Беларускі дзяржаўны тэхналагічны ўніверсітэт

Факультэт інфармацыйных тэхналогій

Кафедра праграмнай інжэнерыі

Лабараторная работа №17

Па дысцыпліне «Асновы алгарытмізацыі і праграмавання»

На тэму « Указатели и ссылки при работе с функциями»

Выканала:

Студэнтка 1 курса 6 группы

Літвінчук Дар'я Валер'еўна

Выкладчык: асс. Андронава М.В.

2023, Мінск

**Варыянт 10**

|  |  |
| --- | --- |
| 10 | 1. Найти номера четных элементов, стоящих на нечетных местах в одномерном массиве.  2. Дана целочисленная прямоугольная матрица. Если максимальное число встречается в матрице более одного раза, то вывести его значение и индексы, а также количество повторений. |

#include <iostream>

#include <random>

#include <stdlib.h>

using namespace std;

// Функция для поиска и вывода позиций четных элементов на четных позициях

void findEvenElementsAtEvenPositions(int arr[10], int n) {

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

// Проверка, является ли элемент четным и находится ли на четной позиции

if (arr[i] % 2 == 0 && (i + 1) % 2 == 0)

cout << i + 1 << " "; // Вывод позиции элемента

}

}

int main() {

setlocale(LC\_ALL, "rus");

cout << "Введите массив: ";

int arr[10];

// Ввод элементов массива

for (int i = 0; i < 10; i++) {

cin >> arr[i];

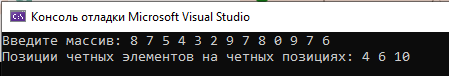
}

cout << "Позиции четных элементов на четных позициях: ";

findEvenElementsAtEvenPositions(arr, 10); // Вызов функции для вывода результатов

cout << endl; // Добавлен перевод строки для читаемости

}

  
  
#include <iostream>

using namespace std;

float ret[4]; // Глобальный массив для возвращаемых значений

// Функция для поиска максимального элемента в матрице и его позиции

float findmax(float\*\* arr, int n, int m)

{

int max = arr[0][0], maxi = 0, maxj = 0, count = 1;

// Обход всех элементов матрицы

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

{

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

{

// Поиск максимального элемента

if (arr[i][j] > max)

{

max = arr[i][j];

maxi = i;

maxj = j;

count = 1;

}

else if (arr[i][j] == max)

{

// Подсчет количества встреч максимального элемента

count++;

}

}

}

// Запись результатов в массив ret

ret[0] = max;

ret[1] = maxi;

ret[2] = maxj;

ret[3] = count;

return 0;

}

int main()

{

setlocale(LC\_ALL, "rus");

int n, m;

cout << "Введите размер матрицы: ";

cin >> n >> m;

float\*\* arr;

arr = new float\* [n];

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

arr[i] = new float[m]; // Создание матрицы

cout << "Введите матрицу: \n";

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

{

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

{

cin >> arr[i][j]; // Ввод матрицы

}

}

findmax(arr, n, m); // Вызов функции поиска максимального элемента

// Использование массива ret для вывода результатов

float max = ret[0], maxi = ret[1], maxj = ret[2], count = ret[3];

cout << "Максимальный элемент: " << max << "[" << maxi + 1 << "]" << "[" << maxj + 1 << "]" << "\nОн встречается " << count << " раз"; // Вывод

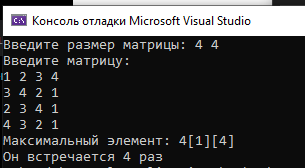
// Освобождение выделенной памяти для матрицы

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

delete[] arr[i];

delete[] arr;

}



**дадатковыя заданні:**

|  |  |
| --- | --- |
| 14 | 1. В одномерном массиве найти минимальный и максимальный элементы. Вычислить их разность.  2. Если в матрице имеется столбец, все элементы которого положительны, то знаки элементов предыдущего столбца изменить на противоположные. |

#include <iostream>

#include <limits>

using namespace std;

// Функция для поиска минимального, максимального элементов и их разности в массиве

void findMinMaxDifference(int\* arr, int size, int& minElement, int& maxElement, int& difference) {

// Проверка на корректный размер массива

if (size <= 0) {

cerr << "Invalid array size." << endl;

return;

}

// Инициализация переменных минимального и максимального элементов

minElement = numeric\_limits<int>::max(); // Инициализация максимальным возможным значением

maxElement = numeric\_limits<int>::min(); // Инициализация минимальным возможным значением

// Поиск минимального и максимального элементов

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

if (arr[i] < minElement) {

minElement = arr[i];

}

if (arr[i] > maxElement) {

maxElement = arr[i];

}

}

// Вычисление разности

difference = maxElement - minElement;

}

int main() {

setlocale(LC\_ALL, "Rus");

int size;

// Ввод размера массива

cout << "Введите размер массива: ";

cin >> size;

