МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам
по курсу

«ИНФОРМАТИКА»

Часть II

2021

ЛАБОРАТОРНАЯ РАБОТА № 2 (Теоретическая часть)
Обработка линейных структур данных

Алгоритм и его программная реализация тесно взаимосвязаны.

• ***Программа*** – упорядоченная последовательность инструкций компьютера (команд) для решения задачи.

• ***Программное обеспечение*** – совокупность программ обработки данных и необходимых для их эксплуатации документов.

• ***Программирование*** – теоретическая и практическая деятельность, связанная с созданием программ.

Программа – результат интеллектуального труда, для которого характерно творчество, поэтому в любой программе присутствует индивидуальность ее разработчика, программа отражает определенную степень искусства программиста. Вместе с тем программирование предполагает и рутинные работы, которые могут и должны иметь строгий регламент выполнения и соответствовать стандартам.

Особенности линейных структур данных в Python

В языке Python отсутствует такая структура, как «массив» (статический массив). Для работы с массивами используются *списки.*

- ***Список*** – это динамическая упорядоченная последовательность объектов (значений, данных), возможно разного типа.

Элементы списка заключены в квадратные скобки [ ] и отделяются друг от друга с помощью запятой.

1. Создание списка через присваивание конкретных значений

Линейный одномерный список

- Пустой список в коде Python:

```
a = []  #[1] Пустой список
```

- Примеры создания списков со значениями:

```
a = [25, 755, -40, 57, -41]   # список целых чисел
a = [1.13, 5.34, 12.63, 4.6, 34.0, 12.8] # список из дробных чисел
a = ["Samara", "Moscow", "Kazan "]   # список из строк
print (a)  # печать списка
```

Разнородные и многомерные списки

- Примеры создания разнородных и многомерных списков со значениями:

```
a = ["Москва", "Иванов", 12, 124]   # смешанный список
a = [[0, 0, 0],[1, 0, 1],[1, 1, 0]]   # список, состоящий из списков
a = ['s', 'p', ['work'], 2] # список из значений и списка
```

[1] # - комментарии в Python

2. Создание списка при помощи функции List()

```
a = []  # Пустой список
a = list ('list')  # 'list' - строка
print(a) # На экране результат['l','i', 's', 't']
```

```
a = []  # Пустой список
a = list ('12345')  # '12345' - строка
print(a) # На экране результат['1', '2', '3', '4', '5']
```

3. Создание списка при помощи функции Split()

```
a = list(map(int, input().split())) # числа вводятся в строку
                                    # через пробел
                                    # a - список
```

```
a = list(map(int, input().split(","))) # числа вводятся в строку
                                       # через запятую
                                       # a - список
```

```
s ="Hello, world"  # s - строка
a =s.split(",")    # a - список
print(a)           # На экране результат ['Hello', ' world']
```

4. Создание списка при помощи модуля random

Для формирования произвольного списка используется модуль генерации случайных чисел random, который необходимо подключить через директиву import в начале программы

```
# Формирование списка случайных чисел
# в диапазоне от b до c (включительно)

import random
. . .
n = int(input("Введите количество элементов списка: "))
a = []
print("Введите диапазон элементов:")
b,c = map(int, input().split())
for i in range(n):
    a.append(random.randint(b,c))
print(a)
print("")
```

Описание функций в Python

Функция в Python - объект, принимающий аргументы и возвращающий значение.

Функция определяется с помощью инструкции def.

Пример описания простейшей функции сложения двух чисел:

```
def summ(a, b):
    return a + b
```

Инструкция **`return`** говорит, какое именно значение нужно вернуть. В нашем случае функция возвращает сумму x и y.

