ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА» (РУТ (МИИТ))

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

ОТЧЁТ О ПРАКТИЧЕСКОЙ РАБОТЕ №4

По дисциплине «Введение в языки программирования»

Выполнил: ст. гр. ТКИ – 111

Насонова А.Н.

Проверил: к.т.н., доц.

Васильева М.А.

Москва 2021

**Оглавление**

[**Задание 4-1** 3](#_Toc93926027)

[**Формулировка задания** 3](#_Toc93926028)

[**Блок-схема алгоритма** 4](#_Toc93926029)

[**Программа на языке C++** 8](#_Toc93926030)

[**Решение тестового примера на C++** 11](#_Toc93926031)

[**Задание 4-2** 12](#_Toc93926032)

[**Формулировка задания** 12](#_Toc93926033)

[**Блок-схема алгоритма** 13](#_Toc93926034)

[**Программа на языке C++** 17](#_Toc93926035)

[**Решение тестового примера на C++** 21](#_Toc93926036)

[**Задание 4-3** 22](#_Toc93926037)

[**Формулировка задания** 22](#_Toc93926038)

[**Блок-схема алгоритма** 23](#_Toc93926039)

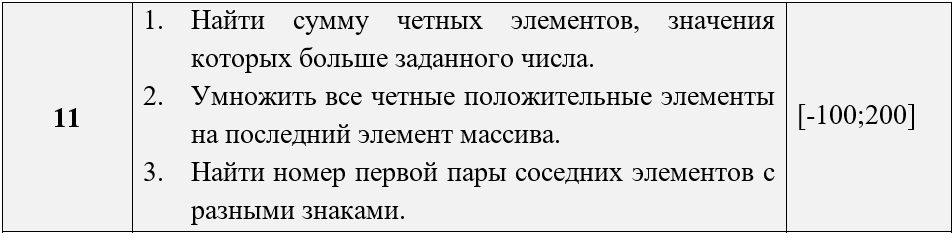
[**Программа на языке C++** 29](#_Toc93926040)

[**Решение тестового примера на C++** 33](#_Toc93926041)

# **Задание 4-1**

# **Формулировка задания**

Протабулировать заданную в таблице функцию. Использовать данные в таблице значения шага и интервала в качестве ввода пользователя для решения тестового примера. При невозможности расчёта функции в конкретной точке выводить её значение и надпись, означающую отсутствие решения.



