Структури даних і алгоритми

Кляцко Семен ТК - 31

Задачі 1

1. Робота з одновимірними масивами. Для заданого масиву цілих чисел (довжина > 100, генеровані випадковим чином) знайти мінімальне значення, максимальне значення, впорядкувати масив (за спаданням, за зростанням).

Вивід:
Enter array length: 150

Array: 419 839 224 148 124 246 864 586 771 246 843 207 145 419 82 82 466 269 190 629 969 760 420 471 299 504 102 721 803 718 398 47 920 180 478 37 371 920 0 929 695 503 31 569 226 271 490 15 380 563 908 168 412 881 984 394 282 432 586 211 291 327 6 521 431 776 32 275 150 747 111 656 795 532 462 528 828 761 449 685 346 94 661 1000 484 83 146 609 374 688 835 239 863 249 287 424 877 585 943 277 804 931 712 571 914 645 645 69 415 512 338 535 193 999 508 806 956 303 132 940 981 347 806 1000 202 810 411 471 650 623 372 700 280 995 289 51 500 723 849 16 614 825 977 814 167 244 744 248 671 750

Min: 0
Max: 1000

Sorting by desc: 1000 1000 999 995 984 981 977 969 956 943 940 931 929 920 920 914 908 881 877 864 863 849 843 839 835 828 825 814 810 806 806 804 803 795 776 771 761 760 750 747 744 723 721 718 712 700 695 688 685 671 661 656 650 645 645 629 623 614 609 586 586 585 571 569 563 535 532 528 521 512 508 504 503 500 490 484 478 471 471 466 462 449 432 431 424 420 419 419 415 412 411 398 394 380 374 372 371 347 346 338 327 303 299 291 289 287 282 280 277 275 271 269 249 248 246 246 244 239 226 224 211 207 202 193 190 180 168 167 150 148 146 145 132 124 111 102 94 83 82 82 69 51 47 37 32 31 16 15 6 0

Sorting by asc: 0 6 15 16 31 32 37 47 51 69 82 82 83 94 102 111 124 132 145 146 148 150 167 168 180 190 193 202 207 211 224 226 239 244 246 246 248 249 269 271 275 277 280 282 287 289 291 299 303 327 338 346 347 371 372 374 380 394 398 411 412 415 419 419 420 424 431 432 449 462 466 471 471 478 484 490 500 503 504 508 512 521 528 532 535 563 569 571 585 586 586 609 614 623 629 645 645 650 656 661 671 685 688 695 700 712 718 721 723 744 747 750 760 761 771 776 795 803 804 806 806 810 814 825 828 835 839 843 849 863 864 877 881 908 914 920 920 929 931 940 943 956 969 977 981 984 995 999 1000 1000

Task A: 1000 999 984 977 956 940 929 920 908 877 863 843 835 825 810 806 803 776 761 750 744 721 712 695 685 661 650 645 623 609 586 571 563 532 521 508 503 490 478 471 462 432 424 419 415 411 394 374 371 346 327 299 289 282 277 271 249 246 244 226 211 202 190 168 150 146 132 111 94 82 69 47 32 16 6 0 15 31 37 51 82 83 102 124 145 148 167 180 193 207 224 239 246 248 269 275 280 287 291 303 338 347 372 380 398 412 419 420 431 449 466 471 484 500 504 512 528 535 569 585 586 614 629 645 656 671 688 700 718 723 747 760 771 795 804 806 814 828 839 849 864 881 914 920 931 943 969 981 995 1000

Task Б: 1000 0 1000 6 999 15 995 16 984 31 981 32 977 37 969 47 956 51 943 69 940 82 931 82 929 83 920 94 920 102 914 111 908 124 881 132 877 145 864 146 863 148 849 150 843 167 839 168 835 180 828 190 825 193 814 202 810 207 806 211 806 224 804 226 803 239 795 244 776 246 771 246 761 248 760 249 750 269 747 271 744 275 723 277 721 280 718 282 712 287 700 289 695 291 688 299 685 303 671 327 661 338 656 346 650 347 645 371 645 372 629 374 623 380 614 394 609 398 586 411 586 412 585 415 571 419 569 419 563 420 535 424 532 431 528 432 521 449 512 462 508 466 504 471 503 471 500 478 490 484