// Проверка на корректный размер массива

if (size <= 0) {

cerr << "Некорректный размер массива." << endl;

return 1;

}

// Динамическое выделение памяти под массив

int\* dynamicArray = new int[size];

// Ввод элементов массива

cout << "Введите элементы массива:" << endl;

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

cout << "Элемент " << i + 1 << ": ";

cin >> dynamicArray[i];

}

// Переменные для хранения результатов

int minElement, maxElement, difference;

// Вызов пользовательской функции

findMinMaxDifference(dynamicArray, size, minElement, maxElement, difference);

// Вывод результатов

cout << "Минимальный элемент: " << minElement << endl;

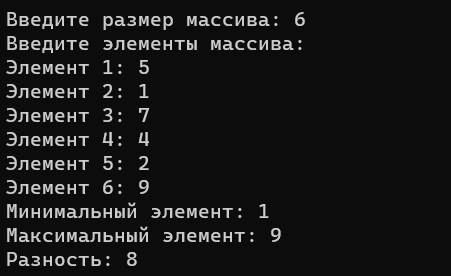
cout << "Максимальный элемент: " << maxElement << endl;

cout << "Разность: " << difference << endl;

// Освобождение динамической памяти

delete[] dynamicArray;

}



#include <iostream>

#include <iomanip> // Для настройки вывода матрицы

using namespace std;

// Функция для изменения знаков элементов предыдущего столбца, если в текущем столбце все элементы положительны

void processMatrix(int\*\* matrix, int rows, int cols) {

for (int col = 1; col < cols; ++col) {

bool allPositive = true;

// Проверка на положительные элементы в текущем столбце

for (int row = 0; row < rows; ++row) {

if (matrix[row][col] <= 0) {

allPositive = false;

break;

}

}

// Если все элементы положительны, меняем знаки предыдущего столбца

if (allPositive) {

for (int row = 0; row < rows; ++row) {

matrix[row][col - 1] \*= -1;

}

}

}

}

// Функция для вывода матрицы

void printMatrix(int\*\* matrix, int rows, int cols) {

for (int row = 0; row < rows; ++row) {

for (int col = 0; col < cols; ++col) {

cout << setw(4) << matrix[row][col];

}

cout << endl;

}

}

// Функция для освобождения динамической памяти, выделенной под матрицу

void deleteMatrix(int\*\* matrix, int rows) {

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

delete[] matrix[i];

}

delete[] matrix;

}

int main() {

setlocale(LC\_ALL, "Rus");

int rows, cols;

// Ввод размеров матрицы

cout << "Введите количество строк матрицы: ";

cin >> rows;

cout << "Введите количество столбцов матрицы: ";

cin >> cols;

if (rows <= 0 || cols <= 0) {

cerr << "Некорректные размеры матрицы." << endl;

return 1;

}

// Динамическое выделение памяти под матрицу

int\*\* dynamicMatrix = new int\* [rows];

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

dynamicMatrix[i] = new int[cols];

}

// Ввод элементов матрицы

cout << "Введите элементы матрицы:" << endl;

for (int row = 0; row < rows; ++row) {

for (int col = 0; col < cols; ++col) {

cout << "Элемент [" << row + 1 << "][" << col + 1 << "]: ";

cin >> dynamicMatrix[row][col];

}

}

// Вызов пользовательской функции для изменения знаков элементов

processMatrix(dynamicMatrix, rows, cols);

// Вывод измененной матрицы

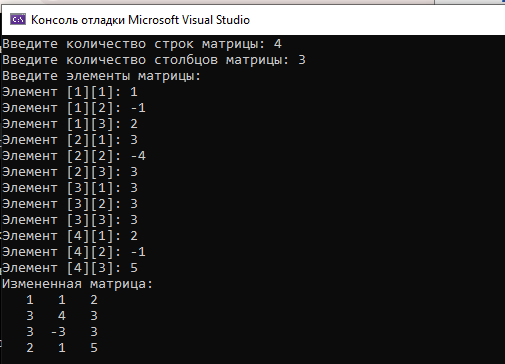
cout << "Измененная матрица:" << endl;

printMatrix(dynamicMatrix, rows, cols);

// Освобождение динамической памяти

deleteMatrix(dynamicMatrix, rows);

}



|  |  |
| --- | --- |
| 11 | 1. Изменить одномерный массив, вычеркнув из него нечетные элементы.  2. Дана целочисленная квадратная матрица. Если она не содержат отрицательных элементов, то определить сумму элементов в тех строках, где отрицательные элементы отсутствуют. |

#include <iostream>

using namespace std;

// Функция для поиска и вывода четных элементов массива

void findEvenElements(int\* arr, int n) {

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

// Проверка, является ли элемент четным

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

cout << arr[i] << " "; // Вывод четного элемента

}

}

int main() {

setlocale(LC\_ALL, "rus");

int n;

cout << "Введите длину массива: ";

cin >> n;

int\* arr;

arr = new int[n]; // Выделение памяти под массив

cout << "Введите массив: ";

