МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Алгоритмы и структуры данных»

Тема: Рандомизированная пирамида поиска– вставка и исключение. Текущий контроль

Студентка гр. 7383 _____ Прокопенко Н.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург

2018

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студентка Прокопенко Н.

Группа 7383

Тема работы: Рандомизированная пирамида поиска – вставка и исключение.

Текущий контроль.

Исходные данные:

Написать программу, генерирующую задания по рандомизированной пирамиде поиска.

Содержание пояснительной записки:

- Содержание
- Введение
- Формулировка задачи
- Решение задачи
- Тестирование
- Заключение
- Приложения А. Исходный код программы

Предполагаемый объем пояснительной записки не менее 15 страниц.

Дата выдачи задания: 19.10.2018

Дата сдачи курсовой работы:

Дата защиты курсовой работы:

Студент гр.7383 _____ Прокопенко Н.

Преподаватель _____ Размочаева Н.В.

АННОТАЦИЯ

В курсовой работе была разработана программа на языке программирования C++, которая позволяет создать рандомизированную пирамиду поиска. Программа, генерирующая варианты заданий по теме «Рандомизированная пирамида поиска – удаление и добавление элементов, построение дерева», а также прорешивающая все эти задания.

SUMMARY

In the course work was developed a program in the programming language C, which allows you to create a randomized search pyramid. A program that generates variants of tasks on the topic "Randomized search pyramid-removing and adding elements, building a tree", as well as solving all these tasks.

Содержание

ВВЕДЕНИЕ

Целью работы является практическое освоение стандартных алгоритмов статического кодирования и декодирования.

Задачей является создать программу, генерирующую задания по теме «Рандомизированная пирамида поиска» – вставка и исключение.

Тестирование будет проводится в OS Linux Mint 18.01.

1. ФОРМУЛИРОВКА ЗАДАЧИ

Написать программу для генерирования вариантов заданий и их решения по теме «Рандомизированная пирамида поиска» – вставка и исключение.

1.2. Теоретические сведения

Рандомизированная пирамида поиска (Treap) – это структура данных, объединяющая в себе бинарное дерево поиска и бинарную кучу (отсюда и второе её название: treap (tree + heap) и дерамида (дерево + пирамида).

Более строго, это бинарное дерево, в узлах которого хранятся пары (x, y), где x – это ключ, а y – это приоритет. Также оно является двоичным деревом поиска по x и пирамидой по y. Предполагая, что все x и все y являются различными, получаем, что если некоторый элемент дерева содержит (x0, y0), то у всех элементов в левом поддереве $x < x0$, у всех элементов в правом поддереве $x > x0$, а также и в левом, и в правом поддереве имеем: $y < y0$.

Дерамиды были предложены Сиделем (Siedel) и Арагон (Aragon) в 1996 г.

Недостатки декартового дерева:

- Большие накладные расходы на хранение: вместе с каждым элементом хранятся два-три указателя и случайный ключ y.
- Скорость доступа O(n) в худшем, хотя и маловероятном, случае.

2. РЕШЕНИЕ ЗАДАЧИ

В структуре node представлен узел БДП, содержащий в себе поля такие, как:

- int key – значение ключа элемента.
- node* left, right – для хранения указателей на правое и левое поддерево.
- int prior – для хранения приоритета.

А также в программе реализованы функции, такие как:

- Функция node* rotateright (node* p) – принимает на вход узел БДП и делает правый поворот вокруг этого узла.
- Функция node* rotateleft(node* p) – принимает на вход узел БДП и делает левый поворот вокруг этого узла.
- Функция node* insert (int key, node* root) добавляет узел с ключом k в дерево root, учитывая его приоритет и ключ. Если ключ k больше (меньше) ключа рассматриваемого узла, то он вставляется вправо (влево) от этого узла, при надобности делается правый или левый поворот.
- Функция node* merge(node *p, node *q) принимает на вход два дерева, которые она объединяет.
- Функция node* remove(node* p, int k) получает на вход дерево и ключ узла k в дереве p. Удаляет узел с заданным ключом, объединяя его правое и левое поддерево с помощью функции merge.
- Функция node* find(node* tree, int key) получает на вход дерево и ключ узла k в дереве p. Проверяет дерево на совпадение ключа узла k в дереве p.
- Функция node* Delete(node* p) получает на вход дерево и удаляет его.
- Функция void printPriority(node* root, ofstream &file) получает на

вход дерево и файловый поток. Записывает значение ключа и его приоритет в файл с генерируемыми заданиями.