Task B: 1000 1000 984 956 940 920 920 914 908 864 828 814 810 806 806 804 776 760 750 744 718 712 700 688 656 650 614 586 586 532 528 512 508 504 500 490 484 478 466 462 432 424 420 412 398 394 380 374 372 346 338 282 280 248 246 246 244 226 224 202 190 180 168 150 148 146 132 124 102 94 82 82 32 16 6 0 15 31 37 47 51 69 83 111 145 167 193 207 211 239 249 269 271 275 277 287 289 291 299 303 327 347 371 411 415 419 419 431 449 471 471 503 521 535 563 569 571 585 609 623 629 645 645 661 671 685 695 721 723 747 761 771 795 803 825 835 839 843 849 863 877 881 929 931 943 969 977 981 995 999

Код створення і заповнення масиву:

```
vector<int> CreateArray(const int& n){

 std::random_device rd; // obtain a random number from hardware
 std::mt19937 gen(rd()); // seed the generator
 std::uniform_int_distribution<> distr(0, 1000); // define the range

 vector<int> arr;

 for (int i = 0; i < n; ++i) {
   arr.push_back(distr(gen));
 }
 return arr;
}
```

Пошук максимального та мінімального значення:

```
auto minNum = min_element(v.begin(), v.end());
auto maxNum = max_element(v.begin(), v.end());
```

Сортування:

```
/*Desc*/ sort(v.begin(), v.end(), greater<int>());

/*Asc*/  sort(v.begin(), v.end());
```

А) в центрі - мінімальне (максимальне) значення, додаємо зліва – справа елементи в порядку зростання (спадання). Вивести масив.

```
vector<int> task1A(vector<int>& v){
 vector<int> res;
 bool insertInBegin =  false;
 for (int & i : v) {
   if(insertInBegin){
     res.insert(res.begin(), i);
   }
   else{
       res.push_back(i);
   }
   insertInBegin = !insertInBegin;
 }
 return res;
}
```

Б) сформувати масив – максимальне, мінімальне, максимальне (з тих, що залишились), мінімальне, … Вивести масив.

```
vector<int> task1B(vector<int>& v) {
 vector<int> res;
 for (int i = 0; i < v.size() / 2 + 1 && res.size() < v.size(); ++i)
{
   res.push_back(v[v.size() - i - 1]);
   if (res.size() < v.size()) {
     res.push_back(v[i]);
   }
 }
 return res;
}
```

В) сформувати масив – елементи з парними номерами спадають, потів – елементи з непарними номерами зростають. Вивести масив.

```
vector<int> task1C(vector<int>& v) {
 vector<int> desc, asc, res;
 asc = v;
 sort(v.begin(), v.end(), greater<int>());
 desc = v;
 for (int i = 0; i < desc.size(); ++i) {
   if (desc[i] % 2 == 0) {
     res.push_back(desc[i]);
   }
 }
 for (int i = 0; i < asc.size(); ++i) {
   if (asc[i] % 2 == 1) {
     res.push_back(asc[i]);
   }
 }
 return res;
}
```

Задачі 2

Робота з матрицями. Для заданого масиву цілих чисел (довжина > 10*10, генеровані випадковим чином) знайти мінімальне значення, максимальне значення, впорядкувати масив за правилом:

Вивід

Enter num of raws: 11
Enter num of columns: 11

Array:
687 486 754 106 380 225 149 581 861 364 498
111 349 545 224 824 613 386 854 391 888 983
412 10 631 712 543 332 50 802 72 668 839
120 464 634 317 0 402 616 197 627 52 800
130 957 39 789 251 744 138 674 963 316 122
355 175 594 731 326 141 978 317 312 720 285
312 595 939 288 575 183 207 347 929 615 611
467 463 740 573 639 221 27 514 602 787 815
776 401 218 535 260 12 859 647 708 256 953
659 496 952 417 196 515 389 995 674 314 598
156 116 143 741 611 618 34 112 148 29 409

