Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

Лабораторная работа 17

По дисциплине «Основы алгоритмизации и программирования»

На тему «Указатели и ссылки при работе с функциями»

Выполнил:

Студент 1 курса 9 группы

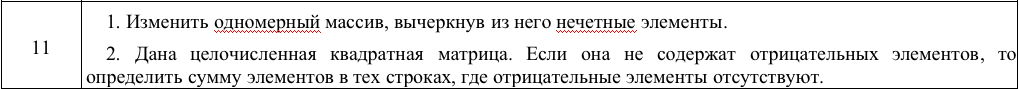
Павлович Ян Андреевич

Преподаватель: асс. Андронова М.В.

2023, Минск

Вариант 11

1. В соответствии со своим вариантом написать главную функцию, в которой имеются вызовы ***функций пользователя***, реализующих задачи из таблицы ниже. Ввод исходных данных и вывод результатов осуществить в главной функции, при этом использовать***динамические* *массивы***. Для передачи параметров в функции пользователя и возвращения результатов применить ***указатели*** и ***ссылки***.



1. Изменить одномерный массив, вычеркнув из него нечетные элементы.

#include <iostream> //для работы с вводом/выводом

**using** **namespace** std; //объявляем программе, что использем пространство имен std

//функция для удаления нечетных элементов в одномерном массиве

**void** delNum(**int**\* mass, **int** a, **int** i) {

**for** (**int** k = i; k < a; k++) {

        \*(mass + k) = \*(mass + k + 1);

    }

}

**int** main() {

    setlocale(LC\_ALL, "rus"); //добавляем возможность работы с руусским языком

**int** a;

**const** **int** N = 20; //задаем максимальное значение для массива

**int**\* mass = **new** **int**[N]; //выделяем память для массива

    cout << "Размер массива: ";

    cin >> a;

    cout << "Введите элементы одномерного массива: ";

    //с помощью цикла вводим данные в массив

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

        cin >> mass[i];

    }

    //с помошью цикла проверяем элементы массива на четность

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

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

            delNum(mass, a, i); //вызываем функцию для удаления нечетного элемента

            i--;

            a--;

        }

    }

    cout << "Измененный массив: ";

    //с помощью цикла выводим измененный массив

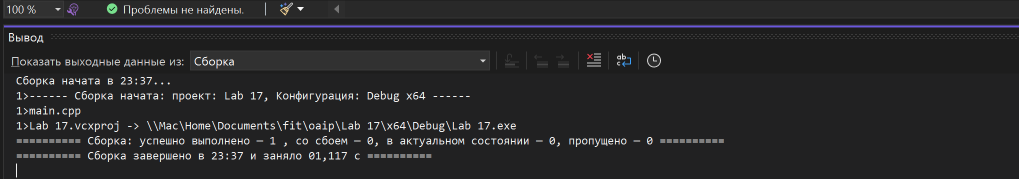
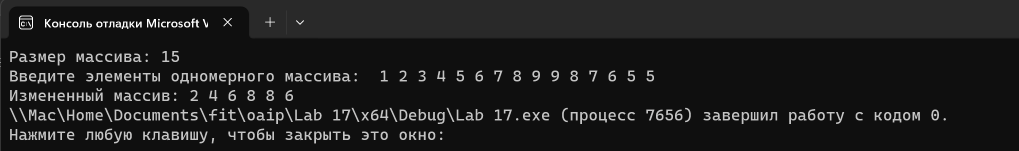
**for** (**int** i = 0; i < a; i++) {

        cout << mass[i] << " ";

}

**delete**[] mass; //очистка динамической памяти

}



2. Дана целочисленная квадратная матрица. Если она не содержат отрицательных элементов, то определить сумму элементов в тех строках, где отрицательные элементы отсутствуют.

