

ИНФОРМАТИКА

9

класс

ОБРАБОТКА ОДНОМЕРНЫХ МАССИВОВ ЦЕЛЫХ ЧИСЕЛ НА ЯЗЫКЕ PYTHON

РАЗРАБОТКА АЛГОРИТМОВ И ПРОГРАММ

КЛЮЧЕВЫЕ СЛОВА

- ◆ массив
- ◆ размерность массива
- ◆ индекс
- ◆ одномерный массив
- ◆ сортировка
- ◆ сортировка пузырьком
- ◆ сортировка выбором
- ◆ двоичный поиск

ТАБЛИЧНЫЕ ВЕЛИЧИНЫ

Линейная таблица (одномерный массив) представляет собой набор однотипных данных, записанных в одну строку или один столбец. Элементы строки (столбца) всегда нумеруются.

1	Понедельник
2	Вторник
3	Среда
4	Четверг
5	Пятница
6	Суббота
7	Воскресенье

Васечкин

1	2	3	4	5
6	6	1	0	0



ТАБЛИЧНЫЕ ВЕЛИЧИНЫ

Прямоугольная таблица (двумерный массив) — это упорядоченный некоторым образом набор строк (столбцов), содержащих одинаковое количество элементов. Строки прямоугольных таблиц имеют свою нумерацию, столбцы — свою.

	1	2	3	4	5
1. Васечкин	6	6	1	0	0
2. Ионов	0	0	0	0	0
3. Радугина	0	0	1	0	0
...
19. Чабанюк	0	0	0	0	0



МАССИВ (PYTHON 3: СПИСОК = МАССИВ)

Массив - это набор элементов одного типа, которым присвоено общее имя. Каждый элемент массива имеет свой номер (**индекс**).

Одномерный массив



Решение разнообразных задач, связанных с обработкой массивов, базируется на решении типовых задач:

- ◆ суммирование элементов массива;
- ◆ поиск элемента с заданными свойствами;
- ◆ сортировка массива.

ОДНОМЕРНЫЕ МАССИВЫ

Размерность массива — это количество индексов, необходимое для однозначного доступа к элементу массива. Массивы с одним индексом называют одномерными, с двумя — двумерными и т. д.

10	50	1	3	50	14	21	50	10	21
----	----	---	---	----	----	----	----	----	----

Здесь:

- ◆ трём равен 3-й элемент
- ◆ десяти равны 0-й и 8-й элементы
- ◆ нет элемента, равного 12

СОЗДАНИЕ МАССИВА

Перед использованием в программе массив необходимо создать, иначе обращение к несуществующему элементу вызовет ошибку и аварийное завершение программы.



Создать массив можно перечислением элементов через запятую в квадратных скобках: $A = [1, 2, -3, 5, 7]$

Так будет создан массив из пяти элементов, каждый из которых равен 0:

$$N = 5$$
$$A = [0] * N$$


ЗАПОЛНЕНИЕ МАССИВА



Небольшие массивы можно вводить с клавиатуры:

```
N = 5
A = [0]*N
for i in range(N):
    A[i] = int(input())
```

Цикл с параметром, выполняющий оператор ввода отдельно для каждого элемента массива

```
for i in range(N):
    print("A[{}]=".format(i), end="\n")
    A[i] = int(input())
```

Цикл с параметром, выводящий подсказку - индекс элемента



ЗАПОЛНЕНИЕ МАССИВА

Задавать значения элементов массива можно по формуле с помощью оператора присваивания.

```
for i in range(N) :  
    A[i] = 2 * i
```

Для работы со случайными числами вначале подключается функция randint модуля random:

```
from random import randint  
for i in range(N) :  
    A[i] = randint(10, 110)
```



ВЫВОД МАССИВА

Массив можно вывести как один объект:

```
print(A)
```

Можно вывести элементы массива на экран в строку:

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

Значения элементов массива можно вывести в столбик:

```
for i in range(len(A)) :  
    print(A[i])
```



Программа, выполняющая:

- ◆ заполнение целочисленного массива A, состоящего из 10 элементов, случайными числами, значения которых изменяются в диапазоне от 0 до 99;
- ◆ вывод массива A на экран.

```
N = 10
```

```
A = [0]*N
```

```
from random import randint
```

```
for i in range(N)
```

```
    A[i] = randint(0, 99)
```

```
for i in range(N)
```

```
    print('A[' , i, '] = ' , A[i])
```

