数组

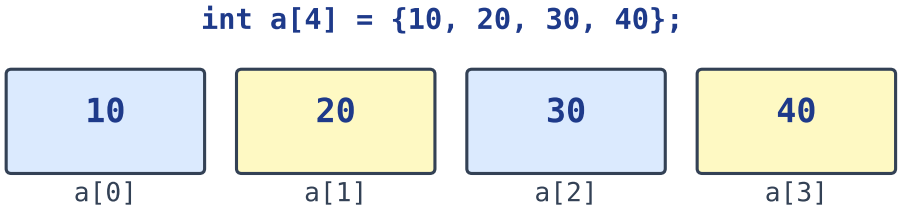
C Reference

1-D arrays

数组 (array) 保存同一类型的多个值。索引 (index) 从 0 开始。C **不会**存储数组的长度, 所以有一个常用技巧来计算元素 (element) 个数:`sizeof(a) / sizeof(a[0])`。

```c
#include <stdio.h>

int main(void) {
    int scores[3] = {88, 71, 95};
    printf("%d\n", scores[0]);     // 88
    scores[1] = 100;
    printf("%d\n", scores[1]);     // 100
    int n = sizeof(scores) / sizeof(scores[0]);
    printf("%d\n", n);             // 3
    return 0;
}
```



Exam tip: indices start at 0; a[i] is one element; length is fixed at creation

Array slots are indexed from 0

Array algorithms

用 `for` 循环遍历 (traverse) 数组，求最大值、总和或计数。始终从 `0` 循环到 `n - 1`。

```c
#include <stdio.h>

int main(void) {
    int a[] = {3, 9, 2, 7};
    int n = sizeof(a) / sizeof(a[0]);
    int max = a[0], total = 0;
    for (int i = 0; i < n; i++) {
        if (a[i] > max) max = a[i];
        total += a[i];
    }
    printf("%d %d\n", max, total);   // 9 21
    return 0;
}
```

2-D arrays

一个 2-D 数组是一个由行和列组成的网格：`grid[row][col]`。使用两个嵌套循环遍历每个单元格。

```c
#include <stdio.h>

int main(void) {
    int grid[2][3] = {{1, 2, 3}, {4, 5, 6}};
    printf("%d\n", grid[1][2]);    // 6
    for (int r = 0; r < 2; r++) {
        for (int c = 0; c < 3; c++) {
            printf("%d ", grid[r][c]);
        }
    }
    printf("\n");                  // 1 2 3 4 5 6
    return 0;
}
```

常见错误

- C 不检查数组边界：在大小为 `n` 的数组中写入 `a[n]` 属于未定义行为。
- 合法下标是 `0` 到 `n - 1`。
- 在大多数表达式中，数组会自动转换为指向首元素的指针，但数组本身并非指针。`sizeof(a)` 给出实际数组的完整字节大小；作为参数的数组会被调整为指针，因此函数需要单独传入长度。