#include <iostream> //для работы с вводом/выводом

**using** **namespace** std; //объявляем программе, что работаем с пространством имен std

// Используем ссылку на bool, чтобы изменять значение внутри функции

**void** check(**bool**& hasNegative, **int** n, **int** i, **int**\*\* matrix) {

    // Проверка наличия отрицательных элементов в строке

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

**if** (matrix[i][j] < 0) {

            hasNegative = **true**;

**break**;

        }

    }

}

**int** main() {

    setlocale(LC\_ALL, "rus"); // добавляем возможность работы с русским языком

**int** n;

    cout << "Введите размерность квадратной матрицы: ";

    cin >> n;

    // Динамическое выделение памяти для матрицы

**int**\*\* matrix = **new** **int**\* [n];

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

        matrix[i] = **new** **int**[n];

    }

    // Ввод элементов матрицы

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

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

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

            cin >> matrix[i][j];

        }

    }

    // Поиск строк без отрицательных элементов и вычисление их суммы

**int** sum = 0;

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

**bool** hasNegative = **false**;

        check(hasNegative, n, i, matrix);

        // Если отрицательных элементов нет, добавляем сумму элементов строки к общей сумме

**if** (!hasNegative) {

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

                sum += matrix[i][j];

            }

        }

    }

    // Вывод результата

    cout << "Сумма элементов строк без отрицательных элементов: " << sum << endl;

    // Освобождение выделенной памяти

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

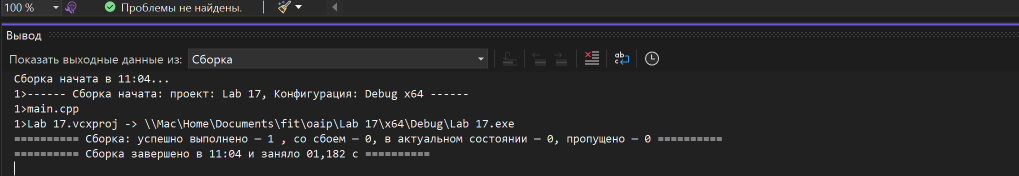
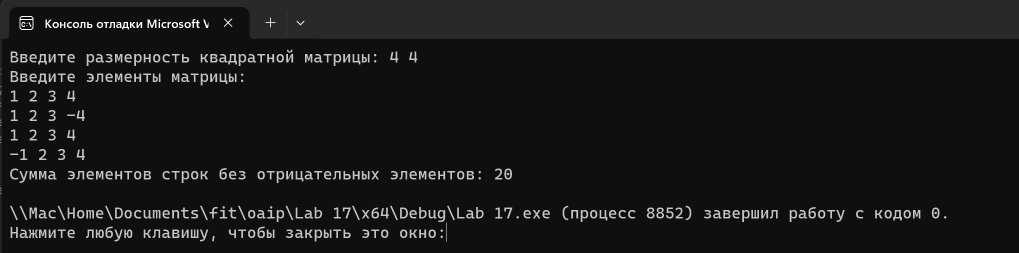
**delete**[] matrix[i];

    }

**delete**[] matrix;

**return** 0;

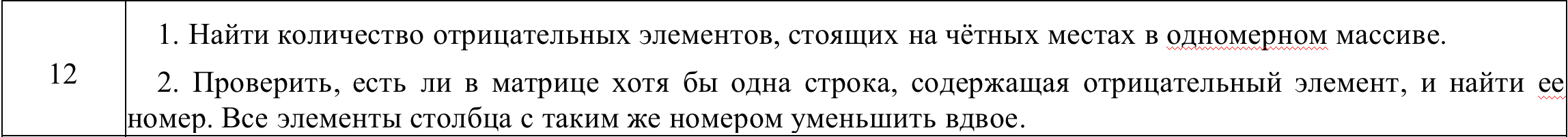
}



**Дополнительные задания**

Вариант 12

5. В соответствии со своим вариантом написать главную функцию, в которой имеются вызовы ***функций пользователя***, реализующих задачи из таблицы ниже. Ввод исходных данных и вывод результатов осуществить в главной функции, при этом использовать***динамические* *массивы***. Для передачи параметров в функции пользователя и возвращения результатов применить ***указатели*** и ***ссылки***.



1. Найти количество отрицательных элементов, стоящих на чётных местах в одномерном массиве.

