

# 静态链表与动态链表

在数据结构中，**静态链表**和**动态链表**是两种常见的链表实现方式，它们在内存分配、结构设计和应用场景上有明显区别。下面我来详细对比它们的异同：

## 概括：

**静态链表**：用数组模拟链表，内存一次性分配，**不支持动态扩展**。

**动态链表**：用指针连接节点，内存按需分配，**支持动态扩展**。

## 详细对比表：

特性	静态链表	动态链表
内存分配方式	静态数组，一次性分配	动态申请（如 <code>malloc / new</code> ）
是否可扩展	不可扩展，大小固定	可动态扩展
节点结构	数组元素 + 游标（cursor）	数据 + 指针（next）
插入/删除	修改游标，效率高	修改指针，效率高
内存碎片	无碎片（连续空间）	可能有碎片（非连续）
实现复杂度	稍高（需管理空闲链表）	相对简单
使用场景	嵌入式、无指针环境、固定数据量场景	通用场景，数据量不确定

## 静态链表（用数组模拟）：

```
#define MAX_SIZE 100
typedef struct {
    int data;
    int next; // 游标，指向下一个元素的数组下标
} Node;

Node staticList[MAX_SIZE];
```

## 动态链表（用指针）：

```
typedef struct Node {  
    int data;  
    struct Node* next;  
} Node;  
  
Node* head = (Node*)malloc(sizeof(Node));
```

总结建议：

场景	适用
数据量固定、无动态内存分配支持（如某些嵌入式系统）	静态链表
数据量不确定、需要频繁插入删除	动态链表