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 №1

По дисциплине «Введение в языки программирования»

Выполнил: ст. гр. ТКИ – 111

Панаргин В.М.

Проверил: к.т.н., доц.

Васильева М.А.

Москва 2021

Оглавление

[Задание 4-1 3](#_Toc91679514)

[Условие задания 3](#_Toc91679515)

[Блок-схема алгоритма 5](#_Toc91679516)

[Программа на языке С++ 7](#_Toc91679517)

[Решение тестовых примеров на языке С++ 11](#_Toc91679518)

[Задание 4-2 12](#_Toc91679519)

[Условие задания 12](#_Toc91679520)

[Блок-схема алгоритма 13](#_Toc91679521)

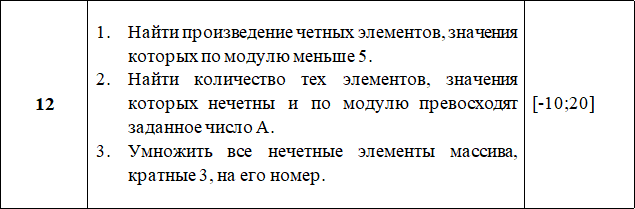
[Программа на языке С++ 15](#_Toc91679522)

[Решение тестовых примеров на языке С++ 19](#_Toc91679523)