Min: 0
Max: 995

Task 2 A:
0 10 12 27 29 34 39 50 52 72 106
111 112 116 120 122 130 138 141 143 148 149
156 175 183 196 197 207 218 221 224 225 251
256 260 285 288 312 312 314 316 317 317 326
332 347 349 355 364 380 386 389 391 401 402
409 412 417 463 464 467 486 496 498 514 515
535 543 545 573 575 581 594 595 598 602 611
611 613 615 616 618 627 631 634 639 647 659
668 674 674 687 708 712 720 731 740 741 744
754 776 787 789 800 802 815 824 839 854 859
861 888 929 939 952 953 957 963 978 983 995

Task 2 Б:

0 111 156 256 332 409 535 611 668 754 861
10 112 175 260 347 412 543 613 674 776 888
12 116 183 285 349 417 545 615 674 787 929
27 120 196 288 355 463 573 616 687 789 939
29 122 197 312 364 464 575 618 708 800 952
34 130 207 312 380 467 581 627 712 802 953
39 138 218 314 386 486 594 631 720 815 957
50 141 221 316 389 496 595 634 731 824 963
52 143 224 317 391 498 598 639 740 839 978
72 148 225 317 401 514 602 647 741 854 983
106 149 251 326 402 515 611 659 744 859 995

Task 2 В:
0 10 12 27 29 34 39 50 52 72 106
314 316 317 317 326 332 347 349 355 364 111
312 581 594 595 598 602 611 611 613 380 112
312 575 731 740 741 744 754 776 615 386 116
288 573 720 888 929 939 952 787 616 389 120
285 545 712 861 983 995 953 789 618 391 122
260 543 708 859 978 963 957 800 627 401 130
256 535 687 854 839 824 815 802 631 402 138
251 515 674 674 668 659 647 639 634 409 141
225 514 498 496 486 467 464 463 417 412 143
224 221 218 207 197 196 183 175 156 149 148

Task 2 Г:
995 618 627 631 634 639 647 659 668 674 674
983 616 380 386 389 391 401 402 409 412 687
978 615 364 196 197 207 218 221 224 417 708
963 613 355 183 72 106 111 112 225 463 712
957 611 349 175 52 10 12 116 251 464 720
953 611 347 156 50 0 27 120 256 467 731
952 602 332 149 39 34 29 122 260 486 740
939 598 326 148 143 141 138 130 285 496 741

929 595 317 317 316 314 312 312 288 498 744
888 594 581 575 573 545 543 535 515 514 754
861 859 854 839 824 815 802 800 789 787 776

Task 2 Д:
0 10 12 27 29 34 39 50 52 72 106
149 148 143 141 138 130 122 120 116 112 111
156 175 183 196 197 207 218 221 224 225 251
326 317 317 316 314 312 312 288 285 260 256
332 347 349 355 364 380 386 389 391 401 402
515 514 498 496 486 467 464 463 417 412 409
535 543 545 573 575 581 594 595 598 602 611
659 647 639 634 631 627 618 616 615 613 611
668 674 674 687 708 712 720 731 740 741 744
859 854 839 824 815 802 800 789 787 776 754
861 888 929 939 952 953 957 963 978 983 995

Task 2 E:
0 10 317 326 332 347 647 659 668 674 861
27 12 317 349 355 674 639 634 687 888 861
29 34 312 314 364 380 627 631 708 712 861
50 39 312 386 389 720 618 616 731 52 861
52 72 260 285 391 401 613 615 740 741 861
111 106 256 402 409 744 611 611 754 720 861
112 116 224 225 412 417 598 602 776 787 861
122 120 221 463 464 789 595 594 800 787 861
130 138 197 207 467 486 575 581 802 815 861
143 141 196 496 498 824 573 545 839 314 861
148 149 156 175 514 515 535 543 854 859 861

