МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра информационных технологий**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6**

**по дисциплине  
 «АЛГОРИТМЫ И АНАЛИЗ СЛОЖНОСТИ»**

Выполнил студент группы 25/2                                       А.А. Козин

Направление подготовки  02.03.03  Математическое обеспечение и администрирование информационных систем

Курс    2

Отчет принял доктор физико-математических наук, профессор                                                                                       А.И. Миков

Краснодар

2021 г.

**Задание**: Дана квадратная матрица размером nXn, нарисовать в ней заданный символ из 0 и 1, где 0 – фон, а 1 – описывают рисунок. Найти количество вхождений символа умножения () в матрице.

**Решение**.

Для нахождения количества вхождений напишем программу:

**Текст программы:**

#include <iostream>

using namespace std;

int\*\* createMatrix(int n)

{

int\*\* Matrix = new int\* [n];

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

{

Matrix[i] = new int[n];

}

return Matrix;

}

int\*\* requestMatrix(int\*\* Matrix, int n)

{

int i, j;

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

{

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

{

cin >> Matrix[i][j];

}

}

return Matrix;

}

int checkMatrix(int\*\* pobMatrix, int n)

{

int i, result = 1;

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

{

if (pobMatrix[i][i] != 1 || pobMatrix[n - i - 1][i] != 1) result = 0;

}

return result;

}

void printMatrix(int\*\* Matrix, int n)

{

int i, j;

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

{

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

{

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

}

cout << endl;

}

}

int createPobMatrix(int\*\* Matrix, int matrixSize, int n)

{

int\*\* pobMatrix = createMatrix(n);

int i, j, k, z, I, J, summ = 0;

for (i = 0; i <= matrixSize - n; i++)

{

for (j = 0; j <= matrixSize - n; j++)

{

I = 0; J = 0;

//Выше мы объявляем глобальную проходку по матрице, ниже - локальную, для задания побочной матрицы, которую мы и будем обследовать.

for (k = i; k < i + n; k++)

{

for (z = j; z < j + n; z++)

{

pobMatrix[I][J] = Matrix[k][z];

J++;

}

I++;

J = 0;

}

cout << i << j << endl;

printMatrix(pobMatrix, n);

cout << endl;

summ += checkMatrix(pobMatrix, n);

}

}

return summ;

}

void main()

{

setlocale(LC\_ALL, "RU");

int n, i;

cout << "Введите размер матрицы:" << endl;

cin >> n;

if (n >= 3) {

int\*\* Matrix = createMatrix(n);

cout << "Введите рисунок:" << endl;

Matrix = requestMatrix(Matrix, n);

int counter = 0;

for (i = 3; i <= n; i += 2)

{

counter += createPobMatrix(Matrix, n, i);

}

cout << "Минимальный размер: 3" << endl;

cout << "Максимальный размер: " << n << endl;

cout << "Ответ:" << counter;

}

else {

cout << "Минимальный размер матрицы должен быть 3" << endl;

}

}

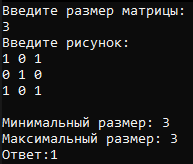
**Описание символа и его ограничения:**

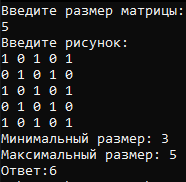
Символ умножения может быть представлен в 3 видах: \*, и точкой. В моем случае ().

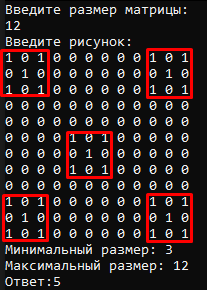
Символ умножения ограничен нижней границей матрицы, если размер матрицы сделать меньше 3, то символ не получиться нарисовать. т.е. меньше вот такого сделать нельзя:

Верхняя граница для символа умножения зависит от максимального размера матрицы. Например, для матрицы размером 7, верхняя граница будет равна 7:

**Программа выдает на консоль результаты в следующем виде:**

Символ умножения встречается на картинке 1 раз, и он во весь рисунок, поэтому количество вхождений равно 1.

В этом рисунке символ умножения встречается 6 раз.  
Первый раз в верхней левой части матрицы, второй раз в верхней правой части и так же снизу еще 2 раза.  
Получается 4, но знак умножения встречается еще во весь размер матрицы 5 и 6 раз в самой середине матрицы.

На последнем рисунке символ умножения встречается 5 раз, символы выделены красным цветом.