Задача 6. Одномерные массивы

1. Дан массив размера *N* и целые числа *K* и *L* (1 ≤ *K* ≤ *L* ≤ *N*). Найти среднее арифметическое элементов массива с номерами от *K* до *L* включительно.

#include <iostream>

using namespace std;

int main() {

const int N = 100;

int arr[N], k, l;

double sum = 0;

int count = 0;

cin >> k >> l;

for (int i = 0; i < N; i++) {

cin >> arr[i];

if (i >= k - 1 && i <= l - 1) {

sum += arr[i];

count++;

}

}

cout << sum / count << endl;

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

2. Дан массив размера *N* и целые числа *K* и *L* (1 < *K* ≤ *L* ≤ *N*). Найти сумму всех элементов мас- сива, кроме элементов с номерами от *K* до *L* включительно.

#include <iostream>

using namespace std;

int main() {

const int N = 100;

int arr[N], k, l;

int sum = 0;

cin >> k >> l;

for (int i = 0; i < N; i++) {

cin >> arr[i];

if (i < k - 1 || i > l - 1) {

sum += arr[i];

}

}

cout << sum << endl;

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

3. Дан массив размера *N* и целые числа *K* и *L* (1 < *K* ≤ *L* ≤ *N*). Найти среднее арифметическое всех элементов массива, кроме элементов с номерами от *K* до *L* включительно.

*#include <iostream>*

*using namespace std;*

*int main() {*

*const int N = 100;*

*int arr[N], k, l;*

*double sum = 0;*

*int count = 0;*

*cin >> k >> l;*

*for (int i = 0; i < N; i++) {*

*cin >> arr[i];*

*if (i < k - 1 || i > l - 1) {*

*sum += arr[i];*

*count++;*

*}*

*}*

*cout << sum / count << endl;*

*return 0;*

*}*

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

4. Дан целочисленный массив размера *N*. Вывести вначале все содержащиеся в данном массиве четные числа в порядке возрастания их индексов, а затем – все нечетные числа в порядке убывания их индексов.

#include <iostream>

using namespace std;

int main() {

const int N = 100;

int arr[N];

int even[N], odd[N];

int even\_count = 0, odd\_count = 0;

for (int i = 0; i < N; i++) {

cin >> arr[i];

if (arr[i] % 2 == 0) {

even[even\_count] = arr[i];

even\_count++;

} else {

odd[odd\_count] = arr[i];

odd\_count++;

}

}

for (int i = 0; i < even\_count; i++) {

cout << even[i] << " ";

}

for (int i = odd\_count - 1; i >= 0; i--) {

cout << odd[i] << " ";

}

cout << endl;

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

5. Дан массив *A* размера *N*. Вывести вначале его элементы с четными номерами (в порядке воз- растания номеров), а затем – элементы с нечетными номерами (также в порядке возрастания номеров).

#include <iostream>

using namespace std;

int main() {

const int N = 100;

int arr[N];

for (int i = 0; i < N; i++) {

cin >> arr[i];

}

for (int i = 0; i < N; i += 2) {

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

}

for (int i = 1; i < N; i += 2) {

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

}

cout << endl;

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

6. Дан массив размера *N*. Найти произведение всех элементов массива, расположенных справа от максимального элементами, не включая максимальный элемент.

#include <iostream>

using namespace std;

int main() {

const int N = 100;

int arr[N];

int maxIndex = 0;

int product = 1;

// Ввод массива

for (int i = 0; i < N; i++) {

cin >> arr[i];

// Находим индекс максимального элемента

if (arr[i] > arr[maxIndex]) {

maxIndex = i;

}

}

// Вычисление произведения

for (int i = maxIndex + 1; i < N; i++) {

product \*= arr[i];

}

cout << product << endl;

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

7. Дан массив размера *N*. Найти сумму всех элементов массива, расположенных между его ми- нимальным и максимальным элементами, включая минимальный и максимальный элементы.

#include <iostream>

using namespace std;

int main() {

const int N = 100;

int arr[N];

int minIndex = 0, maxIndex = 0, sum = 0;

// Ввод массива

for (int i = 0; i < N; i++) {

cin >> arr[i];

// Находим индексы минимального и максимального элементов

if (arr[i] < arr[minIndex]) {

minIndex = i;

}

if (arr[i] > arr[maxIndex]) {

maxIndex = i;

}

}

// Находим сумму элементов между минимальным и максимальным

if (minIndex < maxIndex) {

for (int i = minIndex; i <= maxIndex; i++) {

sum += arr[i];

}

} else {

for (int i = maxIndex; i <= minIndex; i++) {

sum += arr[i];

}

}

cout << sum << endl;

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

8. Дан массив размера *N*. Обнулить элементы массива, расположенные между его минимальным и максимальным элементами (не включая минимальный и максимальный элементы).

#include <iostream>

#include <algorithm>

using namespace std;

int main() {

const int N = 10;

int a[N] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

int min\_index = min\_element(a, a+N) - a;

int max\_index = max\_element(a, a+N) - a;

if (min\_index > max\_index) swap(min\_index, max\_index);

for (int i = min\_index+1; i < max\_index; ++i) {

a[i] = 0;

}

for (int i = 0; i < N; ++i) {

cout << a[i] << " ";

}

cout << endl;

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

9. Дан целочисленный массив размера *N*. Увеличить все четные числа, содержащиеся в массиве, на заданное число. Если четные числа в массиве отсутствуют, то оставить массив без измене- ний.

