1. **Написать программу, которая дает пользователю ввести 5 фамилий студентов, а затем сортирует их по возрастанию.**

**-\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-**

#include <iostream>

#include <Windows.h>

#include <string>

using namespace std;

void sort(string\* arr, int nomer);

void vyvod(string\* arr, int nomer);

int main()

{

system("chcp 1251");

string mass[5] = {};

for (int i = 0; i < 5; i++)

{

cout << " Введите " << i + 1 << " фамилию ";

cin >> mass[i];

}

cout << " Cписок студентов:" << endl;

vyvod(mass, 5);

sort(mass, 5); //Сортируем массив

cout << " Oтсортированный список студентов:" << endl;

vyvod(mass, 5);

return 0;

}

void sort(string\* arr, int nomer)

{

for (int i = 1;i < nomer; i++)

{

string temp = arr[i];

// Вставляем arr [nomer] в правильное положение

int j = i - 1;

while (j >= 0 && temp.length() < arr[j].length())

{

arr[j + 1] = arr[j];

j--;

}

arr[j + 1] = temp;

}

}

void vyvod(string\* arr, int nomer)

{

for (int i = 0; i < nomer; i++)

{

cout << arr[i] << " ";

}

cout << endl << endl;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_---

1. **Есть 3 двумерных массива A, B, C. Количество строк и столбцов для них вводит пользователь с клавиатуры для каждого из них. Реализуйте ■ Создайте одномерный массив, который содержит общие значения для A, B, C; ■ Создайте одномерный массив, который содержит уникальные значения для A, B, C; ■ Создайте одномерный массив, который содержит общие значения для A и C; ■ Создайте одномерный массив, который содержит отрицательные значения для A, B, C без повторений.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

#include <iostream>

#include <iomanip>

#include <ctime>

#include <Windows.h>

using namespace std;

void print\_onedim\_array(int\* array, int n); //функция печати одномерного массива

void twodimensional\_array(int\*\*& lines, int M, int N); //функция создания и заполнения двумерного динамического массива

void delete\_twodimensional\_array(int\*\*& lines, int M, int N); //функция удаления двумерного динамического массива

void print\_twodimensional\_array(int\*\*& lines, int M, int N); //функция печати двумерного динамического массива

int common2(int\*\* lines1, int M1, int N1, int\*\* lines2, int M2, int N2); //функция подсчета количества общих элементов в двух массивах

int common3(int\*\* lines1, int M1, int N1, int\*\* lines2, int M2, int N2, int\*\* lines3, int M3, int N3); //функция подсчета количества общих элементов в трех массивах

void fill\_first\_array(int\*\* lines1, int M1, int N1, int\*\* lines2, int M2, int N2, int\*\* lines3, int M3, int N3, int\*& first\_array); //функция заполнения одномерного массива общиими из 3 массивов

int common\_key(int\*\* lines1, int M1, int N1, int k); //функция, считающая количество повторений ключа в двумерном массиве

void fill\_second\_array(int\*\* lines1, int M1, int N1, int\*\* lines2, int M2, int N2, int\*& second\_array); //функция заполнения одномерного массива общими из 2 массивов

bool check\_coincidence(int\* ptr, int n, int k); //функция информирования о повторяющемся элементе в одномерном массиве

int count\_coincidence(int\* arr, int m, int k); //функция подсчета повторяющихся элементов в одномерном массиве

int\* uniq(int\* arr, int size, int& newSize);//создание нового массива без повторений

bool inArray(int\* arr, int need, int count);//сравнение и определение необходимого массива

int main()

