумя профилями подготовки), профили «Математика» и «Информатика»

Ставрополь
2022

Содержание

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 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.

Часть лабораторных работ построена по принципу последовательного развития решения. Так, решения заданий лабораторной работы 2 использованы при выполнении лабораторных работ 3, 6 и 7 (циклы, файлы, графика), а результаты лабораторных работ 1, 4 и 5 использованы в лабораторной работе 6.

Лабораторные работы должны выполняться последовательно и в полном объёме.

В приложениях к пособию дан справочный материал по функциям модулей Math и Sys, а так же приведены обозначения, используемые для построения блок-схем алгоритмов.

Пособие предназначено для студентов, обучающихся по направлению Педагогическое образование (с двумя профилями подготовки), профили «Математика» и «Информатика», и может быть использовано как дополнительный независимый ресурс преподавателем при подготовке к лекциям, а так же при выполнении практических занятий и лабораторных работ.

Порядок выполнения работы

1. Внимательно прочитать и уяснить условие задачи, которую предстоит решить.

2. Ознакомиться с необходимым теоретическим материалом, используя литературу, указанную в пособии. В качестве основного источника теоретического и практического материала рекомендуется использовать книги [1] и [2] (см. список литературы).

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.

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'е 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 равноширинным шрифтом **Courier New** размером **10-:-12**.

Длинные инструкции можно записать на нескольких строках. Это можно сделать двумя способами:

- в конце строки пометить символ обратного слэш - \. За символом должен следовать символ перевода строки (Enter). Это устаревшая форма;
- поместить выражение в круглые скобки (рекомендуется) или, при определении списка или словаря – квадратные и фигурные скобки.

Подробности можно найти в примерах к лабораторным работам.

Содержание пояснительной записки

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 функции.
6. Файлы.
7. Программирование графики. Модуль turtle.
8. GUI, модуль Tkinter.

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** разделить 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. Объём электронной версии лабораторных работ в формате pdf составляет около 4-х Мб.

Студент, по согласованию с преподавателем, может выбрать другой свободный номер лабораторной работы.

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 заданным формулам. Предварительно подготовьте тестовые примеры с помощью калькулятора или электронной таблицы Excel.

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

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 может быть получен результат в виде комплексного числа или возникнет ошибка, которая приведёт к прерыванию работы программы.

В наших примерах знаменатели $1/2+\sin^2\alpha$ и $1+tg^2(2x/\pi)$ 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. Применить дополнительные математические функций можно после подключения модуля `math`:

```
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 модуля `math` 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,

```
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 на `a`, или `alpha`. Значение цвета можно хранить в переменной `Color`, а объема в переменной `Volume` или `Capacity`.

Обращайте внимание на цветовую раскраску переменной при наборе в редакторе IDLE. 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, введённое пользователем. Строковый тип может быть преобразован, например, к типу `int` или `float`, если введённое значение – число.

Для ввода нескольких значений можно воспользоваться методом `split()`, который позволяет разбить строку на подстроки (*split* – расщеплять).

Например, для ввода значений параметра α 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 вещественного числа целая часть отделяется от десятичной дроби точкой.

Если пользователь не ввел данные (просто нажал Enter) или вместо цифр и точки ввел недопустимые символы, например буквы, программа завершится аварийно на шаге приведения к типу – `float()`. 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 `ValueError` 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:
    a = x = ""
    try:
        a,x= (input('Введите [a: int, x: float]:')
              .split())
        a = int(a)
        x = float(x)
        break
    except ValueError:
        if a == "" and x == "":
            a = x = 0
            continue
        print("a - int, x - float. Пример: 3 4.5")
print(a, x,)
```

В теле "вечного" цикла `while True:`, 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 не определена.

Вывод данных

Вывод данных на экран монитора может быть выполнен функцией print(). Эта функция позволяет выполнять форматированный вывод, как с использованием Си-подобного форматирования, так и с использованием форматной строки Python.

Следующие строки демонстрируют, как можно форматировать вывод.

```
for x in range(1,11):
    print('%2d %3d %7.2f' % (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.

Во второй строке использован Си-подобный формат, в котором формат числа начинается с процента "%". В этом формате аргументы отделяются от форматной части строки так же символом % – процент. В третьей строке используется форматная строка Python, в которой в форматной строке позиции для значений аргументов выделяются фигурными скобками.

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 слова:

```
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 делает 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. Первое выражение примет вид:

```
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 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 офисного приложения, например Excel или Calc. Результаты вычисления по обеим формулам должны совпадать. Отсутствующие в языке функции выразите через имеющиеся.

1. $z_1 = 2\sin^2(3\pi - 2\alpha)\cdot\cos^2(5\pi + 2\alpha)$; $z_2 = \frac{1}{4} - \frac{1}{4}\sin(\frac{5}{2}\pi - 8\alpha)$.

2. $z_1 = \cos\alpha + \sin\alpha + \cos 3\alpha + \sin 3\alpha$; $z_2 = 2\sqrt{2}\cos\alpha\cdot\sin(\frac{\pi}{4} + 2\alpha)$.

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

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

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

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

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

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

8. $z_1 = \cos^4 x + \sin^2 y + \frac{1}{4}\sin^2 2x - 1$; $z_2 = \sin(y+x)\cdot\sin(y-x)$.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$z_2 = 1 - 2c + a$.

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

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

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

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

28. $z_1 = \dfrac{\sin^2 a - \mathrm{tg}^2 a}{\cos^2 a - \mathrm{ctg}^2 a}$; $z_2 = \mathrm{tg}^6 a$;

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

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

Лабораторная работа №2: «Разветвляющиеся вычислительные процессы», Задание 1

Цель работы:

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

Следующий материал представлен в виде оформленной лабораторной работы.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ СТАВРОПОЛЬСКОГО КРАЯ

Государственное бюджетное образовательное учреждение высшего образования «Ставропольский государственный педагогический институт»

Кафедра математики, информатики и цифровых образовательных технологий

Лабораторная работа №2

Разветвляющиеся вычислительные процессы

Задание 1 **Вариант № 1**

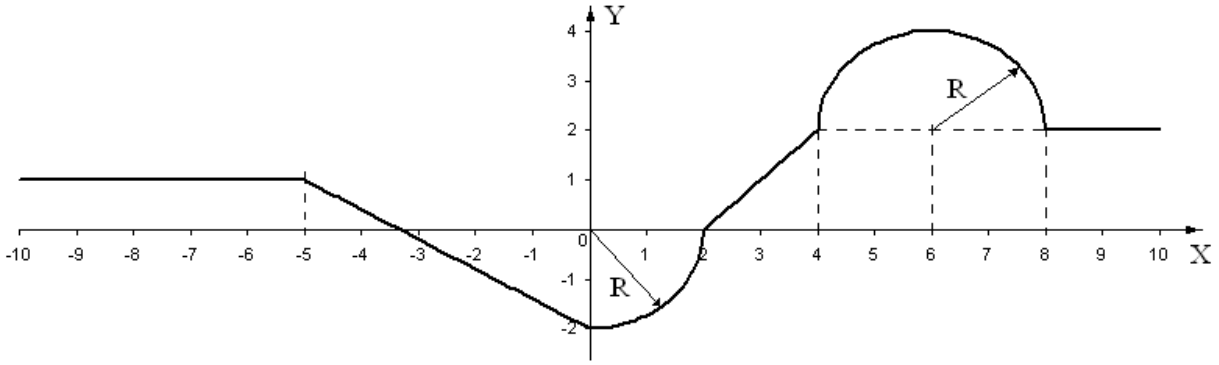
по дисциплине:
«Программирование»

Выполнил
Студент __ курса
группа_____
Иванов И.И.

Оценка_____
_____ О.Х. Шаяхметов
"____" _____202__ год

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 вложенных инструкций, которые выделяются одинаковым количеством пробелов (обычно четырьмя).

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

Вывод данных выполняется инструкцией `print()`, в которой использован форматированный вывод данных.

График функции представлен фрагментами прямых линий, описываемых уравнением $y = kx + b$ и дугами кругов. В общем случае уравнение круга может быть представлено так: $(x-a)^2 + (y-b)^2 = R^2$.

Неизвестные параметры, угол наклона и смещение прямой, а так же координаты центра дуг, определим, используя данные из графика.

Для прямой на интервале (-5, 0) можем записать следующую систему уравнений:

$$\begin{cases} 1 = k \cdot (-5) + b \\ -2 = k \cdot 0 + b \end{cases}$$

Из второго уравнения следует, что b = -2, а из первого – k = -3/5.

Для полукруга с центром (6, 2) уравнение круга примет вид:

$$(x-6)^2+(y-2)^2=2^2$$

Перепишем уравнение так:

$$(y-2)^2=2^2-(x-6)^2$$

$(y-2)^2 = 4-(x-6)^2$ отсюда следует, что $y=2+\sqrt{4-(x-6)^2}$. Знак перед корнем выбран для случая, когда рассматривается верхняя часть полукруга.

Выполнив необходимые вычисления для всех фрагментов функции, мы получим систему уравнений, которую запишем в следующем виде:

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

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 (простая комбинация условного оператора)

```
# -*- 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 вычисленным. 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 (усложненная комбинация условного оператора)

```
# -*- 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

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

ПРИЛОЖЕНИЕ 1
к лабораторной работе №2, задание 1
И.И. Иванов

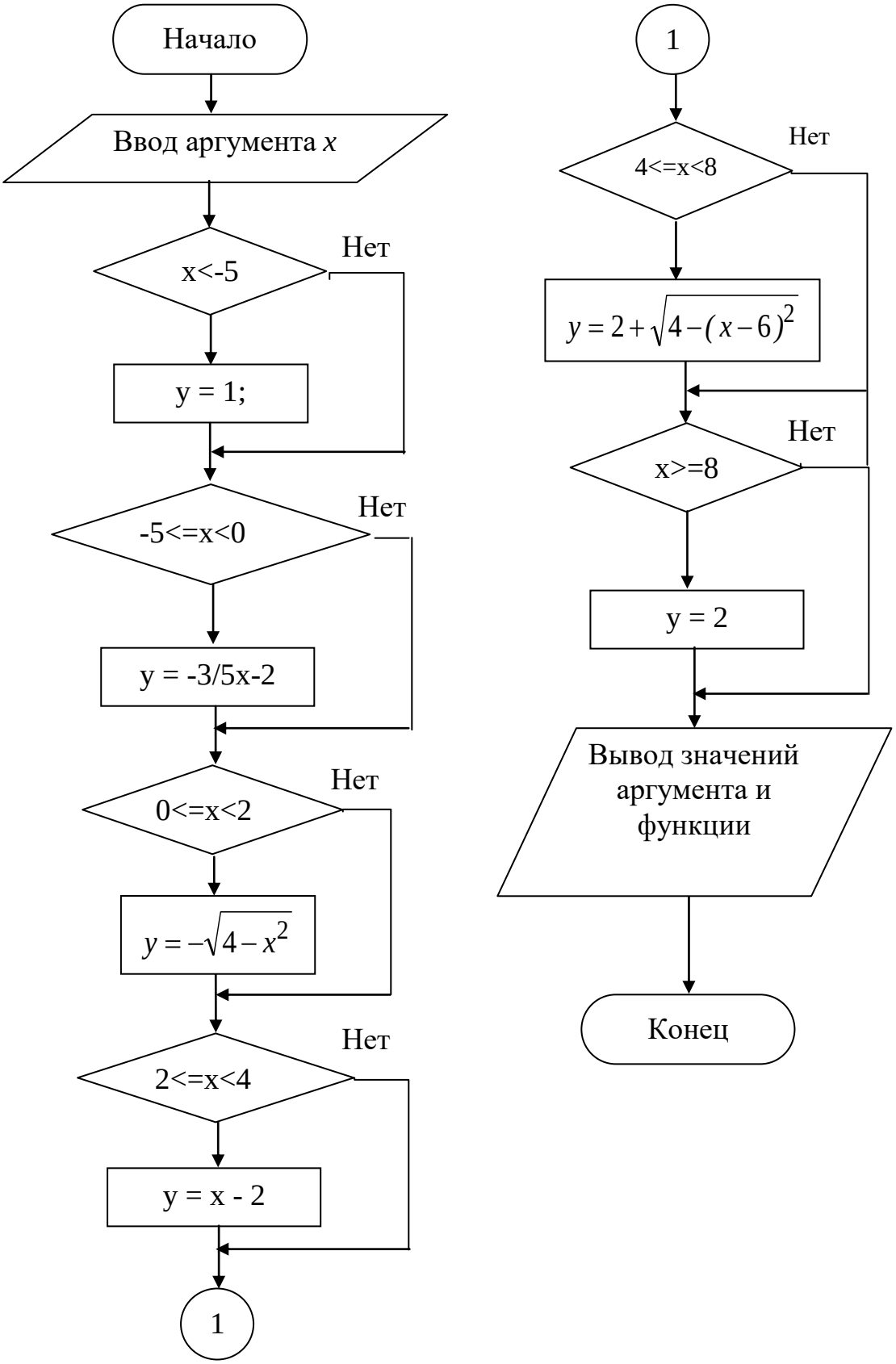


Рис.1 – Блок-схема алгоритма программы (первая часть)

ПРИЛОЖЕНИЕ 2
к лабораторной работе №2, задание 1
И.И. Иванов

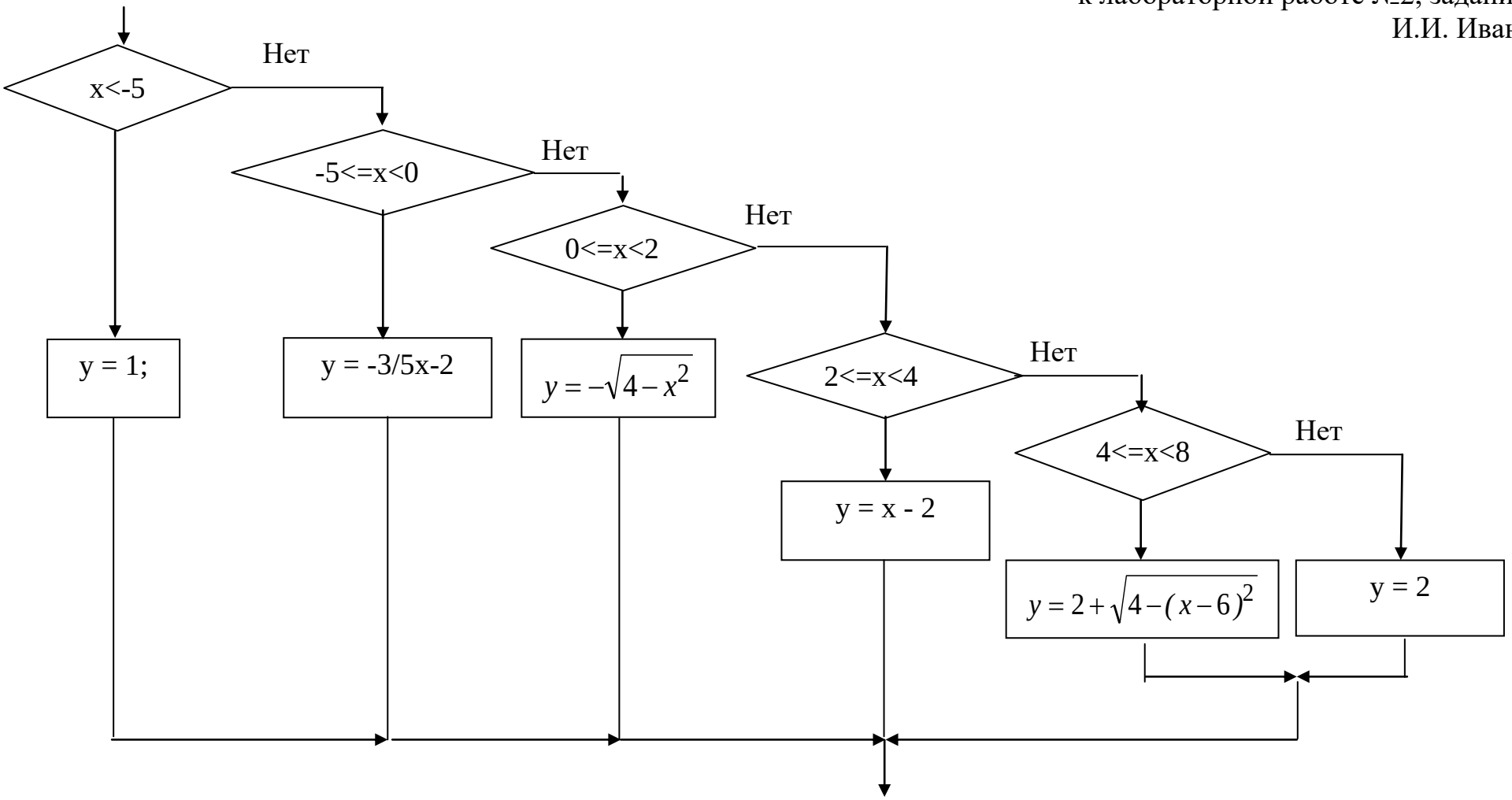


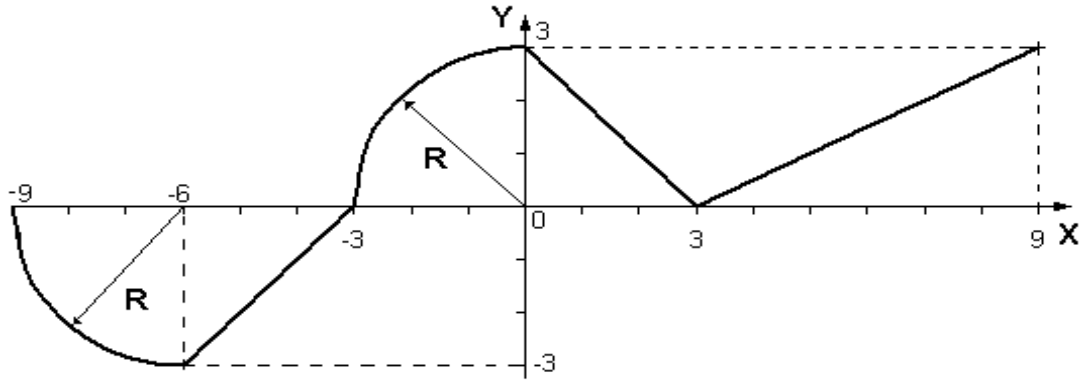
Рис.2 – Блок-схема алгоритма программы (вторая часть: показана только логическая часть)

Задание к лабораторной работе №2 «Разветвляющиеся вычислительные процессы».

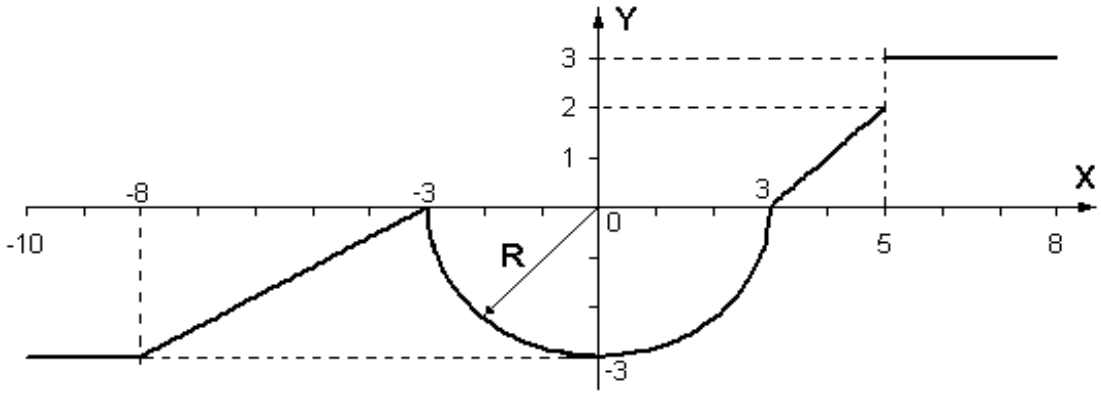
Задание 1

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, согласно Вашего варианта.

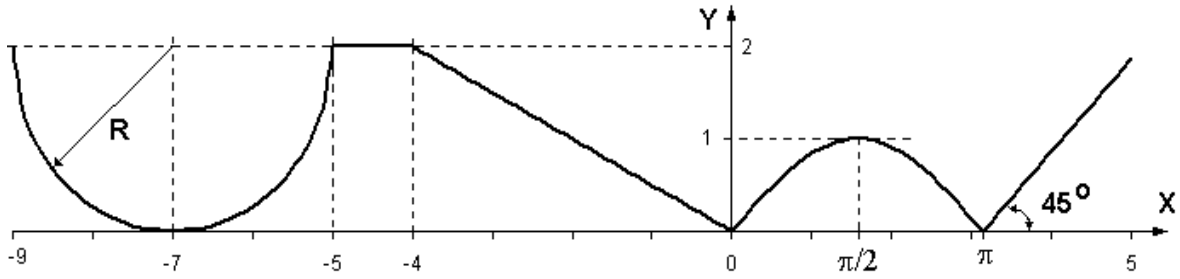
1)



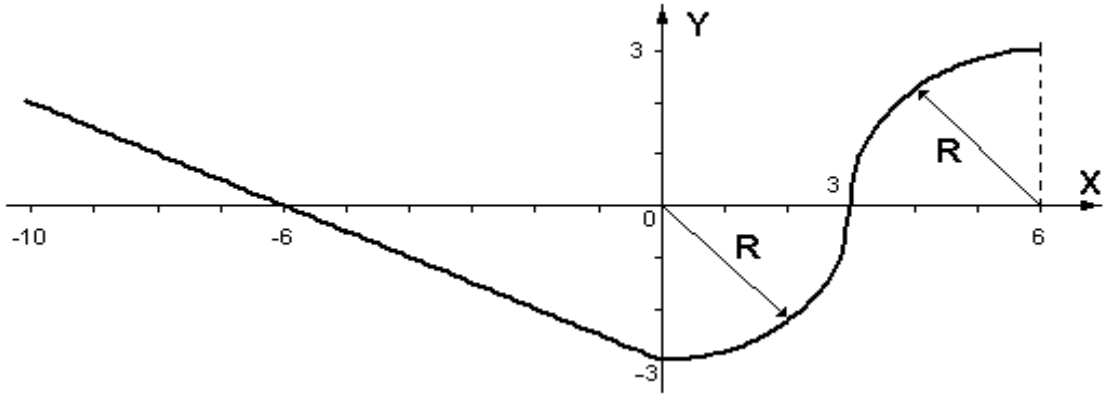
2)



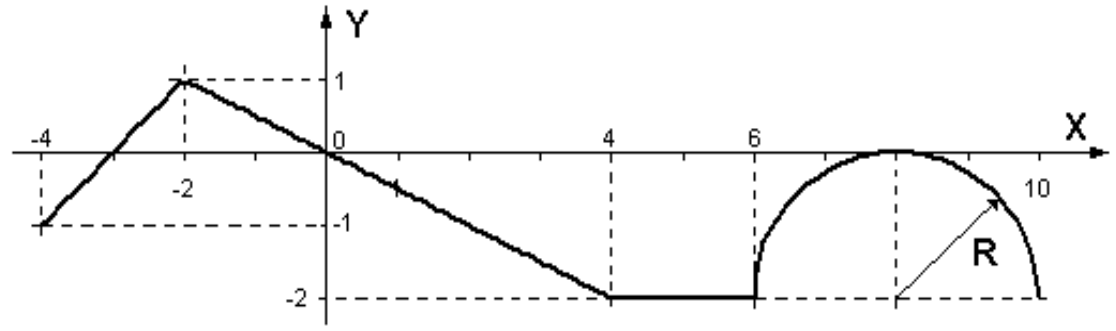
3)



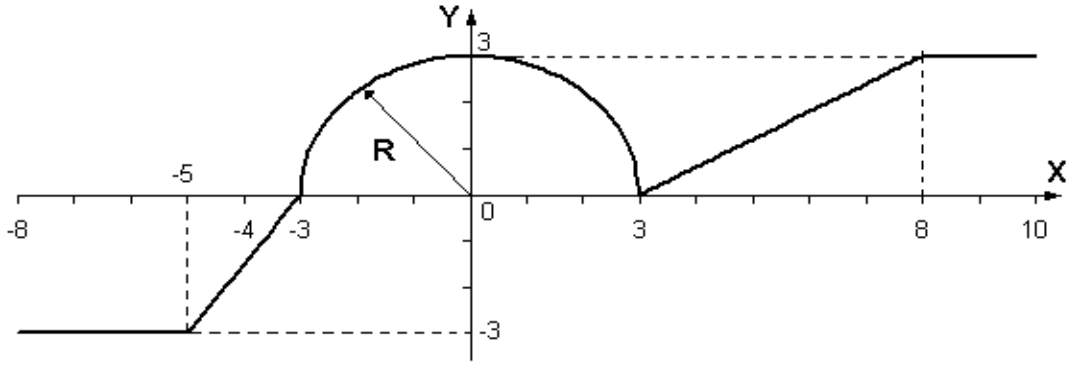
4)