# Задание 4-1

## Условие 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). Составить блок-схему.



## Блок-схема алгоритма

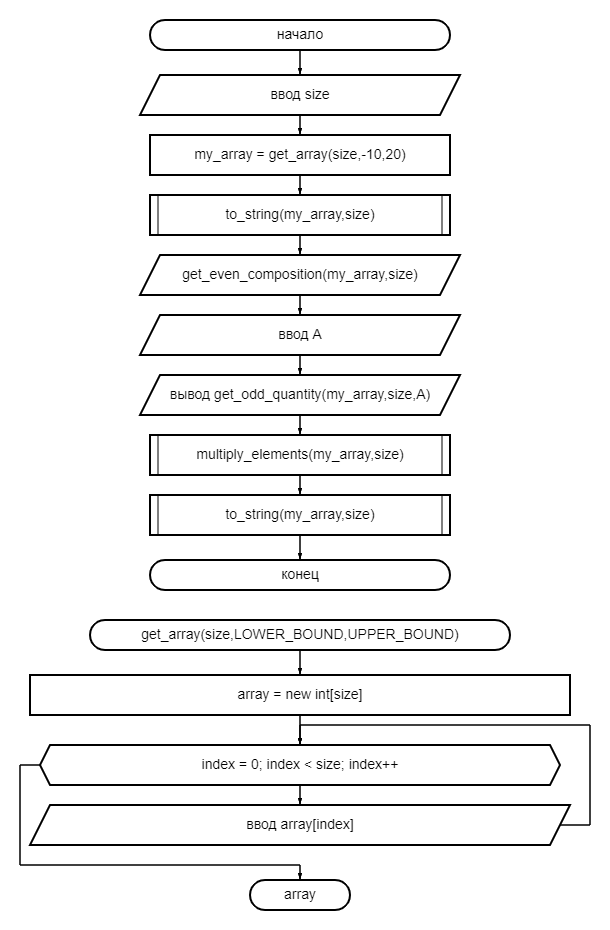


Рисунок 1 - Блок-схема 4-1 часть 1

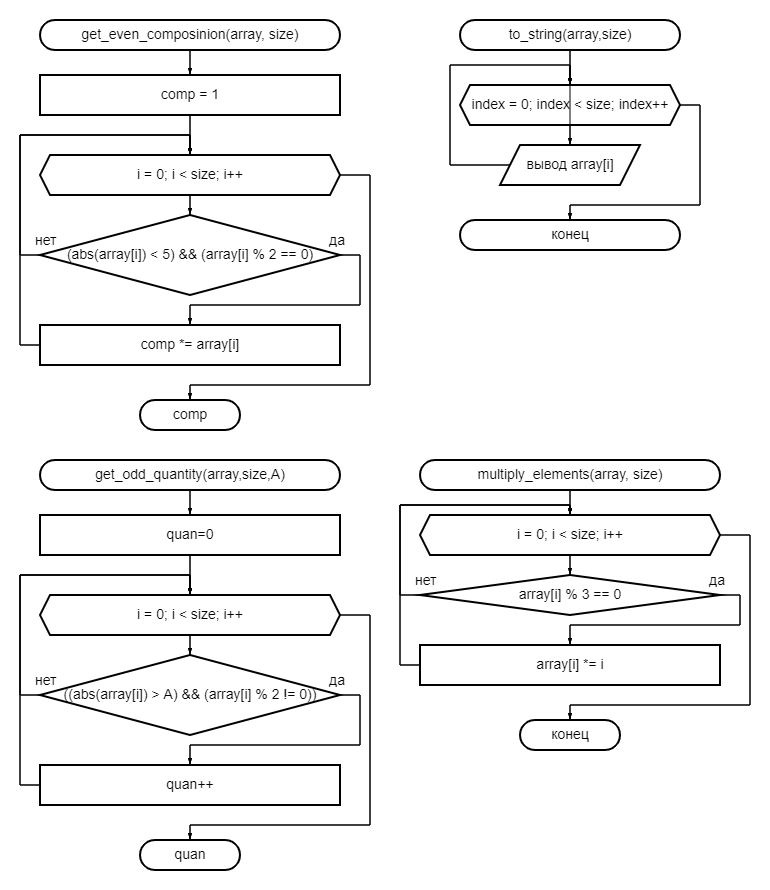


Рисунок 2 - Блок-схема 4-1 часть 2

## Программа на языке С++

#include <iostream>

#include <sstream>

#include <random>

using namespace std;

/\*\*

\* \brief Перечислимый тип способов задания массива.

\*/

enum class input\_types

{

/\*\*

\* \brief ввод вручную.

\*/

MANUALLY,

/\*\*

\* \brief создание с использованием генератора случайных чисел.

\*/

RANDOMLY

};

/\*\*

\* \brief Метод заполнения массива

\* \param size Размер массива

\* \param input\_type Способ создания массива

\* \param LOWER\_BOUND Нижняя граница для случайных чисел

\* \param UPPER\_BOUND Верхняя граница для случайных чисел

\* \return Заполненный массив

\*/

int\* get\_array(const size\_t size, const int input\_type, const int LOWER\_BOUND, const int UPPER\_BOUND);

/\*\*

\* \brief Функция для получения размера массива

\* \param message Мотивоционное сообщение для пользователя

\* \return Размер массива

\*/

size\_t get\_size(const std::string& message);

/\*\*

\* \brief Вывод в строку элементов массива

\* \param size Размер массива

\* \return Строка с элементами

\*/

string to\_string(const int\*, const size\_t size);

/\*\*

\* \brief Находит произведение четных элементов, значение которых по модулю меньше 5

\* \param size Размер массива

\* \return Произведение элементов

\*/

int get\_even\_composition(int\*, const size\_t size);

/\*\*

\* \brief Находит количество нечетных элементов по модулю превышающих заданное число (\a A)

\* \param size Размер массива

\* \param A Число A

\* \return Количество элементов

\*/

int get\_odd\_quantity(int\*, const size\_t size, const int A);

/\*\*

\* \brief Умножает все нечетные элементы массива кратные 3 на их номер

\* \param size Размер массива

\* \return Количество элементов

\*/

void multiply\_elements(int\*, const size\_t size);

/\*\*

\* \brief Точка входа в программу

\* \return 0 в случае успеха

\*/

int main()

{

setlocale(LC\_ALL, "Russian");

auto error\_code = 0;

int\* my\_array = nullptr;

const int LOWER\_BOUND = -10;

const int UPPER\_BOUND = 20;

try

{

const auto size = get\_size("Введите размер массива = ");

cout << "Выберите способ создания массива: " << static\_cast<int>(input\_types::MANUALLY) << " - вручную, " << static\_cast<int>(input\_types::RANDOMLY) << " - заполнить случайными числами ";

int input\_type;

cin >> input\_type;

my\_array = get\_array(size, input\_type, LOWER\_BOUND, UPPER\_BOUND);

cout << "Полученный массив:\n";

cout << to\_string(my\_array, size);

cout << "\nПроизведение четных элементов, значение которых по модулю меньше 5 = " << get\_even\_composition(my\_array, size) << '\n';

cout << "Введите число A = ";

int A;

cin >> A;

cout << "Количество нечетных элементов по модулю превышающих заданное число (A) = " << get\_odd\_quantity(my\_array, size, A) << '\n';

cout << "Массив после умножаения всех нечетных элементов массива кратных 3 на их номер:\n";

multiply\_elements(my\_array, size);

cout << to\_string(my\_array, size);

