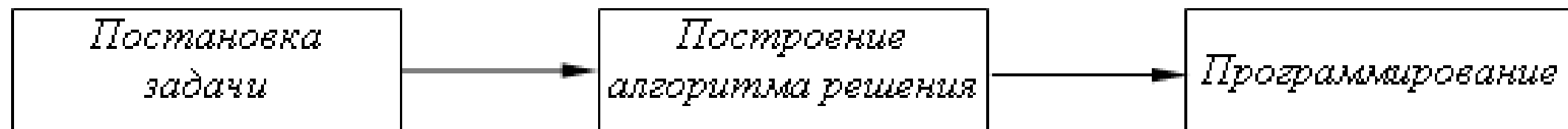


КУРС «ИНФОРМАТИКА»

Часть 1. Построение схем
алгоритмов

2023 – 2024 УЧЕБНЫЙ ГОД

Процесс решения задач на ПК



- **Постановка задачи** – это точная формулировка решения задачи на компьютере с описанием входной и выходной информации рассматриваемой предметной области.

Постановка задачи связана с конкретизацией основных параметров ее реализации, определением источников и структурой входной и выходной информации, востребованной пользователем.



Построение алгоритма решения задачи

- *Алгоритм* – последовательность действий, приводящая к заданному результату за конечное число шагов.


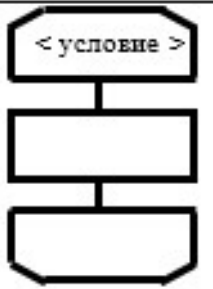

Алгоритм должен обладать следующими свойствами:

- *дискретностью* – разбиением процесса обработки информации на более простые этапы (шаги), выполнение которых компьютером или человеком не вызывает затруднений;
- *определенностью* (детерминированностью) – однозначностью получаемого результата при одних и тех же исходных данных;
- *результативностью* – обязательным получением желаемого результата за конечное число шагов при допустимых исходных данных;
- *массовостью* – применимостью алгоритма для решения определенного класса задач.

Базовые структуры

Графическое обозначение	Наименование	Пояснения	Соответствующие структуры языка Python
	Пуск-останов	Начало, конец, прерывание процесса обработки данных	
	Процесс, действие	Операция, в результате которой изменяются значение данных Может быть составным (из нескольких операторов)	<pre>a=b <оператор 1> ... <оператор k></pre>
	Ввод-вывод данных	Ввод-вывод без указания конкретного носителя	<code>input/print</code>
	Условие	Разветвление алгоритма в зависимости от некоторых условий	<pre>if <условие 1>: <оператор 1> elif <условие 2>: <оператор 2> else: <оператор 3></pre>
	Программа, подпрограмма	Часть алгоритма, требующая дополнительной детализации на последующих шагах	<pre>def <имя_функции> (<параметры>): <операторы></pre>

Циклические структуры

Графическое обозначение	Наименование	Пояснения	Соответствующие структуры языка Python
	Итеративный цикл	Повторяется заданное количество раз Используется для последовательной обработки всех элементов	<pre>for <переменная> in <последовательность>: <оператор> for <переменная> in <функция_range>: <оператор></pre> <p>Есть три способа вызова range():</p> <ol style="list-style-type: none"> 1. range(стоп) 2. range(старт, стоп) 3. range(старт, стоп, шаг)
	Цикл с предусловием	Может ни разу не выполниться (если не сработает условие входа) Используется для поиска элементов (первый встретившийся, последний встретившийся)	<pre>while <условие продолжения>: <оператор></pre>
	Цикл с постусловием	Выполняется хотя бы один раз. Если условие выхода не сработало, возвращается к началу цикла. Используется для организации циклов многократного повторения некоторых блоков программы.	<p>В Python цикл с постусловием отсутствует!</p> <pre>while True: if <условие выхода>: break</pre> <p>Оператор break используется для выхода из цикла. Python – прерывает его досрочно</p>

Вспомогательные структуры

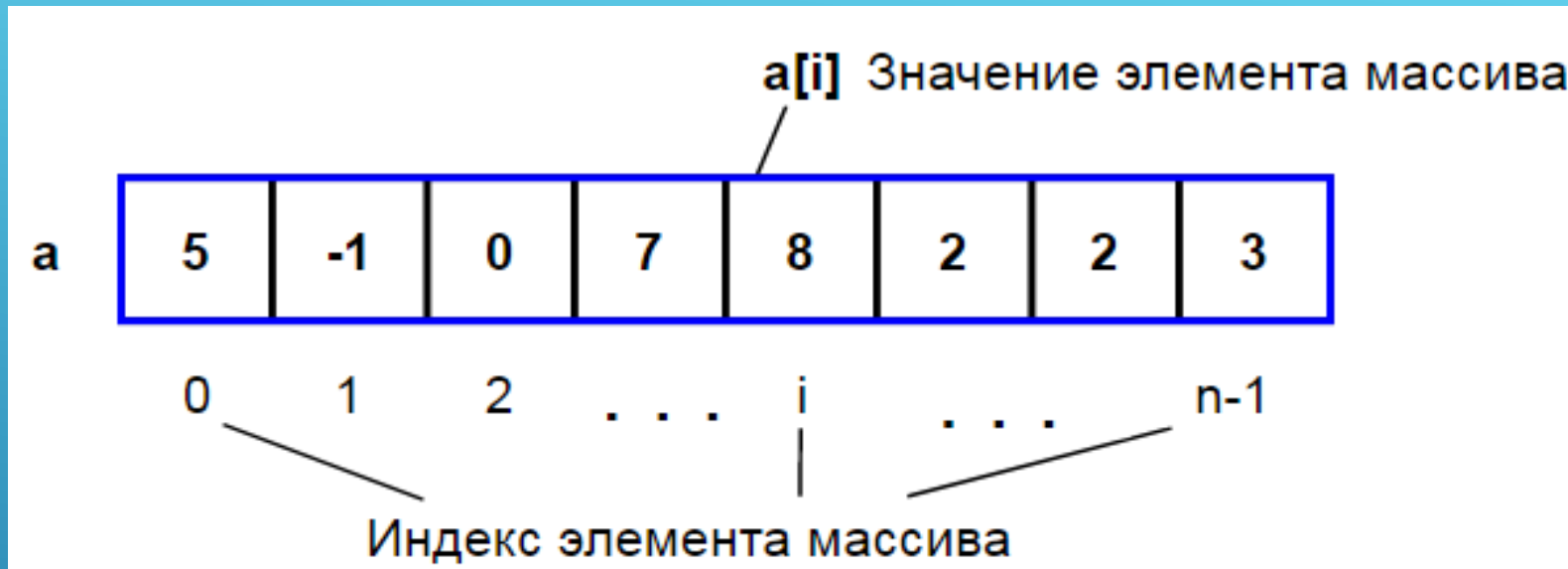
Графическое обозначение	Наименование	Пояснения
	Документ	Ввод-вывод данных, носителем которых является бумага.
	Дисплей	Ввод-вывод данных на монитор.
	Магнитный диск	Ввод-вывод данных на магнитный диск (в файл).
	Комментарий	Связь между элементом схемы и пояснением к нему.
	Соединители	Связь между прерванными линиями на одной страницы, связь между прерванными частями схем на разных стр.

КУРС «ИНФОРМАТИКА»

Часть 2. Построение схем алгоритмов
для одномерных массивов

2023 – 2024 УЧЕБНЫЙ ГОД

Одномерные массивы (векторы)

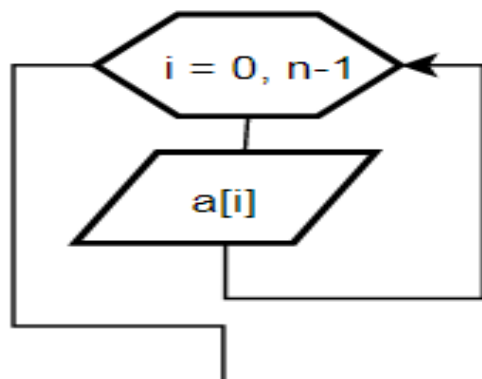


Массив – это совокупность однородных элементов, расположенных в памяти последовательно друг за другом и имеющих общее имя.

