

1MPR05_Simona_Bļinova sb24037

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

1.uzdevums

Programma, kas veic nejauši aizpildīta masīva kārtošānu 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)
    meģinajumi = 1
    while meģinajumi <= 3:
        try:
            a = int(a)
            if a > 0:
                return int(a)
            else:
                raise Exception
        except:
            if meģinajumi < 3:
                meģinajumi += 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

```

```
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, 988, 987, 986, 985, 984, 983, 982, 981, 980, 979, 978, 977, 976, 975, 974, 973, 972, 971, 970, 969, 968, 967, 966, 965, 964, 963, 962, 961, 960, 959, 958, 957, 956, 955, 954, 953, 952, 951, 950, 949, 948, 947, 946, 945, 944, 943, 942, 941, 940, 939, 938, 937, 936, 935, 934, 933, 932, 931, 930, 929, 928, 927, 926, 925, 924, 923, 922, 921, 920, 919, 918, 917, 916, 915, 914, 913, 912, 911, 910, 909, 908, 907, 906, 905, 904, 903, 902, 901, 900, 899, 898, 897, 896, 895, 894, 893, 892, 891, 890, 889, 888, 887, 886, 885, 884, 883, 882, 881, 880, 879, 878, 877, 876, 875, 874, 873, 872, 871, 870, 869, 868, 867, 866, 865, 864, 863, 862, 861, 860, 859, 858, 857, 856, 855, 854, 853, 852, 851, 850, 849, 848, 847, 846, 845, 844, 843, 842, 841, 840, 839, 838, 837, 836, 835, 834, 833, 832, 831, 830, 829, 828, 827, 826, 825, 824, 823, 822, 821, 820, 819, 818, 817, 816, 815, 814, 813, 812, 811, 810, 809, 808, 807, 806, 805, 804, 803, 802, 801, 800, 799, 798, 797, 796, 795, 794, 793, 792, 791, 790, 789, 788, 787, 786, 785, 784, 783, 782, 781, 780, 779, 778, 777, 776, 775, 774, 773, 772, 771, 770, 769, 768, 767, 766, 765, 764, 763, 762, 761, 760, 759, 758, 757, 756, 755, 754, 753, 752, 751, 750, 749, 748, 747, 746, 745, 744, 743, 742, 741, 740, 739, 738, 737, 736, 735, 734, 733, 732, 731, 730, 729, 728, 727, 726, 725, 724, 723, 722, 721, 720, 719, 718, 717, 716, 715, 714, 713, 712, 711, 710, 709, 708, 707, 706, 705, 704, 703, 702, 701, 700, 699, 698, 697, 696, 695, 694, 693, 692, 691, 690, 689, 688, 687, 686, 685, 684, 683, 682, 681, 680, 679, 678, 677, 676, 675, 674, 673, 672, 671, 670, 669, 668, 667, 666, 665, 664, 663, 662, 661, 660, 659, 658, 657, 656, 655, 654, 653, 652, 651, 650, 649, 648, 647, 646, 645, 644, 643, 642, 641, 640, 639, 638, 637, 636, 635, 634, 633, 632, 631, 630, 629, 628, 627, 626, 625, 624, 623, 622, 621, 620, 619, 618, 617, 616, 615, 614, 613, 612, 611, 610, 609, 608, 607, 606, 605, 604, 603, 602, 601, 600, 599, 598, 597, 596, 595, 594, 593, 592, 591, 590, 589, 588, 587, 586, 585, 584, 583, 582, 581, 580, 579, 578, 577, 576, 575, 574, 573, 572, 571, 570, 569, 568, 567, 566, 565, 564, 563, 562, 561, 560, 559, 558, 557, 556, 555, 554, 553, 552, 551, 550, 549, 548, 547, 546, 545, 544, 543, 542, 541, 540, 539, 538, 537, 536, 535, 534, 533, 532, 531, 530, 529, 528, 527, 526, 525, 524, 523, 522, 521, 520, 519, 518, 517, 516, 515, 514, 513, 512, 511, 510, 509, 508, 507, 506, 505, 504, 503, 502, 501, 500, 499, 498, 497, 496, 495, 494, 493, 492, 491, 490, 489, 488, 487, 486, 485, 484, 483, 482, 481, 480, 479, 478, 477, 476, 475, 474, 473, 472, 471, 470, 469, 468, 467, 466, 465, 464, 463, 462, 461, 460, 459, 458, 457, 456, 455, 454, 453, 452, 451, 450, 449, 448, 447, 446, 445, 444, 443, 442, 441, 440, 439, 438, 437, 436, 435, 434, 433, 432, 431, 430, 429, 428, 427, 426, 425, 424, 423, 422, 421, 420, 419, 418, 417, 416, 415, 414, 413, 412, 411, 410, 409, 408, 407, 406, 405, 404, 403, 402, 401, 400, 399, 398, 397, 396, 395, 394, 393, 392, 391, 390, 389, 388, 387, 386, 385, 384, 383, 382, 381, 380, 379, 378, 377, 376, 375, 374, 373, 372, 371, 370, 369, 368, 367, 366, 365, 364, 363, 362, 361, 360, 359, 358, 357, 356, 355, 354, 353, 352, 351, 350, 349, 348, 347, 346, 345, 344, 343, 342, 341, 340, 339, 338, 337, 336, 335, 334, 333, 332, 331, 330, 329, 328, 327, 326, 325, 324, 323, 322, 321, 320, 319, 318, 317, 316, 315, 314, 313, 312, 311, 310, 309, 308, 307, 306, 305, 304, 303, 302, 301, 300, 299, 298, 297, 296, 295, 294, 293, 292, 291, 290, 289, 288, 287, 286, 285, 284, 283, 282, 281, 280, 279, 278, 277, 276, 275, 274, 273, 272, 271, 270, 269, 268, 267, 266, 265, 264, 263, 262, 261, 260, 259, 258, 257, 256, 255, 254, 253, 252, 251, 250, 249, 248, 247, 246, 245, 244, 243, 242, 241, 240, 239, 238, 237, 236, 235, 234, 233, 232, 231, 230, 229, 228, 227, 226, 225, 224, 223, 222, 221, 220, 219, 218, 217, 216, 215, 214, 213, 212, 211, 210, 209, 208, 207, 206, 205, 204, 203, 202, 201, 200, 199, 198, 197, 196, 
```

```
Ievadiet masīva izmēru --> 10000
[ 0 4251 8862 ... 8452 1263 1381]

Naivā kārtāšanas metode: 49995000 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 9995, 9994, 9993, 9992, 9991, 9990, 9989, 9988, 9987, 9986
Burbuļa kārtāšanas metode: 49985409 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 9995, 9994, 9993, 9992, 9991, 9990, 9989, 9988, 9987, 99
Atspoles kārtāšanas metode: 25046442 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 9995, 9994, 9993, 9992, 9991, 9990, 9989, 9988, 9987, 9
Ievietšanas kārtāšanas metode: 25036467 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 9995, 9994, 9993, 9992, 9991, 9990, 9989, 9988, 998
Šella kārtāšanas metode: 248007 salīdzināšanas - 10000, 9999, 9998, 9997, 9996, 9995, 9994, 9993, 9992, 9991, 9990, 9989, 9988, 9987, 9986,
```

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 masiva 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)
    meģinajumi = 1
    while meģinajumi <= 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
```