Создание массива

*Подключение генератора
случайных чисел*

Заполнение массива

Вывод массива

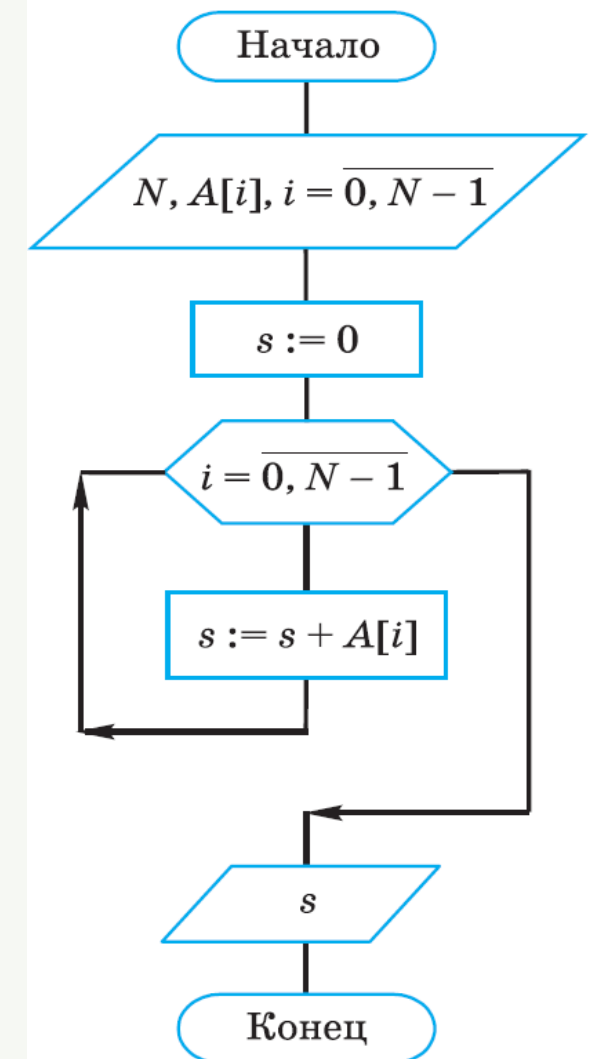
ВЫЧИСЛЕНИЕ СУММЫ ЭЛЕМЕНТОВ МАССИВА

Суммирование элементов массива осуществляется за счёт поочерёдного добавления слагаемых:

Определяется ячейка памяти (переменная **S**), в которой будет последовательно накапливаться результат суммирования

Переменной **S** присваивается начальное значение **0** - число, не влияющее на результат сложения

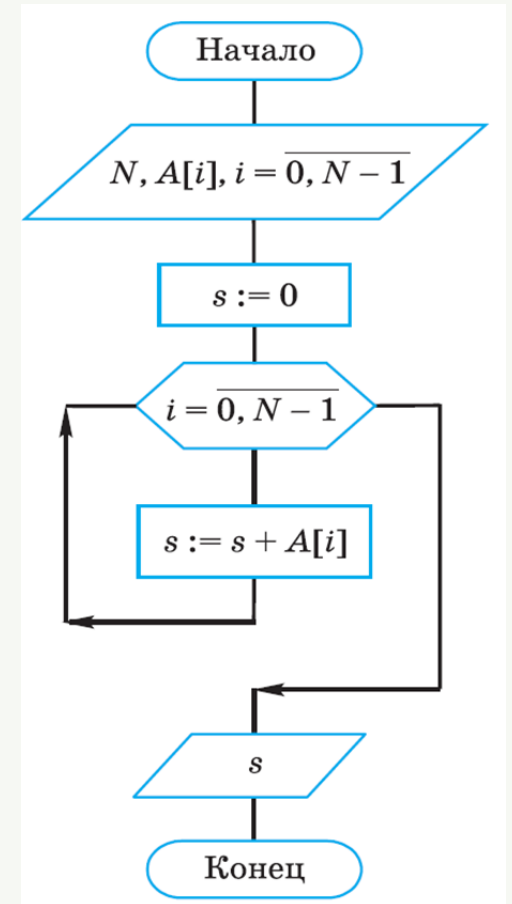
N раз текущее значение переменной **s** складывается со значением текущего элемента массива **A[i]**; полученный результат присваивается переменной **S**.



Σ

Процесс поочерёдного добавления слагаемых в сумму:

Значение i	Операция присваивания	Текущее значение s
	$s := 0$	0
0	$s := s + A[0]$	$0 + A[0]$
1	$s := s + A[1]$	$0 + A[0] + A[1]$
2	$s := s + A[2]$	$0 + A[0] + A[1] + A[2]$
...
$N-1$	$s := s + A[N-1]$	$0 + A[0] + A[1] + A[2] + \dots + A[N-1]$



ПРОГРАММА НА ЯЗЫКЕ PYTHON

```
N = 10
```

```
A = [0]*N
```

Задание массива

```
from random import randint
```

Подключение генератора случайных чисел

```
for i in range(N):
```

```
    A[i] = randint(50, 200)
```

Заполнение массива

```
for i in range(N):
```

```
    print('A[' , i, ']= ' , A[i])
```

