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

Черкас Н.В.

Проверил: к.т.н., доц.

Васильева М.А.

Москва 2021

Оглавление

[Задание 4–1 3](#_Toc85287969)

[Формулировка задания 3](#_Toc85287970)

[Блок-схема алгоритма 4](#_Toc85287971)

[Программа на языке С++ 6](#_Toc85287972)

[Решение тестовых примеров на С++ 10](#_Toc85287973)

[Задание 4–2 11](#_Toc85287975)

[Формулировка задания 11](#_Toc85287976)

[Блок-схема алгоритма 12](#_Toc85287977)

[Программа на языке С++ 14](#_Toc85287978)

[Решение тестовых примеров на С++ 19](#_Toc85287979)

[Задание 4–3 20](#_Toc85287982)

[Формулировка задания 20](#_Toc85287983)

[Блок-схема алгоритма 21](#_Toc85287984)

[Программа на языке С++ 24](#_Toc85287985)

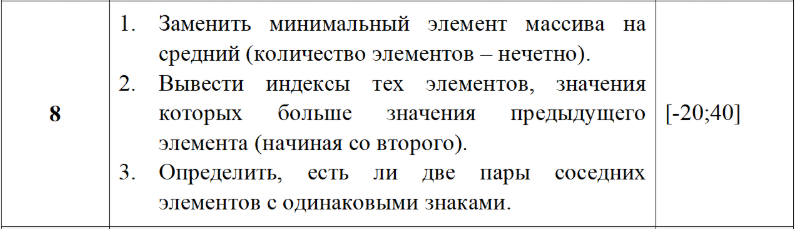
[Решение тестовых примеров на С++ 31](#_Toc85287986)

**Задание 4-1**

**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). Составить блок-схему.

**Таблица 1**

****

**Блок-схема алгоритма**

Начало

Enum, size, choice

Выбор пользователя способа заполнения массива

index = 0; index < size; index++

index = 0; index < size; index++

Ввод элемента массива с клавиатуры

array[index] = rand() % area + minValue

index = 0; index < size; index++

previous \* myArray[index]) >= 0

**True False**

countPairs++

**True False**

ArePairs

Вывод ошибки

Вывод пар элементов

index = 0; index < size; index++

myArray[index] < minArrayValue

**True False**

minArrayValue = myArray[index];minElementIndex = index;

temprary = myArray[minElementIndex]; myArray[minElementIndex] = myArray[(size - 1) / 2]; myArray[(size - 1) / 2] = temprary;

index = 0; index < size; index++

previous < myArray[index]

