СРЕДНЕЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАНИЕ

Серия основана в 2001 году

С.Р. Гуриков

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ НА PYTHON

Учебное пособие

Рекомендовано Учебно-методическим советом СПО
в качестве учебного пособия для студентов учебных заведений,
реализующих программу среднего профессионального образования
по специальностям 09.02.01 «Компьютерные системы и комплексы»,
09.02.02 «Компьютерные сети», 09.02.03 «Программирование в компьютерных системах»,
09.02.04 «Информационные системы (по отраслям)»,
09.02.05 «Прикладная информатика (по отраслям)»

Москва ФОРУМ 2019

ИНФРА-М

УДК 004.43(075.32)
ББК 32.973.1-018я723
Г95

Рецензент:
В.Н. Шакин — кандидат технических наук, доцент, заведующий кафедрой «Информатика», декан общетехнического факультета (ОТФ2) Московского технического университета связи и информатики

Гуриков С.Р.

Г95 Основы алгоритмизации и программирования на Python : учеб. пособие / С.Р. Гуриков. — М. : ФОРУМ : ИНФРА-М, 2019. — 343 с. — (Среднее профессиональное образование).

ISBN 978-5-00091-553-0 (ФОРУМ)
ISBN 978-5-16-013983-8 (ИНФРА-М, print)
ISBN 978-5-16-106723-9 (ИНФРА-М, online)

В учебном пособии рассмотрены основы алгоритмизации и программирования на языке Python. Содержится описание такого материала, как работа линейных, разветвляющихся и циклических структур, обработка списков, кортежей и вложенных последовательностей, создание модулей. Приведены примеры создания объектно-ориентированных и событийно-ориентированных программ. Кроме того, рассмотрены методы работы со строками, функциями, файлами.

В конце каждой главы содержится набор контрольных вопросов и упражнений, задач для самостоятельного решения. В приложении приведены варианты к лабораторным работам по темам, изложенным в учебном пособии.

Содержание книги будет полезно студентам и преподавателям средних специальных учебных заведений, а также школьникам при подготовке к Единому государственному экзамену по дисциплине «Информатика и ИКТ».

УДК 004.43(075.32)
ББК 32.973.1-018я723

ISBN 978-5-00091-553-0 (ФОРУМ)
ISBN 978-5-16-013983-8 (ИНФРА-М, print)
ISBN 978-5-16-106723-9 (ИНФРА-М, online)

ВВЕДЕНИЕ

В условиях глобальных информационных процессов, решения социально-экономических проблем актуализируется важность информатизации образования. На сегодняшний день почти все учебные заведения обладают новейшим компьютерным оборудованием, подключенным к сети Интернет.

Однако известно, что мало иметь современный персональный компьютер, нужно также позаботиться и об установке программного обеспечения. Не секрет, что лицензионные программы стоят немалых денег, которые не всегда есть в бюджете того или иного учебного заведения. В частности, по этой причине компания Microsoft разработала ряд программ лицензирования программных продуктов специально для учебных заведений, например, *Microsoft Academic Open License* – для высших учебных заведений и учебных заведений среднего и среднего специального образования, а также *Microsoft School Agreement* – для начальных и средних учебных заведений.

Вместе с очевидными достоинствами такого подхода имеется и существенный недостаток – формирование зависимости начинающих пользователей от того или иного программного продукта. В результате после окончания учебного заведения вчерашние школьники или студенты задумываются о приобретении необходимых им программ уже на коммерческой основе.

Следует отметить, что в учебных заведениях всех уровней можно использовать и преподавать свободные программные продукты, потому что только с их помощью можно реализовать право человека на распространение, усовершенствование и изучение используемых им программ.

Рынок программного обеспечения сегодня крайне изменчив и нестабилен, но, тем не менее, он находится в непрерывном развитии, и многие технологии обучения, в частности программированию, уже безвозвратно устарели. Ярким примером этого является обучение в средах Turbo Pascal и QBasic. Эти две программы, активно использующиеся в настоящее время в школьных курсах, уже не соответствуют уровню развития современных сред разработки. Более того, в них не поддерживаются принципы объектно-ориентированного подхода, а ведь именно на нем основаны методики разработки большинства современного программного обеспечения.

Отличным решением, с точки зрения преподавания основ программирования и алгоритмизации школьникам и студентам, может стать изучение языка Python, который относится к категории свободно распространяемого программного обеспечения. Следует отметить, что разработчики контрольно-измерительных материалов Единого государственного экзамена по дисциплине «Информатика и ИКТ» уже включили в задания по программированию примеры кодов на этом языке.

Python – современный, активно развивающийся язык программирования, который привлекает разработчиков всего мира. Он используется во многих круп-

ных компаниях, таких как Google, IBM, Microsoft, Red Hat, Yahoo! и др. Это бесплатная система с открытым кодом, поддерживающая динамическую типизацию (не надо заранее объявлять переменные), объектно-ориентированное программирование, интеграцию с другими языками, такими как C, C++, Java, обладает кроссплатформенностью (независимостью от той или иной операционной системы) и многими другими достоинствами.

Тем не менее, проведенный анализ литературы по языку программирования Python, имеющейся на рынке книгопечатной продукции, показал, что ее содержание практически не отвечает целям совместной работы студента и преподавателя в конкретном учебном заведении и на конкретном занятии. В основном книги по программированию на Python имеют характер справочных изданий либо вовлекают читателя в написание игровых моментов. Вероятно, подобное изложение представляет интерес для начинающего, но при этом рабочие программы дисциплин, связанных с обучением программированию, подразумевают совершенно иной алгоритм знакомства с языком.

Таким образом, главная цель настоящего издания – обеспечить прочное и сознательное освоение основ алгоритмизации и программирования, формирование практических умений – профессиональных, учебных, интеллектуальных, необходимых школьнику или студенту. Помимо приобретения чисто практических умений, ценных с точки зрения освоения компьютерной грамотности, ребята получат наглядное представление о возможностях, предоставляемых компьютером человеку, выработают при решении поставленных задач такие профессионально значимые качества, как самостоятельность, ответственность, точность, творческая инициатива.

Учебное пособие построено так, что непосредственно к программированию читатель приступает с самого начала – первая программа описывается уже во второй главе (в *первой главе* приводятся теоретические основы алгоритмизации и программирования).

Во *второй главе* описан процесс создания проекта на Python, рассказано о методах ввода и вывода данных, способах обработки исключительных ситуаций, возникающих в ходе выполнения программ.

В *третьей, четвертой и пятой главах* подробно рассматриваются операторы, относящиеся к трем «китам» программирования – линейному, разветвляющемуся и циклическому алгоритмам.

В *шестой главе* изложен процесс работы с кортежами (последовательность данных, не позволяющая изменять свои значения) и списками (являются изменяемыми последовательностями), а также работа со словарями (структурами данных).

Материал *седьмой главы* даст представление о работе со строками, познакомит с функциями и методами работы с ними, базовыми алгоритмами их обработки.

В *восьмой главе* объясняется работа со вложенными последовательностями, рассматриваются «классические» способы их обработки.

В *девятой главе* речь пойдет о создании пользовательских функций, технике написания и импортирования собственных модулей.

Десятая глава содержит информацию о работе с файлами, умение работать с которыми – необходимая компетенция человека, постигающего азы программирования.

Одиннадцатая и двенадцатая главы посвящены объектно-ориентированному и событийно-ориентированному программированию. Читатель получит представление о создании таких объектов, как классы, научится писать методы, расширяющие функциональность классов, а также создавать новые классы на основе уже существующих. Кроме того, знакомство с программированием виджетов позволит приобрести навыки создания Windows-приложений на языке Python.

В книге приведено около 200 листингов программ, снабженных комментариями. Для удобства их чтения **полужирным шрифтом** выделены основные программные конструкции, функции и методы языка Python. При наборе на компьютере программ, приведенных в учебном пособии, следует учитывать, что в Python отступы (четыре пробела или нажатие клавиши Tab) являются частью синтаксиса программных конструкций.

Материал, изложенный в пособии, особенно будет полезен школьникам при подготовке к ЕГЭ по дисциплине «Информатика и ИКТ», студентам, обучающимся на технических специальностях техникумов, колледжей и вузов. Апробация учебного пособия показала, что учащиеся, не имеющие подготовки в области программирования, не только успешно осваивают представленные в пособии приложения, но и вносят свои очень интересные изменения, направленные на улучшение их работы.

Преподаватели могут оценить учебное пособие с точки зрения методики преподавания. Многие программы специально написаны таким образом, чтобы учащиеся имели возможность доработать или оптимизировать их код, что поможет педагогу создать творческую атмосферу на занятиях. Практически к каждому из разделов приведены примеры решения задач, упражнения, задачи для самостоятельного решения, контрольные вопросы, позволяющие оценить уровень подготовки школьника или студента.

Следует отметить, что учебное пособие не может претендовать на функциональную полноту или на полную оригинальность приведенных методов, алгоритмов и программ. Автор преследовал иную цель: привлечь внимание школьников, студентов и преподавателей к такому активно развивающемуся языку программирования, как Python, дать почувствовать его преимущественные отличия от других языков. Python доступен для свободного скачивания по адресу http://www.python.org/.

Материал предлагаемого учебного пособия был подготовлен автором на основе многолетнего опыта преподавания курса алгоритмизации и программирования школьникам, студентам среднего и высшего профессионального образования, написания книг по программированию в различных средах.

1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

Прежде всего хочется отметить, что первая глава учебного пособия носит ознакомительный характер: здесь представлены теоретические основы алгоритмизации и программирования. Ее чтение подготовленному читателю можно пропустить, а начинающему лучше ее внимательно изучить. Тем не менее, стоит упомянуть, что в тексте будут приводиться примеры различных функций (термин «функция» уже может быть непонятен читателю), результаты их работы, терминологические названия (списки, кортежи и пр.), однако написанное может не объясняться в деталях.

Таким образом, мы сразу должны оговорить, что многое станет более понятным и доступным, как только вы начнете работать со средой программирования Python, напишете свои первые программы на основе чтения второй главы. Тогда вы сможете более осознанно перечитать материал первой главы и сделать соответствующие выводы.

1.1. Алгоритм. Свойства алгоритма. Способы описания алгоритма

Если мы хотим написать программу на каком-либо языке программирования, то сначала нам следует составить алгоритм решения задачи.

Алгоритм – это точное и простое описание последовательности действий для решения данной задачи. Алгоритм содержит несколько шагов, которые должны выполняться в определенной последовательности. Каждый шаг алгоритма может состоять из одной или нескольких простых операций.

Каждый из нас ежедневно использует различные алгоритмы: инструкции, правила, рецепты и т. д. Обычно мы это делаем не задумываясь. Например, открывая дверь ключом, никто не размышляет над тем, в какой последовательности выполнять действия. Однако чтобы кого-нибудь научить открывать дверь, придется четко указать и сами действия, и порядок их выполнения. Например:

1. Достать ключ.
2. Вставить ключ в замочную скважину.
3. Повернуть ключ два раза против часовой стрелки.
4. Вынуть ключ.

Представим, что мы поменяли местами второе и третье действия. Мы сможем выполнить и этот алгоритм, но дверь не откроется, т. е. алгоритм станет невыполнимым.

Для алгоритма важен не только набор действий, но и то, в каком порядке они выполняются. Понятие алгоритма в информатике является фундаментальным.

Таким же, какими являются понятия точки, прямой и плоскости в геометрии, вещества в химии, пространства и времени в физике и т. д.

Свойства алгоритма:

- дискретность (прерывность, раздельность) – алгоритм должен представлять процесс решения задачи как последовательное выполнение простых шагов (этапов);
- определенность – каждый шаг алгоритма должен быть четким и однозначным. Выполнение алгоритма носит механический характер и не требует никаких дополнительных сведений о решаемой задаче;
- результативность – алгоритм должен приводить к решению задачи за конечное число шагов;
- массовость – алгоритм решения разрабатывается в общем виде, т. е. он должен быть применим для решения некоторого класса задач, различающихся лишь исходными данными.

Способы описания алгоритмов

- словесный;
- графический;
- табличный;
- формульный.

1. Словесный способ каждый из нас использует ежедневно, пересказывая собеседнику, например, различные инструкции, правила, кулинарные рецепты, т. е. какую-то последовательность, приводящую к конечному результату.
2. Графический способ представления алгоритмов является более компактным и наглядным по сравнению со словесным. Как часто для лучшего понимания той или иной ситуации нам проще начертить какую-то схему, план, согласно которым мы будем действовать. В программировании данный способ предпочтителен, поскольку позволяет с помощью последовательности функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий, представить ход решения той или иной задачи. Такое представление алгоритма называется структурной схемой алгоритма, или блок-схемой.
3. Табличный способ используется, например, в бухгалтерии при составлении ежегодных отчетов, сводок и т. д.
4. Формульный способ находит свое применение при решении задач из области математики, физики и т. д. Например, при решении квадратного уравнения мы приступаем к нахождению дискриминанта уравнения, а затем, в зависимости от полученного результата, находим корни уравнения по известным всем формулам.

1.2. Назначение функциональных блоков

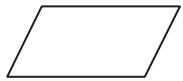
На условные обозначения в схемах алгоритмов, программ, данных и систем распространяется ГОСТ 19.701—90. «ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения». При отрисовке

блок-схем мы будем использовать функциональные блоки, представленные ниже. Заметим, что некоторые их названия или отрисовка не всегда совпадают с указанными в ГОСТ, поэтому при создании схем алгоритмов для курсовых, дипломных работ, безусловно, следует обратиться к существующему стандарту.

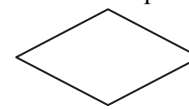
Начало и конец алгоритма

Вычислительная операция

Ввод/вывод данных



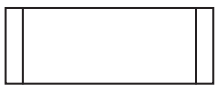
Проверка условия
(логического выражения)



Начало циклического алгоритма

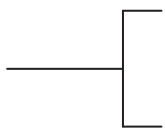
Разрыв соединительных линий
на странице

Вызов пользовательской функции

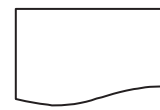


Линии потоков

Комментарий



Вывод на печать



1.3. Основные этапы решения задач

Процесс решения некоторой задачи в среде программирования, как правило, состоит из нескольких этапов, часть из которых выполняется пользователем, а часть – компьютером.

1-й этап. Общая постановка задачи.

На этом этапе описывается содержание задачи, составляется перечень исходных данных.

2-й этап. Разработка математической модели.

Цель этого этапа состоит в установлении формализованных связей между исходными данными и искомыми результатами. Этап заключается в записи расчетных формул или функциональных зависимостей.

3-й этап. Разработка алгоритма.

Этап состоит в описании последовательности действий, в результате которых может быть получено решение задачи.

4-й этап. Разработка программы.

Программа составляется в полном соответствии с разработанным алгоритмом решения задачи.

5-й этап. Отладка программы.

Процесс поиска ошибок в программе и их устранение.

6-й этап. Анализ результатов.

Позволяет принять решение о необходимости внесения изменений в программу, проведении дополнительных расчетов или их окончании.

1.4. Алфавит языка Python

Изучение любого языка начинается с изучения алфавита, из букв складываются слова, из слов – предложения. То же происходит и при изучении языка программирования. Сначала мы должны уяснить, какие символы можно использовать для записи слов языка, из которых можно формировать определенные конструкции. Итак, в алфавит языка Python входят:

1. Латинские буквы от a до z и от A до Z.

В Python есть различия между прописными и строчными буквами алфавита, например, chislo, CHISLO, Chislo – разные имена переменных.

2. Цифры от 0 до 9.
3. Специальные символы, например +, -, *, /.
4. Зарезервированные (служебные) слова: for, if, class, def и т. д.

1.5. Идентификаторы и общие правила их написания

Для того чтобы программа решения задачи обладала свойством массовости, следует употреблять не конкретные значения величин, а использовать их обозначения для возможности изменения по ходу выполнения программы их значений. Для обозначения в программе переменных и постоянных величин используются имена – **идентификаторы** (*identification* – установление соответствия объекта некоторому набору символов).

Программа на Python представляет собой последовательность инструкций, которые называются **операторами.** Необходимо учитывать, следующее:

- в идентификатор не могут входить пробелы, специальные символы алфавита;
- идентификатор начинается только с буквы или со знака подчеркивания;
- идентификатор может состоять из букв, цифр и знака подчеркивания;
- при написании идентификаторов можно использовать как прописные, так и строчные буквы латинского алфавита;
- идентификатор не должен являться зарезервированным словом.

Например:

summa1	правильно
2delta	ошибка
Block_35	правильно
Nomer.doma	ошибка
Сумма	ошибка

1.6. Оператор присваивания

Действия, выполняемые компьютером в процессе решения задачи, записываются в виде операторов алгоритмического языка. Изменение значения переменной осуществляется **оператором присваивания. Присваивание в Python означает связывание значения с некоторым именем переменной.**

Действие, выполняемое этим оператором, обозначается знаком «=».

Слева от этого знака записывается имя той переменной, которой нужно присвоить новое значение (например, x=). Справа может быть:

1. Число, например x=5.

Прокомментируем предложение, выделенное выше полужирным шрифтом. Дело в том, что при выполнении оператора **x=5** в оперативной памяти по некоторому адресу будет создан объект, получивший значение **5**. Затем будет создана переменная **x**, которой будет присвоен адрес этого объекта. Такой механизм занесения значений в ячейки памяти отличает Python от других языков программирования. Однако в дальнейшем при объяснении материала, связанного с операторами присваивания, мы будем использовать упрощенный вариант и выражаться так: «**в ячейку x заносится число 5**».

2. Выражение, например, арифметическое x=(3*a)/2 или логическое
   x=a>1.
3. Другая переменная, например x=a.

Операторы выполняются в той последовательности, в которой они записаны в программе. Например, фрагмент программы нахождения среднего арифметического включает операторы:

```
a=5
b=10
s=a+b
v=s/2
```

Возможна запись оператора присваивания в следующем виде:

= оператор – умножение с присваиванием, например, x=5 идентичен оператору x=x*5.

/= оператор – деление с присваиванием, например, x/=5 идентичен оператору x=x/5.

%= оператор – остаток от деления с присваиванием, например, x%=5 идентичен оператору x=x%5.

+= оператор – сложение с присваиванием, например, x+=5 идентичен оператору x=x+5.

-= оператор – вычитание с присваиванием, например, x-=5 идентичен оператору x=x-5.

1.7. Типы данных

Типы данных относятся к самым фундаментальным понятиям любого языка программирования. Для **определения (объявления)** переменных, интерпретатору или компилятору нужна следующая информация:

- **имя переменной** – по имени осуществляется связь переменной в программе с оперативной памятью компьютера;
- **тип переменной** – позволяет компилятору определить, какого вида информация хранится в переменной;
- **значение переменной** – определяет содержание информации, которая помещается в переменную.

В программах, написанных на Python, нет как такого раздела описания переменных. Сравните: в языках программирования Delphi, Pascal, среде программирования Lazarus вы обязаны объявить переменные, которые будете использовать в своей программе в разделе описания Var. В противном случае они просто не станут работать.

В языке программирования Microsoft Visual Basic можно обойтись без объявления переменных в разделе описания Dim, но тогда пользователь должен знать об особенностях типа данных Variant и придет к выводу о том, что переменные все-таки лучше описывать. В языке программирования Visual C# описать переменную можно непосредственно в тот момент, когда программист начинает ее использовать. Так вот, ничего из перечисленного в Python нет, вернее, типы данных есть, но в объявлении переменной нет необходимости.

В Python используется так называемая **динамическая типизация,** когда типы данных выясняются во время выполнения программы. Переменная сохраняет ссылку на объект определенного типа, а не сам объект. В момент переприсваивания значения другого типа переменная будет ссылаться на другой объект, при этом изменится и тип переменной. Python будет изменять тип переменной в вашей программе в соответствии с теми данными, которые вы собираетесь «отправить» в ту или иную переменную.

Выяснить, на какой тип данных ссылается переменная, поможет функция **type**, имеющая синтаксис:

type(имя переменной) или **type(значение)**

Например, в программе вы можете написать оператор **type(a)** или **type(196228),** и если переменная **a** имела в первом случае, предположим, целочисленный тип, то ответ будет таким: **<class 'int'>**. Во втором случае, очевидно, что **196228** – целое число, поэтому результат будет таким же. Слово **int** как раз и указывает на принадлежность к целочисленному типу.

Перечислим и дадим характеристику основным типам данных в Python.

Целые типы данных. Используются для представления целых чисел. Размер числа ограничен объемом имеющейся оперативной памяти. Как уже было сказано, при вызове функции **type**, как

type(10)

результатом будет

<class 'int'>

Над данными целого типа определены следующие арифметические операции:

+	сложение
-	вычитание
*	умножение
/	деление
//	деление нацело
%	остаток от целочисленного деления

Результат операции **%** можно вычислить по формуле:

a**%**b=a - (a/b)*b.

Ниже приведены результаты выполнения примеров с помощью операций % и //:

36 **%** 6 = 0
5 **%** 2 = 1
2 **%** 5 = 2
20 // 3 = 6
0 **%** 5 = 0

Вещественные типы данных

В языке Python допускается представление вещественных (дробных) значений в форме с плавающей и фиксированной точкой.

В форме **с фиксированной точкой** число представляется последовательностью десятичных цифр со знаком «плюс» или «минус». Например,

7.32 ; 456.721 ; 0.015 ; 192.0 ; - 15.0

Форма **с плавающей точкой** (экспоненциальный формат) используется для представления очень больших или очень маленьких чисел. В этой форме число записывается в виде:

$$\pm\, m\ E\ \pm\ p$$

где **m** – мантисса числа; **E** – символ, обозначающий основание десятичной ССЧ; **p** – порядок (степень) числа;

Например:
7.32e+00
4.56721e+02
1.5e-02
1.92e+02
-1.5e+01 .

За вещественными типами данных закреплен тип **float.**

type(1.5e-02)

Результатом функции **type** будет

<class 'float'>

Строковые типы данных

Значениями строковых переменных являются строковые константы (строки). В языке Python объекты строкового типа обозначены как **str.**

type("Лабораторная работа")

Результатом функции **type** будет

<class 'str'>

В следующем примере показаны некоторые приемы работы со строками. Один из самых популярных операторов в Python – это оператор **print,** который служит для вывода на экран текста, при этом текст должен быть заключен в двойные кавычки. Поскольку Python чувствителен к регистру, оператор **print** записывается строчными буквами, в противном случае (при записи **Print** или **PRINT**) работать он не будет.

Далее следует отметить, что значение, заключенное в кавычки и переданное оператору **print**, является строкой. Кавычки, в которые заключается строковая константа, могут быть одиночными либо двойными. Внутри строкового выражения, заключенного в двойные кавычки, могут встречаться строковые выражения, заключенные в одиночные кавычки, но не в двойные.

Наконец, строка, заключенная в тройные кавычки, позволит вывести строковое выражение, размещенное в нескольких строках.

```
print ("Лабораторная 'Вычисление арифметических выражений' работа")
print("Выполнил", "студент", "МТУСИ")
print('Гуриков Сергей Ростиславович')
print(
        """
        _____________________________
        _____________________________

        """
        )
input("\n\nНажмите ENTER, чтобы выйти")
```

Заключительный в этом коде оператор **input** позволяет остановить запущенную на выполнение программу, и выводит сообщение "Нажмите ENTER, чтобы выйти". В данной программной строке используется, так называемая Escape-последовательность, состоящая из обратного слеша и символа **n,** повторяющаяся несколько раз (\n\n). Следует отметить, что во многих языках программирования в консольном режиме используются Escape-последовательности и Python не исключение. С их помощью можно повысить наглядность выводимых на экран строк, например:

\n – переводит курсор на следующую строку;

\t – равносильно использованию клавиши Tab;

\\ – выводит обратный слеш \;

\' – выводит одиночную кавычку;

\" – выводит двойную кавычку.

Конкатенация (соединение) строк в языке Python возможна с помощью символа +. Например:

```
print("Москва " + "2016" \
    + " МТУСИ")
```

Для того чтобы слова, соединяемые операцией конкатенации, не сливались, необходимо добавлять пробел (пробелы) перед закрытием кавычки либо сразу

после кавычки так, как это показано в примере. Обратный слеш можно использовать при соединении нескольких строк.

Для того чтобы много раз повторить одну и ту же строку, в языке Python допустимо выполнить операцию умножения над строкой, что показано в следующем примере: **print**("МТУСИ"*5).

Итоговый результат рассмотренных выше примеров, представлен на рис. 1.

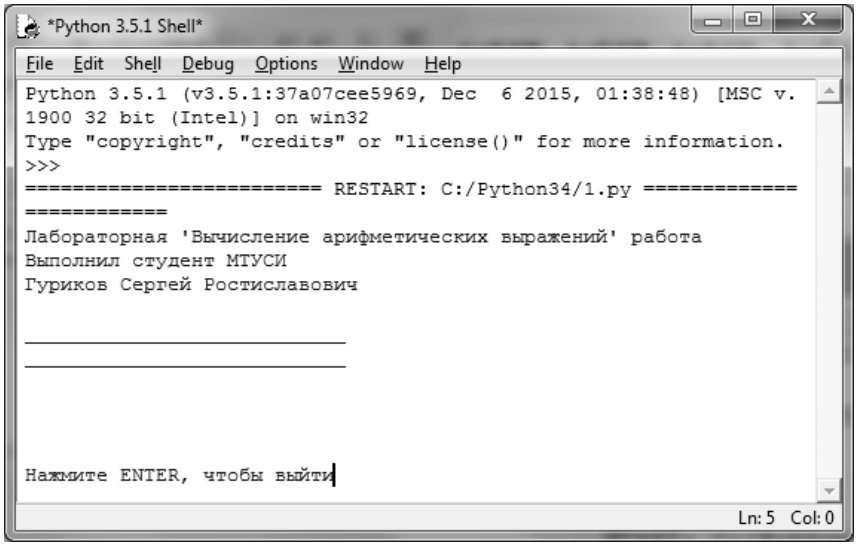


Рис. 1. Результат выполнения операций со строками

Логические типы данных (Булевский тип)

Переменные этого типа могут принимать одно из двух значений: **True** (Истина) или **False** (Ложь). Например, значение логического выражения **5>3** есть **Истина**, а значение **5=3** – **Ложь.** Значение условия **a>5** зависит от значения переменной **a** и не определено до тех пор, пока неизвестно значение величины **a.** Объект логического типа в Python обозначается как **bool.**

Применение функции **type** к объектам логического типа

type(True) или **type**(False)

вызовет ответ

<class 'bool'>, результатом **int(True)** будет 1, а **int(False)** – 0.

Над данными логического типа могут выполняться логические операции. Некоторые из них приведенные в табл. 1.

Таблица 1. **Логические операции**

Операция	Действие	Выражение
and (конъюнкция)	Логическое И	A **and** B
or (дизъюнкция)	Логическое ИЛИ	A **or** B
not (отрицание)	Логическое НЕ	**not** A

В Python, в отличие от некоторых других языков программирования, допустима проверка условия, записанного в таком виде: 1<a<5. Также допустима будет проверка условия (1<a) **and** (a<5), причем значение этого условия истинно только в том случае, если истинны оба входящих в него простых условия.

Мы рассмотрели основные типы данных, которые в дальнейшем будем использовать при изучении языка программирования Python. Еще раз отметим, что в соответствии с динамической типизацией, вы можете изменить тип переменной, просто присвоив ей иное значение. Например, в ходе выполнения операторов

```
a = 2017
a = "Новый год"
a = 3.14
```

переменная **a** последовательно меняет тип данных с целого на строковый, а потом – на вещественный.

1.8. Функции приведения типов

Как известно, компилятор должен иметь возможность определить тип каждой переменной. Кроме того, если одной переменной присваивается значение другой и при этом необходимо преобразование типов (например, от **float** к **int**), такое преобразование должно быть явным.

Приведем список функций языка Python, позволяющих осуществлять явные преобразования типов:

- **y=bool(объект)** – приводит объект к логическому типу. В переменной **y** будет храниться ссылка на объект логического типа, а не сам объект, хотя до выполнения оператора присваивания переменная y могла содержать ссылку на объект другого типа. Такая же ситуация будет и с другими функциями преобразования типов данных;
- **y=int(объект)** – приводит объект к целому типу;
- **y=float(объект)** – приводит объект к вещественному типу;
- **y=str(объект)** – приводит объект к строковому типу;
- **y=list(последовательность)** – преобразует элементы последовательности в список;
- **y=tuple(последовательность)** – преобразует элементы последовательности в кортеж.

Комментарии

Для лучшего понимания программы в ней часто записывается пояснительный текст – **комментарий**. Комментарии выполняют несколько важных функций:

- делают программу легко читаемой, поясняя смысл отдельных программных строк;
- временно отключают фрагменты программы при ее отладке.

В языке Python комментарий к программной строке возможен с использованием символа #. Например,

```
# Вычисляется сумма двух чисел
sum=a+b
```

1.9. Запись математических функций

В связи с невозможностью записи некоторых стандартных математических функций с клавиатуры персонального компьютера в языке Python существуют так называемые встроенные функции, с помощью которых пользователь записывает арифметические выражения.

Некоторые математические функции языка Python представлены в табл. 2. Прежде чем использовать математические функции, необходимо в начале программы написать инструкцию **import math,** однако тогда перед упоминанием каждой функции необходимо будет добавлять имя модуля – **math,** например, y=**math**.sin(x). Другой способ, который позволит избежать многократного вызова модуля **math**, – сделать следующую запись в начале программы: **from math import ***.

Таблица 2. **Общие математические функции модуля Math**

Функция	Запись на Python	Действие
Sin x	math.sin (x)	Возвращает значение функции Sin от числа x
Cos x	math.cos (x)	Возвращает значение функции Cos от числа x
Tg x	math.tan (x) или math.sin (x) / math.cos (x)	Возвращает значение функции Tg от числа x
Ctg x	math.cos (x) / math.sin (x)	Возвращает значение функции Ctg от числа x
$\lvert x \rvert$	math.abs (x)	Возвращает абсолютную величину числа x
e^x	math.exp (x)	Возвращает результат возведения числа **e** в степень x
Ln x	math.Log1p (x)	Возвращает натуральный логарифм от x+1
$\sqrt{X}$	math.sqrt (x)	Возвращает результат извлечения квадратного корня числа x
$Log_{10}x$	math.log (x)	Возвращает логарифм числа x по основанию 10
Cos^2x	math.cos (x) * math.cos (x)	Возвращает результат возведения функции Cos x в квадрат
Acos x	math.acos (x)	Возвращает значение функции арккосинус от числа x
Asin x	math.asin (x)	Возвращает значение функции арксинус от числа x
Atan x	math.atan (x)	Возвращает значение функции арктангенс от числа x
Pi	pi	Возвращает 3.141592653589793
Degrees(x)	math.degrees(x)	Преобразует радианы в градусы
Radians(x)	math.radians(x)	Преобразует градусы в радианы
Floor(x)	math.floor(x)	Возвращает значение, округленное до ближайшего меньшего целого
Ceil(x)	math.ceil(x)	Возвращает значение, округленное до ближайшего большего целого
Factorial(x)	math.factorial(x)	Возвращает факториал числа. 3!=1*2*3

В табл. 3 представлены некоторые встроенные функции для работы с числами, не требующие подключения модуля **math.**

Таблица 3. **Функции для работы с числами**

Функция	Запись на Python	Описание
Round (x)	round(x)	Возвращает результат округления числа x до ближайшего меньшего целого значения для чисел с дробной частью меньше 0.5 или результат округления числа x до ближайшего большего целого значения для чисел с дробной частью больше 0.5
Pow (x, y)	pow(x,y) другой вариант x**y	Возвращает результат возведения числа x в степень y
Max (x, y)	max(список чисел через запятую)	Возвращает большее значение из списка чисел
Min (x, y)	min(список чисел через запятую)	Возвращает меньшее значение из списка чисел
Sum(x,y)	sum(список чисел через запятую)	Возвращает сумму значений элементов последовательности
Int(объект)	int(объект)	Преобразует объект (например, строковое значение, дробное значение) в целое число
Float(объект)	float(объект)	Преобразует объект (например, строковое значение, целое значение) в вещественное число

Python предоставляет возможность совершить операции над числами, если программист укажет исходные данные (операнды) и математические действия в операторе **print,** например, инструкция **print(2016+40-20)** выведет на экран число 2036.

1.10. Операции отношения

Операции отношения представлены в табл. 4.

Таблица 4. **Операции отношения**

Операция	Описание
Операнд1 < Операнд2	Меньше
Операнд1 > Операнд2	Больше
Операнд1 <= Операнд2	Меньше или равно
Операнд1 >= Операнд2	Больше или равно
Операнд1 != Операнд2	Не равно
Операнд1 == Операнд2	Равно

Контрольные вопросы

1. Что называется алгоритмом? Какими свойствами он обладает?
2. Назовите и поясните способы описания алгоритмов.
3. Нарисуйте функциональные блоки, используемые в блок-схемах. Поясните их назначение.
4. Перечислите этапы решения задачи, выполняемые в процессе ее программирования.

5. Что входит в алфавит языка Python? Поясните понятие «идентификатор» и расскажите об общих правилах написания идентификаторов.
6. В чем заключается действие оператора присваивания? Каковы две формы записи дробных чисел?
7. Какова особенность динамической типизации, используемой в языке Python?
8. Дайте характеристику каждого типа данных языка Python.
9. Какие операции определены над данными целого типа?
10. Каково назначение операторов print и input? Приведите примеры использования таких операторов.
11. Какие логические операции могут выполняться над данными логического типа?
12. Назовите функции приведения типов. Приведите примеры.
13. Для каких целей используются комментарии в программах? Как можно закомментировать участок программного кода в Python?
14. Какие инструкции необходимо прописывать в программах, написанных на языке Python, для использования в них математических функций?

2. ВВЕДЕНИЕ В PYTHON

Разработка Python началась в конце 1980-х годов голландцем Гвидо ван Россумом, и первый релиз системы вышел в 1991 году. Python – кроссплатформенный язык программирования, способный работать на аппаратных платформах под управлением Windows, Mac OS, Linux. Если такой список вам показался маленьким, не огорчайтесь, разработчики языка отмечают, что Python может быть использован на компьютерах под управлением 21 операционной системы.

Python – объектно-ориентированный язык, но если понимание основ объектно-ориентированного программирования (ООП) в C#, Java затруднительно для начинающего программиста, то реализация ООП в Python элегантна и проста. Немаловажным преимуществом для человека, решившего начать путь в программирование именно с языка Python, является то обстоятельство, что его интерпретатор распространяется бесплатно и доступен для скачивания на сайте по адресу https://www.python.org/.

2.1. Процесс создания проекта в Python

Язык Python – активно развивающийся язык, и несколько раз в год появляются его новые версии, при этом основные возможности языка не зависят от нового релиза программы.

Рассмотрим процесс установки среды программирования и создания первого проекта. Перейдем на web-ресурс www.python.org, при этом откроется окно, представленное на рис. 2.

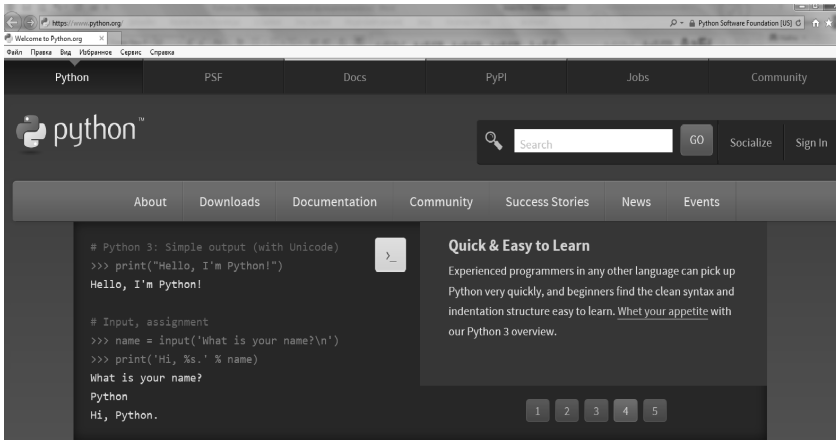


Рис. 2. Главная страница сайта www.python.org

Щелкнув по ссылке **Downloads,** вы сможете не только осуществить выбор операционной системы, под управлением которой работает ваш компьютер, но и версию языка Python. Подобная ситуация представлена на рис. 3.

Рис. 3. Страница выбора версии языка Python

Если компьютер работает под управлением Windows, вам будет предложено выполнить загрузку инсталляционного файла (рис 4.) python-3.x.x.exe (где x.x – номер версии программы), что и следует сделать, указав имя папки, куда будет скачиваться искомый файл.

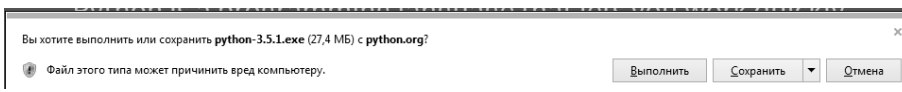


Рис. 4. Запрос о начале загрузки инсталляционного файла

В операционной системе Mac OS ваши действия будут такими же, а именно: выбор версии языка Python и загрузка инсталляционного файла в папку **Загрузки.** Затем необходимо выполнить щелчок на файле с именем python-3.x.x-macos10.6.pkg (где x.x – номер версии программы) (рис. 5) и следовать шагам Мастера по установке программы.

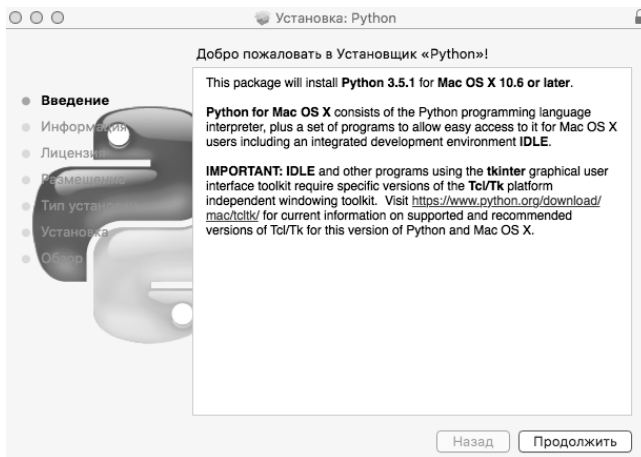


Рис. 5. Окно Мастера установки

В операционной системе Windows, щелкнув на скачанном файле python-3.x.x.exe, вам придется подождать несколько минут, пока не произойдет инсталляция Python на ваш компьютер (рис. 6).

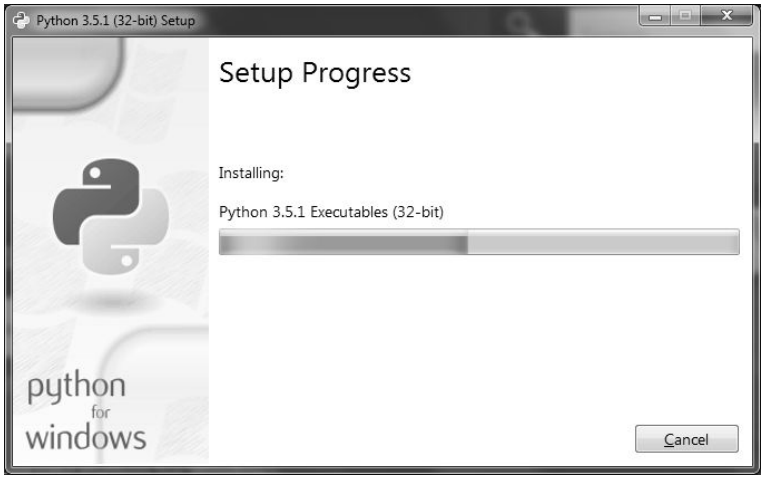


Рис. 6. Процесс инсталляции занимает несколько минут

Затем выполнив команду **Пуск/Python 3.x**, следует выполнить щелчок на ярлыке **IDLE (Python 3.x 32-bit)**, что показано на рис. 7.

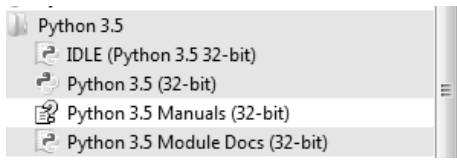


Рис. 7. Ярлыки установленной программы

Откроется окно, представленное на рис. 8.

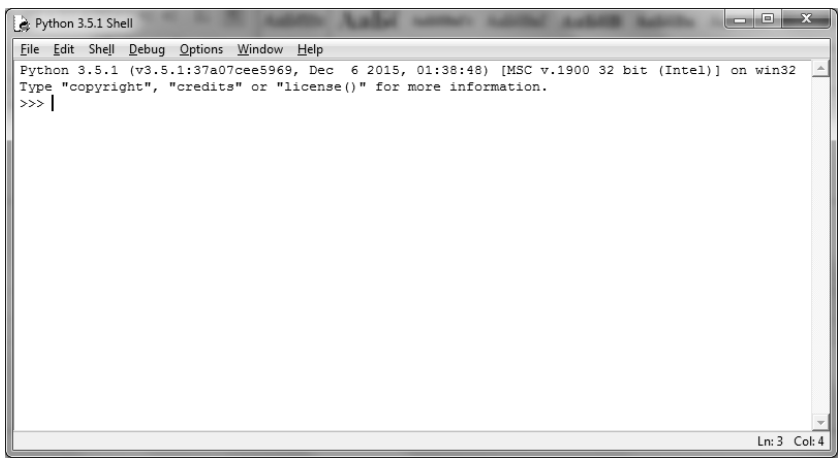


Рис. 8. Окно среды IDLE

IDLE (Integrated Development Environment) – это интегрированная среда разработки на языке Python, с ее помощью мы будем просматривать, редактировать, запускать и отлаживать программы написанные на Python. IDLE создана с помощью графической библиотеки под названием **Tkinter** (англ. Tk interface), которая широко распространена в операционных системах Linux и UNIX-подобных системах. IDLE является активным интерпретатором (программа-переводчик с языка высокого уровня в машинный код), поддерживающим функции многооконного редактора с функцией отмены, подсветкой синтаксиса, отладчика, и написана на Python.

Пункты меню оболочки – стандартные для подобных программ, поддерживающие стандартные функции сохранения, открытия, редактирования и отладки созданных программ, поэтому мы не будем приводить их отдельное описание, а познакомимся с ними непосредственно в ходе работы с IDLE.

Напишем нашу первую программу на языке Python, а именно – программу, которая выводит сообщение "**Привет, мир**!", представляющее собой неформальное приветствие другим людям от человека, делающего первый шаг в программировании на любом языке.

Ее мы напишем, используя оператор **print** и синтаксис данного оператора. **Оператор** (инструкция) в программировании – это действие, которое выполняет компьютер в ответ на запуск программы на выполнение. Каждый оператор имеет свой **синтаксис**, т. е. правила записи, согласно которым данный оператор может быть выполнен в той или иной среде программирования.

Оператор **print** имеет следующий синтаксис:

print("Сообщение")

Следует отметить, что оператор **print** необходимо записывать строчными буквами, поскольку язык Python чувствителен к регистру. Таким образом, оператор **Print** или **PRINT** – неправильно записанные операторы.

Итак, вы записываете оператор **print**("Привет, мир!") сразу после так называемого приглашения, которое в IDLE выглядит как тройное перечисление символа >, то есть **>>>**. Для того чтобы увидеть результат, вам следует нажать клавишу **Enter.** Результатом инструкции, естественно, станет вывод на экран сообщения "Привет, мир!" (рис. 9). Первый шаг в программирование на языке Python сделан.

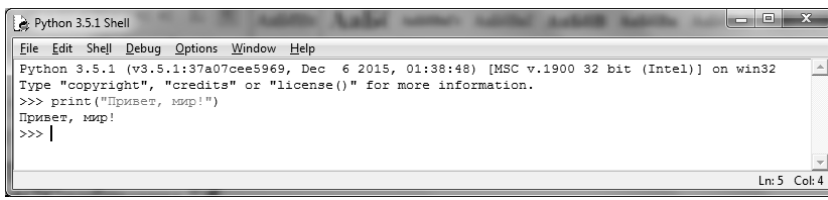


Рис. 9. Результат первой написанной программы

Сохраним нашу первую программу. Для этого выполним команду **File/Save.** Откроется окно, представленное на рис. 10. Для корректного сохранения своих программ лучше всего создать отдельную папку, в данном случае она имеет название **Мои проекты.** По умолчанию, если вы работаете в операционной систе-

ме Windows, программа Python будет установлена по следующему пути: **C:\Users*Имя пользователя на компьютере*\AppData\Local\Programs\Python\Python35-32**.

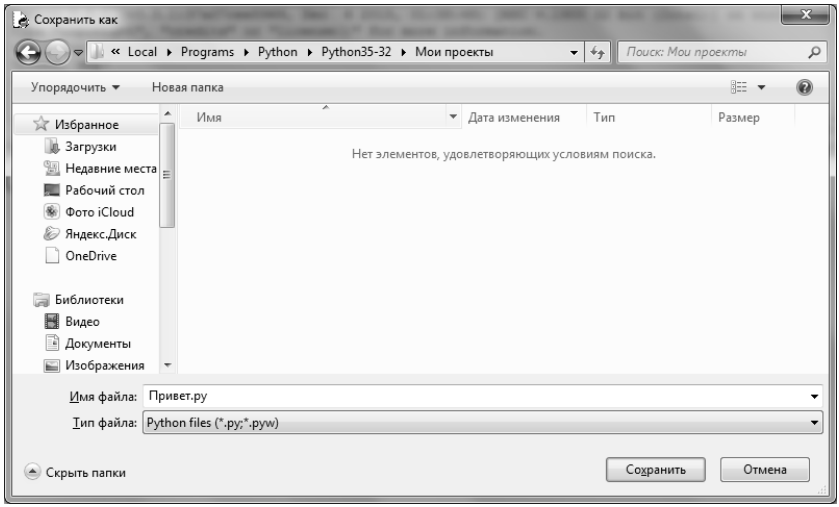


Рис. 10. Окно Сохранить как

Таким образом, если в дальнейшем вы хотите найти свои программы, непосредственно указывая к ним путь, вы будете следовать по вышеуказанному адресу.

Теперь рассмотрим другой способ создания наших программ в IDLE Python. Первый способ позволил сразу увидеть результаты написанной нами программы. Однако такой интерактивный режим, при котором в ответ на запись оператора пользователем происходит мгновенная реакция системы, будет совсем не удобен при разработке более серьезных программ, поскольку практически любая написанная в будущем программа, потребует выполнения такого этапа, как ее отладка. Следовательно, было бы очень удобно сначала написать предполагаемые действия в виде операторов программы, а затем написанный текст «запустить», как принято выражаться среди программистов, на выполнение.

Для этого, открыв ранее описанным способом IDLE, следует выполнить команду **File/New.** Откроется окно, представленное на рис. 11.

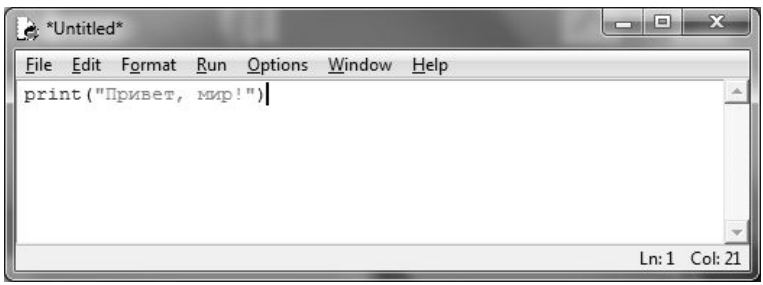


Рис. 11. Окно с программой

Именно в нем мы вновь запишем оператор **print**("Привет, мир!"). **Замечание**. Не делайте отступы от левой границы окна при записи оператора. С чем это связано, будет объяснено позже в параграфе 2.2.

Для того чтобы выполнить нашу программу, придется выбрать пункт меню **Run,** а в нем пункт **Run module.** Однако после этого появится не результат выполнения программы, а просьба о сохранении проекта (рис. 12). Заметим, что как только в дальнейшем вы сделаете любые изменения в вашей программе, даже если до этого она была сохранена, будет появляться окно с просьбой сохранить сделанные изменения и вам остается только нажать на кнопку **Ok**.

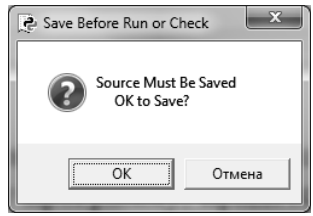


Рис. 12. Вопрос о сохранении программы

В настоящее время, нажав на кнопку **Ok,** следует указать путь к папке, которую вы создали ранее для своих будущих проектов. Поскольку вы уже сохраняли предыдущий проект, вы можете указать то же самое имя проекта, что и ранее. Появится окно, информирующее о том, что такой проект уже существует, и вам необходимо подтвердить свои действия.

Теперь открывается среда (оболочка) Python, в которой вы увидите результаты программы, а именно, сообщение "Привет, мир!" (рис. 13).

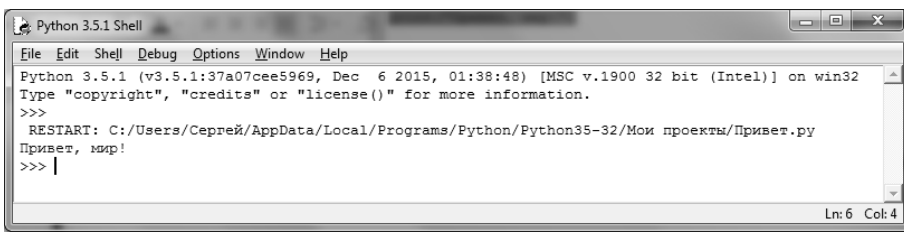


Рис. 13. Результат работы программы

Рассмотрим процесс открытия уже созданной программы. Выполним команду **Пуск/Python 3.x** и сделаем щелчок на ярлыке **IDLE (Python 3.x 32-bit).** В появившемся окне выполним команду **File/Open,** укажем путь к папке, в которой будут храниться проекты, написанные на языке Python, и выберем файл созданной нами программы "Привет, мир!" Подобная ситуация представлена на рис. 14.

Перед нами вновь откроется окно так называемого **сценарного режима**, в котором мы будем писать программы на Python (рис. 15). В нем вы увидите текст ранее созданной программы.

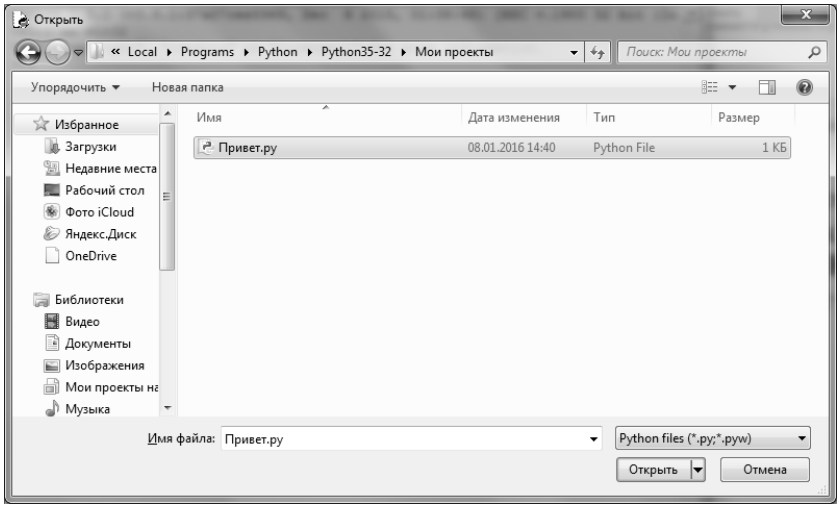


Рис. 14. Окно «Открыть»

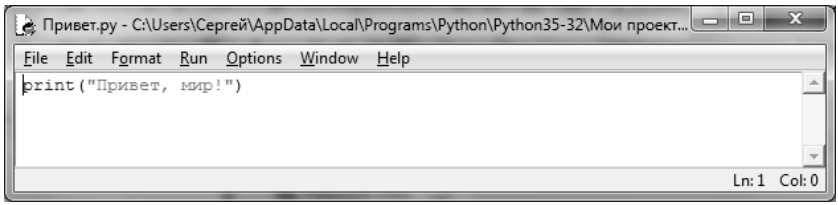


Рис. 15. Окно сценарного режима

Что делать дальше, вы уже знаете, однако напомним, что для запуска программы на выполнение следует выбрать пункт меню **Run,** а в нем пункт **Run module.** Согласно этой команде программа выполнится, и вы сможете увидеть ее результат.

2.2. Методы ввода и вывода данных и обработка исключений

Наступила пора написать более сложную программу, результатом выполнения которой станет сумма двух чисел, вводимых пользователем. На ее примере мы сможем рассмотреть новые операторы, позволяющие осуществить ввод данных и присваивание переменной какого-либо значения.

С действием, которое осуществляет **оператор присваивания,** мы познакомились ранее при чтении параграфа 1.6. Напомним, что в языке Python он имеет вид знака равенства (=).

Функция **input** будет использоваться в кодах программ для получения значений (**ввода данных**), которые будет вводить пользователь с клавиатуры. Она имеет следующий синтаксис:

имя переменной = **input**("Приглашение")

Такая запись очень похожа на синтаксис оператора **print.** Однако отличия все же имеются. Прежде всего, расскажем о самом понятии «**функция**». В ходе изучения языка программирования Python мы должны в совершенстве овладеть программированием на основе использования функций, но сейчас, в качестве небольшого знакомства, следует отметить: никого не удивляет тот факт, что при вычислении функции **sin(x)** пользователь, работая, например, с программой Microsoft Excel, выбирает из категории **Формулы/Вставка/Математика и тригонометрия** функцию **SIN** и начинает с ней работать, вычисляя значение этой функции от какого-то аргумента.

Между тем, функция SIN – это сложный знакопеременный математический ряд, который может быть вычислен по формуле:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1)!} + \ldots$$

Для вычисления суммы такого ряда в среде программирования необходимо написать достаточно сложную программу. Безусловно, легче вызвать уже написанную программистами функцию SIN, которая размещена в библиотеке математических подпрограмм, и работать с ней, используя ее имя – SIN.

Однако следует уяснить, что для получения результата от ее использования в среде программирования вы должны не просто указать ее имя sin и значение аргумента в скобках (sin(x)), но слева от имени функции указать имя переменной, а также оператор присваивания, т. е., вызов функции будет осуществлен в виде оператора **y=sin(x).**

Теперь надо запомнить, что понятие «функция» относится не только к категории математических действий, но и распространяется на другие. Когда мы будем использовать функцию **input** для того, чтобы разместить в ячейке оперативной памяти какое-либо значение, полученное от пользователя (в этом, напомним, заключается действие оператора присваивания), слева от этого оператора, мы должны будем указать имя переменной.

Учитывая вышесказанное, синтаксис функции **input**, записываемой как *имя переменной = **input**("Приглашение")*, не должен вызвать какие-либо вопросы.

В редакторе сценариев напишем программный код, приведенный на рис. 16.

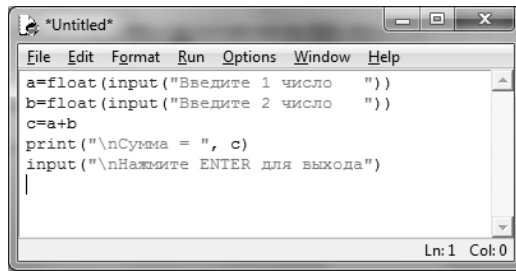


Рис. 16. Текст программы «Сложить два числа»

Для того чтобы было удобнее его прокомментировать, выделим его в отдельный листинг 1.

Листинг 1

```
a=float(input("Введите 1 число   "))
b=float(input("Введите 2 число   "))
c=a+b
print("\nСумма = ", c)
input("\nНажмите ENTER для выхода")
```

Вывод результата в программе может гарантировать оператор **print,** синтаксис которого несколько иной, чем в случае вывода на экран просто строки, а именно,

print("Приглашение", идентификатор), где

- *Приглашение* – строка, содержащая информацию о характере вывода;
- *идентификатор* – имя ячейки памяти, где хранится результат.

Можно еще раз отметить, что Escape-последовательность **\n,** содержащаяся в двух последних строках программы, дает указание интерпретатору перевести курсор на новую строку.

Что же может быть еще непонятно в коде программы **Сложить два числа** в настоящее время? Только применение функции **float** и то, что, несмотря на вышеприведенные рассуждения, функция **input** вызывается без упоминания имени переменной слева от оператора присваивания, который тоже отсутствует в строке программного кода **input("\nНажмите ENTER для выхода")**.

Тем не менее, все должно быть интуитивно понятно. В частности, при объяснении типов данных, используемых в языке Python (см. параграф 1.7), упоминался вещественный тип данных (**float**), используемый, когда программа должна работать с дробными числами. Так вот, функция **input**, как принято говорить, возвращает строковый тип данных. То есть то число, которое будет принято от пользователя, будет восприниматься компьютером не как число, а как строковое значение, а следовательно, никакие математические операции над строками выполнить мы не сможем. Таким образом, функция **float** играет роль функции приведения одного типа данных к другому (в данном случае конвертирует строковый тип в вещественный).

Оператор **input("\nНажмите ENTER для выхода")**, в целом, необязателен. Если его не будет в программе, то, конечно, не изменится ее результат с точки зрения математики, и, кроме того, после получения результата и нажатия **ENTER,** как такого выхода из программы не произойдет. Осуществится лишь переход к приглашению >>>. Тем не менее, психологически это готовит пользователя к тому, что его программа завершена и надо принять какое-то решение: либо осуществить выход из среды программирования, либо возобновить действия с программой.

К тому же использование функции **input** в данном операторе не приводит к получению числового значения от пользователя, а только лишь к считыванию внутреннего кода клавиши **ENTER**, который непосредственно не может изменить результат математического действия сложения чисел. Следовательно, возможен и другой синтаксис функции **input,** который и продемонстрирован в примере. Результат ввода данных и получения результата представлен на рис. 17.

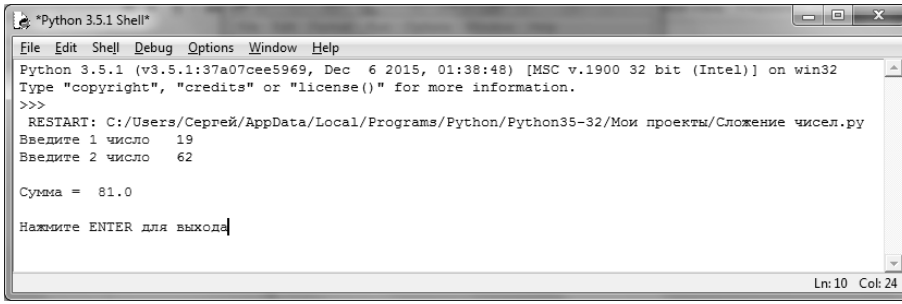


Рис. 17. Результат работы программы

Прерывания, которые будут рассматриваться в данном параграфе, относятся к классу внутренних прерываний и называются **исключениями** (exceptions). Они происходят синхронно выполнению программы и возникают при появлении аварийной ситуации в ходе исполнения некоторой инструкции. Примерами исключений являются деление на нуль, переполнение, обращение к несуществующему файлу и т. д.

Обработчик ошибок в Python использует блок **try…except…finally.** Блок **try…except** должен окружать ту часть кода, где может возникнуть исключительная ситуация. Блок **Finally** всегда исполняется, поэтому в него помещают те инструкции, которые должны выполниться независимо от того, произошло ли исключение.

Программа может прервать свою работу по разным причинам, поэтому типов исключений существует довольно много. В табл. 5 приведем наиболее распространенные среди них.

Таблица 5. **Типы исключений**

Тип исключения	Описание
IOError	Возникает при появлении ошибок, связанных с операциями ввода/вывода
IndexError	Возникает, если в последовательности не найден элемент с указанным индексом
NameError	Возникает, если не найдено имя переменной, имя функции
SyntaxError	Возникает в случае синтаксической ошибки в программе
TypeError	Возникает, если стандартная операция применяется к объекту несоответствующего типа
ValueError	Возникает, если стандартная операция применяется к объекту соответствующего типа, но с неподходящим значением
ZeroDivisionError	Возникает в случае выполнения операции деления на нуль

Приведем пример использования конструкции **try…except…finally** и напишем программу для получения частного от деления двух чисел. Ее код приведен в листинге 2.

Листинг 2

```
a=float(input("Введите 1 число   "))
b=float(input("Введите 2 число   "))
try:
```

```
    c=a/b
    print("\n Частное от деления = ", c)
except ZeroDivisionError:
    print ("Вы делите на нуль!")
finally:
    print("Давайте запустим программу еще раз или \nНажмите ENTER для выхода")
```

Обратите внимание на отступы в данном коде. Дело в том, что в отличие от других популярных языков программирования, таких как C#, Microsoft Visual Basic, Pascal-ориентированных языков, где отступы в блоках кода советуют делать для того, чтобы программу было легче читать и отлаживать, но сам принцип их простановки необязателен, в Python **отступы** являются частью синтаксиса программных конструкций. Для того чтобы сделать отступ в строке кода, достаточно нажать клавишу **Tab** или нажать четыре раза клавишу **Пробел.**

Базовой программной конструкцией в вышеприведенном коде является конструкция **try…except…finally,** поэтому стоит не сделать отступ хотя бы в одной программной строке этого блока, так, например, как это показано на рис. 18 (не сделан отступ в операторе c=a/b), и программа перестанет работать, выдавая ошибку, показанную на рис. 19.

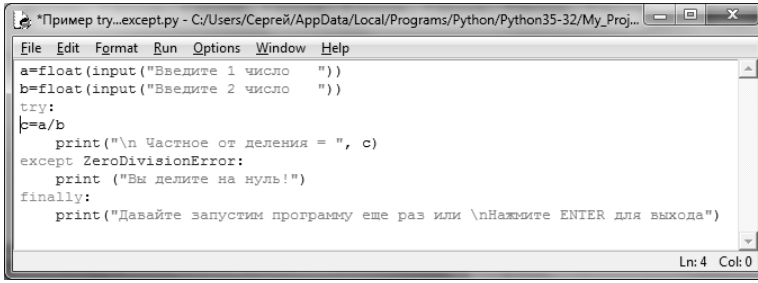


Рис. 18. Не сделан отступ в операторе c=a/b

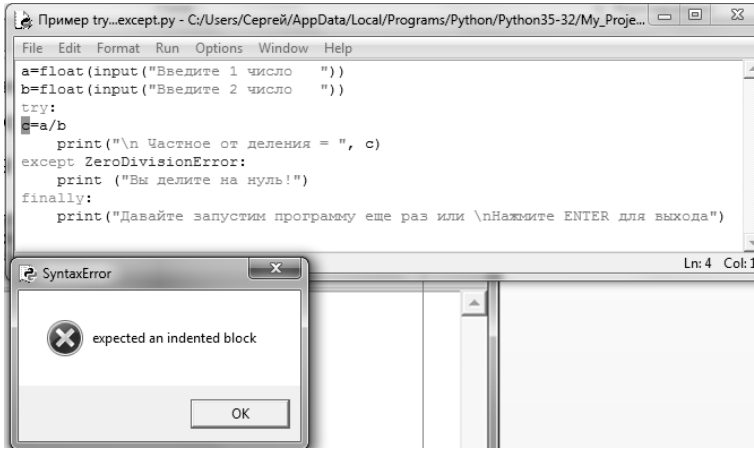


Рис. 19. Ошибка, полученная из-за отсутствия отступа

Также обратите внимание на то, что если вы сделаете отступ в строке программного кода, которая не относится к блоку операторов, например, в **a=float(input("Введите 1 число "))**, то программа также перестанет функционировать и интерпретатор выдаст ошибку.

Значит, если не соблюдать такие правила, начинающему будет очень трудно без посторонней помощи разобраться в сложившейся ситуации. В дальнейшем мы будем давать специальные комментарии по поводу необходимости делать отступы в тех или иных программных конструкциях.

Вернемся к комментарию кода программы. Мы окружили код, где возможно возникновение ошибки деления на нуль, блоком **try…except…finally**, при этом использовали ожидаемый тип исключения **ZeroDivisionError** (см. табл. 5). Приведенный блок перехватывает конкретную ошибку переполнения, возникающую вследствие деления на нуль.

Предположим, мы введем число 5, а второе число – равное 0. При выполнении этого кода появится сообщение: «Вы делите на нуль!». Результат работы программы показан на рис. 20.

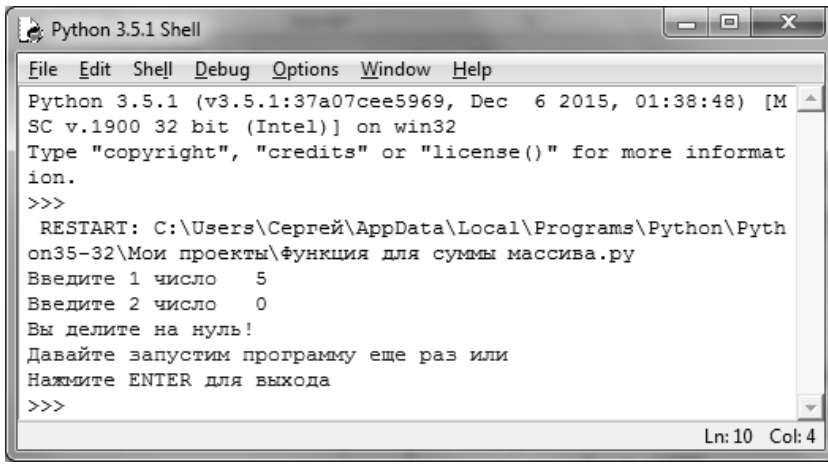


Рис. 20. Программа выдает сообщение «Вы делите на нуль!»

Попробуем усовершенствовать программу. Дело в том, что наша программа не совсем корректна: ведь пользователь по ошибке может вместо чисел ввести обычные символы, расположенные на клавиатуре. Нижеприведенный код, представленный в листинге 3, перехватит другую ошибку, связанную как раз с неверным форматом ввода. В нем, во-первых, оператор **try** разместим в том месте, где возможно возникновение подобной ошибки – это инструкции ввода данных. Во-вторых, воспользуемся тем, что обработка нескольких исключений может быть перехвачена с помощью нескольких вложений конструкции **except,** и включим в наш код исключение **ValueError** (см. табл. 5), которое возникает, если стандартная операция применяется к объекту соответствующего типа, но с неподходящим значением. Наш код примет следующий вид.

Листинг 3

```
try:
   a=float(input("Введите 1 число   "))
   b=float(input("Введите 2 число   "))
   c=a/b
   print("\nЧастное от деления = ", c)
except ZeroDivisionError:
   print ("Вы делите на нуль!")
except ValueError:
   print ("Вы ввели не числовое значение!")
finally:
   print("Давайте запустим программу еще раз")
   input("\nНажмите ENTER для выхода")
```

Теперь программа надежно защищена от неправильного ввода. Предположим, значение первого числа пользователь вводит правильно 5.2, а при вводе второго числа пользователь ошибается и вводит в качестве разделителя не точку, а символ **ю** (поскольку и точка, и символ **ю** расположены на одной клавише). Однако такая ошибка не приведет к исключительной ситуации с точки зрения остановки программы. При ее выполнении появится сообщение «Вы ввели не числовое значение!» (рис. 21).

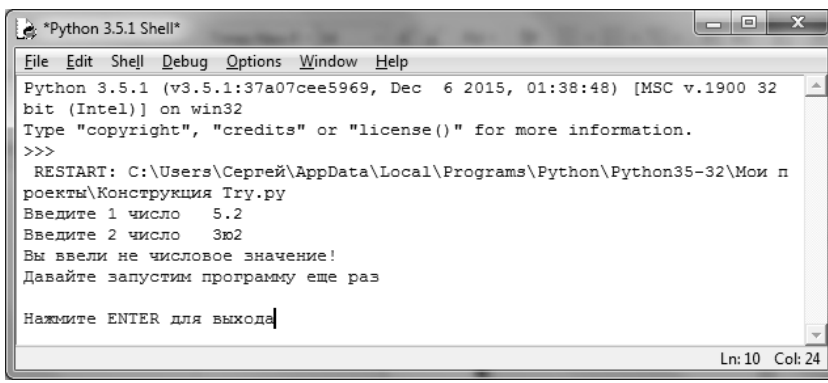


Рис. 21. Программа выдает сообщение «Вы ввели не числовое значение!»

Итак, обработка нескольких исключений может быть перехвачена с помощью нескольких вложений конструкции **except**. Такая ситуация была показана в предыдущем коде. Другой прием заключается в перечислении через запятую типов исключений – так, как это показано в следующей программе.

Листинг 4

```
try:
   a=float(input("Введите 1 число   "))
   b=float(input("Введите 2 число   "))
   c=a/b
```

```
    print("\nЧастное от деления = ", c)
except (ZeroDivisionError, ValueError):
    print ("Вы делите на нуль или вы ввели не числовое значение!")
finally:
    print("Давайте запустим программу еще раз \nНажмите ENTER для выхода")
```

Результат работы программы будет аналогичен представленному на рис. 21, однако код программы стал короче.

Контрольные вопросы

1. Расскажите о назначении IDLE. С какими способами создания программ в IDLE вы познакомились?
2. Какие операторы ввода и вывода данных используются для приложений, разрабатываемых на языке Python? Напишите синтаксис используемых операторов.
3. В каких случаях при разработке концепции глобальной обработки ошибок применяется конструкция try…except…finally? Поясните работу обработчиков исключений на примерах.
4. Назовите основные типы исключений и укажите причины их возникновения.
5. Какова роль отступов в программах, написанных на языке Python?

3. ЛИНЕЙНЫЙ АЛГОРИТМ

Алгоритм называется **линейным,** если он содержит N шагов, и все шаги выполняются последовательно друг за другом от начала до конца. Общий вид линейного алгоритма представлен на рис. 22, где P_1, P_2, … P_n – операторы.

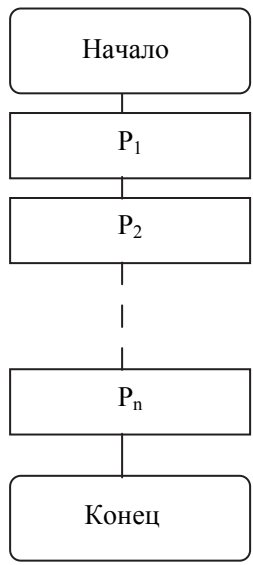


Рис. 22. Общий вид линейного алгоритма

Выполнение упражнений, представленных в некоторых главах учебного пособия, позволит вам лучше понять смысл работы того или иного оператора языка программирования Python. Однако перед тем как приступить к выполнению упражнений, мы еще раз напомним, что **присваивание в Python означает связывание значения с некоторым именем переменной.** Тем не менее, при объяснении материала, связанного с операторами присваивания, мы будем использовать более упрощенный вариант и выражаться в общем виде так: «**в ячейку N заносится число M**».

3.1. Упражнения

Вопрос 1. Какими будут значения переменных m и n после выполнения группы операторов?

```
m=25
n=m+1
m=m-25
```

```
print("Ответ= ", n)
print("Ответ= ", m)
```

Ответ. Действие оператора присваивания заключается в занесении какого-либо значения в соответствующую ячейку памяти, причем старое значение, находящееся в ячейке, исчезает. Таким образом, если в ячейке **m** находится число 25, то в ячейку **n** после выполнения оператора присваивания **n=m+1** заносится число 26. Затем выполняется оператор **m=m–25.** Значение переменной **m** не изменялось, поэтому 25–25=0, т. е. значение переменной **m=0,** а значение переменной **n=26.**

Вопрос 2. Каким будет значение переменной n после выполнения группы операторов?

```
m=20
n=10
m=m/n
n=m*n
n=n+30
print("Ответ= ", n)
```

Ответ. После выполнения оператора присваивания **m=m/n** в ячейку **m** заносится число 2. Выполняется оператор **n= m*n.** В ячейке **m** находится число 2, а в ячейке **n** – число 10, поэтому после выполнения операции умножения в ячейку **n** заносится число 20. Выполняется оператор **n=n+30,** и так как в ячейке **n** находится 20, то значение переменной **n** будет равно **50.**

Вопрос 3. Каким будет значение переменной m после выполнения группы операторов?

```
m=30
n=2
n=m/2
m=n
m=m+n
m=m
print("Ответ = ", m)
```

Ответ. После выполнения оператора **n=m/2** в ячейку **n** заносится число 15. После выполнения оператора **m=n** в ячейке **m** будет также находиться число 15. Следовательно, число 30 заносится в ячейку **m** после выполнения оператора **m=m+n.** Оператор **m=m** переприсваивает значение переменной **m. Ответ в упражнении**: значение переменной **m** равно **30.**

Вопрос 4. Какое число будет выведено в качестве ответа после выполнения группы операторов?

```
m=11
n=m*2
```

```
m=m+n
m=m
print("Результат =", m)
```

Ответ. После выполнения оператора **n=m*2** в ячейку **n** заносится число 22, которое складывается с числом 11 после выполнения оператора **m=m+n.** Оператор **m=m** переприсваивает значение переменной **m.** Следовательно, на экран выводится число **33.**

Вопрос 5. Какое значение будет находиться в ячейке a после выполнения группы операторов?

```
a=26
a="Папа"
b=a
b=54
b=False
a=b
```

Ответ. Динамическая типизация, используемая в Python, позволяет связывать переменную с типом данных в момент присваивания значения. Следовательно, размещая в ячейке **a** сначала числовое значение (a=26), затем строковое (a="Папа"), мы не нарушаем никак правил, связанных с типами данных в Python. Поскольку последним оператором, который выполнится в данном фрагменте программы, будет оператор **a=b**, а в **b** находилось логическое значение **False**, соответственно, и в ячейке **a** будет размещено значение **False.**

Вопрос 6. Какое значение будет находиться в ячейке d после выполнения группы операторов?

```
a=3.333
b=3.332
c=min(a,b)
x=round(c)
d=pow(x,3)
```

Ответ. Упражнение предполагает наличие у читателя знаний об основных математических функциях языка Python. Напомним, что функции min(), round() и pow() не требуют подключения модуля **math**. Функция min() с аргументами **a** и **b** вернет число **3.332** как меньшее из двух чисел. Результатом применения функции round() к аргументу **c** станет значение **3**, поскольку будет возвращен результат округления до ближайшего меньшего целого значения для чисел с дробной частью меньше **0.5**. Функция pow() возвращает результат возведения числа **x** в степень **y**, следовательно, ответом в упражнении станет число **27**.

3.2. Примеры решения задач

Задача 1. Вычислите значения арифметических выражений и выведите на экран результаты вычислений.

Исходные данные: x=1,4444 b=0,318 t=2,1 a=1,3

$$y=9x^2+\mathrm{Sin}^2x\sqrt{a-b}$$

$$z=\sqrt[3]{x^t}\left(ax^3-\frac{x^2}{2!}\right)$$

Запишем арифметические выражения на языке программирования.

y=9*x*x+sin(x)*sin(x)*sqrt(a–b)

z=exp(1/3*log(pow(x,t)))*((a*x*x*x-(x*x)/(1*2)))

Разработка алгоритма решения задачи представлена на рис. 23.

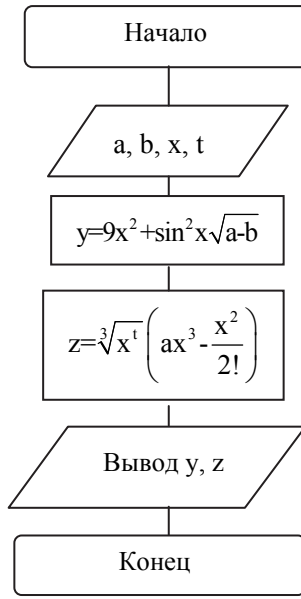


Рис. 23. Алгоритм решения задачи

Код программы, отвечающий за решение задачи, приведен в листинге 5. Мы ввели дополнительную переменную **z1** для вычисления значения **x,** возведенного в степень **t,** для более короткой записи в выражении, вычисляющем значение **z.** Оператор **exit(0)** вызовет появление окна сообщения (рис. 24), в котором спрашивается о том, хотите ли вы завершить запущенный на выполнение процесс (программу). В случае положительного ответа программа прекратит свою работу и, кроме того, закроется интерактивная сессия Python.

Листинг 5

```
from math import *
a = float(input("Введите значение a   "))
```

```
b = float(input("Введите значение b   "))
x = float(input("Введите значение x   "))
t = float(input("Введите значение t   "))
y = 9*x*x+sin(x)*sin(x)*sqrt(a-b)
z1 = log(pow(x,t))
z = pow(z1,(1/3))*((a*x*x*x-(x*x)/(1*2)))
print("\nЗначение y = ", y)
print("\nЗначение z = ", z)
exit(0)
```

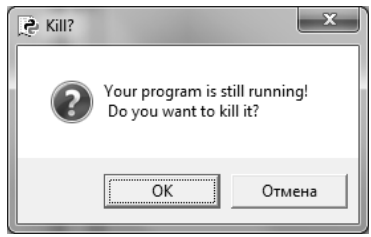


Рис. 24. Программа запрашивает подтверждение о ее остановке

Результат работы программы представлен на рис. 25.

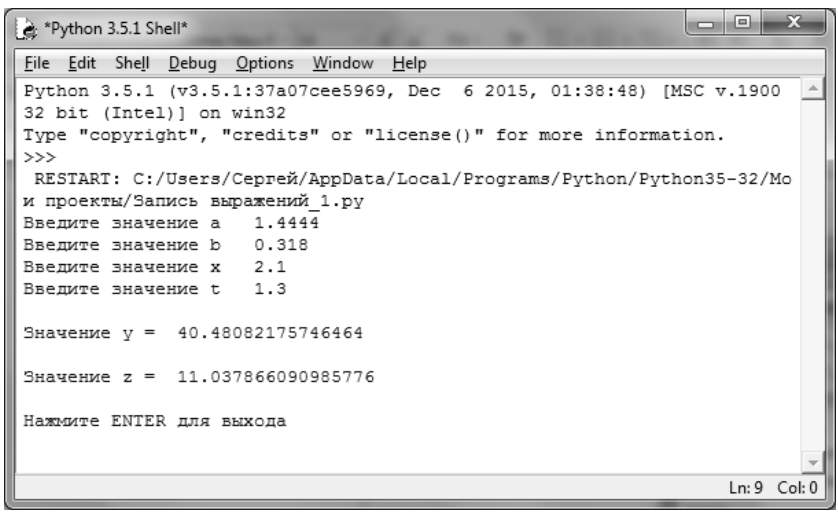


Рис. 25. Результат работы программы

Ограничить количество знаков после точки в вещественной записи числа поможет метод **format.** С самим понятием «метод» мы познакомимся позже, а сейчас посмотрим, как применить его к выводу данных для того, чтобы результат стал более читаемым.

```
print("\nЗначение y = ",'{0:.3f}'.format(y))
print("\nЗначение z = ",'{0:.3f}'.format(z))
```

Метод **format** записывается сразу после точки и применяется к результирующей ячейке, для которой мы хотим уменьшить количество знаков. В фигурных скобках указывается так называемый **спецификатор формата**. **0** – это поле, определяющее индекс позиции, при этом нумерация начинается с нуля (можно задать ширину поля, например 0:10, тогда выводимое число будет «отодвинуто» вправо на 10 позиций), **.3** – точность вычислений, т. е. количество знаков после точки, **f** – вещественное число в десятичном представлении.

Результат работы программы с методом **format** представлен на рис. 26.

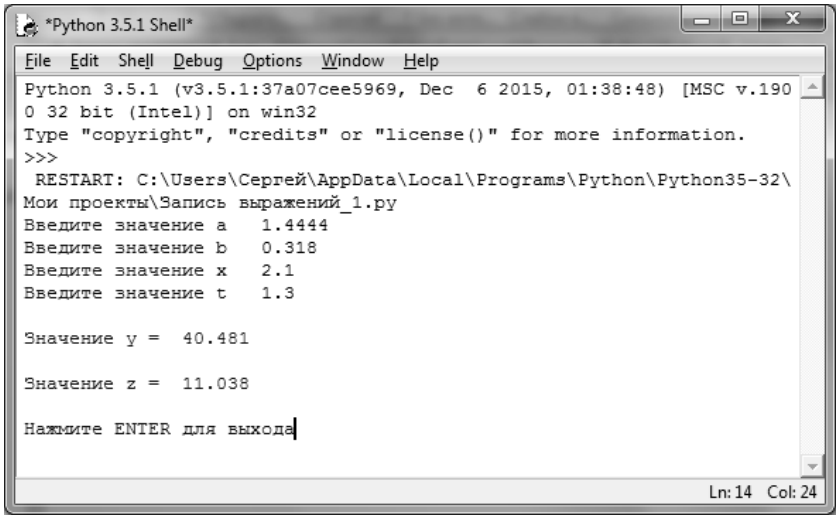


Рис. 26. Результат работы программы с использованием метода **format**

Задача 2. Найдите емкость конденсатора **C**, если известны: площадь пластин **s** и расстояние между ними **L**. Следует учесть, что в конденсатор вставлена металлическая пластина толщиной **d**, параллельная его обкладкам.

$$C = \frac{E_0 S}{L - d},$$

где C – емкость конденсатора;

E_0 – электрическая постоянная, равна $8{,}85 \cdot 10^{-12}$.

Разработка алгоритма решения задачи представлена на рис. 27.

Код программы, отвечающий за решение задачи, приведен в листинге 6.

Листинг 6

```
p=float(input("Введите площадь пластин   "))
l=float(input("Введите расстояние между пластинами   "))
d=float(input("Введите толщину пластины   "))
E0=8.85e-12
c=(E0*p)/(l–d)
print("\nЕмкость конденсатора ",'{0:10.3f}'.format(c))
input("\nНажмите ENTER для выхода")
```

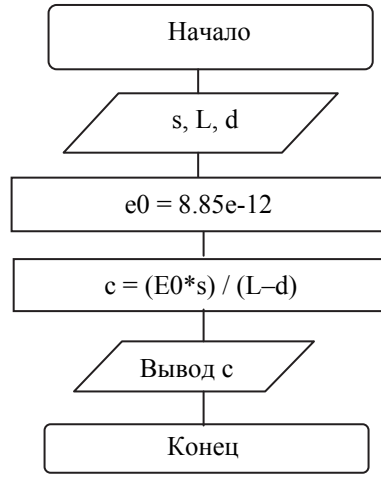


Рис. 27. Алгоритм решения задачи

Контрольные вопросы

1. Какой алгоритм называется линейным?
2. Нарисуйте общий вид линейного алгоритма.
3. С какой целью используется оператор exit(0) в программах, написанных на языке Python?
4. Поясните назначение метода format и приведите примеры его применения.

Задачи для самостоятельного решения

1. Разработайте алгоритм и программу, в которой вычисляются площадь и объем сферы.

$$S_{сф}=4\pi R^2 \qquad V_{сф}=\frac{S\cdot R}{3}.$$

2. Разработайте алгоритм и программу, в которой подсчитывается количество часов минут и секунд в заданном числе суток.

$$ch=24\cdot S$$
$$min=60\cdot ch$$
$$sec=60\cdot min.$$

3. Разработайте алгоритм и программу, в которой вычисляется площадь треугольника по трем сторонам. Вычисление проводиться по формуле Герона.

$$S=\sqrt{p(p-a)(p-b)(p-c)},$$

где S – площадь треугольника;
p=(a+b+c)/2 – полупериметр;
a, b, c – длина стороны треугольника.

4. Разработайте алгоритм и программу решение системы линейных уравнений:

$$a_1x+b_1y=c_1$$

$$a_2x+b_2y=c_2$$

по правилу Крамера:

$$X=\frac{c_1b_2-c_2b_1}{a_1b_2-a_2b_1}.$$

5. Разработайте алгоритм и программу определения объемов цилиндра и конуса с радиусом основания R=5 см и высотой H=8 см.

$$Vц=h\pi R^2$$

$$Vк=\frac{1}{3}h\pi R^2.$$

6. Разработайте алгоритм и программу определения общего сопротивления электрической цепи, если имеются три резистора R_1, R_2, R_3.

$$\frac{1}{R}=\frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3};\ R=\frac{R_1R_2R_3}{R_2R_3+R_1R_3+R_1R_2}$$

4. РАЗВЕТВЛЯЮЩИЙСЯ АЛГОРИТМ

До сих пор мы учились вводить данные и присваивать значения переменным. Теперь надо научиться организовывать различные потоки выполнения приложения в зависимости от ситуации, которая складывается в ходе работы программы.

Алгоритм называется **разветвляющимся,** если последовательность выполнения шагов алгоритма изменяется в зависимости от выполнения некоторых условий. **Условие** – это логическое выражение, которое может принимать одно из двух значений: «ДА» – если **условие верно** (истинно), и «НЕТ» – если **условие неверно** (ложно).

Разветвляющийся алгоритм можно реализовать в программах с помощью простого, сокращенного, составного операторов, а также конструкции многозначных ветвлений. Рассмотрим их более подробно.

4.1. Простой условный оператор

Общий вид в алгоритме конструкции простого условного оператора представлен на рис. 28.

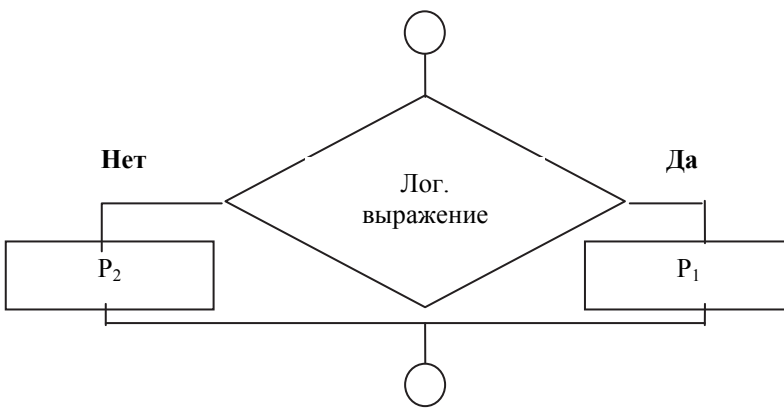


Рис.28. Блок-схема простого условного оператора

Синтаксис простого условного оператора следующий:

if Логическое выражение**:**
    P_1
else:
    P_2

где **if** (если), **else** (иначе) – зарезервированные слова, а P_1, P_2 – операторы.

Простой условный оператор работает по следующему алгоритму: сначала вычисляется логическое выражение. Если результат есть **true** (Истина), то выполняется оператор $\mathbf{P_1}$, а оператор $\mathbf{P_2}$ пропускается. Если результат есть **false** (Ложь), то выполняется оператор $\mathbf{P_2}$, а оператор $\mathbf{P_1}$ пропускается.

Например, в следующем коде в зависимости от результата сравнения двух переменных выводится тот или иной ответ. Оператор **if** – это блочный оператор, поэтому отступы в коде обязательны. Вход в блок операторов осуществляется двоеточием.

Листинг 7

```
a = int(input("Введите значение a   "))
b = int(input("Введите значение b   "))
if a<b:
   print("Тест 1 ветви")
else:
    print("Тест 2 ветви")
```

4.2. Сокращенный условный оператор

Если необходимо выполнить некоторое действие только при истинности проверяемого условия, то в таком случае применяется сокращенный условный оператор. Общий вид в алгоритме конструкции сокращенного условного оператора представлен на рис. 29.

