ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УПРАВЛЕНИЕ ЦИФРОВЫХ ОБРАЗОВАТЕЛЬНЫХ ТЕХНОЛОГИЙ
ПИ (филиал) ДГТУ в г. Таганроге

ЦМК «Прикладная информатика»

Практикум

по дисциплине

«Основы алгоритмизации и программирования»

Автор
Полищук М.С.

Ростов-на-Дону, 2026

Аннотация

Практикум по дисциплине «Основы алгоритмизации и программирования» предназначен для студентов очной формы обучения по специальностям 09.02.08 «Интеллектуальные интегрированные системы»; 09.02.11 «Разработка и управление программным обеспечением»; 09.02.12 «Техническая эксплуатация и сопровождение информационных систем».

Автор

Преподаватель ЦМК «Прикладная информатика»
Полищук М.С.

Оглавление

Введение.....5
Практическое занятие № 1 «Составление блок-схем алгоритмов для решения задач» 7
Практическое занятие № 2 «Операторы ввода - вывода. Написание программ линейной структуры» 12
Практическое занятие № 3 «Написание программ алгоритмов ветвления. Вложенное ветвление».....16
Практическое занятие № 4 «Написание программ циклических алгоритмов» 19
Практическое занятие № 5 «Написание программ алгоритмов смешанной структуры».....22
Практическое занятие № 6 «Трансляция алгоритмов с алгоритмического языка на язык Python. Написание программы по блок – схеме» 26
Практическое занятие № 7 «Формирование сложных условий с использованием логических операций. Вычисление значений логических выражений».....29
Практическое занятие № 8 «Функции и процедуры. Локальные, глобальные переменные: область видимости» 41
Практическое занятие № 9 «Решение задач обработки списков» 43
Практическое занятие № 10 «Обработка вложенных списков» 48
Практическое занятие № 11 «Решение задач со множествами. Операции со строковым типом данных».....50
Практическое занятие № 12 «Программы работы с файлами»52
Практическое занятие № 13 «Алгоритмы вычисления рекуррентных последовательностей».....54
Практическое занятие № 14 «Рекурсивные алгоритмы» 55
Практическое занятие № 15 «Создание рисунков с помощью программного кода. Узоры. Орнаменты. Фракталы» 57
Практическое занятие № 16 «Построение графиков математических функций. Графика, модуль Tkinter» 59

Практическое занятие № 17 «Технологии модульного программирования»..... 62
Практическое занятие № 18 «Разработка приложений с графическим интерфейсом. Элементы окна» 65
Практическое занятие № 19 «Разработка приложений с графическим интерфейсом. Элементы окна» 70
Практическое занятие № 20 «Разработка приложений с графическим интерфейсом. Компьютерная графика» 73
Практическое занятие № 21 «Разработка приложений с графическим интерфейсом. Компьютерная графика» 76
Практическое занятие № 22 «Разработка приложений с графическим интерфейсом. Многооконный интерфейс» 79
Перечень использованных информационных ресурсов . 84

Введение

В учебно-методических указаниях по дисциплине «Основы алгоритмизации и программирования» изложены сведения, необходимые для успешного выполнения практических занятий по данному курсу. Описан процесс работы с инструментарием, применяемым в практических работах, представлен ряд типичных задач и подходы к их решению. Практические занятия посвящены знакомству обучающихся с текстовыми редакторами, графическими редакторами, электронными таблицами, базами данных и компьютерными сетями.

Цель настоящего пособия – помочь обучающимся при выполнении практических работ, выполняемых для закрепления знаний по теоретическим основам и получения практических навыков работы на компьютерах.

Обучающийся должен знать основы составления программы для реализации методов иалгоритмов обработки различных данных; методы и алгоритмы обработки различных данных, решения типовых задач; основные современные методы и средства разработки алгоритмов и программ; формальные методы, технологии и инструменты разработки программного продукта; основные приемы структурного программирования; способы записи алгоритмов на процедурном языке программирования; сферы применения рассматриваемых алгоритмов и методов

Обучающийся должен уметь использовать технологию работы на персональной ЭВМ; использовать основные современные методы и средства разработки алгоритмов и программ; использовать основные приемы структурного программирования; использовать способы записи алгоритмов на процедурном языке программирования.

Общие положения

Практические занятия выполняются каждым обучающимся самостоятельно в полном объеме и согласно содержанию методических указаний. Перед выполнением обучающийся должен отчитаться перед преподавателем за выполнение предыдущего занятия (сдать отчет). Обучающийся должен на уровне понимания и

воспроизведения предварительно усвоить необходимую для выполнения практических занятий теоретическую и информацию.

Обучающийся, получивший положительную оценку и сдавший отчет по предыдущему практическому занятию, допускается к выполнению следующему занятию. Обучающийся, пропустивший практическое занятие по уважительной либо неуважительной причине, закрывает задолженность в процессе выполнения последующих практических занятий.

Форма отчета: титульный лист; введение (цель и задачи); выполнение; заключение

Практическое занятие № 1 «Составление блок-схем алгоритмов для решения задач»

Цель работы Получение практических навыков составления блок- схем алгоритмов

Теоретическая часть

Алгоритм - это точное предписание исполнителю совершить указанную последовательность действий для получения решения задачи за конечное число шагов. Название «алгоритм» произошло от латинской формы имени величайшего среднеазиатского математика Мухаммеда ибн Муса ал-Хорезми (Alhorithmi), жившего в 783 -850 гг. В своей книге «Об индийском счете» он изложил правила записи натуральных чисел с помощью арабских цифр и правила действий над ними «столбиком», знакомые теперь каждом школьнику. В XII веке эта книга была переведена на латынь и получила широкое распространение в Европе. Понятие алгоритма является не только одним из главных понятий математики, но одним из главных понятий современной пауки. Более того, с наступлением эры информатики алгоритмы становятся одним из важнейших факторов цивилизации.

Формы записи алгоритма. Алгоритм, как последовательность шагов или инструкций, может быть представлен в различных формах. На практике наиболее распространены следующие формы представления алгоритмов:

1.Словесная (запись на естественном языке);

2.Графическая (изображения из графических символов);

3.Псевдокоды (полуформализованные описания алгоритмов на условном алгоритмическом языке, включающие в себя как элементы языка программирования, так и фразы естественного языка, общепринятые математические обозначения и др.);

4.Программная (тексты на языках программирования).

Рабочее задание

Вариант 1

1. Приведите пример исполнителя, его системы команд, среды в которой он действует и ситуаций отказов.

2. Перечислите какого типа может быть величина, соответствующая результату вычислений в следующих задачах: - вычисление

кубического корня; - вычисление количества учащихся в университете.

3. Составьте алгоритм определения наименьшего из четырех чисел.

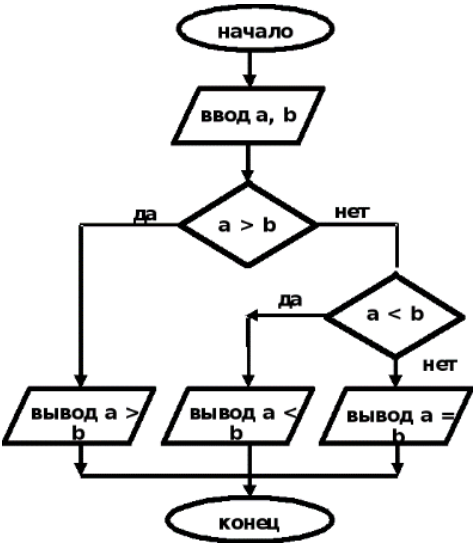


Рисунок 1.1 – Блок-схема определения наименьшего из 4 чисел

4.Определить результат работы алгоритма при различных значениях исходных данных (заполнить таблицу)

	a	b	Результат
Набор 1	23,27	27,23	
Набор 2	402	4002	
Набор 3	0,327	0,30327	

5. Приведите пример задачи (алгоритма), в котором есть входные, промежуточные и выходные данные. Какая это будет информация?

6. Даны значения пяти переменных: a=2, b=4, c=7, d=9, e=1. Составьте линейный алгоритм обмена значениями между данными переменными так, чтобы на выходе алгоритма все переменные изменили свое значение, и все значения переменных были разными.

7. Запишите формулы, как они будут выглядеть в коде

программы:

2x
56:23
$\frac{3}{4}$
$\frac{3x-1}{x+4}$
$a^{0,5}$
$a^{1,5}$
y^2
cos x
$\sin^2 x$

Рисунок 1.2 – Математическая запись

8. Составьте словесно – формульную запись алгоритма, изображенного в виде структурограммы.

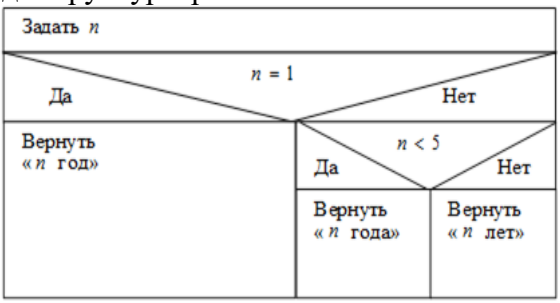


Рисунок 1.3 - Вид структурограммы

9. В чем отличие семантических ошибок в программе от синтаксических?

10. Чем определяется объем памяти, выделяемой для хранения величины? Пример.

Вариант 2

1. Приведите пример исполнителя, его системы команд, среды в которой он действует и ситуаций отказов.

2. Перечислите какого типа может быть величина, соответствующая результату вычислений в следующих задачах:
- вычисление определенного интеграла;
- вычисление количества книг в библиотеке.

3. Составьте алгоритм определения наименьшего из четырех чисел.

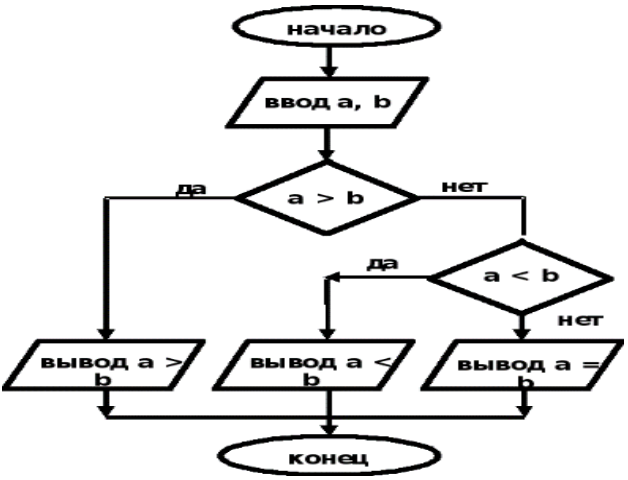


Рисунок 1.4 - Блок-схема определения наименьшего из 4 чисел

4. Определить результат работы алгоритма при различных значениях исходных данных (заполнить таблицу)

	a	b	Результат
Набор 1	23,27	27,23	
Набор 2	402	4002	
Набор 3	0,327	0,30327	

5. Приведите пример задачи (алгоритма), в котором есть входные, промежуточные и выходные данные. Какая это будет информация?
6. Даны значения пяти переменных: a=3, f=4, g=8, d=5, j=7. Составьте линейный алгоритм обмена значениями между данными переменными так, чтобы на выходе алгоритма все переменные изменили свое значение, и все значения переменных были разными.

7. Запишите формулы, как они будут выглядеть в коде программы:

37(25+87,5)-17(4,6+1,9)
$\frac{ab}{c}+d^4$
$\frac{a+2b-3c}{5a+4}$
$\frac{a+b}{a-b}+\frac{ab}{3{,}14}$

Рисунок 1.5 – Вид математической записи

8. Составьте словесно – формульную запись алгоритма, изображенного в виде структурограммы.

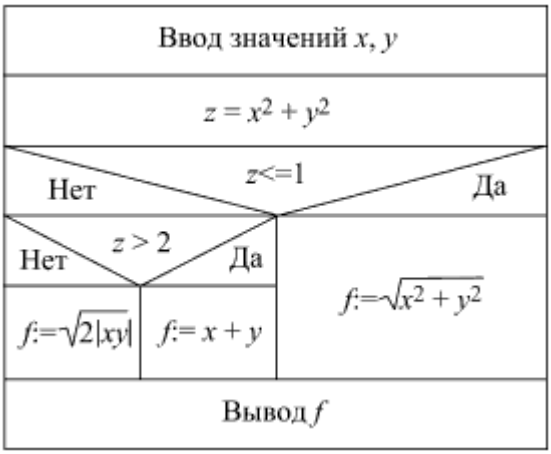


Рисунок 1.6 - Вид структурограммы

9. В чем отличие структурограммы от блок-схемы?

10. От чего зависит перечень допустимых операций, применяемых к величине? Пример.

Время работы: 2 часа.

Практическое занятие № 2 «Операторы ввода - вывода. Написание программ линейной структуры»

Цель работы Получение практических навыков написания программ линейной структуры

Теоретическая часть

Любая Python-программа состоит из последовательности допустимых символов,записанных в определенном порядке и по определенным правилам.

Программа включает в себя:

- комментарии;
- команды;
- знаки пунктуации;
- идентификаторы;
- ключевые слова.

Комментарии в Python обозначаются предваряющим их символом # и продолжаются до конца строки (т.е. в Python все комментарии являются однострочными), при этом не допускается использование перед символом #кавычек.