**True False**

index

myArray

Конец

Рисунок 1 – Блок-схема алгоритма

**Программа на языке C++**

|  |  |
| --- | --- |
|  | #include <iostream> |
|  | #include <string> |
|  | #include <ctime> |
|  |  |
|  | using namespace std; |
|  |  |
|  | /\*\* |
|  | \* \brief Проверка ввода размера массива. |
|  | \* \return Размер массива. |
|  | \*/ |
|  | size\_t GetSize(); |
|  |  |
|  | /\*\* |
|  | \* \brief Вывод элементов массива, значения которых больше значения предыдущего. |
|  | \* \param myArray массив. |
|  | \* \param size размер массива. |
|  | \*/ |
|  | void PrintBiggerElementsIndex(const int\* myArray, const size\_t size); |
|  |  |
|  | /\*\* |
|  | \* \brief Заполнение массива случайными числами. |
|  | \* \param size размер массива. |
|  | \* \param minValue минимальное значение элементов массива. |
|  | \* \param maxValue максимальное значение элементов массива. |
|  | \* \return заполненный массив. |
|  | \*/ |
|  | int\* FillRandomArray(size\_t size, int minValue, int maxValue); |
|  |  |
|  | /\*\* |
|  | \* \brief Вывод массива на консоль. |
|  | \* \param myArray массив. |
|  | \* \param size размер массива. |
|  | \*/ |
|  | void ArrayPrint(const int\* myArray, size\_t size); |
|  |  |
|  | /\*\* |
|  | \* \brief Функция определяет, есть ли 2 пары соседних элементов с одинаковыми знаками. |
|  | \* \param myArray массив. |
|  | \* \param size размер массива. |
|  | \* \return true, если имеются 2 элемента с одинаковыми знаками, false если нет. |
|  | \*/ |
|  | bool ArePairs(const int\* myArray, const size\_t size); |
|  |  |
|  | /\*\* |
|  | \* \brief Замена минимального элемента массива на средний. |
|  | \* \param myArray массив. |
|  | \* \param size размер массива. |
|  | \* \param maxValue максимальное значение, которое может принимать элемент массива. |
|  | \*/ |
|  | void MinToAverageChange(int\* myArray, const size\_t size, const int maxValue); |
|  |  |
|  | /\*\* |
|  | \* \brief Метод, возвращающий заполненный пользователем массив. |
|  | \* \param size размер массива. |
|  | \* \param minValue минимальное значение элементов массива. |
|  | \* \param maxValue максимальное значение элементов массива. |
|  | \* \return заполненный массив. |
|  | \*/ |
|  | int\* FillUserArray(size\_t size); |
|  |  |
|  | /\*\* |
|  | \* \brief Варианы ввода массива. |
|  | \*/ |
|  | enum class ArrayInputWay |
|  | { |
|  | random, |
|  | keyboard |
|  | }; |
|  |  |
|  | /\*\* |
|  | \* \brief Точка входа в программу. |
|  | \* \return 0, в случае успеха. |
|  | \*/ |
|  | int main() |
|  | { |
|  | size\_t size = GetSize(); |
|  |  |
|  | if (size == 0) |
|  | return 1; |
|  |  |
|  | cout << "Как вы хотите заполнить массив?\n"; |
|  | cout << static\_cast<int>(ArrayInputWay::random) << " - random,\n"; |
|  | cout << static\_cast<int>(ArrayInputWay::keyboard) << " - keyboard.\n"; |
|  | cout << "Ваш выбор: "; |
|  | int choice; |
|  | cin >> choice; |
|  |  |
|  | const auto chosen = static\_cast<ArrayInputWay>(choice); |
|  | int\* myArray = nullptr; |
|  |  |
|  | auto minValue = 0; |
|  | auto maxValue = 0; |
|  | cout << "Введите диапазон чисел массива (сначала минимум, потом максимум) " << endl; |
|  | cin >> minValue >> maxValue; |
|  | if (maxValue <= minValue) |
|  | { |
|  | cout << "Введен неправильный диапазон!" << endl; |
|  | } |
|  |  |
|  | switch (chosen) |
|  | { |
|  | case ArrayInputWay::random: |
|  | { |
|  | myArray = FillRandomArray(size, minValue, maxValue); |
|  | break; |
|  | } |
|  | case ArrayInputWay::keyboard: |
|  | { |
|  | myArray = FillUserArray(size); |
|  | break; |
|  | } |
|  | } |
|  |  |
|  | ArrayPrint( myArray, size); |
|  |  |
|  | cout << "Массив с заменённым минимальным элементом: "; |
|  |  |
|  | MinToAverageChange(myArray, size, maxValue); |
|  | ArrayPrint(myArray, size); |
|  |  |
|  | PrintBiggerElementsIndex(myArray, size); |
|  |  |
|  | if (ArePairs(myArray, size)) { |
|  | cout << "Есть 2 пары элементов, знаки которых одинаковы."; |
|  | } |
|  | else{ |
|  | cout << "Нет таких 2 пар соседних элементов, знаки которых одинаковы."; |
|  | } |
|  | cout << endl; |
|  |  |
|  | if (myArray != nullptr) { |
|  |  |
|  | delete[] myArray; |
|  | myArray = nullptr; |
|  |  |
|  | } |
|  | return 0; |
|  |  |
|  | } |
|  |  |
|  | size\_t GetSize(){ |
|  | int size = 0; |
|  | cout << "Введите размер массива"<< endl; |
|  | cin >> size; |
|  | if (size <= 0) |
|  | { |
|  | cout<< "Введён неверный размер"; |
|  | return 0; |
|  | } |
|  | else |
|  | return size; |
|  | }; |
|  |  |
|  | void PrintBiggerElementsIndex(const int\* myArray, const size\_t size){ |
|  | auto previous = 0; |
|  | cout<<"Индексы элементов, значение которых больше значения предыдущего: "; |
|  | for (size\_t index = 1; index < size; index++){ |
|  | previous = myArray[index - 1]; |
|  | if (previous < myArray[index]){ |
|  | cout<<index << " "; |
|  | } |
|  | } |
|  | cout<<"\n"; |
|  | } |
|  |  |
|  | bool ArePairs(const int\* myArray, const size\_t size ) { |
|  | int countPairs = 0; |
|  | int previous = 0; |
|  | for (size\_t index = 1; index < size; index=index + 2) { |
|  | previous = myArray[index - 1]; |
|  | if ((previous \* myArray[index]) >= 0) { |
|  | countPairs++; |
|  | } |
|  | } |
|  | return countPairs >= 2; |
|  |  |
|  | } |
|  |  |
|  | void MinToAverageChange(int\* myArray, const size\_t size, const int maxValue ) |
|  | { |
|  | int temprary = 0; |
|  | if (myArray == nullptr) |
|  | cout << "Массив пуст"; |
|  |  |
|  | auto minArrayValue = maxValue; |
|  | auto minElementIndex = 0; |
|  |  |
|  | for (size\_t index = 0; index < size; index++) { |
|  | if (myArray[index] < minArrayValue) { |
|  | minArrayValue = myArray[index]; |
|  | minElementIndex = index; |
|  | } |
|  | } |
|  |  |
|  | temprary = myArray[minElementIndex]; |
|  | myArray[minElementIndex] = myArray[(size - 1) / 2]; |
|  | myArray[(size - 1) / 2] = temprary; |
|  | } |
|  |  |
|  | void ArrayPrint(const int\* myArray, const size\_t size) |
|  | { |
|  | if (myArray == nullptr) |
|  | { |
|  | cout << "Массива не существует"; |
|  | } |
|  | else { |
|  | cout << "\nМассив:\n"; |
|  | for (size\_t index = 0; index < size; index++) { |
|  | cout << myArray[index] << " "; |
|  | } |
|  | cout << "\n"; |
|  |  |
|  | } |
|  | } |
|  |  |
|  | int\* FillRandomArray(const size\_t size, const int minValue, const int maxValue) |
|  | { |
|  | srand(time(NULL)); |
|  | const auto area = abs(minValue) + abs(maxValue) + 1; |
|  | auto\* array = new int[size]; |
|  | for (size\_t index = 0; index < size; index++) { |
|  | array[index] = rand() % area + minValue; |
|  | cout << array[index] << endl; |
|  | } |
|  |  |
|  | return array; |
|  | } |
|  |  |
|  | int\* FillUserArray(const size\_t size) |
|  | { |
|  | auto\* array = new int[size]; |
|  | cout << "Введите элементы массива" << "\n"; |
|  | for (size\_t index = 0; index < size; index++) |
|  | { |
|  | cin >> array[index]; |
|  | } |
|  | return array; |
|  | } |

