**Лабораторная работа №4**

**Студента группы ПИ-211**

**Зубкова Ильи Леонидовича**

Выполнение: \_\_\_\_\_\_\_\_\_ Защита: \_\_\_\_\_\_\_\_\_\_\_

**Функции**

**Цель работы:** приобрести навыки при разработке функций.

**Вариант №12**

Дан целочисленный массив X размера N. Назовём серией группу подряд идущих одинаковых элементов, а длиной серии – количество этих элементов. Преобразовать массив таким образом, чтобы количество элементов в каждой серии увеличилось на длину самой короткой серии элементов.

**Задания к лабораторной работе**

1. Разбить задачу на подзадачи, разработать алгоритмы решения каждой

из подзадач, описать их в виде блок-схемы.

2. Реализовать два варианта решения задачи: с использованием

стандартных методов и операций работы со списками (если они

существуют), а также с собственной реализацией этих методов и

операций.

3. Набрать текст программы и отладить её в среде разработки PyCharm.

4. Оформить отчет о проделанной работе.

**Выполнение лабораторной работы с собственной реализацией методов**

1. Задачу можно разделить на несколько подзадач: поиск минимальной последовательности и подсчет ее длины, увеличение всех последовательностей на длину минимальной последовательности

Поиск минимальной последовательности и ее длины: poisk\_min\_s(x)

Увеличение всех последовательностей на длину минимальной последовательности: uvelichenie(x, n, min\_s)

**Спецификация подпрограмм**

1. Функция poisk\_min\_s

Входные параметры: x - массив

Выходные параметры: min\_s – длина минимальной последовательности

Описание: ищет минимальную последовательность из символов в массиве и сохраняет ее длину в переменную min\_s

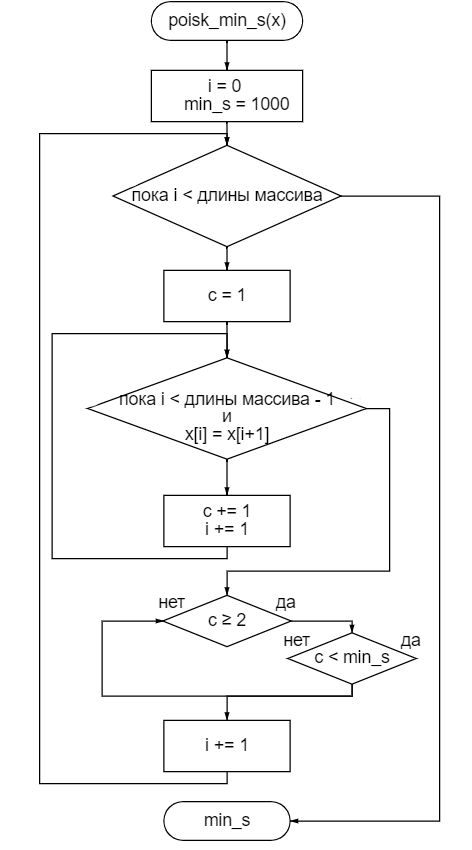


Рисунок 1. Блок-схема для подпрограммы poisk\_min\_s

2. Функция uvelichenie

Входные данные: x – исходный массив, n – длинна массива х, min\_s – длина минимальной последовательности

Выходные данные: x

Описание: функция находит последовательности и увеличивает их на длину минимальной последовательности

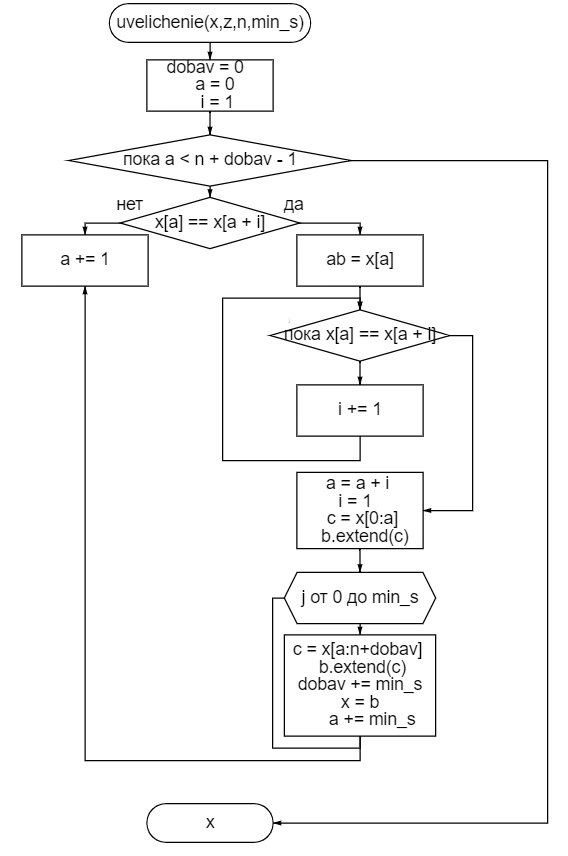


Рисунок 2. Блок-схема функции uvelichenie(x, n, min\_s)

*Текст программы:*

#Функция поиска минимальной последовательности и ее длины

def poisk\_min\_s(x):

i = 0

min\_s = 1000

#проход по массиву

while i < len(x):

c = 1

#сравнивание текущего и следующего символа

while i < len(x)-1 and x[i] == x[i+1]:

c += 1

i = i + 1

#если это последовательность, то

if c >= 2:

if c < min\_s:

#новое минимальное число

min\_s = c

i = i + 1

return min\_s

#Функция увеличения всех последовательностей на длину минимальной последовательности

def uvelichenie(x,n,min\_s):

#переменная для добавки символов в список

dobav = 0

a = 0

i = 1

b = []

c = []

ab = []

#проверка при проходе массива с добавлением #символов

while a < n + dobav - 1:

if x[a] == x[a + i]:

ab = x[a]

while x[a] == x[a + i]:

i += 1

a = a + i

i = 1

c = x[0:a]

#расширим список для вставки символов в #конец

b.extend(c)

for j in range(min\_s):

b.append(ab)

c = x[a:n+dobav]

b.extend(c)

#изменение длины добавления для увеличения

#списка

dobav += min\_s

x = b

b = []

a += min\_s

else:

a += 1

return x

#Основная программа

n = int(input("Введите количество элементов массива: "))

x = list(map(int,input("Введите числа массива через пробел: ").split()))

min\_s = poisk\_min\_s(x)

x = uvelichenie(x,n,min\_s)

print(x)

*Блок-схема полной программы:*

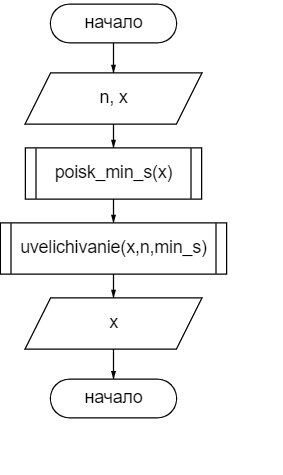


Рисунок 3. Блок-схема всей программы

**Контрольные вопросы**

1. Приведите пример описания функции.

def poisk\_min\_s(x):

i = 0

min\_s = 1000

while i < len(x):

c = 1

while i < len(x)-1 and x[i] == x[i+1]:

c += 1

i = i + 1

if c >= 2:

if c < min\_s:

min\_s = c

Функция начинается со слова def, далее идет имя функции, в скобках указываются данные (переменные), в конце – двоеточие. На следующей строке с отступа идет тело функции.

Если в основную программу требуется передать значение, тогда последняя строка должна начинаться с return. Если ее не указать, то по умолчанию функция возвратит объект None.

2. Что нужно для вызова подпрограммы?

Для вызова подпрограммы надо ввести ее имя и добавить скобки.

Например:

min\_s = poisk\_min\_s(x) # в переменную min\_s вернулся результат функции poisk\_min\_s

3. Какие способы передачи аргументов вы знаете?

· По значению

· По «указателю»

4. В чём различия способов передачи аргументов в подпрограмму?

Если в функцию передается неизменяемый объект, то этот объект передается «по значению». Это значит, что изменение этого объекта внутри функции не изменит его в вызывающем коде.

Если в функцию передается изменяемый объект, то этот объект передается по «указателю». Изменение такого объекта внутри функции повлияет на этот объект в вызывающем коде.

5. Опишите механизм работы подпрограммы.

Пример

Вычислить сумму цифр числа.

def sumD(n): # определение функции с параметром

sumD = 0 while n!= 0: sumD += n % 10 n = n // 10 return sumD # возврат значения функции

# основная программа

print (sumD(int(input())) # вызов функции с параметром

6. Какие типы функций вы знаете?

· Глобальные

Доступны из любой точки программного кода в том же модуле или из других модулей;

· Локальные (вложенные)

Объявляются внутри других функций и видны только внутри них: используются для создания вспомогательных функций, которые нигде больше не используются.

· Анонимные

Не имеют имени и объявляются в месте использования. В Python и представлены лямбда-выражениями;

· Методы

Функции, ассоциированные с каким-либо объектом (например, list.append(), где append() - метод объекта list).

7. Что такое область видимости?

Область видимости переменных в языке программирования Python представляет собой некое пространство имен, в рамках которого функционируют созданные объекты.

Локальные переменные видны только в локальной области видимости, которой может выступать отдельно взятая функция. Глобальные переменные видны во всей программе. "Видны" – значит, известны, доступны. К ним можно обратиться по имени и получить связанное с ними значение.

К глобальной переменной можно обратиться из локальной области видимости. К локальной переменной нельзя обратиться из глобальной области видимости, потому что локальная переменная существует только в момент выполнения тела функции. При выходе из нее, локальные переменные исчезают. Компьютерная память, которая под них отводилась, освобождается. Когда функция будет снова вызвана, локальные переменные будут созданы заново.