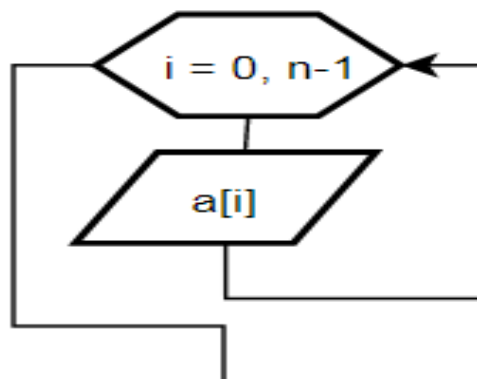
К любому элементу массива можно обратиться по его номеру (индексу)

Базовые алгоритмы обработки

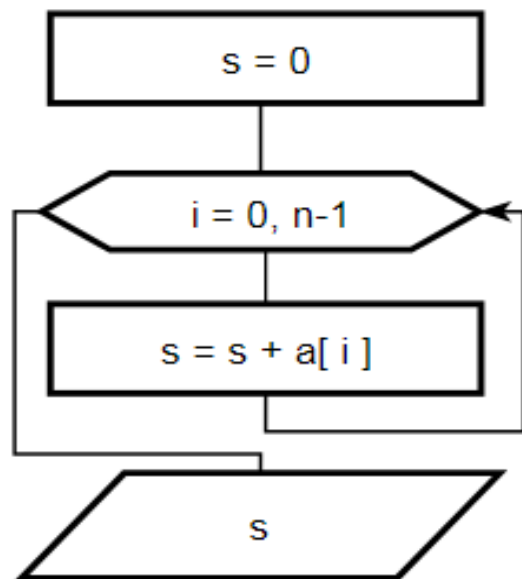
1. Ввод элементов массива



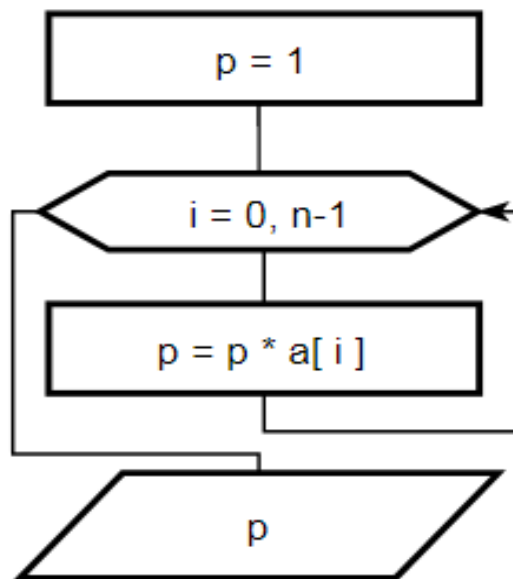
2. Вывод элементов массива



3. Сумма элементов массива



4. Произведение элементов массива



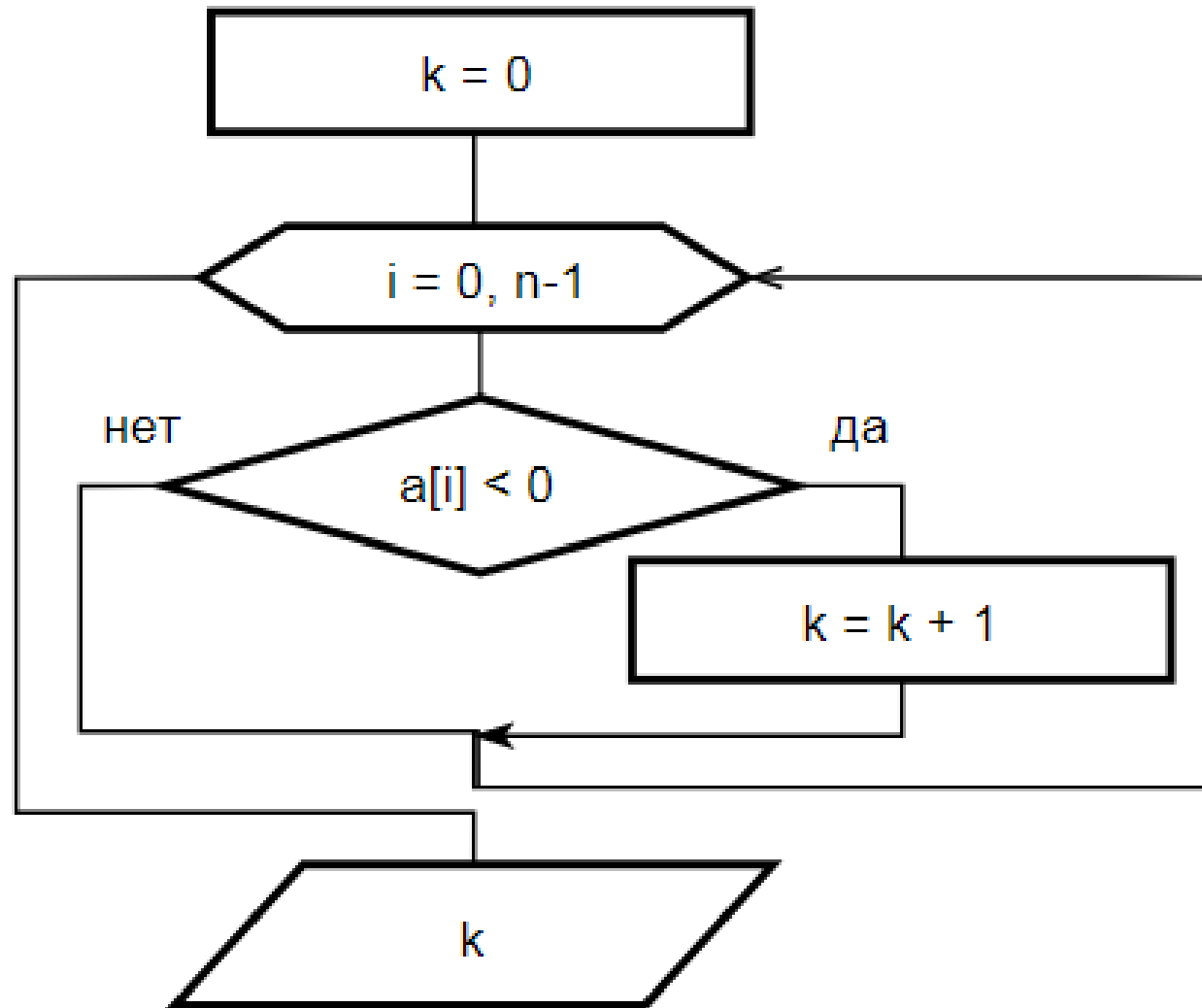
5	-1	0	7	8	2	2	3
0	1	2	...	i	...		n-1

$$s = 0$$

$$p = 1$$

Базовые алгоритмы обработки

5. Количество отрицательных элементов