- Функция void printtree(node* treenode, int l, ofstream &file) получает на вход дерево и файловый поток. Печатает полученные деревья в файл с ответами на генерируемые задания
- Int main – главная функция, которая инициализирует в консоли небольшой диалог с пользователем, спрашивая у него количество требуемых вариантов заданий, типы заданий, которые будут укомплектованы в варинтах заданий. После чего, используя вышеописанные методы и функции, генерирует нужное количество вариантов с выбранными пользователем типом заданий.

Код программы представлен в приложении А.

3. ТЕСТИРОВАНИЕ

Тестирование программы проводилось в OS Linux Mint 18.01. Использованные во время работы программы файлы были размещены в той же папке, что и запускаемый файл программы.

3.1. Первый запуск программы.

Введите кол-во вариантов заданий
5
Введите кол-во узлов пирамиды
8
Если вы хотите задание с исключением какого-либо элемента - введите 1
1
Если вы хотите задание с вставкой какого-либо элемента - введите 2
2
Пирамида #1 сгенерирована
Пирамида #2 сгенерирована
Пирамида #3 сгенерирована
Пирамида #4 сгенерирована
Пирамида #5 сгенерирована

Рисунок 3 – консольное меню при первом тестовом запуске программы.

Вариант №1
Ваша случайная пирамида:
Приоритет ключа[97] - 12
Приоритет ключа[80] - 51
Приоритет ключа[71] - 71
Приоритет ключа[74] - 74
Приоритет ключа[72] - 86
Приоритет ключа[16] - 87
Приоритет ключа[43] - 90
Приоритет ключа[54] - 98
Исключить из пирамиды 74 элемент и нарисовать пирамиду
Вставить в пирамиду 47 элемент и нарисовать пирамиду

Вариант №2
Ваша случайная пирамида:
Приоритет ключа[45] - 5
Приоритет ключа[96] - 92
Приоритет ключа[9] - 14
Приоритет ключа[26] - 16
Приоритет ключа[27] - 34
Приоритет ключа[41] - 47
Приоритет ключа[30] - 53
Приоритет ключа[29] - 76
Исключить из пирамиды 30 элемент и нарисовать пирамиду

Вставить в пирамиду 92 элемент и нарисовать пирамиду

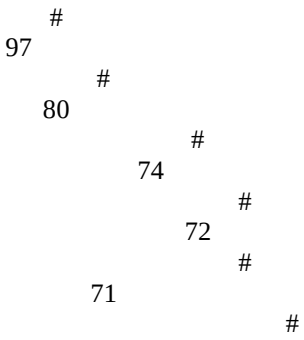
Вариант №3
Ваша случайная пирамида:
Приоритет ключа[14] - 19
Приоритет ключа[76] - 46
Приоритет ключа[84] - 46
Приоритет ключа[50] - 51
Приоритет ключа[18] - 51
Приоритет ключа[49] - 54
Приоритет ключа[2] - 47
Исключить из пирамиды 14 элемент и нарисовать пирамиду
Вставить в пирамиду 45 элемент и нарисовать пирамиду

Вариант №4
Ваша случайная пирамида:
Приоритет ключа[97] - 13
Приоритет ключа[81] - 16
Приоритет ключа[90] - 80
Приоритет ключа[53] - 42
Приоритет ключа[61] - 79
Приоритет ключа[75] - 83
Приоритет ключа[60] - 80
Приоритет ключа[13] - 81
Исключить из пирамиды 81 элемент и нарисовать пирамиду
Вставить в пирамиду 43 элемент и нарисовать пирамиду

