**The Priority Queue ADT**

KV, Winter 2020

**// PriorityQueue.h**

**// after Timothy Budd, Data Structures in C++ (an older textbook).**

**#include <vector>**

**using namespace std;**

**template <class Itr>**

**void push\_heap(Itr, Itr);**

**template <class Itr>**

**void pop\_heap(Itr, Itr);**

**template <class Itr>**

**void adjust\_heap(Itr, unsigned int, unsigned int);**

**template <class T>**

**class PriorityQueue**

**{**

**public:**

**PriorityQueue(){}**

**PriorityQueue(const PriorityQueue<T>& other)**

**{**

**for (int i = 0; i < other.size(); i++)**

**pq.push\_back(other[i]);**

**}**

**bool empty() {return pq.empty();}**

**int size() {return pq.size();}**

**T& top() {return pq.front();}**

**void push(T& newElement)**

**{**

**pq.push\_back(newElement);**

**push\_heap(pq.begin(), pq.end());**

**}**

**void pop()**

**{**

**pop\_heap(pq.begin(), pq.end());**

**pq.pop\_back();**

**}**

**protected:**

**vector<T> pq;**

**};**

**template <class Itr>**

**void push\_heap(Itr start, Itr stop)**

**{**

**// references item at the very "end" of the pq**

**//container structure;**

**unsigned int position = (stop - start) - 1;**

**// parent position of position**

**unsigned int parent = (position - 1)/2;**

**while (position > 0 &&**

**\*(start + position) < \*(start + parent))**

**{**

**swap(\*(start + position), \*(start + parent));**

**position = parent;**

**parent = (position - 1)/2;**

**}**

**}**

**template <class Itr>**

**void pop\_heap(Itr start, Itr stop)**

**{**

**unsigned int lastPosition = (stop - start) - 1;**

**swap(\*start, \*(start + lastPosition));**

**adjust\_heap(start, lastPosition, 0);**

**}**

**template <class Itr>**

**void adjust\_heap(Itr start, unsigned int heapSize, unsigned int position)**

**{**

**while (position < heapSize)**

**{**

**unsigned int childpos = position \* 2 + 1;**

**if (childpos < heapSize)**

**{**

**if (childpos+1 < heapSize &&**

**\*(start + childpos +1) < \*(start +**

**childpos))**

**childpos++;**

**if (\*(start + position) < \*(start +**

**childpos))**

**return;**

**else**

**{**

**swap(\*(start + position), \*(start +**

**childpos));**

**}**

**}**

**position = childpos;**

**}**

**}**

**// PriorityQueueMain.cpp**

**#include <iostream>**

**#include <list>**

**#include "PriorityQueue.h"**

**using namespace std;**

**int main()**

**{**

**PriorityQueue<int> mypq;**

**int k, next;**

**cout << "How many values? ";**

**cin >> k;**

**cout << endl;**

**for (int i = 1; i <= k; i++)**

**{**

**cout << "Next value: ";**

**cin >> next;**

**mypq.push(next);**

**}**

**cout << endl << endl;**

**cout << "Now list and pop all values ..."**

**<< endl << endl;**

**while (!mypq.empty())**

**{**

**cout << mypq.top() << " ";**

**mypq.pop();**

**}**

**cout << endl << endl;**

**return 0;**

**}**