Вывод массива

```
s = 0
```

```
for i in range(N):
```

```
    s +=A[i]
```

Вычисление суммы элементов массива

```
print(' s= ',s)
```

Вывод результатов

ВЫЧИСЛЕНИЕ СУММЫ ЭЛЕМЕНТОВ МАССИВА

$A = [100, 120, 130, 80, 70]$

Значение i	Операция присваивания	Текущее значение s
	$s := 0$	0
$i = 0$	$s := s + A[0]$	100
$i = 1$	$s := s + A[1]$	220
$i = 2$	$s := s + A[2]$	350
$i = 3$	$s := s + A[3]$	430
$i = 4$	$s := s + A[4]$	500

Типовые задачи поиска в массиве

Нахождение наибольшего (наименьшего) элемента массива

Нахождение элемента массива, значение которого равно заданному значению

Нахождение количества (суммы) элементов массива, удовлетворяющих заданному условию



Нахождение значения наибольшего элемента в стопке карточек с записанными числами:



1. Взять верхнюю карточку, записать на доске (запомнить) число как наибольшее.

2. Взять следующую карточку, сравнить числа. Если на карточке число больше, то стереть старую запись и записать это число.

3. Повторить действия, описанные в пункте 2 для всех оставшихся карточек.



! При организации поиска наибольшего элемента массива правильнее искать его индекс.

НАХОЖДЕНИЕ НАИБОЛЬШЕГО ЭЛЕМЕНТА МАССИВА

```
N = 10
```

```
A=[0]*N
```

Задание массива

```
-----  
from random import randint
```

Подключение генератора
случайных чисел

```
-----  
for i in range(N):  
    A[i] = randint(0,99)  
    print('A[' , i, ']=' , A[i])
```

Заполнение и вывод
массива

```
-----  
imax=0
```

```
for i in range(1,N):  
    if A[i] > A[imax]: imax = i
```

Поиск индекса
наибольшего элемента

```
-----  
print('Наибольший элемент:', A[imax])
```

Вывод результата

A = [100, 120, 130, 80, 70]

N = 5

A=[0]*N

```
from random import randint
```

```
for i in range(N):
```

```
    A[i] = randint(0,99)
```

```
    print('A[' ,i, ']=' ,A[i])
```

```
imax=0
```

```
for i in range(1,N):
```

```
    if A[i] > A[imax]: imax = i
```

```
print('Наибольший элемент:', A[imax])
```

imax	i	A[i] > A[imax]
0	1	120 > 100 (Да)
1	2	130 > 120 (Да)
2	3	80 > 130 (Нет)
2	4	70 > 130 (Нет)
Наибольший элемент: 130		

Результат нахождения элемента массива, значение которого равно заданному значению:

◆ k — индекс элемента массива такой, что $A[k] = x$, где x — заданное число;

◆ сообщение о том, что искомого элемента в массиве не обнаружено.



ПОИСК ЭЛЕМЕНТА С ЗАДААННЫМИ СВОЙСТВАМИ

```
N = 10
```

```
A = [0]*N
```

```
from random import randint
```

```
for i in range(N):
```

```
    A[i] = randint(0,99)
```

```
    print('A[',i, ']= ', A[i])
```

```
x = int(input('x='))
```

```
nx=-1
```

```
for i in range(0,N):
```

```
    if A[i] == x: nx = i
```

```
if nx == -1:
```

```
    print('Нет')
```

```
else:
```

```
    print('nx=', nx)
```

Создание массива

Заполнение и вывод массива

*Поиск индекса элемента
массива, равного заданному*

Вывод результатов

A = [100, 80, 120, 130, 80, 70], x = 80

```
N = 10
```

```
A = [0]*N
```

```
from random import randint
for i in range(N):
    A[i] = randint(0,99)
    print('A['+i+', ']= ', A[i])
x = int(input('x='))
nx=-1
for i in range(0,N):
    if A[i] == x: nx = i
if nx == -1:
    print('Нет')
else:
    print('nx=', nx)
```

Какое значение индекса - 1 или 4 -
будет получено в результате
выполнения программы?

A = [100, 80, 120, 130, 80, 70], x = 80

```
N = 10
```

```
A = [0]*N
```

```
from random import randint
for i in range(N):
    A[i] = randint(0,99)
    print('A['+i+', ']= ', A[i])
x = int(input('x='))
nx=-1
for i in range(0,N):
    if A[i] == x: nx = i
if nx == -1:
    print('Нет')
else:
    print('nx=', nx)
```

Какое значение индекса - 1 или 4 -
будет получено в результате
выполнения программы?

4 - индекс последнего
элемента, равного 80

A = [100, 80, 120, 130, 80, 70], x = 80

```
nx = -1
for i in range(N) :
    if A[i] == x:
        nx = i
        break
if nx >= 0:
    print('A[{}]={}'.format(nx, x))
else:
    print('Элемент не найден')
```

Какое значение индекса - 1 или 4 -
будет получено в результате
выполнения программы?