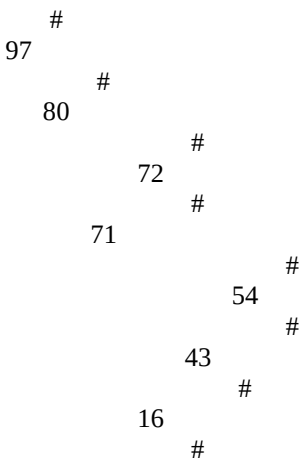
Вариант №5
Ваша случайная пирамида:
Приоритет ключа[92] - 2
Приоритет ключа[61] - 20
Приоритет ключа[88] - 37
Приоритет ключа[82] - 46
Приоритет ключа[77] - 54
Приоритет ключа[78] - 71
Приоритет ключа[21] - 78
Исключить из пирамиды 21 элемент и нарисовать пирамиду
Вставить в пирамиду 23 элемент и нарисовать пирамиду

Рисунок 4 – содержимое файла 1.txt после первого запуска.

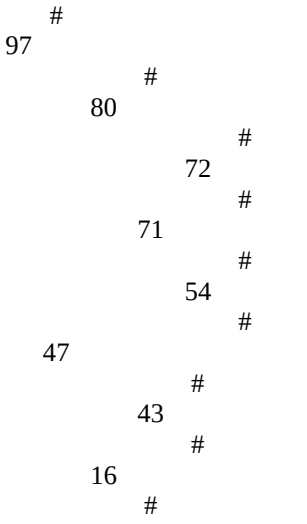
Вариант №1
Иллюстрация сгенерированной пирамиды



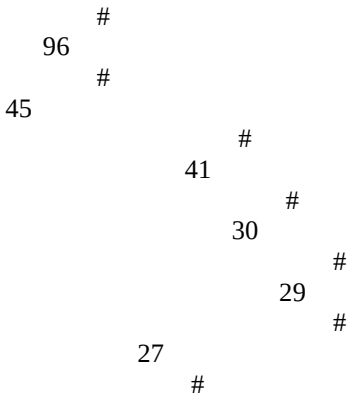
Пирамида перестроена с исключением элемента: 74



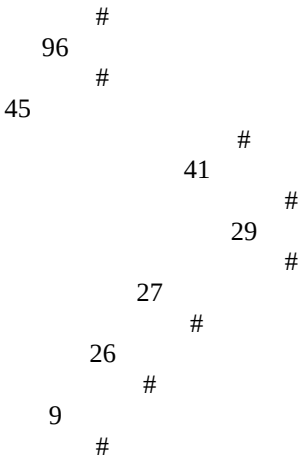
Пирамида перестроена с добавлением элемента: 47



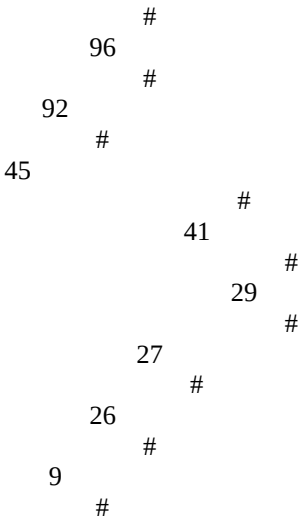
Вариант №2
Иллюстрация сгенерированной пирамиды



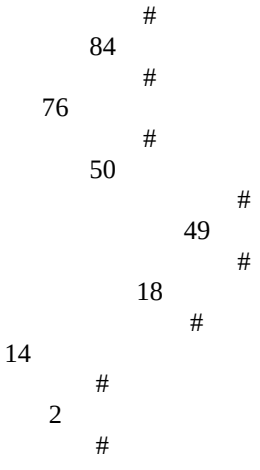
Пирамида перестроена с исключением элемента: 30



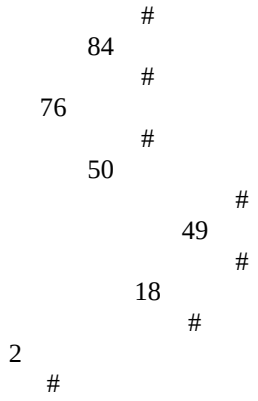
Пирамида перестроена с добавлением элемента: 92



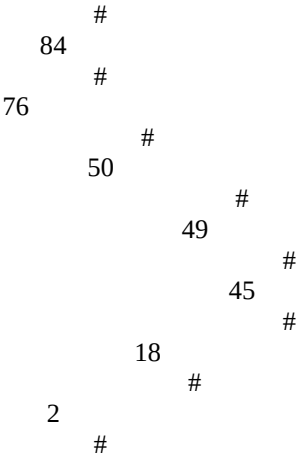
Вариант №3
Иллюстрация сгенерированной пирамиды



