Практическая работа №2
Разработка программ, реализующих циклические алгоритмы над списками.

Цель работы: закрепить знания принципов работы со структурированным типом данных «список», научиться разрабатывать и отлаживать программы обработки списков

Задание на лабораторную работу:

1. Изучить теоретические сведения.
2. Написать программу в соответствии с вариантом.

Теоретические сведения

Список – это изменяемая последовательность произвольных элементов (допускают возможность добавления, удаления и изменения своих элементов).

Доступ к элементам списка по индексу:

```
>>> spam= ['cat’, 'bat’, 'rat', 'elephant']
>>> spam[0]
'cat’
>>> spam[-1]
'elephant'
>>> spam[-3]
'bat'
>>> 'The ' + spam[-1] + ' is afraid of the ' + spam[-4] + '.'
'The elephant is afraid of the cat.'
```

Вывести элементы списка а можно одной инструкцией print(a), при этом будут выведены квадратные скобки вокруг элементов списка и запятые между элементами списка.

Пример: вывести значения списка в столбец

```
Rainbow = ['Red', 'Orange', 'Yellow', 'Green', 'Blue', 'Violet']
print('Выведем радугу')
```

в цикле меняется индекс элемента i, затем выводится элемент списка с индексом i:	в цикле меняется не индекс элемента, а само значение (например, в цикле for elem in ['red', 'green', 'blue'] переменная elem будет последовательно принимать значения 'red', 'green', 'blue':
`for i in range(len(Rainbow)):` `  print(Rainbow[i])`	`for elem in Rainbow:` `  print(elem)`

Ввод элементов списка

Пример: организовать ввод списка можно так (элементы списка вводятся по одному в строке):

```
a = []                          # заводим пустой список
n = int(input())                # считываем количество элементов в
списке
for i in range(n):
    new_element = int(input())        # считываем очередной элемент
    a.append(new_element)       # добавляем его в список
# последние две строки можно было заменить одной:
# a.append(int(input()))
print(a)
```

Пример: ввод списка с помощью строкового метода split() - возвращает список строк, которые получатся, если исходную строку разрезать на части по пробелам (элементы списка вводятся в одну строку через пробел)

```
S = input()           # ввели строку '1 2 3' (разделенную пробелами)
A = S.split()         # получили список строк A = ['1', '2', '3']
```

Если нужно получить список именно из чисел, то можно элементы списка по одному преобразовать в числа:

`a = input().split()`
`for i in range(len(a)):`
`  a[i] = int(a[i])`
	аналогичная операция с генератором:
	`a = [int(s) for s in input().split()]`

	аналогичная операция с функцией list: `a=list(map(int,input().split()))`

Пример заполнения списка в цикле while:

```
Names = []
while True:
    print ('Enter name Or enter nothing to stop ')
    name  =  input()
    if name ==  '':
        break
    Names = Names + [name]  #конкатенация списков
```

Для создания списков можно использовать ***генераторы***: выражения, позволяющие заполнить список некоторой формулой. Общий вид генератора следующий:
[выражение for переменная in последовательность]
где переменная — идентификатор некоторой переменной,
последовательность — последовательность значений, который принимает данная переменная (это может быть список, строка или объект, полученный при помощи функции range),
выражение — некоторое выражение, как правило, зависящее от использованной в генераторе переменной, которым будут заполнены элементы списка.