Примеры вызова функции сложения двух чисел:

```
print(summ(10, 78))
```

```
a = int(input())
b = int(input())
print(summ(a, b))
```

Примеры функций поиска параметров в списке

```
// поиск минимального элемента, кратного заданному числу
def findMax(a, x):
    m = -10**10
    for i in range( len(a)):
        if (a[i]%x ==0) and (a[i]<m):
            m = a[i]
    return m

// поиск количества четных элементов
def findCount(a, x):
    k = 0
    for i in range( len(a)):
        if (a[i]%2 ==0):
            k += 1
    return k

// поиск суммы отрицательных элементов
def findSumNeg(a):
    s = 0
    for i in range( len(a)):
        if (a[i] <0):
            s += a[i]
    return s
```

```
// поиск среднего арифметического нечетных положительных элементов
def findAvg(a):
    s = 0
    k = 0
    for i in range( len(a)):
        if (a[i]%2 !=0) and (a[i] > 0):
            s += a[i]
            k += 1
    if k>0:
        return s/k
    else:
        return -1

// поиск последнего положительного элемента  с использованием while
def lastEl(a):
    i = len(a) - 1
    while (i>-1) and (a[i]<= 0):
        i -=1
    return i
ind = lastEl(a)
if ind >-1:
    print("Индекс последнего положительного элемента = ", ind)
else:
    print("В списке отсутствуют положительные элементы ")
```

Примеры функций сортировки списка в Python

```
// сортировка выбором  линейного списка
def sortChoice(a):
    for j in range(len(a)-1):
        min_ = a[j]
        imin=j
        for i in range(j+1,len(a)):
            if a[i]<min_:
                min_ = a[i]
                imin = i
        a[imin] = a[j]
        a[j] = min_
    return a

// сортировка вставкой  линейного списка
def sortInsert(a):
    for i in range(1,len(a)):
        v = a[i]
        j = i-1
        while(j>=0)  and (v<a[j]):
            a[j+1] = a[j]
            j = j-1
        a[j+1] = v
    return a
```

ЛАБОРАТОРНАЯ РАБОТА № 2 (ЗАДАНИЯ)

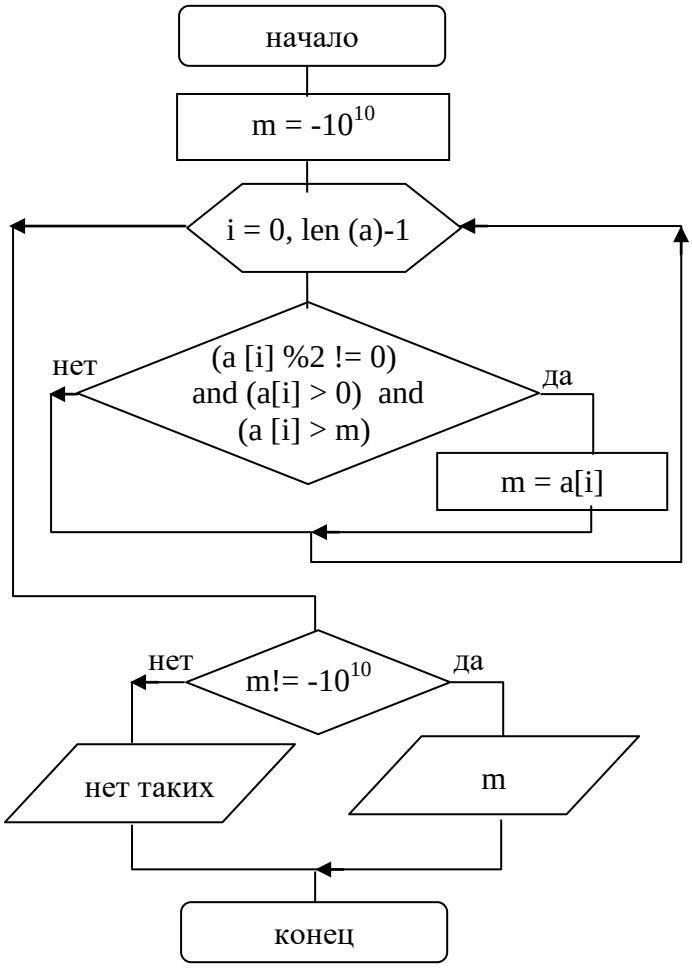
1 часть (В тетради для лабораторных и контрольных работ)

Задание 1.1 Составить алгоритм поиска указанного параметра в списке (без ввода элементов списка).