Знаки пунктуации. *В алфавит Python входит достаточное количество знаков пунктуации, которые используются для различных целей. Например, знаки "+" или " *" могут использоваться для сложения и умножения, а знак запятой "," - для разделения параметров функций.*

Идентификаторы. *Идентификаторы в Python это имена, используемые для обозначенияпеременной, функции, класса, модуля или другого объекта.*

Ключевые слова. *Некоторые слова имеют в Python специальное назначение и представляютсобой управляющие конструкции языка.*

Ключевые слова в Python: ['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

Типы данных

1.None (неопределенное значение переменной)
2.Логические переменные (Boolean Type)

3.Числа (Numeric Type):

- int – целое число
- float – число с плавающей точкой
- complex – комплексное число

4.Списки (Sequence Type)

- list – список
- tuple – кортеж

5.range – диапазон Строки (Text Sequence *Type*)

- str

Ввод и вывод данных. Ввод данных осуществляется при помощи команды **input (**список ввода**)**:

```
a = input()
print(a)
```

В скобках функции можно указать сообщение - комментарий к вводимым данным:

```
a = input ("Введите количество: ")
```

Команда input () по умолчанию воспринимает входные данные как строку символов. Поэтому, чтобы ввести целочисленное значение, следует указать типданных int ():

```
a = int (input ())
```

Для ввода вещественных чисел применяется команда

```
=float(input())
```

Вывод данных осуществляется при помощи команды **print** (список вывода):

```
a = 1
b = 2 print(a)
print (a + b)
print ('сумма = ', a + b)
```

Существует возможность записи команд в одну строку, разделяя их через, однако не следует часто использовать такой способ, это снижает удобочитаемость:

```
a = 1; b = 2; print(a) print (a + b)
print ('сумма = ', a + b)
```

Для команды **print** может задаваться так называемый *сепаратор* — разделитель между элементами вывода:

x=2 y=5
print (x, "+", y, "=", x+y, sep = " ")

Результат отобразится с пробелами между элементами:
2 + 5 = 7

Пример. Напишите программу, которая запрашивала бы у пользователя:

- ФИО ("Ваши фамилия, имя, отчество?")
- возраст ("Сколько Вам лет?")
- место жительства ("Где вы живете?")

После этого выводила бы три строки: "Ваше имя"
"Ваш возраст""Вы живете в"

Решение

Текст программы:

```
a=input('Введите ваши фамилию, имя, отчество ')
b=input('Сколько вам лет? ')
c=input('Где вы живёте? ')
print('Ваше имя ',a)
print('Ваш возраст ',b)
print('Вы живете в ',c)
```

Рисунок 2.1 - Пример программы на Python

```
Введите ваши фамилию, имя, отчество Иванов Иван Иванович
Сколько вам лет? 15
Где вы живёте? Уссурийск
Ваше имя  Иванов Иван Иванович
Ваш возраст  15
Вы живете в  Уссурийск
```

Рисунок 2.2 – Результат выполнения программы

Рабочее задание

1.Установите IDLE Python

https://www.python.org/ftp/python/3.8.5/python-3.8.5.exe

Можно работать в Python online.

2. Напишите программу, которая запрашивала бы у

пользователя:

- фамилия, имя ("Ваши фамилия, имя?")
- возраст ("Сколько Вам лет?")
- место учебы ("Где вы учитесь?")

После этого выводила бы три строки:

"Ваши фамилия, имя" *далее выводится введенная информация*
"Ваш возраст"*далее выводится введенная информация*
"Вы учитесь в" *далее выводится введенная информация*

3. Напишите программу, которая выводит на экран текст «I***like***Python» используя необязательный параметр оператора print - sep.

4. Напишите программу, которая считывает целое число, после чего на экран выводится следующее и предыдущее целое число с пояснительным текстом.

5. Напишите программу, которая находит полное число метров по заданному числу сантиметров. Использовать операцию деления нацело.

6. Написать программу решения задачи. Безумный титан Танос собрал все 6 камней бесконечности и намеревается уничтожить половину населения Вселенной по щелчку пальцев. При этом если население Вселенной является нечетным числом, то титан проявит милосердие и округлит количество выживших в большую сторону. Помогите Мстителям подсчитать количество выживших.

При написании программы использовать операцию деления нацело и операцию получения остатка от деления.

7. Напишите программу для пересчёта величины временного интервала, заданного в минутах, в величину, выраженную в часах и минутах.

```
90

90 мин - это 1 час 30 минут.
```

Рисунок 2.3 - Результат работы программы

8. Напишите программу, в которой рассчитывается сумма цифр положительного трёхзначного числа. Результат –сообщение с указанием суммы цифр в числе.

9. Результаты работы оформить отчетом. По каждой задаче листинг программы текстом и скриншот результата в виде изображения.

Время работы: 2 часа.

Практическое занятие № 3 «Написание программ алгоритмов ветвления. Вложенное ветвление»

Цель работы Получение практических навыков написания программ алгоритмов ветвления.

Теоретическая часть

Алгоритмы ветвящейся (разветвленной) структуры применяются, когда в зависимости от некоторого условия необходимо выполнить либо одно, либо другое действие. В блок-схемах разветвление алгоритма изображается с помощью блока «Решение». Существует два основных вида ветвящихся алгоритмов:

- два альтернативных варианта;
- многовариантный выбор (меню).

Для организации ветвления при многовариантном выборе необходимо изобразить на блок-схеме после блока «Решение» шину, позволяющую перейти к тому или иному варианту в зависимости от выбранного критерия.

Примеры задач, требующих использования ветвящихся алгоритмов:

- решение квадратного уравнения (при оценке дискриминанта необходимо ветвление);
- вычисление функции, заданной графически (при проверке диапазонов значений переменной).

Условный оператор ветвления if, if-else, if-elif-else

Оператор ветвления if в языке Python позволяет выполнить определенный набор инструкций взависимости от некоторого условия. Возможны следующие варианты использования.

Конструкция if

Синтаксис оператора if выглядит так:

if логическое выражение:

команда_1команда_2

...

команда_n

Синтаксис оператора if – else выглядит так:

if логическое выражение:

команда_1

команда_2

...

команда_n

else:

команда_1команда_2

...

команда_n

Конструкция if – elif – else

Для реализации выбора из нескольких альтернативможно использовать конструкцию if – elif – else.

Синтаксис оператора if – elif – else выглядит так:

if логическое выражение_1:

команда_1команда_2

...

команда_n

elif логическое выражение_2:

команда_1команда_2

...

команда_n

elif логическое выражение_3:

команда_1команда_2

...

команда_n

else:
    команда_1команда_2
    ...
    команда_n

Пример. Дано 3 числа. Найти минимальное среди них и вывести на экран.

```
#нахождение минимального из 3-х чисел
a=input('Введите целое число \n')
b=input('Введите целое число \n')
c=input('Введите целое число \n')
if a<b:
    if a<c:
        y=a
    else:
        y=c
else:
    if b<c:
        y=b
    else:
        y=c
print('Минимальное:',y)
```

Рисунок 3.1 – Решение задачи нахождения минимального из трех чисел

Рабочее задание

1.По номеру дня недели вывести название дня недели.

2.Написать программу определения максимального из пяти чисел. Без использования стандартных функций Python.

3.Написать программу, которая вычисляет арифметический квадратный корень. В программе необходимо проверить все исключения.

4.Даны три числа, которые пользователь вводит с клавиатуры. Написать программу, которая выводит эти числа в порядке возрастания.

5.Напишите программу, которая по 3 целым сторонам проверяет существование обычного треугольника и прямоугольного треугольника. Программа получает на вход три целых числа. Программа должна вывести два предложения: существует или не

существует треугольник; если треугольник существует, является ли он прямоугольным.

Входные данные	Выходные данные
3 4 5	Треугольник существует. Треугольник прямоугольный.
100 0 0	Треугольник не существует.
7 7 10	Треугольник существует. Треугольник не прямоугольный.

Рисунок 3.2 – Входные и выходные данные программы

Время работы: 2 часа.

Практическое занятие № 4 «Написание программ циклических алгоритмов»

Цель работы Получение практических навыков написания программ циклических алгоритмов

Теоретическая часть

В Python существуют два типа цикличных выражений:

- Цикл while
- Цикл for

Цикл while в Python

Инструкция while в Python повторяет указанный блок кода до тех пор,пока указанное в цикле логическое выражение будет оставаться истинным.

Синтаксис цикла while:

while логическое выражение:

команда 1
команда 2
...
команда n

После ключевого слова while указывается условное выражение, и покаэто выражение возвращает значение True, будет выполняться блок инструкций, который идет далее.

Все инструкции, которые относятся к циклу while, располагаются напоследующих строках и должны иметь отступ от начала строки (4 пробела).

```
#! Программа по вычислению факториала
number = int(input("Введите число: "))
i = 1
factorial = 1
while i <= number:
    factorial *= i
    i += 1
print("Факториал числа", number, "равен", factorial)
```

Рисунок 4.1 - Пример программы на Python

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
Введите число: 5
Факториал числа 5 равен 120
>>>
```

Рисунок 4.2 - Результат выполнения программы с использованием циклического оператора while

Цикл for в Python:

Цикл for в Python обладает способностью перебирать элементы любого комплексного типа данных (например, строки или списка).

Синтаксис цикла for:

for int **in** range ():
    команда 1
    команда 2
    ...
    команда n

Переменной int присваивается значение первого элемента функции range (), после чего выполняются команды. Затем

переменной int присваивается следующее по порядку значение и так далее до тех пор,пока не будут перебраны все элементы функции range ().

Функция range () является универсальной функцией Python для создания списков (list) содержащих арифметическую прогрессию. Чаще всего она используется в циклах for.

Range (старт, стоп, шаг) - так выглядит стандартный вызов функции range () в Python. По умолчанию старт равняется нулю, шаг единице.

Пример. Найти сумму n элементов следующего ряда чисел: 1 -0.5 0.25 -0.125 ...n. Количество элементов (n) вводится с клавиатуры. Вывести на экранкаждый член ряда и его сумму. Решить задачу, используя циклическую конструкцию for.

Решение:

В данном случае ряд чисел состоит из элементов, где каждый следующий меньше предыдущего в два раза по модулю и имеет обратный знак. Значит,чтобы получить следующий элемент, надо предыдущий разделить на -2.

Какой-либо переменной надо присвоить значение первого элемента ряда (вданном случае это 1). Далее в цикле добавлять ее значение к переменной, вкоторой накапливается сумма, после чего присваивать ей значение следующего элемента ряда, разделив текущее значение на -2. Цикл долженвыполняться n раз.

```
n=int(input('Введите количество элементов последоваетльности: '))
x=1
s=0
print(x)
for i in range(n):
    s+=x
    x/=-2
    print(x)
print('Сумма ряда:',s)
```

Рисунок 4.3 - Пример программы с циклом for

Рабочее задание

1.С помощью цикла for посчитать сумму чисел от 1 до 100 кратных трем.

2.Используя цикл «Пока» вывести на экран степени числа 5 до восьмой степени.
3.Банк принимает вклады под 9% годовых. На счет положена сумма 15000 рублей. Какая сумма будет на счету через 7 лет? Вывести на экран все начисления по годам за 7 лет.
4.Найти разницу A и B. A – сумма всех четных чисел от 1 до 100, B – сумма всех нечетных чисел от 1 до 100.
5.Найти наибольший общий делитель двух чисел. Числа вводятся с клавиатуры.
6.Вывести на экран таблицу умножения.

Время работы: 2 часа.

Практическое занятие № 5 «Написание программ алгоритмов смешанной структуры»

Цель работы Получение практических навыков написания программ алгоритмов смешанной структуры.

Теоретическая часть

Алгоритмы смешанной структуры объединяют в себе элементы различных типов алгоритмов, таких как линейные, ветвящиеся и циклические. Они позволяют эффективно решать сложные задачи, комбинируя различные подходы и стратегии для достижения оптимального решения.

При описании вложенных структур на языке программирования необходимо помнить, что группа операторов объединяется в блок с помощью отступов. При переводе математических выражений на язык программирования, формулы меняют свой вид, используются встроенные функции Python.

Пример

Напишите программу, которая бы вычисляла заданное арифметическое выражение при заданных переменных. Ввод переменных осуществляется с клавиатуры. Вывести результат с 2-мя знаками

$$Z = \frac{9\pi t + 10\cos(x)}{\sqrt{t} - |\sin(t)|} * e^{x}$$

после запятой.

x=10; t=1

example_math_var0.py - C:/Users/maxim/Desktop/Python/example_math_var0.py (3.7.1)

File Edit Format Run Options Window Help

```
import math
x=int(input("Введите переменную x:"))
t=int(input("Введите переменную t:"))
z=((9*math.pi*t+10*math.cos(x))/(math.sqrt(t)-math.fabs(math.sin(t))))*math.pow(math.e,x)
print("z = {0:.2f}".format(z))
```

Рисунок 5.1 – Пример решения задачи

Python 3.7.1 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.1 (v3.7.1:260ec2c36a,
1)] on win32
Type "help", "copyright", "credit
>>>
======== RESTART: C:/Users/maxim/
Введите переменную x:10
Введите переменную t:1
z = 2762685.71
>>>
```

Рисунок 5.2 – Результат работы программы

Рабочее задание

1.По заданным значениям x и y вычислить различные значения z.

1) $z = \dfrac{\min(x, y) + 0{,}5}{1 + \max^2(x, y)}$;

2) $z = \begin{cases} \max(x, y) & \text{при } x < 0, \\ \min(x, y) & \text{при } x \geq 0; \end{cases}$

3) $z = \begin{cases} \min(x, y) & \text{при } y \geq 0, \\ \max(x^2, y^2) & \text{при } y < 0. \end{cases}$

2.Даны три действительных числа (ввести с клавиатуры). выбрать из них те, которые принадлежат интервалу (1, 3).

3.Посчитать количество трехзначных чисел, содержащих цифру 7.

4.По заданной блок – схеме написать программу – алгоритм вычисления корней квадратного уравнения. Проанализировать работу алгоритма. Написать вывод, какой случай не учтен в данной схеме.

