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 №4**

Выполнил:

студент группы ИВТАПбд-21

Галацков И.А.

Проверил(а):

преподаватель кафедры «ВТ»

Исхаков И. И.

Ульяновск

2023

**Постановка задачи**

Из списка A удалить те элементы, которые встречаются и в списке A и в списке B по крайней мере по 2 раза.

Пример: список A[8]: 3 3 4 5 2 3 5 9

список B[7]: 1 2 3 4 5 2 5

По 2 раза в обоих списках встречается только элемент, равный 5.

список A после удаления примет вид: A[6]: 3 3 4 2 3 9.

**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  
listA = [] #Объявление списка  
listB = []  
  
def update(): #Основная логика  
 for i in listA: # с помощью цокла for проходимся по списку A  
 eA = listA.count(i) #Присваиваем в переменную eA количество вхождений текущего элемента  
 if eA >= 2:#Проверяем, чтобы кол. вхождений было больше, либо равно 2  
 for j in listB: # с помощью цокла for проходимся по списку B  
 if j == i: #Если значение элемента списка B равно значению элемента списка A  
 eB = listB.count(j) #Присваиваем в переменную eB количество вхождений текущего элемента  
 if eB >= 2: #Проверяем, чтобы кол. вхождений было больше, либо равно 2  
 listA.remove(i) #Удаляем элемент с этим значением из списка A  
 return listA #Возвращаем измененный список A  
  
def userInput(): #Функция ввода пользователя  
 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)): #обработка ошибки ввода  
 print("Пожалуйста введите целое число из предложенных вариантов")  
 else:  
 return a  
 break  
  
a = userInput()  
  
if a == 0: #Если спик заполняется вручную  
 while True:  
 try:  
 nA = int(input("Пожалуйста введите количество элементов списка A: "))  
 nB = int(input("Пожалуйста введите количество элементов списка B: "))  
 except ValueError:  
 print("Пожалуйста введите целое положительное число")  
 else:  
 if (nA <= 0):  
 print("Пожалуйста введите целое положительное число")  
  
 else:  
 print("Пожалуйста введите элементы списка A: ")  
  
 for i in range(nA):  
 while True:  
 try:  
 e = int(input())  
 except ValueError:  
 print("Пожалуйста введите целое число")  
 else:  
 listA.append(e) # добавление элементов в список  
 break  
 if (nB <= 0):  
 print("Пожалуйста введите целое положительное число")  
  
 else:  
 print("Пожалуйста введите элементы списка B: ")  
  
 for i in range(nB):  
 while True:  
 try:  
 e = int(input())  
 except ValueError:  
 print("Пожалуйста введите целое число")  
 else:  
 listB.append(e)  
 break  
  
 print("Список A до преобразования:")  
 print(listA)  
 print("Список B до преобразования:")  
 print(listB)  
 listA = update()  
 print("Список A после преобразования:")  
 print(listA)  
 break  
  
  
else: #Если спик заполняется атоматически  
 while True:  
 try:  
 nA = int(input("Пожалуйста введите количество элементов списка A: "))  
 nB = int(input("Пожалуйста введите количество элементов списка B: "))  
 except ValueError:  
 print("Пожалуйста введите целое положительное число")  
 else:  
 if (nA <= 0):  
 print("Пожалуйста введите целое положительное число")  
 else:  
 while True:  
 try:  
 lA = int(input("Пожалуйста введите значение левой границы диапазона для генерации случайных чисел списка A: "))  
 rA = int(input("Пожалуйста введите значение правой границы диапазона для генерации случайных чисел списка A: "))  
 except ValueError:  
 print("Пожалуйста введите целое число")  
 else:  
 for i in range(nA):  
 e = random.randint(lA, rA) #Используется рандомайзер, для заполнения, в пределах от lA до rA  
  
 listA.append(e)  
 break  
 if (nB <= 0):  
 print("Пожалуйста введите целое положительное число")  
 else:  
 while True:  
 try:  
 lB = int(input(  
 "Пожалуйста введите значение левой границы диапазона для генерации случайных чисел списка B: "))  
 rB = int(input(  
 "Пожалуйста введите значение правой границы диапазона для генерации случайных чисел списка B: "))  
 except ValueError:  
 print("Пожалуйста введите целое число")  
 else:  
 for i in range(nB):  
 e = random.randint(lB, rB)  
  
 listB.append(e)  
 break  
  
  
 print("Список A до преобразования:")  
 print(listA) #Вывод списка  
 print("Список B до преобразования:")  
 print(listB)  
  
 listA = update()  
 print("Список A после преобразования:")  
 print(listA)  
 break

