***1.Постановка задачи.***

Основные задачи работы:

* изучение различных форм преставления очереди с приоритетами;
* реализация очереди с приоритетами посредством бинарной кучи;
* (псевдо)графическое отображение состояния очереди с приоритетами;
* оценка временной сложности реализованного алгоритма в худшем случае;
* определение временных затрат реализованного алгоритма в среднем случае.

***2.Теория***

**Двоичная куча, пирамида,** или**сортирующее дерево**— такое [двоичное дерево](http://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%BE%D0%B8%D1%87%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE), для которого выполнены три условия:

1. Значение в любой вершине не меньше, чем значения её потомков.
2. Глубина листьев (расстояние до корня) отличается не более чем на 1 слой.
3. Последний слой заполняется слева направо.

Существуют также кучи, где значение в любой вершине, наоборот, не больше, чем значения её потомков. Такие кучи называются min-heap, а кучи, описанные выше — max-heap. В дальнейшем рассматриваются только max-heap. Все действия с min-heap осуществляются аналогично.

Удобная структура данных для сортирующего дерева — [массив](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B5%D0%BA%D1%81%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2) A, у которого первый элемент, A[1] — элемент в корне, а потомками элемента A[i] являются A[2i] и A[2i+1] (при нумерации элементов с первого). При нумерации элементов с нулевого, корневой элемент — A[0], а потомки элемента A[i] — A[2i+1] и A[2i+2]. При таком способе хранения условия 2 и 3 выполнены автоматически.

Операция добавления происходит следующим образом: новый элемент помещается в конец очереди и его приоритет сравнивается с приоритетом элемента, который соответствует его предку, и в случае если приоритет этого элемента больше приоритета предка, то он меняется местами с этим элементом, дальше сравнивается с предком предка и так далее, пока элемент не придется менять или пока не будет рассмотрен элемент, не имеющий предка.

При извлечении элемента из очереди с приоритетами возвращается значение нулевого элемента, после чего на его место ставится последний элемент. Затем происходит спуск этого элемента до своего места. Спуск происходит за счет того, что приоритет этого элемента сравнивается с приоритетом, максимальным среди его потомков. Если он меньше, происходит замена и дальнейший спуск.

***Проверка оценки сложности.***

-Добавление элемента в кучу имеет сложность *О(log (heapSize)) = O(logN),* т.к. функция добавления **HeapAdd** содержит рекурсивную функцию **MoveUp**, которая будет выполнена *log(heapSize) = LogN* раз.

-Удаление элемента из кучи имеет сложность *О(log(heapSize)) = O(LogN).*

-Заполнение двоичной кучи, или преобразование массива случайных чисел в кучу, имеет сложность *О(heapSize\*log(heapSize)) = O(NLogN),* т.к. эта функция содержит в цикле до *heapSize* вызов функции **HeapAdd,** которая содержит в свою очередь функцию **MoveUp.**

***3.Руководство программиста***

Это программа разработана на языке С++ и содержит следующие функции.

***void outxy(int x, int y, int var)*  -** *функция, которая преобразует элементы бинарной кучи(представленная в виде массива) в текстовую форму и выводит по заданным координатам. Параметры: координаты места вывода на экран, элемент кучи.*

***void MoveDown(int index) -*** *функция, перемещает элемент кучи вниз. Параметры: индекс элемента.*

***int GetMax(void)*** - *извлекает элемент из кучи, содержит функцию подсветки при удалении* ***ChangeColorsD.***

***void MoveUp1(int index)* –** *поднимает элемент вверх, пока он больше предка. Параметры: индекс элемента.*

***void MoveUp(int index)* –** *поднимает элемент вверх, пока он больше предка, содержит функцию подсветки* ***ChangeColors****. Параметры: индекс элемента.*

***void ChangeColorsD(int \*mas)*–** *функция подсветки элементов при удалении. Параметры: указатель на массив.*

***void ChangeColors(int \*mas)*–** *функция подсветки элементов при перемещении. Параметры: указатель на массив.*

***void HeapAdd(int x) –*** *добавляет новый элемент в кучу, содержит функцию* ***MoveUp и interfeis.*** *Параметры: индекс элемента.*

