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 №4.3

По дисциплине «Процедурное программирование»

Выполнил: ст. гр. ТКИ – 112

Жеребятин Илья

Проверил: к.т.н., доц.

Васильева М.А.

Москва 2021

**Цель работы**

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.

Задачи:

1. Заменить максимальный элемент каждой строки нулем.
2. Вставить перед всеми строками, первый элемент, которых делится на 3, строку из нулей.

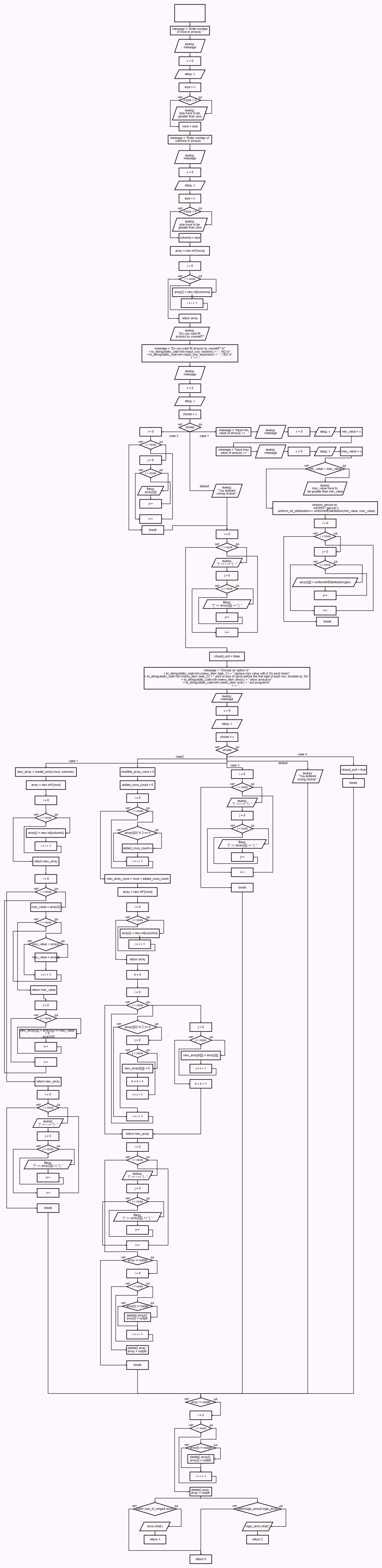
**Проектирование алгоритма**

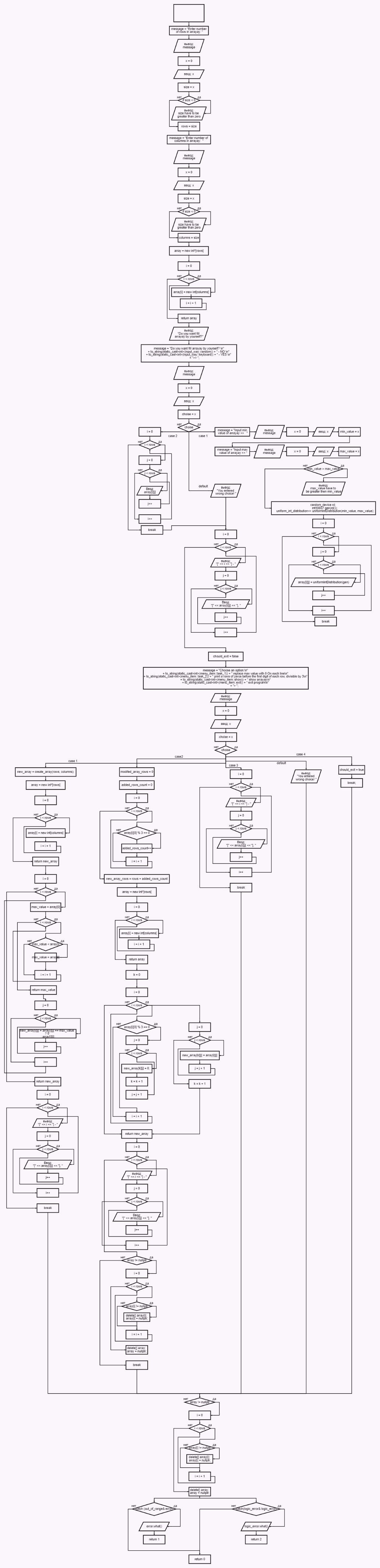
Так как пункты задания по условию требуется организовать в виде функций, следовательно, в виде функций необходимо организовать также и некоторые базовые операции, такие как ввод и вывод массива. Методы — это функции, которые могут выполнять действия над данными, реализуя возможность заполнения массива случайными числами и с помощью клавиатуры.

