Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

Разработка профессиональных приложений

(название дисциплины)

**Отчёт по лабораторной работе №1**

«Обработка списков»

(название (тема) работы)

**Вариант №13**

Выполнил:

студент группы ИВТАПбд-21

Молчанов А. В.

Проверил(а):

преподаватель кафедры «ВТ»

Исхаков И. И.

Ульяновск

2023

**Постановка задачи**

Из списка удалить элементы, имеющие значение меньше среднего арифметического чётных элементов списка.

Пример: из списка A[5]: 8 7 2 6 5 должен получиться список A[3]: 8 7 6 (среднее арифметическое чётных элементов = (8 + 2 + 6) / 3 = 5.(3)).

**Описание реализации**

В соответствии с общими требованиями к программе, она была реализована в двух вариантах: с и без использования стандартных библиотечных методов. Под каждую из реализаций были сформированы отдельные .py файлы, исходный код которых приведён ниже.

Помимо этого, обе реализации соответствуют прочим общим требованиям к программе, а именно:

1. Наличие простейшего пользовательского интерфейса для ввода элементов списка как с клавиатуры самим пользователем (ручной ввод), так и путём автоматической генерации случайных значений (автоматический ввод);
2. Обработка всех возможных исключений некорректного ввода пользователя и вывод соответствующих сообщений об ошибках;
3. Подробные комментарии к исходному коду;
4. Декомпозиция всех возможных элементов программы на отдельные методы.

**Исходный код реализации с использованием стандартных библиотечных методов**

main.py

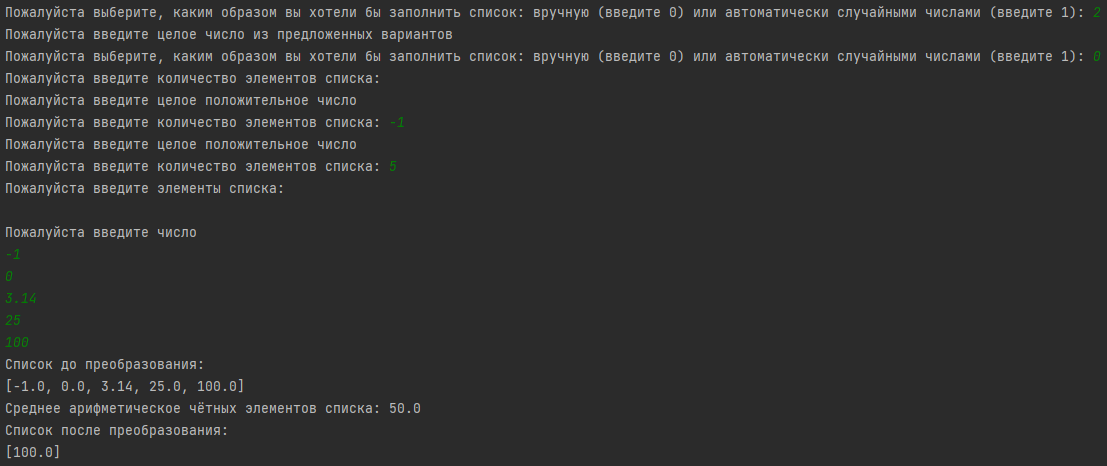
import random  
  
a = 0 # Переменная выбора типа ввода пользователем  
avrg = 0.0 # Переменная среднего арифметического чётных элементов списка  
list = []  
  
  
# Метод пользовательского ввода с обработкой всех исключений  
def userInput():  
 while True:  
 try:  
 a = int(input("Пожалуйста выберите, каким образом вы хотели бы заполнить список: вручную (введите 0) или автоматически случайными "  
 "числами (введите 1): "))  
 except ValueError:  
 print("Пожалуйста введите целое число из предложенных вариантов")  
 else:  
 if ((a != 0) and (a != 1)):  
 print("Пожалуйста введите целое число из предложенных вариантов")  
 else:  
 return a  
 break  
  
