

```

1  /*
2  最大堆的建立
3      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
4      方法1:
5          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
6          时间代价最大为O(NlogN)
7      方法二:
8          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
9          2、调整结点位置, 以满足最大堆的有序特性
10 */
11
12 /*
13
14 方法一
15
16 */
17
18 #include<iostream>
19 #include<malloc.h>
20 const int MinData = -100000; // 哨兵值
21 const int MaxSize = 1005;    // 最大个数
22 using namespace std;
23 typedef struct HeapStruct *Heap;
24 struct HeapStruct{
25     int *data; // 存值的数组
26     int size;  // 当前元素个数
27     int capacity; // 最大容量
28 };
29
30 // 初始化堆
31 Heap Create()
32 {
33     Heap H;
34     H = (Heap)malloc(sizeof(struct HeapStruct));
35     H->data = (int *)malloc((MaxSize+1)*sizeof(int));
36     H->data[0] = MinData;
37     H->size = 0;
38     H->capacity = MaxSize;
39     return H;
40 }
41
42 // 排序, 类似堆的删除操作
43 void sort(Heap H, int i)
44 {
45     int child, parent;
46     int tmp = H->data[i]; // 拿到当前根结点的值
47     for(parent=i; parent*2<=H->size; parent=child)
48     {
49         child = 2*parent;
50         if((child!=H->size) && (H->data[child+1]<H->data[child]))
51             child++;
52         if(tmp <= H->data[child])
53             break;
54         else
55             H->data[parent] = H->data[child];
56     }
57     H->data[parent] = tmp;
58 }
59
60 // 调整
61 void adjust(Heap H)
62 {
63     int i = H->size/2;
64     for(; i>0; i--)
65     {
66         // 从倒数第一个有孩子的结点开始
67         // 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
68         sort(H, i);
69     }
70 }
71
72 // 遍历
73 void bl(Heap H)

```

```

74     {
75         for(int i=1; i<=H->size; i++)
76         {
77             cout<<H->data[i]<<" ";
78         }
79         cout<<endl;
80     }
81
82
83 int main()
84 {
85     Heap H;
86     H = Create();
87     // 8个数字: 5 6 4 3 1 2 8 9
88     int n;
89     cin>>n;
90     for(int i=0; i<n; i++)
91         cin>>H->data[++H->size];
92     adjust(H);
93     bl(H);
94     return 0;
95 }
96
97
98 /*
99
100 方法二
101
102 */
103
104 #include<iostream>
105 #include<malloc.h>
106 const int MinData = -100000; // 哨兵值
107 const int MaxSize = 1005;    // 最大个数
108 using namespace std;
109 typedef struct HeapStruct *Heap;
110 struct HeapStruct{
111     int *data; // 存值的数组
112     int size;  // 当前元素个数
113     int capacity; // 最大容量
114 };
115
116 // 初始化堆
117 Heap Create()
118 {
119     Heap H;
120     H = (Heap)malloc(sizeof(struct HeapStruct));
121     H->data = (int *)malloc((MaxSize+1)*sizeof(int));
122     H->data[0] = MinData;
123     H->size = 0;
124     H->capacity = MaxSize;
125     return H;
126 }
127
128 // 插入
129 void Insert(Heap H, int x)
130 {
131     int i = ++H->size; // 指向数组最后一个
132     for(; x<H->data[i/2]; i/=2)
133         H->data[i] = H->data[i/2];
134     H->data[i] = x;
135 }
136
137 // 遍历
138 void bl(Heap H)
139 {
140     for(int i=1; i<=H->size; i++)
141         cout<<H->data[i];
142 }
143
144 int main()
145 {
146     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
```