#include <iostream>

using namespace std;

int main() {

const int N = 10;

int a[N] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

int delta = 2;

bool has\_even = false;

for (int i = 0; i < N; ++i) {

if (a[i] % 2 == 0) {

has\_even = true;

a[i] += delta;

}

}

if (has\_even) {

for (int i = 0; i < N; ++i) {

cout << a[i] << " ";

}

cout << endl;

} else {

cout << "No even numbers found" << endl;

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

10. Дан целочисленный массив размера *N*. Обнулить все нечетные числа, содержащиеся в масси- ве. Если нечетные числа в массиве отсутствуют, то оставить массив без изменений.

#include <iostream>

using namespace std;

int main() {

const int N = 10;

int a[N] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

bool has\_odd = false;

for (int i = 0; i < N; ++i) {

if (a[i] % 2 != 0) {

has\_odd = true;

a[i] = 0;

}

}

if (has\_odd) {

for (int i = 0; i < N; ++i) {

cout << a[i] << " ";

}

cout << endl;

} else {

cout << "No odd numbers found" << endl;

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

11. Дан массив *A* размера *N*. Найти минимальный элемент из его элементов с четными номерами: *A*2, *A*4, *A*6, ... .

#include <iostream>

using namespace std;

int main()

{

const int N = 10; // размер массива

int A[N] = {5, 3, 8, 2, 9, 1, 6, 4, 7, 0}; // инициализация массива

int min\_even = A[1]; // начальное значение минимального четного элемента

for(int i = 3; i < N; i += 2) // цикл для перебора четных элементов массива

{

if(A[i] < min\_even) // если текущий четный элемент меньше минимального, то обновляем минимальное значение

{

min\_even = A[i];

}

}

cout << "Минимальный элемент из четных элементов массива: " << min\_even << endl;

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

12. Дан массив размера *N*. Найти номера тех элементов массива, которые больше своего правого соседа, и количество таких элементов. Найденные номера выводить в порядке их возрастания.

#include <iostream>

#include <vector>

int main() {

int n;

std::cin >> n;

std::vector<int> a(n);

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

std::cin >> a[i];

}

std::vector<int> indices;

for (int i = 0; i < n - 1; i++) {

if (a[i] > a[i + 1]) {

indices.push\_back(i);

}

}

std::cout << "Indices of elements greater than their right neighbor: ";

for (int i : indices) {

std::cout << i << " ";

}

std::cout << "\n";

std::cout << "Number of elements greater than their right neighbor: " << indices.size() << "\n";

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

13. Дан массив размера *N*. Найти два соседних элемента, сумма которых максимальна, и вывести эти элементы в порядке возрастания их индексов.

#include <iostream>

#include <vector>

int main() {

std::vector<int> a = {2, 5, 1, 6, 3, 9};

int max\_sum = a[0] + a[1];

int max\_index = 0;

for (int i = 1; i < a.size() - 1; i++) {

int sum = a[i] + a[i+1];

if (sum > max\_sum) {

max\_sum = sum;

max\_index = i;

}

}

std::cout << "Max sum is: " << max\_sum << std::endl;

std::cout << "Indices of max sum elements are: " << max\_index << " and " << max\_index + 1 << std::endl;

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

14. Дан массив размера *N*. Поменять местами его минимальный и максимальный элементы.

#include <iostream>

#include <algorithm>

using namespace std;

int main()

{

int n;

cin >> n;

int a[n];

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

cin >> a[i];

int min\_idx = min\_element(a, a + n) - a;

int max\_idx = max\_element(a, a + n) - a;

swap(a[min\_idx], a[max\_idx]);

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

cout << a[i] << " ";

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

15. Даны целые числа *N* (> 2), *A* и *B*. Сформировать и вывести целочисленный массив размера *N*, первый элемент которого равен *A*, второй равен *B*, а каждый последующий элемент равен сумме всех предыдущих.

#include <iostream>

using namespace std;

int main() {

int n, a, b;

cin >> n >> a >> b;

int arr[n];

arr[0] = a;

arr[1] = b;

for (int i = 2; i < n; i++) {

arr[i] = arr[i-1] + arr[i-2];

}

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

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

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Задача 7. Многомерные массивы

1. Дана целочисленная матрица размера *M*×*N*. Найти номер первого из ее столбцов, содержащих только нечетные числа. Если таких столбцов нет, то вывести 0.

#include <iostream>

using namespace std;

int main() {

int M, N;

cin >> M >> N;

int a[M][N];

bool flag;

for (int i = 0; i < N; i++) {

flag = true;

for (int j = 0; j < M; j++) {

cin >> a[j][i];

if (a[j][i] % 2 == 0) {

flag = false;

}

}

if (flag) {

cout << i + 1 << endl;

return 0;

}

}

cout << 0 << endl;

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

2. Дана матрица размера *M*×*N*. Преобразовать матрицу, поменяв местами минимальный и мак- симальный элемент в каждой строке.

