C类补充

堆

```c
#include <stdio.h>

int heap_size, n;
int heap[2000001];
int buf[2000001];
void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}
void push(int d)
{
    int now, next;
    heap[++heap_size] = d;
    now = heap_size;
    while (now > 1)
    {
        next = now >> 1;
        if (heap[now] >= heap[next])
            return;
        swap(&heap[now], &heap[next]);
        now = next;
    }
}
int pop()
{
    int now, next, res;
    res = heap[1];
    heap[1] = heap[heap_size--];
    now = 1;
    while (now * 2 <= heap_size)
    {
        next = now * 2;
        if (next < heap_size && heap[next + 1] < heap[next])
            next++;
        if (heap[now] <= heap[next])
            return res;
        swap(&heap[now], &heap[next]);
        now = next;
    }
}
int top()
{
    return heap[1];
}
int main()
{
    scanf("%d", &n);
    for (int i = 1; i <= n; i++)
    {
        int x;
        scanf("%d", &x);
        if (x == 1)
        {
            scanf("%d", &x);
            push(x);
        }
        else if (x == 3)
        {
            printf("%d", top());
            putchar('\n');
        }
        else
            pop();
    }
    while(heap_size!=0)
    {
        printf("%d ",top());
        pop();
    }
}
```

查找

二分查找

```
int binsearch(int *key, int n, int k)
{
    int low = 0, high = n - 1, mid;
    while (low <= high)
    { // 查找结束的条件
        mid = (low + high) / 2;
        if (k == key[mid])
            return mid; /*  查找成功  */
        if (k > key[mid])
            low = mid + 1; /*  准备查找后半部分 */
        else
            high = mid - 1; /* 准备查找前半部分 */
    }
    return -1; /*   查找失败  */
}
```

lower_bound

```
int lower_bound(LL *a,LL size, LL val)
{
    int first = 0, last = size -1, mid;
    while (first <= last)
    {
        mid = last - (last - first) / 2;
        if (a[mid] >= val)
            last = mid - 1;
        else
            first = mid + 1;
    }
    return first;
}
```

upper_bound

```
int upper_bound(LL *a,LL size, LL val)
{
    int first = 0, last = size - 1, mid;
    while (first <= last)
    {
        mid = last - (last - first) / 2;
        if (a[mid] <= val)
            first = mid + 1;
        else
            last = mid - 1;
    }
    return first;
}
```