A = (100, 80, 120, 130, 80, 70), x = 80

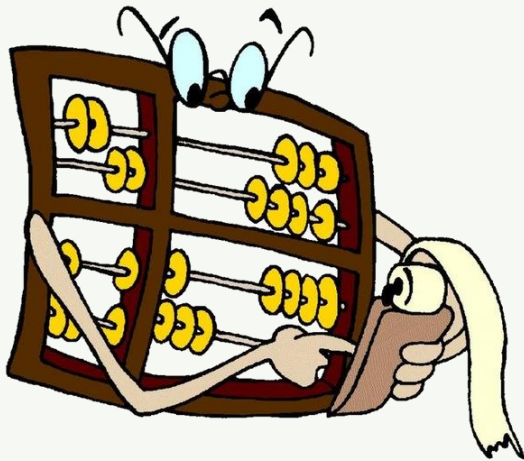
```
nx = -1
for i in range(N):
    if A[i] == x:
        nx = i
        break
if nx >= 0:
    print('A[{}]={}'.format(nx, x))
else:
    print('Элемент не найден')
```

Какое значение индекса - 1 или 4 -
будет получено в результате
выполнения программы?

1 - индекс первого
элемента, равного 80

Подсчёт количества элементов массива, удовлетворяющих некоторому условию:

- ◆ вводится переменная, значение которой увеличивается на единицу каждый раз, когда найден нужный элемент



```
N = 10; A = [0]*N
from random import randint
for i in range(N):
    A[i] = randint(0,99)
    print('A[' ,i, ']= ', A[i])
k = 0
x=int(input('x='))
for i in range(0,N):
    if A[i] > x: k += 1
print('k=', k)
```

10	60	21	53	58	14	28	50	10	51
----	----	----	----	----	----	----	----	----	----

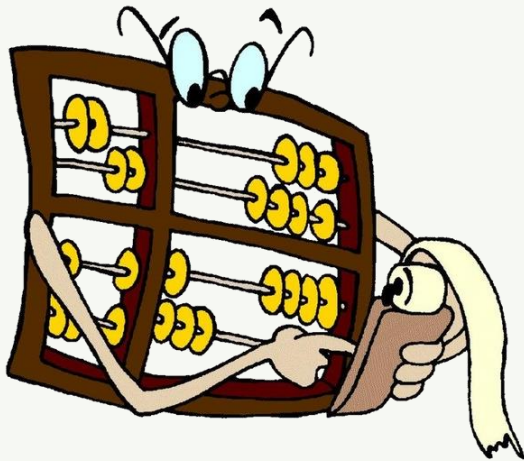
Количество каких элементов массива подсчитывается с помощью следующего фрагмента программы?

```
k=0
for i in range(10):
    if A[i]%2 == 0: k += 1
print('k=', k)
```

10	60	21	53	58	14	28	50	10	51
----	----	----	----	----	----	----	----	----	----

Суммирование значений элементов массива, удовлетворяющих некоторому условию:

- ◆ вводится переменная, к значению которой прибавляется значение найденного элемента массива



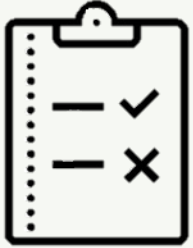
```
N = 10; A = [0]*N
from random import randint
for i in range(N):
    A[i] = randint(0,99)
    print('A['+i, ']= ', A[i])
s = 0
x=int(input('x='))
for i in range(0,N):
    if A[i] > x: s += A[i]
print('s=', s)
```

10	50	1	3	50	14	21	50	10	21
----	----	---	---	----	----	----	----	----	----

Какому условию удовлетворяют элементы массива, значения которых суммируются с помощью следующего фрагмента программы?

```
s=0
for i in range(10):
    if A[i]%2 == 0 and A[i]%10 == 4: s += A[i]
print('s=', s)
```

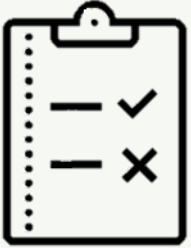
10	60	21	54	58	14	28	50	10	51
----	----	----	----	----	----	----	----	----	----



ПРИМЕР 1

В классе 20 учеников. Известно, что за контрольную работу по информатике они получили отметки «3», «4» и «5». Составим программу, подсчитывающую:

- 1) количество отметок «5», полученных учениками за контрольную работу;
- 2) средний балл, полученный учениками за контрольную работу;
- 3) каких отметок было получено больше всего.



ПРИМЕР 1

В классе 20 учеников. Известно, что за контрольную работу по информатике они получили отметки «3», «4» и «5». Составим программу, подсчитывающую:

- 1) количество отметок «5», полученных учениками за контрольную работу;
- 2) средний балл, полученный учениками за контрольную работу;
- 3) каких отметок было получено больше всего.

