ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ   
«Российский университет транспорта»  
(РУТ (МИИТ))

Институт транспортной техники и систем управления

Кафедра «Управление и защите информации»

отчёт

о практической работе №4

По дисциплине «Процедурное программирование»

Выполнил: ст. гр. ТКИ-111   
Степанов Д. Е.  
Проверил: к. т. н., доцент   
Васильева М. А.

Москва 2021

Оглавление

[Задача 1 3](#_Toc76079436)

[1 Постановка задачи 3](#_Toc76079437)

[1.1 Общее условие 3](#_Toc76079438)

[1.2 Пункты задания 3](#_Toc76079439)

[2 Блок-схема алгоритма 3](#_Toc76079440)

[3 Код программы на языке C++ 8](#_Toc76079441)

[4 Результат работы программы 14](#_Toc76079442)

[Задача 2 15](#_Toc76079443)

[1 Постановка задачи 15](#_Toc76079444)

[1.1 Общее условие 15](#_Toc76079445)

[1.2 Пункты задания 16](#_Toc76079446)

[2 Блок-схема алгоритма 16](#_Toc76079447)

[3 Код программы на языке C++ 21](#_Toc76079448)

[4 Результат работы программы 27](#_Toc76079449)

[Задача 3 28](#_Toc76079450)

[1 Постановка задачи 28](#_Toc76079451)

[1.1 Общее условие 28](#_Toc76079452)

[1.2 Пункты задания 28](#_Toc76079453)

[2 Блок-схема алгоритма 28](#_Toc76079454)

[3 Код программы на языке C++ 33](#_Toc76079455)

[4 Результат работы программы 39](#_Toc76079456)

Задача 1

1 Постановка задачи

* 1. Общее условие

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).

* 1. Пункты задания
     1. Найти сумму элементов, имеющих нечетные индексы.
     2. Подсчитать количество элементов массива, значения которых больше заданного числа А и кратны 5.
     3. Разделить все элементы массива с четными номерами на первый элемент (первый элемент отличен от 0).

1. Блок-схема алгоритма

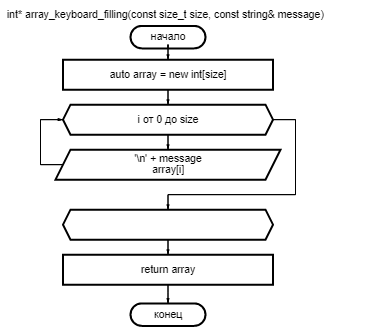


Рисунок 1 ­— array\_keyboard\_filling(size, message)

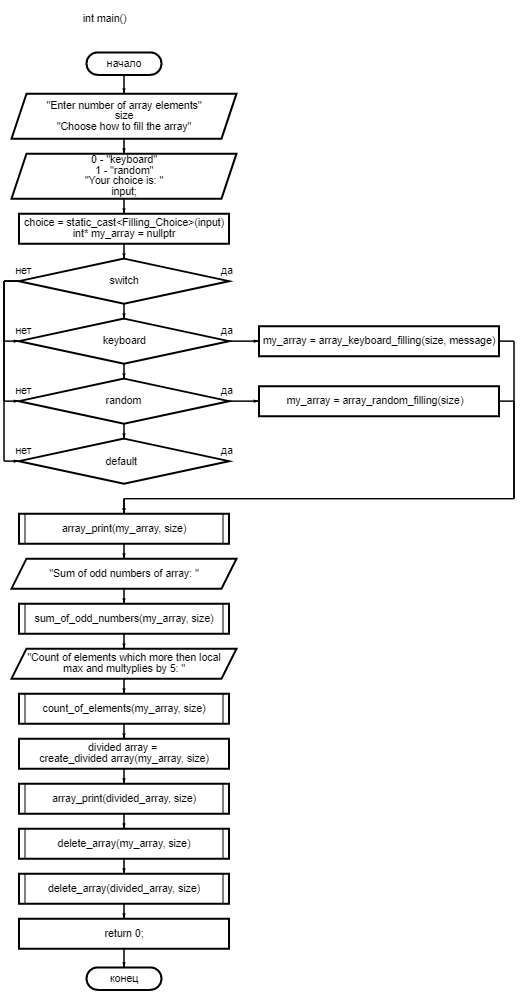


Рисунок 2 — main()

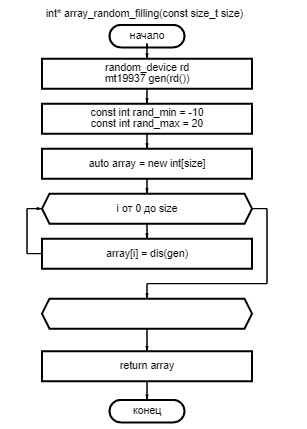


Рисунок 3 — array\_random\_filling(size)

