数组

在编程语言中，**数组数据结构**（英语：array data structure），简称**数组**（英语：Array），是一种数据结构，是数据元素（elements）的集合。它十分类似数学上的“矩阵”，但是在电脑编程语言上，表示方法和意义上略有不同。
数学上的矩阵看起来像这样：

$$a = \begin{bmatrix} 3 & 6 & 2 \\ 0 & 1 & -4 \\ 2 & -1 & 0 \end{bmatrix}$$

而电脑上的数组看起来像这样，例如C语言中的数组：

```
int a[3][3]={
              { 3,  6,  2 },
              { 0,  1, -4 },
              { 2, -1,  0 }
            };
```

虽然数组在各种电脑编程语言中的表示式略有不同，但是几乎每一种编程语言都有这种结构和观念，它已经不只是一种编程专用的术语，而是电脑运作中非常重要的技术和概念。当前电脑能够显示、打印中文字，都是数组的观念应用；字符串也是基于数组的一种重要数据结构。数据库也是数组概念的一种扩充和延伸。

语义

数组概念有双重含义，一是数据类型，二是实体（entity）。

C/C++标准中的数组

C语言标准中规定，一个数组类型描述了连续分配的非空的具有特定元素对象类型的对象集合。这些元素对象的类型称为元素类型（element type）。数组类型由元素类型与元素的数目确定。
在C语言中，可以显式定义一个数组类型，例如：

```
typedef int myArrayType [101];
```

数组名作为数组实体的标识符，具有特殊性，不同于整型、浮点型、指针型或结构类型等变量标识符。这是因为数组是一组元素的聚集，不能把一个聚集看作一个值直接读出（即右值），也不能把一个聚集看作一个地址直接赋值（即左值）。因此，数组名作为左值、右值，在C语言标准中都有特殊规定：

- 作为sizeof的操作数，数组名代表数组对象本身；

- 作为取地址运算符&的操作数，数组名代表数组对象本身；
- 作为字符串字面量用于初始化一个数组；
- 其他情形，表达式中的数组名从数组类型被转化为指向数组首元素的指针类型表达式（右值）。

例如，

```
char s[]="hello";

int main() {
  char (*p1)[6]=&s; //OK!
  char (*p2)[6]=s; //compile error: cannot convert 'char*' to 'char (*)[6]'
  char *p3=&s;//compile error: cannot convert 'char (*)[6]' to 'char*' in initialization
}
```

根据上述C语言标准中的规定，表达式 &s 的值的类型是char (*)[6]，即指向整个数组的指针；而表达式 s 则被隐式转换为指向数组首元素的指针值，即 char* 类型。同理，表达式 s[4] ，等效于表达式 *(s+4)。

数组下标运算符

C语言标准中定义，数组下标运算（Array subscripting）有两个运算数，一个为到类型type的指针表达式，另一个运算符为整数表达式，结果为类型type。但没有规定两个运算数的先后次序。因此，有以下推论：

- 两个运算数可以交换顺序，即 p[N] 与 N[p] 是等价的，为 *(p+N) ；
- 数组下标运算，既可以适用于数组名（实际上隐式把数组名转换为指向数组首元素的指针表达式），也可以适用于指针表达式；
- 整型表达式可以取负值。

例如：

```
int a[10], *p = a;
p[0] = 10;
( p + 1 )[0] = 20;
0[p + 1] = 10;
```

多维数组

普通数组采用一个整数来作下标。多维数组的概念特别是在数值计算和图形应用方面非常有用。我们在多维数组之中采用一系列有序的整数来标注，如在[3,1,5] 。这种整数列表之中整数的个数始终相同，且被称为数组的“维度”。关于每个数组维度的边界称为“维”。维度为k的数组通常被称为k维。

多维数组的数组名字，在表达式中自动转换为数组首元素地址值，但这个首元素实际上是去除数组下标第一维之后的数组剩余部分。例如：

```
int a[10][15];
int (*p)[15]=a;  // a在表达式中自动转换为指向具有15个int的数组的指针值。
```

不完整的数组类型

不完整的数组类型属于不完整类型（incomplete type），即缺乏信息去确定其尺寸。例如：

```
extern int a[]; //外部变量声明
void foo(int b[]){} //函数形参
void bar(int b[][10]){} //多维数组形参
int a[] = { 10, 20 }; //初始化
```

柔性数组成员

C99规定，struct的最后一个成员可以是不完整的数组类型。例如：

```
struct test
{
    int a;
    double b;
    char c[];
};
```

可变长数组

C99引入了可变长数组（variable length array，简称VLA），只能定义在块作用域或函数原型作用域，必须无链接性。其数组长度在编译期可变，但在运行期该数组对象一旦生成就不可改变数组长度了。例如：

```
void foo(int m，int n)
{
   int v[m][n];
   int *p[n];
}
```

程序设计

数组设计之初是在形式上依赖内存分配而成的，所以必须在使用前预先请求空间。这使得数组有以下特性：

1. 请求空间以后大小固定，不能再改变（数据溢出问题）；
2. 在内存中有空间连续性的表现，中间不会存在其他程序需要调用的数据，为此数组的专用内存空间；

3　在旧式编程语言中（如有**中阶语言**之称的C），程序不会对数组的操作做下界判断，也就有潜在的越界操作的风险（比如会把数据写在运行中程序需要调用的核心部分的内存上）。

因为简单数组强烈倚赖电脑硬件之内存，所以不适用于现代的程序设计。欲使用可变大小、硬件无关性的数据类型，Java等程序设计语言均提供了更高级的数据结构：ArrayList、Vector等动态数组。