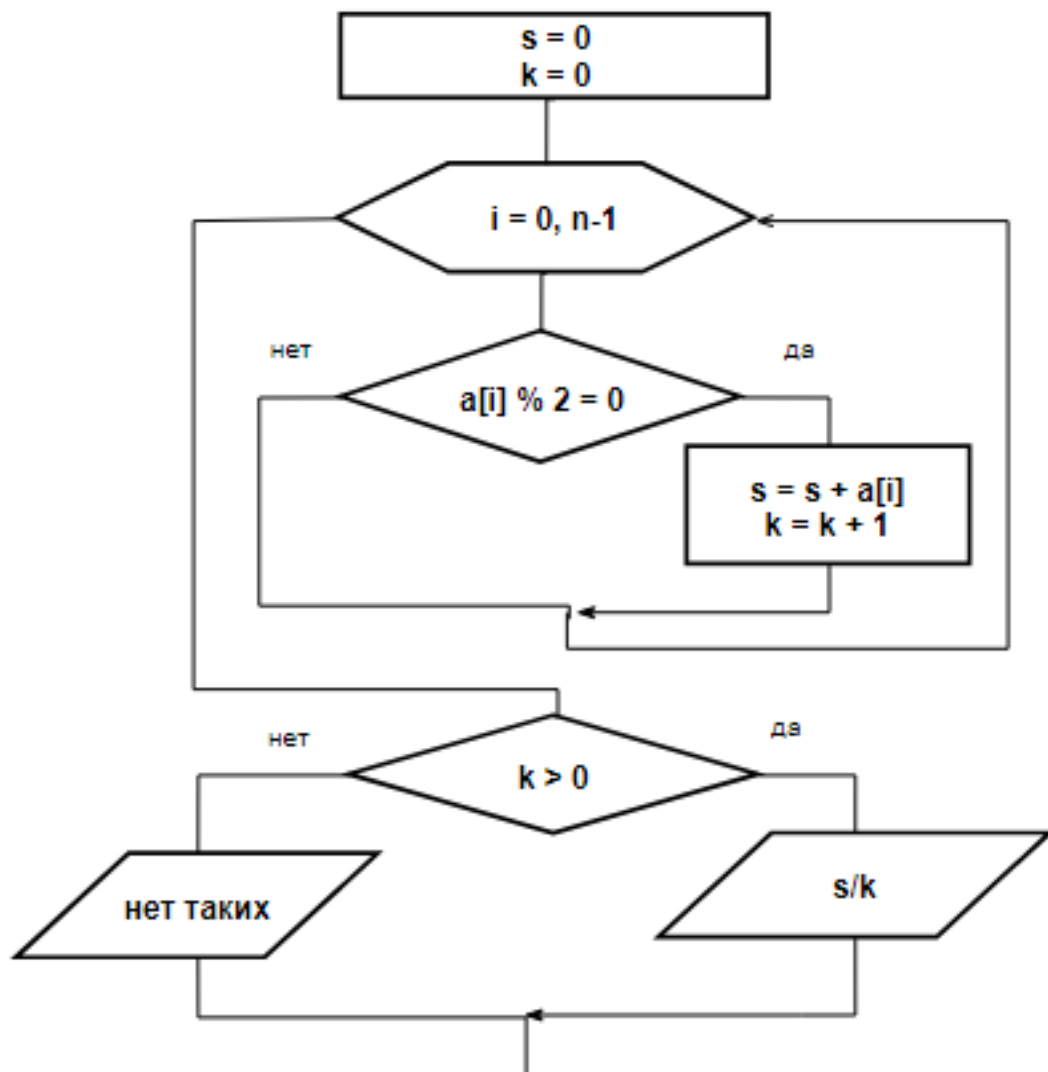


7	-5	9	1	0	-2	4	3	6
0	1	2	...	i	...			n-1

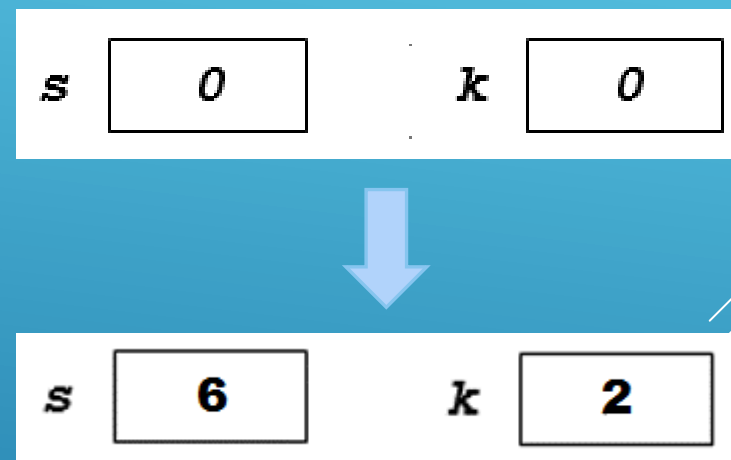
k 0

Базовые алгоритмы обработки

6. Среднее арифметическое четных элементов

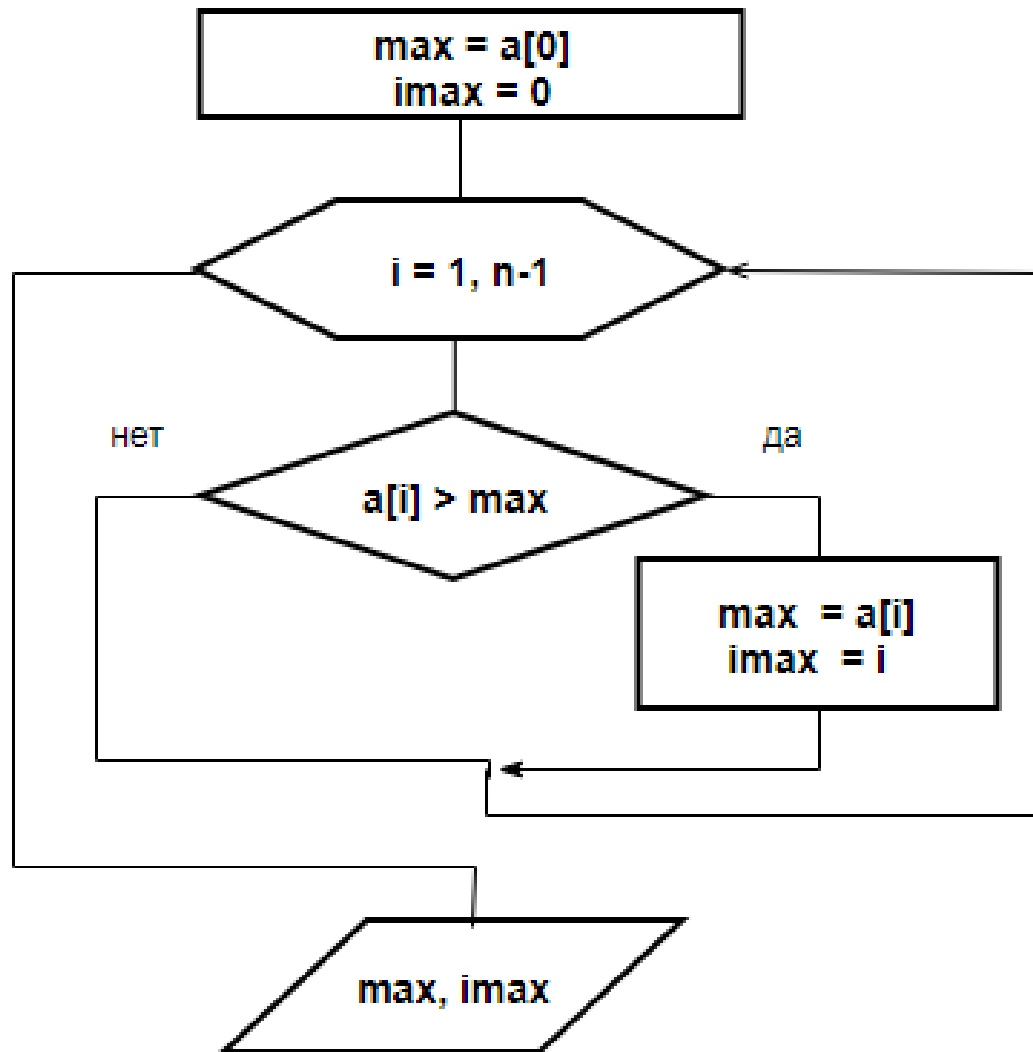


7	-5	9	1	1	-2	4	3	6
0	1	2	...	i	...			n-1

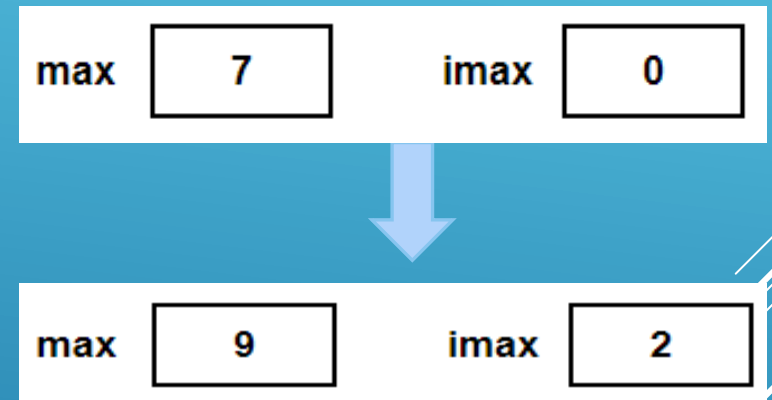


Базовые алгоритмы обработки

7. Максимальный элемент и его номер

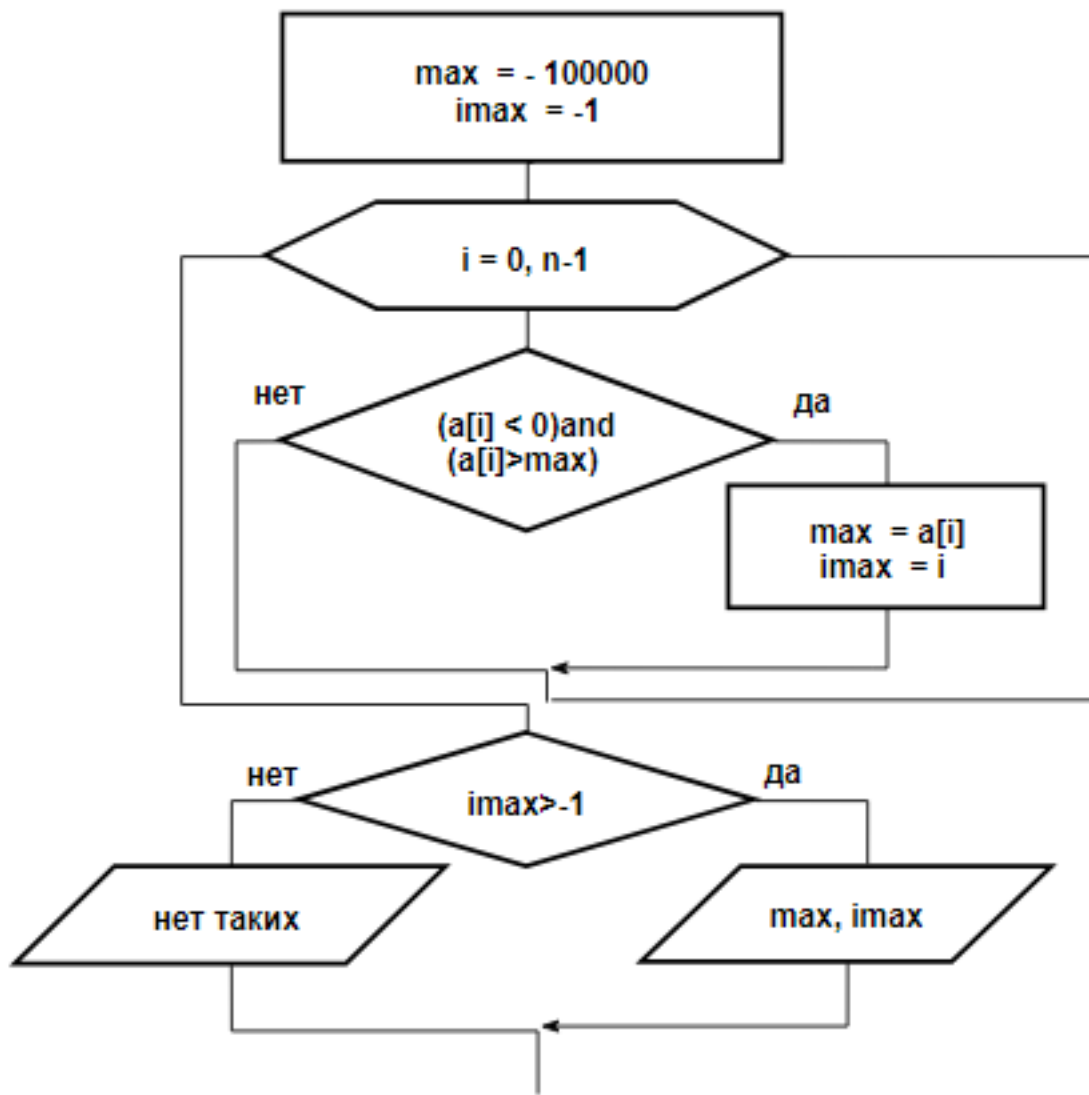


7	-5	9	1	0	-2	4	3	6
0	1	2	...	i	...			n-1