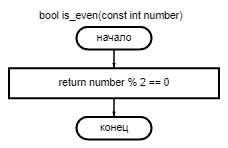


Рисунок 4 — is\_even(number)

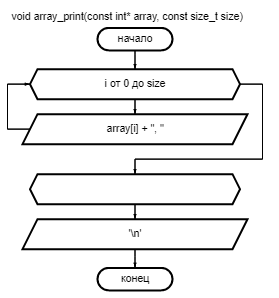


Рисунок 5 — array\_print(array, size)

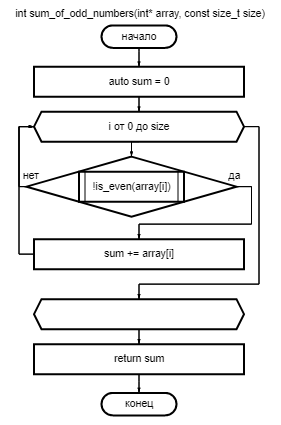


Рисунок 6 — sum\_of\_odd\_numbers(array, size)

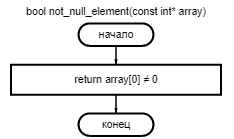


Рисунок 7 — not\_null\_element(array)

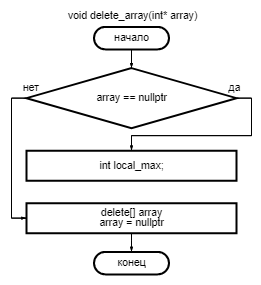


Рисунок 8 — delete\_array(array)

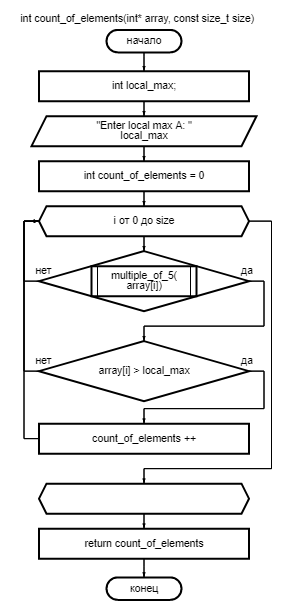


Рисунок 9 — count\_of\_elements(array, size)

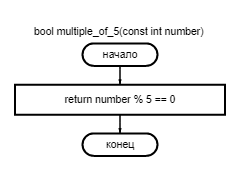


Рисунок 10 — multiple\_of\_5(number)

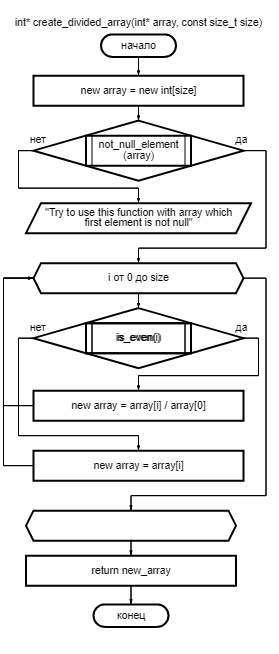


Рисунок 11 — create\_divided\_array(array, size)

1. Код программы на языке C++

#include <iostream>

#include <random>

using namespace std;

/\*\*

\* \brief Filling the array from the keyboard.

\* \param size - array size, message - request for entering a new element.

\* \return Filled array.

\*/

int\* array\_keyboard\_filling(const size\_t size, const string& message);

/\*\*

\* \brief Random filling of an array.

\* \param size - array size.

\* \return Filled array.

\*/

int\* array\_random\_filling(const size\_t size);

/\*\*

\* \brief Prints an array.

\* \param arr - array, size - array size.

\*/

void array\_print(const int\* array, const size\_t size);

/\*\*

\* \brief Сhecks the number for parity .

\* \param number - number.

\* \return True if even else False.

\*/

bool is\_even(const int number);

/\*\*

\* \brief Calculates and prints sum of odd elements of array.

\* \param array - array, size - size of array

\* \return Sum of odd elements.

\*/

int sum\_of\_odd\_numbers(int\* array, const size\_t size);

/\*\*

\* \brief Counts elements of array which more then entered local maximum

\* and multiples of 5.

\* \param array - array, size - size of array.

\* \return Count of elements which more then entered local maximum

\* and multiples of 5.

\*/

int count\_of\_elements(int\* arr, const size\_t size);

/\*\*

\* \brief Checks number for a multiple of 5.

\* \param number - number.

\* \return True if number multiple of 5, else - False.

\*/

bool multiple\_of\_5(const int number);

/\*\*

\* \brief Checks if first element of array is not null

\* \param array - array

\* \return True if 1st element is not null, else - False.