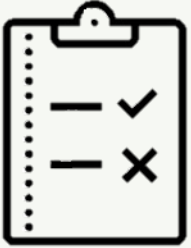
Для решения этой задачи создадим одномерный массив A из 20 случайных целых чисел, принадлежащих отрезку $[3; 5]$, и выведем этот массив на экран:

```
N = 20
A = [0] * N
from random import randint
for i in range(N):
    A[i] = randint(3, 5)
for i in range(N):
    print('A[' + i + ']=' + A[i])
```

В переменной k_5 подсчитаем количество отметок «5», полученных учениками за контрольную работу:

```
k5 = 0
for i in range(N):
    if A[i] == 5:
        k5 += 1
print('k5=' + k5)
```





ПРИМЕР 1

В классе 20 учеников. Известно, что за контрольную работу по информатике они получили отметки «3», «4» и «5». Составим программу, подсчитывающую:

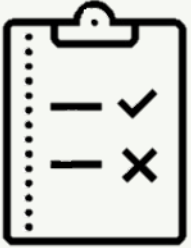
- 1) количество отметок «5», полученных учениками за контрольную работу;
- 2) средний балл, полученный учениками за контрольную работу;
- 3) каких отметок было получено больше всего.

Для подсчёта среднего балла (переменная `av`) предварительно надо найти сумму всех его элементов (переменная `s`):

```
s = 0
for i in range(N):
    s += A[i]
av = s/N
print('av=', av)
```

Для подсчёта количества отметок «4» и «3» введём переменные `k4` и `k3`:

```
k4 = 0; k3 = 0
for i in range(N):
    if A[i] == 4: k4 += 1
    if A[i] == 3: k3 += 1
```



ПРИМЕР 1

В классе 20 учеников. Известно, что за контрольную работу по информатике они получили отметки «3», «4» и «5». Составим программу, подсчитывающую:

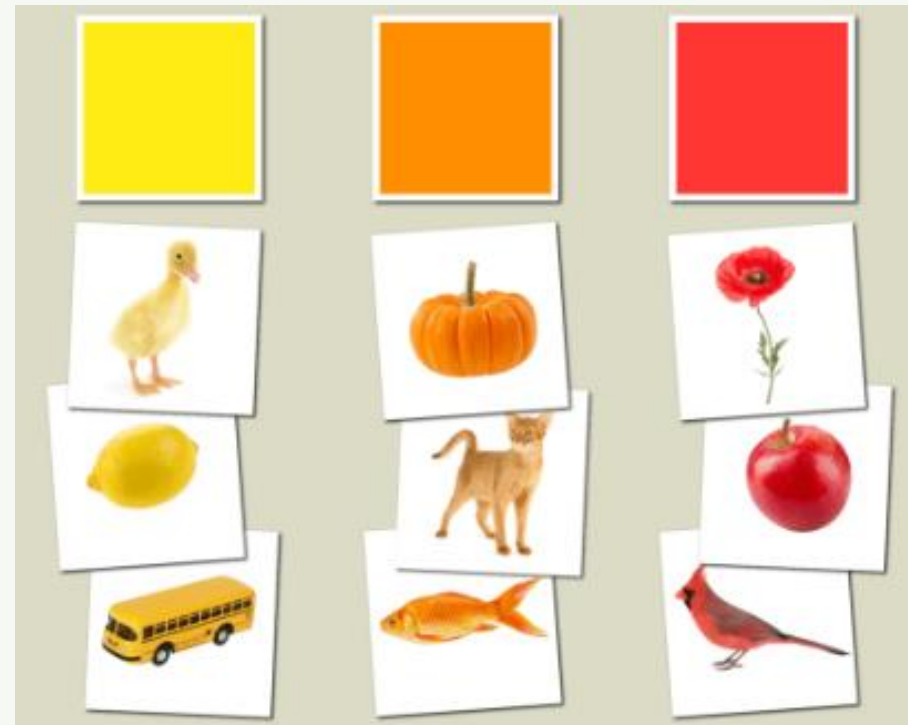
- 1) количество отметок «5», полученных учениками за контрольную работу;
- 2) средний балл, полученный учениками за контрольную работу;
- 3) каких отметок было получено больше всего.

```
max = k5
for i in range(N) :
    if k4 > max: max = k4
    if k3 > max: max = k3
if k5 == max: print('Больше 5')
if k4 == max: print('Больше 4')
if k3 == max: print('Больше 3')
```

СОРТИРОВКА

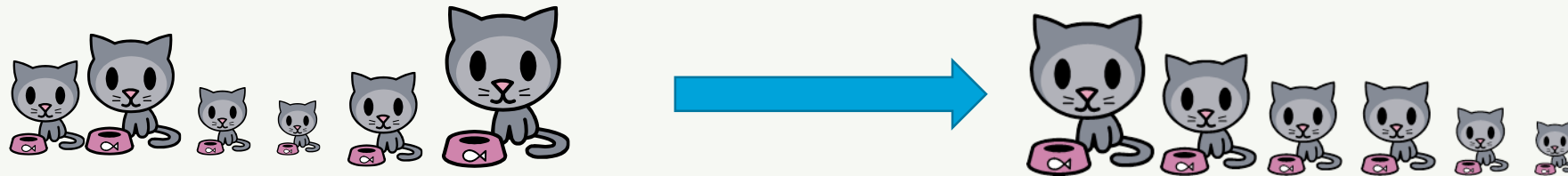
Сортировка (упорядочение) - перераспределение элементов множества в некотором порядке.