#include <iostream>

#include <limits>

using namespace std;

int main() {

const int M = 3, N = 4; // размеры матрицы

int matrix[M][N] = {{3, 2, 5, 4}, {1, 7, 6, 8}, {9, 11, 10, 12}}; // исходная матрица

// проходим по каждой строке матрицы

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

int min\_val = numeric\_limits<int>::max(); // минимальное значение в строке

int max\_val = numeric\_limits<int>::min(); // максимальное значение в строке

int min\_index = 0; // индекс минимального элемента в строке

int max\_index = 0; // индекс максимального элемента в строке

// находим минимальный и максимальный элементы в строке

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

if (matrix[i][j] < min\_val) {

min\_val = matrix[i][j];

min\_index = j;

}

if (matrix[i][j] > max\_val) {

max\_val = matrix[i][j];

max\_index = j;

}

}

// меняем местами минимальный и максимальный элементы в строке

int temp = matrix[i][min\_index];

matrix[i][min\_index] = matrix[i][max\_index];

matrix[i][max\_index] = temp;

}

// выводим измененную матрицу

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

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

cout << matrix[i][j] << " ";

}

cout << endl;

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

3. Дана матрица размера *M*×*N*. Поменять местами строки, содержащие минимальный и макси- мальный элементы матрицы.

#include <iostream>

#include <climits>

using namespace std;

int main() {

const int M = 3, N = 4; // размеры матрицы

int a[M][N] = {{2, 1, 4, 6}, {3, 5, 7, 8}, {9, 0, 4, 1}}; // исходная матрица

int min\_row = 0, max\_row = 0; // номера строк с минимальным и максимальным элементами

int min\_elem = INT\_MAX, max\_elem = INT\_MIN; // минимальный и максимальный элементы матрицы

// находим строки с минимальным и максимальным элементами

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

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

if (a[i][j] < min\_elem) {

min\_elem = a[i][j];

min\_row = i;

}

if (a[i][j] > max\_elem) {

max\_elem = a[i][j];

max\_row = i;

}

}

}

// меняем местами строки

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

swap(a[min\_row][j], a[max\_row][j]);

}

// выводим измененную матрицу

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

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

cout << a[i][j] << " ";

}

cout << endl;

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

4. Дана матрица размера *M*×*N*. Зеркально отразить ее элементы относительно горизонтальной оси симметрии матрицы (при этом поменяются местами строки с номерами 1 и *M*, 2 и *M* – 1 и т. д.).

#include <iostream>

using namespace std;

int main() {

int m, n;

cin >> m >> n;

int matrix[m][n];

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

for (int j = 0; j < n; j++) {

cin >> matrix[i][j];

}

}

for (int i = 0; i < m/2; i++) {

for (int j = 0; j < n; j++) {

swap(matrix[i][j], matrix[m-1-i][j]);

}

}

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

for (int j = 0; j < n; j++) {

cout << matrix[i][j] << " ";

}

cout << endl;

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

5. Дана матрица размера *M*×*N*. Удалить строку, содержащую минимальный элемент матрицы.

#include <iostream>

#include <climits>

using namespace std;

int main() {

int m, n;

cout << "Введите размер матрицы MxN: ";

cin >> m >> n;

int matrix[m][n];

cout << "Введите элементы матрицы:\n";

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

for (int j = 0; j < n; j++) {

cin >> matrix[i][j];

}

}

int minRow = -1, minValue = INT\_MAX;

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

for (int j = 0; j < n; j++) {

if (matrix[i][j] < minValue) {

minValue = matrix[i][j];

minRow = i;

}

}

}

if (minRow != -1) {

for (int i = minRow; i < m - 1; i++) {

for (int j = 0; j < n; j++) {

matrix[i][j] = matrix[i + 1][j];

}

}

m--;

}

cout << "Матрица после удаления строки с минимальным элементом:\n";

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

for (int j = 0; j < n; j++) {

cout << matrix[i][j] << " ";

}

cout << endl;

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

6. Дана матрица размера *M*×*N*. Удалить столбец, содержащий максимальный элемент матрицы.

#include <iostream>

using namespace std;

int main() {

const int M = 3, N = 4; // размеры матрицы

int matrix[M][N] = {{3, 2, 5, 6}, {1, 4, 3, 2}, {9, 2, 7, 8}}; // исходная матрица

int max\_col = 0; // индекс столбца с максимальным элементом

int max\_elem = matrix[0][0]; // максимальный элемент

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

if (matrix[0][j] > max\_elem) {

max\_elem = matrix[0][j];

max\_col = j;

}

}

// удаляем столбец с максимальным элементом

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

for (int j = max\_col; j < N - 1; j++) {

matrix[i][j] = matrix[i][j+1];

}

}

N--; // уменьшаем ширину матрицы на 1

// выводим новую матрицу

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

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

cout << matrix[i][j] << " ";

}

cout << endl;

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

7. Дана матрица размера *M*×*N*. Зеркально отразить ее элементы относительно вертикальной оси симметрии матрицы (при этом поменяются местами столбцы с номерами 1 и *N*, 2 и *N* – 1 и т. д.).

#include <iostream>

using namespace std;

int main()

{

int m, n;

cout << "Enter matrix dimensions (m n): ";

cin >> m >> n;

int matrix[m][n];

cout << "Enter matrix elements:" << endl;

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

{

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

{

cin >> matrix[i][j];

}

}

// Зеркальное отражение элементов относительно вертикальной оси

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

{

for (int j = 0; j < n / 2; j++)

{

int temp = matrix[i][j];

matrix[i][j] = matrix[i][n - 1 - j];

matrix[i][n - 1 - j] = temp;

}

}

cout << "Mirrored matrix:" << endl;

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

{

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

{

cout << matrix[i][j] << " ";

}

cout << endl;

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

8. Дана матрица размера *M*×*N*. Продублировать строку матрицы, содержащую ее максимальный элемент.

