МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №7

по дисциплине: Основы программирования тема: «»

Выполнил: ст. группы ПВ-201 Машуров Дмитрий Русланович

Проверил:

Притчин Иван Сергеевич Брусенцева Валентина

Станиславовна

Белгород 2021 г.

**Лабораторная работа № 7**

**Массивы и указатели**

**Цель работы:** Освоение работы с динамическими массивами в языке Си, осознание связи между массивами и указателями.

**Задания для подготовки к работе**

1. Изучить:
   1. как описываются и инициализируются указатели, массивы указателей, указатели на массивы;
   2. операции над указателями;
   3. модели памяти в Си;
   4. Функции для работы с динамической памятью.
2. Рассмотреть возможные способы размещения матриц в динамической памяти и различные способы доступа к их элементам.
3. Разработать алгоритм и составить программу для решения задачи соответствующего варианта для каждого из следующих случаев задания матрицы:
   1. число строк и число столбцов - константы;
   2. число строк - константа, а число столбцов - исходное данное;
   3. число строк - исходное данное, число столбцов - константа;
   4. число строк и число столбцов - исходные данные^
      1. матрицу разместить с помощью массива указателей на строки;
      2. матрицу разместить с помощью указателя на одномерный массив размером mxn
      3. матрицу разместить как одномерный массив размером mxn и массива указателей на начальные элементы строк.
4. Ввод, вывод и обработку матрицы описать отдельными функциями. Для случаев а) - d), где возможно, использовать одни и те же функции.

**Задание варианта №17**

Дана матрица. Определить k – количество "особых" элементов данной матрицы, считая элемент "особым", если в строке слева от него находятся меньшие элементы, а справа – бóльшие

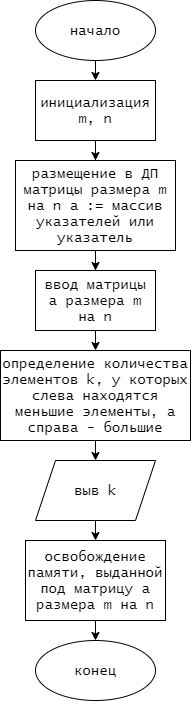
**Выполнение:**

1. **Описание алгоритма и выделение подзадач**

Исходя из условия задачи, будем проверять каждый элемент строки на то, слева от него находятся меньшие элементы, а справа – большие. Если данное условие выполняется, то включаем его в подсчёт

Выделим следующие подзадачи:

1. Ввод матрицы
2. Нахождение количества элементов в строке, у которых слева находятся меньшие элементы, а справа – большие
3. **Блок-схема алгоритма в укрупнённых блоках**

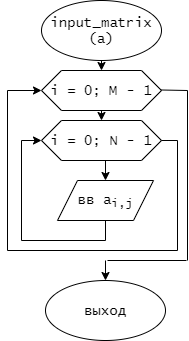
****

1. **Описание структур данных**

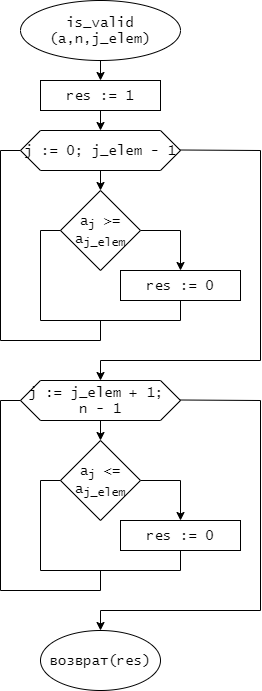
Определю константы, определяющие размер матрицы

#define M 3 //число строк  
#define N 3 //число столбцов

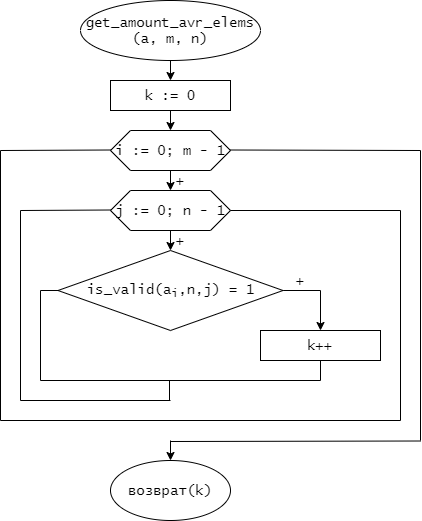
1. **Описание подзадач:**
2. Ввод матрицы
   1. Выделение подзадач
   2. Заголовок: void input\_matrix(int \*\*a)
   3. Назначение: ввод матрицы a размера MxN
   4. Блок-схема:



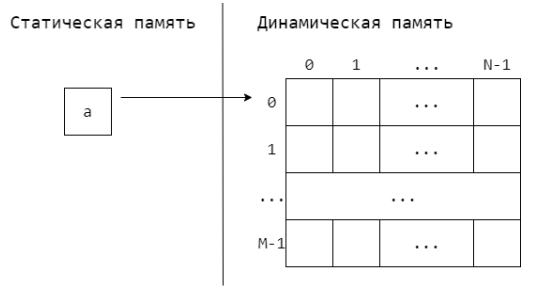
1. Определение, находятся ли слева от элемента только меньшие элементы, а справа – только большие
   1. Заголовок: int is\_valid(cons tint \*a, size\_t n, size\_t i\_elem, size\_t j\_elem)
   2. Назначение: возвращает «1», если в массиве а длины n слева от элемента под индексом j\_elem находятся только меньшие элементы, а справа – только большие, иначе – «0»
   3. Блок-схема:



1. Нахождение количества элементов в строке, у которых слева находятся меньшие элементы, а справа – большие
   1. Выделение подзадач:
      1. Определение, находятся ли слева от элемента только меньшие элементы, а справа – только большие
   2. Заголовок: int get\_amount\_avr\_elems(int (\*a)[N], size\_t m)
   3. Назначение: возвращает количество элементов матрицы a размера m на N, у которых слева находятся только меньшие элементы, а справа – только большие
   4. Блок-схема:



1. **Случаи задания матрицы**
   1. Число строк и столбцов – константы
      1. Размещение



* + 1. Обращение к строке: a[i]
    2. Обращение к элементу: a[i][j]
    3. Размещение в ДП

//выделяет память матрица a размера m на N  
void create\_matrix(int (\*\*a)[N], size\_t m) {  
\*a = (int (\*)[N])calloc(m, N \* sizeof(int));  
}

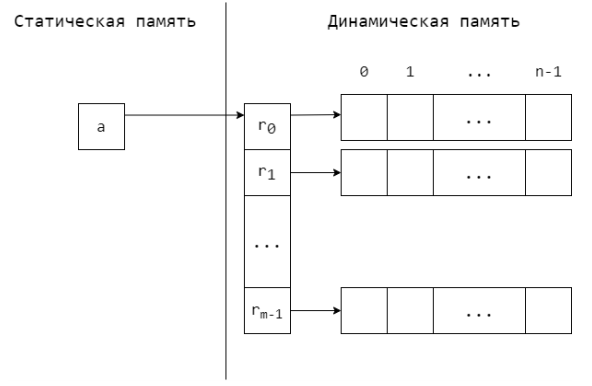
* + 1. Освобождение ДП

//освобождает память, выделенную под матрицу a  
void delete\_matrix(int (\*a)[N]) {  
free(a);  
}

* + 1. Заголовок функции ввода

//ввод матрицы а размера M на N  
void input\_matrix(int \*\*a, size\_t m, size\_t n)

* + 1. Текст программы  
       #include <stdio.h>  
       #include <stdlib.h>  
         
       #define M 3 //число строк  
       #define N 3 //число столбцов  
         
       //возвращает 1, если в массиве а длины n слева от элемента,  
       //под индексом j\_elem находятся только меньшие элменты,  
       //а справа - только большие  
       int is\_valid(const int \*a, size\_t n, size\_t j\_elem) {  
       //проверяем элементы, стоящие слева (если таковые есть)  
       for (size\_t j = 0; j < j\_elem; ++j) {  
       if (a[j] >= a[j\_elem]) {  
       return 0;  
       }  
       }  
         