Пример: создать список, состоящий из 5 нулей при помощи генератора:

```
a = [0 for i in range(5)]
```

Пример: создать список, заполненный случайными числами от 1 до 9 (используя функцию randrange из модуля random):

```
from random import randrange
n = 10
a = [randrange(1, 10) for i in range(n)]
```

Методы списков:

Метод	Описание
list(s)	Преобразует объект s в список. Если аргумент s уже является списком, эта функция создает новый список, используя операцию поверхностного копирования.
s.append(x)	Добавляет новый элемент x в конец списка s.
s.extend(t)	Добавляет новый список t в конец списка s.
s.count(x)	Определяет количество вхождений x в список s.
s.index(x [,start [,stop]])	Возвращает наименьшее значение индекса i, где s[i] == x. Необязательные значения start и stop определяют индексы начального и конечного элементов диапазона, где выполняется поиск. Если нет искомого элемента возбуждается исключение ValueError.
s.insert(i,x)	Вставляет x в элемент с индексом i.

s.pop([i])	Возвращает i-й элемент и удаляет его из списка. Если индекс i не указан, возвращается последний элемент.
s.remove(x)	Отыскивает в списке s элемент со значением x и удаляет его. Или возбуждает исключение ValueError, если искомый элемент отсутствует в списке.
s.reverse()	Изменяет порядок следования элементов в списке s на обратный.
s.sort([key [, reverse]])	Сортирует элементы списка s. key – это функция, которая вычисляет значение ключа. reverse – признак сортировки в обратном порядке. Аргументы key и reverse всегда должны передаваться как именованные аргументы.

Операции со списками:

x in A	Проверить, содержится ли элемент в списке. Возвращает True или False
x not in A	То же самое, что not(x in A)
min(A)	Наименьший элемент списка
max(A)	Наибольший элемент списка
sum(A)	Сумма элементов списка (для числовых значений)

Пример: поиск элемента в списке

```
Students=['Ivan', 'Sasha','Masha']
if 'Ivan' in Students:
    print('Иван здесь')
if 'Ann' not in Students:
    print('Анны нет')
```

Пример: сортировки списка

Не изменяя начального списка	Изменяя начальный список
`Students=['Ivan', 'Sasha','Masha']` `orderedSpisok=sorted(Students)`	`Students=['Ivan', 'Sasha','Masha']` `Students.sort()`

Оператор сложения (+) выполняет ***конкатенацию списков***:

```
a = [1,2,3] + [4,5]   # Создаст список [1,2,3,4,5]
```

!!!! Нельзя выполнить операцию конкатенации для списка и строки, если предварительно не преобразовать список в строку (str) или строку в список (list):

```
>>> str([1, 2]) + "34"         # '[1, 2]34'
>>> [1, 2] + list("34")        # [1, 2, '3', '4']
```

ВЛОЖЕННЫЙ СПИСОК может располагаться в нескольких строках. Величина отступа не имеет значения; Python знает, что до тех пор, пока не встретится закрывающая квадратная скобка, список еще не закончился.

```
span =   ['apples',
        'oranges',
                  'bananas',
'cats']
```

Пример:

```
a = [[1, 2, 3], [4, 5, 6]] # матрица из двух строк и трех столбцов
print(len (a))             # 2
print(a[0])                # [1, 2, 3]
print(a[1])                # [4, 5, 6]
b = a[0]                   # b – ссылка на список из чисел [1, 2, 3]
print(b)                   # [1, 2, 3]
print(a[0][2])             # 3
```

```
a[0][1] = 7                 # в первую строку второй столбец записать
7
print(a)                    # [[1, 7, 3], [4, 5, 6]]
print(b)                    # [1, 7, 3]
b[2] = 9              # в 3-ий элемент списка (по ссылке b) записать
9
print(a[0])                 # [1, 7, 9]
print(b)                    # [1, 7, 9]
```

Пример: ***вывести двумерный список построчно***, разделяя числа пробелами внутри одной строки

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
```

цикл по индексу:	цикл по значениям:

`for i in range(len(a)):`
`  for j in range(len(a[i])):`
`  print(a[i][j], end=' ')`
`  print()`
	`for row in a:`
	`  for elem in row:`
	`  print(elem, end=' ')`
	`  print()`

Создание вложенных списков

Для создания вложенных списков (двумерных массивов) можно использовать ***вложенные генераторы***, разместив генератор списка, являющегося строкой, внутри генератора для строк.

С помощью генератора можно создать список из n элементов, каждый из которых будет списком, состоящих из m нулей

