**5. praktiskais darbs. 2. semestris**

**1. uzdevums**

Sastādīt programmu, kas pieprasa lietotājam ievadīt N skaitļus un atrod šo skaitļu virknes:

1.1. Vidējā aritmētiskā vērtību, ja zināms, ka to var aprēķināt pēc šādas formulas:

1.2. Vidējā kvadrātiskā vērtību, ja zināms, ka to var aprēķināt pēc šādas formulas:

1.3. Vidējā harmoniskā vērtību, ja zināms, ka to var aprēķināt pēc šādas formulas:

1.4. Vidējā ģeometriskā vērtību, ja zināms, ka to var aprēķināt pēc šādas formulas:

1.5. Vidējās absolūtās jeb vidējās lineārās novirzes vērtību, ja zināms, ka to var aprēķināt pēc šādas formulas:

, kur

1.6. Standartnovirzes jeb vidējās kvadrātiskās novirzes vērtību, ja zināms, ka to var aprēķināt pēc šādas formulas:

, kur

*Zināšanai, standartnovirzi aprēķina kā kvadrātsakni no dispersijas*

**Kods:**

# Programmas nosaukums: Formulas no statistikas

# 1. uzdevums (1MPR05\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas pieprasa lietotājam ievadīt N skaitļus un atrod šo skaitļu virknes

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

import numpy

import math

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 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

while True:

try:

x = int(x)

except:

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

else:

return int(x)

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])

return s

def videjais\_aritmetiskais(x):

# Aprēķina masīva vidējo aritmētisko

# x - masīvs

n = len(x)

t = 0

for i in range(0, n):

t = t + x[i]

return t / n

def videjais\_kvadratiskais(x):

# Aprēķina masīva videjo kvadrātisko

# x - masīvs

n = len(x)

t = 0

for i in range(0, n):

t = t + x[i] \* x[i]

return math.sqrt(t / n)

def videjais\_harmoniskais(x):

# Aprēķina masīva videjo harmonisko

# x - masīvs

n = len(x)

t = 0

for i in range(0, n):

if x[i] == 0:

return "Kļūda! Dalīšana ar 0"

elif is\_natural(x[i]):

t = t + 1 / x[i]

else:

return "Kļūda! Visiem skaitļiem jābut pozitīviem!"

return n / t

def videjais\_geometriskais(x):

# Aprēķina masīva videjo ģeomētrisko

# x - masīvs

n = len(x)

s = 1

for i in x:

s = s \* i

if n % 2 == 0: # Pārbauda vai n-sakne ir pāra skaitlis

if s >= 0: # Ja n-sakne pāra skaitlis, tad pārbaudam vai nav negatīvs, ja ir tad nevaram aprēķināt

k = math.pow(s, (1 / n)) # ja ir pozitīvs vai 0, tad viss ir labi, varam aprēķināt pāra-sakni no pozitīvas vērtības

else:

k = "Nevar aprēķināt reālos skaitļos" # ja ir negatīvs un n-sakne ir pāra skaitlis, tad nevaram to aprēķināt reālos skaitlos

else:

k = numpy.sign(s) \* (numpy.abs(s)) \*\* (1 / n) # ja n-sakne ir nepāra skaitlis, tad aprēķināt to šādi (parasta pow(a,b) funkcija nedarbojas ar tādiem skaitliem)

return k

def videjas\_linearas\_novirzes\_vertiba(x):

# Aprēķina masīva vidējās absolūtās jeb vidējās lineārās novirzes vērtību

# x - masīvs

n = len(x)

t = 0

k = 0

for i in range(0, n):

t = t + x[i]

c = t / n

for i in range(0, n):

k = k + abs(x[i] - c)

return k / n

def standartnovirze(x):

# Aprēķina masīva standartnovirzi

# x - masīvs

n = len(x)

t = 0

k = 0

for i in range(0, n):

t = t + x[i]

c = t / n

for i in range(0, n):

k = k + (x[i] - c) \* (x[i] - c)

return math.sqrt(k / n)

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

# Galvenā programmas daļa

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

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

while is\_natural(n) == False:

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

n = izveidot\_masivu\_ar\_garumu(int(n))

print("\nSkaitļu virknes " + izvade(n))

print("Vidējais aritmētiskais: " + str(videjais\_aritmetiskais(n)))

print("Vidējais kvadratiskais: " + str(videjais\_kvadratiskais(n)))

print("Vidējais harmoniskais: " + str(videjais\_harmoniskais(n)))

print("Vidējais ģeomētriskais: " + str(videjais\_geometriskais(n)))

print("Vidējās absolūtās jeb vidējās lineārās novirzes vērtība: " + str(videjas\_linearas\_novirzes\_vertiba(n)))

print("Standartnovīrze: " + str(standartnovirze(n)))

**Testa piemēri:**

1)

Text

Description automatically generated

2)

Text

Description automatically generated

3)

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 with medium confidence

**2. uzdevums**

Sastādīt programmu, kas pieprasa lietotājam ievadīt divus vienāda izmēra datu masīvus ar vērtībām un to skaitu un aprēķina vidējā svētā vērtību, ja zināms, ka to var aprēķināt pēc šādas formulas:

**Kods:**

# Programmas nosaukums: Vidēja svērta vērtība

# 2. uzdevums (1MPR05\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas pieprasa lietotājam ievadīt divus vienāda izmēra datu masīvus ar novērojumiem un aprēķina lineārās korelācijas koeficienta vērtību starp šo abu datu masīvi atbilstošiem elementiem, ja zināms, ka to var aprēķināt pēc šādas formulas:

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

import numpy

import math

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 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

while True:

try:

x = int(x)

except:

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

else:

return int(x)

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 videjais\_aritmetiskais(x):

# Aprēķina masīva vidējo aritmētisko

# x - masīvs

n = len(x)

t = 0

for i in range(0, n):

t = t + x[i]

return t / n

def videja\_sversta\_vertiba(x, y):

# Aprēķina masīvu vidējo svērsto vērtību

# x - pirmais masīvs

# y - otrais masīvs

n = len(x)

t = 0

z = 0

for i in range(0, n):

t = t + x[i] \* y[i]

for i in range(0, n):

z = z + y[i]

return t / z

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

# 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)

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

print("Ievadiet elementu svarus!")