***void HeapAdd1(int x) –*** *добавляет элемент в кучу содержит функцию* ***MoveUp1.***  *Параметры: индекс элемента.*

***void fillHeap(void) –*** *функция осуществляет заполнение кучи случайными числами.*

***void graph() –*** *функция осуществляет графический интерфейс, содержит функцию* ***colors.***

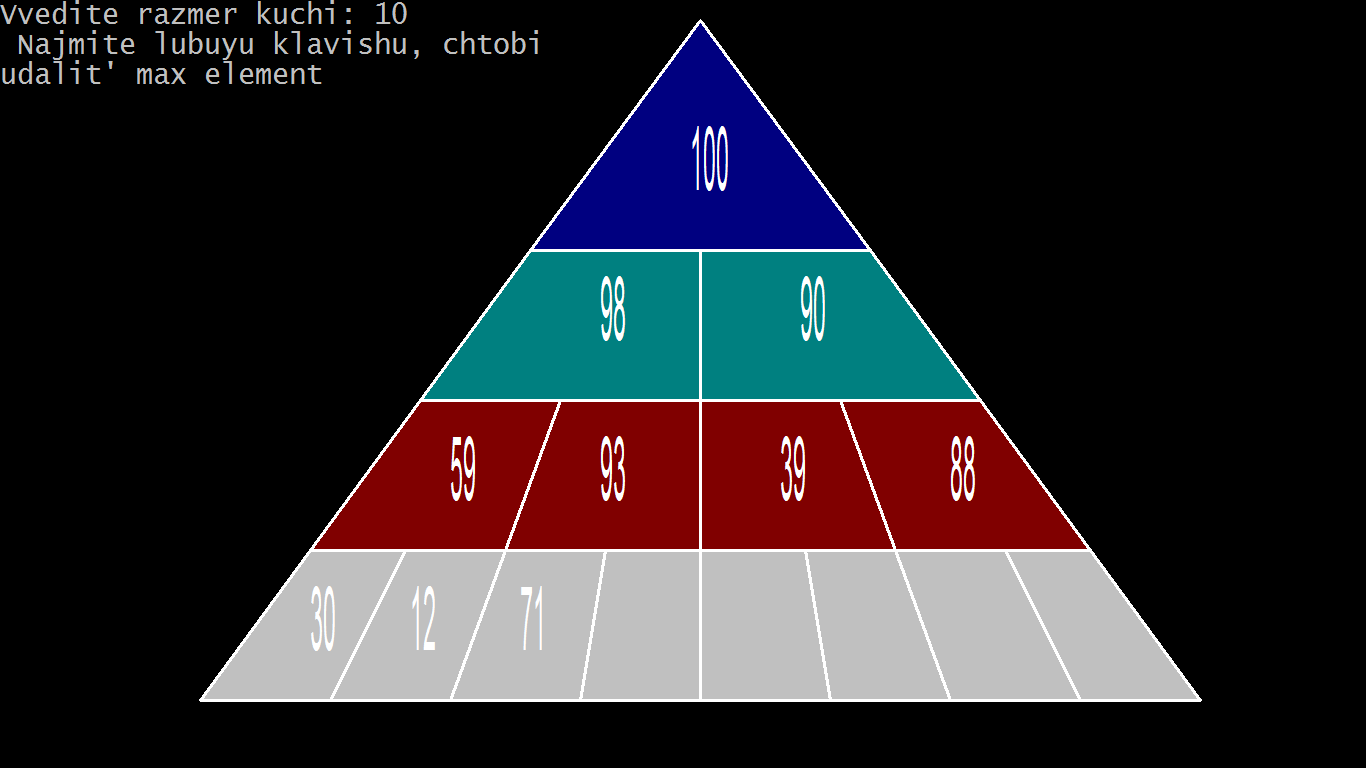
***void interfeis(int \*mas) –*** *функция осуществляет графическое отображение кучи. Параметры: указатель на массив.*