       //проверяем элементы, стоящие справа (если таковые есть)  
       for (size\_t j = j\_elem + 1; j < n ; ++j) {  
       if (a[j] <= a[j\_elem]) {  
       return 0;  
       }  
       }  
         
       return 1;  
       }  
         
       //возвращает количество элементов матрицы а размера m на N,  
       //у которых слева находятся только меньше элементы, а справа -  
       //только большие  
       int get\_amount\_avr\_elems(int (\*a)[N], size\_t m) {  
       int k = 0;  
         
       for (size\_t i = 0; i < m; ++i) {  
       for (size\_t j = 0; j < N; ++j) {  
       if (is\_valid(a[i], N, j)) {  
       k++;  
       }  
       }  
       }  
         
       return k;  
       }  
         
       //ввод матрицы а размера m на N  
       void input\_matrix(int (\*a)[N], size\_t m) {  
       for (size\_t i = 0; i < m; ++i) {  
       for (size\_t j = 0; j < N; ++j) {  
       scanf("%d", &a[i][j]);  
       }  
       }  
       }  
         
       //выделяет память матрица a размера m на N  
       void create\_matrix(int (\*\*a)[N], size\_t m) {  
       \*a = (int (\*)[N])calloc(m, N \* sizeof(int));  
       }  
         
       //освобождает память, выделенную под матрицу a  
       void delete\_matrix(int (\*a)[N]) {  
       free(a);  
       }  
         
       int main() {  
       int (\*a)[N];  
       create\_matrix(&a, M);  
         
       printf("Input matrix (%d x %d)", M, N);  
       input\_matrix(a, M);  
         
       int k = get\_amount\_avr\_elems(a, M);  
       delete\_matrix(a);  
       printf("%d", k);  
       }
  1. Число строк – константа, число столбцов – исходное данное
     1. Размещение



* + 1. Обращение к строке: a[i]
    2. Обращение к элементу: a[i][j]
    3. Размещение в ДП

//выделяет память матрица a размера m на n  
void create\_matrix(int \*\*\*a, size\_t m, size\_t n) {  
\*a = (int \*\*)calloc(m, sizeof(int \*));  
for (size\_t i = 0; i < m; ++i) {  
(\*a)[i] = (int \*)calloc(n, sizeof(int));  
}  
}

* + 1. Освобождение ДП

//освобождает память, выделенную под матрицу a  
void delete\_matrix(int \*\*a, size\_t m) {  
for (size\_t i = 0; i < m; ++i) {  
free(a[i]);  
}  
free(a);  
}

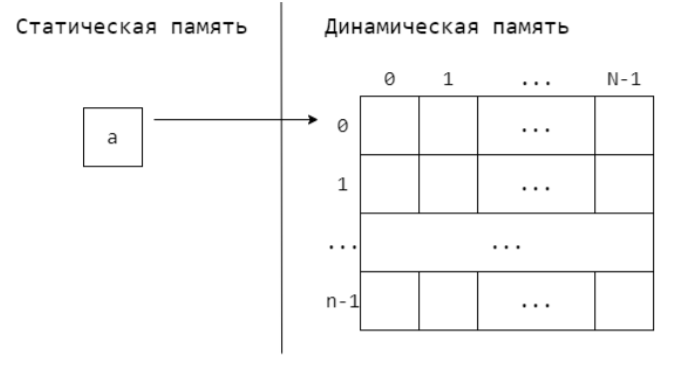
* + 1. Заголовок функции ввода

void input\_matrix(int\*\* a, size\_t m, size\_t n)

* + 1. Текст программы

#include <stdio.h>  
#include <stdlib.h>  
  
#define M 3 //число строк  
  
//возвращает 1, если в массиве а длины n слева от элемента,  
//под индексом j\_elem находятся только меньшие элменты,  
//а справа - только большие  
int is\_valid(const int \*a, size\_t n, size\_t j\_elem) {  
//проверяем элементы, стоящие слева (если таковые есть)  
for (size\_t j = 0; j < j\_elem; ++j) {  
if (a[j] >= a[j\_elem]) {  
return 0;  
}  
}  
  
