# 单链表抽象数据类型（LinkedList ADT）

## 定义

单链表（Singly Linked List）是一种线性数据结构，由一组节点组成，每个节点包含数据域和指向下一个节点的指针。
在 C++ 中，单链表通常采用类（class）实现，封装数据和操作，体现面向对象思想。

## 基本操作

- **初始化**
  - `LinkedList()`：构造函数，初始化空链表
  - 时间复杂度：O(1)
- **析构**
  - `~LinkedList()`：析构函数，释放所有节点
  - 时间复杂度：O(n)
- **清空链表**
  - `clear()`：清空链表内容
  - 时间复杂度：O(n)
- **判空**
  - `empty() const`：判断链表是否为空
  - 时间复杂度：O(1)
- **获取长度**
  - `size() const`：返回链表长度
  - 时间复杂度：O(1)
- **获取元素**
  - `get(index) const`：获取第 index 个元素的值，越界抛出 `std::out_of_range`
  - 时间复杂度：O(n)
- **查找元素**
  - `find(value) const`：查找值为 value 的元素位置，未找到返回 -1
  - 时间复杂度：O(n)
- **插入元素**
  - `insert(index, value)`：在 index 位置插入元素，越界抛出 `std::out_of_range`
  - 时间复杂度：O(n)
- **删除元素**
  - `remove(index)`：删除 index 位置的元素，越界抛出 `std::out_of_range`
  - 时间复杂度：O(n)
- **遍历链表**
  - `traverse(void (*visit)(const T&)) const`：遍历链表，对每个元素调用 visit 函数
  - 时间复杂度：O(n)

## 异常与边界

- 所有越界操作均抛出 `std::out_of_range` 异常

## 接口定义（伪代码）

```
interface LinkedListADT<T> {
    constructor();
    destructor();
    clear(): void;
    empty(): boolean;
    size(): number;
    get(index: number): T;                  // O(n), 越界抛异常
    find(value: T): number;                  // O(n), 未找到返回-1
    insert(index: number, value: T): void; // O(n), 越界抛异常
    remove(index: number): void;             // O(n), 越界抛异常
    traverse(visit: (value: T) => void): void; // O(n)
}
```

# 空间复杂度

- O(n)，n为链表元素个数

# 优点

- 插入、删除操作无需移动大量元素，适合频繁变动场景
- 动态分配内存，无需预设容量
- 支持任意类型元素（模板实现）

# 局限性

- 随机访问效率低（O(n)）
- 需要额外指针存储，空间利用率略低于数组

# 适用场景

- 需要频繁插入、删除操作的线性表
- 元素数量变化频繁且不可预知的场景

# 交互式测试（中文版）

本模块附带交互式测试程序，所有命令行交互均为中文，便于中文用户体验和学习。详见 ../test/test_linkedList.cpp。

示例命令：

- `insert 0 10` 在下标0插入10
- `remove 1` 删除下标1的元素
- `get 2` 获取下标2的值
- `find 99` 查找99首次出现的下标
- `size` 当前元素个数
- `empty` 是否为空
- `print` 打印链表内容
- `help` 显示菜单
- `exit` 或 `0` 退出程序

所有异常和错误提示也为中文，便于理解和调试。