Практикум по алгоритмизации и программированию на Python

Иван Хахаев, 2009

Часть 2. Основные алгоритмы и их реализация на Python.

При разборе задач в этой части будем обращать внимание на постановку задачи (что именно нужно сделать) и собственно алгоритм, который будет описываться как блок-схемой, так и на «псевдоязыке» программирования (подобие «школьного алгоритмического языка»). И только после этого можно приступать к написанию программы на Python с учётом всех тех его особенностей и возможностей, которые были описаны в предыдущей части.

Линейные алгоритмы. Операции с числами и строками.

Линейный алгоритм — алгоритм, в котором вычисления выполняются строго последовательно. Типичная блок-схема линейного алгоритма показана на рис. 1.

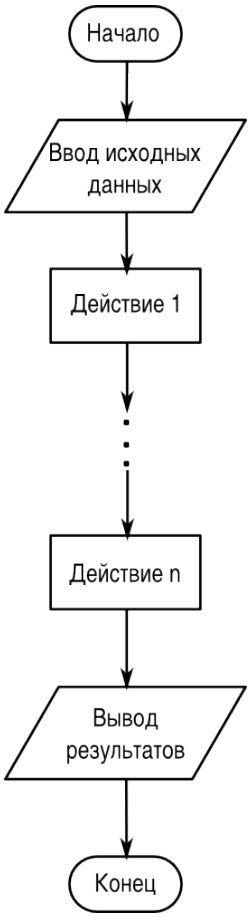

Рисунок 1. Типичная схема линейного алгоритма

Далее рассмотрим типичные задачи с линейной структурой алгоритма.

Задача 1. Дано два числа `a` и `b`. Сделать так, чтобы их значения поменялись местами.

Постановка задачи: Имеются две переменные с какими-то определёнными значениями. Пусть значение `a` равно `x`, а значение `b` равно `y`. Требуется, чтобы значение `a` стало равно `y`, а значение `b` стало равно `x`.

Метод решения (общий): Использовать дополнительную переменную `c`, в которую временно записать начальное значение переменной `a`, присвоить переменной `a` значение переменной `b`, а потом переменной `b` присвоить значение переменной `c`.

Блок-схема такого алгоритма показана на рис. 2.

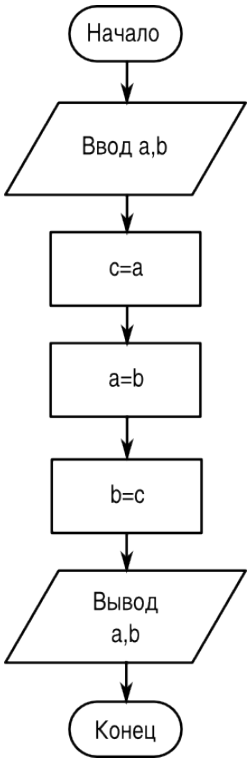

Рисунок 2. Блок-схема алгоритма обмена значениями

Текст программы на «псевдоязыке»:

```
ввод a,b
c=a
a=b
b=c
вывод a,b
```

Метод решения с использованием особенностей Python: использовать два кортежа. В первом будут определены переменные `a` и `b` и их значения, а второй сформируем из этих же переменных, но в обратном порядке.

Текст программы на Python:

```
# -*- coding: utf-8 -*-
#Перестановка местами двух чисел с использованием кортежа
#
(a,b)=input('Введите исходные значения (a, b) через запятую: ')
(a,b) = (b,a)
print 'Новое значение a: ',a,'\n','Новое значение b: ',b
```

Как описано в главе «Чтение из файла и запись в файл» Части 1, комбинация `'\n'` означает директиву на перевод строки для команды `print`.

Задача 2. Известны оклад (зарплата) и ставка процента подоходного налога. Определить размер подоходного налога и сумму, получаемую на руки.

Постановка задачи: Исходными данными являются величина оклада (переменная `oklad`, выражаемая числом) и ставка подоходного налога (переменная `procent`, выражаемая числом). Размер налога (переменная `nalog`) определяется как `oklad*procent/100`, а сумма, получаемая на руки (переменная `summa`) — как `oklad-nalog`.

Блок-схема алгоритма показана на рис. 3.

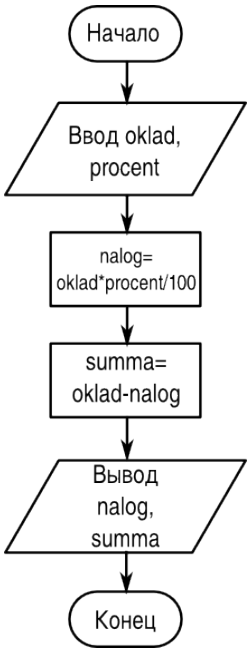

Рисунок 3. Блок-схема задачи о налоге

Текст программы на «псевдоязыке»:

```
ввод oklad,procent
nalog=oklad* procent/100
summa=oklad- nalog
вывод summa,nalog
```

Программа на Python:

```
#  -*- coding: utf-8 -*-
#
oklad=input("Оклад: ")
procent=input("% налога: ")
nalog=oklad*procent/100.0
summa=oklad-nalog
print "Сумма на руки: ",summa
print "Налог: ",nalog
```

Если здесь все числа использовать как целые, то результат может получиться неверным. Поэтому одно из чисел нужно ввести как вещественное (с десятичной точкой). Именно поэтому деление делается не на `100`, а на `100.0`.

Задача 3. Используя данные таблицы

Блюдо	Цена
Борщ	35
Котлета	40
Каша	20
Чай	3

определить общую стоимость обеда в столовой. Определить, во сколько раз возрастёт стоимость обеда, если цена котлеты увеличится вдвое (источник В.А.Молодцов, Н.Б.Рыжикова. Информатика: тесты, задания, лучшие методики. Ростов н/Д: Феникс, 2009).

Постановка задачи (формализованная): Имеется четыре числа, которые требуется просуммировать (обозначим их переменными `a`, `b`, `c` и `d` соответственно). Сумму их значений обозначим `S1`. Требуется найти также величину `S2=S1+b` и определить отношение `S2/S1` (обозначим это отношение переменной `res`). В результате нужно вывести значения переменных `S1` и `res`.

Блок-схема показана на рис. 4

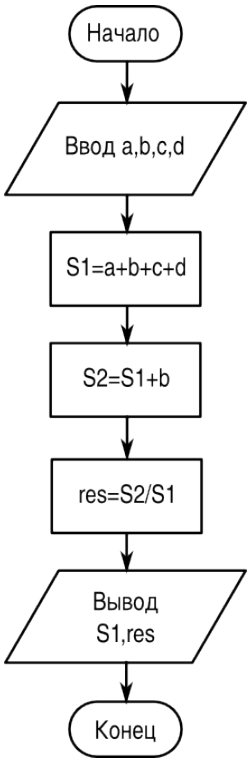

Рисунок 4. Блок-схема задачи об обеде

Текст программы на «псевдоязыке»:

```
ввод a,b,c,d
S1=a,b,c,d
S2=S1+b
res=S2/S1
вывод S1,res
```

В программе на Python разумно будет использовать кортеж :

```
t=(a,b,c,d)=input('Введите значения через запятую: ')
S1=sum(t)
S2=S1+b
res=S2/S1
print 'Начальная стоимость:',S1,'\n','Увеличение, раз:',res
```

И снова при вводе чисел хотя бы одно из них нужно ввести как вещественное.

Задача 4. Преобразовать дату в «компьютерном» представлении (системную дату) в «российский» формат, т.е. день/месяц/год (например, 17/05/2009).

Постановка задачи: Системная дата имеет вид 2009-06-15. Нужно преобразовать это значение в строку, строку разделить на компоненты (символ-разделитель — дефис), потом из этих компонентов сконструировать нужную строку.

Сразу перейдём к программе на Python. Функциями работы с датами и временем в Python «заведует» библиотека (модуль) `datetime`. а конкретно для работы с датами используется объект `date` и его методы.

Воспользуемся знанием методов строк и списков.

```
#  -*- coding: utf-8 -*-
#
# Подключаем нужный программный модуль
from datetime import date
# Получаем текущую дату
d1=date.today()
# Преобразуем результат в строку
ds=str(d1)
print "Системная дата ",ds
# Используем методы строки и списка
lst=ds.split('-')
lst.reverse()
#
rusdate="/".join(lst)
print "Российский стандарт ",rusdate
```

Комментарии в тексте программы помогают понять происходящее.

