

1MPR06_Simona_Bļinova sb24037

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

1.uzdevums

Programma, kas veic masīva kārtošānu 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)
    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)
```

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, 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, 1
```

Testa piemērs(3)

```
Ievadiet masīva izmēru --> 100000
[ 0 74455 61570 ... 78843 28922 38494]
Ātrā jeb Hoara kārtotāšanas metode: 1997886 salīdzināšanas - 100000, 99999, 99998, 99997, 99996, 99995, 99994, 99993, 99992, 99991, 99990, 99989, 99988, 99987, 99986, 99985, 99984, 99983, 99982, 99981, 99980, 99979, 99978, 99977, 99976, 99975, 99974, 99973, 99972, 99971, 99970, 99969, 99968, 99967, 99966, 99965, 99964, 99963, 99962, 99961, 99960, 99959, 99958, 99957, 99956, 99955, 99954, 99953, 99952, 99951, 99950, 99949, 99948, 99947, 99946, 99945, 99944, 99943, 99942, 99941, 99940, 99939, 99938, 99937, 99936, 99935, 99934, 99933, 99932, 99931, 99930, 99929, 99928, 99927, 99926, 99925, 99924, 99923, 99922, 99921, 99920, 99919, 99918, 99917, 99916, 99915, 99914, 99913, 99912, 99911, 99910, 99909, 99908, 99907, 99906, 99905, 99904, 99903, 99902, 99901, 99900, 99899, 99898, 99897, 99896, 99895, 99894, 99893, 99892, 99891, 99890, 99889, 99888, 99887, 99886, 99885, 99884, 99883, 99882, 99881, 99880, 99879, 99878, 99877, 99876, 99875, 99874, 99873, 99872, 99871, 99870, 99869, 99868, 99867, 99866, 99865, 99864, 99863, 99862, 99861, 99860, 99859, 99858, 99857, 99856, 99855, 99854, 99853, 99852, 99851, 99850, 99849, 99848, 99847, 99846, 99845, 99844, 99843, 99842, 99841, 99840, 99839, 99838, 99837, 99836, 99835, 99834, 99833, 99832, 99831, 99830, 99829, 99828, 99827, 99826, 99825, 99824, 99823, 99822, 99821, 99820, 99819, 99818, 99817, 99816, 99815, 99814, 99813, 99812, 99811, 99810, 99809, 99808, 99807, 99806, 99805, 99804, 99803, 99802, 99801, 99800, 99799, 99798, 99797, 99796, 99795, 99794, 99793, 99792, 99791, 99790, 99789, 99788, 99787, 99786, 99785, 99784, 99783, 99782, 99781, 99780, 99779, 99778, 99777, 99776, 99775, 99774, 99773, 99772, 99771, 99770, 99769, 99768, 99767, 99766, 99765, 99764, 99763, 99762, 99761, 99760, 99759, 99758, 99757, 99756, 99755, 99754, 99753, 99752, 99751, 99750, 99749, 99748, 99747, 99746, 99745, 99744, 99743, 99742, 99741, 99740, 99739, 99738, 99737, 99736, 99735, 99734, 99733, 99732, 99731, 99730, 99729, 99728, 99727, 99726, 99725, 99724, 99723, 99722, 99721, 99720, 99719, 99718, 99717, 99716, 99715, 99714, 99713, 99712, 99711, 99710, 99709, 99708, 99707, 99706, 99705, 99704, 99703, 99702, 99701, 99700, 99699, 99698, 99697, 99696, 99695, 99694, 99693, 99692, 99691, 99690, 99689, 99688, 99687, 99686, 99685, 99684, 99683, 99682, 99681, 99680, 99679, 99678, 99677, 99676, 99675, 99674, 99673, 99672, 99671, 99670, 99669, 99668, 99667, 99666, 99665, 99664, 99663, 99662, 99661, 99660, 99659, 99658, 99657, 99656, 99655, 99654, 99653, 99652, 99651, 99650, 99649, 99648, 99647, 99646, 99645, 99644, 99643, 99642, 99641, 99640, 99639, 99638, 99637, 99636, 99635, 99634, 99633, 99632, 99631, 99630, 99629, 99628, 99627, 99626, 99625, 99624, 99623, 99622, 99621, 99620, 99619, 99618, 99617, 99616, 99615, 99614, 99613, 99612, 99611, 99610, 99609, 99608, 99607, 99606, 99605, 99604, 99603, 99602, 99601, 99600, 99599, 99598, 99597, 99596, 99595, 99594, 99593, 99592, 99591, 99590, 99589, 99588, 99587, 99586, 99585, 99584, 99583, 99582, 99581, 99580, 99579, 99578, 99577, 99576, 99575, 99574, 99573, 99572, 99571, 99570, 99569, 99568, 99567, 99566, 99565, 99564, 99563, 99562, 99561, 99560, 99559, 99558, 99557, 99556, 99555, 99554, 99553, 99552, 99551, 99550, 99549, 99548, 99547, 99546, 99545, 99544, 99543, 99542, 99541, 99540, 99539, 99538, 99537, 99536, 99535, 99534, 99533, 99532, 99531, 99530, 99529, 99528, 99527, 99526, 99525, 99524, 99523, 99522, 99521, 99520, 99519, 99518, 99517, 99516, 99515, 99514, 99513, 99512, 99511, 99510, 99509, 99508, 99507, 99506, 99505, 99504, 99503, 99502, 99501, 99500, 99499, 99498, 99497, 99496, 99495, 99494, 99493, 99492, 99491, 99490, 99489, 99488, 99487, 99486, 99485, 99484, 99483, 99482, 99481, 99480, 99479, 99478, 99477, 99476, 99475, 99474, 99473, 99472, 99471, 99470, 99469, 99468, 99467, 99466, 99465, 99464, 99463, 99462, 99461, 99460, 99459, 99458, 99457, 99456, 99455, 99454, 99453, 99452, 99451, 99450, 99449, 99448, 99447, 99446, 99445, 99444, 99443, 99442, 99441, 99440, 99439, 99438, 99437, 99436, 99435, 99434, 99433, 99432, 99431, 99430, 99429, 9
```

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):
    # mēģinājumu skaita skaitītājs (3 mēģinājumi 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):
    # meģinājumu skaitītājs (3 meģinājumi pareizi ievadīt)
    meģinajumi = 1
    while meģinajumi <= 3:
        try:
            a = float(a)
            return float(a)
        except:
            if meģinajumi < 3:
                meģinajumi += 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):
```

```
    # meģinājumu 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 masivā --> ')  
    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]
```