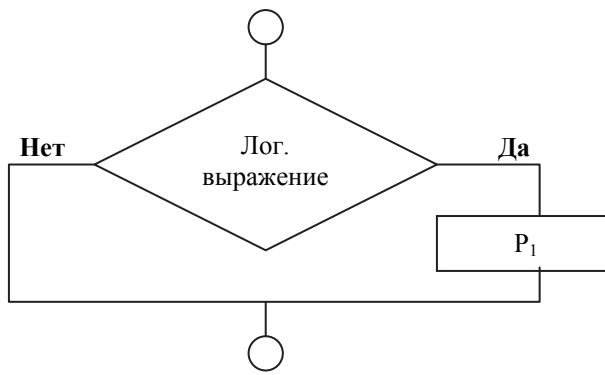


Рис. 29. Блок-схема сокращенного условного оператора

Синтаксис сокращенного условного оператора следующий:

if Лог. выражение**:**
P_1

где **if** (если) – зарезервированное слово, а $\mathbf{P_1}$ – оператор.

Например, в листинге 8 при истинности логического выражения выводится соответствующее сообщение «Тест 1 ветви».

Листинг 8

```
a = int(input("Введите значение a   "))
b = int(input("Введите значение b   "))
if a<b:
   print("Тест 1 ветви")
```

4.3. Составной условный оператор

Если при некотором условии надо выполнить определенную последовательность операторов, то их объединяют в один составной оператор. Общий вид в алгоритме конструкции составного условного оператора представлен на рис. 30.

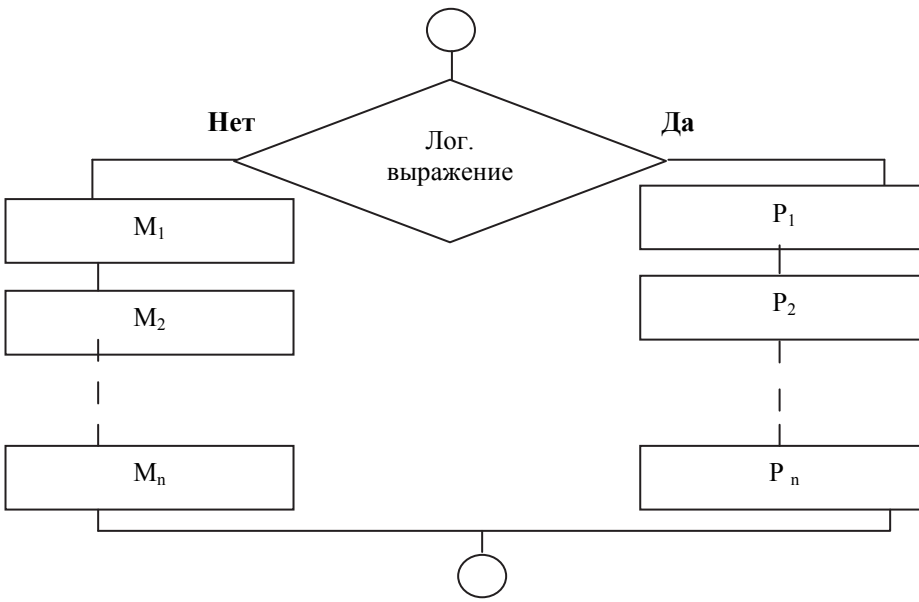


Рис. 30. Блок-схема составного условного оператора

Синтаксис составного условного оператора следующий:

if Лог. выражение**:**
   P_1
   P_2
   .
   .
   P_n
else:
   M_1
   M_2
   .
   .
   M_n

где **if**, **else** – зарезервированные слова, а P_1, P_2,… P_n, M_1, M_2,… M_n – операторы.

Например, в листинге 9 в каждой ветви условного оператора мы хотим выполнить по два оператора. Тогда каждый из них мы должны сместить вправо ровно на четыре пробела от начала строки.

Листинг 9

```
a = int(input("Введите значение a   "))
b = int(input("Введите значение b   "))
if a < b:
   a=a+b
   print("Сумма двух чисел =", a)
else:
   a=a*b
   print("Произведение двух чисел =", a)
```

4.4. Многозначные ветвления

Очень часто приходиться выбирать путь решения задачи не из двух, а из нескольких возможных. В программировании данный ход можно реализовать, используя несколько условных операторов. Общий вид в алгоритме конструкции многозначных ветвлений представлен на рис. 31.

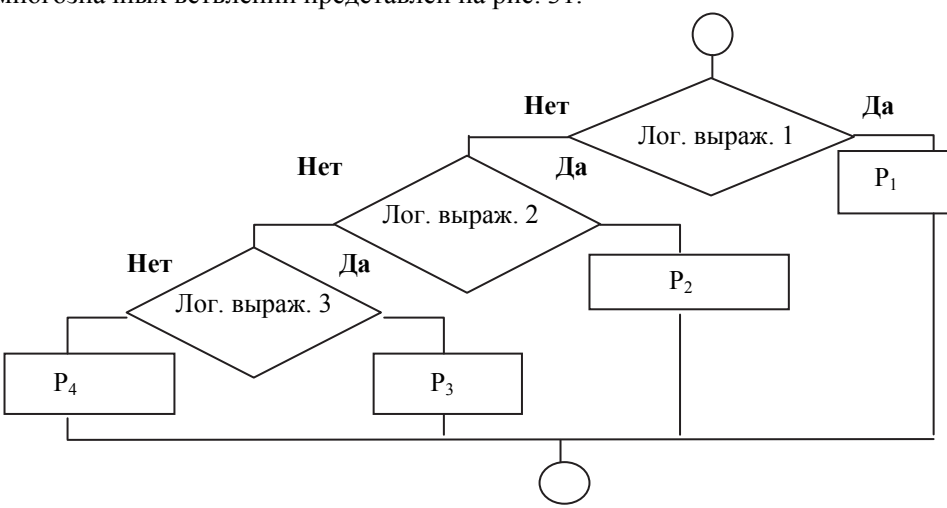


Рис. 31. Блок-схема конструкции многозначных ветвлений

Синтаксис условного оператора в конструкции многозначных ветвлений следующий:

if Лог. выражение1**:**
    P_1
elif Лог. выражение2**:**
    P_2
elif Лог. выражение3**:**

P_3

else

P_4

где **if, elif, else** – зарезервированные слова, а P_1, P_2, P_3, P_4 – операторы.

Алгоритм работы такой конструкции состоит в следующем. Если **Логическое выражение1** истинно, то выполняется оператор или блок операторов, следующих в данной ветви, в противном случае этот оператор или блок пропускается. Если логическое выражение, следующее за оператором **if,** ложно, то анализируется **Логическое выражение2,** следующее за оператором **elif.** Если оно истинно, то выполняется оператор или блок операторов, следующих в данной ветви, в противном случае этот оператор или блок пропускается. Операторы, следующие за последним **else,** выполняются лишь в том случае, если ложны все предыдущие логические выражения. Условные операторы **if** в такой конструкции называются **вложенными.**

Например, в листинге 10 показан процесс тестирования трех ветвей программы. Пользователь может менять исходные данные, которые будут находиться в ячейках **a** и **b**, и всякий раз получать тот или иной результат.

Листинг 10

```
a = int(input("Введите значение a   "))
b = int(input("Введите значение b   "))
if a < b:
   a=a+b
   print("Сумма двух чисел =", a)
elif a==b:
   a=a*b
   print("Произведение двух чисел =", a)
else:
   a=a-b
   print("Разность двух чисел =", a)
```

4.5. Алгоритмы поиска максимального и минимального элементов

Рассмотрим алгоритмы нахождения максимального и минимального значений, которые будут востребованы при дальнейшем изучении языка программирования Python.

Задача 1. Найдите максимальное из двух чисел. Разработка алгоритма решения задачи представлена на рис. 32.

Листинг 11

```
a = int(input("Введите значение a   "))
b = int(input("Введите значение b   "))
if a >= b:
```

```
    max=a
else:
    max=b
print("Максимальное из двух чисел =", max)
```

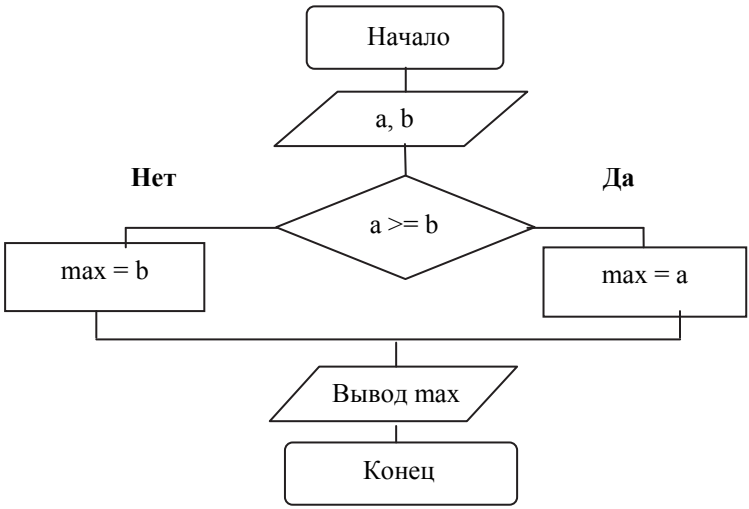


Рис. 32. Алгоритм нахождения максимального из двух чисел

Задача 2. Найдите минимальное из трех чисел. Алгоритм нахождения максимального или минимального элемента может быть запрограммирован несколькими способами. Фрагмент разработки алгоритма решения задачи (первый способ) представлен на рис. 33. Очевидно, что при большом количестве чисел, из которых нужно осуществить выбор экстремального значения, алгоритм станет очень громоздким и запутанным.

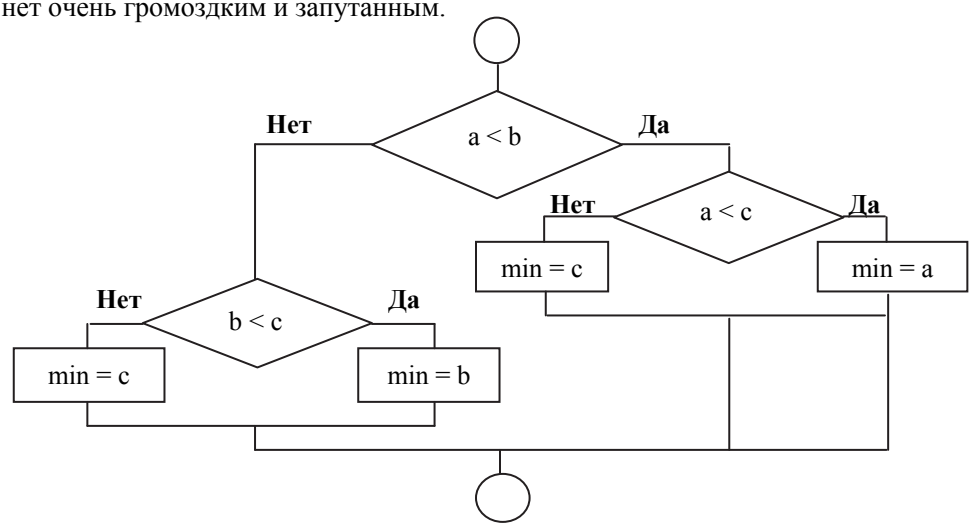


Рис. 33. Алгоритм нахождения минимального из трех чисел (первый способ)

Второй способ, представленный на рис. 34, более простой и наглядный. Сравнив два числа между собой и определив минимальное из них, каждое последующее число будем сравнивать с тем, которое уже находится в ячейке **min,** и осуществлять перезапись ячейки оператором присваивания.

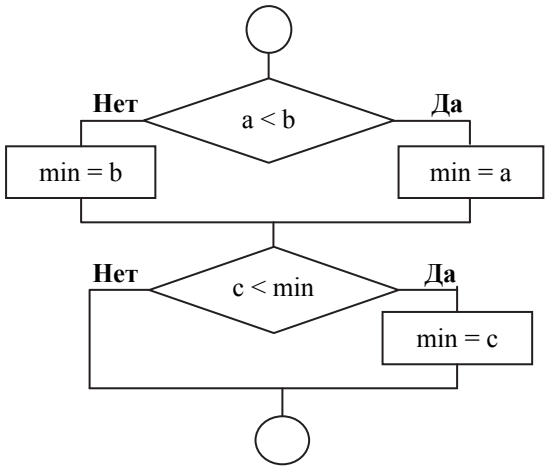


Рис. 34. Алгоритм нахождения минимального из трех чисел (второй способ)

Фрагмент разработки алгоритма решения задачи (третий способ) представлен на рис. 35.

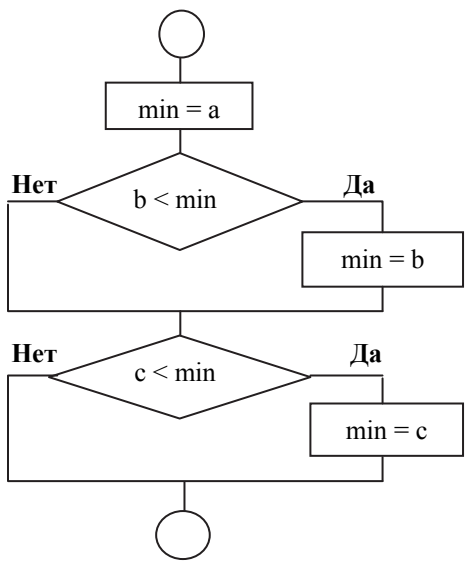


Рис. 35. Алгоритм нахождения минимального из трех чисел (третий способ)

Листинг 12

```
a = int(input("Введите значение a   "))
b = int(input("Введите значение b   "))
```

```
c = int(input("Введите значение c   "))
min=a
if b < min:
   min=b
if c<min:
   min=c
print("Минимальное из трех чисел =", min)
```

На основе рассмотренных алгоритмов решим следующую задачу.

Задача 4. Вычислите значение функции **y**:

$$y = \begin{cases} \min(a1, a2, a3), & \text{если } -1<x<1 \\ \max\{b1, b2, \min\{c1, c2\}\}, & \text{если } x \geq 1 \\ 1, & \text{если } x \leq -1 \end{cases}$$

Разработка алгоритма решения задачи представлена на рис. 36.

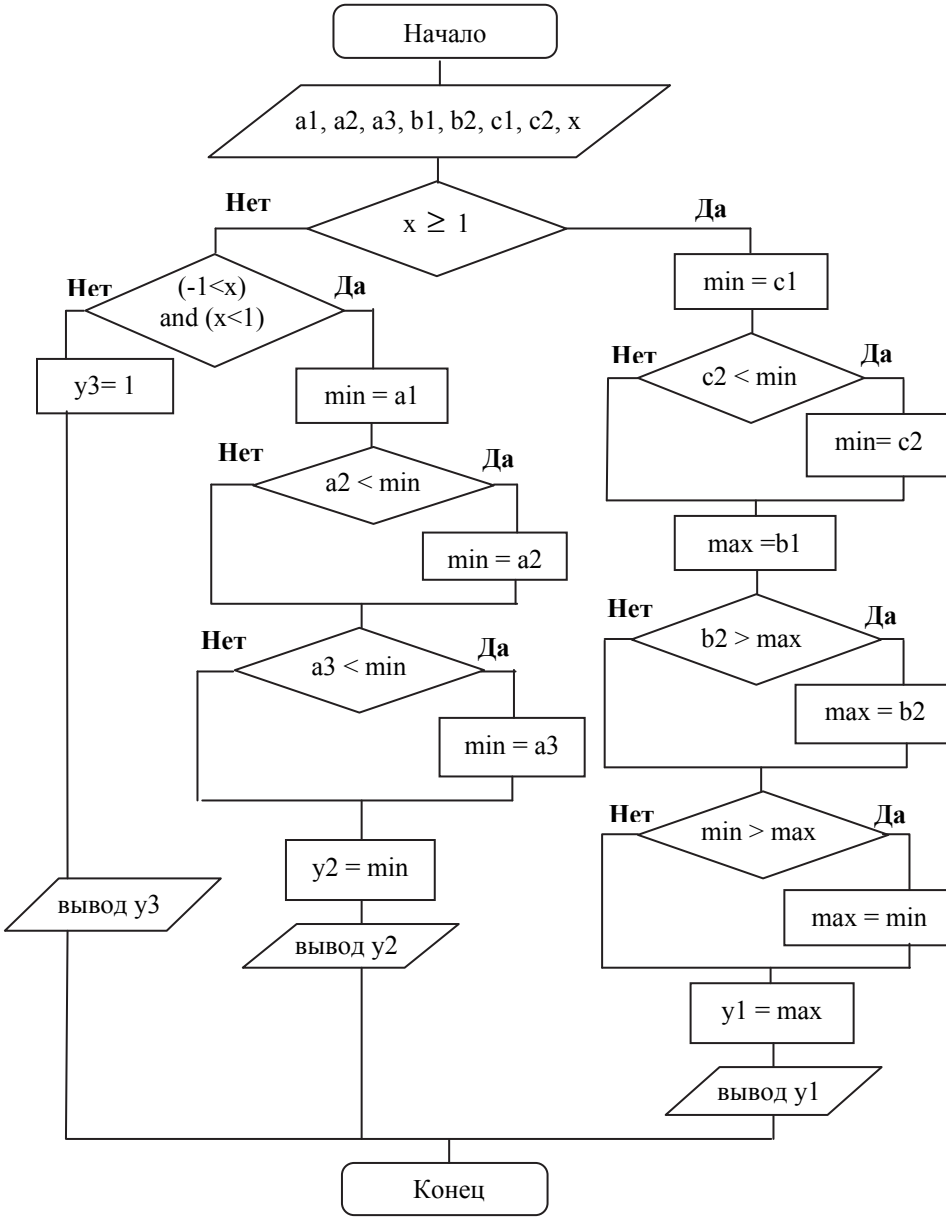


Рис. 36. Алгоритм решения задачи

Ниже приведен код программы, отвечающий за решение задачи. Еще раз **обратите внимание на отступы,** сделанные в листинге 13. Каждый оператор **if** имеет свое вложение операторов согласно алгоритму решения задачи и, соответственно, отступ вправо. Как только условный оператор закончил свою работу, следующий оператор (операторы) пишется (пишутся) без отступа, как, напри-

мер, в следующем фрагменте. После выполнения оператора **max=min** условный оператор **if** закончил свою работу, далее операторы (выделены курсивом) уже не относятся к оператору **if** и при наборе программы смещаются влево.

```
.
.
if min>max:
      max=min
y1=max
print("Тест 2 ветви")
print("y1 =", y1)
.
.
```

Листинг 13

```
a1 = int(input("Введите значение a1   "))
a2 = int(input("Введите значение a2   "))
a3 = int(input("Введите значение a3   "))
b1 = int(input("Введите значение b1   "))
b2 = int(input("Введите значение b2   "))
c1 = int(input("Введите значение c1   "))
c2 = int(input("Введите значение c2   "))
x = float(input("Введите значение x   "))
if x >=1:
   min=c1
   if c2 < min:
      min=c2
   max=b1
   if b2 > max:
      max=b2
   if min>max:
      max=min
   y1=max
   print("Тест 2 ветви")
   print("y1 =", y1)
elif (-1<x) and (x<1):
   min=a1
   if a2 < min:
     min=a2
   if a3<min:
      min=a3
   y2= min
   print("Тест 1 ветви")
   print("y2 =", y2)
else:
   y3 = 1
```

```
print("Тест 3 ветви")
print("y3 =", y3)
```

4.6. Упражнения

Вопрос 1. Каким будет значение переменной s после выполнения группы операторов?

```
n=2.5
f=0.5
d=True
s=0
if n<f:
        s=12
if f>=n:
        s=28
if d:
        s=39
print("s =", s)
```

Ответ. В этом упражнении значение переменной **s** равно **39.** Рассуждать надо следующим образом. Первые два логических выражения ложны, поэтому проверяется третье логическое выражение, в ячейке **d** хранится значение **True** (Истина), поэтому выполняется оператор **s=39.**

Вопрос 2. Каким будет значение переменной j после выполнения группы операторов?

```
w=3
p=5
j=3.5
if (j < p) and (j > w):
        j=j+0.5
        j=j+10
else:
        j=11
print("j =", j)
```

Ответ. Рассуждать нужно следующим образом. Подставив значения переменных **w, p, j** и проверив логическое выражение, получаем, что его значение есть **True** (Истина). Следовательно, выполняется оператор**: j=j+0.5,** а затем оператор **j=j+10**. **Ответ в упражнении**: значение переменной **j=14.0.**

Вопрос 3. Каким будет значение переменной j после выполнения группы операторов?

```
j=3
k=15
m=20
```

```
if j<=k:
   if m>k:
      j=k%2
      j=j%3
   else:
      j=10
print("/nЗначение j= ", j)
```

Ответ. Подставив значения переменных **k, m, j** и проверив оба логических выражения, получаем, что их значения есть **True** (Истина). Следовательно, выполняется оператор: **j=k%2,** а затем: **j=j%3. Ответ в упражнении:** значение переменной **j** равно **1.**

Вопрос 4. Каким будет значение переменной j после выполнения группы операторов?

```
j=6
k=6
if j>k:
   j=j+2
   j=j+3
else:
   j=k-3
   j=j+4
print("j= ", j)
```

Ответ. Подставив значения переменных **k** и **j,** проверив логическое выражение, получаем, что его значение есть **False** (Ложь). Следовательно, выполняются операторы: **j=k-3** и **j=j+4. Ответ в упражнении:** значение переменной **j** равно **7**.

Вопрос 5. Каким будет значение переменной j после выполнения группы операторов?

```
w=3
p=5
j=3.5
if j<p and j>w:
   j=j+0.5
   j=j+12
else:
   j=11
print("j= ", j)
```

Ответ. Подставив значения переменных, получим, что два простых условия **j<p** и **j>w** оказываются истинными, следовательно, и все логическое выражение имеет значение **True.** Таким образом, выполняется оператор: **j=j+0.5,** а затем: **j=j+12.** После их выполнения в ячейке **j** будет число **16.0.**

Вопрос 6. Каким будет значение переменной j после выполнения условного оператора?

```
j=7
k=7
f=10
if j>=k:
   if f<=k:
      k=30%5
      j=(j%2)*k
   else:
      j=1
j=0
```

Ответ. Подставив значения переменных в первое логическое выражение, получим результат: **True** (Истина). Подставив значения переменных во второе логическое выражение, получим, что результат: **False** (Ложь), следовательно, выполняется оператор в ветви **else** (отступ вправо у оператора **j=0** отсутствует, а следовательно, он выполнится уже после выполнения условного оператора). Таким образом, правильный ответ на вопрос, поставленный в упражнении, такой: в ячейке **j** будет находиться значение, равное единице.

Вопрос 7. Каким будет значение переменной f после выполнения группы операторов?

```
x=55
y=5e1
d=False
f=0
if d:
   f=x%2
if x<y:
   f=x
if x>y:
   f=int(2.9)
print("f=", f)
```

Ответ. Проверив первое логическое выражение, получим, что результат **Ложь** – в ячейке **d** хранится значение **False.** Поэтому оператор **f=x%2** не выполняется. Для того чтобы определить, истинно или ложно второе логическое выражение, надо знать, что число **5e1** записано в форме с плавающей точкой и равно **50.** Следовательно, второе логическое выражение ложно, и оператор **f=x** не выполняется. Проверив третье логическое выражение, убеждаемся в том, что оно истинно, так как 55>50. В операторе **f=int(2.9)** используется преобразование к целому типу. Таким образом, после выполнения условного оператора в ячейке **f** будет число **2.**

Вопрос 8. Каким будет значение переменной j после выполнения группы операторов?

```
j=10
k=10
```

```
if j>k:
   j=k-3
else:
   k=k-3
   j=k-3
print("j=", j)
```

Ответ. Проверив логическое выражение, убеждаемся в том, что оно ложно. Следовательно, выполняются операторы в ветви **else.** После выполнения оператора **k=k–3** в ячейке **k** оказывается значение **7,** а после выполнения оператора **j=k–3** в ячейке **j** находится число **4**. **Ответ в упражнении**: **j равно 4.**

4.7. Примеры решения задач

Задача 1. Вычислите значение функции y:

$$y=\begin{cases}\text{Sin}\, x, & \text{если } x \geq 1\\ \text{Cos}\, x, & \text{если } x<1\end{cases}$$

Разработка алгоритма решения задачи представлена на рис. 37.

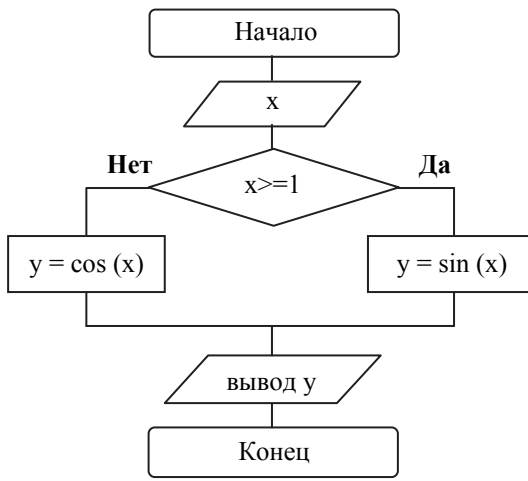


Рис. 37. Алгоритм решения задачи

В листинге 14 приведен код программы, отвечающий за решение задачи.

Листинг 14

```
from math import *
x=float(input("Введите x=  "))
if x>=1:
   y=sin(x)
```

```
else:
   y=cos(x)
print("\nРезультат:  ", y)
```

Задача 2. Вычислите значение функции y:

$$y=\begin{cases}\operatorname{Sin} x, & \text{если } x<0\\ \operatorname{Cos} x, & \text{если } 0\le x\le 1\\ \operatorname{Tg} x, & \text{если } x>1\end{cases}$$

Разработка алгоритма решения задачи представлена на рис. 38.

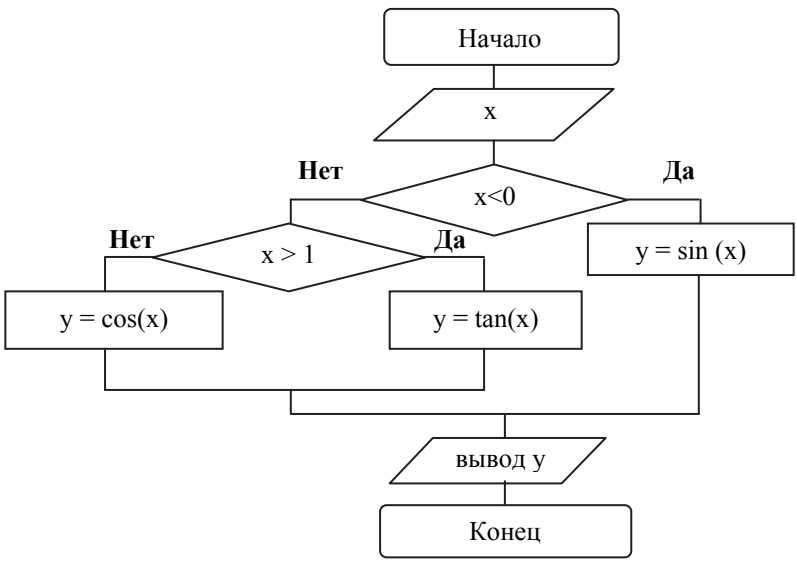


Рис. 38. Алгоритм решения задачи

В листинге 15 приведен код программы, отвечающий за решение задачи.

Листинг 15

```
from math import *
x = float(input("Введите x=  "))
if x<0:
      y=sin(x)
elif x>1:
      y=tan(x)
else:
      y=cos(x)
print("\nРезультат:  ", y)
```

Контрольные вопросы

1. Что называется разветвляющимся алгоритмом?
2. Как записывается простой условный оператор в блок-схемах?
3. Как записывается простой условный оператор в программах?
4. Как работает простой условный оператор?
5. Как записывается сокращенный условный оператор в блок-схемах?
6. Как записывается сокращенный условный оператор в программах?
7. Как работает сокращенный условный оператор?
8. Как записывается составной условный оператор в блок-схемах?
9. Как записывается составной условный оператор в программах?
10. Как работает составной условный оператор?
11. Как записываются многозначные ветвления в блок-схемах?
12. Как записываются многозначные ветвления в программах?
13. Как работает условный оператор if при проверке нескольких условий?

Задачи для самостоятельного решения

1. Разработайте алгоритм и программу вычисления значения y по формуле

$$Y=\begin{cases} X, & \text{если } X \le 0 \\ 2X, & \text{если } X>0 \end{cases}.$$

2. Разработайте алгоритм и программу, которая суммирует только положительные значения, введенные пользователем с клавиатуры.
3. Даны значения трех переменных. Напишите последовательность операторов, подсчитывающих количество значений, которые были равны нулю. Разработайте алгоритм и программу решения данной задачи.
4. Разработайте алгоритм и программу, которая выводит на экран три типа ответа: «Вы имеете удовлетворительную успеваемость», «Вы имеете хорошую успеваемость», «Вы имеете отличную успеваемость», в зависимости от введенного пользователем числа.
5. Разработайте алгоритм и программу решения следующей задачи: найти корни квадратного уравнения по формулам

$$d=b^2-4ac,$$

$$x_1=\frac{-b+\sqrt{d}}{2a},$$

$$x_2=\frac{-b-\sqrt{d}}{2a}.$$

6. Разработайте алгоритм и программу решения следующей задачи. Определить стоимость железнодорожного билета «туда и обратно», если известны расстояние до пункта назначения и длительность пребывания в нем, учитывая, что если расстояние превышает 1000 км, а длительность пребывания превышает 7 дней, то железнодорожная компания дает скидку 30 %.

7. Разработайте алгоритм и программу решения следующей задачи. Даны три числа a, b, c. Проверить, образуют ли они строго возрастающую (a<b<c), строго убывающую (a>b>c) последовательность или не выполняется ни одно из этих двух условий.
8. Разработайте алгоритм и программу решения следующей задачи. Найти минимальное из трех чисел.
9. Разработайте алгоритм и программу решения следующей задачи. Вычислить значение функции y:

$$y=\begin{cases}0, & \text{если } x<0\\ x, & \text{если } 0\le x\le 1.\\ 1, & \text{если } x>1\end{cases}$$

10. Разработайте алгоритм и программу решения следующей задачи. Пользователь вводит два числа. Меньшее из введенных чисел заменяется числом 0, а в случае их равенства – числом 100.

5. ЦИКЛИЧЕСКИЙ АЛГОРИТМ

5.1. Оператор цикла for

Очень часто в программах надо выполнить определенные операторы несколько раз. Нелогично записывать эту последовательность действий двадцать или пятьдесят раз подряд. В этих случаях организуют циклические вычисления. Алгоритм называется **циклическим,** если определенная последовательность шагов выполняется несколько раз в зависимости от заданной величины, которая называется **параметром цикла**. Цикл заканчивается, когда параметр принимает определенное значение. Для организации циклов **с известным количеством** повторений в языке Python используется оператор **for.** Его общий вид в алгоритме представлен на рис. 39.

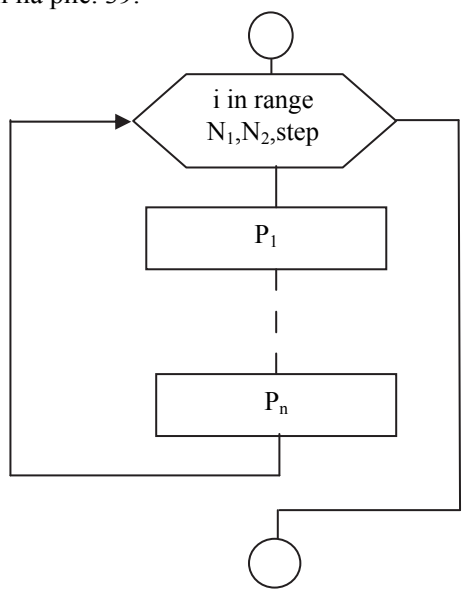


Рис. 39. Общий вид оператора цикла **for**

Отметим, что оператор цикла **for** в языке Python может иметь разные формы записи. Рассмотрим синтаксис первой из них. Назовем ее «цикл по возрастающим значениям параметра». Обратите внимание на **отступы**, которые необходимо делать, если мы хотим, чтобы операторы $P_1,...,P_n$ выполнялись внутри цикла.

for i **in range** (N_1, N_2, **step**)**:**
    P_1
    .
    .
    P_n
    Тело цикла

где **for** (для) – служебное слово; **i** – имя переменной, в которой сохраняются значения элементов; $P_1,...,P_n$ – операторы; **in** – в; **range** – встроенная функция языка Python; **step** – шаг, является необязательным параметром.

Отметим, что аргументами функции **range** могут быть только целые числа. Работа оператора цикла **for** в такой конструкции заключается в следующем. При первом вхождении в цикл параметр цикла **i** принимает значение, равное величине нижней границы $\mathbf{N_1}$, и выполняется оператор или операторы в теле цикла. Затем значение параметра увеличивается на **величину step,** и вновь выполняется тело цикла. Подобные действия будут повторяться до тех пор, пока значение параметра цикла не станет равным величине $\mathbf{N_2-1}$, после чего осуществляется выход из цикла. Если аргумент **step** пропущен в функции **range,** то шаг изменения параметра цикла будет равен единице.

Задача. Найдите сумму целых чисел от 1 до 50. Разработка алгоритма решения задачи представлена на рис. 40.

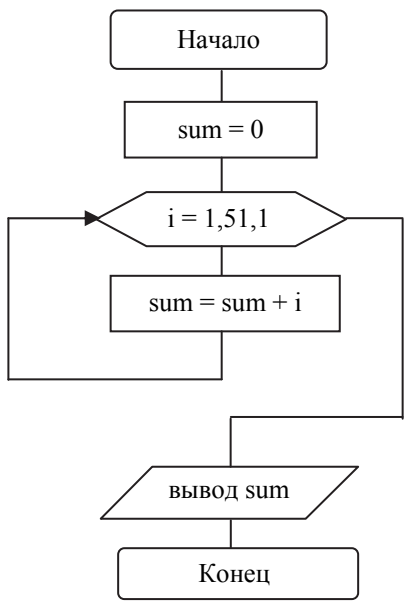


Рис. 40. Алгоритм решения задачи

При первом вхождении цикл параметр цикла примет значение, равное единице, и выполнится оператор **sum=sum+i.** Затем параметр **i** будет последовательно увеличиваться на величину шага, который равен единице, и всякий раз в цикле будет выполняться оператор **sum=sum+i.** Следует отметить, что конечное значение верхней границы 51 достигнуто не будет. **Счетчик,** как иногда называют параметр цикла, остановится на предыдущем значении, т. е. значении, равном 50. В листинге 16 приведен код программы, отвечающий за решение задачи.

Листинг 16

```
sum=0
for i in range (1, 51, 1):
   sum=sum+i
print("Сумма = ", sum)
```

Если учесть, что параметр **step** является необязательным, то заголовок цикла можно переписать иначе (листинг 17).

Листинг 17

```
sum=0
for i in range (1, 51):
   sum=sum+i
print("Сумма = ", sum)
```

Попробуем написать ту же программу, но с **циклом по убывающему значению параметра**. Будем считать ее **второй формой** записи оператора цикла **for.** Последний аргумент в функции **range** теперь равен минус единице. Соответственно, параметр цикла **i** будет изменяться от 50 до 1, значение 0 в перебор включено не будет.

Листинг 18

```
sum=0
for i in range (50, 0, -1):
   sum=sum+i
print("Сумма = ", sum)
```

Третья форма записи оператора цикла **for** может быть организована с использованием функции **range** и одним параметром, указывающим значение верхней границы. Таким образом, параметр цикла будет меняться от 0 до значения N_2-1. Синтаксис оператора в общем виде можно записать следующим образом.

```
for i in range (N2):
        P1
        .
        .        Тело цикла
        Pn
```

Программу «Найти сумму целых чисел от 1 до 50» теперь можно написать, как следующую последовательность операторов.

Листинг 19

```
sum=0
for i in range (51):
   sum=sum+i
print("Сумма = ", sum)
```

Отметим, что результатом выполнения всех рассмотренных программ является число 1275.

Более широкие возможности применения оператора цикла **for** мы рассмотрим в дальнейшем, а сейчас приведем еще один пример, связанный с обработкой последовательности символов, тем более сейчас, когда нам известны основы работы оператора **for,** он не покажется сложным.

Листинг 20

```
slovo=input("Введите слово ")
for i in slovo:
    print(i, end=" ")
```

Как видно из листинга 20, в цикле происходит перебор символов того слова (строки), которое вводит пользователь с клавиатуры. Результат работы программы представлен на рис. 41. В операторе **print,** в отличие от выше рассмотренных примеров, используется небольшое дополнение, а именно, оператор **end=" ",** что позволило расположить результаты работы программы в строке, а не в столбик, как было сделано ранее.

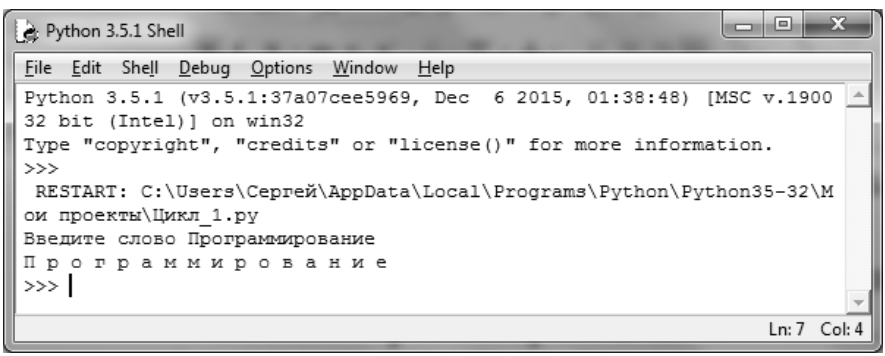


Рис. 41. Результат работы программы

Сложный циклический процесс. Вложенные циклы

Если телом цикла является циклическая структура, то такие циклы называются **вложенными.** Цикл, содержащий в себе другой цикл, называют **внешним,** а цикл, содержащийся в теле другого цикла, называют **внутренним.** Общий вид в алгоритме сложного циклического процесса представлен на рис. 42.

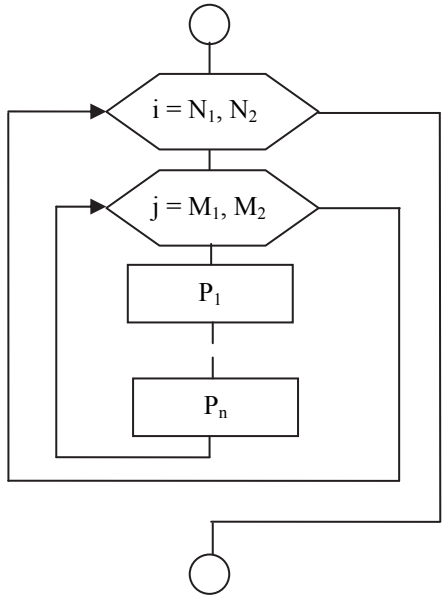


Рис. 42. Общий вид в алгоритме сложного циклического процесса

Синтаксис операторов сложного цикла представлен ниже. Обратите внимание на **отступы,** выполненные для внутреннего цикла и операторов P_1, ..., P_n, которые в нем будут выполняться.

```
for i in range(N1, N2):  #внешний цикл
   for j in range(M1, M2):  #внутренний цикл
      P1
      .
      .                    тело цикла
      Pn
```

Рассмотрим работу сложного циклического процесса. При первом вхождении в цикл параметр внешнего цикла **i** принимает значение равное $\mathbf{N_1}$**.** Управление передается во внутренний цикл, в котором параметр цикла **j** принимает значение, равное $\mathbf{M_1}$**,** и выполняется оператор (операторы), который записан во внутреннем цикле. Затем параметр внутреннего цикла **j** увеличивается на **единицу** (если опущен аргумент **step** в функции **range**), и вновь выполняется тело цикла. Затем параметр внешнего цикла **i** увеличивается на **единицу,** и вновь начинает свою работу внутренний цикл, в котором параметр цикла **j** будет изменяться от $\mathbf{M_1}$ до $\mathbf{M_2}$**,** и при каждом прохождении цикла будут выполняться операторы $\mathbf{P_1}$, …, $\mathbf{P_n}$**.**

Задача. Разработайте приложение, которое осуществляет вывод на экран таблицы умножения для значений от 1 до 5.

Листинг 21

```
for i in range(1,6):
        for j in range(1,6):
            print(i, '*', j, '=', i*j, end='\t')
    print()
```

Итак, согласно алгоритму работы сложного циклического процесса установим границы изменения параметра во внутреннем и внешнем циклах от 1 до 6. Во внутреннем цикле выполнится оператор **print(i, '*', j, '=', i*j, end='\t'),** в котором сочетается вывод самих значений параметров, строковых выражений для вывода знаков умножения и равенства на экран и, собственно, самого действия **i*j,** которое обеспечит перемножение параметров. «Пустой» оператор **print**() (**обратите внимание на отступ**) не выполняется во внутреннем цикле, а служит для пропуска строки между выводом столбцов ответа. Результат работы программы представлен на рис. 43.

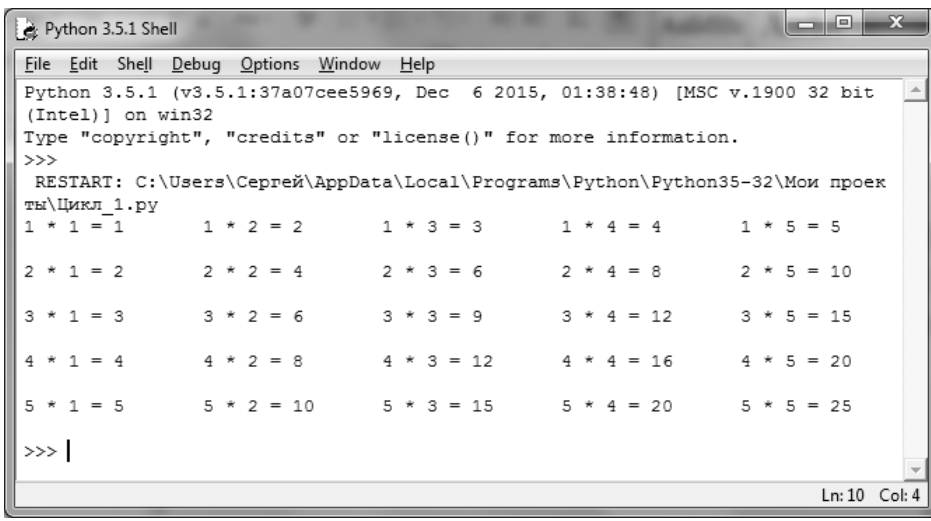


Рис. 43. Вывод таблицы умножения

Упражнения

Вопрос 1. Сколько раз будет выполнен оператор d=5 в теле цикла?

```
d=4
r=15
for i in range(d+1,r,1):
    d=5
```

Ответ. Подставив значения переменных **d** и **r,** получаем, что параметр цикла **i** меняется от 5 до 14. При первом вхождении в цикл параметр цикла примет значение 5, а далее будет автоматически увеличиваться на +1, пока не достигнет величины верхней границы 14. Следовательно, оператор выполнится **10** раз.

Вопрос 2. Определите, какое значение будет в ячейке r после выполнения группы операторов?

```
r=50
s=0
for i in range(5,0,-1):
    s=1
    r=r-s
print("r = ",r)
```

Ответ. При первом вхождении в цикл параметр цикла примет значение 5. В программе используется цикл по убывающим значениям, следовательно, параметр цикла при каждом выполнении тела цикла будет уменьшаться на **-**1. При первом вхождении в цикл переменная **s** принимает значение, равное 1. Выполняется оператор **r=r–s.** После его выполнения в ячейке **r** будет значение 49. Затем параметр цикла уменьшается на единицу и будет равен четырем. Вновь выполняются операторы **s=1** и **r= r–s.** Выполняя их каждый раз и подсчитывая полученные значения, получим в ячейке **r** значение **45.**

Вопрос 3. Определите, какое значение находится в ячейке y после выполнения группы операторов?

```
a=7
d=5
y=0
for i in range(1,4,1):
    y=d
    y=a+2
print ("y = ", y)
```

Ответ. В цикле с оператором **for** параметр цикла будет меняться от 1 до 3. В цикле выполняются два оператора: **y=d** и **y=a+2.** Операторы выполняются три раза. Каждый раз при прохождении цикла в ячейке **y** будет находиться значение 5 (**y=d**), а после выполнения оператора **y=a+2** значение ячейки **y** будет равно 9. Следовательно, после выполнения группы операторов в ячейке **y** будет находиться значение **9.**

Вопрос 4. Определите, какое значение находится в ячейке y после выполнения группы операторов, и какие значения будет принимать ячейка i в теле цикла?

```
a=17.0
d=a
for i in range(3):
    print("i = ",i)
    if a!=d:
        a=a+1
    else:
        y=a
print("y = ",y)
```

Ответ. В цикле с оператором **for** происходит проверка логического выражения **a!=d.** Поскольку значения **a** и **d** равны после выполнения операторов **a=17.0** и **d=a,** логическое выражение все время будет оставаться ложным, а следовательно, выполнение оператора **y=a** в ветви **else** гарантировано при каждом прохождении цикла. Таким образом, с помощью оператора **print** на экран будет выведено значение 17.0. Параметр цикла **i**, согласно теории, рассмотренной выше, будет принимать последовательные значения от 0 до 2.

Вопрос 5. Определите, какое значение находится в ячейке y после выполнения группы операторов?

```
a=1
y=1
for i in range(2,6,1):
   a = a+10
   y= a+10
print("y = ", y)
```

Ответ. При первом вхождении в цикл параметр цикла примет значение нижней границы, равное двум. В цикле будут выполняться два оператора, таким образом, в ячейке **a** оказывается значение 11, а после выполнения оператора **y=a+10** значение, находящееся в ячейке **y**, станет равным **21.** Параметр увеличивается на единицу и будет равен трем, выполняется оператор **a=a+10,** и в ячейке **a** оказывается число 21, а в ячейке **y** находится число 31. Выполнив подобные действия еще два раза, получим в ячейке **y** значение, равное **51.**

Вопрос 6. Определите, какое значение находится в ячейке y после выполнения группы операторов?

```
a=5
d=5
y=0
for i in range(7,1,-1):
   a=d
   y=a+10
print(" y = ",y)
```

Ответ. При первом вхождении в цикл параметр цикла примет значение, равное 7. В ячейке **a** оказывается значение, равное 5, а в ячейке **y** после выполнения оператора **y=a+10** будет значение, равное 15. В цикле по убывающим значениям параметра значение ячейки **i** уменьшится на единицу и будет равно 6. Выполняется оператор **a=d,** и в ячейке **a** снова оказывается число 5. В ячейке **y** после выполнения оператора **y=a+10** будет значение 15. Эти действия будут повторяться до тех пор, пока параметр цикла не достигнет значения границы, равного 2. Происходит выход из цикла. В ячейке **y** находится значение, равное **15.**

Вопрос 7. Определите, какое значение находится в ячейке y после выполнения цикла с оператором for?

```
a=5
d=5
y=0
for i in range(2,6,1):
   y=d+10
y=a+10
y=a*d
print("y = ", y)
```

Ответ. В данном фрагменте программы при выполнении цикла смещен вправо только один оператор **y=d+10.** Следовательно, он и будет выполняться. При первом прохождении цикла в ячейке **y** оказывается значение, равное 15. Параметр цикла увеличивается на единицу. Вновь выполняется оператор **y=d+10,** и так как значение ячейки **d** не изменялось, оно вновь оказывается равным 15. Эти действия будут повторяться до тех пор, пока параметр не достигнет значения верхней границы, равной 5. Происходит выход из цикла, и в ячейке **y** находится значение, равное **15.** Поставив вопрос к данному упражнению по-другому: «Какое значение находится в ячейке **y** после выполнения фрагмента программы?», – следует проанализировать еще два оператора, где изменяется значение **y,** и получить значение, равное **25.**

Вопрос 8. Сколько раз будет выполнен цикл в следующем фрагменте программы?

```
n=2
s=13
y=0
for i in range (s+2,n+3,-1):
   y=y+1
```

Ответ. Подставив исходные значения переменных в заголовок цикла, видим, что параметр цикла **i** в цикле по убывающим значениям будет меняться от 15 до 6, изменяясь при этом с шагом **-1**. При первом вхождении в цикл параметр цикла примет значение, равное 15, затем он будет равен 14, 13, 12 и т. д., и цикл будет выполняться до тех пор, пока параметр не станет равным шести. Таким образом, цикл выполнится **10** раз.

Примеры решения задач

Задача 1. Последовательно вводятся шесть целых чисел. Определите, каких среди них больше: положительных или отрицательных. Разработка алгоритма решения задачи представлена на рис. 44.

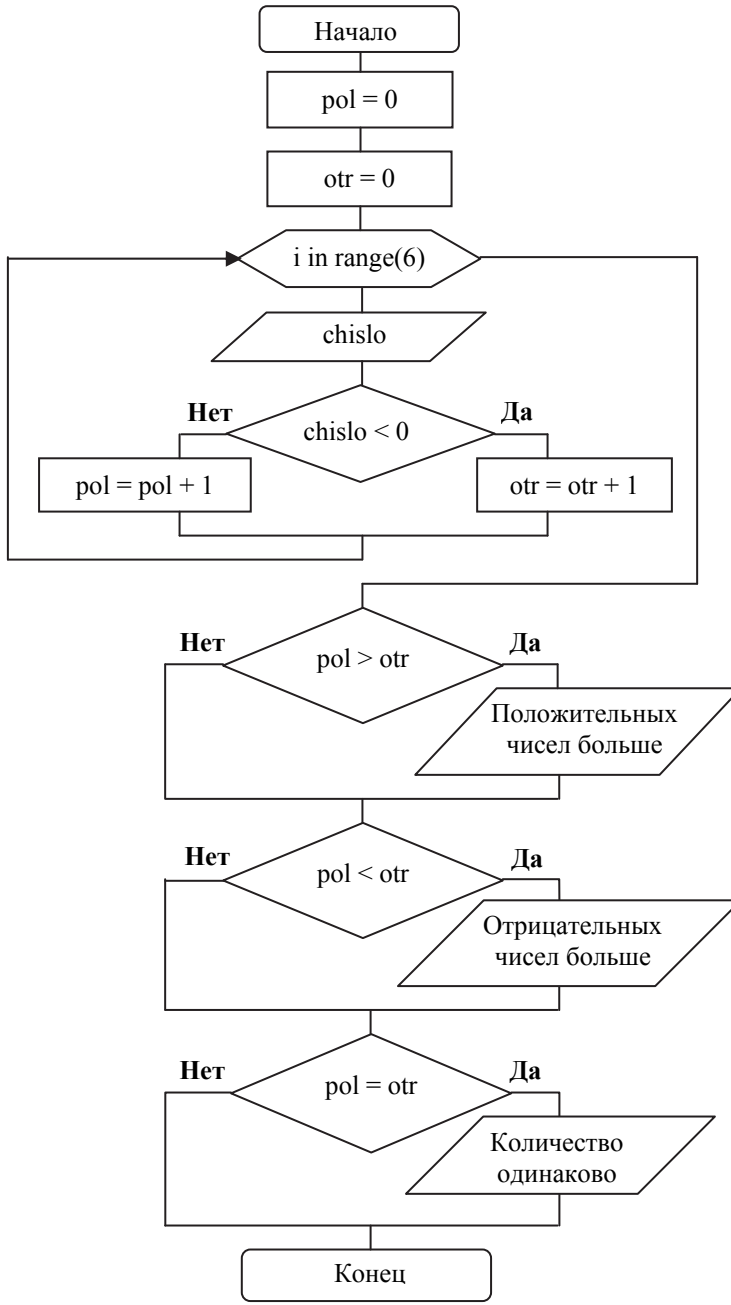


Рис. 44. Алгоритм решения задачи

Комментарий. Ячейки **pol** и **otr** играют роль счетчиков. Счетчики в цикле будут увеличиваться на единицу операторами **otr=otr+1** и **pol=pol+1,** поэтому для того, чтобы конечный результат не был искажен, ячейки предварительно обнуляются операторами **pol=0** и **otr=0.**

В листинге 22 приведен код программы, отвечающий за решение задачи.

Листинг 22

```
pol=0 #Счетчик положительных чисел предварительно обнуляется
otr=0 #Счетчик отрицательных чисел предварительно обнуляется
for i in range (6):
   chislo = int(input("Введите число   "))
   if chislo<0:
      otr=otr+1 #Счетчик отрицательных чисел увеличивается на единицу
   else:
      pol=pol+1 #Счетчик положительных чисел увеличивается на едини-
ицу
if pol>otr:
   print("Положительных чисел больше")
if pol<otr:
   print("Отрицательных чисел больше")
if pol==otr:
   print("Количество чисел одинаково")
```

Задача 2. Последовательно вводится пять вещественных чисел. Найдите минимальное из положительных чисел. Разработка алгоритма решения задачи представлена на рис. 45.

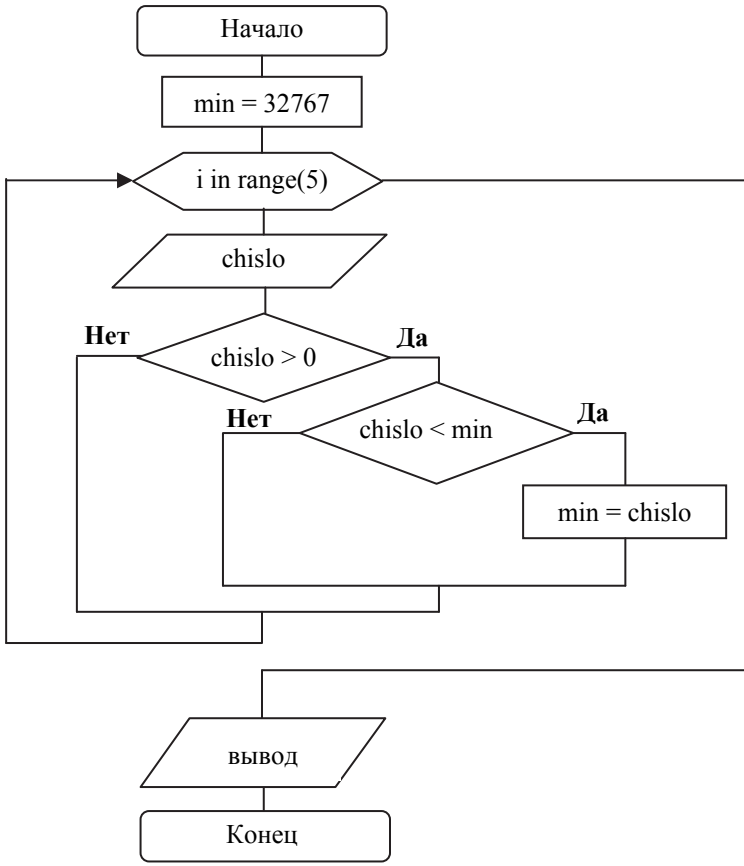


Рис. 45. Алгоритм решения задачи

Комментарий. Для реализации алгоритма поиска минимального числа мы отводим ячейку **min,** в которую предварительно заносим любое большое значение, например, в данной программе **+32767,** оператором **min=32767.** В цикле введенное пользователем число сравнивается с хранящимся в ячейке **min,** и если оно меньше, то введенное число заносится в ячейку **min** оператором **min=chislo.** Таким образом, к моменту завершения цикла в ячейке **min** будет находиться минимальный элемент.

При решении подобных задач следует учесть, что для их более корректной работы можно дополнить код программ методами обработки исключений, изложенными выше, либо сделать дополнительные проверки условий (в данной задаче можно было бы предусмотреть тот случай, когда пользователь вводит все отрицательные числа).

В листинге 23 приведен код программы, отвечающий за решение задачи.

Листинг 23

```
min = 32767
for i in range (5):
   chislo = float(input("Введите число   "))
   if chislo>0: #Проверка на положительность очередного введенного чис-
ла
      if chislo<min:    #Поиск минимального элемента
         min=chislo
print("Минимальное число =" , min)
```

Задача 3. Последовательно вводятся N целых чисел. Найдите максимальное из отрицательных значений. Разработка алгоритма решения задачи представлена на рис. 46.

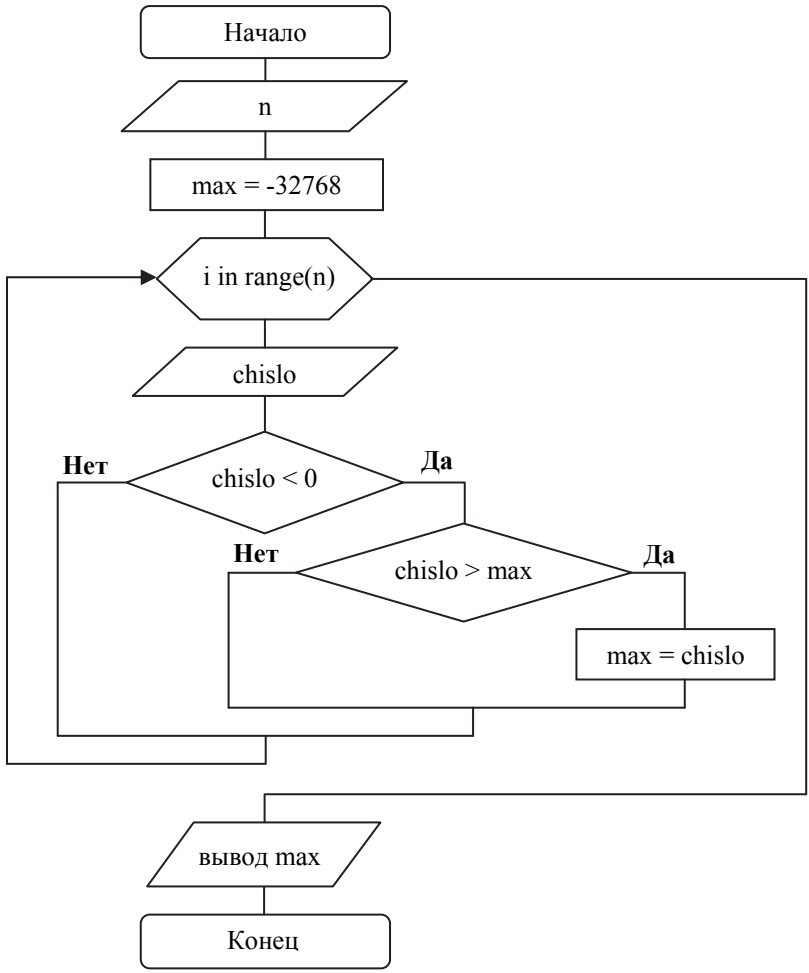


Рис. 46. Алгоритм решения задачи

Комментарий. Алгоритм поиска максимального элемента обратный алгоритму поиска минимального числа. Для реализации алгоритма поиска максимального числа отводим ячейку **max,** в которую предварительно заносим любое малое значение, например, в данной программе отрицательное число **–32768,** оператором **max=-32768.** В цикле введенное пользователем число сравнивается с хранящимся в ячейке **max,** и если оно больше, то введенное число заносится в ячейку **max** оператором **max=chislo.** Таким образом, к моменту завершения цикла в ячейке **max** будет находиться максимальный элемент.

В листинге 24 приведен код программы, отвечающий за решение задачи.

Листинг 24

```
max=-32768
#Ввод количества чисел, которые будут обрабатываться в цикле
n=int(input("Введите количество чисел  "))
for i in range (n):
        chislo=int(input("Введите число   "))
        if chislo<0: #Проверка на отрицательность очередного введенного числа
                if chislo>max: #Поиск максимального элемента
                   max=chislo
print("Максимальное из отрицательных чисел =" , max)
```

Задача 4. Последовательно вводятся пять вещественных чисел. Найдите количество отрицательных чисел и минимальное из отрицательных. Разработка алгоритма решения задачи представлена на рис. 47.

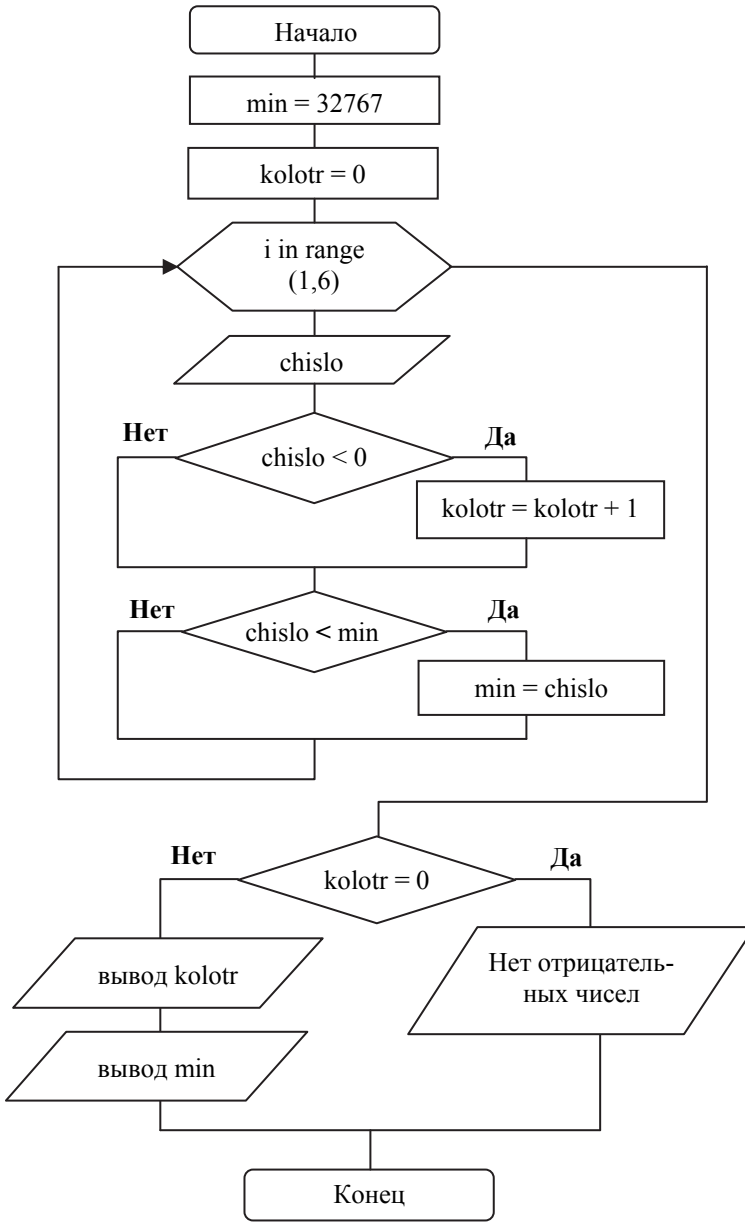


Рис. 47. Алгоритм решения задачи

В листинге 25 приведен код программы, отвечающий за решение задачи.

Листинг 25

```
min= 32767
kolotr=0
for i in range(1,6):
   chislo=float(input("Введите число "))
   if chislo<0:
      kolotr=kolotr+1
   if chislo<min:
      min=chislo
if kolotr==0:
   print("Отрицательных чисел нет")
else:
   print("Количество отрицательных чисел: ",kolotr)
   print("Минимальное из отрицательных чисел: ",min)
```

Задача 5. Последовательно вводятся N вещественных чисел. Найдите количество положительных чисел и максимальное из них. Разработка алгоритма решения задачи представлена на рис. 48.

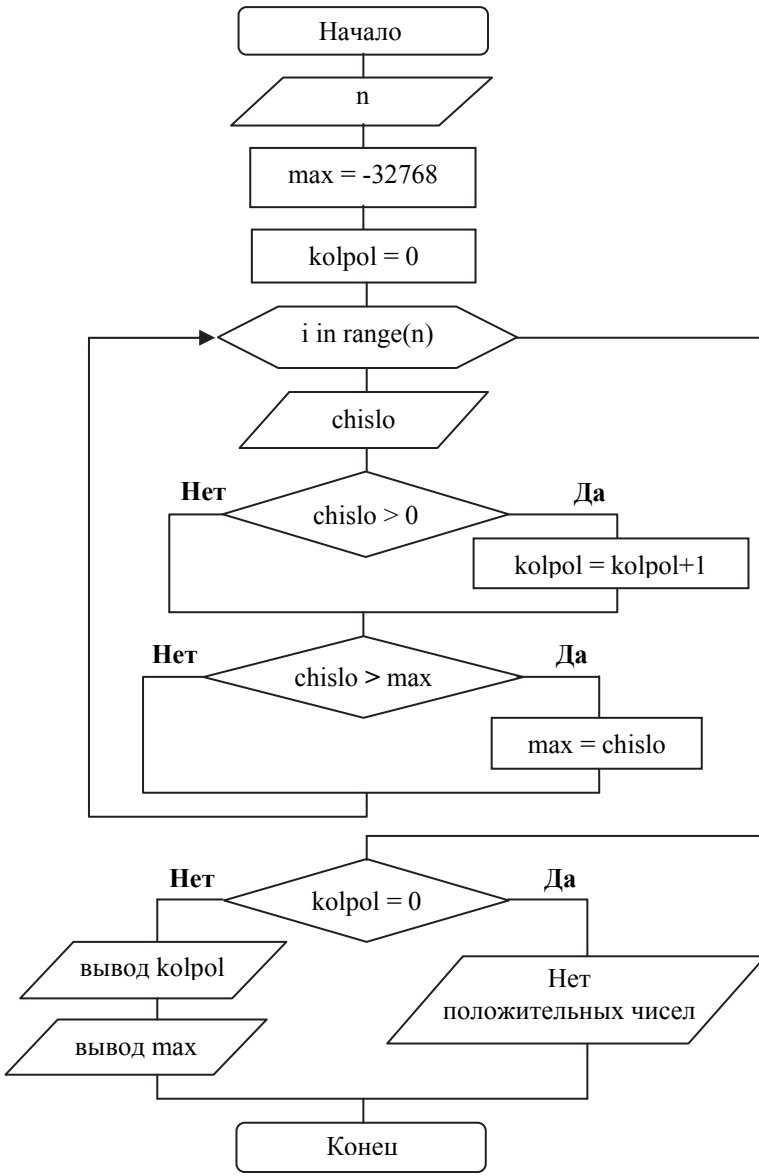


Рис. 48. Алгоритм решения задачи

В листинге 26 приведен код программы, отвечающий за решение задачи.

Листинг 26

```
n=int(input("Введите количество чисел"))
max=-32767
kolpol=0
for i in range(n):
   chislo=float(input("Введите число"))
   if chislo>0:
      kolpol=kolpol+1
   if chislo>max:
      max=chislo
if kolpol==0:
   print("Нет положительных чисел")
else:
   print("Количество положительных чисел =", kolpol)
   print("Максимальное из них =", max)
```

Задача 6. Последовательно вводятся пять целых чисел. Найдите количество положительных чисел и их среднее арифметическое. Разработка алгоритма решения задачи представлена на рис. 49.

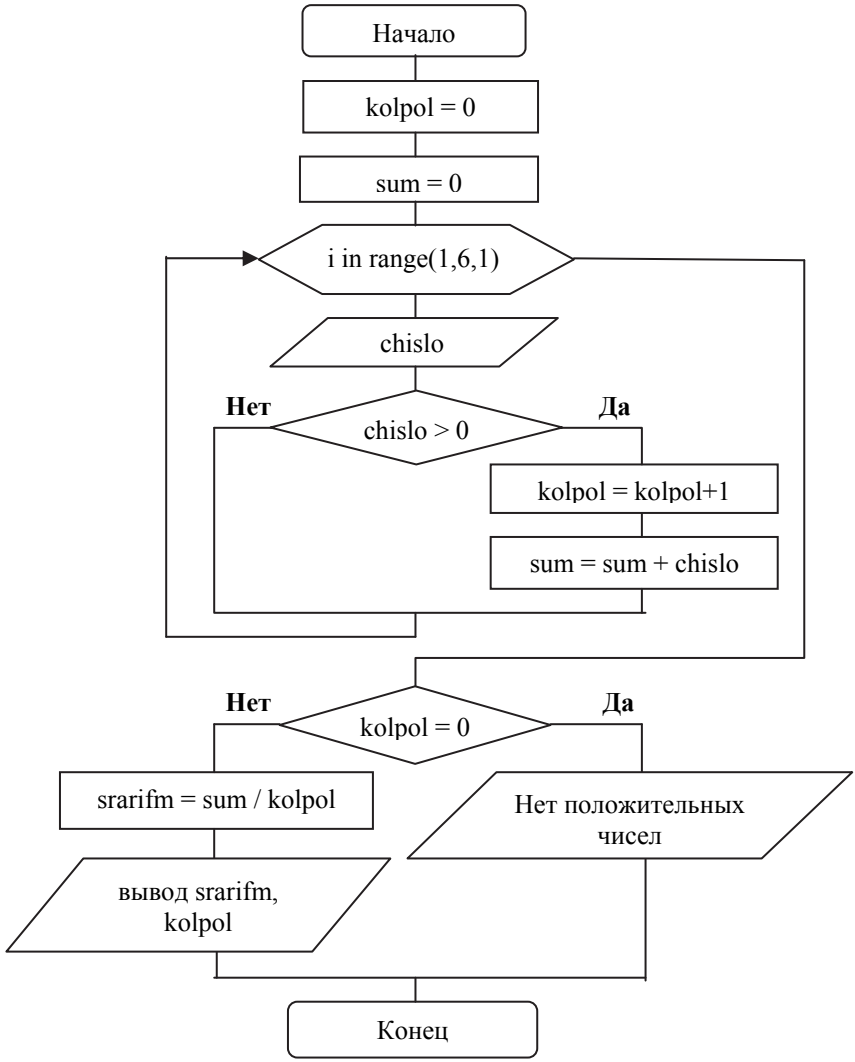


Рис. 49. Алгоритм решения задачи

Комментарий. Поиск среднего арифметического положительных значений заключается в нахождении суммы чисел (**ячейка sum**), нахождении их количества (**ячейка kolpol**) и вычислении среднего арифметического по формуле **srarifm=sum/kolpol.** В листинге 27 приведен код программы, отвечающий за решение задачи.

Листинг 27

```
kolpol = 0
sum = 0
for i in range(1,6,1):
    chislo = int(input("\nВведите число  "))
    if chislo>0:
        kolpol=kolpol+1
        sum=sum+chislo
if kolpol==0:
    print("Нет положительных чисел")
else:
    srarifm = sum/kolpol
    print("Среднее арифметическое положительных чисел =", srarifm)
    print("Количество положительных чисел =", kolpol)
```

Задача 7. Вводится последовательность из N целых чисел. Найдите, сколько в ней чисел, равных числу **100**, а также количество отрицательных чисел. Разработка алгоритма решения задачи представлена на рис. 50.

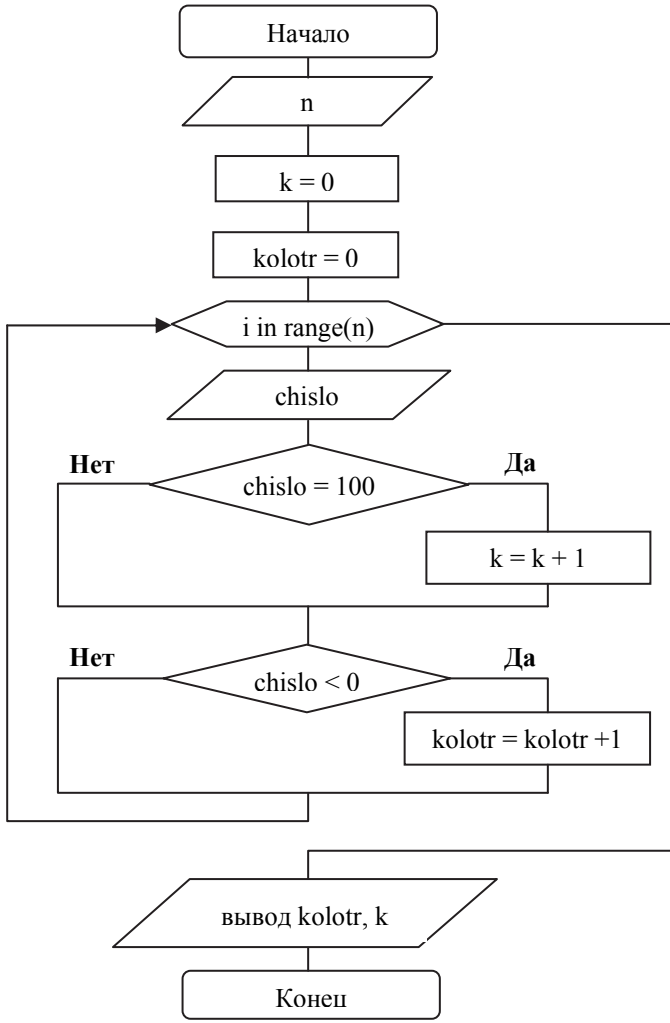


Рис. 50. Алгоритм решения задачи

В листинге 28 приведен код программы, отвечающий за решение задачи.

Листинг 28

```
n=int(input("Введите количество чисел"))
kolotr=0
k=0
for i in range (n):
    chislo=int(input("Введите число: "))
```

```
    if chislo==100:
        k=k+1
    if chislo<0:
        kolotr=kolotr+1
print("\nКоличество чисел равных 100: ",k, "\nКоличество отрицательных чисел: ", kolotr)
```

Задача 8. Последовательно вводится десять вещественных чисел. Определите, сколько из них совпадают с первым числом. Разработка алгоритма решения задачи представлена на рис. 51.

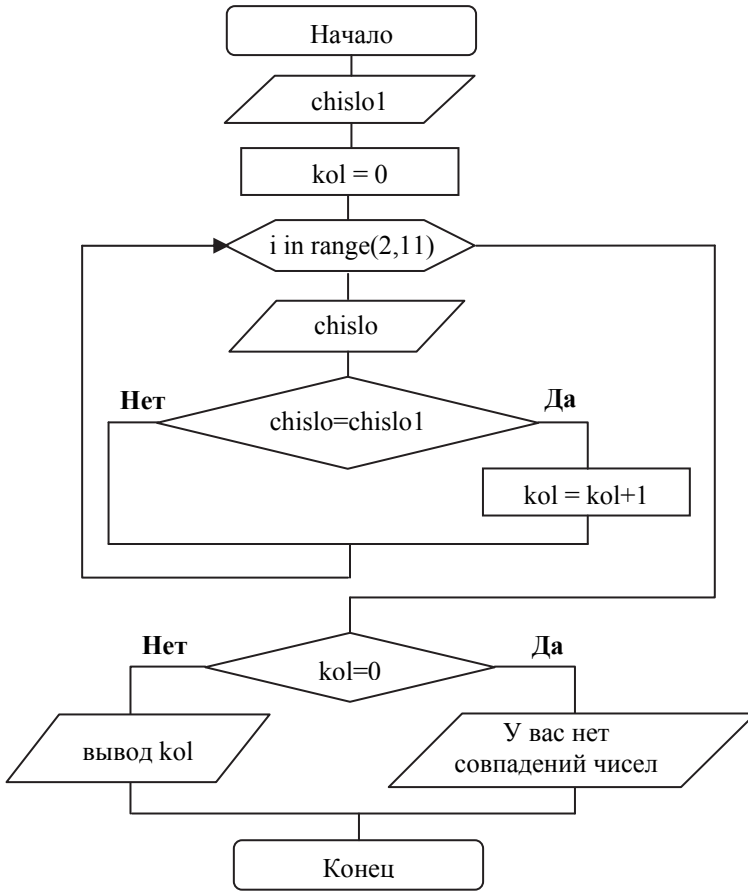


Рис. 51. Алгоритм решения задачи

Комментарий. В ячейку **chislo1** заносим первое число. Все последующие введенные числа, начиная со второго, будем сравнивать в цикле с тем числом, которое находится в ячейке **chislo1.** В случае совпадения будем увеличивать значение ячейки **kol,** которая играет роль счетчика, на единицу. В листинге 29 приведен код программы, отвечающий за решение задачи.

Листинг 29

```
chislo1=float(input("Введите первое число: "))
kol=0
```

```
for i in range (2,11):
    chislo=float(input("Введите следующее число: "))
    if chislo==chislo1:
        kol=kol+1
if kol==0:
    print("У вас нет совпадений чисел ")
else:
    print("С первым числом совпало ", kol)
```

Задача 9. Последовательно вводится десять целых чисел. Найдите разность между максимальным и минимальным из них. Разработка алгоритма решения задачи представлена на рис. 52.

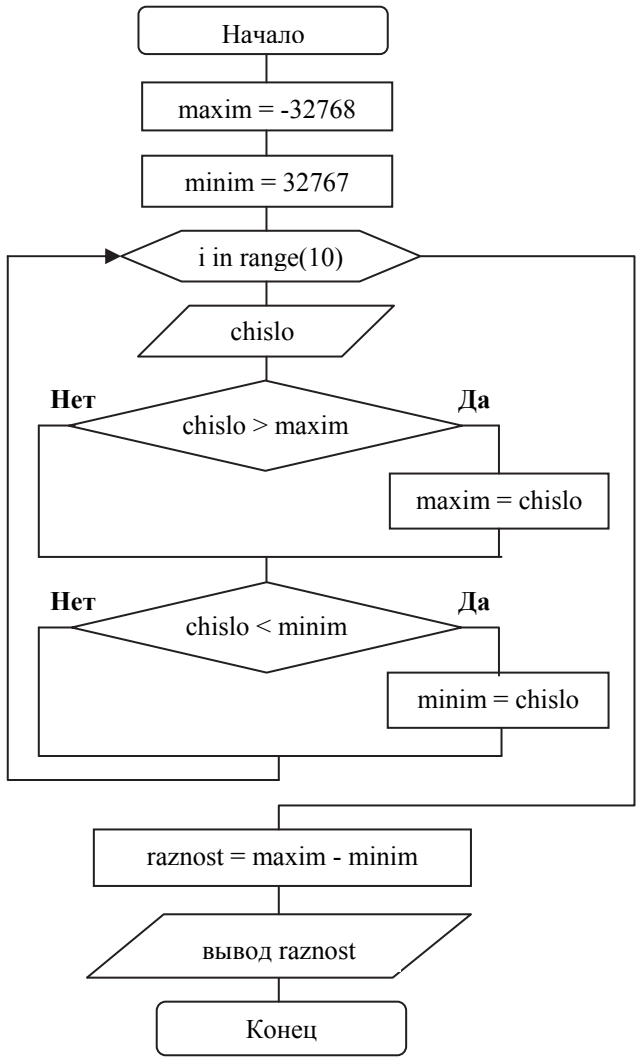


Рис. 52. Алгоритм решения задачи

Комментарий. В задаче реализуются алгоритмы поиска минимального и максимального элемента, описанные выше. По окончании цикла в ячейке **maxim** будет находиться максимальный элемент, а в ячейке **minim** – минимальный элемент. Оператором **raznost=maxim–minim** находится разность чисел. В листинге 30 приведен код программы, отвечающий за решение задачи.

Листинг 30

```
maxim=-32768
minim=32767
for i in range(10):
    chislo=int(input("Введите число "))
    if chislo>maxim:
        maxim=chislo
    if chislo<minim:
        minim=chislo
raznost=maxim-minim
print("Разность между максимальным и минимальным числом =  ", raz-
nost)
```

Задача 10. Последовательно вводится десять целых чисел. Найдите произведение всех положительных и всех отрицательных чисел, предварительно вычислив их сумму. Разработка алгоритма решения задачи представлена на рис. 53.

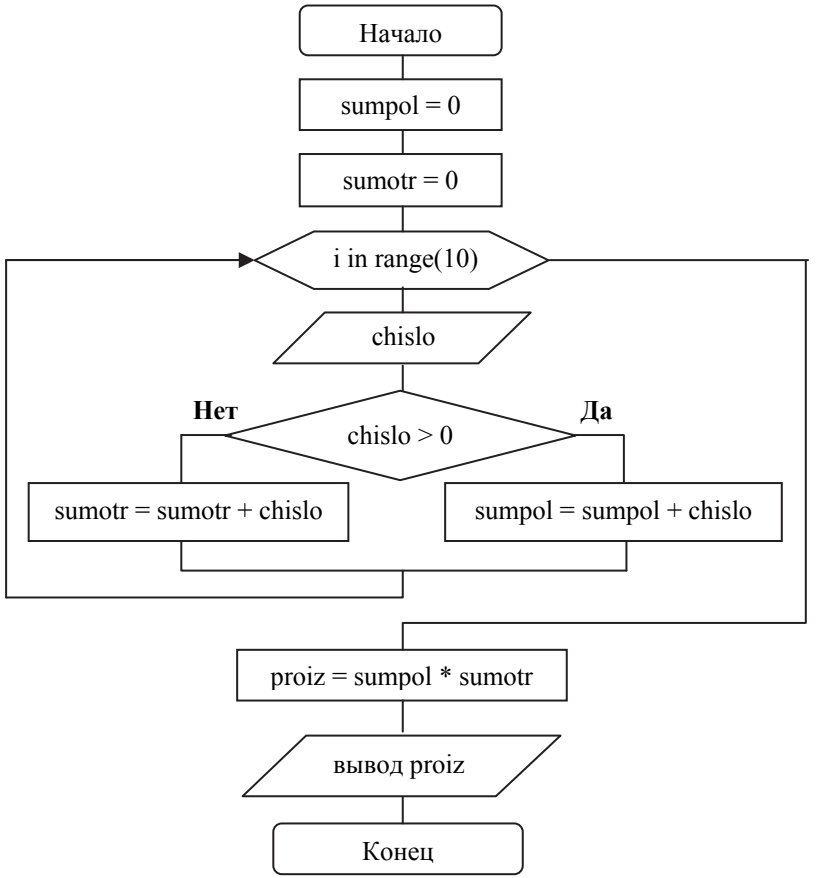


Рис. 53. Алгоритм решения задачи

В листинге 31 приведен код программы, отвечающий за решение задачи.

Листинг 31

```
sumotr=0
sumpol=0
for i in range(10):
    chislo=int(input("Введите число "))
    if chislo>0:
        sumpol=sumpol+chislo
    else:
        sumotr=sumotr+chislo
proiz=sumotr*sumpol
print("Произведение положительных и отрицательных чисел = ",proiz)
```

Контрольные вопросы

1. Дайте определение циклического алгоритма.
2. Расскажите о работе оператора цикла for по возрастающим значениям параметра, нарисовав общий вид алгоритма и синтаксис этого оператора.
3. Расскажите о работе цикла с оператором for по убывающим значениям параметра цикла.
4. Расскажите о работе сложного циклического процесса, нарисовав общий вид алгоритма и синтаксис этого оператора.
5. Какой цикл называется внешним, а какой – внутренним?

Задачи для самостоятельного решения

1. Разработайте алгоритм и программу решения следующей задачи. Имеется N значений температур. Найдите количество отрицательных температур.
2. Разработайте алгоритм и программу решения следующей задачи. Имеется N значений температур. Найдите среднюю температуру.
3. Разработайте алгоритм и программу решения следующей задачи. Имеется N значений температур. Определите среднее значение отрицательных температур.
4. Разработайте алгоритм и программу решения следующей задачи. Имеется N значений температур. Определите максимальную температуру.
5. Разработайте алгоритм и программу решения следующей задачи. Имеется N значений температур. Найдите максимальное и минимальное значения температуры.
6. Разработайте алгоритм и программу решения следующей задачи. Некоторая сумма денег помещена в банк под процент. Определите величину вклада в конце каждого года.

5.2. Оператор цикла while

Для выполнения оператора **for** необходимо задать параметры, которые будут определять, сколько раз должен выполниться оператор (операторы) цикла. Альтернативой циклу с оператором **for** является цикл с неизвестным количеством повторений, в котором оператор (операторы) выполняется, пока логическое выражение не примет определенное значение.

Циклическая структура, в которой число повторений цикла заранее неизвестно, а определяется только в процессе выполнения алгоритма, называется **итеративной.**

Такие циклы нужно применять в тех задачах, где мы не можем знать точно, сколько раз будет повторен цикл. Например, пользователь должен вводить пароль, для того чтобы начать работу с какой-нибудь программой. Какое количество попыток он использует? Неизвестно.

Следовательно, для реализации подобных задач необходимо владеть соответствующими методами их решения. В языке Python для реализации конструкции

цикла с неизвестным количеством повторений служит оператор **while** или **цикл с предпроверкой условия.** Его общий вид в алгоритме представлен на рис. 54.

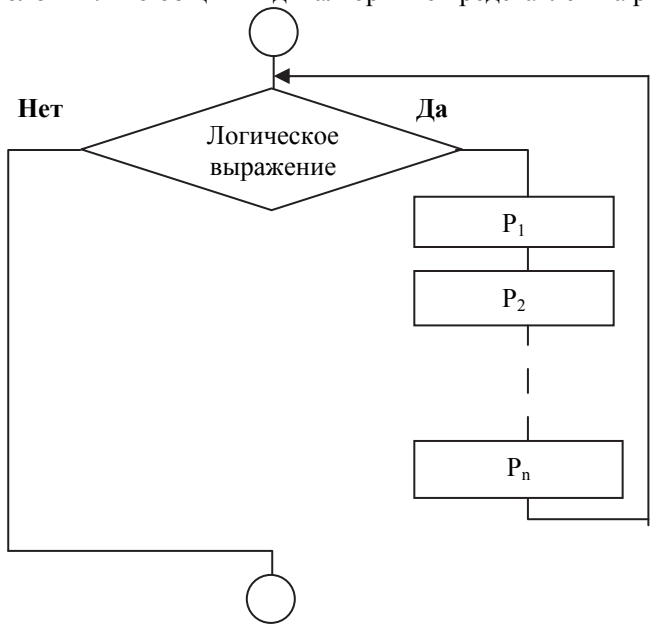


Рис. 54. Общий вид в алгоритме оператора **while**

Синтаксис оператора **while** следующий:

Инициализация начального значения
while логическое выражение**:**
P_1
P_2
.
.
P_n

Здесь приведены обозначения:

P_1, P_2, … P_n – операторы; **while** (пока, до тех пор) – служебное слово языка Python.

Если логическое выражение после служебного слова **while** имеет значение **True** (Истина), то выполняются операторы P_1, P_2, …P_n, после чего проверка логического выражения повторяется. Если логическое выражение имеет значение **False** (Ложь), то происходит выход из цикла. Если условие в заголовке цикла не является истинным с самого начала, цикл **while** не выполняется. Поскольку параметр цикла, который по умолчанию был в конструкции цикла с оператором **for**, в цикле **while** отсутствует, пользователь должен создать переменную, которая бы играла соответствующую роль. Кроме того, при программировании он должен предусмотреть ее инициализацию до выполнения цикла, а в самом цикле выполнить ее увеличение на определенный шаг.

Например, в листинге 32 цикл с оператором **while** обрабатывает данные до тех пор, пока не будет введено слово «Студент».