}

catch (exception& e)

{

cout << e.what();

error\_code = 1;

}

if (my\_array != nullptr)

{

delete[] my\_array;

my\_array = nullptr;

}

return error\_code;

}

size\_t get\_size(const std::string& message)

{

int size = -1;

std::cout << message;

std::cin >> size;

if (size < 0)

{

throw std::out\_of\_range("Incorrect size. Value has to be greater or equal zero.");

}

return size;

}

int\* get\_array(const size\_t size, const int input\_type, const int LOWER\_BOUND, const int UPPER\_BOUND)

{

if (size == 0)

throw out\_of\_range("Неправильный размер массива");

const auto array = new int[size];

//Will be used to obtain a seed for the random number engine

std::random\_device rd;

//Standard mersenne\_twister\_engine seeded with rd()

std::mt19937 gen(rd());

const std::uniform\_int\_distribution<> uniformIntDistribution(LOWER\_BOUND, UPPER\_BOUND);

for (size\_t index = 0; index < size; index++)

{

switch (input\_type)

{

case static\_cast<int>(input\_types::MANUALLY):

{

cout << "Введите " << index + 1 << " элемент массива = ";

cin >> array[index];

break;

}

case static\_cast<int>(input\_types::RANDOMLY):

{

array[index] = uniformIntDistribution(gen);

break;

}

default:

cout << "Ошибка!";

}

}

return array;

}

string to\_string(const int\* array, const size\_t size)

{

if (array == nullptr)

throw invalid\_argument("Массив не существует");

stringstream buffer;

buffer << "{";

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

{

buffer << array[index] << ", ";

}

buffer << array[size - 1] << "}";

return buffer.str();

}

int get\_even\_composition(int\* array, const size\_t size) {

int comp = 1;

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

if ((abs(array[i]) < 5) && (array[i] % 2 == 0))

comp \*= array[i];

}

return comp;

}

int get\_odd\_quantity(int\* array, const size\_t size, const int A) {

int quan = 0;

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

if ((abs(array[i]) > A) && (array[i] % 2 != 0))

quan++;

}

return quan;

}

void multiply\_elements(int\* array, const size\_t size) {

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

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

array[i] \*= i;

}

}

## Решение тестовых примеров на языке С++

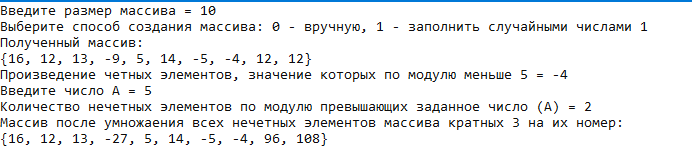
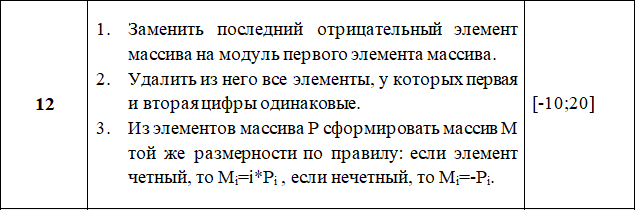


