**4./5. uzdevums (apvienots vienā kodā, lai vieglāk salīdzinātu)**

Sastādīt programmu, kas pieprasa lietotājam ievadīt N skaitļus un sakārto tos dilstošā (neaugošā) secībā, izmantojot **naivās** kārtošanas metodi un uz ekrāna izvada sakārtoto skaitļu masīvu un veikto salīdzināšanas skaitu.

Sastādīt programmu, kas pieprasa lietotājam ievadīt N skaitļus un sakārto tos dilstošā (neaugošā) secībā, izmantojot **burbuļa** kārtošanas metodi un uz ekrāna izvada sakārtoto skaitļu masīvu un veikto salīdzināšanas skaitu.

NB! Jāraksta programma, kas veic sakārtošanu tieši dilstošā (neaugošā) secība! Nedrīkst vispirms sakārtot augošā (nedilstošā) secībā un tad nodrukāt masīvu pretējā secībā.

**Kods:**

# Programmas nosaukums: Naivās kārtošanas metode

# 4./5. uzdevums (1MPR05\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas pieprasa lietotājam ievadīt N skaitļus un sakārto tos dilstošā (neaugošā) secībā, izmantojot **naivās** kārtošanas metodi un uz ekrāna izvada sakārtoto skaitļu masīvu un veikto salīdzināšanas skaitu. Sastādīt programmu, kas pieprasa lietotājam ievadīt N skaitļus un sakārto tos dilstošā (neaugošā) secībā, izmantojot **burbuļa** kārtošanas metodi un uz ekrāna izvada sakārtoto skaitļu masīvu un veikto salīdzināšanas skaitu.

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

import numpy

def is\_natural(n):

# Pārbauda vai simbolu virkne ir naturāls skaitlis vai nav

# Ja ir naturāls skaitlis, tad True. Ja nav tad False.

# n - simbolu virkne, kuru pārbauda.

if str(n).isdigit() and float(n) == int(n) and int(n) > 0:

return True

else:

return False

def izvade(x):

# Izvada masīva elementus pēc kārtas līdz pedējam

# x - masīvs

n = len(x)

s = str(x[0])

for i in range(1, n):

s = s + ", " + str(x[i])

print(s)

def izveidot\_masivu\_ar\_garumu(n):

# Izveido masīvu ar noradīto garumu n

# n - naturāls skaitlis

a = numpy.arange(n)

for i in range(n):

b = input("Ievadiet " + str(i) + ".elementu ===> ")

b = is\_whole(b, i)

a[i] = b

return a

def is\_whole(x, i): # Bezgalīgi daudz reizes ievāda

# Pārbauda vai x ir vesels skaitlis

# x - pārbaudama simbolu virkne

# i - elements pēc kārtas

while True:

try:

x = int(x)

except:

x = input("Kļūda! Ievadiet " + str(i) + ".elementu ===> ")

else:

return int(x)

def naivais\_dilstosa(a):

# Naivā kārtošanas metode

# Sakarto masīva elementus dilstoša (neaugoša) secība un

# izvada paveikto salīdzīnašanas skaits, lai sakārtot masīvu. (izmanto naivo kārtošanas metode)

# a - masīvs

skaititajs = 0

n = len(a)

for j in range(n - 1):

min1 = a[j]

imin = j

for i in range(j + 1, n):

skaititajs = skaititajs + 1

if min1 < a[i]:

min1 = a[i]

imin = i

a[imin] = a[j]

a[j] = min1

print(skaititajs) # Izvada paveikto salīdzīnašanas skaits, lai sakārtot masīvu

def burbulis\_uzlabotais(a):

# Burbulis (uzlabotais) kartošanas metode

# Sakarto masīva elementus dilstoša (neaugoša) secība un

# izvada paveikto salīdzīnašanas skaits, lai sakārtot masīvu. Izmanto burbuļa metodi (uzlaboto)

# a - masīvs

skaititajs = 0

n = len(a)

i = n - 1

paz = True

while paz:

paz = False

for j in range(0, i):

skaititajs = skaititajs + 1

if a[j] < a[j + 1]:

paz = True

x = a[j]

a[j] = a[j + 1]

a[j + 1] = x

i = i - 1

print(skaititajs)

# ---------------------------------------------------------

# Galvenā programmas daļa

# ---------------------------------------------------------

m = input("Ievadiet masīva izmēru N ===> ")

while is\_natural(m) == False:

m = input("Masīva izmērs ir naturāls skaitlis!\nIevadiet masīva izmēru N ===> ")

m = int(m)

b = izveidot\_masivu\_ar\_garumu(m)

c = b

print("Paveikts salīdzīnašanas skaits, lai sakārtot masīvu: (naivais dilstoša)")

naivais\_dilstosa(b) # Salidzinašanas skaits

print("Paveikts salīdzīnašanas skaits, lai sakārtot masīvu: (burbulis uzlabotais)")

burbulis\_uzlabotais(c)

print("Sakārtots skaitļu masīvs dilstoša (neaugoša) secība:")

izvade(b)

**Testa piemēri:**

1)

Text

Description automatically generated

2)

Text

Description automatically generated

3)

Graphical user interface, text

Description automatically generated

4)

Text

Description automatically generated

5)

Text

Description automatically generated

6)

Text

Description automatically generated

7)

Text

Description automatically generated

8)

Text

Description automatically generated

9)

Text

Description automatically generated

10)

Text

Description automatically generated

11)

Text

Description automatically generated

12)

Text

Description automatically generated

13)

Text

Description automatically generated

14)

Text

Description automatically generated

Tad pēc šiem testa piemēriem varam apgalvot, ka burbuļa sakārtošanas metode vidēji strādā labāk nekā naiva sakārtošanas metode