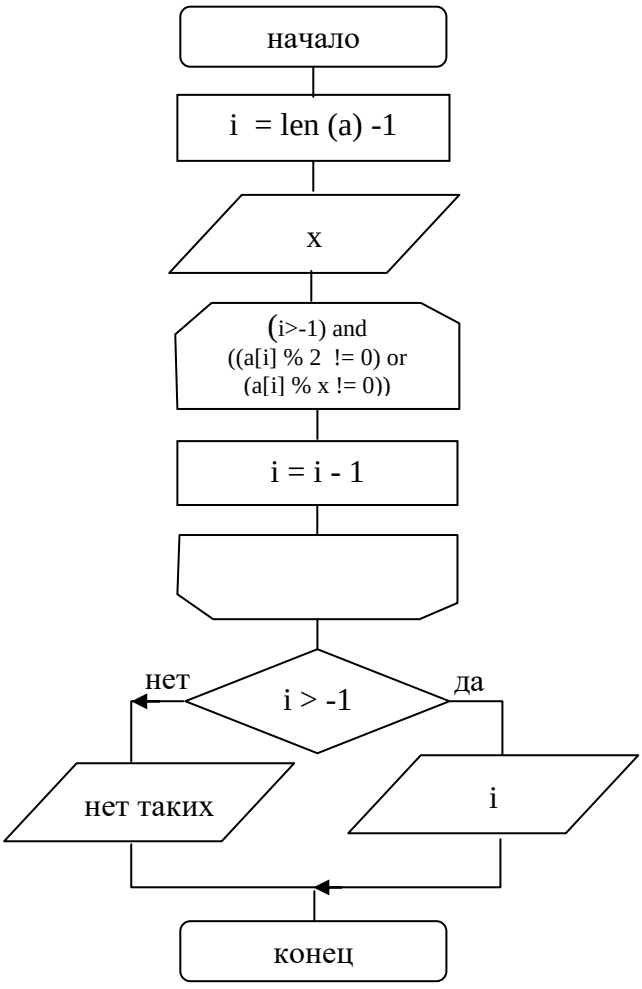
Задание 1.2 Составить алгоритм поиска указанного параметра с использованием цикла While (без ввода элементов списка).

Пример 1 выполнения 1 части лабораторной № 2 (в тетради):

1.1 В списке целочисленных элементов найти максимальный нечетный положительный элемент



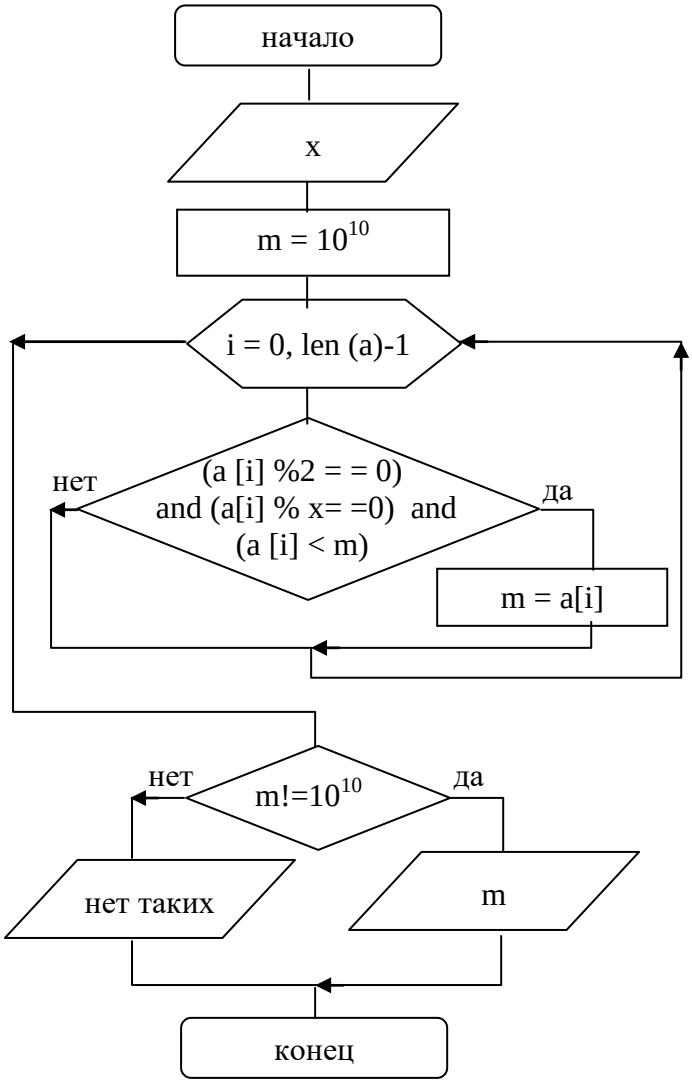
1.2 С использованием цикла while найти в списке индекс последнего четного элемента, кратного заданному числу