Листинг 32

```
name=""
while name !='Студент':
   name=input("Введите ваше имя или слово Студент для выхода: ")
   if name!='Студент':
      print("Ответ= ", name)
```

Задача 1. Найдите сумму целых чисел от 1 до 50, используя оператор цикла **while.**

Ранее мы решали такую же задачу, но с оператором **for.** Что изменится в программе, если применить оператор **while?** Прежде всего надо позаботиться о том, чтобы какая-нибудь переменная менялась в цикле от 1 до 50. Ведь такой величины, как параметр цикла, нет в конструкции **while.** В нашем примере такую роль будет играть переменная **k.** Задав в качестве условия выхода из цикла **k!= 50** и применяя в цикле оператор **sum=sum+k,** мы просуммируем все пятьдесят слагаемых и получим в ответе число **1275**. Разработка алгоритма решения задачи представлена на рис. 55.

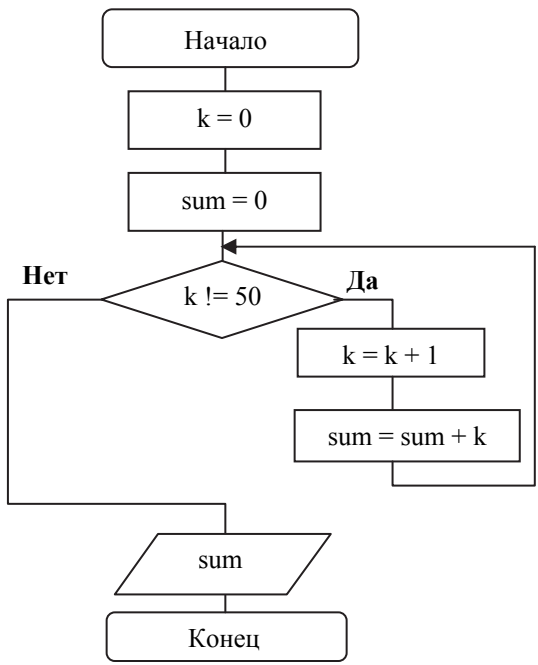


Рис. 55. Алгоритм решения задачи

В листинге 33 приведен код программы, отвечающий за решение задачи.

Листинг 33

```
k=0
sum=0
while k !=50:
   k=k+1
   sum=sum+k
print("Сумма чисел от 1 до 50 = " , sum)
```

Скриншот, приведенный на рис. 56, подтверждает правильность кода программы.

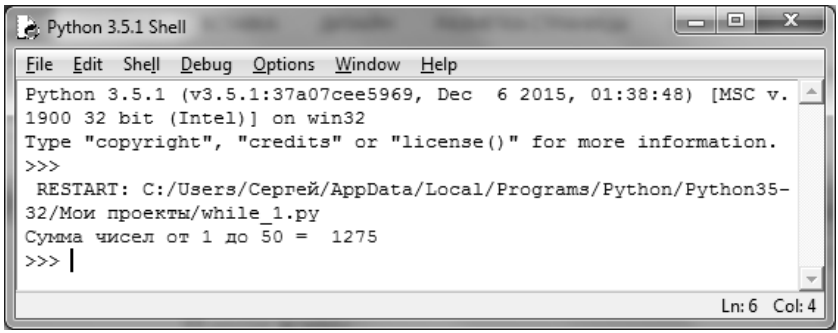


Рис. 56. Сумма чисел от 1 до 50 равна 1275

При вычислении значений функций или каких-либо числовых последовательностей (например, арифметической прогрессии), их часто записывают в виде специальных сумм, называемых **рядами.** Многие числа, функции, алгоритмы численных методов могут быть записаны с помощью рядов или итерационных алгоритмов, которые позволяют вычислять их приближенные значения с заданной точностью. При программировании таких задач используются итерационные методы для организации цикла, в котором производится вычисление некоторой рекуррентной формулы.

Рекуррентная формула – это такая формула, которая сводит вычисления **n**-го члена последовательности к вычислению нескольких предыдущих членов (или, часто, одного предыдущего члена этой последовательности – **n-1**). В общем случае такая формула имеет вид:

$\mathbf{S_n = S_{n-1} + U_n}$,

где $\mathbf{S_n}$ – сумма первых **n** слагаемых ряда, которая образуется из суммы, полученной на предыдущем шаге $\mathbf{S_{n-1}}$;

$\mathbf{U_n}$ – слагаемое, полученное на текущем шаге.

Рассмотрим ряд задач, в которых будут использоваться итеративные циклы и рекуррентные соотношения.

Задача 1. Среди чисел $1, 1+\frac{1}{2}, 1+\frac{1}{2}+\frac{1}{3}, \ldots$ найдите первое число, большее вводимого значения переменной **a.**

Комментарий. Алгоритм решения данной задачи относится к алгоритмам вычисления членов бесконечных последовательностей. Очередной член беско-

нечной последовательности обозначим как **b.** Его номер, который совпадает со значением знаменателя дроби, добавляемой к предыдущему члену для получения значения очередного члена последовательности, обозначим как **n.** Тогда итерационная формула вычисления очередного члена последовательности примет вид:

$$b_n = b_{n-1} + \frac{1}{n}.$$

Разработка алгоритма решения задачи представлена на рис. 57.

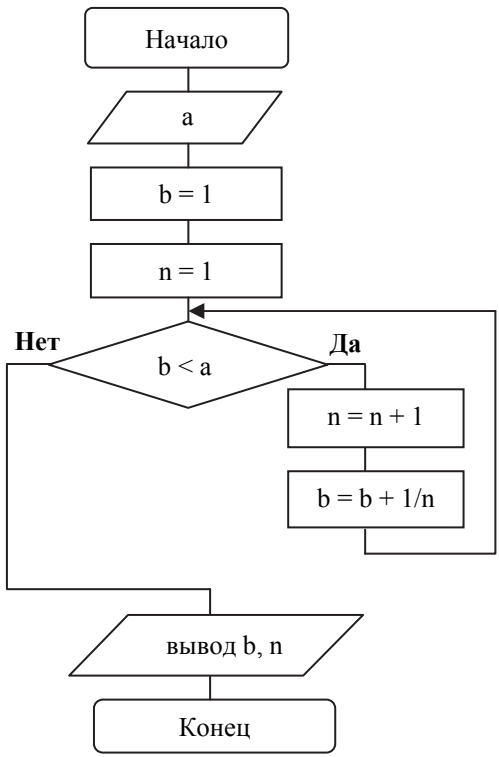


Рис. 57. Алгоритм решения задачи

В листинге 34 приведен код программы, отвечающий за решение задачи.

Листинг 34

```
a=float(input("Введите число a: "))
b=1
n=1
print("\n   n   "+"         b")
print()
while b<a:
   n+=1
   b=b+1/n
   print("  ", n,"         ",b)
```

```
print("Первое число, превышающее значение a: ",'{0:.4f}'.format(b), "Количество итераций: ", n)
```

На рис. 58 показан результат работы программы при введенном значении **a,** равном 5.

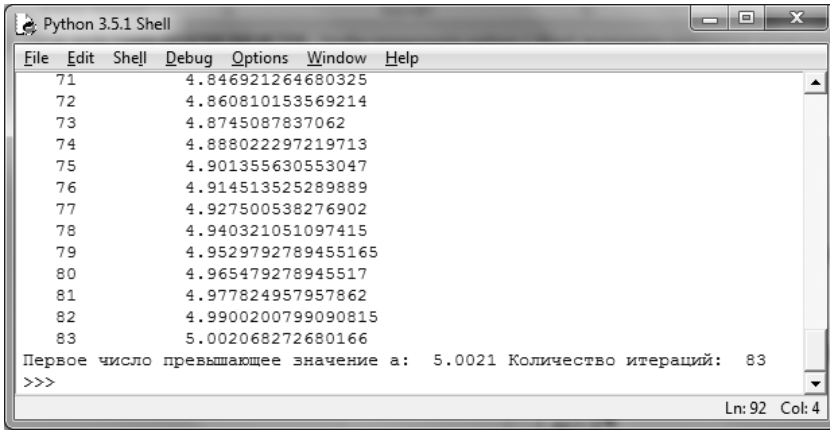


Рис. 58. Результат работы программы при **a**=5

Задача 2. Вычислите с точностью $\varepsilon=10^{-4}$ корень уравнения $\mathbf{e^x + x=0}$, воспользовавшись итерационной формулой $x_{i+1}=-e^{x_i}$, а i = 0, 1, 2, ….; $x_0 = 0$. Закончите итеративный процесс, как только $|x_{i+1}-x_i|$ станет меньше $\varepsilon=10^{-4}$.

Комментарий. Для решения данной задачи необходимо из очередного значения вычисленного корня $\mathbf{x_{i+1}}$ вычитать предыдущее значение корня $\mathbf{x_i}$. Для этого при каждом повторении цикла перед вычислением очередного корня **x** сохраняем в переменной **a** текущее значение **x** (оно становится предыдущим). Цикл заканчивается, когда разность между **x** (т. е. $\mathbf{x_{i+1}}$ и **a** (т. е. $\mathbf{x_i}$) станет меньше $\left(\varepsilon = 10^{-4}\right)$. В программе обозначим ε как **e1.**

Разработка алгоритма решения задачи представлена на рис. 59.

Программный код решаемой задачи представлен в листинге 35.

Листинг 35

```
from math import *
a=1 #Инициализация значения a
x=0
x1=-exp(x)
e1=0.0001
while abs(x-a)>e1:
   a=x
   x=-exp(a)
print("Корень уравнения, вычисленный с заданной точностью",'{0:.14f}'.format(x))
print("Корень уравнения, вычисленный напрямую",x1)
```

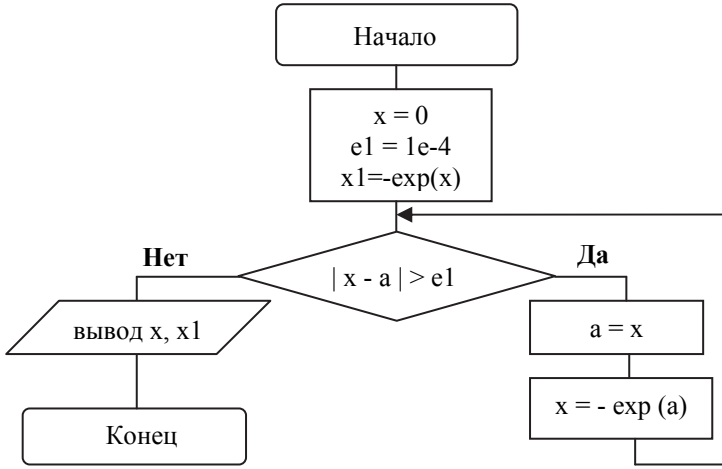


Рис. 59. Алгоритм решения задачи

Результат работы программы представлен на рис. 60. Из него видно, что значения корня уравнения, вычисленное напрямую, имеет более грубое значение.

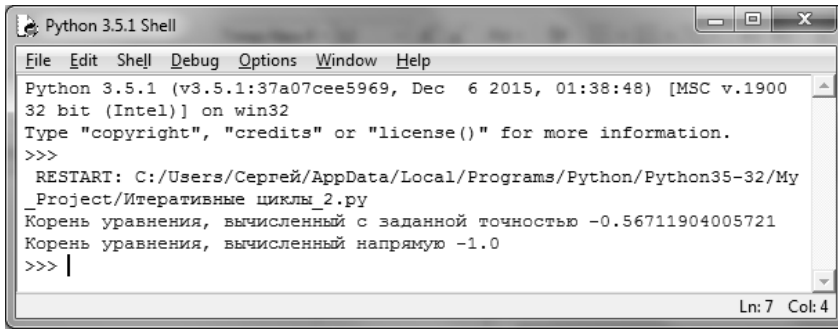


Рис. 60. Результат работы программы.

Задача 3. Разработайте программу, вычисляющую приближенное значение функции **exp(x)** с заданной точностью ε.

Комментарий. Функция **exp(x)** может быть представлена бесконечным рядом

$$\exp(x)=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots+\frac{x^n}{n!}+\ldots$$

Для вычисления значения функции **exp(x)** с точностью ε необходимо просуммировать все члены ряда, которые по модулю превышают заданную точность, т. е. ε. Следовательно, необходимо организовать циклический процесс для вычисления суммы и повторять его пока выполняется условие

$$\left|\frac{x^n}{n!}\right|>\varepsilon$$

Рассматривая приведенную выше формулу можно заметить, что соседние члены ряда (обозначим их как C_n и C_n+1) связаны между собой соотношением

$$C_{n+1}=C_n \cdot \frac{x}{n+1},$$ где n = 0, 1, 2, ..., причем $C_0 = 1$.

Используя это соотношение, можно последовательно вычислить все члены ряда. При этом не возникает необходимости в выполнении в явном виде операций вычисления факториала и возведения в степень. Рассмотренное свойство степенных рядов позволяет организовывать чрезвычайно простые и очень эффективные (в смысле оптимизации скорости вычислений) алгоритмы вычисления их частичных сумм.

Еще раз отметим, что формулы, позволяющие вычислять последующие значения чего-либо на основе предыдущего значения, называются **рекуррентными соотношениями.** Разработка алгоритма решения задачи представлена на рис. 61.

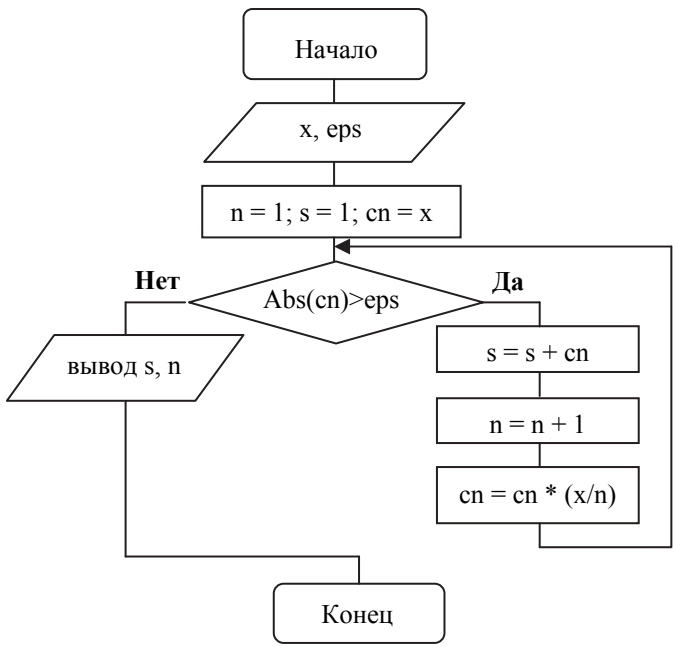


Рис. 61. Алгоритм решения задачи

Из результатов, представленных на рис. 62, видно, что значение функции **s** вычислено более точно, чем значение функции **y,** полученное напрямую посредством оператора **y=exp(x).**

Листинг 36

```
from math import *
x=float(input("Введите значение x: "))
eps=float(input("Введите точность вычислений: "))
s=1
n=1
```

```
cn=x
while abs(cn)>eps:
   s=s+cn
   n=n+1
   cn=cn*(x/n)
y=exp(x)
print("Значение функции, вычисленное с заданной точностью: ", s)
print("Значение функции y, вычисленное через встроенную функцию ", y)
print("Количество членов ряда: ", n)
```

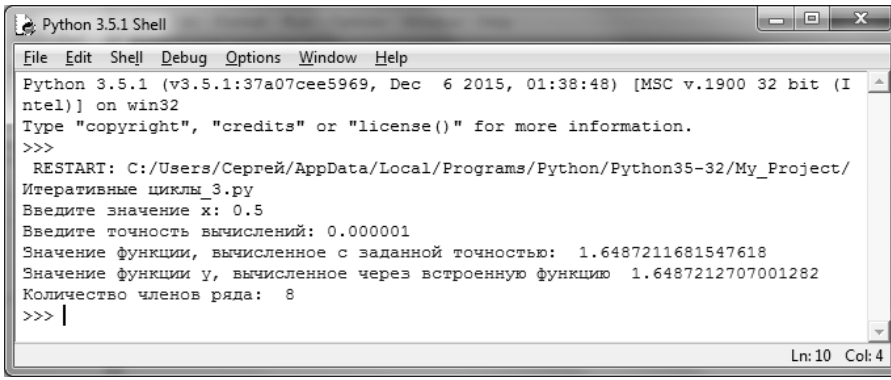


Рис. 62. Результаты работы программы

Задача 4. Вычислите сумму членов знакопеременной убывающей последовательности с заданной точностью ε.

$$\frac{x-1}{1!}-\frac{(x-1)^2}{2!}+\frac{(x-1)^3}{3!}-\ldots+(-1)^n\frac{(x-1)^{n+1}}{(n+1)!}\ldots,$$

Комментарий. Вычисление с заданной точностью ε означает, что суммирование членов ряда надо продолжать до тех пор, пока очередной вычисленный член ряда не станет меньше по абсолютной величине числа ε.

Отметим, что во многих задачах непосредственный подсчет очередного члена связан с вычислительными трудностями. В этом случае целесообразно использовать **рекуррентную формулу,** которая позволяет вычислить значение переменной на следующем шаге, используя ее значение на текущем шаге $a_{n+1}=a_n\cdot q$. Выражение для **q** можно получить, разделив $\mathbf{a_{n+1}}$ член на $\mathbf{a_n}$ член.

Приведем вывод рекуррентной формулы для заданного в задаче ряда. Формула **n**-го члена

$$a_n=(-1)^n\frac{(x-1)^{n+1}}{(n+1)!},$$

тогда формула **n+1** члена

$$a_{n+1}=(-1)^{n+1}\frac{(x-1)^{n+1+1}}{(n+1+1)!}=(-1)^{n+1}\frac{(x-1)^{n+2}}{(n+2)!}.$$

Разделив $\mathbf{a_{n+1}}$ член на $\mathbf{a_n}$, получим выражение для **q**

$$q=\frac{a_{n+1}}{a_n}=\frac{(-1)^{n+1}\cdot(x-1)^{n+2}\cdot(n+1)!}{(n+2)!\cdot(-1)^{n}\cdot(x-1)^{n+1}}=-\frac{(x-1)}{n+2}.$$

Таким образом, рекуррентная формула для данного ряда примет вид

$$a_{n+1}=-a_n\cdot\frac{(x-1)}{n+2}.$$

Выбор начального значения номера члена ряда **n** для нашего случая будет **n=0,** так как при подстановке этого значения в формулу **n-го** члена ряда

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

мы получим значение первого члена, равного **x–1** или **a=x–1.**

В листинге 37 приведен код программы, отвечающий за решение задачи.

Листинг 37

```
from math import *
x=float(input("Введите значение x: "))
e1=float(input("Введите точность вычислений: "))
n=0
a=x-1
s=0
print("   n"+"         a")
while abs(a)>e1:
   print("  ",n,"       ",a)
   s=s+a
   a=-a*(x-1)/(n+2)
   n=n+1
print("Сумма= ",s)
```

На рис. 63 представлены результаты работы программы.

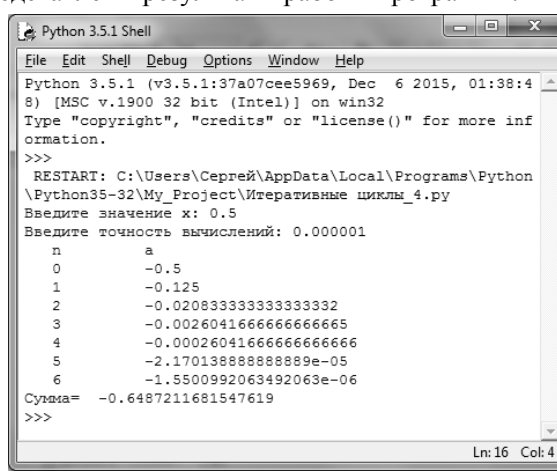


Рис. 63. Результаты работы программы

Задача 5. Вычислите с точностью $\varepsilon=10^{-5}$ корень уравнения $f(x)=x^3-2x^2+x-3=0$, воспользовавшись итерационной формулой

$$x_{i+1}=x_i-\frac{f(x)}{f'(x)} \quad i=0,1,2\ldots, x_0=2{,}2.$$

Комментарий. Проверим правильность решения подстановкой найденного корня в уравнение. Вычислим производную f(x).

$F'(x)=3x^2-4x+1$.

Обозначим ***x*** – текущее приближение к корню, **a** – предыдущее приближение, **f** – значение функции **f(x)** для предыдущего значения, **p** – значение производной **f'(x)** для предыдущего значения, **i** – номер итерации, совпадающий с номером текущего приближения к корню уравнения, **y** – значение функции **f(x)** для найденного с заданной точностью корня уравнения.

Будем считать, что заданная точность ε обеспечена, если модуль разности между текущим и предыдущим значениями корня меньше точности ε, т. е. для нашего случая **|x – a |< ε**. В листинге 38 приведен код программы, отвечающий за решение задачи.

Листинг 38

```
from math import *
x=2.2
e1=0.00001
i=0
a=1
print("   i"+"         x")
while abs(x-a)>e1:
    a=x
    f=pow(x,3)-2*a*a+a-3
    p=3*a*a+4*a+1
    x=a-f/p
    i=i+1
    print("  ",i,"        ",x)
y=pow(x,3)-2*x*x+x-3
print("Искомый корень",'{0:.14f}'.format(x))
print("Значение y при подстановке корня в уравнение",'{0:.14f}'.format(y))
```

Таким образом, из результатов, представленных на рис. 64, видно, что значение **y** при подстановке в уравнение найденного корня, оказывается более точным.

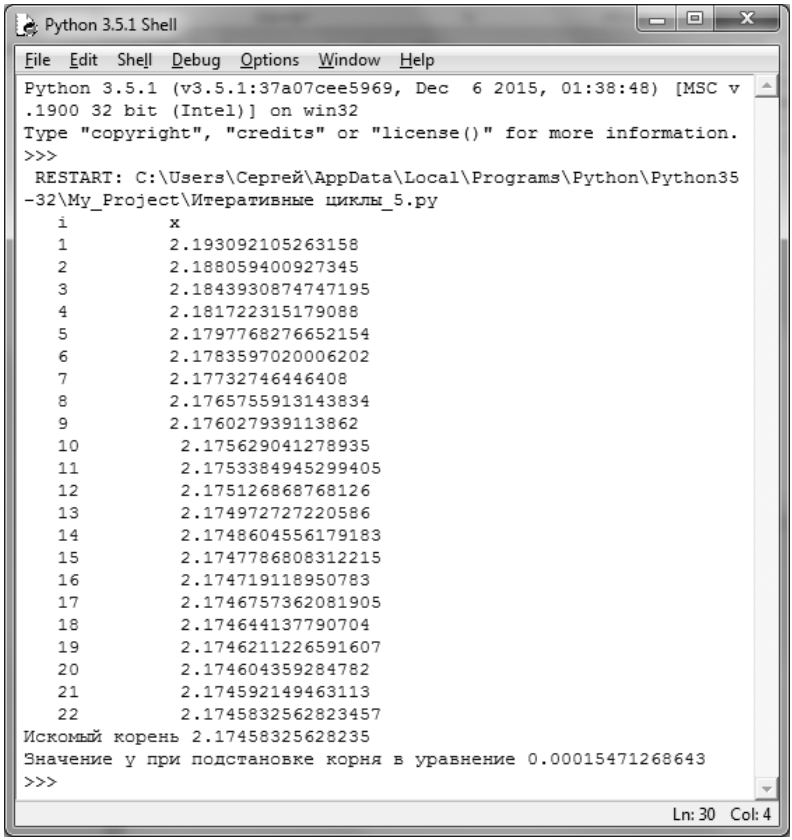


Рис. 64. Результаты работы программы

Упражнения

Вопрос 1. Определите, какое значение находится в ячейке y после выполнения группы операторов?

```
k=0
y=3
while k<y:
   y=y+1
   k=k+3
print("y= ",y)
```

Ответ. В цикле с оператором **while** выход из цикла происходит тогда, когда логическое выражение имеет значение **False** (Ложь). Подставив исходные значения, получаем, что логическое выражение истинно. При выполнении операторов цикла в ячейку **y** заносится число четыре, а в ячейку **k** заносится число три. Вновь проверяется логическое выражение (3<4). Его результат: **Истина.** При втором прохождении цикла в ячейку **y** заносится число 5 (y=y+1), а в ячейку **k** – число шесть (k=k+3). Проверив логическое выражение, убеждаемся в том, что

оно ложно (6<5), следовательно, происходит выход из цикла. В ячейке **y** хранится число **5**.

Вопрос 2. Определите, какое значение находится в ячейке s после выполнения группы операторов?

```
s=7
i=1
while i*i==2:
   s=s+1/i
   i+=i
```

Ответ. В данном примере надо быть внимательным и заметить, что логическое выражение всегда будет иметь значение **False** (Ложь), следовательно, тело цикла выполняться не будет. Значение ячейки **s** останется неизменным и будет равно **7**.

Вопрос 3. Каким будет значение переменной z после выполнения группы операторов?

```
a=1
z=4
while a<=3:
   a+=a
   z=a+1
```

Ответ. Для того чтобы логическое выражение стало ложным и произошел выход из цикла, необходимо, чтобы в ячейке **a** находилось число четыре. Следовательно, при первом прохождении цикла в ячейке **a** будет значение, равное двум, а в ячейке **z** будет находиться число три. Таким образом, цикл выполнится еще два раза. Оператор **z=a+1** увеличит значение ячейки **z** на два. Окончательный ответ: **z** равно **5.**

Вопрос 4. Определите, какое значение находится в ячейке a после выполнения группы операторов?

```
a=2
d=1
while a+d<=7:
   a=a+1
   d=d+1
a=a*d
```

Ответ. При первом вхождении в цикл в ячейке **a** будет находиться значение три (после выполнения оператора **a=a+1**), а в ячейке **d** – значение два (после выполнения оператора **d=d+1**). Сумма значений этих двух ячеек, проверяемая в логическом выражении, будет равна пяти. Условие $5 \leq 7$ истинно, следовательно, цикл выполнится еще раз. Увеличившись на единицу, значения ячеек **a** и **d** примут, соответственно, значения четыре и три. Их сумма будет равна семи, значит, цикл выполнится третий раз. Увеличившись на единицу, значение ячейки **a** будет равно пяти, а значение ячейки **d** – равно четырем. Логическое выражение: **5+4 ≤ 7** оказывается ложным, происходит выход из цикла, и выполняется

оператор **a=a*d**. Умножив пять на четыре, получим число **20,** которое и будет ответом в данном упражнении.

Вопрос 5. Определите, какое значение будет в ячейке n после выполнения группы операторов?

```
n=2
x=1
while x<=4:
   x=x+1
   n=n+x
```

Ответ. Выход из цикла с оператором **while** происходит тогда, когда логическое выражение принимает значение **False** (Ложь). В данном примере выход из цикла произойдет, когда в ячейке **x** будет находиться число, большее по величине, чем число четыре. Видно, что в цикле ячейка **x** увеличивает свое значение на единицу, т. е. выход из цикла произойдет, когда в ячейке **x** будет число пять. Таким образом, в цикле нужно подсчитывать значение ячейки **n,** складывая его со значением переменной **x.** На момент выхода из цикла в ячейке **n** находится число **16.**

Вопрос 6. Каким будет значение переменной a после выполнения группы операторов?

```
a=1
z=1
while a<=3:
   a=a+1
a=a+z
a=a+10
```

Ответ. В цикле с оператором **while** выполняется один оператор **a=a+1.** Это можно понять по отступу, обозначающему выполнение одного оператора в цикле. Выход из цикла произойдет тогда, когда в ячейке **a** будет число четыре. Выполнив оператор **a=a+z,** получаем, что в ячейке **a** находится число пять. Выполняется оператор **a= a+10.** Значение ячейки **a** равно **15.**

Вопрос 7. Определите, какое значение находится в ячейке s после выполнения группы операторов?

```
a=20
d=5
while a – d>=10:
   a=a-1
   d=d+1
s=a+d
```

Ответ. Если вы хорошо знаете теоретический материал, то данное упражнение не покажется вам сложным. Проверив логическое выражение, убеждаемся в том, что оно истинно. Далее вам придется уменьшать значение ячейки **a** на единицу, значение ячейки **d** увеличивать на единицу, каждый раз проверяя разность значений ячеек. Выход из цикла происходит тогда, когда в ячейке **a** будет нахо-

диться число 17, а в ячейке **d** – число 8. Выполнив оператор **s=a+d,** получим, что значение ячейки **s** = **25.**

Вопрос 8. Каким будет значение переменной a после выполнения группы операторов?

```
a=1
z=1
while z<=4:
   z+=z
   z=a+1
   a=z
```

Ответ. В данном примере в теле цикла выполняются три оператора. Можно установить следующую закономерность: значения ячеек **z** и **a** при выполнении операторов будут равны, так как то значение, которое находится в ячейке **z,** переходит в ячейку **a** после выполнения оператора **a=z.** Выход из цикла произойдет тогда, когда в ячейке **z** будет находиться число пять, следовательно, и в ячейке **a** тоже будет храниться число пять, что и является ответом в данном упражнении.

Вопрос 9. Определите, какое значение находится в ячейке p после выполнения группы операторов?

```
p=1
i=2
while p<9:
   p=p*i
i=i+1
p=p*i
```

Ответ. В цикле выполняется оператор **p=p*i**. Остальные операторы выполняются после завершения цикла. Значение ячейки **i** остается неизменным и будет равно двум, меняется только значение ячейки **p** после выполнения оператора **p=p*i.** Таким образом, необходимо умножать на два значение, которое находится в ячейке **p,** и проверять условие выхода из цикла. В ячейку **p** последовательно будут заноситься значения 2, 4, 8, 16. Из этого перечисления видно, что, когда в ячейке **p** находится число 16, происходит выход из цикла. Выполнив оператор **i=i+1,** получаем, что **i** равно трем. Выполнив оператор **p=p*i** (16*3), получим, что в ячейке **p** хранится число **48.**

Примеры решения задач

Задача 1. Вводится последовательность вещественных чисел. Известно, что последний элемент последовательности равен пяти. Найдите количество положительных чисел и минимальное из них. Разработка алгоритма решения задачи представлена на рис. 65.

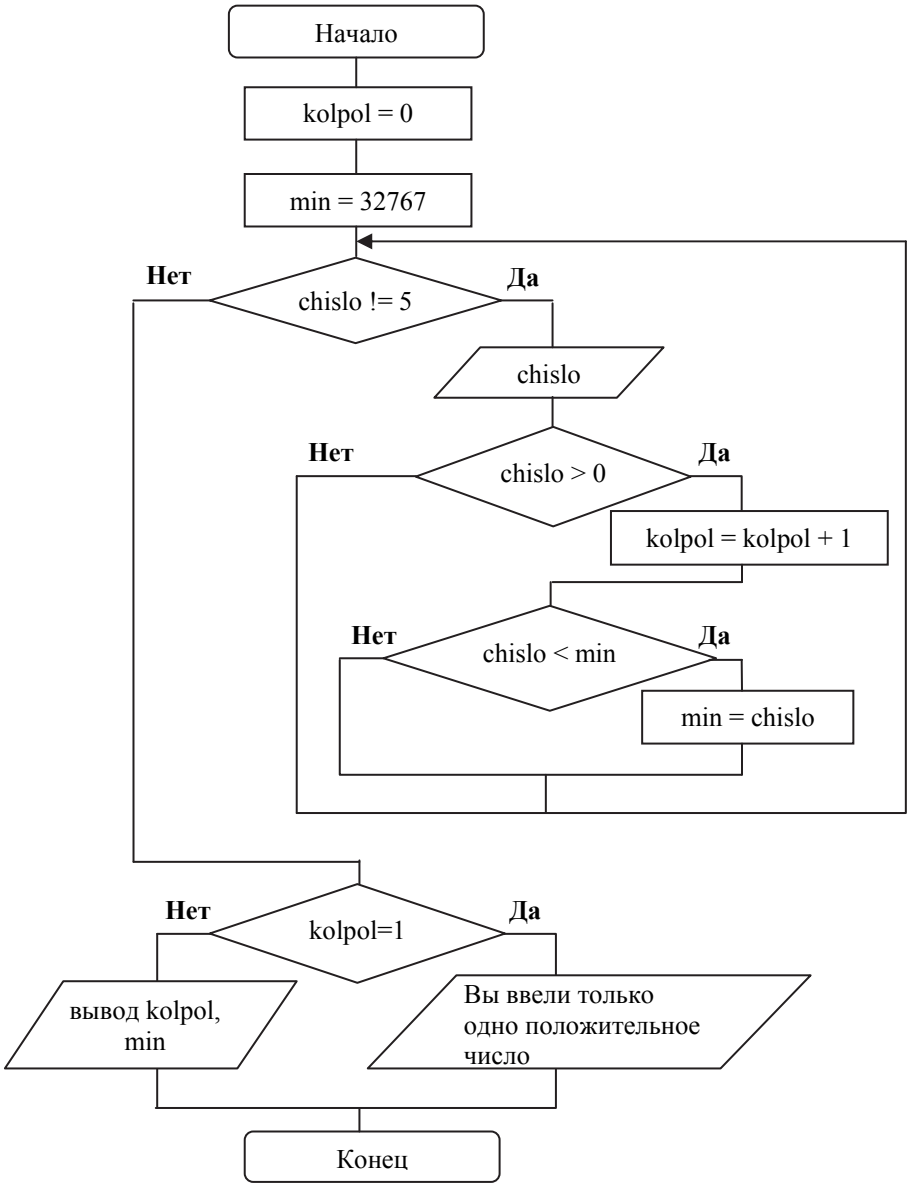


Рис. 65. Алгоритм решения задачи

Комментарий. Выход из цикла с оператором **while** осуществляется тогда, когда логическое выражение принимает значение **False.** Следовательно, до тех пор, пока пользователь вводит значения, не равные числу 5, в цикле будут выполняться соответствующие операторы, необходимые для поиска положительных чисел и минимального из них. В листинге 39 приведен код программы, отвечающий за решение задачи.

Листинг 39

```
kolpol=0
min=32767
chislo=0
while chislo!=5: #Пока число, введенное в цикле, не равно числу 5, выполнять тело цикла
   chislo = float(input("Введите число   "))
   if chislo>0: #Проверка на положительность очередного введенного #числа
      kolpol=kolpol+1
      if chislo<min:  #Реализация алгоритма поиска минимального элемента
         min=chislo
if kolpol==1:
   print("Вы ввели только одно положительное число: 5, предназначенное для выхода из программы")
else:
   print("Количество положительных чисел =" , kolpol)
   print("Минимальное из них =" , min)
```

Задача 2. Вводится последовательность вещественных чисел. Известно, что последний элемент последовательности равен пяти. Определите, каких среди них больше: положительных или отрицательных. Разработка алгоритма решения задачи представлена на рис. 66.

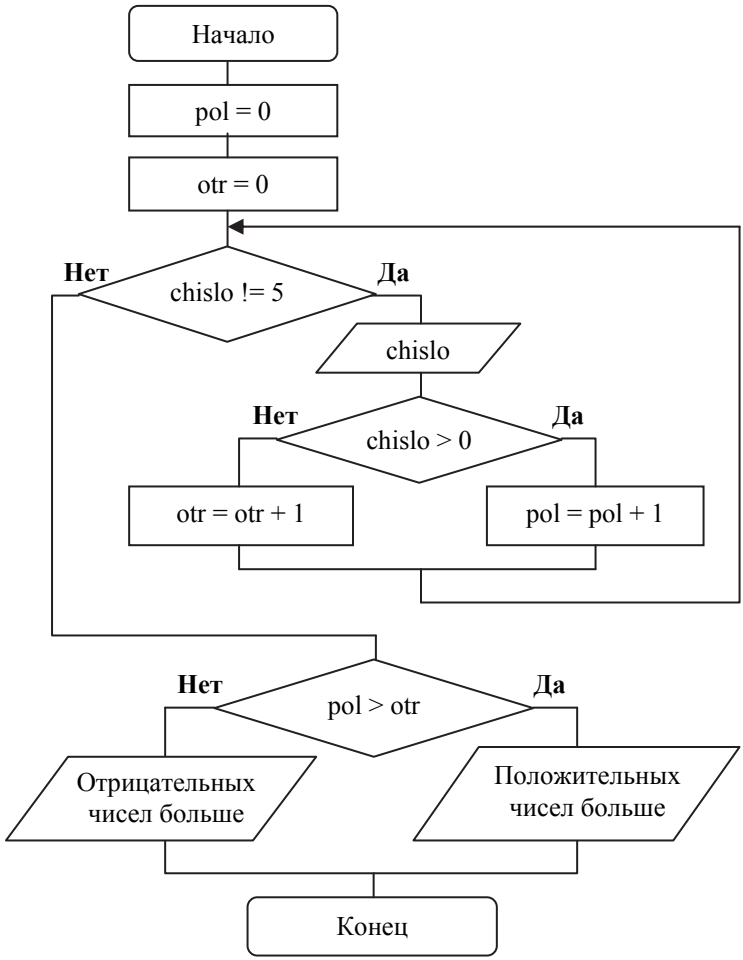


Рис. 66. Алгоритм решения задачи

В листинге 40 приведен код программы, отвечающий за решение задачи.

Листинг 40

```
pol=0
otr=0
chislo=0
while chislo!=5:  #Пока введенное число не равно числу 5, выполнять тело #цикла
    chislo=float(input("Введите число ")) #Ввод очередного числа
```

```
    if chislo>0:  #Проверка: положительно ли введенное число?
        pol=pol+1
    else:
        otr=otr+1
if pol>otr:
    print("Положительных чисел больше ")
else:
    print("Отрицательных чисел больше ")
```

Задача 3. Вводится последовательность вещественных чисел, не равных нулю. Известно, что последний элемент последовательности равен пяти. В программе должна вычисляться сумма всех положительных (кроме последнего, равного пяти) и сумма всех отрицательных чисел. Разработка алгоритма решения задачи представлена на рис. 67.

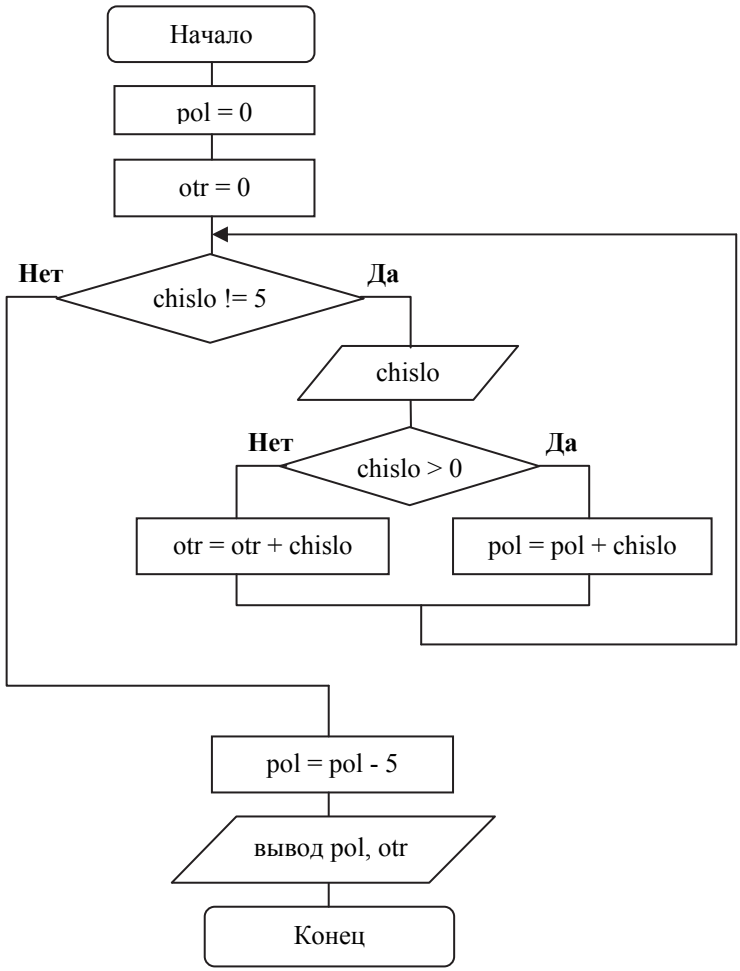


Рис. 67. Алгоритм решения задачи

В листинге 41 приведен код программы, отвечающий за решение задачи.

Листинг 41

```
pol=0
otr=0
chislo=0
while chislo !=5:
```

```
   chislo=float(input("Введите число ")) #Ввод очередного числа
   if chislo>0: #Проверка: положительно ли введенное число?
      pol=pol+chislo #Увеличение значения ячейки на величину числа в случае
#истинности проверяемого условия
   else:
      otr=otr+chislo #Увеличение значения ячейки на величину числа в случае,
#если проверяемое условие ложно
pol=pol – 5 #Из конечной суммы положительных чисел отнимаем 5 для
#того, чтобы не исказился результат
print("Сумма положительных чисел равна ",pol)
print("Сумма отрицательных чисел равна ",otr)
```

Задача 4. Вводится последовательность целых чисел, не равных нулю. Известно, что последний элемент последовательности равен нулю. Найдите среднее арифметическое этих чисел. Разработка алгоритма решения задачи представлена на рис. 68.

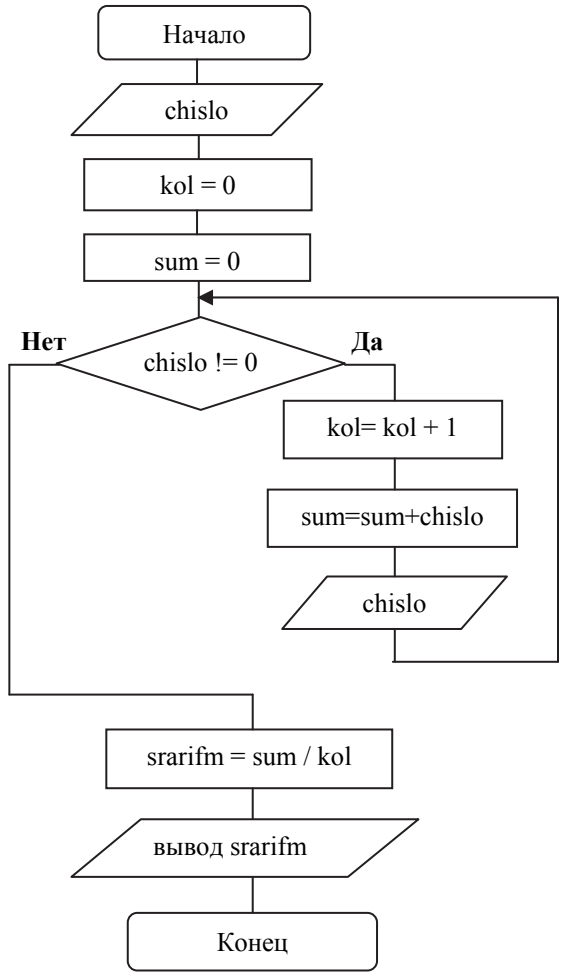


Рис. 68. Алгоритм решения задачи

В листинге 42 приведен код программы, отвечающий за решение задачи.

Листинг 42

```
k=0
sum=0
chislo=int(input("Введите число: "))
while chislo!=0:
    k=k+1
```

```
    sum=sum+chislo
    chislo=int(input("Введите число: "))
srarifm=sum/k
print("Среднее арифметическое чисел: ", srarifm)
input("Нажмите Enter, чтобы завершить программу")
```

Задача 5. Вводится последовательность вещественных чисел, не равных нулю. Известно, что последний элемент последовательности равен нулю. Найдите количество отрицательных чисел и их среднее арифметическое. Разработка алгоритма решения задачи представлена на рис. 69.

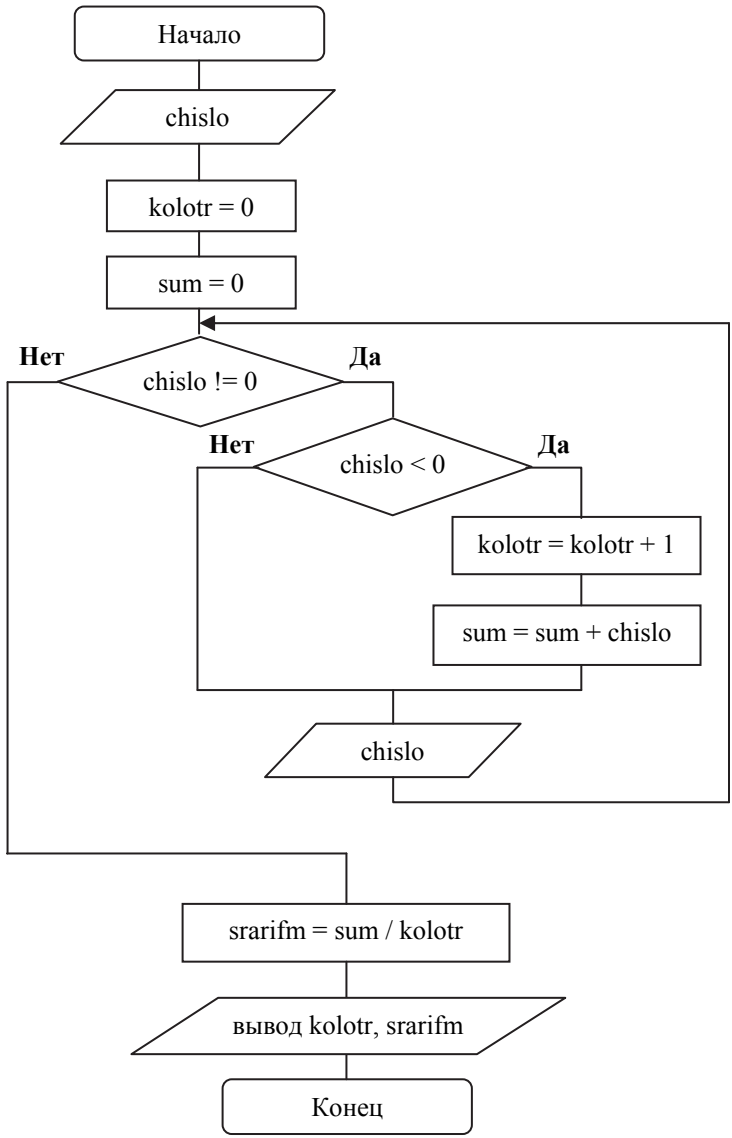


Рис. 69. Алгоритм решения задачи

В листинге 43 приведен код программы, отвечающий за решение задачи.

Листинг 43

```
kolotr=0
sum=0
srarifm=0
chislo=float(input('Введите число '))
while chislo!=0:
   if chislo<0:
      kolotr=kolotr+1
      sum=sum+chislo
   chislo=float(input('Введите число '))
if kolotr==0:
   print("Отрицательных чисел нет")
else:
   srarifm=sum/kolotr
   print('Среднее арифметическое чисел =',srarifm)
   print('Количество отрицательных чисел =',kolotr)
```

Контрольные вопросы

1. Расскажите, в каких случаях применяются циклы с неизвестным числом повторений.
2. Какая циклическая структура может считаться итеративной?
3. Нарисуйте общий вид алгоритма оператора цикла while.
4. Напишите синтаксис оператора цикла while.
5. Расскажите о работе оператора цикла while. Приведите примеры.
6. В каких случаях применяются рекуррентные соотношения? Расскажите об алгоритме вывода рекуррентной формулы.

Задачи для самостоятельного решения

1. Разработайте алгоритм и программу решения следующей задачи. Найдите максимальное значение функции y=Cosx/Sinx на отрезке [a,b] с шагом h.
2. Вводится последовательность вещественных чисел. Известно, что последний элемент последовательности равен 5. Найдите количество положительных чисел и минимальное из них. Разработайте алгоритм и программу.
3. Вводится последовательность целых чисел. Известно, что последний элемент последовательности равен 1. Определите, каких среди них больше: положительных или отрицательных. Разработайте алгоритм и программу.
4. Вводится последовательность вещественных чисел, не равных нулю. Известно, что последний элемент последовательности равен 0. В программе должны вычисляться суммы всех положительных и всех отрицательных чисел. Разработайте алгоритм и программу.

5. Вводится последовательность целых чисел. Определите сумму положительных чисел до первого отрицательного числа. Разработайте алгоритм и программу.
6. Вычислите сумму не равных нулю чисел и выведите эту сумму в качестве ответа. Разработайте алгоритм и программу.
7. В программе определяется количество чисел, сумма которых меньше заданного числа. Разработайте алгоритм и программу.
8. Известны начальный вклад клиента в банк и процент годового дохода. Определите срок, через который вклад превысит 1 млн рублей и величину этого вклада. Разработайте алгоритм и программу.
9. Вычислите сумму четных чисел на отрезке от 10 до 30. Разработайте алгоритм и программу.
10. Определите идеальный вес для взрослых людей по формуле: Идеальный вес = Рост – 100. Выход из цикла: значение роста = 250. Разработайте алгоритм и программу.

6. РАБОТА С КОРТЕЖАМИ И СПИСКАМИ

6.1. Объявление кортежей

Кортежи в Python аналогичны структурам, которые называются **массивами.** Приемы работы с массивами рассматривается при обучении таким популярным языкам программирования, как C, Microsoft Visual Basic, Delphi и др. Стоит отметить, что массивы в них определяются как формальное объединение нескольких однотипных объектов (чисел, символов, строк и т. п.), рассматриваемое как единое целое.

В Python **кортеж** определяется как один из видов последовательностей, с которыми можно работать в этом языке программирования. Отличие от привычных массивов данных (если вы умеете программировать на других языках) заключается в том, что последовательность данных, объединенная в кортеж, **не позволяет изменять свои значения.** В этом есть определенные преимущества. Во-первых, увеличивается скорость обработки элементов кортежа, поскольку системе заранее известно, что значения не будут изменяться. Во-вторых, такая структура, как кортеж, может применяться для образования других структур: например, словари, работа с которыми будет рассмотрена позднее. В-третьих, кортеж занимает меньше памяти, чем, например, списки, и, кроме того, элементы кортежа защищены от случайных изменений.

Если массив, как было отмечено выше, представляет собой совокупность однотипных объектов, то кортеж в Python может содержать совершенно разнородные объекты, например, строковые и числовые значения или звуковые файлы, файлы изображений.

Синтаксис объявления кортежей следующий:

Имя кортежа = (элемент1, элемент2, …элементN)

Например

```
korteg=(1, 2, 3, 4, 5)
```

Возможна и другая запись, например, для значений строкового типа,

```
Имя кортежа=(«Элемент1»,
             «Элемент2»,
             «Элемент3»,
             «ЭлементN»)
```

причем в одной строке может располагаться несколько элементов.

Доступ к каждому элементу кортежа в программе осуществляется с помощью **индекса** – целого числа, служащего своеобразным именем элемента в кортеже. При упоминании в программе элемента кортежа сразу за его именем должен следовать индекс элемента в квадратных скобках, например,

```
korteg[i]
```

Важно отметить, что нумерация элементов кортежа начинается с нуля, а не с единицы. Операции по обработке кортежа осуществляются поэлементно в цикле с оператором **for.**

Задача 1. В кортеже, состоящего из заранее заданных целых чисел найдите сумму элементов. Разработка алгоритма решения задачи представлена на рис. 70.

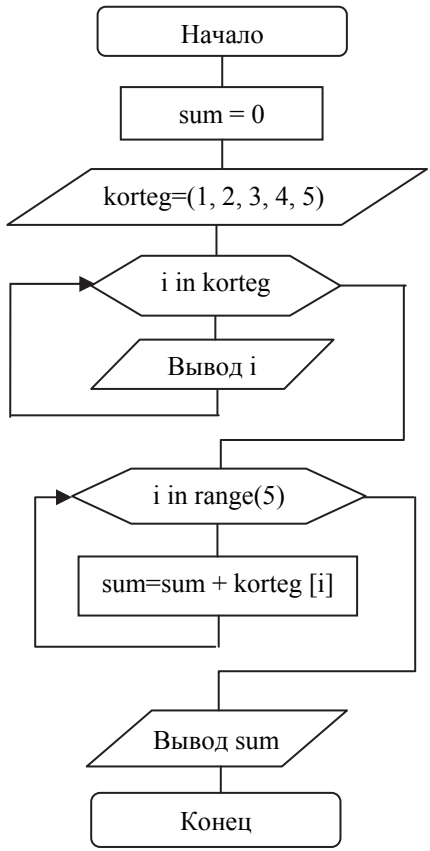


Рис. 70. Алгоритм решения задачи

В листинге 44 приведен код программы, отвечающий за решение задачи, а на рис. 71 показан результат ее выполнения.

Листинг 44

```
sum=0
korteg=(1, 2, 3, 4, 5)
print("Кортеж")
for i in korteg:
        print(i, end= " ")
for i in range(5):
        sum=sum+korteg[i]
print("\nСумма элементов кортежа = ", sum)
```

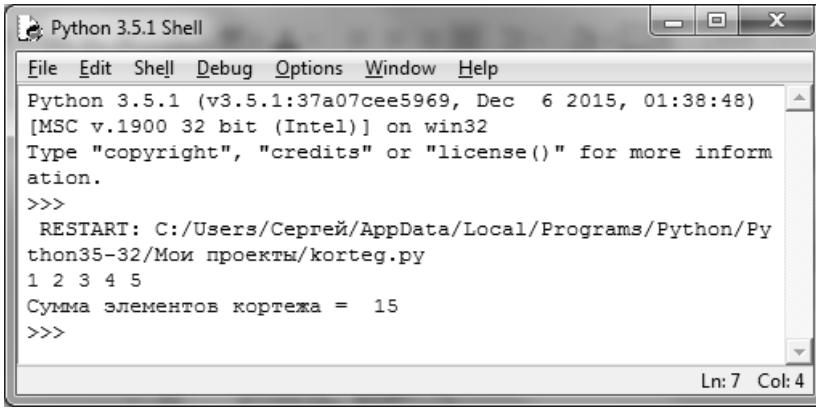


Рис. 71. Результат выполненной программы

6.2. Классические способы обработки кортежей

Решение задач, связанных с обработкой последовательностей данных, базируется на их элементарных приемах обработки. Разбив задачу на логические части, можно значительно ускорить ее решение. Типичными задачами работы с кортежами являются определение наличия в нем заданного элемента, отбор элементов, удовлетворяющих определенным условиям и т. д.

Из всего многообразия существующих приемов, позволяющих манипулировать с данными, представленными в кортежах, можно выделить следующие:

1. Нахождение количества элементов при заданном условии.
2. Нахождение суммы значений элементов при заданном условии.
3. Нахождение произведения значений элементов при заданном условии.
4. Поиск экстремальных значений элементов кортежа (поиск максимального и/или минимального значения).
5. Объединение (сцепление) кортежей.
6. Обмен значений элементов в кортеже.
7. Срезы кортежей.

Ниже описывается реализация перечисленных алгоритмов обработки кортежей. Первые четыре способа чрезвычайно просты, данные алгоритмы неоднократно рассматривались при решении задач в предыдущих главах и в настоящее время не требуют комментариев. Стоит отметить, что в первых четырех листингах очередной элемент кортежа сравнивается с числом 10, однако на практике лучше организовать запрос условия, например, с помощью функции **input.**

Нахождение количества элементов при заданном условии

Листинг 45

```
kol=0
korteg=(13, 2, 23, 14, 5)
for i in korteg:
```

```
            print(i, end=" ")
for i in range(5):
            if korteg[i]>=10:
                kol=kol+1
print("\nКоличество элементов в кортеже >=10 равно ", kol)
```

Нахождение суммы значений элементов при заданном условии

Листинг 46

```
sum=0
korteg=(13, 2, 23, 14, 5)
for i in korteg:
    print(i, end=" ")
for i in range(5):
    if korteg[i]>=10:
        sum=sum+korteg[i]
print("\nСумма элементов в кортеже >=10 равна ", sum)
```

Нахождение произведения значений элементов при заданном условии

Листинг 47

```
pr=1
korteg=(13, 2, 23, 14, 5)
for i in korteg:
    print(i, end=" ")
for i in range(5):
    if korteg[i]>=10:
        pr=pr*korteg[i]
print("\nПроизведение элементов в кортеже >=10 равно ", pr)
```

Нахождение экстремальных значений в кортеже

Листинг 48

```
maxim=-32768
minim=32767
korteg=(13, 2, 23, 14, 5)
for i in korteg:
            print(i, end=" ")
for i in range(5):
            if korteg[i]>maxim:
                maxim=korteg[i]
            if korteg[i]<minim:
                minim=korteg[i]
print("\nМаксимальный элемент в кортеже равен ", maxim)
print("\nМиниимальный элемент в кортеже равен ", minim)
```

Объединение (сцепление) кортежей

Задача. Даны два кортежа. Объедините их между собой.

Комментарий. Очень простая задача, заключающаяся в первоначальном формировании двух исходных кортежей и применении ко второму кортежу операции сложения (+), после чего объединенный кортеж выводится на экран.

Листинг 49

```
korteg=(13, 2, 23, 14, 5)
korteg1=(19,62)
print("\nПервый кортеж")
for i in korteg:
        print(i, end=" ")
print("\nВторой кортеж")
for i in korteg1:
        print(i, end=" ")
print("\nОбъединенный кортеж")
        korteg +=korteg1
for i in korteg:
   print(i, end=" ")
```

На рис. 72 представлен результат работы программы.

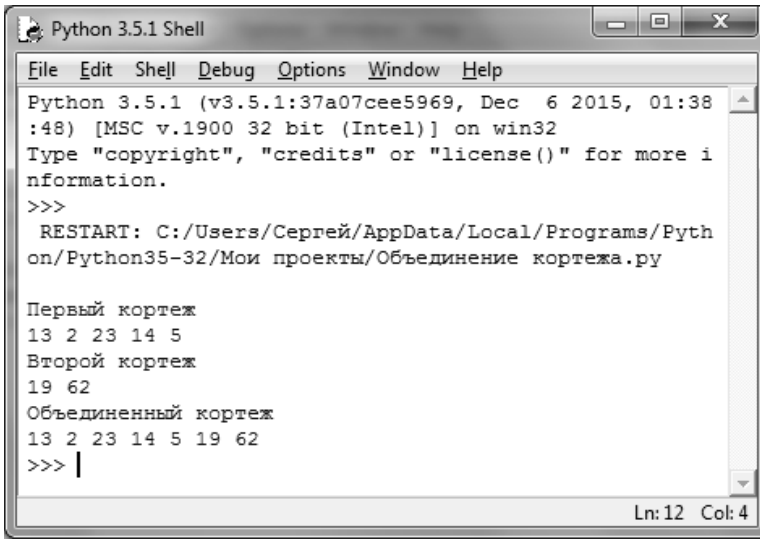


Рис. 72. Сформирован объединенный кортеж

Обмен значений элементов в кортеже

Задача. В кортеже целых чисел найдите максимальный и минимальный элементы, а также осуществите их обмен.

Комментарий. Приемы нахождения максимального и минимального элементов рассматривались ранее, поэтому стоит отметить, что сам обмен значений заключается в выполнении оператора **maxim,minim=minim,maxim**. Код программы представлен в листинге 50.

Листинг 50

```
korteg=(13, 2, 23, 14, 5)
print("\nКортеж")
for i in korteg:
   print(i, end=" ")
minim=32767
maxim=-32768
for i in range(5):
   if korteg[i]>maxim:
     maxim=korteg[i]
   if korteg[i]<minim:
     minim=korteg[i]
print("\nМаксимальный элемент в кортеже равен ", maxim)
print("\nМинимальный элемент в кортеже равен ", minim)
maxim,minim=minim,maxim  #Обмен значений
print("\nПосле обмена ", maxim)
print("\nПосле обмена ", minim)
```

Результат обмена значений элементов можно увидеть на рис. 73.

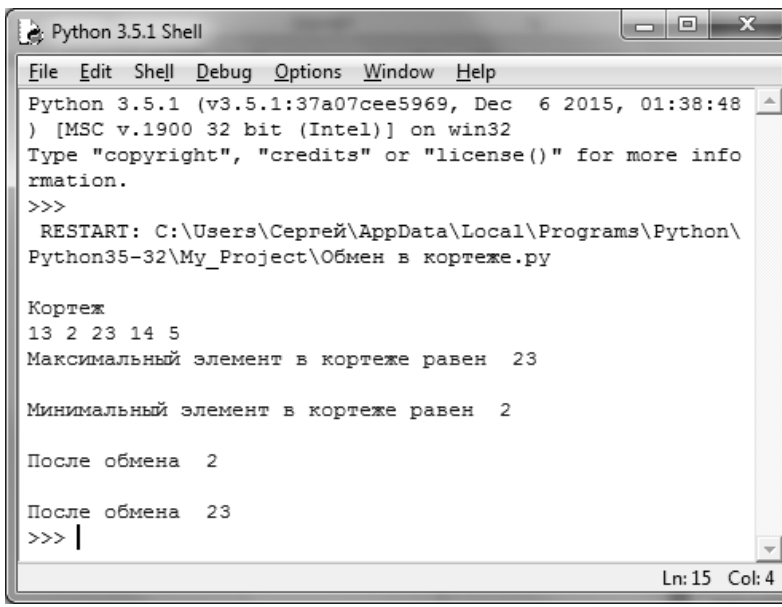


Рис. 73. Результат работы программы

Срезы кортежей

Срез кортежа получается в результате вывода элементов кортежа, находящихся между заранее заданными начальной **a** и конечной **b** позициями элементов. Реализация среза в программе осуществляется с помощью оператора **print**(korteg[a:b]).

Листинг 51

```
korteg=(28, 2, 19, 8, 62, 37, 15, 6)
print("\nКортеж")
for i in korteg:
        print(i, end=" ")
a=int(input("\nВведите начальное значение индекса среза    "))
b=int(input("\nВведите конечное значение индекса среза     "))
print("\nСрез кортежа")
print(korteg[a:b])
```

Результат выполнения программы представлен на рис. 74.

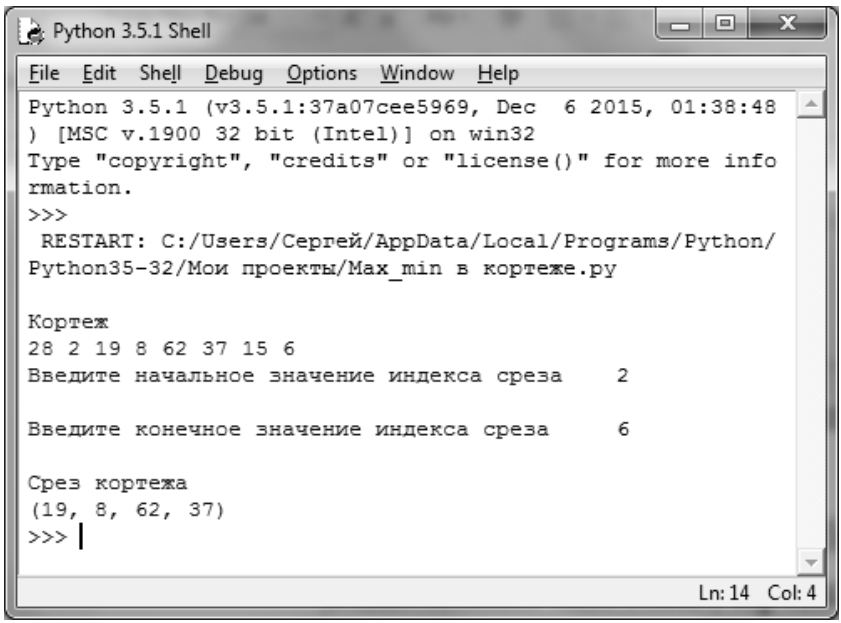


Рис. 74. Результат работы программы

6.3. Работа со списками

Списки в Python функционируют подобно кортежам, с той разницей, что являются изменяемой последовательностью. Таким образом, если такие действия, как добавление нового элемента, удаление элемента, сортировка и др., в кортеже невозможны, то методы работы со списком позволяют это сделать.

Точно так же, как и элементы кортежа, элементы списка содержат лишь ссылку на объект, поэтому в списки могут входить разные типы данных. Данный подход отличается от обработки традиционных массивов в других языках программирования, хотя те, кто пробовал с ними работать, заметит сходства в методах обработки списков в Python с методами обработки массивов.

Рассмотрим синтаксис объявления списков.

Имя списка = [элемент1, элемент2, …элементN]

Например,

spisok=[1, 2, 3, 4, 5]

Типичными задачами при работе со списками являются: определение факта наличия в них заданного элемента и отбор элементов, удовлетворяющих определенным условиям. В обоих случаях используется циклическое сравнение элементов списка с заданным образцом. Для определения факта наличия заданного образца в списке достаточно единственного совпадения, после чего дальнейший просмотр прекращается. Если условие отбора может выполняться для нескольких элементов списка, то необходим просмотр всего списка до конца.

Все рассмотренные выше приемы, позволяющие манипулировать с данными, представленными в кортежах, действительны и для списков, однако их набор может быть существенно расширен в силу изменяемости списков. В частности, можно удалять и добавлять элементы списка, сортировать их.

Пользователь может создавать список, размещая его элементы, которые необходимо обработать, в квадратных скобках (согласно синтаксису, показанному выше), а может генерировать случайным образом. Рассмотрим эти способы подробнее на примере задачи «Найти сумму элементов списка». Точно такую же задачу мы решали, когда работали с кортежами, поэтому будет полезно сравнить способы обработки последовательностей в Python.

Как видно из листинга 52, текст и синтаксис операторов в программе, по сути, не изменились, за исключением того, что элементы списка, которые ввел пользователь для обработки, заключены в квадратные скобки.

Листинг 52

```
sum=0
spisok=[1, 2, 3, 4, 5]
for i in spisok:
        print(i, end="      ")
for i in range(5):
        sum=sum+spisok[i]
print("\nСумма элементов списка = ", sum)
```

На рис. 75 приведен результат выполнения программы.

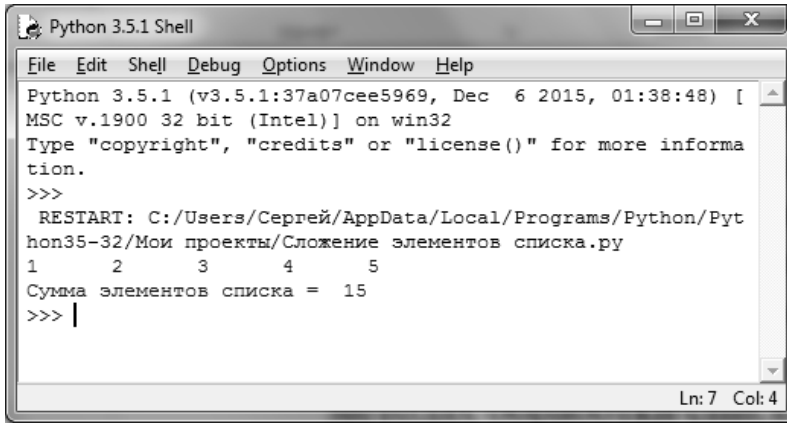


Рис. 75. Результат работы программы

Перейдем к рассмотрению такой возможности, как генерация списков случайным образом. Ее можно осуществить несколькими способами. Во-первых, в известной нам функции **range** можно указать начальное и конечное значения списка, а функция **list** возвратит список. Тогда код предыдущей программы будет выглядеть так, как он представлен в листинге 53.

Листинг 53

```
sum=0
spisok=list(range(1,21))
print(spisok)
for i in range(1,20,1):
        sum=sum+spisok[i]
print("\nСумма элементов списка = ", sum)
```

Во-вторых, можно воспользоваться функцией **sample**, которая из исходной последовательности элементов списка будет возвращать указанное пользователем количество элементов. При этом необходимо воспользоваться модулем **random,** подключив его с помощью инструкции **import.** Как показано в нижеследующей программе (листинг 54), исходный список состоит из девяти элементов. Затем оператором **chislo=random.sample(spisok,5)** генерируются пять элементов из исходного списка.

Листинг 54

```
import random
sum=0
spisok=[10, 20, 30, 40, 50, 60, 70, 80, 90]
chislo=random.sample(spisok,5)
print(chislo, end="     ")
for i in range(0,5):
        sum=sum+chislo[i]
print("\nСумма элементов списка = ", sum)
```

На рис. 76 приведен результат выполнения программы.

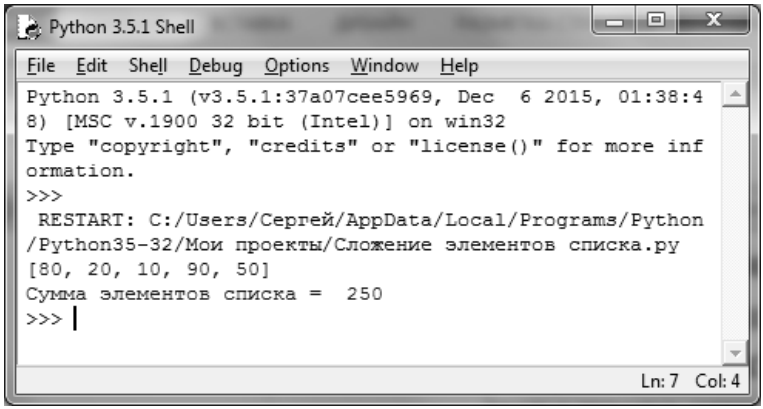


Рис. 76. Результат работы программы

Изменим предыдущий код. Вместо заранее заданного списка сгенерируем его, воспользовавшись функцией **range** и функцией **sample,** параметром которой сделаем число 10, тем самым ограничив количество элементов списка до десяти (листинг 55).

Листинг 55

```
import random
sum=0
chislo=random.sample(range(100),10)
print(chislo)
for i in range(0,10):
        sum=sum+chislo[i]
print("\nСумма элементов списка = ", sum)
```

В-третьих, в модуле **random** есть функция **shuffle** с помощью, которой можно перемешать список случайным образом.

Листинг 56

```
import random
sum=0
spisok=[10, 20, 30, 40, 50]
random.shuffle(spisok)
print(spisok)
for i in range(0,5):
        sum=sum+spisok[i]
print("\nСумма элементов списка = ", sum)
```

На рис. 77 приведен результат выполнения программы.

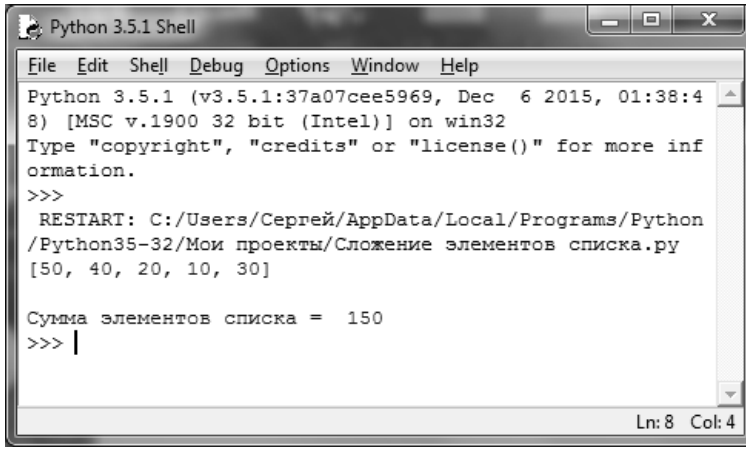


Рис. 77. Результат работы программы

Кроме рассмотренных способов создания списков в программе, можно применить прием динамического создания списка. Как показано в нижеприведенном коде (листинг 57), сначала объявляется пустой список оператором **sp=[].** Затем у пользователя запрашивается количество элементов списка. В цикле пользователь начинает вводить элементы списка, а метод **append()** позволяет добавить их в список.

Листинг 57

```
sp=[]
n=int(input("\nВведите количество элементов списка "))
for i in range(n):
    chislo=int(input("\nВведите число "))
    sp.append(chislo) #Добавляем элементы списка
for i in range(n): #Выводим элементы списка
    print(sp[i], end="  ")
```

Таким образом, мы завершили рассмотрение основных способов создания списков в Python, некоторые из которых будут востребованы в дальнейшем при решении задач.

Для работы со списками в Python предусмотрены методы, некоторые из которых приведены в табл. 6. Рассмотрим их применение на примерах.

Таблица 6. **Методы работы со списками**

Метод	Описание метода
spisok.append(x)	Добавляет значение **x** в конец списка
spisok.insert(i, x)	Вставляет значение **x** в позицию **i**
spisok.extend(spisok1)	Расширяет список, добавляя в конец все элементы списка spisok1
spisok.remove(x)	Удаляет первый элемент в списке, имеющий значение **x**

spisok.pop(i)	Возвращает значение в позиции номер **i** и одновременно удаляет его из списка. Если аргумент не передан, возвращается и удаляется последний элемент списка
spisok.count(x)	Возвращает количество элементов со значением **x**
spisok.sort([reverse = True])	Сортирует элементы по возрастанию. Параметр **reverse** является необязательным и принимает логические значения. Если передать значение **True,** то список сортируется по убыванию
spisok.reverse()	Возвращает обратный список

Задача 1. В список, сгенерированный случайным образом, добавить введенный пользователем элемент.

Комментарий. Как видно из нижеприведенной программы, сначала происходит генерация списка, затем запрашивается число, и с помощью метода **append** происходит добавление числа в конец списка.

Листинг 58

```
import random
spisok=random.sample(range(100),10)
print(spisok)
chislo=int(input("Введите число   "))
spisok.append(chislo)
print(spisok)
```

Результат работы программы показан на рис. 78.

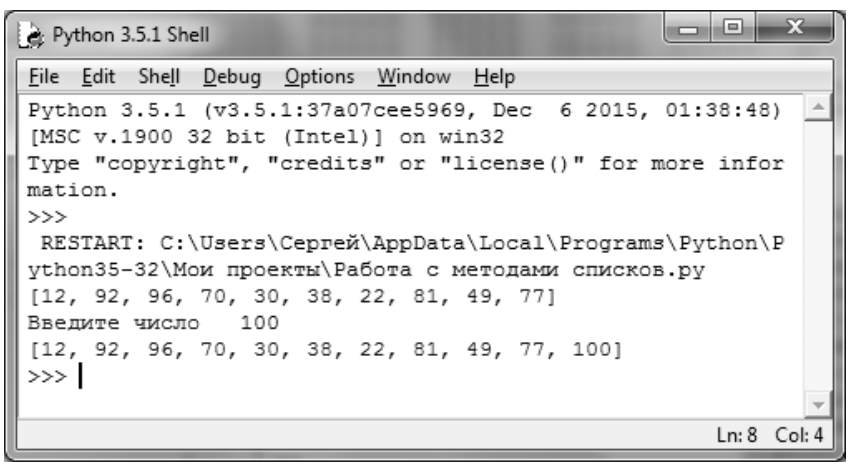


Рис. 78. Число 100 добавлено в конец списка

Задача 2. В список, сгенерированный случайным образом, добавить введенный пользователем элемент на указанную позицию.

Комментарий. В нижеприведенной программе (листинг 59) запрашиваются ввод числа и номер позиции, на которую следует добавить число. Метод **insert()** с соответствующими параметрами **poz, chislo** выполняет вставку элемента на указанную позицию.

Листинг 59

```
import random
spisok=random.sample(range(100),10)
print(spisok)
chislo=int(input("Введите число   "))
poz=int(input("На какую позицию в списке добавить элемент?   "))
spisok.insert(poz, chislo)
print(spisok)
```

Результат работы программы показан на рис. 79.

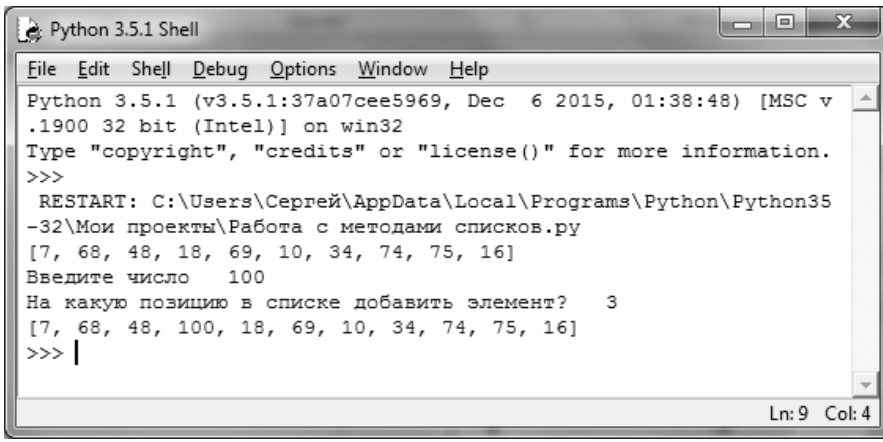


Рис. 79. Число 100 добавлено на третью позицию

Задача 3. Имеются два списка, сгенерированные случайным образом. Добавьте в конец первого списка все элементы второго списка.

Комментарий. Итак, у нас есть два списка **spisok1** и **spisok2,** сгенерированные случайным образом. За счет использования метода **extend(),** примененного к первому списку, мы расширяем его элементами второго списка.

Листинг 60

```
import random
spisok1=random.sample(range(100),10)
print(spisok1)
spisok2=random.sample(range(100),5)
print(spisok2)
spisok1.extend(spisok2)
print(spisok1)
```

Результат работы программы показан на рис. 80.

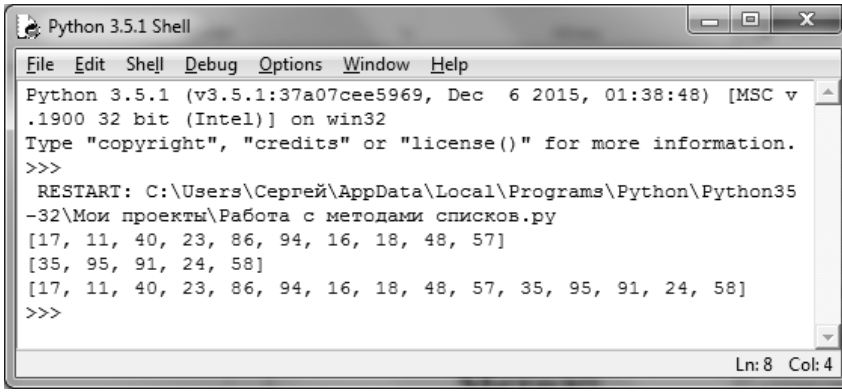


Рис. 80. Пять элементов второго списка добавлены к элементам первого списка

Задача 4. Из заранее сформированного списка следует удалить элемент, введенный пользователем.

Комментарий. Исходный список будет содержать пять элементов (листинг 61). Пользователь вводит значение элемента, подлежащего удалению. Метод **remove(),** примененный к списку, осуществляет указанные действия.

Листинг 61

```
spisok=[10, 2, 30, 4, 50]
for i in spisok:
        print(i, end="     ")
chislo=int(input("\nВведите число   "))
spisok.remove(chislo)
print(spisok)
```

Результат работы программы представлен на рис. 81.

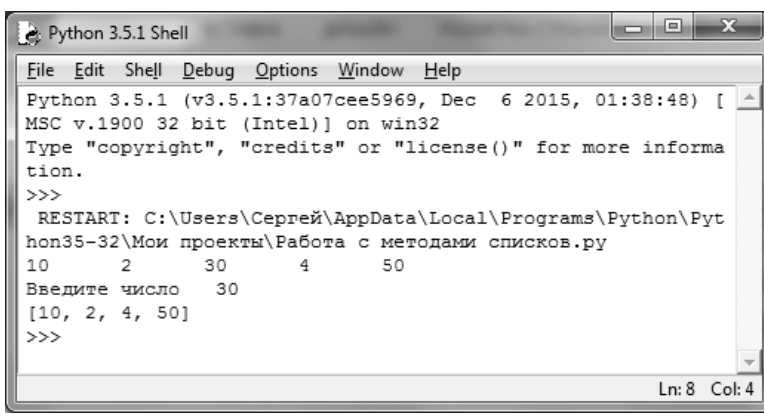


Рис. 81. Число 30, введенное пользователем, отсутствует в результирующем списке

Задача 5. Из исходного списка следует удалить элемент, позицию которого указал пользователь.

Комментарий. Исходный список состоит из пяти элементов. Пользователь вводит индекс элемента, подлежащего удалению. Если аргумент не передан в метод **pop(),** то возвращается и удаляется последний элемент списка. Однако в качестве параметра метода **pop()** мы указываем индекс элемента. Кроме результирующего списка, выводим на экран значение удаленного элемента.

Листинг 62

```
spisok=[10, 2, 30, 4, 50]
for i in spisok:
        print(i, end="      ")
ind=int(input("\nВведите индекс элемента   "))
chislo=spisok.pop(ind)
print(spisok)
print("\nУдаленный элемент =", chislo)
```

Результат работы программы показан на рис. 82.

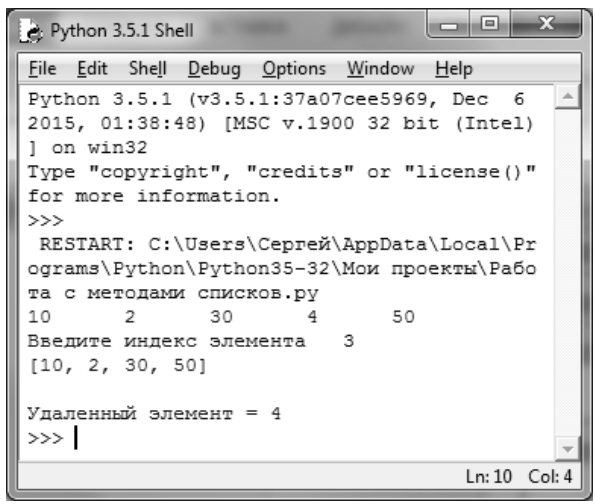


Рис. 82. Элемент с индексом 3 удален из исходного списка

Задача 6. В исходном списке подсчитайте количество элементов с определенным значением.

Комментарий. Мы сформировали список (листинг 63), включив в него несколько повторяющихся элементов. Введенное пользователем значение **znach** является параметром метода **count(),** который вернет количество повторов **znach.**

Листинг 63

```
spisok=[10, 2, 30, 4, 50, 50, 10, 40, 30, 50]
for i in spisok:
        print(i, end="      ")
```

```
znach=int(input("\nВведите значение элемента списка   "))
kol=spisok.count(znach)
print("\nКоличество элементов, имеющих такое значение = ", kol)
```

Результат работы программы показан на рис. 83.

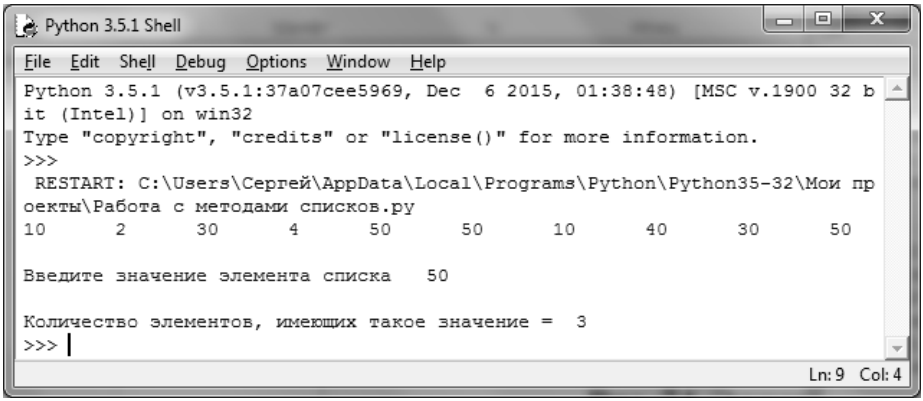


Рис. 83. Количество элементов, имеющих значение 50, равно трем

Задача 7. Список, сгенерированный случайным образом, отсортируйте по возрастанию.

Комментарий. Метод **sort(),** применяемый в Python для сортировки списка, имеет необязательный параметр **reverse.** В нижеприведенном коде (листинг 64) он выставлен в положение **False,** что дает возможность отсортировать список по возрастанию.

Листинг 64

```
import random
spisok=random.sample(range(100),10)
print(spisok)
spisok.sort(reverse = False)
print(spisok)
```

Результат работы программы показан на рис. 84.

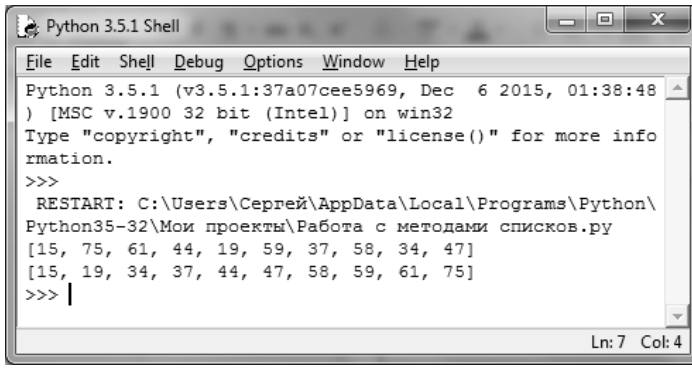


Рис. 84. Отсортированный по возрастанию список

Задача 8. Список, сгенерированный случайным образом, выведите в обратном порядке.

Комментарий. Метод **reverse** без параметров, примененный к исходному списку, сгенерированному случайным образом, возвращает обратный список.

Листинг 65

```
import random
spisok=random.sample(range(100),10)
print(spisok)
spisok.reverse()
print(spisok)
```

Результат работы программы показан на рис. 85.

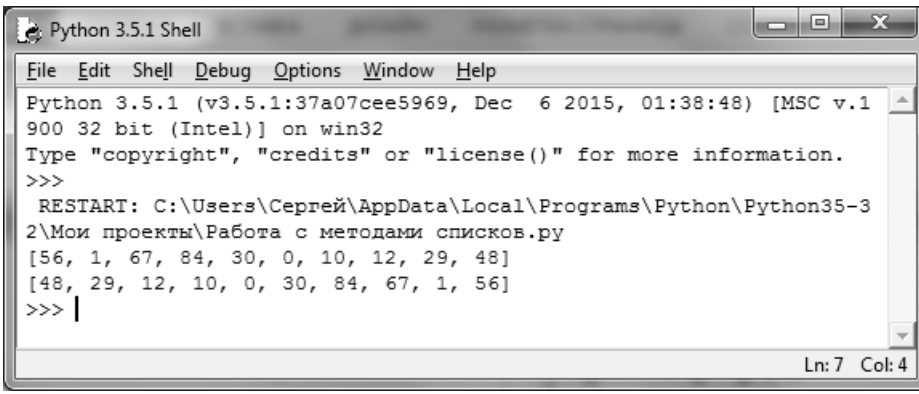


Рис. 85. Список, выведенный в обратном порядке

6.4. Работа со словарями

В Python есть исключительная возможность работать со структурами данных, которые имеют не только номера (индексы) элементов, но и строковые значения. Это несомненное удобство, поскольку часто программисту приходится работать с данными, которые имеют буквенно-цифровые обозначения. Обработка таких данных может быть использована, например, в информационных системах, применяемых в авиа- или железнодорожных компаниях.

Структура данных, позволяющая идентифицировать ее элементы не по числовому индексу, а по произвольному объекту (числу, строке, кортежу), в языке Python называется **словарем.** Некий произвольный объект является **ключом,** который и открывает доступ к набору объектов, содержащихся в словаре.

Синтаксис создания словаря может быть разным, приведем один из них:

Имя словаря ={**ключ:значение**,
ключ:значение,
ключ:значение,
.

.

ключ:значение}

Как видно, каждый элемент словаря состоит из пары «**ключ:значение**». Ключ определяет элемент словаря, а значение соответствует данным, которые соответствуют данному ключу. Каждая пара может записываться в программах на разных строках, а может располагаться в одной.