**Написание алгоритма**

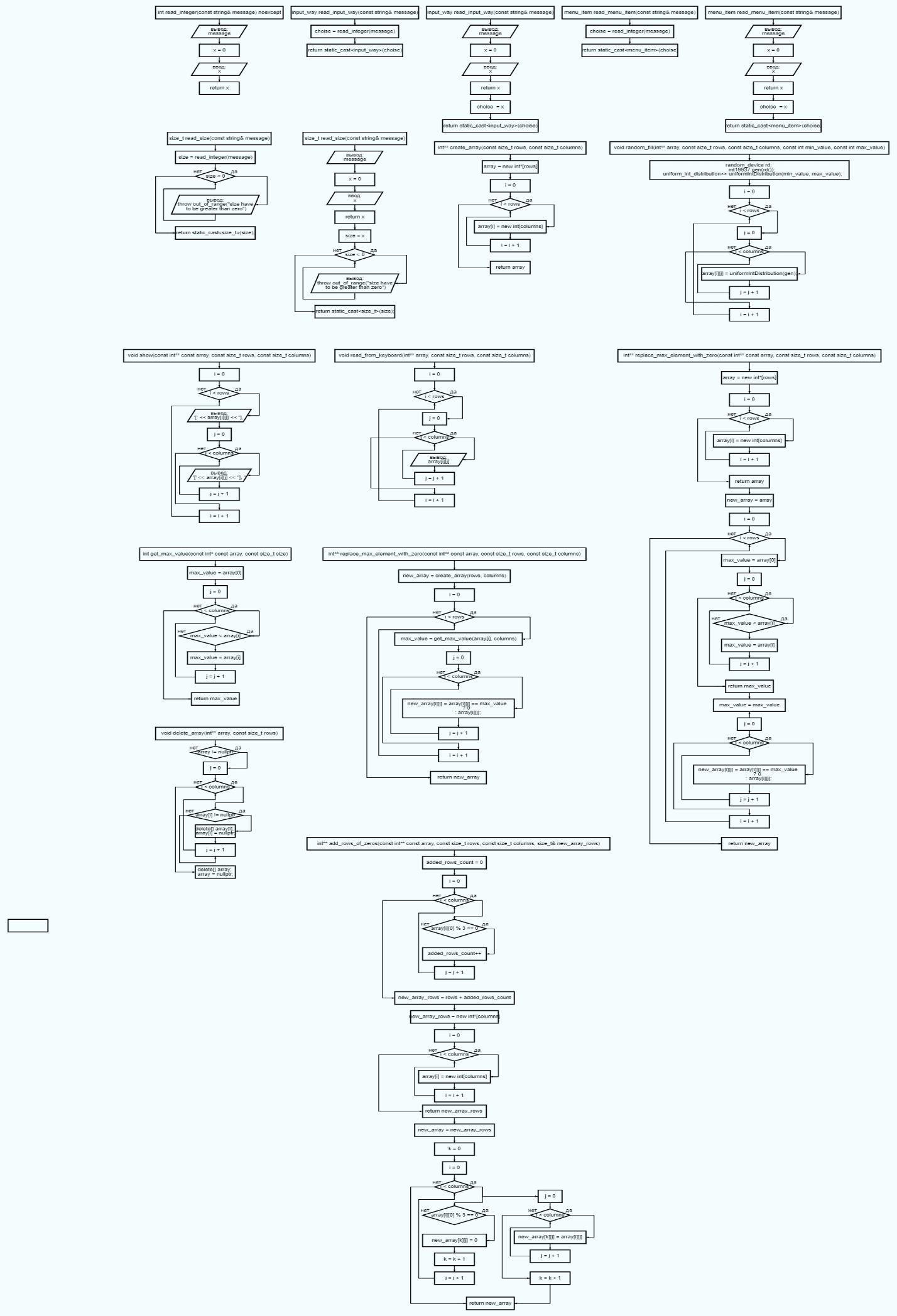
|  |
| --- |
| #include <iostream>  #include <random>  #include <string>    using namespace std;  /\*\*  \* \brief Способ заполнения массива.  \*/  enum class input\_way  {  /\*\*  \* \brief Заполнение массива при помощи ГПСЧ.  \*/  random,    /\*\*  \* \brief Заполнение массива при помощи пользовательского ввода.  \*/  keyboard  };    /\*\*  \* \brief Меню.  \*/  enum class menu\_item  {  /\*\*  \* \brief Показывает массив.  \*/  show,  /\*\*  \* \brief Заменить максимальный элемент массив на 0.  \*/  task\_1,  /\*\*  \* \brief Добавить ряд нулей перед строкой, если первый элемент кратен 3.  \*/  task\_2,  /\*\*  \* \brief Завершает программу.  \*/  exit  };    /\*\*  \* \brief Функция позволяет ввести число предверяя его сообщением \a message.  \* \param message Передаваемое сообщение.  \* \return Введёное пользователем число.  \*/  int read\_integer(const string& message = "") noexcept;    /\*\*  \* \brief Считывает способ запонения массива.  \* \param Мотивирующее сообщение.  \* \return Способ заполнения массива.  \*/  input\_way read\_input\_way(const string& message = "");    /\*\*  \* \brief Считывает пункт меню.  \* \param Мотивирующее сообщение.  \* \return Пункт меню.  \*/  menu\_item read\_menu\_item(const string& message = "");    /\*\*  \* \brief Считывает оазмер по одному из измерений массива.  \* \param Мотивирующее сообщение.  \* \return Способ заполнения массива.  \* \exception out\_of\_range В случае если введён размер меньше или раен нулю.  \*/  size\_t read\_size(const string& message = "");    /\*\*  \* \brief Создаёт двумерный массив.  \* \param rows Количество строк.  \* \param columns Количество столбцов.  \* \return Созданный массив.  \*/  int\*\* create\_array(const size\_t rows, const size\_t columns);    /\*\*  \* \brief Функция заполняет массив случайными числами в заданном промежутке. Возможны перебои при работе с большими массивами.  \* \param array Заполняемый массив.  \* \param rows Количество строк.  \* \param columns Количество столбцов.  \* \param min\_value Левая граница распределения.  \* \param max\_value Правая граница распределения.  \*/  void random\_fill(int\*\* array, const size\_t rows, const size\_t columns, const int min\_value, const int max\_value);    /\*\*  \* \brief Функция позволяет пользователю вручную заполнить массив.  \* \param rows Количество строк.  \* \param columns Количество столбцов.  \*/  void read\_from\_keyboard(int\*\* array, const size\_t rows, const size\_t columns);    /\*\*  \* \brief Функция выводит на экран данный ей массив.  \* \param array Отображаемый массив.  \* \param rows Количество строк.  \* \param columns Количество столбцов.  \*/  void show(const int\*\* const array, const size\_t rows, const size\_t columns);    /\*\*  \* \brief Функция меняет наибольший(наибольшие) элементы на ноль.  \* \param array Исходный массив.  \* \param rows Количество строк.  \* \param columns Количество столбцов.  \* \return Изменённый массив  \*/  int\*\* replace\_max\_element\_with\_zero(const int\*\* const array, const size\_t rows, const size\_t columns);    /\*\*  \* \brief Получает значение максимального элемента \a array.  \* \param Массив элементов.  \* \param Размер массива.  \* \return Максимальный элемент.  \*/  int get\_max\_value(const int\* const array, const size\_t size);    /\*\*  \* \brief Функция добавляет ряд нулей перед строкой, если первый элемент кратен 3.  \* \param array Исходный массив.  \* \param rows Количество строк.  \* \param columns Количество столбцов.  \* \param new\_array\_rows out параметр, хранящий число строк изменённого массива.  \* \return Изменённый элемент.  \*/  int\*\* add\_rows\_of\_zeros(const int\*\* const array, const size\_t rows, const size\_t columns, size\_t& new\_array\_rows);    /\*\*  \* \brief Функция освобождает динамическую память массива  \* \param array Массив для "зачистки"  \* \param rows Количество строк массива  \*/  void delete\_array(int\*\* array, const size\_t rows);    /\*\*  Главная функция программы  0 в случае успеха  \*/  int main()  {  try  {  const auto rows = read\_size("Enter number of rows in arrayay: ");  const auto columns = read\_size("Enter number of columns in arrayay: ");    auto array = create\_array(rows, columns);    auto message = "Do you want fill arrayay by yourself? \n"  + to\_string(static\_cast<int>(input\_way::random)) + " - NO \n"  + to\_string(static\_cast<int>(input\_way::keyboard)) + " - YES \n"  + ">> ";    const auto choise = read\_input\_way(message);  switch (choise)  {  case input\_way::random:  {  const auto min\_value = read\_integer("Input min value of arrayay >> ");  const auto max\_value = read\_integer("Input max value of arrayay >> ");  if (min\_value > max\_value)  {  throw logic\_error("max\_value have to be greater then min\_value");  }    random\_fill(array, rows, columns, min\_value, max\_value);  break;  }  case input\_way::keyboard:  {  read\_from\_keyboard(array, rows, columns);  break;  }  default :  {  throw out\_of\_range("You entered wrong choice!");  }  }    show(const\_cast<const int\*\* const>(array), rows, columns);    bool chould\_exit = false;  do  {  auto message = "Choose an option:\n"  + to\_string(static\_cast<int>(menu\_item::task\_1)) + " replace max value with 0 On each line\n"  + to\_string(static\_cast<int>(menu\_item::task\_2)) + " print a rows of zeros before the first digit of each row, divisible by 3\n"  + to\_string(static\_cast<int>(menu\_item::show)) + " show arrayay\n"  + to\_string(static\_cast<int>(menu\_item::exit)) + " exit program\n"  + "> ";    const auto choise = read\_menu\_item(message);  switch (choise)  {  case menu\_item::exit:  {  chould\_exit = true;  break;  }  case menu\_item::task\_1:  {  const auto new\_array = replace\_max\_element\_with\_zero(const\_cast<const int\*\* const>(array), rows, columns);  show(const\_cast<const int\*\* const>(new\_array), rows, columns);  delete\_array(new\_array, rows);  break;  }  case menu\_item::task\_2:  {  size\_t modifed\_array\_rows = 0;  auto modifed\_array = add\_rows\_of\_zeros(const\_cast<const int\*\* const>(array), rows, columns, modifed\_array\_rows);  show(const\_cast<const int\*\* const>(modifed\_array), modifed\_array\_rows, columns);  delete\_array(modifed\_array, rows);  break;  }  case menu\_item::show:  {  show(const\_cast<const int\*\* const>(array), rows, columns);  break;  }  default:  {  throw out\_of\_range("Not supposed input!");  }  }  } while (!chould\_exit);    delete\_array(array, rows);  }  catch (out\_of\_range& error)  {  cerr << error.what() << endl;  return 1;  }  catch(logic\_error& logic\_error)  {  cerr << logic\_error.what() << endl;  return 2;  }    return 0;  }    int read\_integer(const string& message) noexcept  {  cout << message;  int x = 0;  cin >> x;  return x;  }    input\_way read\_input\_way(const string& message)  {  const auto choise = read\_integer(message);  return static\_cast<input\_way>(choise);  }    menu\_item read\_menu\_item(const string& message)  {  const auto choise = read\_integer(message);  return static\_cast<menu\_item>(choise);  }    size\_t read\_size(const string& message)  {  const auto size = read\_integer(message);  if (size < 0)  {  throw out\_of\_range("size have to be greater than zero");  }  return static\_cast<size\_t>(size);  }    int\*\* create\_array(const size\_t rows, const size\_t columns)  {  auto array = new int\*[rows];  for(size\_t i = 0; i < rows; i++)  {  array[i] = new int[columns];  }    return array;  }    void random\_fill(int\*\* array, const size\_t rows, const size\_t columns, const int min\_value, const int max\_value)  {  random\_device rd;  mt19937 gen(rd());  uniform\_int\_distribution<> uniformIntDistribution(min\_value, max\_value);    for(size\_t i = 0; i < rows; i++)  {  for (size\_t j = 0; j < columns; j++)  {  array[i][j] = uniformIntDistribution(gen);  }  }  }    void read\_from\_keyboard(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++)  {  cin >> array[i][j];  }  }  }    void show(const int\*\* const array, const size\_t rows, const size\_t columns)  {  for (size\_t i = 0; i < rows; i++)  {  cout << '[' << i << "] - ";  size\_t j = 0;    for (; j < columns; j++)  {  cout << '[' << array[i][j] << "], ";  }  cout << endl;    }  }    int\*\* replace\_max\_element\_with\_zero(const int\*\* const array, const size\_t rows, const size\_t columns)  {  int\*\* new\_array = create\_array(rows, columns);    for (size\_t i = 0; i < rows; i++)  {  const auto max\_value = get\_max\_value(array[i], columns);  for (size\_t j = 0; j < columns; j++)  {  new\_array[i][j] = array[i][j] == max\_value  ? 0  : array[i][j];  }  }    return new\_array;  }    int get\_max\_value(const int\* const array, const size\_t size)  {  int max\_value = array[0];  for (size\_t i = 0; i < size; i++)  {  if (max\_value < array[i])  {  max\_value = array[i];  }  }    return max\_value;  }    int\*\* add\_rows\_of\_zeros(const int\*\* const array, const size\_t rows, const size\_t columns, size\_t& new\_array\_rows)  {  int added\_rows\_count = 0;  for (size\_t i = 0; i < rows; i++)  {  if (array[i][0] % 3 == 0)  {  added\_rows\_count++;  }  }    new\_array\_rows = rows + added\_rows\_count;  auto new\_array = create\_array(new\_array\_rows, columns);    size\_t k = 0;  for (size\_t i = 0; i < rows; i++)  {  if(array[i][0] % 3 == 0)  {  for (size\_t j = 0; j < columns; j++)  {  new\_array[k][j] = 0;  }  k++;  }    for (size\_t j = 0; j < columns; j++)  {  new\_array[k][j] = array[i][j];  }  k++;  }    return new\_array;  }    void delete\_array(int\*\* array, const size\_t rows)  {  if (array != nullptr)  {  for (size\_t i = 0; i < rows; i++)  {  if (array[i] != nullptr)  {  delete[] array[i];  array[i] = nullptr;  }  }    delete[] array;  array = nullptr;  }  } |