{

system("chcp 1251");

srand(time(0));

int\*\* mass\_A;

int\*\* mass\_B;

int\*\* mass\_C;

size\_t stroka\_A, stroka\_B, stroka\_C, kolonka\_A, kolonka\_B, kolonka\_C;

cout << " Введите размер для двумерного массива A " << endl;

cin >> stroka\_A >> kolonka\_A;

cout << " Введите размер для двумерного массива B " << endl;

cin >> stroka\_B >> kolonka\_B;

cout << " Введите размер для двумерного массива C " << endl;

cin >> stroka\_C >> kolonka\_C;

twodimensional\_array(mass\_A, stroka\_A, kolonka\_A);

print\_twodimensional\_array(mass\_A, stroka\_A, kolonka\_A);

cout << endl;

twodimensional\_array(mass\_B, stroka\_B, kolonka\_B);

print\_twodimensional\_array(mass\_B, stroka\_B, kolonka\_B);

cout << endl;

twodimensional\_array(mass\_C, stroka\_C, kolonka\_C);

print\_twodimensional\_array(mass\_C, stroka\_C, kolonka\_C);

cout << endl;

const size\_t first\_alement = common3(mass\_A, stroka\_A, kolonka\_A, mass\_B, stroka\_B, kolonka\_B, mass\_C, stroka\_C, kolonka\_C); //размер 1 массива для 3 общих

int\* first\_array = new int[first\_alement]; //создание 1 массива с общими из 3 массивов

fill\_first\_array(mass\_A, stroka\_A, kolonka\_A, mass\_B, stroka\_B, kolonka\_B, mass\_C, stroka\_C, kolonka\_C, first\_array); //заполнение массива общими из 3 массивов

cout << " Общие элементы в массивах A, B, C: " << endl;

print\_onedim\_array(first\_array, first\_alement);

int uniq\_aliment = 0;

for (int i = 0; i < stroka\_A; i++)

{ //проверка на уникальность элементов массива А

for (int j = 0; j < kolonka\_A; j++)

{

if ((common\_key(mass\_B, stroka\_B, kolonka\_B, mass\_A[i][j]) == 0) &&

(common\_key(mass\_C, stroka\_C, kolonka\_C, mass\_A[i][j]) == 0))

{

uniq\_aliment++;

}

}

}

for (int i = 0; i < stroka\_B; i++)

{ //проверка на уникальность элементов массива B

for (int j = 0; j < kolonka\_B; j++)

{

if ((common\_key(mass\_A, stroka\_A, kolonka\_A, mass\_B[i][j]) == 0) &&

(common\_key(mass\_C, stroka\_C, kolonka\_C, mass\_B[i][j]) == 0))

{

uniq\_aliment++;

}

}

}

for (int i = 0; i < stroka\_C; i++)

{ //проверка на уникальность элементов массива C

for (int j = 0; j < kolonka\_C; j++)

{

if ((common\_key(mass\_A, stroka\_A, kolonka\_A, mass\_C[i][j]) == 0) &&

(common\_key(mass\_B, stroka\_B, kolonka\_B, mass\_C[i][j]) == 0))

{

uniq\_aliment++;

}

}

}

int\* uniq\_array = new int[uniq\_aliment];//массив с уникальными элемантами

int counter = 0;//счетчик для перемещения по массиву с уникальными элементами

for (int i = 0; i < stroka\_A; i++)

{

for (int j = 0; j < kolonka\_A; j++)

{

if ((common\_key(mass\_B, stroka\_B, kolonka\_B, mass\_A[i][j]) == 0) &&

(common\_key(mass\_C, stroka\_C, kolonka\_C, mass\_A[i][j]) == 0))

{

uniq\_array[counter] = mass\_A[i][j];

counter++;

}

}

}

for (int i = 0; i < stroka\_B; i++)

{

for (int j = 0; j < kolonka\_B; j++)

{

if ((common\_key(mass\_A, stroka\_A, kolonka\_A, mass\_B[i][j]) == 0) &&

(common\_key(mass\_C, stroka\_C, kolonka\_C, mass\_B[i][j]) == 0))

{

uniq\_array[counter] = mass\_B[i][j];

counter++;

}

}

}

for (int i = 0; i < stroka\_C; i++)

{

for (int j = 0; j < kolonka\_C; j++)

