1MPR06_Simona_Bļinova sb24037

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

1.uzdevums

Programma, kas veic masīva kārtošanu dilstoša secībā ar trim metodēm.

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)
```

1MPR06_1_Simona_Blinova.py:

```
import funkcijas
import random
import copy
import numpy

''' 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 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}'

##### Ātrā jeb Hoara kārtošana #####

def hoara(a, sakuma_vertiba, beigu_vertiba):
    if sakuma_vertiba < beigu_vertiba:
        i = sakuma_vertiba
        j = beigu_vertiba
        solis = -1
        lv = True

        while i != j:
            a[0] += 1
            if lv == (a[i] < a[j]):
                x = a[i]
                a[i] = a[j]
                a[j] = x

                x = i
                i = j
                j = x

                lv = not lv
                solis = -solis

            j += solis

        hoara(a, sakuma_vertiba, i - 1)
```

```
        hoara(a, i + 1, beigu_vertiba)

    return a

##### Saliešanas metode #####

def saliesana(a, sakuma_vertiba, beigu_vertiba):
    garums = len(a)
    b = funkcijas.masiva_izveide(garums)
    if sakuma_vertiba < beigu_vertiba:
        vidus_vertiba = (sakuma_vertiba + beigu_vertiba) // 2
        saliesana(a, sakuma_vertiba, vidus_vertiba)
        saliesana(a, vidus_vertiba + 1, beigu_vertiba)

        for i in range(sakuma_vertiba, beigu_vertiba + 1):
            b[i] = a[i]

        i = sakuma_vertiba
        j = vidus_vertiba + 1
        k = sakuma_vertiba

        while i <= vidus_vertiba and j <= beigu_vertiba:
            a[0] += 1

            if b[i] > b[j]:
                a[k] = b[i]
                i += 1
            else:
                a[k] = b[j]
                j += 1
            k += 1

        if j > beigu_vertiba:
            while i <= vidus_vertiba:
                a[k] = b[i]
                i += 1
                k += 1

    return a

##### Timsort #####

def iev_met(a, sakums, beigas):
    for i in range(sakums+1, beigas+1):
```

```
        elem = a[i]
        j = i-1
        while j >= sakums and a[j] < elem:
            a[0] += 1
            a[j+1] = a[j]
            j = j - 1
        a[j+1] = elem


def apvienosana(a, sakums, vidus, beigas):
    g1 = vidus - sakums + 1
    g2 = beigas - vidus
    viens_a = numpy.copy(a[sakums:sakums+g1])
    otrs_a = numpy.copy(a[vidus+1:vidus+1+g2])
    i = 0
    j = 0
    k = sakums
    while i < g1 and j < g2:
        a[0] += 1
        if viens_a[i] >= otrs_a[j]:
            a[k] = viens_a[i]
            i += 1
        else:
            a[k] = otrs_a[j]
            j += 1
        k += 1

    while i < g1:
        a[k] = viens_a[i]
        i += 1
        k += 1

    while j < g2:
        a[k] = otrs_a[j]
        j += 1
        k += 1


def tim_sort(a):
    INTERVALA_GARUMS = 2
    n = len(a)

    #Atsevišķu intervālu sakārtošana
    for i in range(1, n, INTERVALA_GARUMS):
```

```
        iev_met(a, i, min(i+INTERVALA_GARUMS - 1, n-1))

    izmers = INTERVALA_GARUMS
    while izmers < n:
        for sakums in range(1, n, 2*izmers):
            vidus = min(n-1, sakums+izmers-1)
            beigas = min(n-1, sakums + 2*izmers-1)
            if vidus < beigas:
                apvienosana(a, sakums, vidus, beigas)
        izmers = izmers * 2

    return a


''' 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 trīs veidos'''

hoara_masivs = hoara(copy.deepcopy(masivs), 1, len(masivs)-1)

saliesanas_masivs = saliesana(copy.deepcopy(masivs), 1, len(masivs)-1)

timsort_masivs = tim_sort(copy.deepcopy(masivs))

