МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт цифровых технологий, электроники и физики

Кафедра вычислительной техники и электроники (ВТиЭ)

Лабораторная работа № 006

Программирование с использованием многомерных массивов

Выполнил студент 595 гр.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. В. Лаптев

Проверил: к.т.н,, доцент каф. ВТиЭ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ю.Г. Скурыдин

Лабораторная работа защищена

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Барнаул 2020

**ЗАДАНИЕ №1.**

**1. Формулировка задачи**

Дано натуральное число n и действительная матрица размерностью nx9. Найти среднее арифметическое каждого из столбцов, имеющих чётные номера.

**2. Постановка задачи**

На вход программы поступает целое положительное число n, которое определяет размерность массива по горизонтали (nx9). Действительные элементы матрицы генерируются случайным образом в 9 рядов, по n чисел в каждом. Среди всех столбцов действительной матрицы программа находит чётные и вычисляет средние арифметические значения для каждого чётного столбца в отдельности. На выход программы текстовым сообщением поступают вещественные числа, обозначающие средние арифметические значения чётных столбцов в количестве n / 2.

**3. Математическая модель**

Математическую модель данной программы удобнее всего рассмотреть на примере: у нас имеется матрица 8х9, которая создаётся случайным образом:

429.395 433.393 340.739 440.291 438.063 -228.431 -358.501 367.046

-155.675 -28.1075 -152.257 133.457 -30.427 -417.676 278.695 391.797

-472.427 -49.7452 -73.9769 -434.309 -449.477 -430.525 37.2021 226.02

-112.918 -289.071 -151.25 15.0151 54.1093 -7.416 -498.398 26.1238

-276.955 44.8927 -330.943 44.6181 -377.453 -71.2912 -71.0166 -443.861

138.005 460.891 411.42 -71.1386 389.966 104.526 -247.261 -324.137

71.9932 -419.935 -112.033 203.864 -170.11 -146.855 -288.339 -45.7472

462.813 -344.371 140.019 343.425 -271.92 196.387 -396.313 407.605

30.9763 217.261 -495.621 -440.352 -219.977 51.6068 -335.154 362.896

При помощи условия, путём серийности его проверки, осуществляется поиск всех чётных столбцов матрицы (будет проверяться возможность деления порядкового номера столбца матрицы на 2) и затем для каждого чётного столбца (в данном случае: 2, 4, 6, 8) будут выделены все его элементы (9 значений) и найдено среднее арифметическое для значений, представленных в столбце (т.е. сложены числа в столбце и разделены на 9). И эти значения среднего арифметического и будут выведены на экран. Т.е. в этом случае будут последовательно выведено 4 вещественных числа, по одному для каждого столбца:

Sr. arifmeticheskoe: 2.80092 (для (433.393 - 28.1075 - 49.7452 - 289.071 + 44.8927 + 460.891 - 419.935 - 344.371 + 217.261) / 9)

Sr. arifmeticheskoe: 26.0967 (для (440.291 + 133.457 - 434.309 + 15.0151 + 44.6181 - 71.1386 + 203.864 + 343.425 - 440.352) / 9)

Sr. arifmeticheskoe: -105.519 (для (-228.431 - 417.676 - 430.525 - 7.416 - 71.2912 + 104.526 - 146.855 + 196.387 + 51.6068) / 9)

Sr. arifmeticheskoe: 107.527 (для (367.046 + 391.797 + 226.02 + 26.1238 - 443.861 - 324.137 -45.7472 + 407.605 + 362.896) / 9).

**4. Описание алгоритма**

Начало

4.1 Инициализация переменных j = 1, i = 1 и инициализация переменной размерности массива: n = 8, задание массива matr[n][9]

4.2 Генерация начального случайного значения

Начало цикла (формирование матрицы по вертикали)

4.3 Проверка условия (i <= 9). Если условие не выполняется, то переход к пункту 4.8

Начало вложенного цикла (формирование матрицы по горизонтали)

4.4 Проверка условия (j <= n). Если условие не выполняется, то переход к пункту 4.8

4.5 Заполнение [i-го j-го] члена матрицы случайным образом и вывод его на экран

4.6 j++ и переход к пункту 4.4

Окончание вложенного цикла

4.7 i++ и переход к пункту 4.3

Окончание цикла

Начало цикла (нахождение чётных столбцов матрицы)

4.8 j = 1

4.9 Проверка условия (j <= n). Если условие не выполняется, то окончание работы программы