# **Блок-схема алгоритма**

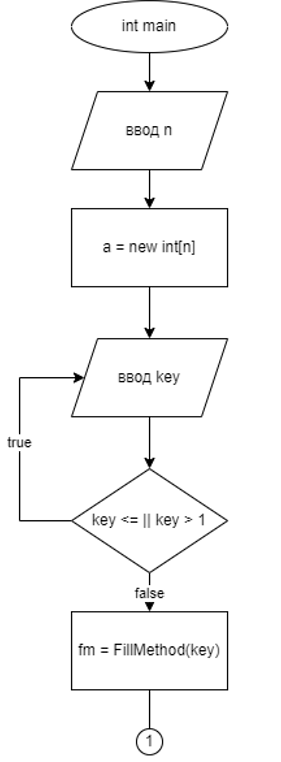
****

Рисунок 1 – Блок-схема алгоритма

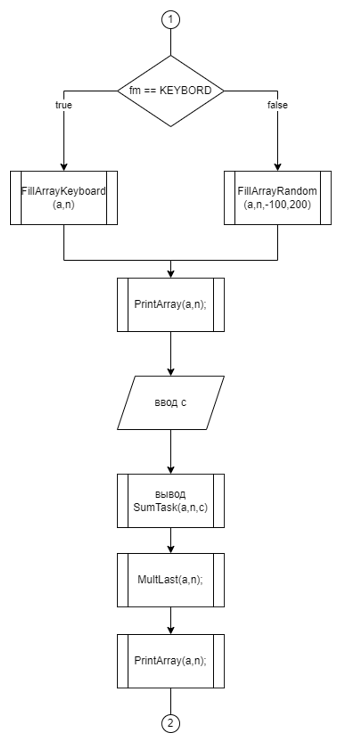


Рисунок 2 – Блок-схема алгоритма

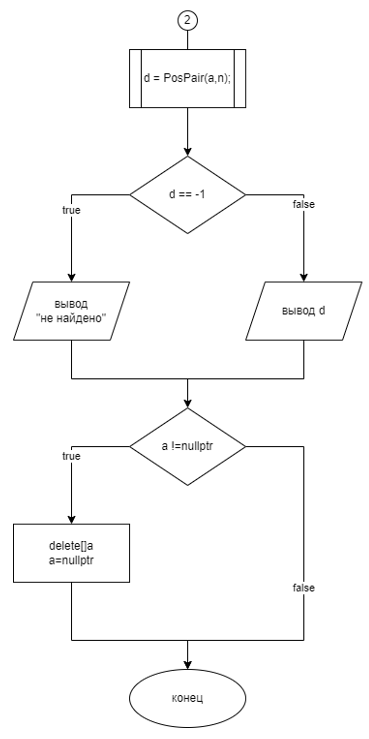


Рисунок 3 – Блок-схема алгоритма

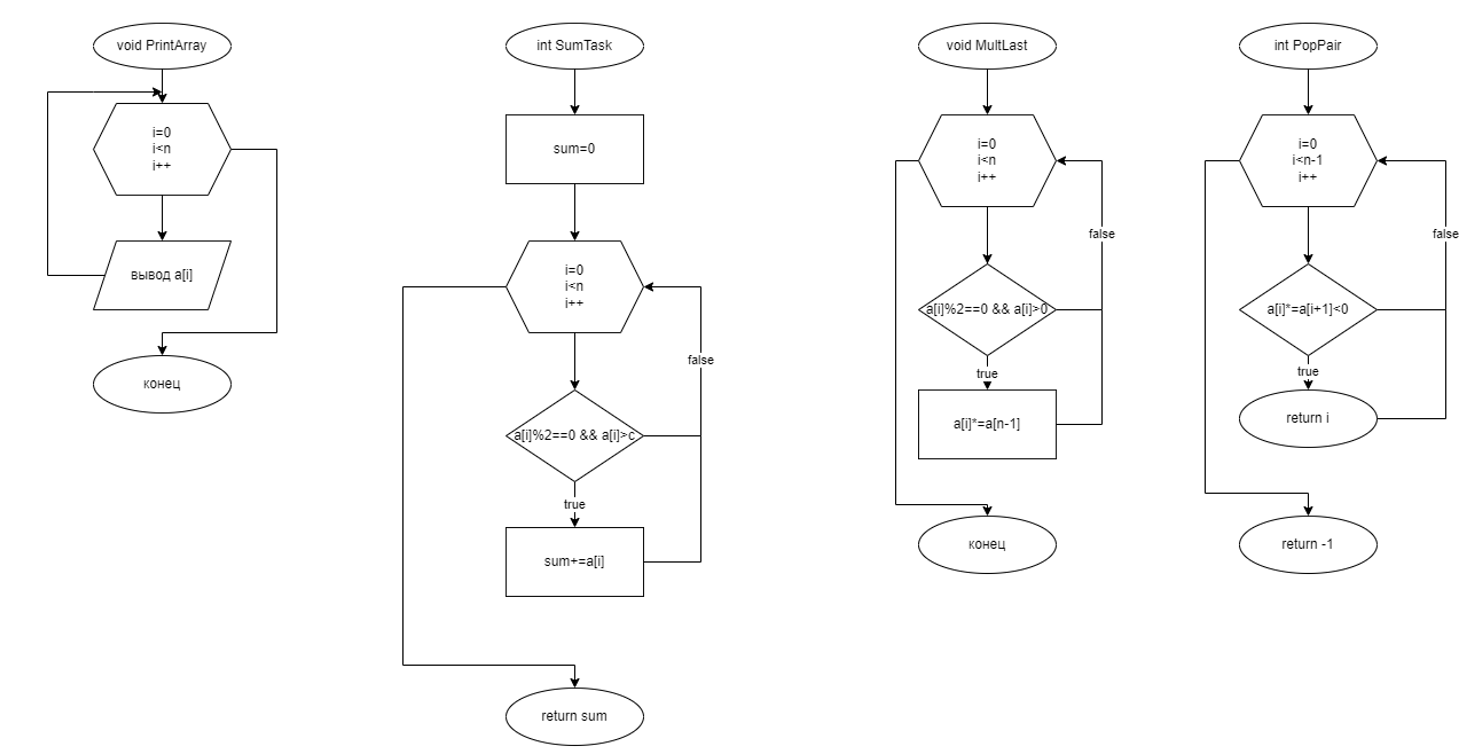


Рисунок 4 – Блок-схема алгоритма

# **Программа на языке C++**

#include <iostream>

#include <random>

using namespace std;

/\*\*

\* \brief Ввод элементов массива с клавиатуры.

\* \param arr - Указатель на массив.

\* \param n - Размер массива.

\*/

void FillArrayKeyboard(int\* arr, const size\_t n);

/\*\*

\* \brief Заполнение элементов массива случайными числами.

\* \param arr - Указатель на массив.

\* \param n - Размер массива.

\* \param min - Левая граница диапазона случайных чисел.

\* \param max - Правая граница диапазона случайных чисел.

\*/

void FillArrayRandom(int\* arr, const size\_t n, const int min, const int max);

/\*\*

\* \brief Вывод элементов массива в консоль.

\* \param arr - Указатель на массив.

\* \param n - Размер массива.

\*/

void PrintArray(const int\* arr, const size\_t n);

/\*\*

\* \brief Подсчет суммы четных элементов массива, больших заданного числа.

\* \param arr - Указатель на массив.

\* \param n - Размер массива.

\* \param c - Заданное число.

\* \return Искомая сумма.

\*/

int SumTask(const int\* arr, const size\_t n, const int c);

/\*\*

\* \brief Перемножение четных положительных элементов массива на значение последнего элемента.

\* \param arr - Указатель на массив.

\* \param n - Размер массива.

\*/

void MultLast(int\* arr, const size\_t n);

/\*\*

\* \brief Нахождение индекса первой пары элементов массива противоположных знаков.

\* \param arr - Указатель на массив.

\* \param n - Размер массива.

\* \return Индекс первого элемента искомой пары.

\*/

int PosPair(const int\* arr, const size\_t n);

/\*\*

\* \brief Перечислимый тип методов заполнения.

\*/

enum class FillMethod { //перечислимый тип

/\*\*

\* \brief Заполнение элементов массива с клавиатуры.

\*/

KEYBORD,

/\*\*

\* \brief Заполнение элементов массива случайными числами.

\*/

RANDOM

};

int main()

{

setlocale(LC\_ALL, "rus");

size\_t n, key;

int size;

cout << "Введите размер массива: ";

cin >> size;

while (size < 1) {

cout << "Размер должен быть положительным! Введите размер: ";

cin >> size;

};

n = size;

int\* arr = new int[n];

do {

cout << "Заполнить массив:\n -" << static\_cast<int>(FillMethod::KEYBORD) << ". С клавиатуры\n -" << static\_cast<int>(FillMethod::RANDOM) << ". Случайными числами\n\nВведите опцию: ";

cin >> key;

} while (key != static\_cast<int>(FillMethod::KEYBORD) && key != static\_cast<int>(FillMethod::RANDOM));

const int left = -100;

const int right = 200;

if (key == static\_cast<int>(FillMethod::KEYBORD))

FillArrayKeyboard(arr, n);

else

FillArrayRandom(arr, n, left, right);

PrintArray(arr, n);

int c;

cout << "Введите заданное число: ";

cin >> c;

cout << "Сумма четных элементов, больших заданного числа: " << SumTask(arr, n, c) << endl;

cout << "Умножаем все четные положительные на последний элемент..." << endl << endl;

MultLast(arr, n);

PrintArray(arr, n);

int position = PosPair(arr, n);

if (position == -1)

cout << "Пары элементов с разными знаками не найдено" << endl;

else

cout << "Индекс пары элементов с разными знаками: " << position << endl;

if (arr != nullptr) {

delete[] arr;

arr = nullptr;

}

return 0;

}

void FillArrayKeyboard(int\* arr, const size\_t n) {

for (size\_t i = 0; i < n; i++) {

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

cin >> arr[i];

}

cout << endl;

}

void FillArrayRandom(int\* arr, const size\_t n, const int min, const int max) {

random\_device rd;

mt19937 gen(rd());

const uniform\_int\_distribution<int> dist(min, max);

for (size\_t i = 0; i < n; i++)

arr[i] = dist(gen);

}

void PrintArray(const int\* arr, const size\_t n) {

for (size\_t i = 0; i < n; i++)

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

cout << endl;

}

int SumTask(const int\* arr, const size\_t n, const int c) {

int sum = 0;

for (size\_t i = 0; i < n; i++)

if (arr[i] % 2 == 0 && arr[i] > c)

sum += arr[i];

return sum;