Цель сортировки — облегчить последующий поиск элементов. Если элементы упорядочены, то среди них можно быстрее отыскать то, что нужно.

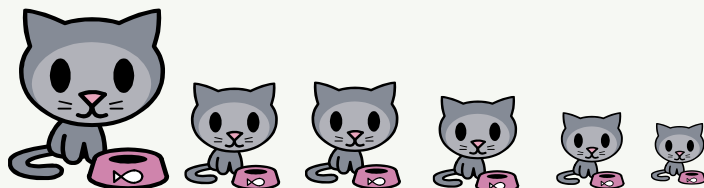
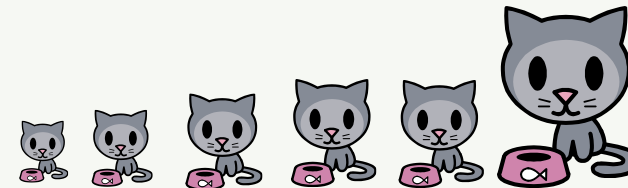


СОРТИРОВКА МАССИВА

Сортировка (упорядочение) массива - это перераспределение его элементов в некотором определённом порядке.



Порядок, при котором в массиве первый элемент имеет самое маленькое значение, а значение каждого следующего элемента не меньше значения предыдущего элемента, называют **неубывающим**.



Порядок, при котором в массиве первый элемент имеет самое большое значение, а значение каждого следующего элемента не больше значения предыдущего элемента, называют **невозрастающим**.

НАЙДИТЕ РАЗНИЦУ



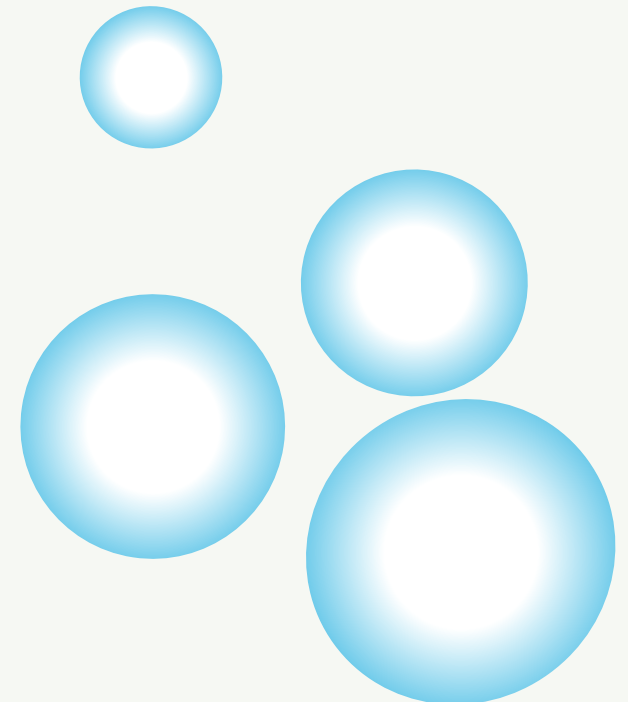
НАЙДИТЕ РАЗНИЦУ



СОРТИРОВКА ПУЗЫРЬКОМ

Сортировка пузырьком — это метод сортировки массивов путём последовательного сравнения и обмена значений соседних элементов, если предшествующий оказывается больше последующего. В процессе выполнения данного алгоритма элементы с большими значениями оказываются в конце массива, а элементы с меньшими значениями постепенно перемещаются по направлению к началу списка.

Своё название алгоритм получил благодаря следующей ассоциации: если сортировать этим алгоритмом массив по неубыванию, то максимальный элемент «тонет», а «лёгкие» элементы поднимаются на одну позицию к началу массива на каждом шаге алгоритма.



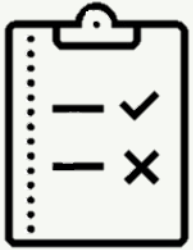
ПОВТОРЯЮЩИЕСЯ ПРОХОДЫ ПО СОРТИРУЕМОМУ МАССИВУ

При программировании алгоритма используются вложенные циклы: внешний и внутренний.

Если в массиве N элементов, то выполняется $N - 1$ проходов внешнего цикла: когда второй элемент становится на своё место, то первый уже однозначно минимальный и находится на своём месте.

Количество проходов внутреннего цикла зависит от номера прохода внешнего цикла: если конец массива уже отсортирован, то выполнять проход по этим элементам нет необходимости.





