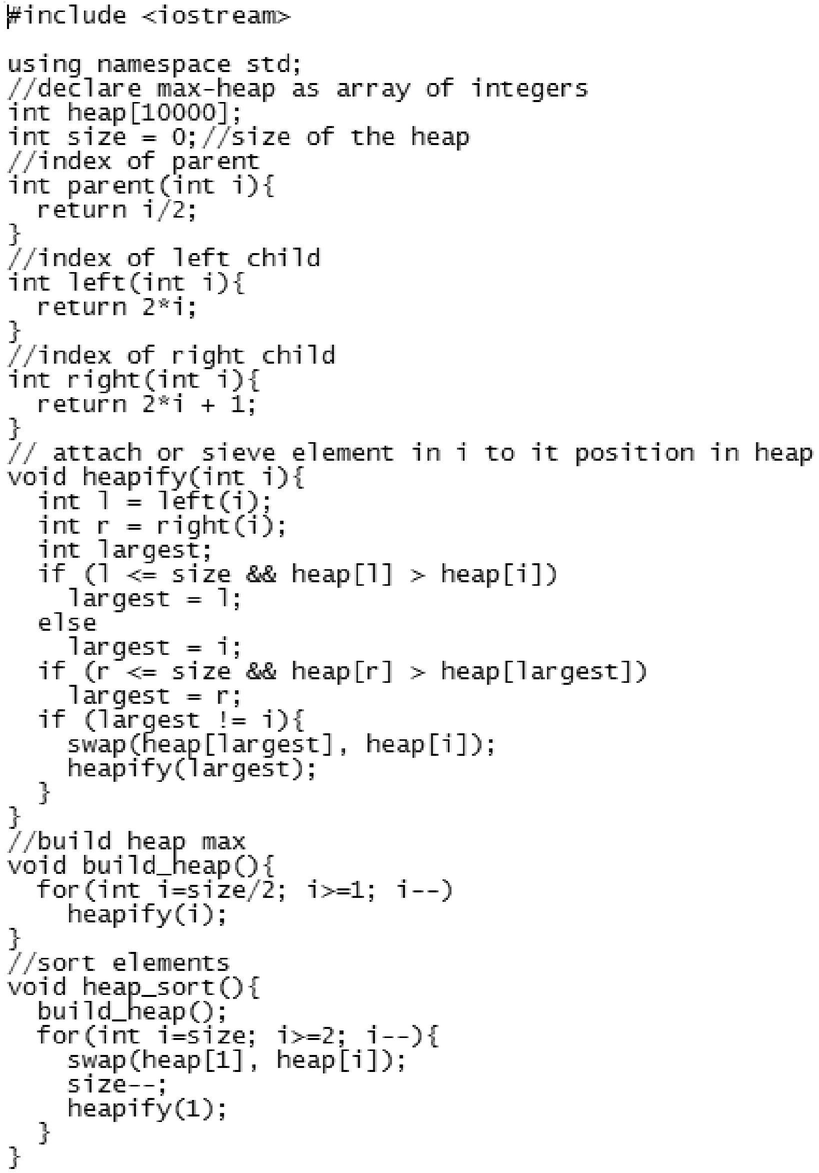
Kazakh-British Technical University

Algorithms and Data Structures, Spring 2011

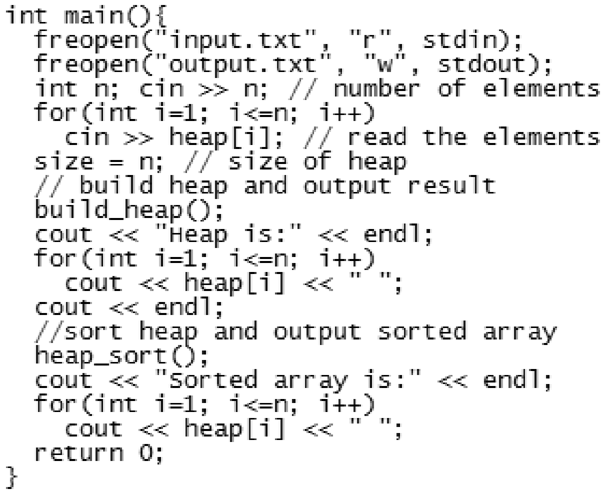
Lecture 6: Heapsort

Reading: [chapter 6] Thomas H. Cormen, Charles E. Leiserson. *Introduction to algorithms – 2-nd edition.* The following lecture note show only implementation of heap sort and priority queue data structure. For detailed explanation read suggested chapter of the book.