Рисунок 3 - Решение тестовых примеров для задания 4-1 на языке С++

# Задание 4-2

## Условие 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

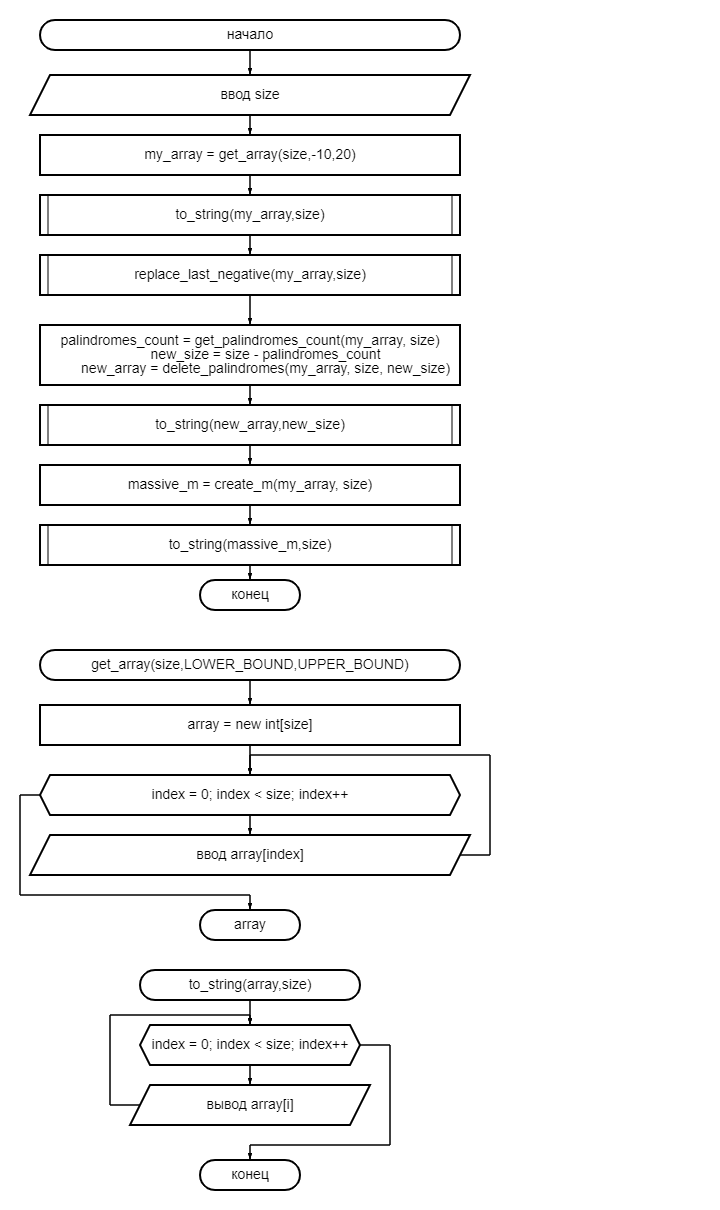


Рисунок 4 - Блок-схема 4-2 часть 1

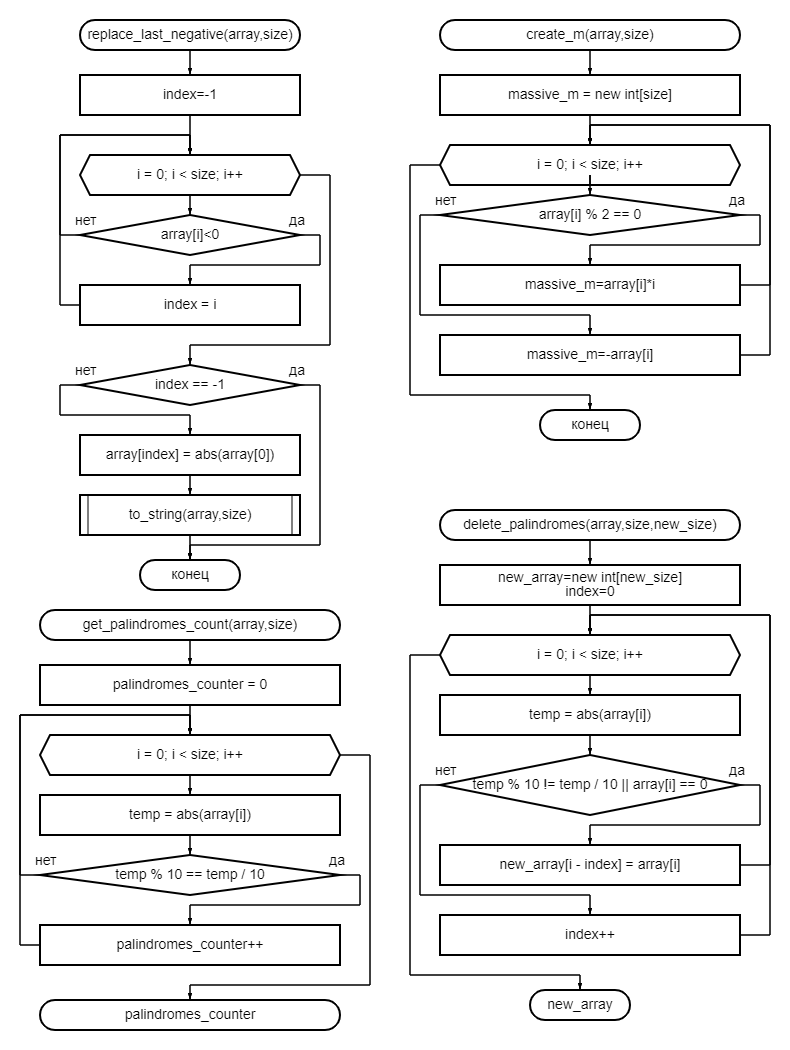


Рисунок 5 - Блок-схема 4-2 часть 2

## Программа на языке С++

#include <iostream>

#include <sstream>

#include <random>

using namespace std;

/\*\*

\* \brief Перечислимый тип способов задания массива.

\*/

enum class input\_types

{

/\*\*

\* \brief ввод вручную.

\*/

MANUALLY,

/\*\*

\* \brief создание с использованием генератора случайных чисел.

\*/

RANDOMLY

};

/\*\*

\* \brief Метод заполнения массива

\* \param size Размер массива

\* \param input\_type Способ создания массива

\* \param LOWER\_BOUND Нижняя граница для случайных чисел

\* \param UPPER\_BOUND Верхняя граница для случайных чисел

\* \return Заполненный массив

\*/

int\* get\_array(const size\_t size, const int input\_type, const int LOWER\_BOUND, const int UPPER\_BOUND);

/\*\*

\* \brief Функция для получения размера массива

\* \param message Мотивоционное сообщение для пользователя

\* \return Размер массива

\*/

size\_t get\_size(const std::string& message);

/\*\*

\* \brief Вывод в строку элементов массива

\* \param size Размер массива

\* \return Строка с элементами

\*/

string to\_string(const int\*, const size\_t size);

/\*\*

\* \brief Заменяет последний отрицательный элемент массива на модуль первого

\* \param size Размер массива

\*/

void replace\_last\_negative(int\*, const size\_t size);

/\*\*

\* \brief Возвращает массив, в котором удалены все числа в котором первая и последняя цифры одинаковые

\* \param size Размер массива

\* \param newsize Размер нового массива