```
n = 3
m = 4
a = [[0] * m for i in range(n)]            # список n×m, заполненный
нулями
```

Внутренний список также можно создать при помощи, например, такого генератора:

```
 [0 for j in range(m)].
```

Вложив один генератор в другой, получим вложенные генераторы:

```
[[0 for j in range(m)] for i in range(n)]
```

Если число 0 заменить на некоторое выражение, зависящее от i (номер строки) и j (номер столбца), то можно получить список, заполненный по некоторой формуле.

Пример заполнения массива целыми числами по образцу в виде таблицы умножения:

```
0  0  0  0  0  0
0  1  2  3  4  5
0  2  4  6  8 10
0  3  6  9 12 15
0  4  8 12 16 20
```

генератор вложенного списка имеет вид:

```
[[i * j for j in range(m)] for i in range(n)]
```

Ввод двумерного массива

Пусть программа получает на вход двумерный массив в виде n строк, каждая из которых содержит m чисел, разделенных пробелами. Строки могут быть разной длины.

```
n = int(input()) # ввести количество строк массива
a = []
for i in range(n):
    a.append([int(j) for j in input().split()])
```

Или с помощью генератора:

```
n = int(input()) # ввести количество строк массива
a = [[int(j) for j in input().split()] for i in range(n)]
```

Пример: подсчитать сумму всех чисел в списке

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
s = 0
```

```
for row in a:
    for elem in row:
        s += elem
print(s)
```

Пример: дан квадратный массив из n строк и n столбцов. Необходимо элементам, находящимся на главной диагонали, присвоить значение 1, элементам, находящимся выше главной диагонали – значение 0, элементам, находящимся ниже главной диагонали – значение 2.

Сначала заполним главную диагональ, для чего нам понадобится один цикл:

```
for i in range(n):
    a[i][i] = 1
```

Затем заполним значением 0 все элементы выше главной диагонали, для чего нам понадобится в каждой из строк с номером i присвоить значение элементам a[i][j] для j=i+1, ..., n-1.

```
for i in range(n):
    for j in range(i + 1, n):
        a[i][j] = 0
```

Аналогично присваиваем значение 2 элементам a[i][j] для j=0, ..., i-1:

```
for i in range(n):
    for j in range(0, i):
        a[i][j] = 2
```

Можно внешние циклы объединить в один и получить более компактное решение:

```
n = 4
a = [[0] * n for i in range(n)]
for i in range(n):
    for j in range(0, i):
        a[i][j] = 2
    a[i][i] = 1
    for j in range(i + 1, n):
        a[i][j] = 0
for row in a:
    print(' '.join([str(elem) for elem in row]))
```

Копирование списков

К спискам можно применить **два вида копирования**: поверхностное копирование и глубокое копирование. При *поверхностном копировании* создается новый объект, но он будет заполнен ссылками на элементы, которые содержались в оригинале. Например:

```
>>> a = [ 1, 2, [3,4] ]
>>> b = list(a)          # Создание поверхностной копии списка a.
>>> b is a
False
>>> b.append(100)        # Добавление нового элемента в список b.
>>> b
[1, 2, [3, 4], 100]
>>> a                    # Обратите внимание, что список a не
изменился
[1, 2, [3, 4]]
>>> b[2] [0] = -100      # Изменение элемента в списке b
>>> b
[1, 2, [-100, 4], 100]
>>> a                    # Обратите внимание, что изменился и список a
[1, 2, [-100, 4]]
```

В данном случае a и b представляют два независимых списка, но элементы, содержащиеся в них, являются общими. Поэтому изменение одного из элементов в списке a приводит к изменению элемента в списке b.