Смысл такой конструкции напоминает использование оператора выбора (Case, Select Case, switch), работа с которым ведется в таких популярных языках программирования, как C#, Microsoft Visual Basic, Delphi, Pascal и др., хотя в Python возможности использования словарей значительно шире. Словари можно использовать, когда необходимо установить соответствие между объектами, подсчитывать количество объектов, хранить данные, связанные с объектом.

При использовании словарей следует учитывать следующие правила:

1. Двух одинаковых ключей в парах «**ключ:значение**» быть не может.
2. Словари не относятся к последовательностям. Таким образом, те функции, которые использовались для работы со списками и кортежами, для словарей неприемлемы.
3. Ключ должен быть неизменяемым, им может быть целое или действительное число, строка, кортеж. Сделано это для того, чтобы добиться выполнения п. 1.
4. Значения могут принимать любой тип данных, быть изменяемыми и неизменяемыми.

Перейдем к рассмотрению примеров работы программ с использованием словаря.

Задача 1. Разработайте программу, которая осуществляет перевод нескольких слов русского языка на английский.

Комментарий. Создание словаря в данной программе (листинг 66) происходит иначе, чем рассказывалось выше. Сначала оператором slovar=**dict**() мы создаем пустой словарь с именем **slovar.** Далее мы создаем в нашем словаре пары, состоящие из ключей (русские слова) и значений (слова английского языка). Когда пользователь вводит слово, которое он хочет перевести на английский язык, организуется перебор элементов словаря, и если какой-то из них совпал с введенным словом, выводится ответ. Подобная ситуация показана на рис. 86.

Листинг 66

```
slovar=dict()
slovar['Мама']='Mother'
slovar['Папа']='Dad'
slovar['Бабушка']='Grandmother'
slovar['Дедушка']='Grandfather'
slovo=input("\nВведите слово для перевода   ")
if slovo in slovar:
        perevod=slovar[slovo]
        print("\n", slovo, " переводится как ", perevod)
```

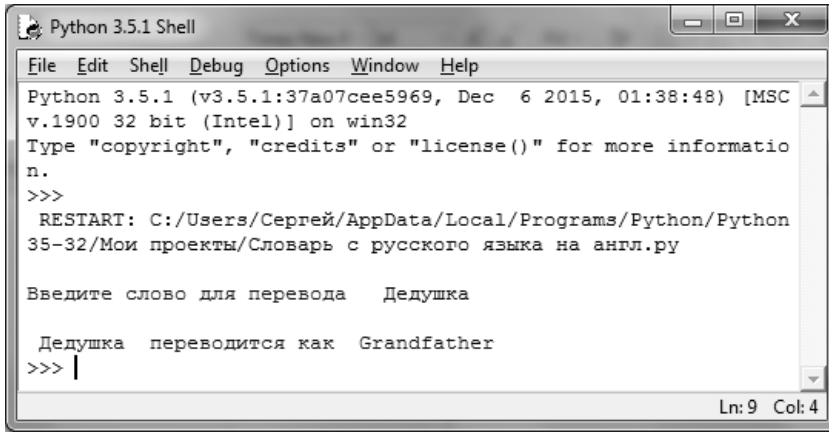


Рис. 86. Результат выбора элемента словаря

Давайте дополним разработанную программу возможностью включения в наш словарь нового слова. Для этого мы организуем приглашение, в котором попросим пользователя ввести новое слово для словаря, а также его перевод (листинг 67). В дальнейшем вы сможете проверить, действительно ли осуществляется перевод нового слова.

Листинг 67

```
slovar=dict()
slovar['Мама']='Mother'
slovar['Папа']='Dad'
slovar['Бабушка']='Grandmother'
slovar['Дедушка']='Grandfather'
slovo=input("\nВведите слово для перевода   ")
if slovo in slovar:
        perevod=slovar[slovo]
        print("\n", slovo, " переводится как ", perevod)
slovo=input("\nКакое слово вы хотите добавить для перевода?   ")
if slovo not in slovar:
        perevod=input("\nВведите его перевод  ")
        slovar[slovo]=perevod
        print("\nМы добавили ваше слово в словарь" )
slovo=input("\nДавайте проверим. Введите слово для перевода   ")
if slovo in slovar:
        perevod=slovar[slovo]
print("\n", slovo, " переводится как ", perevod)
```

Задача 2. Разработайте программу, которая выводит фразу «Привет, программист» на одном из иностранных языков.

Комментарий. При решении данной задачи мы создадим словарь из нескольких пар, в которых есть ключ – страна и значение, которым является приветствие. Все имеющиеся пары мы заключаем в фигурные скобки. Далее с по-

мощью условного оператора **if** проверяем, соответствует ли введенное название страны ключу в словаре и, если условие оказывается истинным, выводим соответствующее значение (листинг 68).

Листинг 68

```
slovar={"Россия":"Привет, программист!",
                "Англия":"Hello, programmer",
                "Германия":"Hallo, Programmierer",
                 "Испания":"Hola, programador",
                 "Италия": "Ciao, programmatori"}
slovo=input("\nВведите название страны   ")
if slovo in slovar:
          perevod=slovar[slovo]
          print("\n", slovo, " Приветствие звучит как ", perevod)
```

Результат работы программы представлен на рис. 87.

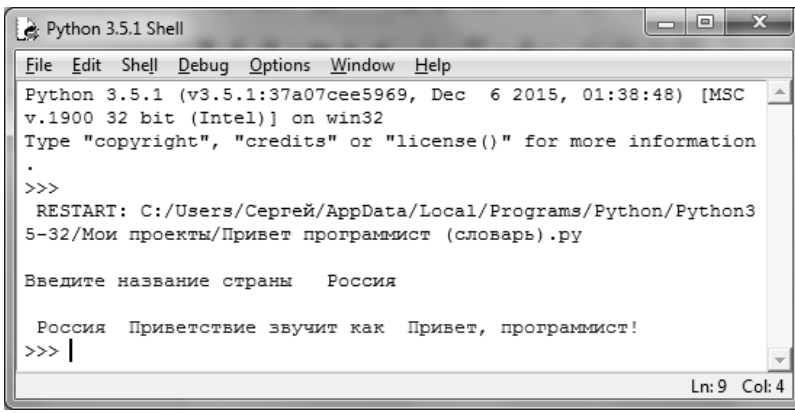


Рис. 87. Название страны «Россия» выбрано в словаре

Задача 3. Разработайте программу, которая имитирует заказ билета в кинотеатр. Ее результатом должен стать вывод информации о выбранном кинофильме, времени сеанса, зале, номере ряда и места.

Комментарий. Создадим три словаря. В первом из них, имеющем имя **film,** элементы словаря будут соответствовать названию фильма и времени сеанса. Предполагается, что в зависимости от времени сеанс будет проходить в определенном зале, поэтому во втором словаре (**time**) элементами будут номер зала и время начала сеанса. Третий словарь (**place**) необходим для хранения номера ряда и места.

После ввода названия фильма условием **if name in film** проверяем наличие фильма в словаре, и если он в нем содержится, то выводим доступное время сеанса для запрошенного фильма. Далее запрашиваем время сеанса, и если оно заявлено в словаре (**if vrem in time**), то относительно заданного времени будет выбран соответствующий зал, после чего пользователя попросят ввести номер ряда и место. Код программы представлен в листинге 69.

Листинг 69

```
film={"Кукла":"18:30  20:00  22:30",
     "Схватка":"9:30  15:00  16:15",
     "Экипаж":"17:45  22:00  00:30"}
time={"18:30":"Зал 3",
     "20:00":"Зал 2",
     "22:30":"Зал 1",
     "9:30":"Зал 3",
     "15:00":"Зал 2",
     "16:15":"Зал 1",
     "17:45":"Зал 3",
     "22:00":"Зал 2",
     "00:30":"Зал 1"}
place={"1":"1-15",
       "2":"1-15",
       "3":"1-15",
       "4":"1-15",
       "5":"1-15",
       "6":"1-15"}
print("Доступные фильмы: Кукла, Схватка, Экипаж")
name=input("\nВведите название фильма: ")
if name in film:
   print("Доступное время сеансов: ",film[name])
   vrem=input("Введите время сеанса: ")
   if vrem in time:
      zal=time[vrem]
      if zal=="Зал 1":
         ryad=input("\nВведите ряд от 1 до 6: ")
         if ryad in place:
            print("Доступные места: ",place[ryad])
            mesto=int(input("Введите место: "))
            if (mesto>0)and(mesto<16):
               print("\nВыбранный фильм: ",name,"\nВыбранное время:
",vrem,"\nЗал: ",zal,"\nРяд: ",ryad,"\nМесто: ",mesto)
            else:
               print("Вы ввели несуществующее место")
         else:
            print("Неверный ряд")
      elif zal=="Зал 2":
         ryad=input("\nВведите ряд от 1 до 6: ")
         if ryad in place:
            print("Доступные места: ",place[ryad])
            mesto=int(input("Введите место: "))
            if (mesto>0)and(mesto<16):
               print("\nВыбранный фильм: ",name,"\nВыбранное время:
```

```
",vrem,"\nЗал: ",zal,"\nРяд: ",ryad,"\nМесто: ",mesto)
            else:
                print("Вы ввели несуществующее место")
        else:
            print("Неверный ряд")
    elif zal=="Зал 3":
        ryad=input("\nВведите ряд от 1 до 6: ")
        if ryad in place:
            print("Доступные места: ",place[ryad])
            mesto=int(input("Введите место: "))
            if (mesto>0)and(mesto<16):
                print("\nВыбранный фильм: ",name,"\nВыбранное время:
",vrem,"\nЗал: ",zal,"\nРяд: ",ryad,"\nМесто: ",mesto)
            else:
                print("Вы ввели несуществующее место")
        else:
            print("Неверный ряд")
    else:
        print("Неверное время")
else:
    print("Некорректное название фильма")
```

Результат выполнения программы представлен на рис. 88.

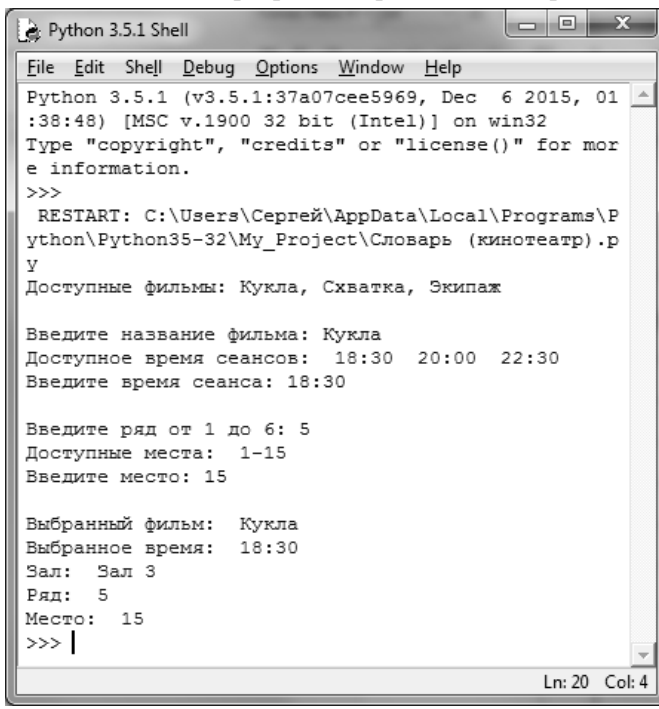


Рис. 88. Для выбранного фильма зарезервированы соответствующие зал, ряд и место

6.5. Примеры решения задач

Задача 1. В кортеже целых чисел вычислите произведение отрицательных элементов, имеющих нечетные индексы. Разработка алгоритма решения задачи представлена на рис. 89.

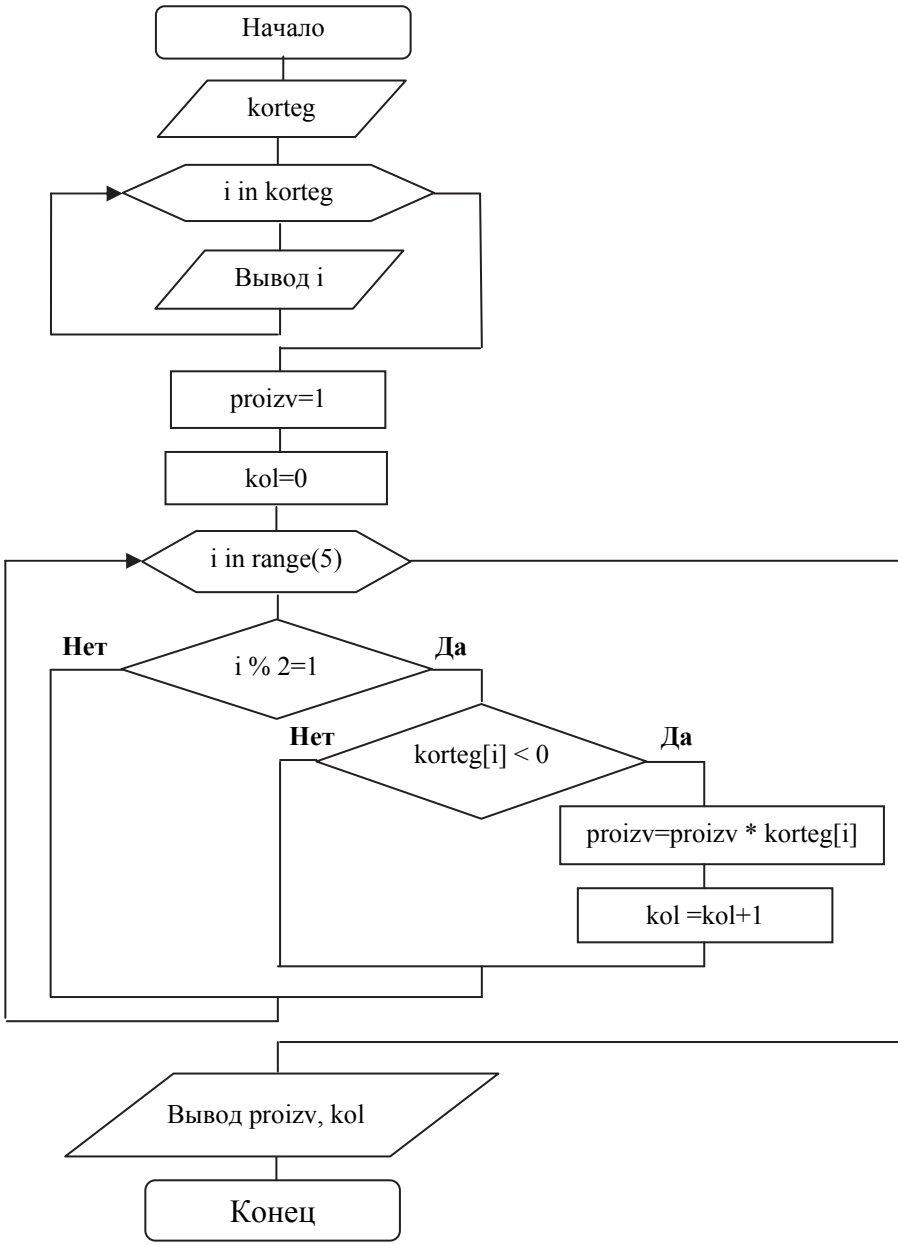


Рис. 89. Алгоритм решения задачи

В листинге 70 приведен код, отвечающий за решение задачи.

Листинг 70

```
korteg=(13, -2, -23, -14, -5)
print("\nКортеж")
for i in korteg:
        print(i, end=" ")
kol=0
proizv=1
for i in range(5):
        if i%2==1:  #Перебор элементов с нечетными индексами
            if korteg[i]<0:
                proizv = proizv*korteg[i] #Нахождение произведения
        # отрицательных элементов
                kol = kol+1 #Подсчет количества
print("\nПроизведение отрицательных элементов, имеющих нечетные
индексы в кортеже, равно ", proizv)
print("\nИх количество равно ", kol)
```

Задача 2. В кортеже целых чисел вычислите среднее арифметическое значение квадратов положительных элементов. Разработка алгоритма решения задачи представлена на рис. 90.

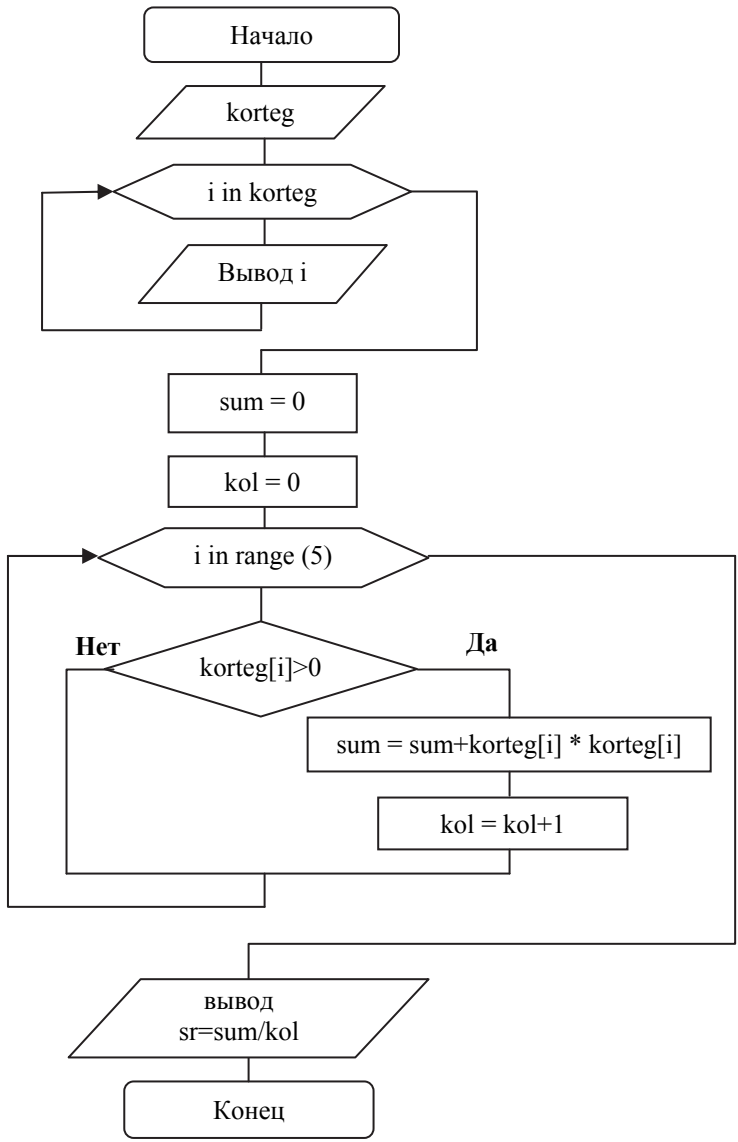


Рис. 90. Алгоритм решения задачи

В листинге 71 приведен код, отвечающий за решение задачи.

Листинг 71

```
korteg=(13, 2, -23, -14, -5)
print("\nКортеж")
```

```
for i in korteg:
   print(i, end=" ")
sum=0
kol=0
for i in range(5):
   if korteg[i]>0:
         sum=sum+korteg[i]*korteg[i] #Нахождение суммы квадратов
         kol=kol+1 #Подсчет количества
sr = sum/kol
print("\nСреднее арифметическое положительных элементов кортежа
равно ", sr)
```

Задача 3. В кортеже целых чисел определите число инверсий. Инверсией называется пара элементов, в которой большее число расположено слева от меньшего. Например, дан кортеж:

Стрелками показано количество инверсий. Разработка алгоритма решения задачи представлена на рис. 91.

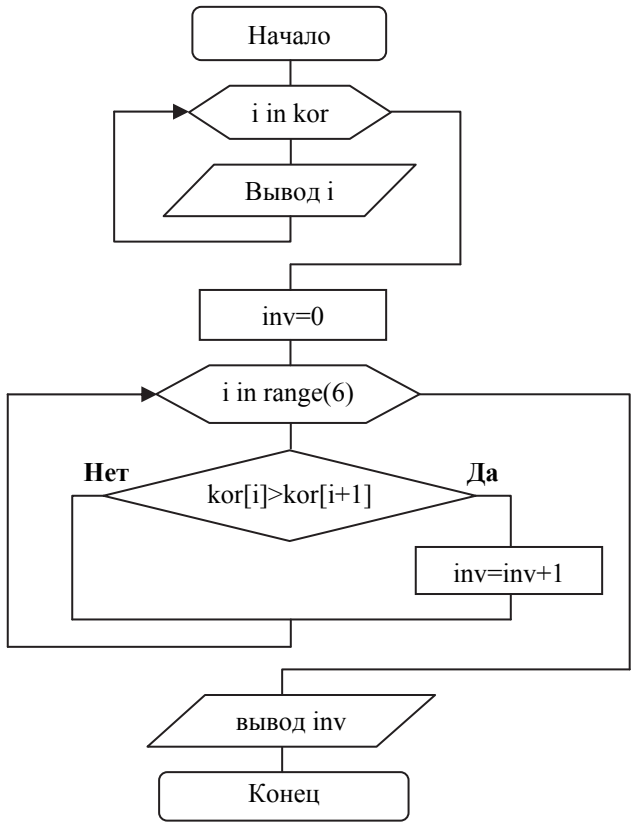


Рис. 91. Алгоритм решения задачи

Комментарий. Организовав цикл от 1 до 6, мы сравниваем каждый элемент последовательности с предыдущим и определяем, не больше ли он него. Нет смысла организовывать цикл от 1 до 7, потому что, выполнив проверку **kor[i]>kor[i+1],** мы седьмой элемент кортежа будем сравнивать с восьмым, а такого не существует. В случае истинности проверяемого условия увеличиваем счетчик на единицу оператором **inv=inv+1** и выводим ответ.

В листинге 72 приведен код, отвечающий за решение задачи.

Листинг 72

```
kor=(24, 35, 29, 44, 8, 22, 4)
print("\nКортеж")
for i in kor:
        print(i, end=" ")
inv=0
for i in range(6):
        if kor[i]>kor[i+1]:
            inv=inv+1
print("\nКоличество инверсий в кортеже равно ", inv)
```

Результат работы программы представлен на рис. 92.

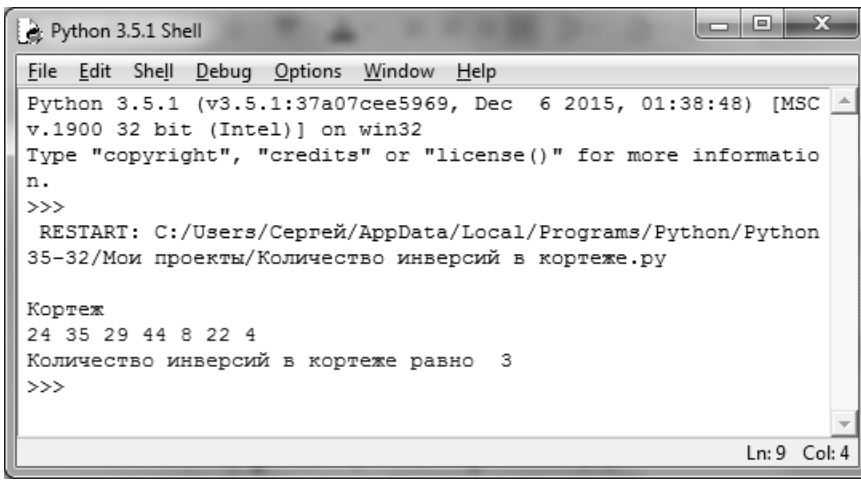
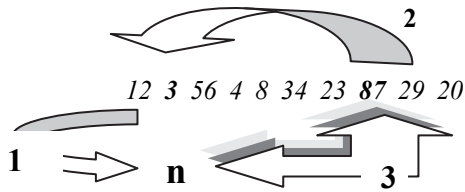


Рис. 92. Результат работы программы

Задача 4. В списке из десяти целых чисел, сформированном случайным образом, найдите максимальный и минимальный элементы, а также осуществите их обмен.

Комментарий. Для того чтобы обменять элементы списка местами, нам потребуются две ячейки **nommax** и **nommin.** Изначально считаем, что номер максимального элемента в списке равен нулю и номер минимального элемента также равен нулю. Отсюда и операторы в программе: **nommax=0** и **nommin=0.** Приемы нахождения максимального и минимального чисел рассматривались при программировании циклических алгоритмов, поэтому сейчас мы на них останавливаться не будем. Для того чтобы осуществить обмен ячеек, необходима третья ячейка – например, **n**. Предположим, сгенерирован список целых чисел:



Минимальный элемент в списке равен **3,** с порядковым номером **1,** а максимальный элемент в списке равен **87,** с порядковым номером **7.** Возьмем третью ячейку – **n. Первым действием** (**1**) будет перемещение в нее найденного минимального числа. **Вторым действием** (**2**) на освободившееся место перемещаем максимальное число. И, наконец, **третьим действием** (**3**) минимальное число из ячейки **n** помещается на то место в списке, где только что находилось максимальное число. В программе эти действия можно было бы описать операторами:

```
n=chislo[1]
chislo[1]=a[7]
a[7]=n
```

Поскольку заранее не известно, какой элемент в списке будет минимальный, а какой максимальный, и каковы будут их номера, такую последовательность операторов заменяем на следующую:

```
n=chislo[nommin]
chislo[nommin]=chislo[nommax]
chislo[nommax]=n
```

В листинге 73 приведен код программы, отвечающий за решение задачи.

Листинг 73

```
import random
chislo=random.sample(range(100),10)
print(chislo)
nommin = 0  #Считаем, что номер минимального элемента в списке
#равен нулю
nommax = 0  #Считаем, что номер максимального элемента в списке
# равен нулю
min = 32767
```

```
max = -32768
for i in range(1,10):
        if chislo[i]<min: #Поиск минимального элемента в списке
            min = chislo[i]
            nommin = i  #Нахождение его номера
        if chislo[i]  max:   #Поиск максимального элемента в списке
            max = chislo[i]
            nommax = i       #Нахождение его номера
print("Минимальное число =" , min)
print("Максимальное число =" , max)
n = chislo[nommin]  #Обмен элементов списка
chislo[nommin] = chislo[nommax]
chislo[nommax] = n
for i in range(0,10):
  print(chislo[i], end='\t')
```

Результат работы программы представлен на рис. 85.

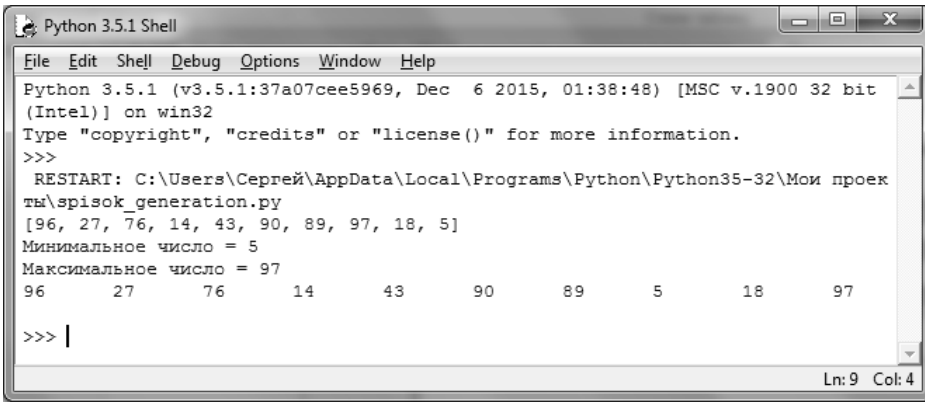


Рис. 93. Минимальное число обменяли с максимальным

Задача 5. Сформируйте кортеж, состоящий из вещественных чисел. Запишите положительные элементы кортежа в список. Определите количество положительных элементов. Найдите произведение элементов списка.

Комментарий. Первоначально сформированный кортеж содержит семь вещественных чисел, среди которых четыре положительных. Пустой список объявляется с помощью оператора **spisok=[].** Оператором **if kort[i]>0** проверим условие на положительность числа, входящего в кортеж, и, если оно истинно, то с помощью метода **append()** добавляем его в список. Счетчик количества положительных чисел **n** увеличивается на единицу оператором **n=n+1.**

Далее организуется цикл с уже известным количеством положительных чисел, хранящихся в ячейке **n,** и в нем находим произведение элементов списка, которое, в качестве ответа, выводим на экран. Код программы приведен в листинге 74, а ее результат представлен на рис. 94.

Листинг 74

```
kort=(5,7,-8,6,-4.5,7.8,-9)
spisok=[]
n=0
proiz=1
print('Кортеж: ')
for i in kort:
   print(i,end="  ")
for i in range (7):
   if kort[i]>0:
      spisok.append(kort[i])
      n=n+1
for i in range(n):
   proiz=proiz*spisok[i]
print('\nСписок: ',kort)
print('\nКоличество положительных элементов равно ', n)
print('\nПроизведение элементов списка ', proiz)
```

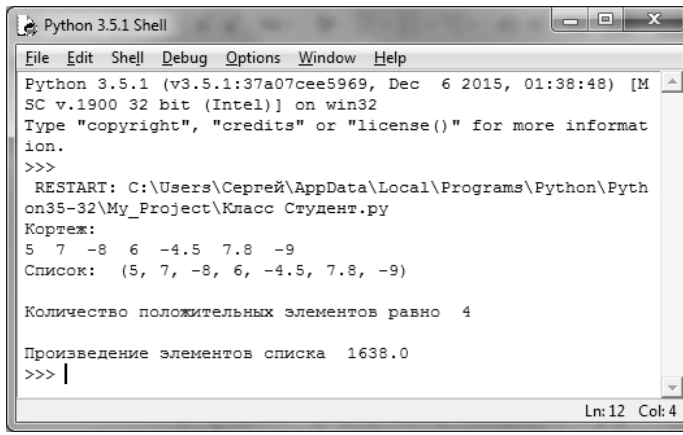


Рис. 94. Результат работы программы

Задача 6. Разработайте программу, заполняющую список из **N** элементов случайными целыми числами разного знака. Выведите на экран компьютера созданный список и сформируйте другой список, элементы которого по модулю больше некоторого введенного значения **C**.

Комментарий. Первоначально сформируем последовательность чисел с использованием функции **sample,** которая возвращает указанное пользователем количество элементов (листинг 75). Затем, после ввода значения **c,** инициализируем список **y,** открываем цикл, в котором проверяем логическое выражение **abs(x[i])>c** и в случае его истинности, используя метод **append(),** добавляем элемент исходной последовательности в список **y.**

Листинг 75

```
import random
from math import *
n=int(input("Введите количество элементов списка: "))
x=random.sample(range(-5,15),n)
for i in x:
   print(i, end=" ")
c=int(input("\nВведите значение переменной C: "))
y=[]
for i in range (n):
   if abs(x[i])>c:
      y.append(x[i])
print("\nЭлементы, превышающие значение C ", y)
```

Результат работы программы представлен на рис. 95.

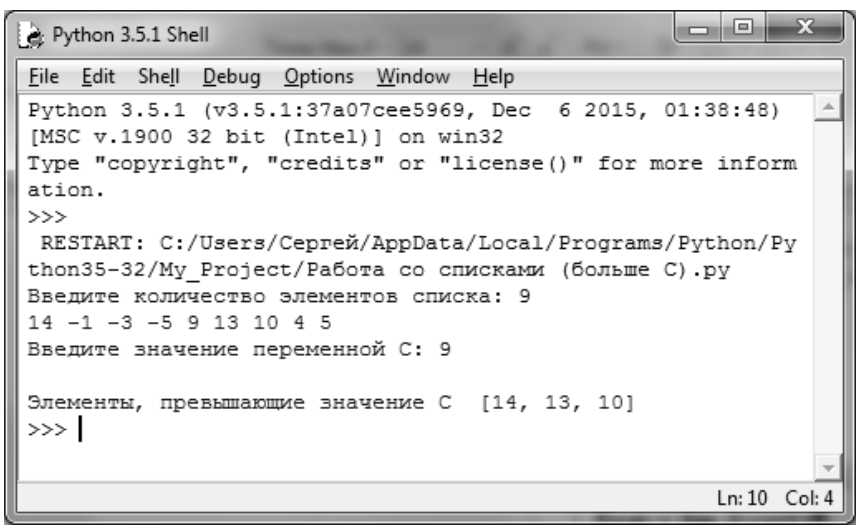


Рис. 95. Результат работы программы

Задача 7. Сформируйте исходный список имен **N** студентов. Организуйте формирование двух результирующих списков по четным и нечетным номерам исходного. Список с нечетными номерами упорядочите по убыванию, с четными номерами – по возрастанию.

Комментарий. Создав три пустых списка (**spisok**, **chetn**, **nechetn**), организуем ввод имен студентов и добавление их методом **append()** в исходный список. В следующем цикле осуществляем проверку номера элемента на четность с использованием операции **%** (деление по модулю). Если остаток равен нулю, то элемент добавляется в список четных номеров, в противном случае формируется список нечетных номеров. Используя методы упорядочивания списков (см. табл. 6), выводим их на экран.

Листинг 76

```
n=int(input('Введите количество студентов '))
spisok=[]
chetn=[]
nechetn=[]
for i in range (n):
   sp=input("Введите имя студента ")
   spisok.append(sp)
print('Список студентов ', spisok)
for i in range(n):
   if i%2==0:
      ch=spisok[i]
      chetn.append(ch)
   else:
      nech=spisok[i]
      nechetn.append(nech)
print("Четные номера списка: ",chetn)
print("Нечетные номера списка ", nechetn)
for i in chetn:
   chetn.sort()
print("Упорядоченный список студентов четных номеров", chetn)
for i in nechetn:
   nechetn.sort(reverse=True)
print('Упорядоченный список студентов нечетных номеров', nechetn)
```

Результат работы программы представлен на рис. 96.

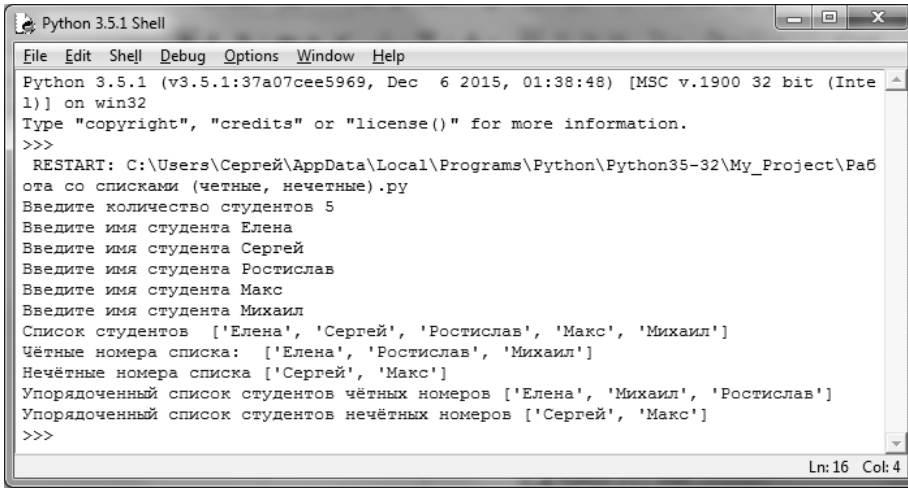


Рис. 96. Выведены списки студентов с четными и нечетными номерами

Контрольные вопросы

1. В чем состоит главная особенность кортежей?
2. Каковы преимущества кортежей с точки зрения их использования в программах?
3. Напишите синтаксис объявления кортежей.
4. Каким образом осуществляется доступ к каждому элементу кортежа при его обработке?
5. Перечислите классические способы обработки кортежей.
6. Каким образом можно реализовать в программе срез кортежа?
7. Поясните, каким образом осуществляется обмен значений элементов кортежа.
8. Поясните, в чем состоит отличие списков, созданных на языке Python, от кортежей.
9. Напишите синтаксис объявления списков.
10. Какие возможности языка Python используются для генерации списков?
11. Перечислите и поясните основные методы работы со списками.
12. Дайте определение такой структуры данных языка Python, как словарь.
13. Напишите синтаксис создания словаря.
14. Какие правила следует использовать при создании словаря?

Задачи для самостоятельного решения

1. В кортеже целых чисел вычислите произведение отрицательных элементов, имеющих нечетные индексы.
2. Из исходного списка целых чисел сформируйте два списка: список четных чисел B и список нечетных чисел C.

3. Определите среднее арифметическое элементов кортежа, удовлетворяющих условию abs(korteg[i])>C. Значение C вводится с клавиатуры.
4. Разработайте программу, в которой определяются максимальный и минимальный элементы среди положительных нечетных элементов целочисленного кортежа A(10).
5. Разработайте программу, заполняющую список из N элементов случайными целыми числами, находящимися в интервале от –20 до 40. Выведите на экран компьютера созданный список. В списке положительные элементы уменьшите вдвое, а отрицательные замените на значения их индексов.
6. Разработайте программу, заполняющую список из N элементов случайными целыми числами, находящимися в интервале от 1 до 40. Определите, сколько процентов всего количества элементов списка составляют нечетные элементы.
7. Разработайте программу, заполняющую список из N элементов случайными целыми числами, находящимися в интервале от 1 до 50. Выведите на экран компьютера созданный список и упорядочите элементы данного списка по возрастанию их значений.
8. Из списка произвольных чисел A[10] сформируйте другой список таким образом, чтобы вначале шли отрицательные элементы исходного списка, затем положительные и, наконец, нулевые.
9. Разработайте программу, заполняющую список из N элементов случайными целыми числами, находящимися в интервале от 1 до 30. Выведите на экран компьютера созданный список и найдите максимальный элемент, его номер и поменяйте местами максимальный и первый элемент списка.
10. Разработайте программу, которая включает заданное число D в список A[10], упорядоченный по возрастанию, с сохранением упорядоченности.
11. Разработайте программу, в которой удалите из списка A, состоящего из n элементов, первые четыре нулевых элемента.
12. Разработайте программу, заполняющую список из N элементов случайными целыми числами, находящимися в интервале от –30 до 50. Выведите на экран компьютера созданный список и определите, есть ли в нем серии элементов, состоящих из знакочередующихся чисел. Если есть, то выведите на экран количество таких серий.
13. Разработайте программу, которая выводит на экран два кортежа A(10), содержащих диаметры и веса шин. Следует отобрать две шины, диаметры которых отличаются не более чем на D см, а вес – не более чем на W килограмм.
14. Из списка произвольных чисел A[10] сформируйте два списка, содержащих номера положительных и отрицательных элементов.
15. Разработайте программу, заполняющую список из N элементов случайными целыми числами, находящимися в интервале от –20 до 40. Выведите на экран компьютера созданный список и вычислите среднее арифметическое значение квадратов положительных элементов.
16. В кортеже целых чисел произведите обмен соседних элементов, стоящих на четных местах, с элементами, стоящими на нечетных местах.

17. Разработайте программу, заполняющую список из N элементов случайными вещественными числами, находящимися в интервале от 1 до 30. Все элементы списка с четными номерами, предшествующие первому по порядку элементу с наибольшим значением, домножьте на него.
18. Разработайте программу, заполняющую список из N элементов случайными целыми числами, находящимися в интервале от –15 до 20. Выведите на экран компьютера созданный список и найдите наибольший элемент из отрицательных.
19. Разработайте программу, заполняющую список из N элементов случайными целыми числами, находящимися в интервале от 1 до 50. Выведите на экран компьютера созданный список и найдите количество тех элементов, значения которых находятся в диапазоне от A до B.
20. Пользователь вводит с клавиатуры элементы списка A[n]. Определите, является ли заданная последовательность чисел a_1 , a_2, ... , a_N монотонно убывающей.
21. Разработайте программу, заполняющую список из N элементов случайными целыми числами, находящимися в интервале от 1 до 30. Замените нулями элементы между максимальным и минимальным значениями, кроме них самих.
22. Разработайте программу, в которой в кортеже целых чисел требуется найти индекс последнего по счету отрицательного элемента.
23. Разработайте программу, которая определяет, имеется ли в заданном целочисленном кортеже A(10) хотя бы одна пара совпадающих по значению чисел.
24. Разработайте программу, которая выводит на экран два кортежа, содержащих кортежи ростов игроков двух команд (в см), и определяет, имеется ли в данных командах игроки одинакового роста.
25. Разработайте программу, заполняющую список из N элементов случайными целыми числами, находящимися в интервале от 1 до 50. Выведите на экран компьютера созданный список и найдите максимальный и минимальный элементы, вычислите их разность.

7. РАБОТА СО СТРОКАМИ

7.1. Основные понятия

Следует отметить, что первоначальные навыки работы со строками в языке программирования Python у нас уже сформированы. Мы знаем, как сделать приглашение к вводу данных или вывести ответ в программе, умеем осуществлять перевод строки, делать табуляцию, форматировать строки. В этой главе мы познакомимся с функциями и методами, предназначенными для работы со строками.

Строка в Python – это объект класса **str,** поэтому ранее, для того чтобы работать с числовыми данными, мы применяли функции приведения типов, например, **int,** как это продемонстрировано в следующем примере:

ind=**int**(**input**("\nВведите индекс элемента "))

На рис. 97 показано, что строки индексируются с нуля, т. е., если имеется оператор **stroka="python",** то первый символ в строке нулевой и обращение к нему будут записываться как **stroka[0].** В то же время к элементам строки в Python может обращаться, указывая отрицательные индексы, например, оператор **print(stroka[-6])** есть ничто иное, как вывод символа **p** на экран.

0	1	2	3	4	5
p	y	t	h	o	n
-6	-5	-4	-3	-2	-1

Рис. 97. Индексация строки

Известно, что в программировании, прежде чем сравнить один символ с другим, его следует преобразовать в число с помощью таблицы **ASCII** (American Standard Code for Information Interchange – Американский стандартный код для обмена информацией), который поддерживает кодирование 128 буквенно-цифровых символов.

Первые 32 кода базовой таблицы, начиная с нулевого, отданы разработчикам аппаратных средств (в первую очередь производителям компьютеров и печатающих устройств). В этой области размещаются так называемые **управляющие коды,** которым не соответствуют никакие символы языков, и соответственно, эти коды не выводятся ни на экран, ни на устройства печати, но используются для функций управления (например, возврата каретки или возврата на один символ).

Национальные стандарты кодировочных таблиц включают международную часть кодовой таблицы без изменений, а во второй половине содержат коды национальных алфавитов, символы псевдографики и некоторые математические знаки.

Начиная с кода 32 по код 127 размещены коды символов английского алфавита, знаков препинания, цифр, арифметических действий и некоторых вспомогательных символов. Базовая таблица кодировки ASCII приведена в табл. 7.

Таблица 7. **Базовая таблица кодировки ASCII**

sp 32	! 33	" 34	# 35	$ 36	% 37	& 38	' 39	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	_ 95
` 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	| 124	} 125	~ 126	

Таким образом, при работе со строками важно знать, что символ "**A**" – это совсем не то же самое, что символ "**a**". Длина строки ограничена лишь объемом оперативной памяти компьютера, и для обращения к элементу строки достаточно указать его индекс в квадратных скобках. Подобным образом ранее мы обращались к элементу списка.

Строки в Python представляют собой неизменяемую последовательность и обрабатываются с использованием цикла с оператором **for.** Важно понять, что изменить символ строки, обратившись к нему по индексу, невозможно.

```
stroka="Python"
stroka[0]="p" #недопустимый оператор
```

При обработке строки к ней можно обратиться непосредственно в цикле, как это показано в листинге 77.

Листинг 77

```
stroka="python"
for i in stroka:
    print(i, end="  ")
```

Результатом работы этого кода станет вывод на экран слова **python**.

7.2. Функции для работы с символами

Рассмотрим основные функции, предназначенные для работы со строками.

1. **Функция len()**

При работе со строками может оказаться полезной функция **len(),** назначение которой заключается в определении длины строки. В операторе **print** мы обращаемся к каждому элементу строки, упоминая имя строки и заключая индекс каждого элемента в квадратные скобки. Естественно, результатом нижеследующего кода станет вывод на экран слова **python.**

Листинг 78

```
stroka="Python"
for i in range(len(stroka)):
   print(stroka[i], end="  ")
```

2. **Функция ord**()

Другая функция – **ord**(), синтаксис которой **ord("Символ")** возвращает код указанного символа. Например, функция **ord(),** которую мы применили к каждому символу строки в нижеследующем примере, возвратит ASCII-код символов строки "Python". Результат работы программы показан на рис. 98.

Листинг 79

```
stroka="Python"
for i in range(len(stroka)):
   print(ord(stroka[i]), end="  ")
```

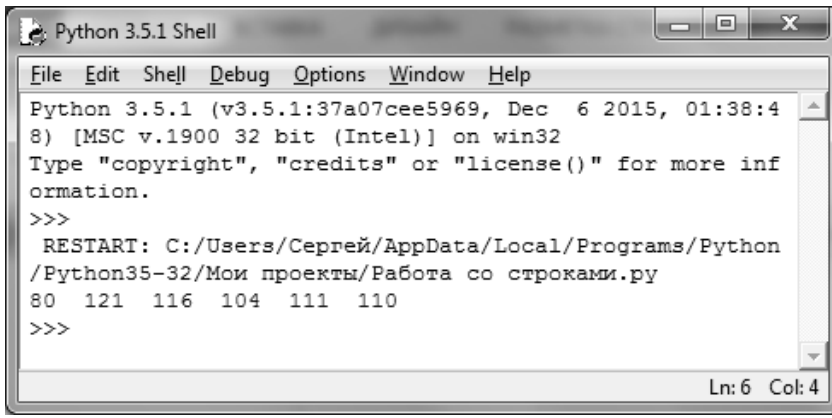


Рис. 98. Выведены ASCII-коды символов

3. **Функция chr().**

Действие функции, синтаксис которой **chr(Число),** прямо противоположно функции **ord(),** а именно, по ASCII-коду символа вернуть символ. Таким образом, функция **chr(),** которую применили к функции **ord(),** которая, в свою очередь, возвращала ASCII-коды символов, вернет исходную строку "Python" в нижеприведенном коде.

Листинг 80

```
stroka="Python"
for i in range(len(stroka)):
   print(chr(ord(stroka[i])), end="  ")
```

7.3. Методы работы со строками

Рассмотрим основные методы работы со строками.

1. **Метод upper()**.

Синтаксис метода: **stroka.upper().** Преобразует все символы строки к верхнему регистру. Например,

```
stroka="Python"
newstr=stroka.upper()
print(newstr)
```

2. **Метод lower().**

Синтаксис метода: **stroka.lower().** Преобразует все символы строки к нижнему регистру. Например,

```
stroka="PYTHON"
newstr=stroka.lower()
print(newstr)
```

3. **Метод swapcase().**

Синтаксис метода: **stroka.swapcase().** Преобразует все символы строки, записанные в нижнем регистре – в верхний и наоборот (рис. 99). Например,

```
stroka="PyThoN"
newstr=stroka.swapcase()
print(newstr)
```

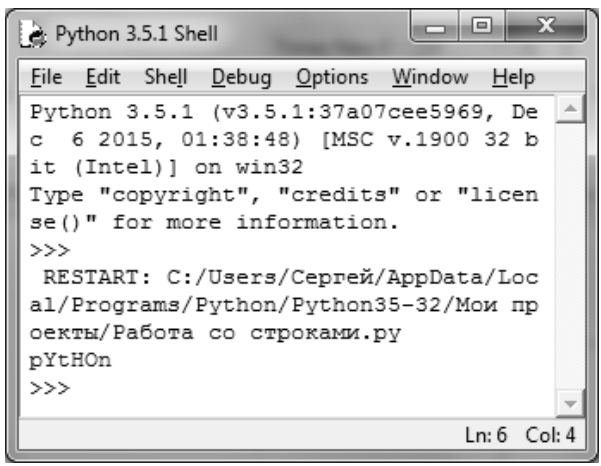


Рис. 99. Результат работы программы

4. **Метод capitalize().**

Синтаксис метода: **stroka.capitalize().** Преобразует первую букву в строке в верхний регистр. Например,

```
stroka="python"
newstr=stroka.capitalize()
print(newstr)
```

5. **Метод title().**

Синтаксис метода: **stroka.title().** Преобразует все первые буквы в строке в верхний регистр (рис. 100). Например,

```
stroka="язык программирования python"
newstr=stroka.title()
print(newstr)
```

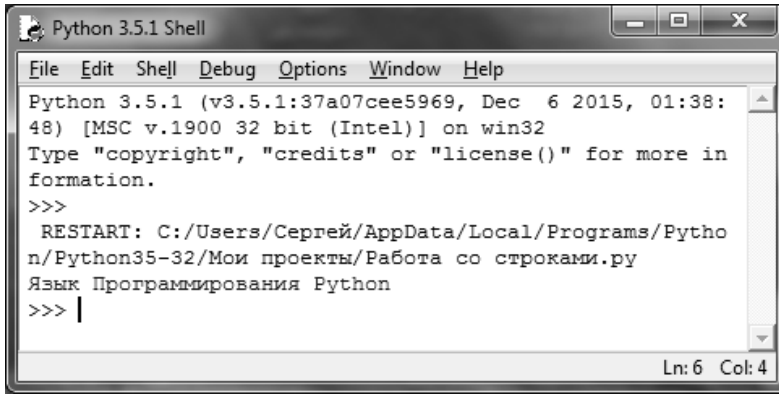


Рис. 100. Результат работы программы

6. **Метод startswith().**

Синтаксис метода: **stroka.startswith(podstroka).** Проверяет, начинается ли строка **stroka** с указанной подстроки **podstroka.** Например, в данном примере проверяется условие наличия подстроки "Язы" в начале строки "Язык программирования Python".

```
stroka="Язык программирования Python"
podstr="Язы"
n=stroka.startswith(podstr)
print(n)
```

Результатом данного кода будет значение **True.**

7. **Метод endswith().**

Синтаксис метода: **stroka.endswith (podstroka).** Проверяет, заканчивается ли строка **stroka** указанной подстрокой **podstroka.** Например, в данном примере проверяется условие наличия подстроки "thon" в начале строки "Язык программирования Python".

```
stroka="Язык программирования Python"
podstr="thon"
n=stroka.endswith(podstr)
print(n)
```

Результатом данного кода будет значение **True.**

8. **Метод replace().**

Синтаксис метода: **stroka.replace**(old, new), где old – подстрока для замены, new – новая подстрока. Метод находит и заменяет в строке **stroka** подстроку **old** подстрокой **new.** Например, в данном примере оператор stroka = stroka.**Replace** ("еще","вот") возвратит строку "Эх раз, вот раз, вот много, много раз".

```
stroka= "Эх раз, еще раз, еще много, много раз"
stroka=stroka.replace("еще","вот")
print(stroka)
```

9. **Метод rfind().**

Синтаксис метода: **stroka.rfind**(podstr), где **podstr** – подстрока. Метод возвращает позицию последнего вхождения подстроки в строку. Если подстрока не обнаружена, то возвращается значение **–1.** После параметра **podstr** могут быть указаны начальная позиция и конечная позиции в строке, где следует искать подстроку. Эти параметры необязательны, и если отсутствует начальная позиция, то поиск будет осуществлен с начала строки. Так, в нижеприведенном примере, последнее вхождение подстроки "еще" будет зафиксировано на позиции 17. Строка, напомним, индексируется с нуля, пробелы и знаки препинания являются символами.

```
stroka= "Эх раз, еще раз, еще много, много раз"
stroka=stroka.rfind("еще")
print(stroka)
```

10. **Метод find().**

Синтаксис метода: **stroka.find**(podstr), где **podstr** – подстрока. В отличие от предыдущего метода, метод **find()** возвращает номер позиции, с которого начинается вхождение подстроки в строку. Если подстрока не обнаружена, то возвращается значение **–1.** Соответственно, если изменить предыдущую программу и вместо метода **rfind()** использовать метод **find(),** то первое вхождение подстроки "еще" в строку "Эх раз, еще раз, еще много, много раз" будет начинаться на позиции 8.

11. **Метод count().**

Синтаксис метода: **stroka.count**(podstr), где **podstr** – подстрока. Метод возвращает количество вхождений подстроки **podstr** в строку **stroka.** Таким образом, в нижеследующем примере ответом, выведенным на экран оператором **print,** будет число 2, поскольку подстрока "еще" входит два раза в строку "Эх раз, еще раз, еще много, много раз".

```
stroka= "Эх раз, еще раз, еще много, много раз"
stroka=stroka.count("еще")
print(stroka)
```

12. **Метод strip()**.

Синтаксис метода: **stroka.strip()**. Метод удаляет начальные и конечные пробелы в строке **stroka.** В нижеприведенном примере в исходной строке присутствуют пробелы в начале и в конце строки. На рис. 101 представлен результат выполнения программы, где пробелы с помощью метода **strip()** удалены из исходной строки.

```
stroka= "    Эх раз, еще раз, еще много, много раз    "
stroka=stroka.strip()
print(stroka)
```

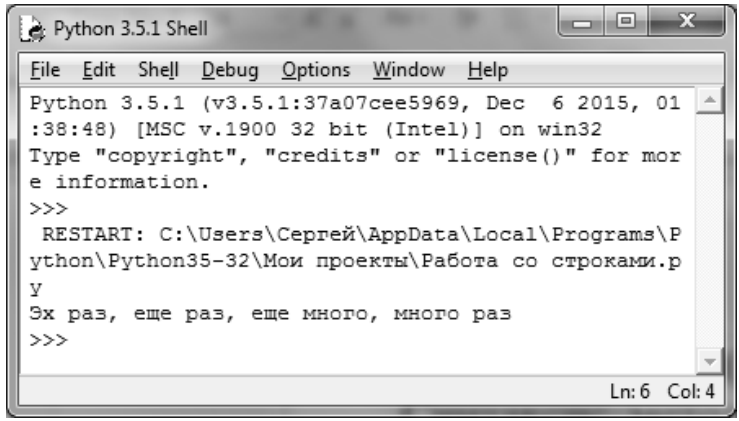


Рис. 101. В исходной строке слева и справа удалены пробельные символы

13. **Методы lstrip() и rstrip().**

Их синтаксис и действие аналогичны рассмотренному методу **strip(),** с той разницей, что метод **lstrip()** удаляет символы пробела слева от начала исходной строки, а метод **rstrip()** – в конце исходной строки.

Отметим, что с помощью методов **strip(), rstrip()** и **lstrip**() можно удалять не только пробелы. Так, если в качестве параметра одного из методов указать символ или последовательность символов, то произойдет его (их) удаление. Например, в нижеследующем примере мы указали в качестве параметра в методе **strip()** символы "Эх". Соответственно, метод возвратит исходную строку, но уже без указанных символов. Подобная ситуация показана на рис. 102.

```
stroka= "Эх раз, еще раз, еще много, много раз"
stroka=stroka.strip("Эх")
print(stroka)
```

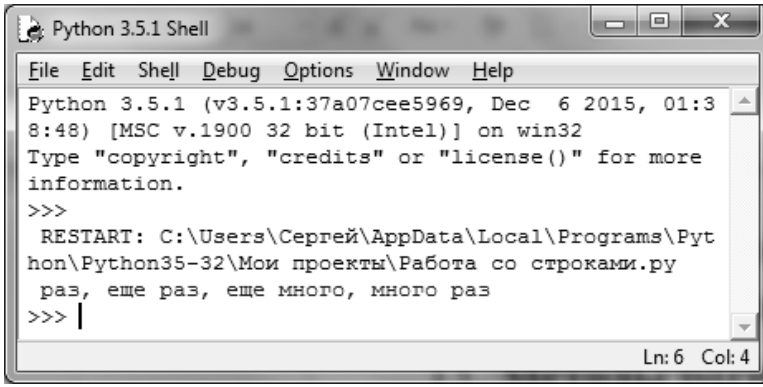


Рис. 102. Результат работы программы

14. **Метод split().**

Синтаксис метода: **stroka.split().** Метод разделяет строку на подстроки и добавляет их в список. Если в качестве параметра присутствует разделитель, то строка будет разбита на подстроки в соответствии с указанным разделителем. В

случае его отсутствия разделителем считается пробел. В коде, представленном в листинге 81, исходная строка методом **split()** преобразована в список, разделителем служит пробел. Далее уже над списком выполняется списочный метод **remove(),** который удаляет тот элемент списка, который вводит пользователь с клавиатуры.

Листинг 81

```
stroka= "Эх раз, еще раз, еще много, много раз"
spisok=stroka.split(" ")
print(spisok)
element=input("\nВведите элемент списка, который хотите удалить   ")
for i in spisok:
        if element in spisok:
                spisok.remove(element)
print(spisok)
```

Результат работы программы представлен на рис. 103. Следует обратить внимание на то, что при формировании списка разделителем был пробел. Соответственно, в исходном списке элемент списка '**раз,**' входит в него два раза, а элемент списка '**раз**', входит в него один раз. При запросе удаляемого элемента, мы ввели '**раз,**', а не '**раз**', поэтому в результирующем списке слово '**раз**' присутствует.

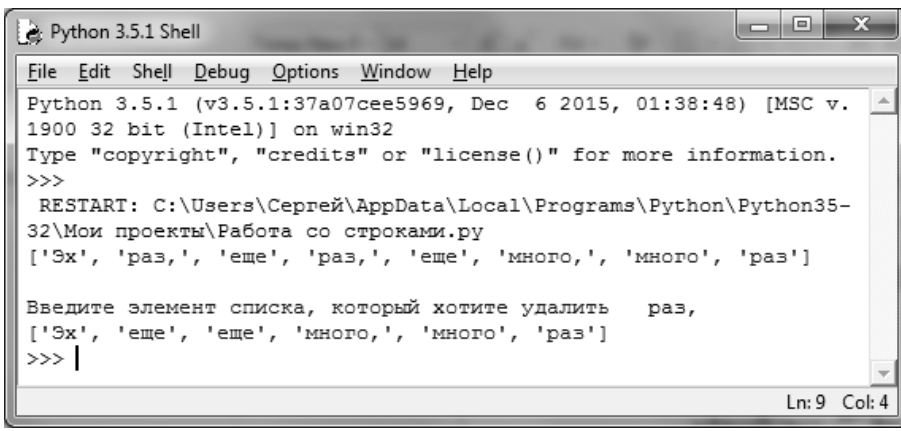


Рис. 103. Удалены два элемента списка 'раз,'

15. **Метод join().**

Синтаксис метода: **разделитель.join(spisok),** где **spisok** – список или кортеж. Еще одним способом преобразования строки в список является использование функции **list(),** а обратное преобразование осуществляется методом **join().** Такие преобразования необходимы для того, чтобы изменить элемент строки по индексу, поскольку изначально строки неизменяемы.

В нижеприведенном примере (листинг 82) из исходной строки с помощью функции **list()** происходит формирование списка. Попробуем заменить элемент

исходного списка на введенный с клавиатуры и для этого осуществляем соответствующие запросы с помощью функции **input().** Вычисляем длину списка с помощью функции **len()** и организуем просмотр списка с помощью оператора цикла **for.** Если очередной элемент списка будет равен введенному элементу, то произойдет замена исходного элемента списка **element** на элемент **element1.** Затем, используя метод **join(),** преобразуем список в строку и выводим ее на экран (рис. 104).

Листинг 82

```
stroka= "Эх раз, еще раз, еще много, много раз"
spisok=list(stroka)
print(spisok)
element=input("\nВведите элемент списка, который хотите заменить   ")
element1=input("\nВведите элемент, на который хотите заменить   ")
n=len(spisok)
for i in range(0,n):
        if spisok[i]==element:
            spisok[i]=element1
stroka1="".join(spisok)
print(stroka1)
```

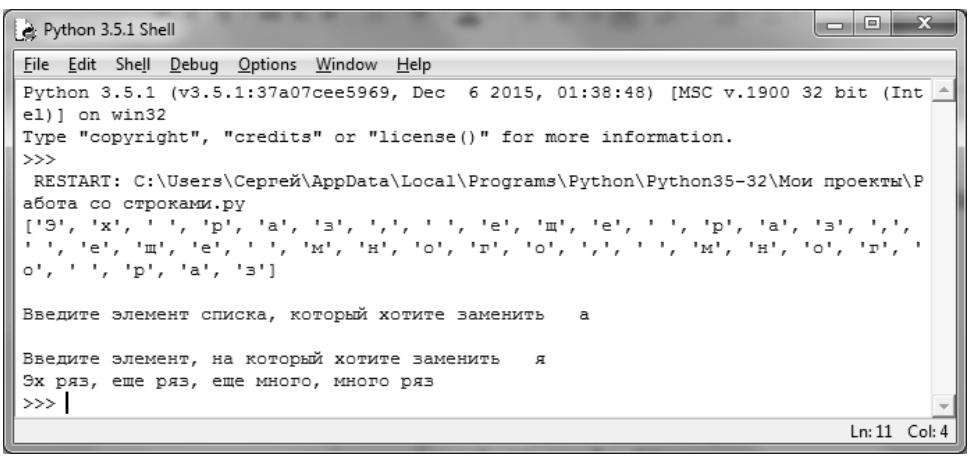


Рис. 104. Замена элемента списка 'а' на 'я'

7.4. Базовые алгоритмы обработки строк

Вышеприведенные примеры с использованием функций и методов познакомили вас с возможностями обработки строк и символов в Python. При решении задач следует научиться использовать в совокупности простые приемы и методы обработки строк (назовем их базовыми алгоритмами), среди которых можно выделить следующие:

1. Определение количества символов.
2. Замена символов в строке.
3. Удаление символа в строке.
4. Вставка символа в строку.
5. Анализ символа на принадлежность к группе.
6. Обращение строки.
7. Алфавитная выборка.
8. Срезы строк.

Данные базовые алгоритмы могут найти свое применение не только для решения учебных задач, но и для разработки простых текстовых редакторов, программного обеспечения предназначенного для верстки текста и т. д.

Определение количества символов

Задача. Пользователь вводит исходную строку **stroka** и символ **simvol.** Подсчитайте, сколько раз заданный символ встречается в исходной строке.

Комментарий. Первоначально после обнуления ячейки **k,** которая будет играть роль счетчика символов, мы преобразуем все символы исходной строки в строчные буквы методом **lower().** Затем в цикле происходит сравнение текущего символа строки **spisok[i]** с искомым символом **simvol.** Параметр цикла **i** будет изменяться от 0 (начало строки) до конца строки (за это отвечает ячейка **n,** в которой уже находится значение функции **len(spisok)**). Подсчет количества найденных символов в строке обеспечивает оператор **k+=1.**

Ниже приведен код программы, отвечающий за решение задачи.

Листинг 83

```
k=0
stroka=input("\nВведите строку   ")
stroka1=stroka.lower()
spisok=list(stroka1)
simvol=input("\nВведите символ, количество которого вы хотите найти
")
n=len(spisok)
for i in range(0,n):
   if spisok[i]==simvol:
      k+=1
print("\nКоличество символов равно", k)
```

Результат выполнения программы представлен на рис. 105.

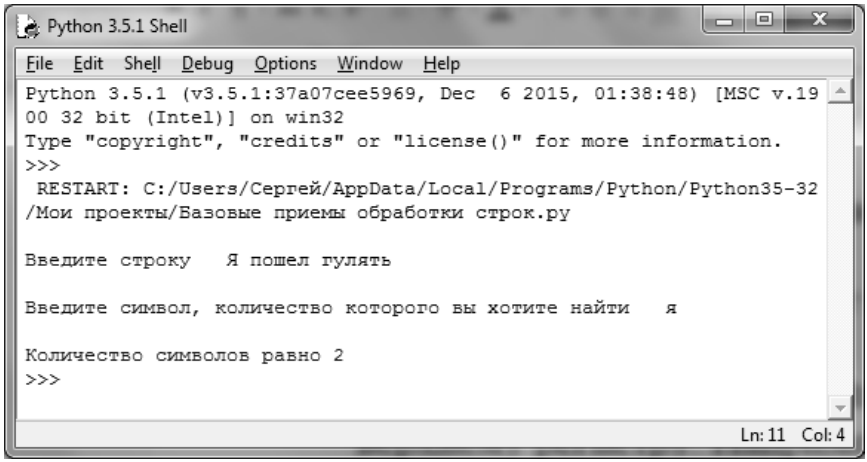


Рис.105. Количество символов **я** в исходной строке равно двум

Замена символов в строке

Задача. Пользователь вводит исходную строку **row** и символ **simv.** Замените пробелы в исходной строке указанным символом.

Комментарий. Ранее при объяснении работы метода **join()** была решена подобная задача. В листинге 84 приведен код, в котором обрабатываемая строка не фиксирована, а вводится с клавиатуры. Методы ее обработки остались прежними. Заметим, что подобную задачу можно решить и с применением метода **replace().**

Листинг 84

```
stroka=input("\nВведите строку   ")
stroka1=stroka.lower()
spisok=list(stroka1)
element=input("\nВведите элемент, который хотите заменить   ")
element1=input("\nВведите элемент, на который хотите заменить   ")
n=len(stroka1)
for i in range(0,n):
   if spisok[i]==element:
      spisok[i]=element1
stroka1="".join(spisok)
print(stroka1)
```

Удаление символов в строке

Задача. Пользователь вводит исходную строку **row** и символ **simv.** Удалите в исходной строке указанный пользователем символ.

Комментарий. Способ основан на применении метода **replace().** Осуществляем ввод символа (листинг 85), который хотим удалить, а затем используем его в качестве параметра в методе **replace().** Второй параметр в методе делаем равным пустым кавычкам. Результат работы программы показан на рис. 106.

Листинг 85

```
stroka=input("\nВведите строку   ")
element=input("\nВведите элемент, который хотите удалить   ")
stroka=stroka.replace(element,"")
print(stroka)
```

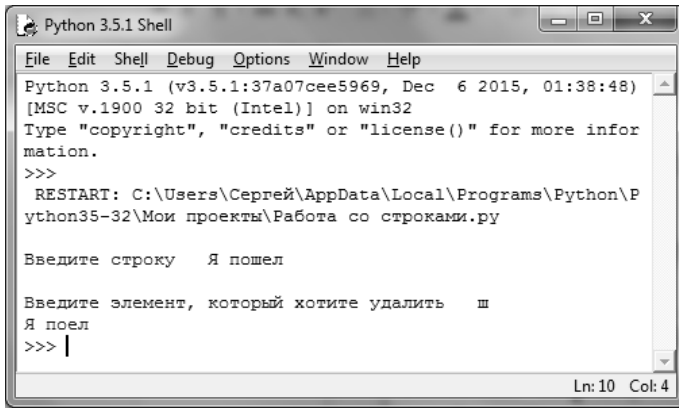


Рис. 106. Символ **ш** удален из исходной строки

Вставка символа в строку

Задача. Пользователь вводит исходную строку **row.** В исходную строку необходимо добавить символ **simvol1** после символа **simvol,** который пользователь вводит с клавиатуры.

Комментарий. Вставка подстроки в заданную позицию осуществляется с помощью оператора **spisok.insert(i+1,element1),** предварительно мы преобразовали строку в список с помощью функции **list.** Изменение параметра **i** в цикле с оператором **for** обеспечивает продвижение по строке. Если очередной элемент списка равен найденному элементу (**if** spisok[i]==element), то применяется метод **insert.** В конце программы, используя метод **join,** делаем обратное преобразование: список превращается в строку, которая и выводится на экран. Напоминаем, что при работе со строками надо учитывать, в каком состоянии находится символ – в верхнем или нижнем регистре.

В листинге 86 приведен код программы, отвечающий за решение задачи.

Листинг 86

```
stroka=input("\nВведите строку   ")
print(stroka)
spisok=list(stroka)
element=input("\nВведите символ после которого вы хотите вставить
новый символ   ")
element1=input("\nВведите символ который вы хотите вставить   ")
n=len(stroka)
for i in range(0,n):
   if spisok[i]==element:
```

```
        spisok.insert(i+1,element1)
stroka="".join(spisok)
print(stroka)
```

Результат выполнения программы представлен на рис. 107.

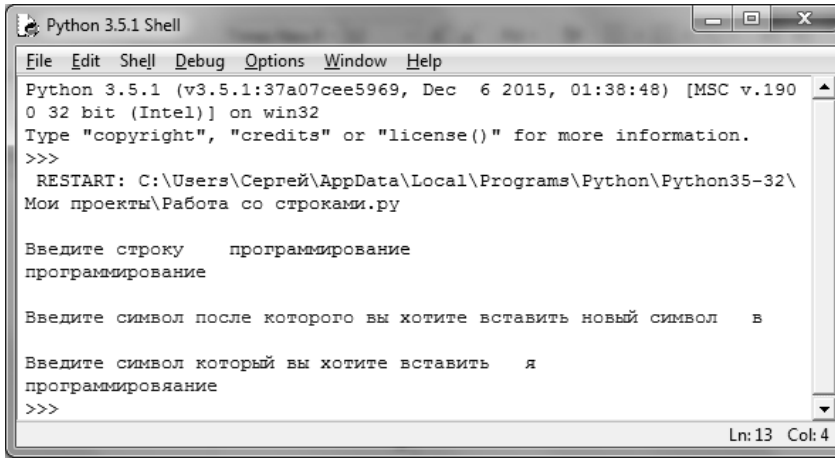


Рис. 107. В исходную строку после заявленного символа "в" добавляется символ "я"

Анализ символа на принадлежность к группе

Задача. Пользователь вводит исходную строку **row.** Определите, сколько в ней символов **simv_1** и **simv_2,** которые пользователь вводит с клавиатуры.

Комментарий. В данной программе, преобразовав строку символов в список, вводим два символа, количество которых мы хотим обнаружить в исходной строке. В цикле с оператором **for** сравниваем каждый элемент списка с введенными пользователем символами, и если условие истинно, счетчик увеличивается на единицу. Ниже приведен код программы, отвечающий за решение задачи.

Листинг 87

```
stroka=input("\nВведите строку    ")
print(stroka)
spisok=list(stroka)
k=0
element=input("\nВведите символ, количество которого вы хотите найти
")
element1=input("\nВведите второй символ, количество которого вы хо-
тите найти  ")
n=len(stroka)
for i in range(0,n):
   if (spisok[i]==element) or (spisok[i]==element1):
      k+=1
print("\nВсего найденных символов:  ", k)
```

Результат выполнения программы представлен на рис. 108.

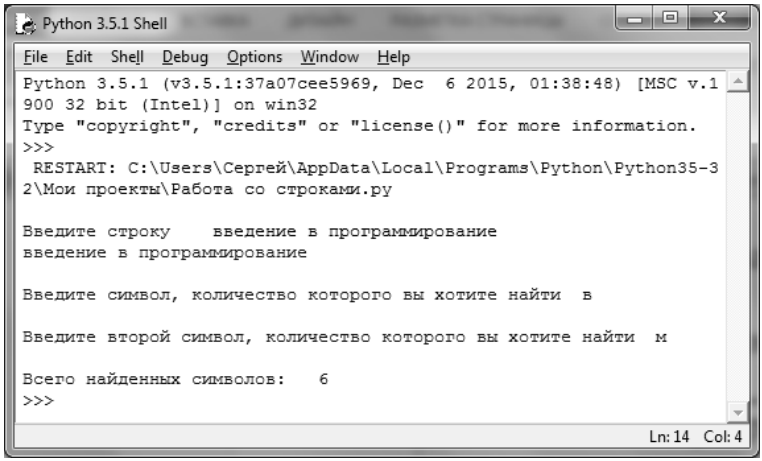


Рис.108. В исходной строке осуществлен подсчет символов "в" и "м"

Обращение строки

Задача. Пользователь вводит строку **stroka.** Переверните ее (т. е. запишите наоборот).

Комментарий. При решении данной задачи можно было бы воспользоваться методом **reverse** для обработки списков, но алгоритм работы данной программы основан на поочередной выборке символов из заданной строки и перемещение их в начало новой строки. Вспомогательная строка **tmp** изначально пуста. С помощью организации цикла просматриваем символы исходной строки от первого к последнему. Каждый из них присоединяется к началу уже накопленной строки, оператором **tmp=stroka[i]+tmp.** В листинге 88 приведен код, отвечающий за решение задачи.

Листинг 88

```
stroka=input("\nВведите строку    ")
print(stroka)
tmp=""
n=len(stroka)
for i in range(0,n):
    tmp = stroka[i] + tmp
print(tmp)
```

Результат выполнения программы представлен на рис. 109.

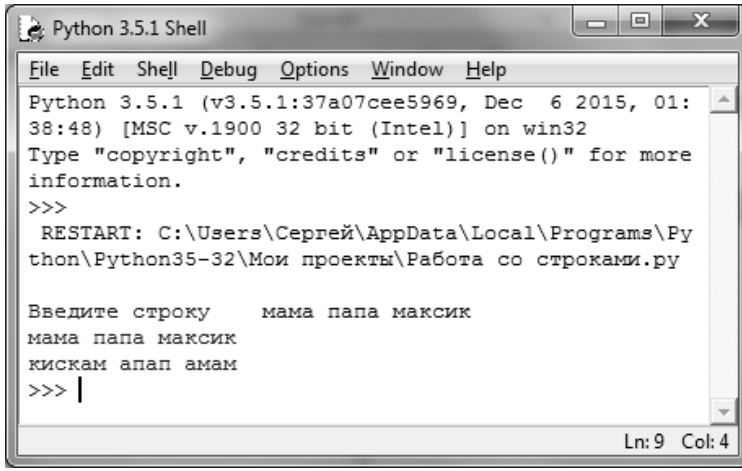


Рис. 109. Обращенная строка

Алфавитная выборка

Задача. Имеется заранее заданный алфавит. Пользователь вводит строку **stroka.** Выберите из строки символы, используемые в алфавите, и выведите их на экран.

Комментарий. При решении данной задачи мы познакомимся с таким понятием, как «константа» в Python. Переменная, набранная прописными буквами, в Python называется **константой.** Существует особенность применения констант в изучаемом нами языке программирования: в отличие от некоторых языков программирования константы в Python можно изменять. Поэтому, создав константу, пользователь должен контролировать ее неизменность.

Итак, мы имеем некий набор символов, условно называемый алфавитом, который хранится в константе **ALFAVIT.** Пользователь вводит строку, и для каждого символа строки (**for** letter **in** stroka:) мы осуществляем проверку условия. Если символ есть в алфавите (**if** letter **in** ALFAVIT:), то в переменной **tmp** накапливаем результирующие символы оператором **tmp+=letter.**

В листинге 89 приведен код, отвечающий за решение задачи.

Листинг 89

```
ALFAVIT="абвгдежзиклмн"
stroka=input("\nВведите строку    ")
print(stroka)
tmp=""
for letter in stroka:
   if letter in ALFAVIT:
      tmp += letter
print(tmp)
```

Результат выполнения программы представлен на рис. 110.

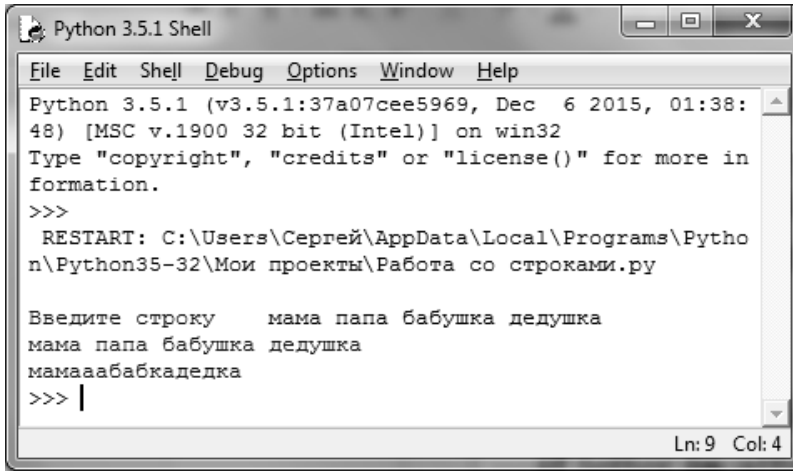


Рис. 110. Выполнена алфавитная выборка

Срезы строк

Задача. Из исходной строки получите символы, расположенные между заданными начальным и конечным значениями.

Комментарий. При изучении темы «Работа с кортежами», мы уже познакомились с понятием «срез» и говорили о том, что срез кортежа получается в результате вывода элементов кортежа, находящихся между заранее заданными начальной (a) и конечной (b) позициями элементов. Механизм работы со срезами строк очень похож на принципы работы со срезами кортежей. Применяя срезы к строкам, мы можем выбрать из них любой символ или последовательность расположенных подряд символов. Для этого нам понадобятся начальная и конечная позиции, которые будут обозначать границы среза. Код, отвечающий за правильность работы программы, представлен в листинге 90.

Листинг 90

```
stroka=input("\nВведите строку. Для выхода нажмите Enter, не вводя значение среза     ")
print(stroka)
flag=None
while flag !="":
        flag=input("\nВведите начало среза, для выхода нажмите Enter   ")
        if flag:
            flag=int(flag)
            kon=int(input("\nВведите конец среза   "))
            print(stroka[flag:kon])
input("\nНажмите Enter, чтобы выйти из программы  ")
```

Результат выполнения программы представлен на рис. 111.

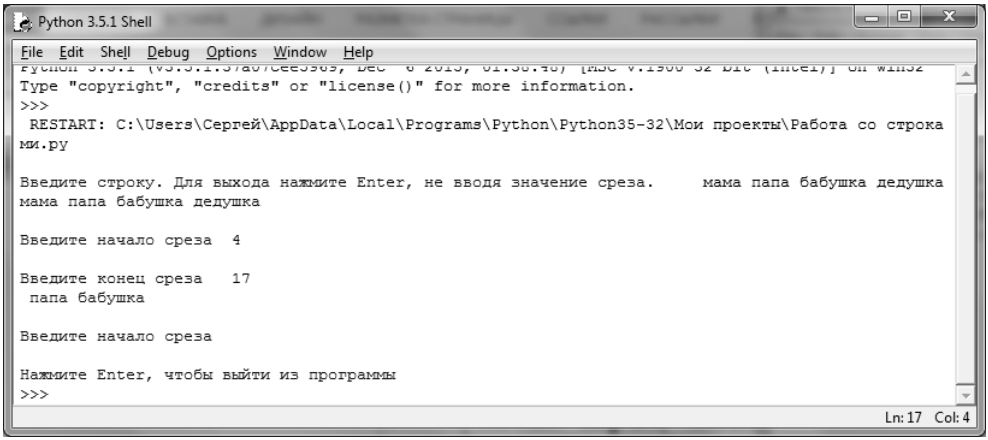


Рис. 111. Срез с начальной и конечной позицией stroka[4:17]

7.5. Примеры решения задач

Задача 1. Пользователь вводит строку **row,** содержащую несколько слов, между которыми один или несколько пробелов. Требуется найти количество символов в самом длинном слове.

Комментарий. В цикле с оператором **for** мы проверяем наличие пробела в строке, указывающее на переход к очередному слову. В переменной **kol** будет храниться количество символов в слове. Соответственно, для того чтобы найти максимальную длину слова, мы вводим в программу такую переменную, как **maxim,** предварительно присвоив ей маленькое значение (например, –10).

Таким образом, если значение ячейки **kol** превышает значение ячейки **maxim,** то находится самое длинное слово в строке при текущем проходе цикла. Следовательно, на выходе из цикла в ячейке **maxim** будет храниться количество символов в самом длинном слове строки.

В листинге 91 приведен код программы, отвечающий за решение задачи.

Листинг 91

```
stroka=input("Введите строку\n ")
kol=0
maxim=-10
for i in range(len(stroka)):
   if stroka[i]!=" ":
      kol=kol+1
      if kol>maxim:
         maxim=kol
   else:
      kol=0
print("\nВ самом длинном слове ", maxim, "символов")
```

Результат выполнения программы представлен на рис. 112.

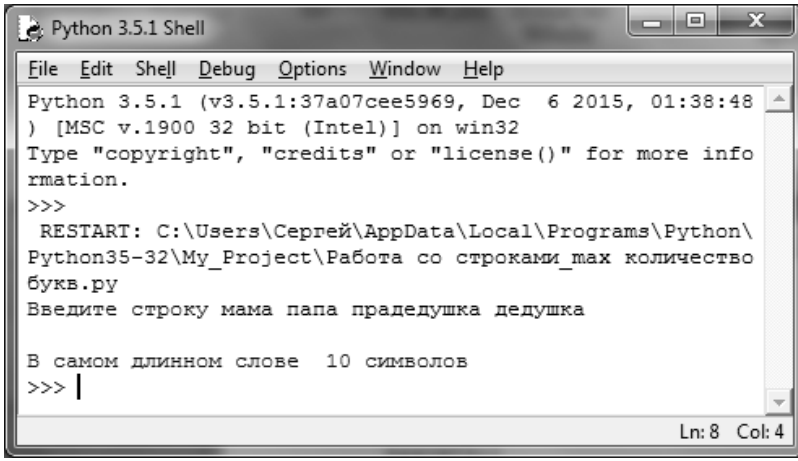


Рис. 112. Самое длинное слово в строке «прадедушка» содержит 10 символов

Задача 2. Пользователь вводит строку **row,** содержащую несколько слов, разделенных запятой, но пробел после запятой отсутствует. Результатом работы программы должна стать исправленная строка.

Комментарий. В переменной **k** хранится длина строки. В цикле с оператором **while** начинаем перебирать символы строки. В качестве условия поиска проверяем такую ситуацию, когда очередной символ строки равен символу «запятая», и при этом символ после запятой не равен пробелу. Как только условие становится истинным, вставляем пробел. Ниже приведен код программы, отвечающий за решение задачи.

Листинг 92

```
row=input("\nВведите строку  ")
k=len(row)
probel=" "
i=1
stroka=row[0]
while i<k:
   stroka = stroka + row[i]
   if((row[i]==",") and (not(row[i+1]==' '))):
      stroka=stroka + probel
   i=i+1
print(stroka)
```

На рис. 113 приведен результат выполненной программы

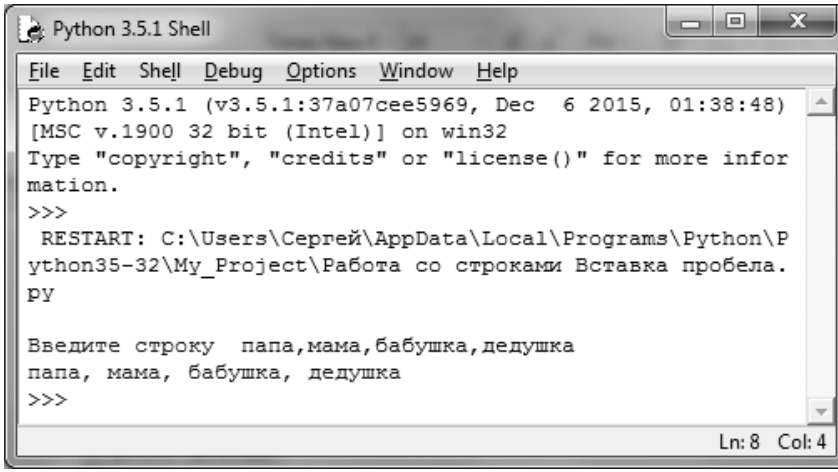


Рис. 113. В исправленной строке после запятой следует пробел

Задача 3. Пользователь вводит строку **row,** содержащую несколько слов и один из символов **simv,** содержащихся в строке. Следует удвоить символ и вывести результирующую строку.

Комментарий. После ввода исходной строки осуществляем ее преобразование в список символов, используя функцию **list.** Оператором **zamena=simv+simv** удваиваем введенный символ **simv.** В цикле с оператором **for** выполним проверку равенства введенного символа символам исходной строки. Если условие будет истинным, такой символ найден, и можно его удвоить, выполнив оператор **spisok[i]=zamena.** Далее осуществляется преобразование списка в строку методом **join()** и вывод ее на экран.

В листинге 93 приведен код программы, отвечающий за решение задачи.

Листинг 93

```
row=input("Введите строку\n")
spisok=list(row)
simv=input("Введите символ из строки для его удваивания\n")
zamena=simv+simv
n=len(spisok) #Определяем длину списка
for i in range (0,n):
   if spisok[i]==simv:
      spisok[i]=zamena
rowzam="".join(spisok)
print("\nСтрока с заданной удвоенной буквой\n",rowzam)
```

Результат выполнения программы представлен на рис. 114.

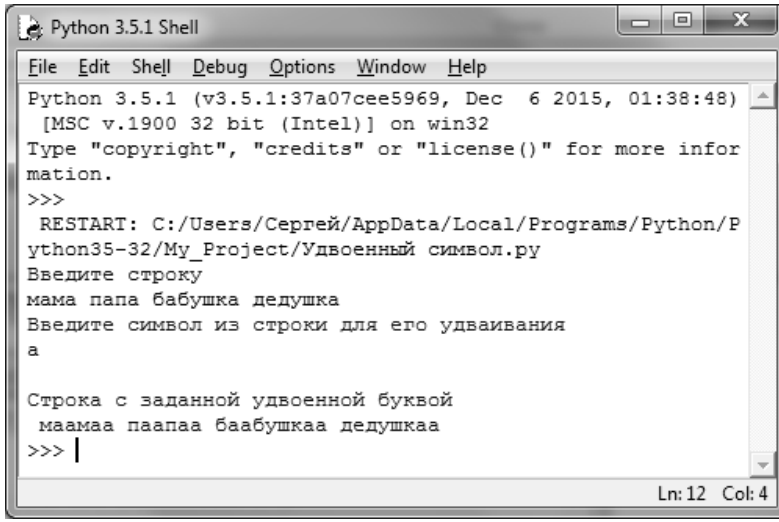


Рис.114. В результирующей строке удвоен символ **а**

Задача 4. Разработайте программу, проверяющую степень надежности пароля пользователя, при этом критерии сложности пароля следующие:

- ненадежный: длина менее 6 символов, либо длина от 6 до 8 символов, и пароль состоит только из цифр или букв;
- средний: длина от 6 до 8 символов, должен включать цифры, строчные и прописные символы, или иметь длину более 8 символов, но пароль при этом не содержит либо прописных букв, либо строчных, либо цифр;
- сложный: длина пароля свыше 8 символов, включает одновременно прописные буквы, строчные буквы и цифры.

Комментарий. В программе использованы различные методы работы со строками, такие как **isalpha(),** проверяющий, состоит ли строка из букв, **isdigit(),** проверяющий, состоит ли строка из цифр, **isupper(),** проверяющий, есть ли в строке символы в верхнем регистре, **islower(),** проверяющий, есть ли в строке символы в нижнем регистре. Код программы, отвечающий за решение задачи, представлен в листинге 94.

Листинг 94

```
print("\n\nДля создания надежного пароля используйте: 1.Заглавные буквы алфавита (от A до Z). 2.Строчные буквы алфавита (от a до z). 3.Цифры (от 0 до 9). 4.Длина пароля >8 символов.")
parol=input("\nВведите пароль: ")
if len(parol)>=8:
    if parol.isalpha()==True:
        print("Пароль средней сложности, рекомендуем поменять пароль!")
        print("Ваш пароль: ",parol)
    if parol.isdigit()==True:
        print("Пароль средней сложности, рекомендуем поменять пароль!")
```

```
        print("Ваш пароль: ",parol)
    else:
        if parol.isupper()==True:
            print("Пароль средней сложности, рекомендуем поменять пароль!")
            print("Ваш пароль: ",parol)
        if parol.islower()==True:
            print("Пароль средней сложности, рекомендуем поменять пароль!")
            print("Ваш пароль: ",parol)
        else:
            print("Вы придумали надежный пароль!")
            print("Ваш пароль: ",parol)
if 6<len(parol)<8:
    if parol.isalpha()==True:
        print("Пароль ненадежный, рекомендуем поменять пароль!")
        print("Ваш пароль: ",parol)
    if parol.isdigit()==True:
        print("Пароль ненадежный, рекомендуем поменять пароль!")
        print("Ваш пароль: ",parol)
    else:
        if parol.isupper()==True:
            print("Пароль средней сложности, рекомендуем поменять пароль!")
            print("Ваш пароль: ",parol)
        if parol.islower()==True:
            print("Пароль средней сложности, рекомендуем поменять пароль!")
            print("Ваш пароль: ",parol)
        else:
            print("Пароль средней сложности, рекомендуем поменять пароль!")
            print("Ваш пароль: ",parol)
if len(parol)<=6:
    print("Пароль ненадежный, рекомендуем поменять пароль!")
    print("Ваш пароль: ",parol)
```

Задача 5. Разработайте программу, которая подсчитывает число слов во введенной с клавиатуры строке. Условимся считать словом любую последовательность символов, которая отделена от других пробелом.