{

if ((common\_key(mass\_A, stroka\_A, kolonka\_A, mass\_C[i][j]) == 0) &&

(common\_key(mass\_B, stroka\_B, kolonka\_B, mass\_C[i][j]) == 0))

{

uniq\_array[counter] = mass\_C[i][j];

counter++;

}

}

}

cout << endl;

cout << " Уникальные элементы в массивах A, B, C: " << endl;

print\_onedim\_array(uniq\_array, counter);

const size\_t second\_alement = common2(mass\_A, stroka\_A, kolonka\_A, mass\_C, stroka\_C, kolonka\_C);

int\* second\_array = new int[second\_alement];

fill\_second\_array(mass\_A, stroka\_A, kolonka\_A, mass\_C, stroka\_C, kolonka\_C, second\_array);//заполнение массива общими из А и С

cout << endl;

cout << " Общие элементы для массивов A и C: " << endl;

print\_onedim\_array(second\_array, second\_alement);

int neg\_alement = 0; //количество отрицательных элементов

for (int i = 0; i < stroka\_A; i++)

{

for (int j = 0; j < kolonka\_A; j++)

{

if (mass\_A[i][j] < 0)

{

neg\_alement++;

}

}

}

for (int i = 0; i < stroka\_B; i++)

{

for (int j = 0; j < kolonka\_B; j++)

{

if (mass\_B[i][j] < 0)

{

neg\_alement++;

}

}

}

for (int i = 0; i < stroka\_C; i++)

{

for (int j = 0; j < kolonka\_C; j++)

{

if (mass\_C[i][j] < 0)

{

neg\_alement++;

}

}

}

int\* temp\_neg\_array = new int[neg\_alement];

int neg\_schet = 0; //счетчик для итерации по массиву отрицательных значений

for (int i = 0; i < stroka\_A; i++)

{

for (int j = 0; j < kolonka\_A; j++)

{

if (mass\_A[i][j] < 0)

{

temp\_neg\_array[neg\_schet] = mass\_A[i][j];

neg\_schet++;

}

}

}

for (int i = 0; i < stroka\_B; i++)

{

for (int j = 0; j < kolonka\_B; j++)

{

if (mass\_B[i][j] < 0) {

temp\_neg\_array[neg\_schet] = mass\_B[i][j];

neg\_schet++;

}

}

}

for (int i = 0; i < stroka\_C; i++)

{

for (int j = 0; j < kolonka\_C; j++)

{

if (mass\_C[i][j] < 0) {

temp\_neg\_array[neg\_schet] = mass\_C[i][j];

neg\_schet++;

}

}

}

cout << endl;

cout << " Отрицательные элементы массивов А,В,С: " << endl;

for (int i = 0; i < neg\_schet; i++)

{

cout << " " << temp\_neg\_array[i] << " ";

}

cout << endl << endl;

cout << " Не повторяющийся отрицателные элементы массивов A,B,C: " << endl;

int newSize = 0;

int\* newArr = uniq(temp\_neg\_array, neg\_schet, newSize);

for (int i = 0; i < newSize; ++i)

{

cout << " " << newArr[i] << " ";

}

cout << endl;

return 0;

}

int count\_coincidence(int\* arr, int m, int k)

{

int res = 0;

for (int i = 0; i < m; i++)

{

if (arr[i] == k)

{

res++;

}

}

if (res == 1)

{

return res = 0;

}

else

{

return res;

}

}

bool check\_coincidence(int\* ptr, int n, int k)

{

int res = 0;

for (int i = 0; i < n; i++)

{

if (ptr[i] == k)

{

res++;

}

}

if (res == 1) return 1;

else return 0;

}

void fill\_second\_array(int\*\* lines1, int M1, int N1, int\*\* lines2, int M2, int N2, int\*& second\_array)

{

int counter = 0;

for (int i1 = 0; i1 < M1; i1++)

{

for (int j1 = 0; j1 < N1; j1++)

{

for (int i2 = 0; i2 < M2; i2++)

{

for (int j2 = 0; j2 < N2; j2++)

