线性表

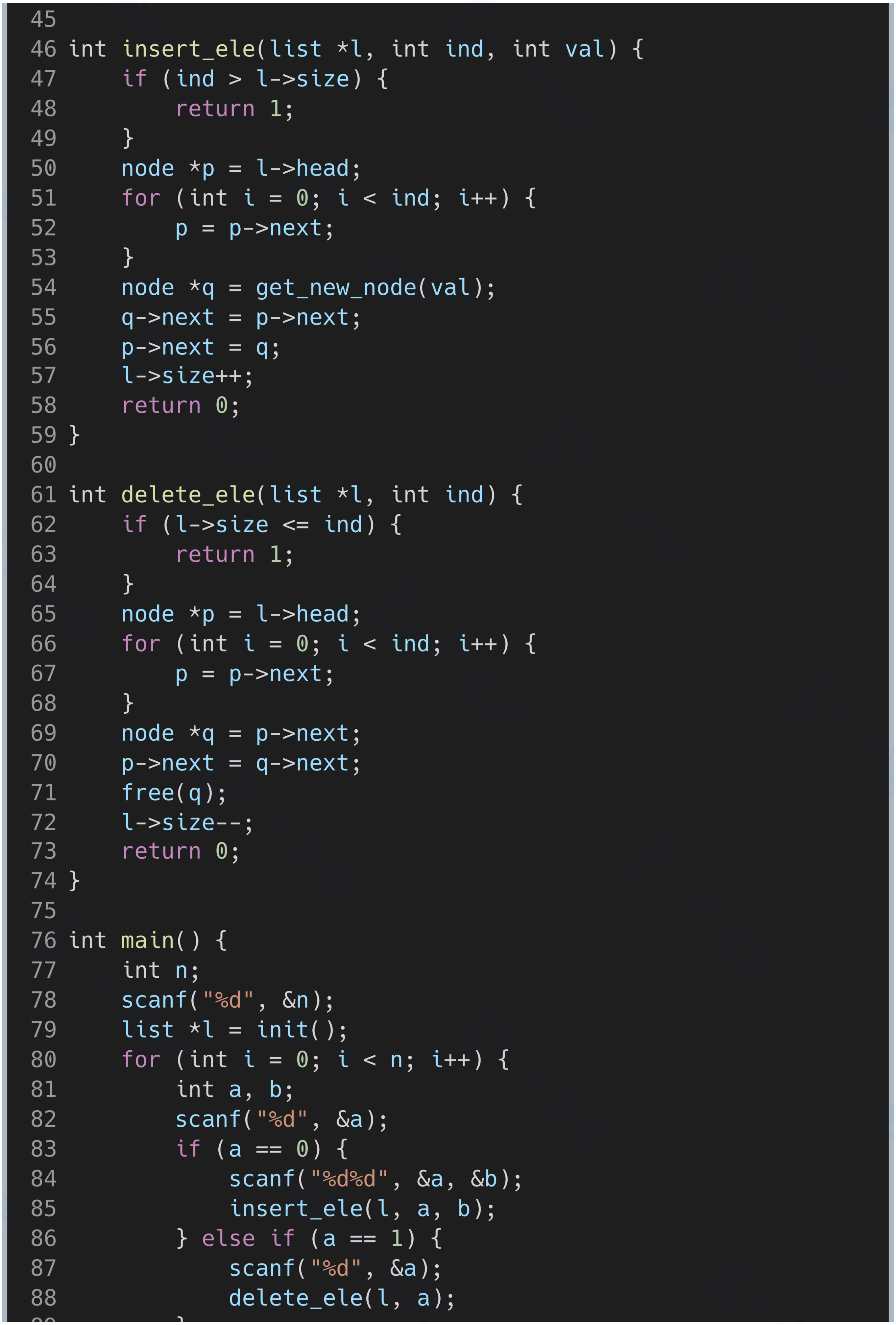
同学们要自己动手，锻炼动手能力！

顺序表

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

typedef struct vector {
    int *data;
    int size, cap;
} vector;

vector *init(int cnt) {
    vector *p = (vector *)malloc(sizeof(vector));
    p->data = (int *)malloc(sizeof(int) * cnt);
    p->size = 0;
    p->cap = cnt;
    return p;
}

void delete_vector(vector *p) {
    free(p->data);
    free(p);
}

void show_vector(vector *v) {
    printf("----- size = %d, cap = %d------\n", v->size, v->cap);
    for (int i = 0; i < v->size; i++) {
        printf("%d ", v->data[i]);
    }
    printf("\n-----------------------\n");
}

int insert_ele(vector *v, int ind, int val) {
    if (ind > v->size) {
        return 1;
    }
    if (v->size == v->cap) {
        v->cap *= 2;
        v->data = (int *)realloc(v->data, sizeof(int) * v->cap);
    }
    for (int i = v->size; i > ind; i--) {
        v->data[i] = v->data[i - 1];
    }
    v->data[ind] = val;
    v->size++;
    return 0;
}

int delete_ele(vector *v, int ind) {
    if (v->size <= ind) {
        return 1;
    }
    for (int i = ind; i < v->size - 1; i++) {
        v->data[i] = v->data[i + 1];
    }
    v->size--;
    return 0;
}

int main() {
    int n, cnt;
    scanf("%d%d", &n, &cnt);
    vector *v = init(cnt);
    for (int i = 0; i < n; i++) {
        int a, b;
        scanf("%d", &a);
        if (a == 0) {
            scanf("%d%d", &a, &b);
            insert_ele(v, a, b);
        } else if (a == 1) {
            scanf("%d", &a);
            delete_ele(v, a);
        }
        show_vector(v);
    }
    delete_vector(v);
    v = NULL;
    return 0;
}
```

单链表

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

typedef struct node {
    int data;
    struct node *next;
} node;

typedef struct list {
    int size;
    struct node *head;
} list;

node *get_new_node(int val) {
    node *p = (node *)malloc(sizeof(node));
    p->data = val;
    p->next = NULL;
    return p;
}

void delete_list(list *p) {
    node *q = p->head;
    for (int i = 0; i <= p->size; i++) {
        node *t = q->next;
        free(q);
        q = t;
    }
    free(p);
}

list *init() {
    list *p = (list *)malloc(sizeof(list));
    p->head = get_new_node(0);
    p->size = 0;
    return p;
}

void show_list(list *l) {
    printf("----- size = %d --------\n", l->size);
    for (node *p = l->head->next; p != NULL; p = p->next) {
        printf("%d->", p->data);
    }
    printf("NULL\n----------------------\n");
}
```



```c

int insert_ele(list *l, int ind, int val) {
    if (ind > l->size) {
        return 1;
    }
    node *p = l->head;
    for (int i = 0; i < ind; i++) {
        p = p->next;
    }
    node *q = get_new_node(val);
    q->next = p->next;
    p->next = q;
    l->size++;
    return 0;
}

int delete_ele(list *l, int ind) {
    if (l->size <= ind) {
        return 1;
    }
    node *p = l->head;
    for (int i = 0; i < ind; i++) {
        p = p->next;
    }
    node *q = p->next;
    p->next = q->next;
    free(q);
    l->size--;
    return 0;
}

int main() {
    int n;
    scanf("%d", &n);
    list *l = init();
    for (int i = 0; i < n; i++) {
        int a, b;
        scanf("%d", &a);
        if (a == 0) {
            scanf("%d%d", &a, &b);
            insert_ele(l, a, b);
        } else if (a == 1) {
            scanf("%d", &a);
            delete_ele(l, a);
        }
        }
        show_list(l);
    }
    delete_list(l);
    l = NULL;
    return 0;
}
```