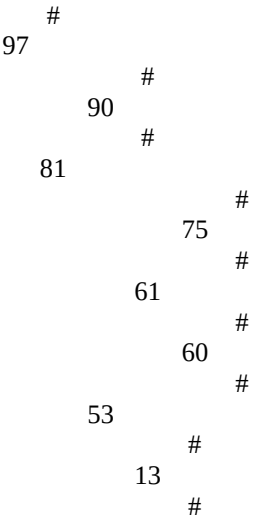
Пирамида перестроена с исключением элемента: 14



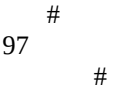
Пирамида перестроена с добавлением элемента: 45

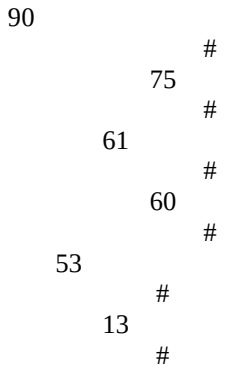


Вариант №4
Иллюстрация сгенерированной пирамиды

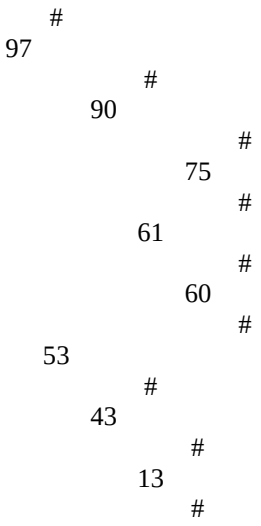


Пирамида перестроена с исключением элемента: 81

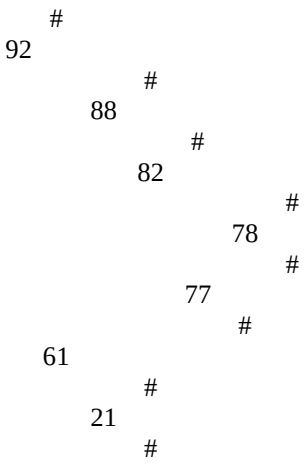




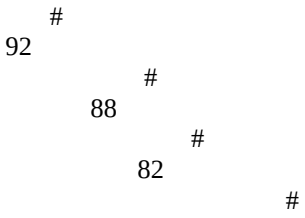
Пирамида перестроена с добавлением элемента: 43



Вариант №5
Иллюстрация сгенерированной пирамиды



Пирамида перестроена с исключением элемента: 21



78
#
77
#
61
#

Пирамида перестроена с добавлением элемента: 23

#
92
#
88
#
82
#
78
#
77
#
61
#
23
#

Рисунок 5 – содержимое файла answ.txt после первого запуска.

3.2. Второй запуск программы.

Введите кол-во вариантов заданий
7
Введите кол-во узлов пирамиды
5
Если вы хотите задание с исключением какого-либо элемента - введите 1
1
Если вы хотите задание с вставкой какого-либо элемента - введите 2
3
Пирамида #1 сгенерирована
Пирамида #2 сгенерирована
Пирамида #3 сгенерирована
Пирамида #4 сгенерирована
Пирамида #5 сгенерирована
Пирамида #6 сгенерирована
Пирамида #7 сгенерирована

Рисунок 6 – консольное меню после второго запуска.

Вариант №1
Ваша случайная пирамида:
Приоритет ключа[82] - 27
Приоритет ключа[25] - 61
Приоритет ключа[37] - 99
Приоритет ключа[2] - 68
Исключить из пирамиды 37 элемент и нарисовать пирамиду

Вариант №2
Ваша случайная пирамида:
Приоритет ключа[68] - 7
Приоритет ключа[97] - 56

Приоритет ключа[18] - 24
Приоритет ключа[25] - 42
Приоритет ключа[47] - 98
Исключить из пирамиды 68 элемент и нарисовать пирамиду

Вариант №3
Ваша случайная пирамида:
Приоритет ключа[16] - 11
Приоритет ключа[48] - 70
Приоритет ключа[80] - 73
Приоритет ключа[63] - 78
Приоритет ключа[10] - 84
Исключить из пирамиды 63 элемент и нарисовать пирамиду

Вариант №4
Ваша случайная пирамида:
Приоритет ключа[11] - 60
Приоритет ключа[98] - 61
Приоритет ключа[66] - 67
Приоритет ключа[95] - 70
Приоритет ключа[49] - 99
Исключить из пирамиды 11 элемент и нарисовать пирамиду

Вариант №5
Ваша случайная пирамида:
Приоритет ключа[44] - 42
Приоритет ключа[78] - 51
Приоритет ключа[95] - 85
Приоритет ключа[60] - 93
Приоритет ключа[21] - 58
Исключить из пирамиды 78 элемент и нарисовать пирамиду

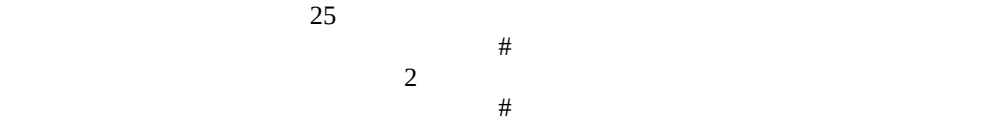
Вариант №6
Ваша случайная пирамида:
Приоритет ключа[19] - 6
Приоритет ключа[45] - 19
Приоритет ключа[85] - 50
Приоритет ключа[92] - 75
Приоритет ключа[34] - 66
Исключить из пирамиды 34 элемент и нарисовать пирамиду

Вариант №7
Ваша случайная пирамида:
Приоритет ключа[38] - 26
Приоритет ключа[74] - 44
Приоритет ключа[78] - 68
Приоритет ключа[84] - 86
Приоритет ключа[28] - 92
Исключить из пирамиды 38 элемент и нарисовать пирамиду

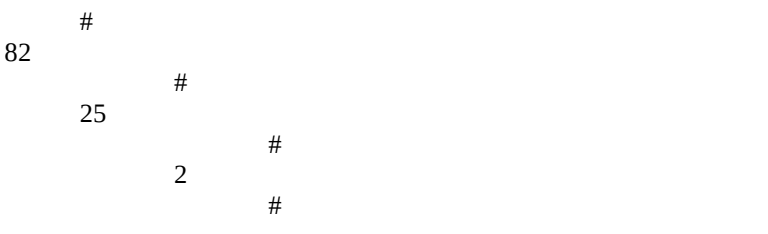
Рисунок 7 – содержимое файла task.txt после второго запуска.

Вариант №1
Иллюстрация сгенерированной пирамиды

#
82
#
37
#



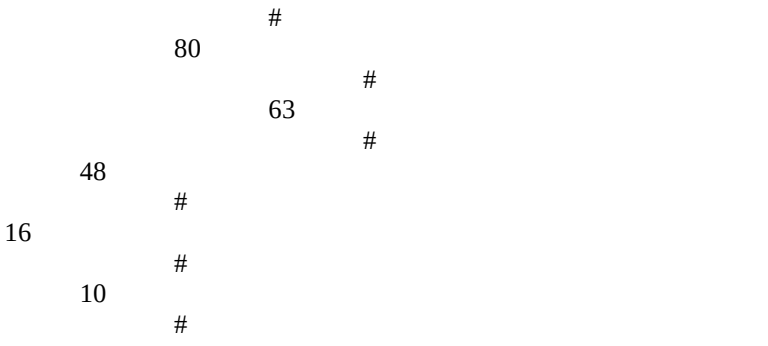
Пирамида перестроена с исключением элемента: 37



Вариант №2
Иллюстрация сгенерированной пирамиды

Пирамида перестроена с исключением элемента: 68

Вариант №3
Иллюстрация сгенерированной пирамиды



Пирамида перестроена с исключением элемента: 63

```
            80
                    #
        48
            #
16
            #
        10
            #
```

Вариант №4
Иллюстрация сгенерированной пирамиды

```
            #
        98
                        #
                95
                        #
            66
                        #
                49
                        #
11
        #
```

Пирамида перестроена с исключением элемента: 11

```
        #
98
                #
            95
                #
        66
                #
            49
                #
```

Вариант №5
Иллюстрация сгенерированной пирамиды

```
                #
            95
                #
        78
                #
            60
                #
44
            #
        21
            #
```

Пирамида перестроена с исключением элемента: 78

```
                #
            95
                #
        60
            #
44
            #
```

```
        21
                #
```

Вариант №6
Иллюстрация сгенерированной пирамиды

```
                        #
                92
                        #
        85
                #
45
                #
        34
                #
19
#
```

Пирамида перестроена с исключением элемента: 34

```
                        #
                92
                        #
        85
                #
45
        #
19
#
```

Вариант №7
Иллюстрация сгенерированной пирамиды

```
                        #
                84
                        #
        78
                #
74
        #
38
        #
28
        #
```

Пирамида перестроена с исключением элемента: 38

```
                        #
                84
                        #
        78
                #
74
        #
28
#
```

Рисунок 8 – содержимое файла answ.txt после второго запуска.

3.3. Третий запуск программы.

```
Введите кол-во вариантов заданий
0
```

Рисунок 9 – консольное меню после третьего запуска.

После третьего запуска программы в файлы task.txt и answ.txt ничего не было записано.

ЗАКЛЮЧЕНИЕ

В ходе работы была реализована программа на языке C++, которая взаимодействует с пользователем с помощью консольного меню, генерируя заданное количество вариантов на тему «Рандомизированная пирамида поиска», а также с заданным количеством узлов дерева.

Текущий контроль осуществлен с помощью записи вариантов заданий в файл, а ответов в другой файл.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Методические указания к лабораторным работам и практическим занятиям, по выполнению курсовой работы по дисциплине «Алгоритмы и структуры данных»»: учеб.-метод. пособие / сост.: С.А Ивановский , Т.Г. Фомичева , О.М. Шолохова.. СПб. 2017. 86 с.

2. Представление и обработка структурированных данных. Практикум по программированию. /С. А. Ивановский, В.А. Калмычков, А.А. Лисс, В.П. Самойленко. – СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2002.

3. Алексеев А. Ю., Ивановский С. А., Фролова С. А. Алгоритмы сортировки: учебное пособие. СПб. : Изд-во СПбГЭТУ «ЛЭТИ», 2009.

4. Кнут Д. Искусство программирования. Том 1: Основные алгоритмы : пер. с англ. – 3-е изд., испр. и доп. – М. : Издательский дом «Вильямс», 2007. (Доп. тираж 2009 г.)

5. Кнут Д. Искусство программирования. Том 3: Сортировка и поиск : пер. с англ. – 2-е изд., испр. и доп. – М. : Издательский дом «Вильямс», 2007.

ПРИЛОЖЕНИЕ А: ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <iostream>
#include <cmath>
#include <cstdlib>
#include <cstring>
#include <fstream>
#include <iomanip>

using namespace std;

long long int maxc = pow(2,32);

struct node {             // структура для представления узлов дерева
    int key;           // ключ-значение
    long long prior;   // приоритет
    node* left;        // указатель на левое поддерево
    node* right;       // указатель на правое поддерево
    node(int k) {
        key = k;       // инициализация структуры
        left = right = NULL;
        prior = 1+rand() % 99; // рандомные числа от 0 до 2^32
    }
};

node* rotateright(node* p) { // правый поворот вокруг узла p
    node* q = p->left;
    if( !q )
        return p;
    p->left = q->right;
    q->right = p;
    return q;
```

```
}

node* rotateleft(node* q) { // левый поворот вокруг узла q
    node* p = q->right;
    if( !p )
        return q;
    q->right = p->left;
    p->left = q;
    return p;
}

node* insert(int key, node* root) { // вставка
    if(!root) {
        node* p = new node(key);
        return (p);
    }
    if(key <= root->key)
    {
        root->left = insert(key, root->left);
        if(root->left->prior < root->prior)
            root = rotateright(root);
    }
    else {
        root->right = insert(key, root->right);
        if(root->right->prior < root->prior)
            root = rotateleft(root);
    }
    return root;
}

node* find( node* tree, int key) {
    if(!tree)
```

```
        return NULL;
    if(key == tree->key)
        return tree;
    if(key < tree->key)
        return find(tree->left, key);
    else
        return find(tree->right, key);
}

node* Delete(node* p){
    if (left)
        delete p->left;
    if (right)
        delete p->right;
    delete p;
    return p = NULL;
}

node* merge(node *p, node *q) {
    if (p == NULL) return q;
    if (q == NULL) return p;

    if (p->prior > q->prior) {
        p->right = merge(p->right, q);
        return p;
    }
    else {
        q->left = merge(p, q->left);
        return q;
    }
}
```

