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: «Массивы и указатели»

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^
        i. матрицу разместить с помощью массива указателей на строки;
        ii. матрицу разместить с помощью указателя на одномерный массив размером mxn
        iii. 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 a) - 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

2. **Блок-схема алгоритма в укрупнённых блоках**

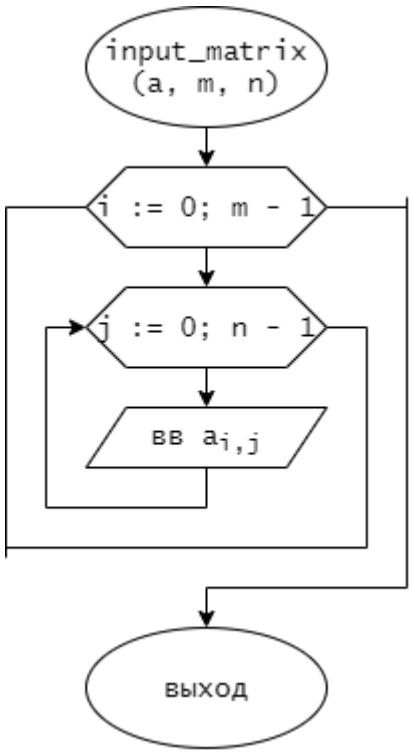
3. Описание структур данных

Определю константы, определяющие размер матрицы

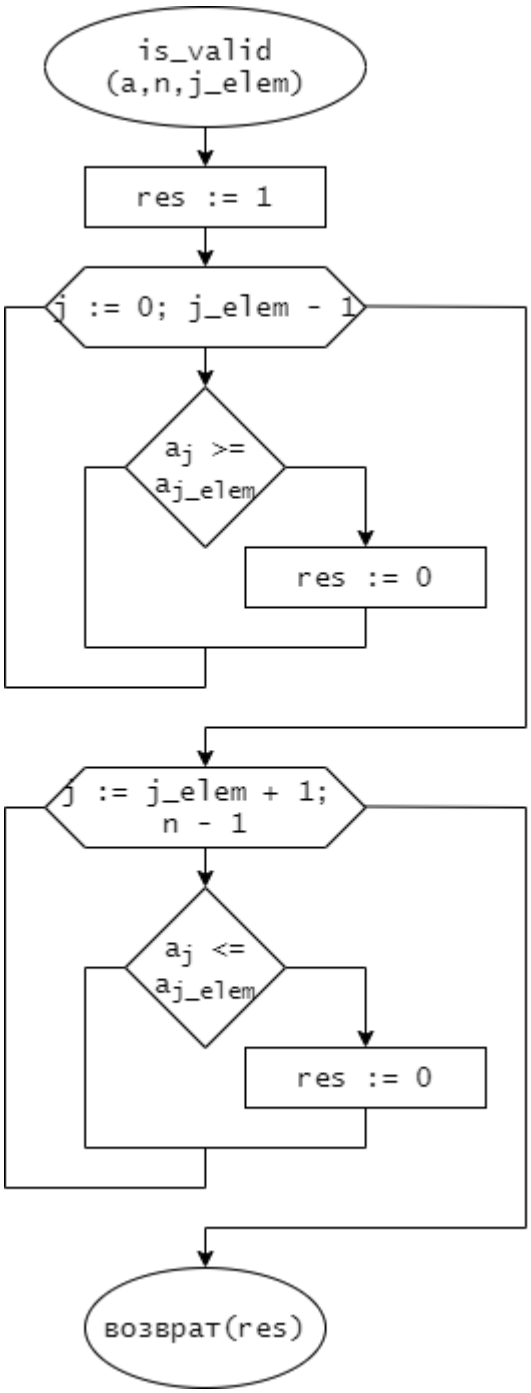
```
#define M 3 //число строк
#define N 3 //число столбцов
```

4. Описание подзадач:

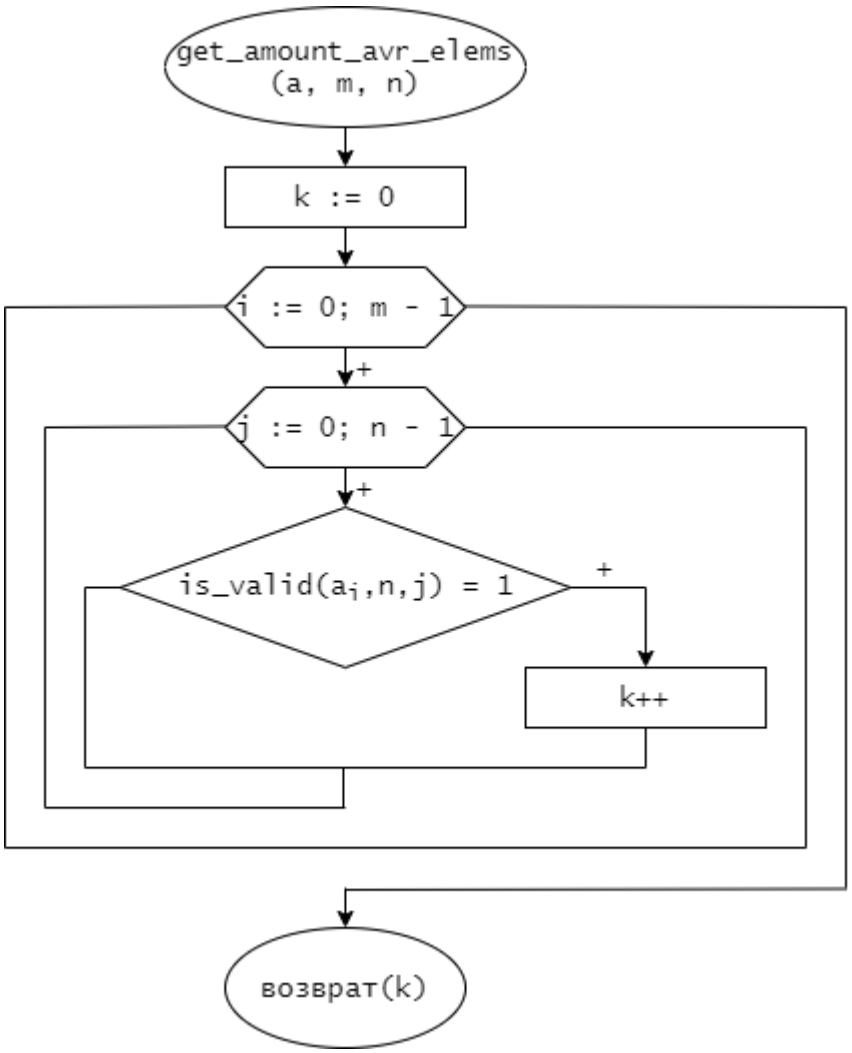
1) Ввод матрицы
    a. Выделение подзадач
    b. Заголовок: `void input_matrix(int a[][N], size_t m)`
    c. Назначение: ввод матрицы `a` размера `mxN` (N = 3)
    d. Блок-схема:



2) Определение, находятся ли слева от элемента только меньшие элементы, а справа – только большие
    a. Заголовок: `int is_valid(cons tint *a, size_t n, size_t i_elem, size_t j_elem)`
    b. Назначение: возвращает «1», если в массиве `a` длины `n` слева от элемента под индексом `j_elem` находятся только меньшие элементы, а справа – только большие, иначе – «0»
    c. Блок-схема:



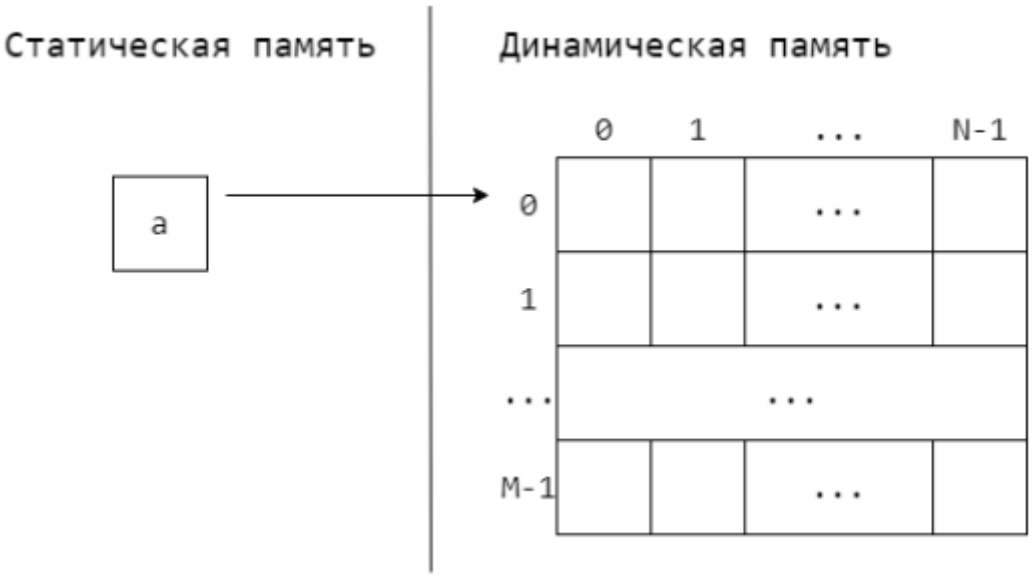
3) Нахождение количества элементов в строке, у которых слева находятся меньшие элементы, а справа – большие
    a. Выделение подзадач:
        i. Определение, находятся ли слева от элемента только меньшие элементы, а справа – только большие
    b. Заголовок: `int get_amount_avr_elems(int (*a)[N], size_t m)`
    c. Назначение: возвращает количество элементов матрицы `a` размера `m` на `N`, у которых слева находятся только меньшие элементы, а справа – только большие
    d. Блок-схема:



5. Случаи задания матрицы

a. Число строк и столбцов – константы

i. Размещение



ii. Обращение к строке: `a[i]`

iii. Обращение к элементу: `a[i][j]`

iv. 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));
}
```

v. Освобождение ДП

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

vi. Заголовок функции ввода

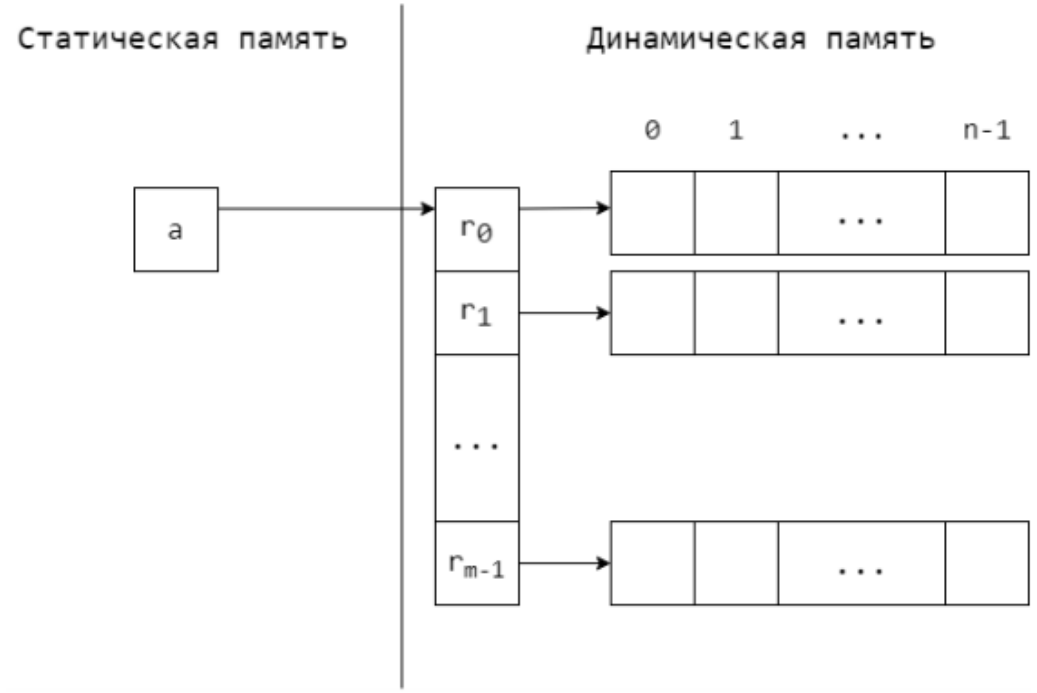
```
//ввод матрицы a размера M на N
void input_matrix(int **a, size_t m, size_t n)
```

vii. Текст программы

```
#include <stdio.h>
#include <stdlib.h>

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

//возвращает 1, если в массиве a 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 a 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 a 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);
}
```

b. Число строк – константа, число столбцов – исходное данное
   i. Размещение



   ii. Обращение к строке: `a[i]`
   iii. Обращение к элементу: `a[i][j]`
   iv. 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));
    }
}
```

   v. 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);
}
```

   vi. Заголовок функции ввода

```
void input_matrix(int** a, size_t m, size_t n)
```

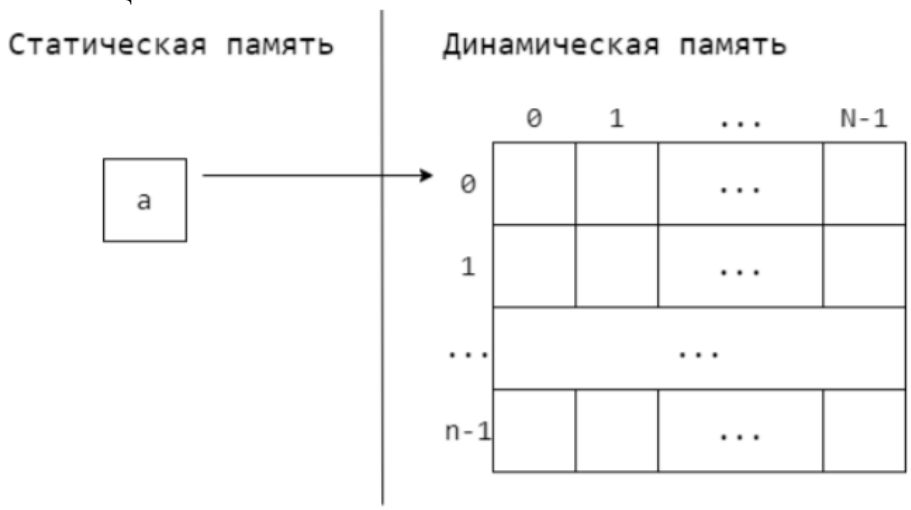
   vii. Текст программы

```
#include <stdio.h>
#include <stdlib.h>

#define M 3 //число строк

//возвращает 1, если в массиве a 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 a 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 a 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);
}
```

c. Число строк – исходное данное, число столбцов – константа
   i. Размещение



   ii. Обращение к строке: `a[i]`
   iii. Обращение к элементу: `a[i][j]`
   iv. 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));
}
```

   v. Освобождение ДП

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

   vi. Заголовок функции ввода

```
void input_matrix(int (*a)[N], size_t m)
```

   vii. Текст программы

```
#include <stdio.h>
#include <stdlib.h>
```

```
#define N 3 //число столбцов

//возвращает 1, если в массиве a 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 a 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 a 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);
}
```

d. Число строк и столбцов – исходные данные
   i. 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)`
      7. Текст программы

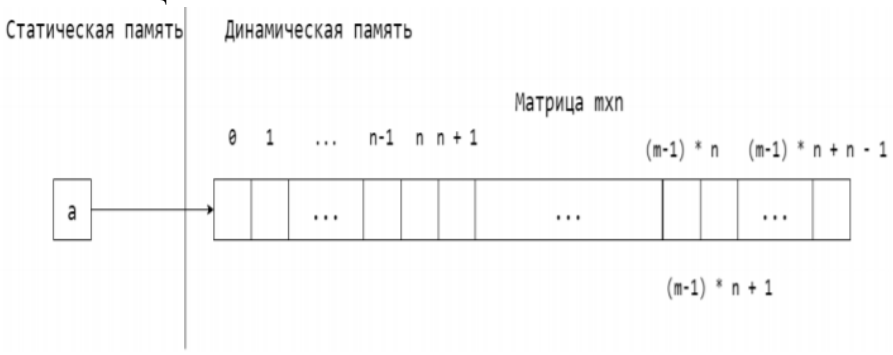
```
#include <stdio.h>
#include <stdlib.h>

//возвращает 1, если в массиве a длины n слева от
элемента,
//под индексом j_elem находятся только меньшие
элменты,
```

```
//a 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 a 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 a 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);
}
```

ii. Матрица – указатель на одномерный массив mxn

1. Размещение



2. Обращение к строке: `a[i * n]`
3. Обращение к элементу: `a[i * n + j]`
4. 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 *));
}
```

5. Освобождение ДП

```
//освобождает память, выделенную под матрицу a
void delete_matrix(int **a) {
    free(a);
```

```
}
```

6. Заголовок функции ввода

```
void input_matrix(int *a, size_t m, size_t n)
```

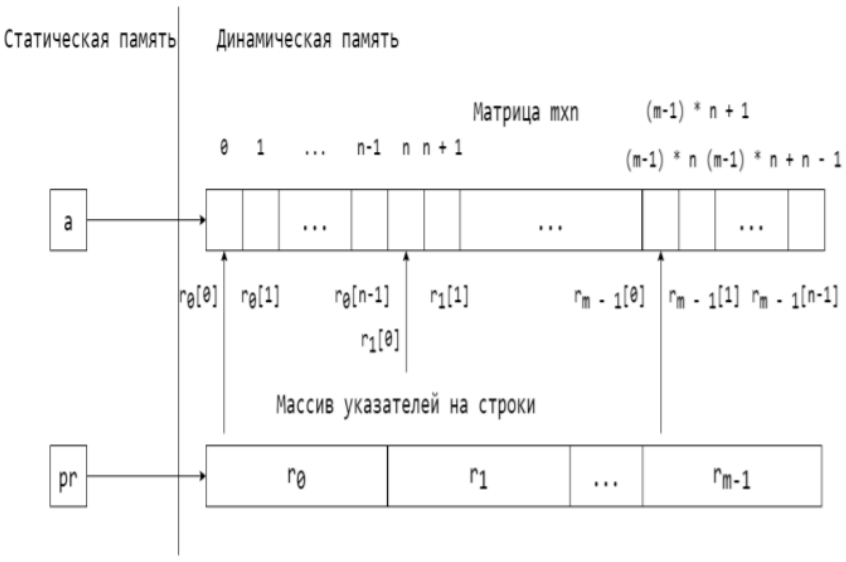
7. Текст программы

```
#include <stdio.h>
#include <stdlib.h>

//возвращает 1, если в массиве a 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 a 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 a 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);
}
```

iii. Матрица – одномерный массив размером mxn и массива указателей на начальные элементы строк

1. Размещение



2. Обращение к строке: `a[i * n], pr[i]`
3. Обращение к элементу: `pr[i][j] | a[i * n + j]`
4. 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;
    }
}
```
5. Освобождение ДП
```
//освобождает память, выделенную под матрицу a
void delete_matrix(int *a, int **pr) {
    free(a);
    free(pr);
}
```
6. Заголовок функции ввода
```
void input_matrix(int **a, size_t m, size_t n)
```
7. Текст программы
```
#include <stdio.h>
#include <stdlib.h>

//возвращает 1, если в массиве a 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 a 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 a 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);
    input_matrix(a, m);

    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);
}
```

6. Тестовые данные

№	Вход	Выход
1	M = N = 3 1 2 3 0 0 0 0 1 0	3
2	M = 3, n = 2 1 2 4 5 1 0	4
3	m = 4, N = 3 1 1 1 0 0 0 0 1 0 1 0 1	0
4	M = 2, n = 5 7 6 4 3 8 1 2 0 4 5	3

7. Результаты

Пример №1:

```
Input matrix (3 x 3)
1 2 3
0 0 0
0 1 0
3
```

Пример №2:

```
Input number of columns
2
Input matrix (3 x 2)
1 2
4 5
1 0
4
```

Пример №3:

```
Input number of rows
4
Input matrix (4 x 3)
1 1 1
0 0 0
0 1 0
1 0 1
0
```

Пример №4:

```
Input number of rows
2
Input number of columns
5
Input matrix (2 x 5)
7 6 4 3 8
1 2 0 4 5
3
```