**Решение тестовых примеров на языке C++**

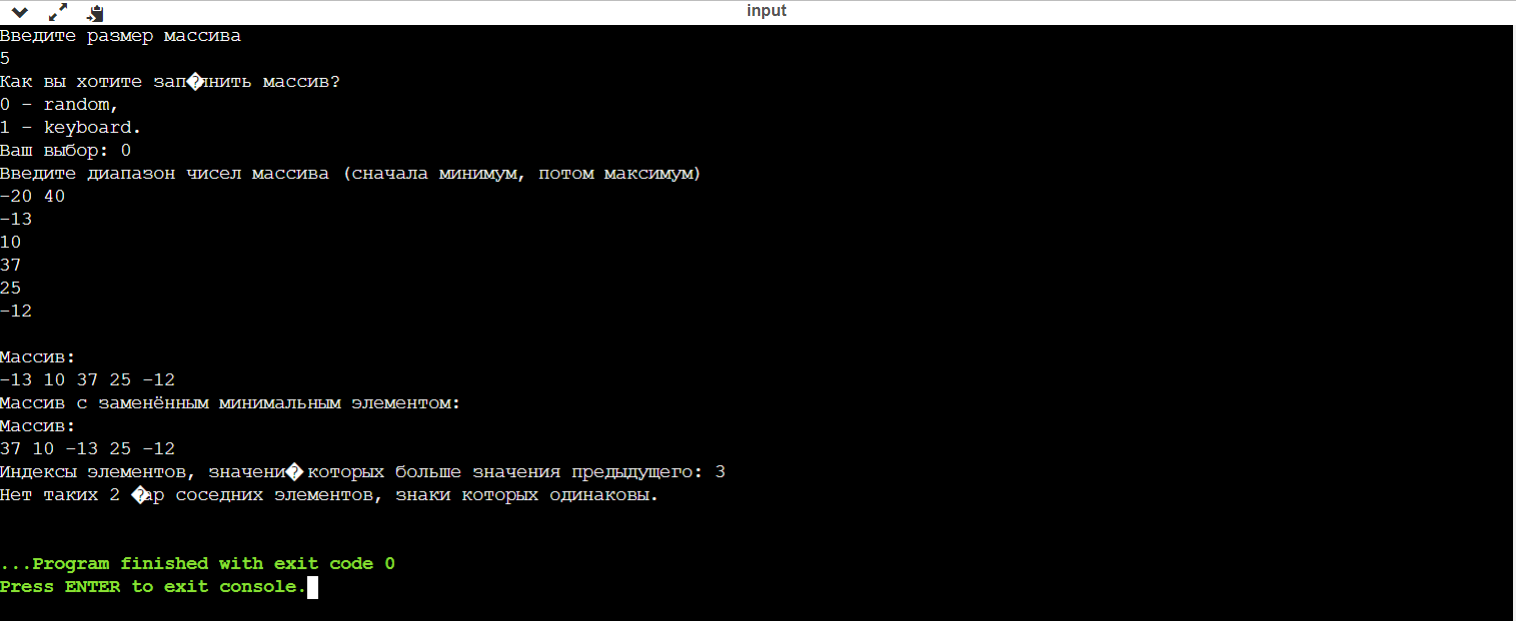
****

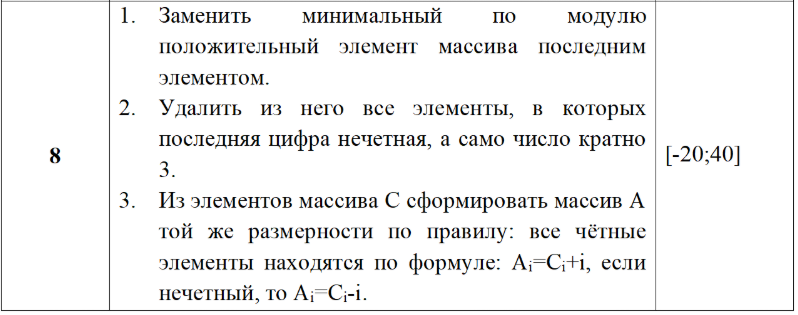
Рисунок 2 – Решение тестовых примеров на C++

**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). Составить блок-схему.

**Таблица 2**

****

**Блок-схема алгоритма**

Начало

Enum, size, choice

Выбор пользователя способа заполнения массива

index = 0; index < size; index++

index = 0; index < size; index++

Ввод элемента массива с клавиатуры

array[index] = rand() % area + minValue

index = 0; index < size; index++

abs(myArray[index]) < minArrayValue

**True False**

|  |
| --- |
| minArrayValue = abs(myArray[index]); |
|  | minElementIndex = index; |

temprary = myArray[minElementIndex]; myArray[minElementIndex] = myArray[size - 1]; myArray[size - 1] = temprary;

index = 0; index < size; index++

myArray[index] % 2 == 1 && myArray[index] % 3 == 0

**True False**

count++;

index = 0; newindex = 0; index < size; index++

!(myArray[index] % 2 == 1 && myArray[index] % 3 == 0)