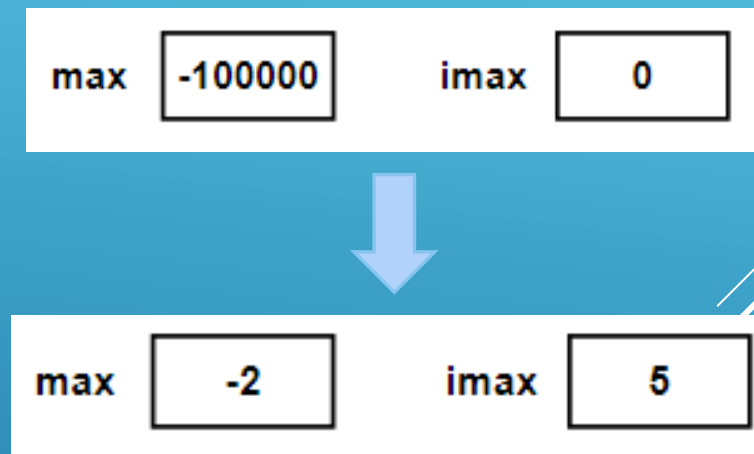


Базовые алгоритмы обработки

8. Максимальный отрицательный элемент и его номер

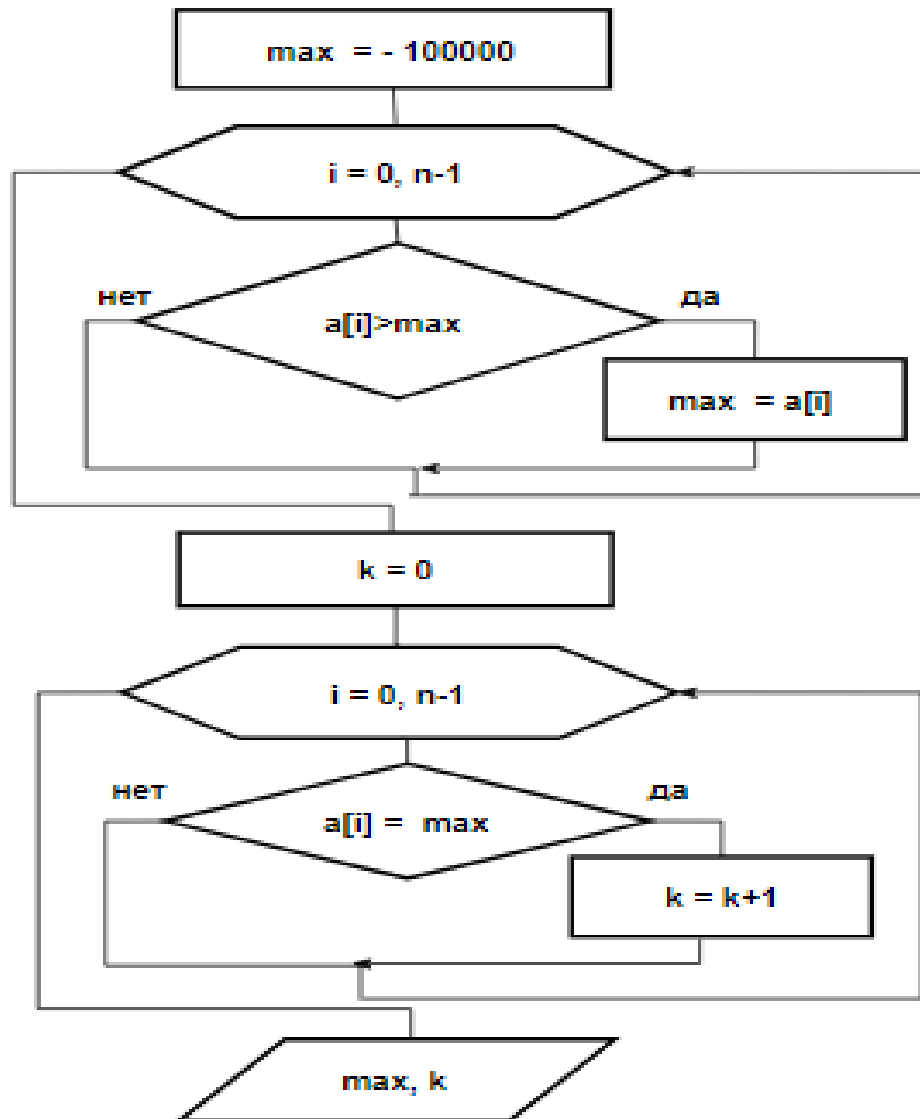


7	-5	9	1	0	-2	4	3	6
0	1	2	...	i	...			n-1



Базовые алгоритмы обработки

9. Количество максимальных элементов



7	-5	9	1	0	-2	9	3	6
0	1	2	...	i	...			n-1

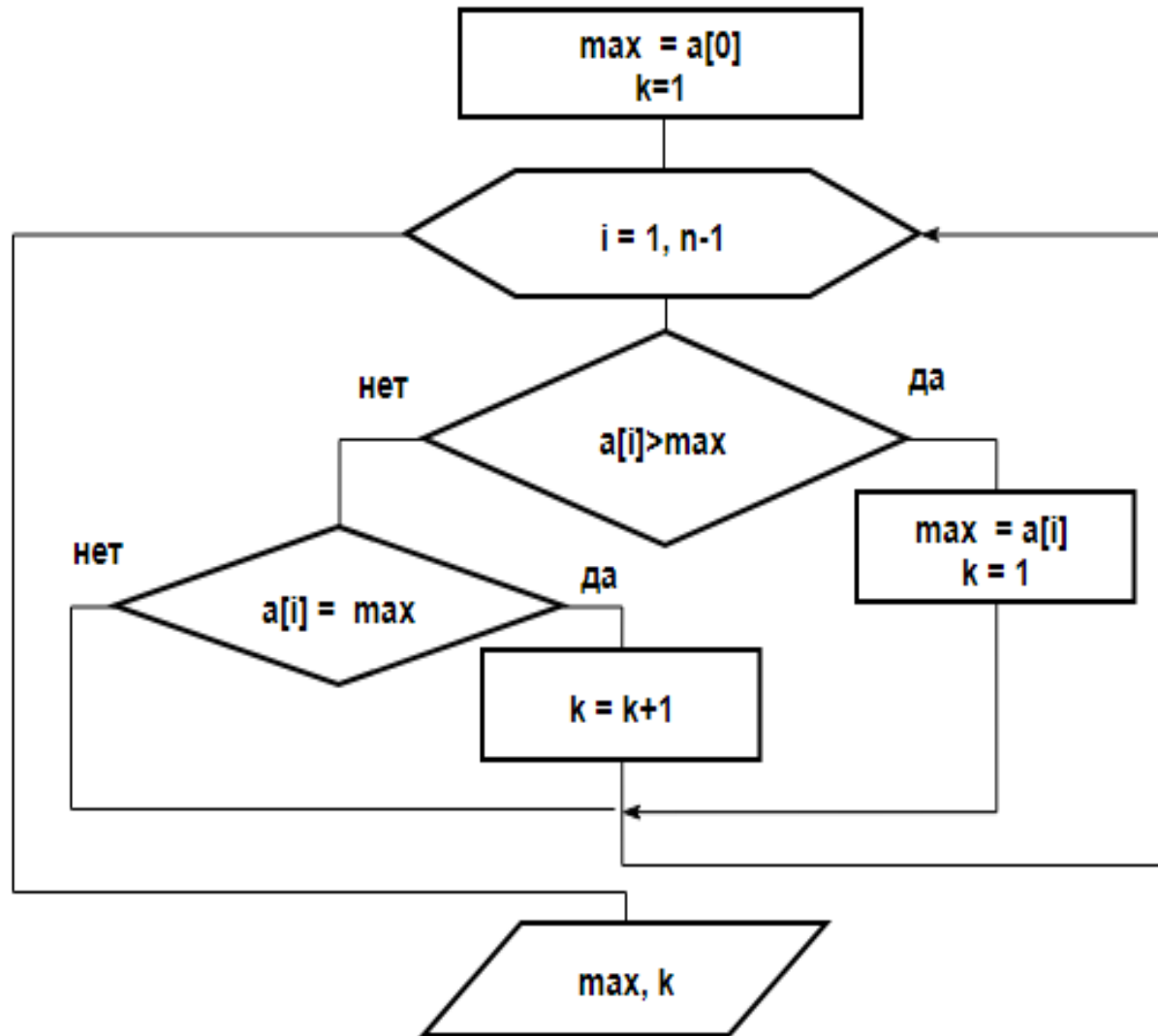
max -100000 k 0



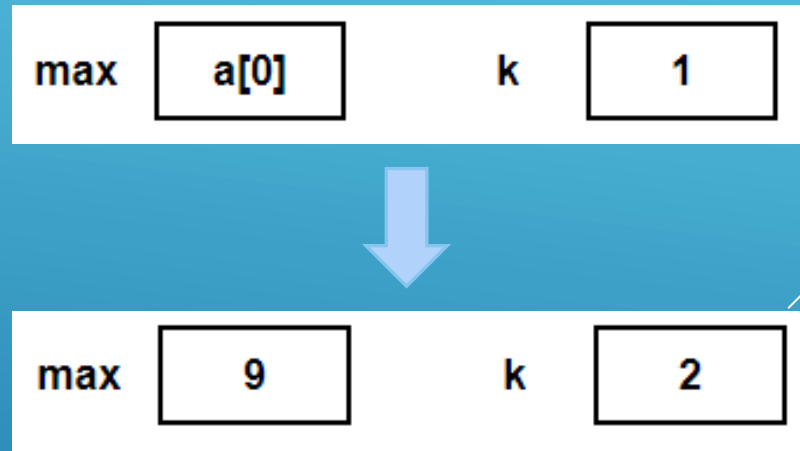
max 9 k 2

Базовые алгоритмы обработки

10. Количество максимальных элементов (за один проход массива)