\*/

bool not\_null\_element(const int\* array);

/\*\*

\* \brief Creates new array, which even numbers of elements are divided by first

\* element of the old array.

\* \param array - array, size - size of array.

\* \return New divided array.

\*/

int\* create\_divided\_array(int\* array, const size\_t size);

/\*\*

\* \brief Users choice of array filling.

\* 0 - keyboard, 1 - random.

\*/

enum class Filling\_Choice { keyboard, random };

/\*\*

\* \brief Deletes an array from memory.

\* \param array - array

\*/

void delete\_array(int\* array);

int main()

{

cout << "Enter number of array elements: ";

size\_t size;

cin >> size;

cout << '\n' << "Choose how to fill the array\n"

<< static\_cast<int>(Filling\_Choice::keyboard) << " - keyboard,\n"

<< static\_cast<int>(Filling\_Choice::random) << " - random,\n"

<< "Your choice is ";

int input;

cin >> input;

cout << '\n';

const auto choice = static\_cast<Filling\_Choice>(input);

int\* my\_array = nullptr;

const string message = "Enter the element of the array: ";

switch (choice)

{

case Filling\_Choice::keyboard:

{

my\_array = array\_keyboard\_filling(size, message);

};

case Filling\_Choice::random:

{

my\_array = array\_random\_filling(size);

};

default: {};

};

array\_print(my\_array, size);

cout << "Sum of odd numbers of array: " << sum\_of\_odd\_numbers(my\_array, size) << '\n';

cout << "Count of elements which more then entered local maximum and multiples of 5: " << count\_of\_elements(my\_array, size) << '\n';

auto divided\_array = create\_divided\_array(my\_array, size);

array\_print(divided\_array, size);

delete\_array(my\_array);

delete\_array(divided\_array);

return 0;

}

int\* array\_keyboard\_filling(const size\_t size, const string& message)

{

auto array = new int[size];

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

cout << '\n' << message;

cin >> array[i];

};

return array;

};

int\* array\_random\_filling(const size\_t size)

{

random\_device rd;

mt19937 gen(rd());

const int rand\_min = -10;

const int rand\_max = 20;

uniform\_int\_distribution <> dis(rand\_min, rand\_max);

auto array = new int[size];

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

array[i] = dis(gen);

};

return array;

};

bool is\_even(const int number)

{

return number % 2 == 0;

}

void array\_print(const int\* array, const size\_t size)

{

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

{

cout << array[i] << ", ";

};

cout << '\n';

};

int sum\_of\_odd\_numbers(int\* array, const size\_t size)

{

auto sum = 0;

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

{

if (!is\_even(i))

{

sum += array[i];

}

}

return sum;

};

int count\_of\_elements(int\* array, const size\_t size)

{

int local\_max;

cout << "Enter local maximum A: ";

cin >> local\_max;

int count\_of\_elements = 0;

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

if ((array[i] > local\_max) && (multiple\_of\_5(array[i])))

{

count\_of\_elements++;

}

}

return count\_of\_elements;

};

int\* create\_divided\_array(int\* array, const size\_t size)

{

auto new\_array = new int[size];

if (not\_null\_element(array))

{

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

{

if (is\_even(i))

{

new\_array[i] = array[i] / array[0];

}

else

{

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

}

}

return new\_array;

}

else

{

cout << "Try to use this function with array, which first element is not null." << '\n';

}

}

bool multiple\_of\_5(const int number)

{

return number % 5 == 0;

}

bool not\_null\_element(const int\* array)

{

return array[0] != 0;

}

void delete\_array(int\* array)

{

if (array == nullptr)

return;

delete[] array;

array = nullptr;

}

1. Результат работы программы

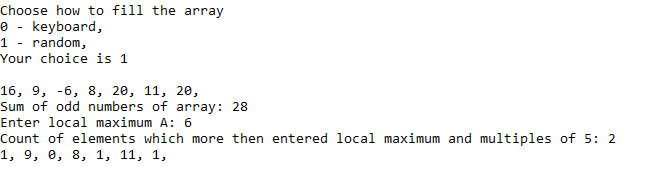


Рисунок 12

Задача 2

1. Постановка задачи
   1. Общее условие

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.