Задачи для самостоятельного решения.

1. Нарисуйте блок-схему к задаче 4 этой главы.
2. Даны действительные числа A,B,C. Найти максимальное и минимальное из этих чисел.
3. Известны длины трёх сторон треугольника. Вычислить периметр треугольника и площадь по формуле Герона (указание: использовать библиотеку `math` и функцию `sqrt()`).
4. Задан вес в граммах. Определить вес в тоннах и килограммах.
5. Известен объем информации в байтах. Перевести в Кбайты, Мбайты.
6. Определить значение функции `Z=1/(XY)` при `X` и `Y` не равных `0`.

Ветвления и оператор выбора.

В решениях задач по алгоритмизации одним из важнейших элементов является так называемое «ветвление», которое хорошо описывается в народных

сказках - «Направо пойдёшь — голову потеряешь, прямо пойдёшь — коня потеряешь...», а проще говоря, ситуация «если ..., то ..., иначе ...». Типовая блок-схема алгоритма с ветвлением (проверкой условия) показана на рис. 5.

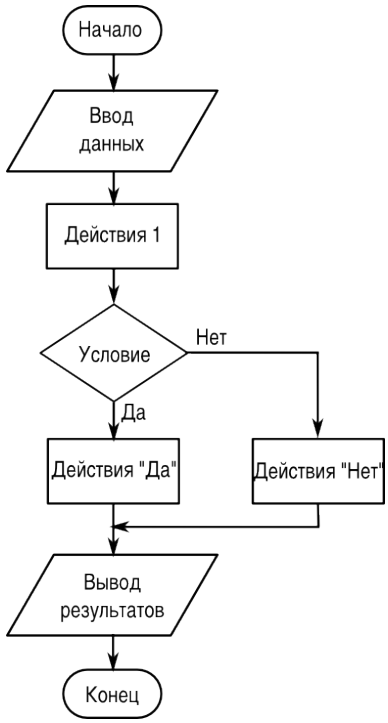

Рисунок 5. Типовая схема алгоритма с ветвлением

Если условие, указанное в блоке «Условие», выполняется, то далее производятся действия, соответствующие «ветви ДА» («Действия ДА»), иначе выполняются действия, соответствующие «ветви НЕТ» (Действия НЕТ»). Условия нужно составлять так, чтобы результат проверки любого условия допускал только два исхода — условие либо выполняется. либо не выполняется.

В случае, когда одной проверкой не удаётся охватить все варианты, используется «цепочка» условий, показанная на рис. 6. Такая ситуация называется «выбор».

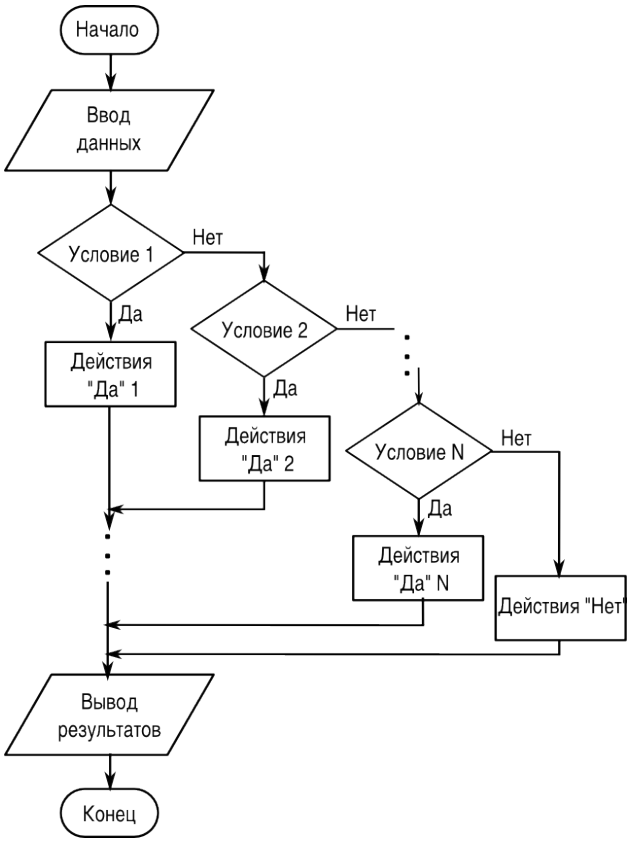

Рисунок 6. Блок-схема алгоритма выбора

В языках программирования для обеспечения проверки условий используется специальный составной оператор `IF` («если»). В этом операторе указывается условие, которое нужно проверить и действия, для ветвей «ДА» и «НЕТ».

Чтобы понять, как работает оператор `IF`, рассмотрим типичные задачи на проверку условий и выбор.

Задача 1. Составить программу ввода значения температуры воздуха `t` и выдачи текста «Хорошая погода!», если `t>10` градусов и текста «Плохая погода!», если `t<=10` градусов (источник В.А.Молодцов, Н.Б.Рыжикова. Информатика: тесты, задания, лучшие методики. Ростов н/Д: Феникс, 2009).

Постановка задачи: Исходными данными является значение t, необходимо сформировать строку `s`. При `t<10 s='Плохая погода!'`, иначе `s='Хорошая погода!'`.

Блок-схема алгоритма показана на рис. 7.

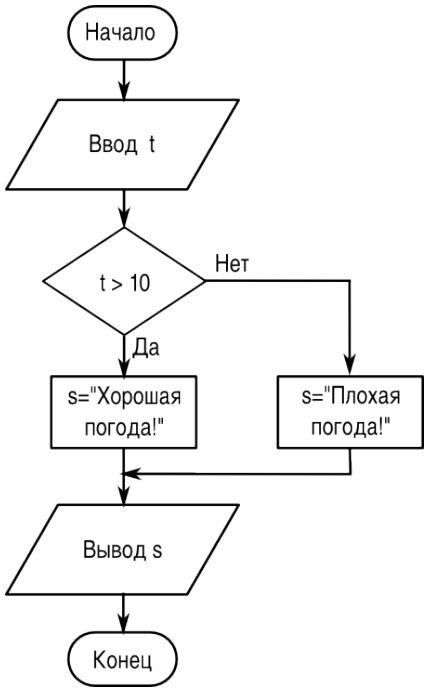

Рисунок 7. Блок-схема алгоритма задачи про погоду

Текст программы на «псевдоязыке»:

```
ввод t
если (t<10) то
    s='Плохая погода!'
иначе
    s='Хорошая погода!'
конец если
вывод s
```

Текст на Python:

```
#  -*- coding: utf-8 -*-
#
t=input('Введите температуру в градусах: ')
if t<10:
    s='Плохая погода!'
else:
    s='Хорошая погода!'
print s
```

Начало каждой «ветви» программы обозначается символом «:». Условие в операторе `IF` («если») записывается без скобок. Как таковое окончание оператора `IF` отсутствует. Python считает, что следующий оператор начинается в строке без отступа. Таким образом, в Python отступы играют важную роль,.

Задача 2 (источник тот же). Составить программу ввода оценки `P`, полученной учащимся, и выдачи текста «Молодец!», если `P=5`, «Хорошо!», если `P=4` и «Лентяй!», если `P<=3`.

Постановка задачи: Дано значение `P`, которое является натуральным числом и не может быть больше `5`. В зависимости от величины `P` нужно сформировать строку `s` по правилам, указанным в условии. Необходимо выполнить две последовательные проверки значения `P`.

Блок-схема алгоритма показана на рис. 8.