''' Rezultātu izvade '''

print(f'Ātrā jeb Hoara kārtošanas metode: {izvade(hoara_masivs)}')
print(f'Saliešanas metode: {izvade(saliesanas_masivs)}')
print(f'Timsort metode: {izvade(timsort_masivs)}')
```

Testa piemērs(1)

```
Ievadiet masīva izmēru --> 10
[ 0  5  8  9  3  7  2  4 10  1  6]
Ātrā jeb Hoara kārtošanas metode: 24 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
Saliešanas metode: 22 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
Timsort metode: 25 salīdzināšanas - 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
```

Testa piemērs(2)

```
Ievadiet masīva izmēru --> 1000
[  0 247 349 ...  47 931 706]
Ātrā jeb Hoara kārtošanas metode: 10273 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 994, 9
Saliešanas metode: 8704 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 994, 993, 992, 991, 99
Timsort metode: 8465 salīdzināšanas - 1000, 999, 998, 997, 996, 995, 994, 993, 992, 991, 990,
```

Testa piemērs(3)

```
Ievadiet masīva izmēru --> 100000
[    0 74455 61570 ... 78843 28922 38494]
Ātrā jeb Hoara kārtošanas metode: 1997886 salīdzināšanas - 100000, 99999, 99998, 99997, 99996, 99995, 9
[omitting some output]
Saliešanas metode: 1536219 salīdzināšanas - 100000, 99999, 99998, 99997, 99996, 99995, 99994, 99993, 99
[omitting some output]
 71920, 71919, 71918, 71917, 71916, 71915, 71914, 71913, 71912, 71911, 71910, 71909, 71908, 71907, 7190
[omitting some output]
877, 57876, 57875, 57874, 57873, 57872, 57871, 57870, 57869, 57868, 57867, 57866, 57865, 57864, 57863,
[omitting some output]
Timsort metode: 1541468 salīdzināšanas - 100000, 99999, 99998, 99997, 99996, 99995, 99994, 99993, 99992
[omitting some output]
 90644, 90643, 90642, 90641, 90640, 90639, 90638, 90637, 90636, 90635, 90634, 90633, 90632, 90631, 9063
[omitting some output]
20, 71919, 71918, 71917, 71916, 71915, 71914, 71913, 71912, 71911, 71910, 71909, 71908, 71907, 71906, 7
[omitting some output]
```

2.uzdevums

Programma, kas veic trīs dažāda izmēra masīvu apvienošanu.

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)

# Funkcija pārbaudei, lai skaitlis būtu reāls skaitlis
def reals_skaitlis(a):
  # meģinajumu skaitītājs (3 meģinājumi pareizi ievadīt)
  meginajumi = 1
  while meginajumi <= 3:
    try:
      a = float(a)
      return float(a)
    except:
      if meginajumi < 3:
        meginajumi += 1
        a = input('Ievadiet reālo skaitli vēlreiz --> ')
      else:
        print('Programma beidz darbību!')
        exit()
```

1MPR06_2_Simona_Blinova.py:

```
import funkcijas
import math

def izveidot_masivu():
  n = input('Ievadiet masīva izmēru --> ')
  n = funkcijas.naturals_skaitlis(n)

  a = funkcijas.masiva_izveide(n)

  for i in range(n):
    skaitlis = input('Ievadiet skaitli masīvā --> ')
    skaitlis = funkcijas.reals_skaitlis(skaitlis)
    a[i] = skaitlis

  return a


def shell_metode_augosa(a):
  garums = len(a)

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

  while solis >= 1:
    for i in range(0, solis):
```

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

            if a[j-solis] > a[j]:
                b = a[j]
                k = j

                while a[k-solis] > b:
                    a[k] = a[k-solis]
                    k -= solis
                    if k == i:
                        break

                a[k] = b

        solis = (solis - 1) // 3

    return a


def shell_metode_dilstosa(a):
    garums = len(a)