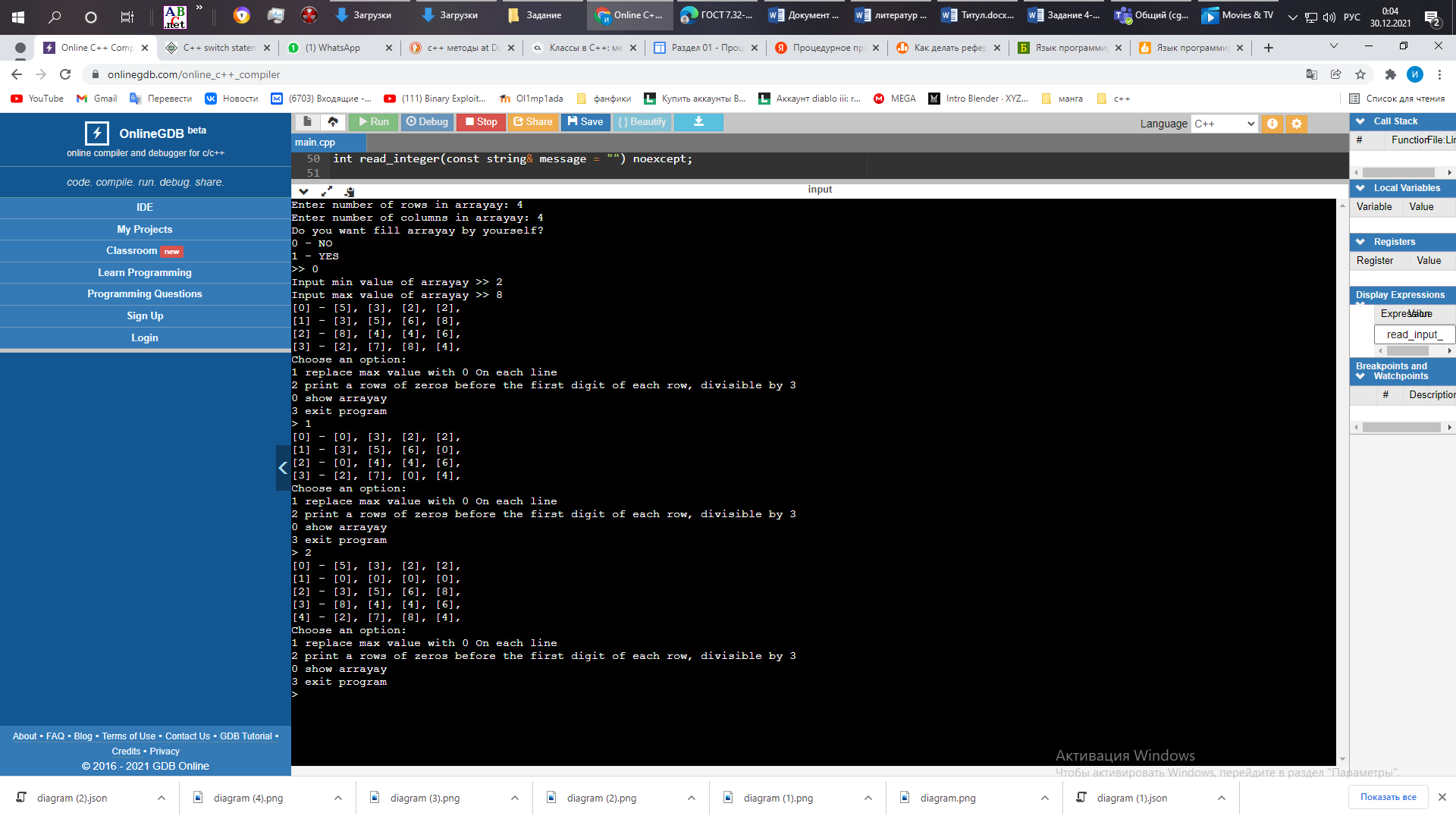
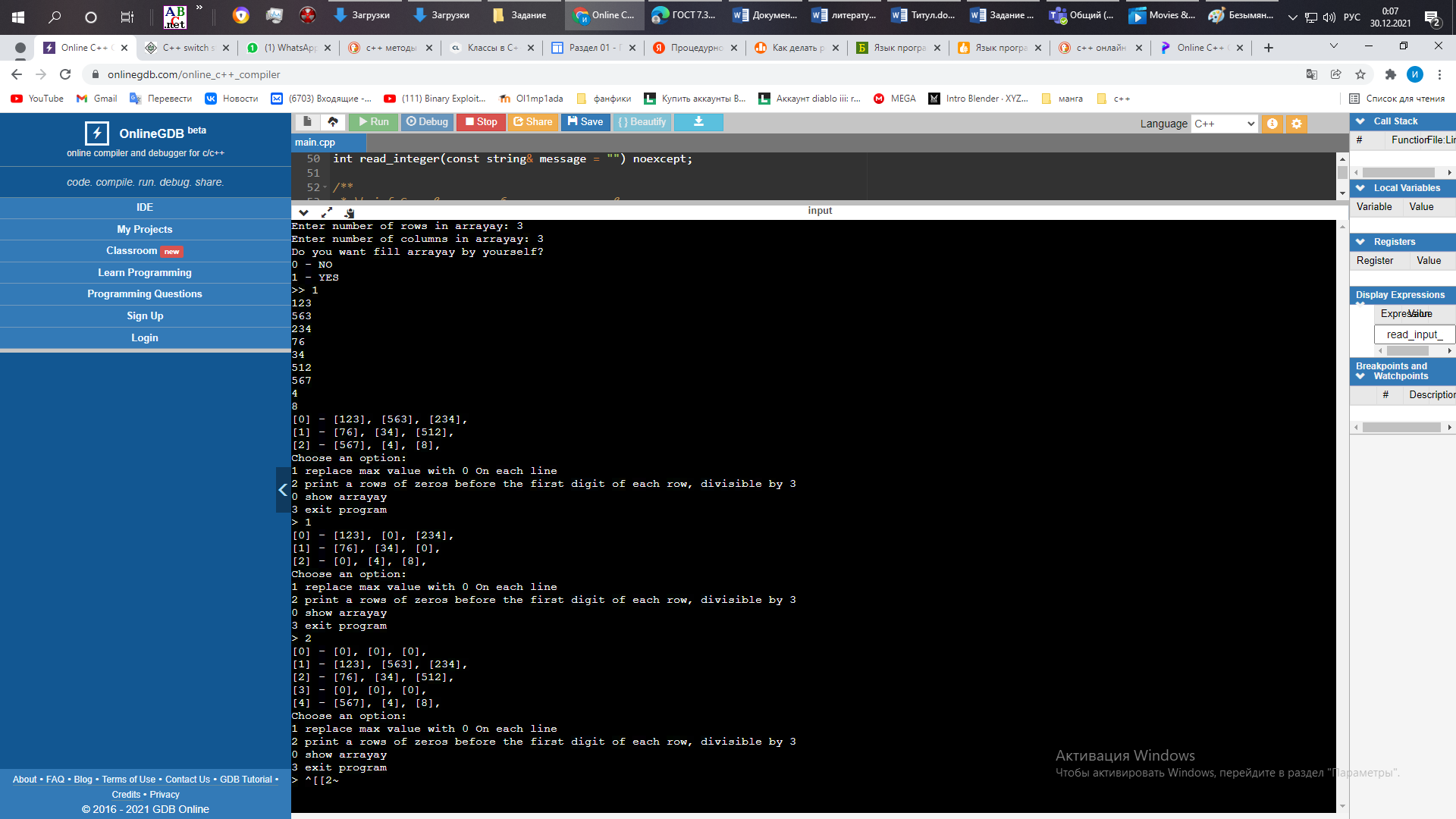
**Блок схема**