Например, ниже создается объект списка, связанный с именем L, на это имя также ссылается элемент списка M. Таким образом, изменение списка с помощью имени L приведет к изменению списка, на который ссылается и список M:

```
>>> L = [1,  2,  3]
>>> M = ['X',  L,   'Y' ]          # Встраивает ссылку из L
>>> M
['X',   [1,   2,   3],   'Y']
>>> L[1] = 0                          # Список M также изменяется
>>> M
['X',   [1,   0,   3],   'Y']
```

Если это является нежелательным, вы всегда можете явно создать копию объекта. Для списков всегда можно создать поверхностную копию с помощью операции извлечения среза с незаданными пределами:

```
>>> L = [1,  2,  3]
>>> M = ['X',  L[:],   'Y']          # Встраивается копия L
>>> L[1] = 0                          # Изменяется только L,  но не M
>>> L
[1,   0,   3]
>>> M
['X',   [1,   2,   3],   'Y']
```

При *глубоком копировании* создается новый объект и рекурсивно создаются копии всех объектов, содержащихся в оригинале. В языке Python нет встроенной функции, выполняющей глубокое копирование объектов. Однако в стандартной библиотеке copy существует функция copy.deepcopy():

```
>>> import copy
 >>> a = [1, 2, [3, 4]]
 >>> b = copy.deepcopy(a)
>>> b[2] [0] = -100
>>> b
[1, 2, [-100, 4]]
>>> a               # Обратите внимание, что список a не изменился
[1, 2, [3, 4]]
```

Задания

Задание 1

B1. Дан список. Найти количество положительных элементов и количество отрицательных.

B2. Пользователь вводит список. Определите наличие строки «привет» в списке. Если такая строка в списке присутствует, то вывести ее на экран, повторив 10 раз.

B3. Пользователь вводит список. Подсчитать, сколько раз в списке встречается число 4. Если больше одного раза, то очистить список.

B4. Напишите программу, которая генерирует сто случайных целых чисел и заполняет ими список. Выводит получившийся список на экран по десять элементов в ряд.

B5. Пользователь вводит количество элементов N в списке и диапазон значений элементов [A,B]. Создайте и выведите список из N случайных элементов от A до B.

В6. Заполнить список из N элементов случайными числами. Найти среднее арифметическое значений элементов списка. Вывести на экран значения, которые меньше среднего арифметического.
В7. Напишите программу, которая выводит чётные числа из списка (введенного пользователем) и останавливается, если встречает число 237.
В8. Поменять местами первый элемент (с индексом 0) и минимальный элемент списка. Например, дан список [1,10,5,0], получим [0,10,5,1].
В9. Заполнить список из N элементов случайными числами. Вывести на экран значения списка, которые лежат в диапазоне от A до B.
В10. Пользователь вводит предложение. Написать программу, которая будет считать слова, начинающиеся с буквы "н" и заканчивающиеся на букву "а".
В11. Задан список [‘bmp’,’jpg’,’jpeg’,’gif’,’png’]. Пользователь вводит полное имя файла в виде: имя.расширение Если расширение файла есть в списке, вывести сообщение «Это картинка».
В12. Дан список целых чисел. Создать новый список, заменив отрицательные числа на -100, положительные - на число 100, ноль оставить без изменений.

