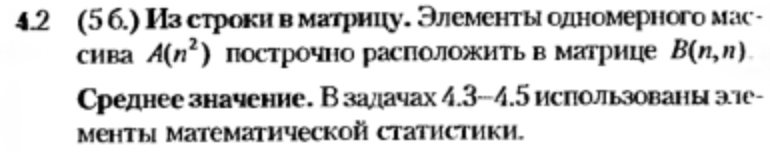
**Отчет курсанта Громова Григория Андреевича группы 22.Б05 о выполнении практического задания на 06.10.2022**

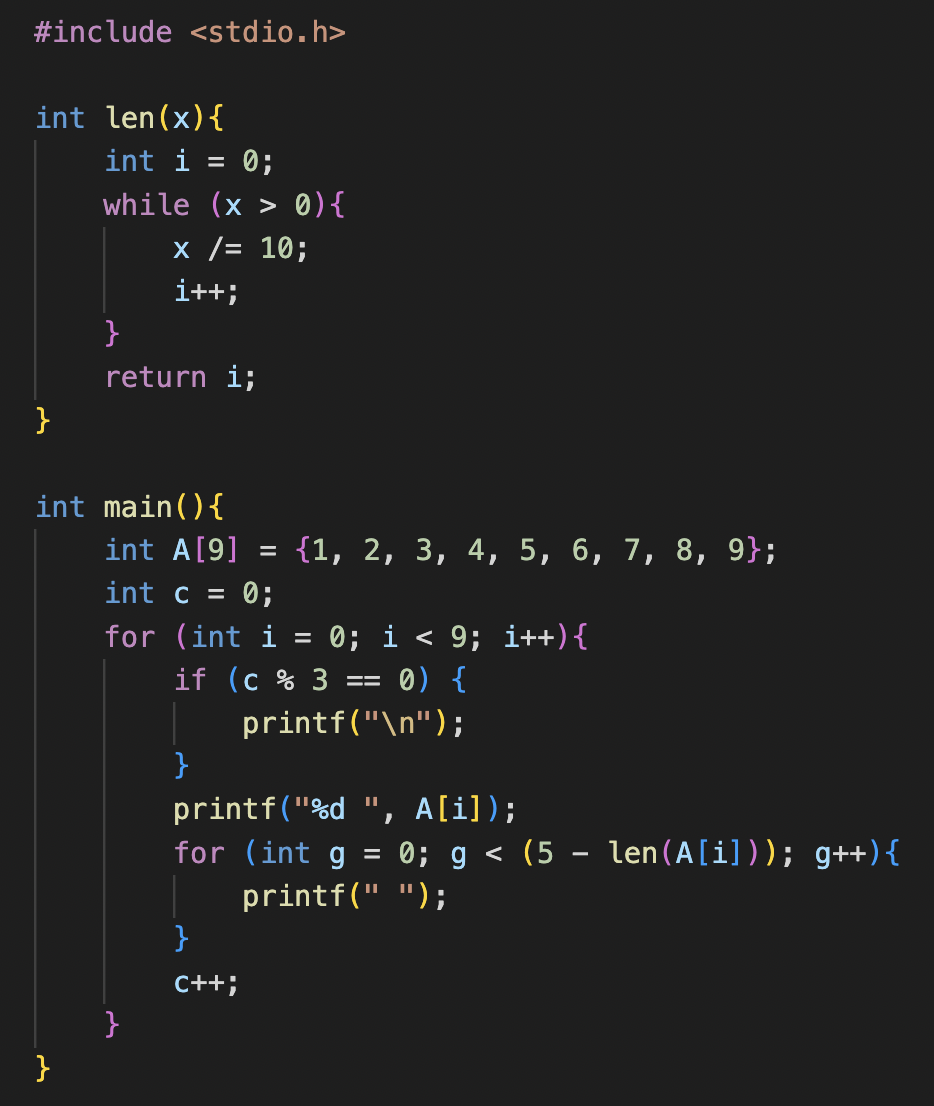
**Одномерные массивы (задания из Юркина)**



**Алгоритм:**

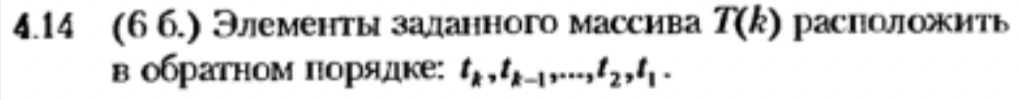
1. Вручную создаем массив, число элементов которого является квадратом произвольного натурального числа n
2. Создаем счетчик и устанавливаем его значение на нуле
3. Создаем цикл с n\*n итерациями
4. При каждом входе в цикл выписываем элемент массива в порядке его расположение в массиве, добавляя после него k пробелов (k = константа - количество разрядов выписываемого элемента массива). Переводим курсор на новую строку с каждой итерацией кратной n.