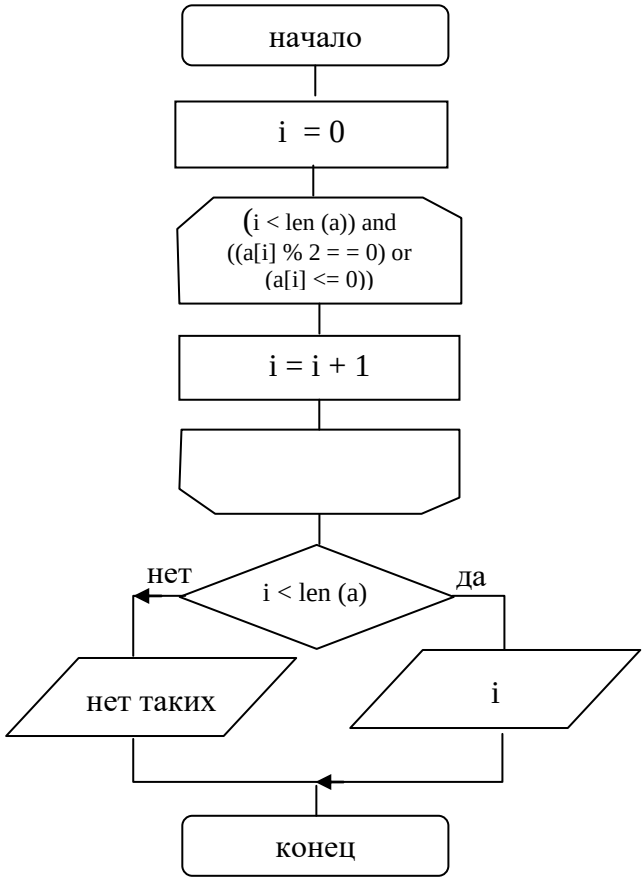


Пример 2 выполнения 1 части лабораторной № 2 (в тетради):

1.1 В списке целочисленных элементов найти минимальный четный элемент, кратный заданному числу

1.2 С использованием цикла while найти в списке индекс первого нечетного положительного элемента





2 часть (Программа) Используя функции с параметрами, составить на Python программу обработки списка целых чисел

Задание 2.1 Поиск параметра.

Задание 2.2 Поиск параметра с использованием *while.*

Задание 2.3 Сортировка списка заданным способом.

В программе должны быть реализованы следующие функции:

1. Выбор пользователем способа заполнения списка.
   1.1. Ввод случайных чисел в заданном диапазоне (количество элементов и диапазон значений элементов задает пользователь) и выдача сформированного списка на экран.
   1.2. Ввод элементов списка пользователем с клавиатуры в одну строку.
2. Поиск указанного параметра (реализация через функцию).
3. Поиск указанного параметра с использованием цикла While (реализация через функцию).
4. Сортировка списка по указанном параметру (реализация через функцию).
5. Вывод результатов на экран с сообщениями для пользователя.

Требования по оформлению программы[2]