* 1. Пункты задания
     1. Заменить последний отрицательный элемент массива предпоследним элементом массива.
     2. Удалить из массива все элементы, первая цифра которых четная.
     3. Из элементов массива сформировать массив той же размерности по правилу: если номер четный, то значение элемента находится по формуле:  
        а если нечетный, то по формуле:

1. Блок-схема алгоритма

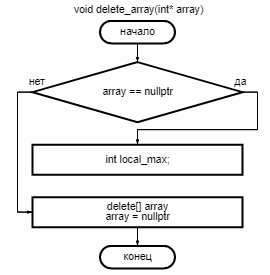


Рисунок 13 — delete\_array(array)

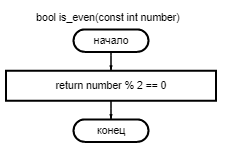


Рисунок 14 — is\_even(number)

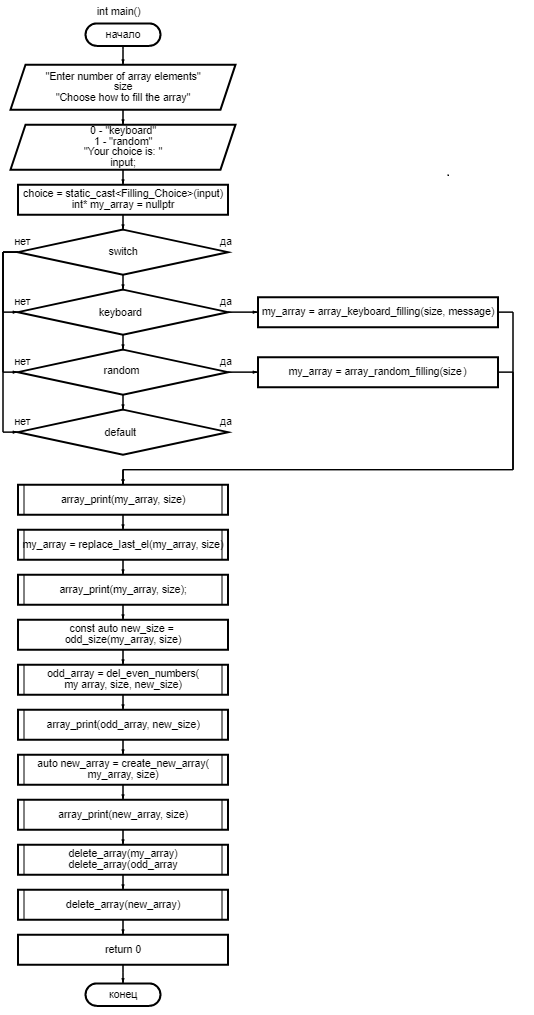


Рисунок 15 — main()

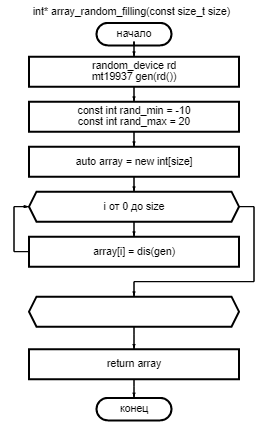


Рисунок 16 — array\_random\_filling(size)

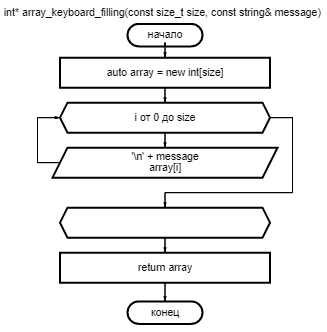


Рисунок 17 — array\_keyboard\_filling(size, message)

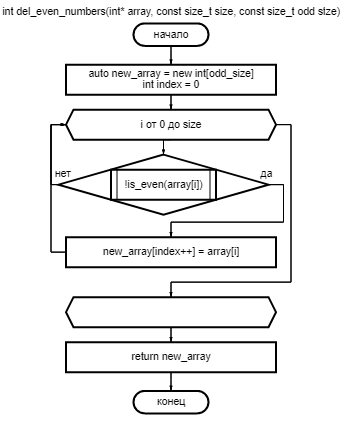


Рисунок 18 — del\_even\_numbers(array, size, odd\_size)

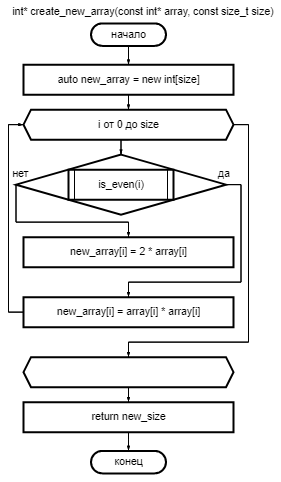


Рисунок 19 — create\_new\_array(array, size)

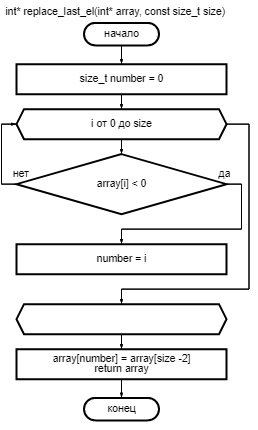


Рисунок 20 — replace\_last\_el(array, size)

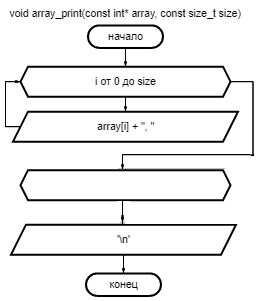


Рисунок 21 — array\_print(array, size)

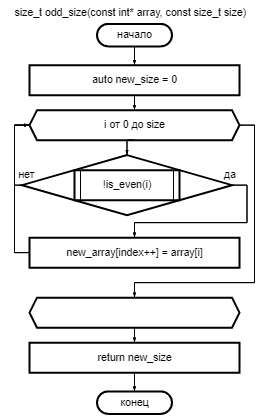


Рисунок 22 — odd\_size(array, size)

1. Код программы на языке C++

#include <iostream>

#include <random>

using namespace std;

/\*\*

\* \brief Filling the array from the keyboard.

\* \param size - array size, message - request for entering a new element.

\* \return Filled array.

\*/

int\* array\_keyboard\_filling(const size\_t size, const string& message);

/\*\*

\* \brief Random filling of an array.

\* \param size - array size.

\* \return Filled array.

\*/

int\* array\_random\_filling(const size\_t size);

/\*\*

\* \brief Replaces the last negative element of the array with the penultimate element.

\* \param arr - array, size - array size.

\* \return Filled array.

\*/

int\* replace\_last\_el(int\* array, const size\_t size);

/\*\*

\* \brief Deletes even elements from the array.

\* \param arr - array, size - array size, odd\_size - size of array without even elements.

\* \return New array without even numbers.

\*/

int\* del\_even\_numbers(int\* array, const size\_t size, const size\_t odd\_size);

/\*\*

\* \brief Creates a new array based on the old array according to rule:

\* if the number of the element of the old array is even - the element

\* of new array is equal to the square of the element of the old one,

\* else - to the doubled element of the old one.

\* \param arr - array, num - array size.

\* \return New array.

\*/

int\* create\_new\_array(const int\* array, const size\_t size);

/\*\*

\* \brief Prints an array.

\* \param arr - array, size - array size, odd\_size - size of array without even elements.

\*/

void array\_print(const int\* array, const size\_t size);

/\*\*

\* \brief Сhecks the number for parity .

\* \param number - number.

\* \return True if even else False.

\*/

bool is\_even(const int number);

/\*\*

\* \brief Counts odd elements of the array.

\* \param arr - array, size - size.

\* \return Count of odd elements.

\*/

size\_t odd\_size(const int\* array, const size\_t size);

/\*\*

\* \brief Users choice of array filling.

\* 0 - keyboard, 1 - random.

\*/

enum class Filling\_Choice { keyboard, random };

/\*\*

\* \brief Deletes an array from memory.

\* \param array - array

\*/

void delete\_array(int\* array);

int main()

{

cout << "Enter number of array elements: ";

size\_t size;

cin >> size;

cout << '\n' << "Choose how to fill the array\n"

<< static\_cast<int>(Filling\_Choice::keyboard) << " - keyboard,\n"

<< static\_cast<int>(Filling\_Choice::random) << " - random,\n"

<< "Your choice is ";

int input;

cin >> input;

cout << '\n';

const auto choice = static\_cast<Filling\_Choice>(input);

int\* my\_array = nullptr;

const string message = "Enter the element of the array";

switch (choice)

{

case Filling\_Choice::keyboard:

{

my\_array = array\_keyboard\_filling(size, message);

break;

};

case Filling\_Choice::random:

{

my\_array = array\_random\_filling(size);

break;

};

default: {};

}

array\_print(my\_array, size);

my\_array = replace\_last\_el(my\_array, size);

array\_print(my\_array, size);

const auto new\_size = odd\_size(my\_array, size);

auto odd\_array = del\_even\_numbers(my\_array, size, new\_size);

array\_print(odd\_array, new\_size);

delete\_array(odd\_array);

auto new\_array = create\_new\_array(my\_array, size);

array\_print(new\_array, size);

delete\_array(new\_array);

delete\_array(my\_array);

return 0;

}

int\* array\_keyboard\_filling(const size\_t size, const string& message)

{

auto array = new int[size];

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

cout << '\n' << message;

cin >> array[i];

};

return array;

};

int\* array\_random\_filling(const size\_t size)

{

random\_device rd;

mt19937 gen(rd());

const int rand\_min = -10;

const int rand\_max = 20;

uniform\_int\_distribution <> dis(rand\_min, rand\_max);

auto array = new int[size];

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

array[i] = dis(gen);

};

return array;