}

void MultLast(int\* arr, const size\_t n) {

for (size\_t i = 0; i < n; i++)

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

arr[i] \*= arr[n - 1];

}

int PosPair(const int\* arr, const size\_t n) {

for (size\_t i = 0; i < n - 1; i++)

if (arr[i] \* arr[i + 1] < 0)

return i;

return -1;

}

# **Решение тестового примера на C++**

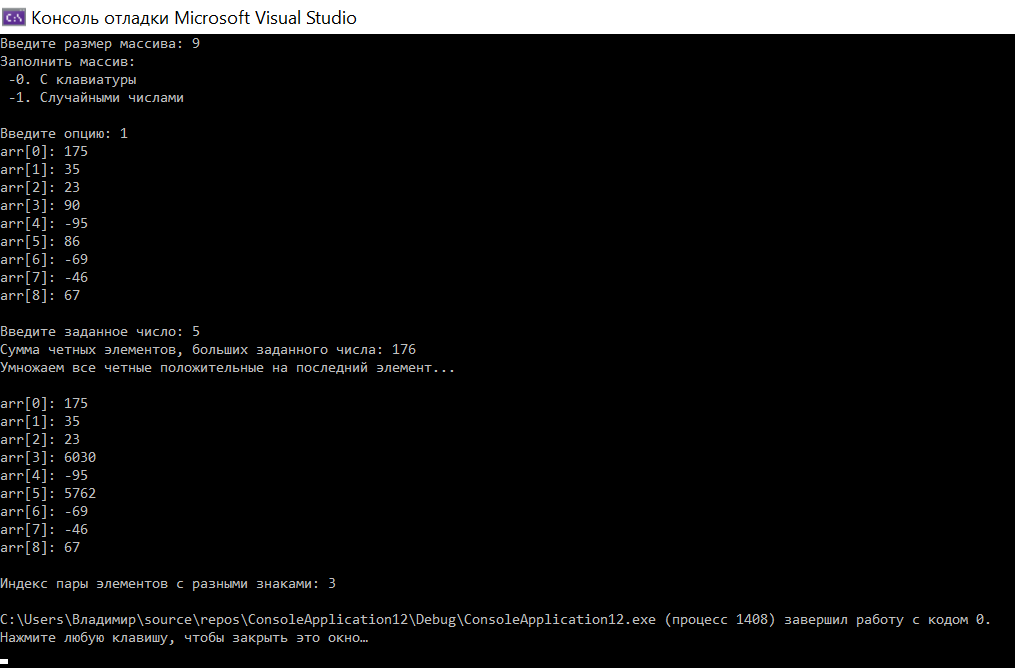
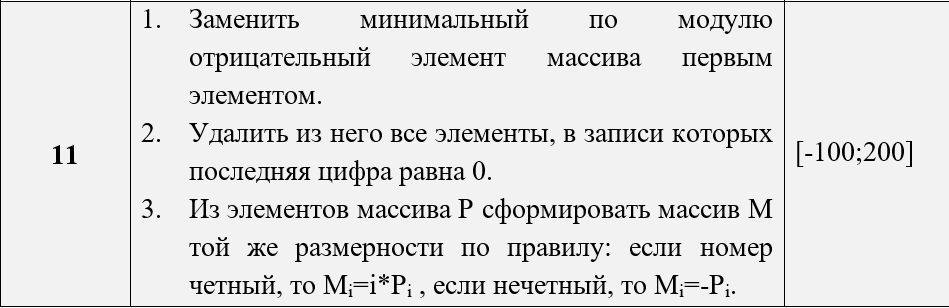


Рисунок 5 – Решение тестового примера на C++

# **Задание 4-2**

# **Формулировка задания**

Создать одномерный массив из *n* целых чисел. Реализовать возможность заполнения массива, как случайными числами, так и с помощью клавиатуры по желанию пользователя. Пункты задания организовать в виде функций (методов). Вывести массив на экран. Составить блок-схему.

