**Отчет курсанта Громова Григория Андреевича группы 22.Б05 о выполнении практического задания на 13.10.2022**

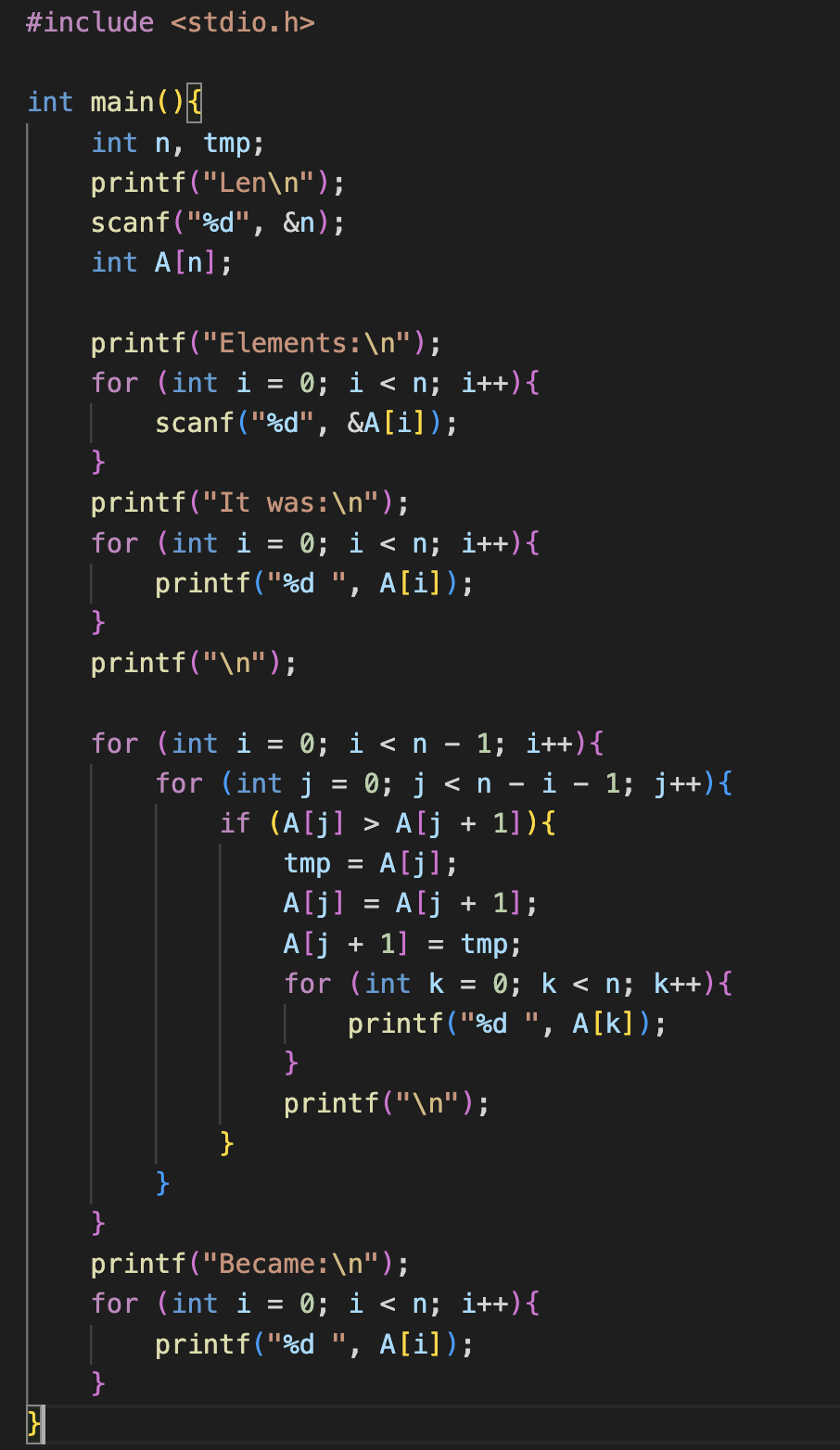
**Сортировки:**

1. Пузырьком

**Алгоритм:**

1. Спрашиваем у пользователя, сколько чисел он собирается ввести
2. Записываем полученное значение в переменную n
3. Создаем массив A[] длинной n
4. N раз запрашиваем у пользователя по одному числу ряд чисел, который пользователь хочет отсортировать по возрастанию
5. Последовательно записываем их в массив A в порядке, в котором они были введены
6. N – 1 раз проходимся по массиву, каждый раз рассматривая по порядку соседние пары чисел: если левое больше правого: меняем их местами, иначе: ничего не делаем. Причем с каждым новым разом рассматриваем на одну пару меньше (мы можем так поступать, потому что с первым проходом мы гарантированно перенесем самое больше число на последнюю позицию, со следующим второе по величине число на предпоследнюю позицию и тд. Таким образом рассматривать крайние правые места становится бессмысленным, ведь там и так уже стоят самые больше числа)
7. Выводим отсортированный массив

**Программа:**



**Текст программы:**

#include <stdio.h>

int main(){

int n, tmp;

printf("Len\n");

scanf("%d", &n);

int A[n];

printf("Elements:\n");

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

scanf("%d", &A[i]);

}

printf("It was:\n");

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

printf("%d ", A[i]);

}

printf("\n");

for (int i = 0; i < n - 1; i++){

for (int j = 0; j < n - i - 1; j++){

if (A[j] > A[j + 1]){

tmp = A[j];

A[j] = A[j + 1];

A[j + 1] = tmp;

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

printf("%d ", A[k]);

}

printf("\n");

}

}

}

printf("Became:\n");

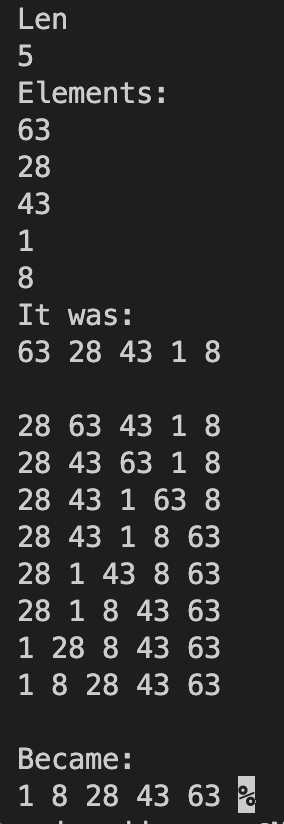
for (int i = 0; i < n; i++){

printf("%d ", A[i]);

}

}

**Пример:**

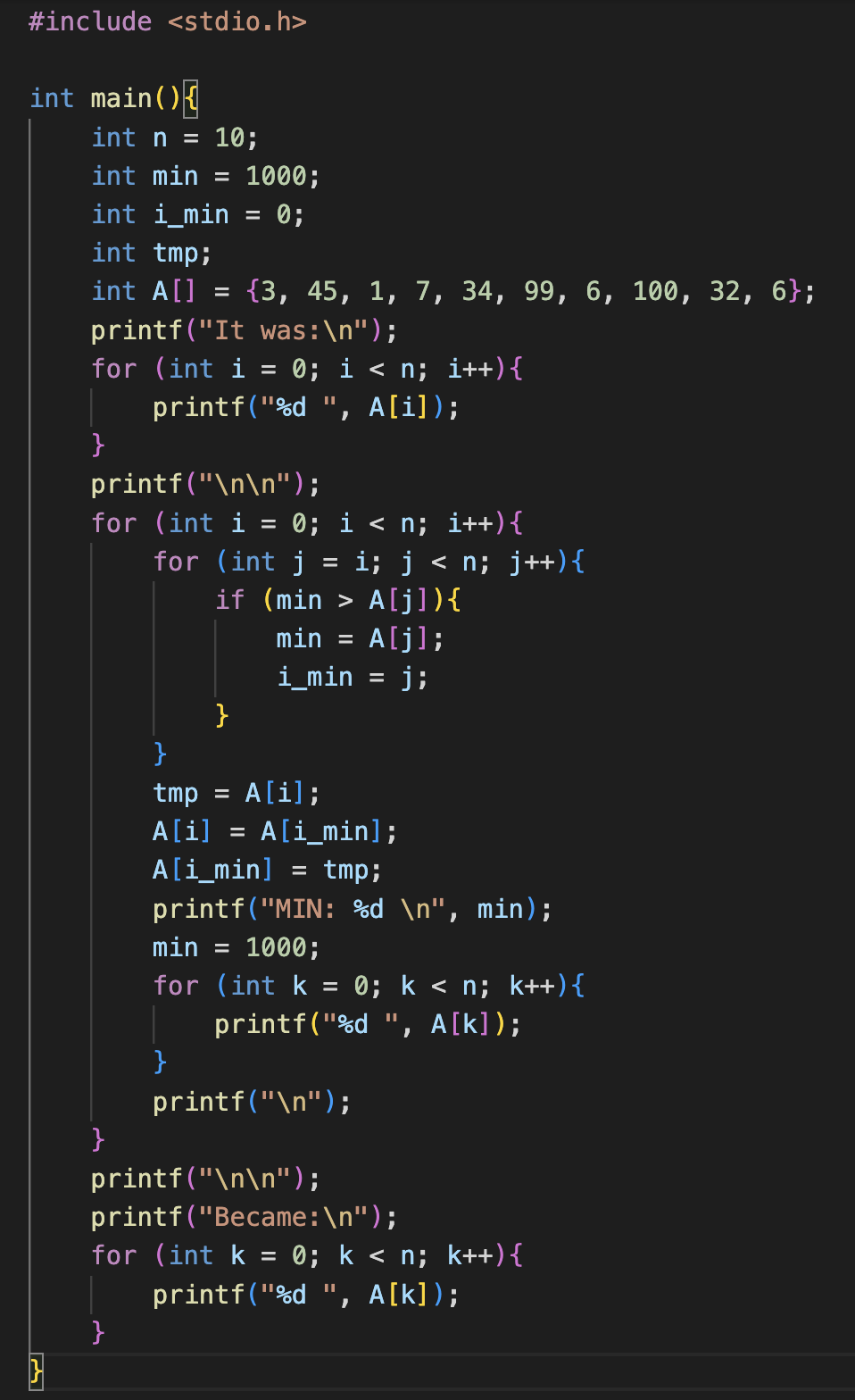


1. Обменом

**Алгоритм:**

1. Заводим цикл с количеством повторений равным длине массива A
2. Далее рассматриваем подмножество массива от A[i] до A[n-1]
3. Ищем в нем наименьший элемент перебором и меняем местами с элементом A[i]
4. Делаем то же самое с каждым новым циклом, отнимая от подмножества массива элемент с индексом **i – 1** (то есть первый элемент подмножества массива в предыдущем цикле) до тех пор, пока это подмножество не станет пустым

**Программа:**



**Текст программы:**

#include <stdio.h>

int main(){

int n = 10;

int min = 1000;

int i\_min = 0;

int tmp;

int A[] = {3, 45, 1, 7, 34, 99, 6, 100, 32, 6};

printf("It was:\n");

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

printf("%d ", A[i]);

}

printf("\n\n");

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

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

if (min > A[j]){

min = A[j];

i\_min = j;

}

}

tmp = A[i];

A[i] = A[i\_min];

A[i\_min] = tmp;

printf("MIN: %d \n", min);

min = 1000;

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

printf("%d ", A[k]);

}

printf("\n");

}

printf("\n\n");

printf("Became:\n");

