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 №9

Выполнил:

студент группы ИВТАПбд-22

Ступин А. О.

Проверил:

преподаватель кафедры

«Вычислительная техника»

Исхаков И.И.

Ульяновск, 2023

**Задание по варианту**

Между списками A и B обменять их самые длинные цепочки из одинаковых элементов.

Пример: список A[10]: 3 2 2 5 2 1 1 1 3 9

список B[8]: 1 2 5 5 4 8 3 3

В списке A самая длинная цепочка:

1 1 1 (элементы с 7 по 9).

В списке B самая длинная цепочка:

5 5 (элементы с 3 по 4).

список A после перестановки в него цепочки из B: A[9]: 3 2 2 5 2 5 5 3 9.

список B после перестановки в него цепочки из A: B[9]: 1 2 1 1 1 4 8 3 3.

**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).

**Особенности реализации**

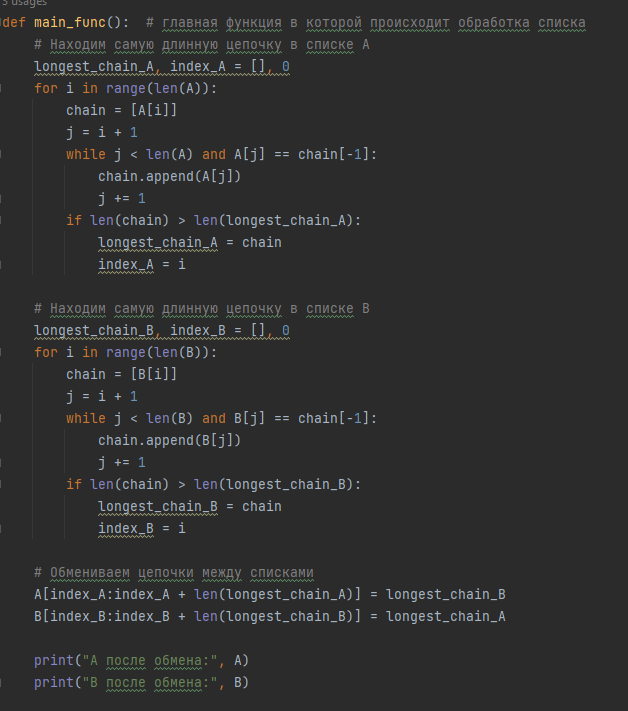
Были созданы два пустых списка А и В, функции manual\_filling[список]() и manual\_filling\_wo\_func[1,2]([список]) для каждого из списков, внутри которых происходит ввод списка с клавиатуры с использованием стандартных функций и без использования стандартных функций соответственно.

Для рандомного заполнения импортируем библиотеку random, вызываем ее метод randint(0, 10) который заполнит списки рандомной длины(от 5 до 20) целочисленными значениями от 0 до 10. Такой небольшой набор значений при такой длине призван обеспечить наличие повторяемых участков в списках при каждом запуске программы.

Для ручного заполнения был создан цикл for, который выполняется до тех пор, пока длина нашего списка не станет равна длине списка.

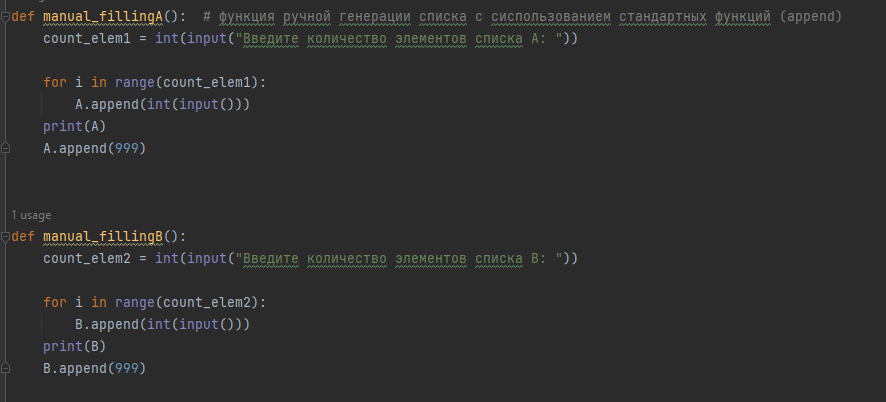
В программе необходимо найти в каждом списке наиболее длинные цепочки повторяющихся элементов и поменять их между списками.

В функции main\_func() происходит поиск цепочек четных чисел и поиск минимального элемента в этих цепочках.

