



ИНЖЕНЕРНО-ТЕХНОЛОГИЧЕСКАЯ ШКОЛА # 777

Алгоритмы обработки массивов Программирование на Python



Поиск в массиве

Найти элемент, равный X:

```
i = 0
while A[i] != X:
    i += 1
print ( "A[" , i , "]" = " , X , sep = " " )
```

? Что плохо?

```
i = 0
while i < N and A[i] != X:
    i += 1
if i < N:
    print ( "A[" , i , "]" = " , X , sep = " " )
else:
    print ( "Не нашли!" )
```

? Что если такого нет?



Поиск в массиве

Вариант с досрочным выходом:

номер найденного
элемента

```
nX = -1
for i in range ( N ):
    if A[i] == X:
        nX = i
        break
if nX >= 0:
    print ( "A[" , nX , "]" = " , X , sep = " " )
else:
    print ( "Не нашли!" )
```

досрочный
выход из цикла



ПОИСК В МАССИВЕ

Варианты в стиле Python:

```
for i in range ( N ) :
    if A[i] == X:
        print ( "A[" , i , "]" = " , X , sep = " " )
        break
else:
    print ( "Не нашли!" )
```

если не было досрочного выхода из цикла

```
if X in A:
    nX = A.index (X)
    print ( "A[" , nX , "]" = " , X , sep = " " )
else:
    print ( "Не нашли!" )
```



Задачи

«А»: Заполните массив случайными числами в интервале $[0,5]$. Введите число X и найдите все значения, равные X .

Пример:

Массив :

1 2 3 1 2

Что ищем :

2

Нашли: $A[2]=2$, $A[5]=2$

Пример:

Массив :

1 2 3 1 2

Что ищем :

6

Ничего не нашли.



Задачи

«В»: Заполните массив случайными числами в интервале $[0,5]$. Определить, есть ли в нем элементы с одинаковыми значениями, стоящие рядом.

Пример:

Массив :

1 2 3 3 2 1

Есть : 3

Пример:

Массив :

1 2 3 4 2 1

Нет



Задачи

«С»: Заполните массив случайными числами. Определить, есть ли в нем элементы с одинаковыми значениями, не обязательно стоящие рядом.

Пример:

Массив :

3 2 1 3 2 5

Есть: 3, 2

Пример:

Массив :

3 2 1 4 0 5

Нет



Максимальный элемент

```
M = A[0]
```

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

```
    if A[i] > M:
```

```
        M = A[i]
```

```
print ( M )
```



Если `range(N)` ?

Варианты в стиле Python:

```
M = A[0]
```

```
for x in A:
```

```
    if x > M:
```

```
        M = x
```



Как найти его номер?

```
M = max ( A )
```




Максимальный элемент и его номер

```
M = A[0]; nMax = 0
for i in range(1, N):
    if A[i] > M:
        M = A[i]
        nMax = i
print ( "A[" , nMax , "]" = " , M , sep = " " )
```

! Что можно улучшить?

! По номеру элемента можно найти значение!

```
nMax = 0
for i in range(1, N):
    if A[i] > A[nMax]:
        nMax = i
print ( "A[" , nMax , "]" = " , A[nMax] , sep = " " )
```



Максимальный элемент и его номер

Вариант в стиле Python:

```
M = max (A)
nMax = A . index (M)
print ( "A[" , nMax , "]" = " , M , sep = " " )
```

номер заданного
элемента (первого из...)



Задачи

«А»: Заполнить массив случайными числами и найти минимальный и максимальный элементы массива и их номера.

Пример:

Массив :

1 2 3 4 5

Минимальный элемент: $A[1]=1$

Максимальный элемент: $A[5]=5$

«В»: Заполнить массив случайными числами и найти два максимальных элемента массива и их номера.

Пример:

Массив :

5 5 3 4 1

Максимальный элемент: $A[1]=5$

Второй максимум: $A[2]=5$



Задачи

«С»: Введите массив с клавиатуры и найдите (за один проход) количество элементов, имеющих максимальное значение.

Пример:

Массив :

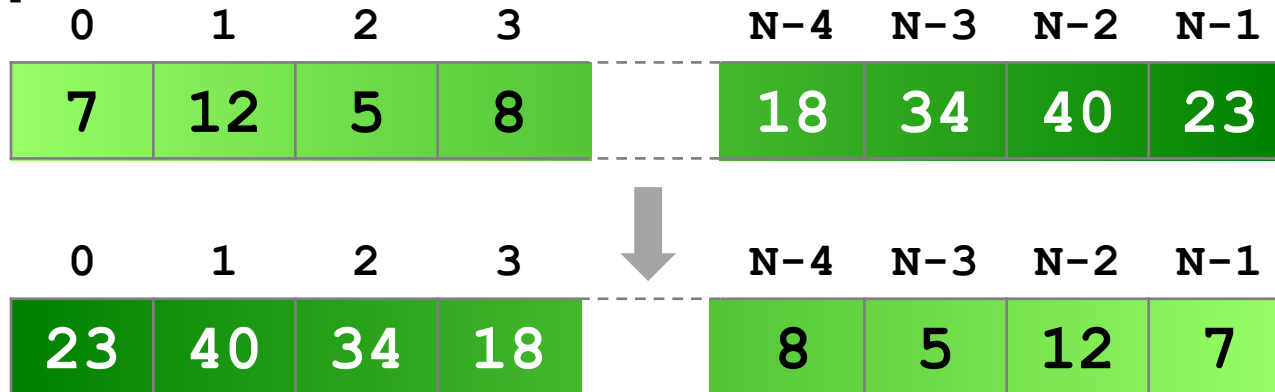
3 4 5 5 3 4 5

Максимальное значение 5

Количество элементов 3



Реверс массива



«Простое» решение:

остановиться на середине!

```
for i in range(N//2):  
    поменять местами A[i] и A[N-1-i]
```

? Что плохо?



Реверс массива

```
for i in range(N//2):  
    c = A[i]  
    A[i] = A[N-1-i]  
    A[N-1-i] = c
```

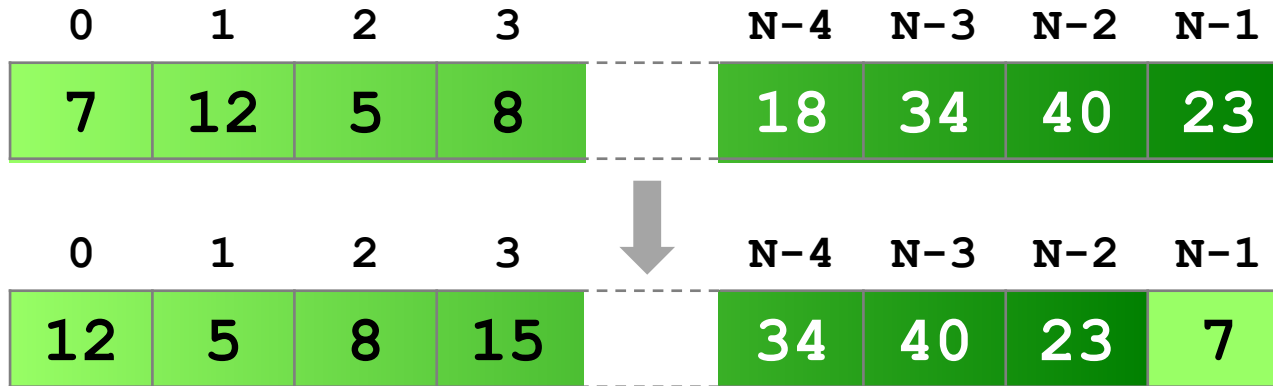
Варианты в стиле Python:

```
for i in range(N//2):  
    A[i], A[N-i-1] = A[N-i-1], A[i]
```

```
A.reverse()
```



Циклический сдвиг элементов



«Простое» решение:

```
for i in range(N-1):  
    A[i] = A[i+1]
```

? Почему не до N?

? Что плохо?



Срезы в Python

0	1	2	3		N-4	N-3	N-2	N-1
7	12	5	8		18	34	40	23

! Последний элемент не входит в срез!

$A[1:3] \rightarrow [12, 5]$

$A[2:3] \rightarrow [5]$

$A[:3] \rightarrow A[0:3] \rightarrow [7, 12, 5]$

с начала

$A[3:N-2] \rightarrow [8, \dots, 18, 34]$

$A[3:] \rightarrow A[3:N] \rightarrow [8, \dots, 18, 34, 40, 23]$

до конца

копия массива

$A[:] \rightarrow [7, 12, 5, 8, \dots, 18, 34, 40, 23]$



Срезы в Python – шаг

0	1	2	3	4	5	6	7	8
7	12	5	8	76	18	34	40	23

шаг

`A[1:6:2]` → [12, 8, 18]

`A[::3]` → [7, 8, 34]

`A[8:2:-2]` → [23, 34, 76]

`A[:: -1]` → [23, 40, 34, 18, 76, 8, 5, 12, 7]

реверс!

`A.reverse()`



Задачи

«А»: Заполнить массив случайными числами и выполнить циклический сдвиг элементов массива вправо на 1 элемент.

Пример:

Массив :

1 2 3 4 5 6

Результат :

6 1 2 3 4 5

«В»: Массив имеет четное число элементов. Заполнить массив случайными числами и выполнить реверс отдельно в первой половине и второй половине.

Пример:

Массив :

1 2 3 4 5 6

Результат :

3 2 1 6 5 4



Задачи

«С»: Заполнить массив случайными числами в интервале $[-100, 100]$ и переставить элементы так, чтобы все положительные элементы стояли в начала массива, а все отрицательные и нули – в конце. Вычислите количество положительных элементов.

Пример:

Массив :

20 -90 15 -34 10 0

Результат:

20 15 10 -90 -34 0

Количество положительных элементов : 3



Отбор нужных элементов

Задача. Отобрать элементы массива A ,
удовлетворяющие некоторому условию, в массив B .

Простое решение:

```
V = []  
сделать для i от 0 до N-1  
    если условие выполняется для A[i] то  
        добавить A[i] к массиву B
```

```
V = []  
for x in A:  
    if x % 2 == 0:  
        V.append(x)
```



Какие элементы выбираем?

добавить x в конец
массива B



Отбор нужных элементов

Задача. Отобрать элементы массива **A**,
удовлетворяющие некоторому условию, в массив **B**.

Решение в стиле Python:

перебрать все
элементы A

```
B = [ x for x in A  
      if x % 2 == 0 ]
```

если **x** – чётное
число



Особенности работы со

$A = [1, 2, 3]$

$B = A$

$A \rightarrow [1, 2, 3]$

$B \rightarrow [1, 2, 3]$



$A[0] = 0$

$A \rightarrow [0, 2, 3]$

$B \rightarrow [1, 2, 3]$

$A = [1, 2, 3]$

$B = A[:]$

копия массива A

$A \rightarrow [1, 2, 3]$

$B \rightarrow [1, 2, 3]$



$A[0] = 0$

$A \rightarrow [0, 2, 3]$

$B \rightarrow [1, 2, 3]$



Копирование списков

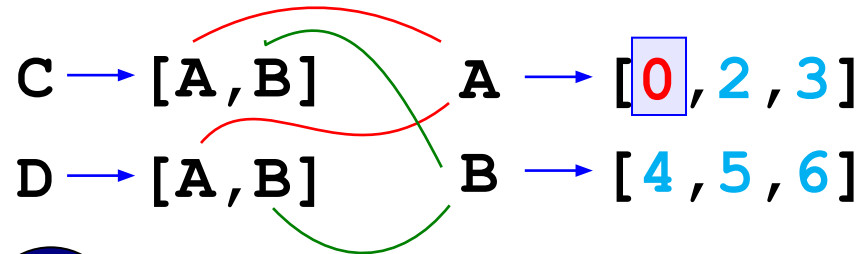
«Поверхностное» копирование:

```
import copy
A = [1, 2, 3]
B = copy.copy(A)
```

A → [1, 2, 3]

B → [4, 5, 6]

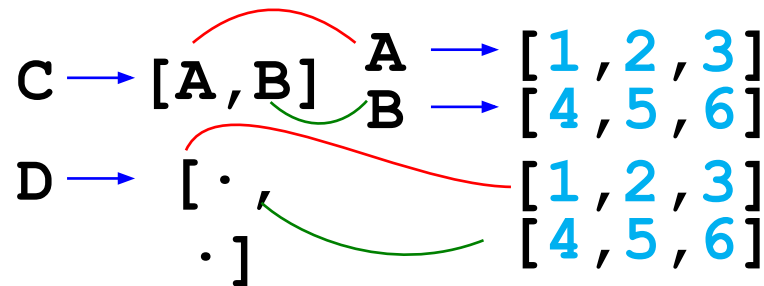
```
A = [1, 2, 3]
B = [4, 5, 6]
C = [A, B]
D = copy.copy(C)
C[0][0] = 0
```



! Влияет на C и D!

«Глубокое» копирование:

```
D = copy.deepcopy(C)
```





Задачи

«А»: Заполнить массив случайными числами в интервале $[-10, 10]$ и отобразить в другой массив все чётные отрицательные числа.

Пример:

Массив А:

-5 6 7 -4 -6 8 -8

Массив В:

-4 -6 -8

«В»: Заполнить массив случайными числами в интервале $[0, 100]$ и отобразить в другой массив все простые числа. Используйте логическую функцию, которая определяет, является ли переданное ей число простым.

Пример:

Массив А:

12 13 85 96 47

Массив В:

13 47



Задачи

«С»: Заполнить массив случайными числами и отобразить в другой массив все числа Фибоначчи. Используйте логическую функцию, которая определяет, является ли переданное ей число числом Фибоначчи.

Пример:

Массив А:

12 13 85 34 47

Массив В:

13 34