#include <iostream>

#include <algorithm>

#include <vector>

int main() {

int M, N;

std::cin >> M >> N;

std::vector<std::vector<int>> matrix(M, std::vector<int>(N));

int max\_element\_row = -1, max\_element\_col = -1;

int max\_element = INT\_MIN;

// читаем матрицу и находим максимальный элемент и его позицию

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

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

std::cin >> matrix[i][j];

if (matrix[i][j] > max\_element) {

max\_element = matrix[i][j];

max\_element\_row = i;

max\_element\_col = j;

}

}

}

// дублируем строку с максимальным элементом

matrix.insert(matrix.begin() + max\_element\_row + 1, matrix[max\_element\_row]);

++M;

// выводим результат

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

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

std::cout << matrix[i][j] << " ";

}

std::cout << std::endl;

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

9. Дана матрица размера *M*×*N*. В каждом ее столбце найти количество элементов, больших среднего арифметического всех элементов этого столбца.

#include <iostream>

using namespace std;

int main() {

const int M = 3;

const int N = 3;

int a[M][N] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

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

int sum = 0;

int count = 0;

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

sum += a[i][j];

count++;

}

double average = sum / count;

int above\_average = 0;

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

if (a[i][j] > average) {

above\_average++;

}

}

cout << "In column " << j+1 << ", there are " << above\_average << " elements above average." << endl;

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

10. Дана матрица размера *M*×*N* (*M* и *N* – четные числа). Поменять местами левую верхнюю и пра- вую нижнюю четверти матрицы.

#include <iostream>

using namespace std;

int main()

{

const int M = 4; // количество строк

const int N = 4; // количество столбцов

int a[M][N]; // матрица размера M x N

// заполнение матрицы

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

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

a[i][j] = i \* N + j + 1;

// вывод исходной матрицы

cout << "Исходная матрица:" << endl;

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

{

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

cout << a[i][j] << " ";

cout << endl;

}

// обмен левой верхней и правой нижней четвертей

for (int i = 0; i < M/2; i++)

for (int j = 0; j < N/2; j++)

swap(a[i][j], a[i+M/2][j+N/2]);

// вывод преобразованной матрицы

cout << "Преобразованная матрица:" << endl;

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

{

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

cout << a[i][j] << " ";

cout << endl;

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

11. Дана матрица размера *M*×*N*. Для каждой строки матрицы с нечетным номером (1, 3, ...) найти среднее арифметическое ее элементов.

#include <iostream>

using namespace std;

int main() {

int m, n;

cout << "Введите размер матрицы MxN: ";

cin >> m >> n;

int matrix[m][n];

cout << "Введите элементы матрицы:\n";

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

for (int j = 0; j < n; j++) {

cin >> matrix[i][j];

}

}

for (int i = 0; i < m; i += 2) {

int sum = 0, count = 0;

for (int j = 0; j < n; j++) {

sum += matrix[i][j];

count++;

}

double avg = (double) sum / count;

cout << "Среднее арифметическое элементов строки " << i+1 << ": " << avg << endl;

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

12. Дана матрица размера *M*×*N*. Найти номер ее строки с наибольшей суммой элементов и вывес- ти данный номер, а также значение наибольшей суммы.

#include <iostream>

using namespace std;

int main() {

int m, n;

cin >> m >> n;

int maxSum = -1, maxSumRow = -1;

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

int rowSum = 0;

for (int j = 0; j < n; j++) {

int x;

cin >> x;

rowSum += x;

}

if (rowSum > maxSum) {

maxSum = rowSum;

maxSumRow = i;

}

}

cout << maxSumRow << " " << maxSum << endl;

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

13. Дана матрица размера *M*×*N*. Перед строкой матрицы, содержащей минимальный элемент мат- рицы, вставить строку из нулей.

#include <iostream>

#include <algorithm>

#include <vector>

using namespace std;

int main() {

int M, N;

cin >> M >> N;

vector<vector<int>> matrix(M, vector<int>(N));

int min\_row = 0, min\_val = INT\_MAX;

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

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

cin >> matrix[i][j];

if(matrix[i][j] < min\_val) {

min\_val = matrix[i][j];

min\_row = i;

}

}

}

matrix.insert(matrix.begin() + min\_row, vector<int>(N, 0));

for(auto& row : matrix) {

for(int i = 0; i < N; i++) {

cout << row[i] << " ";

}

cout << endl;

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

14. Дана целочисленная матрица размера *M*×*N*. Найти номер последней из ее строк, содержащих только четные числа. Если таких строк нет, то вывести 0.

#include <iostream>

using namespace std;

int main() {

const int M = 3, N = 4; // размер матрицы

int matrix[M][N] = {{1, 2, 3, 4}, {6, 8, 10, 12}, {0, 3, 5, 7}}; // матрица

int last\_even\_row = 0; // переменная для хранения номера последней строки с четными числами

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

bool only\_even = true; // флаг для проверки, содержит ли строка только четные числа

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

if (matrix[i][j] % 2 != 0) { // если нашли нечетное число, то строка не содержит только четные числа

only\_even = false;

break;

}

}

if (only\_even) { // если строка содержит только четные числа, то запоминаем ее номер

last\_even\_row = i;

}

}

if (last\_even\_row == 0) { // если не нашли ни одной строки с четными числами, то выводим 0

cout << "0" << endl;

} else { // иначе выводим номер последней строки с четными числами

cout << last\_even\_row + 1 << endl;

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

15. Дана матрица размера *M*×*N*. Поменять местами столбец с номером 1 и последний из столбцов, содержащих только положительные элементы. Если требуемых столбцов нет, то вывести мат- рицу без изменений.