****

# **Блок-схема алгоритма**

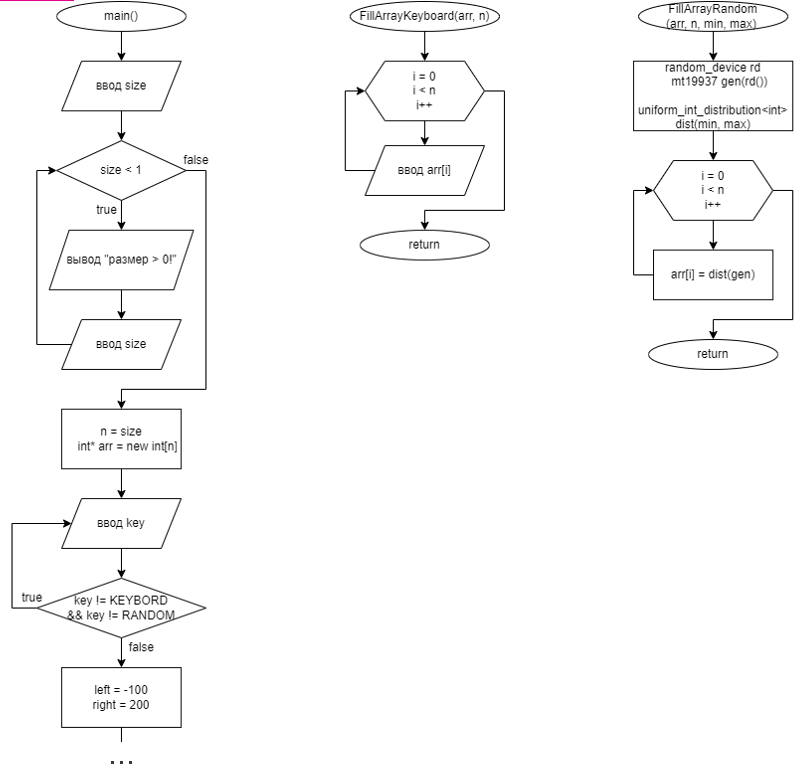
****

Рисунок 6 – Блок-схема алгоритма

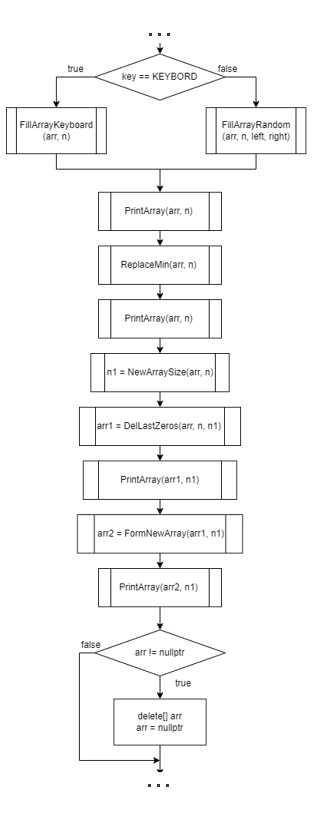
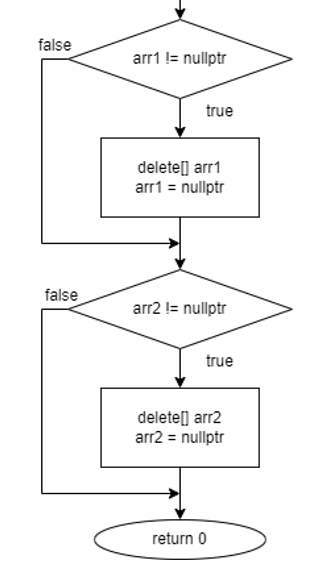
****

Рисунок 7 – Блок-схема алгоритма

**…**

****

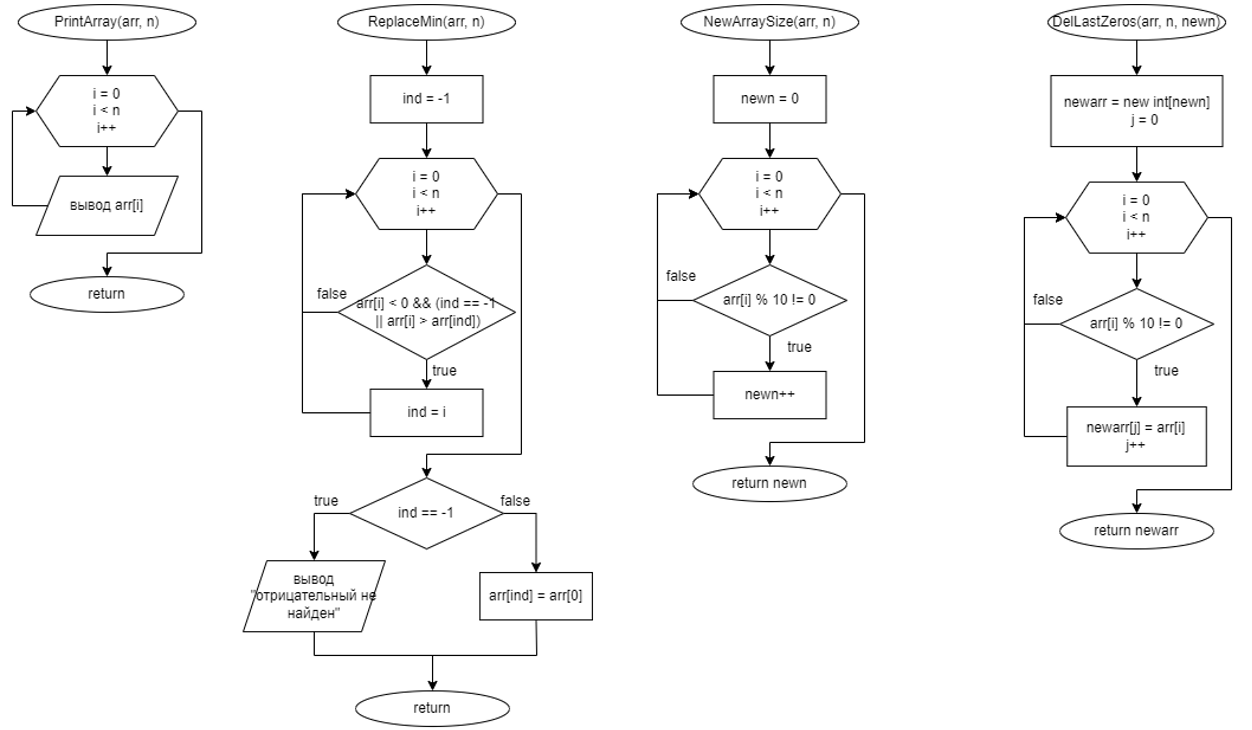
****

Рисунок 8 – Блок-схема алгоритма

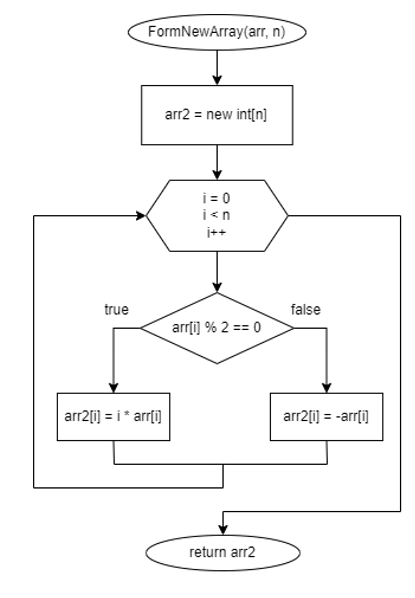
****

Рисунок 9 – Блок-схема алгоритма

# **Программа на языке C++**

#include <iostream>

#include <random>

using namespace std;

/\*\*

\* \brief Ввод элементов массива с клавиатуры.

\* \param arr - Указатель на массив.

\* \param n - Размер массива.

\*/

void FillArrayKeyboard(int\* arr, const size\_t n);

/\*\*

\* \brief Заполнение элементов массива случайными числами.

\* \param arr - Указатель на массив.

\* \param n - Размер массива.

\* \param min - Левая граница диапазона случайных чисел.

\* \param max - Правая граница диапазона случайных чисел.

\*/

void FillArrayRandom(int\* arr, const size\_t n, const int min, const int max);

/\*\*

\* \brief Вывод элементов массива в консоль.

\* \param arr - Указатель на массив.

\* \param n - Размер массива.

\*/

void PrintArray(const int\* arr, const size\_t n);

/\*\*

\* \brief Замена минимального по модулю отрицательного числа.

\* \param arr - Указатель на массив.

\* \param n - Размер массива.

\*/

void ReplaceMin(int\* arr, const size\_t n);

/\*\*

\* \brief Количество элементов, не оканчивающихся на 0.

\* \param arr - Указатель на старый массив.

\* \param n - Размер старого массива.

\* \return - Размер нового массива.