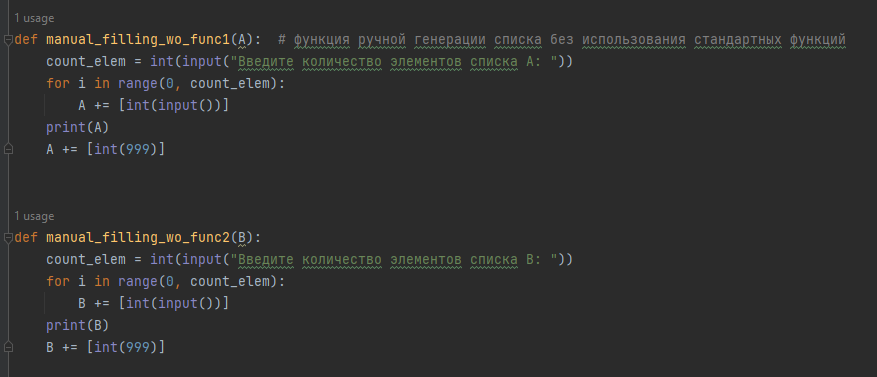


Список longest\_chain хранит самую длинную из последовательностей, index\_A – её начало. Цикл for() проходится по списку, цикл while сравнивает каждый следующий элемент с последним в цепочке chain и, если он равен ему, то добавляет его в цепочку. Наконец, длина chain сравнивается с длиной сохранённой цепочки longest\_chain, и если она окажется больше, то новая цепочка будет longest\_chain. Эти операции выполняются и для списка B. Наконец происходит обмен самых длинных цепочек двух списков.

Функции ручной генерации списка с использованием стандартных методов(append)



Функции ручной генерации списка без использования стандартных методов



**Описание возникших затруднений**

При выполнении лабораторной работы возникли трудности с привыканием к синтаксису и правилам языка Python, последний раз иметь дело с которым приходилось только в 11 классе. Также возникли трудности при работа с методом main\_func(): считалось, что будет достаточно только цикла while, который проходил бы по всему массиву и при этом сразу же сравнивал все элементы и сохранял одинаковые из них в цепочки. Но после неудачных попыток нормально написать такой код было принято решение, что одного итератора недостаточно, поэтому был создан внешний цикл for.

**Описание альтернативных способов решения**

Для поиска последовательностей можно было воспользоваться функциями itertools. В частности, itertools.groupby(data) возвращает последовательность групп одинаковых элементов. Для каждой из групп можно было измерить её длину и найти таким образом наибольшую цепочку.