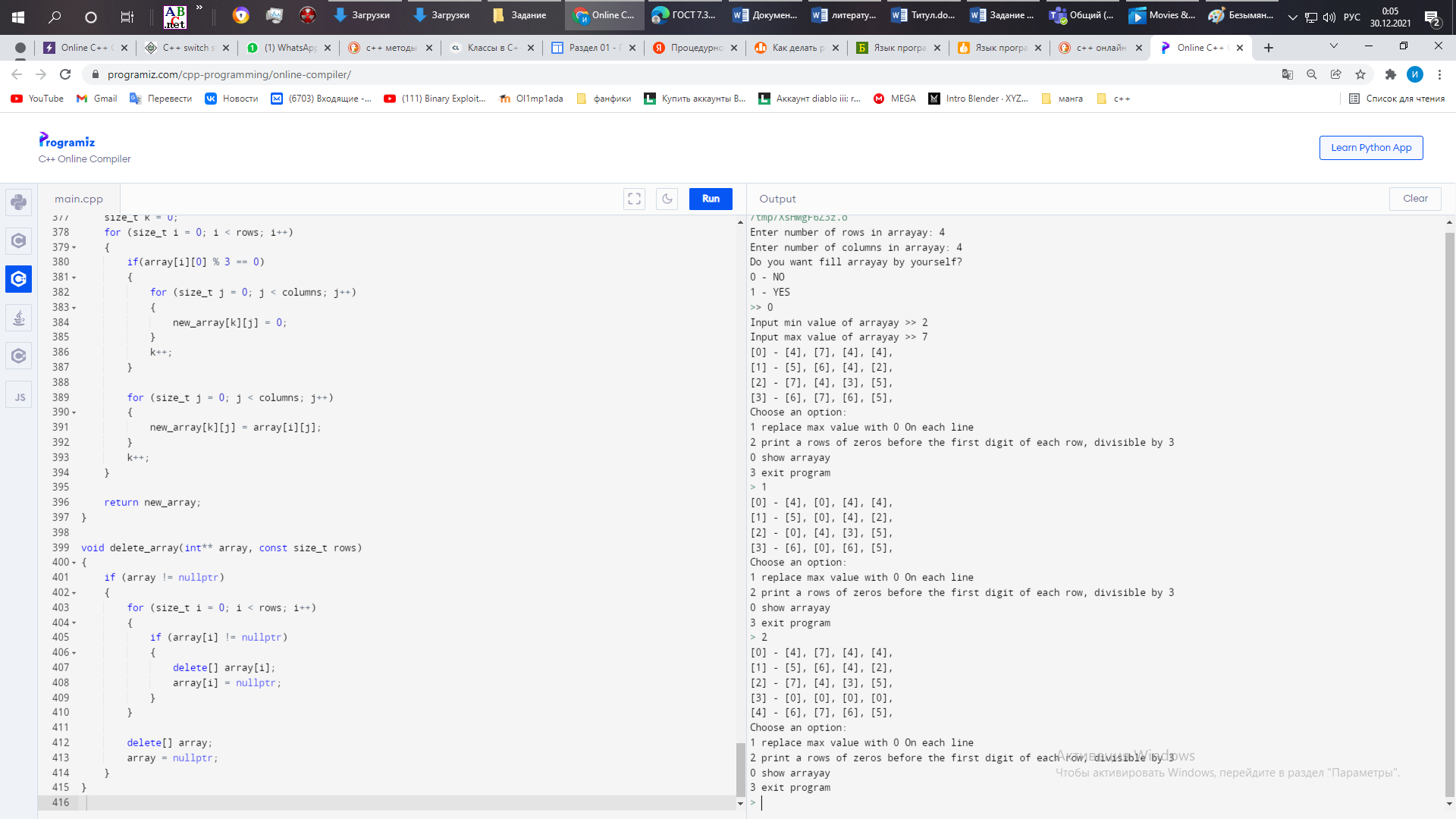
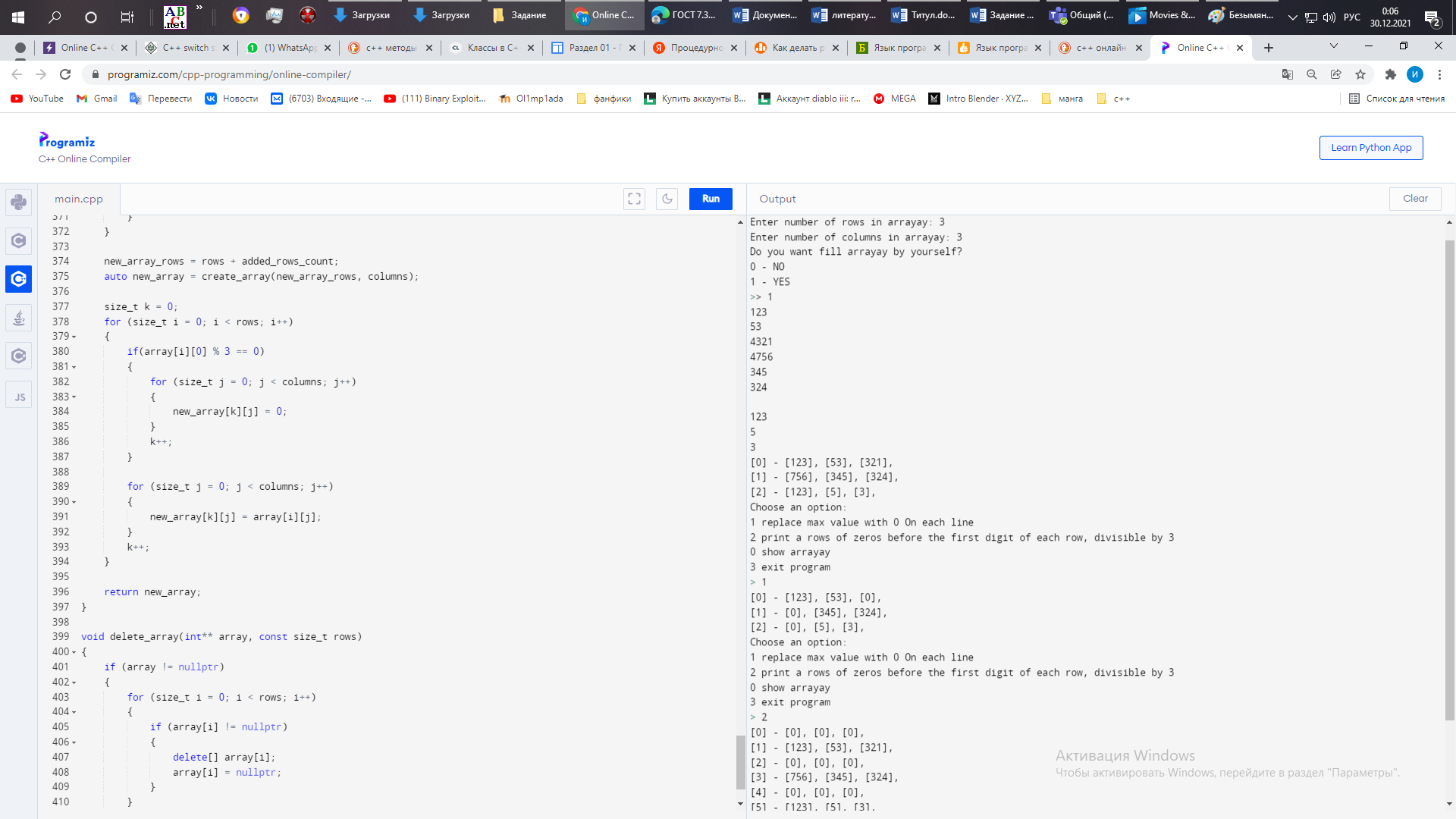