1. Программа на языке Python должна иметь следующую структуру:

```
# РАЗДЕЛ ФУНКЦИЙ (все функции вынести в начало файла с программой)
# Функция поиска заданного параметра
def <имя функции 1> [(<список параметров >)]:
     < тело функции >
     return <результат>

# Функция поиска индекса
def <имя функции 2> [(<список параметров >)]:
     < тело функции >
     < поиск реализован через while>
     return <результат>

# Функция сортировки списка
def <имя функции 3> [(<список параметров >)]:
     < тело функции >
     return <результат>

# ВЫВОД ИНФОРМАЦИИ ПО ЛАБОРАТОРНОЙ РАБОТЕ НА ЭКРАН ВЫПОЛНЕНИЯ ПРОГРАММЫ:
     Лабораторная работа № 2
     № варианта, группа, автор (фамилия и имя полностью)
     1. В списке целочисленных элементов найти (далее по своему
        варианту)
     2. С использованием цикла while найти в списке (далее по своему
        варианту)
     3. Отсортировать список (далее по своему варианту)


# ВВОД СПОСОБА ЗАПОЛНЕНИЯ СПИСКА
     1 – ввод элементов списка в одну строку через пробел
     2 – автоматическая генерация списка из n случайных элементов в
         заданном пользователем диапазоне

# ПОИСК ЗАДАННОГО ПАРАМЕТРА (Задание 1)
     вызов функции поиска параметра, выдача результата с
     комментариями для пользователя

# ПОИСК ЗАДАННОГО ИНДЕКСА (Задание 2)
     вызов функции поиска индекса ЧЕРЕЗ WHILE, выдача результата с
     комментариями для пользователя

# СОРТИРОВКА СПИСКА (Задание 3)
     вызов функции сортировки, выдача исходного и отсортированного
     списка друг под другом с комментариями для пользователя
```

[2] Дополнительные требования приведены в файле «Требования к лабораторной работе № 2_2021.doc»

2. Вариант 1 вида работы программы

```
Лабораторная работа № 2
Вариант № 1. Выполнил студент группы 6101-090301D Иванов П.С.
Задание:
1. В списке целочисленных элементов найти максимальный
   нечетный двузначный элемент
2. С использованием цикла while найти в списке индекс
   последнего четного элемента, кратного заданному числу
3. Отсортировать список (без использования стандартных
   функций сортировки) по возрастанию (сортировка выбором)

Введите способ заполнения списка:
1 - ввод элементов списка в одну строку через пробел:
любое число - автоматическое формирование списка из n элементов:
1

Введите в строку элементы списка:
5 4 11 6 24 73 8 19

Максимальный нечетный двузначный элемент =  73

Введите число, кратность которому нужно проверить: 6
Индекс последнего четного элемента, кратного заданному числу =  4

Исходный список:
[5, 4, 11, 6, 24, 73, 8, 19]
Список после сортировки выбором:
[4, 5, 6, 8, 11, 19, 24, 73]
```

2. Вариант 2 вида работы программы

```
Лабораторная работа № 2
Вариант № 1. Выполнил студент группы 6101-090301D Иванов П.С.
Задание:
1. В списке целочисленных элементов найти максимальный
   нечетный двузначный элемент
2. С использованием цикла while найти в списке индекс
   последнего четного элемента, кратного заданному числу
3. Отсортировать список (без использования стандартных
   функций сортировки) по возрастанию (сортировка выбором)

Введите способ заполнения списка:
1 - ввод элементов списка в одну строку через пробел:
любое число - автоматическое формирование списка из n элементов:
2

Введите количество элементов списка: 8
Введите диапазон элементов:
-10 100
[2, 4, 32, 95, 18, 3, 36, 83]

Максимальный нечетный двузначный элемент =  95

Введите число, кратность которому нужно проверить: 7
В списке отсутствуют четные элементы, кратные заданному числу

Исходный список:
[2, 4, 32, 95, 18, 3, 36, 83]
Список после сортировки выбором:
[2, 3, 4, 18, 32, 36, 83, 95]
```