Комментарий. Метод **capitalize()** переводит первый символ строки в верхний регистр, поэтому в результирующей строке произойдут изменения, если пользователь начнет набирать строку строчными буквами. С помощью метода **split**(" "), который осуществляет разбиение строки по разделителю (в нашем случае – пробел), будут выявлены слова во введенной строке. Код программы, отвечающий за решение задачи, представлен в листинге 95.

Листинг 95

```
k=0
stroka=input('\nВведите предложение: ')
s=stroka.capitalize()
```

```
spisok=stroka.split(" ")
for i in range (len(spisok)):
   k=k+1
print("\nПредложение: ",s,"\nВ этом предложении",k," слов(а)")
```

Задача 6. Дана строка символов. Определите количество слов, являющихся записью десятичного числа.

Комментарий. Воспользуемся методом **isdigit(),** назначение которого состоит в анализе строки на предмет наличия в ней цифр. В цикле с оператором **for** будем проверять каждое слово строки и в случае истинности условия, увеличивать счетчик на единицу. Код программы, отвечающий за решение задачи, представлен в листинге 96.

Листинг 96

```
kol=0
stroka=input("Введите строку символов\n")
spisok=stroka.split(" ")
n=len(spisok)
for i in range(n):
   if spisok[i].isdigit():
      kol=kol+1
stroka1="".join(stroka)
print(stroka1)
print("\Количество слов, являющихся записью десятичного числа ", kol)
```

Результат работы программы представлен на рис. 115.

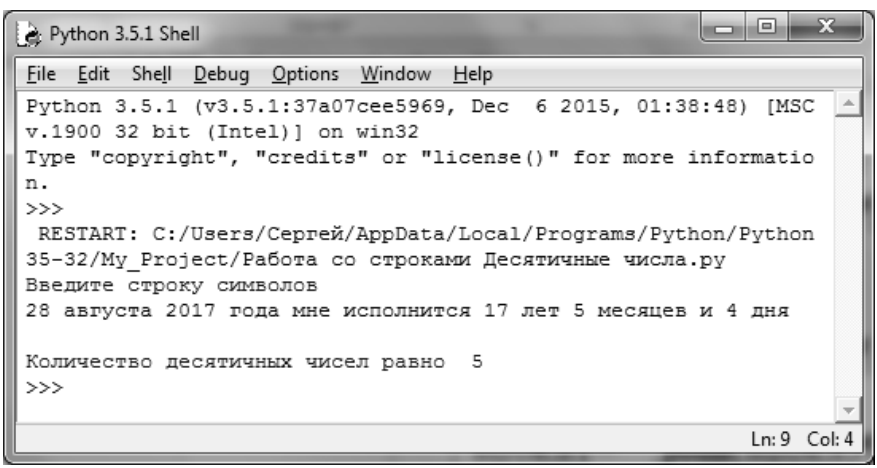


Рис. 115. Во введенной строке 5 десятичных чисел

Контрольные вопросы

1. Как называется кодировка, поддерживающая кодирование буквенно-цифровых символов? Расскажите о ее структуре.
2. Перечислите основные функции для работы с символами. Приведите примеры.
3. Перечислите методы работы со строками, позволяющие преобразовывать символы строки к различным регистрам клавиатуры.
4. Какой метод позволяет разбить строку на подстроки? Напишите его синтаксис.
5. Какой метод отвечает за преобразование строки в список? Напишите его синтаксис.
6. Приведите примеры базовых алгоритмов строк.
7. Каким образом можно осуществить срез строки?

Задачи для самостоятельного решения

1. Разработайте программу, которая подсчитывает число слов во введенной с клавиатуры строке. Условимся считать словом любую последовательность символов, которая отделена от других пробелом.
2. Разработайте программу, которая проверяет, является ли введенная с клавиатуры последовательность символов целым числом, записанным в двоичной системе счисления.
3. Разработайте программу, которая вычисляет среднюю длину слов во введенной с клавиатуры строке.
4. Разработайте программу, которая зашифровывает введенный с клавиатуры текст. Процесс шифровки производится следующим образом: из десятичного кода каждого введенного с клавиатуры символа вычитается число десять. Получившаяся в результате вычитания величина интерпретируется как десятичный код некоторого другого символа, который и выводится на экран компьютера.
5. Дано слово. Определите, является ли оно палиндромом (словом, которое читается одинаково в обоих направлениях, например, «потоп»).
6. Дана строка символов. Определите самое длинное слово в строке и количество слов такой же длины.
7. Дана строка символов. Удалите из нее все пробелы.
8. Дана строка символов. Дано слово. Удалите из строки это слово.
9. Дана строка символов. Выделите подстроку между первой и второй точками.
10. Дана строка символов. Определите длину самого короткого и самого длинного слова.
11. Дана строка символов. Определите, сколько слов начинается и кончается одной и той же буквой.
12. Дана строка символов. Определите, сколько слов содержат хотя бы одну букву «е».

13. Дана строка символов. Определите, является ли она правильным скобочным выражением. Рассматривайте только круглые скобки.
14. Дана строка символов. Определите, сколько слов содержат ровно три буквы «е».
15. Строка содержит только цифры. Удалите все впереди стоящие нули.
16. Дана строка символов. Подсчитайте количество знаков препинания в строке.
17. Дана строка символов. Удалите из строки все запятые.
18. Дана строка символов. Приведено некоторое слово. Вставьте его после каждого пробела.
19. Дана строка символов. Найдите сумму чисел, встречающихся в строке.
20. Дана строка символов. Удалите из строки все числа.
21. Дана строка символов. Удалите из строки слово, имеющее наибольшую длину.
22. Дана строка символов. Вставьте в строку пробел после каждого знака препинания.
23. Дана строка из цифр и латинских букв. Определите, каких букв – гласных (A, E, I, O, U) или согласных – больше в этой строке.
24. Из заданной строки удалите те ее части, которые заключены в кавычки (вместе с кавычками).
25. Задано слово. Замените в этом слове букву A на букву O. Если буквы A в этом слове нет, то выведите соответствующее сообщение.

8. ОБРАБОТКА ВЛОЖЕННЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

8.1. Формирование вложенных последовательностей

Ранее мы познакомились с обработкой таких последовательностей данных с помощью языка программирования Python, как списки и кортежи. Однако очень часто информация находится в таблицах, данные в которых представлены в виде последовательности строк. Получается, что строки как бы вложены друг в друга, т. е. одна последовательность данных вложена в другую.

Можно провести аналогию с другими языками программирования, когда обработка информации, размещенной в таблице, происходит с помощью инициализации так называемых **двумерных массивов** (матриц).

На рис. 116 представлен общий вид квадратной матрицы 5×5. Каждый элемент матрицы имеет имя **A.** Первый индекс – это **номер строки**, второй индекс– **номер столбца**. Диагональ матрицы $A_{00} – A_{44}$ называется **главной,** а диагональ $A_{04} – A_{40}$ – **побочной.**

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} & a_{04} \\ a_{10} & a_{11} & a_{12} & a_{13} & a_{14} \\ a_{20} & a_{21} & a_{22} & a_{23} & a_{24} \\ a_{30} & a_{31} & a_{32} & a_{33} & a_{34} \\ a_{40} & a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

Рис. 116. Общий вид квадратной матрицы

Возможности языка Python по обработке вложенных последовательностей удивят человека, привыкшего к классической обработке матриц в таких языках программирования, как C#, Microsoft Visual Basic, Delphi и др. Например, вложенная последовательность в Python может содержать одновременно список и кортеж, строковые и числовые данные и т. д.

Следует отметить, что точно так же, как и при обработке матриц, операции по обработке, вводу-выводу элементов вложенных последовательностей в Python осуществляются с использованием сложного циклического процесса.

В нижеприведенном примере (листинг 97) список **a** состоит из трех элементов: первый – список 1, 9, 6, 2, второй – список 5, 6, 7, 12 и третий – список 7, 8, 9, 28.

```
a=[
    [1,9,6,2],
    [5,6,7,12],
    [7,8,9,28]
]
```

Выводить на экран такой список будем, организовав первый цикл по номеру строки, а второй цикл – по элементам внутри строки.

Листинг 97

```
a=[
        [1,9,6,2],
        [5,6,7,12],
        [7,8,9,28]
   ]
for i in range(len(a)):
        for j in range(len(a[i])):
            print(a[i][j], end="   ")
        print()
```

Результат работы программы показан на рис. 117.

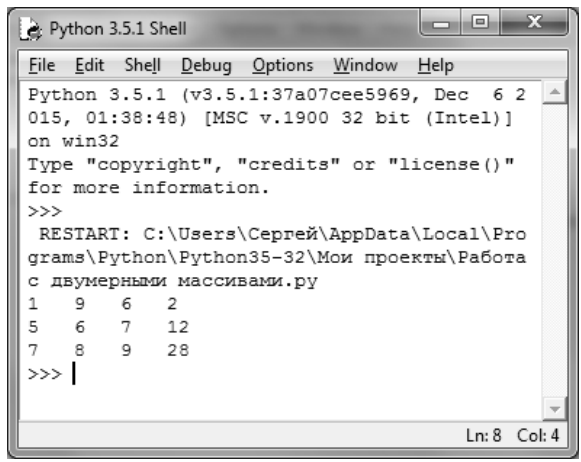


Рис. 117. Вывод на экран заранее заданной вложенной последовательности

Обратиться к элементу вложенного списка можно, указав его индекс. Например,

print(a[0])

На экран будет выведен список: (1, 9, 6, 2).

Чтобы извлечь третий элемент первого списка, следует указать два индекса: первый – номер списка, в котором находится нужный элемент; второй – позиция этого элемента в списке. Так, для того чтобы извлечь третий элемент списка (цифра 2) в первом списке, напишем такой оператор:

print(a[0][3])

Сейчас мы создадим вложенную последовательность – список, состоящий из нескольких кортежей, в которых определим наименование продукта и его стоимость.

a=[("Торт", 800), ("Пирожное", 350), ("Шоколад", 150)]

Для создания вложенных последовательностей можно использовать вложенные генераторы, разместив генератор списка, являющегося строкой, внутри генератора строк (листинг 98). Конечный элемент в данном примере вычисляется по формуле **a[i][j]=i*j.**

Листинг 98

```
n=5 #Количество строк
m=5 #Количество столбцов
a = [[i * j for j in range(m)] for i in range(n)]
for i in range(n):
        for j in range(m):
            print(a[i][j], end="   ")
        print()
```

Немного видоизменив предыдущий код за счет применения функции **sample,** будем из исходной последовательности элементов списка возвращать указанное количество элементов (листинг 99). Лучше, если это будет число пять, потому что матрица 5×5, как правило, используется в учебных целях при решении задач. При этом воспользуемся модулем **random,** подключив его с помощью инструкции **import.**

Листинг 99

```
import random
sum=0
n=5
m=5
a = [[i+j for j in random.sample(range(100),5)] for i in random.sample(range(100),5)]
for i in range(n):
   for j in range(m):
      print(" %3d " a[i][j], end="  ")
   print()
```

Результат работы программы показан на рис. 118.

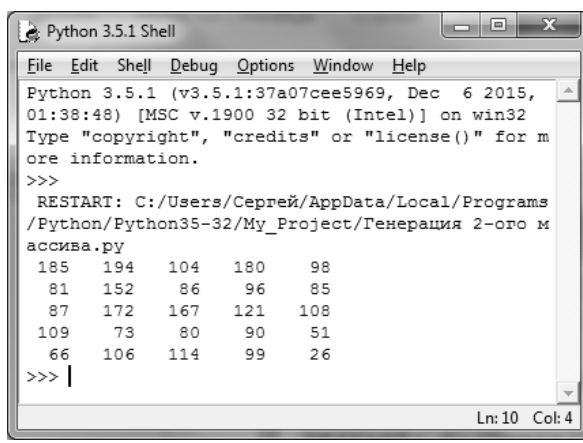


Рис. 118. Вложенная последовательность, сгенированная случайным образом

Задача. Во вложенной последовательности заранее заданных целых чисел найдите сумму всех элементов. Разработка алгоритма решения задачи представлена на рис. 119.

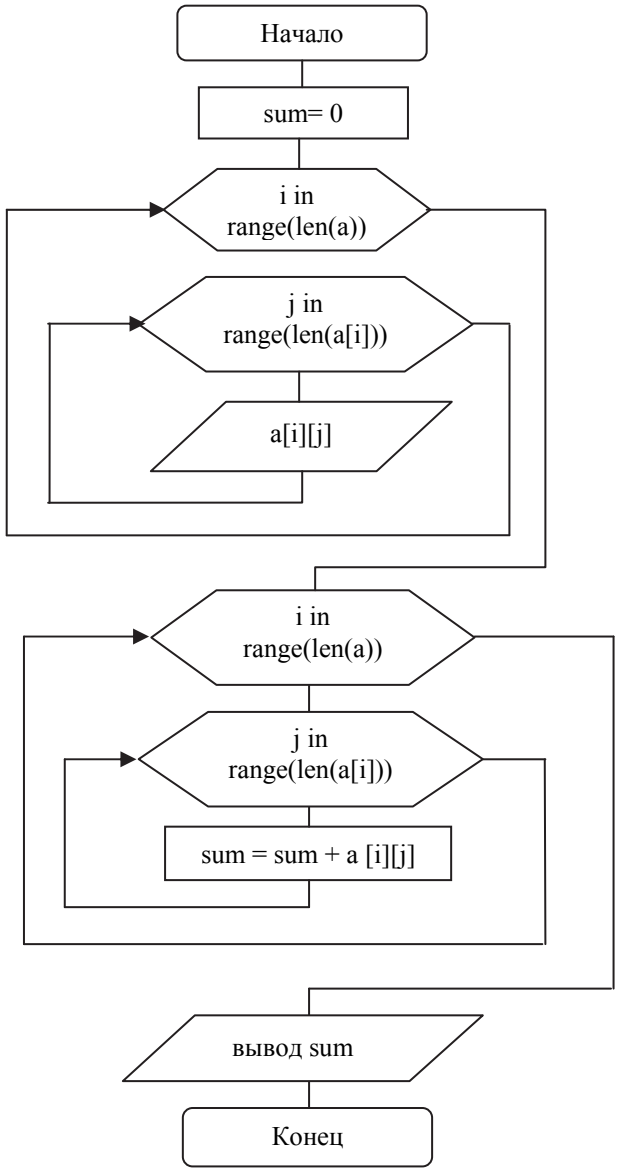


Рис. 119. Алгоритм решения задачи

Ниже приведен код программы, отвечающий за решение задачи, а на рис. 120 показан результат ее выполнения.

Листинг 100

```
sum=0
a=[
        [1,9,6,2],
        [5,6,7,12],
        [7,8,9,28]
   ]
for i in range(len(a)):
        for j in range(len(a[i])):
            print(a[i][j], end="   ")
        print()
for i in range(len(a)):
        for j in range(len(a[i])):
            sum=sum+a[i][j]
print("\nСумма элементов списка = ", sum)
```

Результат нахождения суммы элементов вложенной последовательности представлен на рис. 120.

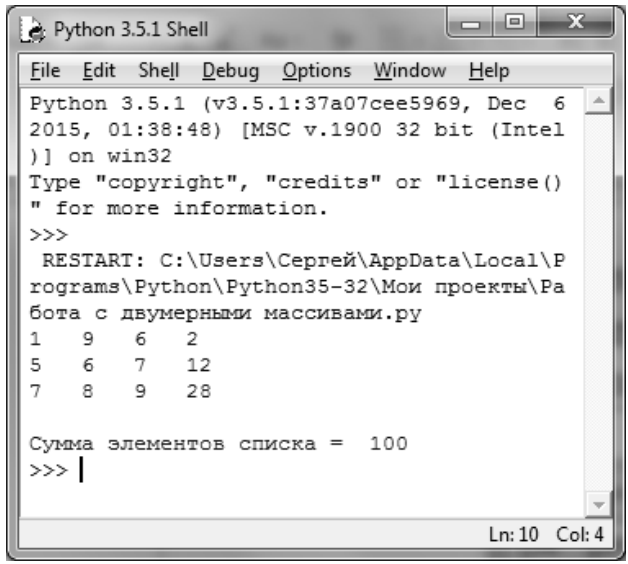


Рис. 120. Результат выполненной программы

В листинге 101 решение задачи, приводящее к нахождению суммы элементов вложенной последовательности, организовано не по индексу элемента, а по значениям списка, где **row** – строка, **elem** – значение элемента в строке.

Листинг 101

```
sum=0
a=[[1,2,3,4],[5,6,7,12],[7,8,9,56]]
for i in range(len(a)):
```

```
        for j in range(len(a[i])):
            print(a[i][j], end="   ")
        print()
for row in a:
        for elem in row:
            sum+=elem
print("\nСумма элементов списка = ", sum)
```

8.2. Базовые алгоритмы обработки вложенных последовательностей

В силу принятого нами подхода работы со списками можно выделить ряд приемов, позволяющих реализовать основные задачи обработки вложенных последовательностей, а именно:

1. Нахождение количества элементов вложенной последовательности при заданном условии;
2. Нахождение суммы значений элементов вложенной последовательности при заданном условии;
3. Нахождение произведения значений элементов вложенной последовательности при заданном условии;
4. Поиск экстремальных значений элементов вложенной последовательности (поиск максимального и/или минимального значения);
5. Обмен столбцов элементов вложенной последовательности;
6. Обмен строк элементов вложенной последовательности;
7. Удаление заданной строки вложенной последовательности;
8. Замена значений элементов вложенной последовательности.

Ниже описывается реализация перечисленных алгоритмов обработки вложенных последовательностей. Первые четыре способа чрезвычайно просты, данные алгоритмы неоднократно рассматривались при решении задач в предыдущих разделах и в настоящее время не требуют комментариев. Однако стоит отметить, что в первых четырех листингах очередной элемент вложенной последовательности сравнивается с числом 10, однако на практике лучше организовать запрос данных от пользователя.

Нахождение количества элементов вложенной последовательности при заданном условии

Листинг 102

```
import random
kol=0
n=5
m=5
a = [[i+j for j in random.sample(range(15),5)] for i in ran-
dom.sample(range(15),5)]
```

```
for i in range(n):
   for j in range(m):
      print(a[i][j], end="      ")
   print()
for i in range(n):
   for j in range(m):
      if a[i][j]>=10:
         kol+=1
print("\nКоличество элементов вложенной последовательности >= 10
равно ", kol)
```

Нахождение суммы значений элементов вложенной последовательности при заданном условии

Листинг 103

```
import random
sum=0
n=5
m=5
a = [[i+j for j in random.sample(range(15),5)] for i in ran-
dom.sample(range(15),5)]
for i in range(n):
   for j in range(m):
      print(a[i][j], end="      ")
   print()
for i in range(n):
   for j in range(m):
      if a[i][j]>=10:
         sum+=a[i][j]
print("\nСумма элементов вложенной последовательности >= 10 равно ",
sum)
```

Нахождение произведения значений элементов вложенной последовательности при заданном условии

Листинг 104

```
import random
pr=1
n=5
m=5
a = [[i+j for j in random.sample(range(15),5)] for i in ran-
dom.sample(range(15),5)]
for i in range(n):
          for j in range(m):
             print(a[i][j], end="      ")
```

```
        print()
for i in range(n):
        for j in range(m):
            if a[i][j]>=10:
                pr*=a[i][j]
print("\nПроизведение элементов вложенной последовательности >= 10 равно ", pr)
```

Нахождение экстремальных значений во вложенной последовательности

Листинг 105

```
import random
max = -32768
min = 32767
n=5
m=5
a = [[i+j for j in random.sample(range(15),5)] for i in random.sample(range(15),5)]
for i in range(n):
        for j in range(m):
            print(a[i][j], end="      ")
        print()
for i in range(n):
        for j in range(m):
            if a[i][j]>max:
                max=a[i][j]
            if a[i][j]<min:
                min=a[i][j]
print("\nМаксимальный элемент вложенной последовательности = ", max)
print("\nМинимальный элемент вложенной последовательности = ", min)
```

Обмен столбцов во вложенной последовательности

Задача. Во вложенной последовательности произвольных чисел поменяйте местами первый и четвертый столбцы. Выведите обе вложенные последовательности на экран.

Комментарий. После генерации элементов вложенной последовательности следует организовать цикл и выполнить в нем три оператора, отвечающих за перестановку столбцов вложенной последовательности:

```
k = a[i][1]
a[i][1] = a[i][4]
a[i][4] = k
```

Поскольку при первом вхождении в цикл параметр цикла **i** примет значение, равное нулю, во вспомогательную ячейку **k** отправляется нулевой элемент первого столбца, т. е. элемент, имеющий координаты $\mathbf{a_{01}}$. На его место приходит эле-

мент с координатами $\mathbf{a_{04}}$**.** На освободившееся место перейдет элемент, находящийся в ячейке **k.** Произошел обмен. Так как все действия осуществляются в цикле **for i in range(n)**, происходит обмен элементов первого и четвертого столбцов.

Разработка алгоритма решения задачи представлена на рис. 121.

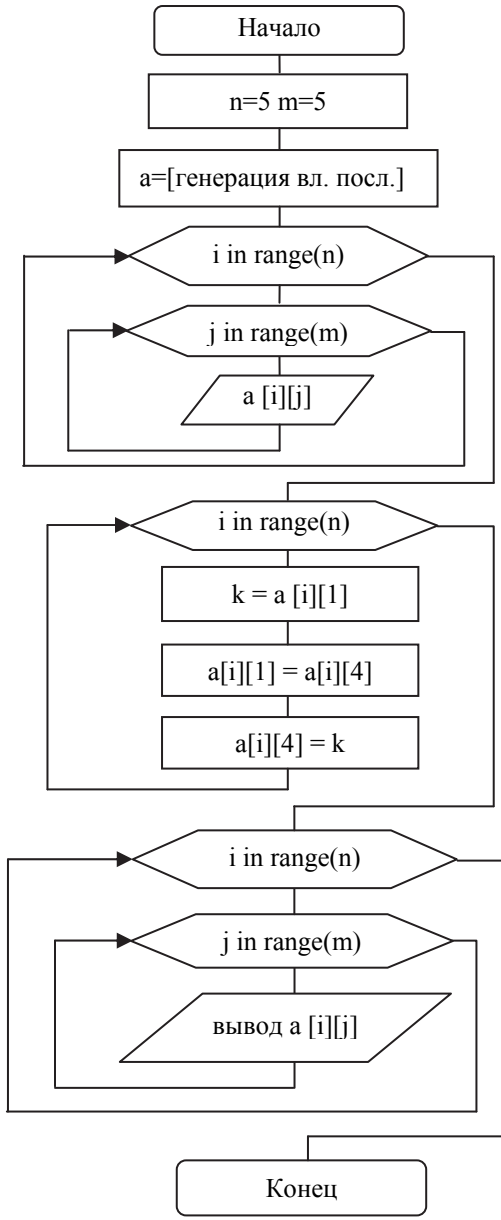


Рис. 121. Алгоритм решения задачи

В листинге 106 приведен код программы, отвечающий за решение задачи.

Листинг 106

```
import random
n=5
m=5
a = [[i+j for j in random.sample(range(15),5)] for i in ran-
dom.sample(range(15),5)]
for i in range(n):
        for j in range(m):
            print(" %3d " a[i][j], end="  ")
        print()
for i in range(n): #Обмен столбцов
        k=a[i][1]
        a[i][1]=a[i][4]
        a[i][4]=k
print()
for i in range(n): #Вывод результирующей последовательности
        for j in range(m):
           print(" %3d " a[i][j], end="  ")
         print()
```

Результат выполненной программы представлен на рис. 122.

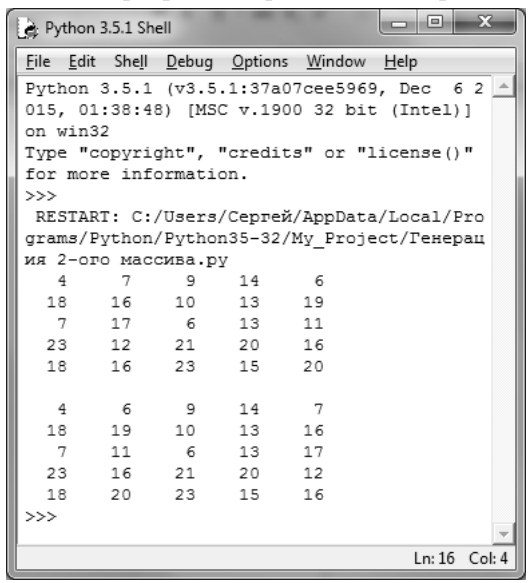


Рис. 122. Результат работы программы: первый столбец исходной вложенной последовательности поменяли с четвертым столбцом

Обмен строк во вложенной последовательности

В данной задаче происходит обмен тех строк, номера которых (**n_st1** и **n_st2**) задает пользователь. Алгоритмически метод ничем не отличается от рассмот-

ренного выше. При обмене следует воспользоваться третьей ячейкой **bufer.** В листинге 107 приведен код программы, отвечающий за решение задачи.

Листинг 107

```
import random
n=5
m=5
a = [[i+j for j in random.sample(range(15),5)] for i in ran-
dom.sample(range(15),5)]
for i in range(n):
   for j in range(m):
      print(a[i][j], end="      ")
   print()
n_st1=int(input("\nВведите номер строки, которая подлежит замене  "))
n_st2=int(input("\nВведите номер строки, которую обменяем  "))
for j in range(n):
   bufer=a[n_st2][j]
   a[n_st2][j]=a[n_st1][j]
   a[n_st1][j]=bufer
print()
for i in range(n):
   for j in range(m):
      print(a[i][j], end="      ")
   print()
```

Результат выполненной программы представлен на рис. 123.

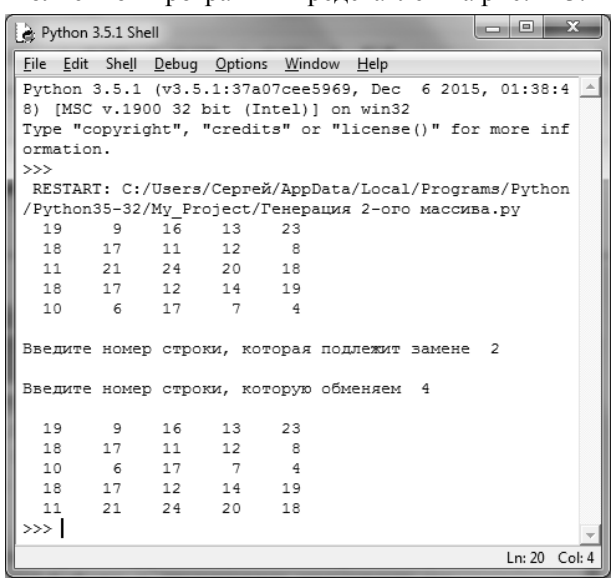


Рис.123. Результат работы программы: вторую строку исходной вложенной последовательности поменяли с четвертой строкой

Удаление заданной строки во вложенной последовательности

Удаление заданной строки с номером **n_st** происходит с помощью сдвига на один шаг вверх всех последующих строк, начиная с **n_st+1.** Задачу удаления столбца можно решить аналогично. В листинге 108 приведен код программы, отвечающий за решение задачи.

Листинг 108

```
import random
n=5
m=5
a = [[i+j for j in random.sample(range(15),5)] for i in ran-
dom.sample(range(15),5)]
for i in range(n):
   for j in range(m):
      print(" %3d " % a[i][j], end="  ")
   print()
n_st=int(input("\nВведите номер строки, которую хотите удалить   "))
for i in range(n_st+1,5):
   for j in range(m):
      a[i-1][j] = a[i][j]
print()
for i in range(n-1):
   for j in range(m):
      print(" %3d " % a[i][j], end="  ")
   print()
```

Результат выполненной программы представлен на рис. 124.

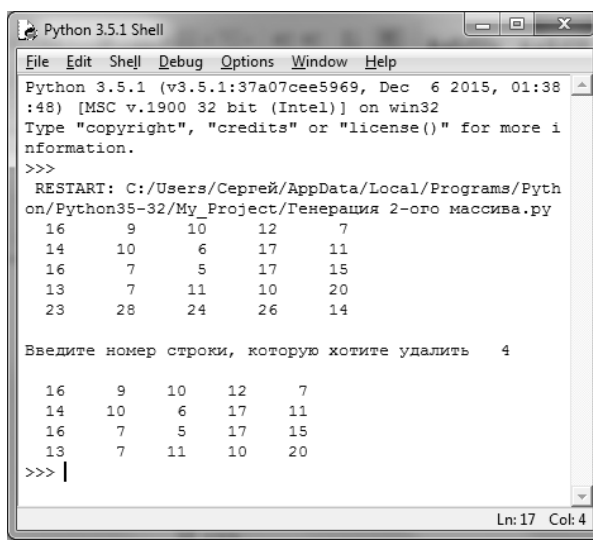


Рис. 124. Удалена четвертая строка исходной матрицы

Замена значений элементов вложенной последовательности

Задача. Все элементы исходной вложенной последовательности замените нулем, а каждый элемент главной диагонали замените его номером.

Вполне логично первоначально осуществить проверку условия: «Находится ли элемент вложенной последовательности на главной диагонали?», т. е. **if i==j**, а затем выполнить прямое присваивание **a[i][j]=i** (где i – номер элемента на главной диагонали) или оператор **a[i][j]=0**, который заменит элементы исходной вложенной последовательности нулем. В листинге 109 приведен код программы, отвечающий за решение задачи.

Листинг 109

```
import random
n=5
m=5
a = [[i+j for j in random.sample(range(15),5)] for i in ran-
dom.sample(range(15),5)]
for i in range(n):
        for j in range(m):
            print(" %3d " % a[i][j], end="  ")
        print()
for i in range(n):
        for j in range(n):
            if i==j: #Проверка: находится элемент на главной диагонали
        # или нет?
                a[i][j]=i
            else:
                a[i][j]=0 #Замена остальных элементов последовательности
#на нуль
print()
for i in range(n):
        for j in range(m):
            print(" %3d " % a[i][j], end="  ")
    print()
```

Результат выполненной программы представлен на рис. 125.

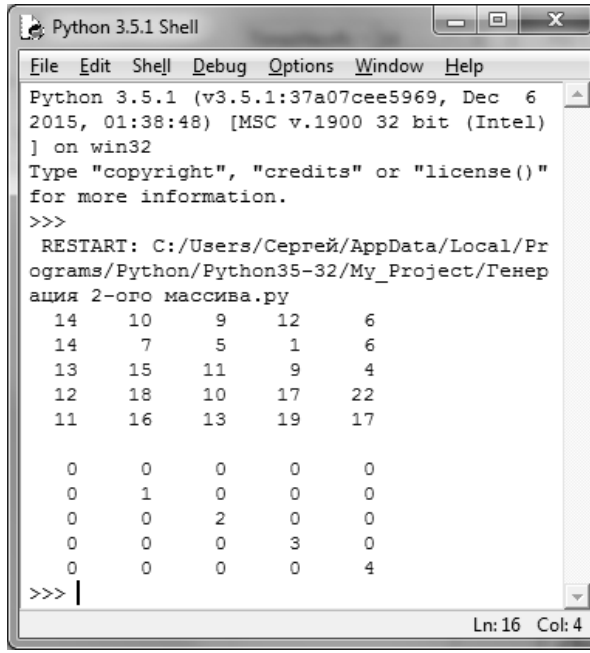


Рис. 125. Все элементы вложенной последовательности заменены нулями, элементы главной диагонали – порядковыми номерами

8.3. Примеры решения задач

Задача 1. Во вложенной последовательности произвольных чисел найдите максимальный элемент на главной диагонали и его номер (индекс). Разработка алгоритма решения задачи представлена на рис. 126.

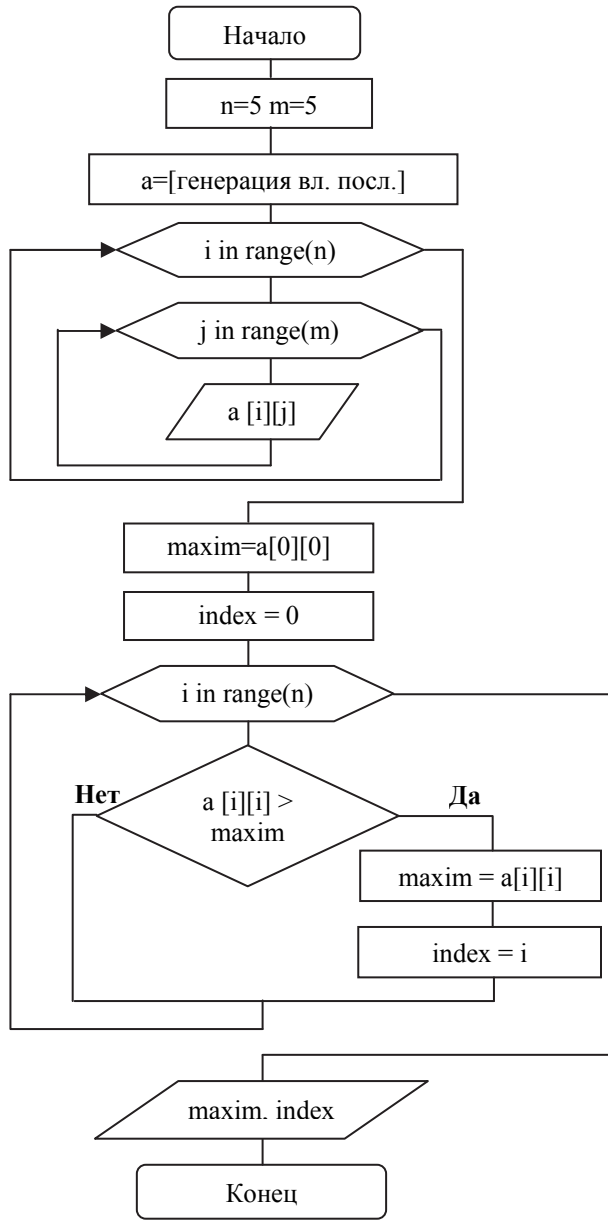


Рис. 126. Алгоритм решения задачи

Комментарий. Элементы главной диагонали рассматриваемой вложенной последовательности имеют следующие индексы: **a[0][0], a[1][1], a[2][2], a[3][3], a[4][4],** поэтому для поиска максимального элемента применим следующий алгоритм решения задачи: максимальным считается нулевой элемент главной диагонали – **a[0][0].** В программе за это действие отвечает оператор **maxim=a[0][0].** Далее осуществляется сравнение очередного элемента главной диагонали с тем элементом, который хранится в ячейке **maxim,** и если он больше него, данный элемент заносится в ячейку **maxim** оператором **maxim=a[i][i].** В листинге 110 приведен код программы, отвечающий за решение задачи.

Листинг 110

```
import random
n=5
m=5
a=[[i+j for j in random.sample(range(15),m)] for i in ran-
dom.sample(range(15),n)]
print("\nИсходная вложенная последовательность")
for i in range(n):
   for j in range(m):
      print(" %3d " % a[i][j],", end=" ")
   print()
maxim=a[0][0] #Максимальным считается нулевой элемент главной
# диагонали
index=0
for i in range(n):
   if a[i][i]>maxim: #Поиск максимального элемента главной диагонали
      maxim=a[i][i]
      index=i    #Фиксация его номера
print("\nМаксимальный элемент главной диагонали =", maxim)
print("\nЕго индекс =", index)
```

Задача 2. Из вложенной последовательности произвольных чисел сформируйте другую вложенную последовательность, содержащую квадраты чисел исходной. Разработка алгоритма решения задачи представлена на рис. 127.

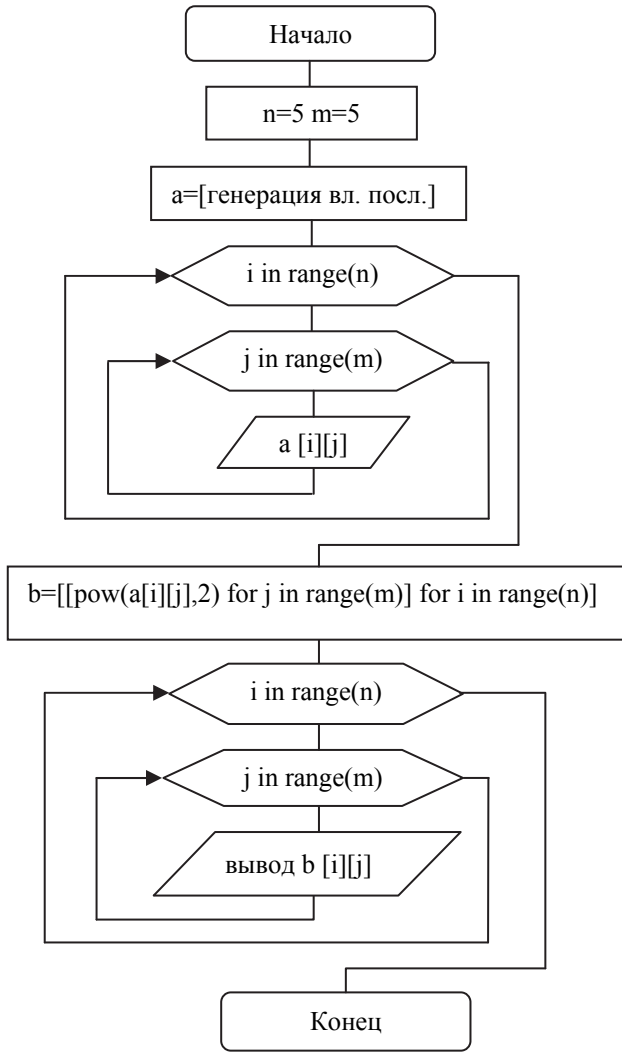


Рис. 127. Алгоритм решения задачи

Комментарий. Формирование вложенной последовательности, содержащей квадраты чисел исходной последовательности, осуществляется в цикле оператором **b=[[**pow**(a[i][j],2)** for **j** in range**(m)]** for **i** in range**(n)].** В листинге 111 приведен код программы, отвечающий за решение задачи.

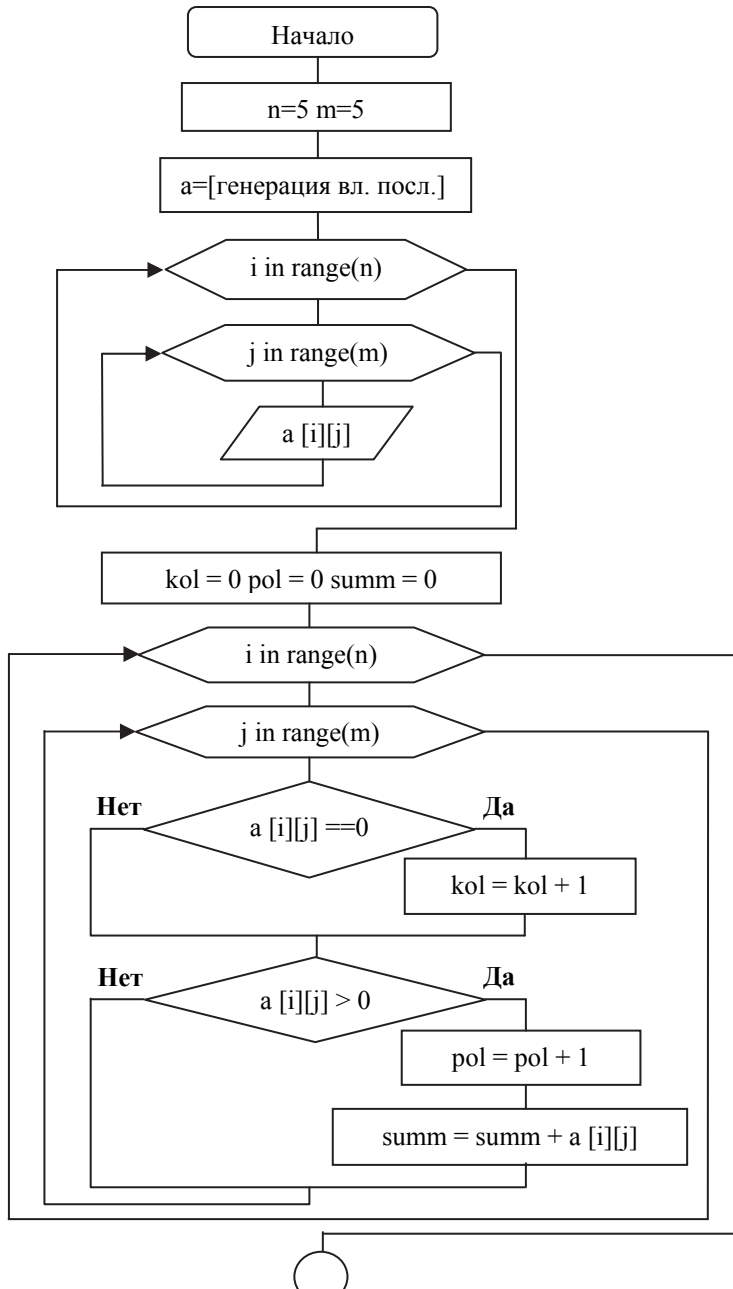
Листинг 111

```
import random
n=5
```

```
m=5
a=[[i+j for j in random.sample(range(15),m)] for i in ran-
dom.sample(range(15),n)]
print("\nИсходная вложенная последовательность")
for i in range(n):
    for j in range(m):
        print(" %3d " % a[i][j],", end=" ")
    print()
print("\nРезультирующая вложенная последовательность")
b=[[pow(a[i][j],2) for j in range(m)] for i in range(n)]
for i in range(n):
    for j in range(m):
        print(" %3d " % b[i][j],", end=" ")
    print()
```

Задача 3. Во вложенной последовательности произвольных чисел вычислите среднее арифметическое положительных элементов и количество элементов равных нулю.

Разработка алгоритма решения задачи представлена на рис. 128.



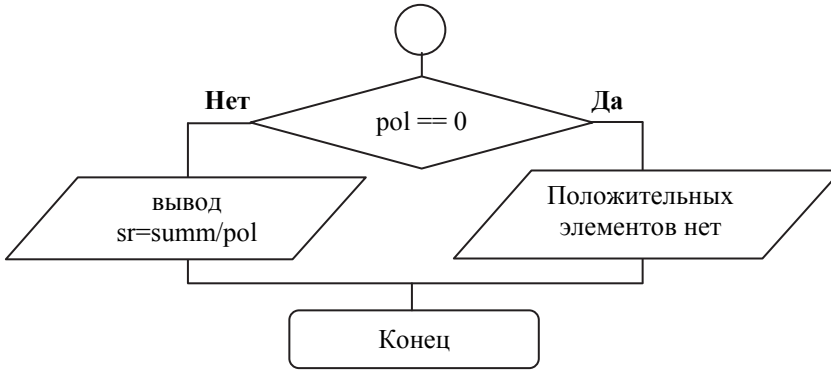


Рис. 128. Алгоритм решения задачи

Комментарий. Организовав циклический процесс, проверяем каждый элемент вложенной последовательности на равенство нулю и в случае истинности условия увеличиваем счетчик на единицу оператором **kol=kol+1.** Затем проверяем каждый элемент вложенной последовательности: является ли он положительным числом? В случае истинности условия накапливаем сумму положительных чисел оператором **summ=summ+a[i][j]** и увеличиваем счетчик на единицу оператором **pol=pol+1.** В программе предусмотрена защита на случай, если во вложенной последовательности нет положительных чисел. Тогда выводится надпись: «Положительных элементов во вложенной последовательности нет». В противном случае находится среднее арифметическое положительных элементов.

В листинге 112 приведен код программы, отвечающий за решение задачи.

Листинг 112

```
import random
n=5
m=5
a=[[i-j for j in random.sample(range(15),m)] for i in ran-
dom.sample(range(15),n)]
print("\nИсходная вложенная последовательность")
for i in range(n):
   for j in range(m):
      print(" %3d " % a[i][j],", end=" ")
   print()
kol=0
pol=0
summ=0
for i in range(n):
   for j in range(m):
      if a[i][j]==0: #Поиск элементов последовательности равных нулю
         kol=kol+1 #Увеличение счетчика на единицу в случае истинности
#условия
```

```
      if a[i][j]>0:  #Поиск положительных элементов последовательности
         pol=pol+1 #Увеличение счетчика на единицу в случае истинности
#условия
         summ=summ+a[i][j] #Нахождение суммы положительных чисел
if pol==0:
   print("\nПоложительных элементов во вложенной последовательности
нет")
else:
   sr=summ/pol
   print("\nСреднее арифметическое положительных элементов =", sr)
   print("\nКоличество нулевых элементов =", kol)
```

Задача 4. Во вложенной последовательности произвольных чисел найдите четные элементы и выведите их как список. Разработка алгоритма решения задачи представлена на рис. 129.

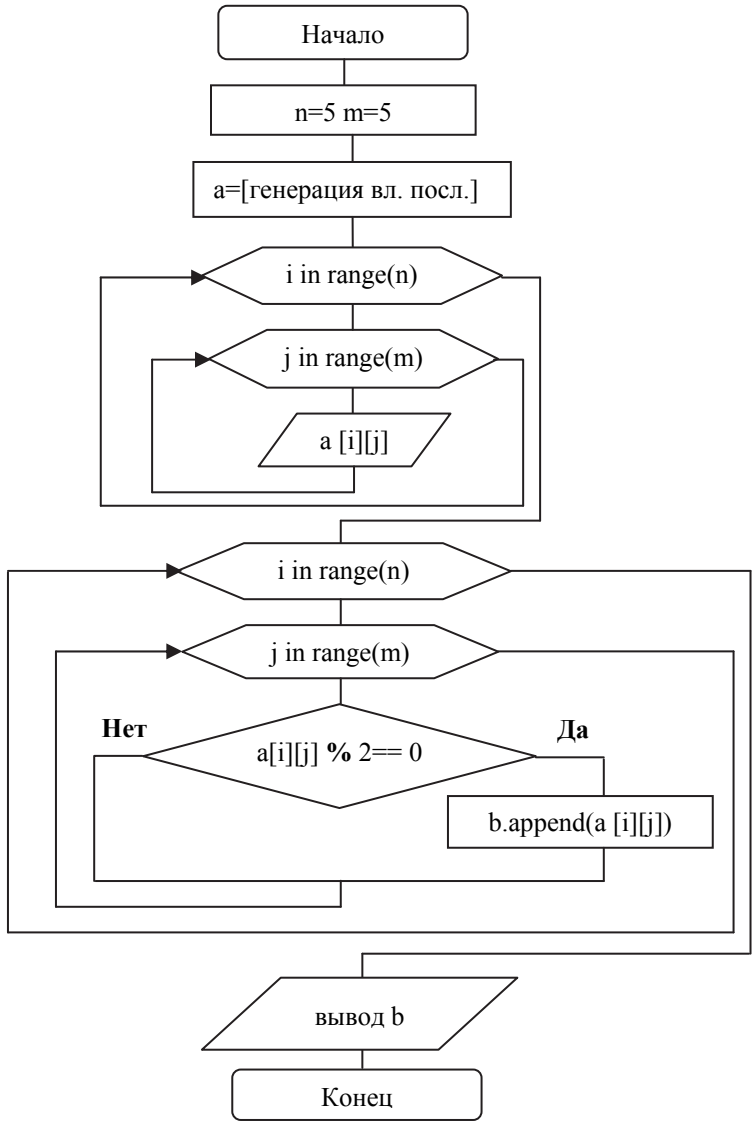


Рис. 129. Алгоритм решения задачи

Комментарий. Проверка на четность очередного элемента последовательности происходит оператором **if (a[i][j]%2==0)**. Далее, применив метод **append(),** формируем список четных элементов. В листинге 113 приведен код программы, отвечающий за решение задачи.

Листинг 113

```
import random
n=5
m=5
b=[]
a=[[i+j for j in random.sample(range(15),m)] for i in ran-
dom.sample(range(15),n)]
print("\nИсходная вложенная последовательность")
for i in range(n):
    for j in range(m):
        print(" %3d " % a[i][j],", end=" ")
    print()
for i in range(n):
    for j in range(m):
        if a[i][j]%2==0:
            b.append(a[i][j])
print("\nСписок четных элементов")
print(b)
```

Задача 5. Во вложенной последовательности произвольных чисел вычислите сумму элементов, сумма индексов которых равна 4. Разработка алгоритма решения задачи представлена на рис. 130.

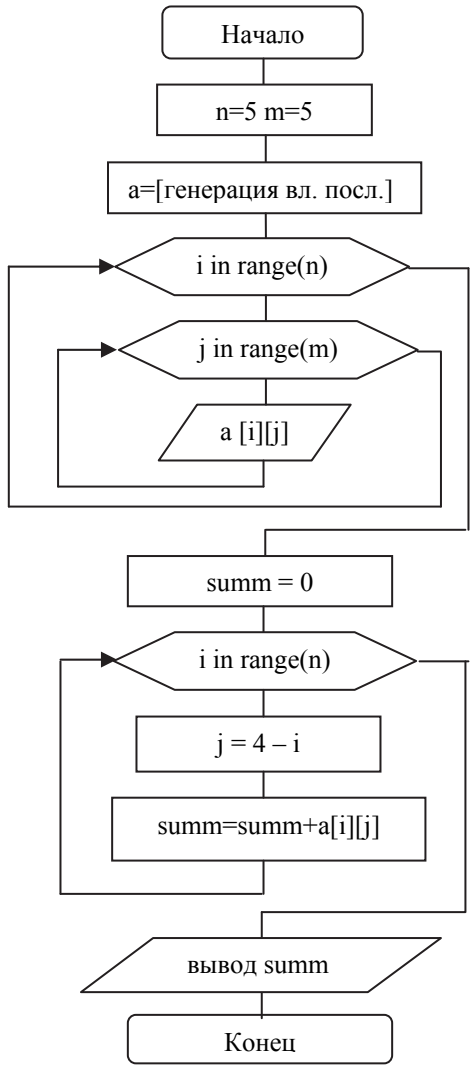


Рис. 130. Алгоритм решения задачи

Комментарий. В данной программе перебирать все элементы вложенной последовательности не следует. По сути, требуется найти сумму элементов $\mathbf{a_{04}}$, $\mathbf{a_{13}}$, $\mathbf{a_{22}}$, $\mathbf{a_{31}}$, $\mathbf{a_{40}}$. Именно у этих элементов сумма индексов равна четырем. Поскольку индексация начинается с нуля, первый индекс (номер строки, за который отвечает параметр цикла **i**) может принимать значения от 0 до 4, а второй (номер столбца, за который отвечает параметр цикла **j**) – от 4 до 0. Для решения задачи достаточно организовать один цикл. В нем будет вычисляться первый индекс элемента, вто-

рой индекс будет вычисляться в цикле оператором **j=4–i.** В листинге 114 приведен код программы, отвечающий за решение задачи.

Листинг 114

```
import random
n=5
m=5
a=[[i+j for j in random.sample(range(15),m)] for i in ran-
dom.sample(range(15),n)]
print("\nИсходная вложенная последовательность")
for i in range(n):
   for j in range(m):
      print(" %3d " % a[i][j],", end=" ")
   print()
summ=0
for i in range(n):
   j = 4 – i
   summ=summ+a[i][j]
print("\nСумма элементов, сумма индексов которых равна четырем =",
summ)
```

Задача 6. Сформируйте вложенную последовательность и найдите максимальный элемент, лежащий ниже главной диагонали, и минимальный элемент, лежащий выше главной диагонали. Найденные максимальный и минимальный элементы поменяйте местами. В листинге 115 приведен код программы, отвечающий за решение задачи.

Листинг 115

```
import random
n=5
m=5
a = [[i+j for j in random.sample(range(15),5)] for i in ran-
dom.sample(range(15),5)]
print("\nИсходная вложенная последовательность")
for i in range(n):
   for j in range(m):
      print(" %5d " % a[i][j],", end=" ")
   print()
str_min=0
stol_min=0
str_max=0
stol_max=0
#Находим минимальный элемент, лежащий выше главной диагонали
minim=a[0][0]
for i in range(n):
   for j in range(m):
```

```
      if i<j:
         if a[i][j]<minim:
            minim=a[i][j]
            str_min=i
            stol_min=j
#Находим максимальный элемент, лежащий ниже главной диагонали
maxim=a[0][0]
for i in range(n):
   for j in range(m):
      if i>j:
         if a[i][j]> maxim:
            maxim=a[i][j]
            str_max=i
            stol_max=j
print("\nМаксимальный элемент, лежащий ниже главной диагонали =",
maxim)
print("\nМинимальный элемент, лежащий выше главной диагонали =",
minim)
#Обмен элементов
bufer=a[str_min][stol_min]
a[str_min][stol_min]=a[str_max][stol_max]
a[str_max][stol_max]=bufer
#Вывод результирующей матрицы
print("\nРезультирующая вложенная последовательность")
for i in range(n):
   for j in range(m):
      print(" %5d " % a[i][j],", end=" ")
   print()
```

Задача 7. Сформируйте вложенную последовательность A[5,5] случайных целых чисел, находящимися в интервале от 1 до 40. В созданной последовательности найдите в каждой строке минимальный элемент и запишите его в выходной список.

Комментарий. После генерации вложенной последовательности и вывода ее на экран организуем сложный циклический процесс, в котором при прохождении внешнего цикла будем считать минимальным нулевой элемент каждой строки (оператор **minim=a[i][0]**). Во внутреннем цикле происходит последовательное сравнение каждого элемента строки с нулевым элементом, в результате чего определяем минимальный элемент. Затем с помощью метода **append()** добавляем найденный элемент в список с последующим выводом на экран. Код программы представлен в листинге 116.

Листинг 116

```
import random
n=5
m=5
a=[]
```

```
spisok=[]
print("\nИсходная вложенная последовательность")
a=[[i for i in random.sample(range(1,41),m)] for j in ran-
dom.sample(range(1,41),n)]
for i in range(n):
    for j in range(m):
        print(" %3d " % a[i][j],", end=" ")
    print()
for i in range(n):
    minim=a[i][0]
    for j in range(m):
        if a[i][j]<minim:
            minim=a[i][j]
    spisok.append(minim)
    print('\n Минимальный элемент ',i,' строки ',spisok[i])
print("\nСписок минимальных элементов")
print(spisok)
```

На рис. 131 представлен результат работы программы.

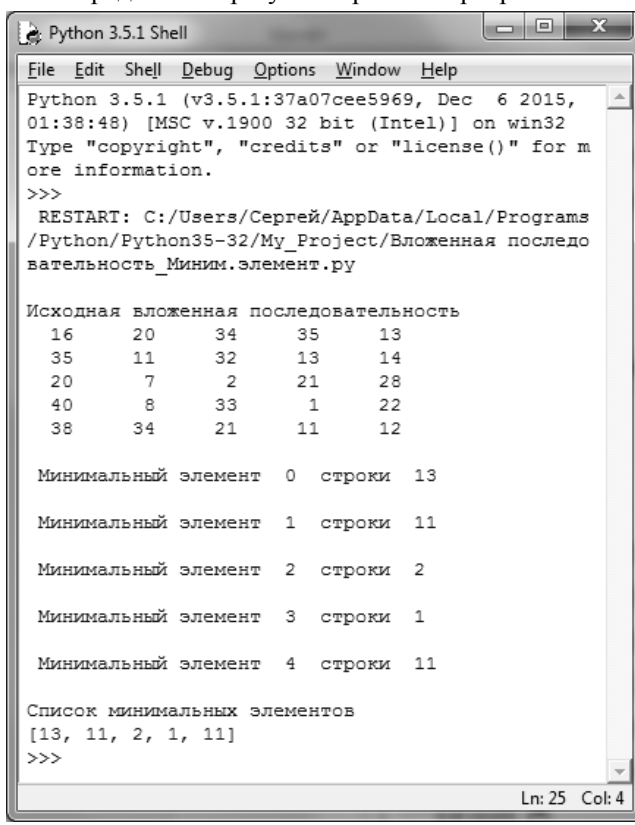


Рис. 131. Найденный список минимальных элементов

Контрольные вопросы

1. Нарисуйте общий вид квадратной матрицы. Какая диагональ называется главной, какая – побочной?
2. Каким образом можно обратиться к элементу вложенного списка?
3. Перечислите базовые алгоритмы обработки вложенных последовательностей.
4. Опишите словесный алгоритм нахождения количества элементов вложенной последовательности при некотором условии.
5. Опишите словесный алгоритм нахождения суммы элементов вложенной последовательности при некотором условии.
6. Опишите словесный алгоритм нахождения экстремальных значений вложенной последовательности при некотором условии.
7. Поясните, каким образом осуществляется обмен столбцов во вложенной последовательности.

Задачи для самостоятельного решения

1. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до 40. Сформируйте список, в который следует записать номера строк максимальных элементов каждого столбца.
2. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до 40. В созданной последовательности найдите сумму элементов, сумма индексов которых равна 4.
3. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до 40. В созданной последовательности найдите наибольший элемент побочной диагонали.
4. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до 80. В созданной последовательности найдите четные элементы и выведите их как список.
5. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от –15 до 40. В созданной последовательности вычислите среднее арифметическое положительных элементов и количество элементов, равных нулю.
6. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до 40. В созданной последовательности для каждой строки вычислите среднее арифметическое элементов, значения которых находятся в заданном диапазоне. Диапазон задан значениями нижней и верхней границ, при этом значения границ в диапазон не входят.

7. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до 20. Сформируйте два списка, в один запишите элементы последовательности, расположенные на главной диагонали и выше, в другой – элементы последовательности, лежащие ниже главной диагонали, и выведите оба списка.
8. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до 50. Выведите на экран только главную и побочную диагонали.
9. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до 40. В результирующей последовательности выведите на экран половину последовательности относительно главной диагонали.
10. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до 40. В созданной последовательности найдите сумму элементов на диагонали, параллельной побочной A14, A23, A32, A41, и сумму элементов на диагонали, параллельной главной A10, A21, A32, A43.
11. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до 40. Преобразуйте вложенную последовательность таким образом, чтобы строки с нечетными индексами были упорядочены по убыванию, с четными – по возрастанию.
12. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до 40. В созданной последовательности вычислите сумму наибольших значений в столбцах, выведите на экран список наибольших элементов.
13. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до 40. Все элементы исходной последовательности замените нулями, а каждый элемент главной диагонали замените его номером.
14. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до 40. Требуется поменять местами вторую и четвертую строки. Выведите обе последовательности на экран.
15. Разработайте программу, заполняющую две вложенные последовательности A[5,5] и B[5,5] случайными целыми числами, находящимися в интервале от 1 до 10. Осуществите перемножение последовательностей и выведите результирующую вложенную последовательность на экран.
16. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до 40. Вычислите две суммы элементов, расположенных выше и ниже главной диагонали.
17. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до

20. Преобразуйте вложенную последовательность так, чтобы первый элемент каждой строки был заменен средним арифметическим элементов этой строки.

18. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до 40. Определите номера столбцов, в которых среднее арифметическое их элементов меньше, чем среднее арифметическое элементов последовательности.
19. Пользователь вводит с клавиатуры элементы вложенной последовательности A[i,j]. Определите, является ли последовательность единичной. Единичной последовательностью называют вложенную последовательность, у которой элементы главной диагонали – единицы, все остальные – нули.
20. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до 40. Преобразуйте вложенную последовательность таким образом, чтобы каждый столбец был упорядочен по убыванию.
21. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от –20 до 20. Вычислите количество положительных элементов последовательности, расположенных по ее периметру и на диагоналях.
22. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до 40. В созданной последовательности вычислите суммы элементов строк.
23. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до 40. В созданной последовательности вычислите сумму максимальных значений в строках, выведите на экран список максимальных элементов.
24. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до 50. Определите максимальный элемент среди элементов последовательности, расположенных выше главной диагонали, и минимальный элемент среди тех, которые находятся ниже главной диагонали.
25. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от –20 до 20. Найдите в строках самые правые наименьшие элементы и определите их местоположение, т. е. выведите на экран номер столбца.
26. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до 60. Вычислите значение среднего арифметического ее элементов, больших, чем число 20.
27. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до 30. Осуществите поворот последовательности на 90° по часовой стрелке и выведите результат на экран.

28. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от 1 до 40. Определите, имеются ли среди ее элементов, лежащих ниже главной диагонали, отрицательные числа.
29. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от –15 до 30. Если хотя бы один элемент строки последовательности отрицателен, то все элементы этой строки замените нулями.
30. Разработайте программу, заполняющую вложенную последовательность A[5,5] случайными целыми числами, находящимися в интервале от –20 до 40. В созданной последовательности вычислите количество, сумму и среднее арифметическое отрицательных чисел.

9. РАБОТА С ФУНКЦИЯМИ. СОЗДАНИЕ МОДУЛЕЙ

Ранее было отмечено, что в ходе изучения языка программирования Python мы должны в совершенстве овладеть программированием на основе использования функций, а также говорилось о том, что встроенная функция SIN(x), которую мы часто используем в программах вычислительного характера, на самом деле представляет собой сложный знакопеременный математический ряд, который может быть вычислен по формуле:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1)!} + \ldots$$

Для вычисления суммы такого ряда в среде программирования необходимо написать достаточно сложную программу. Безусловно, легче вызвать уже написанную программистами функцию sin(x), которая размещена в библиотеке математических подпрограмм, и работать с ней, используя ее имя – sin.

Однако часто, несмотря на громадное количество встроенных функций, программист вынужден создавать свои собственные, нестандартные функции, поскольку при разработке программ участки кода, которые носят определенное функциональное назначение, могут быть использованы повторно. Таким образом, вызвать по имени ранее написанный программный блок, оказывается гораздо рациональнее, чем писать его снова и снова.

Кроме того, следует учесть, что разработка большого программного проекта, это, как правило, труд не одного человека, а коллектива разработчиков. Соответственно, в задачи одной группы программистов может входить разработка только одного из программных блоков (модуля), а в задачи другой группы – разработка другого модуля, выполняющего уже иное предназначение. Таким образом, можно сформулировать ряд преимуществ, которые получает программист от создания собственных функций. Итак, программный код, оформленный в виде функции:

- позволяет связать часто используемую группу операторов программного кода со знакомым именем;
- устраняет повторы фрагментов программного кода, т. е. можно один раз определить функцию и вызывать (обращаться к ней) любое количество раз;
- становится более простым и легко читаемым, так как, разделенный на небольшие части, он легче воспринимается;
- упрощает разработку проектов, так как проект, разделенный на логические (функциональные) единицы, легче разрабатывать и отлаживать; кроме того, если программный проект готовится группой разработчиков, то можно обмениваться готовыми функциями и модулями;
- позволяет повторно использовать функции в других проектах;
- расширяет язык программирования (пользовательские функции могут вы-

полнять задачи, которые не могут быть выполнены встроенными функциями языка).

Перейдем к рассмотрению процесса создания собственных функций.

9.1. Создание пользовательских функций

Для объявления функции используется следующий синтаксис.

def ИмяФункции(Параметр(ы))**:**

Операторы функции

return возвращаемое значение

где

- **def** (от англ. *define* – определять, устанавливать);
- **ИмяФункции** – уникальное имя создаваемой функции. На имя функции распространяются общие правила написания идентификаторов, перечисленные в параграфе 1.5;
- **Параметры** (аргументы) – список необязательных аргументов, разделенных запятыми и используемых в данной функции;
- **Операторы функции** – блок операторов, который выполняет работу функции;
- **return** – необязательный оператор, с помощью которого можно указать место, где в блоке кода функции требуется возвратить значение в вызывающую программу, и каково это возвращаемое значение. После выполнения данного оператора происходит выход из функции, и управление передается в то место программы, откуда эта функция была вызвана.

Вызов пользовательской функции происходит следующим образом:

ИмяПеременной=**ИмяФункции**(Параметр(ы))

Еще раз напомним, что при создании собственной функции следует учитывать, что у нее может не быть параметров, и она может не возвращать значение с помощью оператора **return.** В качестве примера напишем функцию, которая выводит сообщение об ошибке.

```
def func(): #Параметры функции отсутствуют
    print("Ошибка")
func() #вызов функции
```

При вызове функции оператором **func**() на экран будет выведено слово «Ошибка».

Задача 1. Разработайте функцию, которая определяет наибольшее число из двух заданных. Ниже приведен код программы, отвечающий за решение задачи.

Листинг 117

```
def maximum(a,b):
   if a>=b:
      max=a
   else:
```

```
    max=b
  return max

a=int(input("\nВведите первое число  "))
b=int(input("\nВведите второе число  "))
rez=maximum(a,b) #Вызов функции
print("\nМаксимальное из двух чисел =", rez)
```

Комментарий. Переменные **a** и **b,** указанные в заголовке функции, называются **позиционными параметрами.** Такие параметры принимают значения только в том порядке, в котором они переданы. Прокомментируем механизм действия функции, изложенный в упрощенном виде, без учета динамической типизации, свойственной Python.

После запуска программы на выполнение начинает свою работу основная часть программы, которая запрашивает у пользователя значения первого и второго чисел (**a** и **b**). Далее вызывается функция **maximum**(a,b) с указанием параметров **a, b,** которые передаются в переменные **a** и **b,** указанные в заголовке функции: **def maximum(a,b).** Далее над ними выполняются действия, записанные в теле функции, а именно, нахождение максимального значения из двух чисел, и управление вновь передается в основную часть программы на оператор rez=**maximum**(a,b). Результат нахождения максимального значения за счет выполнения оператора присваивания оказывается в ячейке **rez** и выводится на экран оператором **print.**

Добавим к вышесказанному, что между параметрами, объявленными в заголовке функции, и параметрами, указанными при вызове функции, должно быть соответствие, а именно, количество параметров должно быть одинаковым.

Задача 2. Разработайте функцию для вычисления суммы квадратов двух чисел и вывода ее на экран.

Комментарий. Рассмотрим на примере этой задачи несколько особенностей использования функций в языке Python. В коде, приведенном в листинге 118, используются значения, переданные по умолчанию. Таким образом, вызов функции **square** не содержит аргументов.

Листинг 118

```
def square(x=5,y=5):
   c=x*x+y*y
   return c
kvadr=square()
print("\nСумма квадратов двух чисел =", kvadr)
```

В следующем примере (листинг 119), относящемся к текущей задаче, используются **именованные аргументы,** которые позволяют передавать значения в любом порядке. Для того чтобы разобраться, как это происходит, мы добавили несколько операторов **print** в тело функции и будем выводить значения квадратов элементов в зависимости от порядка расположения аргументов.

Листинг 119

```
def square(x,y):
    kv1=x*x
    print("\nКвадрат kv1=", kv1)
    kv2=y*y
    print("\nКвадрат kv2=", kv2)
    c=kv1+kv2
    return c
kvadr=square(x=5,y=6)
print("\nСумма квадратов двух чисел =", kvadr)
```

На рис. 132 показана ситуация, когда **x=5**, **y=6.** Переданные значения поступают на вход функции, и соответственно, переменная **kv1** равна 25, а переменная **kv2** равна 36.

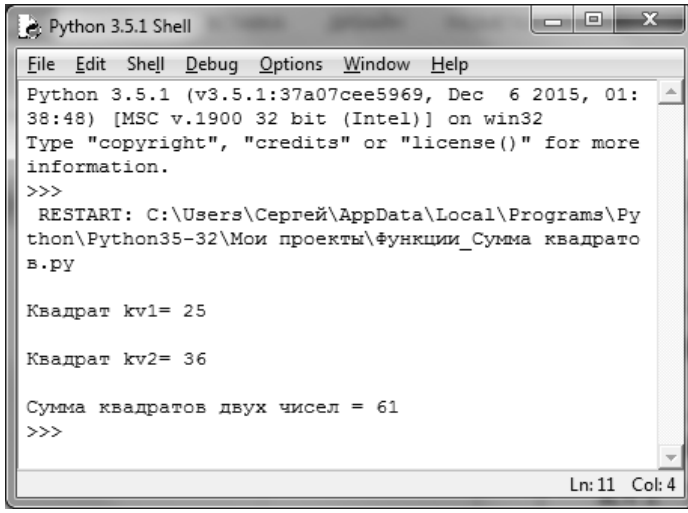


Рис. 132. Результат работы программы

Теперь изменим (листинг 120) при вызове функции порядок аргументов следующим образом: kvadr=**square**(y=5,x=6).

Листинг 120

```
def square(x,y):
        kv1=x*x
        print("\nКвадрат kv1=", kv1)
        kv2=y*y
        print("\nКвадрат kv2=", kv2)
        c=kv1+kv2
        return c
kvadr=square(y=5,x=6)
print("\nСумма квадратов двух чисел =", kvadr)
```

Из рис. 133 видно, что несмотря на изменения в порядке передачи аргументов, значения присваиваются параметрам функции в той последовательности, в которой они были указаны.

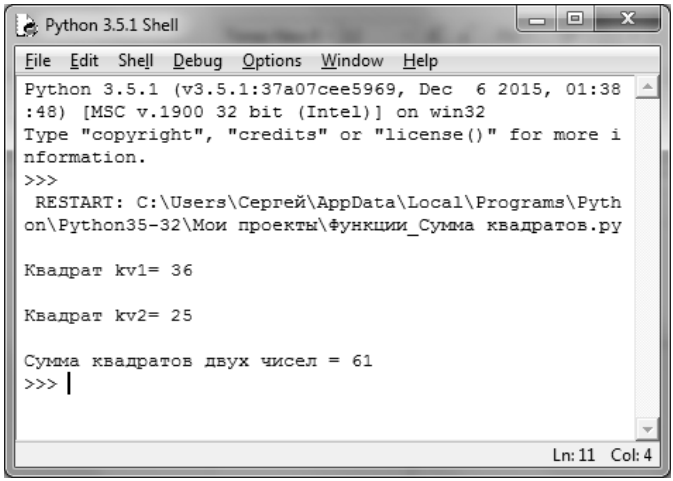


Рис. 133. Результат работы программы

Ранее мы говорили о том, что типы данных, строки, списки являются в Python объектами. Таким образом, в листинге 121 мы в переменной **sslka** оператором **sslka=square** сохраняем ссылку на объект типа «функция».

Листинг 121

```
def square(x,y):
        c=x*x+y*y
        return c
sslka=square
kvadr=sslka(x=5,y=6)
print("\nСумма квадратов двух чисел =", kvadr)
```

Следующей особенностью функций в Python является возможность сделать необязательными некоторые аргументы. Так, если в заголовке функции задать значение одного из параметров, то он становится необязательным (см. листинг 122). При этом следует учесть, что необязательные параметры должны быть перечислены после обязательных. Так, заголовок функции **def square(x=5,y)** вызовет появление сообщения об ошибке.

Листинг 122

```
def square(x,y=6):
        """Сумма квадратов двух чисел""" #Документирование функции
        c=x*x+y*y
        return c
kvadr=square(x=5)
print("\nСумма квадратов двух чисел =", kvadr)
```

Задача 3. Разработайте функцию, суммирующую элементы списка.

Комментарий. Основная проблема заключается в передаче в функцию параметров списка, поскольку количество элементов списка может быть разным. Однако если в заголовке функции перед параметром указать символ *, то в функцию можно передать произвольное количество параметров. Этим способом мы и воспользуемся при решении задачи.

Создав функцию, которая должна возвратить сумму элементов списка, воспользуемся функцией **sample,** которая из исходной последовательности элементов списка будет возвращать указанное пользователем количество элементов. При этом необходимо воспользоваться модулем **random,** подключив его с помощью инструкции **import.** Как показано в нижеследующей программе, исходный список состоит из девяти элементов. Затем оператором **chislo=random.sample(spisok,5)** генерируется пять элементов из исходного списка. Происходит их вывод на экран, и затем осуществляется вызов функции **summa** с параметром ***chislo.** Символ * указывает на то, что происходит так называемая распаковка списка. Код программы представлен в листинге 123.

Листинг 123

```
import random
def summa(*arg):
        s=0
        for i in arg:
            s+=i
        return s
spisok=[10, 20, 30, 40, 50, 60, 70, 80, 90]
chislo=random.sample(spisok,5)
print(chislo, end="      ")
print("\nСумма элементов списка = ", summa(*chislo))
```

Результат работы программы представлен на рис. 134.

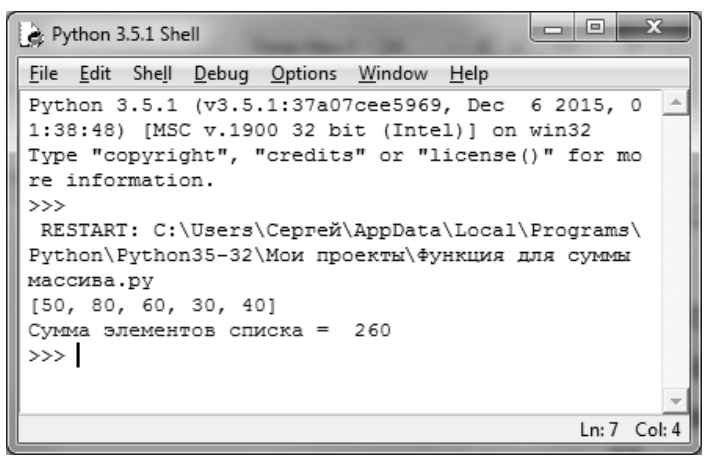


Рис. 134. Результат выполнения программы

Во многих популярных языках программирования, таких как Microsoft Visual Basic, Delphi и др., существует понятие «процедура-подпрограмма». Процедура, в отличие от функции, которая в этих средах программирования может возвращать только одно значение, может вернуть несколько. Может ли функция в Python возвращать несколько значений? Ответ положительный – может. Рассмотрим задачу, которую мы уже решали в параграфе 4.4, но выполним ее с помощью разработки пользовательской функции.

Задача 4. Вычислите значение функции **y**, если:

$$y = \begin{cases} \min(a1, a2, a3), & \text{если } -1 < x < 1 \\ \max\{b1, b2, \min\{c1, c2\}\}, & \text{если } x \geq 1 \\ 1, & \text{если } x \leq -1 \end{cases}$$

Дополнительно выведите на экран номер ветви, по которой производятся вычисления в зависимости от введенных пользователем значений.

Комментарий. Разработанная функция **znach** содержит параметры, указанные в заголовке, а в переменной **n** будет находиться целое число, соответствующее номеру ветви. Обратим внимание на оператор **return**. Для того чтобы функция вернула два значения: значение **y** и значение **n**, мы указываем их через запятую в операторе **return y,n**. Код функции содержится в листинге 124.

Листинг 124

```
def znach(a1,a2,a3,b1,b2,c1,c2,x):
    if x >=1:
        min=c1
        if c2<min:
            min=c2
        max=b1
        if b2>max:
            max=b2
        if min>max:
            max=min
        y=max
        n=2 #Фиксация номера ветви
    elif (-1<x) and (x<1):
        min=a1
        if a2<min:
            min=a2
        if a3<min:
            min=a3
        y=min
        n=1 #Фиксация номера ветви
    else:
        y = 1
        n=3 #Фиксация номера ветви
    return y,n
```

В основной части программы (листинг 125) реализуем ввод данных. Теперь посмотрим, как осуществляется вызов функции. В отличие от предыдущих примеров, где функция возвращала одно значение, мы сейчас указали два значения **rez** и **n,** расположив их через запятую: rez,n=**znach**(a1,a2,a3,b1,b2,c1,c2,x). Далее делаем вывод найденного значения функции и номера ветви программы.

Листинг 125

```
a1 = int(input("Введите значение a1   "))
a2 = int(input("Введите значение a2   "))
a3 = int(input("Введите значение a3   "))
b1 = int(input("Введите значение b1   "))
b2 = int(input("Введите значение b2   "))
c1 = int(input("Введите значение c1   "))
c2 = int(input("Введите значение c2   "))
x = int(input("Введите значение x   "))
rez,n=znach(a1,a2,a3,b1,b2,c1,c2,x) #Вызов функции. Функция возвращает
#несколько значений
print("\nРезультат = ", rez)
print("\nНомер ветви = ", n)
```

В следующем примере будет показан вызов ранее написанной функции другой функцией. Для этого решим следующую задачу.

Задача 5. Создайте программу для вычисления периметра и площади треугольника по заданным координатам трех его вершин.

Задано: x1, y1; x2, y2; x3, y3 – координаты вершин. Требуется определить: **P** – периметр треугольника и **S** – площадь треугольника. Ограничения на значения исходных данных и их соотношения: A>0, B>0, C>0, A+B>C, A+C>B, B+C>A одновременно.

Комментарий. Для решения задачи существуют известные формулы:

$$P = a + b + c;$$

$$s = \sqrt{p_p \cdot (p_p - a) \cdot (p_p - b) \cdot (p_p - c)} \text{ (формула Герона);}$$

$$a = \sqrt{(x1 - x2)^2 + (y1 - y2)^2};$$

$$b = \sqrt{(x3 - x2)^2 + (y3 - y2)^2};$$

$$c = \sqrt{(x3 - x1)^2 + (y3 - y1)^2}.$$

где Pp = p/2– полупериметр; a, b, c – стороны треугольника.

В соответствии с требованиями задания разобьем решение задачи на несколько отдельных задач и создадим соответствующие пользовательские функции:

- функцию **def dl_otr** (), вычисляющую длину отрезка по координатам двух точек;
- функцию **def ps**(), вычисляющую периметр и площадь треугольника.

Код программы представлен в листинге 126.

Листинг 126

```
from math import *
def dl_otr(x1,y1,x2,y2):
        f=sqrt(((x2-x1)*(x2-x1)) + ((y2-y1)*(y2-y1)))
        return f
def ps(x1,y1,x2,y2,x3,y3):
        a=dl_otr (x1, y1, x2, y2) #Вызов функции вычисления длины отрезка
        b=dl_otr (x2, y2, x3, y3) #Вызов функции вычисления длины отрезка
#относительно других значений
        c=dl_otr (x3, y3, x1, y1)
        p=(a + b + c)
        pp = p / 2
        s = sqrt(pp*(pp-a)*(pp-b)*(pp-c))
        return s,p
x1 = int(input("Введите значение x1   "))
y1 = int(input("Введите значение y1   "))
x2 = int(input("Введите значение x2   "))
y2 = int(input("Введите значение y2   "))
x3 = int(input("Введите значение x3   "))
y3 = int(input("Введите значение y3   "))
s,p=ps(x1, y1, x2, y2, x3, y3) #Вызов функции ps
print("\nПериметр треугольника = ", p)
print("\nПлощадь треугольника = ", s)
```

Результат работы программы показан на рис. 135.

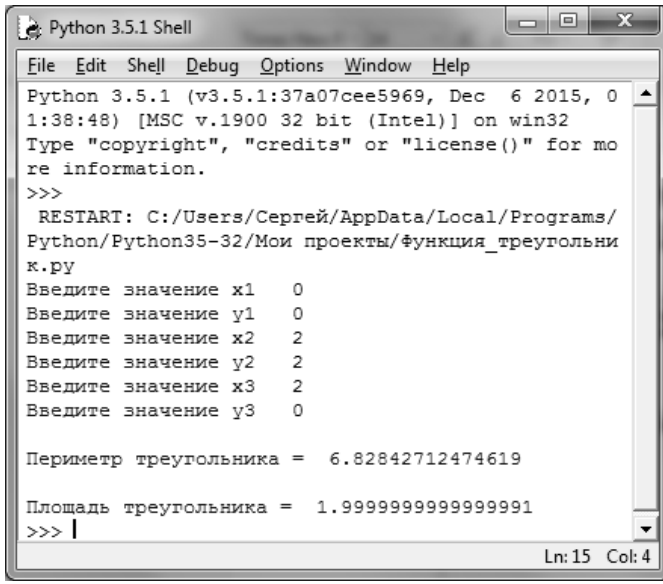


Рис. 135. Результат выполненной программы

9.2. Создание модулей

В предыдущих программах в главе 9 созданные пользовательские функции были доступны только в текущих программах. Например, созданную функцию **def dl_otr**(), вычисляющую длину отрезка по координатам двух точек (Задача 5), можно вызывать только в той программе, где она объявлена, что, безусловно, снижает ее актуальность.

Замечено, что 30–50 % кода в простых приложениях схожи или решают одни и те же задачи, поэтому программисту невыгодно писать один и тот же код. Более профессиональная работа заключается в создании библиотек пользовательских функций, а затем их использовании в программах. В Python такая возможность может быть реализована путем создания **модулей.**

Для сравнения отметим, что в такой свободно распространяемой среде программирования, как Lazarus, или среде программирования Delphi подобный способ реализуется программистом путем создания так называемых библиотек DLL (**Dynamic Link Library** – динамически подключаемая библиотека). Dll-файл – это файл, подключаемый к приложению во время исполнения. Однако в отличие от языка Python, где **модуль** – это обычная программа, сохраненная под своим уникальным именем, в среде Lazarus, Delphi процесс создания Dll-библиотеки содержит определенную последовательность непростых действий (этапов), что требует не только внимательности при программировании, но и временных затрат.

Достоинства модульного построения программ несомненны. Во-первых, как уже было сказано, многие участки программного кода повторяются, и их выгодно оформить в виде функций. Таким образом, очевидно, что, собрав эти функции в единое целое, мы получаем, по сути, их библиотеку. Во-вторых, в современных условиях, программирование не есть удел одиночек, пусть и талантливых, но пытающихся самостоятельно разработать весь функционал своей будущей программной системы. Современные программы создаются коллективами разработчиков, поэтому интегрирование программных модулей, написанных разными группами программистов, объединенные общей задачей – естественный процесс.

Приступим к созданию собственного модуля. Поскольку будущий модуль должен содержать уже разработанную функцию (функции), напишем и отладим программу, которая будет вычислять значение функции **ctg(x).** Встроенная функция нахождения котангенса числа отсутствует в библиотеке математических функций языка Python.

В листинге 127 представлен программный код, в котором написана функция **ctg,** а также осуществляется ее вызов несколько раз.

Листинг 127

```
from math import * #Подключение библиотеки математических функций
def ctg(chislo):
        ctg=cos(chislo)/sin(chislo)
        return ctg
```

```
x=float(input("\nВведите первое число  "))
y=float(input("\nВведите второе число  "))

rez=ctg(x)
rez1=ctg(y)
print("\nПервый вызов функции ctg =", rez)
print("\nВторой вызов функции ctg =", rez1)
```

Убедившись в работоспособности программы, скопируем код функции и, создав новый файл, вставим ее текст в него. Подобная ситуация представлена на рис. 136.

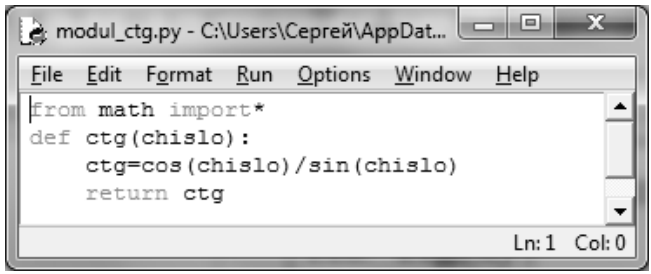


Рис. 136. Текст функции

Выполним команду **File/Save** и сохраним код в текущем каталоге. Лучше всего, если это будет тот каталог, где вы сохраняете программы, написанные на Python. В данном случае файл был сохранен под именем **modul_ctg.py.** Созданный файл можно закрыть.

Что же произошло в результате наших действий? Итогом стало создание **модуля,** который представляет собой обычную программу, написанную на Python. Теперь представим, что мы не предпринимали никаких шагов по созданию программы, а нам сообщили: чтобы воспользоваться вычислением котангенса какого-либо числа, вам нужно подключить в будущей программе модуль, который имеет название **modul_ctg.**

Каким же будет алгоритм использования функции **ctg(x)** в программе? Очень простым. Он будет аналогичен вызову обычной математической функции языка Python.

Листинг 128

```
import modul_ctg
x=float(input("\nВведите первое число  "))
y=float(input("\nВведите второе число  "))
rez=modul_ctg.ctg(x)
rez1=modul_ctg.ctg(y)
print("\nПервый вызов функции ctg =", rez)
print("\nВторой вызов функции ctg =", rez1)
```

В первой строке программного кода (листинг 128) с помощью инструкции **import** мы подключили модуль с именем **modul_ctg.** Ранее эта инструкция ис-

пользовалась нами для подключения стандартных математических функций (**import math**) или для генерации случайных чисел (**import random**). Далее указываем имя модуля, ставим точку и дописываем имя функции. Итак, очевидно, что процесс создания модуля в Python и вызов из него функции не требует использования сложных алгоритмов и большого промежутка времени для реализации задуманного.

9.3. Примеры решения задач

В данном параграфе, помимо новых задач, будут рассмотрены задачи, решения которых уже приводились в предыдущих главах учебного пособия. Это сделано для того, чтобы рассмотреть особенности программирования на основе функций.

Задача 1. Вводится последовательность вещественных чисел. Известно, что последний элемент последовательности равен 5. Разработайте функцию, подсчитывающую количество положительных чисел и минимальное из них. В листинге 129 приведен код программы, отвечающий за решение задачи.

Листинг 129

```
def func(chislo=0):
        kolpol=0
        min_=32767
        while chislo!=5: #Пока число, введенное в цикле, не равно числу
#5, выполнять тело цикла
            chislo = float(input("Введите число   "))
            if chislo>0: #Проверка на положительность очередного
        # введенного числа
                kolpol=kolpol+1
                if chislo<min_:  #Реализация алгоритма поиска
        # минимального элемента
                    min_ = chislo
        return kolpol,min_
k,m=func()
print("Количество положительных чисел =" , k)
print("Минимальное из них =" , m)
```

Задача 2. Из списка заранее заданных целых чисел сформируйте список, состоящий из 5 чисел. Найдите сумму и количество элементов списка, принадлежащих отрезку [0,10]. Подсчет суммы и количества элементов оформите в виде функций.

Разработка алгоритма функции **F_sum,** осуществляющей нахождение суммы чисел, представлена на рис. 137.

