

# Лабораторная работа №7

## «Массивы многомерные»

### Теоретическое введение

#### Многомерные массивы

Многомерный массив — это **массив массивов**. Например, объявим двумерный массив, содержащий 5 массивов по 10 элементов в каждом:

```
int arr2d[5][10];
```

В данном примере:

- 5 — это *старшая* размерность, 10 — *младшая*;
- всего в массиве содержится  $5 \times 10 = 50$  элементов;
- элементы имеют номера от [0][0] до [4][9];
- если массив объявлен внутри функции, его элементы никак не инициализированы и содержат «мусор».

Лучше всего представлять себе такой массив как матрицу, содержащую 5 строк и 10 столбцов:

arr2d[0][0]	arr2d[0][1]	...	arr2d[0][9]
arr2d[1][0]	arr2d[1][1]	...	arr2d[1][9]
arr2d[2][0]	arr2d[2][1]	...	arr2d[2][9]
arr2d[3][0]	arr2d[3][1]	...	arr2d[3][9]
arr2d[4][0]	arr2d[4][1]	...	arr2d[4][9]

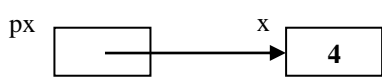
(arr2d[0] — это первая строка целиком, arr2d[1] — вторая и т.д.)

#### Указатели

Указатель — это переменная, которая хранит в себе *адрес* некоторой ячейки памяти. При объявлении такой переменной обязательно надо указывать *тип* того значения, которое хранится по этому адресу (для того, чтобы мог работать механизм контроля типов). Тип самой переменной-указателя обозначается как “Т\*”, где Т — тип значения, на которое он будет указывать. Тот факт, что некоторая переменная-указатель хранит в себе адрес некоторой другой области памяти, на иллюстрациях будем изображать стрелочкой:

Например, пусть указатель px указывает на целочисленную переменную x:

Таблица 1. Графическое изображение переменных и указателей.

Код на C++	Иллюстрация
<pre>int x = 4; int* px; px = &amp;x;</pre>	

Доступ к самой ячейке, на которую указывает указатель, осуществляется при помощи операции *разыменования* указателя (\*). Если *изменить* значение в ячейке посредством разыменования указателя на нее, то это изменение будет заметно и при любом другом способе обращения к этой ячейке:

Таблица 2. Обращение к ячейке памяти через указатель.

Код на C++	Результат на экране
<pre>int x = 4; int* px; px = &amp;x; cout &lt;&lt; x &lt;&lt; " " &lt;&lt; *px &lt;&lt; '\n'; x = 2; cout &lt;&lt; x &lt;&lt; " " &lt;&lt; *px &lt;&lt; '\n'; *px = 3; cout &lt;&lt; x &lt;&lt; " " &lt;&lt; *px &lt;&lt; '\n';</pre>	<pre>4 4 2 2 3 3</pre>

## Указатели на массивы и арифметика указателей

Особенно велика польза от указателей, когда они указывают не просто на какую-то другую переменную, а на некоторую ячейку массива:

```
// создадим двумерный массив
int arr2d[5][10];
// ...тут пропущен код заполнения массива arr2d
cout << "Какую строку массива Вам распечатать: ";
int row;
cin >> row;

// пусть указатель указывает на первый элемент row-той строки
int* p = &arr2d[row][0];

// распечатаем эту строку, используя доступ через указатель
for(int i=0; i<10; i++){
    cout << *(p+i) << ' ';
}
cout << '\n';
```

В данном примере выражение «\*(p+i)» означает:

- вычислить адрес i-ой ячейки, лежащей после той, на которую указывает указатель p (т.е. просто прибавить<sup>1</sup> к хранимому в нем адресу число i);
- разыменовать полученный адрес — т.е. обратиться к самой ячейке, лежащей по этому адресу (а затем вывести полученное значение на экран).

## Самое главное, что надо знать про указатели

В языке Си (и Си++) указатели и массивы *взаимозаменяемы*. В том смысле, что указатель можно использовать как массив:

```
int* p = &some_arr[0];
cout << p[i];           // эквивалентно: cout << *(p+i);
```

а массив — как указатель:

```
int arr[10];
int* p = arr;           // эквивалентно: int* p = &arr[0];
```

## Динамические массивы

Динамический массив — это массив, расположенный в *динамической памяти* и доступный *только через указатель*. Для того чтобы получить в свое распоряжение некоторый фрагмент этой памяти, программе надо явно запросить его при помощи библиотечной функции `malloc` (из `<cstdlib>`):

```
cout << "How many elements: ";
int N;
cin >> N;
int* arr = (int*)malloc(N*sizeof(int)); // запросить N ячеек по 4 байта
```

После этого — как-то с полученной памятью поработать. Исключительно в качестве примера приведем следующий код:

```
int sum = 0;
for(int i=0; i<N; i++){
    cin >> arr[i]    // считываем с клавиатуры N эл-тов в наш массив
    sum += arr[i];   // в это же время считаем их сумму
}
```

---

<sup>1</sup> На самом деле надо прибавлять, конечно же, не просто i, а `i*sizeof(int)`. Именно такой код в реальности и генерирует компилятор, когда встречается подобное выражение. А для того, чтобы он мог так сделать, мы и указываем тип указателя — `int-со-звездочкой`.

```
int average = sum / N; // вычисляем (округленное) среднее арифметическое
// выводим обратно пользователю только те элементы,
// которые не меньше average
for(int i=0; i<N; i++){
    if(arr[i] >= average)
        cout << arr[i] << ' ';
}
cout << '\n';
```

После окончания работы с динамическим массивом его надо вернуть обратно операционной системе при помощи функции `free()`:

```
free(arr);
```

## Динамические многомерные массивы

Раз уж в языке Си бывают массивы массивов, то логично ожидать и наличия массивов указателей:

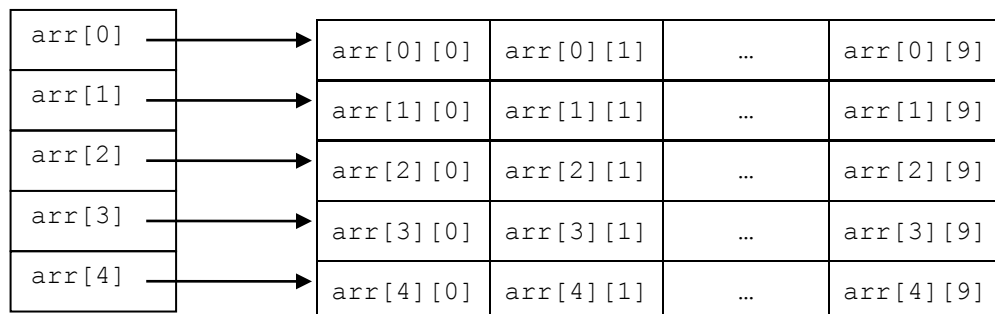
```
int* arr[5]; // массив из 5 указателей на int
```

Каждый из этих 5-ти указателей можно использовать по своему усмотрению. Например — выделить под каждый из них по блоку динамической памяти на 10 `int`'ов:

```
for(int i=0; i<5; i++){
    arr[i] = (int*)malloc(10*sizeof(int));
}
```

Графически это выглядит следующим образом:

arr



Работа с таким массивом осуществляется по общим правилам обращения с массивами и указателями. Например, обратимся двумя эквивалентными способами к 3-му (считая от 0) элементу второй (считая от 0) строки:

```
cin >> *(arr[2]+3); // эквивалентно: arr[2][3]
cout << arr[2][3]; // эквивалентно: *(arr[2]+3)
```

Если же еще вспомнить, что просто `arr` — это то же самое, что `&arr[0]` (в силу эквивалентности массивов и указателей), то это же обращение можно осуществить еще и третьим способом:

```
cout << *(* (arr+2)+3);
```

Когда такой массив больше не нужен — его аналогичным образом нужно удалить (вернуть память системе):

```
for(int i=0; i<5; i++){
    free(arr[i]);
}
```

## Настоящие динамические многомерные массивы

На практике описанный выше «гибридный» вариант используется редко. Гораздо чаще «главный» массив (массив указателей на массивы) тоже создается как динамический.

Ну а поскольку в нем хранятся указатели (элементы типа `int*`), то указатель на этот массив должен иметь на одну звездочку больше (`int**` — указатель на указатель на `int`). Обращение с элементами такого массива полностью аналогично приведенному выше примеру с элементом `arr[2][3]`. Создание же его и удаление — требуют на одну операцию выделения (и освобождения) памяти больше:

```
int** arr;           // указатель на массив указателей на массив int'ов

arr = (int**)malloc(5*sizeof(int*));           // создаем 5 указателей
for(int i=0; i<5; i++){
    arr[i] = (int*)malloc(10*sizeof(int)); // 5 раз создаем по 10 эл-тов
}

// некоторые операции с элементами для иллюстрации:
cin >> *(arr[2]+3);
cout << arr[2][3];

// удаляем - в обратном порядке:
for(int i=0; i<5; i++){
    free(arr[i]);           // эквивалентно: *(arr+i)
}
free(arr);
```

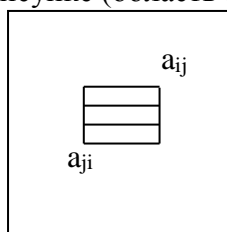
## Задания

### Правила выполнения:

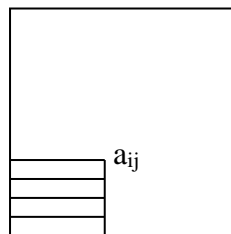
1. Реализовать ввод исходных данных как с клавиатуры, так и посредством генератора случайных чисел.
2. Задание сделать тремя способами: с помощью статических массивов, с помощью динамических массивов и с помощью «гибридных» массивов (в которых «старший» массив указателей — статический, а элементы этого массива уже указывают на области в динамической памяти). Естественно, в статическом массиве обе размерности придется сделать константными, в гибридном — только старшую, и только в динамическом массиве обе размерности задаются с клавиатуры. Подробности см. в «теоретическом введении».
3. Максимальная оценка — 8 баллов.
4. Правила оценивания см. в «критерии\_оценивания.doc».

### Задания:

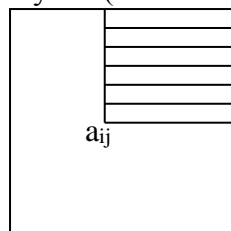
1. Для заданной целочисленной матрицы  $A$  размерностью  $N \times M$  сформировать матрицу  $B$  такой же размерностью, в которой элемент равен 0 в том случае, если все соседи соответствующего элемента матрицы  $A$  меньше его самого, и 1 в противном случае.
2. Определить норму заданной матрицы  $A = \|a_{ij}\|$ , т. е. число  $\max_i \left( \sum_j |a_{ij}| \right)$
3. Определить норму заданной матрицы  $B = \|b_{ij}\|$ , т.е. число  $\max_j \left( \sum_i |b_{ij}| \right)$
4. Расстояние между каждой  $k$ -ой и  $l$ -ой строками матрицы  $A = \|a_{ij}\|$  определяется как  $\sum_{j=1}^n |a_{kj}| * |a_{lj}|$ . Указать номер строки, максимально удаленной от первой строки матрицы.
5. Элемент матрицы называется локальным минимумом, если он строго меньше всех своих соседей. Найти максимум среди всех локальных минимумов заданной матрицы.
6. Найти максимальный среди всех элементов тех строк заданной матрицы, которые упорядочены (либо по возрастанию, либо по убыванию).
7. Подсчитать количество столбцов целочисленной квадратной матрицы порядка  $n$ , которые составлены из попарно различных чисел.
8. Дана квадратная матрица  $A$   $n$ -го порядка. Построить матрицу  $B$ , элемент  $b_{ij}$  которой равен сумме элементов данной матрицы, расположенных в области, определяемой индексами  $i, j$  так, как показано на рисунке (область заштрихована).



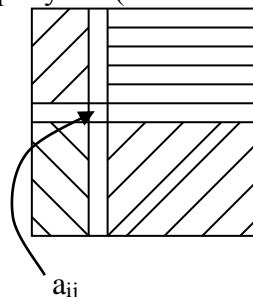
9. Дана квадратная матрица  $A$   $n$ -го порядка. Построить матрицу  $B$ , элемент  $b_{ij}$  которой равен сумме элементов данной матрицы, расположенных в области, определяемой индексами  $i, j$  так, как показано на рисунке (область заштрихована).



10. Дана квадратная матрица  $A$   $n$ -го порядка. Построить матрицу  $B$ , элемент  $b_{ij}$  которой равен сумме элементов данной матрицы, расположенных в области, определяемой индексами  $i, j$  так, как показано на рисунке (область заштрихована).



11. Дана квадратная матрица  $A$   $n$ -го порядка. Построить матрицу  $B$ , элемент  $b_{ij}$  которой равен сумме элементов данной матрицы, расположенных в области, определяемой индексами  $i, j$  так, как показано на рисунке (область заштрихована).



12. Дана квадратная матрица порядка  $n$ . Поставить  $i$ -ую строку на первое место, сдвинув циклически остальные строки.
13. Даны целые положительные числа  $M, N$ , число  $D$  и набор из  $M$  чисел. Сформировать матрицу размера  $M \times N$ , у которой первый столбец совпадает с исходным набором чисел, а элементы каждого следующего столбца равны сумме соответствующего элемента предыдущего столбца и числа  $D$  (в результате каждая строка матрицы будет содержать элементы *арифметической прогрессии*).
14. Даны целые положительные числа  $M, N$ , число  $Q$  и набор из  $N$  чисел. Сформировать матрицу размера  $M \times N$ , у которой первая строка совпадает с исходным набором чисел, а элементы каждой следующей строки равны соответствующему элементу предыдущей строки, умноженному на  $Q$  (в результате каждый столбец матрицы будет содержать элементы *геометрической прогрессии*).
15. Дана целочисленная матрица  $M \times N$ . Переставляя строки заданной матрицы, расположить их в соответствии с ростом характеристик. Характеристикой строки целочисленной матрицы назовем сумму ее положительных четных элементов.
16. Дана целочисленная матрица  $M \times N$ . Переставляя строки заданной матрицы, расположить их в соответствии с ростом характеристик. Характеристикой столбца целочисленной матрицы назовем сумму модулей его отрицательных нечетных элементов.
17. Дана матрица размера  $M \times N$ . Поменять местами столбец с номером 1 и последний из столбцов, содержащих только положительные элементы. Если требуемых столбцов нет, то вывести матрицу без изменений.
18. В данной действительной квадратной матрице порядка  $N$  поменять местами строку, в которой расположен элемент с наибольшим значением, со строкой, содержащей элемент с наименьшим значением (предполагается, что такие элементы единственны).

19. Дана матрица размера  $M \times N$ . Поменять местами столбец с номером  $N$  и первый из столбцов, содержащих только отрицательные элементы. Если требуемых столбцов нет, то вывести матрицу без изменений.
20. Дана матрица размера  $M \times N$ . Зеркально отразить ее элементы относительно горизонтальной оси симметрии матрицы (при этом поменяются местами строки с номерами 1 и  $M$ , 2 и  $M - 1$  и т. д.).
21. Дана матрица размера  $M \times N$ . Зеркально отразить ее элементы относительно вертикальной оси симметрии матрицы (при этом поменяются местами столбцы с номерами 1 и  $N$ , 2 и  $N - 1$  и т. д.).
22. Дана матрица размера  $M \times N$ . Элемент матрицы называется ее *локальным минимумом*, если он меньше всех окружающих его элементов. Заменить все локальные минимумы данной матрицы на нули. При решении допускается использовать вспомогательную матрицу.
23. Дана матрица размера  $M \times N$ . Элемент матрицы называется ее *локальным максимумом*, если он больше всех окружающих его элементов. Поменять знак всех локальных максимумов данной матрицы на противоположный. При решении допускается использовать вспомогательную матрицу.
24. Дана квадратная матрица  $A$  порядка  $M$ . Найти сумму элементов каждой ее диагонали, параллельной главной (начиная с одноэлементной диагонали  $A_{1,M}$ ).
25. Дана квадратная матрица  $A$  порядка  $M$ . Найти сумму элементов каждой ее диагонали, параллельной побочной (начиная с одноэлементной диагонали  $A_{1,1}$ ).
26. В квадратной целочисленной матрице порядка  $n$  определить, что больше: сумма всех элементов матрицы, лежащих над главной диагональю, или произведение положительных элементов, лежащих ниже побочной диагонали.
27. Дана матрица размера  $M \times N$ . Вывести ее элементы в следующем порядке: первая строка слева направо, вторая строка справа налево, третья строка слева направо, четвертая строка справа налево и т. д.
28. Дана матрица размера  $M \times N$ . Вывести ее элементы в следующем порядке: первый столбец сверху вниз, второй столбец снизу вверх, третий столбец сверху вниз, четвертый столбец снизу вверх и т. д.
29. Дана квадратная матрица  $A$  порядка  $M$ . Начиная с элемента  $A_{1,1}$ , вывести ее элементы следующим образом («уголками»): все элементы первой строки; элементы последнего столбца, кроме первого (уже выведенного) элемента; оставшиеся элементы второй строки; оставшиеся элементы предпоследнего столбца и т. д.; последним выводится элемент  $A_{M,1}$ .
30. Дана квадратная матрица  $A$  порядка  $M$ . Начиная с элемента  $A_{1,1}$ , вывести ее элементы следующим образом («уголками»): все элементы первого столбца; элементы последней строки, кроме первого (уже выведенного) элемента; оставшиеся элементы второго столбца; оставшиеся элементы предпоследней строки и т. д.; последним выводится элемент  $A_{1,M}$ .
31. Элемент матрицы называется седловой точкой, если он является одновременно наименьшим в своей строке и наибольшим в своем столбце. Выяснить, имеются ли в заданной матрице седловые точки и, если имеются, то указать индексы всех таких точек.
32. Даны две целочисленные квадратные матрицы порядка  $n$ . Построить вектор  $b = \{b_i\}_{i=1..n}$ , элементы которого  $b_i = 1$ , если выполняются условия, и  $b_i = 0$  в противном случае.
  - а) все элементы  $i$ -ой строки первой матрицы больше соответствующих элементов  $i$ -ой строки второй матрицы;
33. Даны две целочисленные квадратные матрицы порядка  $n$ . Построить вектор  $b = \{b_i\}_{i=1..n}$ , элементы которого  $b_i = 1$ , если выполняются условия, и  $b_i = 0$  в противном случае.
  - б) все элементы  $i$ -ых строк первой и второй матриц отрицательны;

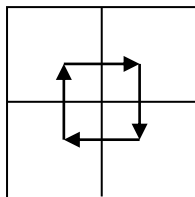
34. Даны две целочисленные квадратные матрицы порядка  $n$ . Построить вектор  $b = \{b_i\}_{i=1..n}$ , элементы которого  $b_i = 1$ , если выполняются условия, и  $b_i = 0$  в противном случае.

с)  $i$ -ые строки первой и второй матриц содержат вместе не более трех положительных элементов;

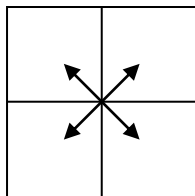
35. Даны две целочисленные квадратные матрицы порядка  $n$ . Построить вектор  $b = \{b_i\}_{i=1..n}$ , элементы которого  $b_i = 1$ , если выполняются условия, и  $b_i = 0$  в противном случае.

д) количество отрицательных и неотрицательных элементов  $i$ -ой строки первой матрицы совпадают с количеством отрицательных и неотрицательных элементов  $i$ -ой строки второй матрицы.

36. Дана действительная квадратная матрица порядка  $2n$ . Получить новую матрицу, переставляя ее блоками размера  $n \times n$  соответственно:



37. Дана действительная квадратная матрица порядка  $2n$ . Получить новую матрицу, переставляя ее блоками размера  $n \times n$  соответственно:



38. Дана действительная квадратная матрица порядка  $2n$ . Получить новую матрицу, переставляя ее блоками размера  $n \times n$  соответственно:

