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 №4.2

По дисциплине «Процедурное программирование»

Выполнил: ст. гр. ТКИ-111   
Чевычалов А. В.  
Проверил: к. т. н., доцент   
Васильева М. А.

Москва 2021

**1 Постановка задачи**

**1.1 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).

**1.2 Пункты задания**

**1.2.1 Найти сумму элементов, имеющих нечетные индексы.**

**1.2.2 Подсчитать количество элементов массива, значения которых больше заданного числа А и кратны 5.**

**1.2.3 Разделить все элементы массива с четными номерами на первый элемент (первый элемент отличен от 0).**

**2 Блок-схема алгоритма**

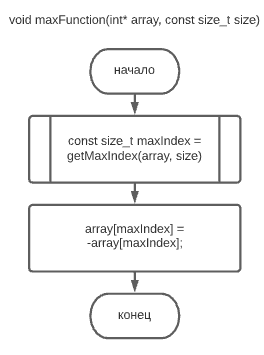
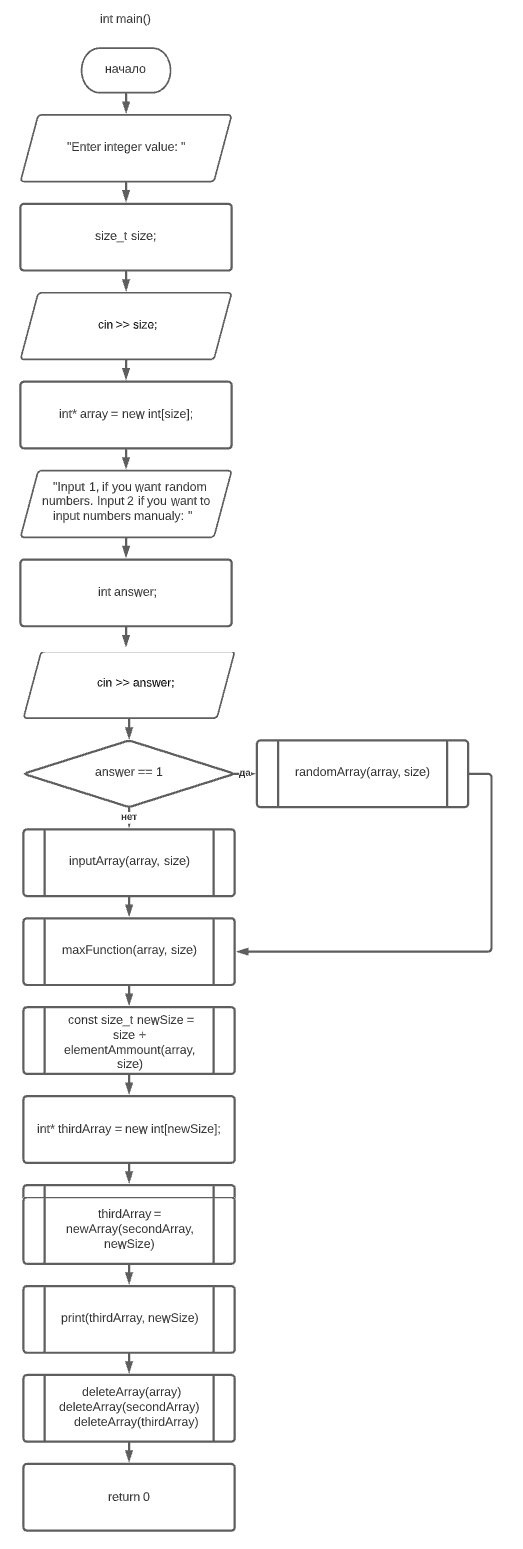
****