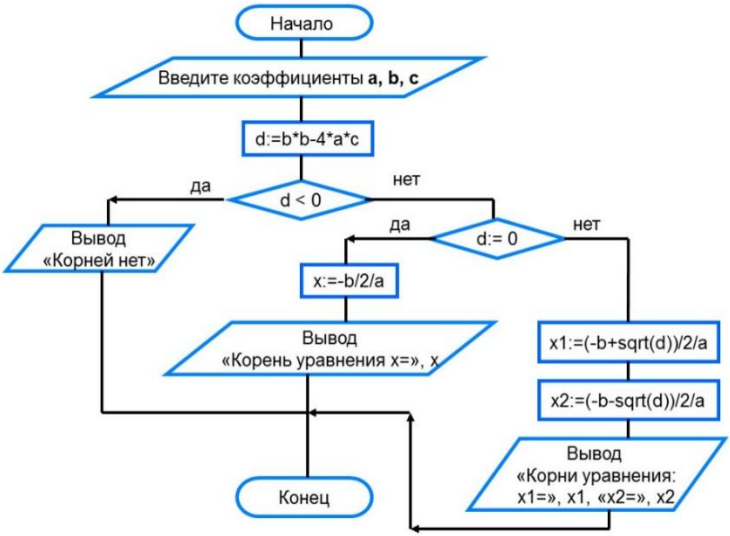


Рисунок 5.1 – Блок –схема алгоритма нахождения корней квадратного уравнения

5.По заданной блок – схеме написать программу вычисления значения z на Python.

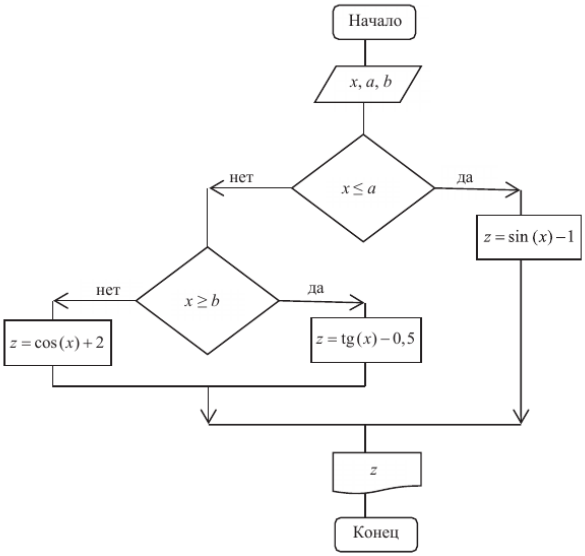


Рисунок 5.2 - Блок –схема алгоритма, по которой необходимо написать программу.

6.Вывести на экран буквы слова «программирование» исключая букву «о». Использовать оператор continue.

7.Вывести на экран буквы слова «программирование» до первой попавшейся буквы «м». Использовать оператор break.

8. Сложить все значения произведений в таблице умножения от 1 до 10. Результат вывести на экран.

9.С клавиатуры вводятся два числа. Найти наибольший общий делитель чисел без использования операций деления и умножения.

Время работы: 2 часа.

Практическое занятие № 6 «Трансляция алгоритмов с алгоритмического языка на язык Python. Написание программы по блок – схеме»

Цель работы Получение практических навыков написания программы по блок- схеме

Теоретическая часть

Псевдокод — это упрощенный язык программирования, который используется для описания алгоритмов. Он не имеет строгого синтаксиса, как настоящие языки программирования, и предназначен для понятного и наглядного описания шагов выполнения задачи. Псевдокод используется для планирования и проектирования алгоритмов до их реализации на конкретном языке программирования.

Школьный алгоритмический язык, также известный как "АЛГОРИТМИКА", является специальным обучающим языком, который используется в школьном курсе информатики для изучения алгоритмов и программирования. Этот язык обычно содержит ограниченный набор команд и конструкций, которые позволяют учащимся освоить основы алгоритмического мышления и программирования.

Школьный алгоритмический язык обычно предоставляет возможность описывать простейшие алгоритмы с использованием команд, таких как присваивание значений переменным, условные операторы (if-else), циклы (for, while), операции с числами и строками.

Рабочее задание

1.Написать программу по блок – схеме.

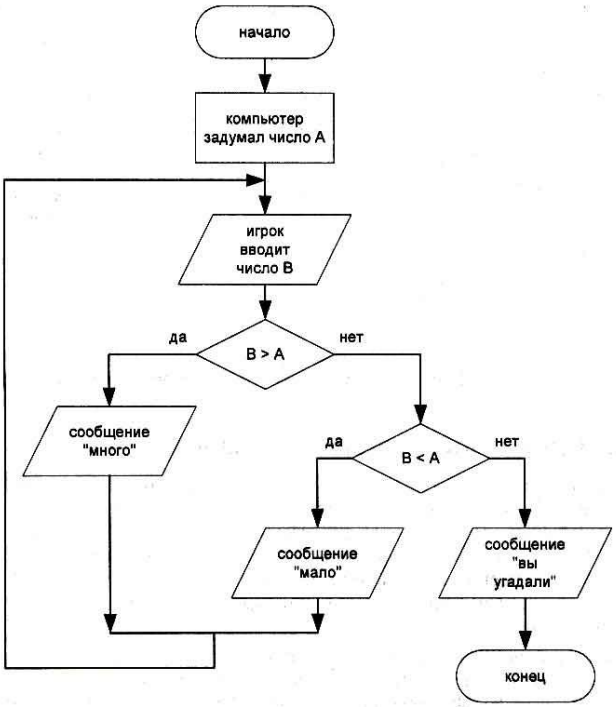


Рисунок 6.1 - Блок –схема алгоритма, по которой необходимо написать программу

2.Написать программу по алгоритму в виде псевдокода:

алг Сумма квадратов (арг цел n, рез цел S)дано | n > 0
надо | S = 1*1 + 2*2 + 3*3 + + n*nнач цел i
| ввод n; S:=0
| нц для i от 1 до n
| | S := S + i * i
| кц
| вывод "S = ",Sкон

Какую задачу решает данная программа?

3.Написать программу по алгоритму в виде псевдокода:

алг произвед a,f
нач
f=1;
a=1;
нц

пока a<=10
f=f*a;
a=a+1;
кц
вывод F
кон

Какую задачу решает данная программа?

4.Написать программу по алгоритму в виде псевдокода:

алг суммавещ a,s
нач
S=0;
A=1;
нц
пока a<=10
S=S+a;
a=a+1;
кц
вывод S
кон

Какую задачу решает данная программа?

5.Написать программу по блок – схеме.

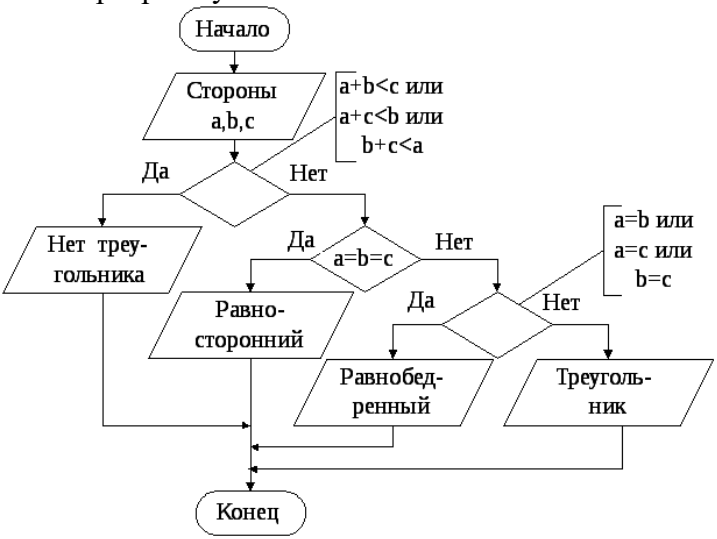


Рисунок 6.2 -Какую задачу решает данная программа?

6.Написать программу по блок – схеме. Примечание «:=» - оператор присваивания значения.

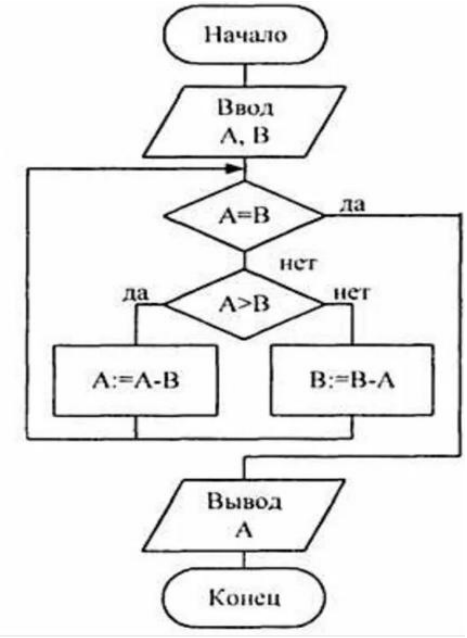


Рисунок 6.3 -Какую задачу решает данная программа?

Время работы: 2 часа.

Практическое занятие № 7 «Формирование сложных условий с использованием логических операций. Вычисление значений логических выражений»

Цель работы Получение практических навыков составления сложных условий и построения таблиц истинности логических выражений при написании программ.

Теоретическая часть

Практическое значение алгебры логики. В вычислительной технике булевы значения - это 0 и 1. Они представляют собой состояние ячейки памяти объемом в 1 бит или наличие/отсутствие напряжения в электрической схеме. Алгебра логики позволяет строить сложные электронные узлы, элементы которых работают согласно этой математической теории.

Логическое высказывание – это любое повествовательное предложение, в отношении которого можно однозначно сказать, истинно оно или ложно.

Простое высказывание, содержащее только одну мысль - логическая переменная. Обозначается прописными (большими) буквами латинского алфавита (например, A, B, C) и могут принимать лишь два значения ИСТИНА (1) и ЛОЖЬ (0). **Логические операции** – логические действия. К ним относятся:

1.Конъюнкция (логическое умножение - И) – обозначение «&» и «∧ ». Результат будет истинным тогда и только тогда, когда оба исходных высказывания истинны

A	B	A &B
0	0	0
0	1	0
1	0	0
1	1	1

2.Дизъюнкция (логическое сложение - ИЛИ) – обозначение «V». Результат будет истинным тогда, когда истинно хотя бы одно из высказываний.

A	B	AvB
0	0	0
0	1	1
1	0	1
1	1	1

3.Инверсия (логическое отрицание – НЕ) - обозначение «¬» и «¯» ($\bar{A}$). Результат будет истинным, если исходное высказывание ложно, и наоборот, ложным - если исходное высказывание истинно.

A	$\bar{A}$
0	1
0	0

4.Эквивалентность - обозначение «~». Результат будет истинным тогда, когда оба исходных высказывания либо истинны, либо ложны.

A	B	A ~ B
0	0	1
0	1	0
1	0	0
1	1	1

5.Импликация (логическое следование) - обозначение «→». Результат будет ложным только тогда, когда из истинного высказывания следует ложное.

A	B	A → B
0	0	1
0	1	1
1	0	0
1	1	1

При выполнении логических операций определен следующий порядок их выполнения: инверсия, конъюнкция, дизъюнкция. Для изменения указанного порядка

используются скобки.

Например, A v B & C →(A v B) & C

Каждое составное высказывание можно выразить в виде логического выражения (формулы), в которое входят логические переменные, обозначающие высказывания, и знаки логических операций, обозначающие логические функции.

Решение логических выражений записывают в виде **таблиц истинности** – таблиц, в которых по действиям показано, какие значения принимает логическое выражение при всех возможных наборах его переменных.

Для составления таблиц истинности необходимо:

- определить количество строк в таблице (2^n, n – количество переменных);
- определить количество столбцов в таблице (= количество логических переменных + количество логических операций);
- установить последовательность выполнения логических операций;
- построить таблицу, указывая названия столбцов и возможные наборы значений исходных логических переменных;
- заполнить таблицу истинности по столбцам.

Логические выражения, у которых последние столбцы таблиц истинности совпадают, называются равносильными – обозначение «=».

В алгебре высказываний можно проводить тождественные преобразования, заменяя одни высказывания равносильными им другими высказываниями, применяя следующие свойства логических операций:

1) коммутативность A & B = B & A
   A v B = B v A
2) закон идемпотентности A & A = A
   A v A = A
3) двойное отрицание
   $\bar{\bar{A}} = A$
4) сочетательные (ассоциативные) законы
   A v (B v C) = (A v B) v C = A v B v C

A & (B & C) = (A & B) & C = A & B & C

5) распределительные (дистрибутивные) законы A & (B v C) = (A & B) v (A &)
   A v (B & C) = (A v B) & (A v C)
6) поглощение
   A v (A & B) = A&склеивание
   (A & B) v ($\bar{A}$ & B) = B
   (A v B) & ($\bar{A}$ v B) = B
7) операции отрицания
   A & $\bar{A}$ = 0
   A v $\bar{A}$ = 1
8) операции с константами A & 1 = A, A v 1 = A & 0 = 0 , A v 0 = A
9) законы де Моргана A&$\bar{B}$ = $\bar{A}$ v$\bar{B}$

   (1-ый закон)¯v$\bar{A}\bar{B}$¯ = A¯ &
   $\bar{B}$ (2ой закон)

Рассмотрим область, ограниченную известными функциями. Требуется определить, попадает ли точка с заданными координатами в обозначенную область.