```
    node* remove(node* p, int k){ // удаление из дерева p первого
найденного узла с ключом k
        if( !p )
            return p;
        if( p->key == k ) {
            node* q = merge(p->left,p->right);
            delete p;
            return q;
        }
        else if( k < p->key )
            p->left = remove(p->left,k);
        else
            p->right = remove(p->right,k);
        return p;
    }

    void printPriority(node* root,ofstream &file) {
        if (!root)
            return;
        file<<"\tПриоритет ключа["<<setw(5)<<right<< root->key <<"]
- "<<setw(5)<<right<<root->prior<<endl;
        printPriority(root->right,file);
        printPriority(root->left,file);
    }

    void printtree(node* treenode, int l, ofstream &file) {
        if(treenode==NULL) {
            for(int i = 0; i<l; ++i)
                file<<"\t";
           file<<'#'<<endl;
            return;
        }
        printtree(treenode->right, l+1,file);
```

```
    for(int i = 0; i < l; i++)
        file << "\t";
    file << treenode->key<<endl;
    printtree(treenode->left,l+1,file);
}

int main() {
    node* treap = NULL; // пирамида поиска
    int call, j=0, i=0;
    int ch=0;
    int iskl=0, vstav=0;
    int vozr, foriskl, forvstav;
    ofstream file;
    file.open("1.txt");
    ofstream answ;
    answ.open("answ.txt");
    cout<<"Введите кол-во вариантов заданий"<<endl;
    cin>>call;
    if (!call)
         return 0;
    if (!ch){
        cout<<"\nВведите кол-во узлов пирамиды"<<endl;
          cin>>ch;
          cin.ignore();
          srand(ch);
      }
    while(call>j) {
          j++;
    int arr[ch];
    srand( time(0)+j );
  if (ch){
     if (iskl==0){
```

```
            cout<<"\nЕсли вы хотите задание с исключением какого-
либо элекмента - введите 1"<<endl;
            cin>>iskl;
           }
         }
         if (vstav==0){
           cout<<"\nЕсли вы хотите задание с вставкой какого-либо
элемент - введите 2"<<endl;
            cin>>vstav;
         }
        cout<<"\n\tПирамида #"<<j<<" сгенерирована"<<endl;
        for(i=0; i<ch; i++){
            arr[i]=1+rand() % 99;
        }
        file<<"\n\tВариант №"<<j;
        file<<"\nВаша случайная пирамида:\n";
        answ<<"_____
_____\n";
        answ<<"\n\tВариант №"<<j;
        answ<<"\nИллюстрация сгенерированной пирамиды\n\n";
        for(i=0; i<ch; i++){
             if(find(treap,arr[i])){
                continue;
            }
            treap = insert(arr[i], treap);
        }
        printPriority(treap,file);
        printtree(treap,0,answ);
        if(iskl==1){
             foriskl=arr[0+rand() % ch];
             file<<"Исключить из пирамиды "<<foriskl<<" элемент и
нарисовать пирамиду\n";
             treap = remove(treap, foriskl);
```

```
                answ<<"\nПирамида перестроена с исключением элемента: "<<foriskl<<"\n\n";
                printtree(treap, 0, answ);
           }

          if(vstav==2){
                srand(time(NULL)+arr[j]);
                forvstav=1+rand() % 99;
                file<<"Вставить в пирамиду "<<forvstav<<" элемент и нарисовать пирамиду\n";
                treap = insert(forvstav, treap);
                answ<<"\nПирамида перестроена с добавлением элемента: "<<forvstav<<"\n\n";
                printtree(treap, 0, answ);
          }
          treap = Delete(treap);
     }
     return 0;
     }
```