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

    while solis >= 1:
        for i in range(0, solis):
            for j in range(solis+i, garums, solis):

                if a[j-solis] < a[j]:
                    b = a[j]
                    k = j

                    while a[k-solis] < b:
                        a[k] = a[k-solis]
                        k -= solis
                        if k == i:
                            break

                    a[k] = b

        solis = (solis - 1) // 3

    return a
```

```
def apvienosana(a, b):
    garums_a = len(a)
    garums_b = len(b)
    garums_c = garums_a + garums_b

    c = funkcijas.masiva_izveide(garums_c)

    ia = indeksa_noteiksana(a)
    ib = indeksa_noteiksana(b)
    ic = 0

    if ia != 0:
        a = shell_metode_augosa(a)
        ia = 0

    if ib != 0:
        b = shell_metode_augosa(b)
        ib = 0

    while ia < garums_a and ib < garums_b:
        if a[ia] < b[ib]:
            c[ic] = a[ia]
            ia += 1
        else:
            c[ic] = b[ib]
            ib += 1
        ic += 1

    if ia < garums_a:
        for i in range(ia, garums_a):
            c[ic] = a[i]
            ic += 1
    else:
        for i in range(ib, garums_b):
            c[ic] = b[i]
            ic += 1

    return c


def indeksa_noteiksana(a):
    garums = len(a)
```

```
    if a[0] < a[garums-1]:
        i = 0
    else:
        i = garums-1

    return i


masivs1 = izveidot_masivu()
print(masivs1)

masivs2 = izveidot_masivu()
print(masivs2)

masivs3 = izveidot_masivu()
print(masivs3)

masivs1 = shell_metode_augosa(masivs1)
masivs2 = shell_metode_dilstosa(masivs2)
masivs3 = shell_metode_augosa(masivs3)

print(masivs1, masivs2, masivs3)

masivs12 = apvienosana(masivs1, masivs2)
masivs = apvienosana(masivs12, masivs3)
print(masivs)
```

Testa piemērs(1)

```
Ievadiet masīva izmēru --> 5
Ievadiet skaitli masīvā --> 5
Ievadiet skaitli masīvā --> 4
Ievadiet skaitli masīvā --> 3
Ievadiet skaitli masīvā --> 2
Ievadiet skaitli masīvā --> 1
[5 4 3 2 1]
Ievadiet masīva izmēru --> 3
Ievadiet skaitli masīvā --> 1
Ievadiet skaitli masīvā --> 2
Ievadiet skaitli masīvā --> 3
[1 2 3]
Ievadiet masīva izmēru --> 2
Ievadiet skaitli masīvā --> 3
Ievadiet skaitli masīvā --> 2
[3 2]
[1 2 3 4 5] [3 2 1] [2 3]
[1 1 2 2 2 3 3 3 4 5]
```

Testa piemērs(2)

```
Ievadiet masīva izmēru --> 3
Ievadiet skaitli masīvā --> -3
Ievadiet skaitli masīvā --> 67
Ievadiet skaitli masīvā --> 3
[-3 67  3]
Ievadiet masīva izmēru --> 4
Ievadiet skaitli masīvā --> 12
Ievadiet skaitli masīvā --> 3
Ievadiet skaitli masīvā --> 89
Ievadiet skaitli masīvā --> 0
[12  3 89  0]
Ievadiet masīva izmēru --> 5
Ievadiet skaitli masīvā --> -54
Ievadiet skaitli masīvā --> 657
Ievadiet skaitli masīvā --> 23
Ievadiet skaitli masīvā --> 6
Ievadiet skaitli masīvā --> 7
[-54 657  23   6   7]
[-3  3 67] [89 12  3  0] [-54   6   7  23 657]
[-54  -3   0   3   3   6   7  12  23  67  89 657]
```

Testa piemērs(3)