};

int\* replace\_last\_el(int\* array, const size\_t size)

{

size\_t number = 0;

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

{

if (array[i] < 0)

{

number = i;

}

}

array[number] = array[size - 2];

return array;

};

int\* del\_even\_numbers(int\* array, const size\_t size, const size\_t odd\_size) {

auto new\_array = new int[odd\_size];

int index = 0;

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

{

if (!is\_even(array[i]))

{

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

}

}

return new\_array;

}

bool is\_even(const int number)

{

return number % 2 == 0;

}

size\_t odd\_size(const int\* array, const size\_t size)

{

auto new\_size = 0;

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

{

if (!is\_even(array[i]))

{

new\_size++;

}

}

return new\_size;

}

int\* create\_new\_array(const int\* array, const size\_t size)

{

auto new\_array = new int[size];

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

{

if (is\_even(i))

{

new\_array[i] = array[i] \* array[i];

}

else

{

new\_array[i] = 2 \* array[i];

}

}

return new\_array;

}

void array\_print(const int\* array, const size\_t size)

{

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

{

cout << array[i] << ", ";

};

cout << '\n';

};

void delete\_array(int\* array)

{

if (array == nullptr)

return;

delete[] array;

array = nullptr;

}

1. Результат работы программы

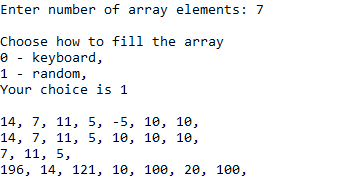


Рисунок 23

Задача 3

1. Постановка задачи
   1. Общее условие

Создать многомерный массив из 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.

* 1. Пункты задания
     1. Заменить все элементы первых трех столбцов на их квадраты.
     2. Вставить после каждой нечетной строки первую строку.

1. Блок-схема алгоритма

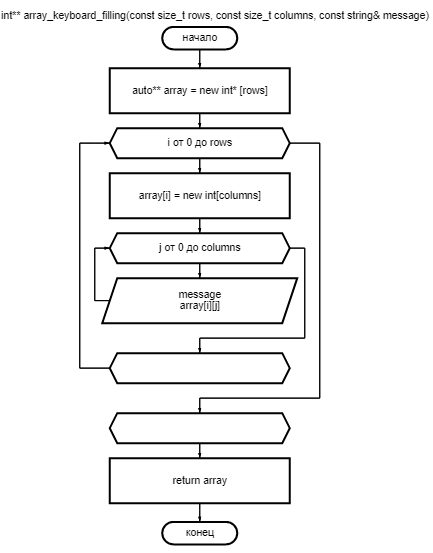


Рисунок 24 — array\_keyboard\_filling(rows, columns, message)

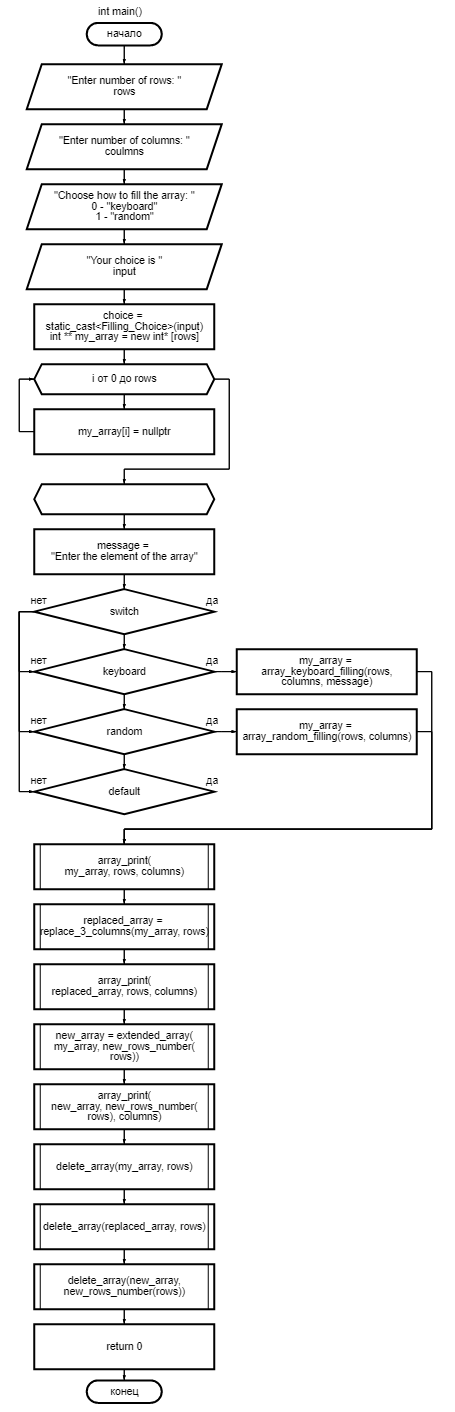


Рисунок 25 — main()