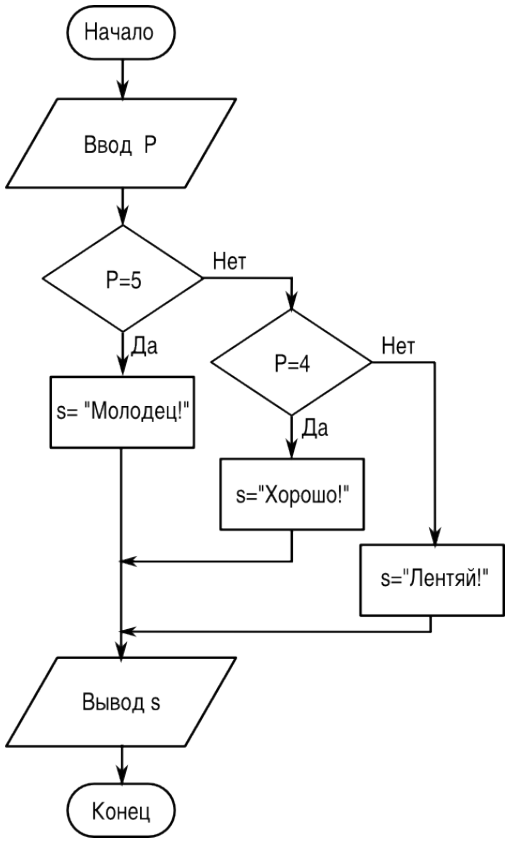

Рисунок 8. Блок-схема алгоритма к задаче про оценки

Текст программы на «псевдоязыке»:

```
ввод P
если (P=5) то
    s='Молодец!'
иначе если (P=4)
    s='Хорошо!'
иначе
    s='Лентяй!'
```

```
конец если
вывод s
```

Программа на Python:

```
#  -*- coding: utf-8 -*-
#
P=input('Ваши баллы? ')
if P==5:
    s='Молодец!'
elif P==4:
    s='Хорошо!
else:
   s='Лентяй!'
print s
```

Ключевое слово `elif` в Python является сокращением от `else if` («иначе если») и может использоваться любое количество раз, создавая различные варианты выбора.

Задачи для самостоятельного решения.

1. Дано натуральное число. Определить будет ли это число: чётным, кратным 4.

2. Дано натуральное число. Определить будет ли это число: нечётным, кратным 5.

3. Дано натуральное число. Определить будет ли это число: нечётным, кратным 7.

4. Дано натуральное число. Определить будет ли это число: чётным, кратным 10.

5. Имеется коробка со сторонами: `AxBxC`. Определить пройдёт ли она в дверь с размерами `MxK`.

6. Дано вещественное число. Определить какое это число: положительное, отрицательное, ноль.

7. Можно ли из бревна, имеющего диаметр поперечного сечения `D`, выпилить квадратный брус шириной `A`?

8. Можно ли в квадратном зале площадью S поместить круглую сцену радиусом `R` так, чтобы от стены до сцены был проход не менее `K`?

9. Дан номер места в плацкартном вагоне. Определить, какое это место: верхнее или нижнее, в купе или боковое.

10. Известна денежная сумма. Разменять её купюрами 500, 100, 10 и монетой

2 руб., если это возможно.

11. Имеются две ёмкости: кубическая с ребром A, цилиндрическая с высотой H и радиусом основания R. Определить поместится ли жидкость объёма M в первую ёмкость, во вторую, в обе.

12. Имеются две ёмкости: кубическая с ребром A, цилиндрическая с высотой H и радиусом основания R. Определить можно ли заполнить жидкостью объёма M первую ёмкость, вторую, обе.

13. Даны действительные числа: X, Y, Z. Определить существует ли треугольник с такими длинами сторон и, если существует, будет ли он прямоугольным.

14. Дано число X. Определить принадлежит ли это число заданному промежутку [a,b].

15. Определить значение функции Z=1/(XY) при произвольных X и Y.

16. Даны действительные числа: A, B, C. Определить выполняются ли неравенства A<B<C или A>=B>=C и какое именно неравенство выполняется.

17. Даны действительные числа X и Y. Вычислить Z. $Z=\sqrt{X*Y}$ при X>Y, $Z=ln(X+Y)$, в противном случае.

18. Даны действительные положительные числа a, b, c, d. Выясните, может ли прямоугольник со сторонами a,b уместиться внутри прямоугольника со сторонами c,d так, чтобы каждая сторона внутреннего прямоугольника была параллельна или перпендикулярна стороне внешнего прямоугольника.

19. Дано действительное число A. Вычислить f(A), если $f(x)=x^2+4x+5$, при $x\le 2$; в противном случае $f(x)=1/(x^2+4x+5)$.

20. Дано действительное число A. Вычислить f(A), если $f(x)=0$, при $x\le 0$; $f(x)=x$, при $0<x\le 1$, в противном случае $f(x)=x^4$.

21. Дано действительное число A. Вычислить f(A), если $f(x)=0$ при $x\le 0$; $f(x)=x^2-x$ при $0<x\le 1$, в противном случае $f(x)=x^2-\sin(\pi*x^2)$.

22. Составит алгоритм и программу для реализации логических операций "И" и "ИЛИ" для двух переменных.

23. Известен ГОД. Определить будет ли этот год високосным, и к какому веку этот год относится.

Указание. При вычислении корней и логарифмов используйте функции sqrt() и log() библиотеки math. В этой же библиотеке определена константа pi (π).

Циклические алгоритмы. Обработка последовательностей и одномерных массивов.

Циклом называется фрагмент алгоритма или программы, который может повторяться несколько раз (в том числе и нуль раз). Каждая циклическая конструкция начинается заголовком цикла и заканчивается конечным оператором. Между ними располагаются операторы, называемые «телом цикла». Количество повторений выполнения команд (операторов), составляющих тело цикла, определяется условием окончания цикла. Условием окончания может быть достижение некоторого значения специальной переменной, называемой параметром цикла (переменной цикла), или выполнение (прекращение выполнения) некоторого условия.

Для организации циклов с параметром в языках программирования используется составной оператор `FOR` («для»), а в циклах с условием чаще всего используется составной оператор `WHILE` («пока»).

В случае цикла с параметром количество повторений («оборотов») цикла известно заранее и задаётся специальным выражением в заголовке цикла, а в случае цикла с условием при каждом следующем повторении требуется проверять условие прекращения цикла. *Если при написании операторов в теле цикла допущена ошибка, условие прекращения цикла может не выполниться никогда и цикл окажется бесконечным* («программа зациклится»).

Пример блок-схемы цикла с параметром (переменной) показана на рис. 9, а пример блок-схемы цикла с условием окончания — на рис. 10. (Для обозначения заголовка цикла с параметром используется специальный графический элемент — блок модификации, в котором указывается закон изменения параметра цикла).

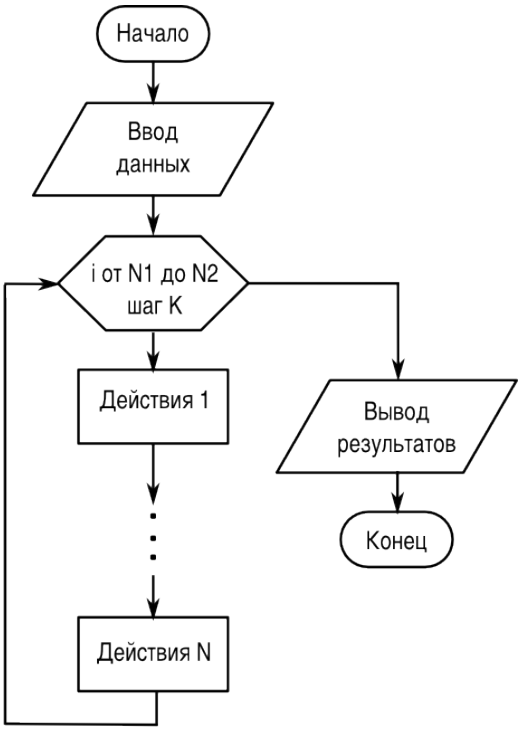

Рисунок 9. Пример блок-схемы цикла с параметром

Для работы с одномерными массивами целесообразно использовать циклы с параметром, поскольку до начала цикла может определено количество повторений. В этом случае цикл с параметром требуется для ввода элементов массива, а для выполнения каких-то действий с этими элементами и вывода результатов также могут потребоваться циклы.