ПРИМЕР 2

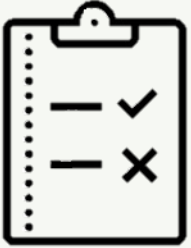
Рассмотрим процесс сортировки **пузырьком** на примере массива $A = [5, 4, 3, 2, 1]$.

```
1 for i in range(N-1):
2     for j in range(N-i-1):
3         if A[j] > A[j+1]:
4             temp = A[j]
5             A[j] = A[j+1]
6             A[j+1] = temp
```

Для обмена значений соседних элементов массива мы использовали промежуточную переменную `temp` и оформили обмен в строках 4–6 программного кода. Можно применить множественное присваивание, тогда код обмена будет выглядеть так:

```
A[j], A[j+1] = A[j+1], A[j]
```

Индекс	0	1	2	3	4	
Значение	5	4	3	2	1	
Проход 1	1.1	4	5	3	2	1
	1.2	4	3	5	2	1
	1.3	4	3	2	5	1
	1.4	4	3	2	1	5
Проход 2	2.1	3	4	2	1	5
	2.2	3	2	4	1	5
	2.3	3	2	1	4	5
Проход 3	3.1	2	3	1	4	5
	3.2	2	1	3	4	5
Проход 4	4.1	1	2	3	4	5
Итог:	1	2	3	4	5	



ПРИМЕР 2

Сортировку пузырьком можно оформить в виде процедуры:

```
def bubble(A):  
    for i in range(N-1):  
        for j in range(N-i-1):  
            if A[j] > A[j+1]:  
                A[j], A[j+1] = A[j+1], A[j]
```



СОРТИРОВКА ВЫБОРОМ

Сортировка элементов массива по невозрастанию выбором осуществляется следующим образом:

1. В массиве выбирается максимальный элемент

2. Максимальный и первый элемент меняются местами, после чего первый элемент считается отсортированным

3. В неотсортированной части массива снова выбирается максимальный элемент; он и первый неотсортированных элемент массива меняются местами

Действия пункта 3 повторяются с неотсортированными элементами массива, пока не останется один неотсортированный элемент (минимальный)

СОРТИРОВКА ВЫБОРОМ



Индекс	0	1	2	3	4	5	6	7	
Значение	0	1	9	2	4	3	6	5	
Шаги	1	0	1	9	2	4	3	6	5
	2	9	1	0	2	4	3	6	5
	3	9	6	0	2	4	3	1	5
	4	9	6	5	2	4	3	1	0
	5	9	6	5	4	2	3	1	0
	6	9	6	5	4	3	2	1	0
	7	9	6	5	4	3	2	1	0
	Итог:	9	6	5	4	3	2	1	0

СОРТИРОВКА ВЫБОРОМ

```
for i in range(N-1):  
    imax = i  
    for j in range(i+1, N):  
        if A[j] > A[imax]: imax = j  
    A[i], A[imax] = A[imax], A[i]
```

ДВОИЧНЫЙ ПОИСК В УПОРЯДОЧЕННОМ ОДНОРОДНОМ МАССИВЕ

В программировании поиск — одна из наиболее часто встречающихся задач невычислительного характера. Можно выделить следующие типовые задачи поиска:

- 1) найти элемент массива, обладающий заданными свойствами;
- 2) найти количество (сумму) элементов массива, обладающих заданными свойствами.

Если мы имеем дело с неотсортированным массивом, то для решения любой из задач поиска в программе необходимо организовать последовательный просмотр элементов массива и сравнение значения очередного просматриваемого элемента с неким образцом.

Для поиска элемента в отсортированном массиве используется метод двоичного поиска.

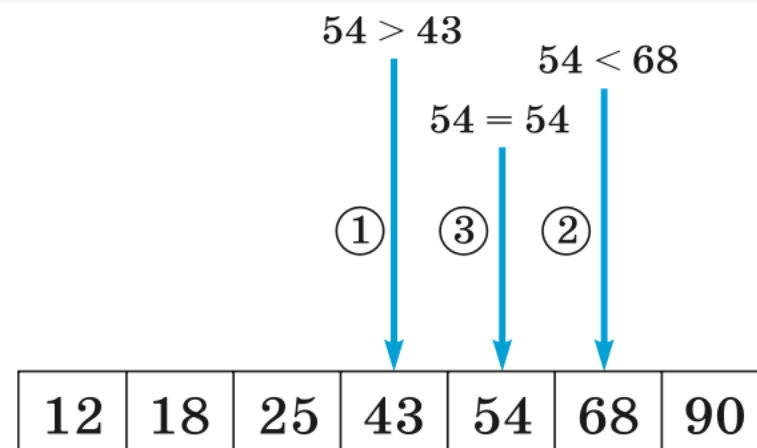


ДВОИЧНЫЙ ПОИСК

Двоичный поиск — классический алгоритм поиска элемента в отсортированном одномерном массиве, использующий дробление массива и его частей на половины.