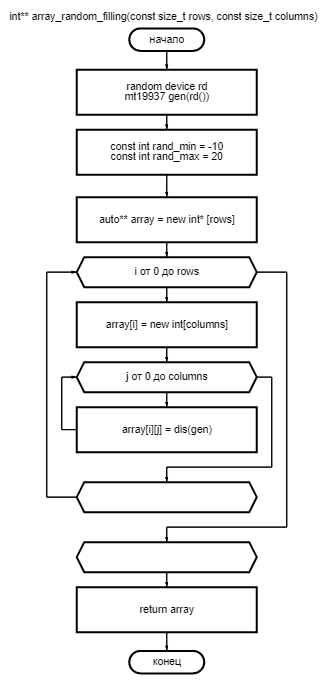


Рисунок 26 — array\_random\_filling(rows, columns)

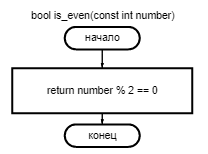


Рисунок 27 — is\_even(number)

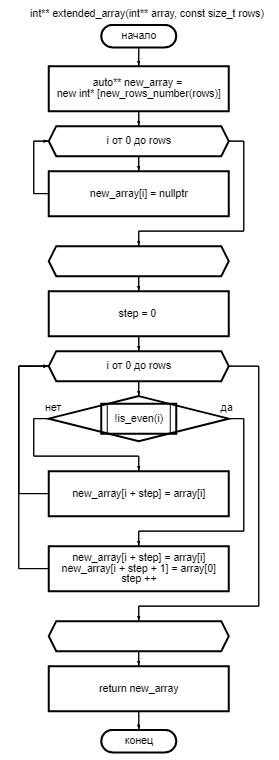


Рисунок 28 — extended\_array(array, rows)

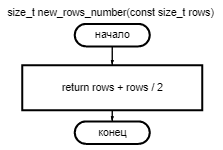


Рисунок 29 — new\_rows\_number(rows)

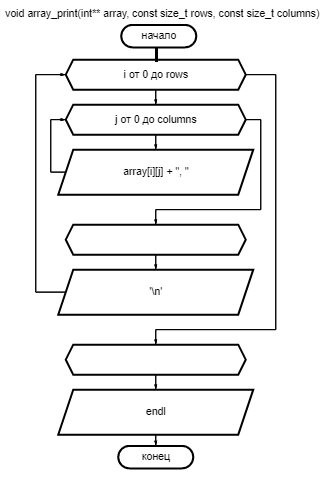


Рисунок 30 — array\_print(array, rows, columns)

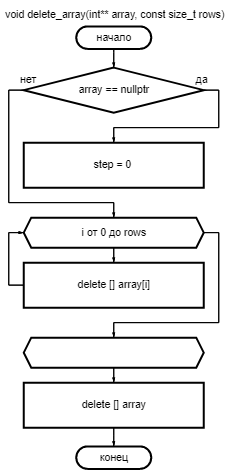


Рисунок 31 — delete\_array(array, rows)

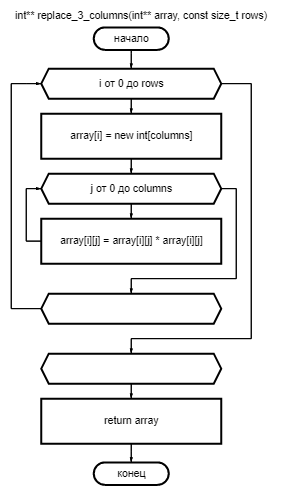


Рисунок 32 — replace\_3\_columns(array, rows)

1. Код программы на языке C++

#include <iostream>

#include <random>

using namespace std;

/\*\*

\* /brief Fills array from keyboard.

\* /params rows - rows, columns - columns, message - request for entering a new element.

\* /return Filled array.

\*/

int\*\* array\_keyboard\_filling(const size\_t rows, const size\_t columns, const string& message);

/\*\*

\* /brief Fills array with random numbers.

\* /params rows - rows, columns - columns;

\* /return Filled array.

\*/

int\*\* array\_random\_filling(const size\_t rows, const size\_t columns);

/\*\*

\* /brief Fills array with random numbers.

\* /params rows - rows.

\* /return Filled array.

\*/

int\*\* replace\_3\_columns(int\*\* array, const size\_t rows);

/\*\*

\* \brief Сhecks the number for parity.

\* \param number - number.

\* \return True if even else False.

\*/

bool is\_even(const int number);

/\*\*

\* \brief Calulates count of rows, if after each odd row insert dirst row.