Задание 2

В1. Пользователь вводит количество строк n и количество столбцов m. Вывести на экран двумерный массив n×m, заполненный случайными числами от 1 до 10.
В2. Задать двумерный массив n×m случайных чисел. Вывести строки от 1 до A (вводит пользователь).
В3. Пользователь вводит количество строк n и количество столбцов m. Вывести на экран двумерный массив n×m, заполненный случайными числами от A до B (вводит пользователь).
В4. В двумерный массив вставить два столбца из нулей: первый перед столбцом с номером k, второй - после него.
В5. Заполнить двумерный массив A*B случайными числами в диапазоне 50-100. Вывести его на экран, найти сумму всех элементов.
В6. Заполнить двумерный массив A*B случайными числами в диапазоне 0-10. Увеличить значения всех элементов на K, введенное пользователем.
В7. Заполнить двумерный массив n×m случайными значениями (0 или 1). Вывести False, если элемент =0 или True, если =1.
В8. Пользователь задает значения массива размерности 3*3. Найти сумму и произведение элементов главной диагонали.
В9. Заполнить двумерный массив A*B случайными числами в диапазоне 1-10. Вывести его на экран, найти произведение всех элементов.
В10. Заполнить двумерный массив A*B случайными числами в диапазоне от -10 до 10. Вывести его на экран, посчитать количество отрицательных элементов.
В11. Количество строк массива задает пользователь. Количество элементов в каждой строке случайно (от 3 до 6). Вывести массив на экран. Посчитать сумму элементов в каждой строке.
В12. Пользователь задает значения массива размерности 3*3. Найти сумму и произведение всех элементов массива.

Задание 3

В1. Задайте двумерный массив размером n×m . Найдите индексы первого вхождения минимального элемента (выведите номер строки и номер столбца).
В2. Задайте двумерный массив размером n×m . Найдите индексы последнего вхождения максимального элемента (выведите номер строки и номер столбца).

В3. Даны два числа n и m. Создайте двумерный массив размером n×m и заполните его символами "-" и "o" в шахматном порядке. В левом верхнем углу должен быть символ "-".

В4. Дан двумерный массив из четного числа столбцов. Столбцы левой половины массива поменять местами со столбцами правой половины.

В5. Дано число n. Создайте массив размером n×n и заполните его по следующему правилу: случайные числа от 1 до 9, все элементы главной диагонали замените на 0, все числа больше 7 замените на отрицательные. Исходный и полученный массивы выведите на экран. Числа в строке разделяйте одним пробелом.

В6. Дан двумерный массив размером n*n. Сформировать одномерный массив из элементов заданного массива, расположенных над побочной диагональю.

В7. Дан двумерный массив и два числа: i и j. Поменяйте в массиве столбцы с номерами i и j местами.

В8. Создайте двумерный массив размером n×n и заполните его случайными двузначными числами. Замените все элементы побочной диагонали на 0. Исходный и полученный массивы выведите на экран. Числа в строке разделяйте одним пробелом.

В9. Дан двумерный массив. Вывести из него все строки с четными номерами.

В10. Дан двумерный массив. Найти минимальную сумму элементов строки.

В11. В квадратном массиве выделим четыре четверти, ограниченные главной и побочной диагоналями (без учета элементов, расположенных на диагоналях). Найти сумму элементов верхней четверти.

В12. В12: Дан двумерный массив и число i. Если элемент массива больше числа i, записать вместо него символ «-».

Задание 4

В1-2: Заполнить матрицу m×n случайными числами. Найти строку с максимальной суммой элементов. Вывести на экран номер строки и саму строку.

В3-4: Заполнить матрицу m×n случайными числами. Найти минимальный элемент в каждой строке и столбце.

В5-6: Заполнить матрицу m×n случайными числами. Найти значение наибольшего четного элемента списка для каждой строки.

В7-8: Определить, имеются ли в двумерном массиве только два одинаковых элемента.

В9-10: Заполнить матрицу m×n случайными числами.
Создайте новый двумерный транспонированный массив. Транспонирование - строки исходного массива становятся столбцами транспонированного, столбцы исходного массива становятся строками транспонированного.