Рисунок 1

Рисунок 2

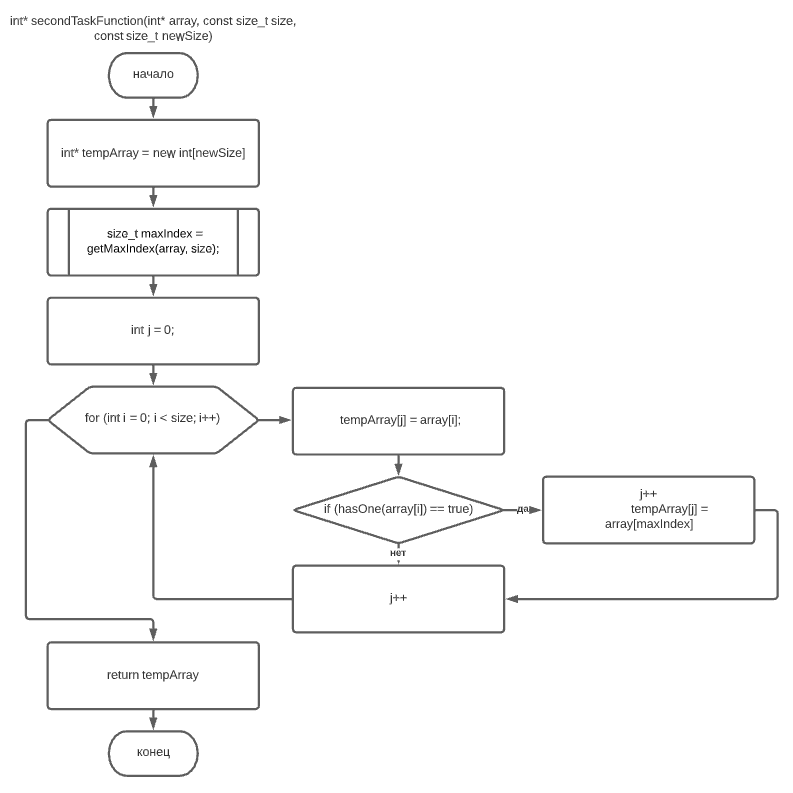
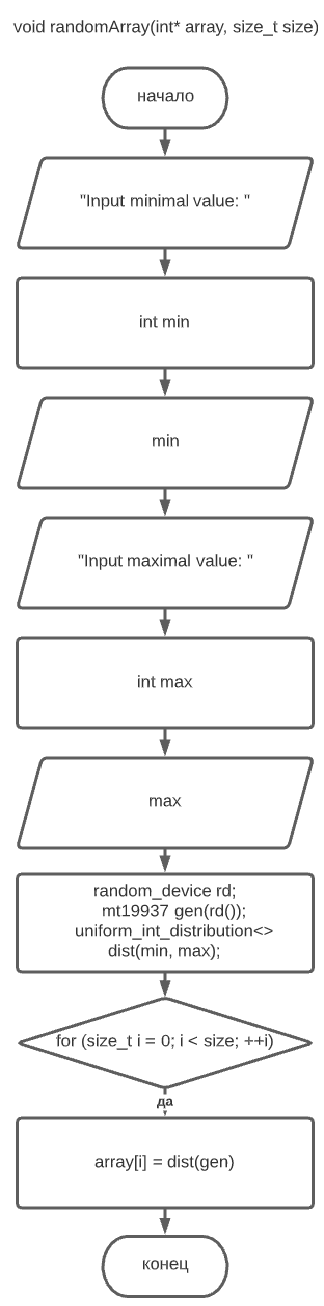
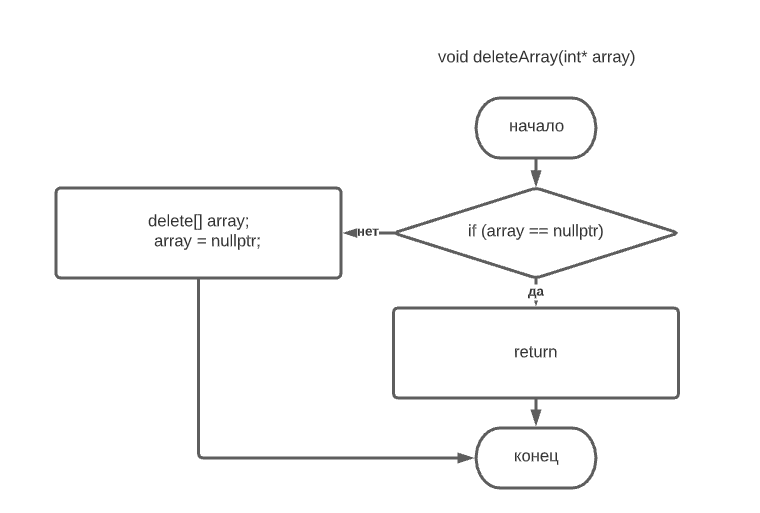
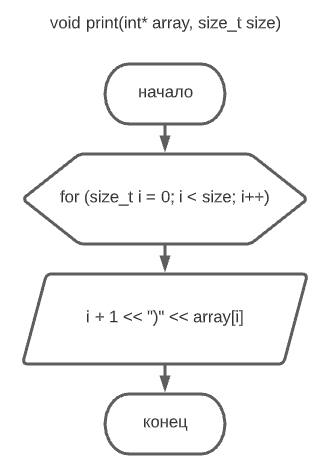