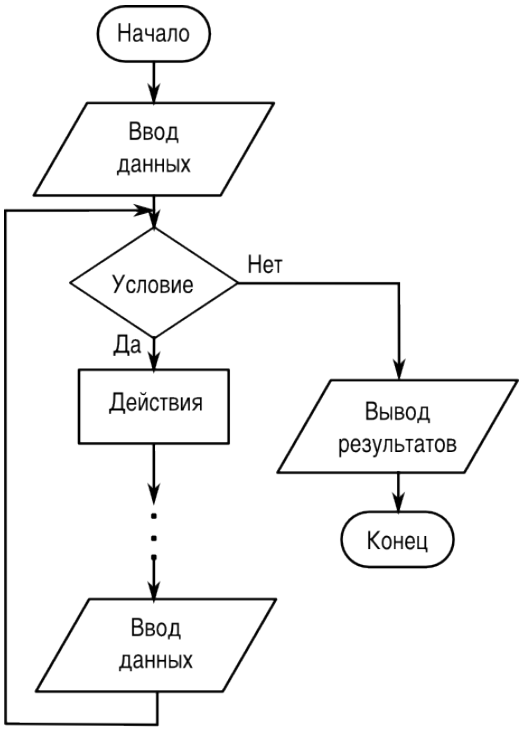

Рисунок 10. Пример блок-схемы цикла с условием

В блок-схеме на рис. 10 действия повторяются, пока выполняется некоторое условие. Когда условие перестаёт выполняться, цикл завершается.

Такие циклы целесообразно использовать в ситуации, когда данные вводятся (поступают из какого-то источника), пока не произойдёт некоторое событие. При этом всю обработку чаще всего приходится выполнять «на лету», не создавая массив, поскольку количество элементов заранее неизвестно.

Рассмотрим типичные задачи. решение которых требует циклических вычислений.

Задача 1. Дан одномерный массив A числовых значений, насчитывающий N элементов. Найти среднее арифметическое элементов массива.

Постановка задачи:

Дано:

`N` – количество элементов в массиве;

`i` - индекс элемента массива (параметр цикла).

`A[i]` – элемент массива;

Найти:

`S`– сумма элементов массива

`C` – среднее арифметическое элементов массива, `C=S/N`.

Блок-схема алгоритма показана на рис. 11.

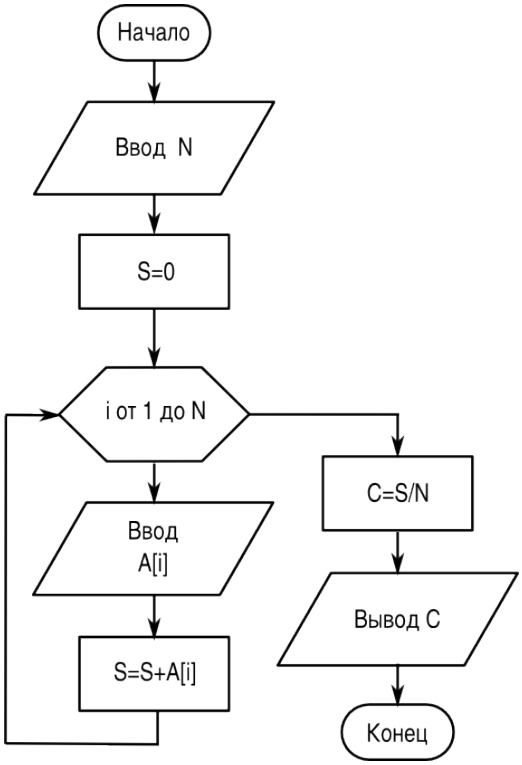

Рисунок 11. Блок-схема алгоритма вычисления среднего значения в массиве

Текст программы на «псевдоязыке»:

```
ввод N
S=0
нц для i от 1 до N
    ввод A[i]
    S=S+A[i]
кц
C=S/N
вывод C
```

Здесь **нц** и **кц** обозначают, соответственно, начало и конец цикла, строка с **нц** является заголовком цикла. Как видно из текста, указываются начальное и конечное значение переменной цикла, которая обязательно должна быть целым числом. В приведённой здесь записи переменная цикла увеличивается на 1 при каждом повторении («шаг переменной цикла» равен 1). Если требуется шаг не равный 1, это указывается специально.

Тело цикла состоит из двух операторов — ввода очередного числа и прибавления этого числа к текущему значению суммы.

На Python можно написать практически то же самое (с учётом особенностей, связанных с использованием функции `range()`).

```
#  -*- coding: utf-8 -*-
#
N=input('Количество элементов: ')
S=0
for i in range(N-1):
    a=input('Введите число: ')
    S=S+a
C=S/N
print 'Результат:',C
```

Поскольку диапазон чисел, формируемых функцией `range()`, начинается с 0, то верхней границей должно быть `N-1`. Поскольку массив хранить не надо, можно просто вводить числа и добавлять их к текущему значению суммы.

Тело цикла начинается после символа «:» и все операторы тела цикла в Python должны иметь одинаковый отступ от начала строки. Как только отступ исчезает, Python считает, что тело цикла закончилось.

А вот вариант решения этой же задачи на Python с использованием списка и методов списка.

```
#  -*- coding: utf-8 -*-
#
N=input('Количество элементов: ')
S=0
lst=[]
for i in range(N-1):
    a=input('Введите число: ')
    lst.append(a)
C=sum(lst)/N
print 'Результат:',C
```

В этом варианте формируется список, а сумма элементов списка вычисляется с помощью встроенной функции. Программа увеличилась на одну строку (создание пустого списка), но зато мы научились формировать список в цикле.

Задача 2. Определить, является ли введённая строка палиндромом («перевёртышем») типа ABBA, kazak и пр.

Постановка задачи: Требуется сравнивать попарно символы с начала и с конца строки `S` (первый и последний, второй и предпоследний и т.д.). Если в каждой такой паре символы одинаковы, строка является палиндромом. Соответственно, каждая проверка пары символов должна получить некоторый

признак (`flag` - «флаг»), который будет равен 1, если символы в паре совпадают и 0, если не совпадают. Окончательный результат обработки строки получится как произведение всех значений «флагов». Если хотя бы один раз «флаг» оказался равен нулю, строка палиндромом не является и произведение всех «флагов» окажется равным 0. Количество пар не превышает половины длины строки `L` (точно равно половине длины для строк с чётным количеством символов и результат целочисленного деления длины строки на 2 для строк с нечётным количеством символов, поскольку «центральный» символ строки с нечётным количеством символов очевидно совпадает сам с собой).

Блок-схема алгоритма показана на рис. 12.

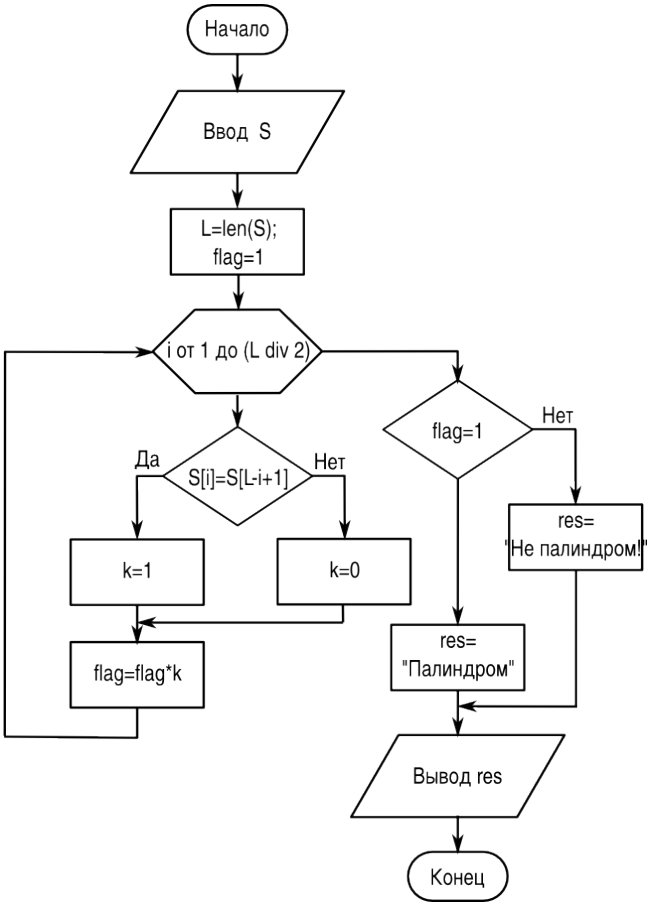

Рисунок 12. Блок-схема алгоритма определения палиндрома

Текст программы на «псевдоязыке»:

```
ввод S
flag=1
```

```
L=длина(S)
N=L div 2
нц для i от 1 до N
    если S[i]=S[L-i+1] то
        k=1
    иначе
        k=0
    конец если
    flag=flag*k
кц
если flag=1 то
    вывод 'Палиндром'
иначе
    вывод 'Не палиндром!'
конец если
```

При проверке каждой пары устанавливается коэффициент `k`, который затем умножается на текущее значение «флага».