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

main\_own.py

import random  
class Node: #Класс для единственного узла в списке  
 def \_\_init\_\_(self, data):  
 self.item = data #Текущий элемент списка  
 self.nref = None #Ссылка на следующий элемент  
 self.pref = None #Ссылка на предыдущий элемент, т.к. узел единственный, то у него нет следующего и предыдущего элементов, их значения None  
  
class DoublyLinkedList: #Класс содержащий функции списка  
 def \_\_init\_\_(self):  
 self.start\_node = None  
  
 def Myappend(self, data):#Добавление элемента в конец списка  
 if self.start\_node is None:#Проверяем список на пустоту  
 new\_node = Node(data) #Значение первого узла  
 self.start\_node = new\_node #Добавляем элемент в пустой список  
 return  
 n = self.start\_node #Присваиваем n значение первого узла  
 while n.nref is not None: #Запускаем цикл до момента пока ссылка на след. эдемент не будет равна None  
 n = n.nref #Следующий узел становится текущим  
 new\_node = Node(data) #Задаем значение  
 n.nref = new\_node #Присваиваем значения след. узлу  
 new\_node.pref = n #Текущий узел становится предыдущим  
  
 def traverse\_list(self): #Обход списка  
 if self.start\_node is None: # ПРоверка на пустоту  
 print("В списке нет элементов")  
 return  
 else: #Если список не пустой то проходимся циклом и выводим каждый элемент  
 n = self.start\_node  
 while n is not None:  
 print(n.item , " ")  
 n = n.nref  
 def Myremove(self, x): #Удаление элемента по значению  
 if self.start\_node is None: #Проверка на пустоту  
 print("В списке нет элемента для удаления")  
 return  
 if self.start\_node.nref is None: #Проверка на наличие след. элемента  
 if self.start\_node.item == x: #Проверяем, тот ли это элемент, который нудо удалить  
 self.start\_node = None #Если да, то удаляем  
 else:  
 print("Элемент для удаления не найден")  
 return  
 if self.start\_node.item == x: #Поверяем значение текущего узла  
 self.start\_node = self.start\_node.nref #Присваиваем текущему узлу ссылку на след. элемент  
 self.start\_node.pref = None #Прыдущий элемент удаляем  
 return  
 n = self.start\_node  
 while n.nref is not None: #если список содержит несколько элементов и удаляемый элемент не является первым элементом, мы просматриваем все элементы в списке  
 if n.item == x:#Проверяем, имеет ли какой-либо из узлов значение, соответствующее значению, которое нужно удалить  
 break;  
 n = n.nref  
 if n.nref is not None:  
 n.pref.nref = n.nref #Установливаем значение следующей ссылки предыдущего узла на следующую ссылку удаляемого узла  
 n.nref.pref = n.pref #Установливаем предыдущее значение следующего узла на предыдущую ссылку удаляемого узла  
 else: #если удаляемый узел является последним узлом  
 if n.item == x:  
 n.pref.nref = None #для следующей ссылки узла, предшествующего последнему узлу, устанавливается значение «None»  
 else:  
 print("Элемент для удаления не найден")  
  
 def mycount(self, x):#Возвращает количество вхождений элемента в список  
 c = 0 #Объявляем счетчик  
 n = self.start\_node  
 while n is not None: #Проходимся по списку  
 if n.item == x: #Если значение элемента списка совпадает с нежным нам значением x  
 c += 1 # То пребавляем счетчик на единицу  
 n = n.nref #И присваиваес текущиму узлу ссылку на следующий  
 else:  
 n = n.nref # Иначе присваиваес текущиму узлу ссылку на следующий  
 return c #Возвращаем количество вхождений  
  
 def update(self):#Основная логика  
 n = self.start\_node  
 while n is not None: # Проходимся по списку A  
 eA = listA.mycount(n.item) #Присваиваем в переменную eA количество вхождений текущего элемента  
 if eA >= 2: #Проверяем, чтобы кол. вхождений было больше, либо равно 2  
 n1 = listB.start\_node  
 while n1 is not None:# Проходимся по списку B  
 if n1.item == n.item: #Если значение элемента списка B равно значению элемента списка A  
 eB = listB.mycount(n1.item) #Присваиваем в переменную eB количество вхождений текущего элемента  
 if eB >= 2:#Проверяем, чтобы кол. вхождений было больше, либо равно 2  
 listA.Myremove(n.item) #Удаляем элемент с этим значением из списка A  
 n1 = n1.nref #присваиваес текущиму узлу ссылку на следующий в списке B  
 n = n.nref#присваиваес текущиму узлу ссылку на следующий в списке A  
 return listA #Возвращаем измененный список A  
  
