Data structures

C Reference

Linked lists

A linked list 链表 is a chain of nodes 节点. Each node holds a value and a pointer to the `next` node; the last one points to `NULL`. Unlike an array it grows one node at a time with `malloc`. Always `free` every node when done.

```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) {   // prepend 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) {                 // free the whole list
        Node *t = head;
        head = head->next;
        free(t);
    }
    return 0;
}
```

Stacks & queues

A stack 栈 is LIFO 后进先出 (last in, first out): push and pop at the same end. A queue 队列 is FIFO 先进先出 (first in, first out): add at the back, remove from the front. Both are easy to build on an array with index variables.

```c
#include <stdio.h>

int main(void) {
```

```
    int stack[10];
    int top = 0;                        // next free slot
    stack[top++] = 1;                   // push
    stack[top++] = 2;
    stack[top++] = 3;
    while (top > 0) {
        printf("%d ", stack[--top]);    // pop: 3 2 1
    }
    printf("\n");
    return 0;
}
```

```
#include <stdio.h>

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