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)

```
Ievadiet masīva izmēru --> 10
[ 0  5  7  8  2  3 10  4  6  1  9]
Naivā kārtošanas metode: 45 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
Burbuļa kārtošanas metode: 45 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
Atspoles kārtošanas metode: 36 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
Ievietošanas kārtošanas metode: 32 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
Šella kārtošanas metode: 32 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
```

Testa piemērs(2)

```
Ievadiet masīva izmēru --> 1000
[  0 320 167 ... 424 688  42]
Naivā kārtošanas metode: 499500 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 994, 993, 992, 991, 990, 989, 
Burbuļa kārtošanas metode: 498554 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 994, 993, 992, 991, 990, 989,
Atspoles kārtošanas metode: 245882 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 994, 993, 992, 991, 990, 98
Ievietošanas kārtošanas metode: 244899 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 994, 993, 992, 991, 990,
Šella kārtošanas metode: 14395 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 994, 993, 992, 991, 990, 989, 98
```

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

```
Ievadiet masīva izmēru --> 10000
[   0 4251 8862 ... 8452 1263 1381]
Naivā kārtošanas metode: 49995000 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 9995, 9994, 9993, 9992, 9991, 9990, 9989, 9988, 9987, 9986
Burbuļa kārtošanas metode: 49985409 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 9995, 9994, 9993, 9992, 9991, 9990, 9989, 9988, 9987, 99
Atspoles kārtošanas metode: 25046442 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 9995, 9994, 9993, 9992, 9991, 9990, 9989, 9988, 9987, 9
Ievietošanas kārtošanas metode: 25036467 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 9995, 9994, 9993, 9992, 9991, 9990, 9989, 9988, 998
Šella kārtošanas metode: 248007 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 9995, 9994, 9993, 9992, 9991, 9990, 9989, 9988, 9987, 9986,
```

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)

```
Ievadiet masīva garumu --> 5
Ievadiet vērtību --> 3
Ievadiet vērtību --> -10
Ievadiet vērtību --> 5
Ievadiet vērtību --> 1
Ievadiet vērtību --> 0
[-10   0   1   3   5]
Kopas mediāna: 1
```

Testa piemērs(2)

```
Ievadiet masīva garumu --> 8
Ievadiet vērtību --> -3
Ievadiet vērtību --> 2.5
Ievadiet vērtību --> -3.33
Ievadiet vērtību --> 23
Ievadiet vērtību --> 12
Ievadiet vērtību --> 6
Ievadiet vērtību --> 9
Ievadiet vērtību --> 13
[-3 -3  2  6  9 12 13 23]
Kopas mediāna: 7.5
```

Testa piemērs(3)

```
Ievadiet masīva garumu --> 1
Ievadiet vērtību --> 5
[5]
Kopas mediāna: 5
```

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)

```
Ievadiet masīva garumu --> 5
Ievadiet vērtību --> -1
Ievadiet vērtību --> 3
Ievadiet vērtību --> 2
Ievadiet vērtību --> 0
Ievadiet vērtību --> 2
[-1  0  2  2  3]
Skaitļu kopas moda: 2
```

Testa piemērs(2)

```
Ievadiet masīva garumu --> 8
Ievadiet vērtību --> 2
Ievadiet vērtību --> 2
Ievadiet vērtību --> 6
Ievadiet vērtību --> 7
Ievadiet vērtību --> 4
Ievadiet vērtību --> 2
Ievadiet vērtību --> 4
Ievadiet vērtību --> 4
[2 2 2 4 4 4 6 7]
Skaitļu kopas moda: 2 4
```

Testa piemērs(3)

```
Ievadiet masīva garumu --> 4
Ievadiet vērtību --> 1
Ievadiet vērtību --> 2
Ievadiet vērtību --> 3
Ievadiet vērtību --> 4
[1 2 3 4]
Moda netika atrasta
```