{

if (lines1[i1][j1] == lines2[i2][j2])

{

second\_array[counter] = lines1[i1][j1];

counter++;

}

}

}

}

}

}

int common\_key(int\*\* lines1, int M1, int N1, int k)

{

int result = 0;

for (int i = 0; i < M1; i++)

{

for (int j = 0; j < N1; j++)

{

if (k == lines1[i][j])

{

result++;

}

}

}

return result;

}

void fill\_first\_array(int\*\* lines1, int M1, int N1,

int\*\* lines2, int M2, int N2,

int\*\* lines3, int M3, int N3, int\*& first\_array)

{

int counter = 0;

for (int i1 = 0; i1 < M1; i1++)

{

for (int j1 = 0; j1 < N1; j1++)

{

for (int i2 = 0; i2 < M2; i2++)

{

for (int j2 = 0; j2 < N2; j2++)

{

for (int i3 = 0; i3 < M3; i3++)

{

for (int j3 = 0; j3 < N3; j3++)

{

if ((lines1[i1][j1] == lines2[i2][j2]) &&

(lines2[i2][j2] == lines3[i3][j3]) &&

(lines1[i1][j1] == lines3[i3][j3]))

{

first\_array[counter] = lines1[i1][j1];

counter++;

}

}

}

}

}

}

}

}

int common3(int\*\* lines1, int M1, int N1, int\*\* lines2, int M2, int N2, int\*\* lines3, int M3, int N3)

{

int result = 0;

for (int i1 = 0; i1 < M1; i1++)

{

for (int j1 = 0; j1 < N1; j1++)

{

for (int i2 = 0; i2 < M2; i2++)

{

for (int j2 = 0; j2 < N2; j2++)

{

for (int i3 = 0; i3 < M3; i3++)

{

for (int j3 = 0; j3 < N3; j3++)

{

if ((lines1[i1][j1] == lines2[i2][j2]) &&

(lines2[i2][j2] == lines3[i3][j3]) &&

(lines1[i1][j1] == lines3[i3][j3]))

{

result++;

}

}

}

}

}

}

}

return result;

}

int common2(int\*\* lines1, int M1, int N1, int\*\* lines2, int M2, int N2)

{

int result = 0;

for (int i1 = 0; i1 < M1; i1++)

{

for (int j1 = 0; j1 < N1; j1++)

{

for (int i2 = 0; i2 < M2; i2++)

{

for (int j2 = 0; j2 < N2; j2++)

{

if (lines1[i1][j1] == lines2[i2][j2])

{

result++;

}

}

}

}

}

return result;

}

void print\_twodimensional\_array(int\*\*& lines, int M, int N)

{

for (int i = 0; i < M; i++)

{

for (int j = 0; j < N; j++)

{

cout << " " << setw(3) << lines[i][j] << " ";

} cout << endl;

}

}

void twodimensional\_array(int\*\*& lines, int M, int N)

{

lines = new int\* [M];

for (int i = 0; i < M; i++)

{

lines[i] = new int[N];

}

for (int i = 0; i < M; i++)

{

for (int j = 0; j < N; j++)

{

lines[i][j] = rand() % 21 - 10;

}

}

}

void delete\_twodimensional\_array(int\*\*& lines, int M, int N)

{

for (int i = 0; i < M; i++)

{

delete lines[i];

}

delete[] lines;

}

void print\_onedim\_array(int\* array, int n)

{

for (int i = 0; i < n; i++)

{

cout << " " << array[i] << " ";

}

cout << endl;

}

bool inArray(int\* arr, int need, int count)

{

for (int i = 0; i < count; i++)

if (arr[i] == need)

{

return true;

}

return false;

}

int\* uniq(int\* arr, int size, int& newSize)

{

int\* localArr = new int[size] {0};

for (int i = 0; i < size; i++)

{

if (inArray(localArr - newSize, arr[i], size))

continue;

++newSize;

\*localArr++ = arr[i];

}

return localArr - newSize;

}