**1MPR05\_Simona\_Bļinova sb24037**

Visos uzdevumos lietoju savu moduli ar funkcijām. (programmu mapē *funkcijas.py*)

**1.uzdevums**

Programma, kas veic nejauši aizpildīta masīva kārtošanu dilstoša secībā piecos veidos.

**Kods:**

Funkcijas no *funkcijas.py*:

# Funkcija pārbaudei, lai skaitlis būtu naturāls skaitlis

def naturals\_skaitlis(a):

# meģinājumu skaita skaitītājs (3 meģinājumi ierakstam)

meginajumi = 1

while meginajumi <= 3:

try:

a = int(a)

if a > 0:

return int(a)

else:

raise Exception

except:

if meginajumi < 3:

meginajumi += 1

a = input('Ievadiet naturālo skaitli vēlreiz --> ')

else:

print('Programma beidz darbību!')

exit()

# Masīva izveides funkcija

def masiva\_izveide(n):

return numpy.arange(n)

*Simona\_Blinova\_1MPR05\_1.py*:

import funkcijas

import random

import copy

import math

''' Funkciju bloks '''

def aizpildit\_masivu(a, elementu\_skaits):

for i in range(elementu\_skaits, 2, -1):

b = random.randint(1, i-1)

c = a[i]

a[i] = a[b]

a[b] = c

return a

def naiva\_kartosana(a):

garums = len(a)

skaititajs = 0

for i in range(1, garums-1):

maksimala\_vertiba = a[i]

max\_vertibas\_indekss = i

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

skaititajs += 1

if maksimala\_vertiba < a[j]:

maksimala\_vertiba = a[j]

max\_vertibas\_indekss = j

a[max\_vertibas\_indekss] = a[i]

a[i] = maksimala\_vertiba

a[0] = skaititajs

return a

def bubble\_metode(a):

garums = len(a)

skaititajs = 0

atkartojumi = garums - 1

pazime = True

while pazime:

pazime = False

for j in range(1, atkartojumi):

skaititajs += 1

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

pazime = True

b = a[j]

a[j] = a[j+1]

a[j+1] = b

atkartojumi -= 1

a[0] = skaititajs

return a

def atspoles\_metode(a):

garums = len(a)

skaititajs = 0

for i in range(2, garums):

skaititajs += 1

if a[i-1] < a[i]:

for j in range(i, 1, -1):

skaititajs += 1

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

b = a[j]

a[j] = a[j-1]

a[j-1] = b

else:

break

a[0] = skaititajs

return a

def ievietosanas\_metode(a):

garums = len(a)

skaititajs = 0

for i in range(2, garums):

skaititajs += 1

x = a[i]

j = i

while j > 1 and a[j-1] < x:

skaititajs += 1

a[j] = a[j-1]

j -= 1

a[j] = x

a[0] = skaititajs

return a

def shell\_metode(a):

garums = len(a)

skaititajs = 0

solis = (3\*\*math.floor(math.log(2\*n+1, 3))-1)//2

while solis >= 1:

for i in range(1, solis+1):

for j in range(solis+i, garums, solis):

skaititajs += 1

if a[j-solis] < a[j]:

b = a[j]

k = j

while a[k-solis] < b:

skaititajs += 1

a[k] = a[k-solis]

k -= solis

if k == i:

break

a[k] = b

solis = (solis - 1) // 3

a[0] = skaititajs

return a

def izvade(a):

garums = len(a)

virkne = ''

for i in range(1, garums):

if i == garums - 1:

virkne += str(a[i])

else:

virkne += str(a[i]) + ', '

return f'{a[0]} salīdzināšanas - {virkne}'

''' Masīva izveides un aizpildes bloks '''

n = input('Ievadiet masīva izmēru --> ')

n = funkcijas.naturals\_skaitlis(n)

masivs = funkcijas.masiva\_izveide(n+1)

masivs = aizpildit\_masivu(masivs, n)

print(masivs)

