#include <iostream>

8

/ \

7 6

/\ /\

4 3 2 5

/\

1

插入9↓，弹出9↑

9

/ \

8 6

/\ /\

7 3 2 5

/\

1 4

using namespace std;

const int MAX = 22;

typedef struct Heap

{

int sizeHeap;

int\* heapData;

}HEAP,\*LPHEAP;

LPHEAP createHeap()

{

LPHEAP heap=(LPHEAP)malloc(sizeof(HEAP));

heap->sizeHeap=0;

heap->heapData=(int\*)malloc(sizeof(int)\*MAX);

return heap;

}

int size(LPHEAP heap)

{

return heap->sizeHeap;

}

int empty(LPHEAP heap)

{

return heap->sizeHeap==0;

}

void moveToCorrectPos(LPHEAP heap, int curPos)//向上渗透，curPos一般取最后一个元素的下标

{

while(curPos>1)

{

int Max=heap->heapData[curPos];

int parentIndex=curPos/2;

if(Max>heap->heapData[parentIndex])

{

heap->heapData[curPos]=heap->heapData[parentIndex];

heap->heapData[parentIndex]=Max;

curPos=parentIndex;//向上移动

}

else

{

break;

}

}

}

void insertHeap(LPHEAP heap, int data) //放到当前堆的最后面并按条件往上移

{

++heap->sizeHeap;

heap->heapData[heap->sizeHeap]=data;

moveToCorrectPos(heap,heap->sizeHeap);

}

int popHeap(LPHEAP heap)

{

int Max=heap->heapData[1];

int curPos=1;

int childIndex=curPos\*2;

while(childIndex<=heap->sizeHeap)

{

int temp = heap->heapData[childIndex];

if(childIndex+1<=heap->sizeHeap && temp<heap->heapData[childIndex+1])

{

temp=heap->heapData[++childIndex];

}

heap->heapData[curPos]=temp;

curPos = childIndex; // 下移一层，childIndex在上面或有更新

childIndex\*=2;

}

heap->heapData[curPos]=heap->heapData[heap->sizeHeap];

--heap->sizeHeap;

return Max;

}

void main()

{

LPHEAP heap=createHeap();

const int elem=9;

for(int i=1;i<elem;++i)

{

insertHeap(heap,i);

}

for(i=1;i<elem;++i)

{

printf("%d\t",heap->heapData[i]);

}

printf("\n");

while(!empty(heap))

{

printf("%d\t",popHeap(heap));

}

system("pause");

}

/\*

堆(优先队列)：动态维护一组数据中最小（大）的一个

9 8 6 7 3 2 5 1 4

9 8 7 6 5 4 3 2 1

\*/