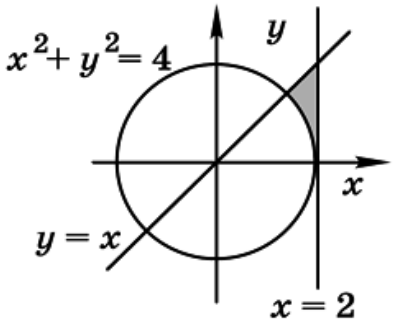


Рисунок 7.1 – Изображение обозначенной области на координатной плоскости

Программа определения попадания точки в заданную область:

```
x = float (input ())
y = float (input ())
```

```
if x<=2 and y <= x and 4<=x*x+y*y:
print('Yes!')
 else:
print('No!')
```

Рабочее задание

Задание 1 (вариант выбираем по номеру компьютера). Написать листинг программы построения таблиц истинности к задачам. Обязательно писать комментарии

Вариант	**Задача**
1	Вычислить значение логического выражения при следующих значениях логических величин A, B и C: A = Истина, B = Ложь, C = Ложь: а) A или B; б) A и B; в) B или C.
2	Вычислить значение логического выражения при следующих значениях логических величин X, Y и Z: X = Ложь, Y = Истина, Z = Ложь: а) X или Z; б) X и Y; в) X и Z.
3	Вычислить значение логического выражения при следующих значениях логических величин A, B и C: A = Истина, B = Ложь, C = Ложь: а) не A и B; б) A или не B; в) A и B или C.
4	Вычислить значение логического выражения при следующих значениях логических величин X, Y и Z: X = Истина, Y = Истина, Z = Ложь: а) не X и Y; б) X или не Y; в) X или Y и Z
5	Вычислить значение логического выражения при следующих значениях логических величин A, B и C: A = Истина, B = Ложь, C = Ложь: а) A или B и не C; г) A и не B или C;
6	Вычислить значение логического выражения при следующих значениях логических величин X, Y и Z:

	X = Ложь, Y = Ложь, Z = Истина: а) X или Y и не Z; г) X и не Y или Z
7	Вычислить значение логического выражения при следующих значениях логических величин X, Y и Z: X = Ложь, Y = Ложь, Z = Истина: а) не X и не Y; б) X и (не Y или Z);
8	Вычислить значение логического выражения при следующих значениях логических величин X, Y и Z: X = Ложь, Y = Ложь, Z = Истина: а) не (X и Z) или Y; б) X или (не (Y или Z)).
9	Вычислить значение логического выражения при следующих значениях логических величин A, B и C: A = Истина, B = Ложь, C = Ложь: а) A или не (A и B) или C; б) не A или A и (B или C);
10	Вычислить значение логического выражения при следующих значениях логических величин X, Y и Z: X = Ложь, Y = Истина, Z = Ложь: а) X и не (Z или Y) или не Z; б) не X или X и (Y или Z);
11	Вычислить значение логического выражения при следующих значениях логических величин X, Y и Z: X = Истина, Y = Ложь, Z = Ложь: а) не X или не Y или не Z; б) (не X или не Y) и (X или Y);
12	Вычислить значение логического выражения при следующих значениях логических величин A, B и C: A = Ложь, B = Ложь, C = Истина: а) (не A или не B) и не C; б) (не A или не B) и (A или B);
13	Вычислить значение логического выражения при следующих значениях логических величин A, B и C: A = Ложь, B = Ложь, C = Истина: а) A и B или A и C или не C и Вычислить значение логического выражения при следующих значениях логических величин X, Y и Z: X = Истина, Y = Ложь, Z = Ложь: а) X и Y или X и Z или не Z.
14	Вычислить значение логического выражения при следующих значениях логических величин X, Y и Z:

	X = Ложь, Y = Истина, Z = Ложь: а) X и Z; б) X или Y; в) X и Y.
15	Вычислить значение логического выражения при следующих значениях логических величин X, Y и Z: X = Ложь, Y = Истина, Z = Ложь: а) X и не Y; б) X или не Y; в) X и Y и Z

Задание 2 (вариант выбираем по номеру компьютера)

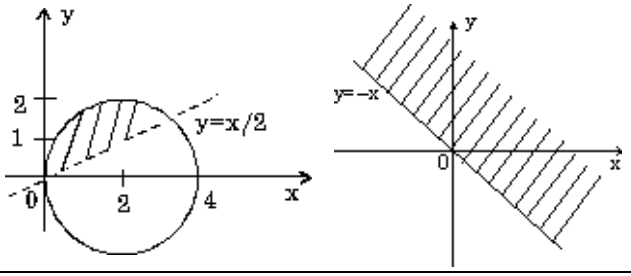
Сформулируйте условие принадлежности точки к заштрихованным областям. Составьте блок – схему алгоритма установления принадлежности точки к заштрихованным областям. Пунктиром обозначены границы, не входящие в данную область. Напишите программу. Результат работы программы должен быть в виде одного из четырех сообщений:

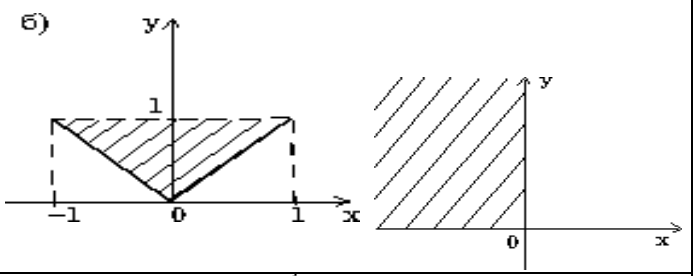
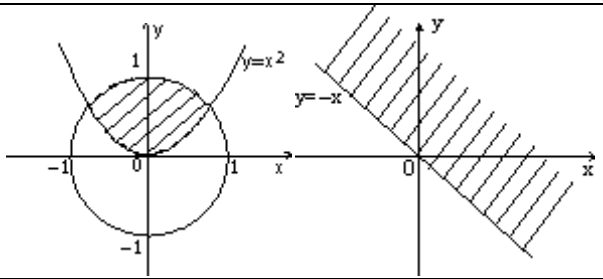
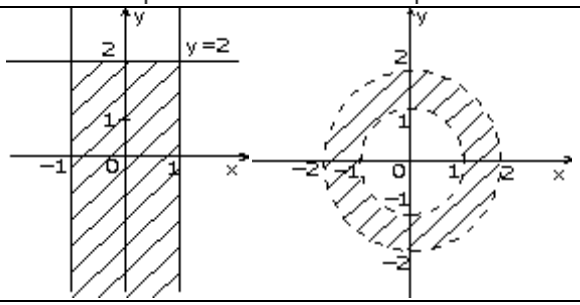
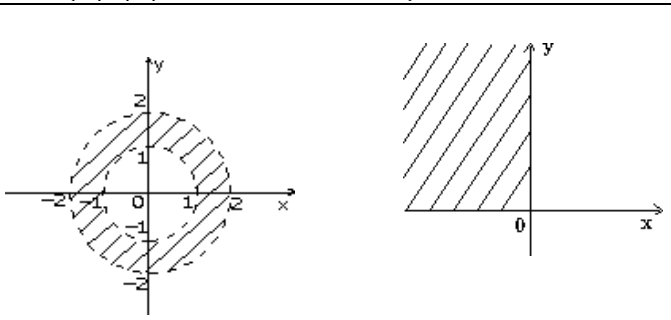
1.точка принадлежит только области 1;

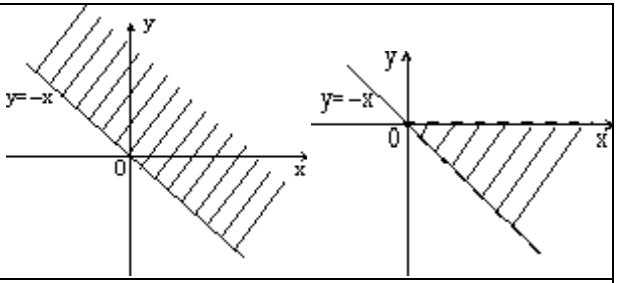
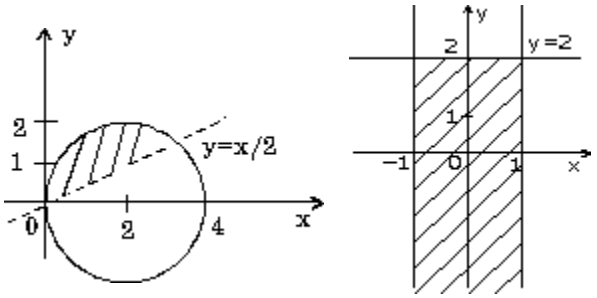
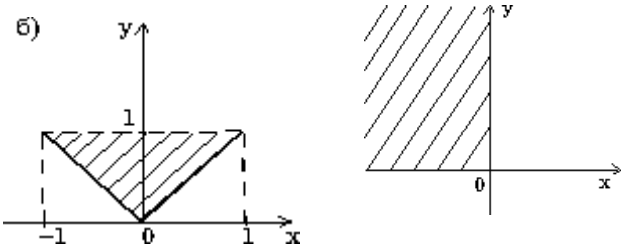
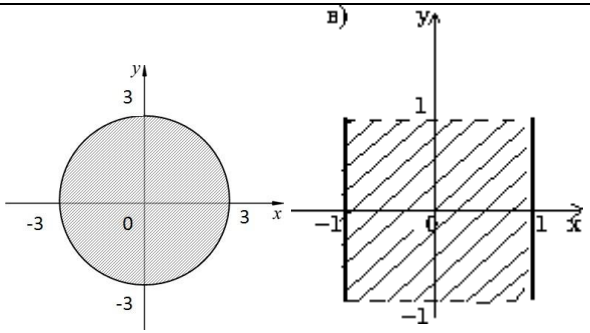
2.точка принадлежит только области 2;

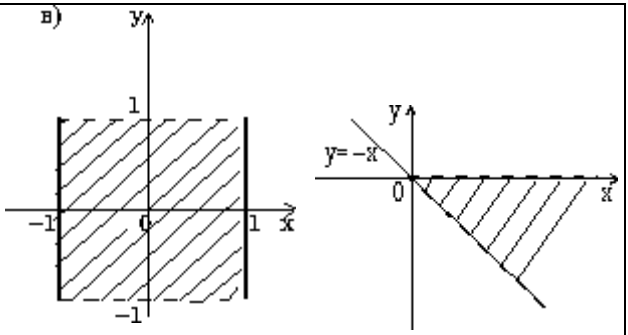
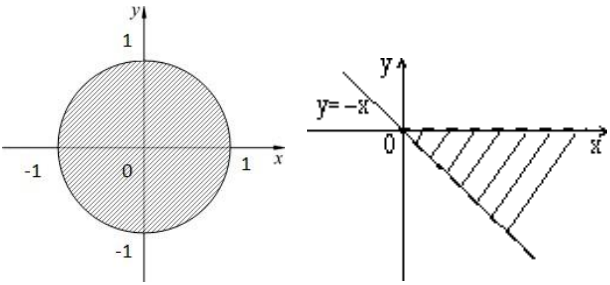
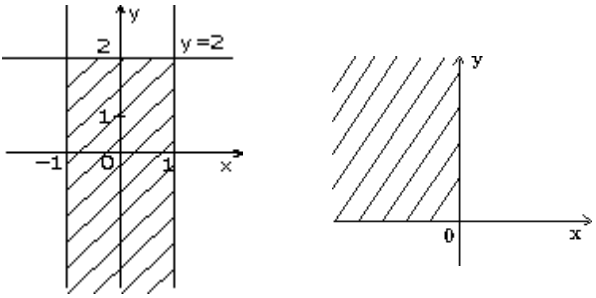
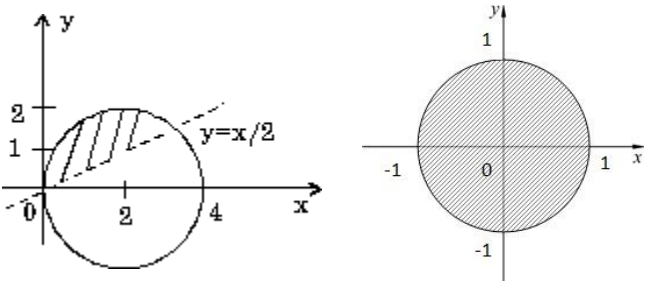
3.точка принадлежит обеим областям;

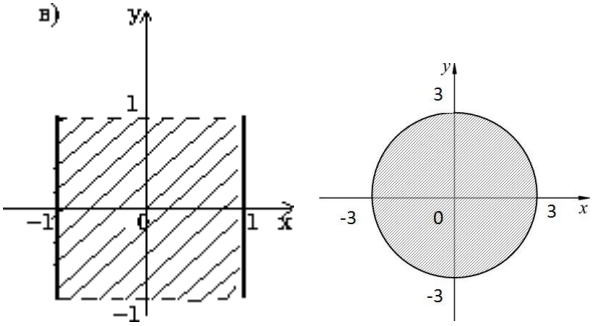
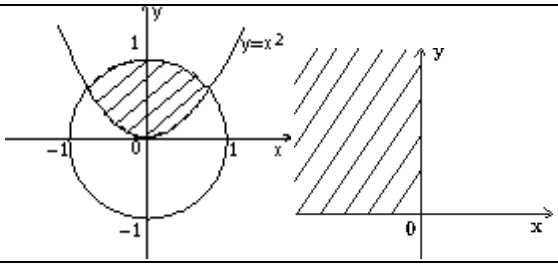
4.точка не принадлежит ни одной из указанных областей.

№.	Задание 2
1	

2	 б) y 1 −1 0 1 x y 0 x
3	 y 1 y=x 2 −1 0 1 x −1 y y=−x 0 x
4	 y 2 y =2 1 −1 0 1 x y 2 1 −2 −1 0 1 2 x −1 −2
5	 y 2 1 −2 −1 0 1 2 x −1 −2 y 0 x

6	y y=–x 0 x y y=–x 0 x
7	y 2 1 y=x/2 0 2 4 x y 2 y=2 1 –1 0 1 x
8	б) y 1 –1 0 1 x y 0 x
9	y 3 -3 0 3 x -3 в) y 1 –1 0 1 x –1

10	 в) y 1 −1 0 1 x −1 y y= −x 0 x
11	 y 1 -1 0 1 x -1 y y= −x 0 x
12	 y 2 y =2 1 −1 0 1 x y 0 x
13	 y 2 1 y=x/2 0 2 4 x y 1 -1 0 1 x -1

14	
15	

Время работы: 2 часа.

Практическое занятие № 8 «Функции и процедуры. Локальные, глобальные переменные: область видимости»

Цель работы Получение практических навыков написания функций в Python. Изучение механизма области видимости переменных.

Теоретическая часть

Вспомогательный алгоритм — алгоритм, который можно использовать в других алгоритмах, указав его имя и, если необходимо, значения параметров. Вспомогательный алгоритм, записанный на языке программирования, называют подпрограммой.

Вспомогательные алгоритмы делятся на функции и процедуры.

В Python не различают отдельно эти два вида подпрограмм. Отличие функции от процедуры в том, что процедура выполняет действия, не сохраняя нигде никаких результатов, а функция всегда возвращает значение. Для завершения выполнения вызова функции используется оператор return, он “возвращает” вызывающему объекту результат (значение выражения, следующего за ключевым словом return). Операторы после операторов return не выполняются. Если оператор return не содержит никакого выражения, то возвращается специальное значение None.