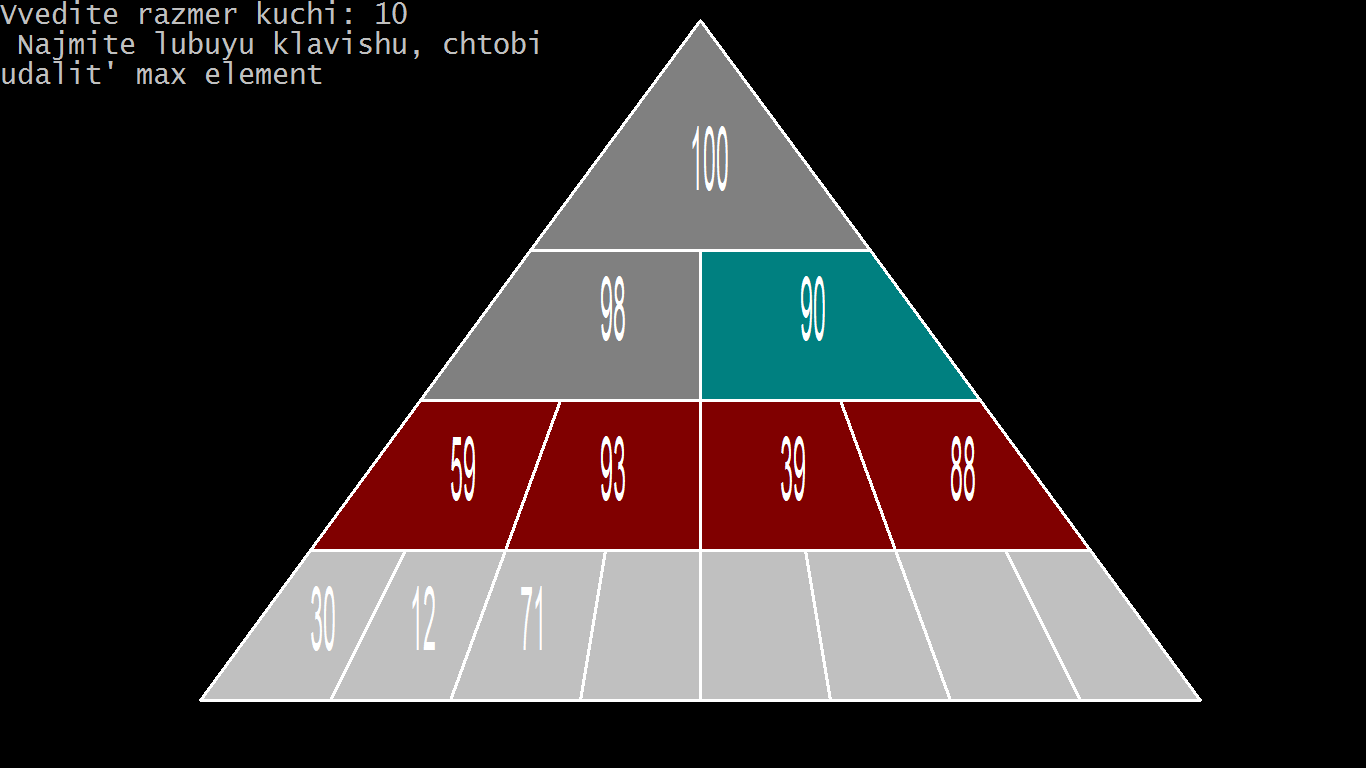
***void colors(int a,int b,int c,int d) –*** *функция осуществляет покраску слоев графического отображения кучи. Параметры: цвета уровней.*

***int main() –*** *основная функция программы, инициирует графический режим, осуществляет взаимодействие с пользователем, запрашивая размер кучи, введение нового элемента и удаление максимального.*