#include <iostream>

#include <vector>

using namespace std;

int main() {

int M, N;

cin >> M >> N;

vector<vector<int>> matrix(M, vector<int>(N));

// заполнение матрицы

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

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

cin >> matrix[i][j];

}

}

int pos\_last\_even\_col = -1;

for (int j = N - 1; j >= 0; j--) {

bool is\_even = true;

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

if (matrix[i][j] % 2 != 0) {

is\_even = false;

break;

}

}

if (is\_even) {

pos\_last\_even\_col = j;

break;

}

}

if (pos\_last\_even\_col != -1) {

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

swap(matrix[i][0], matrix[i][pos\_last\_even\_col]);

}

}

// вывод матрицы

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

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

cout << matrix[i][j] << " ";

}

cout << endl;

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Задача 8. Работа со строками

1. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Найти количество слов в строке.

#include <iostream>

#include <string>

#include <algorithm>

int main() {

std::string str;

std::getline(std::cin, str); // ввод строки

int count = std::count\_if(str.begin(), str.end(), [](char c) {

return std::isspace(c); // проверка на пробел

});

std::cout << count + 1 << std::endl; // вывод количества слов

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

2. Дана строка, состоящая из русских слов, набранных заглавными буквами и разделенных про- белами (одним или несколькими). Найти количество слов, которые начинаются и заканчива- ются одной и той же буквой.

#include <iostream>

#include <string>

#include <sstream>

using namespace std;

int main() {

string input;

getline(cin, input); // вводим строку

stringstream ss(input);

string word;

int count = 0;

while (ss >> word) {

if (word.front() == word.back()) {

count++;

}

}

cout << count << endl;

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

3. Дана строка, состоящая из русских слов, набранных заглавными буквами и разделенных про- белами (одним или несколькими). Найти количество слов, которые содержат хотя бы одну букву «А».

#include <iostream>

#include <string>

int main() {

std::string str;

std::getline(std::cin, str); // ввод строки

int count = 0;

bool found\_a = false; // флаг нахождения буквы А в текущем слове

for (int i = 0; i < str.size(); i++) {

if (str[i] == ' ') { // если текущий символ - пробел, значит, слово закончилось

if (found\_a) { // если в слове была найдена буква А, увеличиваем счетчик

count++;

}

found\_a = false; // обнуляем флаг

} else if (str[i] == 'А') { // если текущий символ - буква А, устанавливаем флаг

found\_a = true;

}

}

if (found\_a) { // учитываем последнее слово, если в нем была найдена буква А

count++;

}

std::cout << count << std::endl; // выводим количество слов с буквой А

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

4. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Найти длину самого короткого слова.

#include <iostream>

#include <string>

using namespace std;

int main() {

string s;

getline(cin, s); // ввод строки с пробелами

int shortest\_word\_length = s.length(); // начальное значение для длины самого короткого слова

int current\_word\_length = 0; // текущая длина слова

for (int i = 0; i < s.length(); i++) {

if (s[i] != ' ') { // если символ не пробел, увеличиваем длину текущего слова

current\_word\_length++;

} else { // если символ пробел, сравниваем длину текущего слова с длиной самого короткого слова

if (current\_word\_length < shortest\_word\_length) {

shortest\_word\_length = current\_word\_length;

}

current\_word\_length = 0; // сброс текущей длины слова для следующего слова

}

}

// проверяем длину последнего слова после последнего пробела

if (current\_word\_length < shortest\_word\_length) {

shortest\_word\_length = current\_word\_length;

}

cout << shortest\_word\_length << endl; // выводим длину самого короткого слова

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

5. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Вывести строку, содержащую эти же слова, разделенные одним пробелом и расположенные в обратном порядке.

#include <iostream>

#include <string>

#include <sstream>

#include <algorithm>

int main() {

std::string input, word;

std::getline(std::cin, input); // Считываем входную строку

std::stringstream ss(input);

std::string reversed; // Переменная для хранения перевернутой строки

while (ss >> word) { // Читаем слова из строки

reversed = word + " " + reversed; // Добавляем текущее слово в начало перевернутой строки

}

std::cout << reversed << std::endl; // Выводим перевернутую строку

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

6. Дана строка, состоящая из русских слов, набранных заглавными буквами и разделенных про- белами (одним или несколькими). Вывести строку, содержащую эти же слова, разделенные одним пробелом и расположенные в алфавитном порядке.

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

// Функция разбиения строки на слова

vector<string> split(string str, char delimiter) {

vector<string> words;

string word;

for (char c : str) {

if (c == delimiter) {

if (!word.empty()) {

words.push\_back(word);

word.clear();

}

} else {

word += c;

}

}

if (!word.empty()) {

words.push\_back(word);

}

return words;

}

// Функция объединения вектора слов в строку

string join(vector<string> words, string delimiter) {

string str;

for (int i = 0; i < words.size(); i++) {

str += words[i];

if (i != words.size() - 1) {

str += delimiter;

}

}

return str;

}

int main() {

string str;

getline(cin, str);

vector<string> words = split(str, ' ');

sort(words.begin(), words.end());

string sorted\_str = join(words, " ");

cout << sorted\_str << endl;

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

7. Дан символ *C* и строки *S*, *S*0. Перед каждым вхождением символа *C* в строку *S* вставить стро- ку *S*0.