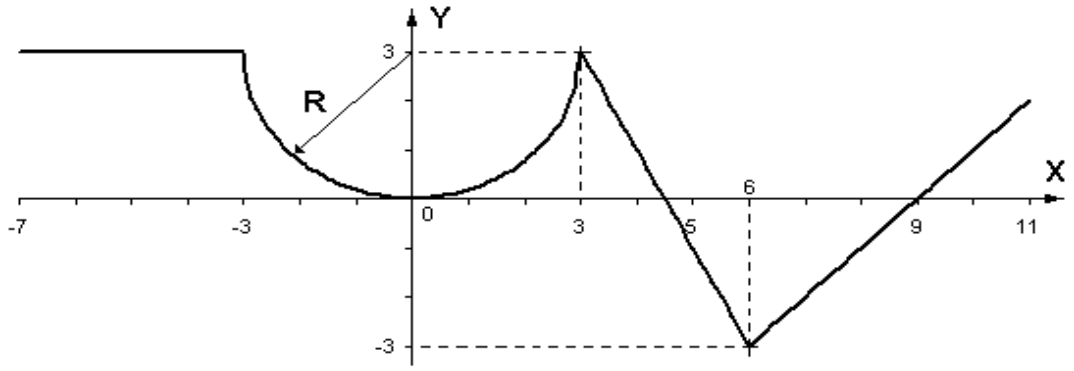
5)


Y
X
1
-1
-2
-4
-2
0
4
6
10
R

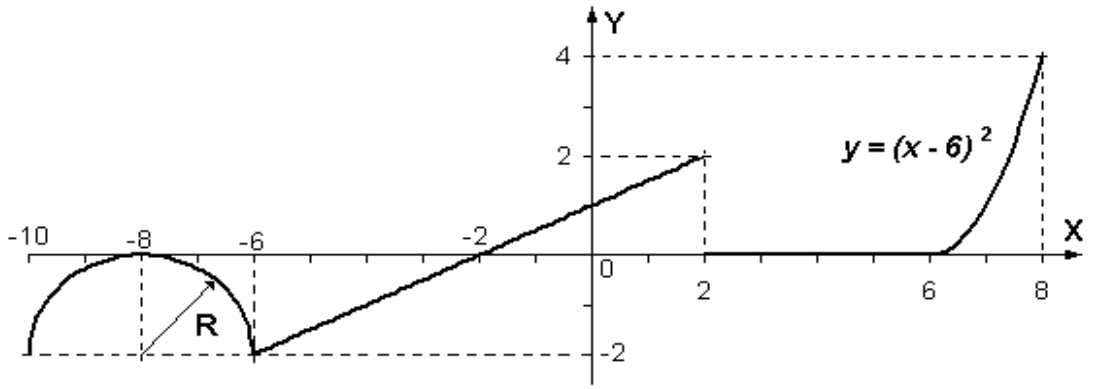
6)


Y
X
3
-3
-5
-8
-4
-3
0
3
8
10
R

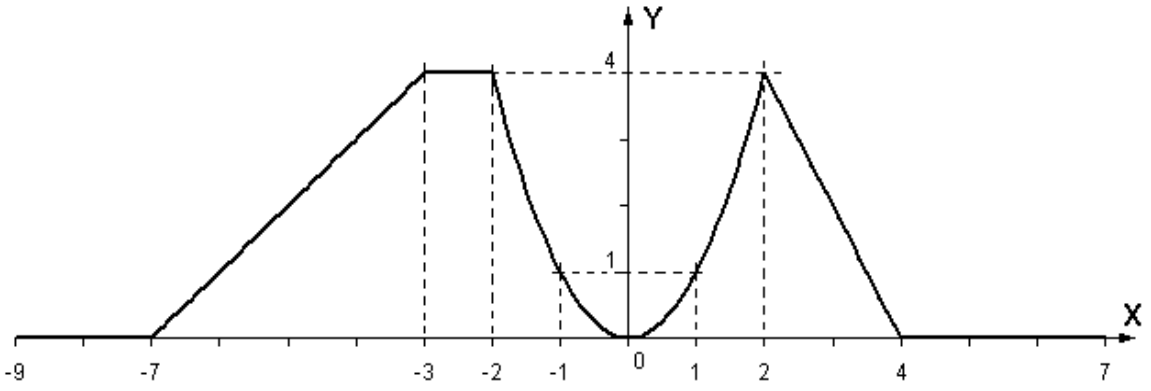
7)


Y
X
3
-3
R
-7
-3
0
3
5
6
9
11

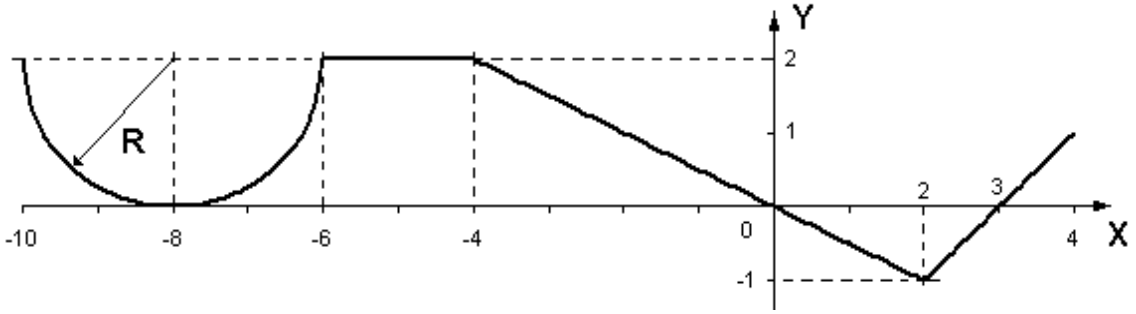
8)


Y
X
4
2
-2
y = (x - 6) ²
-10
-8
-6
-2
0
2
6
8
R

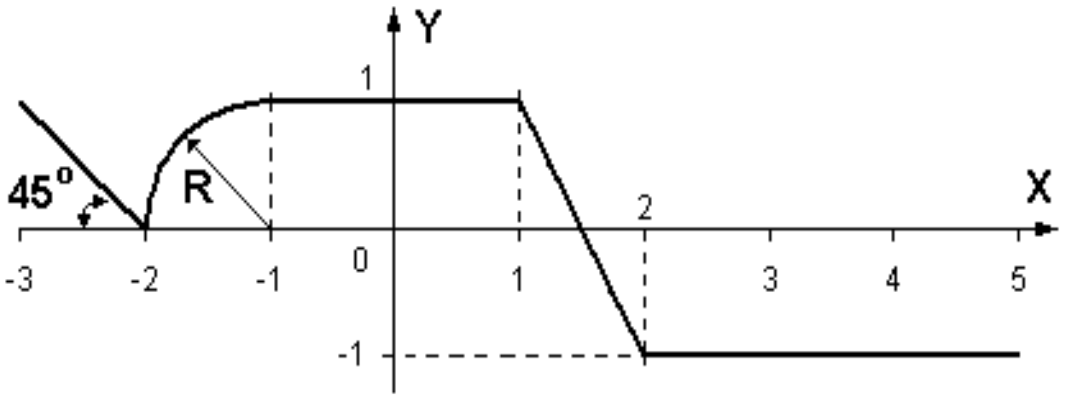
9)


Y
X
4
1
0
-9
-7
-3
-2
-1
1
2
4
7

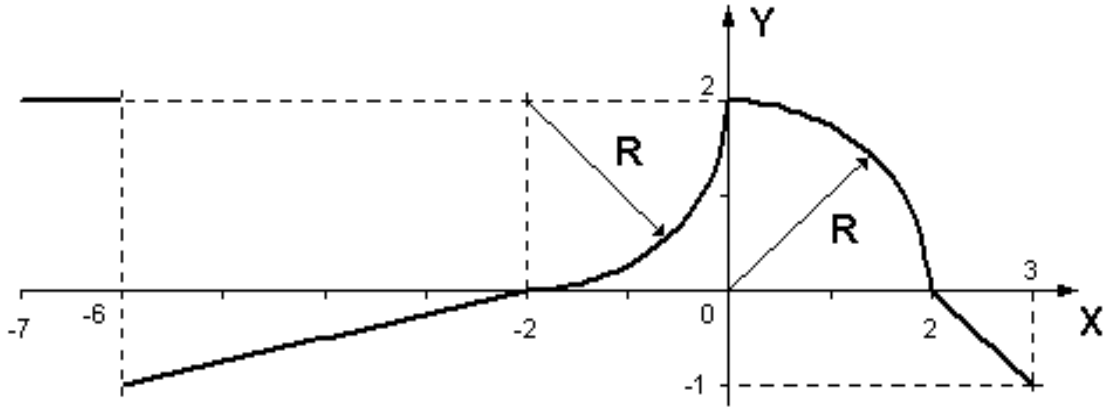
10)


Y
X
R
2
1
-1
-10
-8
-6
-4
0
2
3
4

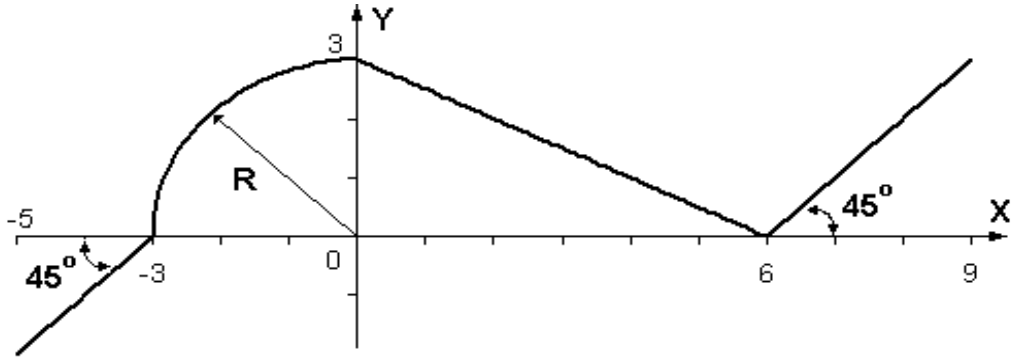
11)


Y
X
45°
R
1
0
-1
-3
-2
-1
1
2
3
4
5

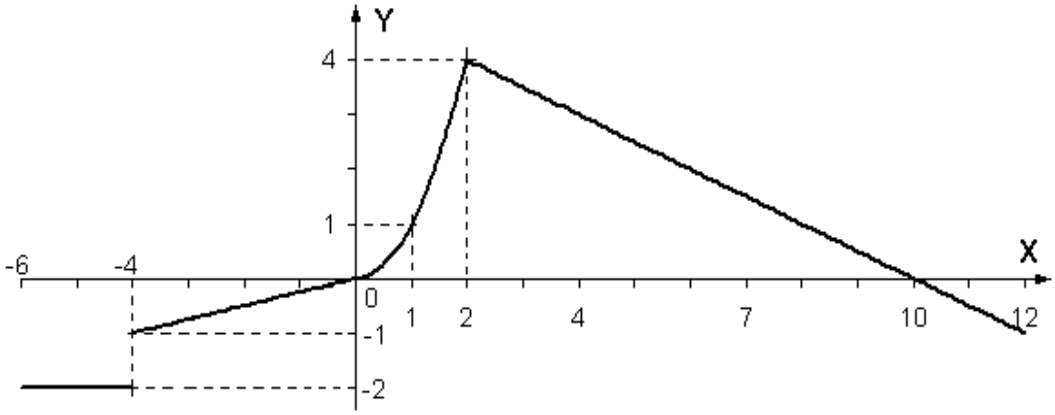
12)


Y
X
R
R
2
0
-1
-7
-6
-2
2
3

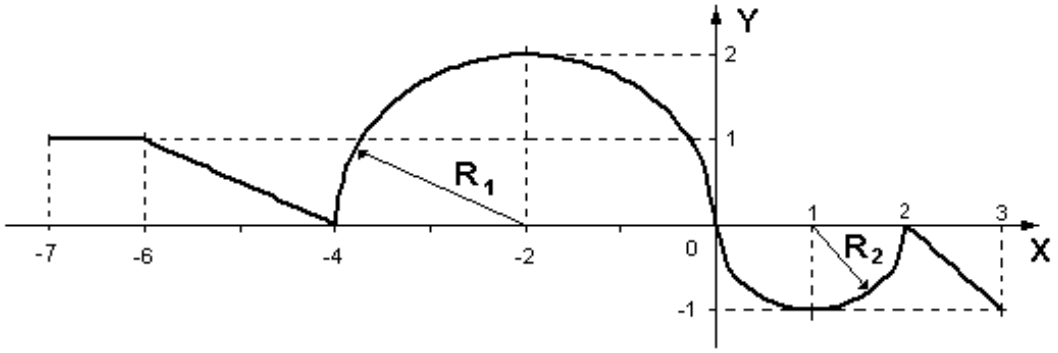
13)


Y
X
3
R
-5
45°
-3
0
6
45°
9

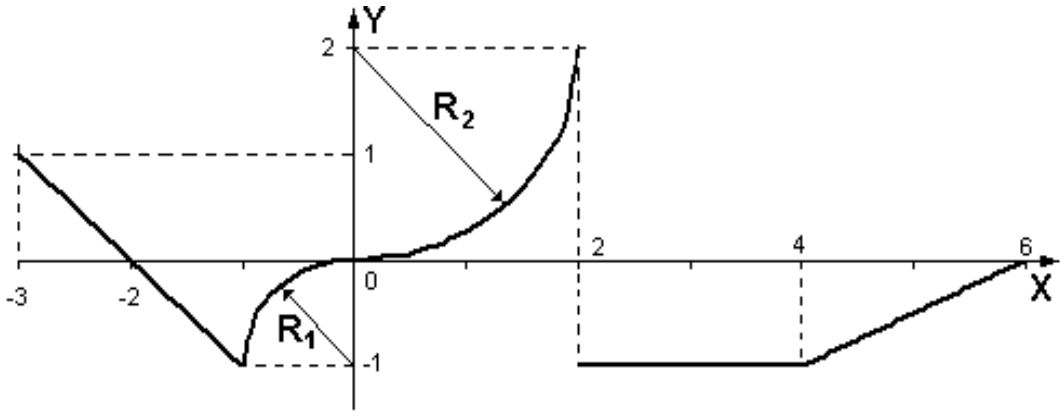
14)


Y
X
4
1
-6
-4
0
1
2
4
7
10
12
-1
-2

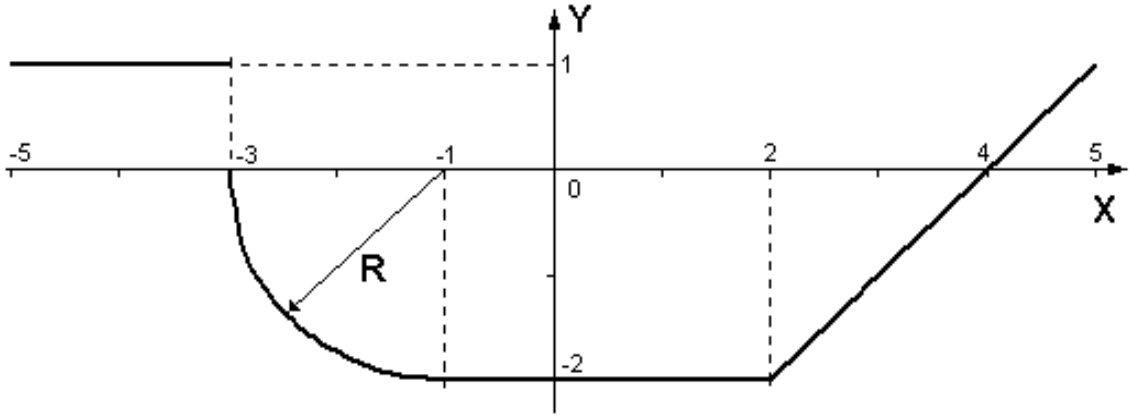
15)


Y
X
2
1
R1
-7
-6
-4
-2
0
1
2
3
R2
-1

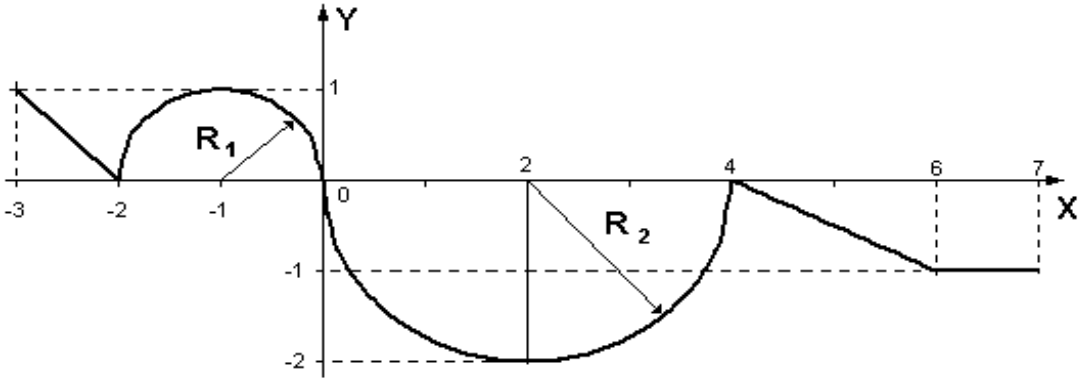
16)


Y
X
2
R2
1
2
4
6
-3
-2
0
R1
-1

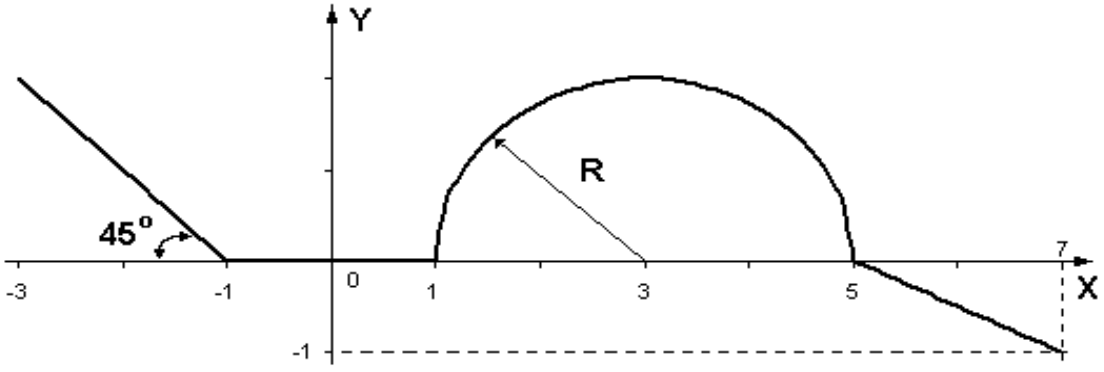
17)


Y
X
1
0
-2
-5
-3
-1
2
4
5
R

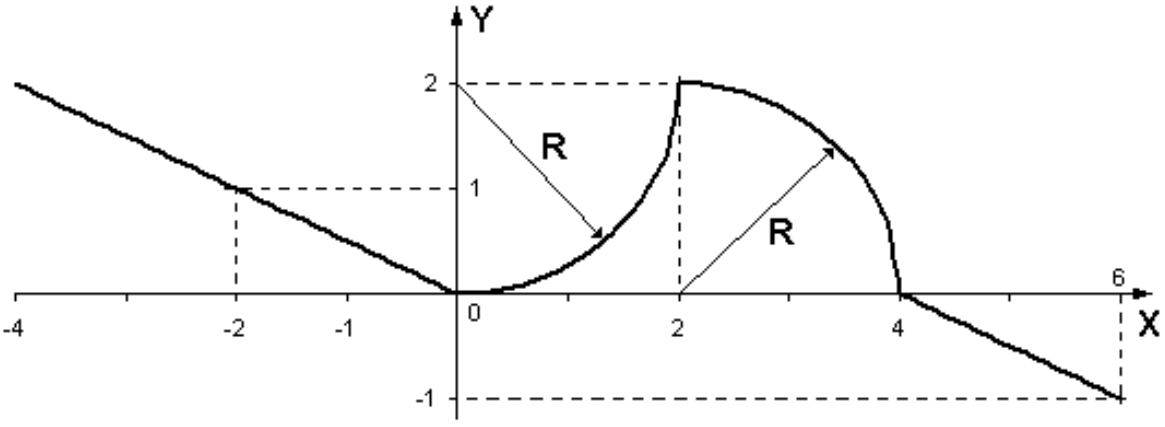
18)


Y
X
1
0
-1
-2
-3
-2
-1
2
4
6
7
R1
R2

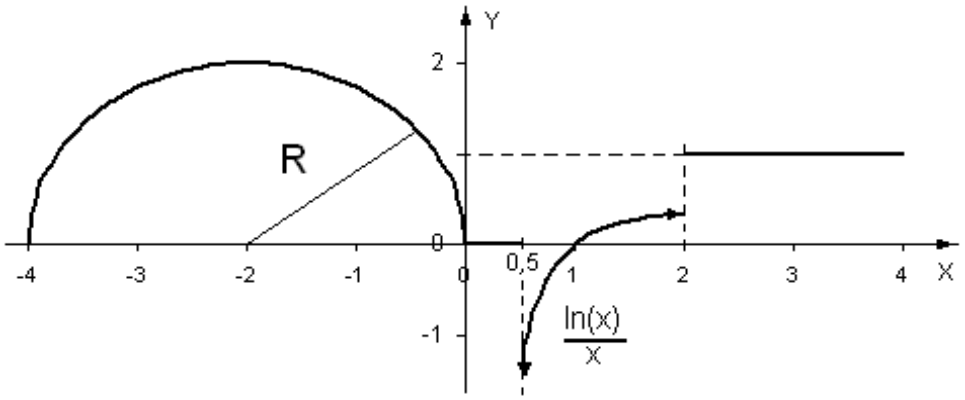
19)


Y
X
0
-1
-3
-1
1
3
5
7
45°
R

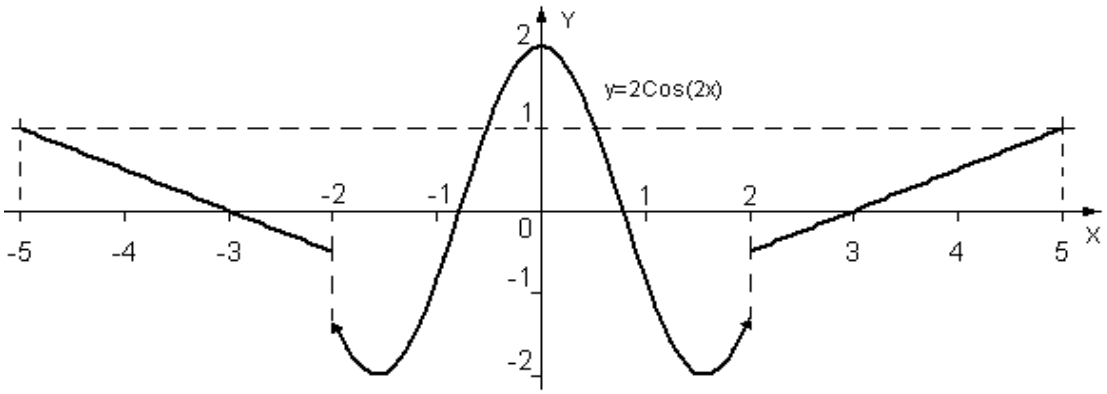
20)