Код створення і заповнення масиву:

```
vector<vector<int>> CreateArray(const int& n1 /*raws*/, const int& n2
/*columns*/){

 std::random_device rd; // obtain a random number from hardware
 std::mt19937 gen(rd()); // seed the generator
 std::uniform_int_distribution<> distr(0, 1000); // define the range

 vector<vector<int>> arr(n1);

 for (int i = 0; i < n1; ++i) {
    for(int j = 0; j < n2; ++j)
    arr[i].push_back(distr(gen));
 }
 return arr;
}
```

Пошук максимального та мінімального значення:

```
int Max(const vector<vector<int>>& vv, int xn, int yn) {
 int max = std::numeric_limits<int>::min();
 for(int x = 0; x < xn; x++) {
    for(int y = 0; y < yn; y++) {
        if(vv[x][y] > max){
            max = vv[x][y];
        }
    }
    }
 return max;
}
```

```
int Min(const vector<vector<int>>& vv, int xn, int yn) {
 int min = std::numeric_limits<int>::max();
 for(int x = 0; x < xn; x++) {
    for(int y = 0; y < yn; y++) {
       if(vv[x][y] < min){
          min = vv[x][y];
       }
    }
 }
 return min;
}
```

A)

```
void task2A(vector<vector<int>> vv, int n1, int n2) {
 vector<int> v;
 for(int i =0; i < n1; ++i) {
    for (int j = 0; j < n2; ++j) {
       v.push_back(vv[i][j]);
    }
 }
 sort(v.begin(), v.end());

 int count = 0;
 for(int i =0; i < n1; ++i) {
    for (int j = 0; j < n2; ++j) {
       vv[i][j] = v[count];
       count++;
    }
 }
 printVector(vv);
}
```

Б)

```
void task2B(vector<vector<int>> vv, int n1, int n2) {
 vector<int> v;
 for (int i = 0; i < n1; ++i) {
    for (int j = 0; j < n2; ++j) {
       v.push_back(vv[i][j]);
    }
 }
    sort(v.begin(), v.end());
    int count = 0;
    for (int column = 0; column < n2; column++) {
       for (int raw = 0; raw < n1; raw++) {
          vv[raw][column] = v[count];
          count++;
       }
    }
 printVector(vv);
}
```

В)

```
void task2C(vector<vector<int>> vv, int n1, int n2) {

 vector<int> v;
 for (int i = 0; i < n1; ++i) {
    for (int j = 0; j < n2; ++j) {
       v.push_back(vv[i][j]);
    }
 }
 sort(v.begin(), v.end());

 int count = 0;
 int left = 0;
 int right = n2-1;
 int up = 0;
 int down = n1 - 1;
 int x  = 0;
 int y = 0;
 while(left < right+1 || up < down+1) {
```

```
    if (y == up && x < right) {
      vv[up][x] = v[count];
      if (x < right) {
        x++;
      }
      if (x == right) {
        up++;
      }
    } else if (x == right && y < down) {
       vv[y][right] = v[count];
      if (y < down) {
        y++;
      }
      if (y == down) {
        right--;
      }
    } else if (y == down && x > left) {
       vv[down][x] = v[count];
      if (x > left) {
        x--;
      }
      if (x == left) {
        down--;
      }
    } else if (x == left && y > up) {
       vv[y][left]  = v[count];
      if (y > up) {
        y--;
      }
      if (y == up) {
        left++;
      }
    } else
    {
      vv[y][x] = v[count];
      break;
    }
    count++;
  }
 printVector(vv);
}
```