**Результаты работы программы**

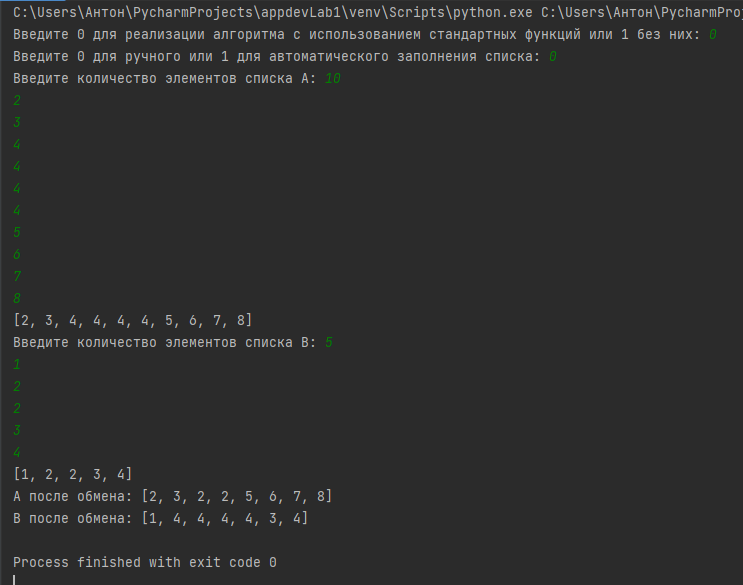


Рис. 1. Ручной ввод с использованием стандартных функций

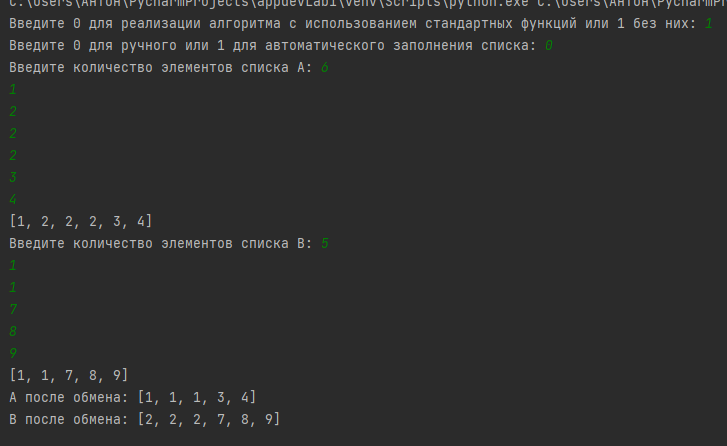


Рис. 2. Ручной ввод без использования стандартных функций

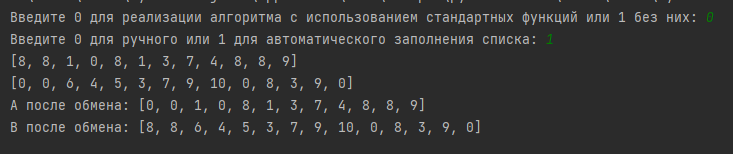


Рис. 3. Автоматическая генерация со стандартными функциями

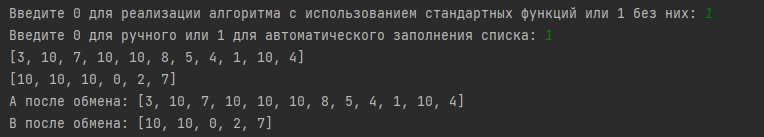


Рис. 4. Автоматическая генерация без стандартных функций

**Приложение 1. Исходный код программы**

import random  
import sys  
  
A = []  
B = []  
  
check\_method = int(input("Введите 0 для реализации алгоритма с использованием стандартных функций или 1 без них: "))  
check = int(input("Введите 0 для ручного или 1 для автоматического заполнения списка: "))  
  
  
def manual\_fillingA(): # функция ручной генерации списка с сиспользованием стандартных функций (append)  
 count\_elem1 = int(input("Введите количество элементов списка А: "))  
  
 for i in range(count\_elem1):  
 A.append(int(input()))  
 print(A)  
 A.append(999)  
  
  
def manual\_fillingB():  
 count\_elem2 = int(input("Введите количество элементов списка B: "))  
  
 for i in range(count\_elem2):  
 B.append(int(input()))  
 print(B)  
 B.append(999)  
  
  
def manual\_filling\_wo\_func1(A): # функция ручной генерации списка без использования стандартных функций  
 count\_elem = int(input("Введите количество элементов списка A: "))  
 for i in range(0, count\_elem):  
 A += [int(input())]  
 print(A)  
 A += [int(999)]  
  
  
def manual\_filling\_wo\_func2(B):  
 count\_elem = int(input("Введите количество элементов списка B: "))  
 for i in range(0, count\_elem):  
 B += [int(input())]  
 print(B)  
 B += [int(999)]  
  
  
def random\_a():  
 A = [random.randint(0, 10) for i in range(random.randint(5, 20))]  
 print(A)  
 A.append(999)  
  
  
def random\_b():  
 B = [random.randint(0, 10) for i in range(random.randint(5, 20))]  
 print(B)  
 B.append(999)  
  
def random\_a\_wo\_func(A):  
 A = [random.randint(0, 10) for i in range(random.randint(5, 20))]  
 print(A)  
 A += [int(999)]  
  
  
def random\_b\_wo\_func(B):  
 B = [random.randint(0, 10) for i in range(random.randint(5, 20))]  
 print(B)  
 B += [int(999)]  
  
  
  
min\_row\_id = 0  
  
  
def main\_func(): # главная функция в которой происходит обработка списка  
 # Находим самую длинную цепочку в списке A  
 longest\_chain\_A, index\_A = [], 0  
 for i in range(len(A)):  
 chain = [A[i]]  
 j = i + 1  
 while j < len(A) and A[j] == chain[-1]:  
 chain.append(A[j])  
 j += 1  
 if len(chain) > len(longest\_chain\_A):  
 longest\_chain\_A = chain  
 index\_A = i  
  
 # Находим самую длинную цепочку в списке B  
 longest\_chain\_B, index\_B = [], 0  
 for i in range(len(B)):  
 chain = [B[i]]  
 j = i + 1  
 while j < len(B) and B[j] == chain[-1]:  
 chain.append(B[j])  
 j += 1  
 if len(chain) > len(longest\_chain\_B):  
 longest\_chain\_B = chain  
 index\_B = i  
  
 # Обмениваем цепочки между списками  
 A[index\_A:index\_A + len(longest\_chain\_A)] = longest\_chain\_B  
 B[index\_B:index\_B + len(longest\_chain\_B)] = longest\_chain\_A  
  
 print("A после обмена:", A)  
 print("B после обмена:", B)  
  
  
def with\_func(A, B): # определение метода генерации списка  
 if check == 0:  
 if check\_method == 0:  
 manual\_fillingA()  
 manual\_fillingB()  
 main\_func()  
 elif check\_method == 1:  
 manual\_filling\_wo\_func1(A)  
 manual\_filling\_wo\_func2(B)  
 main\_func()  
  
 elif check == 1:  
 if check\_method == 0:  
 random\_a()  
 random\_b()  
 # main\_func()  
 elif check\_method == 1:  
 random\_a\_wo\_func(A)  
 random\_b\_wo\_func(B)  
 main\_func()  
 else:  
 print("Ошибка ввода")  
 sys.exit()  
  
  
# определение метода (с использованием стандартных функций или без)  
if check\_method == 0:  
 with\_func(A, B)  
elif check\_method == 1:  
 with\_func(A, B)  
else:  
 print("Введите только 0 или 1")