**True False**

newArray[newindex] = myArray[index]; newindex++;

index = 0; newindex = 0; index < size; index++

myArray[index] % 2 == 1

**True False**

newArray[newindex] = myArray[index] + index

newArray[newindex] = myArray[index] - index

myArray

Конец

Рисунок 3 – Блок-схема алгоритма

**Программа на языке C++**

|  |  |
| --- | --- |
|  | #include <iostream> |
|  | #include <string> |
|  | #include <ctime> |
|  | #include <random> |
|  | using namespace std; |
|  |  |
|  | /\*\* |
|  | \* \brief Проверка ввода размера массива. |
|  | \* \return Размер массива. |
|  | \*/ |
|  | size\_t GetSize(); |
|  |  |
|  | /\*\* |
|  | \*\brief Вычисление нового размера массива. |
|  | \*\param size размер массива. |
|  | \*\return Размер нового массива. |
|  | \*/ |
|  | size\_t NewSize(const int\* myArray, size\_t size); |
|  |  |
|  | /\*\* |
|  | \* \brief Заполнение массива случайными числами. |
|  | \* \param size размер массива. |
|  | \* \param minValue минимальное значение элементов массива. |
|  | \* \param maxValue максимальное значение элементов массива. |
|  | \* \return заполненный массив. |
|  | \*/ |
|  | int\* FillRandomArray(size\_t size, int minValue, int maxValue); |
|  |  |
|  | /\*\* |
|  | \* \brief Вывод массива на консоль. |
|  | \* \param myArray массив. |
|  | \* \param size размер массива. |
|  | \*/ |
|  | void ArrayPrint(const int\* myArray, size\_t size); |
|  |  |
|  | /\*\* |
|  | \* \brief Удаление нечетных элементов, кратных 3. |
|  | \* \param myArray массив. |
|  | \* \param size размер массива. |
|  | \* \return изменённый массив. |
|  | \*/ |
|  | int\* IsOddAndMulpiples3(int\* myArray, size\_t size); |
|  |  |
|  | /\*\* |
|  | \* \brief Создание нового массива по заданным формулам. |
|  | \* \param myArray массив. |
|  | \* \param size размер массива. |
|  | \* \return изменённый массив. |
|  | \*/ |
|  | int\* ArrayChange(int\* myArray, size\_t size); |
|  |  |
|  | /\*\* |
|  | \* \brief Замена минимального элемента массива на средний. |
|  | \* \param myArray массив. |
|  | \* \param size размер массива. |
|  | \* \param maxValue максимальное значение, которое может принимать элемент массива. |
|  | \*/ |
|  | int\* MinToLastChange(const size\_t size, const int maxValue); |
|  |  |
|  | /\*\* |
|  | \* \brief Метод, возвращающий заполненный пользователем массив. |
|  | \* \param size размер массива. |
|  | \* \param minValue минимальное значение элементов массива. |
|  | \* \param maxValue максимальное значение элементов массива. |
|  | \* \return заполненный массив. |
|  | \*/ |
|  | int\* FillUserArray(size\_t size); |
|  |  |
|  | /\*\* |
|  | \* \brief Варианы ввода массива. |
|  | \*/ |
|  | enum class ArrayInputWay |
|  | { |
|  | random, |
|  | keyboard |
|  | }; |
|  |  |
|  | /\*\* |
|  | \* \brief Точка входа в программу. |
|  | \* \return 0, в случае успеха. |
|  | \*/ |
|  | int main() |
|  | { |
|  | setlocale(LC\_ALL, "Russian"); |
|  | size\_t size = GetSize(); |
|  |  |
|  | if (size == 0) |
|  | return 1; |
|  |  |
|  | cout << "Как вы хотите заполнить массив?\n"; |
|  | cout << static\_cast<int>(ArrayInputWay::random) << " - random,\n"; |
|  | cout << static\_cast<int>(ArrayInputWay::keyboard) << " - keyboard.\n"; |
|  | cout << "Ваш выбор: "; |
|  | int choice; |
|  | cin >> choice; |
|  |  |
|  | const auto chosen = static\_cast<ArrayInputWay>(choice); |
|  | int\* myArray = nullptr; |
|  |  |
|  | auto minValue = 0; |
|  | auto maxValue = 0; |
|  | cout << "Введите диапазон чисел массива (сначала минимум, потом максимум) " << endl; |
|  | cin >> minValue >> maxValue; |
|  | if (maxValue <= minValue) |
|  | { |
|  | cout << "Введен неправильный диапазон!" << endl; |
|  | } |
|  |  |
|  | switch (chosen) |
|  | { |
|  | case ArrayInputWay::random: |
|  | { |
|  | myArray = FillRandomArray(size, minValue, maxValue); |
|  | break; |
|  | } |
|  | case ArrayInputWay::keyboard: |
|  | { |
|  | myArray = FillUserArray(size); |
|  | break; |
|  | } |
|  | } |
|  |  |
|  | cout << "Массив с заменённым минимальным элементом: " << endl; |
|  |  |
|  | myArray = MinToLastChange(size, maxValue); |
|  | ArrayPrint(myArray, size); |
|  |  |
|  |  |
|  | int\* newArray2 = new int[NewSize(myArray, size)]; |
|  | cout << "Массив с удалёнными нечетными элементами, кратными 3м" << endl; |
|  | newArray2 = IsOddAndMulpiples3(myArray, size); |
|  | ArrayPrint(newArray2, size); |
|  |  |
|  | if (newArray2 != nullptr) { |
|  |  |
|  | delete[] newArray2; |
|  | newArray2 = nullptr; |
|  |  |
|  | } |
|  |  |
|  | cout << "Заменённый массив по формулам A[i] = C[i] + i и A[i] = C[i] - i" << endl; |
|  | newArray2 = ArrayChange(myArray, size); |
|  | ArrayPrint(newArray2, size); |
|  |  |
|  | if (myArray != nullptr) { |
|  |  |
|  | delete[] myArray; |
|  | myArray = nullptr; |
|  |  |
|  | } |
|  | return 0; |
|  |  |
|  | } |
|  |  |
|  | size\_t GetSize() { |
|  | int size = 0; |
|  | cout << "Введите размер массива" << endl; |
|  | cin >> size; |
|  | if (size <= 0) |
|  | { |
|  | cout << "Введён неверный размер"; |
|  | return 0; |
|  | } |
|  | else |
|  | return size; |
|  | }; |
|  |  |
|  | size\_t NewSize(const int\* myArray, size\_t size) { |
|  | size\_t count = 0; |
|  | for (size\_t index = 0; index < size; index++) { |
|  | if (myArray[index] % 2 == 1 && myArray[index] % 3 == 0) { |
|  | count++; |
|  | } |
|  | } |
|  | size = size - count; |
|  | return size; |
|  | } |
|  |  |
|  | int\* MinToLastChange(const size\_t size, const int maxValue) |
|  | { |
|  | int temprary = 0; |
|  |  |
|  | auto minArrayValue = abs(maxValue); |
|  | size\_t minElementIndex = 0; |
|  |  |
|  | int\* newArray = new int[size]; |
|  | for (size\_t index = 0; index < size; index++) { |
|  | if (abs(newArray[index]) < minArrayValue) { |
|  | minArrayValue = abs(newArray[index]); |
|  | minElementIndex = index; |
|  | } |
|  | } |
|  |  |
|  | temprary = newArray[minElementIndex]; |
|  | newArray[minElementIndex] = newArray[size - 1]; |
|  | newArray[size - 1] = temprary; |
|  |  |
|  | return newArray; |
|  | } |
|  |  |
|  | int\* ArrayChange(int\* myArray, size\_t size) |
|  | { |
|  | int temprary = 0; |
|  | if (myArray == nullptr) { |
|  | cout << "Массив пуст"; |
|  | return nullptr; |
|  | } |
|  |  |
|  | int\* newArray = new int[size]; |
|  | for (size\_t index = 0, newindex = 0; index < size; index++) { |
|  | if (myArray[index] % 2 == 1) { |
|  | newArray[newindex] = myArray[index] - index; |
|  | } |
|  | else { |
|  | newArray[newindex] = myArray[index] + index; |
|  | } |
|  | newindex++; |
|  | } |
|  |  |
|  | return newArray; |
|  | } |
|  |  |
|  | int\* IsOddAndMulpiples3(int\* myArray, size\_t size) |
|  | { |
|  | if (myArray == nullptr) { |
|  | cout << "Массив пуст"; |
|  | return nullptr; |
|  | } |
|  |  |
|  | size\_t count = 0; |
|  | for (size\_t index = 0; index < size; index++) { |
|  | if (myArray[index] % 2 == 1 && myArray[index] % 3 == 0) { |
|  | count++; |
|  | } |
|  | } |
|  |  |
|  | int\* newArray = new int[size - count]; |
|  | for (size\_t index = 0, newindex = 0; index < size; index++) { |
|  | if (!(myArray[index] % 2 == 1 && myArray[index] % 3 == 0)) { |
|  | newArray[newindex] = myArray[index]; |
|  | newindex++; |
|  | } |
|  | } |
|  | return newArray; |
|  | } |
|  |  |
|  | void ArrayPrint(const int\* myArray, const size\_t size) |
|  | { |
|  | if (myArray == nullptr) |
|  | { |
|  | cout << "Массива не существует"; |
|  | } |
|  | else { |
|  | cout << "Массив:\n"; |
|  | for (size\_t index = 0; index < size; index++) { |
|  | cout << myArray[index] << " "; |
|  | } |
|  | cout << "\n"; |
|  |  |
|  | } |
|  | } |
|  |  |
|  | int\* FillRandomArray(const size\_t size, const int minValue, const int maxValue) |
|  | { |
|  | random\_device rd; |
|  |  |
|  | mt19937 gen(rd()); |
|  |  |
|  | const std::uniform\_int\_distribution<> uniformIntDistribution(minValue, maxValue); |
|  |  |
|  | auto\* myArray = new int[size]; |
|  |  |
|  | for (size\_t index = 0; index < size; index++) |
|  | { |
|  | myArray[index] = uniformIntDistribution(gen); |
|  | } |
|  | return myArray; |
|  | } |
|  |  |
|  | int\* FillUserArray(const size\_t size) |
|  | { |
|  | auto\* array = new int[size]; |
|  | cout << "Введите элементы массива" << "\n"; |
|  | for (size\_t index = 0; index < size; index++) |
|  | { |
|  | cin >> array[index]; |
|  | } |
|  | return array; |
|  | } |