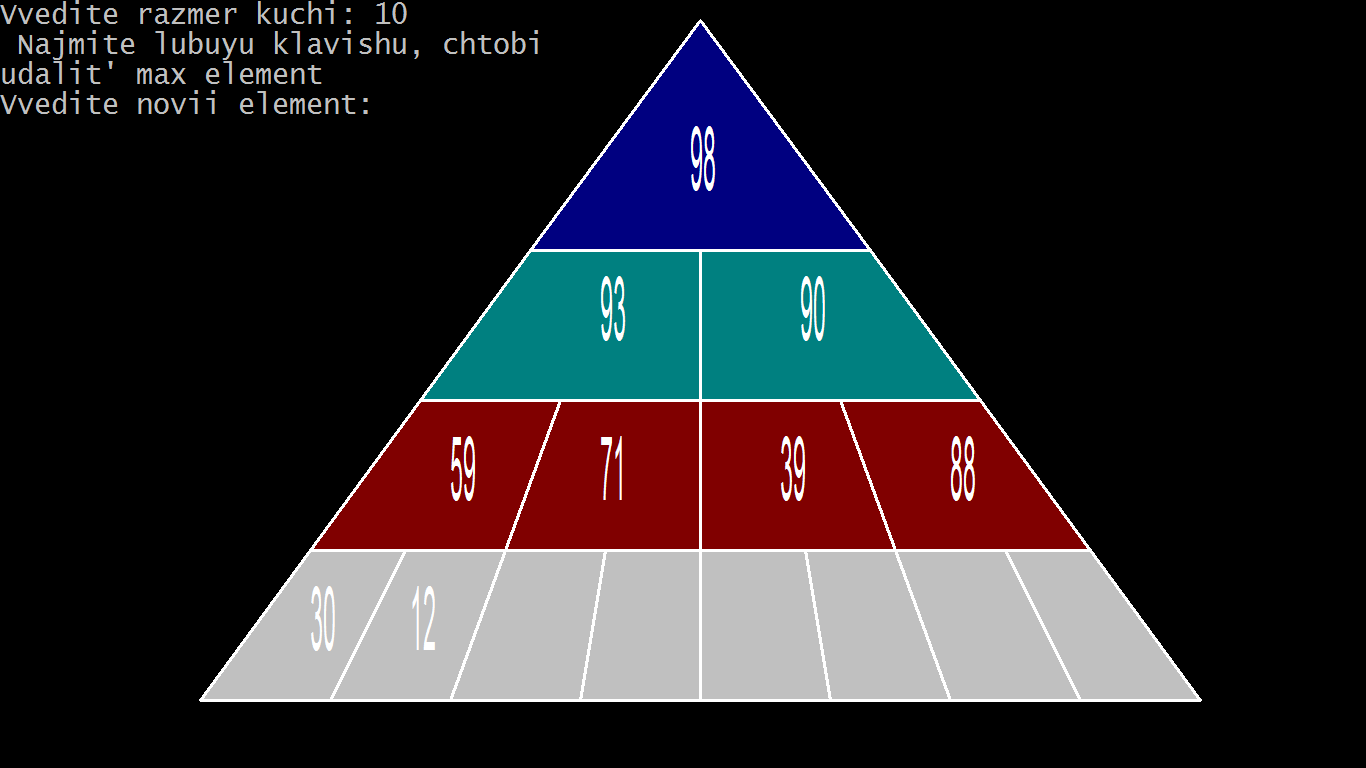
***4.Тестирование***

1) При запуске программы пользователя просят ввести размер кучи, с которым будет работать программа. После введения размера пирамида заполняется.

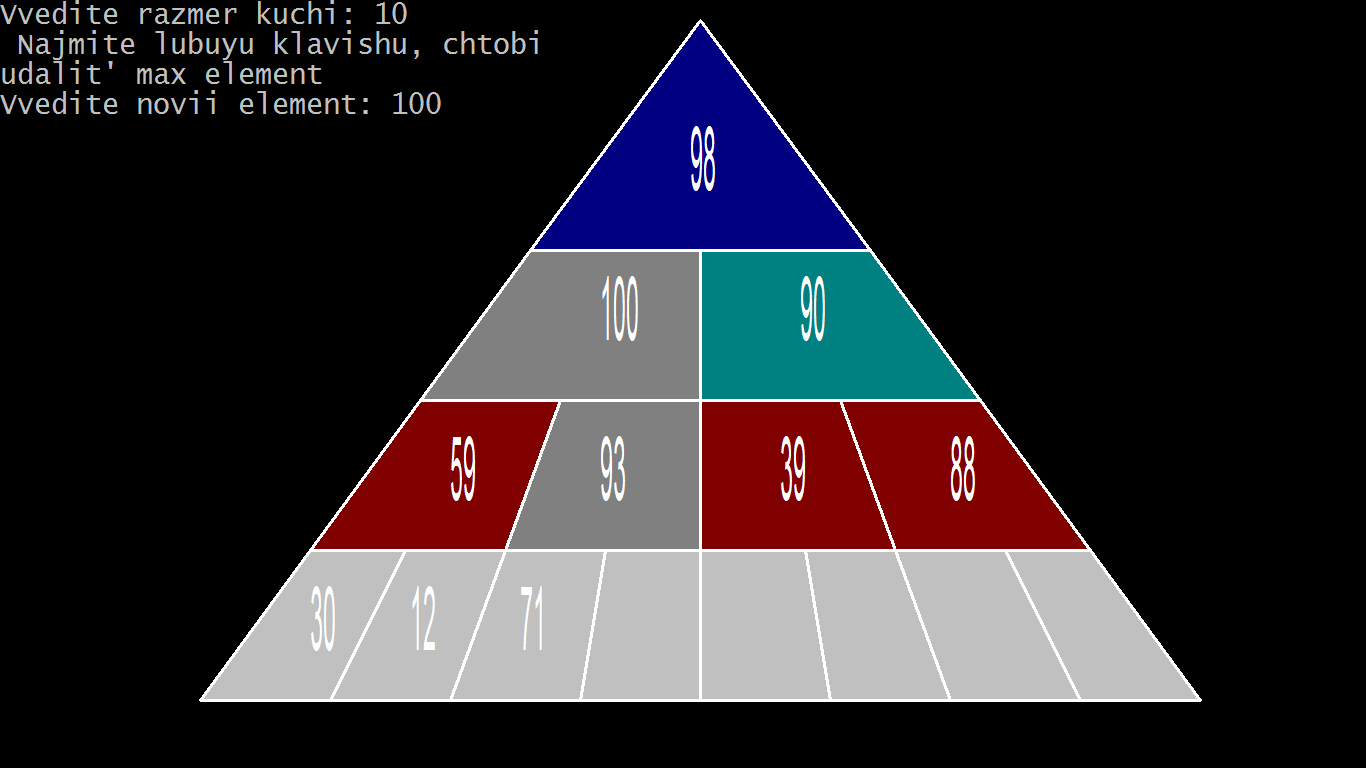




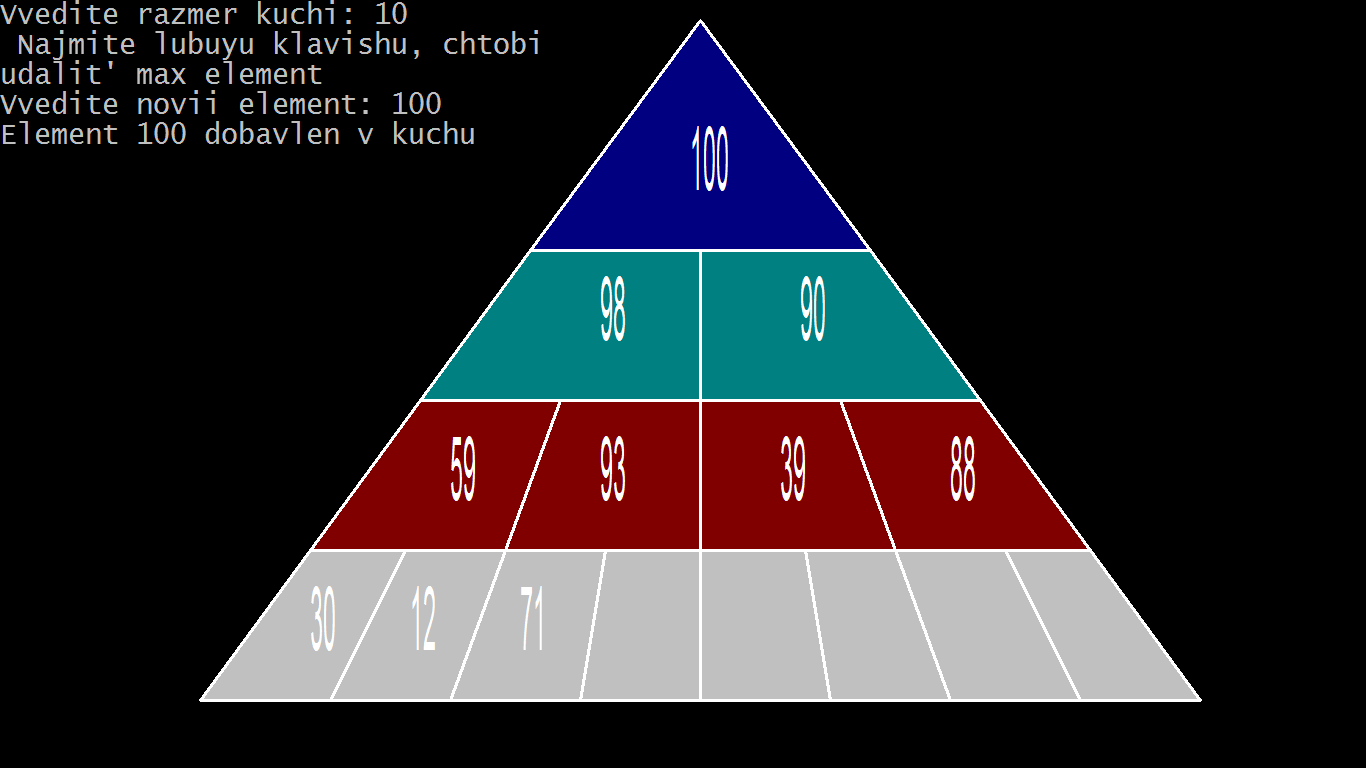
2) Выводится сообщение о том, что при нажатии любой клавиши удалится максимальный элемент. При нажатии на любую кнопку в таблице удаляется максимальный элемент и происходит перестраивание кучи.



3) Пользователя просят добавить новый элемент. При введении нового элемента происходит графическое отображение вставки и перехода на свою позицию этого элемента.



4) Появляется сообщение о том, что элемент добавлен в кучу. Программа завершает свою работу.



В результате тестирования программы, не было обнаружено отклонений от поставленной задачи, а это означает, что программа исправна и готова к использованию.