'''Masīva kopijas kārtošana piecos veidos'''

masivs\_naiva = naiva\_kartosana(copy.deepcopy(masivs))

# print(masivs\_naiva)

masivs\_bubble = bubble\_metode(copy.deepcopy(masivs))

# print(masivs\_bubble)

masivs\_atspole = atspoles\_metode(copy.deepcopy(masivs))

# print(masivs\_atspole)

masivs\_ievietosana = ievietosanas\_metode(copy.deepcopy(masivs))

# print(masivs\_ievietosana)

masivs\_shell = shell\_metode(copy.deepcopy(masivs))

# print(masivs\_shell)

''' Rezultātu izvade '''

print(f'Naivā kārtošanas metode: {izvade(masivs\_naiva)}')

print(f'Burbuļa kārtošanas metode: {izvade(masivs\_bubble)}')

print(f'Atspoles kārtošanas metode: {izvade(masivs\_atspole)}')

print(f'Ievietošanas kārtošanas metode: {izvade(masivs\_ievietosana)}')

print(f'Šella kārtošanas metode: {izvade(masivs\_shell)}')

Testa piemērs(1)

A screenshot of a computer code

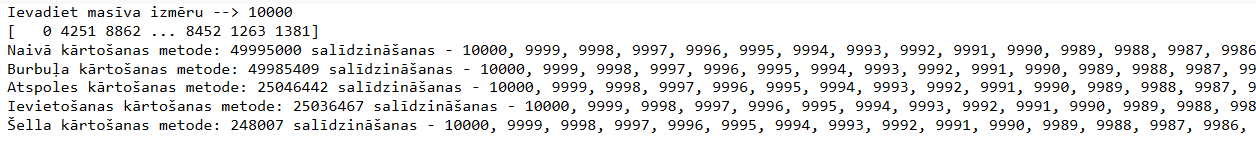
AI-generated content may be incorrect.

Testa piemērs(2)

A number and numbers on a white background

AI-generated content may be incorrect.

Testa piemērs(3) (dators ļoti ilgi skaitīja 100000 un beigas nevarēja pārveidot lielo int ar salīdzīnājumu skaitu, tāpēc paņemu 10000)



**2.uzdevums**

Programma, kas atrod masīvā ievadītas skaitļu kopas mediānu.

**Kods:**

Funkcijas no *funkcijas.py*:

# Funkcija pārbaudei, lai skaitlis būtu naturāls skaitlis

def naturals\_skaitlis(a):

# meģinājumu skaita skaitītājs (3 meģinājumi ierakstam)

meginajumi = 1

while meginajumi <= 3:

try:

a = int(a)

if a > 0:

return int(a)

else:

raise Exception

except:

if meginajumi < 3:

meginajumi += 1

a = input('Ievadiet naturālo skaitli vēlreiz --> ')

else:

print('Programma beidz darbību!')

exit()

# Masīva izveides funkcija

def masiva\_izveide(n):

return numpy.arange(n)

# Datu ievades masīvā funkcija

def datu\_ievade\_masiva(a):

garums = len(a)

for i in range(garums):

vertiba = input('Ievadiet vērtību --> ')

# pārbaude lai reāls, lai varētu atrast min un max

vertiba = reals\_skaitlis(vertiba)

a[i] = vertiba

return a

*Simona\_Blinova\_1MPR05\_2.py*:

import funkcijas

def ievietosanas\_metode(a):

garums = len(a)

for i in range(1, garums):

if a[i-1] > a[i]:

x = a[i]

j = i

while a[j-1] > x:

a[j] = a[j-1]

j -= 1

if j == 0:

break

a[j] = x

return a

n = input('Ievadiet masīva garumu --> ')

n = funkcijas.naturals\_skaitlis(n)

masivs = funkcijas.masiva\_izveide(n)

masivs = funkcijas.datu\_ievade\_masiva(masivs)

#print(masivs)

masivs = ievietosanas\_metode(masivs)

print(masivs)

videjais\_indekss = n // 2

if n % 2 == 0:

vertibu\_summa = masivs[videjais\_indekss] + masivs[videjais\_indekss-1]

mediana = vertibu\_summa / 2

else:

mediana = masivs[videjais\_indekss]

print(f'Kopas mediāna: {mediana}')

Testa piemērs(1)

A computer screen shot of a number

AI-generated content may be incorrect.