  
# Метод ручного заполнения списка пользователем с обработкой всех исключений  
def listUserInput(avrg):  
 k = 0 # Переменная-счётчик количества чётных элементов списка  
  
 for i in range(n):  
 while True:  
 try:  
 e = float(input()) # float используется для универсальности  
 except ValueError:  
 print("Пожалуйста введите число")  
 else:  
 list.append(e) # Метод append добавляет элемент в конец списка  
  
 if (e % 2) == 0: # Если введённое пользователем число e чётное  
 avrg += e # Суммируем его в переменную среднего арифметического  
 k += 1  
 break  
  
 if k != 0:  
 avrg /= k  
  
 return avrg  
  
  
# Метод автоматического заполнения списка случайными числами  
def listRandomInput(avrg):  
 k = 0  
  
 for i in range(n):  
 # Метод randint генерирует случайное целое число в диапазоне от l до r, введённом пользователем с обработкой всех исключений  
 e = random.randint(l, r)  
  
 list.append(e)  
  
 if (e % 2) == 0:  
 avrg += e  
 k += 1  
  
 if k != 0:  
 avrg /= k  
  
 return avrg  
  
  
# Метод для удаления элементов списка, меньших среднего арифметического чётных элементов этого списка  
def listUpdate(n):  
 i = 0  
  
 while i < n: # Проходимся по всем индексам элементов списка  
 if list[i] < avrg: # Если число по индексу i списка строго меньше среднего арифметического чётных элементов этого списка  
 list.pop(i) # Метод pop удаляет i-ый элемент из списка  
 n -= 1 # Уменьшаем размер списка n на 1 только что удалённый элемент  
  
 if i > 0:  
 # После удаления из списка элемента, его место займёт элемент, который до его удаления стоял перед ним.  
 # Чтобы рассмотреть этот элемент, необходимо вернуться на 1 индекс назад (-1) кроме случая, когда мы удалили  
 # из списка первый элемент с индексом 0 (элемента с индексом -1 не существует)  
 i -= 1  
 else:  
 i += 1 # Иначе идём дальше (+1) по циклу и берём индекс следующего элемента списка  
  
 return list  
  
  
a = userInput()  
  
if a == 0: # Если пользователь выбрал ручной ввод  
 while True:  
 try:  
 n = int(input("Пожалуйста введите количество элементов списка: "))  
 except ValueError:  
 print("Пожалуйста введите целое положительное число")  
 else:  
 if (n <= 0):  
 print("Пожалуйста введите целое положительное число")  
 else:  
 print("Пожалуйста введите элементы списка: ")  
  
 avrg = listUserInput(avrg)  
  
 print("Список до преобразования:")  
 print(list)  
  
 print("Среднее арифметическое чётных элементов списка:", avrg)  
  
 list = listUpdate(n)  
  
 print("Список после преобразования:")  
 print(list)  
 break  
else: # Если пользователь выбрал автоматический (случайный) ввод  
 while True:  
 try:  
 n = int(input("Пожалуйста введите количество элементов списка: "))  
 except ValueError:  
 print("Пожалуйста введите целое положительное число")  
 else:  
 if (n <= 0):  
 print("Пожалуйста введите целое положительное число")  
 else:  
 while True:  
 try:  
 l = int(input("Пожалуйста введите значение левой границы диапазона для генерации случайных чисел: "))  
 r = int(input("Пожалуйста введите значение правой границы диапазона для генерации случайных чисел: "))  
 except ValueError:  
 print("Пожалуйста введите целое число")  
 else:  
 if l > r:  
 print("Левая граница диапазона для генерации случайных чисел должна быть меньше или равна правой!")  
 else:  
 avrg = listRandomInput(avrg)  
  
 print("Список до преобразования:")  
 print(list)  
  
 print("Среднее арифметическое чётных элементов списка:", avrg)  
  
 list = listUpdate(n)  
  