\*/

size\_t NewArraySize(const int\* arr, const size\_t n);

/\*\*

\* \brief Удаление элементов оканчивающихся на 0.

\* \param arr - Указатель на старый массив.

\* \param n - Размер старого массива.

\* \param newn - Размер нового массива.

\* \return - Адрес нового массива.

\*/

int\* DelLastZeros(const int\* arr, const size\_t n, const size\_t newn);

/\*\*

\* \brief Формирование нового массива по заданным правилам.

\* \param arr - Указатель на массив.

\* \param n - Размер массива.

\*/

int\* FormNewArray(const int\* arr, const size\_t n);

enum class FillMethod { //перечислимый тип

/\*\*

\* \brief Заполнение элементов массива с клавиатуры.

\*/

KEYBORD,

/\*\*

\* \brief Заполнение элементов массива случайными числами.

\*/

RANDOM

};

int main()

{

setlocale(LC\_ALL, "rus");

size\_t n, key;

int size;

cout << "Введите размер массива: ";

cin >> size;

while (size < 1) {

cout << "Размер должен быть положительным! Введите размер: ";

cin >> size;

};

n = size;

int\* arr = new int[n];

do {

cout << "Заполнить массив:\n -" << static\_cast<int>(FillMethod::KEYBORD) << ". С клавиатуры\n -" << static\_cast<int>(FillMethod::RANDOM) << ". Случайными числами\n\nВведите опцию: ";

cin >> key;

} while (key != static\_cast<int>(FillMethod::KEYBORD) && key != static\_cast<int>(FillMethod::RANDOM));

const int left = -100;

const int right = 200;

if (key == static\_cast<int>(FillMethod::KEYBORD))

FillArrayKeyboard(arr, n);

else

FillArrayRandom(arr, n, left, right);

PrintArray(arr, n);

ReplaceMin(arr, n);

PrintArray(arr, n);

size\_t n1 = NewArraySize(arr, n);

int\* arr1 = DelLastZeros(arr, n, n1);

PrintArray(arr1, n1);

int\* arr2 = FormNewArray(arr1, n1);

cout << "Сформированный массив: " << endl;

PrintArray(arr2, n1);

if (arr != nullptr) {

delete[] arr;

arr = nullptr;

}

if (arr1 != nullptr) {

delete[] arr1;

arr1 = nullptr;

}

if (arr2 != nullptr) {

delete[] arr2;

arr2 = nullptr;

}

return 0;

}

void FillArrayKeyboard(int\* arr, const size\_t n) {

for (size\_t i = 0; i < n; i++) {

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

cin >> arr[i];

}

cout << endl;

}

void FillArrayRandom(int\* arr, const size\_t n, const int min, const int max) {

random\_device rd;

mt19937 gen(rd());

const uniform\_int\_distribution<int> dist(min, max);

for (size\_t i = 0; i < n; i++)

arr[i] = dist(gen);

}

void PrintArray(const int\* arr, const size\_t n) {

for (size\_t i = 0; i < n; i++)

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

cout << endl;

}

void ReplaceMin(int\* arr, const size\_t n) {

int ind = -1; //не найден отрицательный

for (size\_t i = 0; i < n; i++)

if (arr[i] < 0 && (ind == -1 || arr[i] > arr[ind]))

ind = i;

if (ind == -1)

cout << "Отрицательный элемент не найден" << endl;

else {

cout << "Минимальный по модулю отрицательный элемент " << arr[ind] << " заменен на первый " << arr[0] << endl;

arr[ind] = arr[0];

}

}

size\_t NewArraySize(const int\* arr, const size\_t n) {

size\_t newn = 0;

for (size\_t i = 0; i < n; i++) {

if (arr[i] % 10 != 0) {

newn++;

}

}

return newn;

}

int\* DelLastZeros(const int\* arr, const size\_t n, const size\_t newn) {

cout << "Массив без элементов, оканчивающихся на 0: " << endl;

int\* newarr = new int[newn];

size\_t j = 0;

for (size\_t i = 0; i < n; i++) {

if (arr[i] % 10 != 0) {

newarr[j] = arr[i];

j++;

}

}

return newarr;

}

int\* FormNewArray(const int\* arr, const size\_t n) {

int\* arr2 = new int[n];

for (size\_t i = 0; i < n; i++)

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

arr2[i] = i \* arr[i];

else

arr2[i] = -arr[i];

return arr2;

}

# **Решение тестового примера на C++**

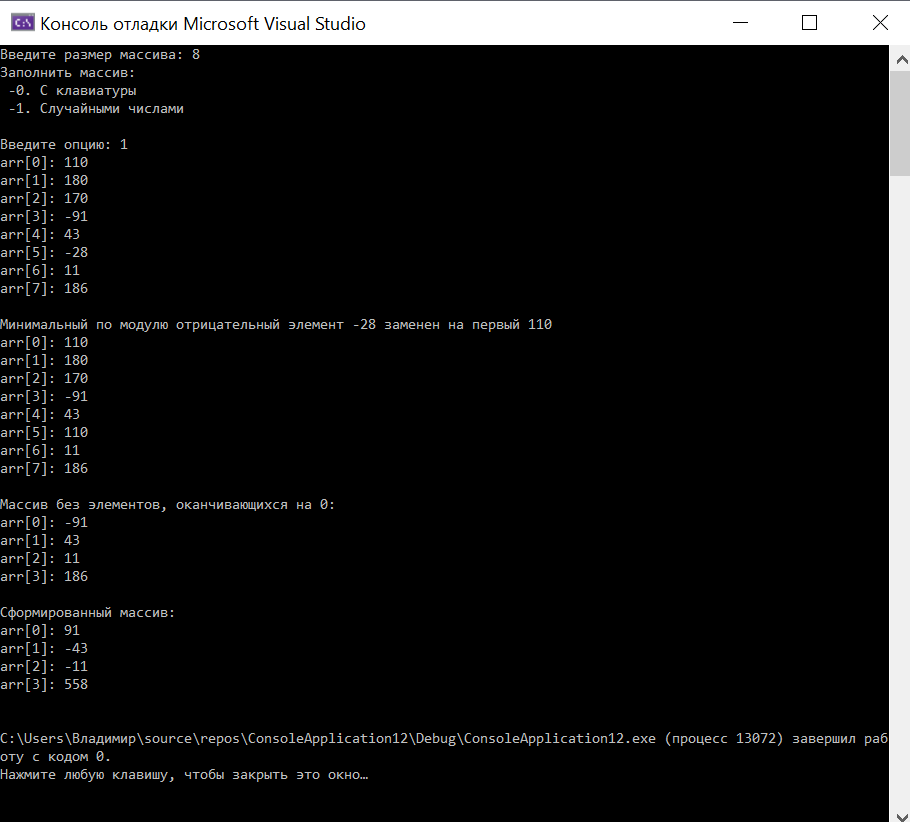
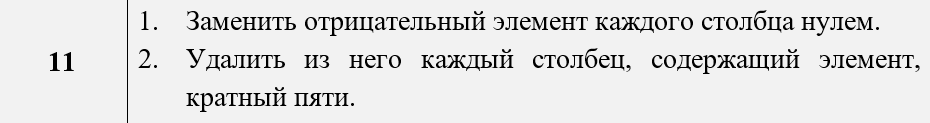
****