a = 0  
listA = DoublyLinkedList() # создадим объект класса DoublyLinkedList  
listB = DoublyLinkedList()# создадим объект класса DoublyLinkedList  
"""  
list.insert\_at\_end(1)  
list.traverse\_list()  
list.delete\_element\_by\_value(1)  
list.traverse\_list()  
"""  
  
  
  
  
def userInput(): #Функция ввода пользователя  
 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)): #обработка ошибки ввода  
 print("Пожалуйста введите целое число из предложенных вариантов")  
 else:  
 return a  
 break  
  
  
  
a = userInput()  
  
if a == 0: #Если спик заполняется вручную  
 while True:  
 try:  
 nA = int(input("Пожалуйста введите количество элементов списка A: "))  
 nB = int(input("Пожалуйста введите количество элементов списка B: "))  
 except ValueError:  
 print("Пожалуйста введите целое положительное число")  
 else:  
 if (nA <= 0):  
 print("Пожалуйста введите целое положительное число")  
  
 else:  
 print("Пожалуйста введите элементы списка A: ")  
  
 for i in range(nA):  
 while True:  
 try:  
 e = int(input())  
 except ValueError:  
 print("Пожалуйста введите целое число")  
 else:  
 listA.Myappend(e) # добавление элементов в список  
 break  
 if (nB <= 0):  
 print("Пожалуйста введите целое положительное число")  
  
 else:  
 print("Пожалуйста введите элементы списка B: ")  
  
 for i in range(nB):  
 while True:  
 try:  
 e = int(input())  
 except ValueError:  
 print("Пожалуйста введите целое число")  
 else:  
 listB.Myappend(e) # добавление элементов в список  
 break  
  
 print("Список A до преобразования:")  
 listA.traverse\_list() #Вывод списка  
 print("Список B до преобразования:")  
 listB.traverse\_list()  
 listA.update()  
 print("Список A после преобразования:")  
 listA.traverse\_list()  
 break  
  
  
else: #Если спик заполняется атоматически  
 while True:  
 try:  
 nA = int(input("Пожалуйста введите количество элементов списка A: "))  
 nB = int(input("Пожалуйста введите количество элементов списка B: "))  
 except ValueError:  
 print("Пожалуйста введите целое положительное число")  
 else:  
 if (nA <= 0):  
 print("Пожалуйста введите целое положительное число")  
 else:  
 while True:  
 try:  
 lA = int(input("Пожалуйста введите значение левой границы диапазона для генерации случайных чисел списка A: "))  
 rA = int(input("Пожалуйста введите значение правой границы диапазона для генерации случайных чисел списка A: "))  
 except ValueError:  
 print("Пожалуйста введите целое число")  
 else:  
 for i in range(nA):  
 e = random.randint(lA, rA) #Используется рандомайзер, для заполнения, в пределах от lA до rA  
  
 listA.Myappend(e)  
 break  
 if (nB <= 0):  
 print("Пожалуйста введите целое положительное число")  
 else:  
 while True:  
 try:  
 lB = int(input(  
 "Пожалуйста введите значение левой границы диапазона для генерации случайных чисел списка B: "))  
 rB = int(input(  
 "Пожалуйста введите значение правой границы диапазона для генерации случайных чисел списка B: "))  
 except ValueError:  
 print("Пожалуйста введите целое число")  
 else:  
 for i in range(nB):  
 e = random.randint(lB, rB)  
  
 listB.Myappend(e)  
 break  
  
 print("Список A до преобразования:")  
 listA.traverse\_list()  
 print("Список B до преобразования:")  
 listB.traverse\_list()  
 listA.update()  
 print("Список A после преобразования:")  
 listA.traverse\_list()  
 break

**Описание возникших затруднений**

Основные затруднения при выполнении данной лабораторной работой были связаны с тем, что до этого момента я никогда не изучал язык программирования Python, поэтому пришлось ознакомиться с основами языка. Синтаксис языка (с использованием отступов) был для меня в новинку.

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