//проверяем элементы, стоящие справа (если таковые есть)  
for (size\_t j = j\_elem + 1; j < n ; ++j) {  
if (a[j] <= a[j\_elem]) {  
return 0;  
}  
}  
  
return 1;  
}  
  
//возвращает количество элементов матрицы а размера m на n,  
//у которых слева находятся только меньше элементы, а справа -  
//только большие  
int get\_amount\_avr\_elems(int \*\*a, size\_t m, size\_t n) {  
int k = 0;  
  
for (size\_t i = 0; i < m; ++i) {  
for (size\_t j = 0; j < n; ++j) {  
if (is\_valid(a[i], n, j)) {  
k++;  
}  
}  
}  
  
return k;  
}  
  
//ввод матрицы а размера m на n  
void input\_matrix(int \*\*a, size\_t m, size\_t n) {  
for (size\_t i = 0; i < m; ++i) {  
for (size\_t j = 0; j < n; ++j) {  
scanf("%d", &a[i][j]);  
}  
}  
}  
  
//выделяет память матрица a размера m на n  
void create\_matrix(int \*\*\*a, size\_t m, size\_t n) {  
\*a = (int \*\*)calloc(m, sizeof(int \*));  
for (size\_t i = 0; i < m; ++i) {  
(\*a)[i] = (int \*)calloc(n, sizeof(int));  
}  
}  
  
//освобождает память, выделенную под матрицу a  
void delete\_matrix(int \*\*a, size\_t m) {  
for (size\_t i = 0; i < m; ++i) {  
free(a[i]);  
}  
  
free(a);  
}  
  
int main() {  
printf("Input number of columns\n");  
int n;  
scanf("%d", &n);  
  
int \*\*a;  
create\_matrix(&a, M, n);  
  
printf("Input matrix (%d x %d)", M, n);  
input\_matrix(a, M, n);  
  
int k = get\_amount\_avr\_elems(a, M, n);  
delete\_matrix(a, M);  
printf("%d", k);  
}

* 1. Число строк – исходное данное, число столбцов – константа
     1. Размещение



* + 1. Обращение к строке: a[i]
    2. Обращение к элементу: a[i][j]
    3. Размещение в ДП

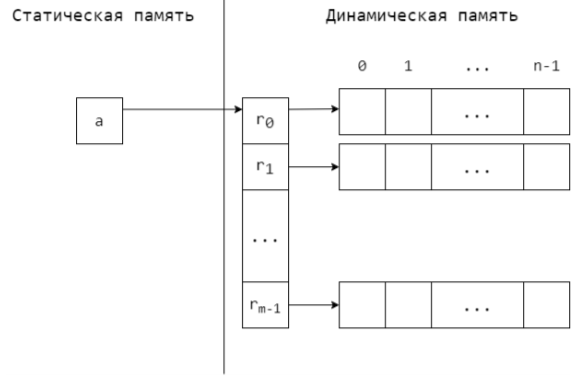
//выделяет память матрица a размера m на n  
void create\_matrix(int (\*\*a)[N], size\_t m) {  
\*a = (int (\*)[N])calloc(m, N \* sizeof(int));  
}

* + 1. Освобождение ДП

//освобождает память, выделенную под матрицу a  
void delete\_matrix(int (\*a)[N]) {  
free(a);  
}

* + 1. Заголовок функции ввода

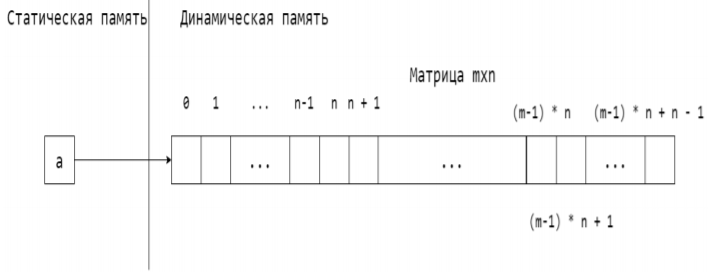
void input\_matrix(int (\*a)[N], size\_t m)

* + 1. Текст программы  
       #include <stdio.h>  
       #include <stdlib.h>  
         
       #define N 3 //число столбцов  
         
       //возвращает 1, если в массиве а длины n слева от элемента,  
       //под индексом j\_elem находятся только меньшие элменты,  
       //а справа - только большие  
       int is\_valid(const int \*a, size\_t n, size\_t j\_elem) {  
       //проверяем элементы, стоящие слева (если таковые есть)  
       for (size\_t j = 0; j < j\_elem; ++j) {  
       if (a[j] >= a[j\_elem]) {  
       return 0;  
       }  
       }  
         
       //проверяем элементы, стоящие справа (если таковые есть)  
       for (size\_t j = j\_elem + 1; j < n ; ++j) {  
       if (a[j] <= a[j\_elem]) {  
       return 0;  
       }  
       }  
         
       return 1;  
       }  
         
       //возвращает количество элементов матрицы а размера m на n,  
       //у которых слева находятся только меньше элементы, а справа -  
       //только большие  
       int get\_amount\_avr\_elems(int (\*a)[N], size\_t m) {  
       int k = 0;  
         
       for (size\_t i = 0; i < m; ++i) {  
       for (size\_t j = 0; j < N; ++j) {  
       if (is\_valid(a[i], N, j)) {  
       k++;  
       }  
       }  
       }  
         
       return k;  
       }  
         
       //ввод матрицы а размера m на n  
       void input\_matrix(int (\*a)[N], size\_t m) {  
       for (size\_t i = 0; i < m; ++i) {  
       for (size\_t j = 0; j < N; ++j) {  
       scanf("%d", &a[i][j]);  
       }  
       }  
       }  
         
       //выделяет память матрица a размера m на n  
       void create\_matrix(int (\*\*a)[N], size\_t m) {  
       \*a = (int (\*)[N])calloc(m, N \* sizeof(int));  
       }  
         
       //освобождает память, выделенную под матрицу a  
       void delete\_matrix(int (\*a)[N]) {  
       free(a);  
       }  
         
       int main() {  
       printf("Input number of rows\n");  
       size\_t m;  
       scanf("%d", &m);  
         
       int (\*a)[N];  
       create\_matrix(&a, m);  
         
       int k = get\_amount\_avr\_elems(a, m);  
       delete\_matrix(a);  
       printf("%d", k);  
       }
  1. Число строк и столбцов – исходные данные
     1. Матрица – массив указателей на строки
        1. Размещение
        2. Обращение к строке: a[i]
        3. Обращение к элементу: a[i][j]
        4. Размещение в ДП
        5. Освобождение ДП
        6. Заголовок функции ввода

void input\_matrix(int \*\*a, size\_t m, size\_t n)

* + - 1. Текст программы

#include <stdio.h>  
#include <stdlib.h>  
  
//возвращает 1, если в массиве а длины n слева от элемента,  
//под индексом j\_elem находятся только меньшие элменты,  
//а справа - только большие  
int is\_valid(const int \*a, size\_t n, size\_t j\_elem) {  
//проверяем элементы, стоящие слева (если таковые есть)  
for (size\_t j = 0; j < j\_elem; ++j) {  
if (a[j] >= a[j\_elem]) {  
return 0;  
}  
}  
  
//проверяем элементы, стоящие справа (если таковые есть)  
for (size\_t j = j\_elem + 1; j < n ; ++j) {  
if (a[j] <= a[j\_elem]) {  
return 0;  
}  
}  
  
return 1;  
}  
  
//возвращает количество элементов матрицы а размера m на n,  
//у которых слева находятся только меньше элементы, а справа -  
//только большие  
int get\_amount\_avr\_elems(int \*\*a, size\_t m, size\_t n) {  
int k = 0;  
  
for (size\_t i = 0; i < m; ++i) {  
for (size\_t j = 0; j < n; ++j) {  
if (is\_valid(a[i], n, j)) {  
k++;  
}  
}  
}  
  
return k;  
}  
  
//ввод матрицы а размера m на n  
void input\_matrix(int \*\*a, size\_t m, size\_t n) {  
for (size\_t i = 0; i < m; ++i) {  
for (size\_t j = 0; j < n; ++j) {  
scanf("%d", &a[i][j]);  
}  
}  
}  
  
//выделяет память матрица a размера m на n  
void create\_matrix(int \*\*\*a, size\_t m, size\_t n) {  
\*a = (int \*\*)calloc(m, sizeof(int \*));  
for (size\_t i = 0; i < m; ++i) {  
(\*a)[i] = (int \*) calloc(n, sizeof(int));  
}  
}  
  