**Решение тестовых примеров на языке C++**

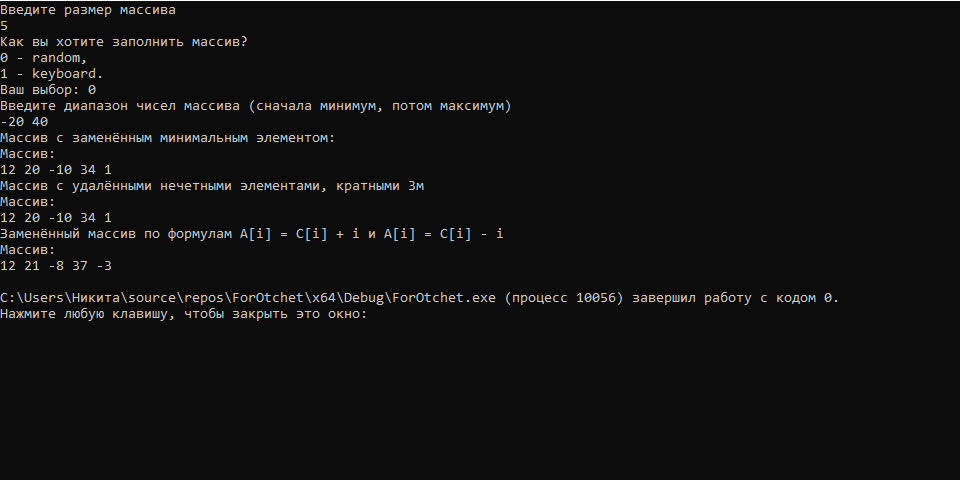
****

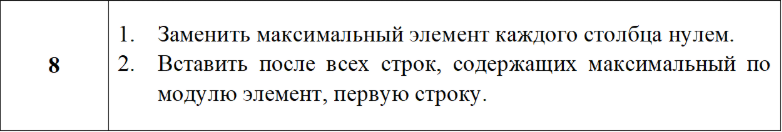
Рисунок 4 – Решение тестовых примеров на языке C++

**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.

**Таблица 3**

****

**Блок-схема алгоритма**

Начало

size, choice

Выбор пользователя способа заполнения массива

Заполнение массива рандомно

Заполнение массива вручную

Вывод массива

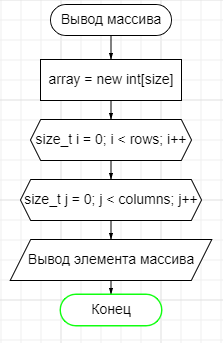
Замена максимального элемента каждого столбца на 0

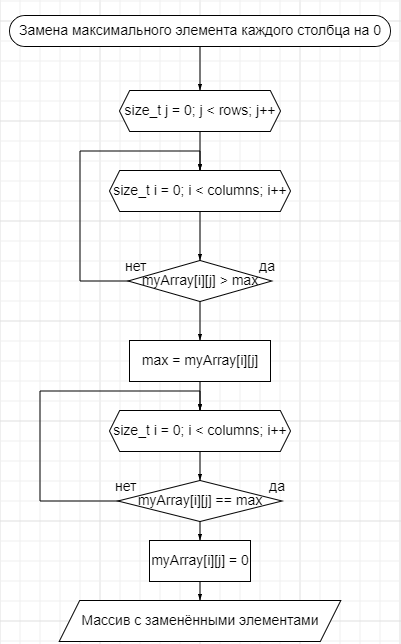
Создание массива с добавленными строками