#include <iostream>

#include <string>

int main() {

char c = 'a'; // символ C

std::string s = "hello world"; // строка S

std::string s0 = "test"; // строка S0

// Перед каждым вхождением символа C в строку S вставить строку S0

std::size\_t pos = 0;

while ((pos = s.find(c, pos)) != std::string::npos) {

s.insert(pos, s0);

pos += s0.length() + 1;

}

// Вывести результат

std::cout << s << std::endl;

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

8. Дана строка-предложение с избыточными пробелами между словами. Преобразовать ее так, чтобы между словами был ровно один пробел.

#include <algorithm>

#include <string>

std::string sentence = "Это строка с избыточными пробелами.";

sentence.erase(std::unique(sentence.begin(), sentence.end(),

[](char a, char b){ return a == ' ' && b == ' '; }),

sentence.end());

if (sentence.front() == ' ') sentence.erase(sentence.begin());

if (sentence.back() == ' ') sentence.pop\_back();

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

9. Дана строка. Подсчитать общее количество содержащихся в ней строчных латинских и рус- ских букв.

#include <iostream>

#include <string>

using namespace std;

int main() {

string s;

getline(cin, s);

int count = 0;

for (char c : s) {

if (islower(c) && isalpha(c)) {

count++;

}

}

cout << "Total number of lowercase letters: " << count << endl;

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

10. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Найти длину самого длинного слова.

#include <iostream>

#include <string>

using namespace std;

int main() {

string s;

getline(cin, s); // ввод строки с пробелами

int max\_len = 0, cur\_len = 0;

for (char c : s) {

if (c == ' ') {

max\_len = max(max\_len, cur\_len); // обновляем максимальную длину

cur\_len = 0; // обнуляем текущую длину

} else {

cur\_len++; // увеличиваем текущую длину

}

}

max\_len = max(max\_len, cur\_len); // учитываем последнее слово

cout << max\_len << endl; // выводим результат

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

11. Дана строка-предложение. Зашифровать ее, поместив вначале все символы, расположенные на четных позициях строки, а затем, в обратном порядке, все символы, расположенные на не- четных позициях (например, строка «Программа» превратится в «ргамамроП»).

#include <iostream>

#include <string>

using namespace std;

int main() {

string str;

getline(cin, str);

string even, odd;

for(int i = 0; i < str.length(); i++) {

if(i % 2 == 0) even += str[i];

else odd += str[i];

}

reverse(odd.begin(), odd.end());

cout << even << odd << endl;

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

12. Дана непустая строка *S* и целое число *N* (> 0). Вывести строку, содержащую символы строки *S*, между которыми вставлено по *N* символов «\*» (звездочка).

#include <iostream>

#include <string>

int main() {

std::string s;

int n;

std::cin >> s >> n;

for (int i = 0; i < s.length(); i++) {

std::cout << s[i];

if ((i + 1) < s.length() && (i + 1) % n == 0) {

std::cout << "\*";

}

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

13. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Вывести строку, содержащую эти же слова, разделенные одним символом «.» (точка). В конце строки точку не ставить.

#include <iostream>

#include <string>

using namespace std;

int main()

{

string s;

getline(cin, s);

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

{

if(s[i] == ' ')

{

cout << ".";

}

else

{

cout << s[i];

}

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

14. Дана строка, содержащая *полное имя файла*, то есть имя диска, список каталогов (путь), соб- ственно имя и расширение (например, d:\ivanov\primer\unit1.pas). Выделить из этой строки имя файла (без расширения).

#include <iostream>

#include <string>

using namespace std;

int main()

{

string path = "d:\\ivanov\\primer\\unit1.pas";

string fileName = path.substr(path.find\_last\_of("\\/") + 1); // извлекаем имя файла с расширением

string fileBaseName = fileName.substr(0, fileName.find\_last\_of(".")); // извлекаем имя файла без расширения

cout << fileBaseName << endl; // выводим имя файла без расширения

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

15. Дана строка, состоящая из русских слов, разделенных пробелами (одним или несколькими). Вывести самое длинное слово в предложении. Если таких слов несколько, то вывести послед- нее из них.

#include <iostream>

#include <string>

using namespace std;

int main() {

string s, longestWord;

int maxLength = 0;

getline(cin, s);

for (int i = 0; i < s.length(); i++) {

if (s[i] != ' ') {

string currentWord = "";

while (s[i] != ' ' && i < s.length()) {

currentWord += s[i];

i++;

}

if (currentWord.length() >= maxLength) {

maxLength = currentWord.length();

longestWord = currentWord;

}

}

}

cout << longestWord;

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Задача 9. Сортировка массивов

Использовать один из простых методов сортировки (вставкой, обменом или выбором).

1. Дан одномерный массив целых чисел. Выбрать из массива все нечетные числа и упорядо- чить их по убыванию.