#include <iostream> //для работы с вводом/выводом

**using** **namespace** std; //объявляем программе, что использем пространство имен std

//функция для реализации условия

**void** MasCheck(**int**\* mas, **int** size) {

**int** negative\_odd\_count = 0; //создание переменной для подсчета кол-ва элементов

    //цикл для нахождеия таких элементов

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

**if** (\*(mas + i) < 0) {

**if** (i % 2 != 0) {

                negative\_odd\_count++; //увеличение счетной переменной на 1

            }

        }

    }

    cout << "Кол-во отрицательных элементов,стоящих на чётных местах в одномерном массиве:" << negative\_odd\_count << endl; //вывод кол-ва элеентов

}

**void** main() {

    setlocale(LC\_ALL, "Russian");// добавляем возможность работы с русским языком

**int** size;//ввод размерности массива

    cout << "Введите размерность одномерного массива: ";

    cin >> size;

**int**\* mass;//создание указателя на оьласть памяти с массивом

    mass = **new** **int**[size];//выделение динамиечкой памяти

    cout << "Введите элементы одномерного массива: ";

**for** (**int** i = 0; i < size; i++) {//цикл для ввода элементов в массив

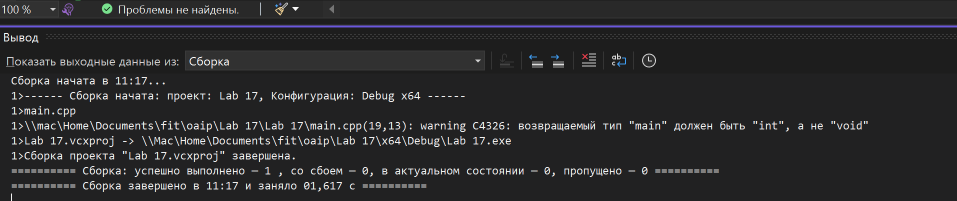
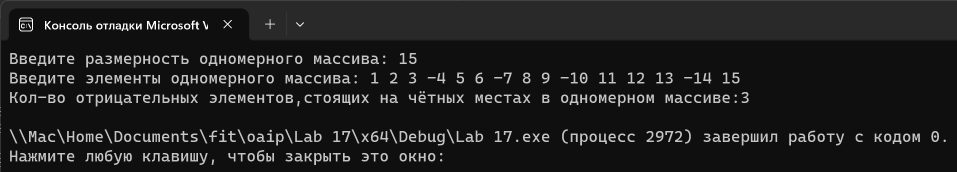
        cin >> mass[i];

    }

    MasCheck(mass, size);

**delete**[]mass;//очистка динамической памяти

}



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> //для работы с вводом/выводом

**using** **namespace** std; //объявляем программе, что использем пространство имен std

**void** MatrixCheck(**int**\*\* matrix, **int** amm\_of\_colomns, **int** amm\_of\_rows)//функиция для изменения матрицы

{

**int** number\_of\_row[100];//создание массива для включения в него номеров строк с отрицательными элемнетами

**int** n = 0;//создане счётной переменной для вычисления размера массива

**for** (**int** i = 0; i < amm\_of\_rows; i++) {//цикл в цикле для проверки каждого элемента матрицы

**for** (**int** j = 0; j < amm\_of\_colomns; j++) {

**if** (matrix[i][j] < 0) {//если элемент отрицательный

                number\_of\_row[n] = i;//добавление номера строки в массив

                n++;//увеличение счётной переменной на 1

            }

        }

    }

    //цикл для вывода номеров строк

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

        cout << "Отрицтельный элемент в строке под номером: " << number\_of\_row[i] + 1 << endl;

    }

    //цикл для уменьшения соответствующих столбцов вдвое

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

**for** (**int** j = 0; j < amm\_of\_rows; j++) {

            matrix[j][number\_of\_row[i]] = matrix[j][number\_of\_row[i]] / 2;

        }

    }

    //вывод результата измененой матрицы

    cout << "Получившаяся матрица:" << endl;

**for** (**int** i = 0; i < amm\_of\_rows; i++) {

**for** (**int** j = 0; j < amm\_of\_colomns; j++) {

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

        }

        cout << "\n";

