```
/*
最大堆的建立
    将已经存在的 N 的元素按最大堆的要求存放在一个一维数组中
    方法1：
        通过插入操作，将 N 个元素一个个相继插入到一个初始为空的堆中去
        时间代价最大为O(NlogN)
    方法二：
        1、将 N 个元素按输入顺序存入，先满足完全二叉树的结构特性
        2、调整结点位置，以满足最大堆的有序特性
*/

/*

方法一

*/

#include<iostream>
#include<malloc.h>
const int MinData = -100000; // 哨兵值
const int MaxSize = 1005;    // 最大个数
using namespace std;
typedef struct HeapStruct *Heap;
struct HeapStruct{
    int *data; // 存值的数组
    int size;  // 当前元素个数
    int capacity; // 最大容量
};

// 初始化堆
Heap Create()
{
    Heap H;
    H = (Heap)malloc(sizeof(struct HeapStruct));
    H->data = (int *)malloc((MaxSize+1)*sizeof(int));
    H->data[0] = MinData;
    H->size = 0;
    H->capacity = MaxSize;
    return H;
}

// 排序，类似堆的删除操作
void sort(Heap H, int i)
{
    int child, parent;
    int tmp = H->data[i]; // 拿到当前根结点的值
    for(parent=i; parent*2<=H->size; parent=child)
    {
        child = 2*parent;
        if((child!=H->size) && (H->data[child+1]<H->data[child]))
            child++;
        if(tmp <= H->data[child])
            break;
        else
            H->data[parent] = H->data[child];
    }
    H->data[parent] = tmp;
}

// 调整
void adjust(Heap H)
{
    int i = H->size/2;
    for(; i>0; i--)
    {
        // 从倒数第一个有孩子的结点开始
        // 以每个有孩子结点的结点作为根结点，对其子树进行堆排序
        sort(H, i);
    }
}

// 遍历
void bl(Heap H)
```

```
{
    for(int i=1; i<=H->size; i++)
    {
        cout<<H->data[i]<<" ";
    }
    cout<<endl;
}


int main()
{
    Heap H;
    H = Create();
    // 8个数字：5 6 4 3 1 2 8 9
    int n;
    cin>>n;
    for(int i=0; i<n; i++)
        cin>>H->data[++H->size];
    adjust(H);
    bl(H);
    return 0;
}


/*

方法二

*/

#include<iostream>
#include<malloc.h>
const int MinData = -100000; // 哨兵值
const int MaxSize = 1005;    // 最大个数
using namespace std;
typedef struct HeapStruct *Heap;
struct HeapStruct{
    int *data; // 存值的数组
    int size;  // 当前元素个数
    int capacity; // 最大容量
};

// 初始化堆
Heap Create()
{
    Heap H;
    H = (Heap)malloc(sizeof(struct HeapStruct));
    H->data = (int *)malloc((MaxSize+1)*sizeof(int));
    H->data[0] = MinData;
    H->size = 0;
    H->capacity = MaxSize;
    return H;
}

// 插入
void Insert(Heap H, int x)
{
    int i = ++H->size; // 指向数组最后一个
    for(; x<H->data[i/2]; i/=2)
        H->data[i] = H->data[i/2];
    H->data[i] = x;
}

// 遍历
void bl(Heap H)
{
    for(int i=1; i<=H->size; i++)
        cout<<H->data[i];
}

int main()
{
    Heap H;
```

```
147         H = Create();
148         int n;
149         cin>>n;
150         for(int i=0; i<n; i++)
151         {
152             int t;
153             cin>>t;
154             Insert(H, t);
155         }
156         bl(H);
157         return 0;
158     }
159
```