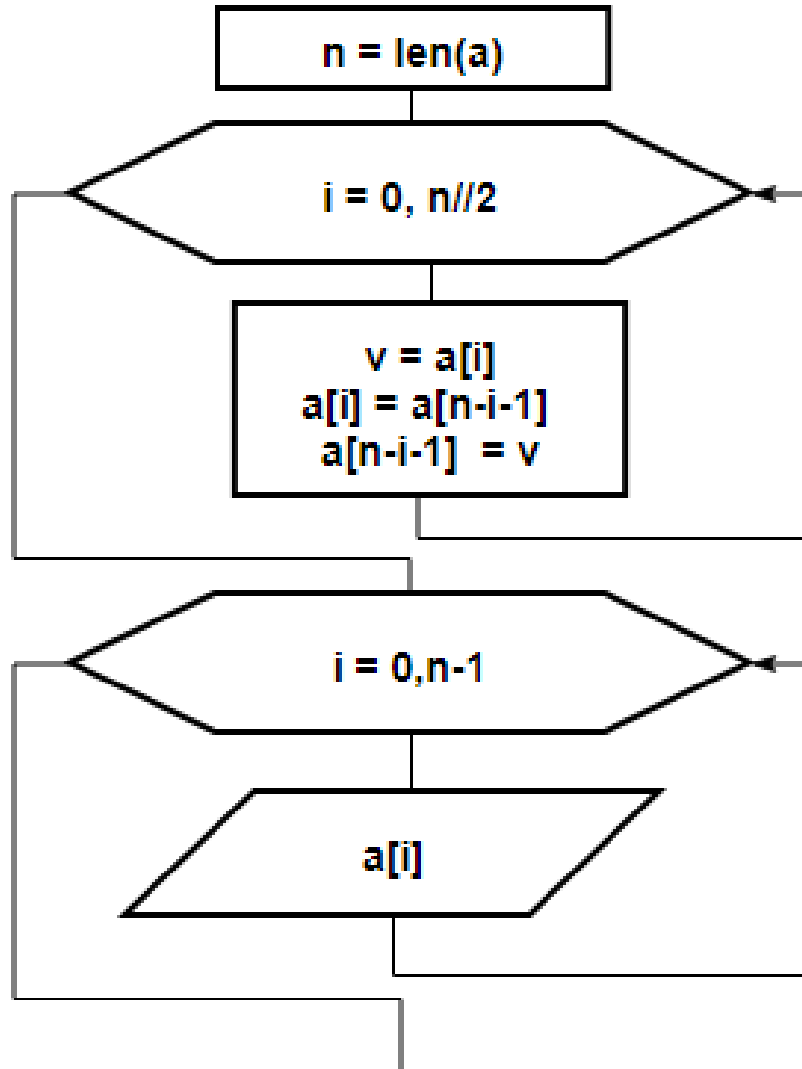


7	-5	9	1	0	-2	9	3	6
0	1	2	...	i	...			n-1



Базовые алгоритмы обработки

11. Переписать массив в обратном порядке



7	-5	9	1	0	-2	4	3	6
0	1	2	...	i	...			n-1

`n = len(a)`

0

`n-1`

1

`n-2`

2

`n-3`

3

`n-4`

`i`

`n-i-1`

`a[i]`

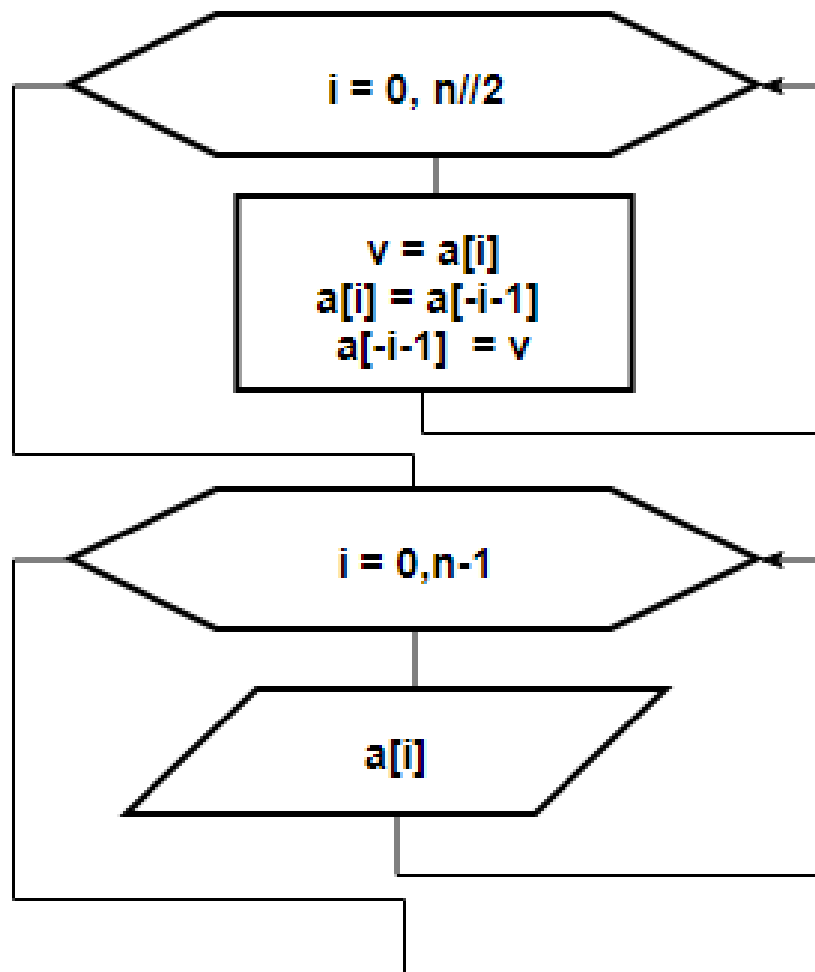
`a[n-i-1]`



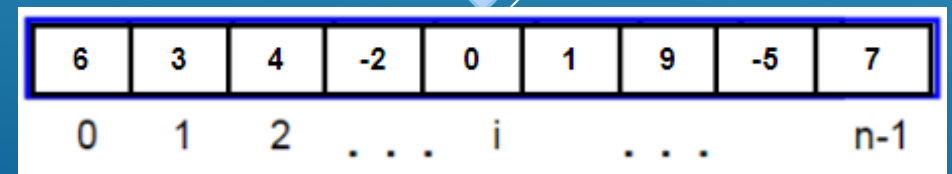
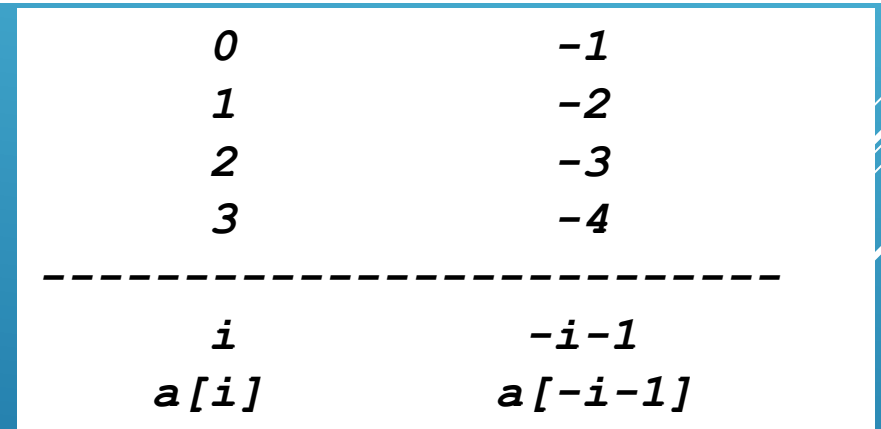
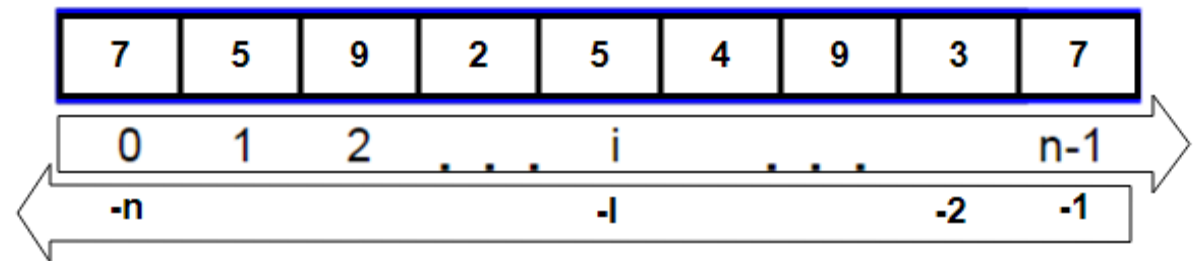
6	3	4	-2	0	1	9	-5	7
0	1	2	...	i	...			n-1

Базовые алгоритмы обработки

11Б. Переписать массив в обратном порядке.