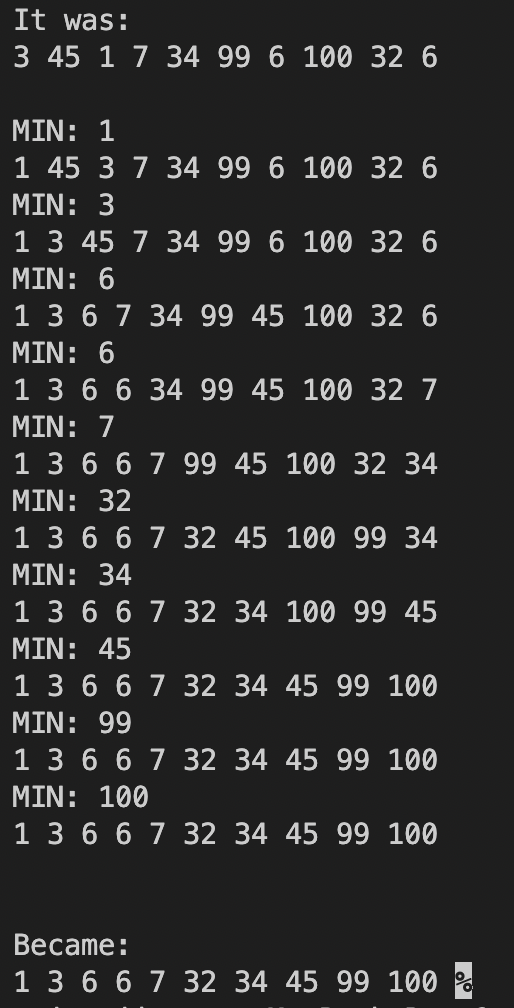
for (int k = 0; k < n; k++){

printf("%d ", A[k]);

}

}

**Терминал:**

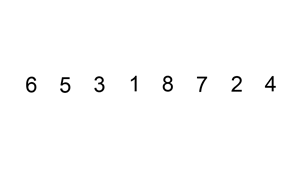
****

1. **Вставкой**

**Суть сортировки:**

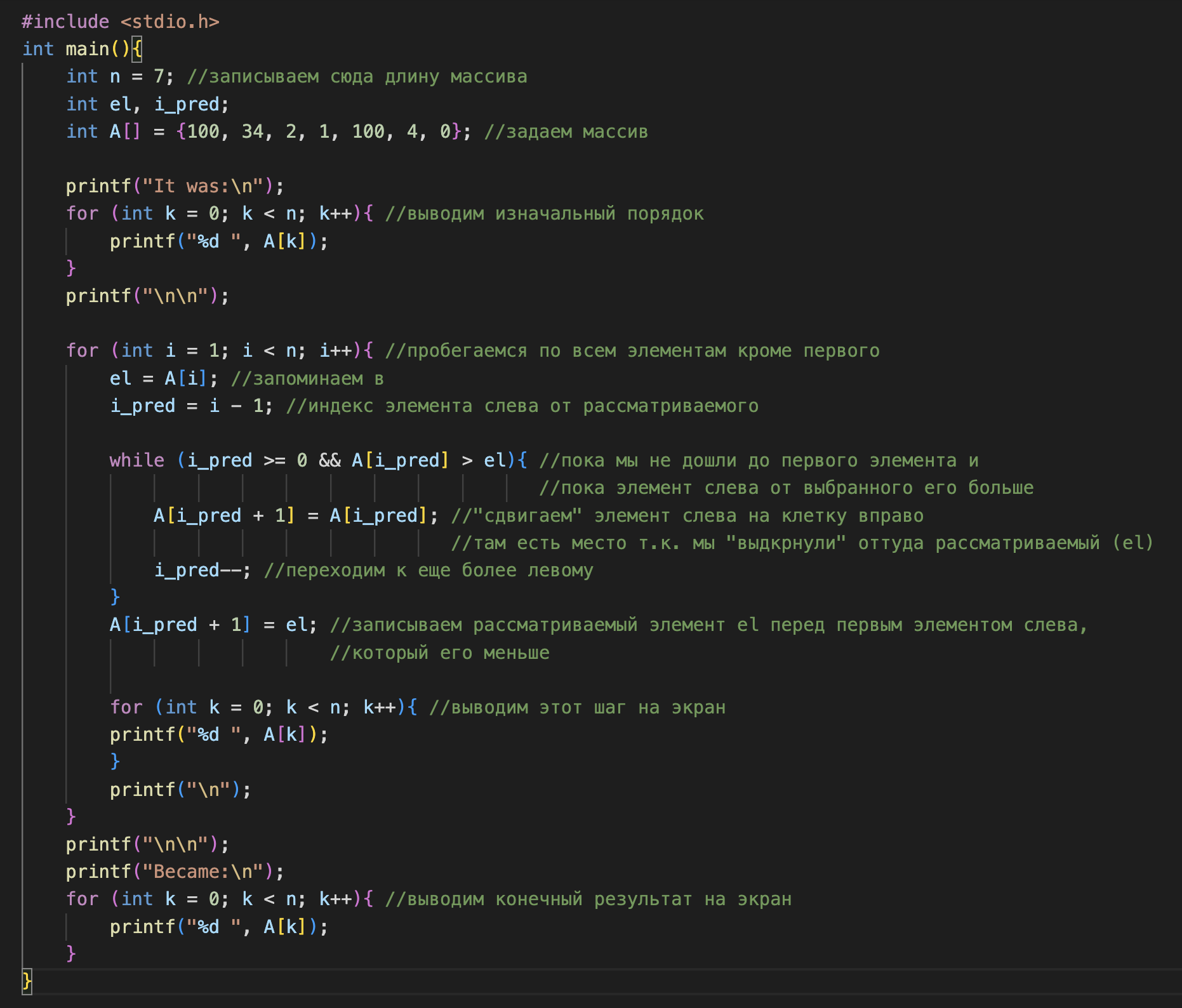
Пробегаемся по всем элементам, начиная со второго, и сравниваем их с уже отсортированным ранее подмножеством массива, которое располагается слева от рассматриваемого элемента и «вклиниваем» его перед первым справа налево элементом, который его меньше.