Рисунок 10 – Решение тестового примера в C++

# **Задание 4-3**

# **Формулировка задания**

Создать многомерный массив nˣm из *n* целых чисел. Реализовать возможность заполнения массива, как случайными числами, так и с помощью клавиатуры по желанию пользователя. Пункты задания организовать в виде функций (методов). Вывести массив на экран.

****

# **Блок-схема алгоритма**

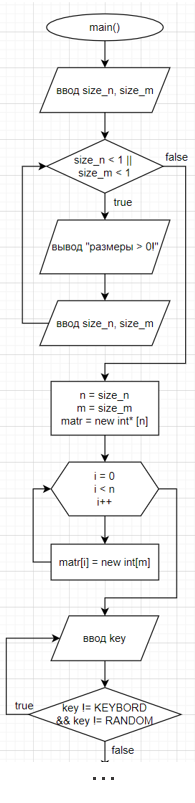
****

Рисунок 11 – Блок-схема алгоритма

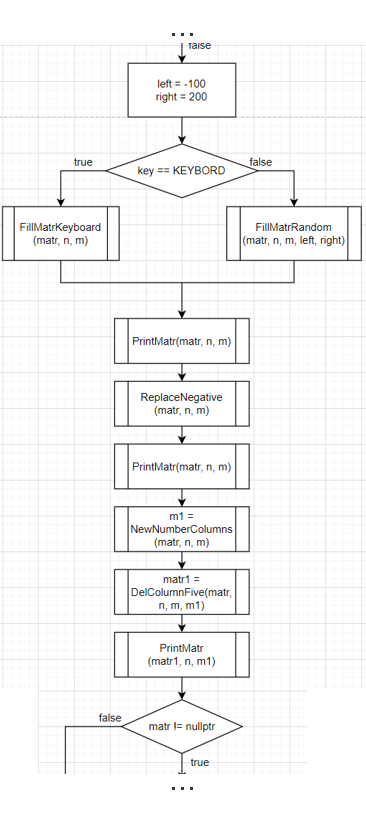
****

Рисунок 12 – Блок-схема алгоритма

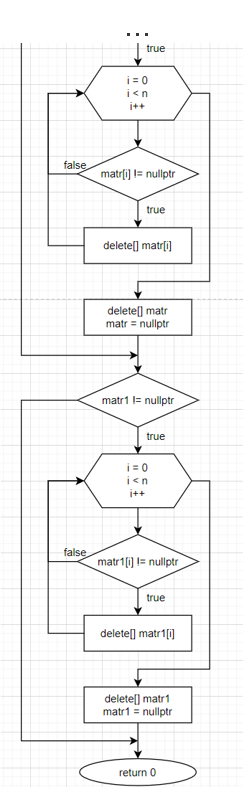
****

Рисунок 13 – Блок-схема алгоритма

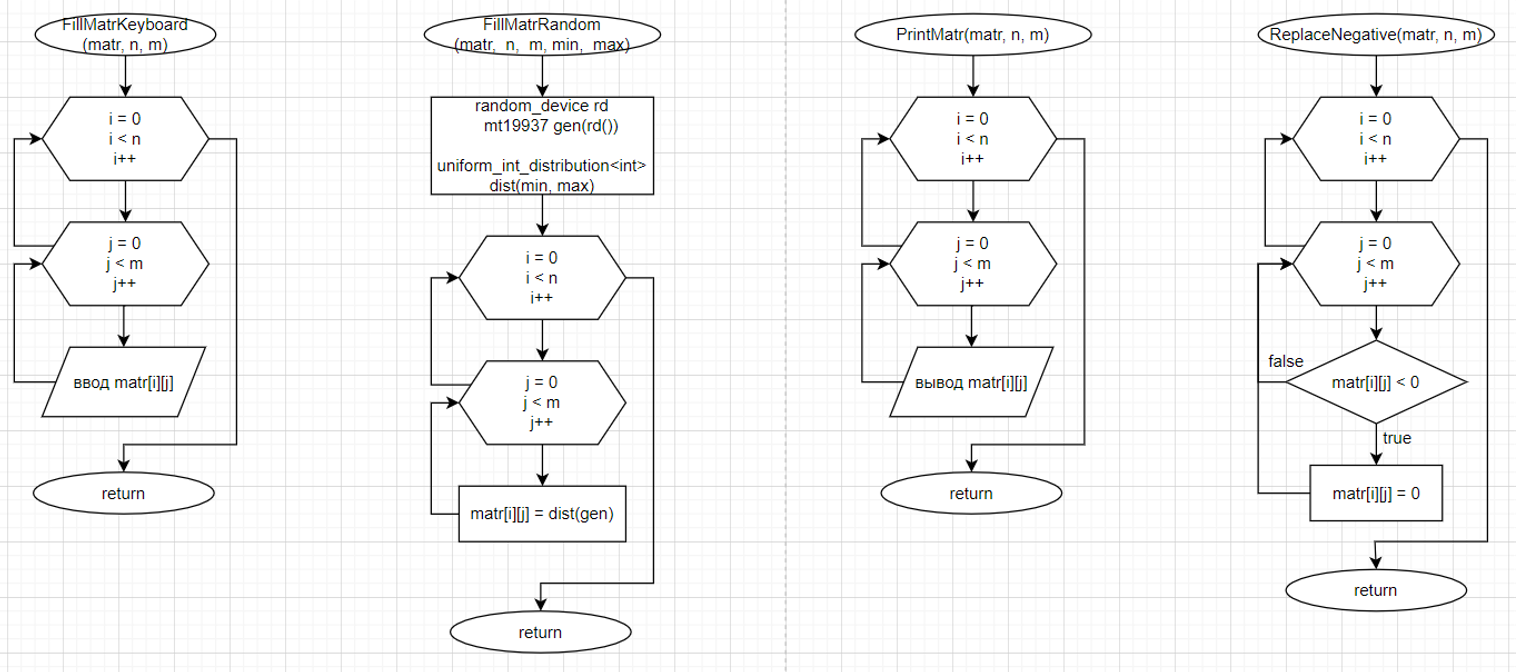
****

Рисунок 14 – Блок-схема алгоритма

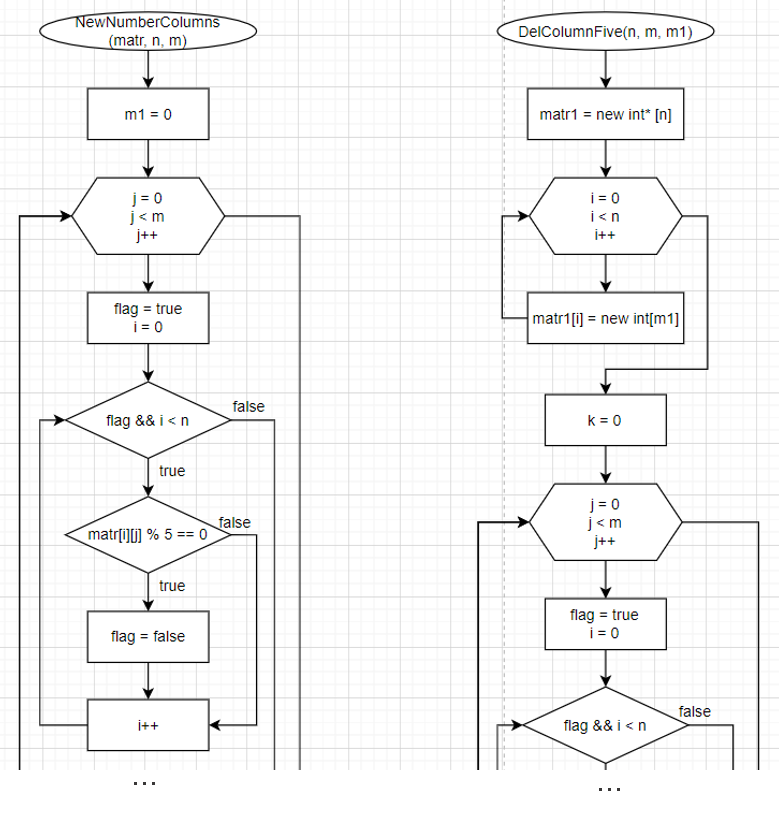
****

Рисунок 15 – Блок-схема алгоритма

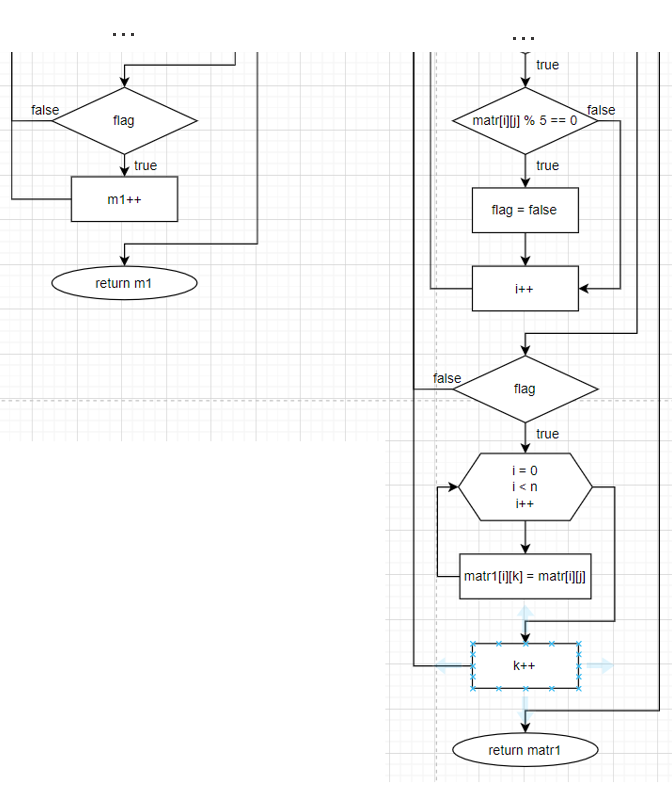
****

Рисунок 16 – Блок-схема алгоритма

# **Программа на языке C++**

#include <iostream>

#include <iomanip>

#include <random>

using namespace std;

/\*\*

\* \brief Ввод элементов матрицы с клавиатуры.

\* \param matr - Указатель на матрицу.

\* \param n - Количество строк.

\* \param m - Количество столбцов.

\*/

void FillMatrKeyboard(int\*\* matr, const size\_t n, const size\_t m);

/\*\*

\* \brief Заполнение элементов матрицы случайными числами.

\* \param matr - Указатель на матрицу.

\* \param n - Количество строк матрицы.