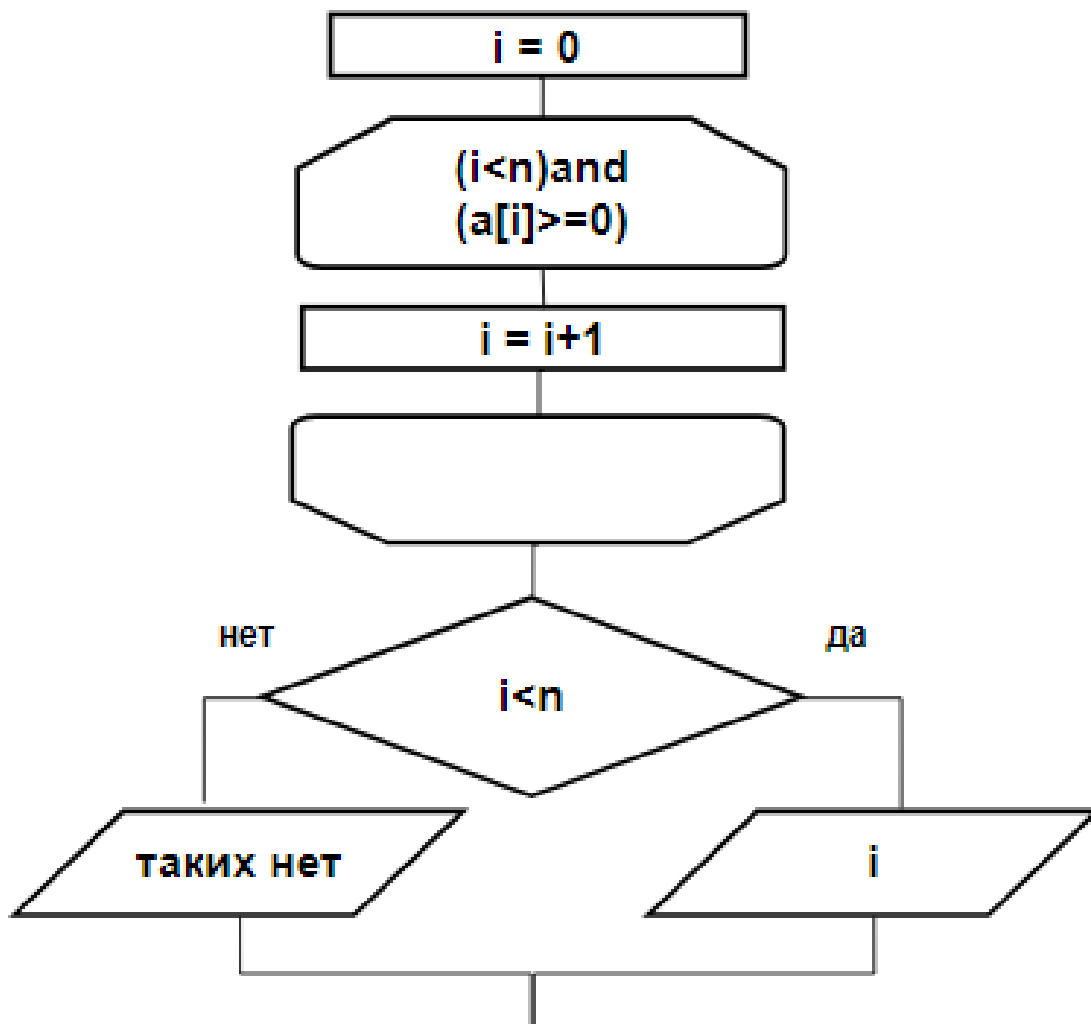


Предыдущий алгоритм можно переписать, используя обратную индексацию Python



Базовые алгоритмы обработки

12. Номер первого отрицательного элемента



7	-5	9	1	0	-2	4	3	6
0	1	2	...	i	...			n-1

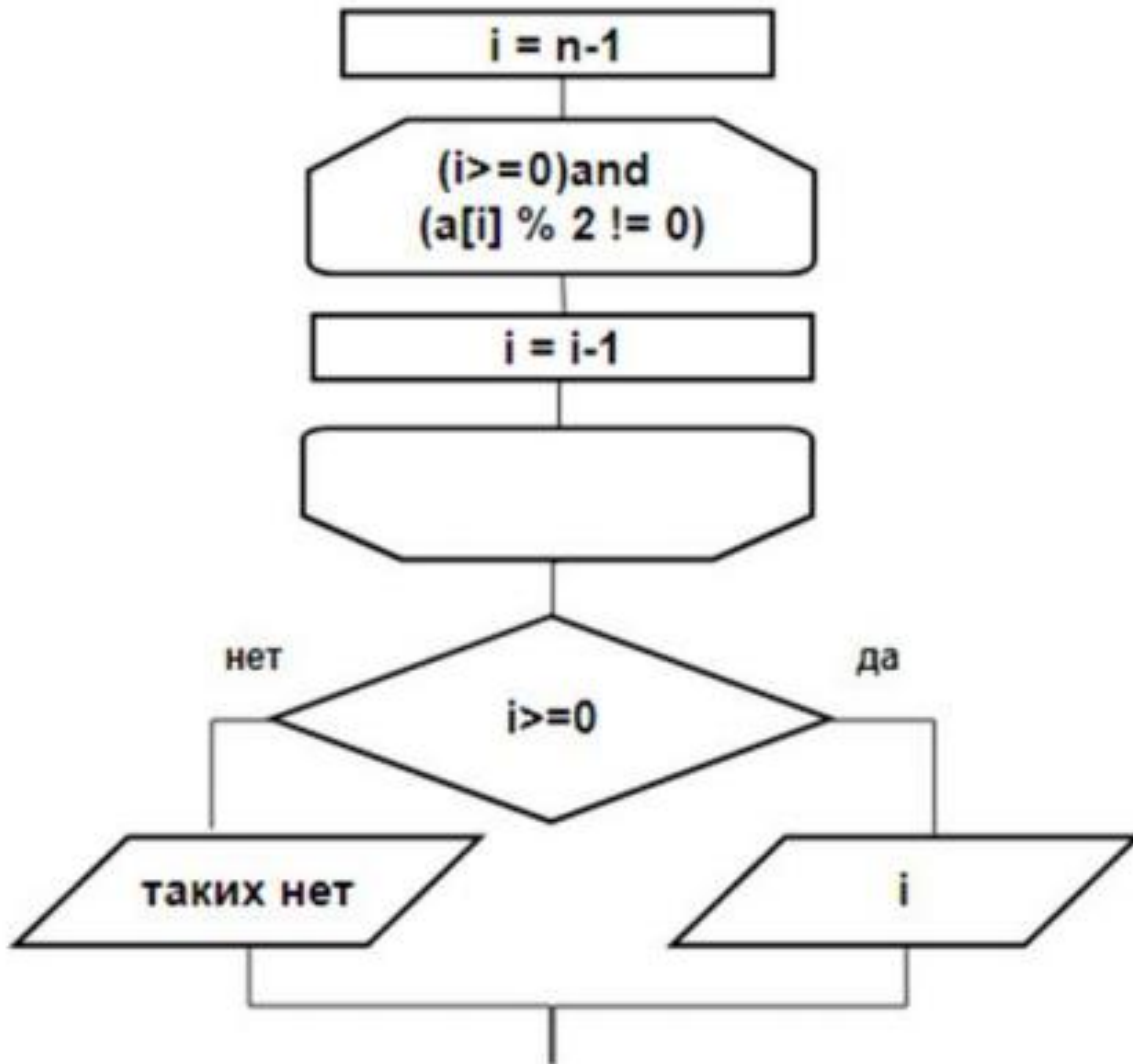
$i = 1$

7	5	9	1	0	2	9	3	6
0	1	2	...	i	...			n-1

$i = 9$
 $i \geq n$
нет таких

Базовые алгоритмы обработки

13. Номер последнего четного элемента



7	-5	9	1	0	-2	4	3	6
0	1	2	...	i	...			n-1

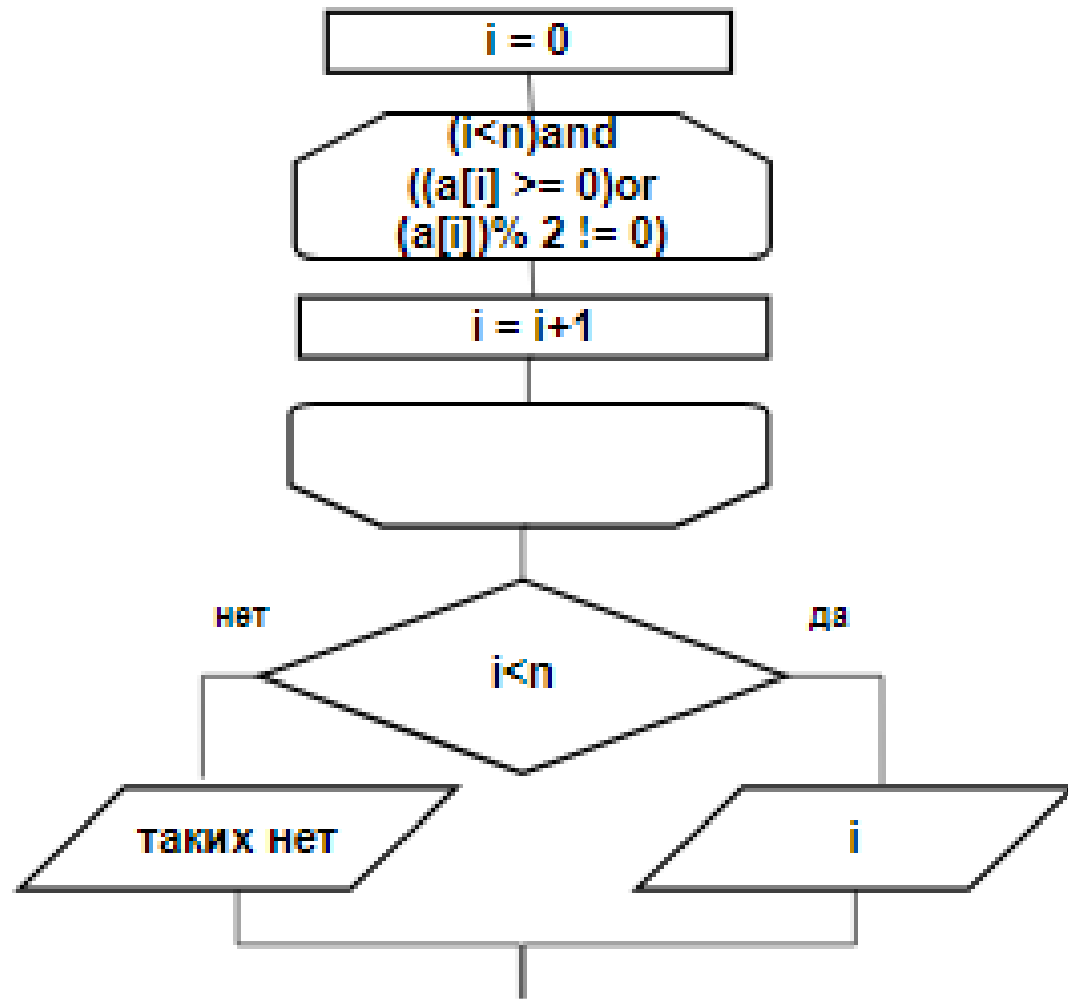
$i = 8$

7	5	9	1	5	3	9	3	7
0	1	2	...	i	...			n-1

$i = -1$
нет таких

Базовые алгоритмы обработки

14. Номер первого отрицательного четного числа



7	-5	9	1	0	-2	4	3	6
0	1	2	...	i	...			n-1

$i = 5$

7	5	9	2	5	4	9	3	7
0	1	2	...	i	...			n-1

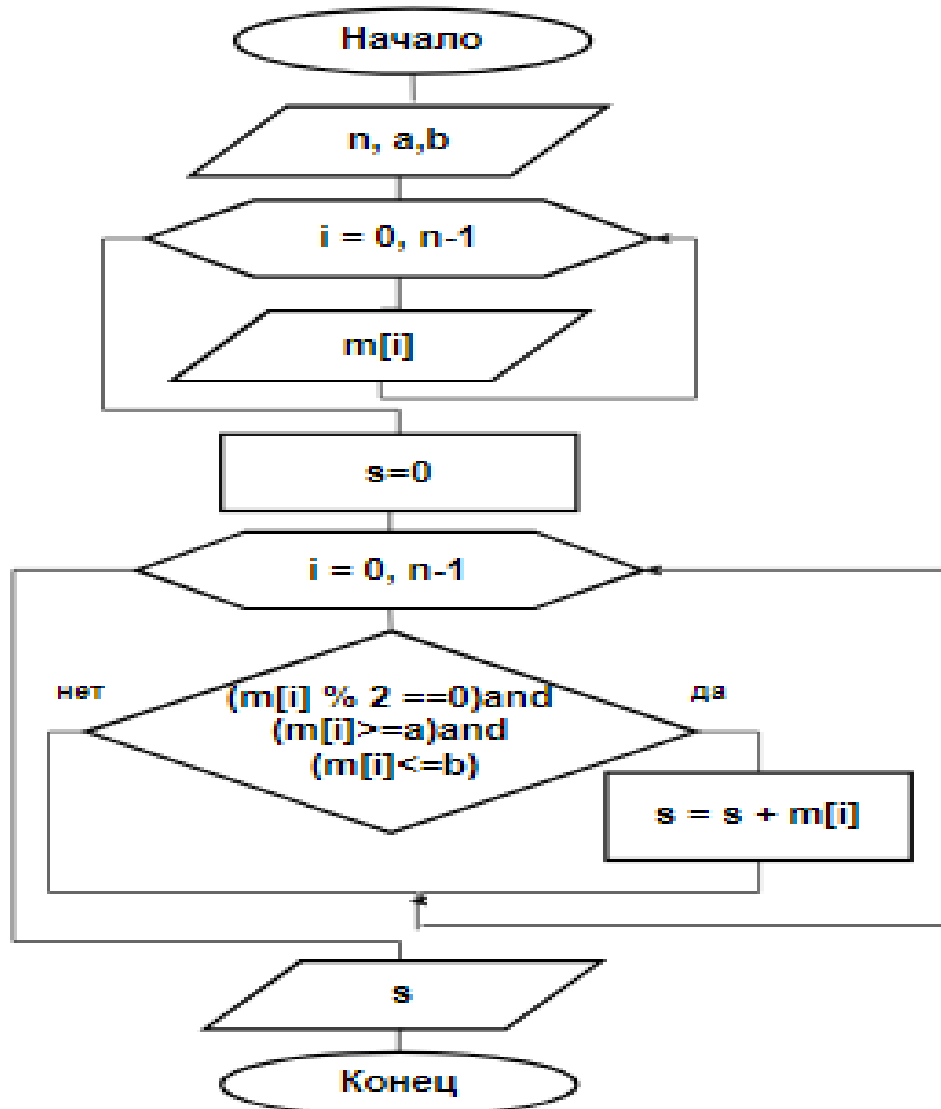
$i = 9$
 $i \geq n$
нет таких

Примеры заданий

№	Задача 1	Задача 2
1	Найти сумму четных (по значению) элементов массива, попадающих в диапазон от a до b . массив m	С помощью цикла WHILE найти индекс первого четного элемента массива
2	Найти произведение всех нечетных (по индексу) элементов массива	С помощью цикла WHILE найти индекс последнего нулевого элемента массива
3	Найти среднее арифметическое отрицательных нечетных (по индексу) элементов массива	С помощью цикла WHILE найти индекс последнего отрицательного нечетного элемента
4	Найти количество нечетных положительных элементов массива	С помощью цикла WHILE найти индекс первого нечетного элемента, кратного 7

Задача 1

Найти сумму четных (по значению) элементов массива, попадающих в диапазон от a до b .



#Ввод длины массива m

```
n = int(input("Введите длину n массива m: "))
```

```
a = int(input("Введите a: ")) #Ввод a
```

```
b = int(input("Введите b: ")) #Ввод b
```

```
print("n=", n, "a=", a, "b=", b) #Вывод n, a, b на консоль
```

m=[] #Объявление массива(списка) m

#Заполнение массива(списка) m

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

```
    m.append(int(input("Введите элемент массива m:")))
```

#Вывод массива m на экран

```
print("Массив:", m)
```

s=0 #Переменная суммы

*#Находим сумму четных (по значению) элементов массива,
попадающих в диапазон от a до b*