В программировании особое внимание уделяется концепции о локальных и глобальных переменных, а также связанное с ними представление об областях видимости. Соответственно, локальные переменные видны только в локальной области видимости, которой может выступать отдельно взятая функция. Глобальные переменные видны во всей программе. "Видны" – значит, известны, доступны. К ним можно обратиться по имени и получить связанное с ними значение.

К глобальной переменной можно обратиться из локальной области видимости. К локальной переменной нельзя обратиться из глобальной области видимости, потому что локальная переменная существует только в момент выполнения тела функции. При выходе из нее, локальные переменные исчезают. Компьютерная память, которая под них отводилась, освобождается. Когда функция будет снова вызвана, локальные переменные будут созданы заново.

Пример программы нахождения площади двух фигур с использованием глобальной переменной result. Значение этой переменной задается внутри каждой из функций и при это доступно в основном коде программы.

```
def rectangle():
    a = float(input("Ширина: "))
    b = float(input("Высота: "))
    global result
    result = a * b

def triangle():
    a = float(input("Основание: "))
    h = float(input("Высота: "))
    global result
    result = 0.5 * a * h

result = 0

figure = input("1-прямоугольник, 2-треугольник:
")

if figure == '1':
    rectangle()

elif figure == '2':
    triangle()

print("Площадь: %.2f" % result)
```

Рабочее задание

1.Напишите процедуру, которая принимает параметр натуральное число N и выводит на экран линию из N символов ‘-‘.

2.Напишите процедуру, которая принимает натуральное число N и выводит на экран квадрат из звездочек со стороной N.

3.Напишите функцию, которая определяет, верно ли что число N – простое.

4.Напишите функцию, которая принимает параметр натуральное число N и выводит сумму всех целых чисел от 1 до N.

5.Напишите программу вычисления площади квадрата и площади круга, используя две разные функции, где результат вычислений присваивается внутри каждой из функций и доступен в основном коде программы.

6.Нарисуйте блок – схему к задаче 5.

7. Объявите функцию с именем is_triangle, которая принимает три стороны треугольника (целые числа) и проверяет, можно ли из переданных аргументов составить треугольник. (Напоминаем, что у любого треугольника длина третьей стороны всегда должна быть меньше суммы двух других). Если проверка проходит, вернуть булево значение True, иначе - значение False.

8. Объявите функцию для проверки числа на четность (возвращается True, если переданное число четное и False, если число нечетное). После объявления функции в цикле необходимо считывать целое числовое значение (функцией input), пока не поступит число 1. Если прочитанное значение четное (проверяется с помощью заданной функции), то оно выводится на экран (в столбик, то есть, каждое значение с новой строки).

9.На вход подается натуральное число N от 1 до 20. Необходимо найти произведение чисел от 1 до N.

- решить задачу с помощью функции;
- решить задачу с помощью процедуры.

10.Написать функцию, которая определяет, соответствует ли введенное пятизначное число маске *2*7*. Если соответствует, функция возвращает сумму цифр числа, если нет - ноль.

Время работы: 2 часа.

Практическое занятие № 9 «Решение задач обработки списков»

Цель работы Получение практических навыков решения задач работы со списками

Теоретическая часть

Список (list) – это структура данных для хранения объектов различных типов.

Списки являются упорядоченными

последовательностями, которые состоят из различных типов данных, заключающихся в квадратные скобки [ ] и отделяющиеся друг от друга с помощью запятой. В одном списке одновременно могут лежать данные разных типов — например, и строки, и числа. В один список можно вложить другой список.

Список (list) — это упорядоченный набор элементов, каждый из которых имеет свой номер, или **индекс**, позволяющий быстро получить к нему доступ. Нумерация элементов в списке начинается с *нуля*. Мы можем без проблем узнать индекс элемента и обратиться по нему. Когда мы создаём объект list, в памяти компьютера под него резервируется место. Нам не нужно переживать о том, сколько выделяется места и когда оно освобождается, — Python всё сделает сам. Например, когда мы добавляем новые элементы, он выделяет память, а когда удаляем старые — освобождает.

Однако в списках Python можно хранить объекты разного размера и типа. Более того, размер массива ограничен, а размер списка в Python — нет. Но всё равно мы знаем, сколько у нас элементов, а значит, можем обратиться к любому из них с помощью индексов.

Существует небольшой трюк: списки в Python представляют собой *массив ссылок*. Каждый элемент такого массива хранит не сами данные, а ссылку на их расположение в памяти компьютера!

Функции для работы со списками в Python — это автономные фрагменты кода, выполняющие определенную задачу со списком, а методы работы со списками в Python — это встроенные функции, которые вызываются над объектом списка для выполнения определенной операции.

Функции списков

- len(list): возвращает длину списка
- min(list): возвращает наименьший элемент списка
- max(list): возвращает наибольший элемент списка
- sorted (): возвращает отсортированный список

Методы списков

- list.append(x) — позволяет добавлять элемент в конец списка;

- list1.extend(list2) — предназначен для сложения списков;
- list.insert(i, x) — служит для добавления элемента на указанную позицию(i — позиция, x — элемент);
- list.remove(x)— удаляет элемент из списка (только первое вхождение);
- list.clear() — предназначен для удаления всех элементов (после этой операции список становится пустым []);
- list.copy() — служит для копирования списков.
- list.count(x) — посчитает количество элементов x в списке;
- list.index(x) — вернет позицию первого найденного элемента x в списке;
- list.pop(i) — удалит элемент из позиции i;
- list.reverse() — меняет порядок элементов в списке на противоположный;
- list.sort() — сортирует список.

Создание списков на Python.

1.Получение списка через присваивание конкретных значений.

Так выглядит в коде Python пустой список:

s = [] # Пустой список

Примеры создания списков со значениями:

```
l=[5,75,-4,7,-51]# список целых чисел
l=[1.13,5.34,12.63,4.6,34.0,12.8]# список из вещественных чисел
l=["Оля", "Владимир", "Михаил", "Дарья"]# список из строк
l=["Москва", "Иванов", 12, 124]   # смешанный список
l=[[0, 0, 0], [1, 0, 1], [1, 1, 0]]   # список, состоящий из списков
l=['s', 'p', ['isok'], 2] # список из значений и списка
```

Списки можно складывать с помощью знака «+»:

```
l=[1, 3]+[4,23]+[5]
print('l=[1, 3]+[4,23]+[5] =',l)
```

Результат:

```
>>>
l=[1, 3]+[4,23]+[5] = [1, 3, 4, 23, 5]
>>>
```

2.Создание списка при помощи функции Split().

Используя функцию split в Python можно получить из строки список.stroka ="Привет, страна"

lst=stroka.split(",")

```
============== RESTART: C:/Users/maxim/Desktop/ex_list_ - строка
stroka =  Здравствуй, Дедушка Мороз
lst=stroka.split(","): ['Здравствуй', ' Дедушка Мороз']
print('lst=stroka.split(","):',lst)
```

Результат:

3.Генераторы списков.

Самым простым способом создания списка является обычное присваивание ему необходимых значений или объектов. В том случае, когда элементов последовательности немного, их можно просто перечислить один за другим. Но если их количество переваливает за десяток, следует всерьез задуматься об использовании генератора списка. ***Данная конструкция обеспечивает его автоматическое заполнение, исходя из определенных инструкций***.

Следующий пример демонстрирует создание в Python списка чисел при помощи генератора. Переменная i является ссылкой на текущий элемент объекта data. Функция range здесь принимает два аргумента, которые устанавливают границы для сгенерированной последовательности целых чисел. Вывод информации на экран происходит через метод print.

```
>>> data = [i for i in range(0, 10)]
>>> print(data)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Генераторы списка Python 3 работают не только с численными значениями. Как показывает следующий пример, генератору можно передать в качестве аргумента строку либо ссылку на нее.

```
>>> data = [i for i in "python"]
>>> print(data)
```

```
['p', 'y', 't', 'h', 'o', 'n']
```

В результате выполнения этого кода будет построен список из символов, которые включала в себя исходная строка. Как и в случае с предыдущим примером, вывести полученный набор данных на экран можно при помощи уже известного метода print.

Генератор с условием

Один из часто используемых способов генерации списков в Python — с условием. Предыдущие примеры демонстрировали довольно тривиальный подход к заполнению списка при помощи генератора. Данные конструкции языка Python ***позволяют задавать определенные условия для выполнения такой операции.*** Например, существует задача перенести в список только четные числовые значения из созданной ранее выборки range.

```
>>> data = [i for i in range(0, 10) if i % 2 == 0]
>>> print(data)
[0, 2, 4, 6, 8]
```

В дело вступает команда if, после которой следует небольшая инструкция, сообщающая о том, что каждый элемент последовательности должен выполнять некоторое условие. В данном случае необходимо возвращать 0 в результате вычисления остатка от целочисленного деления. Как можно увидеть, прошедшие проверку цифры были занесены в список.

Генератор с циклом

В языке Python также существует возможность генерировать список при помощи нескольких циклов, помещая их в квадратные скобки инициализации. В следующем примере используется две конструкции for, осуществляющие поочередный перебор значений для переменных i и j. Числовой диапазон для них (от 0 до 3), как и раньше, задается через методы range.

```
>>> data = [i * j for i in range(0, 3) for j in range(0, 3)]
>>> print(data)
[0, 0, 0, 0, 1, 2, 0, 2, 4]
```

Результатом выполнения данного кода станет создание

списка data на основе поочередного перемножения переменных i и j. Благодаря циклам for, их значения увеличиваются пошагово. Таким образом, генерируется список из девяти элементов.

Рабочее задание

1.Ввести с клавиатуры список фамилий писателей. Отсортировать список в алфавитном порядке и вывести его.

2.Создать список из пяти элементов. Заполнить его случайными числами. Вывести этот список. Найти и вывести сумму его элементов. Вывести наибольший элемент списка.

3.Заполнить массив из 5 элементов случайными числами в интервале [-100,100]. Найти сумму всех отрицательных элементов массива. Если отрицательных элементов в массиве нет, вывести сообщение «отрицательных элементов нет».

4.Создать список из десяти элементов. Заполнить его случайными числами. Заменить все нечетные числа нулями. Вывести исходный и получившийся списки.

5.Заполнить массив **a** случайными числами, отобрать в массив **b** все числа, меньшие пяти. Вывести оба массива.

Время работы: 2 часа.

Практическое занятие № 10 «Обработка вложенных списков»

Цель работы Получение практических навыков написания программ обработки вложенных списков

Теоретическая часть

Часто в задачах приходится хранить прямоугольные таблицы с данными. Такие таблицы называются матрицами или двумерными массивами. В языке программирования Питон таблицу можно представить в виде списка строк, каждый элемент которого является в свою очередь списком, например, чисел. Например, приведём программу, в которой создаётся числовая таблица из двух строк и трех столбцов, с которой производятся различные действия.

запустить выполнить пошагово

```
1  a = [[1, 2, 3], [4, 5, 6]]
2  print(a[0])
3  print(a[1])
4  b = a[0]
5  print(b)
6  print(a[0][2])
7  a[0][1] = 7
8  print(a)
9  print(b)
10 b[2] = 9
11 print(a[0])
12 print(b)
13
```

Здесь первая строка списка a[0] является списком из чисел [1, 2, 3]. То есть a[0][0] == 1, значение a[0][1] == 2, a[0][2] == 3, a[1][0] == 4, a[1][1] == 5, a[1][2] == 6.

Для обработки и вывода списка, как правило, используют два вложенных цикла. Первый цикл перебирает номер строки, второй цикл бежит по элементам внутри строки. Например, вывести двумерный числовой список на экран построчно, разделяя числа пробелами внутри одной строки, можно так:

запустить выполнить пошагово