Г)

```
void task2D( vector<vector<int>> vv, int n1, int n2) {
 vector<int> v;
 for (int i = 0; i < n1; ++i) {
   for (int j = 0; j < n2; ++j) {
     v.push_back(vv[i][j]);
   }
 }
sort(v.begin(), v.end(), greater<int>());
 int count = 0;
 int left = 0;
 int right = n2-1;
 int up = 0;
 int down = n1 - 1;
 int x  = 0;
 int y = 0;
 vector<int> res;
 while(left < right+1 || up < down+1) {
   if (y == up && x > left) {
     vv[up][x] = v[count];
     if (x > left) {
       x--;
     }
     if (x == left) {
       up++;
     }
   } else if (x == right && y > up) {
     vv[y][right] = v[count];
     if (y > up) {
       y--;
     }
     if (y ==up) {
       right--;
     }
   } else if (y == down && x < right) {
     vv[down][x] = v[count];
     if (x < right) {
       x++;
     }
     if (x == right) {
       down--;
```

```
      }
    } else if (x == left && y < down) {
     vv[y][left] = v[count];
      if (y < down) {
        y++;
      }
      if (y == down) {
        left++;
      }
    } else
    {
       vv[y][x] = v[count];
      break;
    }
    count++;
  }
 printVector(vv);
}
```

Д)

```
void task2E(vector<vector<int>> vv, int n1, int n2) {

 vector<int> v;
 for (int i = 0; i < n1; ++i) {
    for (int j = 0; j < n2; ++j) {
      v.push_back(vv[i][j]);
    }
 }
 sort(v.begin(), v.end());

 int count = 0;

 bool printLeftToRight = true;
 for(int i = 0; i < n1; ++i){
    if(printLeftToRight)
    {
      for(int j = 0; j < n2; j++){
        vv[i][j] = v[count];
        count++;
      }
    }
    else{
```

```
      for(int j = n2 - 1; j >= 0; j--){
         vv[i][j] = v[count];
        count++;
      }
    }
    printLeftToRight = !printLeftToRight;
  }
  printVector(vv);
}
```

E)

```
void task2F(vector<vector<int>> vv, int n1, int n2) {

  vector<int> v;
  for (int i = 0; i < n1; ++i) {
    for (int j = 0; j < n2; ++j) {
      v.push_back(vv[i][j]);
    }
  }
  sort(v.begin(), v.end());

  int count = 0;

  bool leftToRight = true;
  bool upToDown = true;
  for(int doubleColumn = 0; doubleColumn < n2 / 2; ++doubleColumn){
    leftToRight = true;
    if(upToDown){
      for(int raw = 0; raw < n1; raw++){
        if(leftToRight){
          vv[raw][doubleColumn*2] = v[count];
          count++;
          vv[raw][doubleColumn*2 + 1] = v[count];
          count++;
        }
        else{
          vv[raw][doubleColumn+1] = v[count];
          count++;
          vv[raw][doubleColumn*2]=  v[count];
          count++;
        }
```

```
        leftToRight = !leftToRight;
      }
    }
    else{
      for(int raw = n1 - 1; raw >= 0; raw--){
        if(leftToRight){
            vv[raw][doubleColumn*2] = v[count];
            count++;
            vv[raw][doubleColumn*2 + 1] = v[count];
            count++;
        }
        else{
            vv[raw][doubleColumn*2+1] = v[count] ;
            count++;
            vv[raw][doubleColumn*2] = v[count];
            count++;
        }
        leftToRight = !leftToRight;
      }
    }
    upToDown = !upToDown;
  }
  if(n2 % 2 == 1){
    if(upToDown){
      for(int i = 0; i < n1; i++){
       vv[i][n2 - 1]  = v[count];
       count++;
      }
    }
    else{
      for(int i = n1 - 1; i >= 0; i--){
         vv[i][n2 - 1]= v[count];
      }
    }
  }
  printVector(vv);
}
```

Задачі 3

Вивід

Enter z:
6
Enter y:
6
Enter x:
6

518 192 888 544 423 268
527 55 87 874 291 5
324 840 815 555 413 991
537 195 191 193 950 654
492 681 559 899 142 509
693 179 773 245 863 357

522 476 768 524 124 134
958 62 488 343 932 227
103 180 898 385 18 580
271 154 2 235 18 359
158 94 227 52 640 592
255 769 575 942 248 750

479 958 462 150 499 212
441 33 475 723 182 738
203 566 155 345 304 683
530 81 36 642 999 526
879 358 897 219 280 738
66 941 450 971 965 431

411 158 862 341 709 490
608 478 693 627 373 764
617 582 432 506 717 827
210 246 531 628 339 87
930 581 288 936 189 24
146 745 478 969 111 369