4.10 sum = 0

4.11 (j % 2 == 0). Если условие не выполняется, то окончание работы программы

Начало вложенного цикла (вычисление среднего арифметического чётных столбцов)

4.12 i = 1

4.13 Проверка условия (i <= 9). Если условие не выполняется, то окончание работы программы

4.14 Вычисление значения среднего арифметического для чётного столбца и вывод значения на экран

4.15 i++ и переход к пункту 4.12

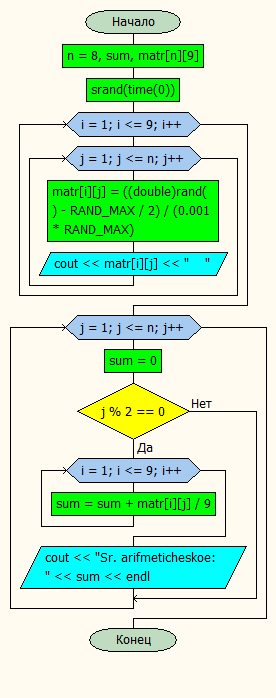
Окончание вложенного цикла

4.16 j++ и переход к пункту 4.8

Окончание цикла

Конец

**5. Опорный граф (блок-схема) алгоритма**



**6. Проект программы с определением замкнутых программных единиц и необходимых структур данных**

В данном случае программа должна состоять из четырех частей – блок ввода, блок вывода блок преобразований, который включает в себя 2 цикла, проверку условий и выполнение математических преобразований, а также блок в котором происходит задание и инициализация массива.

**7. Текст программы:**

**на языке С++**

#include <iostream>

#include <cmath>

#include <math.h>

#include <stdlib.h>

#include <ctime>

using namespace std;

//lab\_6\_1

int main()

{

const int n = 8;

double sum;

double matr[n][9]; // Инициализация и объявление переменных и массива

srand(time(0));

for (int i = 1; i <= 9; i++) // Генерация и вывод на экран матрицы

{

for (int j = 1; j <= n; j++)

{

matr[i][j] = ((double)rand() - RAND\_MAX / 2) / (0.001 \* RAND\_MAX);

cout << matr[i][j] << " ";

}

cout << endl;

}

for (int j = 1; j <= n; j++) // Вычленение чётных столбцов и нахождение их среднего арифметического

{

sum = 0;

if (j % 2 == 0) // Вычленение чётных столбцов

{

for (int i = 1; i <= 9; i++)

sum = sum + matr[i][j] / 9; // Нахождение среднего арифметического для каждого столбца

cout << "Sr. arifmeticheskoe: " << sum << endl;

}

}

return 0;

}

**на языке Pascal**

**Program lab\_6\_1;**

**const n = 8;**

**Var sum: real;**

**i, j: integer;**

**Var A: array[1..n + 1,1..10] of real;** // Инициализация и объявление переменных и массива

**Begin**

**randomize;**

**for i := 1 to 9 do** // Генерация и вывод на экран матрицы

**begin**

**for j := 1 to n do**

**begin**

**a[i, j] := random(-9999999999, 9999999999) / 100000;**

**Write (' ', a[i,j]);**

**end;**

**Writeln ();**

**end;**

**for j := 1 to n do**  // Вычленение чётных столбцов и нахождение их среднего арифметического

**begin**

**sum := 0;**

**if (j mod 2 = 0) then** // Вычленение чётных столбцов

**begin**

**for i := 1 to 9 do**

**sum := sum + a[i,j];**

**Writeln ('Sr. arifmeticheskoe: ', sum / 9);** // Нахождение среднего арифметического для каждого столбца

**end;**

**end;**

**End.**

**8. Проверка работоспособности (тестирование) программы**

Для проверки работоспособности запустим программу и из матрицы 8x9 будут найдены средние арифметические значения для 2, 4, 6, 8 столбцов. Точные значения надо проверять, т.к. матрица генерируется случайным образом.

**9. Сравнительный анализ и оценка эффективности работы программ на разных языках программирования**

В данной задаче программы написанные на разных языках программирования не отличны друг от друга. Единственным отличием, не влияющем на производительность, является различная организация, с точки зрения синтаксиса, случайной генерации символов.

**ЗАДАНИЕ №2.**

**1. Формулировка задачи**

Дана действительная квадратная матрица размерности 2n. Получить новую матрицу, выполнив перестановку её блоков размерности n x n, как показано на рисунке.