Окончательный вывод делается по итоговому значению «флага».

Текст программы на Python может быть очень похожим.

```
#  -*- coding: utf-8 -*-
#
s1=raw_input('Исходная строка: ')
# Определяем длину строки
L=len(s1)
flag=1
for i in range(L//2):
    if s1[i]==s1[-i-1]:
        k=1
    else:
        k=0
    flag=flag*k
if flag==1:
    print 'Палиндром'
else:
    print 'Не палиндром!'
```

Для ввода строки использован оператор `raw_input()`, при этом не требуется строку записывать в кавычках.

Небольшие синтаксические особенности всё-таки есть — условие равенства двух переменных записывает знаком «==», начало каждого составного оператора обозначается символом «:», и опять большую роль играют отступы. Кроме того, чтобы отсчитывать символы с конца строки, использованы «отрицательные» индексы элементов строки.

Однако использование особенностей строк в Python, их функций и методов, позволяет решить эту задачу более изящно. Например, так.

```
#  -*- coding: utf-8 -*-
#
s1=raw_input('Исходная строка: ')
lst=list(s1)
lst.reverse()
s2=''.join(lst)
if s1==s2:
    print 'Палиндром'
else:
    print 'Не палиндром!'
```

Здесь исходная строка преобразуется в список, затем список «переворачивается» и из него с помощью пустой «строки-объединителя» формируется новая строка. Затем строки сравниваются. Цикл оказывается не нужен! Всю работу делает Python.

Если количество повторений операций заранее неизвестно, но известно условие прекращения выполнения операций, используется цикл (составной оператор) `WHILE`. Покажем его использование на следующем примере.

Задача 3. Последовательно вводятся ненулевые числа. Определить сумму положительных и сумму отрицательных чисел. Закончить ввод чисел при вводе 0.

Задача настолько проста, что дополнительных уточнений в качестве постановки задачи не требуется. Пусть сумма положительных чисел называется `SP`, а сумма отрицательных чисел — `SN`.

Блок-схема алгоритма показана на рис. 13.

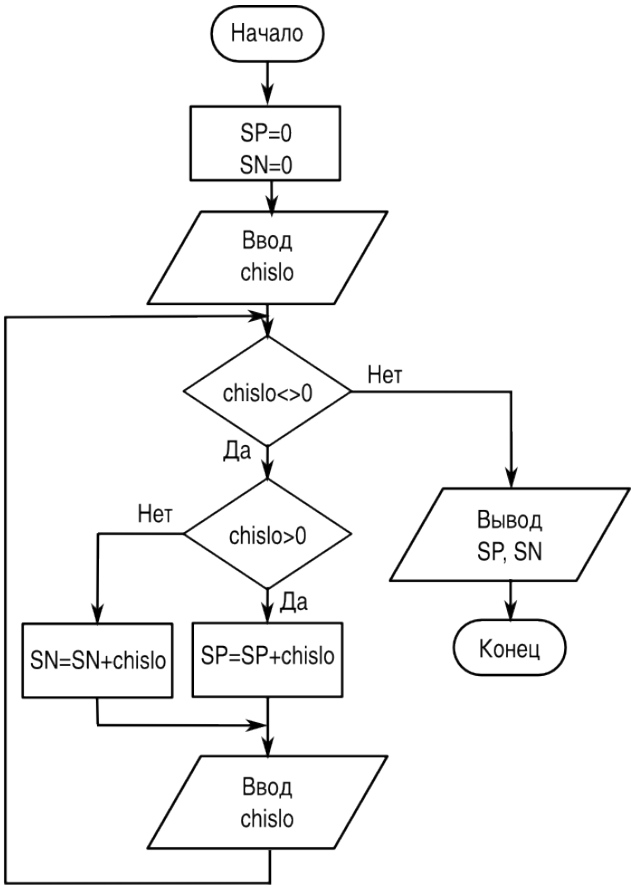

Рисунок 13. Блок-схема алгоритма обработки последовательности

Текст программы на «псевдоязыке»:

```
SP=0
SN=0
ввод chislo
нц пока chislo <> 0
    если chislo >0 то
        SP=SP+chislo
    иначе
        SN=SN+chislo
    конец если
ввод chislo
кц
вывод SP
вывод SN
```

Условие «неравенства» в языках программирования Pascal и BASIC обозначается как «<>», поэтому здесь сохранено это обозначение.

Нужно обратить внимание что проверяемое число нужно определить до начала цикла, поскольку возможна ситуация, что неопределённое значение

окажется равным 0 и программа закончится, не успев начаться. А потом числа вводятся в цикле и каждое вновь поступившее число проверяется на не-равенство 0 (после ввода каждого числа следует проверка условия). Порядок операций и проверок в цикле `WHILE` может оказаться важным для получения верного результата.

Текст программы на Python не имеет каких-то существенных особенностей. Для удобства чтения программа поделена на «блоки» с помощью символа комментария.

```
#  -*- coding: utf-8 -*-
#
SP=0
SN=0
#
chislo=input('Следующее число: ')
#
while chislo != 0:
     if chislo > 0:
        SP=SP+chislo
     else:
        SN=SN+chislo
     chislo=input('Следующее число: ')
#
print 'Сумма положительных:',SP
print 'Сумма отрицательных:',SN
```

Сортировка массива.

Задача сортировки, а также задача поиска максимального или минимального элемента в массиве встречается довольно часто. Средствами Python такие задачи решаются очень просто, но тем не менее рассмотрим общую задачу сортировки массива.

Под сортировкой понимается процедура, в результате выполнения которой изменяется исходный порядок следования данных. Причём новый порядок их следования отвечает требованию возрастания или убывания значений элементов одномерного массива. Например, при сортировке по возрастанию из одномерного массива [3 1 0 5 2 7] получается массив [0 1 2 3 5 7]. Возможны и более сложные критерии сортировки. Символьные данные обычно сортируются в алфавитном порядке.

Один из наиболее наглядных методов сортировки – «метод пузырька».

Пусть необходимо упорядочить элементы массива `A` из `N` элементов по возрастанию.

Просматривая элементы массива «слева направо» (от первого элемента к последнему), меняем местами значения каждой пары соседних элементов в случае

неравенства `A[i]>A[i+1]`, передвигая тем самым наибольшее значение на последнее место. Следующие просмотры начинаем опять с первого элемента массива, последовательно уменьшая на единицу количество просматриваемых элементов. Процесс заканчивается после `N−1` просмотра.

Метод получил такое название, потому что каждое наибольшее значение как бы всплывает вверх.

Фрагмент блок-схемы алгоритма показан на рис. 14.

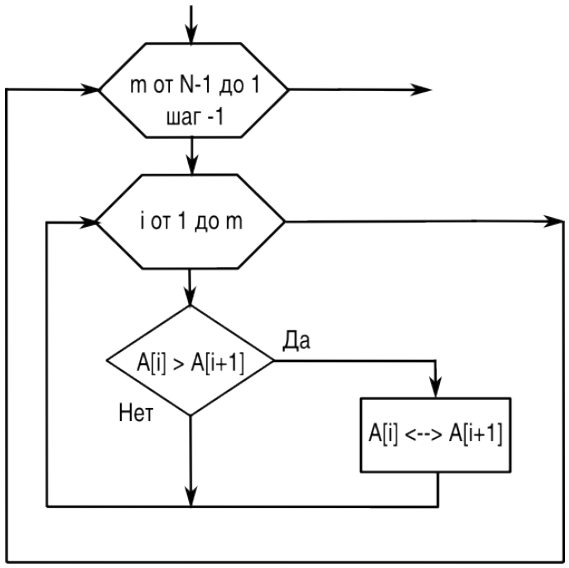

Рисунок 14. Алгоритм сортировки «методом пузырька»

Действие `A[i] <--> A[i+1]` означает перестановку значений элементов массива.

Текст соответствующего фрагмента программы на «псевдоязыке»:

```
ввод N, A
нц для m от N−1 до 1 шаг −1
    нц для i от 1 до m
        если A[ i ] > A[i+1] то
        X=A[i]
        A[i]=B[i+1]
        A[i+1]=X
        конец если
    кц
кц
вывод A
```

В этом фрагменте для перестановки значений элементов массива

используется промежуточная переменная.