\* \param rows - count of rows in old array.

\* \return New count of rows.

\*/

size\_t new\_rows\_number(const size\_t rows);

/\*\*

\* \brief Expands the array by inserting the first row after each odd row.

\* \param array - old array, rows - number of rows of old array.

\* \return New extended array.

\*/

int\*\* extended\_array(int\*\* array, const size\_t rows);

/\*\*

\* \brief Users choice of array filling.

\* 0 - keyboard, 1 - random.

\*/

enum class Filling\_Choice { keyboard, random };

/\*\*

\* \brief Prints an array.

\* \param array - array, rows - number of rows, columns - number of columns.

\*/

void array\_print(int\*\* array, const size\_t rows, const size\_t columns);

/\*\*

\* \brief Deletes an array.

\* \param array - array, rows - number of rows, columns - number of columns.

\*/

void delete\_array(int\*\* array, const size\_t rows);

int main()

{

cout << "Enter number of rows: ";

size\_t rows;

cin >> rows;

cout << "Enter number of columns: ";

size\_t columns;

cin >> columns;

cout << '\n' << "Choose how to fill the array\n"

<< static\_cast<int>(Filling\_Choice::keyboard) << " - keyboard,\n"

<< static\_cast<int>(Filling\_Choice::random) << " - random,\n"

<< "Your choice is ";

int input;

cin >> input;

cout << '\n';

const auto choice = static\_cast<Filling\_Choice>(input);

int\*\* my\_array = new int\* [rows];

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

{

my\_array[i] = nullptr;

}

const string message = "Enter the element of the array";

switch (choice)

{

case Filling\_Choice::keyboard:

{

my\_array = array\_keyboard\_filling(rows, columns, message);

};

case Filling\_Choice::random:

{

my\_array = array\_random\_filling(rows, columns);

};

default: {};

};

array\_print(my\_array, rows, columns);

auto replaced\_array = replace\_3\_columns(my\_array, rows);

array\_print(replaced\_array, rows, columns);

auto new\_array = extended\_array(my\_array, rows);

array\_print(new\_array, new\_rows\_number(rows), columns);

delete\_array(my\_array, rows);

delete\_array(replaced\_array, rows);

delete\_array(new\_array, new\_rows\_number(rows));

return 0;

}

int\*\* array\_keyboard\_filling(const size\_t rows, const size\_t columns, const string& message)

{

auto\*\* array = new int\* [rows];

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

{

array[i] = new int[columns];

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

{

cout << message;

cin >> array[i][j];

}

}

return array;

}

int\*\* array\_random\_filling(const size\_t rows, const size\_t columns)

{

random\_device rd;

mt19937 gen(rd());

const int rand\_min = -10;

const int rand\_max = 20;

uniform\_int\_distribution <> dis(rand\_min, rand\_max);

auto\*\* array = new int\* [rows];

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

{

array[i] = new int[columns];

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

{

array[i][j] = dis(gen);

};

};

return array;

}

bool is\_even(const int number)

{

return number % 2 == 0;

}

int\*\* replace\_3\_columns(int\*\* array, const size\_t rows)

{

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

{

for (size\_t j = 0; j < 3; j++)

{

array[i][j] = array[i][j] \* array[i][j];

}

}

return array;

}

size\_t new\_rows\_number(const size\_t rows)

{

return rows + rows / 2;

}

int\*\* extended\_array(int\*\* array, const size\_t rows)

{

auto\*\* new\_array = new int\* [new\_rows\_number(rows)];

for (size\_t i = 0; i < new\_rows\_number(rows); i++)

{

new\_array[i] = nullptr;

};

auto step = 0;

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

{

if (!is\_even(i))

{

new\_array[i + step] = array[i];

new\_array[i + step + 1] = array[0];

step++;

}

else

{

new\_array[i + step] = array[i];

}

}

return new\_array;

}

void array\_print(int\*\* array, const size\_t rows, const size\_t columns)

{

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

{

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

{

cout << array[i][j] << ',';

};

cout << '\n';

};

cout << endl;

}

void delete\_array(int\*\* array, const size\_t rows)

{

if (array == nullptr)

return;

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

delete[] array[i];

delete[] array;

}

1. Результат работы программы

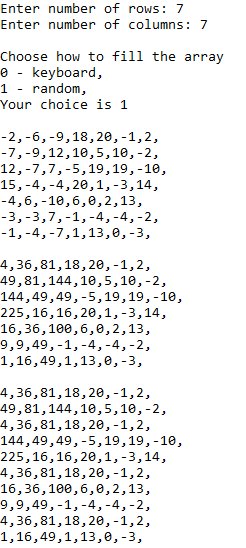


Рисунок 33