Вывод массива с добавленными строками

Массив с добавленными строками

Конец





конец

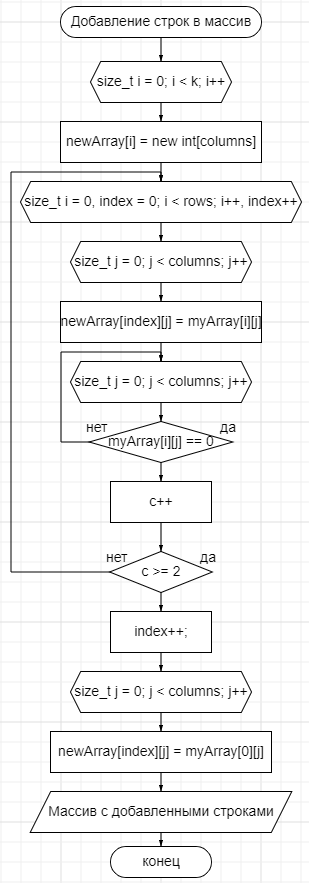


Рисунок 5 – Решение тестовых примеров на языке C++

**Программа на языке C++**

|  |  |
| --- | --- |
|  | #include <iostream> |
|  | #include <string> |
|  | #include <ctime> |
|  | #include <random> |
|  | using namespace std; |
|  |  |
|  | /\*\* |
|  | \* \brief Вычисление нового размера массива. |
|  | \* \param myArray Массив. |
|  | \* \param rows Количество строк. |
|  | \* \param columns Количество столбцов. |
|  | \* \return Количество строк нового массива. |
|  | \*/ |
|  | size\_t NewSize(int\*\* myArray, const int rows, const int columns); |
|  |  |
|  | /\*\* |
|  | \* \brief Замена максимального элемента столбца на 0. |
|  | \* \param myArray Массив. |
|  | \* \param rows Количество строк. |
|  | \* \param columns Количество столбцов. |
|  | \* \param minValue Минимальное значение массива. |
|  | \* \return Изменённый массив. |
|  | \*/ |
|  | void MaxToZeroChange(int\*\* myArray, const int rows, const int columns, const int minValue); |
|  |  |
|  | /\*\* |
|  | \* \brief Добавление строк в массив, в случае выполнения условия. |
|  | \* \param myArray Массив. |
|  | \* \param newArray Новый изменённый массив. |
|  | \* \param rows Количество строк. |
|  | \* \param columns Количество столбцов. |
|  | \* \return Изменённый массив. |
|  | \*/ |
|  | int\*\* StrAddIfTwoZeros(int\*\* myArray, int rows, int columns); |
|  |  |
|  | /\*\* |
|  | \* \brief Заполнение массива случайными числами. |
|  | \* \param rows количество строчек. |
|  | \* \param columns количество столбцов. |
|  | \* \param minValue минимальное значение элементов массива. |
|  | \* \param maxValue максимальное значение элементов массива. |
|  | \* \return заполненный массив. |
|  | \*/ |
|  | int\*\* FillRandomArray(const int rows, const int columns, int minValue, int maxValue); |
|  |  |
|  | /\*\* |
|  | \* \brief Вывод массива на консоль. |
|  | \* \param myArray массив. |
|  | \* \param rows количество строчек. |
|  | \* \param columns количество столбцов. |
|  | \*/ |
|  | void ArrayPrint(int\*\* myArray, const int rows, const int columns); |
|  |  |
|  | /\*\* |
|  | \* \brief Метод, возвращающий заполненный пользователем массив. |
|  | \* \param rows количество строчек. |
|  | \* \param columns количество столбцов. |
|  | \* \param minValue минимальное значение элементов массива. |
|  | \* \param maxValue максимальное значение элементов массива. |
|  | \* \return заполненный массив. |
|  | \*/ |
|  | int\*\* FillUserArray(const int rows, const int columns); |
|  |  |
|  | /\*\* |
|  | \* \brief Варианы ввода массива. |
|  | \*/ |
|  | enum class ArrayInputWay |
|  | { |
|  | random, |
|  | keyboard |
|  | }; |
|  |  |
|  | /\*\* |
|  | \* \brief Точка входа в программу. |
|  | \* \return 0, в случае успеха. |
|  | \*/ |
|  | int main() |
|  | { |
|  | setlocale(LC\_ALL, "Russian"); |
|  | int rows = 0; |
|  | int columns = 0; |
|  | cout << "Введите количество строк и столбцов" << endl; |
|  | cin >> rows >> columns; |
|  |  |
|  | if (rows <= 0 || columns <= 0) { |
|  | cout << "Err"; |
|  | return 1; |
|  | } |
|  |  |
|  | cout << "Как вы хотите заполнить массив?\n"; |
|  | cout << static\_cast<int>(ArrayInputWay::random) << " - random,\n"; |
|  | cout << static\_cast<int>(ArrayInputWay::keyboard) << " - keyboard.\n"; |
|  | cout << "Ваш выбор: "; |
|  | int choice; |
|  | cin >> choice; |
|  |  |
|  | const auto chosen = static\_cast<ArrayInputWay>(choice); |
|  | int\*\* myArray = nullptr; |
|  |  |
|  | auto minValue = 0; |
|  | auto maxValue = 0; |
|  | cout << "Введите диапазон чисел массива (сначала минимум, потом максимум) " << endl; |
|  | cin >> minValue >> maxValue; |
|  | if (maxValue <= minValue) |
|  | { |
|  | cout << "Введен неправильный диапазон!" << endl; |
|  | } |
|  |  |
|  | switch (chosen) |
|  | { |
|  | case ArrayInputWay::random: |
|  | { |
|  | myArray = FillRandomArray(rows, columns, minValue, maxValue); |
|  | break; |
|  | } |
|  | case ArrayInputWay::keyboard: |
|  | { |
|  | myArray = FillUserArray(rows, columns); |
|  | break; |
|  | } |
|  | } |
|  |  |