Задача поиска максимального элемента в массиве решается следующим образом. Пусть `maxA` — требуемое значение максимального элемента. Сначала присваиваем переменной `maxA` значение первого элемента массива, потом сравниваем первый элемент со следующим. Если следующий элемент (второй) больше первого, присваиваем его значение переменной `maxA`, а если нет — переходим к следующему (третьему элементу) и т.д.

Аналогично решается задача поиска минимального элемента в массиве.

В Python эти алгоритмы уже реализованы в функциях `max()`, `min()` и в методе `sort()` (метод `sort()` сортирует список по возрастанию значений элементов).

Задачи для самостоятельного решения.

1. Составьте блок-схему поиска максимального элемента в одномерном массиве.

2. Нарисуйте полную блок-схему алгоритма сортировки массива «методом пузырька».

3. Дан одномерный массив числовых значений, насчитывающий N элементов. Поменять местами элементы, стоящие на чётных и нечётных местах: A(1) с A(2), A (3) с A(4) ..

4. Дан одномерный массив числовых значений, насчитывающий N элементов. Выполнить перемещение элементов массива по кругу вправо, т.е. A(1) -> A(2); A(2) -> A(3); ... A(n) -> A(1).

5. Дан одномерный массив числовых значений, насчитывающий N элементов. Поменять местами первую и вторую половины массива.

6. Дан одномерный массив числовых значений, насчитывающий N элементов. Поменять местами группу из M элементов, начинающихся с № K с группой из M элементов, начинающихся № P.

7. Дан одномерный массив числовых значений, насчитывающий N элементов. Вставить группу из M новых элементов, начиная с № K.

8. Дан одномерный массив числовых значений, насчитывающий N элементов. Сумму элементов массива и количество положительных элементов поставить на первое и второе место.

9. Дан одномерный массив числовых значений, насчитывающий N элементов. Исключить из него M элементов, начиная с номера K.

10. Дан одномерный массив числовых значений, насчитывающий N элементов. Исключить все нулевые элементы.

11. Дан одномерный массив числовых значений, насчитывающий N элементов. После каждого отрицательного элемента вставить новый элемент, равный

квадрату этого отрицательного элемента.

12. Дан одномерный массив числовых значений, насчитывающий N элементов. Определить образуют ли элементы массива, расположенные перед первым отрицательным элементом, возрастающую последовательность.

13. Дан одномерный массив числовых значений, насчитывающий N элементов. Определить образуют ли элементы массива, расположенные перед первым отрицательным элементом, убывающую последовательность.

14. Дан одномерный массив числовых значений, насчитывающий N элементов. Из элементов исходного массива построить два новых. В первый должны входить только положительные элементы, а во второй только отрицательные элементы.

15. Дан одномерный массив числовых значений, насчитывающий N элементов. Добавить столько элементов, чтобы положительных и отрицательных стало бы поровну.

16. Дан одномерный массив числовых значений, насчитывающий N элементов. Добавить к элементам массива такой новый элемент, чтобы сумма положительных элементов стала бы равна модулю суммы отрицательных элементов.

17. Дан одномерный массив числовых значений, насчитывающий N элементов. Дано положительное число T. Разделить это число между положительными элементами массива пропорционально значениям этих элементов, и добавить полученные доли к соответствующим элементам.

18. Дан одномерный массив числовых значений, насчитывающий N элементов. Исключить из массива элементы, принадлежащие промежутку [B;C].

19. Дан одномерный массив числовых значений, насчитывающий N элементов. Вместо каждого нулевого элемента поставить сумму двух предыдущих элементов массива.

20. Дан одномерный массив числовых значений, насчитывающий N элементов. Определить имеются ли в массиве два подряд идущих нуля.

21. Дан одномерный массив числовых значений, насчитывающий N элементов. Подсчитать количество чисел, делящихся на 3 нацело, и среднее арифметическое чётных чисел. Поставить полученные значения на первое и последнее места в массиве (увеличив массив на 2 элемента).

22. Заданы M строк символов, которые вводятся с клавиатуры. Найти количество символов в самой длинной строке. Выровнять строки по самой длинной строке, поставив перед каждой строкой соответствующее количество звёздочек.

23. Заданы M строк символов, которые вводятся с клавиатуры. Из заданных строк, каждая из которых представляет одно слово, составить одну длинную строку, разделяя слова пробелами.

24. Заданы M строк слов, которые вводятся с клавиатуры. Подсчитать количество гласных букв в каждой из заданных строк.

25. Заданы M строк слов, которые вводятся с клавиатуры. Вводится слог (последовательность букв). Подсчитать количество слогов в каждой строке.

26. Заданы M строк слов, которые вводятся с клавиатуры. Вводится слог (последовательность букв). Удалить данный слог из каждой строки.

27. Заданы M строк символов, которые вводятся с клавиатуры. Напечатать все центральные буквы слов нечетной длинны.

28. Заданы M строк символов, которые вводятся с клавиатуры. Каждая строка содержит слово. Записать каждое слово в разрядку - через пробелы.

29. Задана строка символов, в которой встречается символ ".". Поставить после каждого такого символа системное время ПК.

30. Заданы M строк слов, которые вводятся с клавиатуры. Подсчитать количество пробелов в каждой из строк.

31. Заданы M строк символов, которые вводятся с клавиатуры. Каждая строка представляет собой последовательность символов, включающих в себя вопросительные знаки. Заменить в каждой строке все имеющиеся вопросительные знаки звёздочками.

32. Определить сумму чисел с нечётными номерами и произведение чисел с чётными номерами. Подсчитать количество слагаемых и количество сомножителей. При вводе числа 55555 закончить работу.

33. Определить сумму вводимых положительных чисел. Причём числа с нечётными номерами суммировать с обратным знаком, а числа с чётными номерами перед суммированием возводить в квадрат. Подсчитать количество слагаемых. При вводе первого отрицательного числа закончить работу.

34. Даны число P и число H. Определить сумму чисел меньше P, произведение чисел больше H и количество чисел в диапазоне значений P и H. При вводе числа равного P или H, закончить работу.

35. Суммировать вводимые числа, среди которых нет нулевых. При вводе нуля обеспечить вывод текущего значения суммы. При вводе числа 99999 закончить работу.

36. Вводятся положительные числа. Определить сумму чисел, делящихся на положительное число B нацело. При вводе отрицательного числа закончить работу.

37. Для вводимых чисел определить процент положительных и отрицательных чисел. При вводе числа –65432 закончить работу.

Обработка двумерных массивов (матриц).

Двумерные массивы являются аналогами матриц и имеют «прямоугольную» (табличную) структуру. Описываются массивы так же, как одномерные. Разница состоит в том, что у элемента двумерного массива *две* координаты (два индекса) — номер строки и номер столбца, в которых находится элемент.

Ввод массива осуществляется построчно при помощи двух циклов. Пусть `M` — количество столбцов, `N` — количество строк. Элементы массива обозначим как `mas[i,j]`, первый индекс — номер строки, второй — номер столбца.

```
ввод M,N
нц для i от 1 до N
    нц для j от 1 до M
        ввод mas[i,j]
    кц
кц
```

Вывод массива на экран осуществляется при помощи аналогичных циклов.

```
нц для i от 1 до N
    нц для j от 1 до M
        вывод mas[i,j]
    кц
   вывод
кц
```

Здесь «пустой» оператор вывода обеспечивает переход на новую строку.

В Python для работы с многомерными (когда используется два и более индексов) массивами можно использовать вложенные списки (списки списков, списки списков списков и т.д.).

Однако Python предоставляет более удобный инструмент создания и преобразования многомерных массивов — библиотеку `numpy` (Numeric Python).

Создание двумерного массива в Python может выглядеть так:

```
#  -*- coding: utf-8 -*-
#
import numpy
n=input('Количество строк: ')
m=input('Количество столбцов: ')
# Создаём ''нулевую'' матрицу
a=numpy.zeros([n-1,m-1])
# Заполняем матрицу
for i in range(n-1):
    for j in range(m-1):
        print 'Элемент матрицы [',i,'][',j,']'
        a[i,j]=input('Введите элемент: ')
```

#

Сначала с помощью функции (метода) `numpy.zeros()` создаётся двумерный массив (матрица), заполненный нулями, а потом вместо нулей подставляются реальные значения. Индексы элементов, так же как в строках, кортежах и списках, начинаются с 0 (первый — верхний левый — элемент матрицы в Python имеет индекс `[0,0]`).

Оператор `print` выводит индексы очередного элемента матрицы, который нужно ввести.

Задача 1. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Найти среднее арифметическое элементов массива.

Постановка задачи:

Дано:

`n` – количество строк в массиве;

`m` – количество столбцов в массиве;

`A(i,j)` – элемент массива;

`i , j`- индексы элемента массива.

Найти:

`S`– сумма элементов массива (сумма всех `A(i,j)` при всех `i` и `j`)

`K` – количество элементов в массиве (`K=m*n`)

`C` – среднее арифметическое элементов массива (`C=S/K`)

Блок-схема алгоритма решения показана на рис. 15.

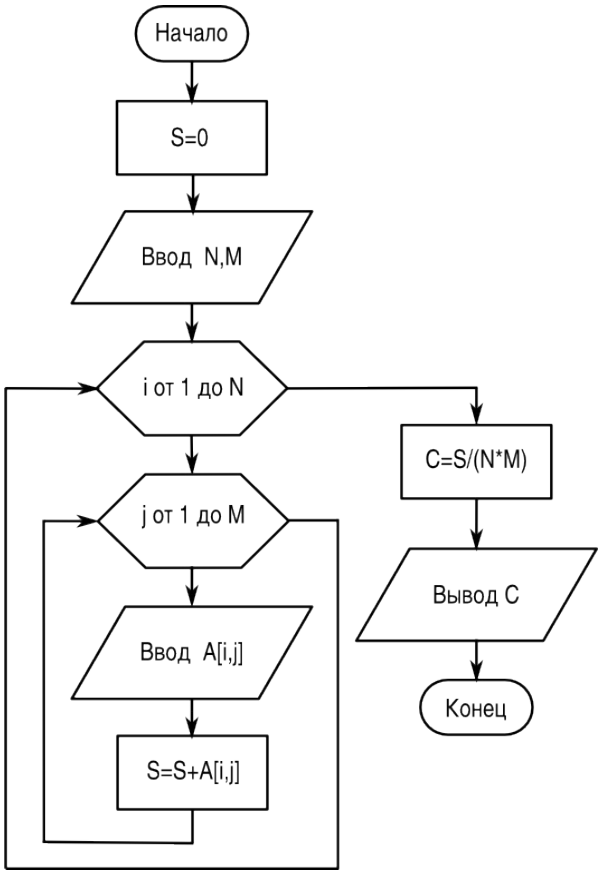

Рисунок 15. Блок-схема алгоритма вычисления среднего значения матрицы

Текст программы на «псевдоязыке»:

```
ввод n,m
S=0
нц для i от 1 до n
    нц для j от 1 до m
        ввод A[i,j]
        S=S+A[i,j]
    кц
кц
K=n*m
C=S/K
вывод C
```

Текст программы на Python:

```
#  -*- coding: utf-8 -*-
#
import numpy
```

```
n=input('Количество строк: ')
m=input('Количество столбцов: ')
S=0.0
# Создаём нулевую матрицу
a=numpy.zeros([n-1,m-1])
# Заполняем матрицу
for i in range(n-1):
    for j in range(m-1):
        print 'Элемент матрицы [',i,'][',j,']'
        a[i,j]=input('Введите элемент: ')
        S=S+a[i,j]
#
K=n*m
C=S/K
print 'Среднее значение по строкам:',C
```

Задачи для самостоятельного решения.

1. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Найти наибольший элемент столбца матрицы A, для которого сумма абсолютных значений элементов максимальна.

2. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Найти наибольшее значение среди средних значений для каждой строки матрицы.

3. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Найти наименьший элемент столбца матрицы A, для которого сумма абсолютных значений элементов максимальна.

4. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Найти наименьшее значение среди средних значений для каждой строки матрицы.

5. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Определить средние значения по всем строкам и столбцам матрицы. Результат оформить в виде матрицы из N+1 строк и M+1 столбцов.

6. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Найти сумму элементов всей матрицы. Определить, какую долю в этой сумме составляет сумма элементов каждого столбца. Результат оформить в виде матрицы из N+1 строк и M столбцов.

7. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Найти сумму элементов всей матрицы. Определить, какую долю в этой сумме составляет сумма элементов каждой строки. Результат оформить в виде матрицы из N строк и M+1 столбцов.

8. Выполнить обработку элементов прямоугольной матрицы A, имеющей N

строк и M столбцов. Определить, сколько отрицательных элементов содержится в каждом столбце и в каждой строке матрицы. Результат оформить в виде матрицы из N+1 строк и M+1 столбцов.

9. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Определить, сколько нулевых элементов содержится в верхних L строках матрицы и в левых K столбцах матрицы.

10. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Перемножить элементы каждого столбца матрицы с соответствующими элементами K-го столбца.

11. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Просуммировать элементы каждой строки матрицы с соответствующими элементами L - той строки.

12. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Нормировать элементы каждой строки по отношению к наибольшему элементу этой строки.

13. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Нормировать элементы каждого столбца матрицы по отношению к наибольшему элементу этого столбца.

14. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Нормировать элементы матрицы по отношению к наибольшему элементу всей матрицы.

15. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Все элементы имеют целый тип. Дано целое число H. Определить, какие столбцы имеют хотя бы одно такое число, а какие не имеют.

16. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Исключить из матрицы строку с номером L. Сомкнуть строки матрицы.

17. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Добавить к матрице строку и вставить ее под номером L.

18. Выполнить обработку элементов квадратной матрицы A, имеющей N строк и N столбцов. Найти сумму элементов, стоящих на главной диагонали, и сумму элементов, стоящих на побочной диагонали.

19. Выполнить обработку элементов квадратной матрицы A, имеющей N строк и N столбцов. Определить сумму элементов, расположенных параллельно главной диагонали (ближайшие к главной).

20. Выполнить обработку элементов квадратной матрицы A, имеющей N строк и N столбцов. Определить произведение элементов, расположенных параллельно побочной диагонали (ближайшие к побочной).

21. Выполнить обработку элементов квадратной матрицы A, имеющей N строк

и N столбцов. Каждой паре элементов, симметричных относительно главной диагонали (ближайшие к главной), присвоить значения, равные полусумме этих симметричных значений.

22. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Исходная матрица состоит из нулей и единиц. Добавить к матрице еще один столбец, каждый элемент которого делает количество единиц в каждой строке четным.

23. Выполнить обработку элементов квадратной матрицы A, имеющей N строк и N столбцов. Найти сумму элементов, расположенных выше главной диагонали, и произведение элементов, расположенных выше побочной диагонали.

24. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Дан номер строки L и номер столбца K, при помощи которых исходная матрица разбивается на четыре части. Найти сумму элементов каждой части.

25. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Определить, сколько нулевых элементов содержится в каждом столбце и в каждой строке матрицы. Результат оформить в виде матрицы из N+1 строк и M+1 столбцов.

26. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Дан номер строки L и номер столбца K, при помощи которых исходная матрица разбивается на четыре части. Найти среднее арифметическое элементов каждой части.

27. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Все элементы имеют целый тип. Дано целое число H. Определить, какие строки имеют хотя бы одно такое число, а какие не имеют.

28. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Исключить из матрицы столбец с номером L. Сомкнуть столбцы матрицы.

29. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Добавить к матрице столбец и вставить ее под номером L.

30. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Добавить к элементам каждого столбца такой новый элемент, чтобы сумма положительных элементов стала бы равна модулю суммы отрицательных элементов. Результат оформить в виде матрицы из N+1 строк и M столбцов.

31. Выполнить обработку элементов прямоугольной матрицы A, имеющей N строк и M столбцов. Добавить к элементам каждой строки такой новый элемент, чтобы сумма положительных элементов стала бы равна модулю суммы отрицательных элементов. Результат оформить в виде матрицы из N строк и M+1 столбцов.

Работа с ассоциативными массивами (таблицами данных)

Ассоциативный массив лучше всего описывается табличным представлением данных, когда каждая строка таблицы описывает характеристики какого-то объекта из множества однородных объектов (типичный пример — список учеников, их домашних телефонов и адресов). Таким образом, по значению из первого столбца такой таблицы (ключу) можно однозначно определить значения из остальных столбцов, т.е. значение ключа ассоциируется с остальными характеристиками объекта (в случае ученика — по фамилии можно найти другую информацию).