```
Ievadiet masīva izmēru --> 2
Ievadiet skaitli masīvā --> -3
Ievadiet skaitli masīvā --> -67
[ -3 -67]
Ievadiet masīva izmēru --> 7
Ievadiet skaitli masīvā --> 1
Ievadiet skaitli masīvā --> 4
Ievadiet skaitli masīvā --> 67
Ievadiet skaitli masīvā --> 32
Ievadiet skaitli masīvā --> 5433
Ievadiet skaitli masīvā --> 24
Ievadiet skaitli masīvā --> -90
[   1    4   67   32 5433   24  -90]
Ievadiet masīva izmēru --> 5
Ievadiet skaitli masīvā --> 11
Ievadiet skaitli masīvā --> 2
Ievadiet skaitli masīvā --> 421
Ievadiet skaitli masīvā --> 34
Ievadiet skaitli masīvā --> 0
[ 11   2 421  34   0]
[-67  -3] [5433   67   32   24    4    1  -90] [  0   2  11  34 421]
[ -90  -67   -3    0    1    2    4   11   24   32   34   67  421 5433]
```

PU

Programma, kas veic trīs dažāda izmēra masīvu apvienošanu.

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)

# Funkcija pārbaudei, lai skaitlis būtu reāls skaitlis
def reals_skaitlis(a):
    # meģinajumu skaitītājs (3 meģinājumi pareizi ievadīt)
    meginajumi = 1
    while meginajumi <= 3:
        try:
            a = float(a)
            return float(a)
        except:
            if meginajumi < 3:
                meginajumi += 1
                a = input('Ievadiet reālo skaitli vēlreiz --> ')
            else:
                print('Programma beidz darbību!')
                exit()
```

1MPR06_PU_Simona_Blinova.py:

```
import funkcijas
import math
import random

def izveidot_masivu():
    n = input('Ievadiet masīva izmēru --> ')
```

```
    n = funkcijas.naturals_skaitlis(n)

    a = funkcijas.masiva_izveide(n)

    for i in range(n):
        skaitlis = input('Ievadiet skaitli masīvā --> ')
        skaitlis = funkcijas.reals_skaitlis(skaitlis)
        a[i] = skaitlis

    return a


def shell_metode_augosa(a):
    garums = len(a)

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

    while solis >= 1:
        for i in range(0, solis):
            for j in range(solis+i, garums, solis):

                if a[j-solis] > a[j]:
                    b = a[j]
                    k = j

                    while a[k-solis] > b:
                        a[k] = a[k-solis]
                        k -= solis
                        if k == i:
                            break

                    a[k] = b

        solis = (solis - 1) // 3

    return a


def shell_metode_dilstosa(a):
    garums = len(a)

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

    while solis >= 1:
```

```
        for i in range(0, solis):
            for j in range(solis+i, garums, solis):

                if a[j-solis] < a[j]:
                    b = a[j]
                    k = j

                    while a[k-solis] < b:
                        a[k] = a[k-solis]
                        k -= solis
                        if k == i:
                            break

                    a[k] = b

        solis = (solis - 1) // 3

    return a


def apvienosana(a, b):
    garums_a = len(a)
    garums_b = len(b)
    garums_c = garums_a + garums_b

    c = funkcijas.masiva_izveide(garums_c)

    # Nosaka indeksu (vai ir augoša vai dilstoša)
    ia = indeksa_noteiksana(a)
    ib = indeksa_noteiksana(b)
    ic = 0

    # pārveido masīvus par augošu, ja noteikts, ka ir dilstoša
    if ia != 0:
        a = shell_metode_augosa(a)
        ia = 0

    if ib != 0:
        b = shell_metode_augosa(b)
        ib = 0

    while ia < garums_a and ib < garums_b:
        if a[ia] < b[ib]:
            c[ic] = a[ia]
```

```
            ia += 1
        else:
            c[ic] = b[ib]
            ib += 1
        ic += 1

    if ia < garums_a:
        for i in range(ia, garums_a):
            c[ic] = a[i]
            ic += 1
    else:
        for i in range(ib, garums_b):
            c[ic] = b[i]
            ic += 1

    return c


def indeksa_noteiksana(a):
    garums = len(a)

    if a[0] < a[garums-1]:
        i = 0
    else:
        i = garums-1

    return i


def random_sort(a):
    # Ja 0, tad dilstoša
    # Ja 1, tad augoša
    pazime = random.randint(0, 1)

    if pazime == 0:
        return shell_metode_dilstosa(a)
    else:
        return shell_metode_augosa(a)


masivs1 = izveidot_masivu()
print(masivs1)

masivs2 = izveidot_masivu()
```

```
print(masivs2)

masivs3 = izveidot_masivu()
print(masivs3)

masivs1 = random_sort(masivs1)
masivs2 = random_sort(masivs2)
masivs3 = random_sort(masivs3)

print(masivs1, masivs2, masivs3)

masivs12 = apvienosana(masivs1, masivs2)
masivs = apvienosana(masivs12, masivs3)
print(masivs)
```