**2. Постановка задачи**

На вход программы поступает значение переменной n (вводится пользователем), при помощи которого определяется размерность матрицы 2n. Затем генерируется случайная действительная матрица 2n x 2n. И происходит последовательная перестановка блоков матрицы n x n таким образом, чтобы в итоге они переместились по часовой стрелке (см. рисунок). На выход программы поступает последовательность чисел (с теми же данными в блоках nxn) из 2n символов в каждой строке и этих строк тоже должно быть 2n, но уже с переставленными по часовой стрелке блоками.

**3. Математическая модель**

Решение данной задачи будет проще рассмотреть на примере реальной работы программы. Значение n = 2 (для того, чтобы перемещение блоков проще было отследить). При помощи генератора случайных чисел создаём случайное действительное значение для каждого элемента матрицы (сама матрица будет иметь размерность 4 и, соответственно, 16 значений). Затем происходит непосредственно выделение блоков n x n и взаимодействия с ними. А именно (в примере):

Исходная матрица:

-156.935 -132.563 253.896 863.125

560.351 -745.562 -891.351 269.753

458.365 -198.359 -678.943 543.982

-325.786 285.861 -702.252 480.068

от верхнего левого значения начинается перемещение (система сравнивает индексы значений матрицы до тех пор, пока условия этого сравнения выполняются) по матрице до 2 элемента матрицы (по индексам (координатам), которые имеет каждое значение матрицы, которые были использованы еще при создании этой матрицы, происходит наращивание индексов на значение переменной n, в данном случае на 2: matr[i][j] = temp), как по вертикали, так и по горизонтали. Когда перемещение завершится (перестанут выполняться условия для продолжения сравнения индексов), то эти 4 значения и будут выделены в 1 блок (2х2) (будут зафиксированы в памяти системы). В примере:

-156.93511 -132.56312 будут составлять левый верхний блок после проверки всех

560.35121 -745.56222 условий и наращивания всех необходимых индексов (индексы, которые имеет каждый элемент я указал более мелким шрифтом).

Затем индексы этого блока получают новые значения (увеличиваются на n в обе стороны: matr[i][j] = matr[i + n][j + n]), таким образом блок меняет своё местоположение внутри матрицы и становится на место правого нижнего блока (меняясь с ним местами). В примере:

-156.93533 -132.56334 получение новых индексов для блока (соответственно этот блок

560.35143 -745.56244 заменяет те значения, которые стояли на этих местах, а те значения в свою очередь получают те индексы, которые имел этот блок. На примере:

(изначально)

-678.94333 543.98234

-702.25243 480.06844

(после преобразований)

-678.94311 543.98212

-702.25221 480.06822

Далее в зависимости от условия происходит следующее: сначала, в случае если условие выполняется (исходя из начальных данных оно выполнится в первый раз), перемещение левого верхнего блока на место левого нижнего (temp = matr[i][j] – выделение блока 2х2, matr[i][j] = matr[i + n][j] – перемещение блока на место левого нижнего, а левого нижнего на место левого верхнего; в этом случае левый верхний блок – этот тот блок, который стоял на месте правого нижнего, но был поменян местами с изначально находившимся на своём месте левым верхним, алгоритм перемещения описан выше). На примере:

Преобразования для левого верхнего блока Преобразования для левого нижнего блока

(было) (было)

-678.94311 543.98212 458.36531 -198.35932

-702.25221 480.06822 -325.78641 285.86142

(стало) (стало)

-678.94331 543.98232 458.36511 -198.35912

-702.25241 480.06842 -325.78621 285.86122,

затем, если условие не выполняется, то происходит перемещение правого нижнего блока на место правого верхнего, а правый верхний, в свою очередь перемещается на место правого (по тем же правилам, что и в перемещении верхнего левого блока на место левого нижнего, а левого нижнего на место левого верхнего). На примере:

Преобразования для правого верхнего блока Преобразования для правого нижнего блока

(было) (было)

253.89613 863.12514 -156.93533 -132.56334

-891.35123 269.75324 560.35143 -745.56244

(стало) (стало)

253.89633 863.12534 -156.93513 -132.56314

-891.35143 269.75344 560.35123 -745.56224

И после этих действий блоки переместятся по часовой стрелке и займут места, необходимые по условию задачи. Затем полученная матрица будет выведена на экран. На примере:

458.36511 -198.35912 -156.93513 -132.56314

-325.78621 285.86122 560.35123 -745.56224

-678.94331 543.98232 253.89633 863.12534

-702.25241 480.06842 -891.35143 269.75344

Для большей понятности дополню словесное объяснение схемами (в этих схемах представлено перемещение не конкретных значений матрицы, а именно блоков nxn или, в примере, 2х2, т.к. условием задачи предусмотрено не перемещение конкретных одиночных элементов матрицы, а перемещение более обширных единиц указанной матрицы):

|  |  |
| --- | --- |
|  |  |
|  |  |

|  |  |
| --- | --- |
| 4 | 1 |
| 3 | 2 |

|  |  |
| --- | --- |
|  |  |
|  |  |

|  |  |
| --- | --- |
| 1 | 2 |
| 4 | 3 |

|  |  |
| --- | --- |
|  |  |
|  |  |

1. 2. 3. 4. 5.

**4. Описание алгоритма**

Начало

4.1 Инициализация переменных и объявление массива и ввод значения n пользователем: temp, n, matr[2 \* n][2 \* n], h, i, j

4.2 srand(time(0))

4.3 i = 0

Начало цикла (формирование матрицы по вертикали)

4.4 Проверка условия (i < 2 \* n). Если условие не выполняется, то переход к пункту 4.10

4.5 j = 0

Начало вложенного цикла (формирование матрицы по горизонтали)

4.6 Проверка условия (j < 2 \* n). Если условие не выполняется, то переход к пункту 4.9

4.7 Заполнение [i-го j-го] члена матрицы случайным образом и вывод его на экран

4.8 j++ и переход к пункту 4.6

Окончание вложенного цикла

4.9 i++ и переход к пункту 4.4

Окончание цикла

4.10 i = 0

Начало цикла

4.11 Проверка условия i < n. Если условие не выполняется, то переход к пункту 4.18

4.12 j = 0

Начало вложенного цикла

4.13 Присвоение переменной temp значения: temp = matr[i][j]

4.14 Смещение i-го j-го члена матрицы: matr[i][j] = matr[i + n][j + n]

4.15 Присвоение переменной temp нового значения матрицы: matr[i + n][j + n] = temp

4.16 j++ и переход к пункту 4.13

Окончание работы вложенного цикла

4.17 i++ и переход к пункту 4.11

Окончание работы цикла

4.18 l = 0

Начало цикла

4.19 Проверка условия l < 2. Если условие не выполняется, то переход к пункту 4.33

4.20 i = 0

Начало вложенного цикла

4.21 Проверка условия i < n. Если условие не выполняется, то переход к пункту 4.19

4.22 Проверка условия l == 0. Если условие не выполняется, то переход к пункту 4.25

4.23 j = 0

4.24 h = n и переход к пункту 4.27

4.25 j = n

4.26 h = 2 \* n

4.27 Проверка условия j < h. Если условие не выполняется, то переход к пункту 4.21

4.28 Присвоение переменной temp значения i-го j-го члена матрицы

4.29 Смещение i-го j-го члена матрицы на n по вертикали

4.30 Присвоение переменной temp нового значения матрицы

4.31 j++ и переход к пункту 4.27

Окончание работы вложенного цикла

Окончание работы цикла

4.32 i = 0

Начало цикла

4.33 Проверка условия i < 2 \* n. Если условие не выполняется, то происходит окончание работы программы

4.34 j = 0

Начало вложенного цикла

4.35 Проверка условия j < 2 \* n. Если условие не выполняется, то переход к пункту 4.33

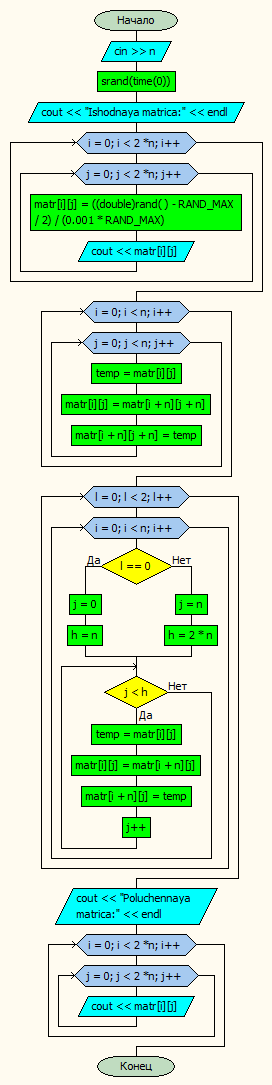
4.36 Вывод полученного значения матрицы на экран и переход к пункту 4. 35

Окончание работы вложенного цикла

Окончание работы цикла

Конец

**5. Опорный граф (блок-схема) алгоритма**



**6. Проект программы с определением замкнутых программных единиц и необходимых структур данных**

В данном случае программа должна состоять из четырёх частей – блок ввода, блок вывода и блок преобразований, включающий 4 цикла, в которых присутствует проверка условия на вхождение и которые необходимы для: генерации случайной матрицы, перестановки блоков матрицы по часовой стрелке (внутри цикла проверяется ряд условий необходимых для перестановки блоков) и вывода полученной матрицы на экран.