    }

}

**void** main() {

    setlocale(LC\_ALL, "Russian"); //добавляем возможность работы с русским языком

**int** amm\_of\_rows;

    cout << "Введите количество строк: ";

    cin >> amm\_of\_rows;

    cin.ignore(cin.rdbuf()->in\_avail()); //очистка буфера после ввода

**int** amm\_of\_colomns;//ввод кол-ва столбцов матрицы

    cout << "Введите количество столбцов: ";

    cin >> amm\_of\_colomns;

    cin.ignore(cin.rdbuf()->in\_avail());//очистка буфера после ввода

**int**\*\* matrix;//создание указателя на область памяти с массивом

    matrix = **new** **int**\* [amm\_of\_rows];//выделение динамической памяти под массив

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

**for** (**int** i = 0; i < amm\_of\_rows; i++) {//цикл в цикле для ввода элементов матрицы

        matrix[i] = **new** **int**[amm\_of\_colomns];//выделение динамической памяти под строку

**for** (**int** j = 0; j < amm\_of\_colomns; j++) {

            cin >> matrix[i][j];//ввод элемента матрицы

        }

    }

    cin.ignore(cin.rdbuf()->in\_avail());//очистка буфера после ввода

    MatrixCheck(matrix, amm\_of\_colomns, amm\_of\_rows);

    //цикл для очистки динамиеской памяти

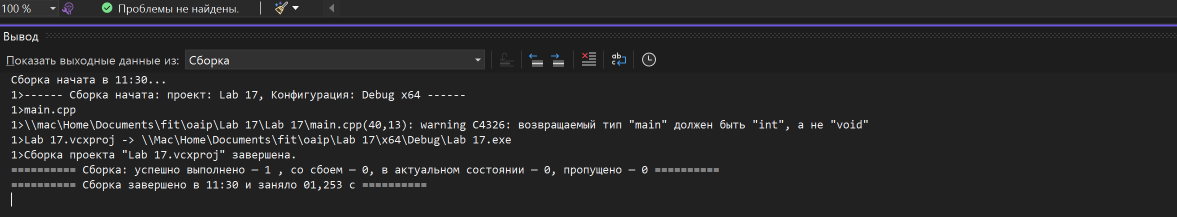
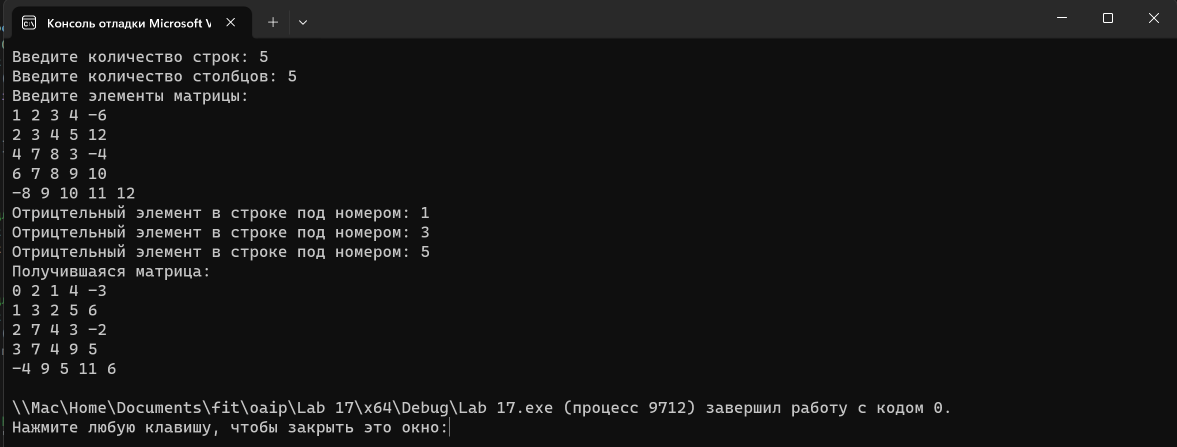
**for** (**int** i = 0; i < amm\_of\_rows; i++) {

**delete**[]matrix[i];

    }

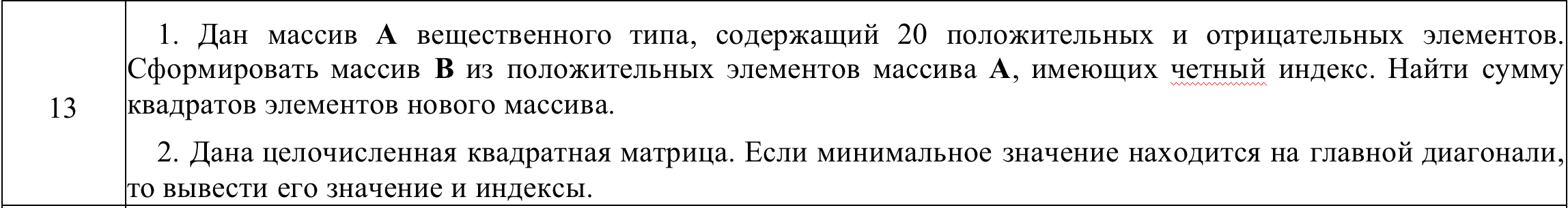
**delete**[]matrix;//очистка динамической памяти под массив

}



Вариант 13

5. В соответствии со своим вариантом написать главную функцию, в которой имеются вызовы ***функций пользователя***, реализующих задачи из таблицы ниже. Ввод исходных данных и вывод результатов осуществить в главной функции, при этом использовать***динамические* *массивы***. Для передачи параметров в функции пользователя и возвращения результатов применить ***указатели*** и ***ссылки***.



1. Дан массив **A** вещественного типа, содержащий 20 положительных и отрицательных элементов. Сформировать массив **B** из положительных элементов массива **A**, имеющих четный индекс. Найти сумму квадратов элементов нового массива.

#include <iostream> //для работы с вводом/выводом

**using** **namespace** std; //объявляем программе, что использем пространство имен std

//функция для построения массива В

**void** BuildMasB(**double**\* A, **int** size\_of\_A) {

**int** amm\_of\_odd\_positive = 0; //создание переменной для определения размерности массива В

**double**\* B; //создание указателя на область памяти с массивом В

    B = **new** **double**[size\_of\_A]; //выделение динамичекой памяти под массив В

    //цикл для проверки элементов массива А на соответствие условию

**for** (**int** i = 0; i < size\_of\_A; i++) {

**if** (i % 2 != 0) {//если элемент четный по номеру

**if** (\*(A + i) > 0) { //если элемент положительный

                \*(B + amm\_of\_odd\_positive) = \*(A + i); //присвоить элемент массиву В

                amm\_of\_odd\_positive++; //увеличение счётной переменной на 1

            }

        }

    }

    //результат

    cout << "Массив В: ";

**for** (**int** i = 0; i < amm\_of\_odd\_positive; i++) {

        cout << \*(B + i) << ' ';

    }

}

**void** main() {

    setlocale(LC\_ALL, "Russian"); //добавляем возможность работы с русским языком

**double**\* A; //создание указателя на область памяти с массивом А

**int** size\_of\_A = 20; //задание размерности массива

    A = **new** **double**[size\_of\_A]; //выделение динамической памяти под массив А

    cout << "Введите 20 элементов массива А: ";

    //цикл для ввода элементов массива

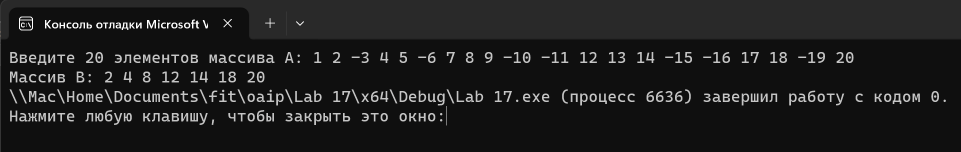
**for** (**int** i = 0; i < size\_of\_A; i++) {