Testa piemērs(2)

A screenshot of a computer

AI-generated content may be incorrect.

Testa piemērs(3)

A white background with black text

AI-generated content may be incorrect.

**PU**

Programma, kas atrod masīvā ievadītas skaitļu kopas modu.

**Kods:**

Funkcijas no *funkcijas.py*:

# Funkcija pārbaudei, lai skaitlis būtu naturāls skaitlis

def naturals\_skaitlis(a):

# meģinājumu skaita skaitītājs (3 meģinājumi ierakstam)

meginajumi = 1

while meginajumi <= 3:

try:

a = int(a)

if a > 0:

return int(a)

else:

raise Exception

except:

if meginajumi < 3:

meginajumi += 1

a = input('Ievadiet naturālo skaitli vēlreiz --> ')

else:

print('Programma beidz darbību!')

exit()

# Masīva izveides funkcija

def masiva\_izveide(n):

return numpy.arange(n)

# Datu ievades masīvā funkcija

def datu\_ievade\_masiva(a):

garums = len(a)

for i in range(garums):

vertiba = input('Ievadiet vērtību --> ')

# pārbaude lai reāls, lai varētu atrast min un max

vertiba = reals\_skaitlis(vertiba)

a[i] = vertiba

return a

*Simona\_Blinova\_1MPR05\_PU.py*:

import funkcijas

def ievietosanas\_metode(a):

garums = len(a)

for i in range(1, garums):

if a[i-1] > a[i]:

x = a[i]

j = i

while a[j-1] > x:

a[j] = a[j-1]

j -= 1

if j == 0:

break

a[j] = x

return a

def moda(a):

garums = len(a)

saraksts = []

i = 1

skaits = 1

moda = a[i-1]

while i < garums:

if moda != a[i]:

saraksts.append((moda, skaits))

moda = a[i]

skaits = 1

else:

skaits += 1

i += 1

saraksts.append((moda, skaits))

return saraksts

n = input('Ievadiet masīva garumu --> ')

n = funkcijas.naturals\_skaitlis(n)

masivs = funkcijas.masiva\_izveide(n)

masivs = funkcijas.datu\_ievade\_masiva(masivs)

# print(masivs)

sakartots\_masivs = ievietosanas\_metode(masivs)

print(sakartots\_masivs)

modas = moda(sakartots\_masivs)

# print(modas)

garums = len(modas)

if garums == n:

print('Moda netika atrasta')

else:

max\_moda = modas[0][1]

moda\_indekss = 0

for i in range(1, garums):

if max\_moda < modas[i][1]:

max\_moda = modas[i][1]

moda\_indekss = i

modu\_saraksts = []

modu\_saraksts.append(modas[moda\_indekss][0])

for i in range(garums):

if i != moda\_indekss:

if max\_moda == modas[i][1]:

modu\_saraksts.append(modas[i][0])

# print(modu\_saraksts)

modu\_virkne = ''

for i in range(len(modu\_saraksts)):

modu\_virkne += str(modu\_saraksts[i]) + ' '

print('Skaitļu kopas moda: ' + modu\_virkne)

Testa piemērs(1)

A screenshot of a computer

AI-generated content may be incorrect.

Testa piemērs(2)

A screenshot of a computer code

AI-generated content may be incorrect.

Testa piemērs(3)

A white background with black text

AI-generated content may be incorrect.