 print("Список после преобразования:")  
 print(list)  
 break  
 break

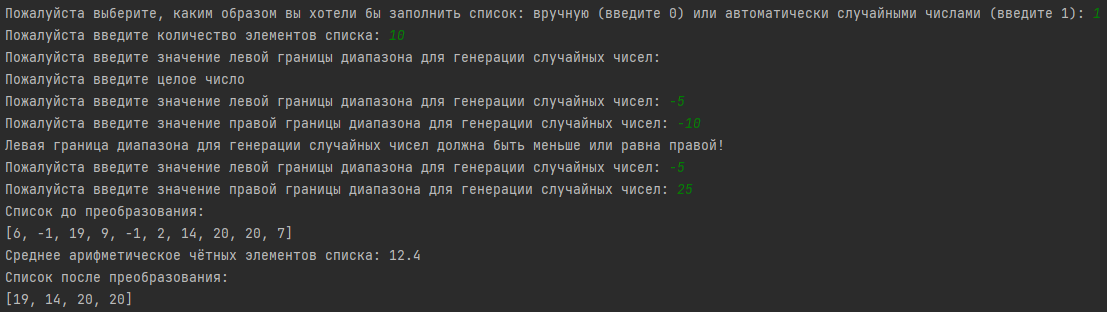
**Исходный код реализации без использования стандартных библиотечных методов**

main\_own.py

import random  
  
  
# Класс элемента списка  
class Node:  
 # Метод \_\_init\_\_ отвечает за инициализацию экземпляров класса элемента списка после их создания  
 def \_\_init\_\_(self, data):  
 self.item = data # data - значение элемента списка  
 self.nref = None # nref - ссылка на следующий элемент списка  
 self.pref = None # pref - ссылка на предыдущий элемент списка  
  
  
# Класс двусвязного списка  
class DoublyLinkedList:  
 def \_\_init\_\_(self):  
 self.start\_node = None # По умолчанию в нём нет элементов  
  
 # Данный метод добавляет элемент в конец списка  
 def insert\_at\_end(self, data):  
 if self.start\_node is None:  
 new\_node = Node(data) # Создаём элемент списка  
 self.start\_node = new\_node  
 return  
 n = self.start\_node  
  
 # Проходимся по всему списку  
 while n.nref is not None:  
 n = n.nref # Получаем ссылку на следующий элемент последнего элемента списка  
  
 # Вставляем новый элемент в конец списка  
 new\_node = Node(data)  
 n.nref = new\_node  
 new\_node.pref = n  
  
 # Данный метод обходит весь список и выводит его  
 def traverse\_list(self):  
 if self.start\_node is None:  
 print("Список пуст!")  
 return  
 else:  
 n = self.start\_node  
  
 print('[', end = '')  
  
 while n is not None:  
 # Если элемент не последний отделяем его запятой  
 if n.nref is not None:  
 print(n.item, end=', ')  
 else: # После последнего элемента закрываем квадратную скобку  
 print(n.item, end = ']')  
  
 n = n.nref # Берём следующий элемент  
  
 # Данный метод удаляет элемент из списка по его индексу  
 def delete\_element\_by\_index(self, i):  
 # Если в качестве индекса была передана строка  
 if type(i) == str:  
 print("Элемента с таким индексом не существует!")  
 return  
  
 n = self.start\_node  
 k = 0 # Переменная-счётчик общего количества элементов списка  
  
 while n is not None:  
 k += 1  
 n = n.nref  
  
 if (i < 0) or (i >= k):  
 print("Элемента с таким индексом не существует!")  
 return  
  
 n = self.start\_node  
 k = 0  
  
 while n is not None:  
 # Если элемент с индексом i был найден  
 if i == k:  
 # Проверяем первый ли он  
 if n.pref is None:  
 list.delete\_at\_start() # Удаляем его  
 n = self.start\_node # Элемент, следующий за удалённым элементом, становится первым в списке  
 n.pref = None # Так как он первый, то ссылки на предыдущий элемент у него не будет  
 elif n.nref is None: # Если элемент с индексом i последний  
 n.pref.nref = None # "Забываем" его у предыдущего элемента  
 else: # Если элемент с индексом i и не первый, и не последний  
 # Аккуратненько вставляем его между двумя элементами списка  
 n.pref.nref = n.nref  
 n.nref.pref = n.pref  
 break  
 else: # Если элемент с индексом i не был найден  
 k += 1  
 n = n.nref # Берём следующий элемент  
  
