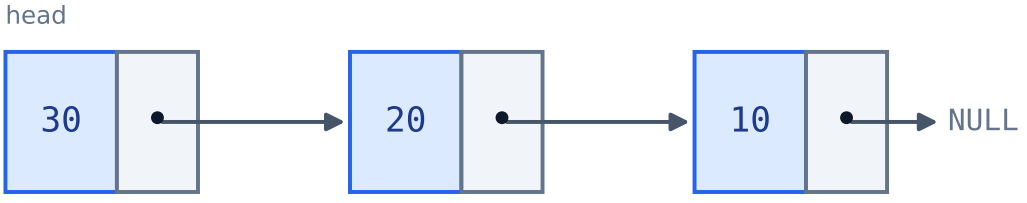
数据结构

C Reference

链表

链表 (linked list) 是一串节点 (node)。每个节点保存一个值和一个指向 `next`(下一个) 节点的指针; 最后一个指向 `NULL`。与数组不同, 它用 `malloc` 一次增加一个节点。用完后一定要 `free` 每个节点。



10-linked-list

链表: 每个节点保存一个值和指向下一个节点的指针, 最后指向 *NULL*

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
    int value;
    struct Node *next;
} Node;

int main(void) {
    Node *head = NULL;
    for (int v = 10; v <= 30; v += 10) {   // 依次头插 10、20、30
        Node *n = malloc(sizeof(Node));
        n->value = v;
        n->next = head;
        head = n;
    }
    for (Node *p = head; p != NULL; p = p->next) {
        printf("%d ", p->value);           // 30 20 10
    }
    printf("\n");

    while (head != NULL) {                 // 释放整个链表
        Node *t = head;
        head = head->next;
        free(t);
    }
    return 0;
}
```

栈与队列

栈 (stack) 是 LIFO 后进先出 (last in, first out): 在同一端入栈和出栈。队列 (queue) 是 FIFO 先进先出 (first in, first out): 在尾部加入, 从头部移除。两者都可以用数组加索引变量轻松实现。

```c
#include <stdio.h>

int main(void) {
    int stack[10];
    int top = 0;                        // 下一个空位
    stack[top++] = 1;                   // 入栈
    stack[top++] = 2;
    stack[top++] = 3;
    while (top > 0) {
        printf("%d ", stack[--top]);    // 出栈:3 2 1
    }
    printf("\n");
    return 0;
}
```

```c
#include <stdio.h>

int main(void) {
    int queue[10];
    int front = 0, back = 0;
    queue[back++] = 1;                          // 入队
    queue[back++] = 2;
    queue[back++] = 3;
    while (front < back) {
        printf("%d ", queue[front++]);          // 出队:1 2 3
    }
    printf("\n");
    return 0;
}
```

常见错误

- 在链表中绝不能弄丢 `head` 指针, 否则整条链表都找不回来。
- 用完后释放每个节点, 解开链接前先遍历。
- pop 之前先检查空链表 (`head` 为 `NULL`)。