**7. Текст программы:**

**на языке С++**

#include <iostream>

#include <iomanip>

#include <cmath>

#include <math.h>

#include <stdlib.h>

#include <ctime>

using namespace std;

// lab\_6\_2

int main()

{

int n;

cout << "Vvedite n: ";

cin >> n;

double matr[2 \* n][2 \* n];

double temp;

srand(time(0));

cout << "Ishodnaya matrica:" << endl;

for (int i = 0; i < 2 \* n; i++) {

for (int j = 0; j < 2 \* n; j++) {

matr[i][j] = ((double)rand() - RAND\_MAX / 2) / (0.001 \* RAND\_MAX);

cout << setw(10) << matr[i][j];

}

cout << endl;

}

cout << endl;

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

temp = matr[i][j];

matr[i][j] = matr[i + n][j + n];

matr[i + n][j + n] = temp;

}

}

for (int l = 0; l < 2; l++) {

for (int i = 0; i < n; i++) {

int j, h;

if (l == 0) {

j = 0;

h = n;

}

else {

j = n;

h = 2 \* n;

}

while (j < h) {

temp = matr[i][j];

matr[i][j] = matr[i + n][j];

matr[i + n][j] = temp;

j++;

}

}

}

cout << "Poluchennaya matrica:" << endl;

for (int i = 0; i < 2 \* n; i++) {

for (int j = 0; j < 2 \* n; j++) {

cout << setw(10) << matr[i][j];

}

cout << endl;

}

return 0;

}

**на языке Pascal**

**Program** lab\_6\_2;

**const** n = 2;

**Var** temp: real;

i, j, h, l: integer;

**Var** A: **array**[0..2 \* n-1, 0..2 \* n-1] **of** real;

**Begin**

randomize;

Writeln('Ishodnaya matrica:');

**for** i := 0 **to** 2 \* n-1 **do** // Генерация и вывод на экран матрицы

**begin**

**for** j := 0 **to** 2 \* n-1 **do**

**begin**

a[i, j] := random(-9999999999, 9999999999) / 100000;

Write (' ', a[i,j]);

**end**;

Writeln ();

**end**;

Writeln();

**for** i := 0 **to** n-1 **do**

**begin**

**for** j := 0 **to** n-1 **do**

**begin**

temp := a[i, j];

a[i,j] := a[i + n, j + n];

a[i + n, j + n] := temp;

**end**;

**end**;

**for** l := 0 **to** 1 **do**

**begin**

**for** i := 0 **to** n-1 **do**

**begin**

**if** l = 0 **then**

**begin**

j := 0;

h := n;

**end**

**else**

**begin**

j := n;

h := 2 \* n;

**end**;

**for** j := (h-n) **to** h-1 **do**

**begin**

temp := a[i, j];

a[i, j] := a[i + n, j];

a[i + n, j] := temp;

**end**;

**end**;

**end**;

Writeln ('Poluchennaya matrica:');