|  | ArrayPrint(myArray, rows, columns); |
|  |  |
|  | MaxToZeroChange( myArray, rows, columns, minValue); |
|  |  |
|  | int\*\* newArray2 = new int\*[NewSize(myArray, rows, columns)]; |
|  |  |
|  | size\_t k = NewSize(myArray, rows, columns); |
|  |  |
|  | for (size\_t i = 0; i < k; i++) { |
|  |  |
|  | newArray2[i] = new int[columns]; |
|  | } |
|  |  |
|  | cout << "Массив с добавленными строками: "; |
|  | newArray2 = StrAddIfTwoZeros(myArray, rows, columns); |
|  | ArrayPrint(newArray2, NewSize(myArray, rows, columns), columns); |
|  |  |
|  | if (newArray2 != nullptr) { |
|  |  |
|  | delete[] newArray2; |
|  | newArray2 = nullptr; |
|  |  |
|  | } |
|  |  |
|  | if (myArray != nullptr) { |
|  |  |
|  | delete[] myArray; |
|  | myArray = nullptr; |
|  |  |
|  | } |
|  | return 0; |
|  |  |
|  | } |
|  |  |
|  |  |
|  | size\_t NewSize(int\*\* myArray, const int rows, const int columns) { |
|  |  |
|  | int countRows = 0; |
|  | int countZeros = 0; |
|  | for (size\_t i = 0; i < rows; i++) { |
|  | countZeros = 0; |
|  | for (size\_t j = 0; j < columns; j++) { |
|  | if (myArray[i][j] == 0) |
|  | countZeros++; |
|  | } |
|  | if (countZeros >= 2) |
|  | countRows++; |
|  | } |
|  |  |
|  | return rows + countRows; |
|  | } |
|  |  |
|  | void MaxToZeroChange(int\*\* myArray, const int rows,const int columns, const int minValue) { |
|  |  |
|  | int max = minValue; |
|  |  |
|  | for (size\_t j = 0; j < rows; j++) |
|  | { |
|  | max = minValue; |
|  |  |
|  | for (size\_t i = 0; i < columns; i++) |
|  | { |
|  | if (myArray[i][j] > max){ |
|  | max = myArray[i][j]; |
|  | } |
|  | } |
|  |  |
|  | for (size\_t i = 0; i < columns; i++) |
|  | { |
|  | if (myArray[i][j] == max) { |
|  | myArray[i][j] = 0; |
|  | } |
|  | } |
|  | } |
|  | ArrayPrint(myArray, rows, columns); |
|  | } |
|  |  |
|  | int\*\* StrAddIfTwoZeros(int\*\* myArray, int rows, int columns) { |
|  |  |
|  | int c = 0; |
|  | size\_t k = NewSize(myArray, rows, columns); |
|  |  |
|  | int\*\* newArray = new int\* [NewSize(myArray, rows, columns)]; |
|  |  |
|  | for (size\_t i = 0; i < k; i++) { |
|  |  |
|  | newArray[i] = new int[columns]; |
|  | } |
|  |  |
|  | for (size\_t i = 0, index = 0; i < rows; i++, index++) { |
|  | c = 0; |
|  |  |
|  | for (size\_t j = 0; j < columns; j++) { |
|  |  |
|  | newArray[index][j] = myArray[i][j]; |
|  |  |
|  | } |
|  | for (size\_t j = 0; j < columns; j++) { |
|  | if (myArray[i][j] == 0) |
|  | c++; |
|  | } |
|  | if (c >= 2) { |
|  |  |
|  | index++; |
|  |  |
|  | for (size\_t j = 0; j < columns; j++) { |
|  |  |
|  | newArray[index][j] = myArray[0][j]; |
|  |  |
|  | } |
|  | } |
|  | } |
|  |  |
|  | return newArray; |
|  | } |
|  |  |
|  | void ArrayPrint(int\*\* myArray, const int rows, const int columns) |
|  | { |
|  | if (myArray == nullptr) |
|  | { |
|  | cout << "Массива не существует"; |
|  | } |
|  | else { |
|  | cout << "Массив:\n"; |
|  | for (size\_t i = 0; i < rows; i++) { |
|  | for (size\_t j = 0; j < columns; j++) { |
|  | cout << myArray[i][j] << " "; |
|  | } |
|  | cout << endl; |
|  | } |
|  | cout << "\n"; |
|  |  |
|  | } |
|  | } |
|  |  |
|  | int\*\* FillRandomArray(const int rows, const int columns, const int minValue, const int maxValue) |
|  | { |
|  | random\_device rd; |
|  |  |
|  | mt19937 gen(rd()); |
|  |  |
|  | const std::uniform\_int\_distribution<> uniformIntDistribution(minValue, maxValue); |
|  |  |
|  | int\*\* myArray = new int\* [rows]; |
|  |  |
|  | for (size\_t index = 0; index < rows; index++) { |
|  | myArray[index] = new int[columns]; |
|  | } |
|  | for (size\_t i = 0; i < rows; i++) |
|  | { |
|  | for (size\_t j = 0; j < columns; j++) |
|  | { |
|  | myArray[i][j] = uniformIntDistribution(gen); |
|  | } |
|  | } |
|  | return myArray; |
|  | } |
|  |  |
|  | int\*\* FillUserArray(const int rows, const int columns) |
|  | { |
|  | int\*\* myArray = new int\* [rows]; |
|  |  |
|  | for (size\_t index = 0; index < rows; index++) { |
|  | myArray[index] = new int[columns]; |
|  | } |
|  | for (size\_t i = 0; i < rows; i++) |
|  | { |
|  | for (size\_t j = 0; j < columns; j++) |
|  | { |
|  | cin>> myArray[i][j]; |
|  | } |
|  | } |
|  | return myArray; |
|  | } |