**Программа:**



**Текст программы:**

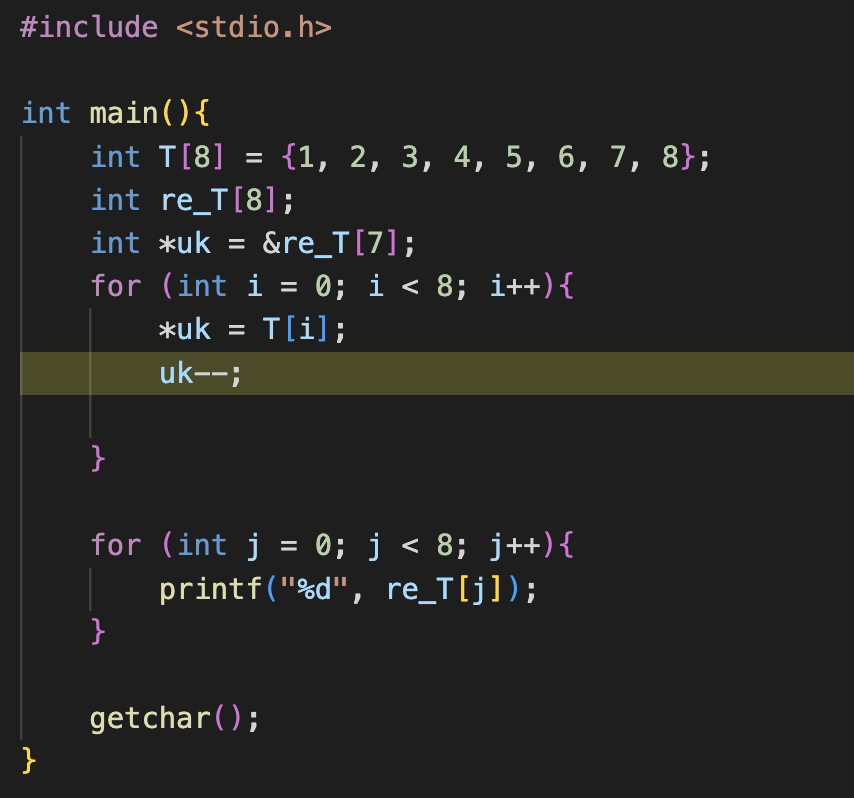
[#include](https://vk.com/im?sel=363705479&st=%23include) <stdio.h>  
  
int len(x){  
int i = 0;  
while (x > 0){  
x /= 10;  
i++;  
}  
return i;  
}  
  
int main(){  
int A[9] = {1, 2, 3, 4, 5, 6, 7, 8, 9};  
int c = 0;  
for (int i = 0; i < 9; i++){  
if (c % 3 == 0) {  
printf("\n");  
}  
printf("%d ", A[i]);  
for (int g = 0; g < (5 - len(A[i])); g++){  
printf(" ");  
}  
c++;   
}   
}

****

**Алгоритм:**

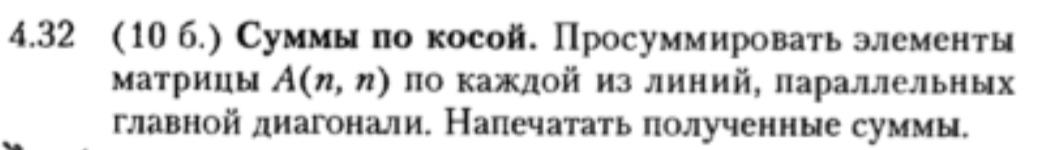
1. Создаем произвольный массив T длины n
2. Создаем пустой массив re\_T длины n
3. Создаем указатель uk и присваиваем ему адрес последней ячейки массива re\_T
4. С помощью цикла перебираем все элементы массива T. Получив элемент записываем его в ячейку, на которую указывает uk. После этого уменьшаем uk на 1, тем самым «направляя» его на предыдущую ячейку массива re\_T
5. Выводим на экран массив re\_T