\* \param m - Количество столбцов матрицы.

\* \param min - Левая граница диапазона случайных чисел.

\* \param max - Правая граница диапазона случайных чисел.

\*/

void FillMatrRandom(int\*\* matr, const size\_t n, const size\_t m, const int min, const int max);

/\*\*

\* \brief Вывод элементов матрицы в консоль.

\* \param matr - Указатель на матрицу.

\* \param n - Количество строк матрицы.

\* \param m - Количество столбцов матрицы.

\*/

void PrintMatr(int\*\* matr, const size\_t n, const size\_t m);

/\*\*

\* \brief Замена отрицательных элементов матрицы на 0.

\* \param matr - Указатель на матрицу.

\* \param n - Количество строк матрицы.

\* \param m - Количество столбцов матрицы.

\*/

void ReplaceNegative(int\*\* matr, const size\_t n, const size\_t m);

/\*\*

\* \brief Количество столбцов, не содержащих элемент кратный 5.

\* \param matr - Указатель на матрицу.

\* \param n - Количество строк старой матрицы.

\* \param m - Количество столбцов старой матрицы.

\* \return - Количество столбцов новой матрицы.

\*/

size\_t NewNumberColumns(int\*\* matr, const size\_t n, const size\_t m);

/\*\*

\* \brief Удаление всех столбцов содержащих элемент кратный 5.

\* \param matr - Указатель на матрицу.

\* \param n - Количество строк старой матрицы.

\* \param m - Количество столбцов старой матрицы.

\* \param m1 - Количество столбцов новой матрицы.

\* \return - Адрес новой матрицы.