Лабораторная работа № 2

Задания по вариантам

№ вар.	1. В списке целочисленных элементов найти …	2. С использованием цикла while найти в списке …	3. Отсортировать список (без использования стандартных функций сортировки) …
1.	Максимальный четный элемент	индекс последнего нечетного элемента, некратного первому элементу списка	по возрастанию младших цифр элементов списка (сортировка Шелла)
2.	Минимальный четный отрицательный элемент	индекс первого положительного четного элемента	по возрастанию (сортировка вставкой)
3.	Минимальный элемент, некратный заданному числу	индекс первого нечетного ненулевого элемента	по убыванию (сортировка выбором)
4.	Максимальный четный положительный элемент	индекс последнего четного элемента, некратного заданному числу	по убыванию младших цифр элементов списка (сортировка выбором)
5.	Максимальный отрицательный элемент, некратный заданному числу	индекс последнего положительного нечетного элемента	по возрастанию (быстрая сортировка)
6.	Максимальный ненулевой элемент	индекс первого отрицательного четного элемента	по убыванию младших цифр элементов списка (сортировка Шелла)
7.	Максимальный элемент, некратный заданному числу	индекс последнего отрицательного нечетного элемента	по убыванию младших цифр элементов списка (сортировка вставкой)
8.	Максимальный нечетный элемент	индекс первого двузначного элемента, кратного заданному числу	по убыванию старших цифр элементов списка (быстрая сортировка)
9.	Минимальный положительный элемент, некратный заданному числу	индекс первого отрицательного нечетного элемента	по возрастанию старших цифр элементов списка (сортировка выбором)
10.	Минимальный элемент, кратный заданному числу	индекс последнего ненулевого элемента	по убыванию (быстрая сортировка)
11.	Минимальный четный положительный элемент	индекс последнего нечетного элемента, некратного заданному числу	по возрастанию старших цифр элементов списка (сортировка Шелла)
12.	Максимальный элемент, кратный заданному числу	индекс первого четного ненулевого элемента	по убыванию старших цифр элементов списка (сортировка вставкой)
13.	Максимальный четный ненулевой элемент	индекс первого нечетного элемента, кратного заданному числу	по возрастанию старших цифр элементов списка (быстрая сортировка)
14.	Минимальный ненулевой элемент	индекс последнего положительного четного элемента	по убыванию старших цифр элементов списка (сортировка выбором)
15.	Минимальный четный элемент	индекс последнего двузначного элемента, некратного заданному числу	по возрастанию младших цифр элементов списка (быстрая сортировка)
16.	Максимальный нечетный отрицательный элемент	индекс первого четного элемента, кратного заданному числу	по возрастанию (сортировка Шелла)
17.	Минимальный положительный элемент, кратный заданному числу	индекс первого положительного нечетного элемента	по возрастанию младших цифр элементов списка (сортировка вставкой)
18.	Минимальный нечетный	индекс последнего нечетного	по возрастанию старших цифр

	отрицательный элемент	элемента, кратного заданному числу	элементов списка (сортировка вставкой)
19.	Максимальный положительный элемент, кратный заданному числу	индекс последнего четного ненулевого элемента	по возрастанию (сортировка выбором)
20.	Максимальный нечетный элемент	индекс последнего отрицательного четного элемента	по убыванию младших цифр элементов списка (быстрая сортировка)
21.	Минимальный отрицательный элемент, некратный заданному числу	индекс первого нулевого элемента	по убыванию (сортировка Шелла)
22.	Минимальный нечетный элемент	индекс последнего двузначного элемента, кратного заданному числу	по убыванию (сортировка вставкой)
23.	Максимальный четный отрицательный элемент	индекс первого двузначного элемента, некратного заданному числу	по возрастанию младших цифр элементов списка (сортировка выбором)
24.	Максимальный положительный элемент, некратный заданному числу	индекс последнего нечетного ненулевого элемента	по убыванию старших цифр элементов списка (сортировка Шелла)
25.	Минимальный четный ненулевой элемент	индекс первого четного элемента, некратного заданному числу	по возрастанию младших цифр элементов списка (шейкерная сортировка)
26.	Максимальный нечетный положительный элемент	индекс первого нечетного элемента, некратного последнему элементу списка	по возрастанию (шейкерная сортировка)
27.	Минимальный отрицательный элемент, кратный заданному числу	индекс последнего нулевого элемента	по убыванию (шейкерная сортировка)
28.	Минимальный нечетный положительный элемент	индекс первого нечетного элемента, некратного заданному числу	по убыванию младших цифр элементов списка (шейкерная сортировка)
29.	Максимальный отрицательный элемент, кратный заданному числу	индекс первого ненулевого элемента	по возрастанию старших цифр элементов списка (шейкерная сортировка)
30.	Максимальный нечетный двузначный элемент	индекс последнего четного элемента, кратного заданному числу	по убыванию старших цифр элементов списка (шейкерная сортировка)