**Решение тестовых примеров на языке C++**

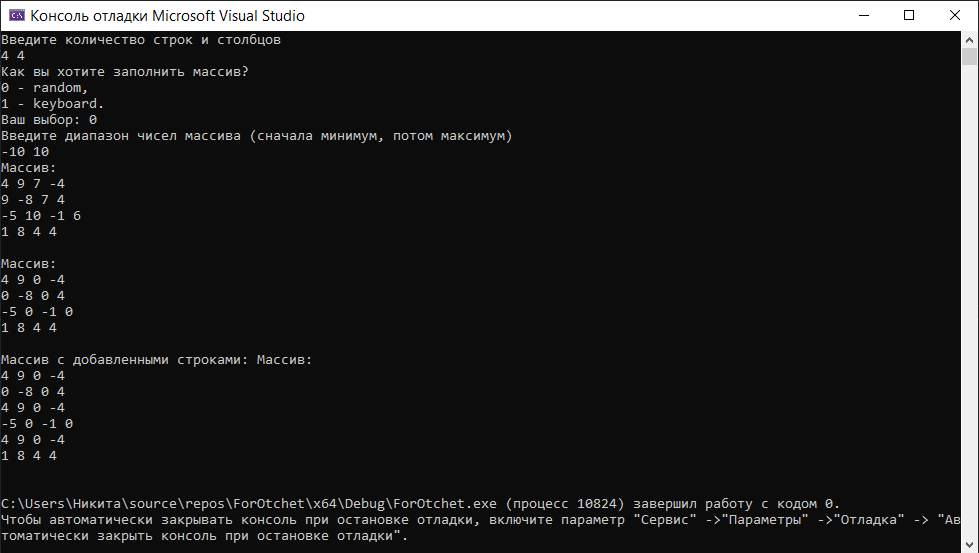
****

Рисунок 6 – Решение тестовых примеров на языке C++