805 803 625 33 38 502
941 712 440 952 23 294
718 579 900 475 302 122
669 682 119 727 56 357
943 574 550 944 406 415
188 671 5 321 143 164

489 880 436 253 986 409
481 270 205 360 440 75
212 579 718 364 173 604
260 272 267 385 350 946
333 558 68 169 831 239
759 101 126 152 219 489

Task 3 Max: 999
Task 3 Min: 2

Task 3 Sort X:
192 268 423 518 544 888
5 55 87 291 527 874
324 413 555 815 840 991
191 193 195 537 654 950
142 492 509 559 681 899
179 245 357 693 773 863

124 134 476 522 524 768
62 227 343 488 932 958
18 103 180 385 580 898
2 18 154 235 271 359
52 94 158 227 592 640
248 255 575 750 769 942

150 212 462 479 499 958
33 182 441 475 723 738
155 203 304 345 566 683
36 81 526 530 642 999
219 280 358 738 879 897
66 431 450 941 965 971

158 341 411 490 709 862
373 478 608 627 693 764
432 506 582 617 717 827
87 210 246 339 531 628
24 189 288 581 930 936
111 146 369 478 745 969

33 38 502 625 803 805
23 294 440 712 941 952
122 302 475 579 718 900
56 119 357 669 682 727
406 415 550 574 943 944
5 143 164 188 321 671

253 409 436 489 880 986
75 205 270 360 440 481
173 212 364 579 604 718
260 267 272 350 385 946
68 169 239 333 558 831
101 126 152 219 489 759

Task 3 Sort Y:
5 55 87 291 527 863
142 193 195 518 544 874
179 245 357 537 654 888
191 268 423 559 681 899
192 413 509 693 773 950
324 492 555 815 840 991

2 18 154 227 271 359
18 94 158 235 524 640
52 103 180 385 580 768
62 134 343 488 592 898
124 227 476 522 769 942
248 255 575 750 932 958

33 81 304 345 499 683
36 182 358 475 566 738
66 203 441 479 642 897
150 212 450 530 723 958
155 280 462 738 879 971
219 431 526 941 965 999

24 146 246 339 531 628
87 189 288 478 693 764
111 210 369 490 709 827
158 341 411 581 717 862
373 478 582 617 745 936
432 506 608 627 930 969

5 38 164 188 321 671
23 119 357 574 682 727
33 143 440 579 718 805
56 294 475 625 803 900
122 302 502 669 941 944
406 415 550 712 943 952

68 126 152 219 385 481
75 169 239 333 440 718
101 205 270 350 489 759
173 212 272 360 558 831
253 267 364 489 604 946
260 409 436 579 880 986

Task 3 Sort Z:
2 18 87 188 271 359
18 94 158 235 440 640
33 103 180 350 489 759
56 134 272 360 558 831
122 227 364 489 604 936
219 255 436 579 840 952

5 38 152 219 321 481
23 119 195 333 524 718
52 143 270 385 580 768
62 212 343 488 592 862
124 267 462 522 745 942
248 409 526 627 880 958

5 55 154 227 385 628
36 169 239 475 544 727
66 203 357 479 642 805
150 212 411 530 681 898
155 280 476 617 769 944
260 415 550 712 930 969

24 81 164 291 499 671
75 182 288 478 566 738
101 205 369 490 654 827
158 268 423 559 717 899
192 302 502 669 773 946
324 431 555 750 932 986

33 126 246 339 527 683
87 189 357 518 682 764
111 210 440 537 709 888
173 294 450 581 723 900
253 413 509 693 879 950
406 492 575 815 943 991

68 146 304 345 531 863
142 193 358 574 693 874
179 245 441 579 718 897
191 341 475 625 803 958
373 478 582 738 941 971
432 506 608 941 965 999

Код створення і заповнення масиву:

```
vector<vector<vector<int>>>  CreateArray(int xn, int yn, int zn){

 std::random_device rd; // obtain a random number from hardware
 std::mt19937 gen(rd()); // seed the generator
 std::uniform_int_distribution<> distr(0, 1000); // define the range

 vector<vector<vector<int>>> arr(zn);

 for (int i = 0; i < zn; ++i) {
     arr[i].resize(yn);
 }


 for (int i = 0; i < zn; ++i) {
   for(int j = 0; j < yn; ++j)
      for(int k = 0; k < xn; ++k){
         arr[i][j].push_back(distr(gen));
      }
 }
 return arr;
}
```

Пошук максимального та мінімального значення:

```
int Max(const vector<vector<vector<int>>>& vvv, int xn, int yn, int
zn) {
 int max = std::numeric_limits<int>::min();
 for(int z = 0; z < zn; z++) {
   for(int y = 0; y < yn; y++) {
     for(int x = 0; x < xn; x++) {
```

```
        if(vvv[z][y][x] > max){
          max = vvv[z][y][x];
        }
      }
    }
  }
  return max;
}

int Min(const vector<vector<vector<int>>>& vvv, int xn, int yn, int
zn) {
  int min = std::numeric_limits<int>::max();
  for(int z = 0; z < zn; z++) {
    for(int y = 0; y < yn; y++) {
      for(int x = 0; x < xn; x++) {
        if(vvv[z][y][x] < min){
          min = vvv[z][y][x];
        }
      }
    }
  }
  return min;
}
```

A)

```
void Task3SortX(vector<vector<vector<int>>>& vvv, int xn, int yn, int
zn){
    for(int z = 0; z < zn; z++){
      for(int y = 0; y < yn; ++y){
        sort(vvv[z][y].begin(), vvv[z][y].end());
      }
    }
}
```

Б)

```
void Task3SortY(vector<vector<vector<int>>>& vvv, int xn, int yn, int
zn){

vector<int> v;
 for(int z = 0; z < zn; z++){
   for(int x = 0; x < xn; x++){
     for(int y = 0; y < yn; y++){
       v.push_back(vvv[z][y][x]);
     }
     sort(v.begin(), v.end());
     for(int y = 0; y < yn; y++){
       vvv[z][y][x] = v[y];
     }
     v.clear();
   }
 }
}
```

В)

```
void Task3SortZ(vector<vector<vector<int>>>& vvv, int xn, int yn, int
zn){

 vector<int> v;
 for(int y = 0; y < yn; y++){
   for(int x = 0; x < xn; x++){
     for(int z = 0; z < zn; z++){
       v.push_back(vvv[z][y][x]);
     }
     sort(v.begin(), v.end());
     for(int z = 0; z < zn; z++){
       vvv[z][y][x] = v[z];
     }
     v.clear();
   }
 }

}
```

Задачі 4

Операції над множинами. Реалізувати операції над множинами, що задані у вигляді масивів. Операції – об’єднання, перетин, доповнення, різниця, симетрична різниця.

Вивід

Task 4
Enter v1 length: 100

Enter v2 length: 100
v1 :
578 304 127 910 235 993 554 165 470 56 190 245 917 572 590 520 332 113 298 496 988 850
262 394 230 547 813 980 789 393 468 13 694 864 685 559 9 574 361 606 483 745 807 40 850
379 570 224 451 359 910 919 980 726 621 460 85 363 38 244 948 48 517 272 197 970 676 24
955 252 370 799 827 372 771 606 94 856 401 925 193 275 116 861 982 861 961 433 84 600
286 142 332 646 942 618 315 128 584 747
371 433 77 266 133 810 532 18 666 413 774 432 570 631 879 107 70 990 670 176 359 572
325 725 867 472 46 898 53 316 790 61 28 851 859 801 131 617 113 716 628 399 865 395 297
945 970 988 499 786 538 39 116 491 323 99 692 385 4 267 334 441 888 941 418 705 1 895
700 569 371 797 138 368 60 101 469 33 89 983 1000 477 166 263 67 51 522 276 637 291 805
249 74 466 778 57 565 767 484 688