**for** i := 0 **to** 2 \* n-1 **do** // Генерация и вывод на экран матрицы

**begin**

**for** j := 0 **to** 2 \* n-1 **do**

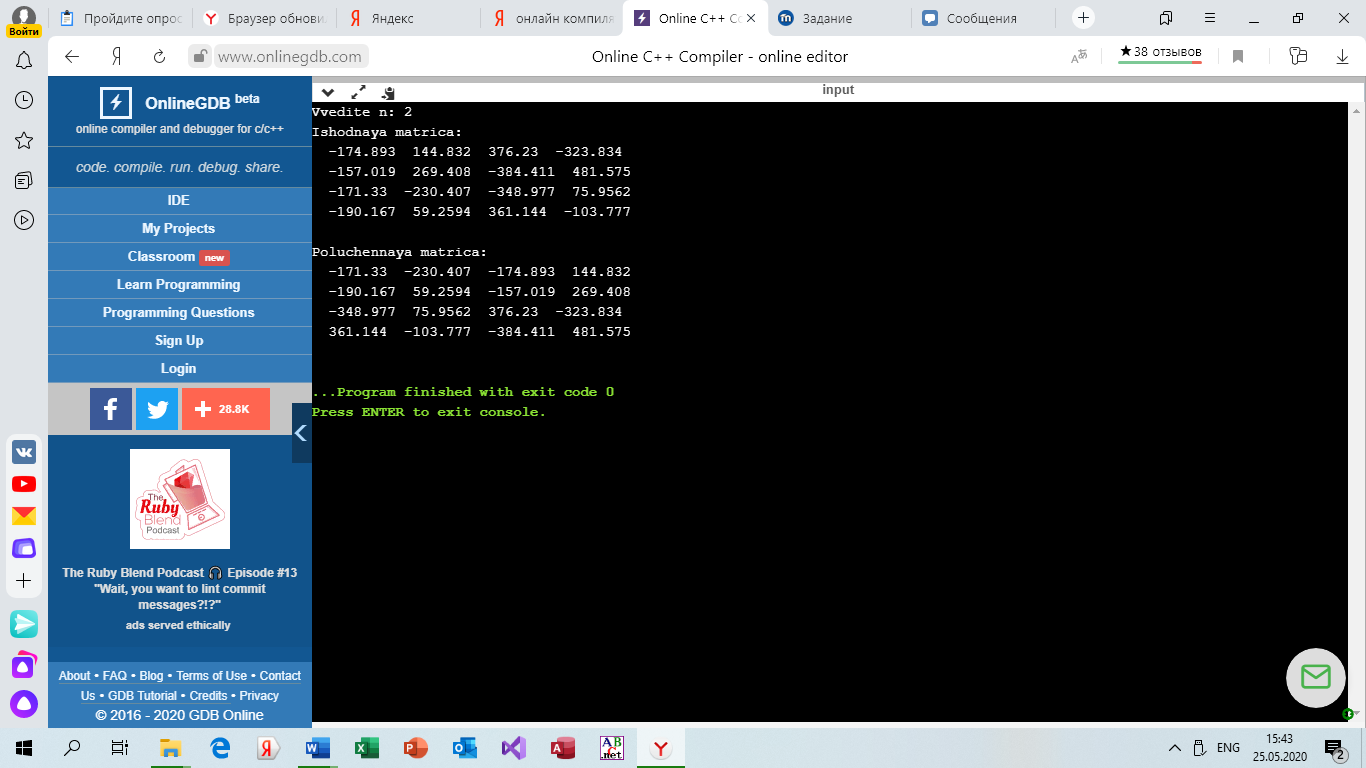
Write (' ', a[i,j]);

Writeln ();

**end**;

**End**.

**8. Проверка работоспособности (тестирование) программы**



**9. Сравнительный анализ и оценка эффективности работы программ на разных языках программирования**

В данной задаче программы написанные на разных языках программирования не отличны друг от друга. Единственным отличием, не влияющем на производительность, является различная организация, с точки зрения синтаксиса, случайной генерации символов.

1. Для многомерного C-массива выделяется единый блок памяти необходимого размера. Значения располагаются последовательно. Самый левый индекс массива изменяется медленнее всего.
2. С++: int mas[n][n][n]; Pascal: a:array[1..n, 1..n, 1..n] of integer;
3. Чтобы взаимодействовать с диагональными элементами матрицы необходимо взаимодействовать с индексами этих элементов, а индексы представляют собой координаты элемента внутри матрицы по вертикали и горизонтали, у диагональных элементов координаты будут совпадать, поэтому при взаимодействии с диагональными элементами нужно обращаться к совпадающим координатам.
4. Эта конструкция означает проверку выполняемости 2 условий одновременно: т.е. должно быть выполнено условие о том, что значение индекса по горизонтали должно быть меньше значения, которое получится при вычитания из значения, которое показывает ранг матрицы, значения индекса матрицы по вертикали, уменьшенного на 1, и одновременно с этим условием должно выполняться ещё и условие, при котором значение матрицы, которое получит такие индексы i, j не должно быть чётным. То есть манипуляции будут производиться, только с треугольной областью, которая будет занимать верхнюю часть матрицы, ограниченную её рангом.
5. Ранг матрицы – наивысший из порядков всевозможных ненулевых миноров.