Двумерные массивы

Объявление, ввод и вывод двумерного массива

Мы помним определение массива - набор пронумерованных элементов, идущих в памяти последовательно. К каждому элементу массива можно обратиться, указав один индекс - номер этого элемента. Но бывает так, что одной нумерации недостаточно. Например, клетки игрового поля (шахматы, сапер) удобнее нумеровать парой чисел, элементы таблицы (Excel, классный журнал) тоже естественным образом нумеруются уже не одним индексом, а двумя. Попробуйте привести свои примеры.[1]

Таким образом естественно возникает аналог обычного массива. Например, массив с двойной нумерацией элементов - двумерный массив

Создать его можно следующим образом: `int A[n][m]`. Данное объявление создает массив из `n` объектов, каждый из которых в свою очередь является массивом типа `int [m]`. Тогда `A[i]`, где `i` принимает значения от `0` до `n-1`, будет в свою очередь одним из `n` созданных обычных массивов, и обратиться к элементу с номером `j` в этом массиве можно как к `A[i][j]`.

Подобные объекты (массивы массивов) также называют двумерными массивами. Двумерные массивы можно представлять в виде квадратной таблицы, в которой первый индекс элемента означает номер строки, а второй индекс – номер столбца. Например, массив `A[3][4]` будет состоять из 12 элементов и его можно записать в виде

```
A[0][0]  A[0][1]  A[0][2]  A[0][3]
A[1][0]  A[1][1]  A[1][2]  A[1][3]
A[2][0]  A[2][1]  A[2][2]  A[2][3]
```

Для считывания, вывода на экран и обработки двумерных массивов необходимо использовать вложенные циклы. Первый цикл – по первому индексу (то есть по всем строкам), второй цикл – по второму индексу, то есть по всем элементам в строках (столбцам). Например, вывести на экран двумерный массив в виде таблицы, разделяя элементы в строке одним пробелом можно следующим образом:

```
int A[n][m];

for (int i = 0; i < n; ++i)
{   // Выводим на экран строку i
    for (int j = 0; j < m; ++j)
    {
        cout << A[i][j] << " ";
    }
    cout << endl;
    // Строка завершается символом перехода на новую строку
}
```

[1] А БЫВАЮТ ЛИ ОБЪЕКТЫ, ГДЕ НУМЕРАЦИЯ ТРОЙКАМИ НАТУРАЛЬНЫХ ЧИСЕЛ УМЕСТНЕЕ? А БОЛЬШИМ КОЛИЧЕСТВОМ ЭЛЕМЕНТОВ?

А считать двумерный массив с клавиатуры можно при помощи еще более простого алгоритма (массив вводится по строкам, то есть в порядке, соответствующему первому примеру):

```
for (i = 0; i < n; ++i)
{
    for (j = 0; j < m; ++j)
    {
        cin >> A[i][j];
    }
}
```

Обработка двумерного массива

Обработка двумерных массивов производится аналогичным образом. Например, если мы хотим записать в массив таблицу умножения, то есть присвоить элементу `A[i][j]` значение `i * j`, это можно сделать следующим образом при помощи вложенных циклов:

```
for (i = 0; i < n; ++i)
{
    for (j = 0; j < m; ++j)
    {
        A[i][j] = i * j;
    }
}
```

Рассмотрим более сложную задачу и несколько способов ее решения. Пусть дан квадратный двумерный массив `int A[n][n]`. Необходимо элементам, находящимся на главной диагонали проходящей из левого верхнего угла в правый нижний (то есть тем элементам `A[i][j]`, для которых `i==j`) присвоить значение `1`, элементам, находящимся выше главной диагонали – значение 0, элементам, нахощящимся ниже главной диагонали – значение 2. То есть получить такой массив (пример для `n==4`):

```
1 0 0 0
2 1 0 0
2 2 1 0
2 2 2 1
```

Рассмотрим несколько способов решения этой задачи. Элементы, которые лежат выше главной диагонали – это элементы `A[i][j]`, для которых `i<j`, а для элементов ниже главной диагонали `i>j`. Таким образом, мы можем сравнивать значения `i` и `j` и по ним определять значение `A[i][j]`. Получаем следующий алгоритм:

```
for (i = 0; i < n; ++i)
{
    for (j = 0; j < n; ++j)
    {
        if (i < j)
        {
            A[i][j] = 0;
        }
        else if (i > j)
        {
            A[i][j] = 2;
        }
```

```
        else
        {
            A[i][j] = 1;
        }
    }
}
```

Данный алгоритм плох, поскольку выполняет одну или две инструкции `if` для обработки каждого элемента. Если мы усложним алгоритм, то мы сможем обойтись вообще без условных инструкций.

Сначала заполним главную диагональ, для чего нам понадобится один цикл:

```
for (i = 0; i < n; ++i)
{
    A[i][i] = 1;
}
```

Затем заполним значением `0` все элементы выше главной диагонали, для чего нам понадобится в каждой из строк с номером `i` присвоить значение элементам `A[i][j]`для `j`=`i+1`, ..., `n-1`. Здесь нам понадобятся вложенные циклы:

```
for (i = 0; i < n; ++i)
{
    for (j = i + 1; j < n; ++j)
    {
        A[i][j] = 0;
    }
}
```

Аналогично присваиваем значение `2` элементам `A[i][j]` для `j`=`0`, ..., `i-1`:

```
for (i = 0; i < n; ++i)
{
    for (j = 0; j < i; ++j)
    {
        A[i][j] = 2;
    }
}
```

Можно также внешние циклы объединить в один и получить еще одно, более компактное решение:

```
for (i = 0; i < n; ++i)
{   // Заполняем строку с номером i
    for (j = 0; j < i; ++j)
    {
        A[i][j] = 2;    // Сначала пишем 2 ниже диагонали
    }
    A[i][j] = 1;        // После завершения предыдущего цикла i==j, пишем 1
    for (++j; j < n; ++j)  // Цикл начинаем с увеличения j на 1
    {
        A[i][j] = 0;    // Записываем 0 выше диагонали
    }
}
```

Форматирование чисел при выводе

Допустим, мы заполним массив таблицей умножения: `A[i][j]=i*j` как в примере в начале раздела. Если мы теперь попробуем вывести этот массив на экран, разделяя элементы в строке одним пробелом, то из-за того, что числа имеют различную длину столбцы таблицы окажутся неровными:

```
0 0 0 0 0 0 0 0 0 0
0 1 2 3 4 5 6 7 8 9
0 2 4 6 8 10 12 14 16 18
0 3 6 9 12 15 18 21 24 27
```

Для того, чтобы получить ровные столбцы необходимо, выводить числа так, чтобы одно выводимое число имело ширину, например, ровно в 3 символа, а “лишние” позиции были бы заполнены пробелами. Тогда получится следующая таблица:

```
0   0   0   0   0   0   0   0   0   0
0   1   2   3   4   5   6   7   8   9
0   2   4   6   8 10 12 14 16 18
0   3   6   9 12 15 18 21 24 27
```

Для того, чтобы выводимое число или строка имело ровно заданную ширину, необходимо перед выводом его на экран для потока `cout` вызвать метод `width` с параметром `3`. Данный метод устанавливает ширину поля для выводимого значения. Получим следующую программу для вывода:

```
for(int i = 0; i < n; ++i)
{
    for(int j = 0; j < m; ++j)
    {
        cout.width(3);
        cout << A[i][j];
    }
    cout << endl;
}
```

Заметим, что мы теперь не выводим пробел после каждого числа, поскольку мы добавили этот пробел к ширине выводимого поля. Функция `width` действует однократно, только на следующее выводимый в поток значение, поэтому ее нужно вызывать перед каждым выводом числа на экран.

Внимание! Если выводимое число или строка имеет большую длину, чем это было установлено функцией `width`, то это число или строка будут выведены полностью, а не будет обрезано до указанного значения. То есть предпочтительней вывести результат некрасиво, нежели неверно.

Упражнения

B. Побочная диагональ

Дано число `n`. Создайте массив размером `n×n` и заполните его по следующему правилу:

Числа на диагонали, идущей из правого верхнего в левый нижний угол равны 1.

Числа, стоящие выше этой диагонали, равны 0.

Числа, стоящие ниже этой диагонали, равны 2.

Полученный массив выведите на экран. Числа в строке разделяйте одним пробелом.

Ввод	Вывод
4	0 0 0 1 0 0 1 2 0 1 2 2 1 2 2 2

C. Поменять строки

Дан двумерный массив. Поменяйте в нем первую и последнюю строку. Полученный массив выведите на экран.

Программа получает на вход два числа: количество строк `n` в массиве и количество столбцов `m`. Далее идет `n` строк, каждая из которых содержит `m` чисел - элементы массива.

Выведите массив на экран разделяя числа в строке одним пробелом.

Ввод	Вывод
3 4 11 12 13 14 21 22 23 24 31 32 33 34	31 32 33 34 21 22 23 24 11 12 13 14

D. Поменять столбцы

Дан двумерный массив и два числа: `i` и `j`. Поменяйте в массиве столбцы с номерами `i` и `j` и выведите результат.

Программа получает на вход размеры массива `n` и `m`, затем элементы массива, затем числа `i` и `j`.

Ввод	Вывод
3 4 11 12 13 14 21 22 23 24 31 32 33 34 0 1	12 11 13 14 22 21 23 24 32 31 33 34

E. Симметричен ли массив?

Дано число n и массив размером n×n. Проверьте, является ли этот массив симметричным относительно главной диагонали. Выведите слово “YES”, если массив симметричный, и слово “NO” в противном случае.

Ввод	Вывод
3 0 1 2 1 2 3 2 3 4	YES

F. Транспонировать прямоугольную матрицу

Дан двумерный массив размером n×m. Симметричный ему относительно главной диагонали массив называется транспонированным к данному. Он имеет размеры m×n: строки исходного массива становятся столбцами транспонированного, столбцы исходного массива становятся строками транспонированного.

Для данного массива постройте транспонированный массив и выведите его на экран. .

Ввод	Вывод
3 4 11 12 13 14 21 22 23 24 31 32 33 34	11 21 31 12 22 32 13 23 33 14 24 34

G. Состязания - 1

В метании молота состязается n спортсменов. Каждый из них сделал m бросков. Победителем считается тот спортсмен, у которого максимален наилучший результат.

Программа получает на вход количество спортсменов n, затем количество бросков m. Далее идет n строк по m чисел в каждой, в i-й строке записаны результаты бросков i-го спортсмена.

Определите победителя соревнований. Программа должна вывести значение наилучшего броска, затем номер спортсмена, совершившего такой бросок (если таким спортсменов несколько - то номер первого из них), затем номер броска, на котором этот спортсмен совершил такой бросок (если таких несколько - то наименьший номер).

Ввод	Вывод
4 3 1 4 2 5 2 5 5 1 4 1 2 4	5 2 1

H. Состязания - 2

В условиях предыдущей задачи выведите число спортсменов, разделивших первое место, то есть показавших наилучший результат.

Ввод	Вывод
4 3 1 2 3 4 5 6 6 2 5 2 3 4	2

I. Состязания - 3

Победителем считается тот спортсмен, у которого сумма бросков за все попытки максимальная.

Выведите наибольшую сумму бросков для одного спортсмена, затем номер спортсмена, у которого такая сумма максимальна. Если таких спортсменов несколько - выведите номер первого из них.

Ввод	Вывод
4 3 5 6 7 6 6 7 7 6 6 4 3 5	19 2

J. Состязания - 4

В соревнованиях побеждает спортсмен, у которого максимален наилучший бросок. Если таких несколько, то побеждает тот, у которого наибольшая сумма по всем броскам. Если таких несколько, то побеждает тот, у которого меньше номер.

Выведите номер победившего спортсмена.

Ввод	Вывод
4 3 8 8 8 5 9 3 9 4 7 6 6 2	3

K. Кинотеатр

В кинотеатре n рядов по m мест в каждом. В двумерном массиве хранится информация о проданных билетах, число 1 означает, что билет на данное место уже продано, число 0 означает, что место свободно. Поступил запрос на продажу k билетов на соседние места в одном ряду. Определите, можно ли выполнить такой запрос.

Программа получает на вход числа n, m, k. Далее идет n строк, содержащих m чисел (0 или 1), разделенных пробелами.

Программа должна вывести номер ряда, в котором есть k подряд идущих свободных мест. Если таких рядов несколько, то выведите номер наименьшего подходящего ряда. Если подходящего ряда нет, выведите число 0.

Ввод	Вывод
3 4 2 0 1 0 1 1 0 0 1 1 1 1 1	2
3 3 3 0 1 0 1 0 0 1 1 1	0

L. Треугольник Паскаля - 1

Даны два числа n и m. Создайте массив `n×m` и заполните его по следующим правилам:

Числа, стоящие в строке 0 или в столбце 0 равны 1 (`A[0][j]=1`, A[i][0]=1). Для всех остальных элементов массива `A[i][j]=A[i-1][j]+A[i][j-1]`, то есть каждый элемент равен сумме двух элементов, стоящих слева и сверху от него. Выведите данный массив на экран, отводя на вывод каждого элемента массива ровно 6 символов.

Ввод	Вывод
4 6	1 1 1 1 1 1 1 2 3 4 5 6 1 3 6 10 15 21 1 4 10 20 35 56

M. Таблица умножения

Даны числа n и m. Создайте двумерый массив размером n×m и заполните его таблицей умножения по формуле `A[i][j]=i*j`. **При заполнении массива нельзя использовать вложенные циклы.**

Выведите получившийся массив на экран (при выводе можно использовать вложенные циклы), отводя на вывод каждого числа ровно 4 символа.

Ввод	Вывод
4 6	0 0 0 0 0 0 0 1 2 3 4 5 0 2 4 6 8 10 0 3 6 9 12 15