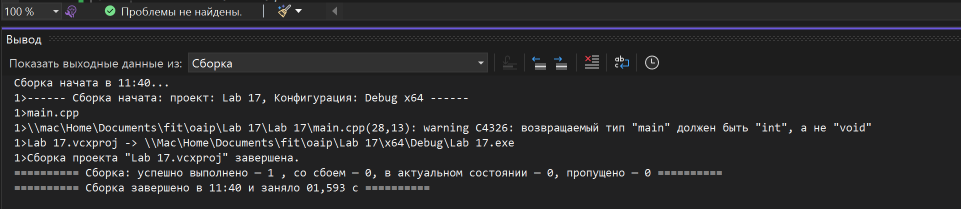
        cin >> \*(A + i);

    }

    BuildMasB(A, size\_of\_A);

}





2. Дана целочисленная квадратная матрица. Если минимальное значение находится на главной диагонали, то вывести его значение и индексы.

#include <iostream> //для работы с вводом/выводом

**using** **namespace** std; //объявляем программе, что использем пространство имен std

//функция для проверки элементов матрицы

**void** MatrixCheck(**int**\*\* matrix, **int** size) {

**int** min = matrix[0][0];

    //цикл для проверки каждого элемента и вычисления минимального

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

**for** (**int** j = 0; j < size; j++) {

**if** (matrix[i][j] < min) {

                min = matrix[i][j];

            }

        }

    }

    cout << "Минимальный элемент матрицы: " << min << endl; //вывод минимального элемента

**for** (**int** i = 0; i < size; i++) { //цикл для проверки главной диагонали матрицы

**for** (**int** j = 0; j < size; j++) {

**if** (i == j) { //если элемент на главной диагонали

**if** (matrix[i][j] == min) { //есди элемент равен минимальному

                    cout << "Значение " << matrix[i][j] << " находится на позиции (" << i + 1 << ";" << j + 1 << ")" << endl;

                }

            }

        }

    }

}

**void** main() {

    setlocale(LC\_ALL, "Russian"); //добавляем возможность работы с русским языком

**int** size;

    cout << "Введите размер квадратной матрицы: ";

    cin >> size;

**int**\*\* matrix; //создание указателя на область памяти с матрицей

    matrix = **new** **int**\* [size]; //выделение динамической памяти под матрицу

    //ввод элементов в матрицу

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

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

        matrix[i] = **new** **int**[size]; //выделение динамической памяти под строку

**for** (**int** j = 0; j < size; j++) {

            cin >> matrix[i][j];

        }

    }

    MatrixCheck(matrix, size);

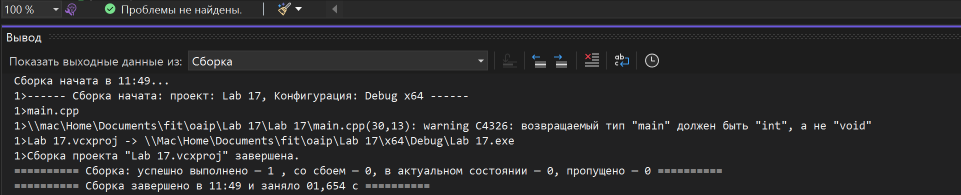
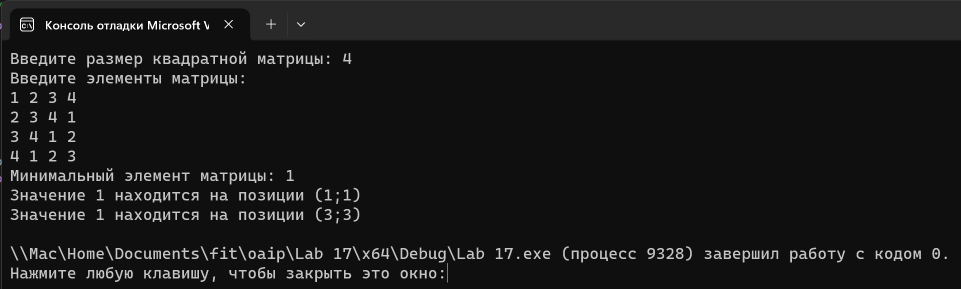
**for** (**int** i = 0; i < size; i++) { //цикл для очистки динамической памяти

**delete**[]matrix[i];

}

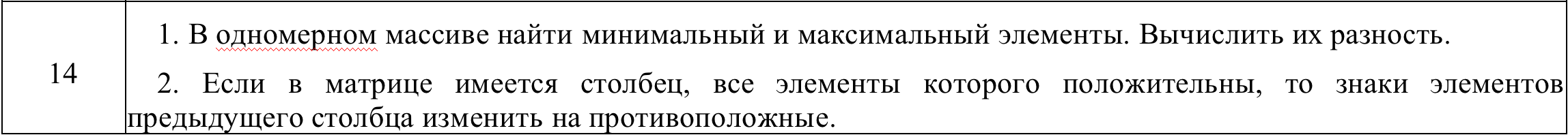
**delete**[]matrix;//очистка динамической памяти под матрицу

}



Вариант 14

5. В соответствии со своим вариантом написать главную функцию, в которой имеются вызовы ***функций пользователя***, реализующих задачи из таблицы ниже. Ввод исходных данных и вывод результатов осуществить в главной функции, при этом использовать***динамические* *массивы***. Для передачи параметров в функции пользователя и возвращения результатов применить ***указатели*** и ***ссылки***.



1. В одномерном массиве найти минимальный и максимальный элементы. Вычислить их разность.

#include <iostream> //для работы с вводом/выводом

**using** **namespace** std; //объявляем программе, что использем пространство имен std

//функиция, возвращаюзая максимальный элемент матрицы

**int** MaxElem(**int**\* mas, **int** size) {

**int**\* max = mas; //создание указателя на массив

    //цикл для вычисления максимльног элемента

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

**if** (\*(mas + i) > \*max) {

            max = (mas + i); //вычисление максимального элемента

        }

    }

**return** \*max; //возращает максимальный элемент

}

//функция, возвращающая минимальный элемент

**int** MinElem(**int**\* mas, **int** size) {

**int**\* min = mas; //создание указателя на массив

    //цикл для нахождения минимальног элемента

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

**if** (\*(mas + i) < \*min) {

            min = (mas + i); //высисление минимального элемента

        }

    }

**return** \*min; //возрващает минимальный элемент

}

//функция для вывода максимального и миимального элементов и операций над ними

**void** Calc(**int**\* mas, **int** size) {

    cout << "Максимальный элемент массива: " << MaxElem(mas, size) << endl; //вывод максимального

    cout << "Минимальный элемент массива: " << MinElem(mas, size) << endl; //вывод минимального

    cout << "Разница между максимальным и минимаьным элементами: " << MaxElem(mas, size) - MinElem(mas, size) << endl; //вывод разности

}

**void** main() {

    setlocale(LC\_ALL, "Russian"); //добавляем возможность работы с русским языком

**int** size; //ввод размерности массива

    cout << "Введите размерность массива: ";

    cin >> size;

**int**\* mas; //создание указателя на массив

    mas = **new** **int**[size]; //выделение динамической памяти под массив

    cout << "Введите элементы массива: ";

    //цикл для ввода элементов массива

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

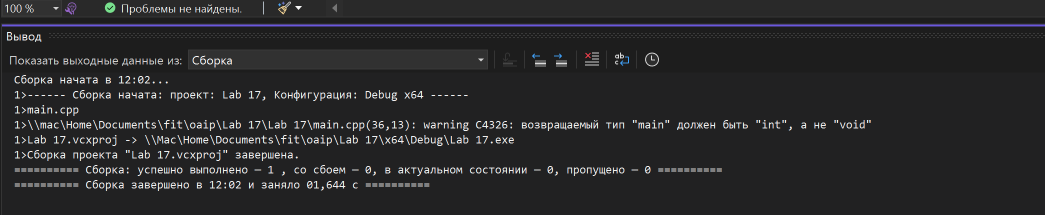
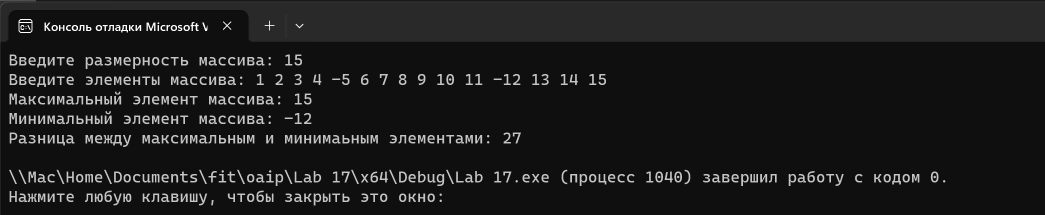
        cin >> mas[i];