```
1 a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
2 for i in range(len(a)):
3     for j in range(len(a[i])):
4         print(a[i][j], end=' ')
5     print()
6
```

Рабочее задание

1.Заполнить двумерный массив 10 на 10 случайными значениями в диапазоне от 1 до 10. Вывести все значения в виде таблицы

2.Дан список 20 случайных чисел list. Написать две функции. Первая возвращает список list_n, где собраны только нечетные значения списка list, вторая функция возвращает список list_c, где

собраны только четные значения списка list. Вывести исходный список и результат работы обеих функций.

3.Дан двумерный список случайных двузначных чисел 3 на 3. Переписать значения списка так, чтобы элементы каждой строки стали в обратном порядке.

Например

Исходный список			Результат		
22	14	37	37	14	22
74	98	51	51	98	74
43	63	29	29	63	43

1.Написать программу, реализующую механизм перемножения двух матриц размером 3х3.

2.Написать программу, которая заносит в двумерный массив и печатает таблицу умножения от 5 до 15.

Вывести исходный список и результат.

Время работы: 2 часа.

Практическое занятие № 11 «Решение задач со множествами. Операции со строковым типом данных»

Цель работы Получение практических навыков написания программ обработки строкового типа данных, работа со множествами

Теоретическая часть

Кортеж (tuple) – это неизменяемая структура данных, которая по своей структуре очень похожа на список, за исключением изменяемости. Кортежи используются, когда необходимо быть уверенным, что элементы структуры данных не будут изменяться в процессе работы программы. Кортежи в памяти занимают меньший объем по сравнению со списками. Кортежи работают быстрее, чем списки (т.е. на операции перебора элементов и т.п. будет тратиться меньше времени). Кортежи можно использовать в качестве ключа у словаря.

Множества - это встроенный тип, предлагающий широкий набор возможностей, которые повторяют теорию **множеств** из математики. Множество задается перечислением всех его элементов

в фигурных скобках. Исключением является пустое множество, которое можно создать при помощи функции set(). Если функции set передать в качестве параметра список, строку или кортеж, то она вернёт множество, составленное из элементов списка, строки, кортежа. Например,

A = {1, 2, 3}
A = set('qwerty')
print(A)

выведет {'e', 'q', 'r', 't', 'w', 'y'}

Каждый элемент может входить в множество только один раз, порядок задания элементов не важен.

Словарь - структура данных, позволяющая идентифицировать ее элементы не по числовому индексу, а по произвольному, называется *словарем или ассоциативным массивом, или хеш-таблицей.* Соответствующая структура данных в языке Питон называется **dict**. Каждый элемент словаря состоит из двух объектов: *ключа* и *значения*. В нашем примере ключом является название страны, значением является название столицы. Ключ идентифицирует элемент словаря, значение является данными, которые соответствуют данному ключу. Значения ключей — уникальны, двух одинаковых ключей в словаре быть не может.

Рабочее задание

№ п.п	Задание – «Строки»
1	Строка t равна “СИНХРОФАЗОТРОН”. Написать алгоритм (программу) составления слов: ФАЗАН ФОТОН РОЗА
2	Строка t равна “КИБЕРНЕТИКА”. Написать алгоритм (программу) составления слов: КАБЕРНЕ БРИКЕТ
3	Ввести с клавиатуры две символьные строки. Сравнить вторые по счету символы этих строк. Если это одна и та же буква, вывести соответствующее сообщение. Сравнить строки между собой, вывести соответствующее сообщение.
4	Проверить, одинаковое ли число открывающихся и закрывающихся скобок в данной строке.

5	Строка t равна “СМАРТФОН”. Написать алгоритм (программу) составления слов: МАРТ СОРТ ТРАСТ.
6	Строка t равна “КОМПЬЮТЕР”. Написать алгоритм (программу) удаления букв, содержащихся в словах: КОТ ТРЮМО. Вывести слово, получившееся после обработки начального слова “КОМПЬЮТЕР”.
7	Дана строка. Добавить в начале, середине и в конце строки последовательность букв abc. Вывести результат на экран.
8	Строка t равна “ГИГАБАЙТ”. Написать алгоритм (программу) составления слов: ТАЙГА БИТА.
9	Строка t равна “ФОРМА”. Написать алгоритм (программу) составления слова: “ИНФОРМАТИКА”, используя начальную строку.

Задание – «Множества»

1.Дан список программ. Некоторые программы из этого списка установлены на 3-х компьютерах. Определить, каких программ из списка нет ни на одном их имеющихся компьютеров, и какие есть в каждом из них.

2.Множество содержит N случайных чисел из интервала [a, b]. Требуется:

а) определить, сколько элементов содержится в множестве;

б) определить, сколько четных чисел данного множества принадлежит интервалу [c, d].

3.Дано множество, состоящее из гласных и согласных букв латинского алфавита. Определить, каких букв больше. Вывести сообщение.

4.Дан текст, содержащий числовую информацию. Определить, какие цифры содержатся в данном тексте.

5.Дан текст, содержащий запись арифметического выражения. Определить, какие арифметические операции использованы в этом выражении.

Время работы: 2 часа.

Практическое занятие № 12 «Программы работы с файлами»

Цель работы Получение практических навыков написания программ работы с файлами

Теоретическая часть

В Python работа с файлами осуществляется с помощью встроенных функций и методов. Для открытия файла используется функция `open()`, которая принимает два аргумента - путь к файлу и режим открытия (например, "r" для чтения, "w" для записи, "a" для добавления).

После открытия файла можно использовать методы для чтения (`read()`, `readline()`, `readlines()`) или записи (`write()`, `writelines()`). После окончания работы с файлом его необходимо закрыть с помощью метода `close()`.

Также в Python существует оператор `with`, который автоматически закрывает файл после завершения работы с ним. Например,

```
with open("file.txt", "r") as file:
    data = file.read()
    print(data)
```

Рабочее задание

1. Дан текстовый файл, в котором записано число Пи. Вывести на экран 15 чисел, начиная со второго знака после запятой.

2. Дан текстовый файл литературного произведения «Алиса в стране чудес». Посчитать сколько всего слов в файле, посчитать сколько раз встречается словосочетание «чеширский кот». Вывести соответствующие сообщения.

3. Написать программу, определяющую, содержится ли введенное четырехзначное число в последовательности чисел дробной части числа пи (использовать файл с записью числа с 1000000 знаков после запятой).

4. С помощью программы создать текстовый файл t1.txt на рабочем столе. Ввести с клавиатуры текстовую строку и записать в этот файл.

5. Считать первые 10 строк файла с текстом «Алиса в стране чудес». Записать этот текст в новый файл «alice2».

Время работы: 2 часа.

Практическое занятие № 13 «Алгоритмы вычисления рекуррентных последовательностей»

Цель работы Получение практических навыков написания программ вычисления рекуррентных последовательностей

Теоретическая часть

Рекуррентно заданные последовательности — это последовательности, для которых задаётся первый (или несколько первых) элемент и алгоритм нахождения каждого следующего элемента через предыдущий.

В математике известно понятие ***рекуррентной последовательности чисел*** (от латинского «*recurrere*» – «*возвращаться*»).

Это понятие вводят так: пусть известно ***k*** чисел $a_1, \ldots, a_k$, которые являются началом числовой последовательности. Следующие элементы этой последовательности вычисляются так:

$a_{k+1}=F(a_1, \ldots, a_k)$; $a_{k+2}=F(a_1, \ldots, a_{k+1})$;

$a_{k+3}=F(a_1, \ldots, a_{k+2})$;…, $a_{k+i}=F(a_1, \ldots, a_{k+i-1})$

Величина ***k*** называется ***глубиной рекурсии.***

С рекуррентными последовательностями связаны следующие задачи:

1) вычисление заданного n-го элемента последовательности;

2) математическая обработка определенной части последовательности (например, вычисление суммы или произведения первых n - членов);

3) подсчет числа элементов на заданном отрезке последовательности, удовлетворяющих определенным свойствам;

4) определение номера первого элемента, удовлетворяющего определенному требованию;

5) вычисление и сохранение в памяти заданного числа элементов последовательности.

Рекуррентная формула - это математическое выражение, которое определяет последовательность чисел или функций через их предыдущие значения. Такая формула часто используется для описания последовательностей, рекурсивных алгоритмов и динамического программирования.

Рабочее задание

1. Определить значение 15-го элемента последовательности чисел Фибоначчи. Рекуррентная формула для вычисления элементов ряда Фибоначчи следующая:

$$F_k = \begin{cases} F_1 = 1; \\ F_2 = 1; \\ F_k = F_{k-1} + F_{k-2}, \ k > 2 \end{cases}.$$

2. Найти сумму первых 20 элементов арифметической прогрессии, если известно, что первый элемент=1, разность между любым и рядом стоящими элементами =8.

3. Вычислить число ПИ с точностью до третьего знака после запятой, через вычисление суммы:

$$\pi = 4 \cdot \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots\right) = 4 \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{2k-1}.$$

4. Вычислить приближенное значение cos(x), где x вводится с клавиатуры (|x|<1, аргумент в радианах), через разложение вряд:

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots = \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k}}{(2k)!}.$$

5.Найти десятый элемент геометрической прогрессии, если известно, что первый элемент=100, а знаменатель геометрической прогрессии=0.5

Время работы: 2 часа.

Практическое занятие № 14 «Рекурсивные алгоритмы»

Цель работы Получение практических навыков разработки рекурсивных алгоритмов

Теоретическая часть

Рекурсия — это способ организации вспомогательного алгоритма (подпрограммы), при котором эта подпрограмма (процедура или функция) в ходе выполнения обращается сама к себе.

В рекурсивном определении должно присутствовать ограничение, **граничное условие**, при выходе на которое дальнейшая инициация рекурсивных обращений прекращается.

Важно учитывать временную и пространственную сложности. Рекурсивные функции занимают больше места в памяти по сравнению с итеративными из-за постоянного добавления новых слоев в стек в памяти. Часто их производительность выше. Рекурсия может быть медленнее, если реализована неправильно. Из-за этого вычисления будут происходить чаще, чем требуется.

Самовызов процедуры ничем не отличается от **вызова другой процедуры**.

Что происходит, если одна процедура вызывает другую:

- в памяти размещаются **параметры**, передаваемые процедуре;
- в другом месте памяти сохраняются **значения** внутренних переменных вызывающей процедуры;
- запоминается адрес возврата в вызывающую процедуру;
- управление передается вызванной процедуре.

Рекурсия распространена в Python, когда ожидаемые вводы не вызовут значительного количества вызовов рекурсивных функций. Пример: рекурсивная функцию, которая определяет, является ли введенная пользователем строка палиндромом.

```
def palindrome(my_str):
    if len(my_str) == 0 or len(my_str) == 1: # граничный случай
        return True
    else: # рекурсивный случай
        head = my_str[0]
        middle = my_str[1:-1] e
        nd = my_str[-1]
        return head == end and palindrome(middle)
st = [i.lower() for i in input() if i.isalpha()] print((palindrome(st)))
```

Рабочее задание

1. Вычислите 7 элемент последовательности чисел Фибоначчи с помощью рекурсивной функции.

2. Напишите рекурсивную функцию summation(n), которая будет принимать положительное целое число n и возвращать сумму чисел от 1 до n.

3. Алгоритм вычисления функций F(n) и G(n) задан следующими соотношениями:
F(1) = G(1) = 1
F(n) = 3·F(n–1) + G(n–1) – n + 5, если n > 1
G(n) = F(n–1) + 3·G(n–1) – 3·n, если n > 1
Чему равно значение F(14) + G(14)?

4. Напишите рекурсивную функцию sum_sub(list), которая будет принимать список целых чисел. Эта функция будет суммировать все нечётные числа и вычитать все чётные числа. В конце она будет возвращать получившееся значение.

5. Напишите рекурсивную функцию вычисления значения выражения:

$$\cfrac{x}{x^2+\cfrac{2}{x^2+\cfrac{4}{x^2+\cfrac{8}{\ddots\,x^2+\cfrac{256}{x^2}}}}}$$

Время работы: 2 часа.

Практическое занятие № 15 «Создание рисунков с помощью программного кода. Узоры. Орнаменты. Фракталы»

Цель работы Получение практических навыков разработки приложений с графическим интерфейсом

Теоретическая часть

Модуль Graph — это набор функций, который представляет собой «обёртку» для создания учебных графических программ на языке Python на основе виджета Canvas библиотеки Tkinter.
Вот некоторые функции модуля Graph:

- mainWindow() — возвращает ссылку на объект главного окна, что позволяет использовать все возможности Tkinter;

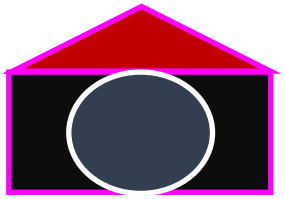
- windowSize(width, height) — устанавливает ширину (width) и высоту (height) рабочей области окна;
- canvas() — возвращает ссылку на объект области рисования, что позволяет использовать все возможности виджета Canvas пакета Tkinter;
- canvasPos(x, y) — устанавливает координаты (x, y) области рисования;
- canvasSize(width, height) — устанавливает ширину (width) и высоту (height) области рисования.
- **Рабочее задание**
- **1.** Написать программу построения рисунка 1:
- 

Рисунок 15.1 – Вид рисунка 1 для построения

2. Написать программу построения рисунка 2:

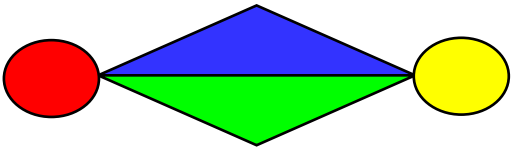


Рисунок 15.2 – Вид рисунка 2 для построения

3. Написать программу построения рисунка 3 (использовать процедуру построения одного треугольника):

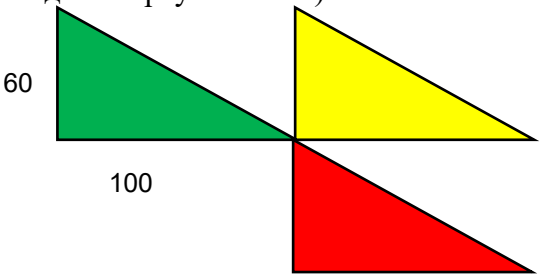


Рисунок 15.3 – Вид рисунка 3 для построения

4. Написать программу построения рисунка 4 использовать

процедуру построения одного треугольника):

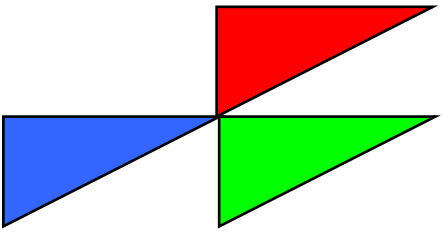


Рисунок 15.4 – Вид рисунка 4 для построения

5. Ввести с клавиатуры число N и вывести на экран N окружностей в ряд.

6. Написать программу построения рисунка 5:

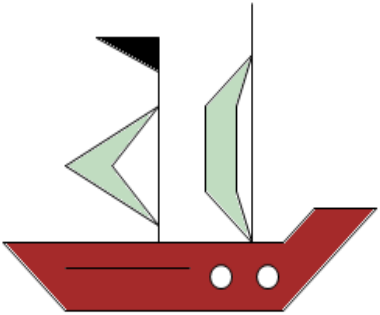


Рисунок 15.5 –Вид рисунка 5 для построения

Время работы: 2 часа.

Практическое занятие № 16 «Построение графиков математических функций. Графика, модуль Tkinter»

Цель работы Получение практических навыков разработки приложений с графическим интерфейсом

Теоретическая часть

Tkinter - стандартная библиотека для создания графического интерфейса пользователя (GUI) в Python. С ее помощью также можно создавать графические изображения. Приведем краткую информацию о построении графических изображений с помощью Tkinter.

1. Создание окна: сначала необходимо создать окно, в котором будет отображаться графическое изображение. Для этого используется класс `Tk()` из библиотеки Tkinter.

2. Создание холста: после создания окна необходимо создать холст, на котором будет отрисовываться графическое изображение. Для этого используется класс `Canvas()`.

3. Отрисовка фигур: с помощью методов холста (`create_line()`, `create_rectangle()`, `create_oval()`, `create_polygon()` и т. д.) можно рисовать линии, прямоугольники, овалы, многоугольники и другие фигуры.

4. Работа с цветами: для задания цветов фигур и фона можно использовать стандартные цветовые имена (например, `"red"`, `"blue"`, `"green"`), шестнадцатеричные значения цветов или RGB значения.

5. Отображение изображения: после отрисовки всех элементов на холсте необходимо вызвать метод `pack()` или `grid()` для отображения холста в окне.

Пример кода для создания графического изображения с использованием Tkinter:

```
from tkinter import *
# Создание окна
root = Tk()
root.title("Графическое изображение")

# Создание холста
canvas = Canvas(root, width=400, height=400)
canvas.pack()

# Отрисовка круга
canvas.create_oval(100, 100, 300, 300, fill="red")

# Отображение изображения
root.mainloop()
```

Этот пример создает окно с красным кругом на холсте. Tkinter позволяет создавать различные графические изображения и

интерактивные элементы для пользовательского интерфейса.

Рабочее задание

1.Нарисовать прямоугольник со скругленными углами.
2.Нарисовать 10 прямоугольников различной степени прозрачности и разного размера.
3.Построить график функции y=1\x+4x-2.
4.Написать программу построения графического изображения

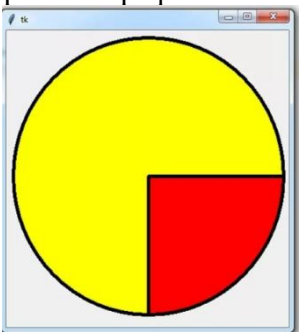


Рисунок 16.1 – Вид рисунка 1 для построения

5.Написать программу построения подобного изображения, чтобы чередовались 2 цвета овалов

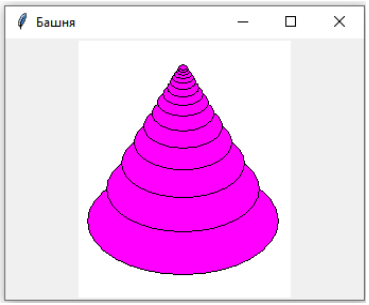


Рисунок 16.2 – Вид рисунка 2 для построения

6.В графическом окне вывести текст.

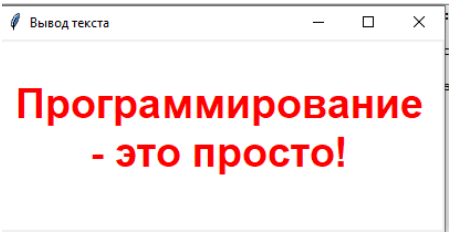


Рисунок 16.3 - Текст для построения

Время работы: 2 часа.

Практическое занятие № 17 «Технологии модульного программирования»

Цель работы Получение практических навыков разработки приложений с графическим интерфейсом

Теоретическая часть

Модули в Python - это просто файлы Python с расширением .py.

Модуль оформляется в виде отдельного файла с исходным кодом. Имя модуля будет именем файла. Модуль Python может иметь набор функций,классов или переменных, определенных и реализованных.

Создание модуля

Чтобы создать свой **модуль в Python** достаточно сохранить код в файл срасширением**.py** Теперь он доступен в любом другом файле.

Например, создадим файл mymodule.py, в которой определим какие-нибудь функции:

```
def hello(): print('Hello, world!')

def fib(n): a = b = 1
  for i in range(n - 2): a, b = b, a + b
  return b
```

Теперь в этой же папке создадим другой файл, например, main.py:

```
import mymodule

mymodule.hello() print(mymodule.fib(10))
```

Выведет:
Hello, world! 55

Модуль нельзя именовать также, как и ключевое, также имена модулей нельзя начинать с цифры и не стоит называть модуль также, как какую-либо из встроенных функций.

Рабочее задание

Задание 1. Составить программу, согласно полученному варианту задания.

№ п/п	Задание
1	Список **A** содержит **N** чисел. Перепишите из списка **A** в список **B** толькоте элементы, значения которых не равны заданному значению **K**. Назначение функции: переписывание из одного списка в другой только тех элементов, которые не совпадают с заданным значением. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.
2	Список **A** содержит **N** упорядоченных по возрастанию чисел. Вставьте в список некоторое значение **K** так, чтобы упорядоченность списка ненарушилась. Назначение функции: вставка заданного значения в упорядоченный список. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.
3	Список **A** содержит **N** элементов, значения которых не определены. Организуйте запрос: «Сколько элементов списка следует заполнить?» Заполните список заданным числовым значением **K**. Назначение функции: заполнение заданного числа элементов заданным значением. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.

4	Список **A** содержит **N** чисел. Найдите количество элементов списка,значения которых превышают заданное значение ***K***. Назначение функции: подсчет количества элементов, превышающих заданное значение. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.
5	Список **A** содержит **N** чисел. Найдите сумму значений элементов списка,меньших заданного значения **K**. Назначение функции: суммированиеэлементов, значения которых меньше заданного. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.
6	Список **A** содержит **N** чисел. Найдите произведение значений элементовсписка. Назначение функции: вычисление произведения элементов списка. Оформите созданную функцию в виде программного модуля иподключите его к основной программе.
7	Список **A** содержит **N** чисел. Удалите из списка значение с порядковым номером **K**. Назначение функции: удаление из списка значения с заданным порядковым номером. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.
8	Список **A** содержит **N** чисел. Значения всех элементов списка, отличающихся от заданного значения **K**, замените другим заданным значением **M**. Назначение функции: замена значений элементов списка. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.
9	Список **A** содержит **N** чисел. Найдите в нем элемент с наибольшимзначением. Назначение функции: поиск в списке элемента с наибольшимзначением. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.

10	Список **A** содержит **N** чисел. Разработайте программу, с помощью которой можно определить количество наибольших элементов в нем. Назначение 1 функции: нахождение максимального элемента. Назначение 2 функции: подсчет количества максимальных элементов. Оформите созданные функции в виде программного модуля и подключите его к основной программе.

Задание 2. Дано **N** целых чисел. Найдите среди них число, у которого сумма цифр имеет максимальное значение. Назначение первой функции: нахождение суммы цифр числа. Назначение второй функции: выбор числа с максимальной суммой цифр. Оформите созданные функции в виде программного модуля и подключите его к основной программе.

Содержание отчета:

Постановка задачи. Текст программы. Решение трех задач. Результаты выполнения программы (скриншоты).

Время работы: 2 часа.

Практическое занятие № 18 «Разработка приложений с графическим интерфейсом. Элементы окна»

Цель работы Получение практических навыков разработки приложений с графическим интерфейсом

Теоретическая часть

Рассмотрим часть графических объектов (виджет), содержащихся в библиотеке Tkinter: кнопки, поля для ввода, метки, флажки, переключатели и списки. Следует понимать, что графический интерфейс пользователя достаточно стандартен, и поэтому любые подобные библиотеки-модули (в том числе и Tkinter) содержат приблизительно одинаковые виджеты.

Каждый класс виджет имеет определенные свойства, значения которых можно задавать при их создании, а также программировать их изменение при действии пользователя и в результате выполнения программы.

Кнопки

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

*переменная = Button (родит_виджет, *свойство=значение, ...+)*

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

```
1. from tkinter import *
2.
3. root = Tk()
4.
5. but = Button(root,
6.         text="Это кнопка", #надпись на кнопке
7.         width=30,height=5, #ширина и высота
8.         bg="white",fg="blue")  #цвет  фона  и
надписи
9.
10. but.pack()
11. root.mainloop()
```

bg и fg – это сокращения от background (фон) и foreground (передний план). Ширина и высота измеряются в знакоместах (количествосимволов).

Метки

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

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

Однострочное текстовое поле

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

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

bd – это сокращение от borderwidth (ширина границы).

Многострочное текстовое поле

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

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

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

Радиокнопки (переключатели)

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

```
from tkinter import *
root = Tk()
var=IntVar()
var.set(0)

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

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

Флажки

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

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

Списки

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

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

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

Рабочее задание

Задание 1. Создайте небольшое окно с заголовком «Привет, пользователь!». В окне расположить кнопку «Здравствуйте!».

Задание 2. Создайте программу на языке программирования Python и с использованием модуля Tkinter генерирующую шаблон, представленный ниже:

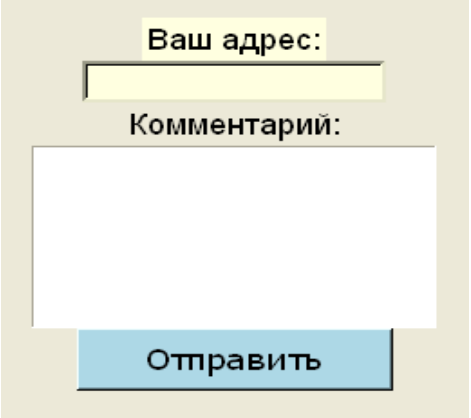


Рисунок 18.1 – Примерное окно для построения

Задание 3. Создайте программу на языке программирования Python с использованием модуля Tkinter



Рисунок 18.2-Шаблон с использованием модуля Tkinter

Задание 4. Написать программу управления положением квадрата на экране. При нажатии клавиш «Стрелка влево» «Стрелка вправо» «Стрелка вверх» «Стрелка вниз» квадрат должен перемещаться в соответствующем направлении.

Время работы: 2 часа.

Практическое занятие № 19 «Разработка приложений с графическим интерфейсом. Элементы окна»

Цель работы Получение практических навыков разработки приложений с графическим интерфейсом

Теоретическая часть

Продолжим рассматривать графические объекты (виджеты), содержащихся в библиотеке Tkinter. Это будут рамка (frame), шкала (scale), полоса прокрутки (scrollbar), окно верхнего уровня (toplevel).

Frame (рамка)

Рамки (фреймы) хороший инструмент для организации остальных виджетов в группы внутри окна, а также оформления.

```
1. from tkinter import *
2.
3. root = Tk()
4.
5. fra1 = Frame(root,width=500,height=100,bg="darkred")
6. fra2 = Frame(root,width=300,height=200,bg="green",bd=20)
7. fra3 = Frame(root,width=500,height=150,bg="darkblue")
8.
9. fra1.pack()
10. fra2.pack()
11. fra3.pack()
12.
13. root.mainloop()
```

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

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

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

Scale (шкала)

Назначение шкалы — это предоставление пользователю выбора какого-то значения из определенного диапазона. Внешне

шкала представляет собой горизонтальную или вертикальную полосу с разметкой, по которой пользователь может передвигать движок, осуществляя тем самым выбор значения.

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

Свойства:

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

Scrollbar (полоса прокрутки)

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

```
1. from tkinter import *
2.
3. root = Tk()
4.
5. tx = Text(root,width=40,height=3,font='14')
6. scr = Scrollbar(root,command=tx.yview)
7. tx.configure(yscrollcommand=scr.set)
8.
9. tx.grid(row=0,column=0)
10. scr.grid(row=0,column=1)
11. root.mainloop()
```

В примере сначала создается текстовое поле (tx), затем полоса прокрутки (scr), которая привязывается с помощью опции

command к полю tx по вертикальной оси (yview). Далее поле tx изменяется (конфигурируется) с помощью метода configure: устанавливается значение опции yscrollcommand.

Здесь используется незнакомый нам пока еще метод grid, представляющий собой другой способ расположения виджет на окне. Toplevel (окно верхнего уровня).

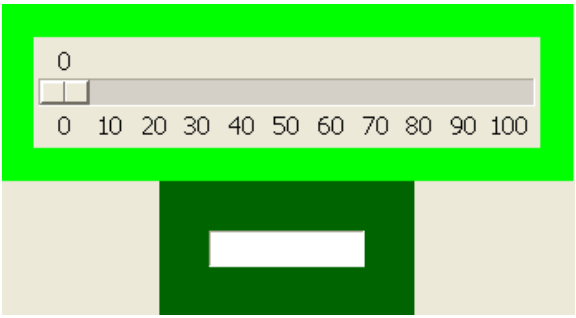
С помощью класс Toplevel создаются дочерние окна, на которых также могут располагаться виджеты. Следует отметить, что при закрытии главного окна (или родительского), окно Toplevel также закрывается. С другой стороны, закрытие дочернего окна не приводит к закрытию главного.

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

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

Рабочее задание

Задание 1. Создайте программу на языке программирования Python и с использованием модуля Tkinter генерирующего шаблоны, представленные ниже.



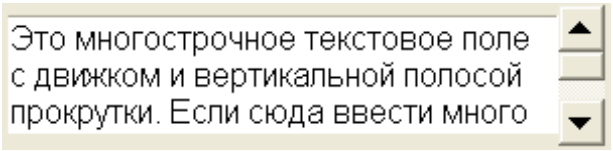


Рисунок 19.1 – Внешний вид виджетов

Задание 2. Создайте приложение, состоящее из главного и двух дочерних окон. На каждом из трех окон должны располагаться один или два любыхграфических объекта.

Задание 3. Создать окно с заголовком «Калькулятор», содержащее 2 поля ввода, кнопку, «Рассчитать!», текстовую метку «Сумма чисел». При нажатии кнопки «Рассчитать» в метку «Сумма чисел» выводится значение суммы чисел из полей «Число1», «Число2». См. рисунок 19.2



Рисунок 19.2 – Вид окна программы

Время работы: 2 часа.

Практическое занятие № 20 «Разработка приложений с графическим интерфейсом. Компьютерная графика»

Цель работы Получение практических навыков разработки приложений с графическим интерфейсом

Теоретическая часть

Для создания меню и подменю в Tkinter используется виджет `Menu`. Вот краткое описание того, как создать меню и подменю в Tkinter:

1. Создание главного меню:

– Создайте объект `Menu` для главного окна или фрейма.
– Добавьте различные команды и подменю к этому объекту `Menu`.