**Функции**



**Приложение** 

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Бьярне Страуструп - Программирование. Принципы и практика с использованием C++ (2016)   
   (URL:[Бьярне Страуструп - Программирование. Принципы и практика с использованием C++ (2016).pdf](file:///C:\Users\renge\Downloads\OneDrive_3_29.12.2021\%D0%91%D1%8C%D1%8F%D1%80%D0%BD%D0%B5%20%D0%A1%D1%82%D1%80%D0%B0%D1%83%D1%81%D1%82%D1%80%D1%83%D0%BF%20-%20%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5.%20%D0%9F%D1%80%D0%B8%D0%BD%D1%86%D0%B8%D0%BF%D1%8B%20%D0%B8%20%D0%BF%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D0%BA%D0%B0%20%D1%81%20%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%D0%BC%20C++%20(2016).pdf))
2. switch statement (C++)  
   (URL:<https://docs.microsoft.com/en-us/cpp/cpp/switch-statement-cpp?view=msvc-170>)
3. try, throw, and catch Statements (C++)  
   (URL: <https://docs.microsoft.com/en-us/cpp/cpp/try-throw-and-catch-statements-cpp?view=msvc-170>)
4. Кочергин А.Н. Научное познание: формы, методы, подходы / А.Н. Кочергин. - М.: Изд-во МГУ, 2011. - 79 с.