    }

    Calc(mas, size); //использование функции для работы с элементами

**delete**[]mas; //очистка динамической памяти

}



2. Если в матрице имеется столбец, все элементы которого положительны, то знаки элементов предыдущего столбца изменить на противоположные.

#include <iostream> //для работы с вводом/выводом

**using** **namespace** std; //объявляем программе, что использем пространство имен std

//функция для изменения матрицы

**void** ColomnCheck(**int**\*\* matrix, **int** amm\_of\_rows, **int** amm\_of\_colomns) {

**int**\* num\_of\_colomns; //создание указателя на область памяти

    num\_of\_colomns = **new** **int**[amm\_of\_rows]; //выделение динамической памяти под массив , содержащий номера положительных столбцов

**int** amm = 0; //создание переменной для подсчета кол-ва положительных столбцов

    //цикл для вычисления таких столбцов

**for** (**int** i = 0; i < amm\_of\_colomns; i++) {

**int** control\_var = 0;

**for** (**int** j = 0; j < amm\_of\_rows; j++) {

**if** (matrix[j][i] > 0) {

                control\_var++;

            }

        }

**if** (control\_var == amm\_of\_rows) {

            \*(num\_of\_colomns + amm) = i;

            amm++;

        }

    }

    cout << "Номера положительных столбцов: ";

    //цикл для вывода номеров положительных столбцов

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

        cout << \*(num\_of\_colomns + i) + 1 << ' ';

    }

    //цикл для изменения заданых по условию столбцов

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

**for** (**int** j = 0; j < amm\_of\_rows; j++) {

            matrix[j][\*(num\_of\_colomns + i) - 1] = -matrix[j][\*(num\_of\_colomns + i) - 1];

        }

    }

    cout << "\nМатрица после изменения:" << endl;

**for** (**int** i = 0; i < amm\_of\_rows; i++) {//цикл в цикле для вывода получившейся матрицы

**for** (**int** j = 0; j < amm\_of\_colomns; j++) {

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

        }

        cout << "\n";

    }

**delete**[] num\_of\_colomns; //очистка динамической памяти

}

**void** main() {

    setlocale(LC\_ALL, "Russian"); //добавляем возможность работы с русским языком

**int** amm\_of\_rows;

    cout << "Введите кол-во строк: ";

    cin >> amm\_of\_rows;

    cin.ignore(cin.rdbuf()->in\_avail()); //очистка буффера после ввода

**int** amm\_of\_colomns;

    cout << "Введите кол-во столбцов: ";

    cin >> amm\_of\_colomns;

    cin.ignore(cin.rdbuf()->in\_avail()); //очистка буффера после ввода

**int**\*\* matrix; //создание указателя на область памяти для массива

    matrix = **new** **int**\* [amm\_of\_rows]; //выделеие динамеческой памяти под матрицу

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

    //цикл для ввода элементов матрицы

**for** (**int** i = 0; i < amm\_of\_rows; i++) {

        matrix[i] = **new** **int**[amm\_of\_colomns];//создание динамической памяти под строку

**for** (**int** j = 0; j < amm\_of\_colomns; j++) {

            cin >> matrix[i][j];//ввод элемента

        }

    }

    cin.ignore(cin.rdbuf()->in\_avail());//очистка буфера после ввода

    ColomnCheck(matrix, amm\_of\_rows, amm\_of\_colomns);

    //цикл для очистки динамической памяти

**for** (**int** i = 0; i < amm\_of\_colomns; i++) {

**delete**[]matrix[i];

    }

**delete**[]matrix;//очистка динамической памяти под матрицу

}