Task 4 Union
578 304 127 910 235 993 554 165 470 56 190 245 917 572 590 520 332 113 298 496 988 850
262 394 230 547 813 980 789 393 468 13 694 864 685 559 9 574 361 606 483 745 807 40 850
379 570 224 451 359 910 919 980 726 621 460 85 363 38 244 948 48 517 272 197 970 676 24
955 252 370 799 827 372 771 606 94 856 401 925 193 275 116 861 982 861 961 433 84 600
286 142 332 646 942 618 315 128 584 747 371 77 266 133 810 532 18 666 413 774 432 631
879 107 70 990 670 176 325 725 867 472 46 898 53 316 790 61 28 851 859 801 131 617 716
628 399 865 395 297 945 499 786 538 39 491 323 99 692 385 4 267 334 441 888 941 418 705
1 895 700 569 371 797 138 368 60 101 469 33 89 983 1000 477 166 263 67 51 522 276 637
291 805 249 74 466 778 57 565 767 484 688

Task 4 Cross
572 113 988 570 359 970 116 433

Task 4 Diff
578 304 127 910 235 993 554 165 470 56 190 245 917 590 520 332 298 496 850 262 394 230 547 813 980 789 393 468 13 694 864 685 559 9 574 361 606 483 745 807 40 850 379 224 451 910 919 980 726 621 460 85 363 38 244 948 48 517 272 197 676 24 955 252 370 799 827 372 771 606 94 856 401 925 193 275 861 982 861 961 84 600 286 142 332 646 942 618 315 128 584 747

Task 4 Symmetric difference
578 304 127 910 235 993 554 165 470 56 190 245 917 590 520 332 298 496 850 262 394 230 547 813 980 789 393 468 13 694 864 685 559 9 574 361 606 483 745 807 40 850 379 224 451 910 919 980 726 621 460 85 363 38 244 948 48 517 272 197 676 24 955 252 370 799 827 372 771 606 94 856 401 925 193 275 861 982 861 961 84 600 286 142 332 646 942 618 315 128 584 747 371 77 266 133 810 532 18 666 413 774 432 631 879 107 70 990 670 176 325 725 867 472 46 898 53 316 790 61 28 851 859 801 131 617 716 628 399 865 395 297 945 499 786 538 39 491 323 99 692 385 4 267 334 441 888 941 418 705 1 895 700 569 371 797 138 368 60 101 469 33 89 983 1000 477 166 263 67 51 522 276 637 291 805 249 74 466 778 57 565 767 484 688

A) Об’єднання

```
vector<int> Task4Union(const vector<int>& v1, const vector<int>& v2){
 vector<int> res = v1;
 vector<int> temp = v1;

 for(auto& el : v2){
   auto it = find(temp.begin(), temp.end(), el);
   if(it != temp.end()){
     temp.erase(it);
   }
   else{
     res.push_back(el);
   }
 }
 return res;
}
```

Б) Перетин

```
vector<int> Task4Cross(const vector<int>& v1, const vector<int>& v2){
 vector<int> temp = v2;
 vector<int> res;
 for(const auto& el : v1){
   auto it = find(temp.begin(), temp.end(), el);
   if(it != temp.end()){
     res.push_back(el);
     temp.erase(it);
   }
 }
 return res;
}
```

В) Різниця

```
vector<int> Task4Diff(const vector<int>& v1, const vector<int>& v2){
 vector<int> cross = Task4Cross(v1,v2);
 vector<int> res;
 for(const auto& el : v1){
   auto it = find(cross.begin(), cross.end(), el);
   if(it == cross.end()){
     res.push_back(el);
   }
   else{
     cross.erase(it);
   }
 }
 return res;
}
```

Г) Симетрична різниця

```
vector<int> Task4DiffSym(const vector<int>& v1, const vector<int>&
v2){
 vector<int> union_ = Task4Union(v1, v2);
 vector<int> cross = Task4Cross(v1,v2);
 vector<int> res;

 for(const auto& el : union_){
   auto it = find(cross.begin(), cross.end(), el);
   if(it == cross.end()){
     res.push_back(el);
   }
   else{
     cross.erase(it);
   }
 }
 return res;
}
```