**Программа:**



**Текст программы:**

[#include](https://vk.com/im?sel=363705479&st=%23include) <stdio.h>  
  
int main(){  
int T[8] = {1, 2, 3, 4, 5, 6, 7, 8};  
int re\_T[8];  
int \*uk = &re\_T[7];  
for (int i = 0; i < 8; i++){  
\*uk = T[i];  
uk--;  
  
}  
  
for (int j = 0; j < 8; j++){  
printf("%d", re\_T[j]);  
}  
  
getchar();  
}



**Алгоритм:**

1. Зададим произвольный двумерный массив M[n][n] (для примера возьмем n = 4)
2. Рассмотрим соответствие индексации и чисел в матрице

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 |

|  |  |  |  |
| --- | --- | --- | --- |
| 00 | 01 | 02 | 03 |
| 10 | 11 | 12 | 13 |
| 20 | 21 | 22 | 23 |
| 30 | 31 | 32 | 33 |

То есть, по сути, нам нужно получить наборы индексов:

00 11 22 33

01 12 23

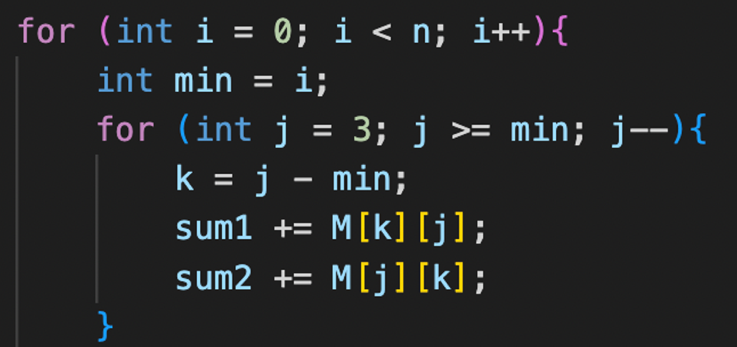
02 13

03

10 21 32

20 31

30

1. Подберём формулу, которая будет возвращать эти числа):