Testa piemērs(1)

```
Ievadiet masīva izmēru --> 4
Ievadiet skaitli masīvā --> 0
Ievadiet skaitli masīvā --> 234
Ievadiet skaitli masīvā --> -45
Ievadiet skaitli masīvā --> 13
[  0 234 -45  13]
Ievadiet masīva izmēru --> 3
Ievadiet skaitli masīvā --> 1
Ievadiet skaitli masīvā --> 2
Ievadiet skaitli masīvā --> 3
[1 2 3]
Ievadiet masīva izmēru --> 7
Ievadiet skaitli masīvā --> -9
Ievadiet skaitli masīvā --> 0
Ievadiet skaitli masīvā --> 68
Ievadiet skaitli masīvā --> 43
Ievadiet skaitli masīvā --> 23
Ievadiet skaitli masīvā --> 1
Ievadiet skaitli masīvā --> 7
[-9  0 68 43 23  1  7]
[234  13   0 -45] [1 2 3] [68 43 23  7  1  0 -9]
[-45  -9   0   0   1   1   2   3   7  13  23  43  68 234]
```

Testa piemērs(2)

```
Ievadiet masīva izmēru --> 1
Ievadiet skaitli masīvā --> 876
[876]
Ievadiet masīva izmēru --> 6
Ievadiet skaitli masīvā --> -97
Ievadiet skaitli masīvā --> -14
Ievadiet skaitli masīvā --> 8
Ievadiet skaitli masīvā --> 45
Ievadiet skaitli masīvā --> 32
Ievadiet skaitli masīvā --> 5
[-97 -14   8  45  32   5]
Ievadiet masīva izmēru --> 3
Ievadiet skaitli masīvā --> 976
Ievadiet skaitli masīvā --> 34
Ievadiet skaitli masīvā --> 6
[976  34   6]
[876] [ 45  32   8   5 -14 -97] [  6  34 976]
[-97 -14   5   6   8  32  34  45 876 976]
```

Testa piemērs(3)

```
Ievadiet masīva izmēru --> 4
Ievadiet skaitli masīvā --> 1
Ievadiet skaitli masīvā --> 1
Ievadiet skaitli masīvā --> 1
Ievadiet skaitli masīvā --> 1
[1 1 1 1]
Ievadiet masīva izmēru --> 8
Ievadiet skaitli masīvā --> -5
Ievadiet skaitli masīvā --> -243
Ievadiet skaitli masīvā --> 9876
Ievadiet skaitli masīvā --> 43
Ievadiet skaitli masīvā --> 5
Ievadiet skaitli masīvā --> 26
Ievadiet skaitli masīvā --> 134
Ievadiet skaitli masīvā --> 65
[  -5 -243 9876   43    5   26  134   65]
Ievadiet masīva izmēru --> 2
Ievadiet skaitli masīvā --> 89
Ievadiet skaitli masīvā --> -89
[ 89 -89]
[1 1 1 1] [9876  134   65   43   26    5   -5 -243] [ 89 -89]
[-243  -89   -5    1    1    1    1    5   26   43   65   89  134 9876]
```