\*/

int\*\* DelColumnFive(int\*\* matr, const size\_t n, const size\_t m, const size\_t m1);

enum class FillMethod { //перечислимый тип

/\*\*

\* \brief Заполнение элементов массива с клавиатуры.

\*/

KEYBORD,

/\*\*

\* \brief Заполнение элементов массива случайными числами.

\*/

RANDOM

};

int main()

{

setlocale(LC\_ALL, "rus");

size\_t n, m, key; //n-количетсво строк(высота), m-количество столбцов(ширина)

int size\_n, size\_m;

cout << "Введите размеры матрицы: ";

cin >> size\_n >> size\_m;

while (size\_n < 1 || size\_m < 1) {

cout << "Размеры должны быть положительными! Введите размеры: ";

cin >> size\_n >> size\_m;

};

n = size\_n;

m = size\_m;

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

for (size\_t i = 0; i < n; i++)

matr[i] = new int[m];

do {

cout << "Заполнить массив:\n -" << static\_cast<int>(FillMethod::KEYBORD) << ". С клавиатуры\n -" << static\_cast<int>(FillMethod::RANDOM) << ". Случайными числами\n\nВведите опцию: ";

cin >> key;

} while (key != static\_cast<int>(FillMethod::KEYBORD) && key != static\_cast<int>(FillMethod::RANDOM));

const int left = -100;

const int right = 200;

if (key == static\_cast<int>(FillMethod::KEYBORD))

FillMatrKeyboard(matr, n, m);

else

FillMatrRandom(matr, n, m, left, right);

PrintMatr(matr, n, m);

cout << "Заменяем отрицательные на нули: " << endl;

ReplaceNegative(matr, n, m);

PrintMatr(matr, n, m);

cout << "Удаляем столбцы, содержащие элементы, кратные пяти: " << endl;

size\_t m1 = NewNumberColumns(matr, n, m);

int\*\* matr1 = DelColumnFive(matr, n, m, m1);

PrintMatr(matr1, n, m1);

if (matr != nullptr) {

for (size\_t i = 0; i < n; i++) {

if (matr[i] != nullptr)

delete[] matr[i];

}

delete[] matr;

matr = nullptr;

}

if (matr1 != nullptr) {

for (size\_t i = 0; i < n; i++) {

if (matr1[i] != nullptr)

delete[] matr1[i];

}

delete[] matr1;

matr1 = nullptr;

}

return 0;

}

void FillMatrKeyboard(int\*\* matr, const size\_t n, const size\_t m) {

for (size\_t i = 0; i < n; i++)

for (size\_t j = 0; j < m; j++) {

cout << "matr[" << i << "][" << j << "]: ";

cin >> matr[i][j];

}

cout << endl;

}

void FillMatrRandom(int\*\* matr, const size\_t n, const size\_t m, const int min, const int max) {

random\_device rd;

mt19937 gen(rd());

const uniform\_int\_distribution<int> dist(min, max);

for (size\_t i = 0; i < n; i++)

for (size\_t j = 0; j < m; j++)

matr[i][j] = dist(gen);

}

void PrintMatr(int\*\* matr, const size\_t n, const size\_t m) {

for (size\_t i = 0; i < n; i++) {

for (size\_t j = 0; j < m; j++)

cout << setw(5) << matr[i][j];

cout << endl;

}

cout << endl;

}

void ReplaceNegative(int\*\* matr, const size\_t n, const size\_t m) {

for (size\_t i = 0; i < n; i++)

for (size\_t j = 0; j < m; j++)

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

matr[i][j] = 0;

}

size\_t NewNumberColumns(int\*\* matr, const size\_t n, const size\_t m) {

size\_t m1 = 0; //количество столбцов в новой матрице

for (size\_t j = 0; j < m; j++) { //бежим по индексам столбцов

bool flag = true; //столбец не удалять

size\_t i = 0; //текущий номер строки

while (flag && i < n) {

if (matr[i][j] % 5 == 0) {

flag = false;

}

i++;

}

if (flag) m1++;

}

return m1;

}

int\*\* DelColumnFive(int\*\* matr, const size\_t n, const size\_t m, const size\_t m1) {

int\*\* matr1 = new int\* [n];

for (size\_t i = 0; i < n; i++)

matr1[i] = new int[m1];

size\_t k = 0;

for (size\_t j = 0; j < m; j++) { //бежим по индексам столбцов

bool flag = true; //столбец не удалять

size\_t i = 0; //текущий номер строки

while (flag && i < n) {

if (matr[i][j] % 5 == 0) {

flag = false;

}

i++;

}

if (flag) {

for (size\_t i = 0; i < n; i++) {

matr1[i][k] = matr[i][j];

}

k++;

}

}

return matr1;

}

# **Решение тестового примера на C++**

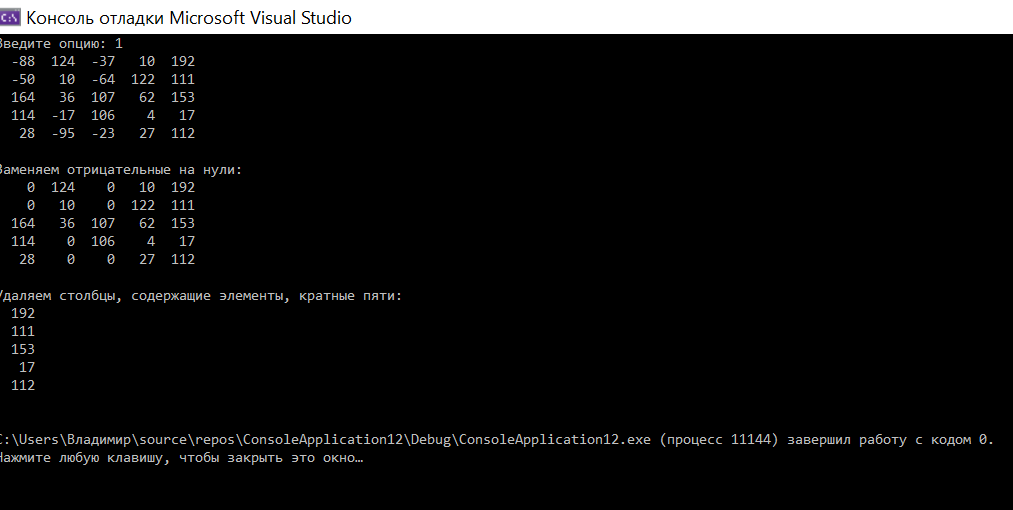
****

Рисунок 17 – Решение тестового примера на C++