Входные данные	Выходные данные
matrix = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12],]	matrix_new=[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]

В11-12: Заполнить матрицу m×n случайными числами. Найти сумму элементов каждой строки и каждого столбца.
Например, матрица 2×3:
1 0 2 *3*
3 5 2 *10*
4 5 4

Задание * (B1-B4)

Дан список T, в котором содержится температура за каждый день. Необходимо сделать новый список, в котором для каждого дня будет отображено - сколько дней необходимо ждать потепления. Если для конкретного дня нет возможности найти день, когда станет теплей, то в список занести 0 (т.е. программа ходит по списку и ищет следующий элемент, который больше текущего)

```
Пример:
Дано: T = [73, 74, 75, 71, 69, 72, 76, 73]
Результат:
[«1 дн.», «1 дн.», «4 дн.», «2 дн.», «1 дн.», «1 дн.», «0 дн.», «0 дн.»]
```

Задание * (B5-B8)

Всем известно, что со временем клавиатура изнашивается, и клавиши на ней начинают залипать.

При изготовлении клавиатуры изначально для каждой клавиши задается количество нажатий, которое она должна выдерживать. Если знать эти величины для используемой клавиатуры, то для определенной последовательности нажатых клавиш можно определить, какие клавиши в процессе их использования сломаются, а какие — нет.

Требуется написать программу, определяющую, какие клавиши сломаются в процессе заданного варианта эксплуатации клавиатуры.

Входные данные

Первая строка входных данных содержит целое число n ($1 \le n \le 1000$) —количество клавиш на клавиатуре.

Вторая строка содержит n целых чисел —количество нажатий, выдерживаемых i-ой клавишей.

Третья строка содержит целое число k ($1 \le k \le 100000$) — общее количество нажатий клавиш,

последняя строка содержит k целых чисел pj ($1 \le pj \le n$) — последовательность нажатых клавиш.

Выходные данные

Программа должна вывести n строк, содержащих информацию об исправности клавиш. Если i-я клавиша сломалась, то i-ая строка должна содержать слово YES, если же клавиша работоспособна — слово NO.

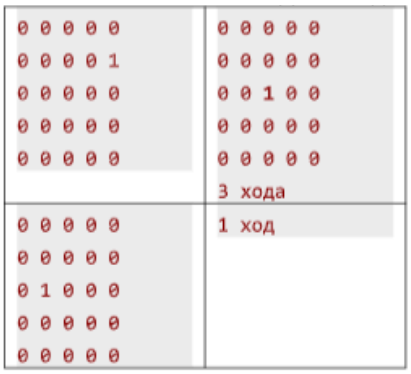
```
Пример
входные данные
5
1 50 3 4 3
16
1 2 3 4 5 1 3 3 4 5 5 5 5 5 4 5
выходные данные
YES
NO
NO
NO
YES
```

Задание * (B9-B13)

Дана матрица 5 × 5, состоящая из 24-х нулей и единственной единицы. Поместить единицу в ее центре (в клетке, которая находится на пересечении третьей строки и третьего столбца), посчитав сколько ходов нужно сделать.

За один ход разрешается применить к матрице одно из двух следующих преобразований:

1. Поменять местами две соседние строки матрицы
2. Поменять местами два соседних столбца матрицы



Контрольные вопросы:

1. Что означает эта операция: a = [ ]?
2. Как можно присвоить значение 'hello' в качестве третьего элемента списка, хранящегося в переменной spam? (Предполагается, что в переменной spam содержится список [2, ‘A’, 6, 8, 10])
3. Каково значение выражения spam [int(‘3’*2) /11], если переменная spam содержит ['a', 'b', 'c','d']?
4. Каково значение выражения spam[-1], если переменная spam содержит ['a', 'b', 'c','d']?
5. Каково значение выражения spam[:2], если переменная spam содержит ['a', 'b', 'c','d']?
6. Каково значение выражения bacon.index ('cat'), если переменная bacon содержит список (3.14, 'cat', 11, 'cat', True)?
7. Как будет выглядеть список, хранящийся в переменной bacon, после вызова: bacon.append (99), если переменная bacon содержит список (3.14, 'cat', 11, 'cat', True)?
8. Как будет выглядеть список, хранящийся в переменной bacon, после вызова: bacon.remove ('cat'), если переменная bacon содержит список (3.14, 'cat', 11, 'cat', True)?