Пример создания главного меню:

```
from tkinter import*
root = Tk()
menu_bar = Menu(root)
# Создание меню "File"
file_menu = Menu(menu_bar, tearoff=0)
file_menu.add_command(label="Open")
file_menu.add_command(label="Save")
file_menu.add_separator()
file_menu.add_command(label="Exit", command=root.quit)
# Добавление меню "File" в главное меню
menu_bar.add_cascade(label="File",
menu=file_menu)

# Установка главного меню
root.config(menu=menu_bar)
root.mainloop()
```

2. Добавление подменю:

– Создайте объект `Menu` для подменю.
– Добавьте команды к этому объекту `Menu`.
– Добавьте подменю к главному меню.

Пример добавления подменю:

```
# Создание подменю "Edit"
edit_menu = Menu(menu_bar, tearoff=0)
edit_menu.add_command(label="Cut")
edit_menu.add_command(label="Copy")
edit_menu.add_command(label="Paste")
# Добавление подменю "Edit" в главное меню
menu_bar.add_cascade(label="Edit",
menu=edit_menu)
```

3. Обработка событий:

– Добавьте функции обработчики для выполнения действий при выборе пунктов меню.

– Эти функции должны быть связаны с командами в меню.

Пример добавления функции обработчика:

```
def open_file():
    print("Opening file...")
file_menu.add_command(label="Open",          com-
mand=open_file)
```

Таким образом, вы можете создавать главные меню и подменю в Tkinter, добавлять команды и функции обработчики для выполнения различных действий при выборе пользователем определенных пунктов меню.

Рабочее задание

Задание 1. Создать окно, содержащее два пункта меню: «Файл», «Правка».

Меню «Файл» содержит подпункты:

- Создать;
- Редактировать;
- Выход.

Меню: «Правка» содержит подпункты:

- Копировать;
- Вырезать;
- Вставить.

Задание 2. Создать окно приложения, содержащее следующие виджеты:

Button1 – при нажатии помещает в ScrolledText текст.

Button2 – при нажатии очищает ScrolledText.

Задание 3. Создать окно, содержащие два поля ввода, кнопку и текстовую метку. Первое поле – ввод числа. Второе поле – ввод числа. При нажатии кнопки в текстовую метку выводится результат – сумма двух чисел из полей ввода.

Время работы: 2 часа.

Практическое занятие № 21 «Разработка приложений с графическим интерфейсом. Компьютерная графика»

Цель работы Получение практических навыков разработки приложений с графическим интерфейсом

Теоретическая часть

Tkinter - это стандартная библиотека Python для создания графических пользовательских интерфейсов. В Tkinter есть несколько графических примитивов, которые можно использовать для рисования на холсте (Canvas). Некоторые из графических примитивов в Tkinter:

1. Линия (Line): Для создания линии на холсте.
2. Прямоугольник (Rectangle): Для создания прямоугольника на холсте.
3. Овал (Oval): Для создания овала на холсте.
4. Текст (Text): Для добавления текста на холст.
5. Изображение (Image): Для добавления изображения на холст.

Чтобы поместить изображение в окно программы Tkinter, сначала нужно загрузить изображение с помощью модуля `PIL` (Pillow) и затем использовать метод `create_image` холста (Canvas) для добавления изображения на холст. Пример кода:

```
from tkinter import *
from PIL import Image, ImageTk

root = Tk()

# Загрузка изображения
image = Image.open("example.jpg")
photo = ImageTk.PhotoImage(image)

# Создание холста
canvas = Canvas(root, width=image.width, height=image.height)
canvas.pack()

# Добавление изображения на холст
canvas.create_image(0, 0, anchor=NW, image=photo)
```

```
root.mainloop()
```

Tkinter поддерживает следующие форматы графических файлов для использования с изображениями: GIF, PGM, PPM, PNG и BMP. Однако, для загрузки и отображения изображений в других форматах, например, JPEG, необходимо использовать модуль `PIL` (Pillow) для преобразования изображения в поддерживаемый формат.

Рабочее задание

Задание 1. Создайте приложение, в главном окне которого располагаются кнопки: записать в файл (при нажатии информация из поля ввод записывается в файл и поле очищается), читать из файла (при нажатии информация из файла выводится в поле ввода). См. рисунок 21.1.

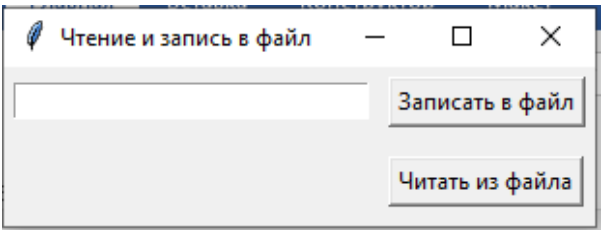


Рисунок 21.1 – Вид программы 1

Задание 2. Создать окно с заголовком «Геометрические примитивы». При нажатии «Нарисовать окружность» должна рисоваться окружность, при нажатии «Нарисовать квадрат» должен рисоваться квадрат (возможны варианты в оформлении).





Рисунок 21.2 – Вид программы 2

Задание 3. Создать окно простейшей анимации (с использованием растрового изображения), выполнение которой начинается при нажатии соответствующей кнопки.



Рисунок 21.3 – Вид программы 3.

Задание 4. Добавить в предыдущем задании кнопку «Стоп», при нажатии которой анимация будет останавливаться, а при повторном нажатии кнопки «Стоп» движение возобновляется.

Время работы: 4 часа.

Практическое занятие № 22 «Разработка приложений с графическим интерфейсом. Многооконный интерфейс»

Цель работы Получение практических навыков разработки приложений с графическим интерфейсом

Теоретическая часть

Для создания многооконного приложения с помощью Tkinter вам потребуется создать несколько окон (оконных объектов) и управлять их отображением. Ниже приведен пример кода, который демонстрирует создание многооконного приложения с использованием Tkinter:

```
from tkinter import *
# Функция для создания нового окна
def create_new_window():
    new_window = Toplevel(root)
    new_window.title("Новое окно")
    label = Label(new_window, text="Это новое
окно")
    label.pack()
# Создание основного окна
root = Tk()
root.title("Главное окно")
# Кнопка для создания нового окна
button = Button(root, text="Открыть новое
окно", command=create_new_window)
button.pack()
root.mainloop()
```

В этом примере мы создаем основное окно с кнопкой "Открыть новое окно". При нажатии на кнопку создается новое окно с надписью "Это новое окно". Можно создавать сколько угодно новых окон, используя функцию `Toplevel()` для создания нового окна. Таким образом, вы можете создать многооконное приложение в Tkinter, добавляя несколько окон и управляя ими с помощью соответствующих функций и элементов управления.

Справочная информация по виджетам Tkinter:
https://metanit.com/python/tkinter/1.2.php

Чтобы сделать окно закрывалось при нажатии кнопки в Tkinter, вы можете использовать метод `destroy()` для закрытия

окна. Ниже приведен пример кода, который показывает, как закрыть окно при нажатии кнопки:

```
from tkinter import *
def close_window():
    root.destroy()

root = Tk()
root.title("Закрытие окна")
label = Label(root, text="Нажмите кнопку, чтобы закрыть окно")
label.pack()
button = Button(root, text="Закрыть окно", command=close_window)
button.pack()
root.mainloop()
```

В этом примере при нажатии на кнопку "Закрыть окно" вызывается функция `close_window()`, которая использует метод `destroy()` для закрытия главного окна `root`. Таким образом, при нажатии на кнопку окно будет закрыто.

Вы можете адаптировать этот пример для закрытия любого другого окна в вашем приложении, просто передавая нужное окно в качестве аргумента функции `destroy()`.

Рабочее задание

Задание 1. Написать программу, при запуске которой открывается небольшое окно, заголовок окна «Форма авторизации».

В окне три текстовых метки, два поля «Логин», «Пароль» и кнопка «Вход». В зависимости от того, какой логин и пароль введены, откроется новое окно с заданным содержанием: «Приветствую, Администратор!» или «Приветствую, пользователь». Пароли администратора и пользователя должны храниться в текстовом файле (или разных файлах). В случае ввода неверных данных вывести соответствующее сообщение. Рисунок 23.1

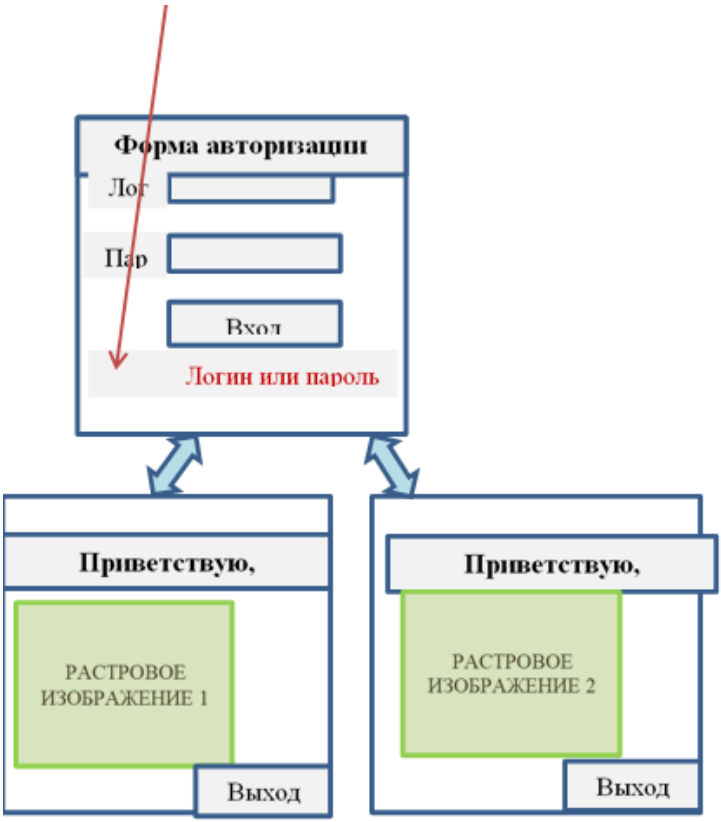


Рисунок 22.1 – Схема интерфейса разрабатываемого приложения

Задание 2. Добавить защиту пароля: при вводе пароля заменять вводимые символы «*».

Задание 3. Добавить в каждое из дочерних окон растровое изображение с кнопкой «Выход», при нажатии которой снова открывается форма авторизации.

Задание 4. В дочернем окне «Приветствую, пользователь» добавить форму загрузки изображения и опцию размещения изображения по центру окна.

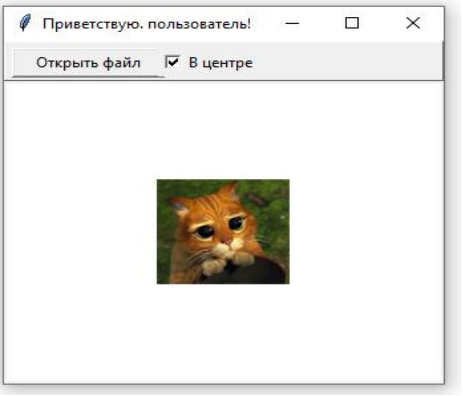


Рисунок 22.2 –Интерфейс разрабатываемого приложения

Пример кода:

Окно пользователя.py - C:\Users\efilonova\Downloads\Окно пользователя.py (3.10.6)

File Edit Format Run Options Window Help

```
from simpletk import *
from tkinter import filedialog

app = TApplication("Приветствую. пользователь!")
app.position = (200, 200)
app.size = (300, 300)

panel = TPanel(app, relief="raised", height=35, bd=1)
panel.align = "top"

image = TImage(app, bg="white")
image.align = "client"
#image.picture = "flower.gif"

def selectFile(sender):
  fname = filedialog.askopenfilename(
     filetypes=[("Файлы GIF", "*.gif"),
                ("Все файлы", "*.*")] )
  if fname:
    image.picture = fname

openBtn = TButton(panel, width=15, text="Открыть файл")
openBtn.position = (5, 5)
openBtn.onClick = selectFile

def cbChanged(sender):
  image.center = sender.checked
  image.redrawImage()

centerCb = TCheckBox(panel,text="В центре")
centerCb.position = (115, 5)
centerCb.onChange = cbChanged

app.Run()
```

Рисунок 22.3 – Код разрабатываемого приложения

Задание 5. В дочернем окне «Приветствую, Администратор!» добавить поля записи новых логинов и паролей пользователей в файл *.txt (дозапись в файл). Новые пользователи должны иметь доступ по этим логинам и паролям.

Время работы: 4 часа.

ПЕРЕЧЕНЬ ИСПОЛЬЗОВАННЫХ ИНФОРМАЦИОННЫХ РЕСУРСОВ

1.Лубашева, Т.В. Основы алгоритмизации и программирования: учебное пособие/ Т.В. Лубашева, Б.А. Железко. - Минск: РИПО, 2016. - 378 с.: ил. - Библиогр. в кн. - ISBN 978-985-503-625-9; [Электронный ресурс].-URL: http://biblioclub.ru/index.php?page=book&id=463632 .

2.Коврижных, А.Ю. Основы алгоритмизации и программирования: практикум. В 2 ч. Ч. 2. Расчетные работы [Электронный ресурс]: учебно-методическое пособие / А.Ю. Коврижных, Е.А. Конончук, Г.Е. Лузина. — Электрон. дан. — Екатеринбург: УрФУ, 2016. — 44 с. — Режим доступа: https://e.lanbook.com/book/98291 .

3.Коврижных, А.Ю. Основы алгоритмизации и программирования: практикум. В2 ч. Ч. 1. Задачи и упражнения [Электронный ресурс]: учебно-методическое пособие / А.Ю. Коврижных, Е. К, Г.Е. Лузина. — Электрон. дан. — Екатеринбург: УрФУ, 2016. — 52 с. — Режим доступа: https://e.lanbook.com/book/98290 .

4.Златопольский Д.М. Основы программирования на языке Python. – М.: ДМК Пресс, 2017. – 284 с.

5.Свейгарт, Эл. Автоматизация рутинных задач с помощью Python: практическое руководство для начинающих. Пер. с англ. — М.: Вильямс, 2016. – 592 с.