Y
X
2
1
0
-1
-4
-2
-1
2
4
6
R
R

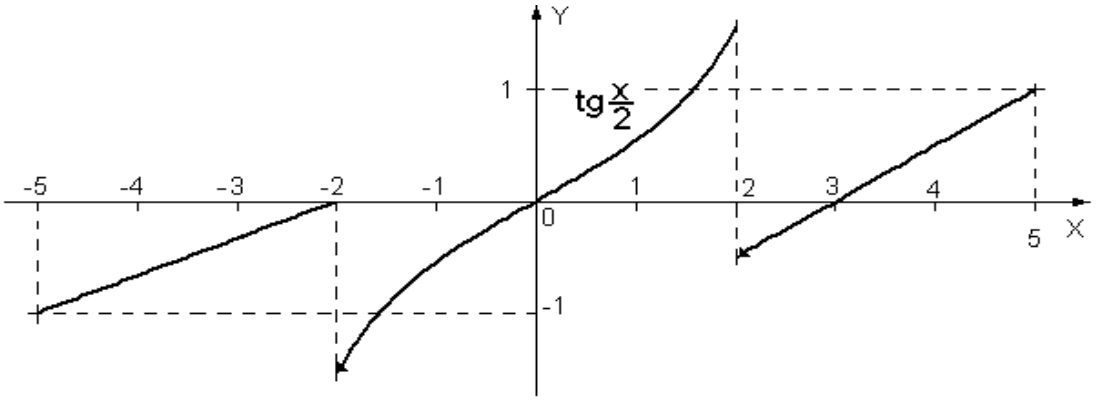
21)


Y
2
R
0
-4
-3
-2
-1
0
0,5
1
2
3
4
X
-1
ln(x)
x

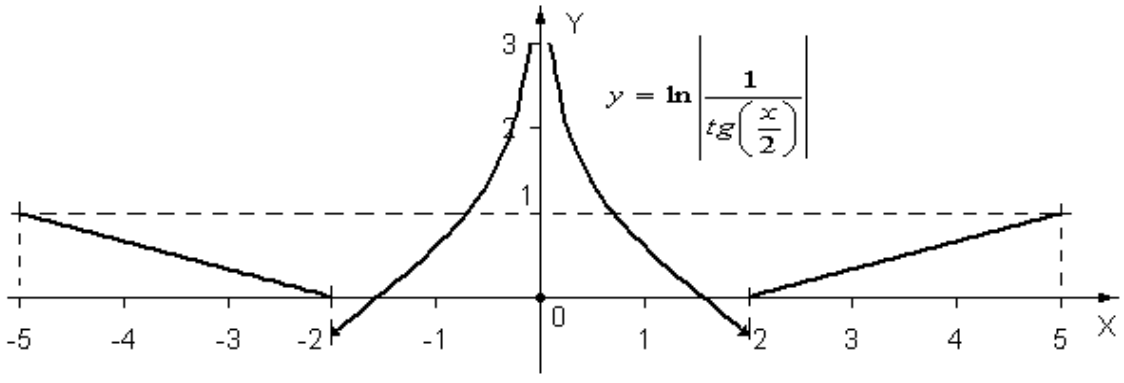
22)


Y
2
y=2Cos(2x)
1
-2
-1
1
2
-5
-4
-3
0
-1
-2
3
4
5
X

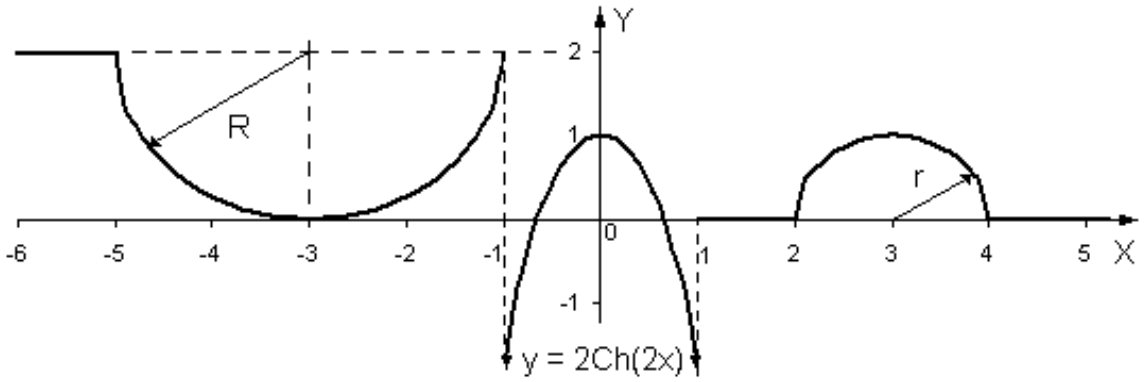
23)


Y
1
tg x/2
-5
-4
-3
-2
-1
1
2
3
4
0
5
X
-1

24)

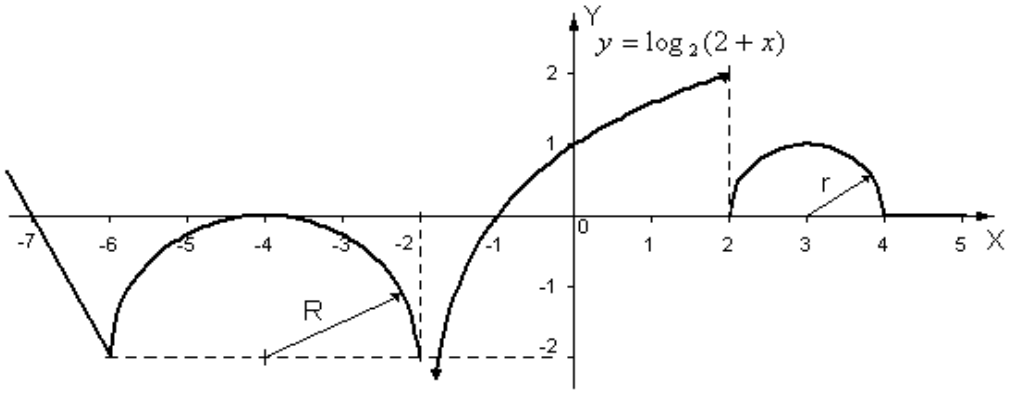

Y
3
y = ln |1 / tg(x/2)|
2
1
0
-5
-4
-3
-2
-1
1
2
3
4
5
X

25)

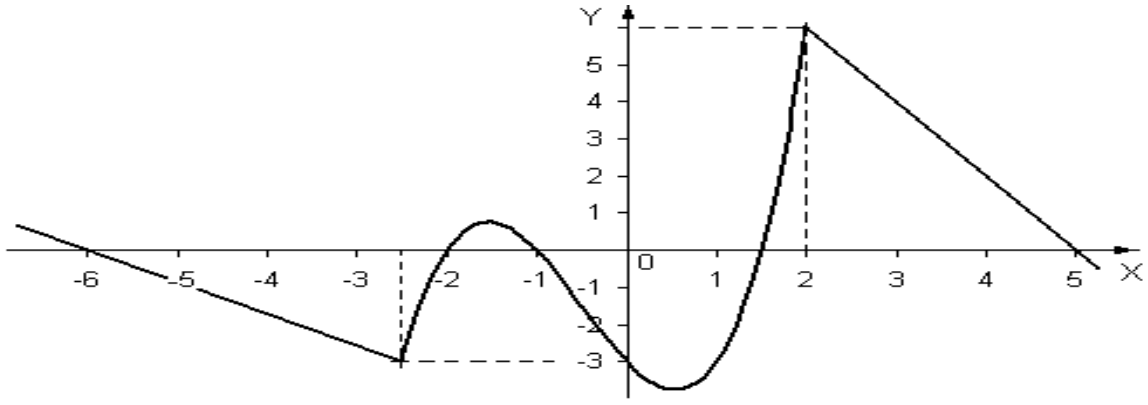


Гиперболический косинус может быть описан формулой: $Ch(x) = \frac{1}{2}\left(e^{x} + e^{-x}\right)$.

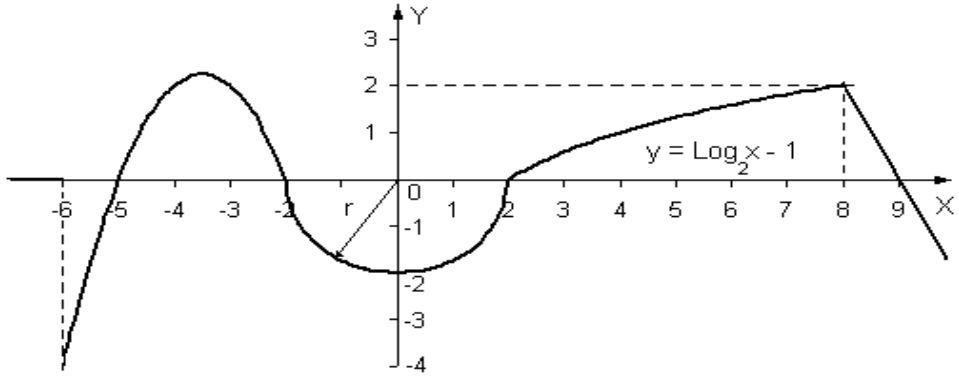
26)



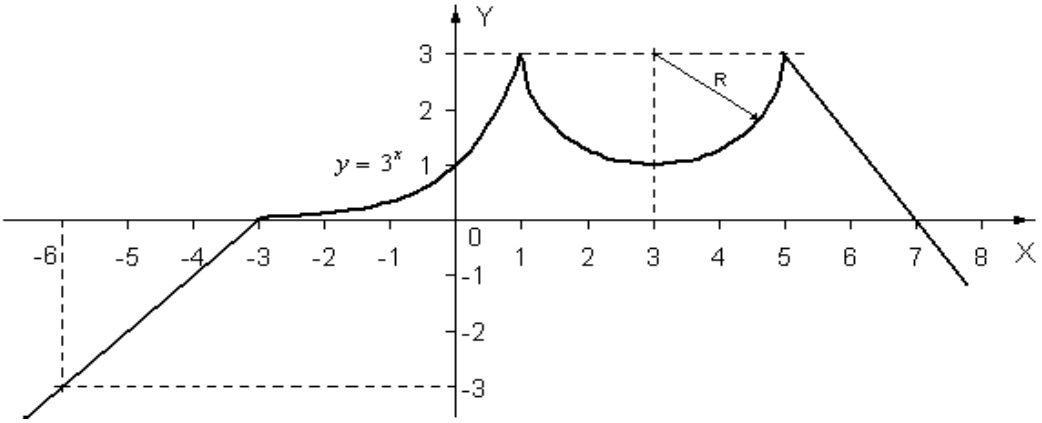
27)



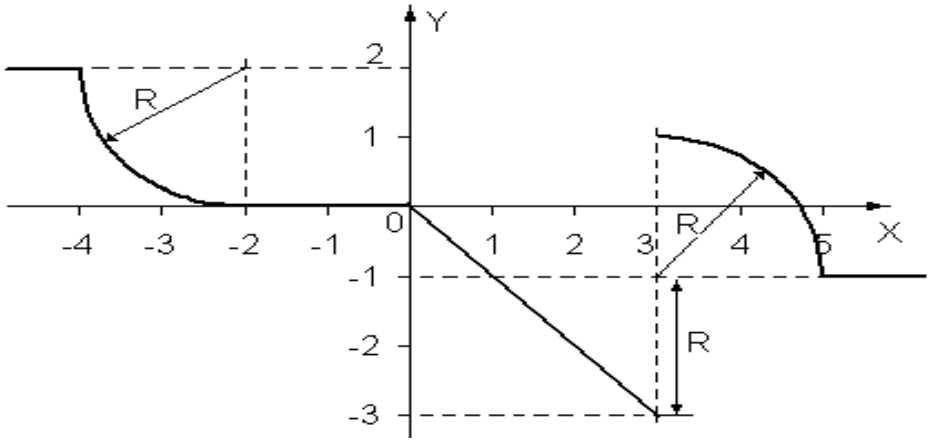
28)



29)



30)

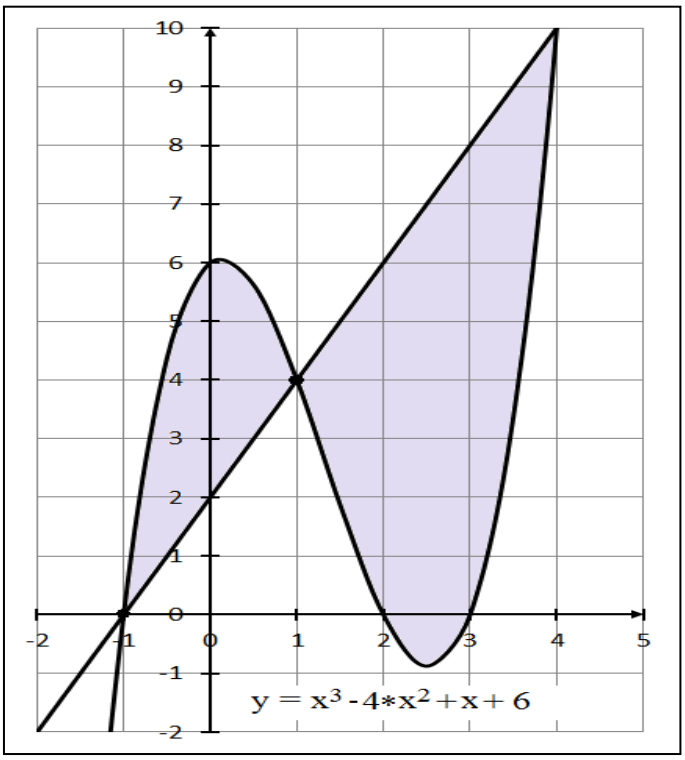


Дополнительное требование к заданию №30: Программа должна быть написана так, что бы решения получались при различных значениях `R`, вводимых с клавиатуры. Центр положения левой четверти окружности изменяется в соответствии с введённым радиусом, а правой – остаётся постоянным при `X = 3`, `Y = -1`.

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).

Необходимо 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 знать уравнение прямой (уравнение кривой приведено на рисунке). Из рисунка получаем координаты двух точек, через которые проходит прямая линия (-1, 0) и (1,4). Подставив значения выбранных точек в уравнение общего вида `y = kx + b`, получим систему уравнений. Решив эту систему, найдём значения для параметров k и b. В итоге уравнение прямой примет вид: `y = 2x + 2`.

Точка с координатами (`x, y`), введённая пользователем, будет попадать в заштрихованную область на интервале [-1, 1] в том случае, если `x` будет не меньше -1 и не больше 1. Кроме этого, `y` должен быть не меньше значения полученного для прямой в точке `x` и не больше значения, вычисленного для кубической параболы в этой точке.