#include <iostream>

using namespace std;

int main() {

int arr[] = {4, 2, 8, 5, 1, 7, 3, 6}; // исходный массив

int n = sizeof(arr)/sizeof(arr[0]); // размер массива

int odd[n], j = 0; // массив для нечетных чисел и его размер

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

if (arr[i] % 2 != 0) {

odd[j] = arr[i];

j++;

}

}

// сортировка выбором

for (int i = 0; i < j-1; i++) {

int maxIndex = i;

for (int k = i+1; k < j; k++) {

if (odd[k] > odd[maxIndex]) {

maxIndex = k;

}

}

int temp = odd[i];

odd[i] = odd[maxIndex];

odd[maxIndex] = temp;

}

// вывод отсортированных нечетных чисел

for (int i = 0; i < j; i++) {

cout << odd[i] << " ";

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

2. Дан одномерный массив целых чисел. Выбрать из массива все положительные числа и упо- рядочить их по возрастанию.

#include <iostream>

using namespace std;

int main()

{

const int SIZE = 10;

int arr[SIZE] = {4, -2, 8, 0, -5, 1, 3, -9, 6, -7};

int sorted[SIZE];

int i, j, key, n;

n = 0;

for (i = 0; i < SIZE; i++) {

if (arr[i] > 0) {

key = arr[i];

j = n;

while (j > 0 && sorted[j - 1] > key) {

sorted[j] = sorted[j - 1];

j--;

}

sorted[j] = key;

n++;

}

}

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

cout << sorted[i] << " ";

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

3. Дан одномерный массив целых чисел. Выбрать из массива все положительные числа и упо- рядочить их по убыванию.

#include <iostream>

void selectionSort(int arr[], int n) {

for (int i = 0; i < n - 1; i++) {

int max\_idx = i;

for (int j = i + 1; j < n; j++) {

if (arr[j] > arr[max\_idx]) {

max\_idx = j;

}

}

std::swap(arr[i], arr[max\_idx]);

}

}

int main() {

const int n = 10;

int arr[n] = {1, 4, -2, 7, 5, -3, 8, 10, -6, 2};

// выбираем положительные элементы

int pos[n];

int pos\_size = 0;

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

if (arr[i] > 0) {

pos[pos\_size++] = arr[i];

}

}

selectionSort(pos, pos\_size);

for (int i = 0; i < pos\_size; i++) {

std::cout << pos[i] << " ";

}

std::cout << std::endl;

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

4. Дан одномерный массив целых чисел. Выбрать из массива все четные числа и упорядочить их по возрастанию.

#include <iostream>

using namespace std;

int main() {

const int n = 10;

int a[n] = {2, 8, 4, 5, 7, 3, 6, 1, 9, 0};

int b[n], j = 0;

// Выбираем все четные числа и копируем их в массив b

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

if (a[i] % 2 == 0) {

b[j] = a[i];

j++;

}

}

// Сортируем массив b по возрастанию

for (int i = 1; i < j; i++) {

int k = b[i];

int l = i - 1;

while (l >= 0 && b[l] > k) {

b[l + 1] = b[l];

l--;

}

b[l + 1] = k;

}

// Выводим отсортированный массив b

for (int i = 0; i < j; i++) {

cout << b[i] << " ";

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

5. Дан одномерный массив целых чисел. Выбрать из массива все числа больше заданного чис- ла *k* и упорядочить их по убыванию.

#include <iostream>

using namespace std;

int main()

{

const int n = 10; // размер массива

int a[n] = {2, 4, 5, 1, 7, 8, 9, 3, 6, 0}; // исходный массив

int k = 4; // заданное число k

int temp;

// выбираем элементы массива, которые больше k

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

{

if (a[i] > k)

{

// сортируем выбранные элементы по убыванию

for (int j = i + 1; j < n; j++)

{

if (a[j] > k && a[j] > a[i])

{

temp = a[i];

a[i] = a[j];

a[j] = temp;

}

}

}

}

// выводим отсортированный массив

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

{

if (a[i] > k)

{

cout << a[i] << " ";

}

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

6. Дан одномерный массив целых чисел. Выбрать из массива все числа больше 10 и упорядо- чить их по возрастанию.

#include <iostream>

using namespace std;

int main() {

const int n = 10;

int arr[n] = {5, 2, 15, 9, 12, 7, 20, 3, 8, 11};

int k = 10; // заданное число k

int count = 0;

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

if (arr[i] > k) {

count++;

}

}

int temp;

for (int i = 0; i < count-1; i++) {

int max\_idx = i;

for (int j = i+1; j < n; j++) {

if (arr[j] > arr[max\_idx] && arr[j] > k) {

max\_idx = j;

}

}

if (max\_idx != i) {

temp = arr[i];

arr[i] = arr[max\_idx];

arr[max\_idx] = temp;

}

}

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

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

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

7. Дан одномерный массив целых чисел. Выбрать из массива все числа кратные 5 и упорядо- чить их по убыванию.

#include <iostream>

using namespace std;

int main() {

const int n = 10;

int arr[n] = {20, 25, 10, 8, 15, 30, 5, 18, 35, 40};

int k = 5;

// Выборка чисел кратных 5

int b[n], j = 0;

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

if (arr[i] % k == 0) {

b[j] = arr[i];

j++;

}

}

// Сортировка выбранных чисел по убыванию

for (int i = 0; i < j - 1; i++) {

for (int l = i + 1; l < j; l++) {

if (b[i] < b[l]) {

int temp = b[i];

b[i] = b[l];

b[l] = temp;

}

}

}

// Вывод отсортированных чисел

for (int i = 0; i < j; i++) {

cout << b[i] << " ";

}

cout << endl;

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

8. Дан одномерный массив целых чисел. Выбрать из массива все числа меньше заданного чис- ла *k* и упорядочить их по возрастанию.

