Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Кафедра прикладной математики и кибернетики

**ОТЧЕТ**

по лабораторной работе 1

по дисциплине «**Объектно-ориентированное программирование**»

|  |  |  |
| --- | --- | --- |
| Выполнил:  студент гр. ИС-242  «11» сентября 2023 г. | \_\_\_\_\_\_\_\_\_\_\_\_ | /Любицкий М.Е./ |
|  |  |  |
| Проверил:  Доцент Кафедры ПМиК  «11» сентября 2023 г. | \_\_\_\_\_\_\_\_\_\_\_\_ | /Ситняковская Е.И./ |

Оценка «\_\_\_\_\_\_\_\_\_\_\_\_\_»

Новосибирск 2023

**ОГЛАВЛЕНИЕ**

[ОГЛАВЛЕНИЕ 2](#_Toc145363856)

[ЗАДАНИЕ 3](#_Toc145363857)

[ВЫПОЛНЕНИЕ РАБОТЫ 5](#_Toc145363858)

[ПРИЛОЖЕНИЕ 8](#_Toc145363859)

# **ЗАДАНИЕ**

**Задание 1.**

Создать одномерный динамический массив типа int, заполнить его случайными числами, вывести на экран. Размер массива необходимо хранить в первом элементе массива.

Необходимо реализовать следующие функции:

int\* genRandArray(int size, int maxValue);

void print(int\* arr);

main должен выглядеть следующим образом:

int main(){

int size = rand()%10;

int maxValue = 100;

int\* arr = genRandArray(size, maxValue);

print(arr);

//очистка выделенной памяти

}

Пример вывода:

7: 44 11 24 41 10 57 100

**Задание 2.**

Создать двумерный динамический массив типа int, заполнить его случайными числами, вывести на экран. Размер каждого одномерного массива – произвольный (матрица не обязана быть прямоугольной). Количество строк матрицы хранить в отдельной переменной в main.

Функции, реализованные в первом задании, рекомендуется использовать во втором.

Необходимо реализовать следующие функции:

int\*\* genRandMatrix(intsize, intmaxValue);

void printMatrix(int\*\* matrix);

int main(){

int size=rand()%10;

int maxValue = 100;

int\*\* matrix = genRandMatrix(size, maxValue);

print(matrix);

//очистка памяти

}

Пример вывода:

4

1: 15

5: 54 23 15 5 12

7: 1 32 51 42 51 100 12

3: 50 37 17

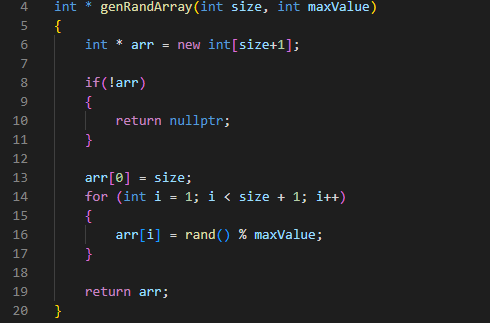
# **ВЫПОЛНЕНИЕ РАБОТЫ**

**Задание 1:**

Функцияint \* genRandArray(int size, int maxValue)

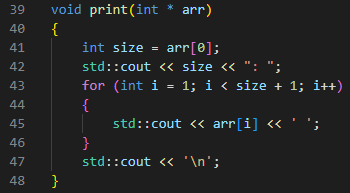
Функция принимает на вход размер массива и максимальное значение в массиве.

В теле функции выделяем память под элементы массива и еще один под размер массива, который нужно хранить в первом элементе массива. Если память выделилась, то возвращается указатель на сгенерированный массив, иначе возвращается нулевой указатель.

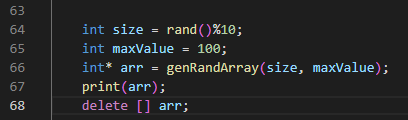


Функция void print(int \* arr)

Функция принимает на вход указатель на массив и выводит его размер и элементы массива.



В теле функции main вызваем две предыдущие функции и после освобождаем выделенную память на наш массив.



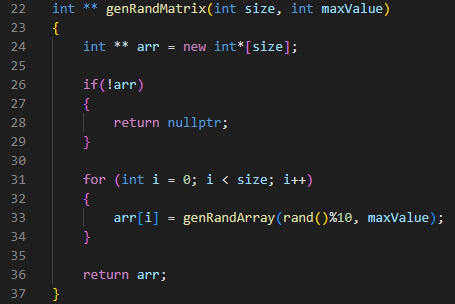
После запуска видим такой результат:



**Задание 2:**

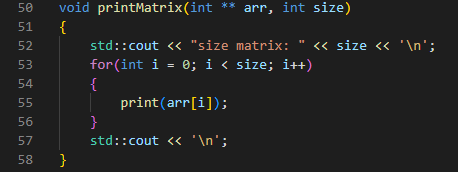
Функция int \*\* genRandMatrix(int size, int maxValue)

Функция принимает на вход количество строк в матрице и максимальное значение, которое будет в этой матрице. В теле функции выделяем массив указателей, после используем функцию из предыдущего задания и генерируем по массиву по адресу каждого указателя. Если память выделилась, то возвращаем указатель на матрицу, иначе нулевой указатель.

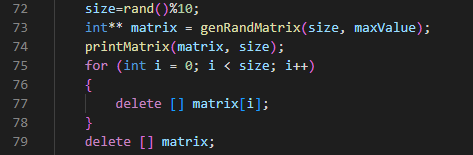


Функция void printMatrix(int \*\* arr, int size)

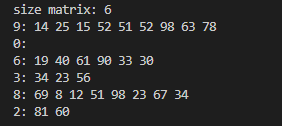
Функция принимает указатель на матрицу и количество строк в этой матрице. Далее в теле цикле вызываем функцию из предыдущего задания для вывода массива. По итогу получаем вывод всей матрицы.



В теле функции main вызваем две предыдущие функции и после освобождаем выделенную память на нашу матрицу.



После запуска программы видим такой результат:



# **ПРИЛОЖЕНИЕ**

Исходный код с комментариями:

**main.cpp**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82 | #include <iostream>  #include <cstdlib>  int \* genRandArray(int size, int maxValue)  {  if(!arr)      {          return nullptr;      }      int \* arr = new int[size+1];      arr[0] = size;      for (int i = 1; i < size + 1; i++)      {          arr[i] = rand() % maxValue;      }        return arr;  }  int \*\* genRandMatrix(int size, int maxValue)  {      int \*\* arr = new int\*[size];  if(!arr)      {          return nullptr;      }      for (int i = 0; i < size; i++)      {          arr[i] = genRandArray(rand()%10, maxValue);      }      return arr;  }  void print(int \* arr)  {      int size = arr[0];      std::cout << size << ": ";      for (int i = 1; i < size + 1; i++)      {          std::cout << arr[i] << ' ';      }      std::cout << '\n';  }  void printMatrix(int \*\* arr, int size)  {      std::cout << "size matrix: " << size << '\n';      for(int i = 0; i < size; i++)      {          print(arr[i]);      }      std::cout << '\n';  }  int main()  {      srand(time(NULL));      int size = rand()%10;      int maxValue = 100;      int\* arr = genRandArray(size, maxValue);      print(arr);      delete [] arr;      std::cout << "---------------\n";      size=rand()%10;      int\*\* matrix = genRandMatrix(size, maxValue);      printMatrix(matrix, size);      for (int i = 0; i < size; i++)      {          delete [] matrix[i];      }      delete [] matrix;      return 0;  } |