**Иллюстрация:**

****

**Алгоритм:** см. в комментариях в пункте «Программа»

**Программа:**

****

**Текст программы:**

#include <stdio.h>

int main(){

int n = 7;

int el, i\_pred;

int A[] = {100, 34, 2, 1, 100, 4, 0};

printf("It was:\n");

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

printf("%d ", A[k]);

}

printf("\n\n");

for (int i = 1; i < n; i++){

el = A[i];

i\_pred = i - 1;

while (i\_pred >= 0 && A[i\_pred] > el){

A[i\_pred + 1] = A[i\_pred];

i\_pred--;

}

A[i\_pred + 1] = el;

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

printf("%d ", A[k]);

}

printf("\n");

}

printf("\n\n");

printf("Became:\n");

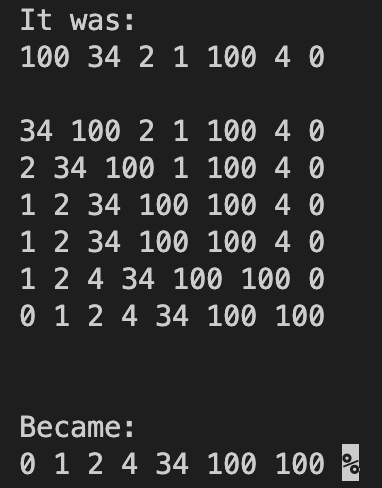
for (int k = 0; k < n; k++){

printf("%d ", A[k]);

}

}

Терминал:



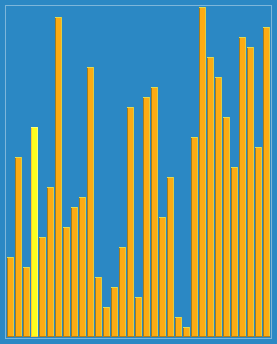
**Дополнительные сортировки:**

1. Гномья

**Суть сортировки:**

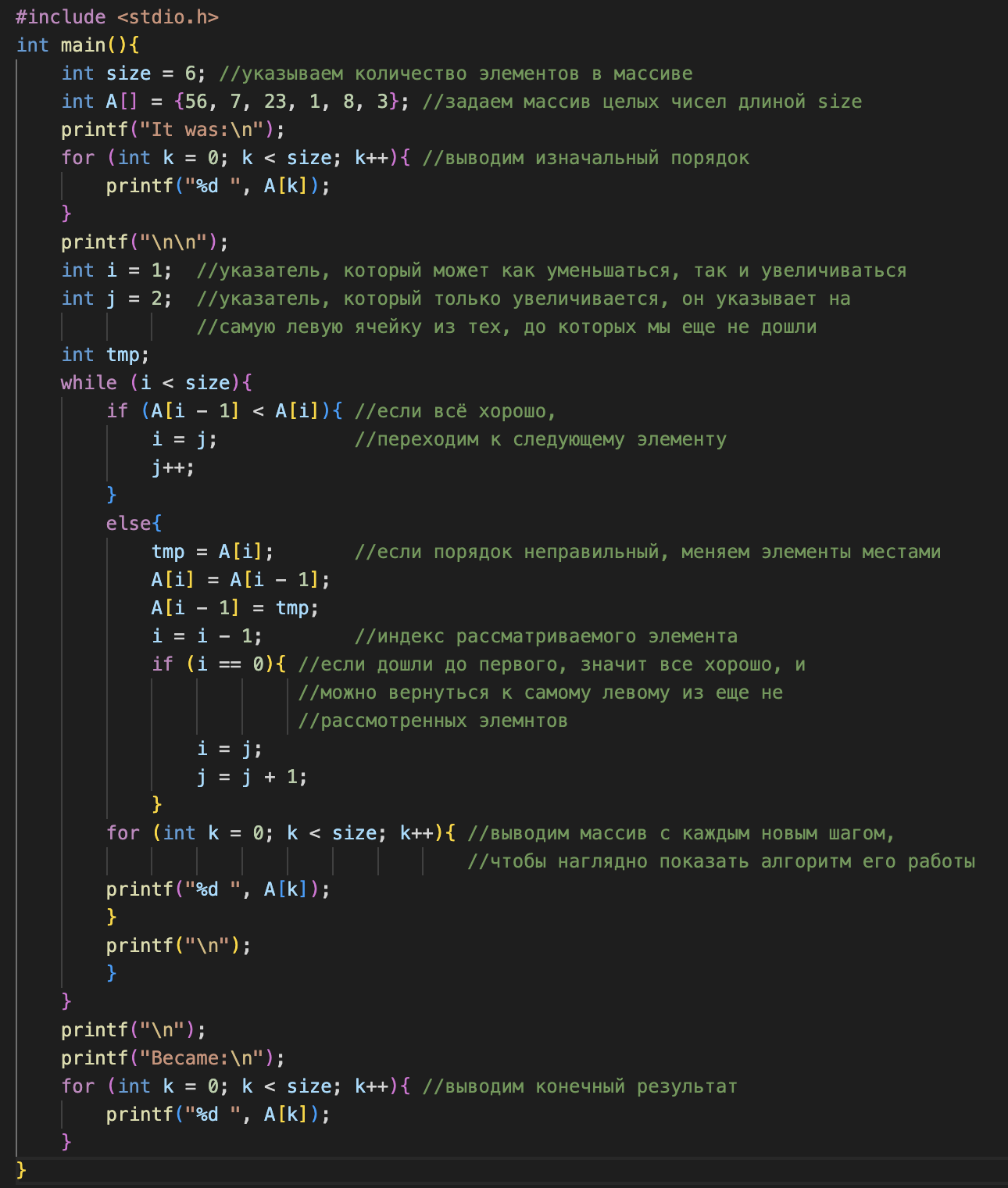
Это нечто среднее между сортировкой пузырьком и сортировкой вставкой. Рассмотрим версию алгоритма, которая сортирует числа от меньшего к большему. Мы находим первый элемент, который меньше того, что слева от него, и меняем его с левым, до тех пор, пока он не станет больше левого от него.