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main()

{

int k = 10;

vector<int> arr = {4, 7, 1, 9, 2, 5, 8};

vector<int> result;

for (int i = 0; i < arr.size(); i++) {

if (arr[i] < k) {

result.push\_back(arr[i]);

}

}

sort(result.begin(), result.end());

for (int i = 0; i < result.size(); i++) {

cout << result[i] << " ";

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

9. Дан одномерный массив целых чисел. Выбрать из массива все числа меньше 15 и упорядо- чить их по убыванию.

#include <iostream>

using namespace std;

int main() {

int arr[] = {5, 8, 12, 3, 17, 10, 6, 20};

int n = sizeof(arr) / sizeof(arr[0]);

int k = 15;

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

if(arr[i] < k) {

for(int j=i+1; j<n; j++) {

if(arr[j] < k && arr[i] < arr[j]) {

swap(arr[i], arr[j]);

}

}

}

}

cout << "Sorted array: ";

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

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

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

10. Дан одномерный массив целых чисел. Выбрать из массива все числа кратные 3 и упорядо- чить их по возрастанию.

#include <iostream>

using namespace std;

void selectionSort(int arr[], int n) {

int i, j, min\_idx;

for (i = 0; i < n-1; i++) {

min\_idx = i;

for (j = i+1; j < n; j++)

if (arr[j] % 3 == 0 && arr[j] < arr[min\_idx])

min\_idx = j;

swap(arr[min\_idx], arr[i]);

}

}

int main() {

int arr[] = {8, 12, 9, 18, 6, 21};

int n = sizeof(arr) / sizeof(arr[0]);

selectionSort(arr, n);

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

if (arr[i] % 3 == 0)

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

cout << endl;

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

11. Дан одномерный массив целых чисел. Выбрать из массива все числа кратные заданному числу *k* и упорядочить их по убыванию.

#include <iostream>

using namespace std;

void selectionSort(int arr[], int n) {

for (int i = 0; i < n - 1; i++) {

int minIndex = i;

for (int j = i + 1; j < n; j++) {

if (arr[j] % k == 0 && arr[j] > arr[minIndex]) {

minIndex = j;

}

}

if (minIndex != i) {

swap(arr[i], arr[minIndex]);

}

}

}

int main() {

int arr[] = {10, 15, 20, 25, 30, 35, 40, 45, 50, 55};

int n = sizeof(arr) / sizeof(arr[0]);

int k = 5;

selectionSort(arr, n);

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

if (arr[i] % k == 0) {

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

}

}

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

12. Дан одномерный массив целых чисел. Выбрать из массива все отрицательные числа и упо- рядочить их по возрастанию.

#include <iostream>

using namespace std;

void insertionSort(int arr[], int n) {

for (int i = 1; i < n; i++) {

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key) {

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

j = j - 1;

}

arr[j+1] = key;

}

}

int main() {

int arr[] = {-2, 6, 3, -8, 10, -15};

int n = sizeof(arr)/sizeof(arr[0]);

// выбираем все отрицательные числа

int negArr[n], negCount = 0;

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

if (arr[i] < 0) {

negArr[negCount] = arr[i];

negCount++;

}

}

// сортируем отрицательные числа по возрастанию

insertionSort(negArr, negCount);

// выводим отсортированный массив

for (int i = 0; i < negCount; i++) {

cout << negArr[i] << " ";

}

cout << endl;

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

13. Дан одномерный массив целых чисел. Выбрать из массива все числа на нечетных позициях и упорядочить эти числа по убыванию.

void sort\_odd\_positions(int arr[], int n) {

for (int i = 1; i < n; i += 2) {

int key = arr[i];

int j = i - 2;

while (j >= 0 && arr[j] < key) {

arr[j + 2] = arr[j];

j -= 2;

}

arr[j + 2] = key;

}

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

14. Дан одномерный массив целых чисел. Выбрать из массива все двузначные числа и упорядо- чить эти числа по возрастанию.

#include <iostream>

using namespace std;

void selectionSort(int arr[], int n) {

for (int i = 0; i < n-1; i++) {

int min\_idx = i;

for (int j = i+1; j < n; j++) {

if (arr[j] >= 10 && arr[j] <= 99 && arr[j] < arr[min\_idx]) {

min\_idx = j;

}

}

if (min\_idx != i) {

swap(arr[i], arr[min\_idx]);

}

}

}

int main() {

int arr[] = {1, 25, 6, 85, 11, 76, 99, 16};

int n = sizeof(arr) / sizeof(arr[0]);

selectionSort(arr, n);

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

if (arr[i] >= 10 && arr[i] <= 99) {

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

}

}

cout << endl;

return 0;

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

15. Дан одномерный массив целых чисел. Выбрать из массива все числа на четных позициях и упорядочить эти числа по возрастанию.   
#include <iostream>

using namespace std;

void insertion\_sort(int arr[], int n) {

for (int i = 2; i < n; i += 2) {

int key = arr[i];

int j = i - 2;

while (j >= 0 && arr[j] > key) {

arr[j + 2] = arr[j];

j -= 2;

}

arr[j + 2] = key;

}

}

int main() {

int arr[] = {5, 9, 3, 8, 4, 7, 2, 6};

int n = sizeof(arr) / sizeof(arr[0]);

insertion\_sort(arr, n);

for (int i = 0; i < n; i += 2) {

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

}

return 0;

}