1 Heapsort



```
#include <iostream>

using namespace std;
//declare max-heap as array of integers
int heap[10000];
int size = 0;//size of the heap
//index of parent
int parent(int i){
  return i/2;
}
//index of left child
int left(int i){
  return 2*i;
}
//index of right child
int right(int i){
  return 2*i + 1;
}
// attach or sieve element in i to it position in heap
void heapify(int i){
  int l = left(i);
  int r = right(i);
  int largest;
  if (l <= size && heap[l] > heap[i])
    largest = l;
  else
    largest = i;
  if (r <= size && heap[r] > heap[largest])
    largest = r;
  if (largest != i){
    swap(heap[largest], heap[i]);
    heapify(largest);
  }
}
//build heap max
void build_heap(){
  for(int i=size/2; i>=1; i--)
    heapify(i);
}
//sort elements
void heap_sort(){
  build_heap();
  for(int i=size; i>=2; i--){
    swap(heap[1], heap[i]);
    size--;
    heapify(1);
  }
}
```



```
int main(){
  freopen("input.txt", "r", stdin);
  freopen("output.txt", "w", stdout);
  int n; cin >> n; // number of elements
  for(int i=1; i<=n; i++)
    cin >> heap[i]; // read the elements
  size = n; // size of heap
  // build heap and output result
  build_heap();
  cout << "Heap is:" << endl;
  for(int i=1; i<=n; i++)
    cout << heap[i] << " ";
  cout << endl;
  //sort heap and output sorted array
  heap_sort();
  cout << "Sorted array is:" << endl;
  for(int i=1; i<=n; i++)
    cout << heap[i] << " ";
  return 0;
}
```

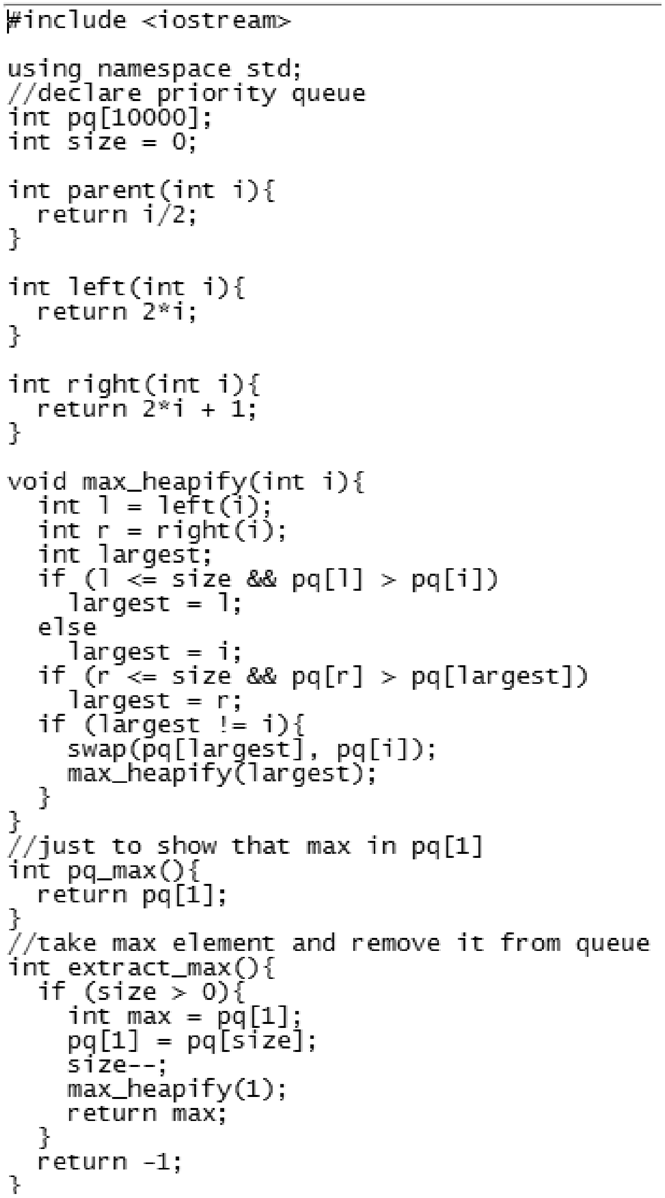
Input

```
10
4 1 3 2 16 9 10 14 8 7
```