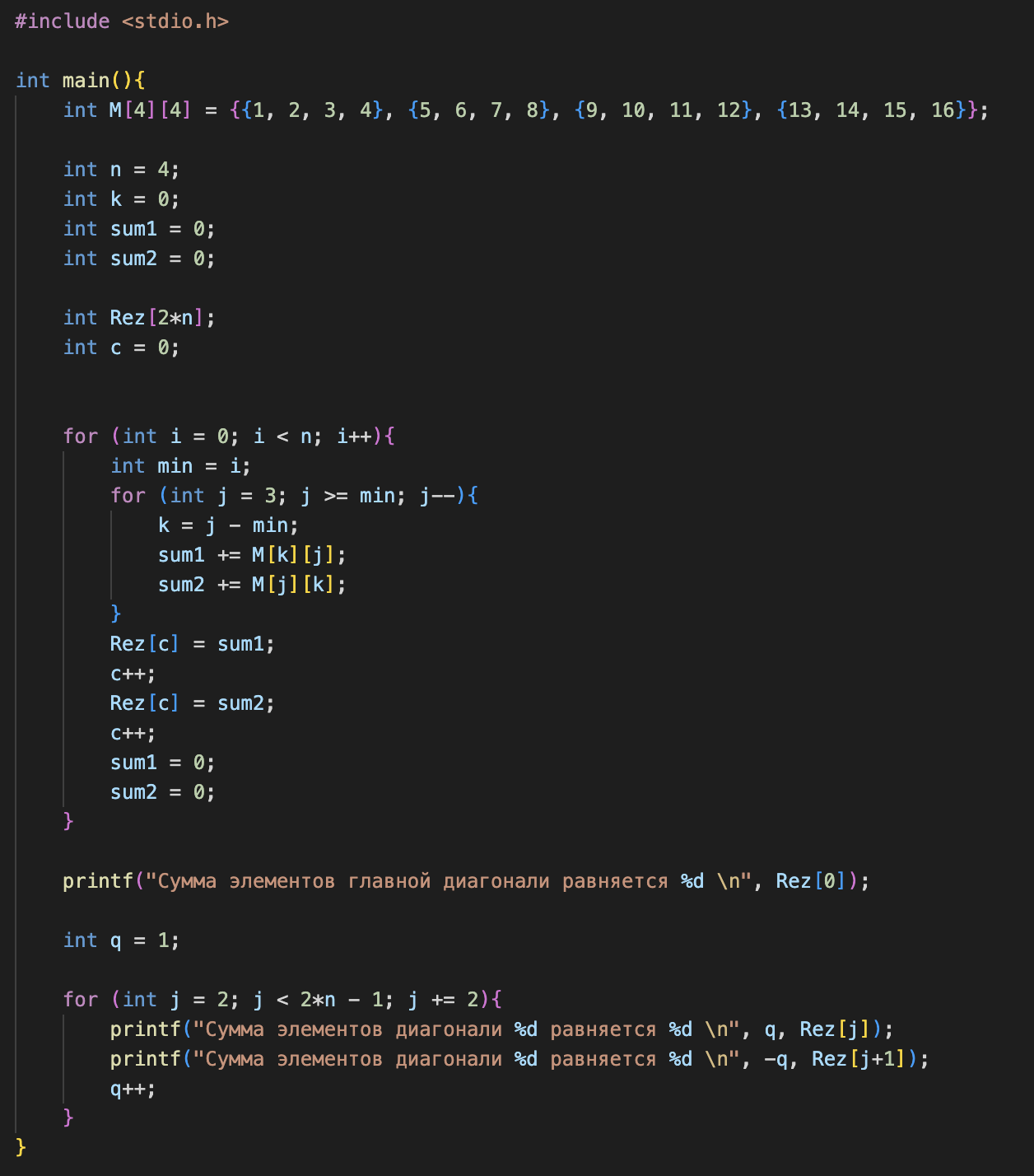
n – размерность матрицы

sum1 – сумма элементов итой диагонали сверху от главной (01 12 23, 02 13, 03)

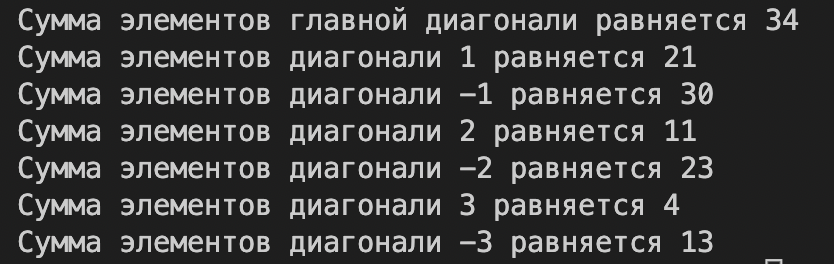
sum2 - итая диагональ снизу от главной (10 21 32, 20 31, 30)

1. Запомним sum1 и sum2 в заранее подготовленный массив Rez
2. Выводим значения на экран в удобном формате

**Программа:**

****

После запуска программы в терминал выводится:



**Текст программы:**

[#include](https://vk.com/im?sel=363705479&st=%23include) <stdio.h>  
  
int main(){  
int M[4][4] = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}, {13, 14, 15, 16}};  
  
int n = 4;  
int k = 0;  
int sum1 = 0;   
int sum2 = 0;  
  
int Rez[2\*n];  
int c = 0;  
  
  
for (int i = 0; i < n; i++){  
int min = i;  
for (int j = 3; j >= min; j--){  
k = j - min;  
sum1 += M[k][j];  
sum2 += M[j][k];  
}  
Rez[c] = sum1;  
c++;  
Rez[c] = sum2;  
c++;  
sum1 = 0;  
sum2 = 0;  
}  
  
printf("Сумма элементов главной диоганали равняется %d \n", Rez[0]);  
  
int q = 1;  
  
for (int j = 2; j < 2\*n - 1; j += 2){  
printf("Сумма элементов диоганали %d равняется %d \n", q, Rez[j]);  
printf("Сумма элементов диоганали %d равняется %d \n", -q, Rez[j+1]);  
q++;  
}  
}