//освобождает память, выделенную под матрицу a  
void delete\_matrix(int \*\*a, size\_t m) {  
for (size\_t i = 0; i < m; ++i) {  
free(a[i]);  
}  
  
free(a);  
}  
  
int main() {  
printf("Input number of rows\n");  
int m;  
scanf("%d", &m);  
  
printf("Input number of columns\n");  
int n;  
scanf("%d", &n);  
  
int \*\*a;  
create\_matrix(&a, m, n);  
  
printf("Input matrix (%d x %d)\n", m, n);  
input\_matrix(a, m, n);  
  
int k = get\_amount\_avr\_elems(a, m, n);  
delete\_matrix(a, m);  
printf("%d", k);  
}

* + 1. Матрица – указатель на одномерный массив mxn
       1. Размещение
       2. 

Обращение к строке: a[i \* n]

* + - 1. Обращение к элементу: a[i \* n + j]
      2. Размещение в ДП

//выделяет память матрица a размера m на n  
void create\_matrix(int \*\*a, size\_t m, size\_t n) {  
\*a = (int \*)calloc(m, n \* sizeof(int \*));  
}

* + - 1. Освобождение ДП

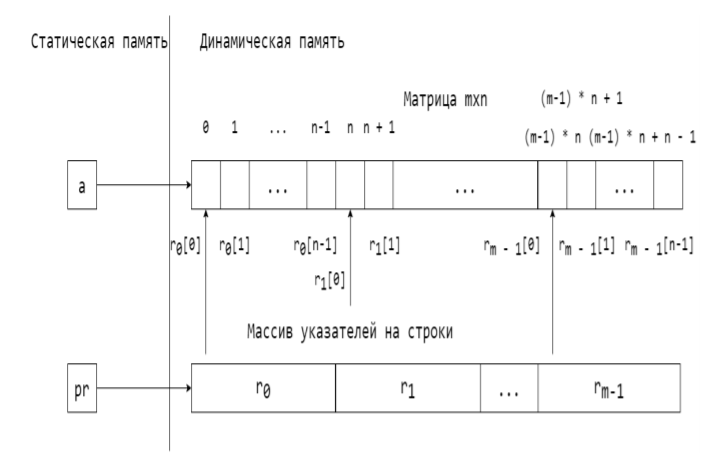
//освобождает память, выделенную под матрицу a  
void delete\_matrix(int \*\*a) {  
free(a);  
}

* + - 1. Заголовок функции ввода

void input\_matrix(int \*a, size\_t m, size\_t n)

* + - 1. Текст программы

#include <stdio.h>  
#include <stdlib.h>  
  
//возвращает 1, если в массиве а длины n слева от элемента,  
//под индексом j\_elem находятся только меньшие элменты,  
//а справа - только большие  
int is\_valid(const int \*a, size\_t n, size\_t j\_elem) {  
//проверяем элементы, стоящие слева (если таковые есть)  
for (size\_t j = 0; j < j\_elem; ++j) {  
if (a[j] >= a[j\_elem]) {  
return 0;  
}  
}  
  
//проверяем элементы, стоящие справа (если таковые есть)  
for (size\_t j = j\_elem + 1; j < n ; ++j) {  
if (a[j] <= a[j\_elem]) {  
return 0;  
}  
}  
  
return 1;  
}  
  
//возвращает количество элементов матрицы а размера m на n,  
//у которых слева находятся только меньше элементы, а справа -  
//только большие  
int get\_amount\_avr\_elems(int \*\*a, size\_t m, size\_t n) {  
int k = 0;  
  
for (size\_t i = 0; i < m; ++i) {  
for (size\_t j = 0; j < n; ++j) {  
if (is\_valid(a[i], n, j)) {  
k++;  
}  
}  
}  
  
return k;  
}  
  
//ввод матрицы а размера m на n  
void input\_matrix(int \*\*a, size\_t m, size\_t n) {  
for (size\_t i = 0; i < m; ++i) {  
for (size\_t j = 0; j < n; ++j) {  
scanf("%d", &a[i][j]);  
}  
}  
}  
  
//выделяет память матрица a размера m на n  
void create\_matrix(int \*\*a, size\_t m, size\_t n) {  
\*a = (int \*)calloc(m, n \* sizeof(int \*));  
}  
  