1. Определение значения среднего элемента — элемента в середине массива.
2. Сравнение значения среднего элемента с искомым значением. Если значение среднего элемента оказывается равным искомому, поиск завершается. Иначе, в зависимости от того, больше оно или меньше значения среднего элемента, дальнейший поиск будет происходить только в правой или только в левой половине массива. Для этого одна из границ исследуемого массива сдвигается. Если искомое значение больше значения среднего элемента, то нижняя граница сдвигается за средний элемент на один элемент справа. Если искомое значение меньше значения среднего элемента, то верхняя граница сдвигается на элемент перед средним.
3. Описанный выше алгоритм повторяется для выбранной половины массива.

Выполнение алгоритма на примере поиска числа 54 в массиве $A = [12, 18, 25, 43, 54, 68, 90]$:



```
value = int(input())
low = 0
high = N - 1
mid = N // 2
while A[mid] != value and low <= high:
    if value > A[mid]:
        low = mid + 1
    else:
        high = mid - 1
    mid = (low + high) // 2
if low > high:
    print("No value")
else:
    print("ID =", mid)
```

МАССИВЫ И ПОСЛЕДОВАТЕЛЬНОСТИ ЦЕЛЫХ ЧИСЕЛ

Если требуется обработать некоторое множество однотипных целочисленных данных, то в зависимости от решаемой задачи можно пойти одним из двух путей:

- ◆ сохранить все элементы множества в памяти компьютера как массив и организовать обработку массива;
- ◆ завести одну переменную, в которую последовательно считывать каждый отдельный элемент множества и сразу же производить его обработку.



Камера наблюдения, установленная в населённом пункте, регистрирует в автоматическом режиме скорость проезжающих мимо неё автомобилей. Программа, которую вам необходимо составить, получает на вход значения скорости проехавших $N \leq 1000$ автомобилей.

Программа должна анализировать скорость каждого автомобиля и анализировать получаемую информацию.



МАССИВ ИЛИ ПОСЛЕДОВАТЕЛЬНОСТЬ?

Надо выяснить:

- 1) количество автомобилей, проехавших со скоростью, превышающей 60 км/ч
- 2) максимальную скорость проехавших автомобилей
- 3) количество автомобилей, проехавших с максимальной скоростью
- 4) среднюю скорость проехавших автомобилей
- 5) количество автомобилей, проехавших со скоростью, ниже средней



Массив — это набор элементов одного типа, которым присвоено общее имя. Каждый элемент массива имеет свой номер (индекс).

Размерность массива — это количество индексов, необходимое для однозначного доступа к элементу массива. Массивы с одним индексом называют одномерными, с двумя — двумерными и т. д.

Под сортировкой (упорядочением) массива понимают перераспределение значений его элементов в некотором определённом порядке.

Сортировка пузырьком, сортировка выбором — классические алгоритмы сортировки одномерных массивов. Как правило, вместо самостоятельного написания алгоритмов сортировки программисты используют встроенные возможности сортировки языка программирования.

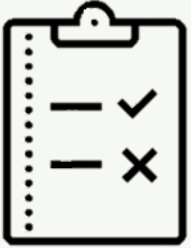
Двоичный поиск — классический алгоритм поиска элемента в отсортированном одномерном массиве, использующий дробление массива и его частей на половины.



ВОПРОСЫ И ЗАДАНИЯ

Для учеников спортивной школы, которые остаются на вторую тренировку, ввели полдник. На полдник каждый получает стакан молока (200 мл) и печенье (4 шт.). Составьте программу, рассчитывающую, сколько нужно закупить пакетов молока (объём одного пакета — 900 мл) и пачек печенья (в одной пачке 10 штук), если известно общее количество учеников спортивной школы и количество тренировок (1 или 2) у каждого из них в день.

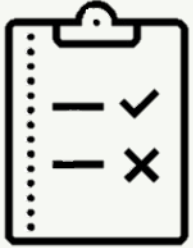




ВОПРОСЫ И ЗАДАНИЯ

Дано натуральное десятичное число $N \leq 32\,000$. Напишите программу, в которой:

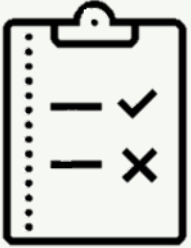
- 1) из цифр данного числа формируется одномерный целочисленный массив;
- 2) определяются наибольшая и наименьшая цифры данного числа;
- 3) находятся сумма и произведение цифр, образующих данное число.



ВОПРОСЫ И ЗАДАНИЯ

Требуется упорядочить по весу в порядке неубывания N непрозрачных банок с чаем, имея в своём распоряжении только чашечные весы без гирь. Опишите возможный алгоритм решения этой задачи.





ВОПРОСЫ И ЗАДАНИЯ

Запишите алгоритм сортировки пузырьком с использованием цикла `while`.



Массив - это поименованная совокупность однотипных элементов, упорядоченных по индексам, определяющим положение элементов в массиве.

