Массив (тип данных)

**Массив** (в некоторых языках программирования также таблица, ряд, матрица) — структура данных, хранящая набор значений (элементов массива), идентифицируемых по индексу или набору индексов, принимающих целые (или приводимые к целым) значения из некоторого заданного непрерывного диапазона. Одномерный массив можно рассматривать как реализацию абстрактного типа данных — вектор.

Размерность массива — это количество индексов, необходимое для однозначной адресации элемента в рамках массива. По количеству используемых индексов массивы делятся на одномерные, двумерные, трёхмерные и т. д.

Форма или структура массива — сведения о количестве размерностей и размере (протяжённости) массива по каждой из размерностей[[3]](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)#cite_note-_e9ff6c56b259ec52-3); может быть представлена одномерным массивом.

Особенностью массива как структуры данных (в отличие, например, от связного списка) является константная вычислительная сложность доступа к элементу массива по индексу. Массив относится к структурам данных с произвольным доступом.

В простейшем случае массив имеет константную длину по всем размерностям и может хранить данные только одного, заданного при описании, типа. Ряд языков поддерживает также **динамические массивы**, длина которых может изменяться по ходу работы программы, и **гетерогенные массивы**, которые могут в разных элементах хранить данные различных типов.

## **Общее описание**

Массив — упорядоченный набор элементов, каждый из которых хранит одно значение, идентифицируемое с помощью одного или нескольких индексов. В простейшем случае массив имеет постоянную длину и хранит единицы данных одного и того же типа, а в качестве индексов выступают целые числа.

Количество используемых индексов массива может быть различным: массивы с одним индексом называют одномерными, с двумя — двумерными, и т. д. Одномерный массив — нестрого соответствует вектору в математике; двумерный («строка», «столбец»)— матрице. Чаще всего применяются массивы с одним или двумя индексами; реже — с тремя; ещё большее количество индексов — встречается крайне редко.

**Пример фиксированного массива на языке Паскаль**

*{Одномерный массив целых чисел.*

*Нумерация элементов от 1 до 15}*

a: **array** [1..15] **of** Integer;

*{Двумерный массив символов.*

*Нумерация по столбцам по типу Byte (от 0 до 255)*

*по строкам от 1 до 5}*

multiArray : **array** [Byte, 1..5] **of** Char;

*{Одномерный массив из строк.*

*Нумерация по типу word (от 0 до 65536)}*

rangeArray : **array** [Word] **of** **String**;

**Пример фиксированного массива на С/С++**

int Array[10]; *// Одномерный массив: целых чисел, размера 10;*

*// Нумерация элементов — от 0 до 9.*

double Array[12][15]; *// Двумерный массив:*

*// вещественных чисел двойной точности,*

*// размера 12 на 15;*

*// Нумерация: по строкам — от 0 до 11,*

*// по столбцам — от 0 до 14.*

В некоторых языках программирования многомерные массивы создаются на основе одномерных, у которых элементы являются массивами[[6]](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)#cite_note-McMillan2014-6).

**Пример двумерного массива на JavaScript**

*//Создание двумерного массива чисел:*

**var** array = [

[11, 12, 13, 14, 15, 16], *// Первая строка-массив*

[21, 22, 23, 24, 25, 26], *// Вторая*

[31, 32, 33, 34, 35, 36] *// Третья*

];

*// Вывод массива на консоль:*

array.forEach((subArray) => { *// Для каждого под-массива,*

subArray.forEach((item) => { *// для каждого его элемента,*

console.log(item); *// — вывести этот элемент на консоль.*

});

});

Поддержка индексных массивов (свой синтаксис объявления, функции для работы с элементами и т. д.) есть в большинстве высокоуровневых языков программирования. Максимально допустимая размерность массива, типы и диапазоны значений индексов, ограничения на типы элементов определяются языком программирования и (или) конкретным транслятором.

В языках программирования, допускающих объявления программистом собственных типов, как правило, существует возможность создания типа «массив». В определении такого типа задаются типы и/или диапазоны значений каждого из индексов и тип элементов массива. Объявленный тип в дальнейшем может использоваться для определения переменных, формальных параметров и возвращаемых значений функций. Некоторые языки поддерживают для переменных-массивов операции присваивания (когда одной операцией всем элементам массива присваиваются значения соответствующих элементов другого массива).

**Объявление типа «массив» в языке Паскаль**

**type**

TArrayType = **array** [0..9] **of** Integer;

*(\* Массивы, имеющие заданные параметры:*

*1. Размер — 10 ячеек;*

*2. Тип элементов, пригодных для хранения —*

*— целые числа диапазона [−32 768; 32 767],*

*— объявляются типом операндов, называющимся "TArrayType". \*)*

**var**

arr1, arr2, arr3: TArrayType;

*(\* Объявление трёх переменных-массивов одного типа*

*(вышеуказанного "TArrayType"). \*)*

В языке программирования APL массив является основным типом данных (при этом нуль-мерный массив называется скаляром, одномерный — вектором, двумерный — матрицей)[[4]](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)#cite_note-_906d4f9efcd27ba4-4). Помимо присваивания массивов в этом языке поддерживаются операции векторной и матричной арифметики, каждая из которых выполняется одной командой, операции сдвига данных в массивах, сортировка строк матрицы и т. п.

## **Специфические типы массивов**

### Динамические массивы

Динамическими называются массивы, размер которых может изменяться во время выполнения программы. Обычные (не динамические) массивы называют ещё *фиксированными* или *статическими*.

Динамические массивы могут реализовываться как на уровне языка программирования, так и на уровне системных библиотек. Во втором случае динамический массив представляет собой объект стандартной библиотеки, и все операции с ним реализуются в рамках той же библиотеки. Так или иначе, поддержка динамических массивов предполагает наличие следующих возможностей:

1. Описание динамического массива. На уровне языка это может быть специальная синтаксическая конструкция, на уровне библиотеки — библиотечный тип данных, значение которого объявляется стандартным образом. Как правило, при описании (создании) динамического массива указывается его начальный размер, хотя это и не обязательно.
2. Операция определения текущего размера динамического массива.
3. Операция изменения размера динамического массива.

Ниже приведён пример конструкций для работы с динамическими массивами на Delphi.

**var** *// Описания динамических массивов*

byteArray : **Array** **of** Byte; *// Одномерный массив*

multiArray : **Array** **of** **Array** **of** **string**; *// Многомерный массив*

...

SetLength(byteArray, 1); *// Установка размера массива в 1 элемент.*

byteArray[0] := 16; *// Запись элемента.*

SetLength(byteArray, Length(byteArray)+1); *// Увеличение размера массива на единицу*

byteArray[Length(byteArray) - 1] := 10; *// Запись значения в последний элемент.*

WriteLn(byteArray[Length(byteArray) - 1]); *// Вывод последнего элемента массива.*

...

SetLength(multiArray, 20, 30); *// Установка размера двумерного массива*

multiArray[10,15] := 12;

SetLength(multiArray, 10, 15); *// Уменьшение размера*

WriteLn(Length(multiArray), ' ', Length(multiArray[0])

### Гетерогенные массивы

*Гетерогенным* называется массив, в разные элементы которого могут быть непосредственно записаны значения, относящиеся к различным типам данных. Массив, хранящий указатели на значения различных типов, не является гетерогенным, так как собственно хранящиеся в массиве данные относятся к единственному типу — типу «указатель». Гетерогенные массивы удобны как универсальная структура для хранения наборов данных произвольных типов. Реализация гетерогенности требует усложнения механизма поддержки массивов в трансляторе языка.

## **Реализация**

Типовым способом реализации статического гомогенного (хранящего данные одного типа) массива является следующий:

1. Под массив выделяется непрерывный блок памяти объёмом S\*m1\*m2\*m3…mn, где S — размер одного элемента, а m1…mn — размеры диапазонов индексов (то есть количество значений, которые может принимать соответствующий индекс).
2. При обращении к элементу массива A[i1, i2, i3, …, in] адрес соответствующего элемента вычисляется как B+S\*((…(i1p\*m1+i2p)\*m2+…+i(n-1)p)\*mn-1+inp), где B — база (адрес начала блока памяти массива), ikp — значение k-го индекса, приведённое к целому с нулевым начальным смещением. Порядок следования индексов в формуле вычисления адреса может быть различным. Приведённый соответствует реализации в большинстве компиляторов языка Си; в Фортране порядок индексов противоположен[[3]](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)#cite_note-_e9ff6c56b259ec52-3).

Таким образом, адрес элемента с заданным набором индексов вычисляется так, что время доступа ко всем элементам массива одинаково. (Здесь одинаковость времени доступа следует понимать как отсутствие теоретической зависимости времени доступа от положения элемента и размера массива. В действительности особенности конкретной вычислительной платформы могут дать определённый разброс времени доступа. Например, CAS-латентность ОЗУ приводит к увеличению времени доступа к данным, расположенным в другой колонке (странице) ОЗУ, по отношению к предыдущим считанным данным. В практике программирования такими тонкостями, за редчайшими исключениями, пренебрегают.)

Первый элемент массива, в зависимости от языка программирования, может иметь различный индекс. Различают три основных разновидности массивов: с отсчетом от нуля (zero-based), с отсчетом от единицы (one-based) и с отсчетом от специфического значения, заданного программистом (n-based). Отсчет индекса элемента массивов с нуля более характерен для низкоуровневых языков программирования, хотя встречается и в языках высокого уровня, например, в том же Си. В ряде языков (Паскаль, Ада, Модула-2) диапазон индексов может определяться как произвольный диапазон значений любого типа данных, приводимого к целому, то есть целых чисел, символов, перечислений, даже логического типа (в последнем случае массив имеет два элемента, индексируемых значениями «Истина» и «Ложь»).

Обычным способом реализации гетерогенных массивов является отдельное хранение самих значений элементов и размещение в блоке памяти массива (организованного как обычный гомогенный массив, описанный выше) указателей на эти элементы. Поскольку указатели на значения любых типов, как правило, имеют один и тот же размер, удаётся сохранить простоту вычисления адреса, хотя возникают дополнительные накладные расходы на размещение значений элементов и обращение к ним.

Для динамических массивов может использоваться тот же механизм размещения, что и для статических, но с выделением некоторого объёма дополнительной памяти для расширения и добавлении механизмов изменения размера и перемещения содержимого массива в памяти.

Также динамические и гетерогенные массивы могут реализовываться путём использования принципиально иных методов хранения значений в памяти, например, одно- или двухсвязных списков. Такие реализации могут быть более гибкими, но требуют, как правило, дополнительных накладных расходов. Кроме того, в них обычно не удаётся выдержать требование константного времени доступа к элементу.

## **Достоинства**

* лёгкость вычисления адреса элемента по его индексу (поскольку элементы массива располагаются один за другим)
* одинаковое время доступа ко всем элементам
* малый размер элементов: они состоят только из информационного поля.

## **Недостатки**

* для статического массива — отсутствие динамики, невозможность удаления или добавления элемента без сдвига других.
* для динамического и/или гетерогенного массива — более низкое (по сравнению с обычным статическим) быстродействие и дополнительные накладные расходы на поддержку динамических свойств и/или гетерогенности.
* при работе с массивом в стиле C (с указателями) и при отсутствии дополнительных средств контроля — угроза выхода за границы массива и повреждения данных.