Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

**Отчет по лабораторной работе №1**

Дисциплина: «Разработка профессиональных приложений»

Обработка списков

Вариант № 14

Выполнил:

студент группы ИВТАПбд-21

Мухаметзянов Т. А.

Проверил:

преподаватель кафедры

«Вычислительная техника»

Исхаков И.И.

Ульяновск, 2023

**Задание по варианту**

Необходимо реализовать 14 вариант задания, а именно: из списка удалить элементы, стоящие после максимального и имеющие значение меньше среднего арифметического всех элементов списка.

**Пример:**

из списка A[5]: 8 6 9 4 5 должен получиться список A[3]: 8 6 9 (среднее арифметическое четных элементов =(8+6+9+4+5)/5=6.4).

**Требования к программе:**

1. Ввод элементов списка должен быть доступен с клавиатуры и путем автоматической генерации
2. Программа должна корректно обрабатывать некорректный ввод.
3. Исходный код должен быть откомментирован
4. Необходимо реализовать правильную декомпозицию программы на методы.
5. Необходимо реализовать предложенный алгоритм без использования стандартных функций и с использованием (необходимо реализовать два отдельных метода).

**Описание реализации**

**Общее:**

Были созданы функции methodsList() и customList() с пустыми параметрами, внутри данных функций происходит ввод с клавиатуры длины будущего списка независимо от того будет список состоять из рандомных чисел или вводимых пользователем. После пользователь вводит строку в которой указывает желаемый вариант заполнения списка (random or manually).

Для рандомного заполнения импортируем библиотеку random, вызываем ее метод randint(0, 10) который заполнит наш список пользовательской длины целочисленными значениями от 0 до 9.

Для ручного заполнения создаем цикл while, который выполняется до тех пор, пока длина нашего списка не станет равна вводимой длине. Дальше на каждой новой итерации цикла вводим строку и если это строка является числом (узнаем с помощью метода isdigit() **(для реализации методами))** (узнаем с помощью try except **(для ручной реализации)**, то добавляем ее в наш список, иначе выводим сообщение об ошибке и завершаем программу.

В нашей программе нам необходимо найти максимальный элемент списка, его индекс в списке, а также среднее арифметическое всех элементов

**Реализация с помощью стандартных методов:**

Максимальный элемент с помощью метода **max**(list)

Индекс максимального в списке с помощью метода list.**index(max(list)**



Среднее арифметическое всех элементов списка:

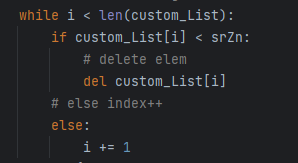


Удаление элемента по условиям:

Переменная, которая будет указывать на следующий элемент после максимального в списке:

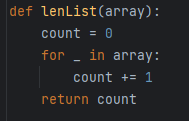


Цикл while в котором проверяется по очереди каждый элемент после максимального и если он меньше среднего арифметического, то удаляем его с помощью функции del, если нет, то просто идем к следующему элементу:

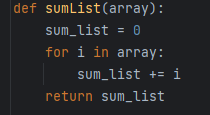


**Ручная реализация:**

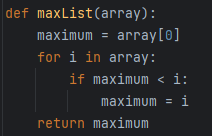
Функция нахождения длины списка:



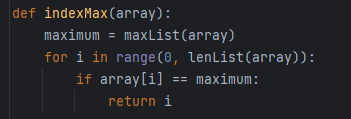
Функция нахождения суммы элементов списка:



Функция нахождения максимального элемента списка:



Функция нахождения индекса максимального элемента списка:



Ср.Ар:



Цикл while в котором проверяется по очереди каждый элемент после максимального и если он меньше среднего арифметического, то мы меняем исходный список, присваивая ему два новых (от начала списка до того элемента, который будет убран и от стоящего после убранного до конца) – подобно методы slice()

**Описание возникших затруднений**

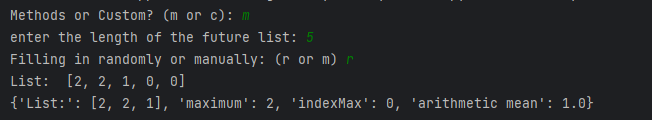
При написании кода трудностей не возникло.

**Описание альтернативных способов решения**

Ключевое слово del при удалении можно заменить методами списка – pop() и remove() отличие лишь в том, что remove() удаляет элемент по первому его вхождению, pop() возвращает удаленный элемент по его индексу, а del удаляет элемент по индексу. Так как нам не нужно возвращать удаленный элемент было выбрано ключевое слово del.

Выбран метод append() вместо extend() так как нам необходимо на каждой итерации цикла while добавлять 1 элемент, а не расширять список добавляя в него новый объект.

**Результаты работы программы**



**1 рис. Рандомное заполнение в функции со стандартными методами**



**2. рис. Заполнение с клавиатуры в функции со кастомными методами**

**Код программы**

import random  
  
  
# custom lengthList  
def lenList(array):  
 count = 0  
 for \_ in array:  
 count += 1  
 return count  
  
  
# custom sumList  
def sumList(array):  
 sum\_list = 0  
 for i in array:  
 sum\_list += i  
 return sum\_list  
  
  
# custom maxList  
def maxList(array):  
 maximum = array[0]  
 for i in array:  
 if maximum < i:  
 maximum = i  
 return maximum  
  
  
# custom indexMaxElem  
def indexMax(array):  
 maximum = maxList(array)  
 for i in range(0, lenList(array)):  
 if array[i] == maximum:  
 return i  
  
  
# methodsFunc  
def methodsList():  
 custom\_List = []  
 # input length  
 length = int(input("enter the length of the future list: "))  
 # return [] if length == 0  
 if length == 0:  
 return []  
 match input("Filling in randomly or manually: (r or m) "):  
 # random inputList  
 case "r":  
 custom\_List = [random.randint(0, 10) for \_ in range(0, length)]  
 case "m":  
 while len(custom\_List) < length:  
 elem = input()  
 # checking for a number  
 if elem.isdigit():  
 # add in list  
 custom\_List.append(int(elem))  
 else:  
 print("It`s not a number!")  
 return {"List": custom\_List}  
 case \_:  
 print("Error!")  
 print("List: ", custom\_List)  
 # indexMax  
 indexMaximum = custom\_List.index(max(custom\_List))  
 # arithmetic mean  
 srZn: float = sum(custom\_List) / len(custom\_List)  
 # start our while for delete elem  
 i = indexMaximum + 1  
 # while index < lengthList  
 while i < len(custom\_List):  
 if custom\_List[i] < srZn:  
 # delete elem  
 del custom\_List[i]  
 # else index++  
 else:  
 i += 1  
 return {'List:': custom\_List, 'maximum': max(custom\_List), 'indexMax': indexMaximum, 'arithmetic mean': srZn}  
  
  
# customMethods  
def customList():  
 custom\_List: list = []  
 length: int = int(input("enter the length of the future list: "))  
 if length == 0:  
 return []  
 match input("Filling in randomly or manually: (r or m) "):  
 case "r":  
 custom\_List = [random.randint(0, 10) for \_ in range(0, length)]  
 case "m":  
 while lenList(custom\_List) < length:  
 elem: str = input()  
 # try except for valid input  
 try:  
 custom\_List += [int(elem)]  
 except Exception:  
 print("It`s not a number!")  
 return {"List": custom\_List}  
 case \_:  
 print("Error!")  
 print("List: ", custom\_List)  
 srZn: float = sumList(custom\_List) / lenList(custom\_List)  
 i: int = indexMax(custom\_List)  
 while i < lenList(custom\_List):  
 if custom\_List[i] < srZn:  
 # новый массив создается по сути из двух: из тех элементов, что идут до нашего  
 # удаляемого и тех что идут после него  
 custom\_List = custom\_List[:i] + custom\_List[i + 1:]  
 else:  
 i += 1  
 return {'List:': custom\_List, 'maximum': maxList(custom\_List), 'indexMax': indexMax(custom\_List),  
 'arithmetic mean': srZn}  
  
  
match input("Methods or Custom? (m or c): "):  
 case "c":  
 print(customList())  
 case "m":  
 print(methodsList())  
 case \_:  
 print("Error!")