**Иллюстрация:**



**Алгоритм:**

см. в комментариях в пункте «Программа»

****

**Текст программы:**

#include <stdio.h>

int main(){

int size = 6;

int A[] = {56, 7, 23, 1, 8, 3};

printf("It was:\n");

for (int k = 0; k < size; k++){

printf("%d ", A[k]);

}

printf("\n\n");

int i = 1;

int j = 2;

int tmp;

while (i < size){

if (A[i - 1] < A[i]){

i = j;

j++;

}

else{

tmp = A[i];

A[i] = A[i - 1];

A[i - 1] = tmp;

i = i - 1;

if (i == 0){

i = j;

j = j + 1;

}

for (int k = 0; k < size; k++){

printf("%d ", A[k]);

}

printf("\n");

}

}

printf("\n");

printf("Became:\n");

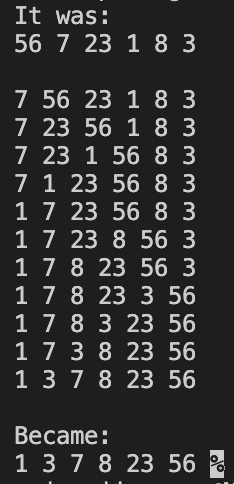
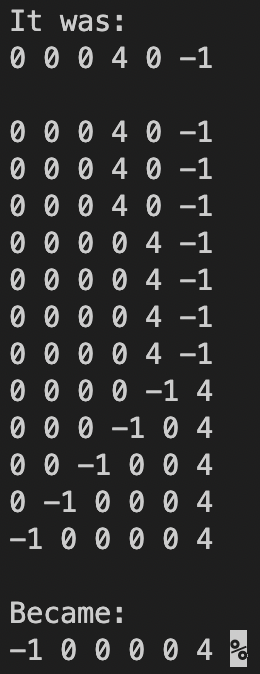
for (int k = 0; k < size; k++){

printf("%d ", A[k]);

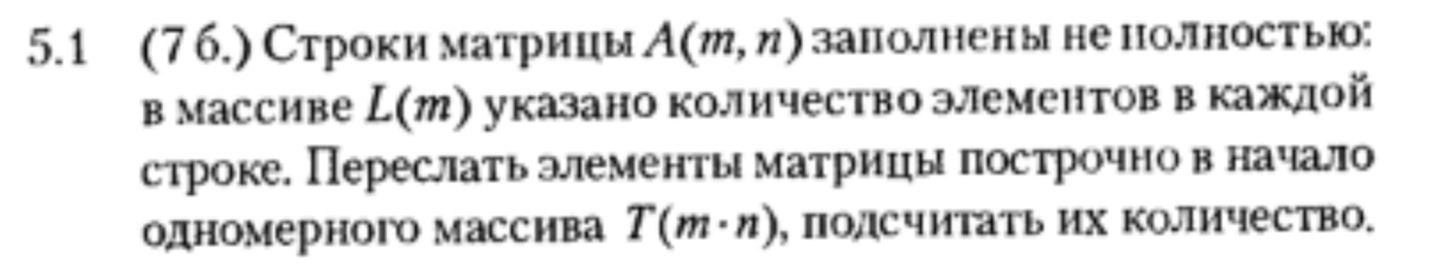
}

}

**Терминал:**

**   **

**Задачи на многомерные массивы**

****

Я посчитал наличие массива L(m) избыточным для реализации программы и, чтобы не усложнять структуру программы, при написании игнорировал его существование.

Замечание: так как в моей реализации пользователь вручную вводит матрицу построчно необходимо было ввести символ, который бы означал пустую ячейку. Зарезервируем под эти нужды цифру “0”, и примем, что пользователь может заполнять ячейки матрицы любыми целыми числами кроме 0.

Математическая модель: очевидна

Алгоритм изложен в комментариях к тексту программы:

#include <stdio.h>

#include <stdlib.h>

int main(){

int m; //число строк

int n; //число столбцов

//инициализируем переменные:

printf("Введите количество строк\n");

scanf("%d", &m);

printf("Введите количество столбцов\n");

scanf("%d", &n);

int A[m][n]; //создаем массив

int counter = 0; //количество непустых ячеек

int a, b; //вспомогательные счетчики для циклов

//заполним массив

for (a = 0; a < m; a ++){

printf("Введите элементы строки %d, вводя вместо пропуска '0': \n", a + 1);

for (b = 0; b < n; b++){

scanf("%d", &A[a][b]); //считываем введенное пользователем число и записываем

//его в соответствующую порядку ввода ячейку массива A

if (A[a][b] != 0){

counter++; //если ячейка непустая, то увеличиваем счетчик

}

}

}

int T[counter]; //создаем массив, в который будем записывать ненулевые элементы матрицы

int c = 0; //счетчик для перемещения по массиву T

//перешлем построчно ненулевые элементы матрицы подряд в массив T

for (a = 0; a < m; a ++){

for (b = 0; b < n; b++){

if (A[a][b] != 0){

T[c] = A[a][b];

c++;

}

}

}

printf("\n");

printf("Вот элементы вашей матрицы:");