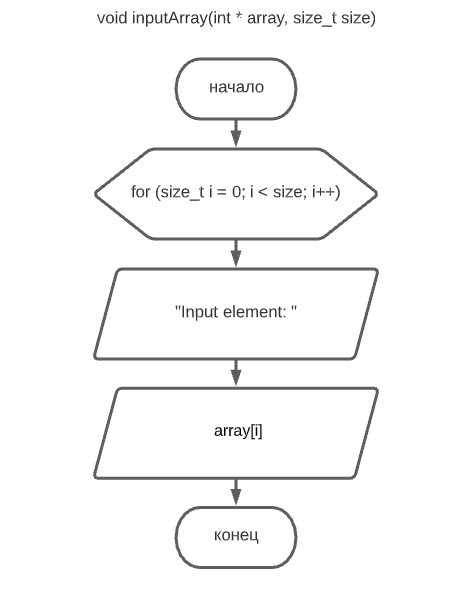
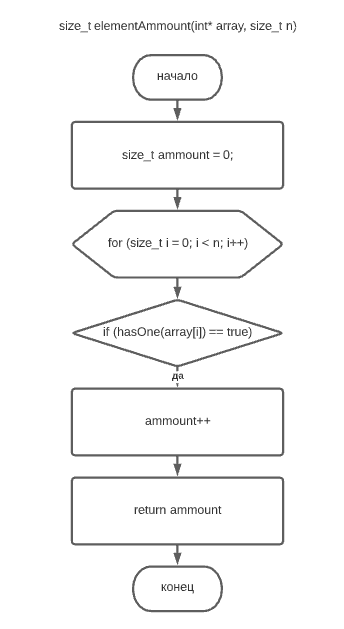
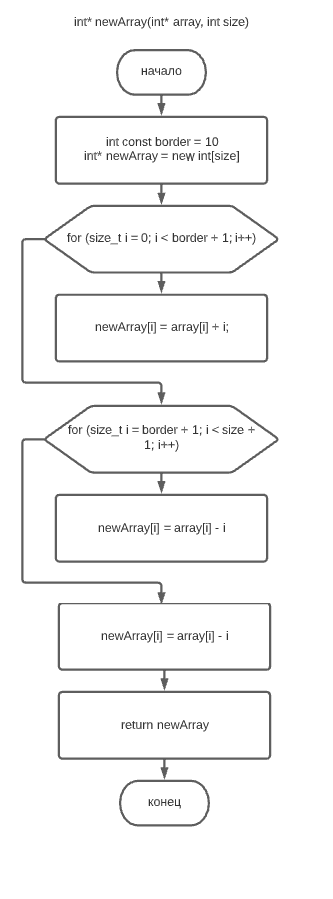
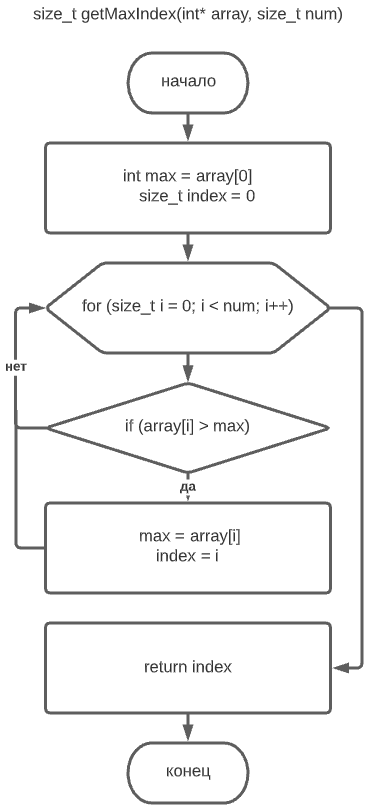