\* \return Массив, в котором удалены все числа в котором первая и последняя цифры одинаковые

\*/

int\* delete\_palindromes(int\*, size\_t size, size\_t newsize);

/\*\*

\* \brief Считает количество чисел с одинаковой первой и последней цифрой (\a index)

\* \param size Размер массива

\* \return Количество чисел с одинаковой первой и последней цифрой

\*/

size\_t get\_palindromes\_count(int\*, size\_t size);

/\*\*

\* \brief Создает из исходного массива новый массив P по определенному правилу

\* \param size Размер массива

\* \return массив P

\*/

int\* сreate\_m(int\*, const size\_t size);

/\*\*

\* \brief Точка входа в программу

\* \return 0 в случае успеха

\*/

int main()

{

setlocale(LC\_ALL, "Russian");

auto error\_code = 0;

int\* my\_array = nullptr;

const int LOWER\_BOUND = -10;

const int UPPER\_BOUND = 20;

try

{

auto size = get\_size("Введите размер массива = ");

cout << "Выберите способ создания массива: " << static\_cast<int>(input\_types::MANUALLY) << " - вручную, " << static\_cast<int>(input\_types::RANDOMLY) << " - заполнить случайными числами ";

int input\_type;

cin >> input\_type;

my\_array = get\_array(size, input\_type, LOWER\_BOUND, UPPER\_BOUND);

cout << "Полученный массив:\n";

cout << to\_string(my\_array, size);

replace\_last\_negative(my\_array, size);

size\_t palindromes\_count = get\_palindromes\_count(my\_array, size);

size\_t new\_size = size - palindromes\_count;

int\* new\_array = delete\_palindromes(my\_array, size, new\_size);

cout << "\nМассив после удаления всех элементов с одинаковой начальной и конечной цифрой:\n";

cout << to\_string(new\_array, new\_size);

int\* massive\_m = сreate\_m(my\_array, size);

cout << "\nМассив M:\n";

cout << to\_string(massive\_m, size);

if (new\_array != nullptr) {

delete[] new\_array;

new\_array = nullptr;

}

if (massive\_m != nullptr) {

delete massive\_m;

massive\_m = nullptr;

}

}

catch (exception& e)

{

cout << e.what();

error\_code = 1;

}

if (my\_array != nullptr)

{

delete[] my\_array;

my\_array = nullptr;

}

return error\_code;

}

size\_t get\_size(const std::string& message)

{

int size = -1;

std::cout << message;

std::cin >> size;

if (size <= 0)

{

throw std::out\_of\_range("Incorrect size. Value has to be greater or equal zero.");

}

return size;

}

int\* get\_array(const size\_t size, const int input\_type, const int LOWER\_BOUND, const int UPPER\_BOUND)

{

const auto array = new int[size];

std::random\_device rd;

//Standard mersenne\_twister\_engine seeded with rd()

std::mt19937 gen(rd());

const std::uniform\_int\_distribution<> uniform\_int\_distribution(LOWER\_BOUND, UPPER\_BOUND);

for (size\_t index = 0; index < size; index++)

{

switch (input\_type)

{

case static\_cast<int>(input\_types::MANUALLY):

{

cout << "Введите " << index + 1 << " элемент массива = ";

cin >> array[index];

break;

}

case static\_cast<int>(input\_types::RANDOMLY):

{

array[index] = uniform\_int\_distribution(gen);

break;

}

default:

cout << "Ошибка!";

}

}

return array;

}

string to\_string(const int\* array, const size\_t size)

{

if (array == nullptr)

throw invalid\_argument("Массив не существует");

stringstream buffer;

if (size == 0) {

buffer << "{}";

}

else {

buffer << "{";

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

{

buffer << array[index] << ", ";

}

buffer << array[size - 1] << "}";

}

return buffer.str();

}

void replace\_last\_negative(int\* array, const size\_t size) {

size\_t index = -1;

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

if (array[i] < 0) {

index = i;

}

}

if (index == -1) {

cout << "\nВ массиве нет отрицательных чисел";

}

else {

array[index] = abs(array[0]);

cout << "\nМассив после замены последнего отрицательного элемента на модуль первого:\n";

cout << to\_string(array, size);

}

}

int\* delete\_palindromes(int\* array, size\_t size, size\_t new\_size) {

int\* new\_array = new int[new\_size];

size\_t index = 0;

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

int temp = abs(array[i]);

if (temp % 10 != temp / 10 || array[i] == 0) {

new\_array[i - index] = array[i];

}

else {

index++;

}

}

return new\_array;

}

size\_t get\_palindromes\_count(int\* array, size\_t size) {

size\_t palindromes\_counter = 0;

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

int temp = abs(array[i]);

if (temp % 10 == temp / 10) {

palindromes\_counter++;

}

}

return palindromes\_counter;

}

int\* сreate\_m(int\* P, const size\_t size) {

int\* massive\_m = new int[size];

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

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

massive\_m[i] = P[i] \* i;

}

else {

massive\_m[i] = -P[i];

}

}

return massive\_m;

}

## Решение тестовых примеров на языке С++

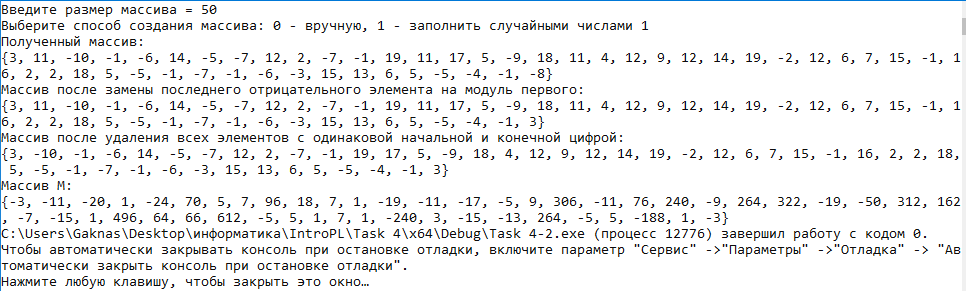


Рисунок - Решение тестовых примеров для задания 4-2 на языке С++