//освобождает память, выделенную под матрицу a  
void delete\_matrix(int \*\*a) {  
free(a);  
}  
  
int main() {  
printf("Input number of rows\n");  
int m;  
scanf("%d", &m);  
  
printf("Input number of columns\n");  
int n;  
scanf("%d", &n);  
  
int \*\*a;  
create\_matrix(a, m, n);  
  
printf("Input matrix (%d x %d)\n", m, n);  
input\_matrix(a, m, n);  
  
int k = get\_amount\_avr\_elems(a, m, n);  
delete\_matrix(a);  
printf("%d", k);  
}

* + 1. Матрица – одномерный массив размером mxn и массива указателей на начальные элементы строк
       1. Размещение
       2. 

Обращение к строке: a[i \* n], pr[i]

* + - 1. Обращение к элементу: pr[i][j] | a[i \* n + j]
      2. Размещение в ДП

//выделяет память матрица a размера m на n  
void create\_matrix(int \*\*a, int \*\*\*pr, size\_t m, size\_t n) {  
\*a = (int \*)calloc(m, n \* sizeof(int));  
\*pr = (int \*\*) calloc(m, sizeof(int \*));  
for (int i = 0; i < m; ++i) {  
(\*pr)[i] = \*a + i \* n;  
}  
}

* + - 1. Освобождение ДП

//освобождает память, выделенную под матрицу a  
void delete\_matrix(int \*a, int \*\*pr) {  
free(a);  
free(pr);  
}

* + - 1. Заголовок функции ввода

void input\_matrix(int \*\*a, size\_t m, size\_t n)

* + - 1. Текст программы

#include <stdio.h>  
#include <stdlib.h>  
  
//возвращает 1, если в массиве а длины n слева от элемента,  
//под индексом j\_elem находятся только меньшие элменты,  
//а справа - только большие  
int is\_valid(const int \*a, size\_t n, size\_t j\_elem) {  
//проверяем элементы, стоящие слева (если таковые есть)  
for (size\_t j = 0; j < j\_elem; ++j) {  
if (a[j] >= a[j\_elem]) {  
return 0;  
}  
}  
  
//проверяем элементы, стоящие справа (если таковые есть)  
for (size\_t j = j\_elem + 1; j < n ; ++j) {  
if (a[j] <= a[j\_elem]) {  
return 0;  
}  
}  
  
return 1;  
}  
  
//возвращает количество элементов матрицы а размера m на n,  
//у которых слева находятся только меньше элементы, а справа -  
//только большие  
int get\_amount\_avr\_elems(int \*\*a, size\_t m, size\_t n) {  
int k = 0;  
  
for (size\_t i = 0; i < m; ++i) {  
for (size\_t j = 0; j < n; ++j) {  
if (is\_valid(a[i], n, j)) {  
k++;  
}  
}  
}  
  
return k;  
}  
  
//ввод матрицы а размера m на n  
void input\_matrix(int \*\*a, size\_t m, size\_t n) {  
for (size\_t i = 0; i < m; ++i) {  
for (size\_t j = 0; j < n; ++j) {  
scanf("%d", &a[i][j]);  
}  
}  
}  
  
//выделяет память матрица a размера m на n  
void create\_matrix(int \*\*a, int \*\*\*pr, size\_t m, size\_t n) {  
\*a = (int \*)calloc(m, n \* sizeof(int));  
\*pr = (int \*\*) calloc(m, sizeof(int \*));  
for (int i = 0; i < m; ++i) {  
(\*pr)[i] = \*a + i \* n;  
}  
}  
  
//освобождает память, выделенную под матрицу a  
void delete\_matrix(int \*a, int \*\*pr) {  
free(a);  
free(pr);  
}  
  
int main() {  
printf("Input number of rows\n");  
int m;  
scanf("%d", &m);  
  
printf("Input number of columns\n");  
int n;  
scanf("%d", &n);  
  
int \*a;  
int \*\*pr;  
create\_matrix(&a, &pr, m, n);  
  
printf("Input matrix (%d x %d)\n", m, n);  
input\_matrix(pr, m, n);  
  
int k = get\_amount\_avr\_elems(pr, m, n);  
delete\_matrix(a, pr);  
printf("%d", k);  
}