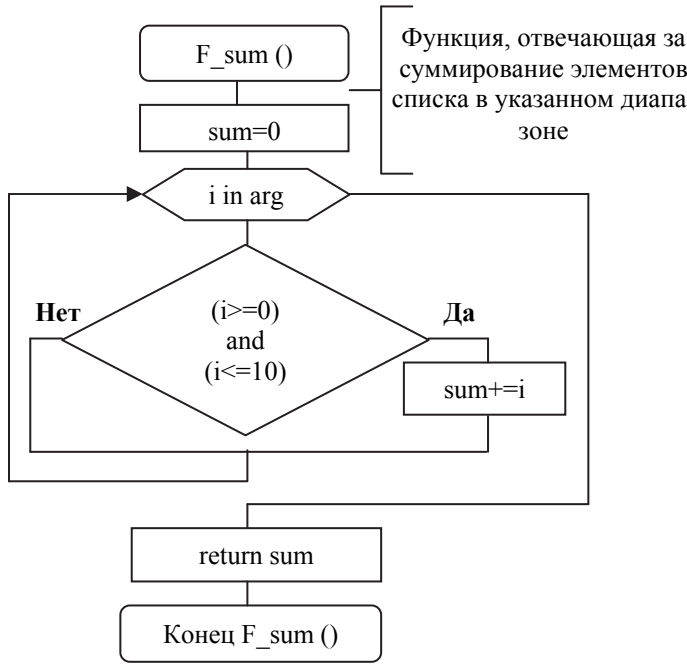


Рис. 137. Алгоритм функции F_sum

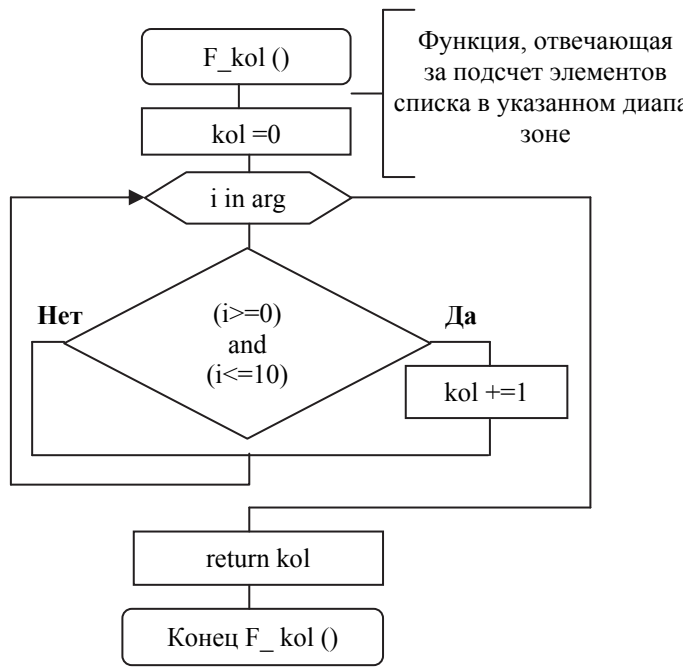


Рис. 138. Алгоритм функции F_kol

Разработка алгоритма функции **F_kol,** осуществляющей нахождение количества чисел, представлена на рис. 138.

Общая блок-схема решения задачи представлена на рис. 139.

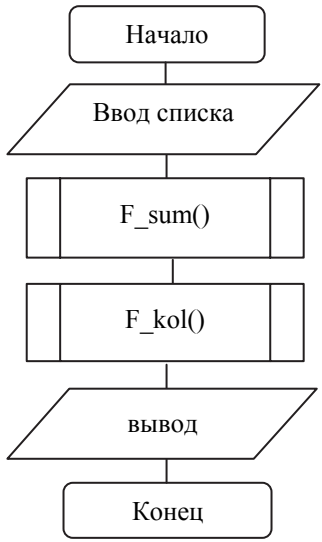


Рис. 139. Общая блок-схема решения задачи

Приведем код функций **F_sum** и **F_kol** в листинге 130.

Листинг 130

```
def F_sum(*arg):
    sum=0
    for i in arg:
        if (i>=0) and (i<=10):
            sum+=i
    return sum

def F_kol(*arg):
    kol=0
    for i in arg:
        if (i>=0) and (i<=10):
            kol+=1
    return kol
```

В листинге 131 приведен основной код программы, в котором происходит вызов ранее созданных функций и вывод на экран результатов вычислений.

Листинг 131

```
spisok=(5, 10, 9, 7, 40, 50, 70, 80, 1, 2)
chislo=random.sample(spisok,5)
print(chislo, end="      ")
```

```
print("\nСумма элементов >=0 и <=10 = ", F_sum(*chislo))
print("\nКоличество таких элементов = ", F_kol(*chislo))
```

На рис. 140 представлен результат работы программы.

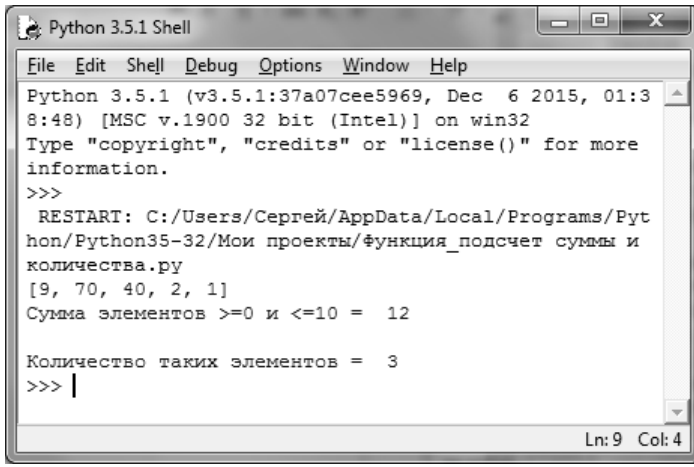


Рис. 140. Результат работы программы

Задача 3. Выполните табулирование (построение таблицы) функции **y=sin(x)**, если известно начальное значение интервала, на котором изменяется функция, конечное значение интервала и шаг ее изменения. Требования к программе:

1. Вычисление функции y=sin(x) оформите в виде пользовательской функции.
2. Задачу табулирования функции также выполните в виде пользовательской функции. Вызовите из нее ранее написанную функцию для вычисления sin(x).

Комментарий. В алгоритме решения задачи табулирования функции используется регулярная циклическая структура, которая в программе реализована оператором **for.** Предварительно в программе происходит вычисление числа повторений цикла по формуле:

$$n=\left[\frac{b-a}{h}\right]+1,$$

где a – начальное значение интервала;
b – конечное значение интервала;
h – шаг.

В листинге 132 представлен код программы табулирования функции.

Листинг 132

```
from math import*
def func(x):
    y=sin(x)
    return y
```

```
def tab(a,b,h):
    x=a
    n=round((b-a)/h)+1 #Вычисление количества повторений
    for i in range (1, n+1):
        z=func(x)
        print("x=  ", x, "  z=  ", '{0:.3f}'.format(z))
        x=x+h
a=float(input("\nВведите начальное значение интервала  "))
b=float(input("\nВведите конечное значение интервала  "))
h=float(input("\nВведите шаг  "))
tab(a,b,h)
```

На рис. 141 представлен результат работы программы.

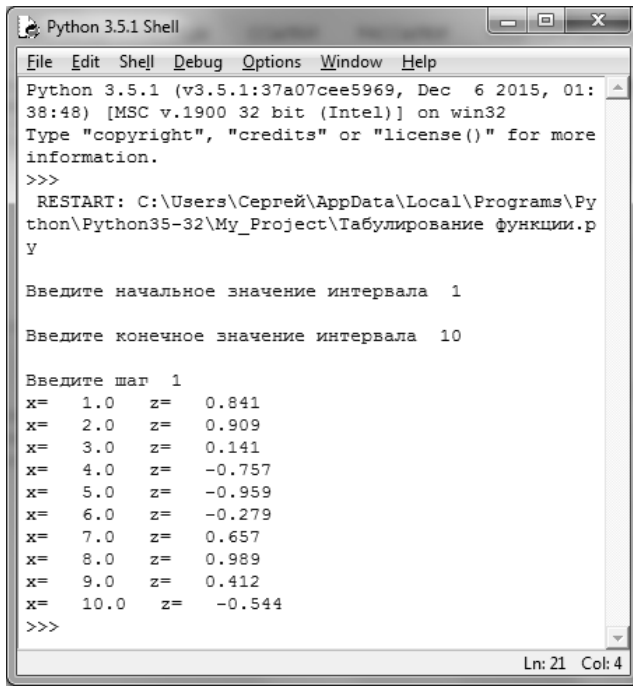


Рис. 141. Результаты табулирования функции

Задача 4. Пользователь вводит исходную строку **stroka** и символ **simvol.** Напишите функцию для подсчета количества повторений заданного символа в исходной строке.

Комментарий. Первоначально после обнуления ячейки **k,** которая будет играть роль счетчика символов, мы преобразуем все символы исходной строки в строчные буквы методом **lower().** Затем в цикле происходит сравнение текущего символа строки **spisok[i]** с искомым символом **simvol.** Параметр цикла **i** будет изменяться от 0 (начало строки) до конца строки (за это отвечает ячейка **n,** в ко-

торой уже находится значение функции **len(spisok)**). Подсчет количества найденных символов в строке обеспечивает оператор **k+=1**.

В листинге 133 приведен код программы, отвечающий за решение задачи.

Листинг 133

```
def func(sp):
    n=len(sp)
    k=0
    for i in range(0,n):
        if spisok[i]==simvol:
            k+=1
    return k

stroka=input("\nВведите строку   ")
stroka1=stroka.lower()
spisok=list(stroka1)
simvol=input("\nВведите символ, количество повторений которого вы хотите найти   ")
kol_simv=func(spisok) #Вызов функции
print("\nКоличество символов равно", kol_simv)
```

Контрольные вопросы

1. Перечислите преимущества, которые получает программист в результате создания собственных функций.
2. Напишите синтаксис, в соответствии с которым создаются пользовательские функции.
3. Каким образом происходит вызов функции?
4. Как называются переменные, которые указываются в заголовке функции? Объясните механизм действия функции на примере.
5. Поясните особенности работы с аргументами функции.
6. Каким образом можно передать в функцию произвольное количество параметров? Приведите примеры.
7. Приведите примеры того, как происходит вызов ранее написанной функции другой функцией.
8. Раскройте особенности модульного построения программ. Обоснуйте достоинства такого способа программирования.
9. Опишите словесный алгоритм создания модулей в языке Python.
10. Каким образом можно подключить созданный пользовательский модуль к программе, написанной на языке Python?

Задачи для самостоятельного решения

1. Список A содержит N чисел. Перепишите из списка A в список B только те элементы, значения которых не равны заданному значению znach. 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 элементов, значения которых не определены. Организуйте запрос: «Сколько элементов списка следует заполнить?» Заполните список заданным числовым значением znach. Назначение функции: заполнение заданного числа элементов заданным значением. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.
3. Список A содержит N чисел. Найдите количество элементов списка, значения которых превышают заданное значение znach. Назначение функции: подсчет количества элементов, превышающих заданное значение. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.
4. Список A содержит N чисел. Найдите сумму значений элементов списка, меньших заданного значения znach. Назначение функции: суммирование элементов, значения которых меньше заданного. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.
5. Список A содержит N чисел. Найдите произведение значений элементов списка. Назначение функции: вычисление произведения элементов списка. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.
6. Подсчитайте количество элементов заданной вложенной последовательности A[i,j], значения которых превышают заданное значение znach. Назначение функции: подсчет количества элементов, значения которых превышают заданное значение. Оформите созданную функцию в виде программного модуля и подключите 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. Задана исходная вложенная последовательность A[i,j]. Умножьте ее на заданное число chislo. Назначение функции: умножение заданной вложенной последовательности на число. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.
9. В заданной вложенной последовательности A[i,j] значения всех элементов, отличающихся от заданного значения znach1, замените другим заданным значением znach2. Назначение функции: замена значений всех элементов, отличающиеся от заданного значения, другим заданным значением. Офор-

мите созданную функцию в виде программного модуля и подключите его к основной программе.

10. Список A содержит N упорядоченных по возрастанию чисел. Вставьте в список некоторое значение znach так, чтобы упорядоченность списка не нарушилась. Назначение функции: вставка заданного значения в упорядоченный список. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.
11. Списки A и B содержат N и M целых чисел, соответственно. Разработайте программу, которая выводит сообщение о том, в каком списке произведение элементов имеет большее значение. Назначение 1 функции: нахождение произведения элементов каждого из данных массивов. Назначение 2 функции: сравнение найденных значений. Оформите созданные функции в виде программного модуля и подключите его к основной программе.
12. 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.
13. Список A содержит N чисел. Заданное число вставьте на место с порядковым номером K. Назначение функции: вставка в список заданного значения на место с заданным порядковым номером. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.
14. Список A содержит N чисел. Найдите в нем элемент с наибольшим значением. Назначение функции: поиск в списке элемента с наибольшим значением. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.
15. Разработайте программу для определения числа сочетаний из n по m. Число определяется по формуле $C_n^m = \frac{n!}{m!(n-m)!}$. Вычисление факториала оформите в виде функции. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.
16. Из заданной вложенной последовательности A[i,j] удалите строку с порядковым номером N_st. Назначение функции: удаление строки по заданному порядковому номеру. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.
17. Разработайте программу, которая заполняет список A[10] случайными целыми числами от 1 до 99. Определите, сколько в нем имеется простых чисел. (Число, которое делится только на единицу или само на себя, называется простым). Процесс определения того, является ли число простым, оформите в виде функции. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.
18. Разработайте программу, в которой осуществляется подсчет количества встречающегося в заданной целочисленной вложенной последовательности A[5,5] максимальное по величине число. Оформите в виде функций: 1. Нахождение максимального элемента. 2. Подсчет числа вхождений максималь-

ного элемента. Оформите созданные функции в виде программного модуля и подключите его к основной программе.

19. Список A содержит N чисел. Значения всех элементов списка, отличающихся от заданного значения znach, замените другим заданным значением znach1. Назначение функции: замена значений элементов списка. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.
20. Заданы вложенные последовательности A и B согласованного размера. Найдите произведение обеих последовательностей. Назначение функции: вычисление произведения двух последовательностей. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.
21. Найдите сумму значений элементов заданной вложенной последовательности A[i,j], меньших заданного значения Znach. Назначение функции: суммирование элементов, значения которых меньше заданного значения. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.
22. Разработайте программу решения группы квадратных уравнений pix2 + qix + ri =0, где p, q, r – списки вещественных чисел, состоящие из k элементов. Решение одного уравнения оформите в виде функции. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.
23. 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.
24. Дана вложенная последовательность n x m. Найдите индексы всех максимальных элементов вложенной последовательности. Назначение функции: нахождение индексов максимального элемента. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.
25. Сформируйте список, каждый элемент которого равен сумме отрицательных элементов соответствующей строки заданной вложенной последовательности. Назначение функции: подсчет суммы отрицательных элементов в каждой строке вложенной последовательности. Оформите созданную функцию в виде программного модуля и подключите его к основной программе.

10. РАБОТА С ФАЙЛАМИ

Как записать информацию в файл и как прочитать информацию из файла? Два вопроса, ответы на которые должен получить начинающий программист, знакомясь с тем или иным языком программирования.

Сначала отметим, файл – это поименованная совокупность любых данных, размещенная на внешнем запоминающем устройстве, хранимая, пересылаемая и обрабатываемая как единое целое. Файл может содержать программу, числовые данные, текст, закодированное изображение и пр. Физически файлы реализуются как участки памяти на внешних носителях, например, на жестких дисках, флэш-носителях и т. д. Обслуживает файлы специальный модуль (программа) операционной системы, называемый драйвером. Он обеспечивает доступ к информации, записанной на носитель, по имени файла и распределяет пространство между файлами.

Для выполнения этих функций драйвер файловой системы хранит не только информацию пользователя, но и свою собственную служебную информацию. В служебных областях носителя хранится список всех файлов и каталогов, а также различные дополнительные справочные таблицы, служащие для повышения скорости работы драйвера файловой системы. Каждый файл должен иметь имя, зарегистрированное в каталоге.

При больших объемах данных повторное введение их при каждом новом запуске программы вызывает большие неудобства, кроме того, модель файла, как отмечено выше, лежит в основе большинства механизмов доступа к устройствам, используемым в подсистемах ввода – вывода. К файловой системе имеет доступ и любая прикладная программа, для этого в языках программирования, в том числе и в Python, имеются специальные средства в виде функций, классов для работы с файлами.

Работа с любым файлом, с точки зрения программирования, состоит из трех этапов:

- открытие файла;
- чтение или запись информации из файла или в файл;
- закрытие файла.

Для каждого из вышеперечисленных типов файлов существуют свои операторы, с помощью которых программируется то или иное действие. Рассмотрим последовательно каждое из них.

10.1. Запись информации в текстовый файл

В соответствии с этапами работы с файлами, перечисленными выше, первое, что нужно сделать, – **открыть файл.** Для этого в Python существует функция **open,** синтаксис которой следующий:

Файловая переменная**=open** (*FileName*, *Mode, Encoding*)

Файловая переменная – переменная, в которой будет находиться значение, возвращенное функцией **open.**

FileName – представляет собой путь к открываемому файлу. Необходимо отметить, что при работе с файлами **необходимо их сохранение в каталогах, не имеющих русскоязычное имя.** В противном случае компилятор будет выдавать ошибку. Например, путь C:\Users**Сергей**\AppData\Local\Programs\Python\Python35-32\My_project содержит имя каталога «Сергей», что недопустимо.

Mode – режим доступа. Может принимать значения, представленные в табл. 8.

Таблица 8. **Возможные значения режима Mode функции open()**

Значение	Описание
"r"	Чтение информации из файла
"w"	Запись информации в файл. Если файл существует, его содержимое полностью заменится; если файл не существует, он будет создан
"a"	Дозапись в текстовый файл. Если файл существует, новые данные будут дописаны в конец. Если файл не существует, он будет создан
"a+"	Дозапись и чтение из текстового файла. Если файл существует, новые данные будут дописаны в конец. Если файл не существует, он будет создан
"r+"	Чтение и запись в текстовый файл. Если файл не существует, возникает исключение IOError
"w+"	Чтение и запись из текстового файла. Если файл существует, он будет перезаписан. Если файл не существует, он будет создан

Encoding – указание кодировки символов. В текстах программ будем указывать кодировку utf-8 (англ. Unicode Transformation Format, формат преобразования Юникода, 8-битный).

Второе действие, которое следует осуществить при работе с файлом, – **запись в него информации.** Для этого в Python предусмотрен метод **write(),** синтаксис которого следующий:

Имя_файловой_переменной.write(данные)

И, наконец, третье действие, выполняемое при работе с файлом, – его **закрытие,** которое выполняется методом **close().** Синтаксис метода:

Файловая_переменная.close()

Теперь приведем пример, в котором покажем, как происходит создание файла и внесение в него информации.

Задача 1. Разработайте программу, которая позволит записать сведения об автомобилях в текстовый файл.

Комментарий. В результате выполнения оператора **zap=open("avto.txt", "w+", encoding='utf-8')** мы создаем в текущем каталоге файл с именем **avto.txt** и открываем его в режиме **w+**, что позволит при отладке программы многократ-

но создавать и перезаписывать файл. Организация цикла **for** i **in range(1,3)** позволит вводить данные о двух автомобилях, указывая их марку, год выпуска и цвет. При этом мы работаем с переменной **zap,** поскольку именно ей присвоили результат работы функции **open.** Закончив выполнение операторов внутри цикла, закрываем файл оператором **zap.close**(). Код программы представлен в листинге 134.

Листинг 134

```
zap=open("avto.txt", "w+", encoding='utf-8') #Открываем файл на запись
for i in range(1,3):
    marka=input("\nВведите марку автомобиля   ")
    zap.write(marka)
    zap.write("\n")  #Перевод курсора на следующую строку
    god=input("\nВведите год выпуска    ")
    zap.write(god)
    zap.write("\n")
    color=input("\nВведите цвет машины     ")
    zap.write(color)
    zap.write("\n")
zap.close() #Закрываем файл
```

Результат работы программы, содержащийся в файле **avto.txt**, показан на рис. 142.

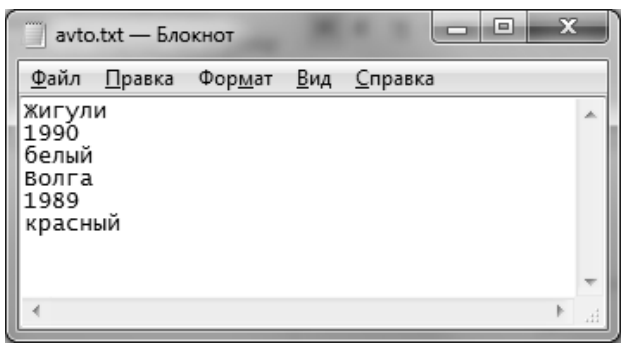


Рис. 142. Содержимое текстового файла

10.2. Чтение информации из текстового файла

Теперь рассмотрим, каким образом прочитать информацию, которая уже находится в текстовом файле. Для этого мы вновь воспользуемся оператором **open,** синтаксис которого останется тем же самым, за исключением значений параметра **Режим доступа.**

Задача 1. Разработайте программу, которая позволит прочитать информацию об автомобилях, содержащуюся в текстовом файле.

В операторе **open("avto.txt", "r", encoding='utf-8')** мы изменили только режим доступа. При записи в файл использовался режим **"w+"**, сейчас **"r",** который, как было сказано выше, используется при чтении информации из файла. Метод **read()** с пустыми круглыми скобками используется по отношению к файловой переменной **chtenie** для чтения **всей** информации из файла. Далее методом **close()** мы закрываем файл. Код программы, с помощью которого мы прочитаем информацию из ранее созданного текстового файла, выглядит так, как он представлен в листинге 135.

Листинг 135

```
chtenie=open("avto.txt", "r", encoding='utf-8')
print (chtenie.read())
chtenie.close()
```

Результат работы программы представлен на рис. 143.

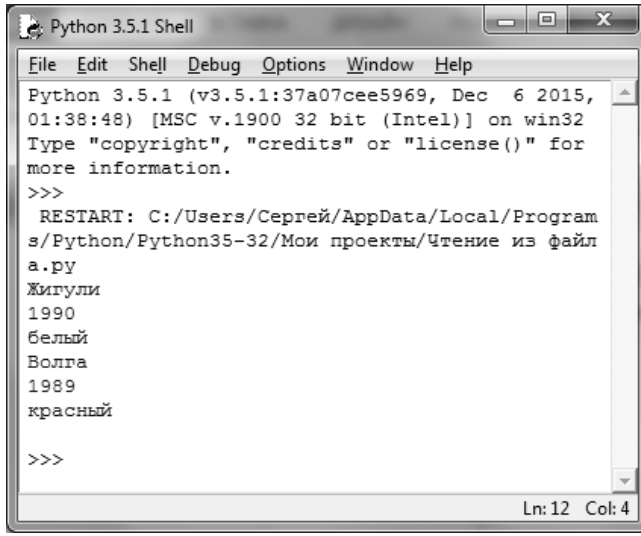


Рис. 143. Информация, прочитанная из файла

При работе с файлами возможно возникновение ошибок, связанных, например, с обработкой пути к файлу. Соответственно, такой тип ошибок можно перехватить, используя обработку исключений. Если предположить, что в указанном ниже примере (листинг 136) ошибочно задан путь к файлу **avto.txt,** то программа выдаст сообщение "Ошибка при открытии файла". Принципы работы конструкции **try**…**except** рассматривались нами в параграфе 2.2.

Листинг 136

```
try:
   chtenie=open("c:\\Primer\\avto.txt", "r", encoding='utf-8')
except:
   print("Ошибка при открытии файла")
```

```
else:
    print (chtenie.read())
    chtenie.close()
```

Если в скобках метода **read()** указать значение, например десять (листинг 137), то с помощью такой записи можно будет прочитать десять символов строки. Вновь использованный в программе метод **read(),** с указанием какого-либо значения, например, пять (**print** (chtenie.**read**(5))), даст возможность прочитать следующие пять символов строки.

Листинг 137

```
chtenie=open("avto.txt", "r", encoding='utf-8')
print (chtenie.read(10)) #читаем десять символов строки
print (chtenie.read(5)) #читаем следующие пять символов строки
chtenie.close()
```

Если нужно прочитать некий текст, содержащийся в файле, построчно, то тогда следует использовать метод, отличие которого от метода **read()** заключается в том, что читать символы можно только в текущей строке. Обратите внимание: метод **read()** в закомментируемом ниже операторе (листинг 138) прочтет содержимое всего файла, между тем, как метод **readline()** прочтет только первую строку (рис. 144).

Листинг 138

```
chtenie=open("avto.txt", "r", encoding='utf-8')
#print (chtenie.read())
print (chtenie.readline())
chtenie.close()
```

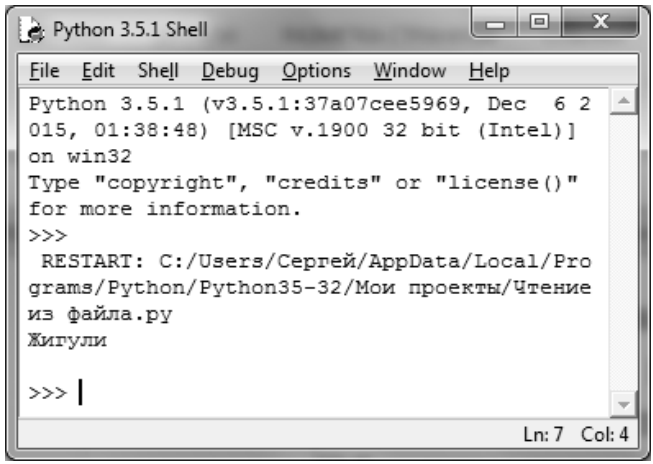


Рис. 144. Построчное чтение информации с помощью метода readline()

10.3. Запись информации в двоичный файл

Двоичные файлы или, как их еще называют, **бинарные файлы,** представляют собой набор байтов, а не набор символов, которые содержатся в текстовых файлах. Бинарные файлы хранят информацию в том виде, в котором она представлена в памяти компьютера, поэтому, открыв такой файл с помощью текстового редактора, мы получим набор нечитаемых символов. Подобная ситуация представлена на рис. 145.

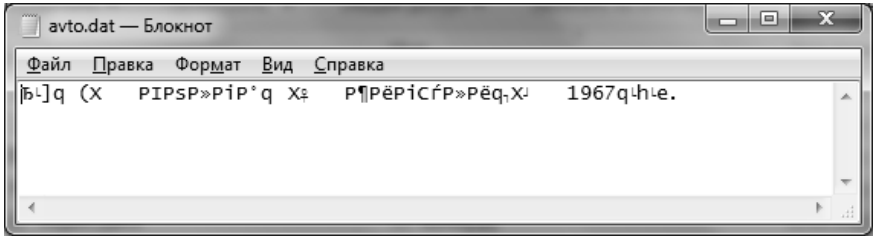


Рис. 145. Содержимое бинарного файла, открытого в редакторе Блокнот

В Python существует такое понятие, как **консервация данных.** Она позволяет хранить в файлах не просто набор символов, а более сложные структуры, например, списки или словари. Такое представление данных позволяет получить представление некоторого объекта в виде набора байтов, причем, сохраненный и переданный на другой компьютер, такой набор может быть расконсервирован, т. е. восстановлен.

Для совершения всех этих операций, в Python предусмотрены два модуля: модуль **pickle** (*от англ. pickling – консервация*) и модуль **shelve** (от англ. ***shelving*** *– стеллаж, размещение на полках*). Соответственно, первый метод позволяет консервировать структуры данных, а второй метод – осуществить доступ к объектам, хранящихся «на полках».

Для того чтобы записать информацию в бинарный файл, его требуется открыть функцией **open,** синтаксис которой следующий:

Файловая переменная**=open** (*FileName*, *Mode*)

Где:

Файловая переменная – переменная, в которой будет находиться значение, возвращенное функцией **open;**

FileName – представляет собой путь к открываемому файлу;

Mode – режим доступа. Может принимать значения, представленные в табл. 9.

Таблица 9. **Возможные значения режима Mode функции open() при работе с бинарными файлами**

Значение	Описание
"rb"	Чтение информации из файла. Если файл не существует, возникает исключение IOError
"wb"	Запись информации в файл. Если файл существует, его содержимое полностью заменится; если файл не существует, он будет создан

"ab"	Дозапись в двоичный (бинарный) файл. Если файл существует, новые данные будут дописаны в конец. Если файл не существует, он будет создан
"rb+"	Чтение и запись в двоичный (бинарный) файл. Если файл не существует, возникает исключение IOError
"wb+"	Чтение и запись из двоичного (бинарного) файла. Если файл существует, он будет перезаписан. Если файл не существует, он будет создан
"ab+"	Дозапись и чтение из двоичного (бинарного) файла. Если файл существует, новые данные будут дописаны в конец. Если файл не существует, он будет создан

Второе действие, выполняемое над бинарным файлом, заключается в непосредственной записи информации в файл. Запись осуществляется с помощью метода **dump(),** имеющего следующий синтаксис:

pickle.dump(данные, файловая_переменная)

Задача 1. Разработайте базу данных, содержащую сведения об автомобилях. О каждом автомобиле известно следующее:

- марка автомобиля;
- цвет автомобиля;
- год выпуска автомобиля.

Требуется создать:

- файл с законсервированной информацией обо всех автомобилях;
- осуществить чтение и расконсервацию данных из файла;
- создать новый файл, в котором организовать размещение списков на полках и осуществить вывод хранящейся информации по указанному пользователем автомобилю.

Комментарий. В самом начале программы мы создадим три списка **avto1, avto2, avto3** в которых находятся сведения о каждом автомобиле: марка, цвет и год выпуска. Четвертый список **avto4** создадим в диалоговом режиме. Далее, оператором **zap=open("avto.dat","wb")** открывается файл с именем **avto.dat** на запись двоичной информации, на что указывает режим **"wb"** (см. табл. 9). С помощью последующих четырех методов **dump()** происходит так называемая **сериализация объектов** (последовательность перевода какой-либо структуры данных в последовательность битов).

В следующей части программы происходит расконсервация данных. Она основана на применении функции **load(),** которая читает данные из файла и преобразовывает их в объект. Ее синтаксис следующий:

Имя_переменной=**pickle.load**(имя_файловой_переменной)

В один файл можно сохранить сразу несколько объектов, что и происходит в программе. Далее оператором **print** выводим списки автомобилей на экран. Код программы представлен в листинге 139.

Листинг 139

```
import pickle
avto1=["Волга","Белый","1970"]
avto2=["Жигули", "Красный", "1980"]
avto3=["Победа","Зеленый", "1950"]
marka=input("\nВведите марку автомобиля   ")
color=input("\nВведите цвет автомобиля   ")
god=input("\nВведите год автомобиля   ")
avto4=[marka,color,god]
zap=open("avto.dat","wb")
pickle.dump(avto1,zap) #консервация данных
pickle.dump(avto2,zap)
pickle.dump(avto3,zap)
pickle.dump(avto4,zap)
zap.close()
#--------------------------------
zap=open("avto.dat","rb+")
avto1=pickle.load(zap)  #расконсервация данных
avto2=pickle.load(zap)
avto3=pickle.load(zap)
avto4=pickle.load(zap)
print(avto1)
print(avto2)
print(avto3)
print(avto4)
zap.close()
```

На рис. 146 представлена информация, извлеченная из файла **avto.dat**.

```
Python 3.5.1 (v3.5.1:37a07cee5969, Dec  5 2015, 21:12:44)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable.
Visit http://www.python.org/download/mac/tcltk/ for current information
.

=== RESTART: /Volumes/KINGSTON/Консервация авто с добавлением элементов.
py ===

Введите марку автомобиля   Тойота

Введите цвет автомобиля   Белый

Введите год автомобиля   1985
['Волга', 'Белый', '1970']
['Жигули', 'Красный', '1980']
['Победа', 'Зеленый', '1950']
['Тойота', 'Белый', '1985']
>>>
```

Рис. 146. Данные об автомобилях

Создадим новый проект Python и, подключив модуль **shelve,** откроем файл **avto1.dat** с помощью функции **open**. Использование оператора **shelve.open** предполагает наличие файла с консервированными объектами.

Теперь мы можем вывести данные по каждому автомобилю, запросив, например, его название. Полка (**polka**) подобна словарю, и сведения, хранящиеся на полке, доступны по соответствующему ключу: "avto1", "avto2", "avto3", "avto4".

Подобно банкам с консервированными огурцами, которые хозяйка размещает для хранения на полках в кухонном шкафу, мы разместим сведения о каждом из автомобилей на виртуальных полках. Поэтому спискам было решено дать соответствующие имена: polka["avto1"], polka["avto2"] и polka["avto3"], polka["avto4"]. Как в словаре, каждому значению, хранящемуся на полке, соответствует ключ, который записывается в квадратных скобках: "avto1", "avto2", "avto3", "avto4". С помощью метода **sync()** объекта-полки, осуществляется запись данных. Код программы представлен в листинге 140.

Листинг 140

```
import shelve
polka=shelve.open("avto1.dat")
polka["avto1"]=["Волга","Белый","1970"]
polka["avto2"]=["Жигули", "Красный", "1980"]
polka["avto3"]=["Победа","Зеленый", "1950"]
marka=input("\nВведите марку автомобиля   ")
color=input("\nВведите цвет автомобиля   ")
god=input("\nВведите год автомобиля   ")
polka["avto4"]=[marka,color,god]
polka.sync()
#---------------------------------
marka=input("\nВведите марку автомобиля   ")
if marka=="Волга":
   print("Данные по машине",marka, polka["avto1"])
if marka=="Жигули":
   print("Данные по машине",marka,polka["avto2"])
if marka=="Победа":
   print("Данные по машине",marka,polka["avto3"])
if marka=="Тойота":
   print("Данные по машине",marka,polka["avto4"])
polka.close()
```

Запрос по автомобилю «Тойота» представлен на рис. 147.

```
Python 3.5.1 (v3.5.1:37a07cee5969, Dec  5 2015, 21:12:44)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: /Volumes/KINGSTON/polki_avto.py ========
===========

Введите марку автомобиля   Тойота

Введите цвет автомобиля   Белый

Введите год автомобиля   1985

Введите марку автомобиля   Тойота
Данные по машине Тойота ['Тойота', 'Белый', '1985']
>>>
```

Рис. 147. Результаты запроса

10.4. Примеры решения задач

Задача 1. Разработайте программу, которая формирует список из пяти случайных элементов. Данные списка консервируются и записываются в двоичный файл. Во второй программе реализуйте чтение списка из файла, создайте функцию, которая осуществляет нахождение суммы сгенерированных элементов списка, находящихся в промежутке от 0 до 10 включительно. Результат выведите в текстовый файл.

В листинге 141 приведен код программы, который осуществляет формирование списка и его консервацию.

Листинг 141

```
import pickle, random
spisok=(5, 10, 9, 7, 40, 50, 70, 80, 1, 2)
chislo=random.sample(spisok,5)
zap=open("mass.dat","wb")
pickle.dump(chislo,zap)
zap.close()
```

В листинге 142 приведен код программы, который осуществляет извлечение списка из файла, нахождение суммы его элементов и запись результата в текстовый файл.

Листинг 142

```
import pickle
zap=open("mass.dat","rb+")
chislo=pickle.load(zap)
print(chislo)
zap.close()
def F_sum(*arg): #Функция подсчитывает сумму элементов
   sum=0
   for i in arg:
```

```
    if (i>=0) and (i<=10):
        sum+=i
  return sum
summa=F_sum(*chislo) #Вызов функции
print("\nСумма элементов >=0 и <=10 = ",summa)
summa_=str(summa) #Преобразование полученного результата в строковое
#значение
f=open("rezult.txt", "w+", encoding='utf-8') #Открываем файл на запись
f.write(summa_)
f.close()
```

Результаты работы программы представлены на рис. 148.

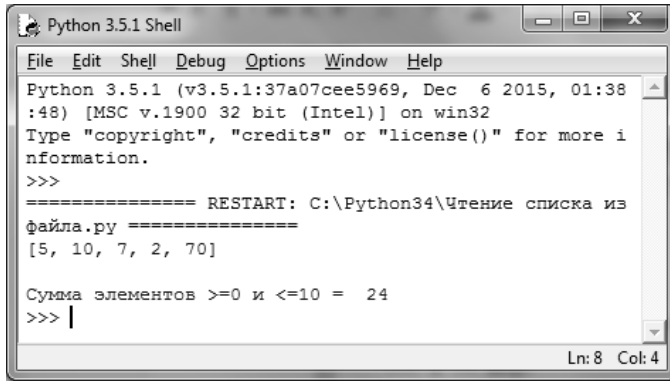


Рис. 148. Осуществлены чтение из файла и нахождение суммы элементов списка, находящихся в соответствующем диапазоне

Задача 2. Разработайте программу, формирующую кортеж из чисел, которые вводит пользователь с клавиатуры. Данные кортежа консервируются и записываются в двоичный файл. Во второй программе осуществите чтение кортежа из файла, создайте функцию, которая осуществляет нахождение минимального из положительных чисел. Результат выведите в текстовый файл.

В листинге 143 приведен код программы, которая осуществляет формирование кортежа и его консервацию.

Листинг 143

```
import pickle
sp=[]  #Объявление кортежа
n=int(input("\nВведите количество будущих элементов кортежа  "))
for i in range(n):
   chislo=int(input("\nВведите число   "))
   sp.append(chislo)
#-----------------------------------------
zap=open("korteg.dat","wb")
pickle.dump(sp,zap)
zap.close()
```

В листинге 144 приведен код программы, которая осуществляет извлечение кортежа из файла, нахождение минимального из положительных чисел и запись результата в текстовый файл.

Листинг 144

```
import pickle
zap=open("korteg.dat","rb+")
chislo=pickle.load(zap)
print(chislo)
zap.close()
#-----------------------------------------
def F_min(*arg): #Функция осуществляет поиск минимального числа
   min = 32767
   for i in arg:
      if i>0: #Проверка на положительность
         if i<min:    #Поиск минимального элемента
            min=i
   return min
min_=F_min(*chislo) #Вызов функции
print("Минимальное число из положительных равно  ", min_)
minim=str(min_) #Преобразование полученного результата в строковое
#значение
f=open("rezult.txt", "w+", encoding='utf-8') #Открываем файл на запись
f.write(minim)
f.close()
```

Результаты работы программы представлены на рис. 149.

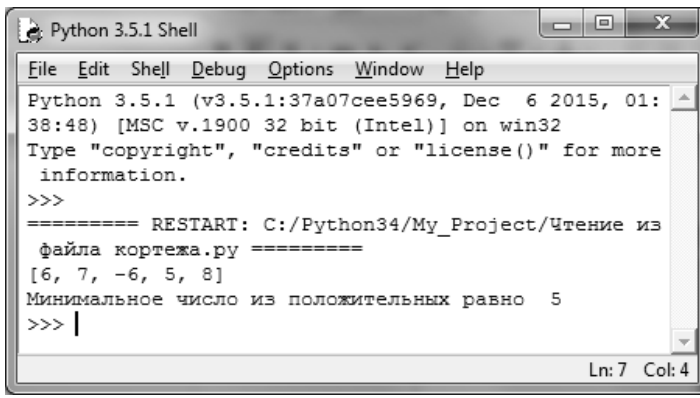


Рис.149. Найден минимальный из положительных элементов

Задача 3. Сформируйте вложенную последовательность произвольных чисел. Данные, находящиеся в последовательности, законсервируйте и запишите в двоичный файл. Далее реализуйте чтение последовательности из файла и осуществите поиск четных элементов в данной вложенной последовательности, создав

соответствующую функцию. Сформируйте из них список, который запишите в текстовый файл.

Комментарий. Итак, в листинге 145 представлен код первой программы, в которой мы формируем вложенную последовательность и, используя модуль **pickle,** консервируем данные в бинарном файле **posled.dat.**

Листинг 145

```
import random, pickle
n=5
m=5
posled=[[i+j for j in random.sample(range(100),5)] for i in random.sample(range(100),5)]
print("\nИсходная последовательность:")
for i in range(n):
    for j in range(m):
        print(" %3d " % posled[i][j],end='')
    print()
#----------Запись в двоичный файл----------
zap=open("posled.dat", "wb")
pickle.dump(posled,zap)
zap.close()
```

На рис. 150 представлены результаты формирования вложенной последовательности.

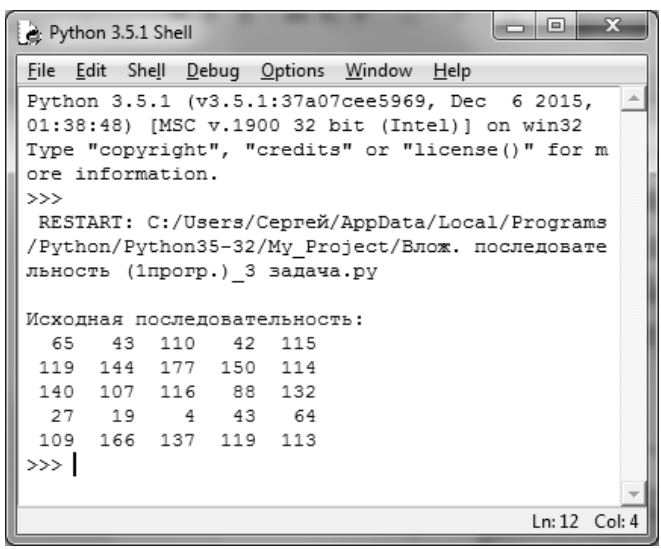


Рис. 150. Сформированная вложенная последовательность

В листинге 146 представлен программный код, в котором мы создаем пустой список оператором **chet=[]**, используемый в дальнейшем для формирования списка четных элементов. С помощью функции **pickle.load()** расконсервируем дан-

ные, хранящиеся в файле **posled.dat.** Функция **F_chet()** реализует поиск четных элементов последовательности оператором **if posled[i][j]%2==0,** а с помощью метода **append()** добавляет найденный элемент в список **chet.**

Вызвав функцию оператором **chet=F_chet(*posled),** выведем сформированный список четных элементов на экран (рис. 151). После этого записываем исходную последовательность и список четных элементов в текстовый файл **result.txt,** предварительно осуществив преобразование данных в строковый тип. Результат записи в текстовый файл показан на рис. 152.

Листинг 146

```
import random, pickle
chet=[]
n=5
m=5
#----------Чтение двоичного файла----------
zap=open("posled.dat","rb+")
posled=pickle.load(zap)
for i in range(n):
   for j in range(m):
      print(" %3d " % posled[i][j],end='')
   print()
zap.close()
#----------Функция вычисления четных элементов----------
def F_chet(*arg):
   for i in range (n):
      for j in range(m):
         if posled[i][j]%2==0:
            chet.append(posled[i][j])
   return chet
#----------Вызов функции и вывод результата----------
chet=F_chet(*posled)
print("\nСписок, состоящий из четных чисел последовательности:")
print(chet)
#----------Запись в текстовый файл----------
posled_=str(posled)
chet_=str(chet)
f=open("result.txt", "w+", encoding='utf-8')
f.write('Исходная последовательность: \n')
for i in range (n):
   f.write(str(posled[i]))
   f.write('\n')
f.write('\nЧетные числа последовательности: \n')
f.write(chet_)
f.close()
```

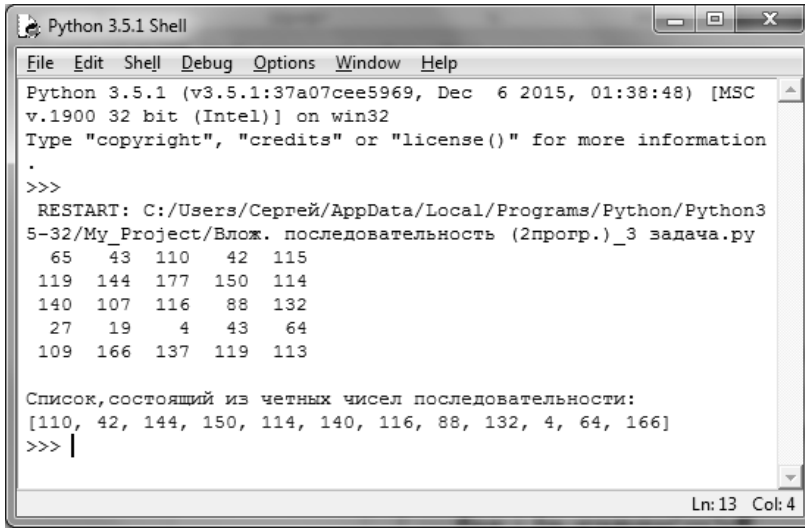


Рис. 151. Чтение данных из файла и формирование списка

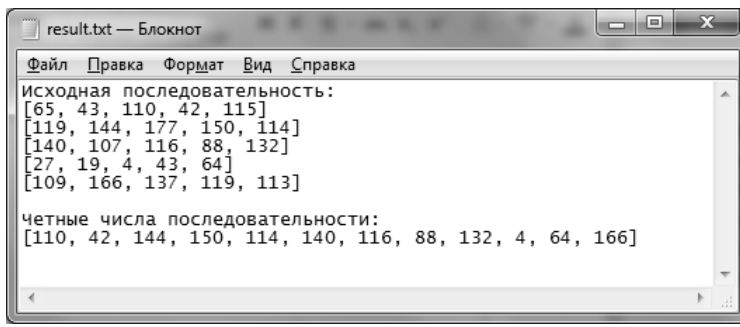


Рис. 152. Данные записаны в текстовый файл

Задача 4. Сформируйте вложенную последовательность A[i,j] случайных чисел в диапазоне [-10;10], в которой осуществите нахождение среднего арифметического и количество элементов, равных нулю.

Заполнение элементами вложенной последовательности, ее вывод, а также нахождение среднего арифметического и количества элементов, равных нулю, оформите в виде функций пользователя. Запишите данные последовательности и результаты вычислений в текстовый и двоичный файлы, создав соответствующие функции. Создайте три модуля, в которых будут находиться функции, отвечающие за следующее:

- генерация, вывод вложенной последовательности, а также нахождение среднего арифметического и количества элементов, равных нулю;
- запись/чтение данных в/из двоичный/го файл/а;
- запись данных в текстовый файл.

Основная программа должна содержать ввод количества строк и столбцов, поступающих на обработку, и вызовы созданных ранее модулей.

Комментарий. В листинге 147 представлен код программы, оформленный в виде модуля и сохраненный под именем **module_sps.py.** Он содержит в себе функцию **rnd(n,m),** отвечающую за генерацию элементов вложенной последовательности, функцию **prnt(a,n,m),** осуществляющую вывод элементов последовательности на экран, а также функцию **task(a,n,m),** выполняющую поиск количества элементов, равных нулю, и нахождение среднего арифметического значения.

Листинг 147

```
import random
#----Заполнение случайными числами----
def rnd(n,m):
    a=[]
    for i in range(n):
        a.append([])
        for j in range(m):
            a[i]=random.sample(range(-10,10),m)
    return a
#--------Вывод вложенной последовательности-----
def prnt(a,n,m):
    for i in range (n):
        for j in range (m):
            print(" %3d " % a[i][j],end='')
        print()
#---Нахождение среднего арифметического и количества нулевых элементов--
def task(a,n,m):
    k=0
    kol=0
    summ=0
    for i in range (n):
        for j in range (m):
            if a[i][j]==0:
                k+=1
            if a[i][j]>0:
                summ=summ+a[i][j]
                kol+=1
    if kol==0:
        print("Нет положительных чисел")
    else:
        srarifm=summ/kol
        print('\nСреднее арифметическое положительных элементов =', srarifm)
        print('\nКоличество элементов, равных нулю =', k)
    return srarifm,k
```

В листинге 148 представлен код модуля, сохраненный под именем **module_txt.py**. В нем содержится функция **prnt_txt(a,n,srarifm,k),** назначением которой является запись исходной последовательности, а также результатов вычислений в текстовый файл **task.txt.**

Листинг 148

```
#-----Запись в текстовый файл----------
def prnt_txt(a,n,srarifm,k):
  txt=open("task.txt","w+",encoding='utf-8')
  for i in range(n):
    txt.write(str(a[i]))
    txt.write('\n')
  txt.write("\nСреднее арифметическое элементов ="+ str(srarifm))
  txt.write("\nКоличество элементов, равных нулю ="+ str(k))
  txt.close()
```

Модуль, сохраненный под именем **module_dat.py,** содержит в себе функцию **prnt_dat(a,n,srarifm,k),** с помощью которой осуществляется запись в двоичный файл **task2.dat** исходной последовательности, а также результатов нахождения среднего арифметического значения и количества элементов, равных нулю. Здесь же создана функция **read_dat(),** отвечающая за вывод списка строк исходной последовательности. Код программы представлен в листинге 149.

Листинг 149

```
import pickle
#--------Запись в двоичный файл----------
def prnt_dat(a,n,srarifm,k):
   dat2=open("task2.dat", "wb")
   pickle.dump(a,dat2)
   pickle.dump(srarifm,dat2)
   pickle.dump(k,dat2)
   dat2.close()
#-------Чтение из двоичного файла-------
def read_dat():
   print("\nПоследовательность из списков строк")
   dat2=open("task2.dat", "rb+")
   a=pickle.load(dat2)
   dat2.close()
   return a
```

По условию задачи основная программа должна содержать в себе вызовы всех ранее созданных модулей. Таким образом, с помощью инструкции **import** мы подключили три ранее созданных модуля для их дальнейшего использования в программе и затем последовательно осуществляем их вызов. Код программы представлен в листинге 150.

Листинг 150

```
import module_sps, pickle, module_dat, module_txt
n=int(input("Введите количество строк: "))
m=int(input("Введите количество столбцов: "))
#-----Обращение к модулю для генерации последовательности----
a=module_sps.rnd(n,m)
#-----Обращение к модулю для вывода последовательности----
module_sps.prnt(a,n,m)
#-----Обращение к модулю для вызова функции---------------
srarifm,k=module_sps.task(a,n,m)
#-----Обращение к модулю для записи в текстовый файл-----
module_txt.prnt_txt(a,n,srarifm,k)
#-----Обращение к модулю для записи в двоичный файл------
module_dat.prnt_dat(a,n,srarifm,k)
#-----Обращение к модулю для чтения из двоичного файла------
b=module_dat.read_dat()
print(b)
```

На рис. 153 показаны результаты работы основной программы, а на рис. 154 результаты, записанные в текстовый файл.

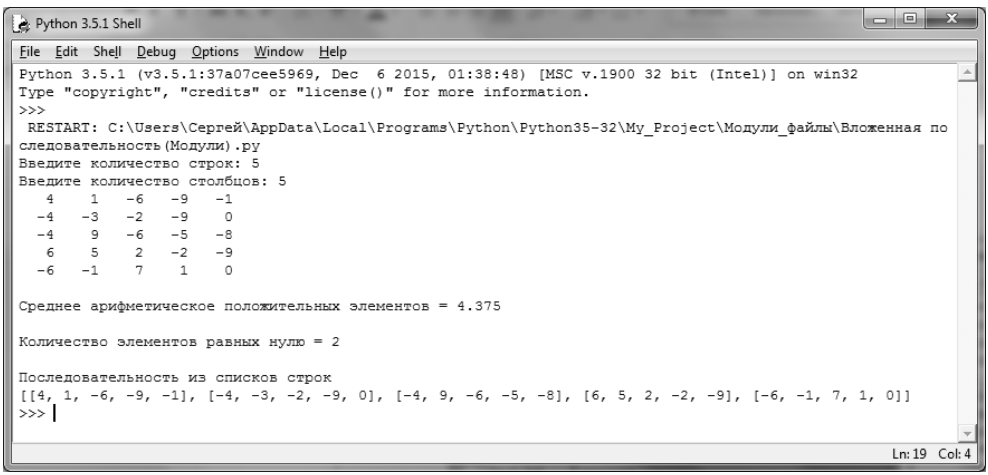


Рис. 153. Результаты выполнения программы

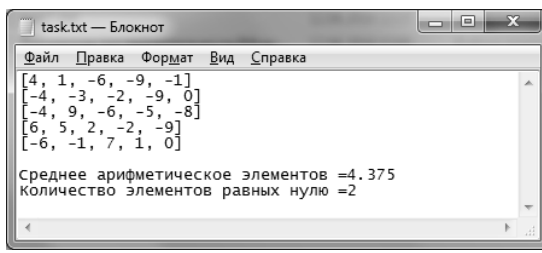


Рис. 154. Результаты, записанные в текстовый файл

Контрольные вопросы

1. Из каких этапов, с точки зрения программирования, состоит работа с любым файлом?
2. Напишите синтаксис функции open(), предназначенной для открытия файла. Поясните назначение параметров функции.
3. Какие возможные значения режима Mode функции open() вы знаете?
4. Напишите синтаксис функции write(), предназначенной для записи информации в файл.
5. Напишите синтаксис функции close(), предназначенной для закрытия файла.
6. Каким образом осуществляется чтение информации из файла? Приведите пример.
7. Объясните, каким образом происходит обработка ошибок, возникающих при работе с файлами. Приведите пример.
8. Раскройте особенности методов read() и readline().
9. Дайте характеристику бинарных файлов.
10. Какова цель консервации данных, используемой в языке Python?
11. Прокомментируйте назначение модулей pickle и shelving.
12. Какая инструкция используется для записи информации в бинарный файл? Напишите ее синтаксис.
13. Какие возможные значения режима Mode функции open(), используемой при работе с бинарными файлами, вы знаете?
14. Каким образом осуществляется запись информации в бинарный файл? Приведите пример программной конструкции.

Задачи для самостоятельного решения

1. Подсчитайте количество сдвоенных символов 'аа', 'оо', 'кк' в тексте, расположенном в текстовом файле, затем удалите повторяющийся символ. Полученную строку запишите в другой файл.
2. Подсчитайте число слов в предложении, записанном в текстовом файле.
3. Найдите в текстовом файле самое длинное и самое короткое слово.
4. Из строки, расположенной в текстовом файле, исключите все символы, входящие в нее более одного раза. Полученную строку сохраните в другом файле.
5. Проверьте, правильно ли расставлены в тексте, расположенном в текстовом файле, круглые скобки. Если неправильно – исправить, а результат сохранить в файле.
6. В последовательности символов, заданной в текстовом файле, подсчитайте общее количество символов '+','-', '*'.
7. В текстовом файле, в предложении, содержащем не менее двух слов, поменяйте местами первое и последнее слова, а затем результат сохраните в другом файле.

8. В текстовом файле две строки текста. Необходимо сформировать третью строку, состоящую из символов, входящих одновременно в обе исходные строки, и дописать ее в текстовый файл.
9. Откорректируйте текст, расположенный в текстовом файле, заменив в нем все вхождения одной буквы на другую. Результат запишите в другой файл.
10. Перепишите текстовый файл таким образом, чтобы все слова исходного текста были перевернуты. Результат запишите в другой файл.
11. В исходном текстовом файле замените все вхождения подстроки F на подстроку Z. Результат запишите в другой файл.
12. Для заданного символа определите, сколько раз он встречается во введенном тексте файла. Результат запишите в другой файл.
13. Из текста, расположенного в файле, исключите группы символов, расположенных между круглыми скобками. Результат запишите в другой файл.
14. Из текста, расположенного в файле, исключите однобуквенные слова. Результат запишите в другой файл.
15. Из текста, расположенного в файле, удалите лишние пробелы, разделяющие слова.
16. Дан текстовый файл, содержащий целые числа. Найдите сумму четных элементов в каждой строке и допишите их в конец файла.
17. Дан текстовый файл, содержащий целые числа. Найдите количество отрицательных элементов в каждой строке. Результат запишите в другой файл.
18. Дан текстовый файл, содержащий целые числа. Найдите номер минимального элемента в каждой строке. Результат запишите в другой файл.
19. Дан текстовый файл, содержащий целые числа. Найдите номер первого (слева направо) четного элемента в каждой строке. Результат запишите в другой файл.
20. Дан текстовый файл, содержащий целые числа. Найдите среднее арифметическое положительных чисел в файле. Результат запишите в другой файл.
21. Дан текстовый файл, содержащий целые числа. Найдите разность максимального и минимального чисел в файле. Результат запишите в другой файл.
22. Даны два текстовых файла. Запишите в третий только те строки, которые есть и в первом, и во втором файлах.
23. Дан текстовый файл. Допишите в него следующие данные: количество строк, количество символов в каждой строке, количество цифр в каждой строке.
24. Дано некоторое конечное множество слов и текстовый файл. Разработайте программу исключения из текстового файла заданных слов.

11. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Мы закончили рассмотрение основ алгоритмизации и программирования на языке Python. Материал, изложенный в предыдущих главах, является фундаментом при изучении любого популярного языка программирования.

Теперь перейдем к обсуждению понятий, которые лежат в основе **объектно-ориентированного программирования** (ООП), представляющего иной подход к решению задач по разработке программного обеспечения, чем, например, технология структурного программирования. Разработка программного обеспечения средствами ООП имеет массу преимуществ, причем это относится не только к созданию более эффективного программного кода, но и к модификации и расширению возможностей уже имеющихся программных систем.

Два основных аспекта объектно-ориентированного программирования – классы и объекты. **Класс** создает новый тип, а **объекты** являются **экземплярами** класса. Аналогично, когда мы говорим о «переменных типа int», это означает, что переменные, которые хранят целочисленные значения, являются экземплярами (объектами) класса **int.**

Объекты могут хранить данные в обычных переменных, которые принадлежат объекту. Переменные, принадлежащие объекту или классу, называют **полями.** Объекты могут также обладать функционалом, т. е. иметь функции, принадлежащие классу. Такие функции в ООП принято называть **методами** класса. Эта терминология важна, так как она помогает нам отличать независимые функции и переменные от тех, что принадлежат классу или объекту. Все вместе (поля и методы) принято называть **атрибутами** класса.

Поля бывают двух типов: они могут принадлежать каждому отдельному экземпляру объекта класса или всему классу. Они называются **переменными экземпляра** и **переменными класса,** соответственно.

11.1. Создание классов

Создание класса имеет следующий синтаксис:

Class ИмяКласса**:**

    Описание атрибутов

Создание простейшего класса показано в листинге 151.

Листинг 151

```
class Student:
    pass #Ключевое слово, необходимое при создании пустого класса

st1=Student() #Создаем экземпляр класса
```

```
st1.name="Сергей" #Создаем атрибуты экземпляра класса
st1.age=18
st1.univer="МТУСИ"
print (st1.name,st1.age,st1.univer) #Обращение к атрибутам класса
```

Как видно из примера, при обращении к атрибутам класса используется следующий синтаксис:

Экземпляр класса.Имя атрибута

На рис. 155 показан результат работы программы.

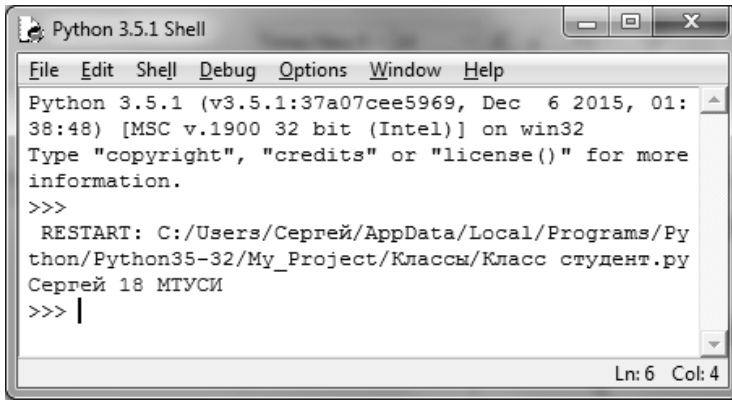


Рис. 155. Результат работы программы

В следующем примере определим **методы класса,** представляющие собой функции. Методы класса имеют одно отличие от обычных функций: они должны иметь дополнительно имя, добавляемое к началу списка параметров. Однако при вызове метода никакого значения этому параметру присваивать не нужно – его укажет Python. Эта переменная указывает на сам объект экземпляра класса и по традиции она называется **self.** Хотя этому параметру можно дать любое имя, настоятельно рекомендуется использовать только имя **self,** использование любого другого имени не приветствуется.

Поясним роль **self** на примере. Предположим, у нас есть класс с именем **MyClass** и экземпляр этого класса с именем **myobject.** При вызове метода этого объекта, например, **myobject.method(a,b),** Python внутренне преобразует такой вызов в **MyClass.method(myobject, a, b).** Это также означает, что если какой-либо метод не принимает аргументов, у него все равно будет один аргумент – **self.**

Итак, создадим класс **Student** (листинг 152). В данном случае он создан с опорой на фундаментальный встроенный тип **object.** В составе класса объявляется метод (функция) **message** с параметром (аргументом) **self.** На этом создание класса заканчивается. Далее создаем новый объект класса **Student** оператором **st1=Student(),** и вызываем метод **message()** оператором **st1.message().**

Листинг 152

```
class Student(object):
        def message(self):
```

```
        print("Здравствуйте, я – студент")
st1=Student()
st1.message() #Обращение к методу класса
```

Как видно из примера, при обращении к методу класса используется следующий синтаксис:

Экземпляр класса.Имя метода

Результат работы программы показан на рис. 156.

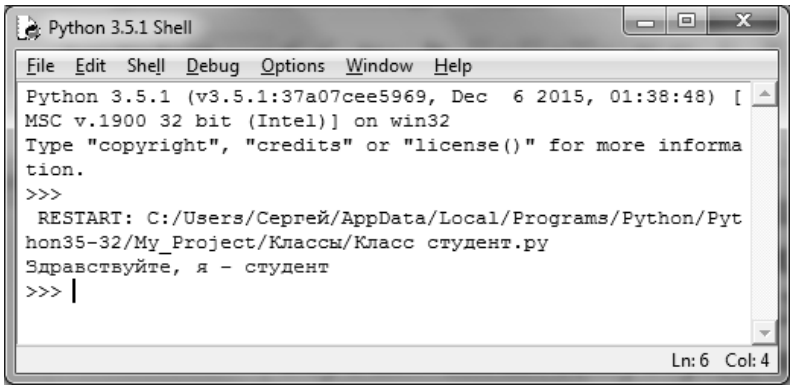


Рис. 156. Результат работы метода massage()

11.2. Создание конструкторов

При создании экземпляра класса создается метод **__init__**, который необходим для инициализации объекта. Такой метод называется **конструктором** (метод-конструктор). Обратите внимание на двойное нижнее подчеркивание слева и справа от ключевого слова **init.**

Его синтаксис следующий:

def __init__(self, значение1, …значениеN)**:**
  инструкции

значение1, …значениеN – необязательные параметры.

В листинге 153 мы определим метод **__init__** так, чтобы он принимал параметр **name** (наряду с обычным **self**). Далее мы создаем новое поле с именем **name.** Обратите внимание, что это две разные переменные, даже несмотря на то, что они обе названы **name.** Это не проблема, так как точка в выражении **self.name** обозначает, что существует нечто с именем **name,** являющееся частью объекта **self,** и другое **name** – локальная переменная. Поскольку мы в явном виде указываем, к какому имени обращаемся, путаницы не возникнет. Оператором st1=**Student**("Сергей") создаем новый объект класса **Student,** у которого атрибут **name** равен строке "Сергей".

Листинг 153

```
class Student:
  def __init__(self, name):
```

```
        self.name = name
    def message(self):
        print("Здравствуйте, я – студент", self.name)
st1=Student("Сергей")
st1.message()
```

Результат работы программы представлен на рис. 157.

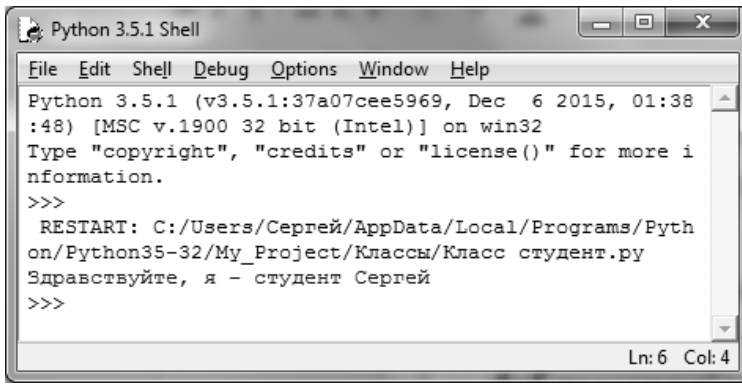


Рис. 157. Результат работы программы

Важно отметить, что при создании нового экземпляра класса мы не вызываем метод **__init__** явным образом, а передаем аргументы в скобках после имени этого класса. В этом и заключается специальная роль данного метода. После этого имеется возможность использовать поле **self.name** в наших методах, что и продемонстрировано в методе **message().** Метод-конструктор **__init__** будет автоматически вызываться при возникновении очередного объекта класса **Student**, что и продемонстрировано в нижеследующем примере (листинг 154). Оператором st2=**Student**("Макс") создается второй экземпляр класса **Student,** а метод **message()** выводит на экран имя второго студента.

Листинг 154

```
class Student:
    def __init__(self, name):
        self.name = name
    def message(self):
        print("Здравствуйте, я – студент", self.name)
st1=Student("Сергей")
st2=Student("Макс")
st1.message()
st2.message()
```

Предположим, мы хотим осуществить подсчет созданных нами студентов. Для этого мы создадим переменную класса **Student,** назвав ее **n** (листинг 155). В терминах ООП она будет называться **атрибутом класса.**

Листинг 155

```
class Student:
    n=0
    def __init__(self, name):
        self.name = name
        Student.n +=1
    def message(self):
        print("Здравствуйте, я – студент", self.name)
    def kol():
        print('У нас {0:d} студента.'.format(Student.n))
    kol=staticmethod(kol)
st1=Student('Сергей')
st2=Student('Макс')
st3=Student('Ростислав')
st1.message()
st2.message()
st3.message()
Student.kol()
```

Результат работы программы представлен на рис. 158.

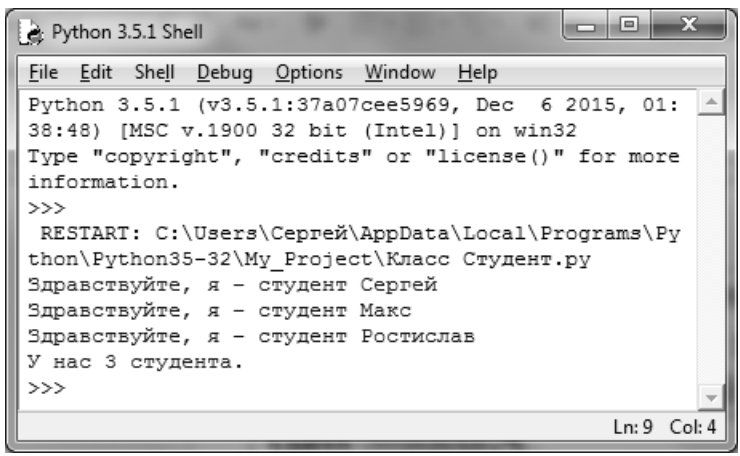


Рис. 158. Результат работы программы

В методе-конструкторе значение атрибута класса увеличивается на единицу оператором **Student.n+=1.** Переменная **name** принадлежит объекту (ей присваивается значение при помощи **self**), и поэтому является переменной объекта. Значение атрибута увеличивается на единицу всякий раз, когда создается новый объект. Таким образом, мы обращаемся к переменной класса **Student** как **Student.n**, а не **self.n.**

Метод **kol()** принадлежит классу, а не объекту. В списке параметров отсутствует параметр **self,** потому что метод **kol()** будет применяться к классу, а не к объекту. Таким образом, методу не нужно передавать ссылку на объект, и пара-

метр для сохранения такой ссылки не нужен. Такой способ определения метода называется **статическим** и определяется как **staticmethod.**

Другим способом объявления статического метода в Python является использование так называемых **декораторов (декорация** – придание чему-либо красивого вида). В этом случае используется команда **@staticmethod,** размещаемая в программе перед методом. С учетом вышесказанного предыдущая программа будет написана так, как это показано в листинге 156.

Листинг 156

```
class Student:
    n=0
    def __init__(self, name):
        self.name = name
        Student.n +=1
    def message(self):
        print("Здравствуйте, я – студент", self.name)
    @staticmethod #Декоратор
    def kol():
        print('У нас {0:d} студента.'.format(Student.n))
st1=Student('Сергей')
st2=Student('Макс')
st3=Student('Ростислав')
st1.message()
st2.message()
st3.message()
Student.kol()
```

Декораторы можно считать упрощенным способом вызова явного оператора, который использовался в предыдущем коде.

11.3. Инкапсуляция

Скрытие внутреннего устройства объектов называется **инкапсуляцией** и является одним из основных принципов ООП. Такой подход позволяет обезопасить внутренние данные (поля) объекта от изменений (возможно, разрушительных) со стороны других объектов; проверять корректность данных, поступающих от других объектов, повышая тем самым надежность программного кода; передавать внутреннюю структуру и код объекта любым способом, не меняя его внешние характеристики (интерфейс), и при этом никакой переделки других объектов не требуется.

По умолчанию все атрибуты и методы объекта являются **открытыми.** Соответственно, из основного кода программы можно использовать значения атрибутов и вызывать методы. Но для реализации инкапсуляции можно сделать атрибуты и методы класса **закрытыми.**

Предположим, при создании класса **Student** мы хотим оставить открытыми атрибуты, отвечающие за имя студента и город, в котором он проживает. Информацию о возрасте студента мы хотим сделать недоступной. Для того чтобы сделать подобный атрибут закрытым, мы начнем запись его в программе с двух нижних подчеркиваний, например, **self.__vozr.** В следующем коде показано, что доступ к закрытому атрибуту внутри объявления класса может быть осуществлен очень просто. Результатом выполнения программы (листинг 157) станет вывод сообщения: «Здравствуйте, я – студент Сергей из Москвы 17».

Листинг 157

```
class Student:
   def __init__(self,name,city,vozr):
      self.name=name  #Открытые атрибуты
      self.city=city
      self.__vozr=vozr  #Закрытый атрибут

   def message(self):
      print("Здравствуйте, я – студент", self.name, "из", self.city, self.__vozr)
st1=Student('Сергей', 'Москвы','17')
st1.message()
```

Продемонстрировать ситуацию, когда закрытие атрибута приводит к ограничению доступа к нему из основного кода, можно на следующем примере. Попытка обратиться к атрибуту **vozr** вне объявления класса **Student** (т. е. просто из командной строки) приводит к ошибке, в которой сообщается о том, что объект **Student** не имеет атрибута **vozr** (рис. 159). Такая же ошибочная ситуация возникнет даже в том случае, если указать атрибут с двумя нижними подчеркиваниями, например, **print**(st1.__vozr).

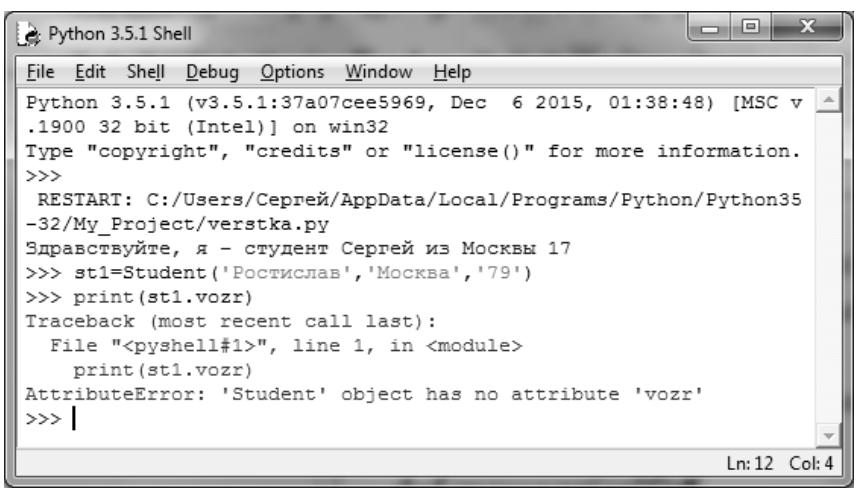


Рис. 159. Сообщение об отсутствии атрибута vozr

Итак, рассказав о методах создания закрытых атрибутов и способах доступа к ним, можно перейти к разговору о создании **закрытых методов.** Для этого дополним код методом **__talk(),** с помощью которого выведем сообщение: "Начнем урок".

```
    def __talk(self):
        print("Начнем урок")
```

Вызовем созданный метод (листинг 158) из метода **message(),** созданного ранее, оператором **self.__talk().**

Листинг 158

```
class Student:
    def __init__(self,name,city,vozr):
        self.name=name  #Открытые атрибуты
        self.city=city
        self.__vozr=vozr  #Закрытый атрибут
    def __talk(self):  #Закрытый метод
        print("Начнем урок")
    def message(self):
        print("Здравствуйте, я – студент", self.name, "из", self.city,
self.__vozr)
        self.__talk()
st1=Student('Сергей', 'Москвы','17')
st1.message()
```

Результат работы программы представлен на рис. 160.

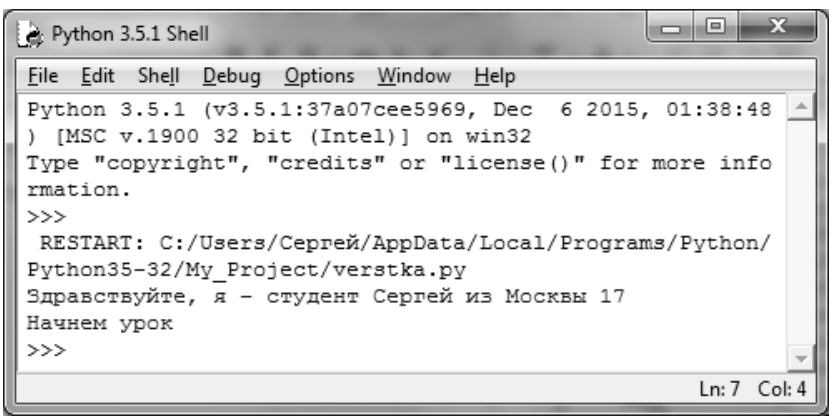


Рис. 160. Результат работы программы. Вызван метод message()

После создания закрытого метода можно продемонстрировать ситуацию, когда из пользовательского кода доступ к такому методу не может быть осуществлен. В частности, обращение к классу **Student** с указанием метода **__talk()** вызовет исключение **AttributeError** с сообщением об отсутствии соответствующего атрибута (рис. 161).

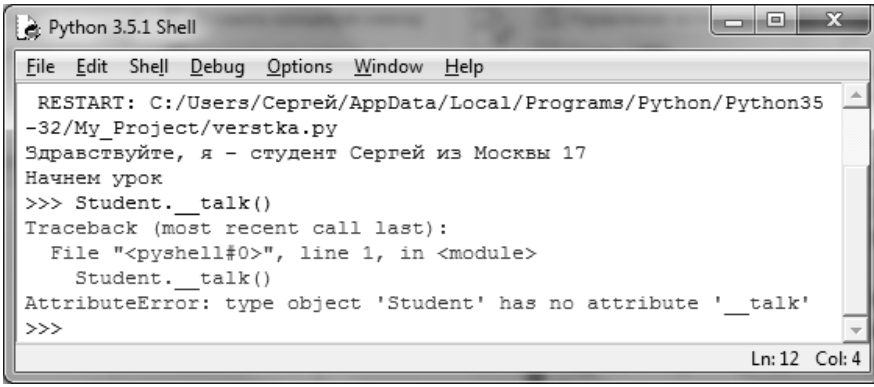


Рис. 161. Сообщение об ошибке

Тем не менее, существует возможность вызвать закрытый метод (или получить значение скрытого атрибута), обратившись к имени класса с помощью следующего синтаксиса:

_ИмяКласса__ИмяАтрибута

Давайте получим доступ к закрытому атрибуту в нашей программе, написав оператор **print(st1._Student__vozr),** и вызовем закрытый метод, оператором **st1._Student__talk().** Подобная ситуация показана на рис. 162.

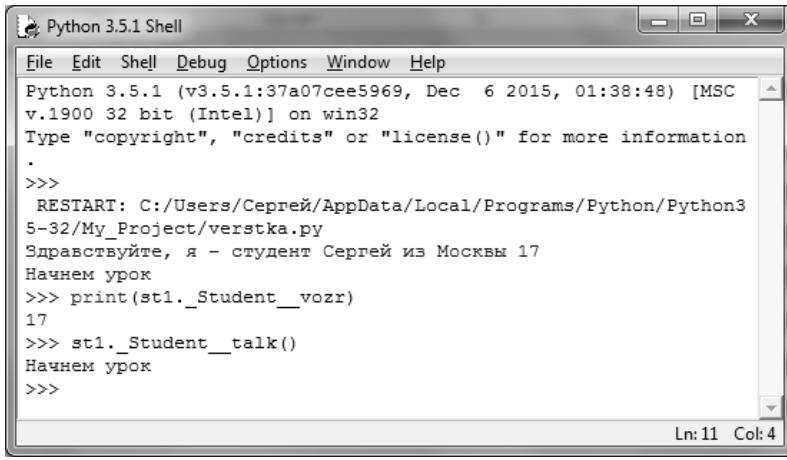


Рис. 162. Вызов закрытого атрибута и обращение к закрытому методу

После работы над созданием закрытых атрибутов и методов может возникнуть естественный вопрос: «А зачем их делать закрытыми, если доступ к ним все равно имеется?» Ответ можно дать такой: во-первых, случайно воспользоваться закрытым атрибутом или методом не удастся, а, во-вторых, инкапсуляция, т. е. ограничение доступа к методам или переменным, в Python существует на уровне договоренности между программистами, поэтому не стоит стремиться закрывать абсолютно все методы и атрибуты созданного класса.

11.4. Создание свойств

Получить доступ к атрибуту, в том числе закрытому, в Python можно с помощью **свойств.** Чтобы создать свойство, следует написать метод, который вернет значение, однако перед тем, как метод будет объявлен, надо написать декоратор **@property.** Продемонстрируем создание свойства на примере, для чего напишем метод **vozr,** с помощью которого получим доступ к закрытому атрибуту **__vozr.**

Метод **vozr** в листинге 159 предваряем декоратором **@property,** а имя свойства будет совпадать с именем метода.

Листинг 159

```
class Student:
    def __init__(self, name,city,vozr):
        self.name=name
        self.city=city
        self.__vozr=vozr
    @property
    def vozr(self):
        return self.__vozr
    def message(self):
        print("Здравствуйте, я – студент", self.name, "из", self.city, self.vozr)

st1=Student('Сергей', 'Москвы','17')
st1.message()
```

Теперь через свойство **vozr** получить доступ к значению атрибута внутри или вне объявления класса. Непосредственно в коде оператором **self.vozr** вызывается метод **vozr,** который вернет атрибут **__vozr.** Обращение к свойству **vozr** из клиентского кода показано на рис. 163. Происходит обращение к методу **vozr**, которое возвратит значение **__vozr,** т. е. число 17.

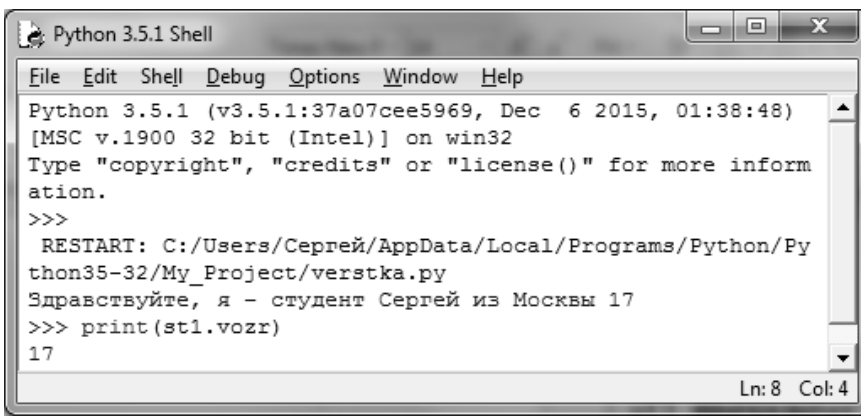


Рис. 163. Результат обращения к методу vozr из клиентского кода

11.5. Наследование

Наследование в ООП предоставляет возможность создавать класс с полями, свойствами и методами, которые могут быть использованы в других классах. В терминах ООП базовый класс называется **наследуемым** (родительским, суперклассом), а производный от него – **наследующим** (подклассом, производным). При наследовании объект производного класса является частным случаем объекта базового класса. Например, автомобиль является частным случаем транспортного средства. В производном классе будет доступ ко всем атрибутам и методам базового класса. Возможности базового класса при этом остаются неизменными.

Так, можно создать базовый класс **Student** и определить в нем метод **information()** (информация). Затем на основе этого класса создать два отдельных класса: **Attendance** (посещаемость) и **Perfomance** (успеваемость). Они используют ту же логику для получения информации, что и класс **Student**, поэтому могут унаследовать от него метод **information().** Это позволяет программисту один раз написать общий код, который затем будет проще поддерживать. Функциональность объектов нового класса можно расширить за счет добавления собственных атрибутов и методов.

Синтаксис производного класса следующий:

```
class Имя_производного_класса(Имя_родительского_класса):
    Оператор1
    Оператор2
    .
    .
    ОператорN
```

В листинге 160 показано создание базового класса **Human** (Человек), а также двух дочерних: **Prepod** (Преподаватель) и **Student** (Студент).

Листинг 160

```
class Human():
    name=""
class Prepod(Human):
    kafedra=""  #Название кафедры
    dolgnost="" #Должность
class Student(Human):
    fakultet=""  #Название факультета
    gruppa=""    #Номер группы
```

Если мы создадим метод в базовом классе **Human,** то он будет наследоваться в методах **Prepod** и **Student,** что и показано в листинге 161.

Листинг 161

```
class Human():
    name=""
    def __init__(self):  #Конструктор базового класса
```

```
    print("Создан класс Человек")
#---------------------------------------
class Prepod(Human):
   kafedra=""  #Название кафедры
   dolgnost="" #Должность
#---------------------------------------
class Student(Human):
   fakultet=""  #Название факультета
   gruppa=""    #Номер группы
#---------------------------------------
h=Human()
pr=Prepod()
st=Student()
```

На рис. 164 представлен результат работы программы, и видно, что при создании объектов вызывается конструктор, унаследованный от базового класса.

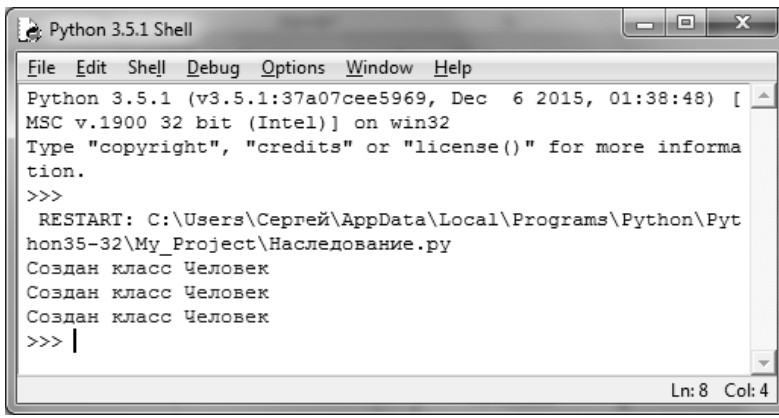


Рис. 164. Три раза вызван конструктор базового класса

В следующем примере (листинг 162) показано, что, если дочерние классы содержат собственные методы, то они и будут выполняться при вызове соответствующих объектов.

Листинг 162

```
class Human():
   name=""
   def __init__(self):  #Конструктор базового класса
      print("Создан класс Человек")
#---------------------------------------
class Prepod(Human):
   kafedra=""  #Название кафедры
   dolgnost="" #Должность
   def __init__(self):  #Конструктор дочернего класса
      print("Создан класс Преподаватель")
```

```
#----------------------------------------
class Student(Human):
   fakultet=""  #Название факультета
   gruppa=""    #Номер группы
   def __init__(self):  #Конструктор дочернего класса
      print("Создан класс Студент")
#------Создание экземпляров классов -------
h=Human()
pr=Prepod()
st=Student()
```

Результат работы программы представлен на рис. 165.

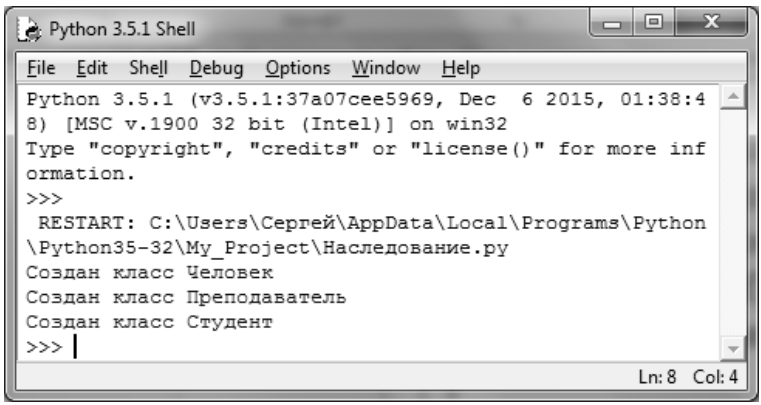


Рис. 165. Вызваны методы дочерних классов

Для того чтобы вызвать конструктор базового класса из конструктора дочернего класса, напишем в каждом из дочерних конструкторов оператор **Human.__init__**(self), так как это показано в листинге 163.

Листинг 163

```
class Human():
   name=""
   def __init__(self):  #Конструктор базового класса
      print("Создан класс Человек")
#----------------------------------------
class Prepod(Human):
   kafedra=""  #Название кафедры
   dolgnost="" #Должность
   def __init__(self):  #Конструктор дочернего класса
      Human.__init__(self)
      print("Создан класс Преподаватель")
#----------------------------------------
class Student(Human):
   fakultet=""  #Название факультета
```

```
    gruppa=""    #Номер группы
    def __init__(self): #Конструктор дочернего класса
        Human.__init__(self)
        print("Создан класс Студент")
#---------------------------------------
h=Human()
pr=Prepod()
st=Student()
```

Тогда результат выполнения программы будет выглядеть следующим образом (рис. 166). Три раза вызвав конструктор базового класса, в котором осуществлялся вывод строки "Создан класс Человек", мы столько же раз получим соответствующий вывод по результатам выполнения программы.

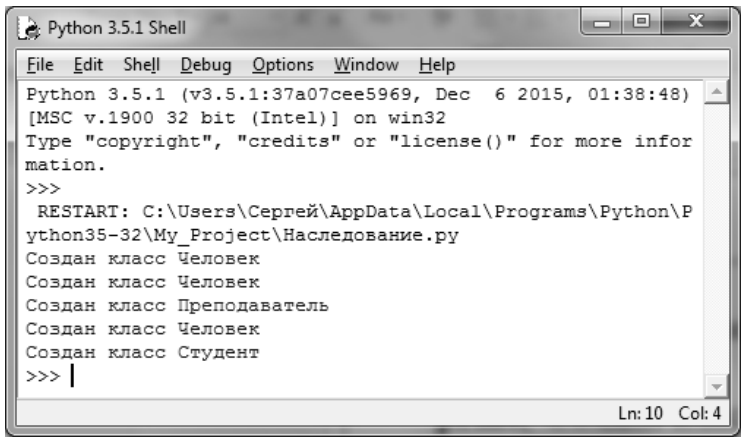


Рис. 166. Вызов конструктора родительского класса

В следующем примере (листинг 164) создан родительский класс **Human** (Человек). В нем реализован метод-конструктор с атрибутом **name** и метод **info.** Два дочерних класса **Prepod** и **Student** будут наследовать родительский класс и, кроме того, обладать атрибутами **kafedra** (Название кафедры), **dolgnost** (Должность) в классе **Prepod** и **fakultet** (Факультет), **gruppa** (Группа) в классе **Student**. В основной части кода метод-конструктор **def __init__** вызывается из каждого дочернего класса оператором **Human.__init__**(self,name), т. е. код родительского класса используется многократно.

Листинг 164

```
class Human():
    def __init__(self,name):
        self.name=name
        print("Создан класс Человек")
    def info(self):
        print(self.name, end=" ")
#---------------------------------------
```

```
class Prepod(Human):
    def __init__(self,name,kafedra,dolgnost):
        Human.__init__(self,name)
        self.kafedra=kafedra  #Название кафедры
        self.dolgnost=dolgnost  #Должность
        print('Создан класс Преподаватель: ',self.name)
    def info(self):
        Human.info(self)
        print('Кафедра:"{0}" Должность:"{1}"'.format(self.kafedra,
self.dolgnost))
#----------------------------------------
class Student(Human):
    def __init__(self,name,fakultet,gruppa):
        Human.__init__(self,name)
        self.fakultet=fakultet #Название факультета
        self.gruppa=gruppa  #Номер группы
        print('Создан класс Студент:  ',self.name)
    def info(self):
        Human.info(self)
        print('Факультет:"{0}" Группа:"{1}"'.format(self.fakultet, self.gruppa))
#----------------------------------------
pr=Prepod('Сергей Гуриков','Информатика','Доцент')
pr.info()
print()
st=Student('Макс Рыжиков','ОТФ-1','БАП1601')
st.info()
```

Результат выполнения программы показан на рис. 167.

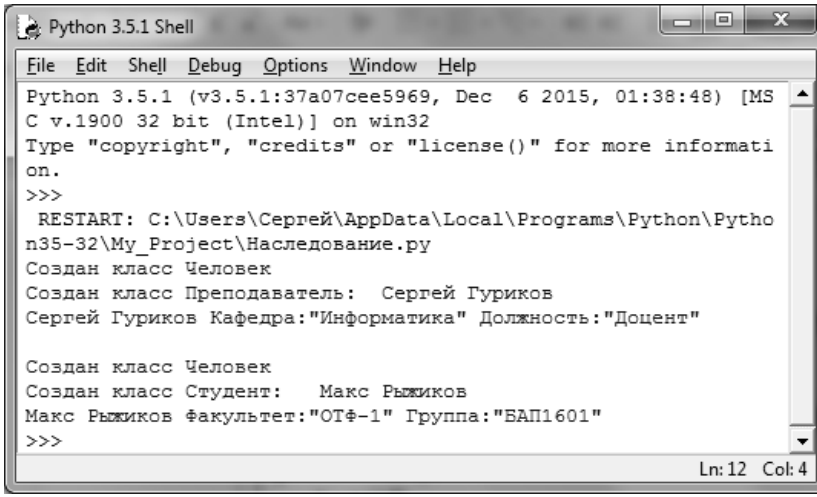


Рис. 167. Результат работы программы

11.6. Примеры решения задач

Задача 1. Создайте класс с методом класса, в котором определялась бы сумма двух целых чисел.

Комментарий. Программа (листинг 165) начинается с объявления класса **Summa,** в нем объявляется метод **summ** с параметром **self.** Напомним, что методу необходимо знать, данные какого объекта ему предстоит обрабатывать, поэтому при создании объекта класса **Summa** параметр **self** становится ссылкой на этот объект. После написания метода **summ** создадим объект **rez1** класса **Summa** оператором rez1=**Summa().** Оператором rez1.**summ()** вызовем метод **summ()** по отношению к объекту класса **Summa,** который связан с переменной **rez1.**

Листинг 165

```
class Summa():
   def summ(self):
      a=int(input('Введите число   '))
      b=int(input('Введите число   '))
      rez=a+b
      print(rez)
rez1=Summa()
rez1.summ()
```

Задача 2. Создайте класс с методом-конструктором, в котором следует определить атрибуты экземпляра класса, необходимые для сложения двух целых чисел. Напишите метод, в котором бы определялась сумма двух целых чисел.

Комментарий. Как уже известно, с помощью метода **__init__** можно присвоить начальные значения атрибутам класса. В листинге 166 строками кода **self.a=chislo1** и **self.b=chislo2** мы создаем атрибуты **a**, **b** у объекта и присваиваем им переменные **chislo1, chislo2.** В созданном далее методе **summ** мы обращаемся к атрибутам **a, b** как **self.a** и **self.b.** Чтобы использовать атрибуты и метод класса, создаем экземпляр класса **rez,** в котором значения первого и второго числа указываются после имени класса **Summa** путем запроса с помощью функции **input.** Вызов метода **summ()** завершает программу.

Листинг 166

```
class Summa():
   def __init__(self,chislo1, chislo2):
      self.a=chislo1
      self.b=chislo2
   def summ(self):
      x=self.a+self.b
      print('Сумма чисел = {0}'.format(x))
rez=Summa(int(input("Введите первое число: ")),int(input("Введите второе число: ")))
rez.summ()
```

В листинге 167 представлен еще один код программы, условие которой изложено в задаче 2. Изменения произошли в методе **summ()** за счет использования оператора **return,** который будет возвращать значение **x,** полученное от сложения двух чисел.

Листинг 167

```
class Summa():
   def __init__(self,chislo1, chislo2):
      self.a=chislo1
      self.b=chislo2
   def summ(self):
      x=self.a+self.b
      return x
rez=Summa(int(input("Введите первое число: ")),int(input("Введите второе число: ")))
print(rez.summ())
```

Задача 3. Создайте класс с методами, формирующими вложенную последовательность. Пользователю должна быть предоставлена возможность заполнить ее либо случайными числами в интервале [-10; 10], либо осуществить ввод данных с клавиатуры.

Комментарий. Метод-конструктор класса инициализирует два атрибута объекта **Formation**: **n** и **m** – количество строк и столбцов. Метод **rnd()** отвечает за генерацию элементов последовательности случайным образом, а метод **manually()** – за ввод элементов последовательности вручную. В методе **prnt()** реализована возможность вывода на экран элементов вложенной последовательности. С помощью метода **clear()** мы переопределим список в его исходное состояние.

В главном методе **main()** пользователю предоставляется возможность определить количество строк и столбцов. Инструкцией ekzemp=**Formation**(n,m) создается новый объект **ekzemp** класса **Formation.** В программе реализована система меню, с помощью которого предлагается выбрать один из методов ввода последовательности. Далее по отношению к экземпляру класса применяются методы **rnd()** или **manually().** В листинге 168 представлен код программы, на рис. 168 – результаты ее выполнения.

Листинг 168

```
import random
class Formation():
   a=[]
   def __init__(self,n,m):
      self.n=n
      self.m=m

   def rnd(self):
      for r in range(self.n):
```

```
            self.a.append([])
            for c in range(self.m):
                self.a[r]=random.sample(range(-10,10),self.m)

    def manually(self):
        for r in range(self.n):
            self.a.append([])
            for c in range(self.m):
                elem=float(input("Введите элемент ("+str(r)+";"+str(c)+"): "))
                self.a[r].append(elem)

    def clear(self):
        self.a=[]

    def prnt(self):
        for r in range(self.n):
            for c in range(self.m):
                print(" %1.2f  " % self.a[r][c],end='')
            print()
def main():
    n=int(input("Введите количество строк: "))
    m=int(input("Введите количество столбцов: "))
    ekzemp=Formation(n,m)
    select = None
    while select != "3":
        print \
    ("""
         Меню для заполнения последовательности
         1 - Случайным образом
         2 - Вручную
         3 - Выход
     """)
        select=input("Выберите способ заполнения последовательности: ")
        print()
        if select=="1":
            ekzemp.rnd()
            ekzemp.prnt()
        elif select=="2":
            ekzemp.clear()
            ekzemp.manually()
            ekzemp.prnt()
        elif select=="3":
            print("Выход из программы")
        else:
            print("Такого пункта меню ",select, " нет", )
```

```
main()
input('\nНажмите клавишу Enter для выхода')
```

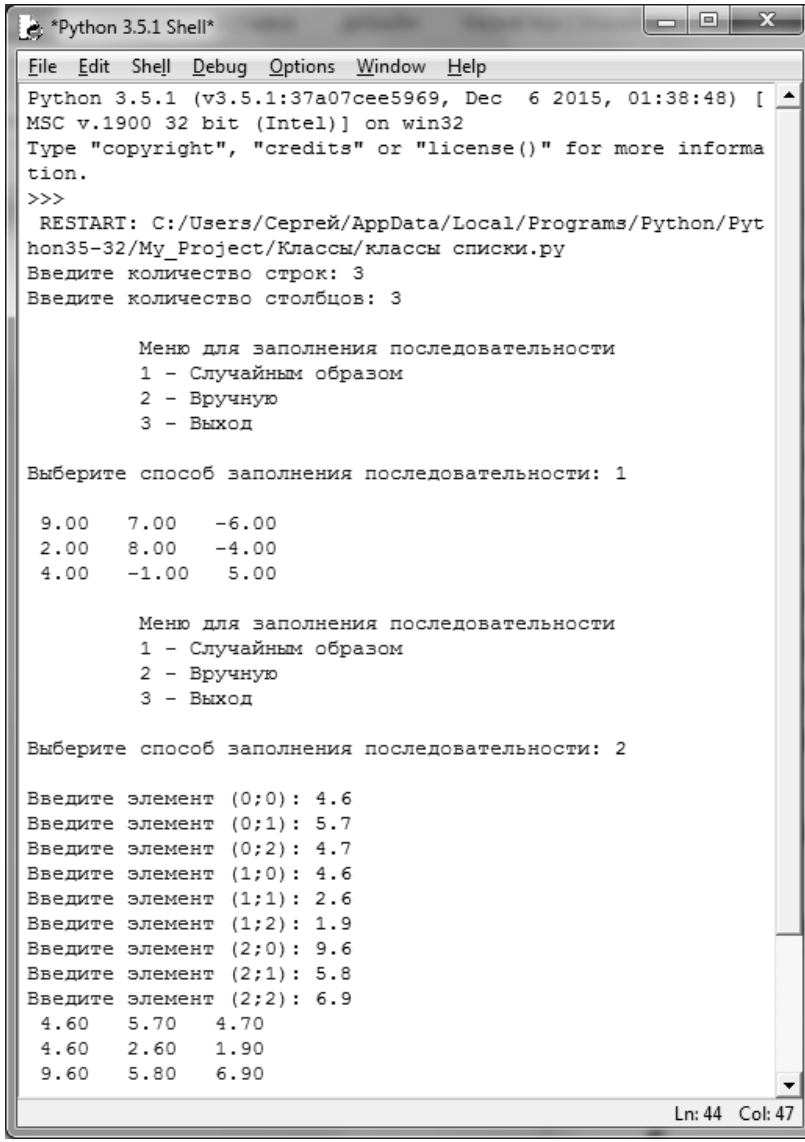


Рис. 168. Формирование вложенной последовательности вручную

Задача 4. Разработайте класс с соответствующими методами, обеспечивающий нахождение значения функции **r** и вывод на экран результатов вычислений. Исходные данные:

$$r = \begin{cases} \sqrt{|x|+|y|+|z|}, & |z|^{x+y} < 3 \\ \min\left\{\sqrt{|x|}, \sqrt{|y|}, \sqrt{|z|}\right., & |z|^{x+y} > 4 \\ \max\left\{x, y^3\right\} + a, & 3 \le |z|^{x+y} \le 4 \end{cases}$$

Комментарий. Реализация нахождения максимального и минимального значений рассматривалась нами параграфе 4.4, поэтому уделим внимание только созданию класса и методам. Первоначально создадим метод инициализации **__init__()** с атрибутами экземпляра класса **x, y, z, a** (листинг 169). В методе **rez()** выполним нахождение значения функции **r** и вывод результатов на экран. Далее, создав экземпляр класса **ekz,** вызовем метод **rez().**

Листинг 169

```
from math import*
class Fun():
   def __init__(self,x,y,z,a):
      self.x=x
      self.y=y
      self.z=z
      self.a=a

   def rez(self):
      if (abs(self.z)**(self.x+self.y))<3:
         r=sqrt(abs(self.x)+abs(self.y)+abs(self.z))
         print('Проверка первой ветви программы\nОтвет:',r)
      elif (abs(self.z)**(self.x+self.y))>4:
         minim=sqrt(abs(self.x))
         if sqrt(abs(self.y))<minim:
            minim=sqrt(abs(self.y))
         elif sqrt(abs(self.z))<minim:
            minim=sqrt(abs(self.z))
         r=minim
         print('Проверка второй ветви программы\nОтвет:',r)
      else:
         maxim=self.x
         if (self.y**3)>maxim:
            maxim=(self.y**3)
         r=maxim+self.a
         print('Проверка третьей ветви программы\nОтвет:',r)
ekz=Fun(float(input('Введите число x ')),float(input('Введите число y
')),float(input('Введите число z ')),float(input('Введите число a ')))
ekz.rez()
input('\nНажмите клавишу Enter для выхода')
```

Задача 5. Создайте класс **ПЕРСОНА** с методами, позволяющими вывести на экран информацию о персоне, а также определить ее возраст (в текущем году). Создайте дочерние классы: **АБИТУРИЕНТ** (фамилия, дата рождения, факультет), **СТУДЕНТ** (фамилия, дата рождения, факультет, курс), **ПРЕПОДАВАТЕЛЬ** (фамилия, дата рождения, факультет, должность, стаж), со своими методами вывода информации на экран и определения возраста. Создайте список из **n** персон, выведите полную информацию из базы на экран, а также организуйте поиск персон, чей возраст попадает в заданный диапазон.

Комментарий. В родительском классе **Persona()** определим, в соответствии с условием задачи, метод **vozrast(),** служащий для определения возраста и метод **info(),** позволяющий вывести информацию о персоне. Далее создаем три дочерних класса: **Abiturient(Persona), Student(Persona), Prepodavatel(Persona),** основанные на классе **Persona().** Соответственно, все дочерние классы будут наследовать методы родительского класса. Чтобы вызвать конструктор базового класса, можно использовать функцию Python – **super().** Заметим, что при использовании функции **super()** можно не передавать в явном виде параметр **self.** Таким образом, вместо инструкции **Persona.__init__(self, name, date, tdate)** в дочернем классе **Student(Persona)** используется инструкция **super().__init__(name,date, tdate).** Подобная ситуация повторяется и в других классах.

В основной части программы мы организуем ввод количества персон и запрашиваем текущий год. В соответствии с количеством персон начинаем осуществлять запрос данных в зависимости от введенной пользователем категории персоны: абитуриент, студент, преподаватель. Создав экземпляры объектов **abitur, stud** и **prepod** классов **Abiturient, Student, Prepodavatel,** передаем им соответствующие значения.

В конце программы осуществляем поиск персон, чей возраст попадает в заданный диапазон. Для этого у нас уже сформирован список персон, хранящийся в соответствующем списке **itogspisok.** Получив критерии **diapazon1** и **diapazon2,** программа благополучно отфильтрует нужные сведения. Полный код программы представлен в листинге 170.

Листинг 170

```
class Persona():
    vozr = 0
    def __init__(self,name,date,tdate):
        self.name=name
        self.date=date
        self.tdate = tdate
    def vozrast(self):
        self.vozr=self.tdate-self.date
    def info(self):
        print("\nСоздан класс Персона")
        print('Фамилия Имя:"{0}" Возраст:"{1}"'.format(self.name,self.vozr))
#------------Создание дочерних классов----
```

```
class Abiturient(Persona):
    def __init__(self,name,date,tdate, fakultet):
        super().__init__(name,date, tdate)
        self.fakultet=fakultet
    def info(self):
        Persona.vozrast(self)
        Persona.info(self)
        print("Создан класс Абитуриент")
        print('Факультет:"{0}"'.format(self.fakultet))

class Student(Persona):
    def __init__(self,name,date,tdate,fakultet,kurs):
        super().__init__(name,date,tdate)
        self.fakultet=fakultet
        self.kurs=kurs
    def info(self):
        Persona.vozrast(self)
        Persona.info(self)
        print("Создан класс Студент")
        print('Факультет:"{0}" Курс:"{1}"'.format(self.fakultet,self.kurs))

class Prepodavatel(Persona):
    def __init__(self,name,date,tdate,fakultet,dolgnost,staj):
        super().__init__(name,date,tdate)
        self.fakultet=fakultet
        self.dolgnost=dolgnost
        self.staj=staj
    def info(self):
        Persona.vozrast(self)
        Persona.info(self)
        print("Создан класс Преподаватель")
        print('Факультет:"{0}" Должность:"{1}"
Стаж:"{2}"'.format(self.fakultet,self.dolgnost,self.staj))
#--------Основная часть программы----------
itogspisok=[]
n=int(input("Введите количество персон: "))
tdate1=int(input("Введите текущий год: "))
for i in range (n):
    persona=input("Кто вы? 1) Абитуриент, 2) Студент, 3) Преподаватель) ")
    name1=input("Введите вашу фамилию и имя: ")
    date1=int(input("Введите год рождения: "))
    spisok=[name1, date1]
    itogspisok.append(spisok)
    if persona=="1":
        fakultet1=input("Введите факультет: ")
```

```
        abitur=Abiturient(name1,date1, tdate1, fakultet1)
        abitur.info()
        print()
    if persona=="2":
        fakultet1=input("Введите факультет: ")
        kurs1=input("Введите курс: ")
        stud=Student(name1,date1,tdate1,fakultet1,kurs1)
        stud.info()
        print()
    if persona=="3":
        fakultet1=input("Введите факультет: ")
        dolgnost1=input("Введите должность: ")
        staj1=input("Введите стаж работы: ")
        prepod=Prepodavatel(name1,date1,tdate1,fakultet1,dolgnost1,staj1)
        prepod.info()
        print()
#---------Организация поиска персон, чей возраст попадает в заданный
#диапазон-------------------
print("Далее будут отобраны персоны, чей год рождения попадает в задан-
ный промежуток")
diapazon1=int(input("Введите начало промежутка: "))
diapazon2=int(input("Введите конец промежутка: "))
print("\nПерсоны, удовлетворяющие заданному условию:")
for i in range (len(itogspisok)):
    if (itogspisok[i][1]>=diapazon1) and (itogspisok[i][1]<=diapazon2):
        print("Фамилия Имя: ", itogspisok[i][0], "Год рождения:
",itogspisok[i][1])
```

Результаты работы программы представлены на рис. 169.

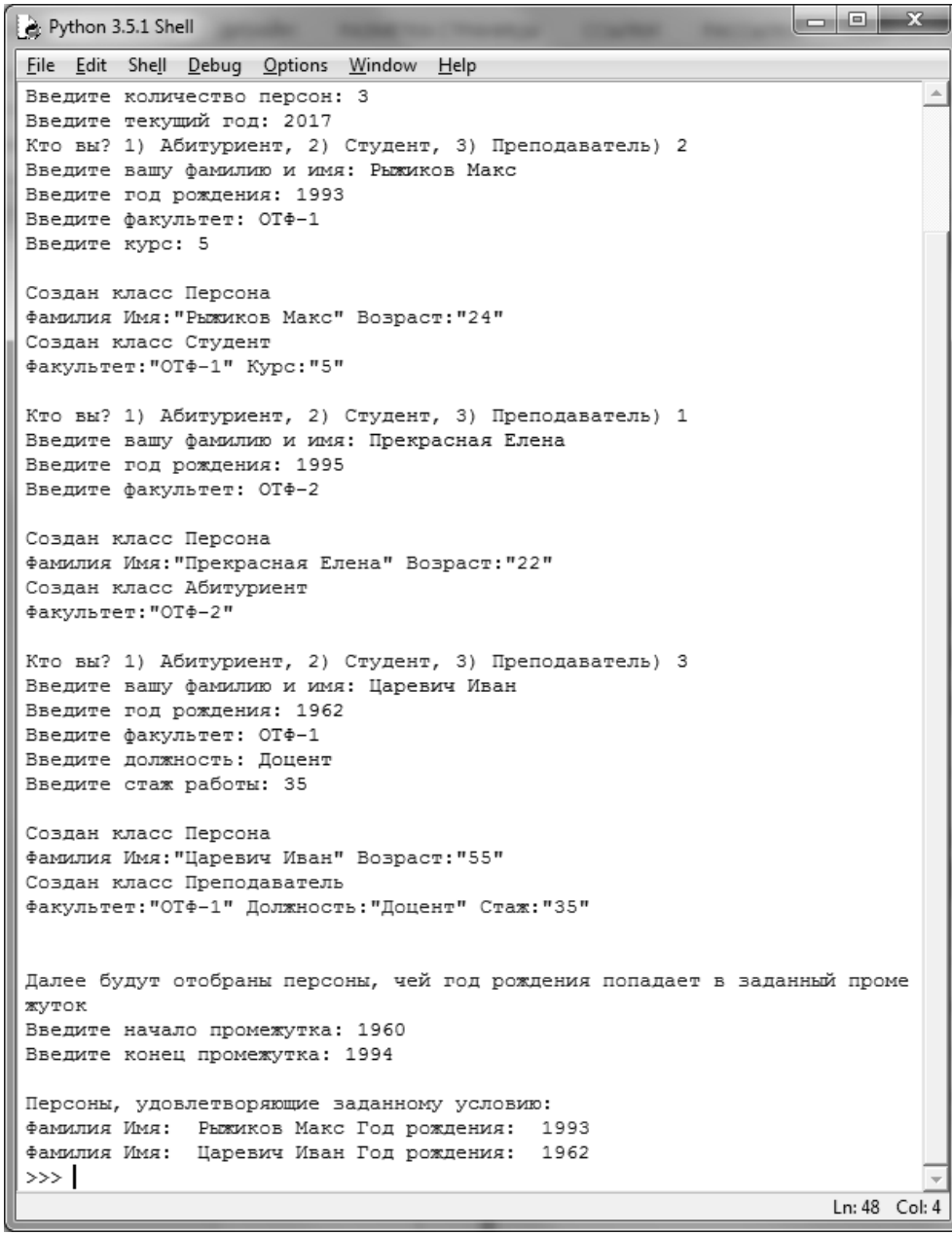


Рис. 169. Результаты работы программы

Контрольные вопросы

1. Назовите и поясните два основных аспекта объектно-ориентированного программирования.
2. Поля, методы, атрибуты – дайте характеристику.
3. Напишите синтаксис создания класса в языке Python.
4. Какой синтаксис используется при обращении к атрибуту класса?
5. Чем методы класса отличаются от обычных функций?
6. Поясните роль параметра self.
7. Какой синтаксис используется при обращении к методу класса?
8. С какой целью создается метод __init__? Напишите его синтаксис.
9. Объясните роль статических методов языка Python. Какие методы объявления статических методов вы знаете?
10. В чем заключается такой принцип ООП, как инкапсуляция?
11. Расскажите о методах создания закрытых атрибутов и способах доступа к ним.
12. С какой целью создаются свойства, и как происходит обращение к ним из клиентского кода?
13. Раскройте особенности одного из основных принципов ООП – наследования. Приведите синтаксис создания производного класса.

Задачи для самостоятельного решения

1. Создайте класс **ФИГУРА** с методами вычисления площади и периметра, а также методом, выводящим информацию о фигуре на экран. Создайте дочерние классы **ПРЯМОУГОЛЬНИК, КРУГ, ТРЕУГОЛЬНИК** со своими методами вычисления площади и периметра. Создайте список **n** фигур и выведите полную информацию о фигурах на экран.
2. Создайте класс **ИЗДАНИЕ** с методом, позволяющим вывести на экран информацию об издании, а также определить, является ли данное издание искомым. Создайте дочерние классы **КНИГА** (название, фамилия автора, год издания, издательство), **СТАТЬЯ** (название, фамилия автора, название журнала, его номер и год издания), **ЭЛЕКТРОННЫЙ РЕСУРС** (название, фамилия автора, ссылка, аннотация) со своими методами вывода информации на экран. Создайте список из **n** изданий, выведите полную информацию из списка, а также организуйте поиск изданий по фамилии автора.
3. Создайте класс **ТРЕУГОЛЬНИК,** заданный длинами двух сторон и угла между ними, с методами вычисления площади и периметра треугольника, а также методом, выводящим информацию о фигуре на экран. Создайте дочерние классы **ПРЯМОУГОЛЬНЫЙ, РАВНОБЕДРЕННЫЙ, РАВНОСТОРОННИЙ** со своими методами вычисления площади и периметра. Создайте список **n** треугольников и выведите полную информацию о треугольниках на экран.

4. Создайте класс **ТРАНСПОРТ** с методами, позволяющими вывести на экран информацию о транспортном средстве, а также определить грузоподъемность транспортного средства. Создайте дочерние классы **АВТОМОБИЛЬ** (марка, номер, скорость, грузоподъемность), **МОТОЦИКЛ** (марка, номер, скорость, грузоподъемность, наличие коляски, при этом если коляска отсутствует, то грузоподъемность равна нулю), **ГРУЗОВИК** (марка, номер, скорость, грузоподъемность, наличие прицепа, при этом если есть прицеп, то грузоподъемность увеличивается в два раза) со своими методами вывода информации на экран и определения грузоподъемности. Создайте список из **n** машин, выведите полную информацию на экран, а также организуйте поиск машин, удовлетворяющих требованиям грузоподъемности.
5. Создайте класс **ТОВАР** с методами, позволяющими вывести на экран информацию о товаре, а также определить, может ли приобрести товар покупатель, имеющий заданную сумму денег. Создайте дочерние классы **ПРОДУКТ** (название, цена, дата производства, срок годности), **ПАРТИЯ** (название, цена за штуку, количество штук, дата производства, срок годности), **ТЕЛЕФОН** (название, цена) со своими методами вывода информации на экран и определения соответствия заданной цене. Создайте список из **n** товаров, выведите полную информацию из базы на экран, а также организуйте поиск товара, который может приобрести покупатель, имеющий заданную сумму денег.
6. Создайте класс **ТОВАР** с методами, позволяющими вывести на экран информацию о товаре, а также определить, предназначен ли он для заданного возраста потребителя. Создайте дочерние классы **ИГРУШКА** (название, цена, производитель, материал, возраст, на который рассчитана), **КНИГА** (название, автор, цена, издательство, возраст, на который рассчитана), **СПОРТИНВЕНТАРЬ** (название, цена, производитель, возраст, на который рассчитан) со своими методами вывода информации на экран и определения соответствия возрасту потребителя. Создайте список из **n** товаров, выведите полную информацию из базы на экран, а также организуйте поиск товаров для потребителя в заданном возрастном диапазоне.
7. Создайте класс **ТЕЛЕФОННЫЙ_СПРАВОЧНИК** с методами, позволяющими вывести на экран информацию о записях в телефонном справочнике, а также определить соответствие записи критерию поиска. Создайте дочерние классы **ПЕРСОНА** (фамилия, адрес, номер телефона), **ОРГАНИЗАЦИЯ** (название, адрес, телефон, факс, контактное лицо), **ДРУГ** (фамилия, адрес, номер телефона, дата рождения) со своими методами вывода информации на экран и определения соответствия заданной фамилии. Создайте список из **n** записей, выведите полную информацию из базы на экран, а также организуйте поиск в базе по фамилии.
8. Создайте класс **КЛИЕНТ** с методами, позволяющими вывести на экран информацию о клиентах банка, а также определить соответствие клиента критерию поиска. Создайте дочерние классы **ВКЛАДЧИК** (фамилия, дата

открытия вклада, размер вклада, процент по вкладу), **КРЕДИТОР** (фамилия, дата выдачи кредита, размер кредита, процент по кредиту, остаток долга), **ОРГАНИЗАЦИЯ** (название, дата открытия счета, номер счета, сумма на счету) со своими методами вывода информации на экран и определения соответствия дате (открытия вклада, выдаче кредита, открытия счета). Создайте список из **n** клиентов, выведите полную информацию из базы на экран, а также организуйте поиск клиентов, начавших сотрудничать с банком в заданную дату.

9. Создайте класс **ПРОГРАММНОЕ_ОБЕСПЕЧЕНИЕ** с методами, позволяющими вывести на экран информацию о программном обеспечении, а также определить соответствие возможности использования (на текущую дату). Создайте дочерние классы **СВОБОДНОЕ** (название, производитель), **УСЛОВНО_БЕСПЛАТНОЕ** (название, производитель, дата установки, срок бесплатного использования), **КОММЕРЧЕСКОЕ** (название, производитель, цена, дата установки, срок использования) со своими методами вывода информации на экран и определения возможности использования на текущую дату. Создайте список из **n** видов программного обеспечения, выведите полную информацию из базы на экран, а также организуйте поиск программного обеспечения, которое допустимо использовать на текущую дату.
10. Создайте класс **ТРАНСПОРТ** с методами, позволяющими вывести на экран информацию о транспортном средстве, а также определить, находится ли транспортное средство в пределах заданных координат. Создайте дочерние классы **САМОЛЕТ** (марка, максимальные скорость и высота, количество пассажиров, координаты), **АВТОМОБИЛЬ** (марка, номер, год выпуска, координаты), **КОРАБЛЬ** (название, координаты, скорость, количество пассажиров, порт приписки) со своими методами вывода информации на экран и определения присутствия транспортного средства в пределах заданных координат. Создайте список из **n** транспортных средств, выведите полную информацию из базы на экран, а также организуйте поиск транспортных средств, которые сейчас находятся в пределах заданных координат.
11. Создайте класс **ИГРУШКА** с методами, позволяющими вывести на экран информацию о товаре, а также определить соответствие игрушки критерию поиска. Создайте дочерние классы **КУБИК** (цвет, цена, материал, размер ребра), **МЯЧ** (цена, цвет, диаметр, материал), **МАШИНКА** (название, цена, производитель, цвет) со своими методами вывода информации на экран и определения соответствия заданному цвету. Создайте список из **n** игрушек, выведите полную информацию из базы на экран, а также организуйте поиск игрушек заданного цвета.
12. Создайте класс **ТЕЛО** с методами вычисления площади поверхности и объема, а также методом, выводящим информацию о фигуре на экран. Создайте дочерние классы **ПАРАЛЛЕЛЕПИПЕД, ШАР, ПИРАМИДА** со своими методами вычисления площади и объема. Создайте список **n** фигур и выведите полную информацию о фигурах на экран.

13. Создайте класс **УРАВНЕНИЕ** с методами вычисления корня уравнения и вывода результата на экран. Создайте дочерние классы **ЛИНЕЙНОЕ, КВАДРАТНОЕ** со своими методами вычисления корней и вывода на экран. Создайте список **n** уравнений и выведите полную информацию об уравнениях на экран.
14. Создайте класс **ВАЛЮТА** с методами перевода денежной суммы в рубли и вывода на экран. Создайте дочерние классы **ДОЛЛАР, ЕВРО** со своими методами перевода и вывода на экран. Создайте список **n** валютных денежных сумм и выведите полную информацию о них на экран.
15. Создайте класс **ПРОГРЕССИЯ** с методами вычисления j-го элемента прогрессии, ее суммы и методом, выводящим сумму на экран. Создайте дочерние классы: **АРИФМЕТИЧЕСКАЯ, ГЕОМЕТРИЧЕСКАЯ** со своими методами вычисления. Создайте список **n** прогрессий и выведите сумму каждой из них экран.

12. СОБЫТИЙНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

В этой главе мы перейдем к созданию программ, в основе которых лежит **графический пользовательский интерфейс** (англ. Graphical User Interface, GUI). Дадим определения некоторых понятий.

Как известно, под **интерфейсом** понимают совокупность средств, обеспечивающих взаимодействие пользователя и программ вычислительной системы. При разработке интерфейса пользователь должен быть заинтересован в максимально простом и удобном способе ввода и вывода данных. Важна и эстетическая форма подачи программы: это и размер формы, ее фон, шрифт надписей, звуковое и графическое сопровождение. Все визуальные объекты, такие как кнопка (**Button**), поле ввода (**Text**), надпись (**Label**) и др., являются объектами. В Python они называются **виджетами.**

Объекты управляются с помощью свойств и методов. **Свойства** – это совокупность характеристик, описывающих объект. Например, можно задать цвет, размер, положение на экране командной кнопки.

Методы – это те действия, которые может выполнить объект. Например, метод **delete** (Удалить) удаляет содержимое текстового окна, метод **insert (Вставить)** позволяет вставить строку в текстовое поле. Многое и в реальной жизни можно описать с помощью объектов, свойств и методов. Например, вы – **объект** под названием Студент. Ваши **свойства:** Факультет, Номер группы, Успеваемость. **Методы,** которые вы можете выполнять: Посещать учебное заведение, Учиться, Уважать преподавателей. Для доступа к свойствам и методам объектов используется следующий синтаксис:

Имя объекта. Свойство или метод

Например:

root.**title**("Создание окна") *#Метод title() изменяет заголовок окна*

Графические интерфейсы пользователя генерируют **события** в ответ на взаимодействие пользователя программы с графическим интерфейсом. Типичными видами взаимодействия являются перемещение мыши, щелчок мыши, щелчок по кнопке и т. д.

В языке Python отсутствует встроенная поддержка создания приложений с графическим интерфейсом, поэтому для создания GUI мы воспользуемся кроссплатформенной графической библиотекой **tkinter** (англ. **tk interface**), которая входит в стандартную библиотеку Python.

12.1. Создание формы и виджетов Кнопка, Текстовое поле, Надпись

Итак, создадим окно Windows, имеющее заголовок и определенные размеры (рис. 170). Для этого напишем программу, код которой представлен в листинге 171.

Листинг 171

```
from tkinter import *
root = Tk()
root.title("Создание окна")
root.geometry("300x200")
root.mainloop()
```

В первой строке программного кода мы объявили модуль **tkinter** инструкцией **from tkinter import*.** Во второй строке происходит обращение к классу **Tk()** библиотеки **tkinter,** и создается базовое окно. Переменную, связанную с объектом-окном, принято называть **root,** хотя делать это не обязательно. Понятно, что в третьей строке мы задали название заголовка окна, а в четвертой – установили его размеры. В последней строке вызываем метод **mainloop()**, чтобы начать событийный цикл базового окна.

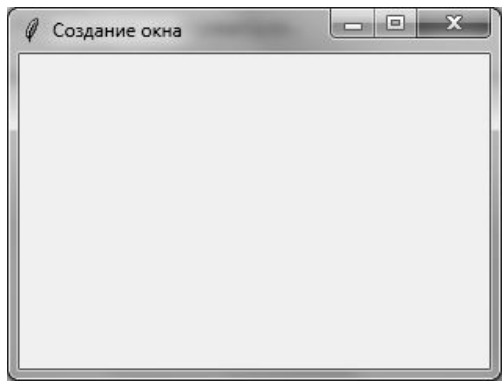


Рис. 170. Созданное окно

Теперь создадим на форме кнопку, при щелчке на которой будет появляться сообщение: «Привет, мир!» (листинг 172). Начало программы совпадает с предыдущим кодом. Далее оператором app=**Frame**(root) создается рамка, внутри которой могут располагаться другие элементы управления (их, как было отмечено ранее, еще называют **виджетами**). Метод **grid()** связан с менеджером размещения, о котором пойдет речь позже. Для того чтобы щелчок на кнопке вызывал сообщение, напишем функцию **vivod().** После этого создаем экземпляр объекта **Button,** в параметрах которого, в частности, указываем высоту и ширину кнопки, ее название, а также привязку к вышеопределенной функции-обработчику **vivod().**

Листинг 172

```
from tkinter import *
root = Tk()
root.title("Создание окна")
root.geometry("300x200")
app=Frame(root)
app.grid()
def vivod():
   print ("Привет, мир!")
btn=Button(app,text="Ok",width=20,height=5,command=vivod)
btn.grid()
root.mainloop()
```

Результат работы программы представлен на рис. 171, 172.

Рис. 171. Виджет «Кнопка», созданный на форме

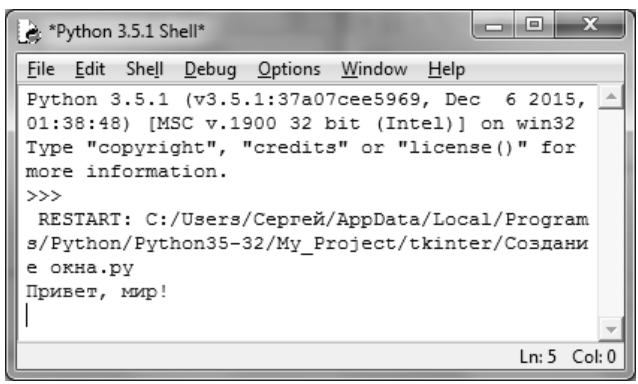


Рис. 172. Результат выполнения щелчка на кнопке

В предыдущей главе пособия мы познакомились с разработкой программ, в основе которых лежат принципы ООП. Теперь эти знания должны пригодиться нам при создании событийно-ориентированных приложений. Перепишем код предыдущей программы, организовав соответствующий класс (листинг 173).

Напомним, что функция **super()** в Python служит для вызова одноименного метода из базового класса. Например, вызов конструктора базового класса **Class1.__init__**(self) можно записать как **super**().**__init__**().

Листинг 173

```
from tkinter import *
class Application(Frame):
    def __init__(self,fr):
        super(Application,self).__init__(fr) #Инициализация рамки
        self.grid()
        self.widget()

    def vivod(self):
        print ("Привет, мир!")

    def widget(self):                #Задаем параметры кнопки
        self.btn=Button(self)
        self.btn["text"]="Ok"
        self.btn["command"]=self.vivod
        self.btn["width"]=30
        self.btn["height"]=10
        self.btn.grid(row=2,column=2)
        self.lbl=Label(self,text="Первое GUI-приложение")
        self.lbl.grid(row=0,column=0)

root = Tk()
root.title("Создание окна")
root.geometry("500x500")
app=Application(root)
root.mainloop()
```

Результат выполнения программы, с точки зрения создания виджетов **label** и **button,** представлен на рис. 173.

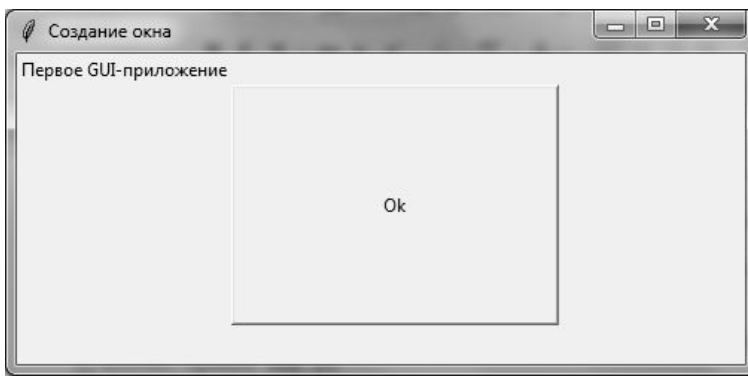


Рис. 173. Результат выполнения программы

В методе **grid(),** который используется в операторах **self.btn.grid(row=2, column=2)** и **self.lbl.grid(row=0,column=0),** указаны два параметра **row** и **col.** Они определяют положение объектов **label** и **button** относительно родительского элемента управления. Рамку базового окна можно представить как сетку, левый верхний угол которой соответствует координатам 0,0. Таким образом, указывая значения **row** и **col,** можно передвигать виджет по сетке. Заметим, что проще попробовать эти действия на практике, чем пытаться виртуально представить некие координаты смещения виджетов.

В следующем примере разработаем программу, реализующую возможность ввода пароля на форму и вывод соответствующих сообщений в случае, если пароль верен или неверен. Таким образом, в нашей будущей программе предполагается создание следующих виджетов: кнопки, двух текстовых окон, в одно из которых будет вводиться пароль, а в другом – появляться соответствующее сообщение о его проверке и метки, в которой будет располагаться приглашение «Введите пароль». Внешний вид программы представлен на рис. 174.

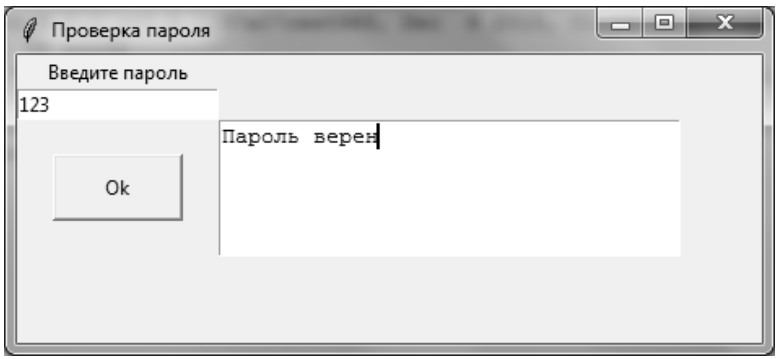


Рис. 174. Интерфейс программы

В нижеприведенном коде (листинг 174) основным является метод **proverka(),** с помощью которого мы получим соответствующее сообщение. В нем находятся два новых для нас оператора: parol=self.**textbox.get()** и self.textbox2.**insert**(0.0,pr). С помощью метода **get(),** который применяется относительно текстового окна, мы «забираем» находящийся в нем текст, т. е. в нашем случае строку с паролем. Во втором операторе метод **insert()** позволит вставить текст, а именно, «Пароль верен» или «Пароль неверен», во второе текстовое окно, поскольку он будет содержаться в ячейке **pr.** Указание значения **0.0** в качестве параметра метода **insert()** отвечает за вставку текста в начало текстового окна.

Листинг 174

```
from tkinter import *
class Application(Frame):
    def __init__(self,fr):
        super(Application,self).__init__(fr)
        self.grid()
        self.widget()
```

```
#Создаем метод для проверки пароля, введенного пользователем
   def proverka(self):
      parol=self.textbox.get()
      if parol=="123":
         pr="Пароль верен"
      else:
         pr="Пароль неверен"
      self.textbox2.insert(0.0,pr)
#Создаем элемент Надпись
   def widget(self):
      self.lbl=Label(self,text="Введите пароль")
      self.lbl.grid(row=0,column=0)
#Создаем текстовое поле для ввода пароля
      self.textbox=Entry(self)
      self.textbox.grid(row=1,column=0)
#Создаем кнопку
      self.btn=Button(self)
      self.btn["text"]="Ok"
      self.btn["command"]=self.proverka #Указываем имя метода
      self.btn["width"]=10
      self.btn["height"]=2
      self.btn.grid(row=2,column=0)
#Создаем текстовое поле для вывода в него ответа
      self.textbox2=Text(self,width=35,height=5)
      self.textbox2.grid(row=2,column=1)

root = Tk()
root.title("Проверка пароля")
root.geometry("500x400")
app=Application(root)
root.mainloop()
```

12.2. Создание виджета Флажок

С помощью флажков (**Checkbox**) пользователь выбирает один или несколько элементов из предложенных ему альтернатив. Разработаем программу на основе тестового вопроса, использующую флажки. Внешний вид формы представлен на рис. 175.

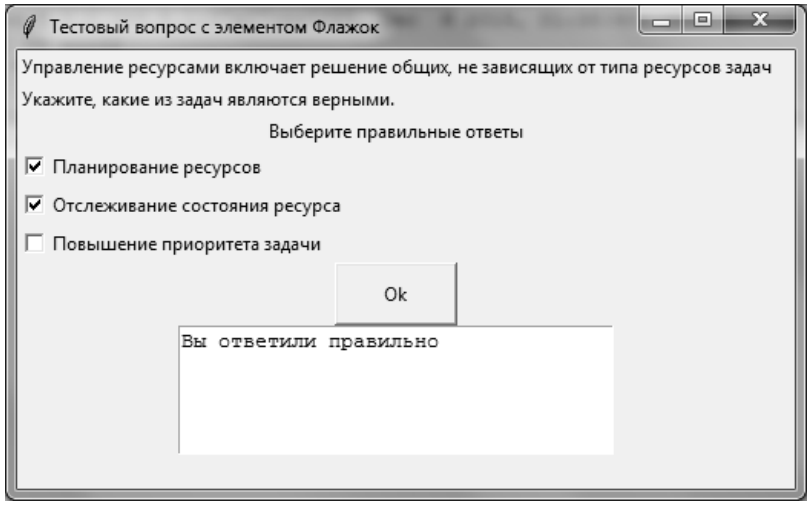


Рис. 175. Тестовый вопрос с элементом управления Флажок

Флажок имеет два состояния: включен – выключен. За текущее состояние флажка в программе будут отвечать атрибут **self.otvet1** для первого флажка, **self.otvet2** для второго флажка и **self.otvet3** для третьего флажка. Для обработки данных разных типов в библиотеке **tkinter** содержатся несколько классов таких, например, как **IntVar()** – для обработки целых чисел, **DoubleVar()** – для обработки дробных чисел, **BooleanVar()** –для обработки логических значений. В нашей программе атрибуты **self.otvet1**, **self.otvet2** и **self.otvet3** мы свяжем с классом **BooleanVar()**, что позволит определять статус флажков, а затем передавать его параметру **znach** каждого флажка.

Как известно логические переменные могут принимать только два значения: **True** (Истина) и **False**(Ложь), поэтому в методе **proverka** мы будем использовать метод **get()**, который и позволит узнать какой из флажков выбрал пользователь. Также в методе **proverka** мы создадим счетчик правильных ответов **n**, который может понадобиться для подсчета правильных ответов, если создавать тестовую программу, состоящую из нескольких вопросов.

Параметр **sticky** позволяет выровнять элемент управления относительно ячейки, расположенной на сетке менеджера размещения. При этом могут быть использованы разные параметры: N – вверх, S – вниз, E – вправо, W – влево. Ниже приведен программный код (листинг 175), отвечающий за создание интерфейса и осуществление выбора флажков.

Листинг 175

```
from tkinter import *
class Application(Frame):
    def __init__(self,fr):
        super(Application,self).__init__(fr)
        self.grid()
        self.widget()
```

```
   def proverka(self):
      n = 0
      if self.otvet1.get() and self.otvet2.get(): #Если пользователь выбрал
# первый и второй флажок
         pr="Вы ответили правильно"
          n+=1  #Увеличиваем счетчик на единицу
      else:
         pr="Вы ответили неправильно"
      self.textbox2.delete(0.0, END)
      self.textbox2.insert(0.0,pr)

   def widget(self):
      Label(self,
          text = "Управление ресурсами включает решение общих, не за-
висящих от типа ресурсов задач"
          ).grid(row = 0, column = 0, sticky = W)

      Label(self,
          text = "Укажите, какие из задач являются верными."
          ).grid(row = 1, column = 0, sticky = W)

      self.lbl=Label(self,text="Выберите правильные ответы")
      self.lbl.grid(row=2,column=0)

      self.otvet1 = BooleanVar()
      Checkbutton(self,
                text = "Планирование ресурсов",
                znach = self.otvet1
                ).grid(row = 3, column = 0, sticky = W)

      self.otvet2 = BooleanVar()
      Checkbutton(self,
                text = "Отслеживание состояния ресурса",
                znach = self.otvet2
                ).grid(row = 4, column = 0, sticky = W)

      self.otvet3 = BooleanVar()
      Checkbutton(self,
                text = "Повышение приоритета задачи",
                znach = self.otvet3
                ).grid(row = 5, column = 0, sticky = W)

      self.btn=Button(self)
      self.btn["text"]="Ok"
```

```
        self.btn["command"]=self.proverka
        self.btn["width"]=10
        self.btn["height"]=2
        self.btn.grid(row=6,column=0)

        self.textbox2=Text(self,width=35,height=5)
        self.textbox2.grid(row=7,column=0)

root = Tk()
root.title("Тестовый вопрос с элементом Флажок")
root.geometry("500x400")
app=Application(root)
root.mainloop()
```

12.3. Создание виджета Переключатель

В отличие от флажков, позволяющих пользователю выбрать один или несколько вариантов, переключатели (**Radiobutton**) реализуют выбор только одного из предложенных вариантов. Разработаем программу на основе тестового вопроса, использующую переключатели. Внешний вид формы представлен на рис. 176.

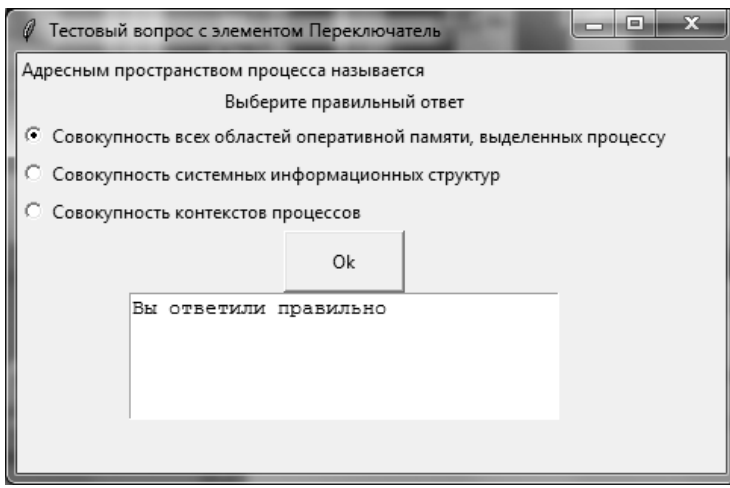


Рис. 176. Тестовый вопрос с элементом управления Переключатель

При программировании виджета Флажок у каждого из флажков был свой атрибут **self.otvet1, self.otvet2, self.otvet3,** который был связан с объектом **BooleanVar().** Теперь в этом нет необходимости, поскольку может быть выбран только один Переключатель. Параметру **znach** каждого элемента **RadioButton** будем присваивать значение одного и того же атрибута **self.otvet,** предварительно связав его с классом **StringVar().**

Значение **Value** каждого переключателя будет получать строковое значение, являющееся ответом на вопрос. Чтобы получить значение переключателя, ис-

пользуется метод **get()** в операторе pr=self.otvet.**get()** метода **proverka.** После проверки логического выражения с помощью условного оператора **if** в переменную **pr** будет занесено соответствующее сообщение о правильности или неправильности ответа. С помощью метода **insert()** в операторе self.textbox2.insert(0.0,**pr**) значение переменной **pr** добавляется в текстовое окно на форме. Ниже приведен программный код (листинг 176), отвечающий за создание интерфейса и осуществление выбора переключателя.

Листинг 176

```
from tkinter import *
class Application(Frame):
   def __init__(self,fr):
      super(Application,self).__init__(fr)
      self.grid()
      self.widget()

   def proverka(self):
      pr=self.otvet.get()
      if pr=="Совокупность всех областей оперативной памяти, выделенных процессу":
         pr="Вы ответили правильно"
      else:
         pr="Вы ответили неправильно"
      self.textbox2.delete(0.0, END)
      self.textbox2.insert(0.0,pr)

   def widget(self):
      Label(self,
          text = "Адресным пространством процесса называется"
          ).grid(row = 0, column = 0, sticky = W)

      self.lbl=Label(self,text="Выберите правильный ответ")
      self.lbl.grid(row=1,column=0)

      self.otvet = StringVar()
      self.otvet.set(None)

      Radiobutton(self,
                text = "Совокупность всех областей оперативной памяти, выделенных процессу",
                znach = self.otvet,
                value = "Совокупность всех областей оперативной памяти, выделенных процессу"
                ).grid(row = 2, column = 0, sticky = W)
```

```
        Radiobutton(self,
                text = "Совокупность системных информационных структур",
                znach = self.otvet,
                value = "Совокупность системных информационных структур"
                ).grid(row = 3, column = 0, sticky = W)

        Radiobutton(self,
                text = "Совокупность контекстов процессов",
                znach = self.otvet,
                value = "Совокупность контекстов процессов"
                ).grid(row = 4, column = 0, sticky = W)

        self.btn=Button(self)
        self.btn["text"]="Ok"
        self.btn["command"]=self.proverka
        self.btn["width"]=10
        self.btn["height"]=2
        self.btn.grid(row=5,column=0)

        self.textbox2=Text(self,width=35,height=5)
        self.textbox2.grid(row=6,column=0)

root = Tk()
root.title("Тестовый вопрос с элементом Переключатель")
root.geometry("500x400")
app=Application(root)
root.mainloop()
```

12.4. Примеры решения задач

Задача 1. Разработайте GUI-программу «Найти сумму двух чисел» с использованием класса.

Комментарий. В начале программы создадим класс **Application,** производный от **Frame,** в котором объявим метод-конструктор **def __init__**(self,fr), из которого, в свою очередь, вызовем метод **widget().** В методе **widget(),** как и в предыдущих программах, займемся созданием меток, текстовых окон, кнопки, т. е. всего того, что называется пользовательским интерфейсом.

С точки зрения решения задачи основным является метод **summa(),** в котором мы используем метод **get()** экземпляра **Entry.** Поскольку метод **get()** возвращает текстовое содержимое элемента, необходимо применение функции **float** для конвертации значения в вещественный тип (**int** – для конвертации значения в целый тип).

В основной части кода создается базовое окно **root,** устанавливаются его заголовок и размеры. Затем создается экземпляр класса **Application** родительского

элемента **root,** после чего происходит запуск событийного цикла методом **mainloop().** Внешний вид разработанного приложения представлен на рис. 177.

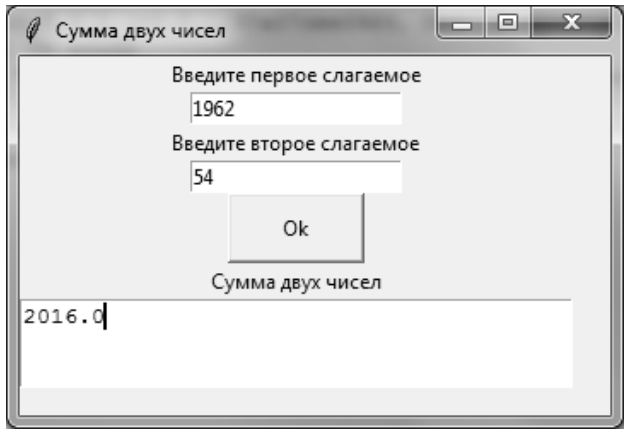


Рис. 177. Интерфейс приложения

Ниже приведен программный код (листинг 177), отвечающий за создание интерфейса и решение задачи.

Листинг 177

```
from tkinter import *
class Application(Frame):
    def __init__(self,fr):
        super(Application,self).__init__(fr)
        self.grid()
        self.widget()

#Создаем метод для нахождения суммы
    def summa(self):
        a=float(self.a.get())
        b=float(self.b.get())
        sum=a+b
        self.otvet.delete(0.0,END)
        self.otvet.insert(0.0,sum)

#Создаем элемент Надпись
    def widget(self):
        self.lbl=Label(self,text="Введите первое слагаемое")
        self.lbl.grid(row=0,column=1)

        #self.a = IntVar()
#Создаем текстовое поле для ввода первого слагаемого
        self.a=Entry(self)
        self.a.grid(row=1,column=1)
```

```
#Создаем элемент Надпись
      self.lbl=Label(self,text="Введите второе слагаемое")
      self.lbl.grid(row=2,column=1)

      #self.b = IntVar()
#Создаем текстовое поле для ввода второго слагаемого
      self.b=Entry(self)
      self.b.grid(row=3,column=1)

#Создаем кнопку
      self.btn=Button(self)
      self.btn["text"]="Ok"
      self.btn["command"]=self.summa #Указываем имя метода
      self.btn["width"]=10
      self.btn["height"]=2
      self.btn.grid(row=4,column=1)

#Создаем элемент Надпись для вывода ответа
      self.lbl=Label(self,text="Сумма двух чисел")
      self.lbl.grid(row=5,column=1)

#Создаем текстовое поле для вывода в него ответа
      self.otvet=Text(self,width=40,height=3)
      self.otvet.grid(row=6,column=1)

root = Tk()
root.title("Сумма двух чисел")
root.geometry("350x270")
app=Application(root)
root.mainloop()
```

Задача 2. Разработайте GUI-программу, вычисляющую значение функции **z**.

$$z = \begin{cases} \min\{a, \max\{x, y, b\}\} & 3 \le x \le 4 \\ a^3 - y\sqrt{b}\,\lg^2|x| & \text{в противном случае} \end{cases}$$

Комментарий. Метод **func()** в программе (листинг 178) отвечает за решение основной задачи – нахождение значения функции **z.** Оператором self.textbox5.**insert**(0.0,z) осуществим вывод найденного значения в пятое текстовое окно на форме. Методом **widget()** создаем пользовательский интерфейс (рис. 178).

Листинг 178

```
from math import*
from tkinter import*
```

```
class Application(Frame):
   def __init__(self,fr):
      super(Application,self).__init__(fr)
      self.grid()
      self.widget()
#Создаем метод для нахождения значения функции
def func(self):
      x=float(self.textbox1.get())
      y=float (self.textbox2.get())
      a=float (self.textbox3.get())
      b=float (self.textbox4.get())
      if x>=3 and x<=4:
         maxi=x
         if y>maxi:
            maxi=y
         if b>maxi:
            maxi=b
         mini=maxi
         if a<mini:
            mini=a
         z=mini
      else:
         z=pow(a,3)-y*sqrt(b)*pow(log(x),2)
      self.textbox5.insert(0.0,z)
#Создаем виджеты на форме
   def widget(self):
      self.lbl1=Label(self,text="Введите число x", font=("Times New Ro-
man",16),fg="blue")
      self.lbl1.grid(row=0,column=0)
      self.textbox1=Entry(self,width=67,font=("Times New Roman",16))
      self.textbox1.grid(row=1,column=0)
#---------------------------------------
      self.lbl2=Label(self,text="Введите значение y", font=("Times New Ro-
man",16),fg="blue")
      self.lbl2.grid(row=2,column=0)
      self.textbox2=Entry(self,width=67,font=("Times New Roman",16))
      self.textbox2.grid(row=3,column=0)
#---------------------------------------
      self.lbl3=Label(self,text="Введите значение a", font=("Times New Ro-
man",16),fg="blue")
      self.lbl3.grid(row=4,column=0)
      self.textbox3=Entry(self,width=67,font=("Times New Roman",16))
      self.textbox3.grid(row=5,column=0)
#---------------------------------------
      self.lbl4=Label(self,text="Введите значение b", font=("Times New Ro-
```

```
man",16),fg="blue")
        self.lbl4.grid(row=6,column=0)
        self.textbox4=Entry(self,width=67,font=("Times New Roman",16))
        self.textbox4.grid(row=7,column=0)
#--------------------------------------
        self.btn=Button(self,font=("Times New Roman",14,"bold"),fg="red")
        self.btn["text"]="Вычислить функцию"
        self.btn["command"]=self.func
        self.btn["width"]=20
        self.btn["height"]=2
        self.btn.grid(row=8,column=0)
        self.lbl5=Label(self,text="Значение функции", font=("Times New Ro-
man",14),fg="green")
        self.lbl5.grid(row=9,column=0)
        self.textbox5=Text(self,width=30,height=1,font=("Times New Roman",14))
        self.textbox5.grid(row=10,column=0)

root=Tk()
root.title("Многозначные ветвления")
root.geometry("740x550")
app=Application(root)
root.mainloop()
```

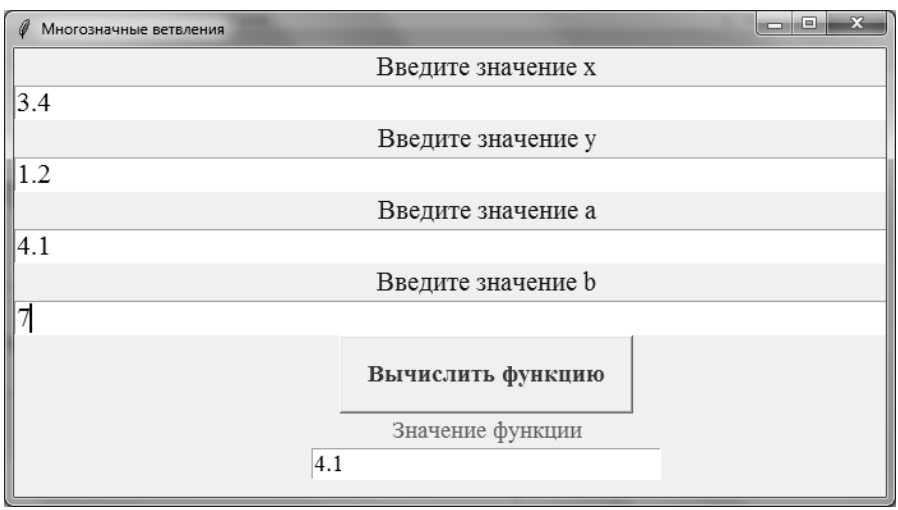


Рис. 178. Результаты выполненной программы

Задача 3. Разработайте GUI-программу, в которой имеется возможность ввода строки символов. Определите, сколько слов содержат хотя бы одну букву «е».

Комментарий. Метод **func()** предназначен для определения количества слов, содержащих хотя бы одну букву «е». В нем методом **get()** мы получаем исходную строку и преобразуем ее с помощью метода **lower()** в строчные буквы. Как известно, метод **split(),** используемый в программе, разделяет строку на под-

строки и добавляет их в список (переменная **spisok)**. Определив длину строки оператором n=**len**(spisok), организуем цикл, в котором каждый элемент строки проверяем на наличие буквы «е». Метод **count()** в операторе self.textbox3.**insert**(0.0,stroka1.**count**("e")) возвратит количество вхождений подстроки «е» в исходной строке в текстовое окно на форме. Код программы представлен в листинге 179, а ее результаты – на рис. 179.

Листинг 179

```
from tkinter import*
class Application(Frame):
   def __init__(self,fr):
      super(Application,self).__init__(fr)
      self.grid()
      self.widget()

   def func(self):
      stroka=self.textbox.get()
      stroka1=stroka.lower()
      spisok=stroka1.split(" ")
      ind=0
      n=len(spisok)
      for i in range (0,n):
         if "e" in spisok[i]:
            ind=ind+1
      self.textbox2.insert(0.0,ind)
      self.textbox3.insert(0.0,stroka1.count("e"))

   def widget(self):
      self.lbl=Label(self,text="Введите строку",font=("Times New Ro-
man",14,"bold"))
      self.lbl.grid(row=0,column=0)

      self.textbox=Entry(self,width=70,font=("Times New Roman",14))
      self.textbox.grid(row=1,column=0)

      self.btn=Button(self,font=("Times New Roman",14,"bold"),fg="red")
      self.btn["text"]="Вычислить"
      self.btn["command"]=self.func
      self.btn["width"]=10
      self.btn["height"]=1
      self.btn.grid(row=2,column=0)

      self.lbl=Label(self,text="Количество слов, содержащих букву
E",font=("Times New Roman",14,"bold"))
      self.lbl.grid(row=3,column=0)
```

```
        self.textbox2=Text(self,width=2,height=1,font=("Times New Ro-
man",32,"bold"))
        self.textbox2.grid(row=4,column=0)

        self.lbl=Label(self,text="Количество повторений буквы E в
строке",font=("Times New Roman",14,"bold"))
        self.lbl.grid(row=5,column=0)

        self.textbox3=Text(self,width=2,height=1,font=("Times New Ro-
man",32,"bold"))
        self.textbox3.grid(row=6,column=0)

root=Tk()
root.title("Работа со строками")
root.geometry("635x280")
app=Application(root)
root.mainloop()
```

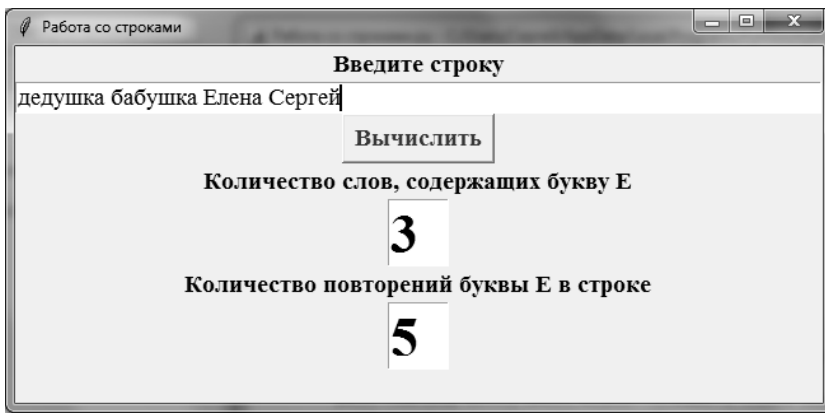


Рис. 179. Результат работы программы

Задача 4. Разработайте GUI-программу, в которой необходимо определить разницу между максимальным и минимальным значениями чисел, введенных пользователем в текстовое окно.

Комментарий. В методе **raznica()** использован классический прием нахождения максимального (ячейка maxi) и минимального (ячейка mini) значений. Первоначально операторами **maxi=-32768** и **mini=32678** заносим в соответствующие ячейки заведомо малое и, соответственно, большое значение для того, чтобы в цикле с оператором **for** последовательно сравнивать очередной элемент строки с найденным максимальным или минимальным значениями. Для того чтобы программа «понимала», что ей приходится работать с числами, используем преобразование типов данных «строковое – в целое числовое», применяя

функцию **int().** Код программы представлен в листинге 180, а результаты ее выполнения на рис. 180.

Листинг 180

```
from tkinter import*
class Application(Frame):
    def __init__(self,fr):
        super(Application,self).__init__(fr)
        self.grid()
        self.widget()

    def raznica(self):
        maxi=-32768
        mini=32678
        chislo=self.textbox.get()
        b=chislo.split(" ")
        for i in range(len(b)):
            if int(b[i])>maxi:
                maxi=int(b[i])
            if int(b[i])<mini:
                mini=int(b[i])
        raznost=maxi-mini
        self.textbox2.insert(0.0,raznost)

    def widget(self):
        self.lbl=Label(self,text="Введите числа через пробел", font=("Times New
Roman",16))
        self.lbl.grid(row=0,column=0)

        self.textbox=Entry(self,width=67,font=("Times New Roman",16))
        self.textbox.grid(row=1,column=0)

        self.btn=Button(self,font=("Times New Roman",12,"bold"),fg="green")
        self.btn["text"]="Вычислить разницу"
        self.btn["command"]=self.raznica
        self.btn["width"]=20
        self.btn["height"]=3
        self.btn.grid(row=3,column=0)

        self.lbl=Label(self,text="Разница между максимальным и минимальным
элементом", font=("Times New Roman",14))
        self.lbl.grid(row=4,column=0)

        self.textbox2=Text(self,width=15,height=1,font=("Times New Roman",14))
        self.textbox2.grid(row=10,column=0)
```

```
root=Tk()
root.title("Работа со строками")
root.geometry("740x220")
app=Application(root)
root.mainloop()
```

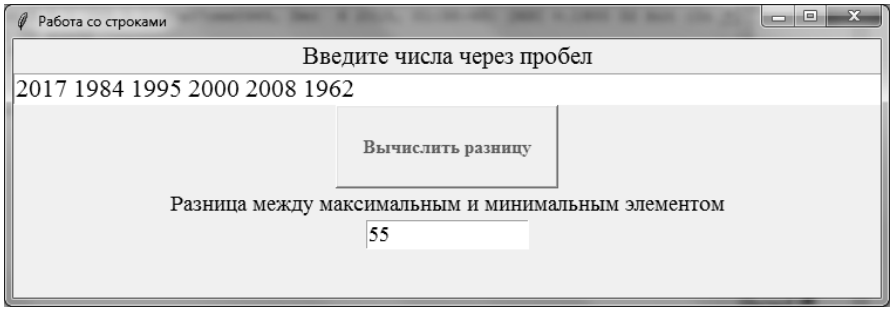


Рис. 180. Найдена разница между максимальным и минимальным значениями

Контрольные вопросы

1. Что понимают под интерфейсом?
2. Какой синтаксис используется для доступа к свойствам и методам?
3. Опишите словесный алгоритм создания в Python базового окна при использовании библиотеки tkinter.
4. Какой метод необходимо использовать для получения данных из текстового окна?
5. Какой метод отвечает за вставку текста в текстовое окно, расположенное на форме?
6. Объясните основные принципы размещения виджетов на форме.
7. Какие классы используются в Python для обработки дробных или целых чисел, логических значений?
8. Какой параметр элемента управления позволяет выровнять его относительно сетки? Какие значения он может принимать?

Задачи для самостоятельного решения

1. Разработайте тестовую GUI-программу на тему: Российские кинофильмы.
2. Разработайте тестовую GUI-программу на тему: Спорт.
3. Разработайте тестовую GUI-программу на тему: Автомобили.
4. Разработайте тестовую GUI-программу на тему: Российская эстрада.
5. Разработайте тестовую GUI-программу на тему: Программирование.
6. Разработайте тестовую GUI-программу на тему: Животные.
7. Разработайте тестовую GUI-программу на тему: Города России.

8. Разработайте тестовую GUI-программу на тему: Достопримечательности моего города.
9. Разработайте тестовую GUI-программу на тему: Кулинария.
10. Разработайте тестовую GUI-программу на тему: Мотоциклы.
11. Разработайте тестовую GUI-программу на тему: Живопись.
12. Разработайте тестовую GUI-программу на тему: Книги.
13. Разработайте тестовую GUI-программу на тему: Мультфильмы.
14. Разработайте тестовую GUI-программу на тему: Зарубежное кино.
15. Разработайте тестовую GUI-программу на тему: Зарубежная эстрада.
16. Разработайте тестовую GUI-программу на тему: Военная техника.
17. Разработайте тестовую GUI-программу на тему: Классическая музыка.
18. Разработайте тестовую GUI-программу на тему: Домашние животные.
19. Разработайте тестовую GUI-программу на тему: Информатика.
20. Разработайте тестовую GUI-программу на тему: Моря и океаны.
21. Разработайте тестовую GUI-программу на тему: Великие пустыни мира.
22. Разработайте тестовую GUI-программу на тему: Герои олимпиад.
23. Разработайте тестовую GUI-программу на тему: История России.
24. Разработайте тестовую GUI-программу на тему: Русские поэты.
25. Разработайте тестовую GUI-программу на тему: Дикие животные.
26. Разработайте тестовую GUI-программу на тему: Российские киноактеры.
27. Разработайте тестовую GUI-программу на тему: Интернет.
28. Разработайте тестовую GUI-программу на тему: Русские народные сказки.
29. Разработайте тестовую GUI-программу на тему: Легенды российского балета.
30. Разработайте тестовую GUI-программу на тему: Оргтехника.

ПРИЛОЖЕНИЕ
ВАРИАНТЫ ДЛЯ ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ

Варианты по теме «Запись арифметических выражений»

№ варианта	Расчетные формулы	Исходные данные
1	$a = \frac{2\cos(x - \pi/6)}{1/2 + \sin^2 y} + \cos^2 x^3 \cdot e^x$ $b = 1 + \frac{z^2}{3 + z^2/5}$	x=1,42 y=1,220 z=3,5
2	$y = \left\|x^{y/x}\right\| - \sqrt[3]{y/x}$ $u = (y - x)\frac{y - z/(y - x)}{1 + (y - x)^2}$	x=1,825 y=18,225 z=-3,298
3	$s = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!}$ $f = x\left(\sin x^3 + \cos^2 y\right)$	x=0,335 y=0,025
4	$y = e^{-bt}\sin(at + b) - \sqrt{\left\|bt + a\right\|}$ $s = b\sin\left(at^2\cos 2t\right) - 1$	a=-0,5 b=1,7 t=0,44
5	$w = \sqrt{x^2}x^3 - x/\sqrt{a^2 + b^2}$ $y = \sqrt{x^2 + b - b^2} - b^2\sin^3(x + a)/x$	a=1,5 b=15,5 x=3,2
6	$s = x^3 tg(x + b)^2 + a/\sqrt{x + b}$ $q = \frac{bx^2 - a}{e^{ax} - 1}$	a=16,5 b=3,4 x=0,61
7	$r = x^2(x + 1)/b - \sin^2(x + a)$ $s = \sqrt{xb/a} + \cos^2(x + b)^3$	a=0,7 b=0,05 x=0,5
8	$y = \sin\left(x^2 + a\right)^2 - x/b$ $z = \frac{x^2}{a} + \cos(x + b)^3$	a=1,1 b=0,004 x=0,2

9	$f = \sqrt[3]{mtgt + \lvert c \sin t \rvert}$ $z = m \cos(bt \sin t) + c$	m=2 b=0,7 g=2 c=-1 t=1,2
10	$y = btg^2 x - \dfrac{a}{\sin^2(x/a)}$ $d = ae^{-\sqrt{a}} \cos(bx/a)$	a=3,2 b=17,5 x=-4,811
11	$f = \ln(a + x^2) + \sin^2(x/b)$ $z = e^{-cx} \dfrac{x + \sqrt{x+a}}{x - \sqrt{\lvert x-b \rvert}}$	a=10,2 b=9,2 c=0,512 x=2,2
12	$y = \dfrac{a^{2x} + b^{-x} \cos(a+b)x}{x+1}$ $r = \sqrt{x^2 + b} - b^2 \sin^3(x+a)/x$	a=0,3 b=0,9 x=0,6113
13	$z = \sqrt{ax \sin 2x + e^{-2x}(x+b)}$ $w = \cos^2 x^3 - x/\sqrt{a^2 + b^2}$	a=0,5 b=3,1 x=1,414
14	$U = \dfrac{a^2 x + e^{-x} \cos bx}{bx - e^{-x} \sin bx + 1}$ $h = 2 \cdot 10^{-15} + \cos x - tg^2 x$	a=0,5 x=0,315 b=2,9
15	$z = \dfrac{\sin x}{\sqrt{1 + m^2 \sin^2 x}} - cm \ln mx$ $s = e^{-ax}\sqrt{x+1} + e^{-bx}\sqrt{x+1,5}$	m=0,7 c=2,1 x=1,7 a=0,5 b=1,0816
16	$y = \dfrac{b + \sqrt{b^2 + 4ac}}{2a} - a^3 c + b^{-2}$ $z = \dfrac{a}{c} \cdot \dfrac{b}{d} - \dfrac{ab - c}{cd}$	a=2,345 b= 3,123 c=0,57 d=1,36
17	$s = \dfrac{\sin x + \cos y}{\cos x - \sin y} \cdot tgxy$ $z = \dfrac{x+y}{x+1} - \dfrac{xy - 12}{34 + x}$	x=3,033 y=0,014
18	$f = \dfrac{3 + e^{y-1}}{1 + x^2 \lvert y - tgx \rvert}$ $z = 6x - \dfrac{x^3}{3} + \dfrac{x^5}{5}$	x=2,03 y=1,599

19	$d = \ln\left(y - \sqrt{\lvert x\rvert}\right)\left(x - \dfrac{y}{x + \dfrac{x^2}{4}}\right)$ $z = (1 - tgx)^{ctgx} + \cos(x - y)$	x=1,333 y=5,014
20	$s = \dfrac{\ln\lvert\cos x\rvert}{\ln(1 + x^2)}$ $f = \left(\dfrac{x+1}{x-1}\right)^x + 18xy^2$	x=-1,255 y=5,23
21	$z = \left(1 + \dfrac{1}{x^2}\right)^x - 12x^2y$ $f = \dfrac{x^2 - 7x + 10}{x^2 - 8x + 12}$	x=1,005 y=3,01
22	$z = \dfrac{\cos x}{\pi - 2x} + 16x \cdot \cos(xy) - 2$ $f = 2^{-x} - \cos x + \sin(2xy)$	x=1,236 y=0,003
23	$z = 2ctg(3x) - \dfrac{1}{12x^2 + 7y - 5}$ $f = \lvert x^2 - x^3\rvert - \dfrac{7x}{y^3 - 15x}$	x=-1,777 y=2,66
24	$z = x \cdot \ln x + \dfrac{y}{\cos x - \dfrac{x}{3}}$ $f = \sin\sqrt{x+1} - \sin\sqrt{x-1}$	x=1,022 y=0,333
25	$z = e^x - \dfrac{y^2 + 12xy - 3x^2}{18y - 1}$ $f = \dfrac{1 + \sin\sqrt{x+1}}{\cos(12y - 4)}$	x=1,111 y=-0,223
26	$z = 2ctg(3x) - \dfrac{\ln\cos x}{\ln(1 + x^2)}$ $f = e^x - x - 2 + (1 + x)^x$	x=0,663 y=3,112
27	$z = 3^x - 4x + \left(y - \sqrt{\lvert x\rvert}\right)$ $k = x - 10^{\sin x} + \cos(x - y)$	x=4,001 y=1,295

28	$f = \dfrac{1+\sin^2(x+y)}{2+\left\lvert x-\dfrac{2x}{1+x^2y^2}\right\rvert}+x$ $z=\sqrt{x}\left\lvert -2\cdot 10^4+\sin x\right\rvert+\dfrac{e^{5x}}{tgy}$	x=6,174 y=-1,65
29	$y=\sqrt{\left(x^3+\sin(x)\right)}+\dfrac{a+1,456}{2,5}-0,003$ $d=5x^5\cdot 3a^5-\sqrt{a+b}$	x=0,388 a=9,456 b=2,111
30	$m=\dfrac{90-saj^5}{1\cdot 10^{12}}+\sin x\cdot tgx$ $p=\sqrt{ba}+ctgx+\dfrac{-a+1\cdot 10^{-5}}{3x^5}$	x=1,104 a=2,03 b=-1,6 j=3,456
31	$z=x\cdot e^{2x}+\dfrac{3,5\cdot 10^6}{\cos x-\dfrac{x}{3}}$ $f=\dfrac{n^2-\cos^3 x+1,23}{x^2-8x+ctgx^3}$	x=2,542 n=0,3
32	$f=\dfrac{3,54+e^{y-1}}{1\cdot 10^5+x^2\left\lvert y-\sqrt{x}\right\rvert}$ $y=\dfrac{b+\sqrt{\cos^2 m+tgx^2}}{2a}-a^5x+e^b b^{-2}$	x=4,001 y=1,295 a=5,23 m=1,023 b=1,36
33	$w=\cos^2 x^2-x\dfrac{\sqrt{a^2+b^2}}{3,23\cdot 10^5}$ $z=\sqrt{x}\cdot\left\lvert 2,74\cdot 10^4+\sin^3 x\right\rvert+\dfrac{e^{5x}}{ctgy}$	x=6,174 y=-1,65 a=4,45 b=0,233
34	$z=e^{-cx}\dfrac{x\cdot\left(\sqrt{x+a}\right)}{x-\left(\sqrt{\lvert x-b\rvert}\right)^3}$ $d=5\cdot\cos^5 x^2\cdot 3a^5-\sqrt{ctgx}$	x=0,388 a=9,456 b=2,111 c=1,11
35	$s=e^{-ax}\sqrt{x+1}+e^{-bx}\cdot\left\lvert tgx^3\right\rvert$ $p=\sqrt{ba}+e^{2x}\cdot x+\dfrac{-a+3,66\cdot 10^{-5}}{3x^6}$	x=1,104 a=2,03 b=-1,6

Варианты по теме «Многозначные ветвления в программах»

№ варианта	Функция	Условие
1	$y = \begin{cases} at^2 \ln t \\ 1 \\ e^{at} \cos bt \end{cases}$	$1 \le t \le 2$ $t > 1$ $t < 2$
2	$y = \begin{cases} \pi x^2 - 7 / x^2 \\ ax^3 + 7\sqrt{x} \\ \ln\left(x + 7\sqrt{x}\right) \end{cases}$	$x < 1{,}3$ $x = 1{,}3$ $x > 1{,}3$
3	$w = \begin{cases} ax^2 + bx + c \\ a / x + \sqrt{x^2 + 1} \\ (a + bx) / \sqrt{x^2 + 1} \end{cases}$	$x > 1{,}2$ $x = 1{,}2$ $x < 1{,}2$
4	$Q = \begin{cases} \pi x^2 - 7 / x^2 \\ ax^3 + 7\sqrt{x} \\ \ln\left(x + 7\sqrt{\lvert x + a \rvert}\right) \end{cases}$	$x < 1{,}4$ $x = 1{,}4$ $x > 1{,}4$
5	$y = \begin{cases} 1{,}5\cos^2 x \\ 1{,}8ax \\ (x - 2)^2 + 6 \end{cases}$	$x < 1$ $1 < x < 2$ $x > 2$
6	$w = \begin{cases} x\sqrt[3]{x - a} \\ x \sin ax \\ e^{-ax} \cos ax \end{cases}$	$x > a$ $x = a$ $x < a$
7	$Q = \begin{cases} bx - \ln bx \\ 1 \\ bx + \ln bx \end{cases}$	$bx < 1$ $bx = 1$ $bx > 1$
8	$y = \begin{cases} \sin x \ln x \\ \cos^2 x \\ -5 \cdot 10^{-31} \cdot x \end{cases}$	$x > 3{,}5$ $x = 3{,}5$ $x < 3{,}5$

9	$f = \begin{cases} \ln(x+1) \\ \sin^2\sqrt{\lvert ax\rvert} \\ \cos x^2 + 2\cdot 10^{-5} \end{cases}$	$x > 1$ $x = 1$ $x < 1$
10	$z = \begin{cases} \left(\ln^3 x + x^2\right)/\sqrt{x+t} \\ \sqrt{x+t} + 1/x \\ \cos x + t\sin^2 x \end{cases}$	$x < 0{,}5$ $x = 0{,}5$ $x > 0{,}5$
11	$s = \begin{cases} \dfrac{a+b}{e^x + \cos x} \\ (a+b)/(x+1) \\ e^x + \sin x \end{cases}$	$x < 2{,}8$ $2{,}8 < x < 6$ $x > 6$
12	$y = \begin{cases} a\ln x + \sqrt[3]{\lvert x\rvert} \\ 2a\cos x + 3x^2 \\ 5\cdot 10^{-7} + \operatorname{tg} x \end{cases}$	$x < 1$ $1 < x < 10$ $x > 10$
13	$w = \begin{cases} \dfrac{a}{i} + bt^2 + c \\ i \\ ai + bi^3 \end{cases}$	$i < 4$ $4 \le i \le 6$ $i > 6$
14	$z = \begin{cases} a\sin\left(\dfrac{i^2+1}{n}\right) \\ \cos\left(i + \dfrac{1}{n}\right) \\ 15y^6 \end{cases}$	$\sin\dfrac{i^2+1}{n} < 0$ $\sin\dfrac{i^2+1}{n} > 0$ $\dfrac{i^2+1}{n} = 1$
15	$w = \begin{cases} \sqrt{at^2 + b\sin t + 1} \\ at + b \\ \sqrt{at^2 + b\cos t + 1} \end{cases}$	$t < 0{,}1$ $t = 0{,}1$ $t > 0{,}1$

16	$w = \begin{cases} -x^2 + 3x + 9 \\ \dfrac{x}{x^3 - 6} \\ 3{,}156 \cdot 10^3 \end{cases}$	$x < 3$ $x = 5$ $x > 3$
17	$w = \begin{cases} -x^2 + 3x + 9 \\ \dfrac{x}{x^3 - 6} \\ 3{,}156 \cdot 10^3 \end{cases}$	$x = 5$ $x < 3$ $x = 3$
18	$Q = \begin{cases} 9 - \cos z + \sqrt{x} \\ \dfrac{1}{x^2 + 1} \\ 15x^6 + 3 \cdot 10^2 \end{cases}$	$x \le 3$ $x > 3$ $x = z$
19	$y = \begin{cases} -5 \cdot 10^5 + sd^5 \\ \dfrac{1}{x + 6} \\ \cos z \end{cases}$	$x \le 1$ $x > 1$ $x = z$
20	$z = \begin{cases} -3x + 9 \\ \dfrac{1}{x - 7} \\ 3 \cdot 10^{-5} \end{cases}$	$x \le 7$ $x > 7$ $x = a\ b$
21	$y = \begin{cases} 3x - 9 \\ \dfrac{1}{x^2 - 4} \\ \sin x + tgx - ctgx \end{cases}$	$x \le 7$ $x > 12$ $x = 1 \cdot 10^{-5}$
22	$Q = \begin{cases} x^2 + e^x + \lvert -2.123 + x \rvert \\ 4 \\ \sqrt[3]{x^z} \end{cases}$	$0 \le x \le 3$ $x > 3$ $x = z$
23	$W = \begin{cases} x^2 + 4x + 5 \\ \dfrac{1}{x^2 + 4x + 5} \\ \sqrt[4]{6x^3} + tgx \end{cases}$	$x < 0$ $x > 2$ $x = 2$

24	$f = \begin{cases} x^2 - x \\ x^2 - \sin \pi x^2 \\ -x + \sqrt[4]{13x^5} \end{cases}$	$0 \le x \le 1$ $x > 1$ $x = z$
25	$y = \begin{cases} \sin x - \left\lvert \frac{x^{15}}{1 \cdot 10^5} \right\rvert \\ \cos x \\ ctgz + tga \end{cases}$	$x \le 0$ $x > 0$ $x = z$
26	$Q = \begin{cases} x^2 + e^x \cdot 3x^2 + tgx^3 \\ 3.111 \cdot 10^5 \\ e^x + \cos^{2x} x - 3,7777 \end{cases}$	$0 \le x \le 3$ $x < \pi$ $x = t$
27	$W = \begin{cases} x^2 + \sqrt[3]{ctgx} + 9,222 \cdot 10^5 \\ \frac{1}{x^2 + 4x + 5} \\ \sqrt[4]{2x^3} + \ln x - \cos^2 x \end{cases}$	$x < 0$ $x > 7$ $x = 5$
28	$f = \begin{cases} x^2 - 2x^{3z} + 3,222 + 2,5^5 \\ 3x^2 - \sin \pi x^2 + ctgx^3 \\ -x + \sqrt[4]{13x^5 + 9x} - e^{3x} \end{cases}$	$0 \le x \le 1$ $x > 5$ $x = n$
29	$y = \begin{cases} \sin x - \left\lvert \frac{x^{15}}{1 \cdot 10^5} \right\rvert \\ \cos x + tg^2 x + 3,333 \cdot 10^{10} \\ ctgz + tg^3 a - \sqrt{x^2} \end{cases}$	$x \le 0$ $x > 2,55$ $x = 7$
30	$\mu = \begin{cases} 3x^2 + \ln x^2 \cdot tgx^3 \\ 2,5 \cdot 10^5 + \sqrt{x \cdot n} \\ e^x + \cos^{2x} x - 3,7777 \end{cases}$	$0 \le x \le 5$ $x > 2\pi$ $x = b$
31	$\omega = \begin{cases} \sqrt[3]{x^4 + 3x^3} + 5,111 \cdot 10^{15} \\ \frac{\cos^2 x}{x^2 + 4x + tgx^5} \\ \ln x - \cos^2 x + e^{3x} \cdot \sqrt{x} \end{cases}$	$x < 0$ $x > 1$ $x = n$

32	$\nu = \begin{cases} x^2 - 2x^{3z} + 3,222 + 2,5^5 \\ \sin \pi x^3 + tgx^3 \\ \sqrt[4]{x^5 + \cos 9x} - e^{3x} \end{cases}$	$0 \le x \le 3$ $x > 15$ $x = 2,33$
33	$\beta = \begin{cases} \left\| \frac{x^{15}}{3 \cdot 10^{-15}} \right\| + ctgx^2 + 2\sin x \\ tg^2 x + 5,45 \cdot 10^{16} - \sqrt[3]{54x^3} \\ ctgb + tg^3 s - \sqrt{\ln x} \end{cases}$	$x \le 0$ $x > 5$ $x = 15$
34	$s = \begin{cases} tgx^3 + 1,222 \cdot \frac{2}{5} \cdot \cos x \cdot 5! \\ \sqrt{x \cdot tgx + ctgx^2} \\ \cos^{2x} + 2x - \ln x^2 + 3,1 \cdot 10^{15} \end{cases}$	$0 \le x \le 5$ $x < \pi$ $x = d$
35	$\omega = \begin{cases} 5,334 \cdot 10^{-15} + \left\| -2x \cdot \cos x \right\| \\ \frac{\ln x^2 x}{tgx^5 + \sin x^5 + 3,111} \\ \sqrt{x} + \sqrt[3]{\sin x^3} \end{cases}$	$x < 0$ $x > 3$ $x = q$

Варианты по теме «Программирование алгоритмов разветвляющихся структур с использованием поиска максимального и минимального значений»

№ варианта	Функция	Условие
1	$e = \begin{cases} x^3 + \lg(xy)^{cd} \\ 3\min\{x, y, \max\{cx, dy\}\} \\ 2^{cd} - x \end{cases}$	$xy > 3$ $2 \le x \le 3$ $xy < 0$

2	$z=\begin{cases}1-e^{xy+ab}\\ b-\min\{ax,y\}\\ \max\left\{x^3,e^y,\sqrt{\lvert\ln y^2\rvert}\right\}\end{cases}$	$xy>0$ $xy=0$ $xy<0$
3	$z=\begin{cases}\max\left\{\frac{a}{b},\frac{b}{x},\sqrt{y}\right\}\\ \min\left\{\frac{a}{x},\sqrt{y}\right\}\\ 2^{x+y}\end{cases}$	$x>0$ и $y>0$ $x<0$ и $y>0$ в противном случае
4	$g=\begin{cases}y\sqrt{1+(zx)^2}\\ \min\{a+x,\max\{y,z\}\}\\ -be^y\end{cases}$	$y>0$ и $xy^2>0$ $y>0$ и $xy^2\le 0$ в противном случае
5	$f=\begin{cases}\min\left\{\frac{x-a}{x},\sqrt{a}+x,\sin x\right\}\\ \max\left\{\sqrt{x,ax}\right\}\\ ax+b\end{cases}$	$0<x\le 1$ $x>1$ $x\le 0$
6	$d=\begin{cases}\min\left\{x^3,e^{-x+1},\max\{\lg x,x+y\}\right\}\\ 1+x^2\\ c^2+d\cos(x+y)\end{cases}$	$x>0$ и $e^{-x}\ge y$ $x\le 0$ и $e^{-x}\ge y$ в противном случае
7	$z=\begin{cases}\min\{a,\max\{x,y,b\}\}\\ a^3-y\sqrt{b}\lg^2\lvert x\rvert\end{cases}$	$3\le x\le 4$ в противном случае
8	$z=\begin{cases}\min\left\{a-\cos x,\frac{a}{b+y},\sin^2 y\right\}\\ \max\left\{a^3,\ln\left(x^2+y^2\right)\right\}\\ a+\cos^3(x-y)\end{cases}$	$x<y$ $y\le x<y+5$ $x\ge y+5$

9	$d = \begin{cases} \cos\left(1 - \dfrac{cx^2}{b}\right) \\ \max\left\{c, x, \min\left\{\sqrt{x}, \sqrt{b} + c\right\}\right\} \\ e^{bx+c} \end{cases}$	$0 \le x \le 1$ $x > 1$ $x < 0$
10	$p = \begin{cases} \min\left\{\max\left\{x, y^2\right\}, \sqrt{x}, \sqrt{y + c}\right\} \\ \max\left\{\sin x, \cos y, \operatorname{tg}(xy)\right\} \end{cases}$	$y \ge 0$ в противном случае
11	$z = \begin{cases} x\sqrt{b^2 + c^2} \\ \min\left\{\sqrt{b}, x^2, x + c\right\} \\ \max\left\{\ln b, x + c\right\} \end{cases}$	$x > 1$ $x \le 0$ в противном случае
12	$P = \begin{cases} \max\left\{x, y\right\} \\ \min\left\{x, b\right\} \\ \min\left\{\sin a, \cos b, \max\left\{x^2, a + b\right\}\right\} \end{cases}$	$x > 1$ $-1 \le x \le 0$ в противном случае
13	$r = \begin{cases} \sqrt{\lvert x\rvert + \lvert y\rvert + \lvert z\rvert} \\ \min\left\{\sqrt{\lvert x\rvert}, \sqrt{\lvert y\rvert}, \sqrt{\lvert z\rvert}\right\} \\ \max\left\{x, y^3\right\} + a \end{cases}$	$\lvert z\rvert^{xy} < 3$ $\lvert z\rvert^{x+y} > 4$ $3 \le \lvert z\rvert^{x+y} \le 4$
14	$r = \begin{cases} \min\left\{x, y, cx, dy\right\} \\ \max\left\{\lg 2x, \sqrt{x + y}, yx\right\} \end{cases}$	$0 \le x \le 1$ и $y > 2$ в противном случае
15	$s = \begin{cases} a\sin x + b\cos x \\ \max\left\{x^3, e^x, 10^3\right\} \\ \min\left\{\dfrac{\sin x}{x}, \max\left\{a^x, x^3\right\}, x\ln^2 x\right\} \end{cases}$	$x < 2$ $x > 3$ в противном случае

16	$t = \begin{cases} \left(1 - \sqrt{x^2 + a}\ \max\{x, y\}\right) \\ \min\left\{x^2, \sin y, \cos(ay)\right\} \\ a^2 + \dfrac{x}{y} \end{cases}$	$xy < 0$ $xy > 2$ в противном случае
17	$u = \begin{cases} \ln(x)\min\{x, z\} \\ \max\left\{x^2, z^2 - a^2, \min\{x, z\}\right\} \end{cases}$	$x > 0$ и $z > 0$ в противном случае
18	$f = \begin{cases} \min\left\{bx^2, cx^3, \max\left\{\sqrt{\lvert c\rvert}, \sqrt{\lvert x\rvert}\right\}\right\} \\ b - cx^2 \\ \operatorname{arctg}\dfrac{b^2}{c^2 + x^2} \end{cases}$	$x < 3$ $x > 3$ в противном случае
19	$v = \begin{cases} \dfrac{a + b + c}{2} \cdot \min\left\{x, y, \dfrac{x + y}{x - y}\right\} \\ \min\{x, y\} \\ y(a + b + c) \end{cases}$	$x < 3$ $x > 0$ и $y > 1$ в противном случае
20	$h = \begin{cases} x^3 \lvert a\rvert \sin y \\ \max\{x, y, z\} \\ \min\{y, z\} \end{cases}$	$\lvert x\rvert^y < z$ $z \le \lvert x\rvert^y \le -z$ в противном случае
21	$z = \begin{cases} \min\{x, y\} \\ \dfrac{\max\{x, y, b\}}{\min\{x, y, b\}} \\ b + y\sin x \end{cases}$	$xy < 2$ $xy > 5$ в противном случае
22	$\beta = \begin{cases} \min\left\{c, \sqrt{x}\ \min\{y, z\}, y - z\right\} \\ \dfrac{xyz}{1 - c} \end{cases}$	$x > 0$ и $yz > 0$ в противном случае
23	$z = \begin{cases} ay^2 \cos \\ \dfrac{\max\{x, y\}}{\min\{a, x, y\}} \end{cases}$	$xy > 2$ $xy \le 2$

24	$z = \begin{cases} \min\left\{\frac{x}{a}, \frac{\sqrt{\lvert x-y\rvert}}{a}\right\} \\ \max\{ax, 2y, \sin(x+y)\} \end{cases}$	$x < y$ и $y > 0$ в противном случае
25	$z = \begin{cases} \min\left\{ax, y, \max\left\{\sqrt{x}, \sin y\right\}\right\} \\ \frac{b+2x}{e^{ay}} \end{cases}$	$x \le 0$ и $y > 2$ в противном случае
26	$y = \begin{cases} x\sqrt[3]{a+b} \\ \frac{1}{3}\max\{a,b,c\} \end{cases}$	$a < b < c$ в противном случае
27	$y = \begin{cases} \min\left\{\frac{x-a}{x}, \sqrt{a+x}\right\} \\ \max\left\{\sqrt{\lvert x, a^x\rvert}\right\} \\ 0 \end{cases}$	$x \ge 5$ $x < 0$ $x > 1$

Варианты по теме «Табулирование функции»

№ вари-анта	Функция	Параметры		Шаг
		начальное значение	конечное значение	h
1	F = x – Sin (x) + 3 Tg (x)	1	6	0,5
2	$F = Sin^2(x) + Ctg(x)$	1	5	0,5
3	$F = Cos^2(x) + 1$	1	8	0,5
4	F = Tg (x) + Sin (x)	1	6	0,5
5	F = Ctg (x) + Cos (x)	1	6	0,5
6	F = 5 Sin (x) + Tg (x)	0	10	0,7
7	F = Cos (x) + Arctg (x)	0	10	0,7
8	F = Arctg (x) + 3 Sin (x)	2	8	0,5
9	F = Sin (x) – Cos (x)	0	15	0,7
10	$F = x^2$ Sin (x) + 2 Cos (x)	0	13	0,7
11	F = Sin(1 / x) + 5 Sin^2 (x)	1	6	0,5
12	F = Cos(1 / x) + 2 Tg(X)	1	8	0,5
13	F = Sin (x^2) + 4 Sin (x)	1	11	0,5
14	F = Cos (x^2) + Cos^2 (x)	1	13	0,7

15	F = 3 Sin (x) + Tg (x)	0	8	0,5
16	F = Cos (x) + Ctg^2 (x)	1	9	0,5
17	F = Tg (x / 2)	0	9	0,5
18	F = Tg (x / 2) + Cos (x)	2	14	0,7
19	F = Ctg (x / 3) + 2 Sin (x)	3	13	0,7
20	F = Sin (x / 4) / 2	1	8	0,5
21	F = Cos (x / 4) / 2	1	14	0,7
22	F = Ctg (x / 4)	1	9	0,5
23	F = Sin (x) + 3 Cos (x / 2)	0	8	0,5
24	F = 2 Cos (x^2) + Sin (2 x)	0	16	0,7
25	F = 2 Ctg (x) – Cos^2 (x)	1	8	0,5
26	F = 2 x Cos (x)	1	7	0,5
27	F = Sin (x) + 5 Tg (x)	1	13	0,5
28	F = 4 Sin(X) + Ctg(x)	0	8	0,5
29	F = 3 Cos (x) + Sin(x / 2)	1	9	0,5
30	F = x + Tg (x)	0	9	0,5
31	F = 2 Sin (x) + Cos^2 (x)	2	14	0,7
32	F = Arctg (x) + Cos (x)	3	13	0,7
33	F = Tg (x) + 2 Cos^2 (x)	1	8	0,5
34	F = Tg^2 (x) + 5 Cos(x)	1	14	0,7
35	F = 9 Sin (x) + 2 Cos (x)	1	9	0,5

Варианты по теме «Программирование алгоритмов регулярных циклических структур»

№	Задача
1	Получите таблицу значений функции y=f(x) при изменении x на отрезке[a,b] с шагом h. $y = \begin{cases} x^2 - 1, & \text{если } x \le 0 \\ \cos x, & \text{если } 0 < x \le 1,5 \\ \sin(x-1), & \text{если } x > 5 \end{cases}$ Отрезок $[-4,4]$, шаг $h = 0,5$
2	Постройте таблицу значений и найдите наибольшее значение функции y=f(x) при изменении x на отрезке[a,b] с шагом h. Y=3$\cos^2$ (2x+1). Отрезок $[-\pi,\pi]$, шаг $h=\frac{\pi}{8}$.