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

// Ввод элементов массива

cin >> arr[i];

}

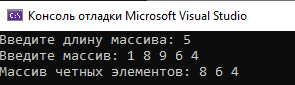
cout << "Массив четных элементов: ";

findEvenElements(arr, n); // Вызов функции для вывода четных элементов

cout << endl; // Добавлен перевод строки для читаемости

delete[] arr; // Освобождение выделенной памяти для массива

}

  
  
  
#include <iostream>

using namespace std;

// Функция для проверки наличия отрицательных элементов в столбцах матрицы

bool containsNegativeInColumns(float\*\* arr, int n, int m)

{

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

{

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

{

if (arr[i][j] < 0)

{

return false; // Найден отрицательный элемент в столбце, возвращаем false

}

}

}

return true; // Все столбцы не содержат отрицательных элементов, возвращаем true

}

int main()

{

setlocale(LC\_ALL, "rus");

int n, m;

cout << "Введите размер матрицы: ";

cin >> n >> m;

float\*\* arr;

arr = new float\* [n];

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

arr[i] = new float[m]; // Выделение памяти под матрицу

cout << "Введите матрицу: \n";

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

{

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

{

cin >> arr[i][j]; // Ввод матрицы

}

}

bool noNegativeColumns = containsNegativeInColumns(arr, n, m);

if (noNegativeColumns)

{

int sum = 0;

cout << "Матрица не содержит отрицательные элементы";

// Подсчет суммы элементов в первых трех строках

for (int i = 0; i < 3; i++)

{

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

sum += arr[i][j];

}

cout << "\nСумма: " << sum; // Вывод суммы

}

else

{

cout << "Матрица содержит отрицательные элементы";

}

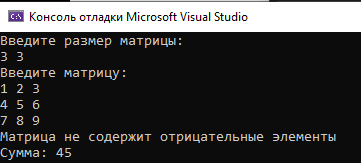
// Освобождение выделенной памяти для матрицы

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

delete[] arr[i];

delete[] arr;

}



|  |  |
| --- | --- |
| 12 | 1. Найти количество отрицательных элементов, стоящих на чётных местах в одномерном массиве.  2. Проверить, есть ли в матрице хотя бы одна строка, содержащая отрицательный элемент, и найти ее номер. Все элементы столбца с таким же номером уменьшить вдвое. |

#include <iostream>

#include <random>

#include <stdlib.h>

using namespace std;

// Функция для подсчета количества нечетных элементов на четных позициях

int countOddAtEvenPositions(int\* arr, int n) {

int count = 0;

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

if (arr[i] % 2 == 1 && (i + 1) % 2 == 0)

count++; // Увеличиваем счетчик для каждого нечетного элемента на четной позиции

}

return count;

}

int main() {

int n;

setlocale(LC\_ALL, "rus");

cout << "Введите длину массива: ";

cin >> n;

int\* arr;

arr = new int[n]; // Выделение памяти под массив

cout << "Введите массив: ";

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

// Ввод элементов массива

cin >> arr[i];

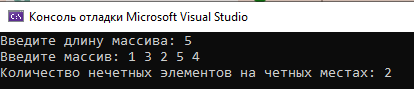
}

int oddCount = countOddAtEvenPositions(arr, n); // Вызов функции для подсчета

cout << "Количество нечетных элементов на четных местах: " << oddCount << endl; // Вывод результата

delete[] arr; // Освобождение выделенной памяти для массива

}

  
  
  
#include <iostream>

using namespace std;

// Функция для поиска номера столбца с отрицательным элементом

int findColumnWithNegative(float\*\* arr, int n, int m)

{

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

{

bool hasNegative = false;

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

{

if (arr[i][j] < 0)

{

hasNegative = true; // Поиск столбца с отрицательными элементами

break;

}

}

if (hasNegative)

{

return j; // Возвращаем номер столбца

}

}

}

int main()

{

setlocale(LC\_ALL, "rus");

int n, m;

cout << "Введите размер матрицы: ";

cin >> n >> m;

float\*\* arr;

arr = new float\* [n];

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

arr[i] = new float[m]; // Создание матрицы

cout << "Введите матрицу: \n";

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

{

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

{

cin >> arr[i][j]; // Ввод матрицы

}

}

int colnum = findColumnWithNegative(arr, n, m); // Номер столбца с отрицательным элементом

cout << "Номер строки с отрицательным элементом: " << colnum + 1 << endl;

// Деление на 2 элементов столбца

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

arr[i][colnum] = arr[i][colnum] / 2;

cout << "Новая матрица:\n";

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

{

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

{

cout << arr[i][j] << " "; // Вывод новой матрицы

}

cout << endl;

}

// Освобождение выделенной памяти для матрицы

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

delete[] arr[i];

delete[] arr;

}  
  
