

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:
 - 1.1 как описываются и инициализируются указатели, массивы указателей, указатели на массивы;
 - 1.2 операции над указателями;
 - 1.3 модели памяти в Си;
 - 1.4 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:
 - 3.1 число строк и число столбцов - константы;
 - 3.2 число строк - константа, а число столбцов - исходное данное;
 - 3.3 число строк - исходное данное, число столбцов - константа;
 - 3.4 число строк и число столбцов - исходные данные[^]
 - 4.i матрицу разместить с помощью массива указателей на строки;
 - 4.ii матрицу разместить с помощью указателя на одномерный массив размером $m \times n$
 - 4.iii матрицу разместить как одномерный массив размером $m \times n$ и массива указателей на начальные элементы строк.
- 4 Ввод, вывод и обработку матрицы описать отдельными функциями. Для случаев а) - д), 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 – большие

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

Определяю константы, определяющие размер матрицы

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

4 Описание подзадач:

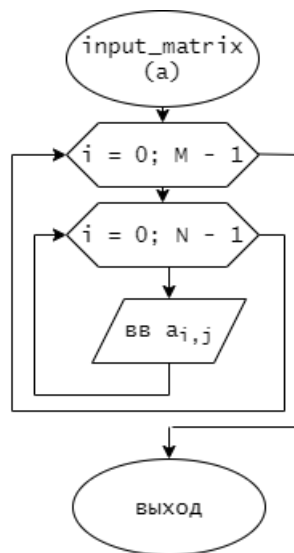
1 Ввод матрицы

1.a Выделение подзадач

1.b Заголовок: `void input_matrix(int **a)`

1.c Назначение: ввод матрицы a размера MxN (M = N = 3)

1.d Блок-схема:



2 Нахождение количества элементов в строке, у которых слева находятся меньшие элементы, а справа – большие

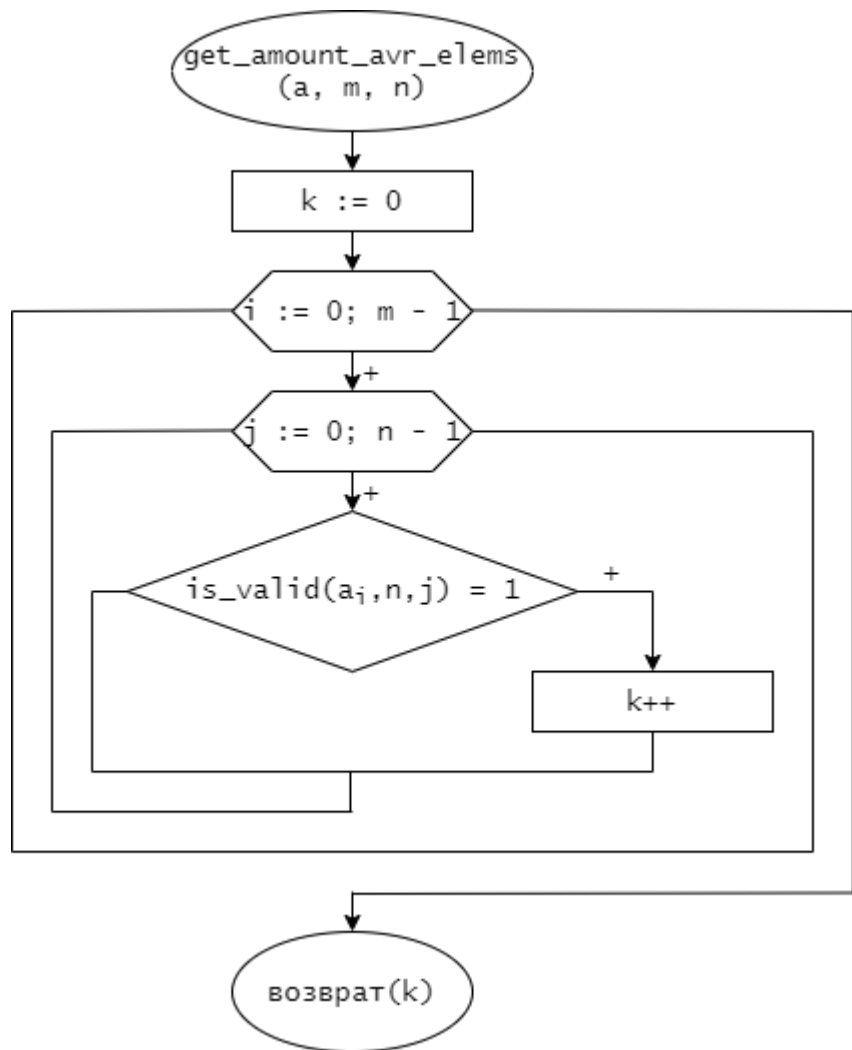
2.a Выделение подзадач:

a.i Определение, находятся ли слева от элемента только меньшие элементы, а справа – только большие

2.b Заголовок: `int get_amount_avr_elems(int (*a)[N], size_t m)`

2.c Назначение: возвращает количество элементов матрицы `a` размера `m` на `N`, у которых слева находятся только меньшие элементы, а справа – только большие

2.d Блок-схема:

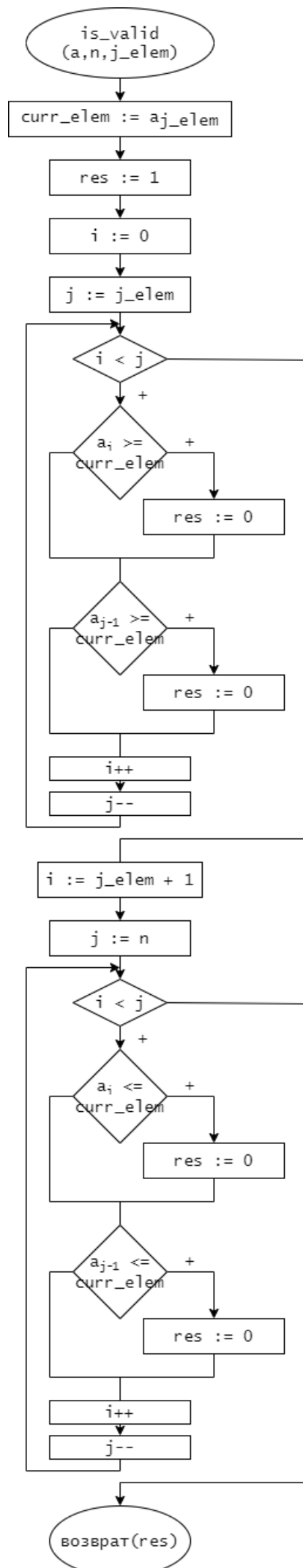


a.i. Определение, находятся ли слева от элемента только меньшие элементы, а справа – только большие

a.i.1 Заголовок: `int is_valid(const int *a, size_t n, size_t i_elem, size_t j_elem)`

a.i.2 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»

a.i.3 Блок-схема:



5 Случаи задания матрицы

5.а Число строк и столбцов – константы

а.і Размещение



а.іі Обращение к строке: $a[i]$

а.ііі Обращение к элементу: $a[i][j]$

а.ііv 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));
}
```

а.іv Освобождение ДП

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

а.іvi Заголовок функции ввода

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

а.іvii 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 элементы,
//а справа - только большие
int is_valid(const int *a, size_t n, size_t j_elem) {
    int curr_elem = a[j_elem];

    //проверяем элементы, стоящие слева
    size_t i = 0, j = j_elem;
    while (i < j) {
        if (a[i] >= curr_elem) {
            return 0;
        }
    }
```



```

        if (a[j - 1] >= curr_elem) {
            return 0;
        }

        i++; j--;
    }

    //проверяем элементы, стоящие справа
    i = j_elem + 1, j = n;
    while (i < j) {
        if (a[i] <= curr_elem) {
            return 0;
        }

        if (a[j - 1] <= curr_elem) {
            return 0;
        }

        i++; j--;
    }

    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);

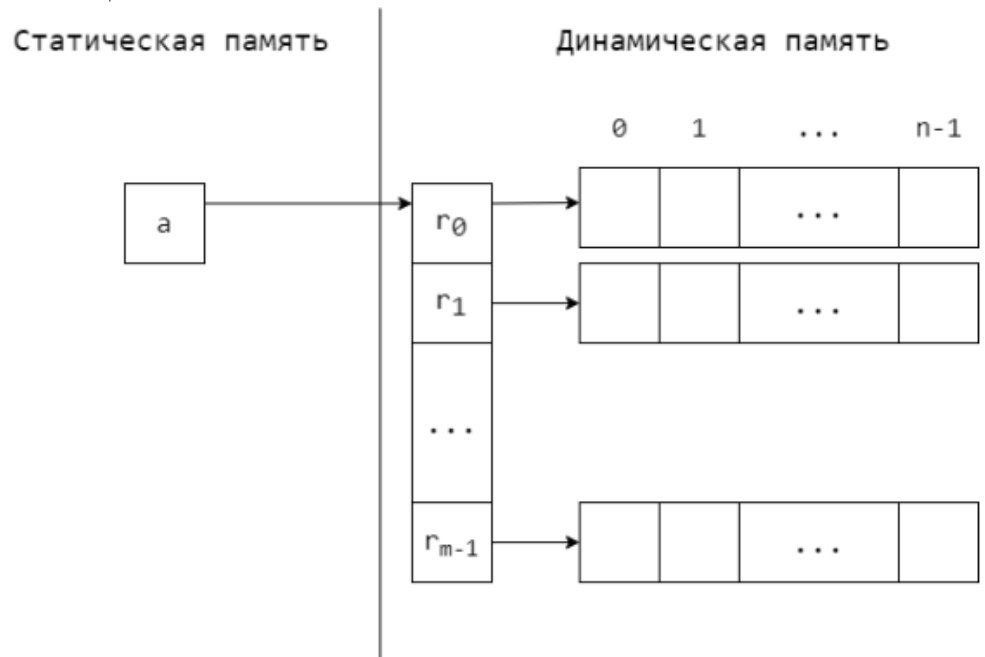
    printf("Input matrix (%d x %d)\n", M, N);
    input_matrix(a, M);

    int k = get_amount_avr_elems(a, M);
    delete_matrix(a);
    printf("%d", k);
}

```

5.b Число строк – константа, число столбцов – исходное данные

b.i Размещение



b.ii Обращение к строке: $a[i]$

b.iii Обращение к элементу: $a[i][j]$

b.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));
    }
}

```

b.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);
}

```

b.vi Заголовок функции ввода

```

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

```

b.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 элементы,
//а справа - только большие
int is_valid(const int *a, size_t n, size_t j_elem) {
    int curr_elem = a[j_elem];

    //проверяем элементы, стоящие слева
    size_t i = 0, j = j_elem;
    while (i < j) {
        if (a[i] >= curr_elem) {
            return 0;
        }

        if (a[j - 1] >= curr_elem) {
            return 0;
        }

        i++; j--;
    }

    //проверяем элементы, стоящие справа
    i = j_elem + 1, j = n;
    while (i < j) {
        if (a[i] <= curr_elem) {
            return 0;
        }

        if (a[j - 1] <= curr_elem) {
            return 0;
        }

        i++; j--;
    }

    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)\n", M, n);
    input_matrix(a, M, n);

    int k = get_amount_avr_elems(a, M, n);
    delete_matrix(a, M);
    printf("%d", k);
}

```

5.с Число строк – исходное данные, число столбцов – константа

с.i Размещение



с.ii Обращение к строке: $a[i]$

с.iii Обращение к элементу: $a[i][j]$

с.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));
}
```

с.v Освобождение ДП

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

с.vi Заголовок функции ввода

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

с.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 элементы,
//а справа - только большие
int is_valid(const int *a, size_t n, size_t j_elem) {
    int curr_elem = a[j_elem];

    //проверяем элементы, стоящие слева
    size_t i = 0, j = j_elem;
    while (i < j) {
        if (a[i] >= curr_elem) {
            return 0;
        }
    }

    if (a[j - 1] >= curr_elem) {
        return 0;
    }
}
```

```

        i++; j--;
    }

    //проверяем элементы, стоящие справа
    i = j_elem + 1, j = n;
    while (i < j) {
        if (a[i] <= curr_elem) {
            return 0;
        }

        if (a[j - 1] <= curr_elem) {
            return 0;
        }

        i++; j--;
    }

    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);

printf("Input matrix (%d x 3)\n", m);
input_matrix(a, m);

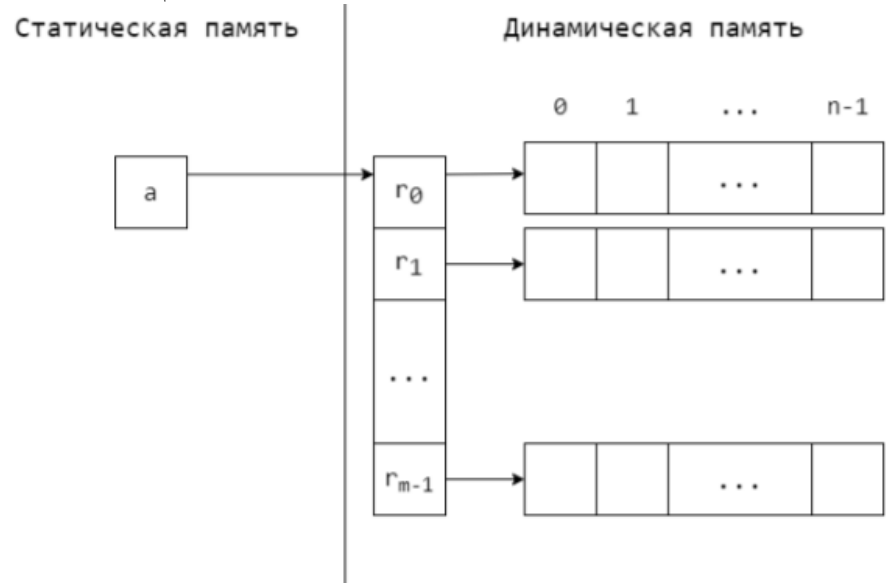
int k = get_amount_avr_elems(a, m);
delete_matrix(a);
printf("%d", k);
}

```

5.d Число строк и столбцов – исходные данные

d.i Матрица – массив указателей на строки

i.1 Размещение



i.2 Обращение к строке: $a[i]$

i.3 Обращение к элементу: $a[i][j]$

i.4 Размещение в ДП

i.5 Освобождение ДП

i.6 Заголовок функции ввода

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

i.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 элементы,
//а справа - только большие
int is_valid(const int *a, size_t n, size_t j_elem) {
    int curr_elem = a[j_elem];

    //проверяем элементы, стоящие слева
    size_t i = 0, j = j_elem;
    while (i < j) {
        if (a[i] >= curr_elem) {
            return 0;
        }
    }
}

```

```

        if (a[j - 1] >= curr_elem) {
            return 0;
        }

        i++; j--;
    }

    //проверяем элементы, стоящие справа
    i = j_elem + 1, j = n;
    while (i < j) {
        if (a[i] <= curr_elem) {
            return 0;
        }

        if (a[j - 1] <= curr_elem) {
            return 0;
        }

        i++; j--;
    }

    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);

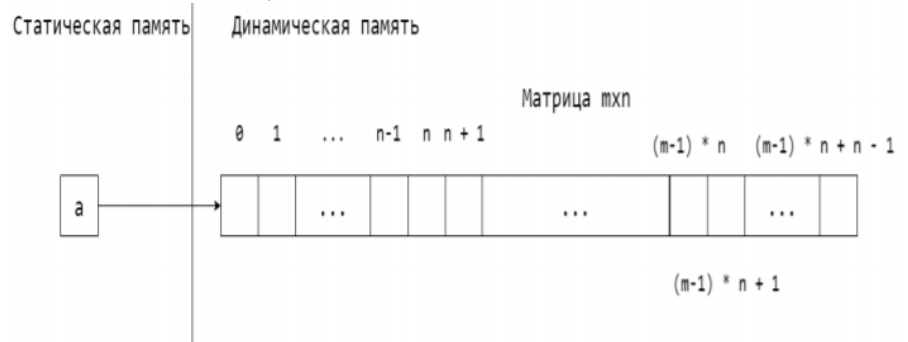
    int **a;
    create_matrix(&a, m, n);

    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);
}
```

d.ii Матрица – указатель на одномерный массив `mxn`

ii.1 Размещение



ii.2 Обращение к строке: $a[i * n]$, где i – номер строки

ii.3 Обращение к элементу: $a[i * n + j]$, где i – номер строки, j – номер столбца

ii.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 *));
}
```

ii.5 Освобождение ДП

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

ii.6 Заголовок функции ввода

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

ii.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) {  
    int curr_elem = a[j_elem];
```

```
    //проверяем элементы, стоящие слева
```

```
    size_t i = 0, j = j_elem;
```

```
    while (i < j) {
```

```
        if (a[i] >= curr_elem) {  
            return 0;
```

```
        }
```

```
        if (a[j - 1] >= curr_elem) {  
            return 0;
```

```
        }
```

```
        i++; j--;
```

```
    }
```

```
    //проверяем элементы, стоящие справа
```

```
    i = j_elem + 1, j = n;
```

```
    while (i < j) {
```

```
        if (a[i] <= curr_elem) {  
            return 0;
```

```
        }
```

```
        if (a[j - 1] <= curr_elem) {  
            return 0;
```

```
        }
```

```
        i++; j--;
```

```
    }
```

```
    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);
    }

```

d.iii Матрица – одномерный массив размером $m \times n$ и массив указателей на начальные элементы строк

iii.1 Размещение



iii.2 Обращение к строке: $a[i * n] \mid pr[i]$

iii.3 Обращение к элементу: $pr[i][j] \mid a[i * n + j]$

iii.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;
    }
}
```

iii.5 Освобождение ДП

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

iii.6 Заголовок функции ввода

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

iii.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 элементы,
//а справа - только большие
int is_valid(const int *a, size_t n, size_t j_elem) {
    int curr_elem = a[j_elem];

    //проверяем элементы, стоящие слева
    size_t i = 0, j = j_elem;
    while (i < j) {
        if (a[i] >= curr_elem) {
```

```

        return 0;
    }

    if (a[j - 1] >= curr_elem) {
        return 0;
    }

    i++; j--;
}

//проверяем элементы, стоящие справа
i = j_elem + 1, j = n;
while (i < j) {
    if (a[i] <= curr_elem) {
        return 0;
    }

    if (a[j - 1] <= curr_elem) {
        return 0;
    }

    i++; j--;
}

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);

    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
```