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, введённые 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 X и Y для 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", end=" ")
else:
    print("не попадает", end=" ")
print("в область.")
```

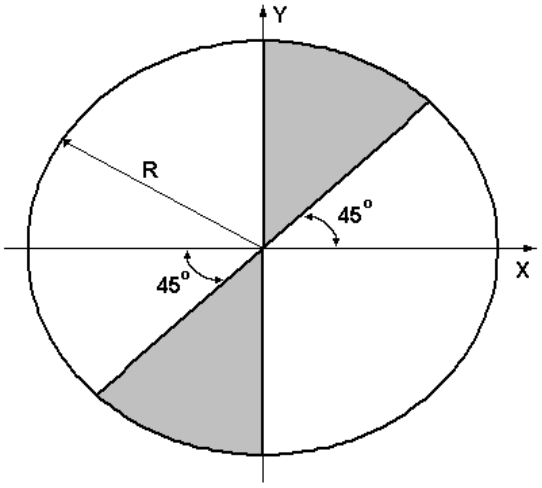
Замечание к тексту программы: Для переноса длинных строк в Python использованы круглые скобки.

Задание к лабораторной работе №2 «Разветвляющиеся вычислительные процессы».

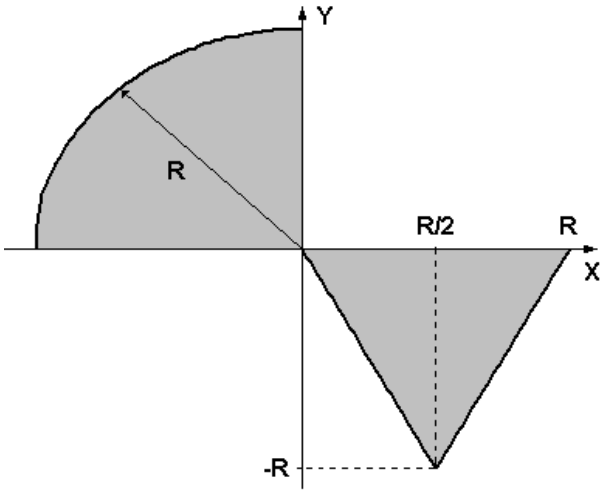
Задание 2

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. Выберите свой вариант для решения.

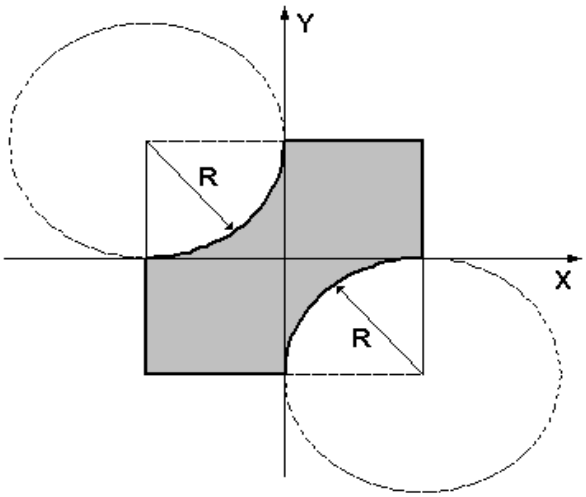
1)



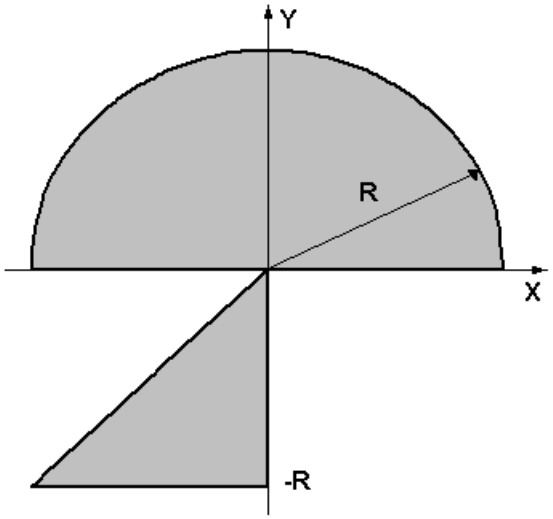
2)



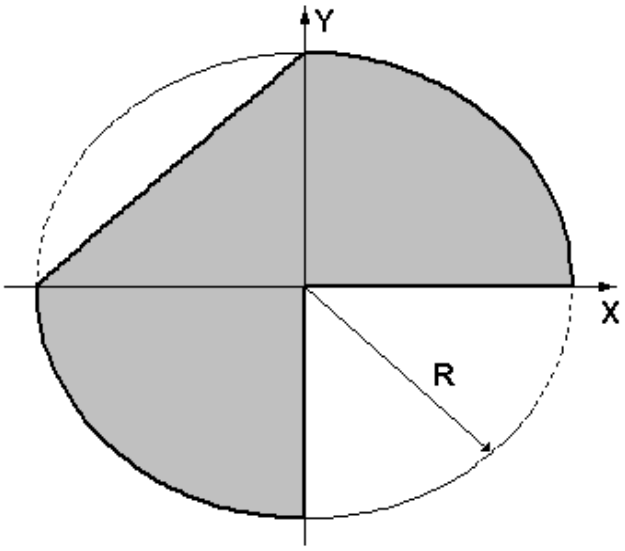
3)



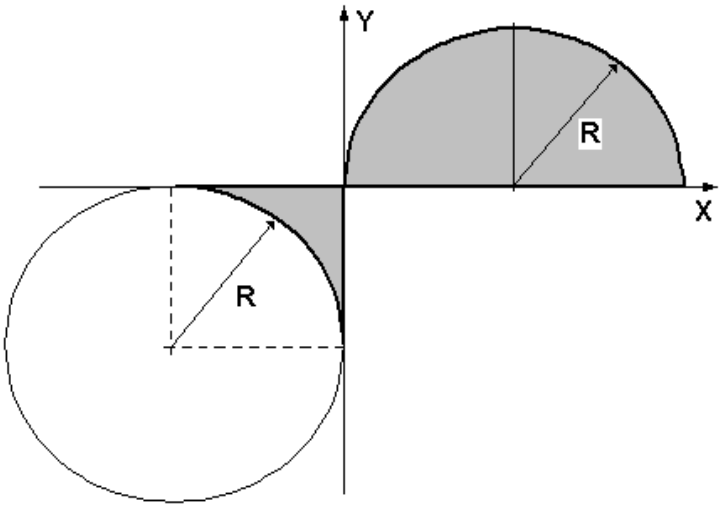
4)



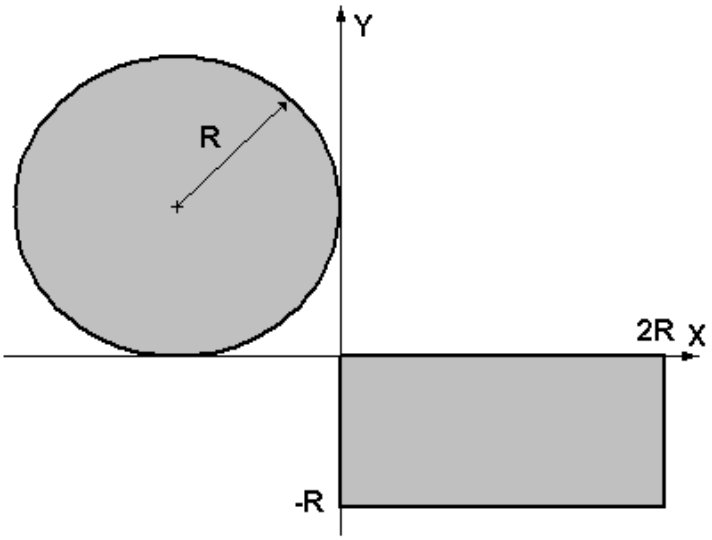
5)



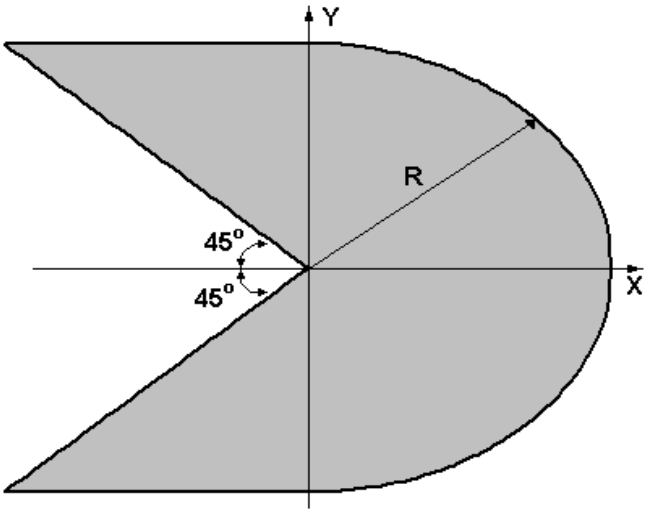
6)



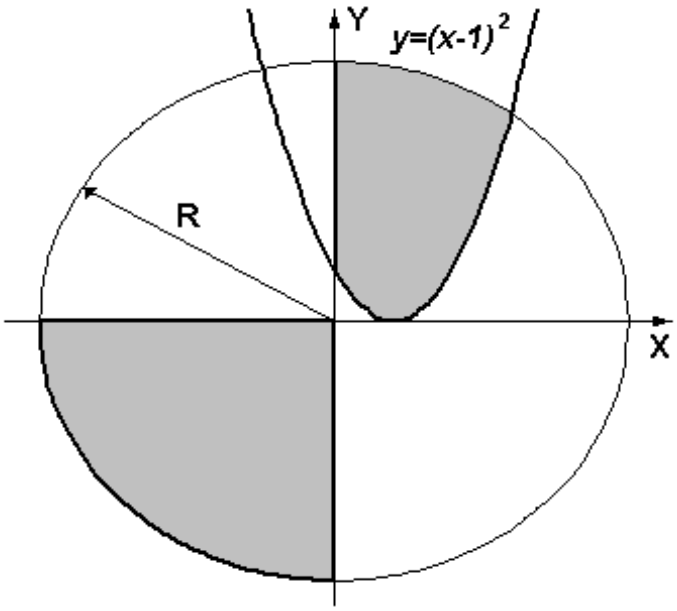
7)



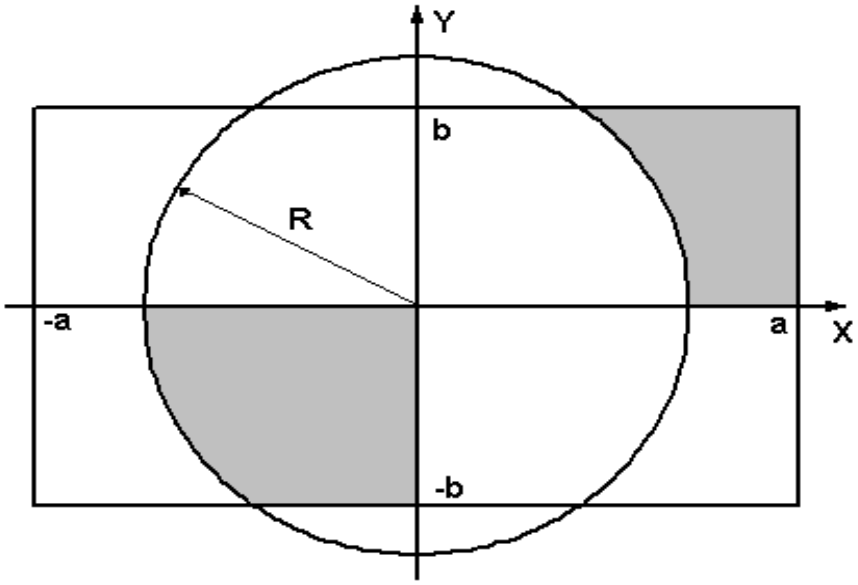
8)



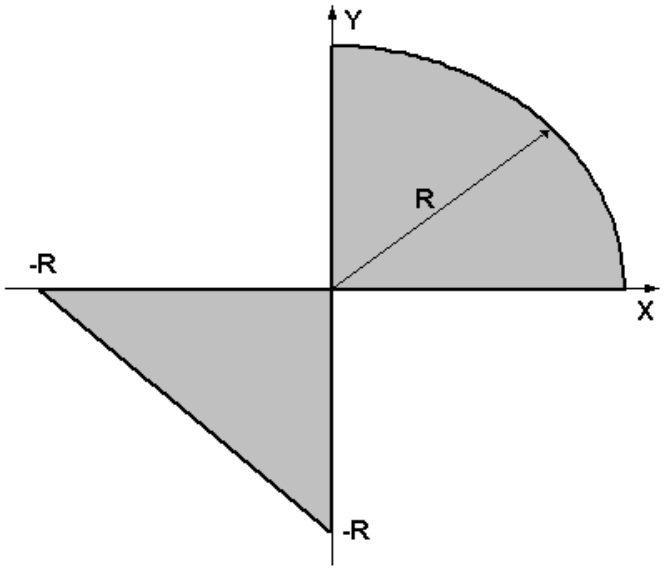
9)



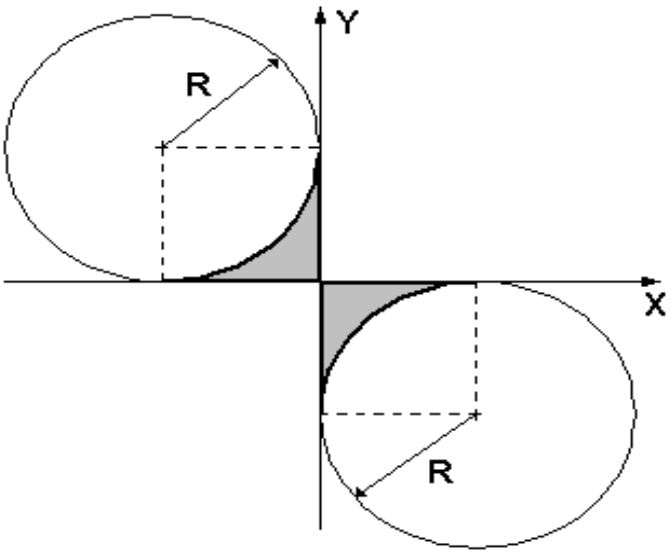
10)



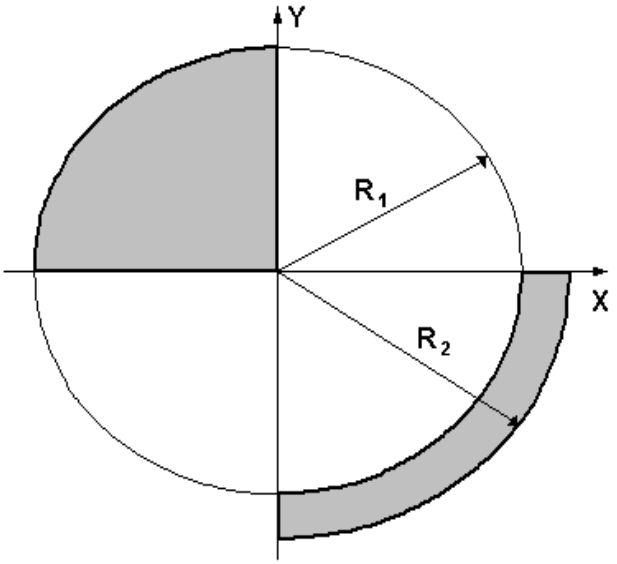
11)



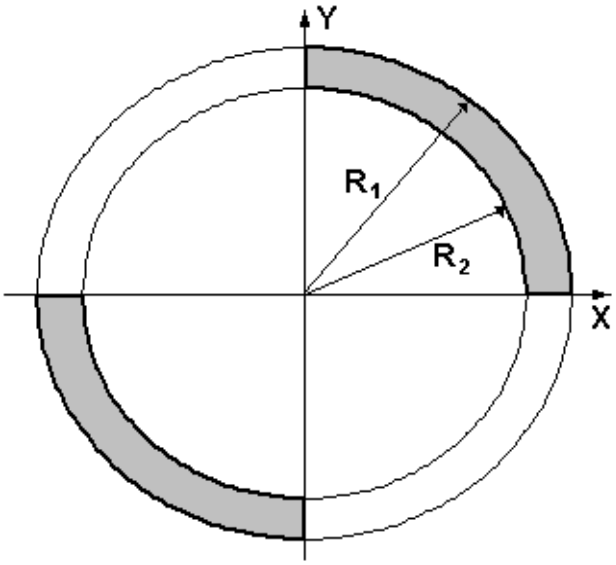
12)



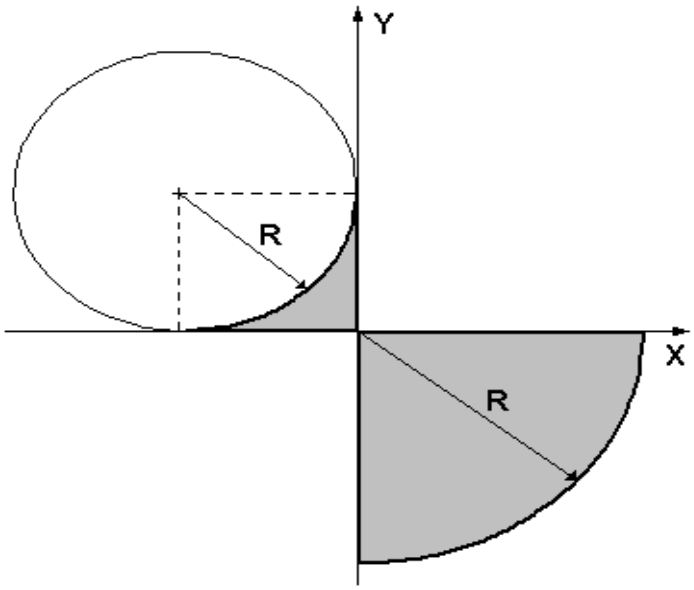
13)



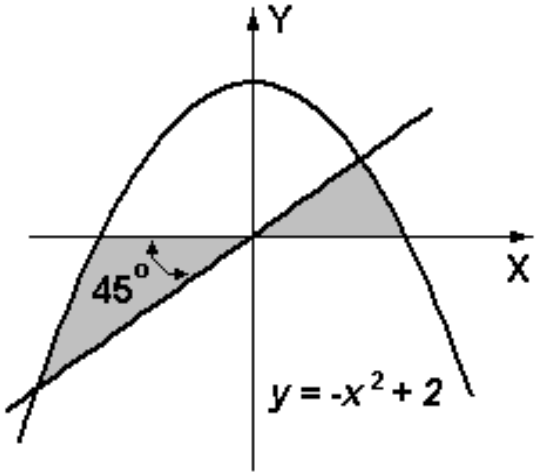
14)



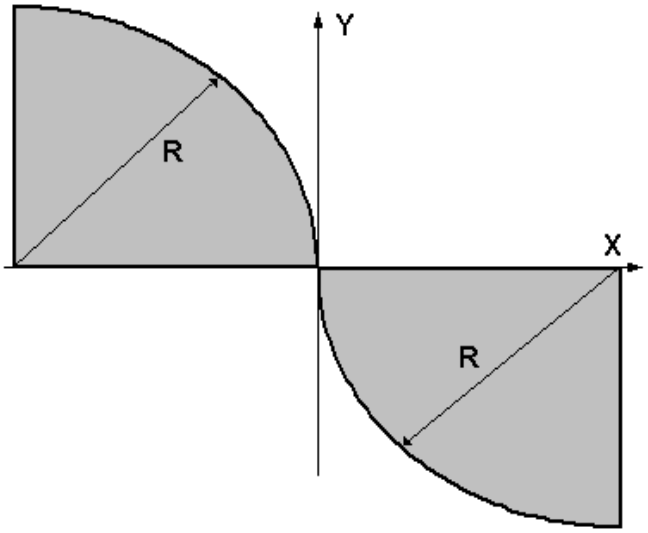
15)



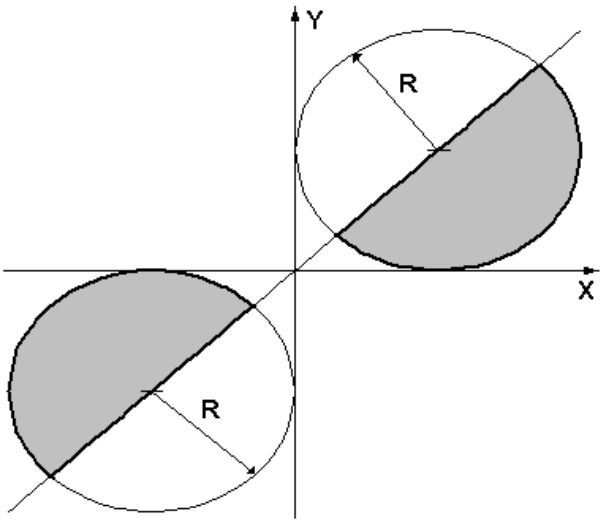
16)



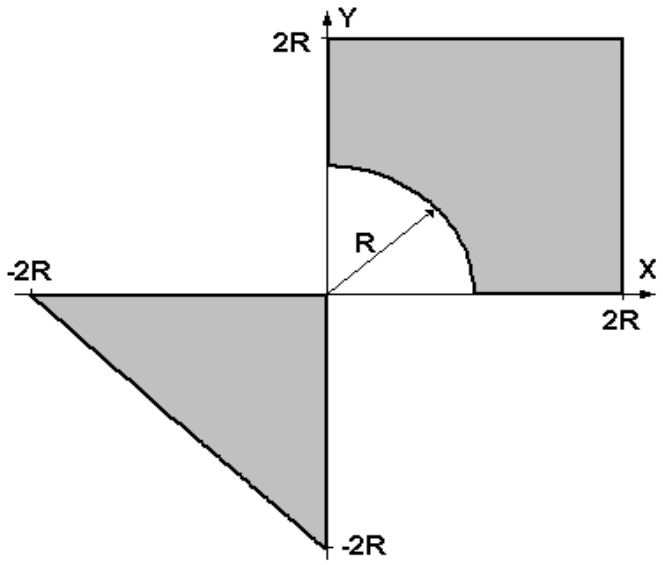
17)



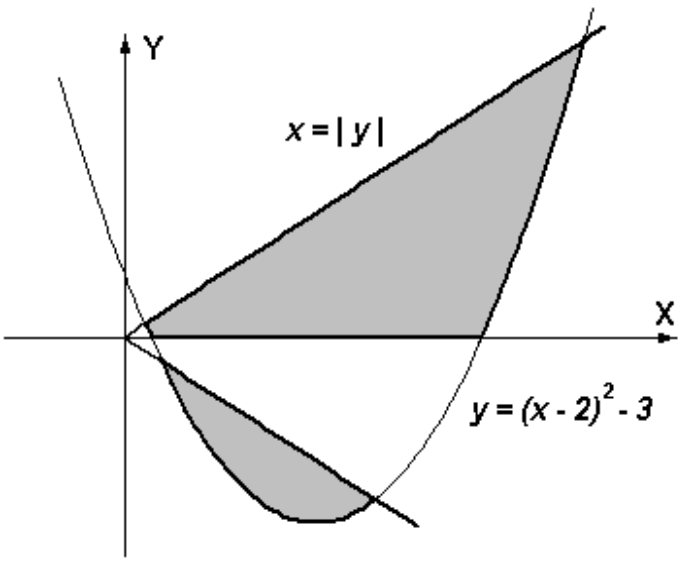
18)



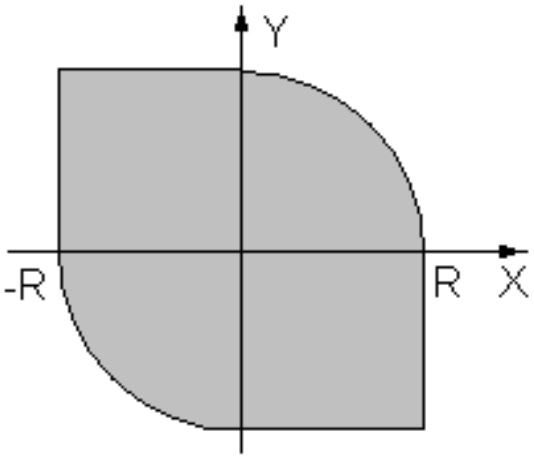
19)



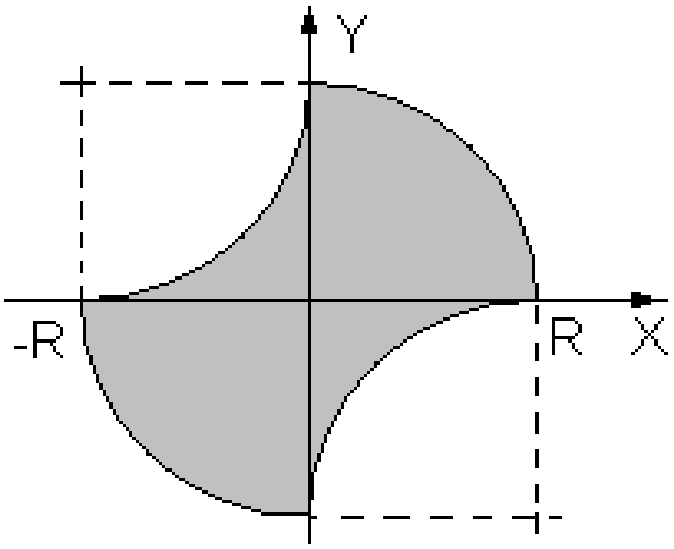
20)



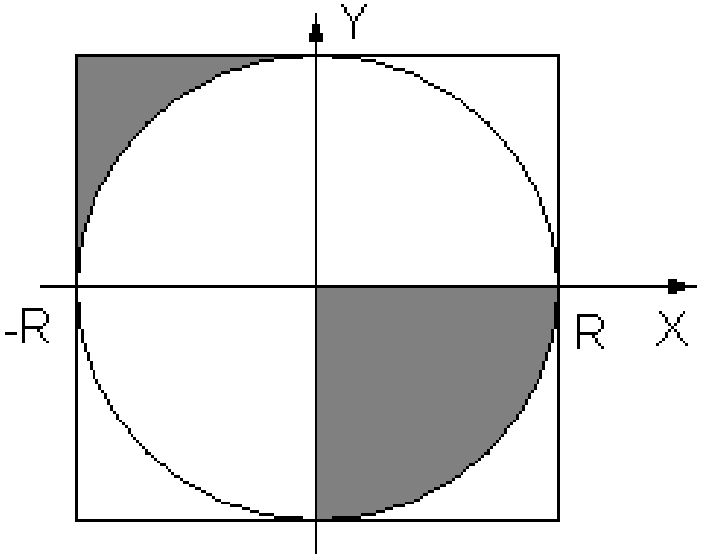
21)



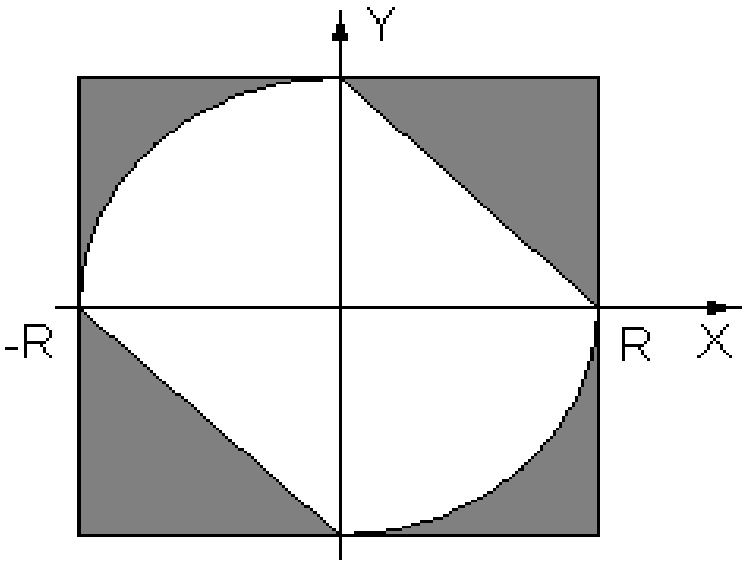
22)



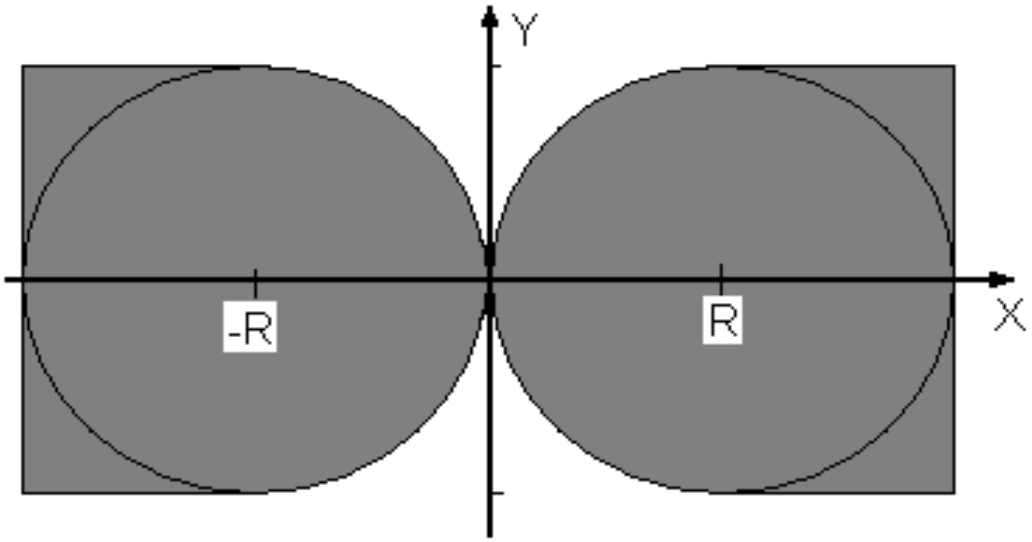
23)



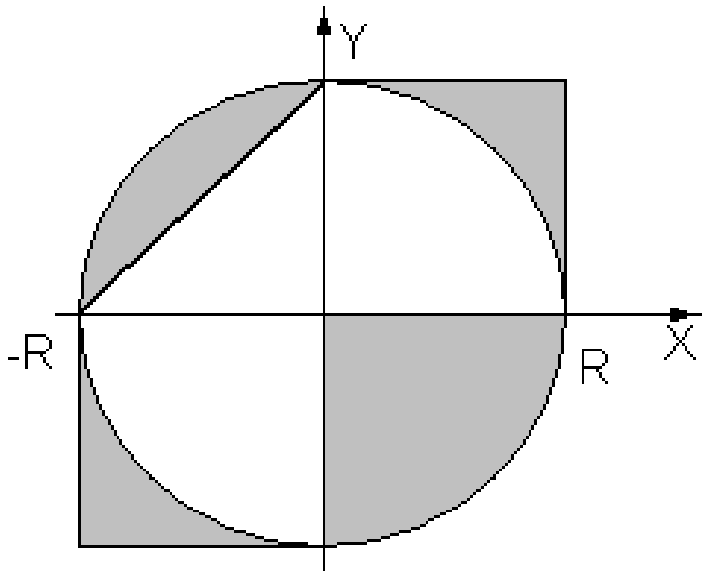
24)



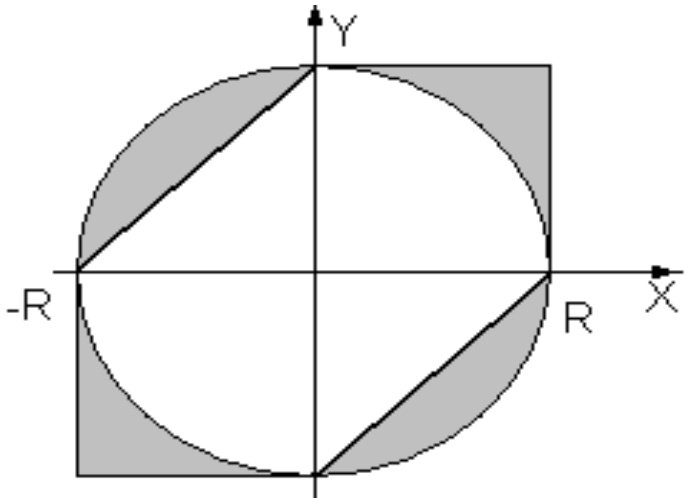
25)



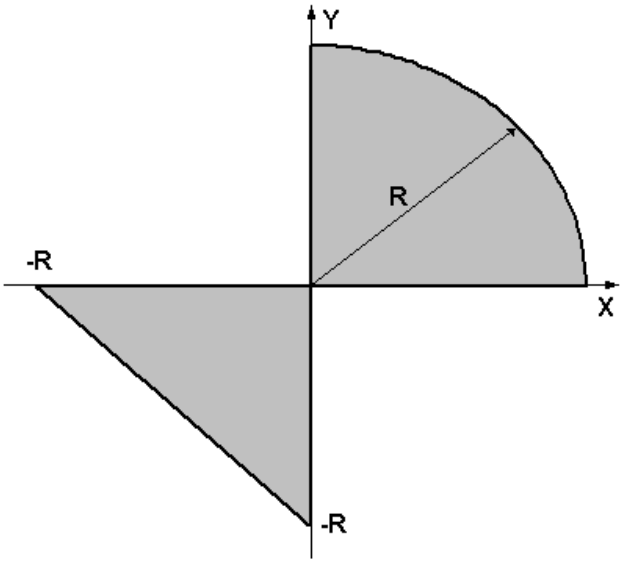
26)



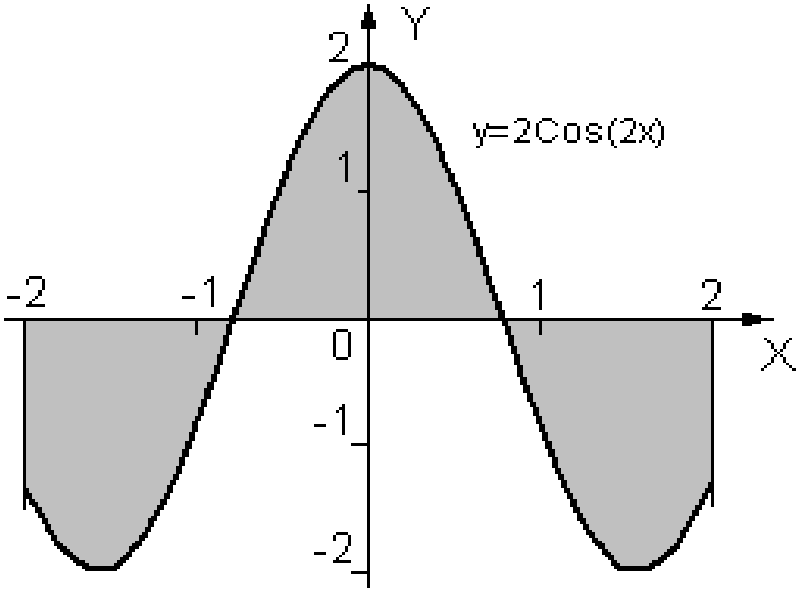
27)



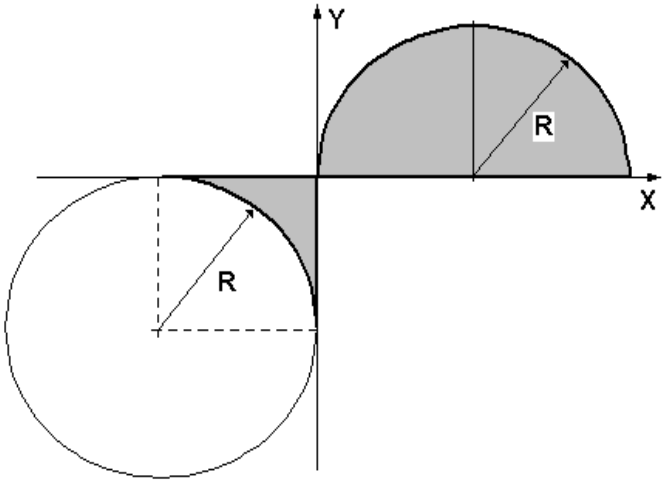
28)



29)



30)



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 в виде таблицы значения функции, заданной графически (см. лабораторная работа № 2, задание 1), на интервале от Хнач до Хкон 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 не использовался 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 Xbeg, Xend, dx.
2. Присвоить текущему значению Xt начальное значение: Xt = Хнач.
3. Вычислить значение функции и вывести в виде строки таблицы.
4. Вычислить новое значение аргумента Xt = Xt + Dx.
5. Если значение аргумента меньше Xend, 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= -10.00 Xend=  10.00
  Dx=   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
+--------+--------+
```

Задания 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. 23), на интервале от `Хнач` до `Хкон` 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 и оператор цикла с параметром:

```
for <Текущий элемент> in <Последовательность>:
    <Инструкции внутри цикла>
[else:
    <Этот код выполняется, если в теле цикла>
    <не использовался 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>)
```

Эта функция генерирует псевдослучайное число в диапазоне от <Начало> до <Конец>.

При этом правая граница не входит в интервал генерируемых значений (интервал открыт справа). 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 изменялись. 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
```

Задания 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).

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 Хнач до Хкон с шагом dx с точностью ε – эпсилон.

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

$$Si(x)=\int_0^x \frac{Sin(x)}{x}dx=\sum_{n=0}^{\infty}(-1)^n\cdot\frac{x^{2\cdot n+1}}{(2\cdot n+1)!\cdot(2\cdot n+1)}=x-\frac{x^3}{3!\cdot 3}+\frac{x^5}{5!\cdot 5}-...+\quad |x|<\infty$$

Теоретическое введение

При решении подобных задач, в которых имеется общая формула вычисления элемента ряда, 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. Следует обращать внимание на границы допустимых значений аргумента.

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

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

При выполнении преобразования выражение факториала в знаменателе было преобразовано к виду:

(2·(n+1)+1)! = (2·n+3)! = (2·n+1)!·(2·n+2)·(2·n+3)

2. Для решения задачи нам потребуется два цикла. Первый цикл While (с предусловием), будет обеспечивать изменение значения переменной X от Хнач до Хкон с шагом dx. Второй цикл – это цикл с постусловием. Он нужен для итерационных вычислений элементов ряда с $|a_n|<\varepsilon$.

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 `Хнач`, `Хкон`, `dx` и параметр точности `ε`.
2. Вывести "шапку" таблицы.
3. Инициировать `Xt` начальным значением (`Хнач`).
4. В цикле по `Xt`.
5. Инициировать переменную для подсчёта суммы членов ряда и переменную, которая отвечает за номер члена ряда (`n`).
6. В цикле по `an`.
7. Вычислить `k`, элемент ряда `an`, сумму элементов ряда и номер элемента.
8. Если модуль элемента ряда меньше `ε`, то прервать цикл (`break`) по `an`, иначе перейти к п.6.
9. Вывести строки таблицы: значение `Xt`, вычисленное значение функции и количество просуммированных членов ряда
10. Вычислить новое значение переменной `Xt =Xt + dx`.
11. Если значение аргумента меньше `Хкон`, то перейти к пункту 4.
12. Завершить рисование таблицы, и работу программы.

Описание входных и выходных данных

Поскольку тип переменных и точность представления не ограничены условием задачи, то входные переменные (`Хнач`, `Хкон`, `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 монитора в виде таблицы значения функции, заданной с помощью ряда Тейлора, на интервале от `Хнач` до `Хкон` с шагом `dx` и точностью `ε`. Таблицу снабдить заголовком и шапкой. Каждая строка таблицы должна содержать значение аргумента, значение функции и количество просуммированных членов ряда.

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

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

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

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

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

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

7. $arcctg(x) = \frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}\cdot x^{2\cdot n+1}}{2\cdot n+1} = \frac{\pi}{2}-x+\frac{x^3}{3}-\frac{x^5}{5}+\ldots, \qquad x\le 1$

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

9. $arctg(x) = \sum_{n=0}^{\infty}\frac{(-1)^{n+1}\cdot x^{2\cdot n+1}}{(2\cdot n+1)} = x-\frac{x^3}{3}+\frac{x^5}{5}-\frac{x^7}{7}+\ldots, \qquad |x|\le 1$

10. $Arth(x) = \sum_{n=0}^{\infty}\frac{x^{2\cdot n+1}}{2\cdot n+1} = x+\frac{x^3}{3}+\frac{x^5}{5}+\frac{x^7}{7}+\ldots, \qquad |x|<1$

11. $Arcth(x) = \sum_{n=0}^{\infty}\frac{1}{(2\cdot n+1)\cdot x^{2\cdot n+1}} = \frac{1}{x}+\frac{1}{3\cdot x^3}+\frac{1}{5\cdot x^5}+\ldots\ , \qquad |x|>1$

12.

$$\frac{1}{x+3}=\frac{1}{2}-\frac{(x+1)}{2^2}+\frac{(x+1)^2}{3}-\dots=\sum_{n=0}^{\infty}\frac{(-1)^n\cdot(x+1)^n}{2^{n+1}}$$

, –3 < x < 1

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

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

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

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

17. $\ln(x)=\sum_{n=0}^{\infty}\frac{(x-1)^{n+1}}{(n+1)\cdot(x+1)^{n+1}}=\frac{x-1}{x}+\frac{(x-1)^2}{2\cdot x^2}+\frac{(x-1)^3}{3\cdot x^3}+\dots, \qquad x>\frac{1}{2}$

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

19. $\arcsin(x)=x+\sum_{n=1}^{\infty}\frac{1\cdot3\cdot\dots\cdot(2\cdot n-1)\cdot x^{2\cdot n+1}}{2\cdot4\cdot\dots\cdot2\cdot n\cdot(2\cdot n+1)}=x+\frac{x^3}{2\cdot3}+\frac{1\cdot3\cdot x^5}{2\cdot4\cdot5}+\frac{1\cdot3\cdot5\cdot x^7}{2\cdot4\cdot6\cdot7}+$

$+\frac{1\cdot3\cdot5\cdot7\cdot x^9}{2\cdot4\cdot6\cdot8\cdot9}+\dots, \qquad |x|<1.$

20. $\arccos(x)=\frac{\pi}{2}-\left(x+\sum_{n=1}^{\infty}\frac{1\cdot3\cdot\dots\cdot(2\cdot n-1)\cdot x^{2\cdot n+1}}{2\cdot4\cdot\dots\cdot(2\cdot n)\cdot(2\cdot n+1)}\right)=$

$$\arccos(x)=\frac{\pi}{2}-\left(x+\sum_{n=1}^{\infty}\frac{1\cdot3\cdot\dots\cdot(2\cdot n-1)\cdot x^{2\cdot n+1}}{2\cdot4\cdot\dots\cdot(2\cdot n)\cdot(2\cdot n+1)}\right)=$$

$$=\frac{\pi}{2}-\left(x+\frac{x^3}{2\cdot3}+\frac{1\cdot3\cdot x^5}{2\cdot4\cdot5}+\frac{1\cdot3\cdot5\cdot x^7}{2\cdot4\cdot6\cdot7}+\frac{1\cdot3\cdot5\cdot7\cdot x^9}{2\cdot4\cdot6\cdot8\cdot9}+\dots\right),\ |x|<1$$

21. $sh(x)=\frac{1}{2}\cdot(e^x-e^{-x})=\sum_{n=0}^{\infty}\frac{x^{2\cdot n+1}}{(2\cdot n+1)!}=x+\frac{x^3}{3!}+\frac{x^5}{5!}+\frac{x^7}{7!}+ \qquad \left[x^2<\infty\right]$

22 $ch(x)=\frac{1}{2}\cdot(e^{x}+e^{-x})=\sum_{n=0}^{\infty}\frac{x^{2\cdot n}}{(2\cdot n)!}=1+\frac{x^{2}}{2!}+\frac{x^{4}}{4!}+\frac{x^{6}}{6!}+\qquad\left[x^{2}<\infty\right]$

23 $\ln\left(\frac{x+1}{x}\right)=2\cdot\left[\frac{1}{2x+1}+\frac{1}{3\cdot(2x+1)^{3}}+\frac{1}{5\cdot(2x+1)^{5}}+\ldots\right]\qquad(2x+1)^{2}>1$

24. $(1+x)^{\frac{1}{4}}=1+\frac{1}{4}\cdot x-\frac{1}{4}\cdot\frac{3}{8}\cdot x^{2}+\frac{1}{4}\cdot\frac{3}{8}\cdot\frac{7}{12}\cdot x^{3}-\frac{1}{4}\cdot\frac{3}{8}\cdot\frac{7}{12}\cdot\frac{11}{16}\cdot x^{4}+\ldots\qquad|x|\le 1$

25. $(1-x)^{\frac{1}{2}}=1-\frac{1}{2}\cdot x-\frac{1}{2}\cdot\frac{1}{4}\cdot x^{2}-\frac{1}{2}\cdot\frac{1}{4}\cdot\frac{3}{6}\cdot x^{3}-\frac{1}{2}\cdot\frac{1}{4}\cdot\frac{3}{6}\cdot\frac{5}{8}\cdot x^{4}-\ldots\qquad|x|\le 1$

26. $(1+x)^{-\frac{1}{3}}=1-\frac{1}{3}\cdot x+\frac{1}{3}\cdot\frac{4}{6}\cdot x^{2}-\frac{1}{3}\cdot\frac{4}{6}\cdot\frac{7}{9}\cdot x^{3}+\frac{1}{3}\cdot\frac{4}{6}\cdot\frac{7}{9}\cdot\frac{10}{12}\cdot x^{4}-\ldots\qquad|x|\le 1$

27. $(1+x)^{-3}=1-\frac{1}{1\cdot 2}(2\cdot 3\cdot x-3\cdot 4\cdot x^{2}+4\cdot 5\cdot x^{3}-5\cdot 6\cdot x^{4}+\ldots)\qquad|x|\le 1$

28. $(1-x)^{-4}=\frac{1}{1\cdot 2\cdot 3}(2\cdot 3\cdot 4\cdot x+3\cdot 4\cdot 5\cdot x^{2}+4\cdot 5\cdot 6\cdot x^{3}+5\cdot 6\cdot 7\cdot x^{4}+\ldots)\qquad|x|\le 1$

29. $(1+\mathrm{x})^{-5}=\frac{1}{1\cdot 2\cdot 3\cdot 4}(2\cdot 3\cdot 4\cdot 5\cdot \mathrm{x}-3\cdot 4\cdot 5\cdot 6\cdot \mathrm{x}^{2}+4\cdot 5\cdot 6\cdot 7\cdot \mathrm{x}^{3}-5\cdot 6\cdot 7\cdot 8\cdot \mathrm{x}^{4}+\ldots$

$|x|\le 1$.

30. $\ln(1+2\mathrm{x})=\ln 7+\frac{2(\mathrm{x}-3)}{7}-\frac{2^{2}(\mathrm{x}-3)^{2}}{2\cdot 7^{2}}+\cdots(-1)^{\mathrm{n}+1}\cdot\frac{2^{\mathrm{n}}(\mathrm{x}-3)^{\mathrm{n}}}{\mathrm{n}\cdot 7^{\mathrm{n}}},\quad -\frac{1}{2}<x\le\frac{13}{2}$

Лабораторная работа №4: «Одномерные массивы»

Цель работы:

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

Замечание: Для организации массивов в Python могут использоваться разные варианты, например, списки или решения, предлагаемые в модулях `array`, `numpy`, В этих решениях могут быть готовые методы и функции, упрощающие программирование, например поиск максимума, минимума или сортировка. В этом руководстве предлагается использовать алгоритмы поиска решений без использования готовых методов и функций.

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 `Amax` 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 массива 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']
```

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 4-х 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.

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 for с генерацией последовательности целых чисел, используемых для доступа 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]` сравнить со значением в `Max1`;
- Если `abs(A[i]) > Max1`, то `Max2 = Max1` и `Max1 = abs(A[i]);`
- Иначе модуль элемента `A[i]` сравнить с `Max2`. Если `abs(A[i]) > Max2`, то `Max2 = abs(A[i]);`

Для сжатия массива по заданному принципу (нулевые элементы размещаются в конце массива при сохранении порядка ненулевых элементов), используется следующий алгоритм:

– Массив просматривается с использованием индексируемой переменной `i`, дополнительный индекс `j` указывает на первый (нулевой) элемент массива;

– При обнаружении не нулевого элемента этот элемент копируется в элемент с индексом `j` и индекс `j` инкрементируется, указывая на следующую освободившуюся позицию;

– После просмотра массива все элементы, начиная с элемента с индексом `j` обнуляются.

Описание алгоритма

1. Запросить количество элементов N и пороговое значение `Amax`.
2. Инициировать 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 `Amax`.
4. Инициировать `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. Инициировать переменную `j`.
7. В цикле от 0 до `N-1`. Если значение элемента больше нуля, то копировать его 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])
j = 0
for i in range(n):  # Сортировка массива
    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
```

```
print("Конечное состояние")
for i in range(n): # Массив после сортировки
    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: 6
Пороговое значение A: 2.8
Начальное состояние
 -2.655   2.194   1.034   2.804  -1.490  -0.474
Конечное состояние
 -2.655   2.194   1.034  -1.490  -0.474   0.000
max1=  2.655 max2=  2.655 sum= -0.474
```

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 C.
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 $\overline{a} = \frac{1}{n} \cdot \sum_{i=1}^{n} a_i$.
2. Сумму квадратов разностей между элементами массива и полученным средним значением $S = \sum (a_i - \overline{a})^2$.

Упорядочить элементы массива по убыванию их разности со средним значением ($a_i - \overline{a}$): 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 ($Min \le a \le Max$); 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: $|a_i - a|$

Вариант 25

С использованием модуля *Random* сформировать одномерный массив, состоящий из *n* вещественных элементов в котором элементы принимают случайное значение, и выполняется условие: $a_1 < a_2 < a_3 < \ldots < a_n$.

1. Для заданного числа *y*, такого, что $a_1 < y < a_n$, определить такой *k*, чтобы $a_k < y < a_{k+1}$.
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: $a_1 > a_2 > a_3 > \ldots > a_n$.

1. Для заданного числа *y*, такого, что $a_1 < y < a_n$, определить такой *k*, чтобы $a_k < y < a_{k+1}$.
2. Вычислить сумму элементов массива, расположенных после элемента a_k.

Упорядочить по убыванию элементы, стоящие до элемента a_k .

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 *X0, Y0* 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 *X0, Y0*, 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. Инициировать 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.

Обратите внимание на строку, выделенную жирным. Элемент массива, ранее получивший числовое значение инициируется строковым значением. При этом Python выполняет эту операцию без выдачи предупреждения (элементы списка могут быть разного типа). Здесь это сделано только для примера.

Листинг программы

```
from random import *
n=int(input("Введите размер кв. матрицы (N): "))
A=[]                    # Пустой список
B=[]
for i in range(n):
    A.append([0]*n)     # Вложенный список
                        # инициированный 0
    B.append([3]*n)     # Вложенный список
                        # инициированный 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) # случайное
                                  # 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'              # А 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

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

    58 75 20
    17 99 45
    26 49 73

    58 75 20
    17 99 AB
    26 49 73
```

Отметим следующие моменты, касающиеся такой реализации кода:

- элементом списка может быть объект любого типа, это снижает эффективность кода;
- в случае обработки массивов (набор 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 функии, которые скрывают алгоритм решения, например, такие, как поиск максимума или минимума.

Для решения нашего задания, см. постановку задачи выше, 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()`. Когда генерацию псевдослучайных чисел необходимо выполнить в диапазоне `[a, b)`, где `b > a`, следует использовать следующее выражение:

```
y= a + (b - a)*Rand,
```

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

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

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

Инициализацию матрицы, вычисление среднего и дисперсии, а так же 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.

Выделяют два вида подпрограмм – это процедуры и функции. Процедуры и функции могут возвращать результат через параметры. Кроме этого функция всегда возвращает результат через собственное имя и, в отличие от процедур, может участвовать в выражениях.

В Python используется только понятие "функция". Объект функция создаётся инструкцией `def`, за которой следует имя функции и, при необходимости, параметры, которые указываются в круглых скобках. Заголовок функции завершается двоеточием. Тело функции образуется блоком инструкций, которые следуют далее и имеют отступ в четыре пробела, по умолчанию. При отсутствии инструкций в теле функции, например в процессе тестирования, записывают инструкцию `pass`, которая ничего не делает. Выход из функции выполняется после выполнения последней инструкции тела функции. Такой инструкцией может быть, но необязательно, и инструкция `return`. В этом случае функция, через свое имя, возвращает объект, который 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.

Функции могут быть вложенными в другие функции, но делать это не рекомендуется, см., например, М. Лутц, Изучаем Python, 2011.

Замечание: Переменные, используемые при описании функции, обычно 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.

Замечание о последовательности описания функций в Python:

При вызове функции (`f2()`) из другой функции (`f1()`) сама функция `f2()` может быть описана после `f1()`. Здесь важно только то, что вызов функции `f1()` выполнен после описания `f2()`. Это вызвано тем обстоятельством, что в Python вначале формируется байт код (это интерпретатор) и только затем работает виртуальная машина Python. Таким образом, к моменту выполнения кода (вызов `f1()`) размещение функции `f2()` известно и код может быть обработан без проблем.

Пример:

```
def f1(y):
    f2(y)  # Опережающая ссылка

def f2(x): # Вызываемая функция
    print(x)

f1(56)     # Вызов функции
```

Описание алгоритма

Рассмотрим алгоритм решения нашей задачи.

1. Запросить размер массива N. Поскольку форма массива не определена, решим задачу для квадратной матрицы (NxN).
2. Изготовить массив – инициировать 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 псевдослучайными числами в диапазоне `[a,b)`. Размер матрицы `n*n` элементов.

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, а 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)

Эта функция выводит массив 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, avr, 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]) - avr) > sigma:
                Matr[Row][Col] = avr
    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}"
      .format(mDisp, sqrt(mDisp))))
# Корректировка
NMatr = CorrectMatr(MyMatr, mMidl, sqrt(mDisp))
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

Midl =  -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 **A** назовем такие элементы a_{kl}, для которых выполняются условия: $i-1<=k<=i+1$, $j-1<=m<=j+1$, $(k, m) \neq (i, j)$.

Это все элементы, которые окружают центральный элемент по горизонтали, вертикали и диагоналям. Для крайних левых элементов соседей слева нет, так же и для правых крайних – соседей справа нет.

$$\begin{pmatrix} \mathbf{a}_{1,1} & \mathbf{a}_{1,2} & \dots & a_{1,j-1} & a_{1,j} & a_{1,j+1} & \dots \\ \mathbf{a}_{2,1} & \mathbf{a}_{2,2} & \dots & a_{2,j-1} & a_{2,j} & a_{2,j+1} & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{i-1,1} & a_{i-1,2} & \dots & \mathbf{a}_{i-1,j-1} & \mathbf{a}_{i-1,j} & \mathbf{a}_{i-1,j+1} & \dots \\ a_{i,1} & a_{i,2} & \dots & \mathbf{a}_{i,j-1} & \mathbf{a}_{i,j} & \mathbf{a}_{i,j+1} & \dots \\ a_{i+1,1} & a_{i+1,2} & \dots & \mathbf{a}_{i+1,j-1} & \mathbf{a}_{i+1,j} & \mathbf{a}_{i+1,j+1} & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{pmatrix}$$

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 10x10.

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. Понятие соседей дано в варианте 9.

Подсчитать количество локальных минимумов заданной матрицы размером 10 x 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

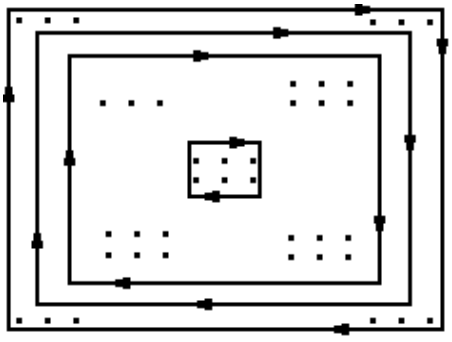
Осуществить циклический сдвиг элементов прямоугольной матрицы на n элементов вправо или вниз (в зависимости от введенного режима), n может быть больше количества элементов в строке или столбце.

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

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

Вариант 13

Осуществить циклический сдвиг элементов матрицы размером m * n (m строк x 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 14

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

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

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 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 **A** имеет седловую точку a_{ij}, если a_{ij} является минимальным элементом в i-й строке и максимальным в j-м столбце.

Вариант 20

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

Вариант 21

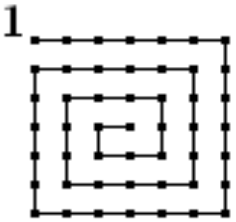
Дана целочисленная квадратная матрица. Определить:

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

Вариант 22

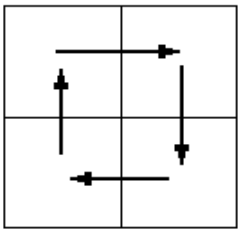
Напишите программу, формирующую квадратную матрицу, элементы которой являются натуральными числами, расположенными в порядке возрастания от 1 до n^2 (n – порядок матрицы) согласно схеме, приведённой на рисунке.

Вычислить сумму элементов, расположенных на главной диагонали полученной матрицы.



Вариант 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

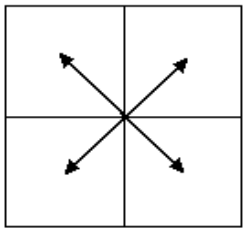
Даны две матрицы одного порядка m * n (m строк x n столбцов).
Написать программу сложения, вычитания и транспонирования матриц.

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

Вариант 26

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 27

Написать программу, которая заполняет матрицу размерности 4x13 числами от 1 до 52 случайным образом: повторение чисел не допускается. Полученный результат вывести на экран монитора.

Вариант 28

Произведением двух матриц **A** на **B** называется такая матрица **C**, для которой: $c_{ik} = a_{i1} \cdot b_{1k} + a_{i2} \cdot b_{2k} + ...a_{in} \cdot b_{nk} = \sum_{j=1}^{n} a_{ij} \cdot b_{jk}$.

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:

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

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

Сумма матриц **A** и **B** одинакового размера есть матрица **C** того же размера с элементами $c_{ik} = a_{ik} + b_{ik}$ при всех i и k. Таким образом, сложение матриц одинакового размера выполняется поэлементно.

Произведение матрицы на действительное число λ будет матрица, у которой все элементы умножены на λ: **C**=λ· **A**=|λ·a_{ik}|.

Произведением двух матриц **A** на **B** называется такая матрица **C**, для которой: $c_{ik} = a_{i1} \cdot b_{1k} + a_{i2} \cdot b_{2k} + ...a_{in} \cdot b_{nk} = \sum_{j=1}^{n} a_{ij} \cdot b_{jk}$.

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 вычисляющую экспоненту от матрицы с суммированием первых n элементов ряда.

Пример представления единичной матрицы, матрицы размерностью 2x2 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:

a) случайным образом заполняет элементы двухмерной матрицы размерности 6x6. Повторение символов не допускается;

b) 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) формирую 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 значительных трудозатрат 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

Файловый тип данных введён в языках программирования для работы с внешними устройствами – файлами на диске, портами ввода/вывода, принтерами и т.д. Файловый тип подразделяют на текстовый и бинарный (двоичный).

Текстовый файл содержит данные типа строка (`str`), а бинарный – типа `bytes`.

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: $0000\ 0010\ 1101\ 1011_2$;
- в текстовом: $37\ 33\ 31_{string}$.

Бинарные файлы относятся к файлам с прямым доступом. В них хранится информация в двоичном виде. При записи или чтении бинарного файла информация не подвергается дополнительному преобразованию.

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 (ошибка) 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 и выполним вычисления;

– результат вычисления 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 (предпочтительно).

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 – это данные, 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>` подставляется один из символов, см. Таблица 6.1.

Таблица 6.1. - Режимы работы с файлами

mod	Режим	Примечание
'r'	чтение файла (read)	файл должен существовать, если файл не существует, то возбуждается исключение: FileNotFoundError
'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: FileNotFoundError
'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: FileExistsError
'x+'	создать файл для чтения и записи	если файл существует, то возбуждается исключение: FileExistsError

Вместе с указанием режима может следовать модификатор, определяющий режим открытия файла: `t` – текстовый или `b` – 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 из файла:

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

```
2)      for line in fi:         # для всех строк файла
           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 НЕТ НИКАКИХ символов.

Во втором примере итерации (чтение строк) выполняются циклом `for`. Строки по очереди считываются 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). Эти строки – памятка, которой можно воспользоваться при подготовке файла в текстовом редакторе;

– в цикле читаем строку, и расщепляем 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 груп, на которые делится строка.

Если указан разделитель `sep` (sep – от separate – разнимать) 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 неограниченно. 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`).

Записывать 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("lab1_pb_in.txt", "rt") #читать файл
fo = open("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 вещест. типу
    x = float(c)
# Вывод в файл
    fo.write("I {0: .1f} I {1: .1f} I {2: 5.4f} I"
             .format(a, x, f1(a, x)))
    fo.write("{0: 6.4f} I\n".format(f2(x)))
    fo.write("+------+------+---------+"
             "--------+\n")
fi.close()   # закроем файлы
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_pb_in.txt", "rt") # читать файл
fo = open(r"lab1_pb_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"
             .format(a, x, f1(a, x)))
    fo.write("{0: 6.4f} I\n".format(f2(x)))
    fo.write("+-------+------+---------+"
             "--------+\n")
```

```
fi.close()    # закроем файлы
fo.close()
```

Замечание: Обратите внимание на то, как выполнен перенос длинных строк. В функции `f1()` в круглые скобки взято все выражение. В функции `f2()` использованы круглые скобки функции `pow()`, а в методе записи данных в файл `fo.write()` – круглые скобки метода.

Результат работы программы (текстовый файл lab1_pb_ou.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, avr, 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])-avr)>sigma:
                Matr[Row][Col] = avr
    return(Matr)
n,m=input("Введите размер матрицы (N M):").split()
n = int(n)
m = int(m)
tstMatr=MakeMatr(n, m, -7, 7)  # готовим матрицу
str_o = 'Resalt after initiation:'
fh_i = open("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("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")
# 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:'
fstr=" ".join(('\nMidl= {0:5.4f} Disp= {1:6.4f}',
               'Sigm= {2:6.4f}'))
str_o1 = fstr.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
```

Замечание: В этом примере методом `.join()` собрана форматная строка `fstr`, которая использована для сохранения результатов вычисления в файле, см. следующую инструкцию. Пробел в двойных кавычках перед методом – это разделитель, используемый при объединении строк. Вместо пробела можно вставить и комбинацию символов.

Результат работы программы (текстовый файл lab1_pd_ou.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
```

Это важно: При просмотре файла через Блокнот записи будут смотреться как одна строка. Откройте файлы через WordPad. В этом случае вы увидите другое представление данных. Сможете ли вы найти причину такой неоднозначности представления. Обратитесь за разъяснением к преподавателю.

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: «Программирование графики». Модуль turtle

Цель 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 для Задания 1 и Задания 2, а так же один из графических модулей Python, написать скрипты, 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 испытаний. Оценить площадь заштрихованной части фигуры методом Монте-Карло и сравнить с реальной площадью. 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`. В этом модуле имеется виджет Canvas, позволяющий рисовать графические изображения.

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 пиксел состоит из трех прямоугольных элементов, позволяющих формировать цвет пиксела 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 пиксел. 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 `Xrc, Yrc` – 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 пикселах, 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? Рис.7.1 поясняет сказанное.

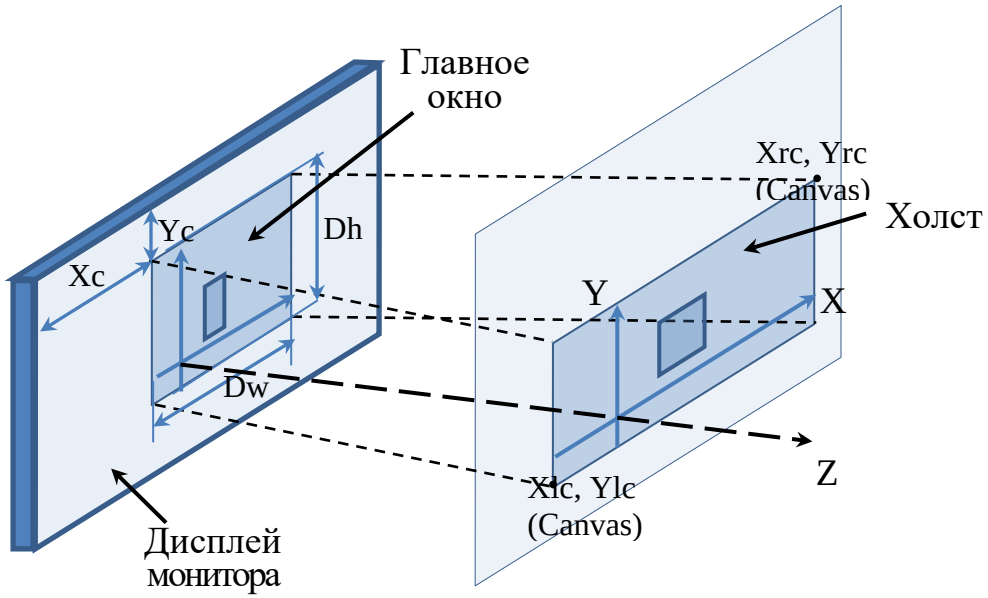


Рис.7.1. – Дисплей, главное окно, холст и мировая система координат

Холст размещается в плоскости проектирования (Z=0). Координаты Xlc и Ylc определяют положение левого нижнего угла холста, а Xrc, Yrc – правого. Размеры холста составляют: Cw = Xrc – Xlc и Ch = Yrc – Ylc. 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 пикселами 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.7.1, 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: Xmin, Xmax – диапазон изменения аргумента, а Ymin, Ymax – 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 Dw и Dh, соответственно, то для исключения искажений углов и элементов графика следует удовлетворить следующее условие:

```
k = Dw / Dh = (Xmax – Xmin) / (Ymax – Ymin)
```

$$k = \frac{Dw}{Dh} = \frac{Xmax - Xmin}{Ymax - Ymin}$$

или, например, Dh = Dw / k $$Dh = \frac{Dw}{k}$$

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 `a` и `b`. Первую точку с координатами (0, 0) не указываем, см. рис.8.2;
- 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.

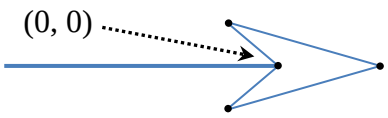


Рис.7.2.– Характерные точки стрелки, использованные в программе

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. Ниже приводится листинг программы и результат работы, см. рис.7.3.

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
# регистрируем новую форму черепашке
    tr.register_shape("myArrow",p)
    tr.resizemode("myArrow")
    tr.shapesize(1,2,1) # растягиваем (пример)
    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] # надпись
    else:                     # для оси Y
        tr.tiltangle(90)      # угол для формы
        tr.goto(0,txy[1]+0.2) # к месту стрелки
        pw = [0.2, int(txy[1])] # надпись
    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

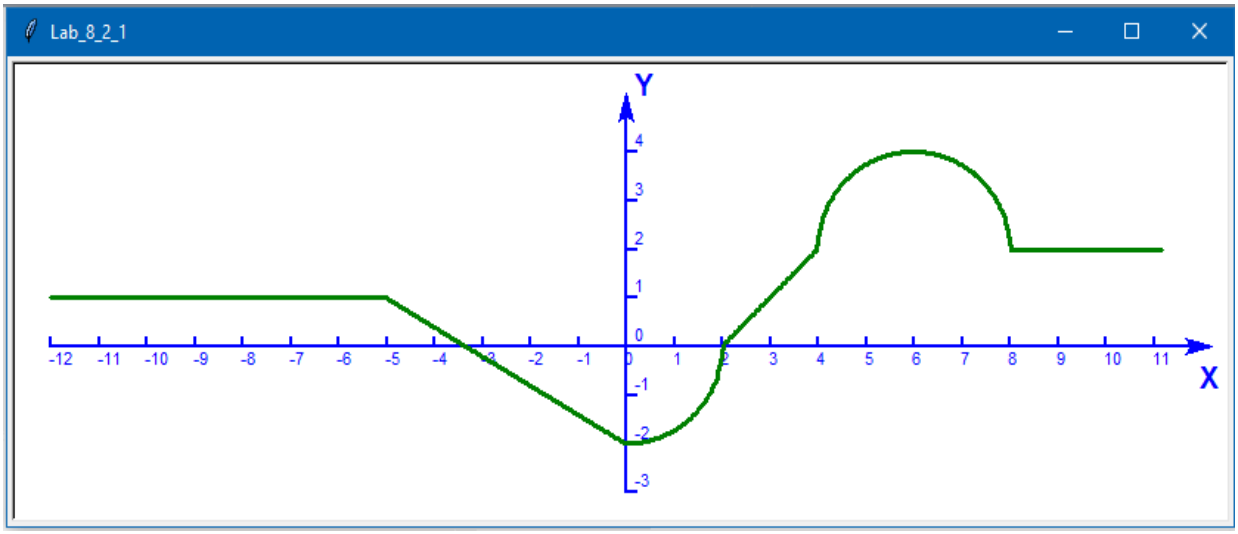


Рис.7.3.– График функции

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 $\sqrt{D/N}\sqrt{\frac{D}{N}}$, 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: $S_f = \int_a^b (f_2(x) - f_1(x)) \cdot dx$ $S_f = \int_a^b \left(f_2(x) - f_1(x)\right) \cdot dx$, где $f_2(x)$ и $f_1(x)$ – функции, которыми ограничена фигура, a и b – границы фигуры слева и справа, соответственно. При этом $f_2(x) >= f_1(x)$. Уравнения для наших линий следующие:

$f_1(x) = 2 \cdot x + 2$ и $f_2(x) = x^3 - 4x^2 + x + 6$ $f_2(x) = x^3 - 4 \cdot x^2 + x + 6$

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:

$$S_f = \int_{-1}^{1} \left(f_2(x) - f_1(x)\right) \cdot dx + \int_{1}^{4} \left(f_1(x) - f_2(x)\right) \cdot dx$$

Подробного решения не приводим. Вычисление дает: S_f = 253/12 ≈ 21.0833

$$S_f = \frac{253}{12} \approx 21.083з$$

Для вычисления площади фигуры методом Монте-Карло опишем около нее прямоугольник с основанием 7 и высотой 13 условных единиц (диапазон a_X = [–2, 5] и a_Y = [–2, 11]). Площадь прямоугольника составляет:

S = [5– (–2)] * [11– (–2)] = 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)
fstr = "N = {0:8d}\nNf = {1:8d}\nSf = {2:8.2f}"
meseg = fstr.format(Nmax, mfun, Sf)
tr.write(meseg, font=("Arial",12,"bold"))
print(meseg)
# Комментировать при работе в IDLE
# tr.done()
```

Результат работы программы

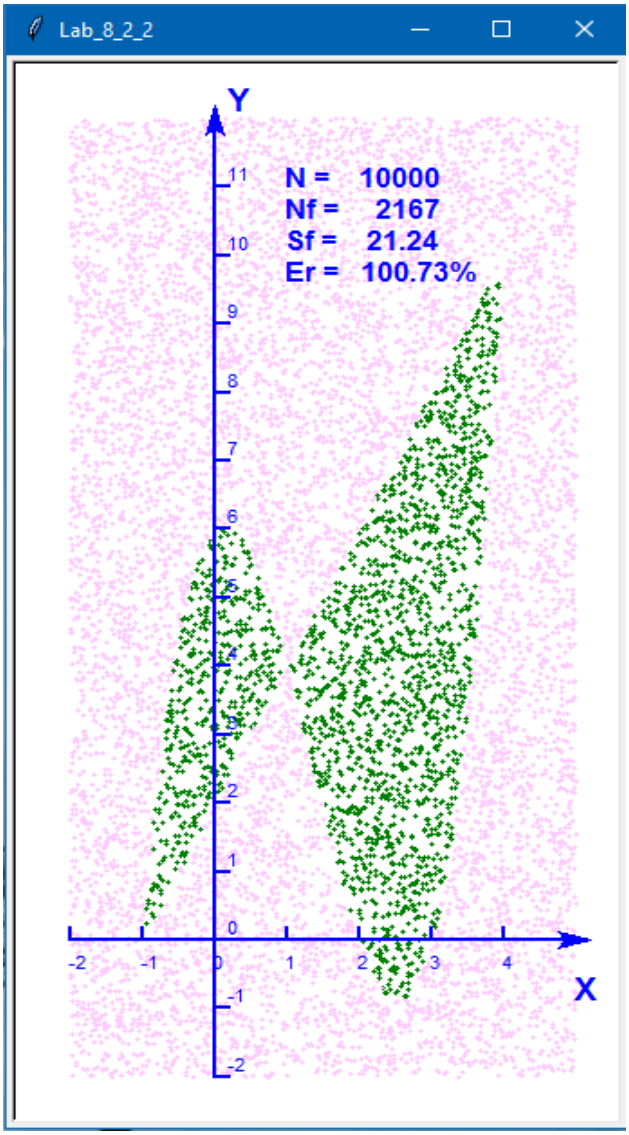


Рис.8.4. – Результат работы программы

Список рекомендованной литературы

1. Прохоренок Н.А., В.А. Дронов, Python 3и 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 №7 "Программирование графики". Модуль turtle

Выполнить в графической форме лабораторные работы №3 Задания 1, 2, 3.

При выполнении Задания 1 решить обратную задачу – нарисовать график функции.

В Задании 2, используя метод Монте-Карло определить площадь заштрихованной части фигуры. Получить точечное изображение фигуры, используя до 10000 испытаний. Выполнить оценку точности вычисления площади в процентах по отношению к реальной площади. Реальную площадь получить геометрическими методами, а при необходимости использовать интегральное исчисление.

В Задании 3 нарисовать график функции, значения которой вычисляются через ряд с точностью 10^{-3}.

Изображение графиков 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.

Лабораторная работа №8: «GUI, модуль Tkinter»

Цель работы:

В данной работе студенты познакомятся:

- с принципами организации событийно-управляемого программирования;
- с основами организации графического интерфейса пользователя (GUI) и виджетами модуля `Tkinter`*);
- со способами построения графиков функций методами этого модуля.

Замечание: Студенты, при решении задач этой лабораторной работы, не ограничиваются выбором модуля `Tkinter`. Приветствуется выбор других модулей или библиотек, например, PyQt.

Постановка задачи

Вывести на экран в графическом режиме графики двух функций на интервале от `Xнач` до `Xкон` с шагом `dx`. Первая функция задана с помощью ряда Тейлора, ее вычисление должно выполняться с точностью `ε`. Значение параметра `b`, определяющего смещение функции по оси ординат, вводится с клавиатуры. Графики должны быть плавными и различаться цветами.

$$y(x) = 2 \cdot \left(1 - \frac{2^2 \cdot x^2}{3!} + \frac{2^4 \cdot x^4}{5!} - \cdots + (-1)^n \cdot \frac{2^{2 \cdot n} \cdot x^{2 \cdot n}}{(2 \cdot n+1)!} + \cdots\right) \quad -\infty < x < \infty$$

$$z(x) = \frac{Sin(2 \cdot x)}{x} + b$$

Теоретическое введение

Для выполнения такой работы нам необходимо познакомиться с модулями Python, которые можно использовать для построения графика функции и организации интерфейса с пользователем. Кроме этого нам необходимо познакомиться с понятием "событие" и принципами организации программ, использующих эти события для построения GUI.

Событийно-управляемое программирование

Под событием в операционной системе понимается любое действие пользователя при его взаимодействии с программным интерфейсом: нажатие клавиш клавиатуры, одиночные и двойные щелчки кнопками мыши, перемещение мыши в окне. Кроме этих событий в системе существуют события, которые вызываются как самой операционной системой, так и другими приложениями: работа системного таймера, информационный обмен между запущенными процессами.

*) В пособии, по тексту, имя модуля пишется с большой буквы, как имя собственное. В скрипте имя модуля следует писать с маленькой буквы (tkinter) для Python 3. В Python 2 имя модуля пишется с большой буквы - Tkinter.

Все события, возникающие в вычислительной системе, воспринимается операционной системой. Эти события преобразуется в сообщение – запись, содержащую необходимую информацию о событии.

Например, это может быть код клавиши и переход её состояния – была клавиша нажата или отпущена. При возникновении события от мыши, в сообщении будет содержаться информация о том, какая кнопка была нажата, был ли это одиночный или двойной клик, а так же координаты маркера мыши в момент клика.

Сообщения поступают в общую очередь, откуда распределяются по очередям приложений.

Каждое приложение имеет свой цикл обработки сообщений, в котором выбирается сообщение из очереди приложения. Затем, через операционную систему, вызывается подпрограмма, предназначенная для обработки этого сообщения.

На рис.8.1 представлена структура программы, управляемой событиями.

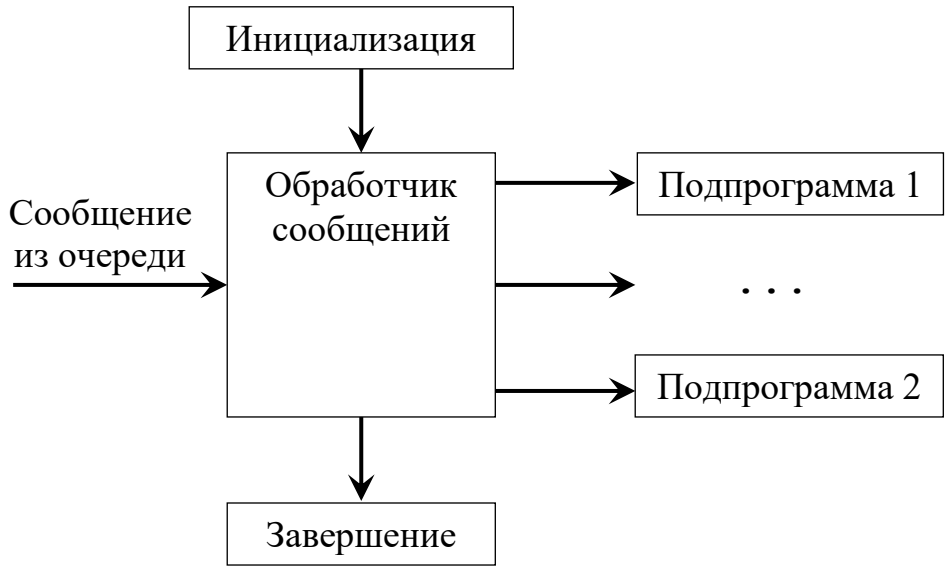


Рис.8.1. – Структура программы, управляемой событиями

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

Модуль Tkinter

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

Стандартным модулем, который устанавливается вместе с Python по умолчанию, является модуль `Tkinter`. У этого модуля понятный и ясный синтаксис и его можно использовать в простых программах с графическим интерфейсом.

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

Для использования методов и функций модуля необходимо выполнить импорт модуля. Это реализуется строкой:

```
from tkinter import *
```

Другая форма импорта – с назначением псевдонима (алиас – калька с английского, alias – псевдоним):

```
import tkinter as tk
```

Виджеты

В Tkinter визуальные компоненты, используемые для построения пользовательского графического интерфейса, называют виджетами (*widget*, от англ. *window gadget*).

Виджет – элеме́нт интерфе́йса – примитив графического интерфейса пользователя (GUI), имеющий стандартный внешний вид и выполняющий стандартные действия. Иногда можно встретить и другое название контро́л (англ. *control*). Вот небольшой перечень виджетов, которыми мы воспользуемся при решении нашей задачи:

– кнопка (`Button`) – используется для запуска команды, или, например, для выбора следующего действия. Например, кнопка "Рисовать" запустит процесс рисования графика, а по кнопке "Выход" скрипт завершит работу;

– однострочное текстовое поле (`Entry`) – этот виджет предназначен для вывода или ввода только одной, как правило, небольшой строки, например, название предмета или фамилия;

– метка (Label) – используется, как правило, для информирования пользователя. Например, метка может быть описанием однострочного текстового поля Entry: назначение поля;

– холст, канва, полотно (Canvas) – область рисования графических изображений;

– 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. Ознакомьтесь самостоятельно с этими виджетами (Spinbox, Combobox).

Некоторые виджеты, например Entry, Button, получают фокус, что позволяет связать такой виджет с клавиатурой. Фокус может быть переведён на виджет, кликом мыши на нём или с использованием клавиши Tab.

События, связанные с виджетом (клик мыши, нажатие клавиш клавиатуры) могут быть привязаны к методу (функция), который будет обрабатывать событие. Например, при клике на полотне может быть вызван метод, который позволяет получить координаты курсора мыши в момен клика.

Важно: Виджеты, описываемые в скрипте обязательно должны быть обработаны упаковщиком, который размещает их на форме.

Существуют три метода упаковки виджетов:

Метод pack

В этом методе указывается, как виджет будет заполнять пространство формы: где будет размещаться, и как будет изменяться при изменении размеров формы.

Метод place

Этот метод позволяет программисту контролировать положение виджета путём задания координат, и его размеров. 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, на скольких строках и столбцах будет размещаться виджет.

В нашем скрипте будем использовать этот упаковщик.

Заметим, что не рекомендуется смешивать упаковщики при формировании одной формы. Больше информации о виджетах и методах их упаковки можно получить в соответствующей литературе или в сети Интернет.

Интерфейс программы

Выделим в нашем скрипте две части. В головной части мы поместим описания функций, которые будут выполнять некоторый набор операций, а во второй – описание виджетов, которые будут организовывать интерфейс, для взаимодействия с пользователем.

Нам потребуются следующие виджеты:

- `Canvas` – полотно для графического представления функций;
- `Entry` – поле для ввода данных, например, смещение графика по оси ординат или вывода данных о положении курсора мыши;
- `Button` – кнопка для задания действия: вывод обновлённого графика, после изменения значений, задаваемых через поле Entry; завершение работы программы;
- `Label` – метка, для описания полей Entry.

Уточним техническое задание к интерфейсу скрипта.

Полотно (Canvas) будет занимать основную верхнюю часть формы. Поля ввода, метки с описанием этих полей и кнопки будут размещены в нижней части формы. В скрипте опишем следующие поля:

– поля для вывода координат указателя мыши в момент клика;

– поля для задания границ области координат: абсциссы и ординаты. Задание этих полей позволит просматривать форму графика в разных областях координат;

– поля для задания смещения функции и шага, который будет использоваться при нанесении разметки на осях координат.

– кнопки для вывода графика и завершения работы скрипта.

Макет окна представим в виде рисунка, см. рис.8.2

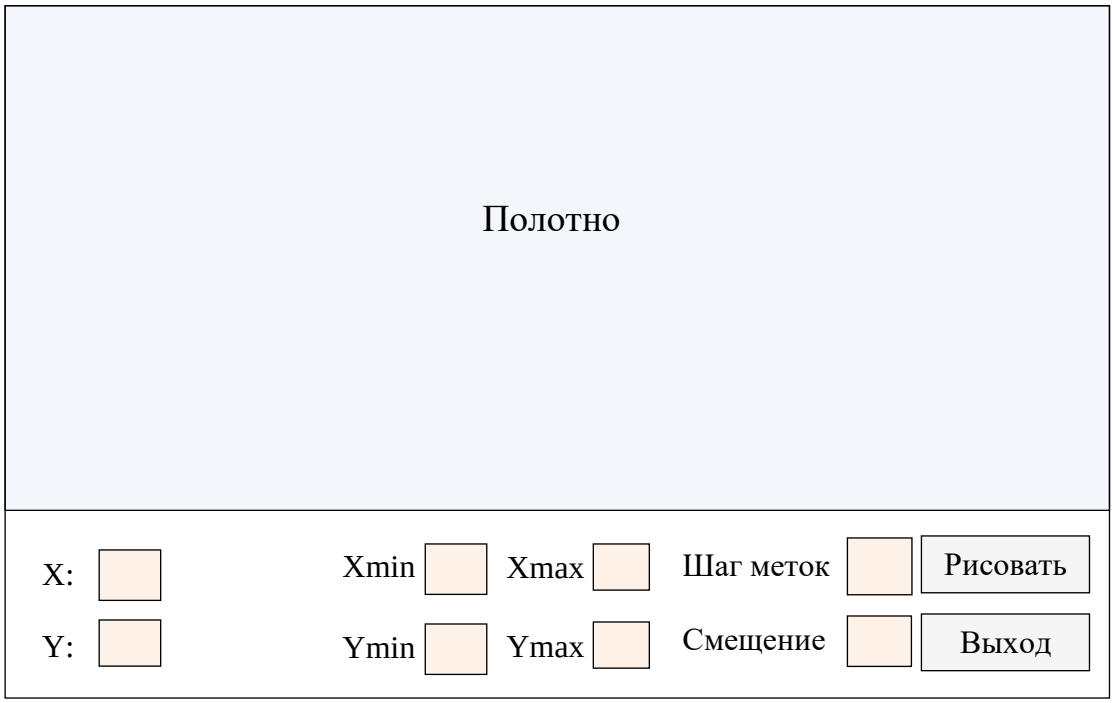


Рис.8.2. - Макет окна программы

Прежде, чем мы начнём описывать наш макет в виде скрипта познакомимся кратко с параметрами используемых виджетов.

Canvas – полотно

Методы виджета Canvas позволяют создавать графические примитивы, с помощью которых формируется рисунок:

– линия – create_line(). Для рисования линии необходимо задать начальные и конечные координаты концов линии, её толщину, цвет, тип (сплошная или пунктир), тип концов (наличие или отсутствие стрелок);

– замкнутый контур – create_poligon(). Для рисования полигональной линии задаётся список координат, через которые пройдёт контур, параметры, описанные для линии, цвет заливки фигуры. Кроме этого можно задать способ проведения линии между точками. В этом случае координаты точек

будут играть весовую роль (линия не будет проходить через точки), но будут влиять на форму получаемой фигуры;

– многоугольник – create_rectangle. Этот примитив позволяет рисовать прямоугольники;

– овал – create_oval. Для рисования эллипса и окружности задаются координаты вершин прямоугольника (квадрата), в который он будет вписан. При этом определяются координаты верхнего левого угла и нижнего правого, цвет заполнения. Также как и для линии можно задавать ширину контура, цвет, и т.п.;

– круг, дуга – create_arc(). Этот примитив используется для рисования дуг, секторов или сегментов (определяется параметром `style`) . Он подобен овалу, но имеет большее число параметров;

– текст – create_text(). Этот примитив используется для вывода текстовой информации в графическом поле, которым представляется полотно. Текст может быть многострочным. В качестве параметров задаются координаты якоря текстового поля (anchor), относительно которого текст выравнивается в этом поле. Можно задавать цвет фона и символов, тип шрифта, размер, …

Следует отметить важную особенность полотна. Все рисуемые примитивы возвращают идентификатор, который можно присвоить переменной, а затем использовать её значение, например, для удаления с полотна изображения примитива.

Система координат полотна

Разберёмся с системой координат, которая реализована на полотне (Canvas).

Отметим, что в этой системе координат единицей измерения является пиксел. В модуле Tkinter нет метода, позволяющего создать собственную пользовательскую систему координат, как это реализовано в модуле Turtle, см. лабораторную работу №7.

Организация системы координат на полотне показана на рис.8.3.

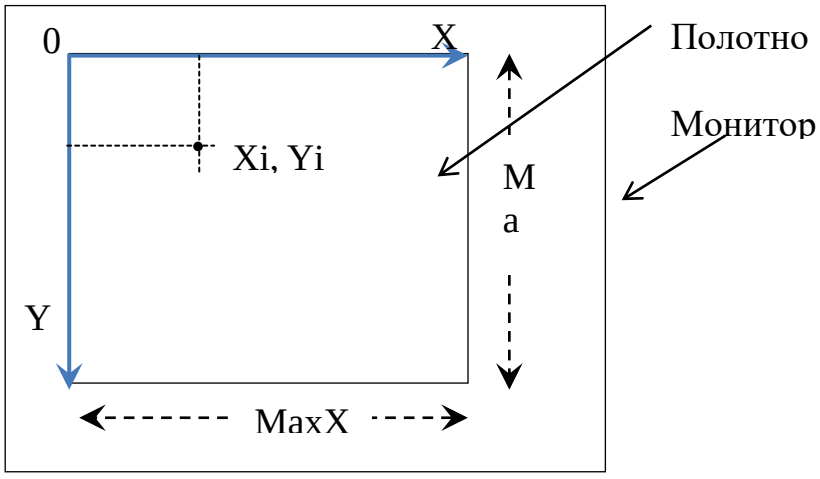


Рис.8.3.– Система координат полотна и параметры окна

На этом рисунке показано, что ось абсцисс направлена с лева направо, а ось ординат – сверху вниз. Следует обратить внимание на то, что размеры полотна (MaxX, MaxY), а так же координаты точки (Xi, Yi) указываются в пикселах.

Учитывая эту особенность модуля, а так же то, что пользователь должен вводить или получать координаты в собственной системе координат, нам придётся преобразовывать данные, введённые пользователем, в пикселы для рисования линий графика и возвращать данные, преобразованные в систему координат пользователя.

Выведем формулы преобразования координат графика, определённых в пользовательской системе измерения, в пикселы. Пусть:

– Xmin, Xmax, Ymin, Ymax – минимальные и максимальные координаты, отображаемые на полотне для осей X и Y, соответственно. Данные представляются в условных единицах измерения длины: сантиметры, метры;

– MaxX, MaxY – размеры полотна по осям X и Y, соответственно в пикселах.

Тогда масштабирующие коэффициенты, показывающие, сколько пикселов приходится на единицу длины пользовательской системы измерения, по осям абсцисс и ординат вычисляются так:

$$Kx = \frac{MaxX}{Xmax - Xmin} \qquad Ky = \frac{MaxY}{Ymax - Ymin}$$

Таким образом, для некоторой точки с координатами Xi, Yi, выраженными в системе координат графика (сантиметры, метры, …), мы можем получить координаты Xp, Yp, которые отображают положение точки в пикселах полотна:

$$Xp = Kx \cdot Xi \; ; \qquad Yp = MaxY - Ky \cdot Yi \; .$$

Во втором выражении учтён, тот факт, что ось ординат в системе координат полотна направлена вниз.

Напишем скрипт, который покажет, как используется виджет Canvas.

```
from tkinter import *  # Импорт модуля
root = Tk() # Создание главного окна
root.title("Графика") # Надпись в заголовке окна
root.resizable(False, False) # Не менять размер окна
                             # по ширине и высоте
Kp = 0.7 # используем 70% экрана
MaxX = root.winfo_screenwidth()* Kp  # Текущие размеры
MaxY = root.winfo_screenheight()* Kp # дисплея монитора
                                     # в пикселах
cv = Canvas(root, width = MaxX, height = MaxY,
            bg = "white")
cv.grid(row = 0, columnspan = 11)
```

В первой строке скрипта выполняется импорт модуля Tkinter.

Обратите внимание на то, что имя модуля записано с маленькой буквы. Затем создаётся главное окно скрипта, и выводится его название в заголовке формы.

Изменение размеров окна потребует дополнительного внимания и обработки, например, перерисовки графиков. Снимем эту проблему, запретив изменение размеров окна. Для этого используется метод:

```
resizable(bW, bH),
```

где параметры bW – изменение размера по ширине, и bH – изменение размера по высоте, принимают логическое значение False или True.

Через методы главного окна можно получить размеры дисплея монитора.

```
winfo_screenwidth()     # Ширина в пикселах
winfo_screenheight()    # Высота в пикселах
```

Эти размеры сильно отличаться для разных мониторов, поэтому зададим размеры полотна на 70% меньше размеров дисплея.

Привязка виджета к форме выполняется при его описании. При этом формируется идентификатор виджета, значение которого может быть присвоено переменной. Обратите внимание на строку, описывающую виджет Canvas:

```
cv = Canvas(root,...)
```

root – имя главного окна, cv – имя полотна, которое мы будем указывать при рисовании примитивов. Заметим, что на форме может быть создано несколько полотен.

Упаковка виджетов, в нашем примере, будет выполняться методом grid(). В качестве аргументов мы передаём номер строки, на которой будет размещаться полотно, и сколько столбцов оно будет занимать. Мы указали нулевую строку, поскольку виджет будет размещаться в верхней части формы. Количество занимаемых столбцов указано 11. Обратитесь к рис.8.2 и посчитайте, сколько виджетов будет размещено по горизонтали в первой строке, если каждый виджет будет занимать одну ячейку воображаемой таблицы. Заметим, что нумерация строк и столбцов начинается с нуля и первая строка занята полотном.

Создайте скрипт с указанным текстом и посмотрите на результат.

Добавим в нашу форму метки (Label), которые мы нарисовали на нашем макете, рис.8.3.

Для описания меток используется метод Label(), в аргументах которого надо указывать имя окна, к которому относится метка. Дополнительно можно указать текст, ширину окна, в котором надо поместить текст, параметры текста (цвет, тип шрифта и его размер), а так же параметры выравнивания текста (по левому краю, по центру, по правому краю). Значение выравнивания задаётся либо словом CENTER – центр, либо символом или комбинацией символов. Символы – это первые буквы

английских слов, описывающие стороны света: **N**orth – Север, **E**ast – Восток, **S**outh – Юг и **W**est – Запад, рис.8.4.

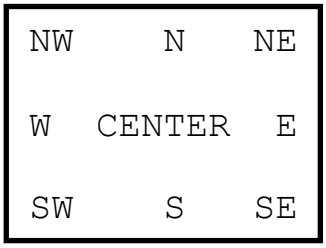


Рис.8.4. – Направления выравнивания текста

Идентификационный код объекта, который возвращается методом `Label()`, присвоим переменным. Опишем все восемь меток нашего макета.

```
lba0 = Label(root, text = "X:", width = 10,
             fg = "blue", font = "Ubutu, 12")
lba0.grid(row = 1, column = 0, sticky='e')
lba1 = Label(root, text = "Y:", width = 10,
             fg = "blue", font = "Ubutu, 12")
lba1.grid(row = 2, column = 0, sticky='e')
lba2 = Label(root, text = "Xmin:", width = 10,
             fg = "blue", font = "Ubutu, 12")
lba2.grid(row = 1, column = 2, sticky='e')
lba3 = Label(root, text = "Xmax:", width = 10,
             fg = "blue", font = "Ubutu, 12")
lba3.grid(row = 1, column = 4, sticky='e')
lba4 = Label(root, text = "Ymin:", width = 10,
             fg = "blue", font = "Ubutu, 12")
lba4.grid(row = 2, column = 2, sticky='e')
lba5 = Label(root, text = "Ymax:", width = 10,
             fg = "blue", font = "Ubutu, 12")
lba5.grid(row = 2, column = 4, sticky='e')
lba6 = Label(root, text = "Шаг меток:", width = 10,
             fg = "blue", font = "Ubutu, 12")
lba6.grid(row = 1, column = 6, sticky='e')
lba7 = Label(root, text = "Смещение:", width = 10,
             fg = "blue", font = "Ubutu, 12")
lba7.grid(row = 2, column = 6, sticky='e')
```

Места размещения меток, указанные в упаковщике `grid()`, были вычислены в соответствии с ранее разработанным макетом, см. рис.8.2. Добавьте этот код к предыдущему и посмотрите результат.

Для завершения формы необходимо добавить поля для ввода и вывода данных – виджеты `Entry`. Эти поля описываются в полной аналогии с описанием меток, например:

```
ent0 = Entry(root, width = 5, font = "Ubuntu, 12")
ent0.grid(row = 1, column = 1)
```

Вставить значение строкового типа в поле `Entry` можно методом `insert()`:

```
ent0.insert(index, s),
```

где строка `S` вставляется перед символом, позиция которого задается параметром `index`.

Удалить символы из виджета можно методом `delete()`:

```
ent0.delete(first, last=None),
```

где параметр `first` задаёт начальную позицию удаляемых символов, а параметр `last` последнюю. При этом позиция, заданная в `last` не включается в удаляемый диапазон. Если `last` не задан, то удаляется только один символ. Если вторым параметром задано значением `END`, то удаляются все сиволы.

Прочитать значение, введённое пользователем в поле, можно методом `get()`:

```
var = float(ent0.get())
```

Прочитанное значение имеет строковый тип. Таким образом, при вставке числового значения его необходимо преобразовать к строковому типу `str()`. При чтении из поля `Entry` числового значения необходимо выполнить обратное преобразование к ожидаемому типу. В нашем скрипте мы будем использовать вещественные значения: `float()`.

Задание:

Добавьте виджеты `Entry` самостоятельно. Их должно быть восемь. Имена переменных, с помощью которых можно обращаться к этим виджетам, составьте по тому же принципу, что и имена для меток. Т.е. `entN`, где `N` – номер виджета.

Два первых поля (`ent0` и `ent1`) предназначены для вывода координат мыши. Инициализируйте их нулевым значением, а оставшиеся поля, предназначенные для ввода, инициализируйте начальными значениями:

```
Xmin = -10.0; Xmax = 10.0 # Границы по абсциссе
Ymin = -5.0; Ymax = 5.0   # Границы по ординате
dX = 50  # Шаг меток по осям
dY = 1.0 # Смещение графика
```

Замечание: Некоторые переменные, использованные в нашем скрипте, будут глобальными, что позволит использовать их в различных функциях без процедуры передачи. Мы поступили так, что бы сократить число параметров, передаваемых в функции.

Назначение переменных глобальными должно делаться более обоснованно.

Проверьте работу скрипта.

Теперь можно добавить кнопки (`Button`). При описании этих виджетов можно задать надпись, которая будет размещена на кнопке, параметры текста (шрифт, цвет) и что очень важно, привязать нажатие

кнопки к вызову функции, которая получит событие от нажатия кнопки и обработает его.

```
btn1 = Button(root, width = 20, bg = "#ccc",
              text = "Рисовать")
btn1.grid(row = 1, column = 10)
btn1.bind("<Button-1>", Draw)
```

В первой строке указан идентификатор кнопки (`btn1`), параметр root показывает, что кнопка находится на форме, остальные параметры задают размер, цвет фона и текст. Метод `grid()` поместит кнопку в первой строке в десятой колонке.

Для обработки события, связанного с кнопкой, будет вызываться функция `Draw()`. Обратите внимание, что переменная `Draw` записана без круглых скобок.

Код для второй кнопки будет аналогичен:

```
btn2 = Button(root, width = 20, bg = "#ccc",
              text = "Выход")
btn2.grid(row = 2, column = 10)
btn2.bind("<Button-1>", Final)
```

Перед тем, как перейти к описанию функций (головной части), добавим связку между событием "клик кнопкой мыши на полотне" с соответствующим обработчиком. Поскольку идентификатор полотна у нас имеет имя `cv`, то связующая строка примет вид:

```
cv.bind('<Button-1>', showXY),
```

где `showXY` функция, которая будет вызвана по событию `'<Button-1>'`. При желании можно в качестве события выбрать двойной клик кнопкой мыши. В этом случае нам следует указать код `<Double-Button-1>`.

Прочитайте больше о событиях и связывании событий в Интернете.

Заключительный шаг для формирования окна скрипта – это организация непрерывного цикла, в котором будут обрабатываться события, см. рис.8.1.

Для организации цикла, в котором будет находиться скрипт в ожидании событий, используется метод `mainloop()`:

```
root.mainloop()
```

Основные шаги по подготовке окна выполнены. В процессе написания скрипта созданы три условия для возникновения событий: кнопка "Рисовать", кнопка "Выход" и событие, связанное с кликом левой кнопки мыши. Реализуем обработку этих событий в виде функций.

```
def Drav(event):
    '''Вызов функций для рисования графиков'''
    pass
#
```

```
def Final(event):
    '''Завершение работы'''
    window_deleted()
#
def window_deleted():
    ''' Завершаем работу по [X]'''
    if askyesno("Выход", "Завершить работу?"):
        root.destroy()
#
def showXY(event):
   '''Вывод координат мыши'''
    global ID1, ID2
    x = event.x; y = event.y
    ent0.delete(0,END)  # Удалим старые
    ent1.delete(0,END)  # значения
# Вычислим и введём новые
    ent0.insert(0, str(round(Xmin + x/Kx, 2)))
    ent1.insert(0, str(round(Ymin + (MaxY - y)/Ky, 2)))
    cv.delete(ID1)  # Удалим старые
    cv.delete(ID2)  # линии
# Нарисуем новые линии
    ID1 = cv.create_line(0,y,MaxX,y, dash = (3,5))
    ID2 = cv.create_line(x,0,x,MaxY, dash = (3,5))
```

Отметте, что в качестве параметра функций указана переменная `event`. Через эту переменную в функцию передаётся описание события.

Функция `Draw()` будет вызывать другие функции. Они нами ещё не разработаны, и мы используем инструкцию pass. Это позволит тестировать скрипт раньше.

Функция `Final()` должна сработать по нажатию кнопки "Выход" и завершить работу скрипта. Поскольку завершение работы скрипта может быть при случайном клике, создадим окно, в котором будет запрос на подтверждение закрытия и две кнопки "Да" и "Нет".

В Tkinter имеется модуль `tkinter.messagebox`, в котором имеется набор окон с разной функциональной направленностью. В нашем случае это:

```
askyesno("Выход", "Завершить работу?")
```

Здесь `askyesno()` метод, формирующий окно запроса с двумя кнопками. В качестве параметров указываются строки. Первая строка – заголовок окна, а вторая – сообщение, которое появится в окне. При нажатии на кнопку возвращается логическое значение: True – нажата кнопка "Да" и False – нажата кнопка "Нет".

Существует ещё один путь закрытия окна – это кнопка [X] в правом верхнем углу формы. Для обработки этого события используется метод `protocol()`, который перехватывает событие, связанное с закрытием окна по кнопкой [X]:

```
root.protocol('WM_DELETE_WINDOW', window_deleted)
```

Здесь `'WM_DELETE_WINDOW'` – наименование протокола, связанного с кнопкой закрытия окна ([X]), а `window_deleted` – функция, обрабатывающая событие. Отметим, что при обработке этого события его параметры в функцию не передаются.

Оформим сказанное в виде двух функций. Одна вызывается при закрытии окна по кнопке [X] и выводит запрос на подтверждение, а вторая вызывает первую. Само закрытие окна происходит при вызове метода `destroy()`:

```
root.destroy()
```

Функция `showXY()` вызывается по событию "одиночный клик левой кнопкой мыши" на полотне и выводит координаты точки на полотне, в которой был сделан клик, в поля `Entry`. Реализация этой функции усложнена по следующим соображениям:

– координаты точки – это два числа отображающие положение точки в пикселах в системе координат полотна. Их необходимо преобразовать в систему координат пользователя. Для этого используем знание о масштабе по каждой из осей. Результат преобразования чисел будет вещественным, поэтому округлим их до двух цифр после запятой. Перед выводом в поле `Entry` числа необходимо преобразовать в строковый тип;

– точку, в которой произведён клик, надо визуализировать. Для этого нарисуем две перпендикулярные пунктирные линии, которые будут проходить через заданную точку параллельно координатным осям. Метод, используемы при рисовании линий, возвращает идентификатор, который мы используем для удаления линий перед новой прорисовкой графика.

Обратитесь к коду функций, приведённому выше, для более полного усвоения этого материала. Заметим, что метод `cv.delete(ID1)` не вызывает ошибки в том случае, когда `ID` не определён: перед первой прорисовкой мы удаляем несуществующий объект.

Координатные оси

Для оценки значений, которые принимает функция, на полотне рисуются координатные линейки с разметкой. Функция `plotXY()` рисует линейки по краям полотна.

Для прорисовки координатной линейки вначале рисуется прямоугольник, а затем на его сторонах наносятся метки в виде коротких линий и текстовые значения, соответствующие этим меткам. При нанесении значения выполняется его вычисление, округление до двух знаков после запятой, а затем преобразование в строковый тип.

В дополнение к этому мы реализовали метод считывания координат полотна по событию, связанному с кликом левой кнопкой мыши.

Рисование функций

Для рисования функций вычисляются значений функции в диапазоне от `Xmin` до `Xmax`. При этом используется пользовательская система координат. Значения абсциссы и ординаты объединяются и формируют список, который возвращается при вызове функции. Поскольку в задании требуется изобразить переходы между вычисляемыми точками функции плавными, вычисления выполняются с шагом в один пиксел по оси X. В пользовательской системе координат шаг будет равен `1/Kx`, где `Kx` – масштабный коэффициент по оси X.

Рисование функций начинается с появления события, связанного с кликом по кнопке "Рисовать". При этом вызывается функция Drav(), в которой последовательно выполняются следующие операции:

Очистка полотна от ранее нарисованных объектов;

Получение данных из полей Entry. Данные, при считывании, контролируются на допустимый диапазон значений. Если на этом шаге возникает ошибка, то выводится предупреждающее сообщение и пользователь может ввести новые данные. Работа скрипта не прерывается;

Рисуется координатная линейка по краям полотна;

Вызывается функция, прорисовывающая график. При вызове, в качестве аргументов задаётся имя функции, которую необходимо нарисовать и цвет, которым будет выполнен график.

Листинг скрипта

Здесь представлен листинг скрипта, решающий поставленную задачу. Он вполне рабочий, но может быть существенно улучшен, например, тем, что основную часть скрипта можно оформить в виде отдельного модуля. Пользователь сможет использовать такой модуль для отображения и других функций в графической форме. При этом ему достаточно будет подготовить собственный скрипт, в который может быть импортирован наш модуль и описаны функции, для которых нужно построить график.

Но эта тема здесь не развивается. Студентам предлагается самостоятельно подумать над возможными улучшениями скрипта.

```
from tkinter import *
from tkinter.messagebox import *
from math import sin, cos, pi, exp
#
def Sin2xDevx(): # Функция из задания
    eps = 0.0001
    Lfun = []
    x = Xmin
    while x <=Xmax:
        an = 1
        Sum = an
        n = 1
```

```
        while (abs(an) > eps):
            an *= -(2**2)*(x**2)/((2*n)*(2*n+1))
            Sum += an
            n += 1
        Lfun.append((x, 2*Sum))
        x += 1 / Kx
    return Lfun

def Sin2xOnX():  # Функция из задания
    Lfun = []
    x = Xmin
    while x <= Xmax:
        if x != 0:
            fun = sin(2*x)/x + dY
        else:
            fun = 2.0
        Lfun.append((x, fun))
        x += 1 / Kx
    return Lfun

def GetData():
    '''Получить данные'''
    global Xmax, Xmin, Ymax, Ymin
    global dX, dY, Kx, Ky
    tmpXmax = float(ent3.get())
    tmpXmin = float(ent2.get())
    tmpYmax = float(ent5.get())
    tmpYmin = float(ent4.get())
    tmpdY = float(ent7.get())
    tmpdX = float(ent6.get())
    if ((tmpXmin >= tmpXmax) or
        (tmpYmin >= tmpYmax) or
        (tmpdX <= 0)):
        showwarning(title = "Ошибка задания границ",
                    message = "Должны выполняться "
                              " неравенства:\n"
                              "Xmax > Xmin;\n"
                              "Ymax > Ymin;\n"
                              "Шаг меток > 2")
    else:
        Xmax = tmpXmax
        Xmin = tmpXmin
        Ymax = tmpYmax
        Ymin = tmpYmin
        dX = tmpdX
        dY = tmpdY
```

```
        Kx = MaxX / abs((Xmax - Xmin))
        Ky = MaxY / abs((Ymax - Ymin))
def SetMark(a, b, LrBt = 1):
    '''Нанесение меток'''
    ax_XY = []
    if LrBt:  # слева и справа
        ax_XY.append((a, b))
        ax_XY.append((a + 10, b))
    else:     # внизу и вверху
        ax_XY.append((a,b))
        ax_XY.append((a, b - 10))
    cv.create_line(ax_XY, fill='black', width = 2)
def plotXY():
    ''' Рисуем координатные линейки'''
# Прямоугольник
    ax_XY = []
    ax_XY.append((5, 5))
    ax_XY.append((MaxX - 5, MaxY - 5))
    cv.create_rectangle(ax_XY, fill="white",
                        outline = "green", width = 2)
#
# Разметка левой и правой сторон
    y = Ymin
    y_pix = MaxY
    flg = False
    while y < Ymax:
        textY = str(round(y, 2))  # Текст метки
        SetMark(0, y_pix, 1)      # Слева
        if flg:   # Надписываем каждую вторую метку
            cv.create_text(15, y_pix, text = textY,
                           anchor = W)
        SetMark(MaxX-10, y_pix, 1) # Справа
        if flg:
            cv.create_text(MaxX - 15, y_pix,
                           text = textY, anchor = E)
        y += dX                    # Ед. пользователя
        y_pix -= dX * Ky           # Ед. в пикселах
        flg = not flg              # Разметка через раз
#
# Разметка сверху и снизу
    x = Xmin
    x_pix = 0
    flg = False
    while x < Xmax:
```

```
        textX = str(round(x, 2))  # Текст метки
        SetMark(x_pix, 0, 0)      # Вверху
        if flg:
            cv.create_text(x_pix, 15, text = textX,
                           anchor = N)
        SetMark(x_pix, MaxY, 0)   # Внизу
        if flg:
            cv.create_text(x_pix, MaxY - 15,
                           text = textX, anchor = S)
        x += dX
        x_pix += dX*Kx
        flg = not flg

def Draw(event):
    '''Подготовка полотна и вызов
       функций для рисования'''
    cv.delete("all") # Очистка полотна
    GetData()        # Получить данные
    plotXY()         # Нарисовать координатные линейки
    Fdraw(Sin2xDevx, 'blue') # Нарисовать функцию 1
    Fdraw(Sin2xOnX, 'red')   # Нарисовать функцию 2
    print('Рисуем')          # Тестовое сообщение

def Fdraw(func, color):
    '''Получение значений функции
       Преобразование в пикселы
       Рисование на полотне'''
    Lxy = func() # Значения функции в ед. пользователя
    Lpix = []    # Значения функции в пикселах
    for xy in Lxy:
        x = Kx * (xy[0] - Xmin)
        if xy[1] != None:
            y = MaxY - Ky * (xy[1] - Ymin)
        else:
            y = 0
        Lpix.append((x,y))
    cv.create_line(Lpix, fill = color)

def Final(event):
    ''' Завершаем работу '''
    window_deleted()

def window_deleted():
    ''' Завершаем работу по [X]'''
    if askyesno("Выход", "Завершить работу?"):
        root.destroy()

def showXY(event):
```

```
    global ID1, ID2
    x = event.x; y = event.y
    ent0.delete(0,END)
    ent1.delete(0,END)
    ent0.insert(0, str(round(Xmin + x/Kx, 2)))
    ent1.insert(0, str(round(Ymin + (MaxY - y)/Ky, 2)))
    cv.delete(ID1)
    cv.delete(ID2)
    ID1 = cv.create_line(0,y,MaxX,y, dash = (3,5))
    ID2 = cv.create_line(x,0,x,MaxY, dash = (3,5))
#
root = Tk()
root.title("Графика")
# Обработчик закрытия окна. Нажата кнопка [X]
root.protocol('WM_DELETE_WINDOW', window_deleted)
root.resizable(False, False)   # Не меняем размер окна
#
Kp = 0.7 # 70% от дисплея
# Начальные параметры полотна. Ед. изм. - пикселы
MaxX = root.winfo_screenwidth() * Kp
MaxY = root.winfo_screenheight() * Kp
#
cv = Canvas(root, width = MaxX,
            height = MaxY, bg = "white")
cv.grid(row = 0, columnspan = 9)
cv.bind('<Button-1>', showXY) # Клик на полотне
#
# Окно графика. Ед. изм. - пользовательские
Xmin = -10.0; Xmax = 10.0
Ymin = -5.0; Ymax = 5.0
Xmid = 0; Ymid = 0 # Положение начала координат
#
# Смещение и шаг аргумента. Ед. изм. - пользовательские
dY = 1.0   # смещение второго графика
dX = 1.0   # шаг меток на координатных линейках
# Идентификаторы курсоров, рисуемых по клику на полотне
ID1 = 0; ID2 = 0
# Масштабные коэфициены: Пиксел / пользов.ед.
Kx = MaxX / abs((Xmax - Xmin))
Ky = MaxY / abs((Ymax - Ymin))
#
lba0 = Label(root, text = "X:", width = 10,
             fg = "blue", font = "Ubutu, 12")
lba0.grid(row = 1, column = 0, sticky='e')
ent0 = Entry(root, width = 5, font = "Ubuntu, 12")
```

```
ent0.grid(row = 1, column = 1, sticky='w')
ent0.insert(0, 0)
lba1 = Label(root, text = "Y:", width = 10,
             fg = "blue", font = "Ubutu, 12")
lba1.grid(row = 2, column = 0, sticky='e')
ent1 = Entry(root, width = 5, font = "Ubuntu, 12")
ent1.grid(row = 2, column = 1, sticky='w')
ent1.insert(0, 0)

lba2 = Label(root, text = "Xmin:", width = 10,
             fg = "blue", font = "Ubutu, 12")
lba2.grid(row = 1, column = 2, sticky='e')
ent2 = Entry(root, width = 5, font = "Ubuntu, 12")
ent2.grid(row = 1, column = 3)
ent2.insert(0, Xmin)

lba3 = Label(root, text = "Xmax:", width = 10,
             fg = "blue", font = "Ubutu, 12")
lba3.grid(row = 1, column = 4, sticky='e')
ent3 = Entry(root, width = 5, font = "Ubuntu, 12")
ent3.grid(row = 1, column = 5)
ent3.insert(0, Xmax)

lba4 = Label(root, text = "Ymin:", width = 10,
             fg = "blue", font = "Ubutu, 12")
lba4.grid(row = 2, column = 2, sticky='e')
ent4 = Entry(root, width = 5, font = "Ubuntu, 12")
ent4.grid(row = 2, column = 3)
ent4.insert(0, Ymin)

lba5 = Label(root, text = "Ymax:", width = 10,
             fg = "blue", font = "Ubutu, 12")
lba5.grid(row = 2, column = 4, sticky='e')
ent5 = Entry(root, width = 5, font = "Ubuntu, 12")
ent5.grid(row = 2, column = 5)
ent5.insert(0, Ymax)

lba6 = Label(root, text = "Шаг меток:", width = 10,
             fg = "blue", font = "Ubutu, 12")
lba6.grid(row = 1, column = 6, sticky='e')
ent6 = Entry(root, width = 5, font = "Ubuntu, 12")
ent6.grid(row = 1, column = 7)
ent6.insert(0, dX)

lba7 = Label(root, text = "Смещение:", width = 10,
```

```
                fg = "blue", font = "Ubutu, 12")
lba7.grid(row = 2, column = 6, sticky='e')
ent7 = Entry(root, width = 5, font = "Ubuntu, 12")
ent7.grid(row = 2, column = 7)
ent7.insert(0, dY)

btn1 = Button(root, width = 20, bg = "#ccc",
              text = "Рисовать")
btn1.grid(row = 1, column = 8)
btn1.bind("<Button-1>", Draw)

btn2 = Button(root, width = 20, bg = "#ccc",
              text = "Выход")
btn2.grid(row = 2, column = 8)
btn2.bind("<Button-1>", Final)

root.mainloop()  # Цикл ожидания событий
```

Результат работы скрипта

На рис.8.5 представлена реализованная форма с графиками.

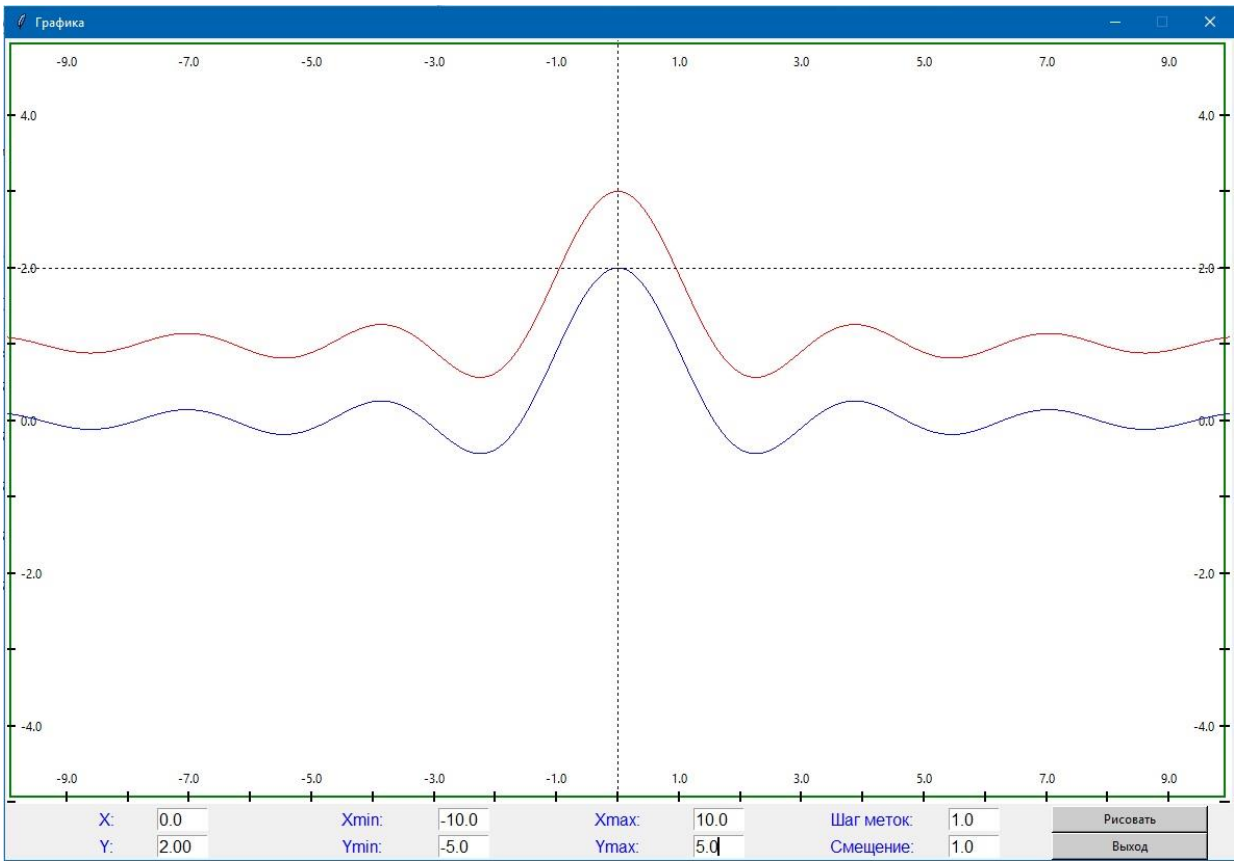


Рис.8.5. – Результат работы скрипта

Задание к лабораторной работе №8
«GUI, модуль Tkinter»

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 `Хнач` до `Хкон` 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.

В вариантах с 12 по 20 требуется построить графическое изображение в виде секторных или столбцовых диаграмм. Для решения этих задач необходимо познакомиться с виджетами модуля Tkinter полнее, чем это изложено выше.

Вариант 1

$$y(x) = 2 \cdot \sum_{n=0}^{\infty} \frac{1}{(2 \cdot n + 1) \cdot x^{2 \cdot n + 1}} = 2 \cdot \left(\frac{1}{x} + \frac{1}{3 \cdot x^3} + \frac{1}{5 \cdot x^5} + \ldots \right), \qquad |x| > 1;$$

$$z(x) = \ln \frac{x+1}{x-1} + b\,.$$

Вариант 2

$$y(x) = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^n}{n!} = \left(1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \ldots \right), \qquad |x| < \infty;$$

$$z(x) = e^{-x} + b;$$

Вариант 3

$$y(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!} = \left(1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots \right), \qquad |x| < \infty$$

$$z(x) = e^x + b\,.$$

Вариант 4

$$y(x) = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2 \cdot n + 1) \cdot x^{2 \cdot n + 1}} = \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3 \cdot x^3} - \frac{1}{5 \cdot x^5} + \ldots, \qquad x > 1;$$

$$z(x) = arctgx + b\,.$$

Вариант 5

$$y(x) = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot x^{2 \cdot n + 1}}{(2 \cdot n + 1)} = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots, \qquad |x| <= 1$$

$$z(x) = arctgx + b\,.$$

Вариант 6

$$y(x)=2\cdot\sum_{n=0}^{\infty}\frac{1}{(2\cdot n+1)\cdot x^{2\cdot n+1}}=2\cdot\left(\frac{1}{x}+\frac{1}{3\cdot x^3}+\frac{1}{5\cdot x^5}+\ldots\right),\qquad |x|>1;$$

$$z(x)=Arthx+b.$$

Вариант 7

$$y(x)=-\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2\cdot n+1)\cdot x^{2\cdot n+1}}=-\frac{\pi}{2}-\frac{1}{x}+\frac{1}{3\cdot x^3}-\frac{1}{5\cdot x^5}+\ldots,\qquad x<-1$$

$$z(x)=arctgx+b.$$

Вариант 8

$$y(x)=\sum_{n=0}^{\infty}\frac{(-1)^n\cdot x^{2\cdot n}}{n!}=1-x^2+\frac{x^4}{2!}-\frac{x^6}{3!}+\frac{x^8}{4!}-\ldots,\qquad |x|<\infty$$

$$z(x)=e^{-x^2}+b.$$

Вариант 9

$$y(x)=\sum_{n=0}^{\infty}\frac{(-1)^n\cdot x^n}{(2\cdot n)!}=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6}+\ldots,\qquad |x|<\infty$$

$$z(x)=\cos x+b.$$

Вариант 10

$$y(x)=\sum_{n=0}^{\infty}\frac{(-1)^n\cdot x^{2\cdot n}}{(2\cdot n+1)!}=1-\frac{x^2}{3!}+\frac{x^4}{5!}-\frac{x^6}{7!}+\ldots,\qquad |x|<\infty$$

$$z(x)=\frac{\sin x}{x}+b.$$

Вариант 11

$$y(x)=2\cdot\sum_{n=0}^{\infty}\frac{(x-1)^{2\cdot n+1}}{(2\cdot n+1)\cdot(x+1)^{2\cdot n+1}}=2\cdot\left(\frac{x-1}{x+1}+\frac{(x-1)^3}{3\cdot(x+1)^3}+\frac{(x-1)^5}{5\cdot(x+1)^5}+\ldots\right),\qquad x>0;$$

$$z(x)=\ln x+b.$$

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 Продаж 1996 г. тыс. шт.	Доля Рынка 1996 г., %	Объем Продаж 1995 г. тыс. шт.	Доля Рынка 1995 г., %	Рост 95/96, %
1	Compaq	7 036	10,3	5 757	9,8	22
2	IBM	6 081	8,9	4 785	8,1	27
3	Packard	4 247	6,2	4 392	7,5	-3
	Bell, NEC	3 587	5,2	4 627	7,9	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 WinPython.

Версии компонент: Python вер: 3.5.4, Tkinter вер: 8.6.4, IDLE вер: 3.5.4.

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. Более того, они написаны так, чтобы не 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.

PS: Это учебно-методическое пособие вместе с дополнительным материалом предполагается распространять в электронном виде. О месте размещения и способе получения электронной версии следует обратиться к авторам.

ПРИЛОЖЕНИЕ 1

Таблица 1. – Функции и методы в Python 3

Константы (Math)	**Описание и пример использования**
pi	Возвращает число π. `math.pi => 3.141592653589793`
e	Возвращает число e. `math.e => 2.718281828459045`
Функции (Sys)	

int([<Объект>[,<Система счисления>]])	Преобразует объект в целое число. Второй параметр указывает систему счисления преобразуемого числа. По умолчанию используется десятичная система счисления.
	`>>>int(7.5), int('71',10), int('0o71',8)`*)
	`(7, 71, 57)`
	`>>>int("0xA",16)`
	`10`
	`>>>int(), int("0b11111111",2)`
	`(0, 255)`

float([<Число или строка>])	Преобразует целое число или строку в вещественное число.
	`>>>float(7), float("7.1"), float("12.")`
	`(7.0, 7.1, 12.0)`
	`>>> float("inf"), float("-inf"), float("nan")`
	`(inf, inf, nan)`

bin(<Число>)	Преобразует десятичное число в двоичное. Возвращает строковое представление числа.
	`>>>bin(255), bin(1), bin(-45)`
	`('0b11111111', '0b1', '-0b101101')`

oct(<Число>)	Преобразует десятичное число в восьмеричное. Возвращает строковое представление числа.
	`>>>oct(7), oct(8), ovt(64)`
	`('0o7', '0o10', '0o100')`

hex(<Число>)	Преобразует десятичное число в шестнадцатеричное. Возвращает строковое представление числа.
	`>>>hex(10), hex(16), hex(255)`
	`('0xa', '0x10', '0xff')`

round(<Число>[, Кол-во знаков после точки])	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.
	`>>>round(1.49), round(1.50), round(1.51)`
	`(1, 2, 2)`
	`>>>round(2.49), round(2.50), round(2.51)`
	`(2, 2, 3)`
	Второй параметр определяет число цифр после запятой. В этом случае округление выполняется по правилу: отбрасываемая часть

*) 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)

	меньше 0.5 – округление вниз;
	больше или равно 0.5 – округление вверх.
	`>>>round(2.449,1),round(2.450,1),round(2.451, 1)`
	`(2.49, 2.5, 2.5)`

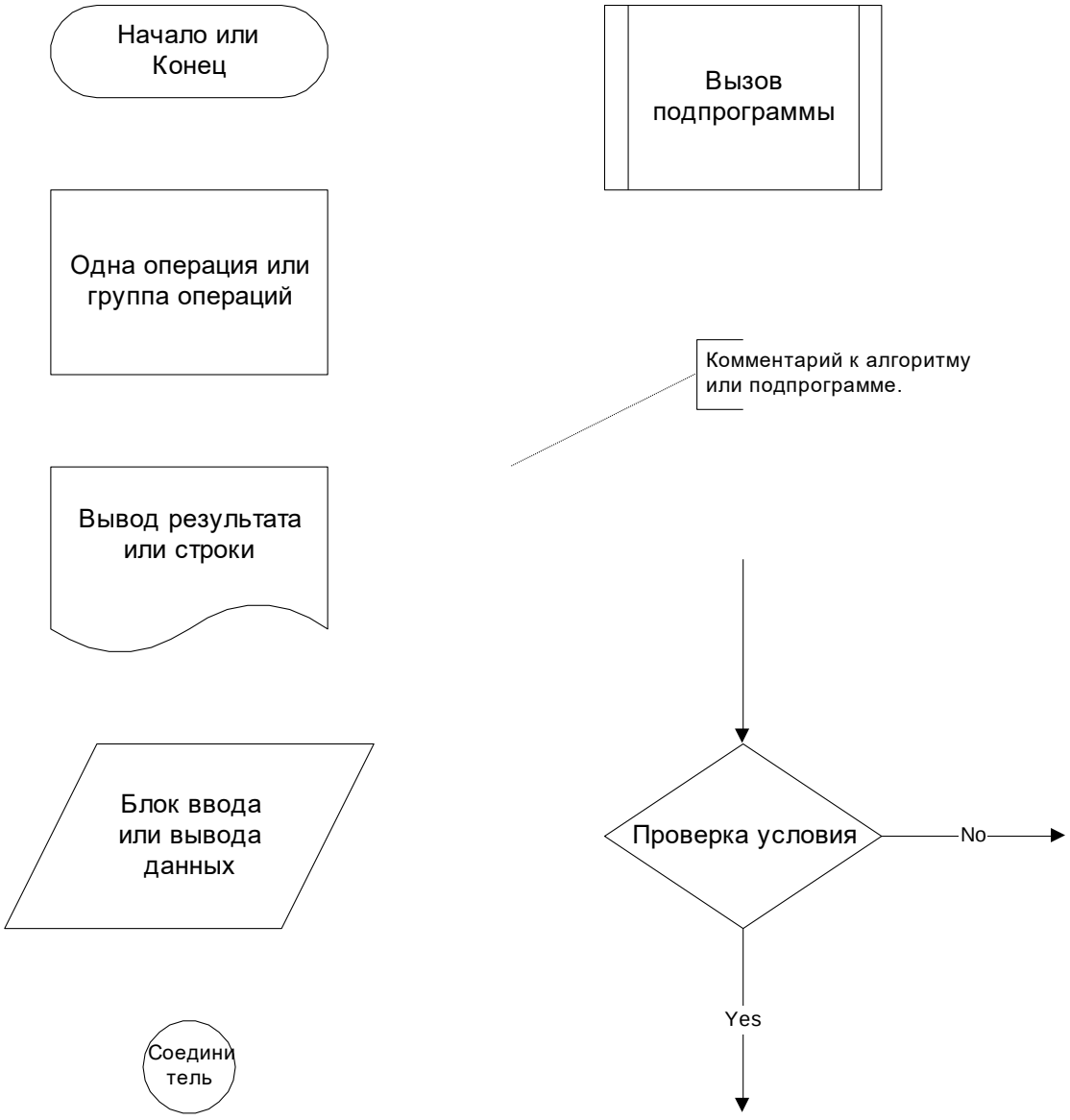
abs(<Число>)	Возвращает абсолютное значение.
pow(<Число>, <Степень>, [,<Делитель>])	Возвращает число, возведенное в степень. Степень может быть вещественного типа. Если указан третий параметр, то возвращается остаток от деления результата на значение этого параметра. `>>>pow(3, 3), pow(3, 3, 2), pow(3, 3.25)` (27, 1, 35.533998349717294)
max(<Список чисел через запятую>)	Возвращается максимальное число из списка.
min(<Список чисел через запятую>)	Возвращается минимальное число из списка.

sum(<Последовательность> [, <Начальное значение>)	Возвращает сумму значений элементов последовательности (списка, кортежа) плюс начальное значение.
	`>>>sum([1, 2, 3], 7), sum((1, 2, 3))`
	`(13, 6)`

divmod(x, y)	Возвращает кортеж из двух значений: целая часть и остаток от деления: `x // y` и `x % y`.
Функции (Math)	
sin(x), cos(x), tan(x)	Стандартные тригонометрические функции. Угол задается в радианах.
asin(x), acos(x), atan(x)	Обратные тригонометрические функции
degrees(x)	Преобразование радиан в градусы
radians(x)	Преобразование градусов в радианы
exp(x)	Экспонента
log(<Число>[, <База>])	Логарифм числа по основанию <База>. Если база не указана, то вычисляется натуральный логарифм.
log10(x), log2(x)	Десятичный и двоичный логарифмы (по базе 10 и 2 соответственно)
sqrt(x)	Квадратный корень
ceil(x)	Округление до ближайшего большего целого
floor(x)	Округление до ближайшего меньшего целого
fabs(x)	Возвращает абсолютное значение. тоже, что и `abs(x)`
fmod(x, y)	Остаток от деления. Тоже, что и x % y.
factorial(n)	Факториал числа.
fsum(<список чисел>)	Возвращает точную сумму чисел из заданного списка. >>>sum([.1, .1, .1, .1, .1, .1, .1, .1, .1, .1]) 0.9999999999999999 >>>fsum([.1, .1, .1, .1, .1, .1, .1, .1, .1, .1]) 1.0

ПРИЛОЖЕНИЕ 2

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…

Список литературы

1. Много интересных и полезных новостей из мира Python: https://pythondigest.ru/
2. Игра для обучения программированию: https://checkio.org
3. Online IDE: https://repl.it
4. Визуализатор online на Python: http://pythontutor.com/visualize.html#mode=edit
5. Задания по Python на основе рейтинга: https://www.hackerrank.com
6. Видео лекции «Программирование на языке Python для сбора и анализа данных»
7. Курс «Программирование на Python (Институт биоинформатики)»
8. Курс «Python: основы и применение» (Институт биоинформатики)
9. Видео лекции «Python 3 Basics Tutorial Series»
10. Курс Программирование на Python от Mail.Ru Group