3	Постройте таблицу и вычислите сумму значений функции y=f(x) при y>0 при изменении x на отрезке [a,b] с шагом h. $Y = \frac{1}{x + 2\pi} - \sin x$. Отрезок $[-\pi,\pi]$, шаг $h=\frac{\pi}{8}$
4	Получите таблицу значений функции y=f(x) при изменении x на отрезке[a,b] с шагом h. $Y = \begin{cases} e^x, & \text{если } x > 2 \\ x + 4, & \text{если } -2 \le x \le 2 \\ 0, & \text{если } x < -2 \end{cases}$ Отрезок [-2,2], шаг h=0.25
5	Постройте таблицу и найдите наибольшее значение функции y=f(x) при изменении x на отрезке[a, b] с шагом h. $Y=0.5\, e^{\sin x} - x - 1$. Отрезок [0,10], шаг h=0.5
6	Постройте таблицу и вычислите произведение значений функции y=f(x) при y>0 при изменении x на отрезке [a, b] с шагом h. $Y=x^4 + x^3 - 10x - 34x - 25$ Отрезок [0,10], шаг h=0.5
7	Получите таблицу значений функции y=f(x) при изменении x на отрезке[a, b] с шагом h. $Y = \begin{cases} \sin x, \text{ если } -1 \le x \le 1 \\ 5\cos x, \text{ в противном случае.} \end{cases}$ Отрезок [-2,2], шаг h=0.25
8	Постройте таблицу и найдите наибольшее значение функции y=f(x) при изменении x на отрезке[a, b] с шагом h. $Y = xe^{-x}$ Отрезок [0.1, 1.5], шаг h=0.1
9	Постройте таблицу и вычислите сумму значений функции y=f(x) при y<0 и при изменении x на отрезке [a,b] с шагом h. Y=0.5-0.1- sinx. Отрезок $[0,2\pi]$, шаг $h=\frac{\pi}{8}$
10	Получите таблицу значений функции y=f(x) при изменении x на отрезке[a, b] с шагом h. $Y = \begin{cases} e^x, & \text{если } x > 1 \\ 2x - 1, & \text{если } x < 0 \\ -1, & \text{если } 0 \le x \le 1 \end{cases}$ Отрезок [-2,2], шаг h=0.25
11	Постройте таблицу и найдите наибольшее значение функции y=f(x) при изменении x на отрезке[a,b] с шагом h.

	$Y = 2^{-x} e^{x}$. Отрезок [-1,1], шаг h=0.1
12	Постройте таблицу и вычислите произведение значений функции y=f(x) при y<0 при изменении x на отрезке [a,b] с шагом h. $Y = x^{8} - 0{,}4x^{3} - 1{,}24$ Отрезок [-1.5, 1.5], шаг h=0.15
13	Получите таблицу значений функции y=f(x) при изменении x на отрезке[a,b] с шагом h. $Y = \begin{cases} x^{\frac{1}{3}}, & \text{если } x > 6 \\ 2\sin x, & \text{если } x < 5 \\ \sqrt{x+1}, & \text{если } 5 \le x \le 6. \end{cases}$ Отрезок [2,12], шаг h=0.5
14	Постройте таблицу и найдите наибольшее значение функции y=f(x) при изменении x на отрезке[a,b] с шагом h. $Y = e^{-x^{2}} + x + 1$ Отрезок [-5,5], шаг h=0.5
15	Постройте таблицу и вычислите сумму значений функции y=f(x) при 0.5<y<1.5 при изменении x на отрезке [a,b] с шагом h. Y=1+cos10x. Отрезок $[-\pi / 2, \pi / 2]$, шаг $h=\pi/16$
16	Получите таблицу значений функции y=f(x) при изменении x на отрезке[a,b] с шагом h. $Y = \begin{cases} 5e^{x}, & \text{если } 0 \le x \le 5 \\ 2\sin x, & \text{если } x > 5 \\ \lvert x \rvert, & \text{если } x < 0 \end{cases}$ Отрезок [-2,6], шаг h=0.5
17	Постройте таблицу и найдите наибольшее значение функции y=f(x) при изменении x на отрезке[a,b] с шагом h. $Y=x^{3}-6x^{2}+9x+4$ Отрезок [2,4], шаг h=0.1
18	Постройте таблицу и вычислите произведение значений функции y=f(x) при y>0 при изменении x на отрезке [a,b] с шагом h. $Y=x^{5}+5x^{4}-2x^{3}-4x^{2}+7x-3$. Отрезок [-0.5, 1.5], шаг h=0.1
19	Получите таблицу значений функции y=f(x) при изменении x на отрезке[a,b] с шагом h.

	$Y=\begin{cases} x^2, & \text{если } -2 \le x \le 3 \\ 0, & \text{если } x > 3 \\ 4\cos x, & \text{если } x < -2 \end{cases}$ Отрезок [-3, 1.5], шаг h=0.4
20	Постройте таблицу и найдите наибольшее значение функции y=f(x) при изменении x на отрезке[a,b] с шагом h. $Y=\frac{\ln^2 x}{x}$ Отрезок [6,8], шаг h=0.2
21	Постройте таблицу и вычислите сумму значений функции y=f(x) при y<1.2 при изменении x на отрезке [a,b] с шагом h. Y=sin(4x)-2 Отрезок $[-\pi,\pi]$, шаг h=$\frac{\pi}{8}$
22	Получите таблицу значений функции y=f(x) при изменении x на отрезке[a,b] с шагом h. $Y=\begin{cases} e^{x-2}, & \text{если } 0 \le x \le 2 \\ \lg x, & \text{если } x > 2 \\ 0{,}1, & \text{если } x < 0 \end{cases}$ Отрезок [-4,4], шаг h=0.5
23	Постройте таблицу и найдите наименьшее значение функции y=f(x) при изменении x на отрезке[a,b] с шагом h. Y=x+1/x Отрезок [0.1, 1.5], шаг h=0.1
24	Постройте таблицу и вычислите произведение значений функции y=f(x) при y>0 при изменении x на отрезке [a,b] с шагом h. $Y=x^3-6x^2+19{,}8$ Отрезок [-3,0], шаг h=0.15
25	Постройте таблицу и найдите наибольшее значение функции y=f(x) при изменении x на отрезке[a,b] с шагом h. $Y=\operatorname{arctg}(x)-\frac{\ln(1+x^2)}{2}$ Отрезок [0.1, 1.5], шаг h=0.1
26	Постройте таблицу и вычислите сумму значений функции y=f(x) при y>0 при изменении x на отрезке [a,b] с шагом h. $Y=\frac{\cos 8x}{\sqrt{1+10x}}$. Отрезок $[0, \pi]$, шаг h=$\frac{\pi}{8}$

27	Постройте таблицу и найдите наибольшее значение функции y=f(x) при изменении x на отрезке[a,b] с шагом h. $Y = x\sqrt[3]{x-1}$ Отрезок [0.1, 1.5], шаг h=0.1
28	Постройте таблицу и вычислите произведение значений функции y=f(x) при y>0 при изменении x на отрезке [a,b] с шагом h. $Y=x^4+39x^3+958x^2-1081x-1987$ Отрезок [1.9, 2.1], шаг h=0.01
29	Найдите наименьшее значение функции y=f(x) при изменениях x на отрезке[a,b] с шагом h. $Y=5*\sin(2x+1)^2$ Отрезок $[-\pi,\pi]$, шаг $h=\frac{\pi}{8}$
30	Найдите наибольшее значение функции y=f(x) при изменениях x на отрезке[a,b] с шагом h. $Y=5*\sin(2x+1)^4 - \cos(x)$ Отрезок $[-\pi,\pi]$, шаг $h=\frac{\pi}{8}$

Варианты по теме «Табулирование функции с использованием циклов с неизвестным количеством повторений»

№ варианта	Задача
1	Вычислите значения y, соответствующие каждому значению x ($xn \le x \le xk$, шаг изменения x равен dx), по формуле $y = \frac{\sqrt{\lvert a - x^2 \rvert \ln(a+x)}}{\sqrt[3]{x^2 + \sqrt[5]{a}}}$. Вычислите сумму, произведение и количество значений y. Контрольный расчет проведите при a=2,17; x_n=-1,5; x_k=0,5; dx=0,2.
2	Вычислите значения z, соответствующие каждому значению x ($xn \le x \le xk$, шаг изменения x равен dx), по формуле $z = \frac{\sqrt[4]{x^3 + ax}}{\ln\sqrt{a^2 + \sqrt{x}}}$. Определите среднее арифметическое вычисленных z. Контрольный расчет проведите при a=5,27; x_n=1; x_k=10; dx=1.

3	Вычислите значения t, соответствующие каждому значению x ($xn \le x \le xk$, шаг изменения x равен dx), по формуле $t = \frac{\left\lvert a - b\sqrt[3]{x}\right\rvert}{b\ln\left\lvert a^2 + x\right\rvert}$. Определите $F = \frac{\sum t}{\prod t}$. Контрольный расчет проведите при a=3,5; b=6,8; x_n=-3; x_k=3; dx=0,5.
4	Вычислите значения t, соответствующие каждому значению x ($xn \le x \le xk$, шаг изменения x равен dx), по формуле $t = \frac{\sqrt[3]{ax}}{a + x\lg(a + x)}$. Вычислите сумму положительных значений t, произведение отрицательных t, количество t. Контрольный расчет проведите при a=1,23; x_n=-0,5; x_k=0,5; dx=0,1.
5	Вычислите значения z, соответствующие каждому значению x ($xn \le x \le xk$, шаг изменения x равен dx), по формуле $z = \frac{\left(ax^2\right)\sqrt[3]{\frac{1}{(a + x)^2}}}{a\ln\left(a + x^2\right)}$. Определите $F = \prod_{z<a} z + \sum_{z \ge a} z$. Контрольный расчет проведите при a=2,62; x_n=-3; x_k=3; dx=0,6.
6	Вычислите значения t, соответствующие каждому значению x ($xn \le x \le xk$, шаг изменения x равен dx), по формуле $t = \frac{a + \sqrt{ax}}{\sqrt{a} + \ln(a + x)}$. Вычислите сумму значений $t \ge a$, произведение всех значений t, количество отрицательных t. Контрольный расчет проведите при a=3,72; x_n=1; x_k=3; dx=0,2.
7	Определите значения t, соответствующие каждому значению x ($xn \le x \le xk$, шаг изменения x равен dx), по формуле $t = (a + b)^2\sqrt{\frac{a + x}{b + x}}\ln(a + x)$. Вычислите количество отрицательных значений x. Найдите минимальное значение среди вычислительных значений t. Контрольный расчет проведите при a=6,13; b=3,42; x_n=-2; x_k=3; dx=0,5.
8	Определите значения y, соответствующие каждому значению x ($xn \le x \le xk$, шаг изменения x равен dx), по формуле $y = \frac{a^2 + x\sqrt[3]{x}}{\sqrt{a} + \sqrt[3]{x}}$. Найдите максимальное значение y и среднее значение среди положительных элементов x. Контрольный расчет проведите при a=2,89; x_n=-50; x_k=50; dx=10.
9	Вычислите значения z, соответствующие каждому значению x ($xn \le x \le xk$, шаг изменения x равен dx), по формуле

	$z = a\sqrt[4]{\frac{ax}{\ln^3(a+x)}}$. Определите разницу между минимальным и максимальным значениями z. Контрольный расчет проведите при a=2,94; x_n=1,5; x_k=5,5; dx=0,4.
10	Найдите значения z, соответствующие каждому значению x ($xn \le x \le xk$, шаг изменения x равен dx), по формуле $z = \frac{\sqrt[3]{(a^2 - 2ab + x)}}{(a+b)^2 + e^x}$. Определите минимальное значение среди значений $z \le 0$, максимальное среди z > 0 и количество вычисленных z. Контрольный расчет проведите при a=4,32; b=8,13; x_n=-3; x_k=4; dx=0,7.
11	Вычислите значения z, которые соответствуют каждому значению x (x=1, dx=0,5), по формуле $z = \ln x\sqrt{\frac{x}{x^3+1}}$. Считайте z до тех пор, пока подкоренное выражение больше или равно 0,02. Определите k – количество вычисленных z
12	Вычислите значения y, соответствующие каждому значению x ($xn \le x \le xk$, шаг изменения x равен dx), по формуле $y = \frac{Ln(x)\,Tg(x)\sqrt{x}}{\sqrt{x^2}}$. Вычислите сумму, произведение и количество значений y. Контрольный расчет проведите при x_n=1; x_k=10; dx=0,5
13	Вычислите значения z, соответствующие каждому значению x ($xn \le x \le xk$, шаг изменения x равен dx), по формуле $z = \frac{\sqrt[3]{5x^3 + x}}{tgx\sqrt{a^2} + \sin^2 x}$. Определите сумму, количество и среднее арифметическое вычисленных z. Контрольный расчет проведите при x_n=1; x_k=10; dx=1
14	Вычислите значения t, соответствующие каждому значению x ($xn \le x \le xk$, шаг изменения x равен dx), по формуле $t = \frac{\left\|a^3 - b\sqrt[3]{x}\right\|}{\left\|a^2 + x\right\| x^2}$. Определите $F = \frac{\sum t}{\prod t}$. Контрольный расчет проведите при a=4,5; b=4,2; x_n=1; x_k=10; dx=1
15	Вычислите значения t, соответствующие каждому значению x ($xn \le x \le xk$, шаг изменения x равен dx), по формуле $t = \frac{\sin x^2 + 3,2}{a + x\,e^{(a+x)}}$. Вычислите сумму положительных значений t,

	произведение отрицательных t, количество t. Контрольный расчет проведите при a=5,11; x_n=-5; x_k=5; dx=0,5
16	Вычислите значения z, соответствующие каждому значению x ($xn \le x \le xk$, шаг изменения x равен dx), по формуле $z = 5x^2 + \sqrt[3]{a}\sqrt{a^5 + 3{,}8 \cdot 10^6}$. Определите $F = \prod z + \sum z$. Контрольный расчет проведите при a=3,94; x_n=-1; x_k=7; dx=0,8
17	Вычислите значения t, соответствующие каждому значению x ($xn \le x \le xk$, шаг изменения x равен dx), по формуле $t = \sqrt{a} + e^{(a+x)} \sin(5x)$. Вычислите сумму значений t при $t \ge a$, произведение всех значений t, количество отрицательных t. Контрольный расчет проведите при a=1,02; x_n=1; x_k=5; dx=0,2
18	Определите значения t, соответствующие каждому значению x ($xn \le x \le xk$, шаг изменения x равен dx), по формуле $t = (a + b + x)^2 \sqrt{\frac{a^5 + x}{b^5 + x}}$. Вычислите количество значений t меньше 1. Найдите минимальное значение среди вычислительных значений t. Контрольный расчет проведите при a=2,43; b=3,15; x_n=-5; x_k=3; dx=0,5
19	Определите значения y, соответствующие каждому значению x ($xn \le x \le xk$, шаг изменения x равен dx), по формуле $y = \sqrt[3]{a^5 + x + \sin^2 x} \cdot 1 \cdot 10^{-5}$. Найдите максимальное значение y и среднее значение среди положительных элементов x. Контрольный расчет проведите при a=2,43; x_n=-50; x_k=50; dx=10
20	Вычислите значения z, соответствующие каждому значению x ($xn \le x \le xk$, шаг изменения x равен dx), по формуле $z = ctg(x) + \sqrt{a^5} + e^{2x^2}$. Определите разницу между минимальным и максимальным значениями z. Контрольный расчет проведите при a=5,31; x_n=1,5; x_k=5,5; dx=0,4.
21	Найдите значения z, соответствующие каждому значению x ($xn \le x \le xk$, шаг изменения x равен dx), по формуле $z = 0{,}312 + \frac{5\, e^{5x}}{3 \times 10^3} \cdot \sqrt{a}$. Определите максимальное значение среди z>0 и количество вычисленных z. Контрольный расчет проведите при a=3,32; x_n=-5; x_k=10; dx=1
22	Вычислите значения z, которые соответствуют каждому значению x (x=1, dx=1), по формуле $z = \sin x / 1 \cdot 10^6 + \sqrt{x^2 + 5}$. Считайте z до тех пор, пока подкоренное выражение больше или равно 0,05. Определите k – количество вычисленных z, сумму вычисленных z, минимальное среди вычисленных z

23	Вычислите значения z, которые соответствуют каждому значению x (x=1, dx=0,1), по формуле $z = 5 \cdot \sqrt{x} + \sqrt{\frac{x}{3x^2 + tgx}}$. Считайте z до тех пор, пока подкоренное выражение больше или равно 0,1. Определите k – количество вычисленных z, разность между максимальным и минимальным значением z
24	Вычислите значения z, которые соответствуют каждому значению x (x=1, dx=0,5), по формуле $z = tgx \cdot \sqrt{x} + \sqrt{\frac{x}{\lvert x^3 \rvert + \cos x}}$ Считайте z до тех пор, пока подкоренное выражение больше или равно 0,03. Определите k – количество вычисленных z, произведение положительных z
25	Вычислите значения z, которые соответствуют каждому значению x (x=1, dx=0,3), по формуле $z = e^x \cdot x^5 + \sqrt{\frac{x^3}{x \sin x + x^5}}$. Считайте z до тех пор, пока подкоренное выражение больше или равно 0,03. Определите k – количество вычисленных z, произведение максимального и минимального значений z
26	Вычислите значения z, которые соответствуют каждому значению x (x=0,5, dx=0,5), по формуле $z = \frac{x^3 + x}{x + 2} + \sqrt{\frac{x}{x^5 + \sin x^5}}$. Считайте z до тех пор, пока подкоренное выражение больше или равно 0,01. Определите k – количество вычисленных z, минимальное значение среди значений $z \le 0$
27	Вычислите значения z, которые соответствуют каждому значению x (x=0,5, dx=0,01), по формуле $z = \frac{x^5 - x^2}{x + 2} + \sqrt{\frac{x}{\sin \frac{\pi x^5}{12}}}$. Считайте z до тех пор, пока подкоренное выражение больше или равно 0,05. Определите k – количество вычисленных z, среднее арифметическое вычисленных z
28	Вычислите значения z, которые соответствуют каждому значению x (x=1, dx=0,5), по формуле $z = x^5 - x^2 \cdot \sqrt{\frac{x}{\frac{x^5}{\cos^2 x}}}$. Считайте z до тех пор, пока подкоренное выражение больше или равно 0,01. Определите k – количество вычисленных z, вычислите $\prod z + \sum z$

29	Вычислите значения z, которые соответствуют каждому значению x (x=0,5, dx=0,05), по формуле $z = \ln x^2 \sqrt{\frac{x}{x^5 \cdot 5}}$. Считайте z до тех пор, пока подкоренное выражение больше или равно 0,05. Определите k – количество вычисленных z, вычислите $\prod_{z<0} z + \sum_{z>0} z$
30	Вычислите значения z, которые соответствуют каждому значению x (x=1, dx=0,5), по формуле $z = \cos(3x) \cdot \sqrt{\frac{5x^2}{3 + x^3 \cdot 2x}}$. Считайте z до тех пор, пока подкоренное выражение больше или равно 0,05. Определите k – количество вычисленных z, вычислите сумму положительных значений z, среднее арифметическое отрицательных значений

Варианты по теме «Программирование алгоритмов итеративных циклических структур»

№ варианта	Задача
1	Вычислите с точностью ε = 0.00001 константу Эйлера (основание натурального логарифма), воспользовавшись разложением в ряд: $e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!} + \ldots$ Сравните результат со значением, полученным с помощью соответствующей встроенной функции
2	Вычислите и выведите те члены последовательности $x, \frac{x^2}{2!}, \frac{x^3}{3!}, \ldots \frac{x^n}{n!}, \ldots,$ значения, которых больше ε = 0.001 при x = 0.2
3	Вычислите с точностью ε = 0.00001 значение функции $y = \sqrt{x}$ при x = 2, воспользовавшись рекуррентной формулой: $y_{i+1} = 0,5\left[y_i + \frac{x}{y_i} \right]; \quad i = 0,1,2\ldots; y_0 = \frac{x}{2}.$ Сравните результат со значением, полученным с помощью соответствующей встроенной функции
4	Вычислите и выведите те члены последовательности $3x, 8x^2, \ldots n(n+2)x^n, \ldots,$ значения которых больше ε = 0.01, при x = 0.6

5	Вычислите с точностью ε = 0.00001, воспользовавшись разложением в ряд: $\operatorname{arctg}(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots + (-1)^n \frac{x^{2n+1}}{2n+1} + \ldots$ и соотношением $\pi = 4 \cdot \operatorname{arctg}(1)$ Сравните результат со значением, полученным с помощью соответствующей встроенной функции
6	Вычислите с точностью ε = 0.00001 значение функции $y = \frac{1}{\sqrt{x}}$ при x = 2, воспользовавшись формулой: $y_{i+1} = 1.5y_i - 0.5xy_i^3; i = 0,1,2,\ldots; y_0 = 1$ Сравните результат со значением, полученным с помощью соответствующей встроенной функции
7	Вычислите sin 0.5 с точностью ε = 0.0001, воспользовавшись разложением в ряд: $\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1)!} + \ldots$ Сравните результат со значением, полученным с помощью соответствующей встроенной функции
8	Для функции $z = \frac{x^k}{k(x-0.5)}$ найдите наименьшее целое положительное значение k, при котором z < 0.001 (x=0.5)
9	Вычислите cos 0.6 с точностью ε = 0.00001, воспользовавшись разложением в ряд: $\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots + (-1)^n \frac{x^{2n}}{(2n)!} + \ldots$ Сравните результат со значением, полученным с помощью соответствующей встроенной функции
10	Вычислите с точностью ε = 0.0001 корень уравнения x – cosx = 0, воспользовавшись формулой: $X_{i+1} = \cos x_i$; i = 0,1,2…, $x_0 = 0$. Проверьте правильность решения подстановкой найденного корня в уравнение
11	Вычислите и выведите те члены последовательности, $\frac{x^2}{2!}, -\frac{x^3}{3!}, \ldots (-1)^n \frac{x^n}{n!}, \ldots,$ значения которых по модулю больше ε = 0.001 при x = 0.5
12	Для функции $z = \frac{x^k}{(x+k)^2}$ при x = 2.5 найдите наименьшее целое положительное значение k, при котором z > 100

13	Вычислите корень уравнения $f(x) = x^4+2x^2-x-1=0$ с точностью $\varepsilon=0.0001$, воспользовавшись итерационной формулой $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}; i = 0,1,2\ldots; x_0 = 0.$ Проверьте правильность решения подстановкой найденного корня в уравнение
14	Вычислите значение $\sqrt{2}$ с точностью $\varepsilon = 0.00001$, воспользовавшись представлением в виде в виде цепной дроби: $\sqrt{2}-1 = \cfrac{1}{2+\cfrac{1}{2+\cfrac{1}{\ldots}}}$ Значение дроби равно пределу числовой последовательности, члены которой вычисляются по рекуррентной формуле до достижения заданной точности $a_n = \frac{1}{2+a_{n-1}}, n = 1,2,3,\ldots, a_0 = 0,5.$ Сравните результат со значением, полученным с помощью соответствующей встроенной функции
15	Вычислите и выведите те члены последовательности $\frac{x^3}{3}, \frac{-x^5}{15}, \ldots, (-1)^{n+1}\frac{x^{2n+1}}{4n^2-1},$ значения которых по модулю больше $\varepsilon = 0.001$ при $x = 0.3$
16	Найдите наименьшее целое положительное n, при котором: $1+\frac{1}{\sqrt{2}}+\frac{1}{\sqrt{3}}+\ldots+\frac{1}{\sqrt{n}}+\ldots > 4$
17	Вычислите sh 0.3 с точностью до $\varepsilon= 0.00005$, воспользовавшись разложением в ряд: $sh(x) = x+\frac{x^3}{3!}+\frac{x^5}{5!}+\ldots+\frac{x^{2n+1}}{(2n+1)!}+\ldots$ Сравните результат со значением, полученным с помощью встроенной функции для вычисления e^x, используя соотношение: $sh(x) = \frac{e^x-e^{-x}}{2}$
18	Вычислите корень уравнения $x-0.5(\sin x^2-1)=0$ с точностью $\varepsilon = 0.001$, воспользовавшись итерационной формулой: $x_{i+1} = 0,5(\sin_i^2-1); i = 0,1,2,\ldots, x_0 = -0,25$ Проверьте правильность решения подстановкой найденного корня в уравнение

19	Вычислите ln(2) с точностью ε = 0.001, воспользовавшись представлением в виде ряда: $\ln 2 = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \ldots + (-1)^{n+1}\frac{1}{n} + \ldots$ Сравните результат со значением, полученным с помощью соответствующей встроенной функции
20	Вычислите с точностью ε = 0.00001 корень уравнения $f(x) = tg(x) - x = 0$ воспользовавшись итерационной формулой $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}, i = 0,1,2,\ldots, x_0 = 4.6.$ Проверьте правильность решения подстановкой найденного корня в уравнение
21	Вычислите ch 0.7 с точностью до ε = 0.00005, воспользовавшись разложением в ряд: $ch(x) = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \ldots + \frac{x^{2n}}{(2n)!} + \ldots$ Сравните результат со значением, полученным с помощью встроенной функции e^x, используя соотношение: $ch(x) = \frac{e^x + e^{-x}}{2}$
22	Пусть $y_0 = 0; y_k = \frac{y_{k-1} + 1}{y_{k-1} + 2}; k = 1, 2, \ldots.$ Дано действительное число ε>0. Найдите первый член y_n, для которого выполнено условие $y_n - y_{n-1} < \varepsilon$
23	Дано действительное a>0. Последовательность $x_0, x_1, \ldots$ образована по закону $x_0 = \begin{cases} \min(2a, 0.95), & \text{при } a < 1 \\ a/5, & \text{в остальных случаях.} \end{cases}$ $x_n = 4/5x_{n-1} + a/5x_{n-1}^4, \; n = 1,2,\ldots$ Найдите первый член x_n, для которого $\frac{5}{4}a\lvert x_{n+1} - x_n\rvert < 10^{-6}$ Вычислите для найденного значения x_n разность $a - x_n^5$.
24	Дано действительное b>0. Последовательность $a_1, a_2, \ldots$ образована по следующему закону: $a_1 = b; a_k = a_{k-1} - 1/k, k = 2, 3,..$ Найдите первый отрицательный член последовательности $a_1, a_2, \ldots$
25	Дано действительное b<0. Последовательность $a_1, a_2, \ldots$образована по следующему закону: $a_1 = b; a_k = (a_{k-1} + 1)/(1 - \sin^2 k), k = 2, 3, \ldots .$ Найдите первый неотрицательный член последовательности

26	Дано действительное x. Вычислите приближенное значение бесконечной суммы с точностью ε=0,00001: $(x-1)/x+(x-1)^2/(2x^2)+(x-1)^3/(3x^3)+\ldots$ $(x>1/2)$
27	Дано действительное x. Вычислите приближенное значение $1/x+1/(3x^3)+1/(5x^5)+\ldots$ $(x>1)$ бесконечной суммы с точностью ε=0.0001
28	Вычислите приближенное значение бесконечной суммы с точностью ε=0,0001 (справа от суммы дается ее точное значение, с которым можно сравнить полученный результат): $1+1/2^2+1/3^2+\ldots$ $\pi^2/6$
29	Вычислите приближенное значение бесконечной суммы с точностью ε=0,0001 (справа от суммы дается ее точное значение, с которым можно сравнить полученный результат): $1-1/3+1/5-1/7+\ldots$ $\pi/4$
30	Вычислите приближенное значение бесконечной суммы с точностью ε=0,0001 (справа от суммы дается ее точное значение, с которым можно сравнить полученный результат): $1/(1*3)+1/(2*4)+1/(3*5)+\ldots$ 3/4
31	Вычислите приближенное значение бесконечной суммы с точностью ε=0,0001 (справа от суммы дается ее точное значение, с которым можно сравнить полученный результат): $1/(1*2*3)+1/(2*3*4)+1/(3*4*5)+\ldots$ 1/4
32	Даны действительные числа x, ε ($x\neq 0$, $\varepsilon>0$). Вычислите с точностью ε=0,001: $\sum_{k=0}^{\infty}\frac{(-1)^k x^{2k+1}}{k!(2k+1)}$
33	Даны действительные числа x, ε ($x\neq 0$, $\varepsilon>0$). Вычислите с точностью ε: $\sum_{k=0}^{\infty}\frac{(-1)^k x^{4k+1}}{(2k)!(4k+1)}$
34	Даны действительные числа *x*, ε ($x\neq 0$, $\varepsilon>0$). Вычислите с точностью ε: $\sum_{k=0}^{\infty}\frac{x^{2k}}{2^k k!}$
35	Дано действительное число x. Последовательность $a_1,a_2,\ldots$ образована по указанному ниже закону. Получите сумму $a_1+\ldots+a_k$, где k – наименьшее целое число, удовлетворяющее двум условиям: $k>10$ и $\lvert a_{k+1}\rvert<10^{-5}$: $a_n=\frac{(-1)^n x^{2n}}{n(n+1)(n+2)}$

Варианты по теме «Программирование алгоритмов формирования и обработки списков»

№ варианта	Задача
1	Сформируйте список x= [-1.5, 0.1, 12, 0, -2.2, 0.5, -1, 0.3]. Вычислите и выведите сумму и количество элементов списка, принадлежащих отрезку [0,1]
2	Сформируйте список a = [5, -2, 0, 3, 4, 12, 7]. Вычислите и выведите среднее арифметическое значение положительных элементов списка
3	Сформируйте список x = [-1.5, 0, 0.8, 2.2, 3, 0.5, 0.1]. Перепишите элементы списка, принадлежащие отрезку [-1,1], в список y и выведите его
4	Сформируйте списки a [5] и b [5], состоящие из произвольных чисел и определите, в каком из списков больше положительных элементов. Выведите новый список c, состоящий из положительных элементов списков a и b
5	Сформируйте список z = [-2, 0, 3.5, 7, -12, 5, -1, 3]. Расположите в списке r сначала положительные, а затем неположительные элементы списка z. Выведите список r
6	Сформируйте список a=[2.35,-4.15,0,-3.1, 7.8, 6.3,-3.05,1.5]. Найдите и выведите среднее геометрическое положительных элементов списка a и индекс элемента, наиболее близкого к среднему геометрическому
7	Сформируйте список a[10], элементы которого должны быть сформированы по правилу $a[i+1]=a[i]+i^2$; a[1]:=2 и вычислите среднее арифметическое его элементов
8	Сформируйте список y=[2.5, 4.9, 10.2, -7.12, 3.1, -2, 6]. Сформируйте из него новый список z, элементами которого будут являться положительные элементы списка y, и упорядочите по возрастанию список z
9	Сформируйте список y=[-6.3, 0.8, 12, -4, 13, 2.5, 7, 8, -9, 10]. Найдите и выведите максимальный и минимальный элементы списка. Поменяйте их местами. Выведите полученный список y
10	Введите списки x=[4.1, 16, 0, -3.2, 12] и y=[4, 5.1, 6]. Объедините их в один список z, поместив элементы списка y между третьим и четвертым элементами списка x. Выведите список z
11	Введите списки z = [0, 1.6, 6.4, 3.8, -7, 1, -2] и a=[5,4,6.4,1]. Найдите среди элементов списков a и z два одинаковых элемента с наименьшими индексами и выведите их значения и индексы

12	Сформируйте список n= [3, 5, 7, 9, 11, 13, 15]. Переставьте элементы списка n в обратном порядке и выведите его
13	Сформируйте список m= [-1, 0, 10, -3, -5, 6, -2, 3, 4]. Сформируйте и выведите список n, элементами которого являются индексы положительных элементов списка m
14	Сформируйте список l = [13, 4, -2, 6, 7, -1, -5, 2, -3, 4]. Вычислите и выведите m[0]n[0]+m[1]n[1]+…+m[k]n[k], где m[0], m[1],…m[p] – отрицательные элементы списка l, взятые в порядке их следования; n[0], n[1],…n[q] – положительные элементы списка l, взятые в обратном порядке их следования; k = min[p,q]
15	Сформируйте список k = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. Переставьте пары элементов k[i], k[i+1], где i = 0, 2, 4, 6, 8. Выведите полученный список
16	В списке, состоящем из 15 вещественных элементов (значения элементов списка задайте, используя генератор случайных чисел), вычислите: 1) сумму отрицательных элементов списка; 2) произведение элементов списка, расположенных между максимальным и минимальным элементами
17	Сформируйте список x=[6,3.8, -9.3, 2.87,16, 5, 0.2, -3.1,1,10]. Сформируйте и выведите список Y, вычислив его по правилу: $\begin{cases} Y_0 = \sqrt{X_0},\ Y_1 = \sqrt{X_0 + X_1},\ Y_2 = \sqrt{X_0 + X_1 + X_2}, \text{ если подкор. выражение неотр.} \\ Y_i = 0, \text{ в противном случае} \end{cases}$
18	Сформируйте список m=[6, 10, 7, 14, 12, 12,-2, 3,-9, 6,-10]. Вычислите и выведите количество и сумму тех элементов списка, которые делятся на 2 и не делятся на 3
19	Сформируйте список m = [14, 6, 3, 0, 7, 12, -3, 1, 5, 2]. Вычислите и выведите произведение элементов списка, кратных 3.
20	Сформируйте список l = [7, 6, 15, 17, 12, -12, 4, 0, -10, -22]. Замените в списке нулями те элементы, модуль которых при делении на 5 дает в остатке 2. Выведите полученный список
21	Сформируйте список k = [1, 2, 3, 4, 6, 5, 8, 9, 10]. Если элементы списка образуют возрастающую последовательность выведите сообщение "ДА"; в противном случае – сообщение "НЕТ"
22	Введите упорядоченный список q = [1.5, 2, 3.1, 4.2, 6, 7.5, 8.3, 9]. Удалите из списка элемент с индексом 4 и вставьте элемент с вводимым значением s так, чтобы не нарушилась упорядоченность. Выведите полученный список
23	В списке, состоящем из 15 вещественных элементов (значения элементов списка задайте, используя генератор случайных чисел), вычислите:

	1) сумму положительных элементов списка; 2) произведение элементов списка, расположенных между максимальным по модулю и минимальным по модулю элементами
24	Сформируйте список b=[-15.1, 0.8, 32.3, 7.5, -1.5, 2.4, -6.3, 15.5]. Подсчитайте и выведите среднее арифметическое значение элементов списка и количество элементов, больших среднего арифметического
25	В списке, состоящем из 20 вещественных элементов (значения элементов списка задайте, используя генератор случайных чисел), вычислите: 1) номер минимального элемента списка; 2) сумму элементов списка, расположенных между первым и вторым отрицательными элементами. Преобразуйте список таким образом, чтобы сначала располагались все элементы, модуль которых не превышает 1, а потом – все остальные
26	В списке, состоящем из 20 целых элементов (значения элементов списка задайте, используя генератор случайных чисел), вычислите: 1) номер максимального элемента списка; 2) произведение элементов списка, расположенных между первым и вторым нулевыми элементами. Преобразуйте список таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй половине – элементы, стоявшие в четных позициях
27	Введите списки x = [-6, 0.5, 0.12, 13, -10.1] и y = [13, 2.1, 14, 6, -2]. Создайте список r [10] такой, в котором элементы с нечетными номерами являются элементами списка x, с четными номерами – списка y. Выведите список r
28	Введите списки a = [-2, 0, 3.1, 4.6, 10], b = [4, 7, 9.1, 12, 63]. Сформируйте из элементов списков a и b список c. Упорядочите его по возрастанию и выведите список c
29	В списке, состоящем из 20 целых элементов (значения элементов списка задайте, используя генератор случайных чисел), вычислите: 1) произведение элементов списка с четными номерами; 2) преобразуйте список таким образом, чтобы сначала располагались все положительные элементы списка, а потом – все отрицательные (элементы равные нулю, считать положительными)
30	В списке, состоящем из 20 элементов целого типа (значения элементов списка задайте, используя генератор случайных чисел), вычислите: 1) минимальный элемент списка; 2) сумму элементов списка, расположенных между первым и последним положительными элементами.

	Преобразуйте список таким образом, чтобы сначала располагались все элемент равные нулю, а потом – все остальные
31	Сформируйте список x= [-1, 2, -3, 4, -5, 0, 6, 8, 9]. Перепишите в список y подряд положительные элементы списка x. Выведите список y
32	Сформируйте список g = [-3.1, 2.8, 5, 7.7, -7.5, 7.6, 3, 0]. Определите и выведите минимальный элемент списка и его номер
33	Сформируйте список b = [7.35, 0.12, -7, 3.12, 2.87, -4.12, 5.32, 0, 6.5]. Определите и выведите максимальный элемент списка и его номер
34	В списке, состоящем из 20 вещественных элементов (значения элементов списка задайте, используя генератор случайных чисел), вычислите: 1) максимальный элемент списка; 2) сумму элементов списка, расположенных до последнего положительного элемента
35	В списке, состоящем из 20 вещественных элементов (значения элементов списка задайте, используя генератор случайных чисел), вычислите: 1) сумму элементов списка с нечетными номерами; 2) сумму элементов списка, расположенных между первым и последним отрицательными элементами

Варианты по теме «Программирование алгоритмов формирования и обработки вложенных последовательностей»

№ варианта	Задача
1	Сформируйте вложенную последовательность L [9,9] по правилу: $$L_{i,j} = \begin{cases} i \cdot j \cdot r, & \text{если } i \neq j \\ 0, & \text{в противном случае,} \end{cases}$$ где r – случайное число из отрезка [0,1]. Найдите в каждой строке наибольший элемент и поменяйте его местами с элементом главной диагонали. Выведите полученную последовательность
2	Сформируйте вложенную последовательность L [10,10] по правилу: $$L_{i,j} = -5 + 10 \cdot r,$$ где r – случайное число из отрезка [0,1]. Запишите на место отрицательных элементов последовательности

	нули, а на место положительных элементов – единицы. Выведите нижнюю треугольную последовательность
3	Сформируйте вложенную последовательность N [12,12] по правилу: $$N_{i,j} = (-1)^j \cdot (i + j).$$ Вычислите и выведите сумму и количество положительных элементов последовательности, находящихся под главной диагональю
4	Сформируйте вложенную последовательность N [15,10] по правилу: $$N_{i,j} = \begin{cases} \dfrac{i+j}{i-j}, & \text{если } i \neq j \\ 1, & \text{в противном случае.} \end{cases}$$ Найдите строки с наибольшей и наименьшей суммой элементов. Выведите найденные строки и суммы их элементов
5	Сформируйте вложенную последовательность N [10,10] по правилу: $$N_{i,j} = -10 + 20 \cdot r,$$ где r – случайное число из отрезка [0,1]. Из положительных элементов последовательности N сформируйте последовательность M[10,max], где max – максимальное число положительных элементов строки последовательности N, располагая их в строках последовательности M подряд. Запишите нули на место отсутствующих элементов последовательности M. Выведите полученную последовательность
6	Сформируйте вложенную последовательность K [8,8] по правилу: $$K_{ij}=8\cdot(i - j) + j.$$ Транспонируйте последовательность (поменяйте местами строки и столбцы) и выведите элементы главной диагонали и диагонали, расположенной под главной, разместив их в двух строках экрана
7	Сформируйте вложенную последовательность K [7,8] по правилу: $$k_{i,j} = 1000 \cdot r,$$ где r – случайное число из отрезка [0,1]. Найдите наибольший и наименьший элементы последовательности и поменяйте их местами. Выведите полученную последовательность
8	Сформируйте вложенную последовательность N [12,12] по правилу: $$N_{i,j} = (-1)^j \cdot (i + j), .$$ Вычислите и выведите сумму и количество положительных элементов последовательности, находящихся под главной диагональю
9	Сформируйте вложенную последовательность M [8,8] по правилу: $M_{i,j} = -10 + 100 \cdot r$, где r – случайное число из отрезка [0,1]. Найдите в каждой строке последовательности максимальный и минимальный элементы и поменяйте их местами соответственно с первым и последним элементами строки. Выведите полученную последовательность

10	Сформируйте вложенную последовательность K [10,12] по правилу $$K_{i,j} = -20 + 40 \cdot r,$$ где r – случайное число из отрезка [0,1]. Определите в каждом столбце количество простых чисел и запишите его в соответствующий элемент последовательности L. Выведите последовательность L
11	Сформируйте вложенную последовательность L [8,8] по правилу: $$L_{i,j} = 250 \cdot r,$$ где r – случайное число из отрезка [0,1]. Среди элементов последовательности L найдите и выведите наибольший элемент и его индексы. Все элементы последовательности разделите на найденный наибольший элемент и выведите полученную последовательность
12	Сформируйте вложенную последовательность K [5,5] по правилу: $M_{i,j} = -30 + 60 \cdot r$, где r – случайное число из отрезка [0,1]. Поменяйте местами элементы строки с номером k с элементами строки с номером t (значения k и t вводятся с клавиатуры). Выведите полученную последовательность M по строкам
13	Сформируйте вложенную последовательность X [4,4] по правилу: $$X_{i,j} = \frac{i}{i+j}.$$ Вычислите и выведите сумму и количество элементов последовательности X, находящихся под главной диагональю и удовлетворяющих условию $0.2 < X_{i,j} < 0.3$
14	Сформируйте вложенную последовательность Z [10,4] по правилу: $$z_{ij} = \begin{cases} \sin(i+j), & \text{если } i < j \\ 1, & \text{если } i = j \\ \operatorname{arctg}\dfrac{i+j}{2i+3j}, & \text{если } i > j. \end{cases}$$ Вычислите максимальное значение суммы модулей элементов в столбцах последовательности и выведите этот столбец
15	Сформируйте вложенную последовательность A [9,3] по правилу $$A_{i,j} = \sin(i + j/2).$$ Определите наименьший элемент в каждой строке последовательности и запишите его в соответствующий элемент списка B. Выведите список B
16	Сформируйте вложенную последовательность B [6,3] по правилу: $$B_{i,j} = \sin\frac{i^2 - j^2}{6}$$ и вычислите выражение

	$\sqrt{\sum_{i=1}^{6}\prod_{j=1}^{3}\left(B_{ij}\right)^2}$
17	Сформируйте вложенную последовательность B [4,4] по правилу: $B_{i,j} = -3 + 6r,$ где r – случайное число на отрезке [0,1]. Сформируйте и выведите последовательность A[4,4], получаемую из последовательности B перестановкой в каждой строке наибольшего по абсолютной величине элемента с диагональным
18	Сформируйте вложенную последовательность N [10,10] по правилу: $N_{i,j} = i + 2j.$ Получите и выведите список K[10], где K_i – наименьшее из значений элементов, находящихся в начале i-й строки последовательности N до элемента, принадлежащего главной диагонали включительно
19	Сформируйте вложенную последовательность X [4,4] по правилу: $X_{i,j} = \cos\left(i^2 + j^2\right).$ Получите и выведите список Y[4], где Y_i – значение первого по порядку положительного элемента i-й строки; если такого элемента нет, то принять Y_i=1
20	Сформируйте вложенную последовательность Z [10,3] по правилу: $Z_{i,j} = -8.2 + 16.4 \cdot r,$ где r – случайное число из отрезка [0,1]. Получите и выведите список P, где P_i – сумма элементов, расположенных за первым отрицательным элементом в i-й строке; если все элементы строки неотрицательны, то принять P_i=100
21	Сформируйте вложенную последовательность F [10,3] по правилу: $F_{i,j} = -1 + 2 \cdot r,$ где r – случайное число из отрезка [0,1]. Получите и выведите список R, где R_j – сумма элементов, предшествующих последнему отрицательному элементу j-го столбца; если все элементы столбца неотрицательны, то принять R_j= -1
22	Сформируйте вложенную последовательность R [8,3] по правилу: $R_{i,j} = i \cdot e^{-j}.$ Найдите и выведите значение и индексы элемента, являющегося одновременно наименьшим в своей строке и наибольшим в своем столбце. При отсутствии такого элемента выведите сообщение
23	Сформируйте вложенную последовательность N [5,5] по правилу: $N_{i,j} = \begin{cases} i + 5j, & \text{если } i \le 3 \\ 7i + 2\left(j - 1\right), & \text{если } i > 3. \end{cases}$

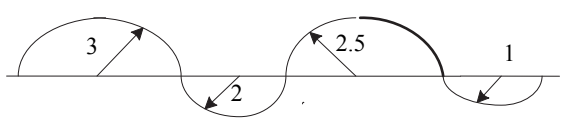
	Найдите и выведите значение и индексы двух одинаковых элементов. Если таковых не окажется, выведите сообщение об этом
24	Сформируйте вложенную последовательность M [5,5] по правилу: $$M_{i,j} = 10i + j,$$ Поменяйте местами третью строку с третьим столбцом и выведите полученную последовательность
25	Сформируйте вложенную последовательность K [8,8] по правилу: $$K_{i,j} = -15 + 30 \cdot r,$$ где r – случайное число из отрезка [0,1]. Получите и выведите последовательность L [7,7], полученную путем удаления из последовательности K строки и столбца, на пересечении которых находится наибольший по модулю элемент
26	Сформируйте вложенную последовательность M [7,7] по правилу: $$M_{i,j} = -5 + 10 \cdot r,$$ где r – случайное число из отрезка [0,1]. Введите список K = [-5, -3, -1, 1, 3]. Замените нулями в последовательности M те элементы, для которых имеются равные среди элементов списка K. Выведите полученную последовательность M
27	Сформируйте вложенную последовательность T [4, 4] по правилу: $t_{i,j} = 5(i + j) - j$. Удалите из нее столбцы, содержащие элементы меньше 10
28	Сформируйте вложенную последовательность K[6,3] по правилу $$K_{i,j} = -10 + 20 \cdot r,$$ где r – случайное число из отрезка [0,1]. Сформируйте и выведите последовательность L[6,3], получаемую из последовательности K перестановкой строк: первой с последней, второй с предпоследней и т. д.
29	Сформируйте вложенные последовательности A[4,4] и B[4,4] по правилу: $$A_{i,j} = \begin{cases} \frac{1}{i+j+1}, i \le j \\ \frac{1}{i+j-1}, i > j \end{cases}, \quad B_{i,j} = \begin{cases} \frac{1}{i+j+1}, i < j \\ 0, i = j \\ \frac{-1}{i+j-1}, i > j \end{cases}.$$ Получите последовательность R[4,4] путем умножения элементов каждой строки последовательности A на наибольший из элементов соответствующей строки последовательности B. Выведите последовательность R

30	Сформируйте вложенные последовательности K[3,3] и L[3,3] по правилу: $K_{i,j} = \frac{1}{i + j - 1}$; $L_{i,j} = \frac{1}{i + j}$. Получите последовательность M путем прибавления к элементам каждого столбца последовательности K произведения элементов соответствующей строки последовательности L. Выведите последовательность M
31	Сформируйте вложенную последовательность M[6,6] по правилу: $M_{i,j} = -30 + 60 \cdot r$, где r – случайное число из отрезка [0,1]. Сформируйте и выведите последовательность L, получающуюся из массива M перестановкой столбцов: первого с последним, второго с предпоследним и т. д.
32	Сформируйте вложенную последовательность A [3, 4] по правилу: $a_{ij} = \frac{i^2 + j^2}{2}$. Сформируйте и выведите список B[4], каждый элемент которого есть среднее арифметическое элементов, соответствующей строки последовательности a
33	Сформируйте вложенную последовательность M[5, 6] по правилу: $$m_{ij} = 2 + 300r,$$ где r – случайное число из отрезка [0, 1]. В каждой строке последовательности подсчитайте суммы тех элементов, которые являются простыми числами, и запишите значения этих сумм в список k
34	Сформируйте вложенную последовательность C [5, 5] по правилу: $$c_{ij} = \begin{cases} \cos(i + j), & \text{если } i < j \\ 0, & \text{если } i = j \\ \operatorname{tg}(i - j), & \text{если } i > j. \end{cases}$$ Получите из последовательности c список p, который упорядочен по возрастанию своих значений
35	Сформируйте вложенную последовательность D [3, 2] по правилу: $d_{ij} = \frac{i^2 - j^2}{2}$. Перепишите отрицательные элементы последовательности d в список t

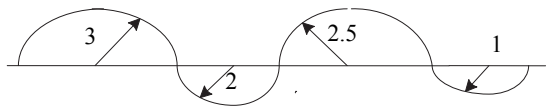
Варианты по теме «Работа с функциями»

№ варианта	Задача
1	Определите периметры правильных n – угольников (10 – угольника, 50 – угольника, 100 – угольника), вписанных в окружность заданного радиуса R. **Назначение функции:** вычисление стороны правильного n-угольника $a = 2 \cdot R \cdot \sin\frac{180}{n}$; вычисление периметра n-угольника $p = n \cdot a$
2	Определите длины всех медиан треугольника, заданного длинами сторон a, b, c. **Назначение функции:** вычисление медианы, проведенной к стороне $a : m = \sqrt{2b^2 + 2c^2 - a^2}$
3	Определите углы между тремя векторами, направленными из общей начальной точки с координатами (0,0) в конечные точки: точку (2;5); точку (7;6); точку (9;3). **Назначение функции:** вычисление угла между 2-мя векторами, проведенными из точки (0, 0) в точки (x1, y1) и (x2, y2) по формуле $\cos a = \frac{x_1 \cdot x_2 + y_1 \cdot y_2}{\sqrt{x_1^2 + y_1^2} \cdot \sqrt{x_2^2 + y_2^2}}$;
4	Вычислите площадь пятиугольника, заданного прямоугольными координатами своих вершин: A1(3;2), A2(9;6), A3(14;2), A4(10;-3), A5(7;-2). Используйте формулу площади треугольника: $S = \frac{1}{2} \cdot \left\lvert (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1) \right\rvert$ **Назначение функции:** вычисление площади треугольника по заданным координатам его вершин: (x1;y1), (x2;y2), (x3;y3)
5	Определите длины всех биссектрис треугольника, заданного длинами сторон a, b, c. **Назначение функции:** вычисление биссектрисы угла α $La = \frac{\sqrt{bc\left((b+c)^2 - a^2\right)}}{b+c}$.
6	Вычислите расстояние между двумя точками A и B, заданными сферическими координатами. Соотношение между сферическими координатами и декартовыми

	$x = r \cdot \sin\theta \cdot \cos\varphi;$ $y = r \cdot \sin\theta \cdot \sin\varphi;$ $z = r \cdot \cos\theta;$ $0 \le r \le \infty; -\pi < \varphi \le \pi; 0 \le \theta \le \pi;$ $R = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2 + (z_B - z_A)^2}$ **Назначение функции:** вычисление декартовых координат точки по заданным сферическим координатам
7	Вычислите значение площади полной поверхности треугольной пирамиды, если известны длины всех ребер: $\|AB\| = 3; \|AD\| = 5; \|DB\| = 4; \|DC\| = \sqrt{35}; \|BC\| = \sqrt{19}; \|AC\| = 5.$ Для вычисления площади треугольника использовать формулу Герона: $S = \sqrt{p(p-a)(p-b)(p-c)}$; $p = \frac{a+b+c}{2}$, где a, b, c – длины сторон треугольника. **Назначение функции:** вычисление площади треугольника по формуле Герона
8	Определите высоту, на которой будет мяч, подброшенный вертикально вверх с высоты y_0=1м и начальной скоростью V_0=20 м/сек через время t = 1 с, 3 с и 4 с. **Назначение функции:** вычисление высоты в момент t: $y(t) = y_0 + v_0\, t - g \cdot t^2 / 2$, где g = 9.8 м/сек
9	Определите площади правильных n-угольников (10-угольника; 50-угольника; 100-угольника), вписанных в окружность радиуса R. **Назначение функции:** вычисление стороны правильного n-угольника $a = 2R\sin\frac{180°}{n}$, где $r = R\cos\frac{180°}{n}$ – радиус вписанной окружности; площадь n-угольника $S = \frac{1}{2} a \cdot n \cdot r.$
10	Определите площадь кольца, внутренний радиус которого равен R1; а внешний R2 (R2>R1). **Назначение функции:** вычисление площади круга радиуса R: $S = \pi R^2$
11	Вычислите полярные координаты трех точек, заданных прямоугольными координатами в правой полуплоскости. Формулы преобразования координат: $r = \sqrt{x^2 + y^2}$; $\varphi = \operatorname{arctg}\frac{y}{x}$. **Назначение функции:** вычисление полярных координат по заданным прямоугольным координатам точки правой полуплоскости

12	Определите площадь каждого из трех секторов с радиусами R1, R2, R3 и с центральными углами α, β, γ. **Назначение функции:** вычисление площади сектора радиуса R с центральным углом α (в градусах) равна $S = \pi R^2 \cdot \frac{\alpha}{360}$
13	Вычислите $y = Sh(x) + tg(x+1) - tg^2(2 + Sh(x-1))$ **Назначение функции:** вычисление $Sh(x) = \frac{\ell^x - \ell^{-x}}{2}$
14	Определите стороны треугольника, заданного величинами своих углов и радиусом описанной окружности. **Назначение функции:** применить теорему синусов $\frac{a}{\sin\alpha} = \frac{b}{\sin\beta} = \frac{c}{\sin\gamma} = 2R \quad (\alpha + \beta + \gamma = 180°)$.
15	Определите площадь каждого из трех кругов, ограниченных тремя окружностями, длины которых L1, L2, L3 известны. **Назначение функции:** вычисление площади круга $S = \pi R^2$ предварительно следует вычислить R по формуле $L = 2\pi R$
16	Определите углы треугольника, длины сторон которого a, b, c заданы. **Назначение функции:** примените теорему половинного угла $tg\frac{\gamma}{2} = \sqrt{\frac{(p-a)\cdot(p-b)}{p\cdot(p-c)}}$ где $p = \frac{a+b+c}{2}$, Y – угол, противолежащий стороне c
17	Вычислите $y = \frac{3x^3 - 4x^2 + 2}{2x^2 + 3x - 1}$; $z = \frac{8x^3 + 2x^2 + x}{x^3 + 4x - 2}$. **Назначение функции:** вычисление $f(a,b,c,d,x) = ax^3 + bx^2 + cx + d$.
18	Определите общую длину дуги, образованной полуокружностями  **Назначение функции:** вычисление длины половины окружности $L = \pi R$
19	Вычислите стороны треугольника A и B при условии, что заданы сторона C и углы треугольника α и β. **Назначение функции:** вычисление стороны треугольника по формуле $A = C\frac{\sin\alpha}{\sin\gamma}; \gamma = 180° - \alpha - \beta$, где α – угол, противолежащий стороне A, Y – угол, противолежащий стороне C

20	Вычислите значения медиан треугольника, сторонами которого являются медианы исходного треугольника со сторонами a, b, c. **Назначение функции:** вычисление длины медианы, проведенной к стороне **a**: $ma = 0,5\sqrt{2b^2 + 2c^2 - a^2}$
21	Вычислите $y = \frac{2.56x^2 + 3.4x + 8.1}{3.6x^2 - 5.2}; z = \frac{3x^2 - 2}{7x + 2\pi};$ **Назначение функции:** вычисление $f(a,b,c,x) = ax^2 + bx + c$
22	Вычислите $y = ch(x+2) - 3ch(x) + tg^2(1 - ch(2x-3))$ **Назначение функции:** вычисление $ch(x) = \frac{\ell^x + \ell^{-x}}{2}$
23	Определите длину дуги каждого из трех секторов с радиусами R1, R2, R3 и с центральными углами α, β, γ. **Назначение функции:** длина дуги сектора радиуса R с центральным углом α (в градусах) равна $L = 2\pi R \cdot \frac{\alpha}{360}$
24	Вычислите координаты точки пересечения двух прямых: $\begin{matrix} a_1x + b_1y = c_1 \\ a_2x + b_2y = c_2 \end{matrix}$, вычисляемые по формулам: $x = \frac{\Delta x}{\Delta}; y = \frac{\Delta y}{\Delta},$ где $\Delta = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}; \Delta x = \begin{vmatrix} c_1 & b_1 \\ c_2 & b_2 \end{vmatrix}; \Delta y = \begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}.$ **Назначение функции:** вычисление определителя второго порядка
25	Вычислите $y = \frac{3.5x^5 - 4x^3 + 5x + 7}{2x^5 + 4.3x^3 - 1}; z = \frac{1.8x^3 + 2x + 6}{0.5x^5 + 4x^3 - 2}.$ **Назначение функции:** вычисление $f(a,b,c,d,x) = ax^5 + bx^3 + cx + d$
26	Определите все углы треугольника при заданных значениях сторон **a, b, c.** **Назначение функции:** вычисление угла по теореме косинусов: $\alpha = \arccos(b^2 + c^2 - a^2) / (2 \times b \times c);$
27	Вычислите определенный интеграл $y = \int_a^b f(x)dx$ для функции $f(x) = \sqrt{2x+1}$ $y \cong \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{6}\right) + f(b)\right].$

28	Вычислите площадь трех кругов S1, S2 и S3 с заданными диаметрами d1, d2 и d3. **Назначение функции:** вычисление $S = \pi \times d^2 / 4$
29	Вычислите $y = \dfrac{3.5x^3 - 2.8x^2 + 6.7}{x^3 + x^2 - 1}$; $z = \dfrac{12.8x^3 + 0.2x^2 + 0.76}{5.5x^3 + 0.4x^2 - 2.8}$. **Назначение функции:** вычисление $f(a,b,c,x) = ax^3 + bx^2 + c$.
30	Определите общую площадь фигуры, образованной полукругами  **Назначение функции:** вычисление площади полукруга $S = \dfrac{\pi R^2}{2}$.
31	Вычислите значения: g(1.2,s)+g(t,s)-g(2s-1,s+t), где s и t – действительные числа. **Назначение функции:** вычисление $g(a,b) = \dfrac{a^2 + b^2}{a^2 + 2ab + 3b^2 + 4}$
32	Вычислите $y = \dfrac{3.3x^4 - 8.5x + 7}{2x - 5}$ и $z = \dfrac{4x + 8}{5x^4 - 3x}$. **Назначение функции:** вычисление $f(a,b,c,x) = ax^4 + bx + c$.
33	Вычислите $y = \dfrac{\sin\dfrac{4\pi}{3} - 7\sin\dfrac{5\pi}{2}}{5\sin\dfrac{8\pi}{5} + \sin\dfrac{7\pi}{3}}$; $z = \dfrac{y^2 + \sin\dfrac{\pi}{2}}{1 - y + \sin\dfrac{4\pi}{5}}$. **Назначение функции:** вычисление $f(n,k) = \sin\dfrac{\pi \cdot n}{k}$.
34	Вычислите $y = \dfrac{0.35x^3 - 4.8x}{2x^3 + 4.3x - 15}$; $z = \dfrac{5.18x^3 - 6.16}{3.5x^3 + 4.9x - 7.5}$. **Назначение функции:** вычисление $f(a,b,c,x) = ax^3 + bx + c$.
35	Вычислите $Y = \dfrac{2^{3x+2} + 3^{4x}}{\left(\dfrac{1}{2}\right)^x + \left(\dfrac{1}{3}\right)^{2x}}$; $z = \sin\dfrac{x}{2} + (x+2)^{x+1}$; **Назначение функции:** вычисление $a^x = e^{x\ln a}$, $a > 0$

Варианты по теме «Обработка строковых данных»

№ варианта	Задача
1	Разработайте программу, которая проверяет, является ли введенная с клавиатуры последовательность символов целым числом, записанным в двоичной системе счисления
2	Разработайте программу, которая вычисляет среднюю длину слов во введенной с клавиатуры строке
3	Разработайте программу, которая зашифровывает введенный с клавиатуры текст. Процесс шифровки производится следующим образом: из десятичного кода каждого введенного с клавиатуры символа вычитается число десять. Получившаяся в результате вычитания величина интерпретируется как десятичный код некоторого другого символа, который и выводится на экран компьютера
4	Дано слово. Определите, является ли оно палиндромом (словом, которое читается одинаково в обоих направлениях, например «потоп»)
5	Дана строка символов. Определите самое длинное слово в строке и количество слов такой же длины
6	Дана строка символов. Определите количество слов, являющихся записью десятичного числа
7	Дана строка символов. Удалите из нее все пробелы
8	Дана строка символов. Дано слово. Удалите из строки это слово
9	Дана строка символов. Выделите подстроку между первой и второй точкой
10	Дана строка символов. Определите длину самого короткого и самого длинного слова
11	Дана строка символов. Определите, сколько слов начинается и кончается одной и той же буквой
12	Дана строка символов. Определите, сколько слов содержат две буквы «с»
13	Дана строка символов. Определите, является ли она правильным скобочным выражением. Рассматривайте только круглые скобки
14	Дана строка символов. Определите, сколько слов содержат ровно три буквы «е»
15	Строка содержит только цифры. Удалите все впереди стоящие нули
16	Дана строка символов. Подсчитайте количество знаков препинания в строке
17	Дана строка символов. Удалить из строки все запятые
18	Дана строка символов. Приведено некоторое слово. Вставьте его после каждого пробела.

19	Дана строка символов. Найдите сумму чисел, встречающихся в строке.
20	Дана строка символов. Удалите из строки все числа
21	Дана строка символов. Удалите из строки слово, имеющее наибольшую длину
22	Дана строка символов. Вставьте в строку пробел после каждого знака препинания
23	Дана строка символов. Сформируйте строку, состоящую из слов исходной строки, записанных наоборот
24	Дана строка из цифр и латинских букв. Определите, каких букв, гласных (A, E, I, O, U) или согласных, больше в этой строке
25	Из заданной строки удалите те ее части, которые заключены в кавычки (вместе с кавычками)
26	Задано слово. Замените в этом слове букву A на букву O. Если буквы A в этом слове нет, то выведите соответствующее сообщение

Варианты по теме «Работа с текстовыми файлами»

№ варианта	Задача
1	Осуществите ввод двух вложенных последовательностей A[5,5] в пустой текстовый файл. Разработайте программу умножения двух последовательностей. Результирующую последовательность выведите на экран и в текстовый файл
2	Сформируйте файл вещественных чисел. Удалите из него максимальный и минимальный элементы. Выведите результаты выполнения программы на экран
3	Создайте файл, элементами которого будут строковые переменные – буквы греческого алфавита (первые пять). Разработайте программу, с помощью которой по порядковому номеру буквы в алфавите можно определить ее название, и наоборот, по названию буквы определить ее положение в алфавите
4	Создайте файл, в котором будут храниться коэффициенты квадратного уравнения. Найдите корни квадратного уравнения и выведите на экран соответствующее сообщение
5	Разработайте программу, которая создает текстовый файл, записывает в него построчно информацию (количество строк заранее не известно, а признаком окончания ввода является ввод в конце очередной строки символа *). Подсчитайте количество строк, которое содержится в этом файле
6	Сформируйте файл исходных данных. Из компонентов исходного файла целых чисел сформируйте списки четных и нечетных

	чисел. Определите наибольший четный компонент файла и наименьший нечетный. Результат запишите в текстовый файл
7	Сформируйте файл исходных данных. Из компонентов исходного файла целых чисел сформируйте список удвоенных нечетных чисел. Упорядочите его по возрастанию элементов. Результат запишите в текстовый файл
8	Сформируйте файл исходных данных. Из компонентов исходного файла вещественных чисел сформируйте список, записав в него компоненты, расположенные в файле до минимального элемента и после максимального элемента. Результат запишите в текстовый файл
9	Сформируйте файл исходных данных. Из компонентов исходного файла вещественных чисел сформируйте список, удвоив каждый его элемент через два пробела. Результат запишите в текстовый файл и выведите на экран
10	Сформируйте файл исходных данных. В файле в виде строк хранятся анкеты студентов. Строка содержит фамилию и инициалы, год рождения, пол. Подсчитайте, сколько анкет относится к лицам мужского пола. Пол задан одной из подстрок «муж», «жен». Результат выведите на экран
11	Разработайте программу, позволяющую протестировать студента. Последовательность вопросов и варианты ответов должны находиться в текстовом файле. Количество вопросов не более пяти. Текст вопроса и ответов не должен занимать более одной строки экрана. Программа выставляет оценки «5» – все правильные ответы, «4» – более 80 % ответов верные, «3» – более 60 % ответов верные, «2» – менее 60 % ответов верные
12	Разработайте программу, которая по желанию пользователя выводит таблицу пересчета из дюймов в миллиметры на экран или в файл
13	Разработайте программу, которая создает текстовый файл, записывает в него построчно информацию (количество строк заранее не известно, а признаком окончания ввода является ввод в конце очередной строки символа *). Подсчитайте количество строк в имеющемся файле
14	Сформируйте файл исходных данных. Сформируйте список положительных чисел, делящихся на пять без остатка, используя элементы исходного списка целых чисел. Упорядочите его по убыванию элементов. Результат запишите в текстовый файл
15	Сформируйте файл исходных данных. Из компонентов исходного файла вещественных чисел сформируйте список, записав в него компоненты, расположенные в файле между минимальным и максимальным элементами. Результат запишите в текстовый файл
16	Создайте файл, элементами которого будут строковые переменные. Из компонентов исходного файла сформируйте текстовый файл, удвоив все буквы «а». Результат выведите на экран

17	Разработайте программу, которая создает файл, содержащий n целочисленных элементов (количество элементов вводит пользователь) и подсчитывает сумму элементов, содержащихся в этом файле. Результат запишите в текстовый файл
18	Сформируйте файл исходных данных. Из компонентов исходного файла целых чисел сформируйте список отрицательных чисел. Вычислите количество нулевых компонентов файла. Результат запишите в текстовый файл
19	Сформируйте файл исходных данных. Из компонентов исходного файла целых чисел сформируйте список чисел, записав в него только ненулевые компоненты, находящиеся после максимального элемента. Результат запишите в текстовый файл
20	Сформируйте файл исходных данных. Из компонентов исходного файла, содержащего символы и целые числа, найдите сумму цифр, встречающихся в нем. Результат запишите в текстовый файл и выведите на экран
21	Сформируйте файл исходных данных. Постройте конкатенацию (последовательную запись) исходного файла самого с собой. Результат запишите в текстовый файл и выведите на экран
22	Разработайте программу, которая записывает в текстовый файл фамилию и номер телефона товарища по группе в формате ***-**-**. Найдите количество номеров телефонов, у которых совпадают первые три цифры. Сформируйте результирующий файл таких телефонов. Если таких номеров нет, то программа должна выводить соответствующее сообщение
23	Сформируйте файл исходных данных. Удалите в исходном файле текст после первой точки. Результат запишите в текстовый файл и выведите на экран
24	Разработайте программу, которая создает файл, содержащий следующую введенную пользователем информацию о прочитанной им книге: номера глав и названия этих глав, причем номера должны записываться в файл как целочисленные элементы, а названия – как строковые. Количество глав в прочитанной книге пользователь вводит с клавиатуры. Выведите на экран содержимое файла, причем номер и название каждой главы должны выводиться в отдельной строке
25	Разработайте программу, которая позволяет найти нужные сведения в телефонном справочнике, хранящемся в текстовом файле. Программа позволяет запрашивать фамилию человека и выводить его телефон. Если в справочнике есть одинаковые фамилии, то программа должна вывести список всех людей, имеющих эти фамилии
26	Сформируйте файл исходных данных. Постройте конкатенацию (последовательную запись) исходного файла самого с собой, только записанного задом наперед. Результат запишите в текстовый файл и выведите на экран

Варианты по теме: «Объектно-ориентированное программирование»

№ варианта	Задача
1	Опишите класс ТОЧКА. Для точки задаются декартовы координаты x, y. Включите в описание класса методы, позволяющие вывести координаты точки на экран, рассчитать расстояние от начала координат до точки, переместить точку на плоскости на вектор (a,b) и свойство, позволяющее умножить координаты точки на скаляр
2	Опишите класс ТРЕУГОЛЬНИК, заданный длинами сторон. Включите в описание класса методы, позволяющие вывести длины сторон треугольника на экран, рассчитать периметр, площадь и высоты треугольника и свойство, позволяющее установить, существует ли треугольник с данными длинами сторон
3	Опишите класс ПРЯМОУГОЛЬНИК, заданный длинами сторон. Включите в описание класса методы, позволяющие вывести длины сторон прямоугольника на экран, рассчитать периметр, площадь и диагональ прямоугольника, изменить его размеры, умножив его длины на скаляр, и свойство, позволяющее установить, является ли данный прямоугольник квадратом
4	Опишите класс ДЕНЬГИ, заданный двумя полями, которые определяют номинал купюры и количество купюр. Включите в описание класса методы, позволяющие вывести номинал и количество купюр, определить, хватит ли денежных средств на покупку товара на сумму N рублей, и метод определения, сколько штук товара стоимости m рублей можно купить на имеющиеся денежные средства, а также свойство, позволяющее рассчитатать сумму денег
5	Опишите класс УГОЛ, заданный величиной в градусах и минутах (двумя полями). Включите в описание класса методы, позволяющие реализовать перевод в радианы, привести величину угла к диапазону 0–360°, увеличить и уменьшить угол на заданную величину и свойство, позволяющее определить, является ли угол тупым
6	Опишите класс для работы со списком целых чисел размером n. Включите в описание класса методы ввода элементов списка, вывода списка на экран, нахождения максимального и минимального элементов списка и их индексов, а также свойства, позволяющие: определить, является ли список упорядоченным по убыванию, домножить все элементы списка на скаляр
7	Опишите класс для работы со вложенной последовательностью целых чисел размером n x m. Включите в описание класса мето-

	ды ввода элементов последовательности с клавиатуры, вывода на экран, вычисления суммы элементов i-го столбца, а также свойства: для вычисления количества нулевых элементов и позволяющее установить значение всех элементов главной диагонали последовательности равным скаляру
8	Опишите класс для работы со вложенной последовательностью вещественных чисел размером n x m. Включите в описание класса методы: ввода элементов последовательности с клавиатуры, вывода на экран, сортировки элементов каждой строки последовательности в порядке убывания, а также свойства: возвращающее общее количество элементов в последовательности; позволяющее увеличить значение всех элементов последовательности на скаляр
9	Составьте описание класса многочлена вида ax^2+bx+c. Включите в класс метод вывода описания многочлена на экран и метод вычисления значения многочлена для заданного аргумента, а также свойства, позволяющие: определить, имеет ли квадратное уравнение действительные корни; умножить многочлен на на скаляр
10	Составьте описание класса многочлена вида ax+b. Включите в класс методы: вывода описания многочлена на экран, вычисления значения многочлена для заданного аргумента, вычисления корня линейного уравнения с проверкой неравенства коэффициента b нулю, а также свойство, позволяющее возвести многочлен в квадрат
11	Опишите класс ТРЕУГОЛЬНИК, заданный длиной одной стороны и двумя прилежащими углами (в градусах). Включите в описание класса методы, позволяющие вычислить две другие стороны и третий угол треугольника, и свойство, доступное только для чтения, позволяющее установить вид треугольника (равносторонний, равнобедренный, прямоугольный и т. п.)
12	Опишите класс СЧЕТ. Для банковского счета задаются фамилия владельца, номер счета, процент начисления за год и сумма в рублях. Включите в описание класса методы: пополнения счета, снятия денег со счета, перевода суммы в доллары и в евро, а также свойство, позволяющее начислить процент за заданное количество месяцев
13	Опишите класс ДАТА, заданный тремя атрибутами для года, месяца и дня. Включите в описание класса методы, позволяющие вычислить дату следующего дня, определить, сколько дней осталось до конца месяца, и свойство, позволяющее выяснить, является ли год высокосным
14	Опишите класс ЗАРПЛАТА. Для класса задаются атрибуты: фамилия, имя и отчество, год поступления на работу, оклад в рублях, процент надбавки, количество отработанных дней в месяце, количество рабочих дней в месяце, начисленная и удержанная суммы. Включите в описание класса методы: вычисления начисленной

	суммы, вычисления удержанной суммы, вычисления суммы, выдаваемой на руки, а также свойство только для чтения, позволяющее определить стаж работы (вычисляется как полное количество лет, прошедших с момента зачисления на работу до задаваемого текущего года). Начисленная сумма вычисляется за отработанные дни месяца плюс надбавка. Удержания – подоходный налог 13 %
15	Опишите класс ДАТА, заданный тремя атрибутами для года, месяца и дня. Включите в описание класса методы, позволяющие вычислить дату предыдущего дня, вычислить дату через заданное число дней, и свойство, позволяющее определить время года (зима, весна, лето, осень)
16	Опишите класс ВРЕМЯ, заданный тремя полями для часов, минут и секунд. Включите в описание класса методы, позволяющие перевести время в секунды, изменить время на заданное количество секунд, и свойство только для чтения, позволяющее определить время суток (утро, день, вечер, ночь)
17	Опишите класс ДРОБЬ, заданный двумя целыми числами для числителя и знаменателя несократимой дроби. Включите в описание класса методы умножения и деления дроби на заданное целое число и метод сокращения дроби, который обязательно вызывается при выполнении умножения и деления. Также предусмотрите свойство, позволяющее вывести десятичное представление дроби в виде вещественного числа
18	Опишите класс БАНКОМАТ. В классе задаются поля для хранения идентификационного номера банкомата, минимальной и максимальной сумм денег, которые позволено снять клиенту за один раз. Сумма денег, оставшаяся в банкомате, представляется шестью полями – номиналами российских рублей (10, 50, 100, 500, 1000, 5000), значениями которых является количество купюр данного достоинства. Включите в описание класса методы загрузки денег в банкомат и снятия определенной суммы денег, а также свойство, позволяющее вывести на экран сумму денег в банкомате в виде строки
19	Опишите класс ОКНО для работы с моделями экранных окон. В качестве полей задаются координаты левого верхнего угла и размеры окна по вертикали и по горизонтали (целые числа), заголовок окна, состояние (видимое/невидимое). Включите в описание класса методы передвижения окна по горизонтали и по вертикали, изменения высоты и ширины окна с проверкой на пересечение границ экрана и свойство, позволяющее установить, является ли данное окно квадратным
20	Опишите класс ТРЕУГОЛЬНИК, заданный координатами трех вершин. Включите в описание класса методы вычисления сторон треугольника, и свойство, позволяющее установить, принадлежит ли начало координат данному треугольнику

Литература

1. *Гуриков С.Р.* Введение в программирование на языке Visual C#: учеб. пособие. – М.: ФОРУМ : ИНФРА-М, 2013. – 448 с. – (Высшее образование. Бакалавриат).
2. *Доусон М.* Программируем на Python. – СПб.: Питер, 2016. – 416 с.
3. *Прохоренок Н.А.* Python 3 и PyQt. Разработка приложений. – СПб.: БХВ-Петербург, 2015. – 704 с.
4. *Шакин В.Н.* Базовые средства программирования на Visual Basic в среде Visual Studio.NET: учеб. пособие. – М.:ФОРУМ : ИНФРА-М, 2015. – 304 с.

Содержание

По вопросам приобретения книг обращайтесь:
Отдел продаж «ИНФРА-М» (оптовая продажа):
127282, Москва, ул. Полярная, д. 31В, стр. 1
Тел. (495) 280-15-96; факс (495) 280-36-29
E-mail: books@infra-m.ru

•

Отдел «Книга—почтой»:
тел. (495) 280-15-96 (доб. 246)

ФЗ № 436-ФЗ	Издание не подлежит маркировке в соответствии с п. 1 ч. 4 ст. 11

Учебное издание

Гуриков Сергей Ростиславович

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ НА PYTHON

УЧЕБНОЕ ПОСОБИЕ

ООО «Издательство Форум»
127282, Москва, ул. Полярная, д. 31В, стр. 1
E-mail: forum-book@yandex.ru
Тел.: (495) 280-15-96

ООО «Научно-издательский центр ИНФРА-М»
127282, Москва, ул. Полярная, д. 31В, стр. 1
Тел.: (495) 280-15-96, 280-33-86. Факс: (495) 280-36-29
E-mail: books@infra-m.ru http://www.infra-m.ru

Подписано в печать 08.11.2018.
Формат 70×100/16. Бумага офсетная. Гарнитура Times.
Печать цифровая. Усл. печ. л. 27,87.
ППТ30. Заказ № 00000

ТК 683004-970143-181116

Отпечатано в типографии ООО «Научно-издательский центр ИНФРА-М»
127282, Москва, ул. Полярная, д. 31В, стр. 1
Тел.: (495) 280-15-96, 280-33-86. Факс: (495) 280-36-29