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

print("\nPirmas virknes elementi:")

izvade(t)

print("Otrās virknes elementi (svars):")

izvade(c)

print("Vidēja svērta vērtība ir:")

print(videja\_sversta\_vertiba(t, c))

**Testa piemēri:**

1)

Text

Description automatically generated

2)

Text

Description automatically generated

3)

Text

Description automatically generated

4)

Text

Description automatically generated

5)

Calendar

Description automatically generated with medium confidence

6)

Text

Description automatically generated

7)

Text

Description automatically generated

**3. uzdevums**

Sastādīt programmu, kas pieprasa lietotājam ievadīt divus vienāda izmēra datu masīvus ar novērojumiem un aprēķina lineārās korelācijas koeficienta vērtību starp šo abu datu masīvi atbilstošiem elementiem, ja zināms, ka to var aprēķināt pēc šādas formulas:

,kur *, ,*

**Kods:**

# Programmas nosaukums: Lineārās korelācijas koeficienta vērtība

# 3. uzdevums (1MPR05\_Vladislavs\_Babaņins)

# Uzdevuma formulējums: Sastādīt programmu, kas pieprasa lietotājam ievadīt divus vienāda izmēra datu masīvus ar novērojumiem un aprēķina lineārās korelācijas koeficienta vērtību starp šo abu datu masīvi atbilstošiem elementiem, ja zināms, ka to var aprēķināt pēc šādas formulas:

# Programmas autors: Vladislavs Babaņins

# Versija 1.0

import numpy

import math

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 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

while True:

try:

x = int(x)

except:

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

else:

return int(x)

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 videjais\_aritmetiskais(x):

# Aprēķina masīva vidējo aritmētisko

# x - masīvs

n = len(x)

t = 0

for i in range(0, n):

t = t + x[i]

return t / n

def linearas\_korelacijas\_koeficients(x, y):

# Aprēķina masīvu linearas korelacijas koeficientu

# x - pirmais masīvs

# y - otrais masīvs

n = len(x)

vidx = videjais\_aritmetiskais(x)

vidy = videjais\_aritmetiskais(y)

s = 0

tx = 0

ty = 0

for i in range(n):

s = s + (x[i] - vidx) \* (y[i] - vidy)

tx = tx + (x[i] - vidx) \* (x[i] - vidx)

ty = ty + (y[i] - vidy) \* (y[i] - vidy)

p = math.sqrt(tx \* ty)

if p != 0:

return s / p

else:

return "Kļūda! Dalīšana ar 0"

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

# 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)

print("Ievadiet pirmā masīva skaitļus!")

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

print("Ievadiet otrā masīva skaitļus!")

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

print("\nPirmā skaitļu virkne:")

izvade(t)

print("Otrā skaitļu virkne:")

izvade(c)

print("Lineāras korelācijas koeficients:")

print(linearas\_korelacijas\_koeficients(t, c))

**Testa piemēri:**

1)

Text

Description automatically generated

2)

Text

Description automatically generated

3)

Text

Description automatically generated

4)

A picture containing text

Description automatically generated

5)

A picture containing calendar

Description automatically generated

6)

Text

Description automatically generated

7)

Text

Description automatically generated

8)

Text

Description automatically generated

9)

Text

Description automatically generated with medium confidence

10)

Text

Description automatically generated

**4. uzdevums**

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.

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. 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.

# 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):

if min1 < a[i]:

skaititajs = skaititajs + 1 # skaititājs

min1 = a[i]

imin = i

a[imin] = a[j]

a[j] = min1

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

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

# 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)

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

naivais\_dilstosa(b) # Salidzinašanas skaits

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

izvade(b)

**Testa piemēri:**

1)

Text

Description automatically generated

2)

Text

Description automatically generated with medium confidence

3)

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)

Shape

Description automatically generated with low confidence

11)

Text

Description automatically generated

12)

A picture containing text

Description automatically generated

13)

Text

Description automatically generated

14)

Text

Description automatically generated with medium confidence

**5. uzdevums**

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: Burbuļa kārtošanas metode

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

# Uzdevuma formulējums: 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 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):

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

skaititajs = skaititajs + 1

paz = True

x = a[j]

a[j] = a[j + 1]

a[j + 1] = x

i = i - 1

print(skaititajs)

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)

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

# 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)

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

burbulis\_uzlabotais(b) # Salidzinašanas skaits

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

izvade(b)

**Testa piemēri:**

1)

Text

Description automatically generated

2)

Text

Description automatically generated

3)

A picture containing text, keyboard

Description automatically generated

4)

Text

Description automatically generated

5)

Text

Description automatically generated

6)

Text

Description automatically generated

7)

Text

Description automatically generated with medium confidence