TASK 2. IMPLEMENTATION OF A BINARY HEAP

PURPOSE

The main purpose of this work is to learn an internal structure of binary heap, approaches to place elements into the heap and to remove elements from it.

TASK

The task is to implement the binary heap in templated class **priority_queue**. The methods of this class are given below. Each variant has some additional methods which are not implemented in STL container **priority_queue**.

Methods to implement in each variant:

- push(…) – to add some element to the queue;
- pop() – to pop the top element from the queue;
- top() – returns the value of the top element;
- size() – returns the number of elements in the queue;
- empty() – returns true if the queue contains no element, returns false otherwise;
- clear() – remove all elements from the queue (there is no such method in STL analogue).

Special methods for each variant:

1. erase(…) – to erase some element with the specified value.
2. second_top(), third_top() – returns the second and third maximum (or minimum) element.
3. merge(…another priority_queue…) – to merge two **priority_queue**’s.
4. total_erase(…) – to erase all the elements with the specified value.
5. check(…) – returns true if there is element in the queue with the specified value.
6. help_to_sort(…reference to a vector of the same type…) – sorts the specified vector using this **priority_queue**.
7. switch_order() – switch maximum-heap to minimum-heap and vice versa.
8. middle() – returns the value of the middle element.
9. construct(…vector of the same type…) – to construct a **priority_queue** out of the specified vector.
10. between(… , …) – the number of elements with values between two specified values.
11. remove_tier(…) – remove all elements from the specified tier. In case there is no element at some tier, the method should do nothing.