//выведем T

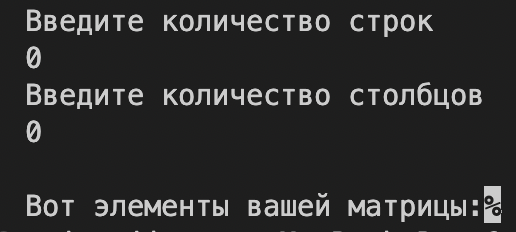
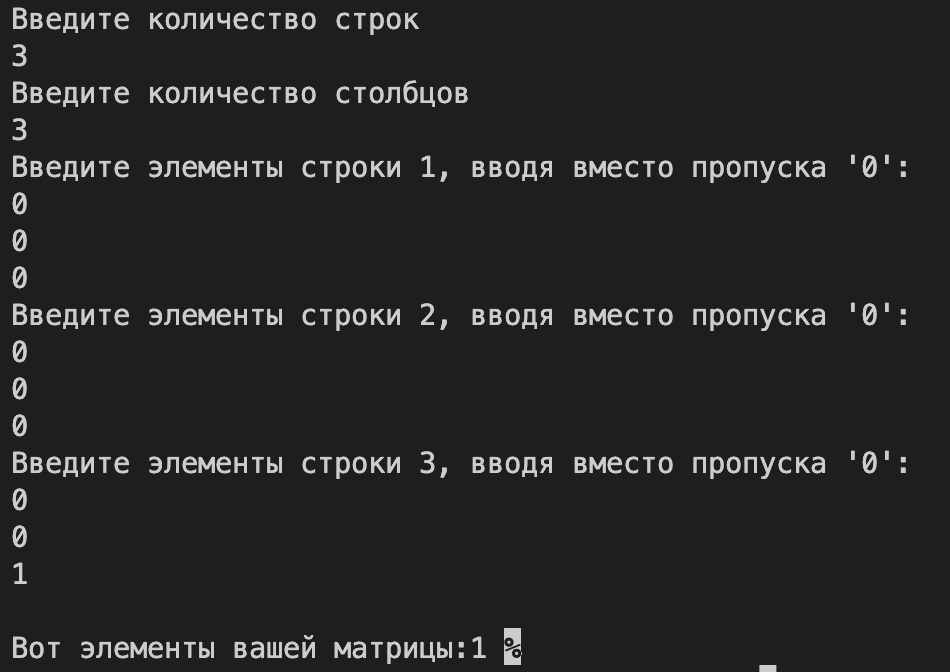
for (a = 0; a < counter; a++){

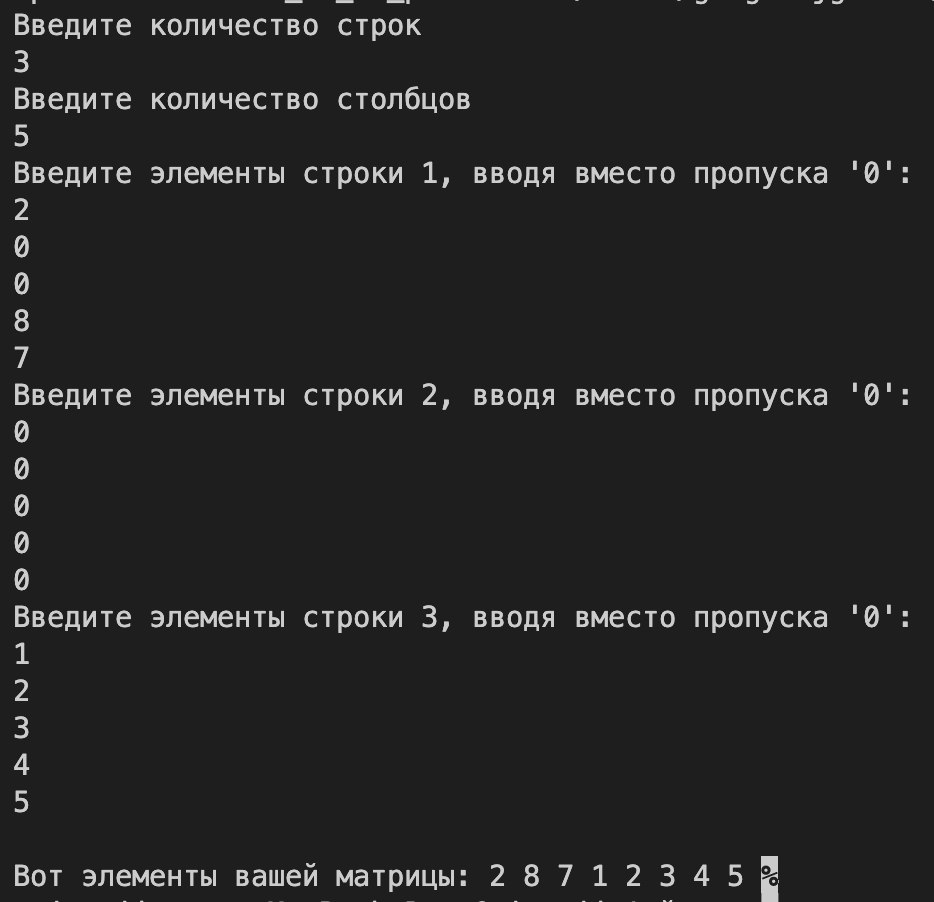
printf("%d ", T[a]);

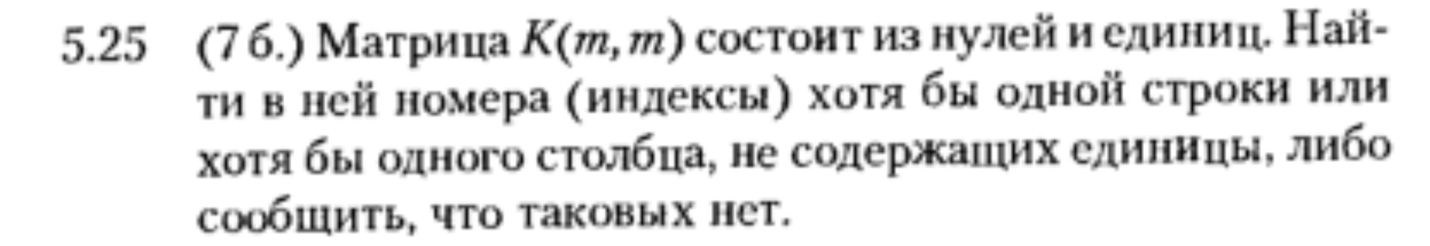
}

}

Скриншоты результатов:







Математическая модель: очевидна

Алгоритм изложен в комментариях к тексту программы:

#include <stdio.h>

#include <stdlib.h>

int main(){

int m; //число строк и столбцов

//инициализируем переменную:

printf("Введите размер квадратной матрицы\n");

scanf("%d", &m);

int A[m][m]; //создаем массив под эту матрицу

int a, b; //вспомогательные счетчики для циклов

//заполним массив через клавиатуру

for (a = 0; a < m; a ++){

printf("Введите элементы строки %d\n", a + 1);

for (b = 0; b < m; b++){

scanf("%d", &A[a][b]); //считываем введенное пользователем число и записываем

//его в соответствующую порядку ввода ячейку массива A

}

}

printf("Вот ваша матрица:");

for (a = 0; a < m; a ++){ //выводим матрицу в терминал

for (b = 0; b < m; b++){

printf("%d ", A[a][b]);

}

printf("\n");

}

printf("\n");

int counter = 0; //счетчик количества нулей в строке

int t = 0; //переменная, в которую мы будем записывать номер строки, где нет единиц

int flag = 0;

for (a = 0; a < m; a ++){

for (b = 0; b < m; b++){

if (A[a][b] == 0){

counter++; //если встретили 0, увеличиваем счетчик на 1

}

}

if (counter == m){ //если количество нулей в строке совпало с количество

//элементов в строке, значит все элементы нули, и эта

//строка нам подходит

t = a + 1; //запишем ее номер

flag = 1; //запишем, что мы нашли искомую строку

}

counter = 0; //обнулим счетчик

if (flag == 1){

break; //выйдем из цикла, если искомая строка найдена (так как в условии

// сказано, что достаточно найти хотя бы одну)

}

}

if (t > 0){ //если хоть бы одна строка была, то t было присвоено какое-то положительно значение

printf("Строка %d не содержит единиц", t);

}

else { //если нет, значит t не изменялось, значит ни одной такой строки найден не было

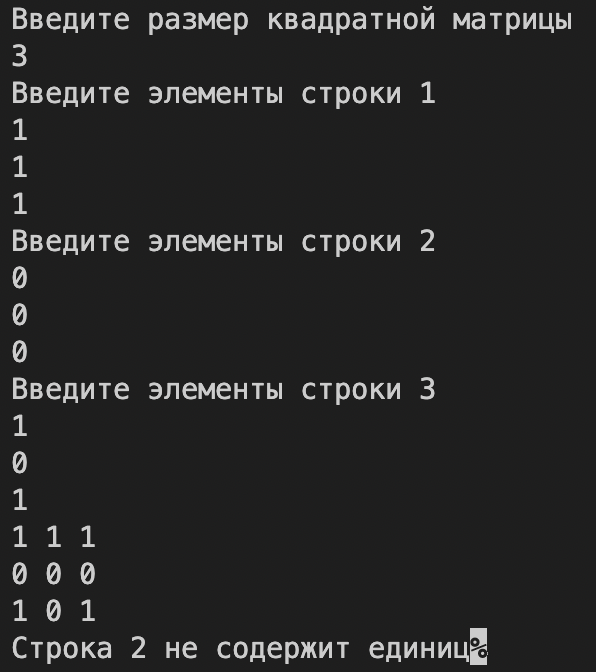
printf("Не нашлось строки, не содержащей ни одной единицы");