 # Данный метод находит элемент в списке по его индексу  
 def find\_element\_by\_index(self, i):  
 if type(i) == str:  
 print("Элемента с таким индексом не существует!")  
 return  
  
 n = self.start\_node  
 k = 0  
  
 while n is not None:  
 k += 1  
 n = n.nref  
  
 if (i < 0) or (i >= k):  
 print("Элемента с таким индексом не существует!")  
 return  
  
 n = self.start\_node  
 k = 0  
  
 while n is not None:  
 # Если элемент с индексом i был найден  
 if i == k:  
 return n.item # Возвращаем его значение  
 break  
 else:  
 k += 1  
 n = n.nref  
  
 # Данный метод удаляет первый элемент из списка  
 def delete\_at\_start(self):  
 if self.start\_node is None:  
 print("Список пуст!")  
 return  
  
 # Если в списке всего 1 элемент  
 if self.start\_node.nref is None:  
 self.start\_node = None  
 return  
  
 self.start\_node = self.start\_node.nref # Делаем первым элементом тот элемент, который стоял после первого  
 self.start\_prev = None # Ссылки на предыдущий элемент у первого элемента списка нет  
  
  
a = 0 # Переменная выбора типа ввода пользователем  
avrg = 0.0 # Переменная среднего арифметического чётных элементов списка  
list = DoublyLinkedList() # Создание объекта экземпляра класса кастомного списка  
  
def userInput():  
 while True:  
 try:  
 a = int(input("Пожалуйста выберите, каким образом вы хотели бы заполнить список: вручную (введите 0) или автоматически случайными "  
 "числами (введите 1): "))  
 except ValueError:  
 print("Пожалуйста введите целое число из предложенных вариантов")  
 else:  
 if ((a != 0) and (a != 1)):  
 print("Пожалуйста введите целое число из предложенных вариантов")  
 else:  
 return a  
 break  
  
  
# Метод ручного заполнения списка пользователем с обработкой всех исключений  
def listUserInput(avrg):  
 k = 0 # Переменная-счётчик количества чётных элементов списка  
  
 for i in range(n):  
 while True:  
 try:  
 e = float(input()) # float используется для универсальности  
 except ValueError:  
 print("Пожалуйста введите число")  
 else:  
 list.insert\_at\_end(e) # Кастомный метод insert\_at\_end добавляет элемент в конец списка  
  
 if (e % 2) == 0: # Если введённое пользователем число e чётное  
 avrg += e # Суммируем его в переменную среднего арифметического  
 k += 1  
 break  
  
 if k != 0:  
 avrg /= k  
  
 return avrg  
  
  
# Метод автоматического заполнения списка случайными числами  
def listRandomInput(avrg):  
 k = 0  
  
 for i in range(n):  
 # Метод randint генерирует случайное целое число в диапазоне от l до r, введённом пользователем с обработкой всех исключений  
 e = random.randint(l, r)  
  
 list.insert\_at\_end(e)  
  
 if (e % 2) == 0:  
 avrg += e  
 k += 1  
  
 if k != 0:  
 avrg /= k  
  
 return avrg  
  
  
# Метод для удаления элементов списка, меньших среднего арифметического чётных элементов этого списка  
def listUpdate(n):  
 i = 0  
  
 while i < n: # Проходимся по всем индексам элементов списка  
 e = list.find\_element\_by\_index(i) # Кастомный метод find\_element\_by\_index находит элемент в списке по его индексу  
  
 if e < avrg: # Если число по индексу i списка строго меньше среднего арифметического чётных элементов этого списка  
 list.delete\_element\_by\_index(i) # Кастомный метод delete\_element\_by\_index удаляет элемент из списка по его индексу  
 n -= 1 # Уменьшаем размер списка n на 1 только что удалённый элемент  
  
 if i > 0:  
 # После удаления из списка элемента, его место займёт элемент, который до его удаления стоял перед ним.  
 # Чтобы рассмотреть этот элемент, необходимо вернуться на 1 индекс назад (-1) кроме случая, когда мы удалили  
 # из списка первый элемент с индексом 0 (элемента с индексом -1 не существует)  
 i -= 1  
 else:  
 i += 1 # Иначе идём дальше (+1) по циклу и берём индекс следующего элемента списка  
  
 return list  
  
  
a = userInput()  
  
if a == 0: # Если пользователь выбрал ручной ввод  
 while True:  
 try:  
 n = int(input("Пожалуйста введите количество элементов списка: "))  
 except ValueError:  
 print("Пожалуйста введите целое положительное число")  
 else:  
 if (n <= 0):  
 print("Пожалуйста введите целое положительное число")  
 else:  
 print("Пожалуйста введите элементы списка: ")  
  
 avrg = listUserInput(avrg)  
  
 print("Список до преобразования:")  
 list.traverse\_list() # Кастомный метод traverse\_list обходит весь список и выводит его  
  
 print()  
  
 print("Среднее арифметическое чётных элементов списка:", avrg)  
  
 list = listUpdate(n)  
  
 print("Список после преобразования:")  
 list.traverse\_list()  
 break  
else: # Если пользователь выбрал автоматический (случайный) ввод  
 while True:  
 try:  
 n = int(input("Пожалуйста введите количество элементов списка: "))  
 except ValueError:  
 print("Пожалуйста введите целое положительное число")  
 else:  
 if (n <= 0):  
 print("Пожалуйста введите целое положительное число")  
 else:  
 while True:  
 try:  
 l = int(input("Пожалуйста введите значение левой границы диапазона для генерации случайных чисел: "))  
 r = int(input("Пожалуйста введите значение правой границы диапазона для генерации случайных чисел: "))  
 except ValueError:  
 print("Пожалуйста введите целое число")  
 else:  
 if l > r:  
 print("Левая граница диапазона для генерации случайных чисел должна быть меньше или равна правой!")  
 else:  
 avrg = listRandomInput(avrg)  
  
 print("Список до преобразования:")  
 list.traverse\_list()  
  
 print()  
  
 print("Среднее арифметическое чётных элементов списка:", avrg)  
  
 list = listUpdate(n)  
  
 print("Список после преобразования:")  
 list.traverse\_list()  
 break  
 break

**Пример работы программы**

****

****

**Описание возникших затруднений**

Основные затруднения при выполнении данной лабораторной работой были связаны с тем, что до этого язык программирования Python мной не изучался, и я не написал на нём ни одной строчки кода, поэтому поначалу был не привычен синтаксис данного языка по сравнению с синтаксисом ЯП C++, а также структура написания программ на нём (отступы).

Также, при реализации программы без использования стандартных библиотечных методов пришлось вспомнить принципы работы двусвязных списков, проведя немало времени в отладчике PyCharm.

**Описание альтернативных способов решения**

Так как тема данной лабораторной работы — это обработка списков, то их альтернативой в языке программирования Python могут выступать: кортежи, множества, словари, деки, очереди, массивы и так далее, но всё же они не являются такими универсальными в работе как списки, не позволяют изменять данные по ходу программы (кортежи), имеют ограниченную область применения по сравнению с ними (множества), имеют переусложнённую структуру и функционал по сравнению со списками (словари) и так далее.

Таким образом, альтернативные способы решения у данной задачи есть, но они будут более громоздкими и переусложнёнными по сравнению с реализацией этой же задачи при помощи списков.

**Вывод**

В ходе выполнения данной лабораторной работы были получены первичные представления о языке программирования Python, его синтаксисе и структуре, а также о работе со списками в нём. Отдельно хотелось бы отметить, что достаточно жёсткие требования к реализации программы значительно повысили моё качество написания комментариев к своему коду