Рисунок 3



**3. Код программы на языке С++**

#include <iostream>

#include <random>

using namespace std;

/\*\*

\* \brief Функция заполнения массива случайными числами из заданного интервала

\* \param array данный массив

\* \param size размер массива

\* \param min минимальное случайное число

\* \param max максимальное случайное число

\*/

void randomArray(int\* array, size\_t size);

/\*\*

\* \brief Функция заполнения массива из ввода с клавиатуры

\* \param array данный массив

\* \param size размер массива

\*/

void inputArray(int\* array, size\_t size);

/\*\*

\* \brief Функция вывода массива на экран

\* \param array данный массив

\* \param size размер массива

\*/

void print(int\* array, size\_t size);

/\*\*

\* \brief Функция, показывающая, есть ли в числе цифра 1

\* \param value данное число

\*/

bool hasOne(int value);

/\*\*

\* \brief Функция замены максимального элемента массива на противоположный по знаку

\* \param array Введённый массив

\* \param size Количество элементов в массиве

\*/

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

/\*\*

\* \brief Функция нахождения массива в котором после всех элементов, в которых есть цифра 1, вставлен максимальный элемент

\* \param array Введённый массив

\* \param size Количество элементов в начальном массиве

\* \param newSize Количество элементов в конечном массиве

\* \return Возвращает необходимый массив

\*/

int\* secondTaskFunction(int\* array, const size\_t size, const size\_t newSize);

/\*\*

\* \brief Функция нахождения индекса максимума массива

\* \param array Введённый массив

\* \param size Количество элементов в массиве

\* \return Возвращает индекс максимума массива

\*/

size\_t getMaxIndex(int\* array, size\_t size);

/\*\*

\* \brief Функция формирования нового массива, сформированного по правилу из 3 задачи

\* \param array Введённый массив

\* \param size Количество элементов в массиве

\* \return Возвращает новый массив

\*/

int\* newArray(int\* array, size\_t size);

/\*\*

\* \brief Функция нахождения количества элементов, в которых есть цифра 1

\* \param array Введённый массив

\* \param size Количество элементов в массиве

\* \return Возвращает количество элементов, в которых есть цифра 1

\*/

size\_t elementAmmount(int\* array, size\_t size);

/\*\*

\* \brief Функция удаления массива с проверкой на nullptr

\* \param array Массив

\*/

void deleteArray(int\* array);

/\*\*

\* \brief Точка входа в программу

\* \return Возвращает нуль, в случае успеха

\*/

int main()

{

cout << "Enter integer value: ";

size\_t size;

cin >> size;

int\* array = new int[size];

cout << "Input 1, if you want random numbers. Input 2 if you want to input numbers manualy: ";

int answer;

cin >> answer;

if (answer == 1)

{

randomArray(array, size);

}

else if (answer == 2)

{

inputArray(array, size);

}

maxFunction(array, size);

const size\_t newSize = size + elementAmmount(array, size);

int\* secondArray = new int[newSize];

secondArray = secondTaskFunction(array, size, newSize);

int\* thirdArray = new int[newSize];

thirdArray = newArray(secondArray, newSize);

print(thirdArray, newSize);

deleteArray(array);

deleteArray(secondArray);

deleteArray(thirdArray);

return 0;

}

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

{

const size\_t maxIndex = getMaxIndex(array, size);

array[maxIndex] = -array[maxIndex];

}

int\* secondTaskFunction(int\* array, const size\_t size, const size\_t newSize)

{

int\* tempArray = new int[newSize];

size\_t maxIndex = getMaxIndex(array, size);

int j = 0;

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

{

tempArray[j] = array[i];

if (hasOne(array[i]) == true)

{

j++;

tempArray[j] = array[maxIndex];

}

j++;

}

return tempArray;

}

size\_t getMaxIndex(int\* array, size\_t num)

{

int max = array[0];

size\_t index = 0;

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

{

if (array[i] > max)

{

max = array[i];

index = i;

}

}

return index;

}

int\* newArray(int\* array, size\_t size)

{

int const border = 10;

int\* newArray = new int[size];

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

{

newArray[i] = array[i] + i;

}

for (size\_t i = border + 1; i < size + 1; i++)

{

newArray[i] = array[i] - i;

}

return newArray;

}

size\_t elementAmmount(int\* array, size\_t n)

{

size\_t ammount = 0;

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

{

if (hasOne(array[i]) == true)

ammount++;

}

return ammount;

}

void randomArray(int\* array, size\_t size)

{

cout << "Input minimal value: ";

int min;

cin >> min;

cout << "Input maximal value: ";

int max;

cin >> max;

random\_device rd;

mt19937 gen(rd());

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

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

{

array[i] = dist(gen);

}

}

void inputArray(int \* array, size\_t size)

{

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

{

cout << "Input element: ";

cin >> array[i];

}

}

void print(int\* array, size\_t size)

{

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

{

cout << i + 1 << ")" << array[i] << "\n";

}

}

bool hasOne(int value)

{

value = abs(value);

int index = 0;

while (value > 0)

{

if (value % 10 == 1)

{

index = 1;

}

value = value / 10;

}

if (index == 1)

return true;

else

return false;

}

void deleteArray(int\* array)

{

if (array == nullptr)

return

delete[] array;

array = nullptr;

}

**4 Результат работы программы**