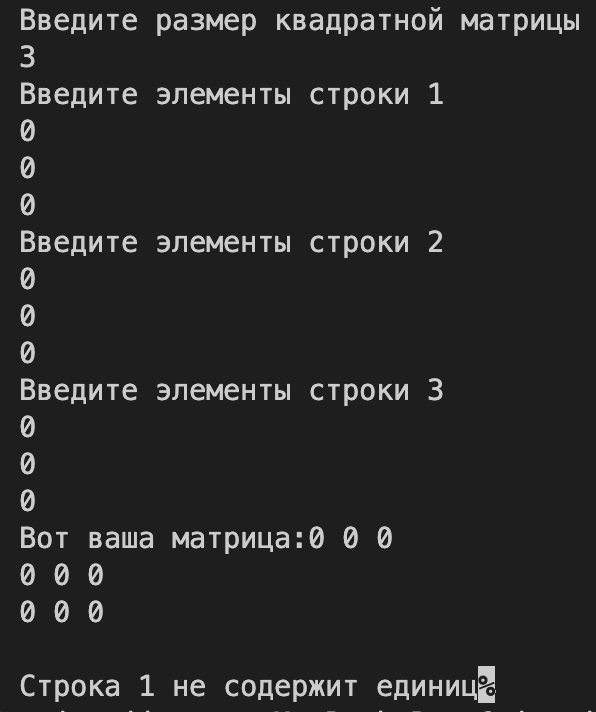
}

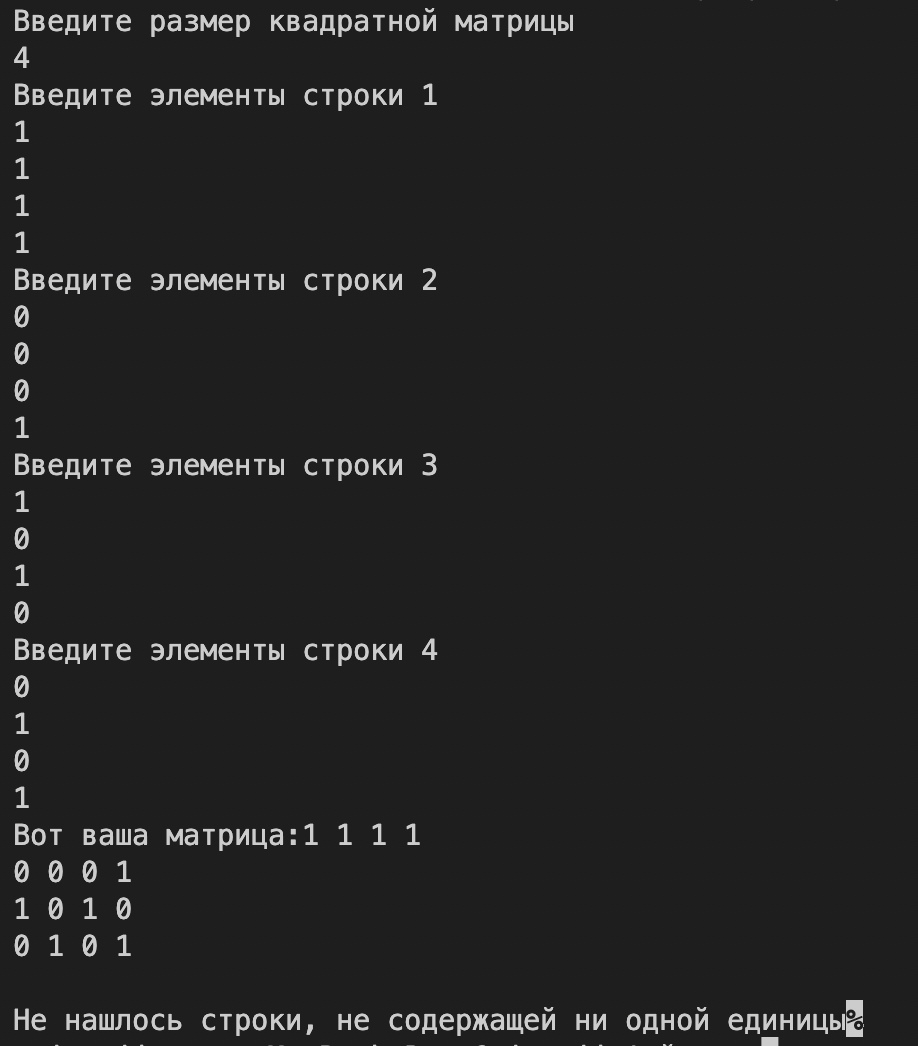
return 0;

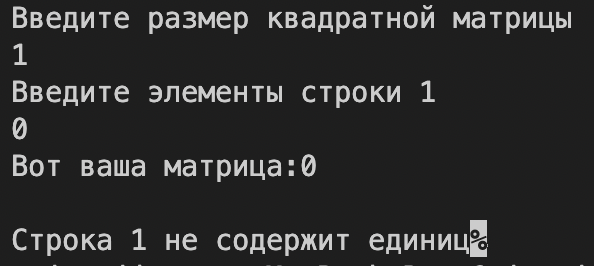
}

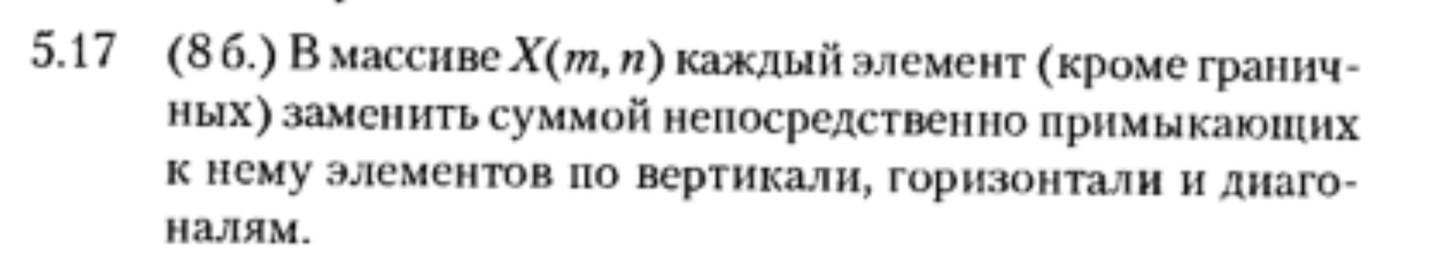
Скриншоты результатов:











Математическая модель: очевидна

Алгоритм изложен в комментариях к тексту программы:

#include <stdio.h>

#include <stdlib.h>

int main(){

int m; //число строк

int n; //число столбцов

//инициализируем переменные:

printf("Введите количество строк\n");

scanf("%d", &m);

printf("Введите количество столбцов\n");

scanf("%d", &n);

int A[m][n]; //создаем массив

int a, b; //вспомогательные счетчики для циклов

//заполним массив

for (a = 0; a < m; a ++){

printf("Введите элементы строки %d\n", a + 1);

for (b = 0; b < n; b++){

scanf("%d", &A[a][b]); //считываем введенное пользователем число и записываем

//его в соответствующую порядку ввода ячейку массива A

}

}

printf("Вот ваша матрица:\n");

for (a = 0; a < m; a++){ //выводим измененную матрицу в терминал

for (b = 0; b < n; b++){

printf("%d ", A[a][b]);

}

printf("\n");

}

//создадим дубликат матрицы

int B[m][n];

for (a = 0; a < m; a ++){

for (b = 0; b < n; b++){

B[a][b] = A[a][b];

}

}

//каждый неграничный элемент заменим суммой элементов прилегающих к нему со всех сторон ячеек

for (a = 1; a < m - 1; a++){

for (b = 1; b < n - 1; b++){

A[a][b] = B[a][b+1] + B[a][b-1] + B[a+1][b] + B[a-1][b] + B[a+1][b+1] + B[a+1][b-1] + B[a-1][b+1] + B[a-1][b+1];

}

}

printf("Вот измененная матрица:\n");

for (a = 0; a < m; a++){ //выводим измененную матрицу в терминал

for (b = 0; b < n; b++){

printf("%d ", A[a][b]);

}

printf("\n");

}

}

Скриншоты результатов:

