ლაბ 7: პრიორიტეტების რიგი

განსახილველი საკითხები:

- კლასი
- იმპლემენტაცია
- პროგრამა დრაივერი >>>
- სავარჯიშოები >>>

კლასი

როდესაც კლასის თარგის გაყოფა ხდება ორ ფაილად (კლასად და იმპლემენტაციად), პროგრამა დრაივერში უნდა მივუთითოთ ორივე - სათაურის ფაილიც და C++ ფაილიც.

რადგან კლასი თარგს წარმოადგენს, მისი მეთოდებისთვის პარამეტრების მითითება თავისებურად ხდება, რასაც ყურადღება უნდა მივაქციოთ.

კლასი "my_ptiority_queue.h":

```
#pragma once
template<typename T, typename Container = deque<T> >
class my_priority_queue
{
private:
	Container a;
public:
	my_priority_queue() {}
	template<typename InIt>
	my_priority_queue(InIt first, InIt last);
	void show();
	int size();
	bool empty();
	void pop();
	void push(T t);
	T top();
	void increase_key(int i, T k);
};
```

იმპლემენტაცია

იმპლემენტაცია მოვათავსოთ ფაილში "my_ptiority_queue.cpp":

```
#include<iostream>
#include<vector>
#include<deque>
#include<algorithm>
using namespace std;
#include "my_ptiority_queue.h"

template<typename T, typename Container = deque<T> >
template<typename InIt>
my_priority_queue<T, Container>::my_priority_queue(InIt first, InIt last)
{
	while (first != last)
	{
		a.push_back(*first);
```

```
			++first;
		}
		make_heap(a.begin(), a.end());

}
template<typename T, typename Container = deque<T> >
void my_priority_queue<T, Container>::show()
{
		for (int i = 0; i < a.size(); ++i)
			cout << a[i] << '\t';
		cout << endl;
}
template<typename T, typename Container = deque<T> >
int my_priority_queue<T, Container>::size()
{
		return a.size();
}
template<typename T, typename Container = deque<T> >
bool my_priority_queue<T, Container>::empty()
{
		return (0 == a.size());
}
template<typename T, typename Container = deque<T> >
void my_priority_queue<T, Container>::pop()
{
		if (a.size() == 0)
		{
			cout << "Error! queue is empty!" << endl;
			return;
		}
		pop_heap(a.begin(), a.end());
		a.pop_back();
}
template<typename T, typename Container = deque<T> >
void my_priority_queue<T, Container>::push(T t)
{
		a.push_back(t);
		push_heap(a.begin(), a.end());
}
template<typename T, typename Container = deque<T> >
T my_priority_queue<T, Container>::top()
{
		return a[0];
}
template<typename T, typename Container = deque<T> >
void my_priority_queue<T, Container>::increase_key(int i, T k)
{
		if (i < 0 || i >= a.size() || k <= a[i])
			cout << "Error!" << endl;
		else
		{
			a[i] = k;
			while (i>0 && a[(i - 1) / 2] < a[i])
			{
				T tmp = a[i];
```

```
                a[i] = a[(i - 1) / 2];
                a[(i - 1) / 2] = tmp;
                i = (i - 1) / 2;
            }
        }
}
```

<<< პროგრამა დრაივერი. განვიხილოთ შემდეგი მარტივი პროგრამა. როდესაც დავრწმუნდებით, რომ კლასის მეთოდები კორექტულადაა იმპლემენტირებული, უნდა წავშალოთ show() ფუნქცია და მისი იმპლემენტაცია.

```
#include<iostream>
#include<list>
using namespace std;
#include"my_ptiority_queue.h"
#include"my_ptiority_queue.cpp"

int main()
{
	list<int> lst{ 1, 2, 4, 7, 12, 4, 5, 6 };
	my_priority_queue<int> q(lst.begin(), lst.end());
	cout << "Priority queue contains:" << endl;
	q.show();
	cout << "Priority queue after first pop():" << endl;
	q.pop();
	q.show();
	cout << "Priority queue after pushing 313:" << endl;
	q.push(313);
	q.show();
	cout << "Poping elements until queue is empty:" << endl;
	while (!q.empty())
	{
		cout << q.top() << endl;
		q.pop();
	}
	cout << "Filling..." << endl;
	q.push(4);
	q.push(41);
	q.push(14);
	q.push(432);
	q.push(33);
	q.push(-12);
	q.push(353);
	q.push(-102);
	q.show();
	cout << "Encreasing key number 6 up to 333:" << endl;
	q.increase_key(6, 333);
	q.show();
	cout << "Again poping elements until queue is empty:" << endl;
	while (!q.empty())
	{
		cout << q.top() << endl;
		q.pop();
	}
}
```

შედეგს აქვს სახე:

```
Priority queue contains:
12      7       5       6       2       4       4       1
Priority queue after first pop():
7       6       5       1       2       4       4
Priority queue after pushing 313:
313     7       5       6       2       4       4       1
Poping elements until queue is empty:
313
7
6
5
4
4
2
1
Filling...
432     41      353     4       33      -12     14      -102
Encreasing key number 6 up to 333:
432     41      353     4       33      -12     333     -102
Again poping elements until queue is empty:
432
353
333
41
33
4
-12
-102
Press any key to continue . . .
```

<<< სავარჯიშოები

1. ვთქვათ, ფაილში “data.txt” მოთვსებულია მონაცემები რამდენიმე ტბის შესახებ:

   ```
   Paravani      37.5  2073  3.3
   Paliastomi    18.2  -0.3  3.2
   Tabackuri     14.2  1997  40.2
   Jandari       10.6  291   7.2
   Ritsa         1.49  884   101
   Bazaleti      1.22  879   7
   ```

   შევქმნათ კლასი, რომლის ობიექტები შეესაბამება ფაილის სტრიქონებს და პროგრამა დრაივერი, რომელიც ფაილიდან მონაცემებს შეიტანს პრიორიტეტების რიგში. შეასრულეთ მარტივი მანიპულაციები ამგვარად აგებულ რიგზე.

2. შექმენით მსგავსი პროგრამა ვულკანებისთვის.