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

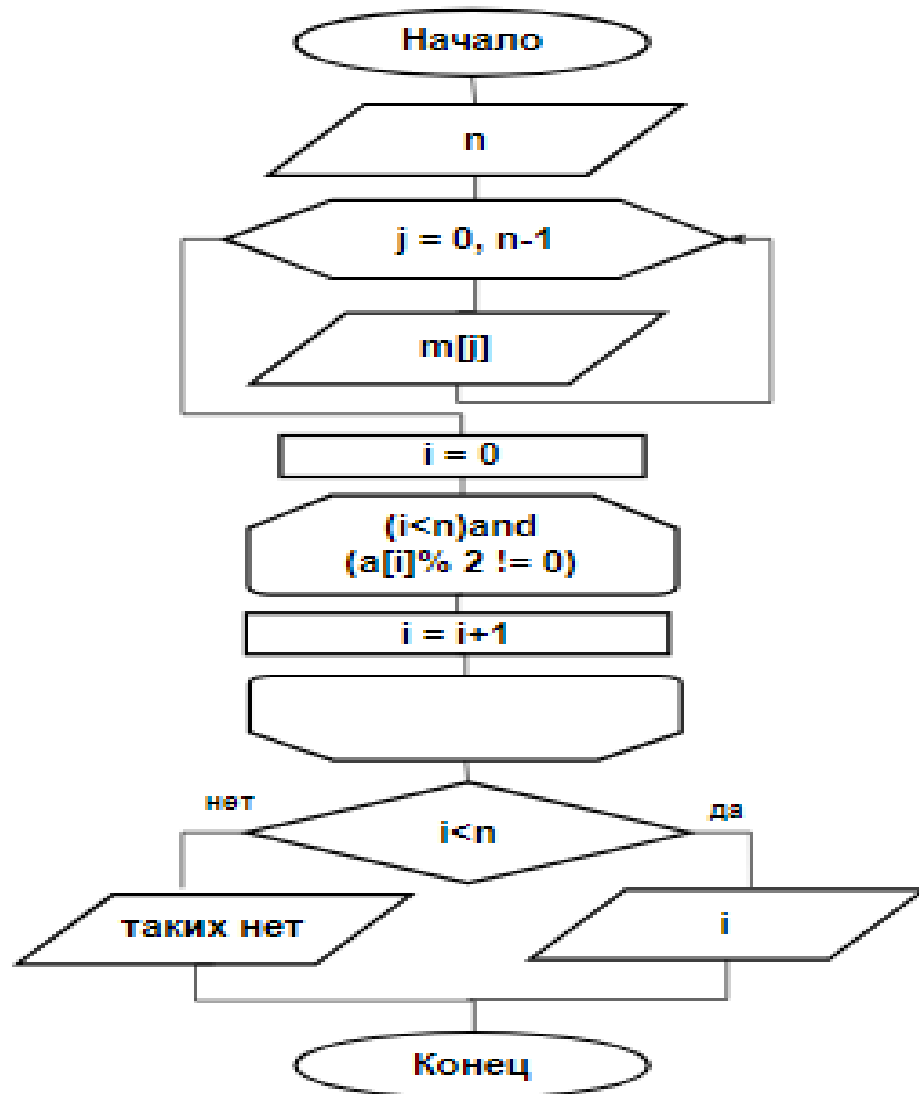
```
    if ((m[i] % 2 == 0) and (m[i] >= a) and (m[i] <= b)):
```

```
        s = s + m[i]
```

```
print("Сумма =", s) #Вывод суммы
```

Задача 2

С помощью цикла WHILE найти индекс первого четного элемента массива.



#Ввод длины массива m

```
n = int(input("Введите длину n массива m: "))
```

#Объявление массива(списка) m

```
m=[]
```

#Заполнение массива(списка) m

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

```
    m.append(int(input("Введите элемент массива m:")))
```

#Вывод массива m на экран

```
print(m)
```

i=0 #индекс массива m

#находим индекс первого четного элемента массива

```
while((i<n)and(m[i]%2!=0)):
```

```
    i=i+1
```

#проверяем значение индекса

```
if(i<n):
```

```
    print("i =",i)
```

```
else:
```

```
    print("таких нет")
```

Содержание контрольной работы № 1

Обработка одномерных массивов

1 Задача (for, while – выбор по целесообразности)

1.1 Схема алгоритма

1.2 Программа обработки

2 Задача (обязательное использование while)

2.1 Схема алгоритма

2.2 Программа обработки

КУРС «ИНФОРМАТИКА»

Часть 3. Памятка по Python

2023 – 2024 УЧЕБНЫЙ ГОД

Особенности Python

- Динамическая типизация. Тип данных переменной определяется исходя из значения, которое ей присвоено. Для объявления переменной не указывается ее тип. В процессе работы программы мы можем изменить тип переменной, присвоив ей значение другого типа

Пример

Объявление переменной и ее инициализация

```
f = 0
```

```
print(f)
```

повторное объявление переменной тоже работает

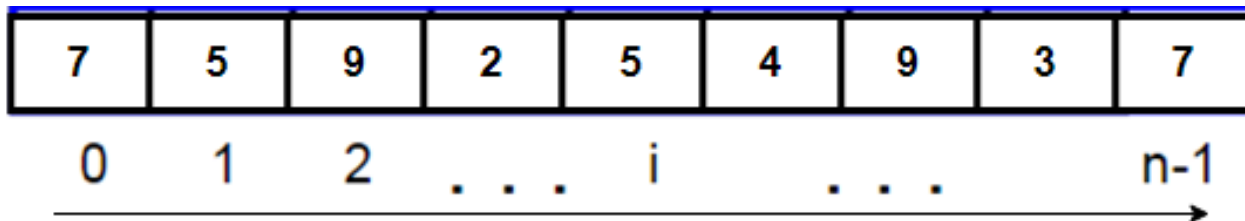
```
f = 'пример'
```

```
print(f)
```

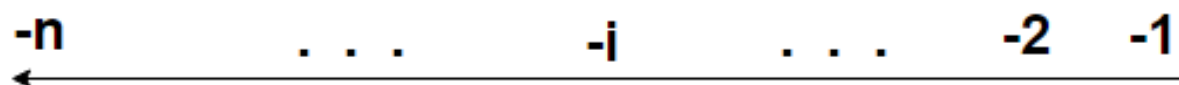
- Однострочный комментарий начинается с символа #

Особенности Python

- Отсутствие явной структуры данных массива. Вместо массивов используются списки. Список содержит набор элементов и поддерживает операции добавления / обновления / удаления / поиска. Список Python допускает элементы разных типов.
- Нумерация элементов массива начинается с 0
- Поддерживаются отрицательные индексы, при этом нумерация идёт с конца



**Прямой порядок
индексации**



**Обратный порядок
индексации**

Операции с числами

Операция	Описание	Пример использования
%	Получение остатка от деления	<pre>print(7 % 2) # Получение остатка от # деления числа 7 на 2. # Результат - 1</pre>
//	Целочисленное деление двух чисел	<pre>print(7 / 2) # 3.5 print(7 // 2) # 3</pre>
**	Возведение в степень	<pre>print(6 ** 2) # Возводим число 6 в # степень 2. Результат - 36</pre>
+=	Присвоение результата сложения	<pre>number = 10 number += 5 print(number) # 15 number -= 3 print(number) # 12 number *= 4 print(number) # 48</pre>
-=	Присвоение результата вычитания	
*=	Присвоение результата умножения	
/=	Присвоение результата от	
//=	деления	
**=	Присвоение	
%=	результата целочисленного	
	деления	
	Присвоение степени числа	
	Присвоение остатка от деления	

Операции сравнения

Операция	Описание	Пример использования
==	Возвращает True, если оба операнда равны. Иначе возвращает False.	# выполняет сложение, если a=0 If (a==0) : a = a+1
!=	Возвращает True, если оба операнда НЕ равны. Иначе возвращает False.	# выполняет сложение, если a не равно 0 If (a!=0) : a = a+1
> (больше чем)	Возвращает True, если первый операнд больше второго.	
< (меньше чем)	Возвращает True, если первый операнд меньше второго.	
>= (больше или равно)	Возвращает True, если первый операнд больше или равен второму.	
<= (меньше или равно)	Возвращает True, если первый операнд меньше или равен второму.	

Логические операторы

Операция	Описание	Пример использования
and	Логический оператор "И". Условие будет истинным если оба операнда истина.	<pre>a = 4 if ((a%2==0) and (a>=0)) : print ("a четное и положительное число")</pre>
or	Логический оператор "ИЛИ". Если хотя бы один из операндов истинный, то и все выражение будет истинным.	<pre>a = 5 if ((a%2==0) or (a>=0)) : print ("a четное или положительное число")</pre>
not	Логический оператор "НЕ". Изменяет логическое значение операнда на противоположное.	<pre>a = 5 if not (a % 3 == 0): print("5 не делится нацело на 3")</pre>

Список функций

Название функции	Описание	Пример использования
<code>print()</code> <code>print(переменная1, переменная2, ..., переменная N)</code>	Вывод заданных объектов на экран. <code>print()</code> без аргументов выводит пустую строку.	# вывод одной переменной print (a) # вывод нескольких переменных print ("a=" , a, "b=", b)
<code>input()</code> <code>input(<строка подсказки>)</code>	Ввод пользовательских данных из консоли. Если в функцию передан необязательный аргумент подсказки, то он записывается в стандартный вывод без завершающей строки. Затем функция читает строку из ввода и преобразует ее в СТРОКУ(str).	# Ввод строки: s = input ()
<code>int()</code>	С помощью функции <code>int()</code> можно конвертировать другой тип данных в целое число.	# Ввод числа: x = int (input("Введите x: "))

Список функций

Название функции	Описание	Пример использования
<code>range(stop)</code> <code>range(start, stop[, step])</code>	Генерирует список чисел, который обычно используется для работы с циклом <code>for</code> . Аргументы функции <code>range</code> должны быть целыми числами. Если аргумент <code>step</code> пропущен, по умолчанию используется 1. Если <code>start</code> аргумент пропущен, по умолчанию используется 0. Цикл выполняется до <code>stop-1</code>	<pre># 5 чисел начиная с 0 for i in range(5): print(i) # диапазон чисел от 1 до 10 с шагом 2 for i in range (1,10,2): print (i)</pre>
<code>append(item)</code>	Добавляет элемент <code>item</code> в конец списка	<pre># Заполнение списка m m = [] for i in range(5): m.append(i) print(m) #m=[0, 1, 2, 3, 4]</pre>
<code>len(<список>)</code>	Функция <code>len()</code> возвращает длину (количество элементов) в списке	<pre># вывод длины массива m print("Длина списка =", len(m))</pre>

Спасибо за внимание