Если в ассоциативном массиве только два столбца («ключ» и «значение»), то такой массив называется «хэш». Такие ассоциативные массивы очень часто используются в современных информационных системах (например, пары «логин-пароль»).

В области моделирования процессов и явлений часто встречаются задачи, в которых значению «ключа» соответствует несколько параметров (например, номеру химического элемента однозначно соответствует название, атомный вес, валентность, количество протонов и пр.). В таких задачах простые хэш-массивы использовать уже неудобно.

Эффективный алгоритм обработки ассоциативных массивов (поиска значений, добавления и удаления значений и ключей, сортировки и пр.) в значительной степени зависит от используемого языка программирования и определённых в этом языке типов и структур данных. Так, в языке программирования Basic ассоциативный массив образуется из нескольких согласованных одномерных массивов. В языке программирования Pascal для представления ассоциативных массивов используется структура данных «запись» (record). В Python для ассоциативных массивов определена специальная структура данных — словарь, но мы рассмотрим работу с ассоциативными массивами с помощью списков и функций работы со списками.

Рассмотрим задачу из области экономического анализа.

Оценить экономическую деятельность нескольких предприятий. Известны названия предприятий, значения планового объёма розничного товарооборота и значения фактического объёма розничного товарооборота.

Требуется определить:

1. процент выполнения плана каждым предприятием
2. количество предприятий, недовыполнивших план
3. наибольший плановый товарооборот

4. упорядочить предприятия по возрастанию планового товарооборота.

Обозначим количество предприятий как `k` и сформируем три списка — список названий предприятий (пусть он называется `name`), список значений планового товарооборота (назовём его `plan`), список значений фактического товарооборота (с именем `fact`). На основании этих данных создадим список значений процентов выполнения плана (пусть он называется `procent`).

Количество предприятий, недовыполнивших план, будем определять в результате сравнения процента выполнения со 100 процентами в цикле по всем предприятиям.

Текст программы на Python может выглядеть, как показано ниже.

```
#  -*- coding: utf-8 -*-
#
# k - количество предприятий
# name - список названий предприятий
# plan - список значений планового товарооборота
# fact - список значений фактического товарооборота
# procent - список значений % выполнения плана
#
k=input("Количество предприятий: ")
name=[]
plan=[]
fact=[]
#
for i in range(k):
    n=raw_input("Название: ")
    name.append(n)
    p1=input("План: ")
    plan.append(p1)
    p2=input("Факт: ")
    fact.append(p2)
#
procent=map(lambda x,y: x*100/y,fact,plan)

fakty=zip(name,procent)
plany=zip(plan,name)

plany.sort()

print 16*"="
print "Процент выполнения плана каждым предприятием:"
#
nedo=0
for i in range(k):
    s1=fakty[i][0]
    s2=fakty[i][1]
    if s2 < 100:
```

```
        nedo=nedo+1
    #
    print s1, ": ",s2

print "Количество предприятий, недовыполнивших план: ", nedo
print "Наибольший плановый товарооборот: ", max(plan)
#
print "Предприятия по возрастанию плана:"

for i in range(k):
    s1=plany[i][1]
    s2=plany[i][0]
    print s1, ": ",s2
```

Здесь с помощью функции `map()` и «одноразовой»-`lambda`-функции создаётся список процентов выполнения плана и с помощью функции `zip()` формируется два итоговых ассоциативных массива. Сортировка таких ассоциативных массивов производится «по первому столбику», поэтому важен порядок аргументов в функции `zip()`, а также порядок индексов при выводе результатов.

Пример решения задачи показан на рис. 16.

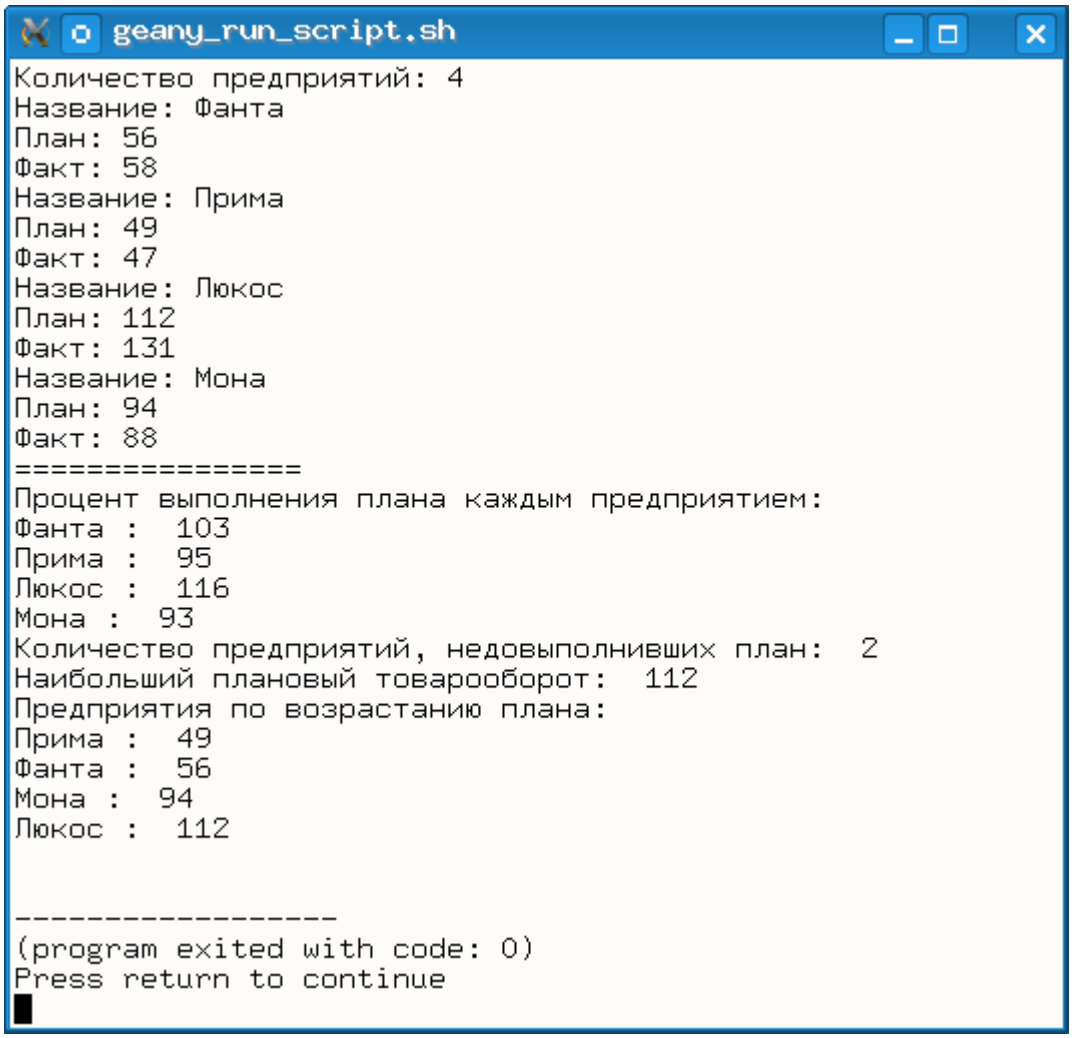

Рисунок 16. Пример решения задачи с ассоциативным массивом

Задачи для самостоятельного решения.

1. Используя данные таблицы

Блюдо	Цена
Борщ	35
Котлета	40
Каша	20
Чай	3

отсортировать блюда по возрастанию цены. Вывести отсортированный вариант списка блюд.

2. Имеется список учеников и результаты трёх тестов (баллы от 0 до 100). Определить средний балл каждого ученика по трём тестам, вывести список

учеников по убыванию среднего балла.

3. Известны данные о количестве мальчиков и девочек в нескольких классах. Отсортировать названия классов по возрастанию процента мальчиков, определить количество классов, к которых мальчиков больше, чем девочек и вывести названия этих классов отдельно.

4. Решить задачу, связанную с оценкой экономической деятельности группы предприятий на основе известных данных:

- Название предприятий
- Плановый объем розничного товарооборота
- Фактический объем розничного товарооборота

Требуется определить:

i) процент выполнения плана каждым предприятием

ii) количество предприятий, недовыполнивших план на 10% и более

iii) наименьший плановый товарооборот

iv) упорядочить предприятия по убыванию планового товарооборота.