Introduction to C Programming

Jichang Zhao

jichang@buaa.edu.cn

Arrays

Objectives

- One-Dimensional Arrays
- Array Initialization
- Arrays as Function Arguments
- Two-Dimensional Arrays
- Search and Sort（8.8 补充材料不作要求，后面会部分涉及）

Introduction

- **Atomic variable**
  - variable whose value cannot be further subdivided into a built-in data type
  - Also called a **scalar variable**
- **Data structure (aggregate data type)**
  - data type with **two** main characteristics

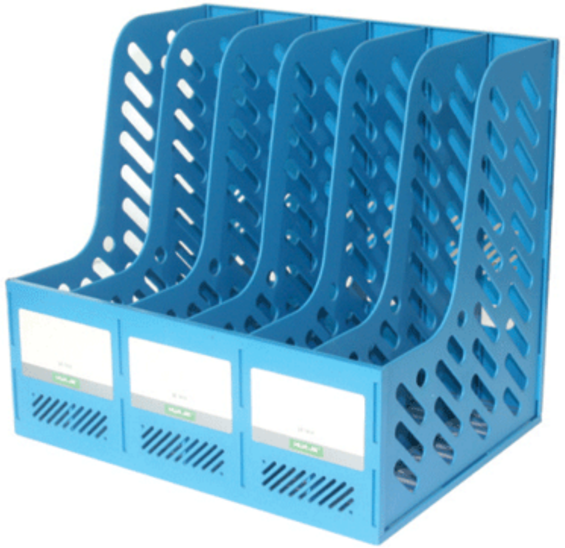
1. Its values can be **decomposed into individual data elements**, each of which is either atomic or another data structure
2. It provides an access scheme **for locating individual data** elements within the data structure

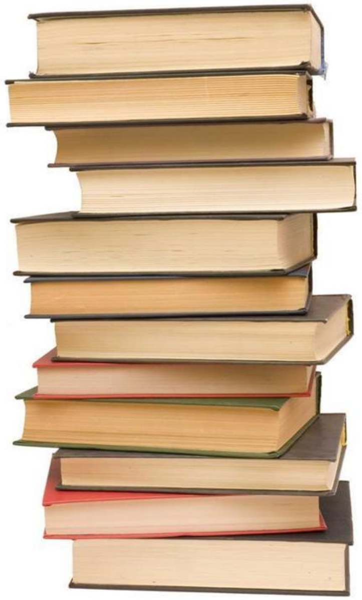
补充：C语言有两种聚合类型，分别是array和structure

Introduction

- One of the simplest data structures, called an **array**
  - is used to **store and process** a set of values
  - all of **the same data type**
  - forms a **logical group**

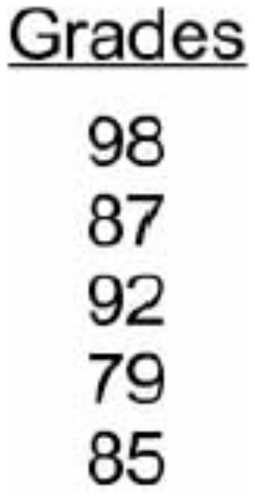
Introduction





One-Dimensional Arrays

- A **one-dimensional array**, also called a **single-dimensional array** and a **single-subscript array**, is a list of values of **the same data type** that is stored using a single group name



Grades
98
87
92
79
85

Figure 8.2
A list of grades

One-Dimensional Arrays

- To create a one-dimensional array:
  - `#define NUMELS 5`
  - **`int`** `grades[NUMELS]; //`“**静态**”
  - 补充：数组的类型是必需，用于计算移动偏移量
  - 补充：数组的长度是“必需”的，用于决定空间分配
  - In C, the starting index value for all arrays is **0**
    - 补充：简化了编译器对偏移量的计算
  - Each item in an array is called **an element or component** of the array
  - Any element can be accessed by giving the name of the array and the element’s position
    - The position is the element’s **index** or **subscript**
    - Each element is called an **indexed variable** or a **subscripted variable**

One-Dimensional Arrays

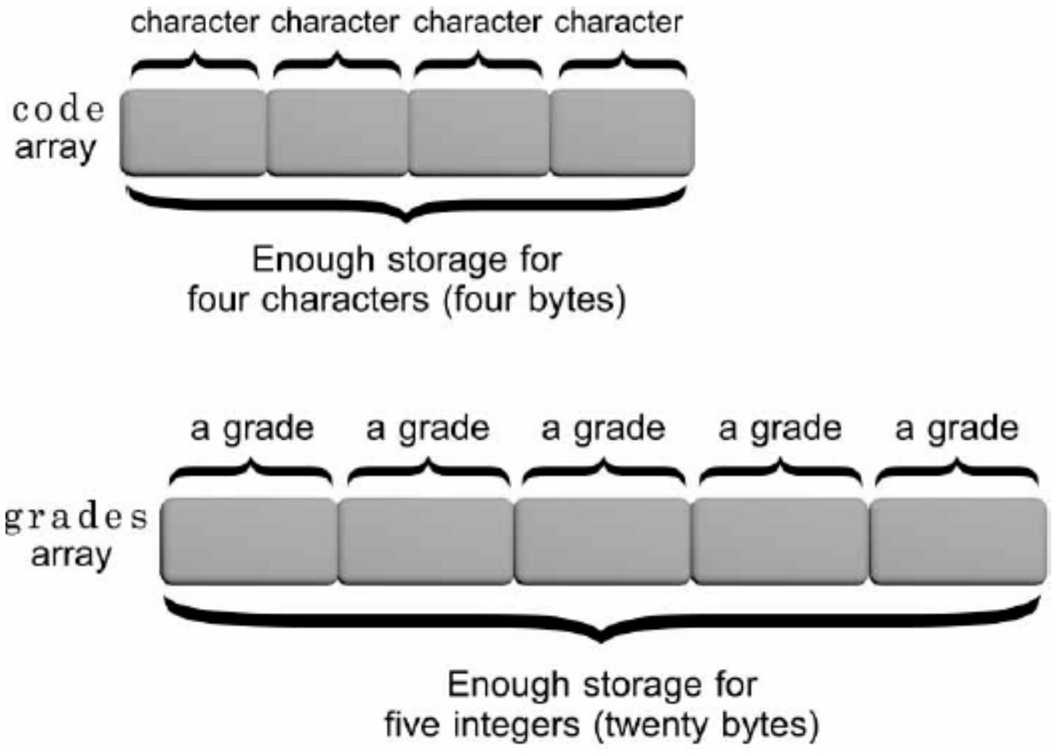

character
character
character
character
code array
Enough storage for four characters (four bytes)
a grade
a grade
a grade
a grade
a grade
grades array
Enough storage for five integers (twenty bytes)

Figure 8.3 The code and grades arrays in memory

One-Dimensional Arrays

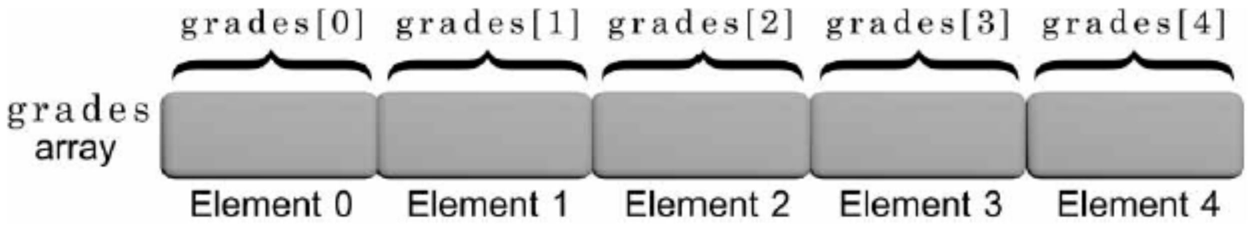


Figure 8.4 Identifying individual array elements

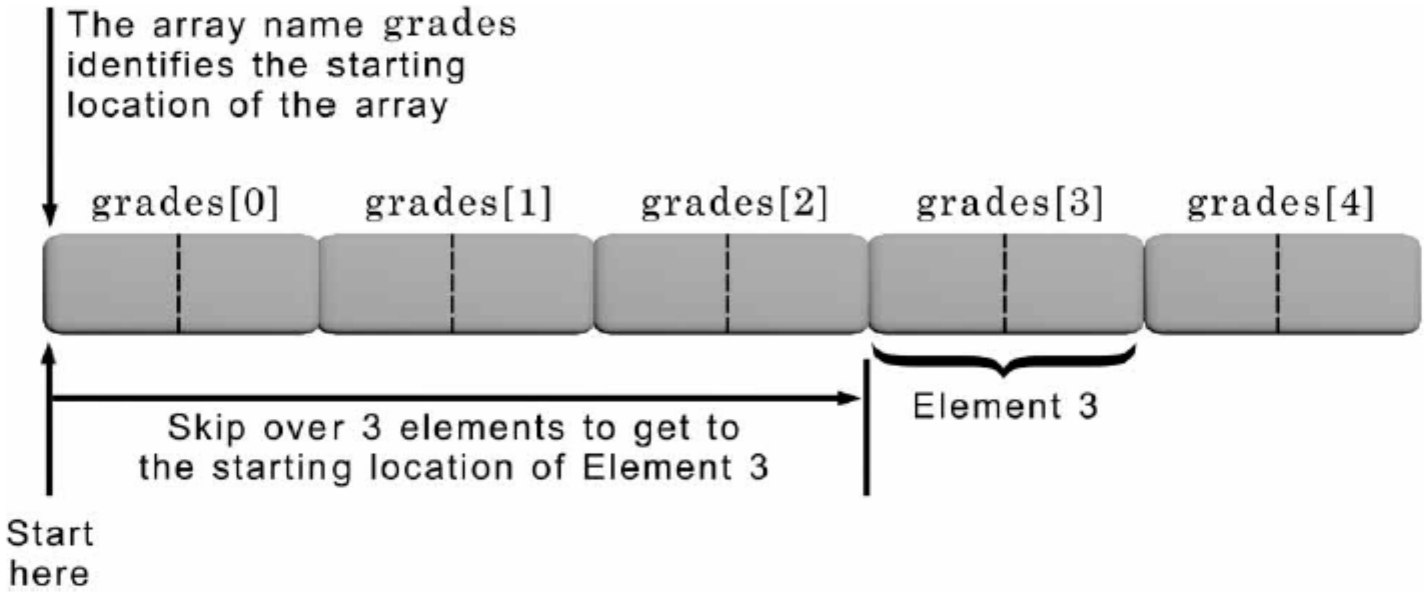

The array name grades identifies the starting location of the array
grades[0]
grades[1]
grades[2]
grades[3]
grades[4]
Element 3
Skip over 3 elements to get to the starting location of Element 3
Start here

Figure 8.5 Accessing element 3

One-Dimensional Arrays

- Subscripted variables can be used anywhere scalar variables are valid
  - `grades[0] = 98;`
  - `grades[1] = grades[0] - 11;`
- **Any expression that evaluates an integer may be used as a subscript**

```
#define NUMELS 5
total = 0; /* initialize total to zero */
for (i = 0; i < NUMELS; i++)
  total = total + grades[i]; /* add a grade */
```

Input and Output of Array Values

- Individual array elements can be assigned values using individual assignment statements or, interactively, using the `scanf()` function

```
#define NUMELS 5
for(i = 0; i < NUMELS; i++)
{
  printf("Enter a grade: ");
  scanf("%d", &grades[i]);
}
```

- Be careful: **C does not check the value of the index being used (called a bounds check)**
- 补充：注意下标不能超出数组的长度减1，否则会出现未定义行为(如ba.c)

Array Initialization

- The individual elements of all global and `static` arrays (local or global) are, by default, **set to 0** at compilation time
- **The values within `auto` local arrays are undefined**
- Examples of initializations:
  - `int grades[5] = {98, 87, 92, 79, 85};`
  - `double length[7] = {8.8, 6.4, 4.9, 11.2};//补0`
  - `int a[200]={0};`
  - `int flags[1000]={[14]=48,[9]=7,[2]=29};//指示符`
  - `char codes[6] = {'s', 'a', 'm', 'p', 'l', e'};`
  - `char codes[] = {'s', 'a', 'm', 'p', 'l', 'e'};`
  - `char codes[] = "sample"; /* size is 7 */`
- **补充：用“”得到的字符串是一种“特殊”的数组，末尾用\0进行标记，因此多占用一个字节。**

Array Initialization

- The **NULL** character, which is the escape sequence \0, is automatically appended to all strings by the C compiler
  - 补充：可以通过sizeof(数组名称)获取数组所占空间的大小
  - 补充：sizeof(数组) / sizeof(数组类型) 得到数组的长度

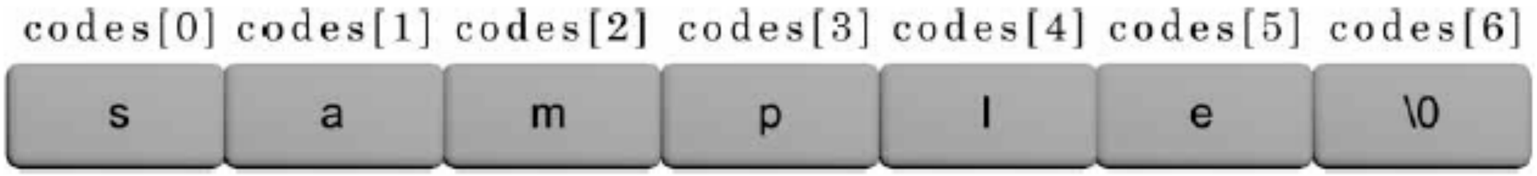


Figure 8.6 A string is terminated with a special sentinel

补充：变长数组

- 变长数组
  - variable-length array, VLA
  - C99标准支持(gcc –std=c99 source.c -o source)
  - 数组的长度可以不预先指定，在运行时再根据相关结果确定
  - `int n;`
  - `scanf(“%d”,&n);`
  - `int a[n];`
  - 注意：`int n ,a[n];`会产生什么错误？
  - DEMO：vla.c
  - 变长数组不能初始化

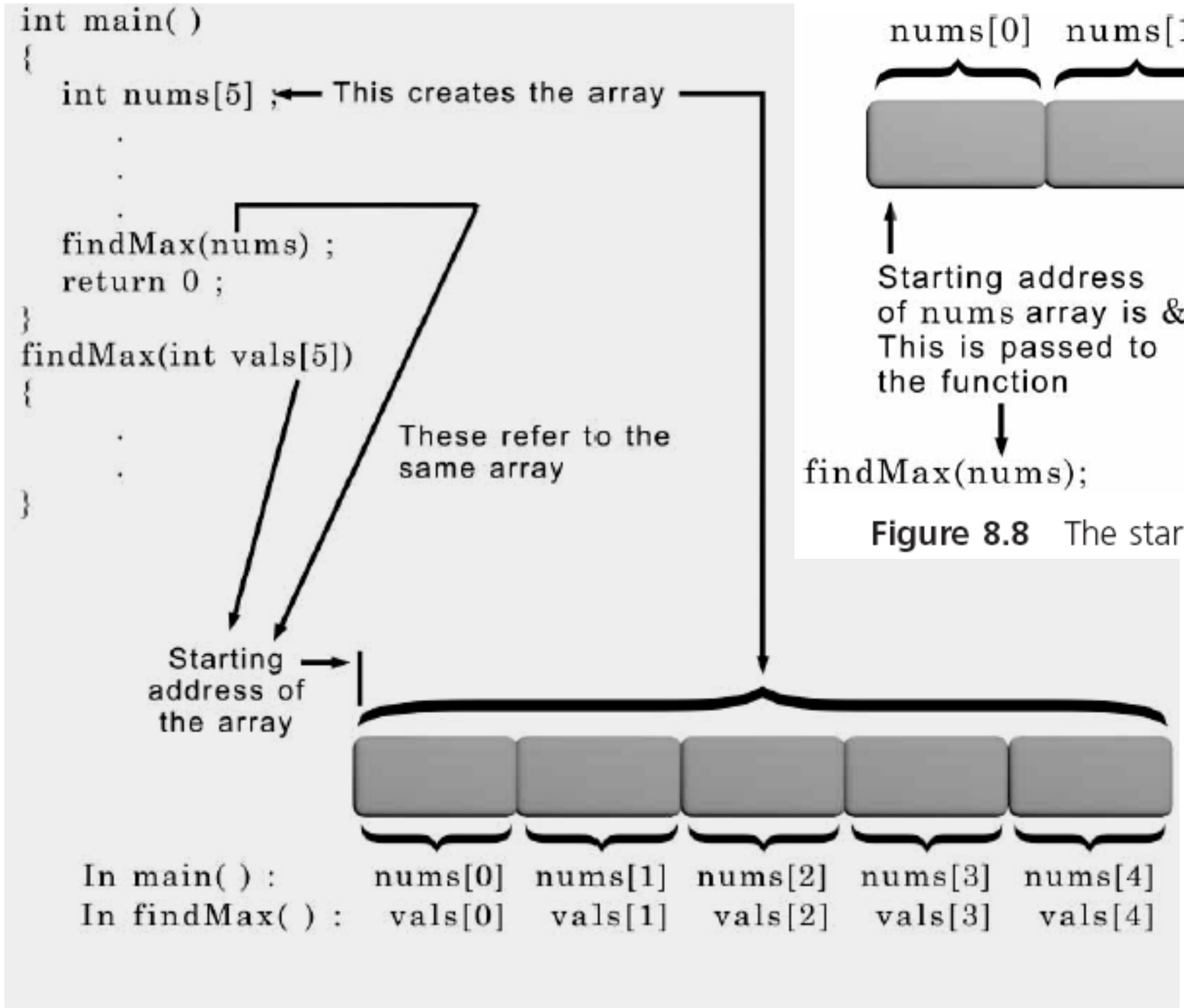
补充：复合字面量

- 复合字面量是指通过指定其包含的元素而创建的没有名字的数组
- C99标准支持
- `(int[]){3,0,3,4,1}`
- `(int[10]){8,10}`
- `(const int []){0,1}`

Arrays as Function Arguments

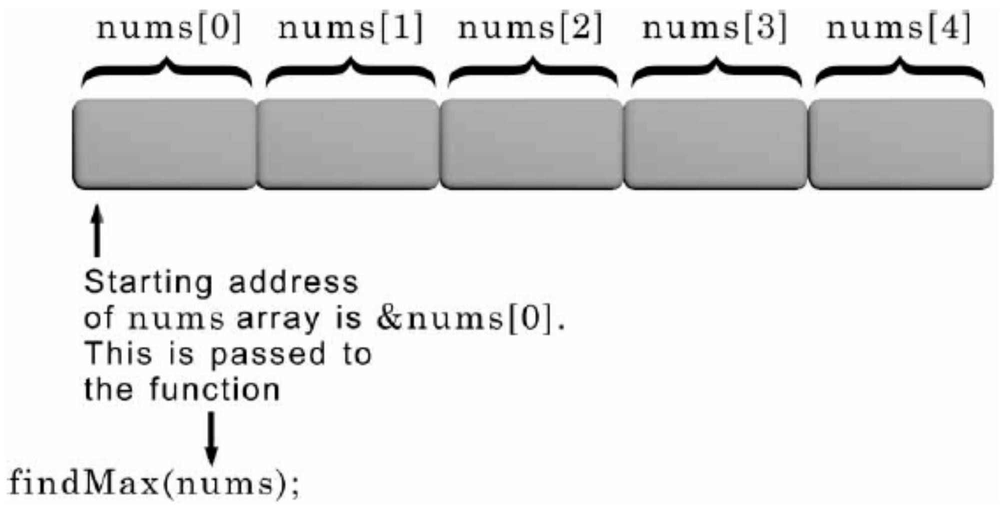
- Individual array elements are passed to a function by including them as subscripted variables in the function call argument list
  - `findMin(grades[2], grades[6]);`
  - **Pass by value**
- When passing a complete array to a function, the called function receives access to the actual array, rather than a copy of the values in the array
  - `findMax(grades);`
  - Pass by address
  - 补充：能够节省空间和时空成本，避免大量数据的复制
  - 补充：同样可以使用const保护数组const char flags[5];

Arrays as Function Arguments



int main( )
{
int nums[5] ; ← This creates the array
.
.
.
findMax(nums) ;
return 0 ;
}
findMax(int vals[5])
{
.
.
}
These refer to the same array
Starting address of the array
In main( ) : nums[0] nums[1] nums[2] nums[3] nums[4]
In findMax( ) : vals[0] vals[1] vals[2] vals[3] vals[4]

Figure 8.7 Only one array is created



nums[0] nums[1] nums[2] nums[3] nums[4]
Starting address of nums array is &nums[0]. This is passed to the function
findMax(nums);

Figure 8.8 The starting address of the array is passed

DEMO: pa.c

Arrays as function Arguments

- 数组型实参
  - 函数如何确定数组的长度？
    - 建议将长度作为另一个参数进行传递
    - `return_type func(array_type array[], int length);`
    - `length`一定要等于或小于实际长度
  - 将数组传递给函数时，不要在数组名后加`[]`，如`func(a[],10);`是错误的，应该是`func(a,10);`
  - 在数组参数声明中使用`static(C99)`
  - `int sum_array(int a[static 3],int n];`
  - 提示编译器该数组至少有3个元素
  - 编译器得以在调用函数时预先从内存中取出这些元素值，而非在函数执行时遇到实际需要这些元素时的语句时才取出
  - 对程序行为无影响，既帮助编译器据此生成更快的指令

Two-Dimensional Arrays

- A two-dimensional array, or table, consists of both rows and columns of elements

```
int val[3][4];
```

补充：注意不要写成`int val[3,4];`这时等同于 `int val[4];`

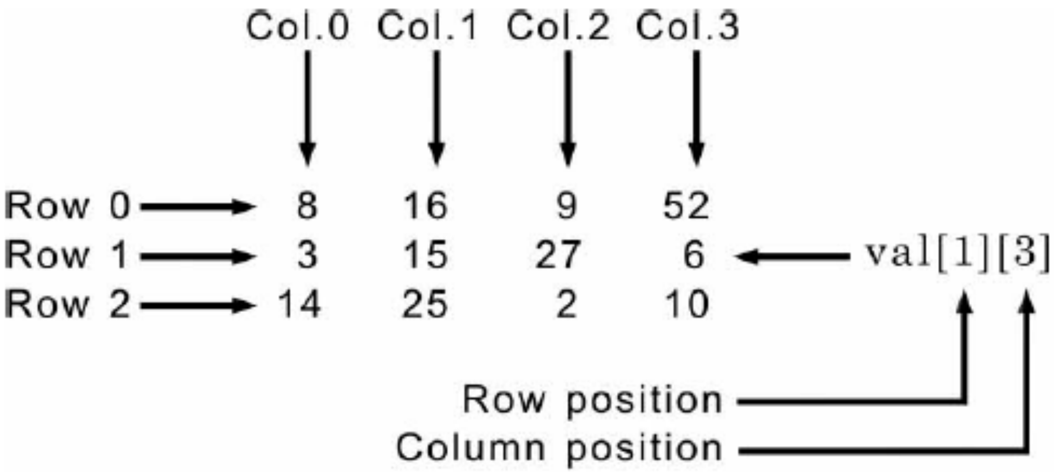


Figure 8.9 Each array element is identified by its row and column

Two-Dimensional Arrays

- Initialization:

```
#define NUMROWS 3
#define NUMCOLS 4
int val[NUMROWS][NUMCOLS] = { {8,16,9,52},
                              {3,15,27,6},
                              {14,25,2,10} };
```

- The inner braces can be omitted:

```
int val[NUMROWS][NUMCOLS] = {8,16,9,52,3,15,27,
  6,14,25,2,10};
```

- Initialization is done in row order

Two-Dimensional Arrays

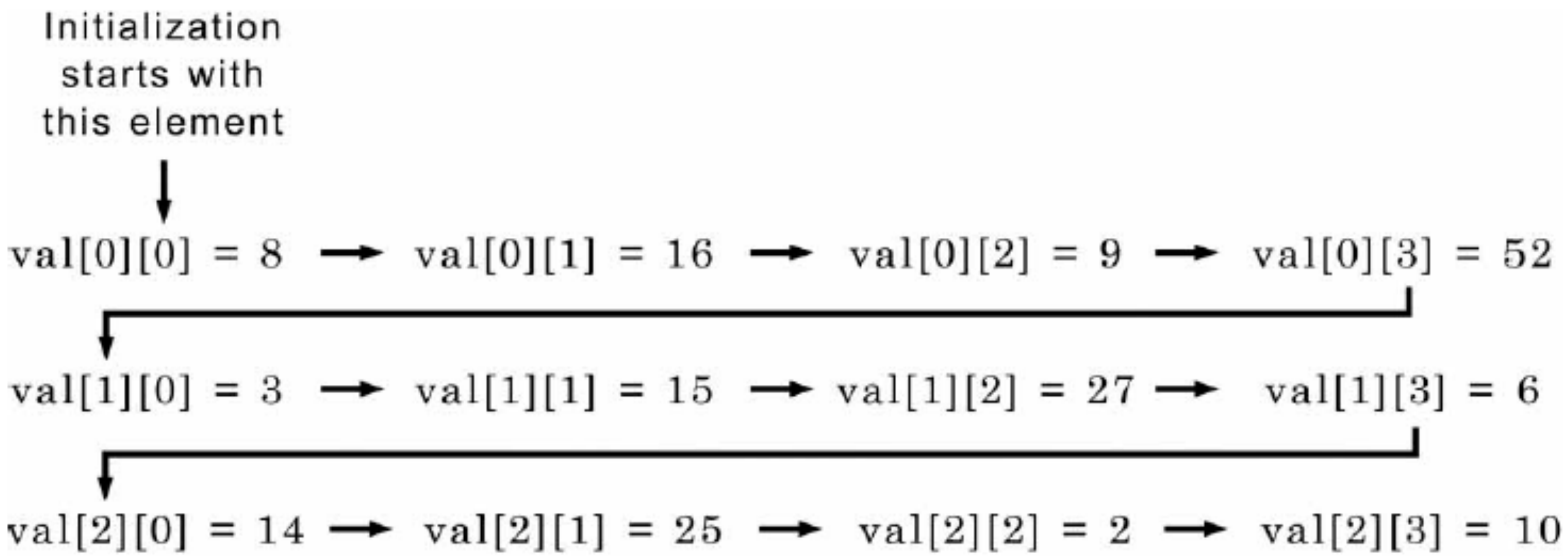


Figure 8.10 Storage and initialization of the `val[]` array

补充：数组元素的定位算法

- 参考教材：P317
  - 数据元素i的地址=数据开始地址+偏移量
- 一维数组
  - 偏移量=i * 单个元素的长度（字节数）
- 二维数组
  - 完整一行的长度=指定的最大列数*单个元素的长度
  - 偏移量=行索引值*完整一行的长度+列索引值*单个元素的长度

Larger Dimensional Arrays

- A three-dimensional array can be viewed as a book of data tables (the third subscript is called the **rank**)
  - `int response[4][10][6];`
  - 补充：如何表示一张图片？如何表示一个多特征的实体？
- A four-dimensional array can be represented as a shelf of books where the fourth dimension is used to declare a desired book on the shelf
- A five-dimensional array can be viewed as a bookcase filled with books where the fifth dimension refers to a selected shelf in the bookcase
- Arrays of three, four, five, six, or more dimensions can be viewed as mathematical *n*-tuples

补充：函数形参为多维数组

- 只能省略第一维的长度
  - int sum_two_dimentional_array(int a[][LEN], int n);
  - LEN：列长
  - n：行长
- static只能用于第一维
  - 仅能用于指定行数
  - int a[static 3][LEN]

Case Study

- Countdown Display
  - 《现代方法第2版》p125, 6; p171, 7

Homework

- 1, P298第4题
- 2, P302第2题
- 3, P306第5题 (将数组作为参数)
- 4, P311第10题
- 5, 《现代方法第2版》p127, 15(加密和解密均要实现)
- 6, 《现代方法第2版》p127, 16
- 7. 《现代方法第2版》p127, 14(终止符号也要读入)
- 8. 《现代方法第2版》p126, 9 (注意死循环)