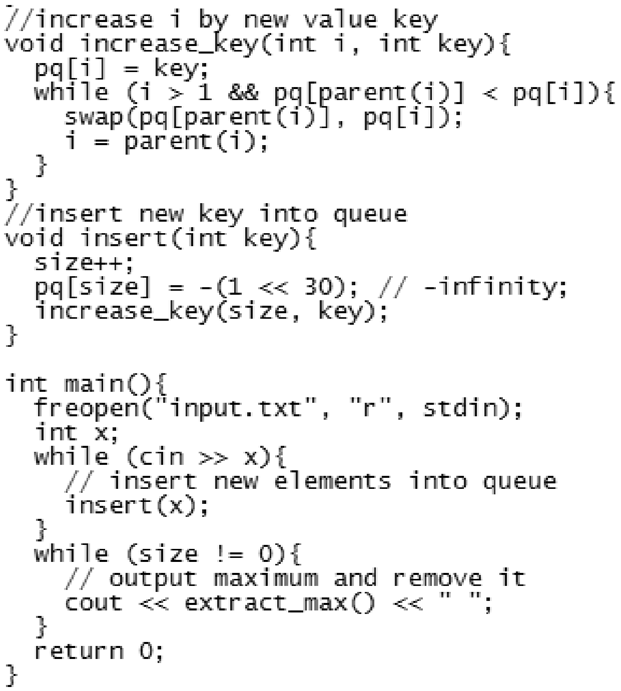
Output

```
Heap is:
16 14 10 8 7 9 3 2 4 1
Sorted array is:
1 2 3 4 7 8 9 10 14 16
```

2 Priority Queue



```
#include <iostream>

using namespace std;
//declare priority queue
int pq[10000];
int size = 0;

int parent(int i){
  return i/2;
}

int left(int i){
  return 2*i;
}

int right(int i){
  return 2*i + 1;
}

void max_heapify(int i){
  int l = left(i);
  int r = right(i);
  int largest;
  if (l <= size && pq[l] > pq[i])
    largest = l;
  else
    largest = i;
  if (r <= size && pq[r] > pq[largest])
    largest = r;
  if (largest != i){
    swap(pq[largest], pq[i]);
    max_heapify(largest);
  }
}
//just to show that max in pq[1]
int pq_max(){
  return pq[1];
}
//take max element and remove it from queue
int extract_max(){
  if (size > 0){
    int max = pq[1];
    pq[1] = pq[size];
    size--;
    max_heapify(1);
    return max;
  }
  return -1;
}
```



```
//increase i by new value key
void increase_key(int i, int key){
  pq[i] = key;
  while (i > 1 && pq[parent(i)] < pq[i]){
    swap(pq[parent(i)], pq[i]);
    i = parent(i);
  }
}
//insert new key into queue
void insert(int key){
  size++;
  pq[size] = -(1 << 30); // -infinity;
  increase_key(size, key);
}

int main(){
  freopen("input.txt", "r", stdin);
  int x;
  while (cin >> x){
    // insert new elements into queue
    insert(x);
  }
  while (size != 0){
    // output maximum and remove it
    cout << extract_max() << " ";
  }
  return 0;
}
```

Input

```
4 1 3 2 16 9 10 14 8 7
```

Output

```
16 14 10 9 8 7 4 3 2 1
```

References

[1] [chapter 6] Thomas H. Cormen, Charles E. Leiserson. *Introduction to algorithms – 2-nd edition.* – USA : MIT Press, 2001. – 1180p.