Задача 1

Полностью объяснить задачу из второго задания № 10. Под «полностью объяснить» подразумевается все — от свойств используемых структур данных до строчек кода.

Формат сдачи — предоставленный исходный код плюс устная беседа по нему.

Задача 2

Разобрать алгоритм быстрой сортировки массива (см. ниже). Оценить асимптотическую сложность алгоритма.

Формат сдачи — названная асимптотическая сложность (как «О большое от чего-то») плюс устная беседа по алгоритму.

Задача 3

На базе любой удобной задачи второго задания реализовать структуру данных и интерфейс к ней (указаны ниже). Программа при запуске должна читать команды из файла commands.txt, выполнять их, печатать результат на экран и завершаться. Никаких интерактивных действий от пользователя не предполагается.

Структура данных: бинарное дерево

Тип данных: строка не более 15 символов

Состав инструкций в commands.txt:

INSERT A — добавить элемент со значением A, ничего не печатать. Если такой элемент уже есть — добавить еще одну копию.

FIND A — найти элемент со значением A. Если элемент найден, напечатать FOUND. Если не найден — напечатать NOT FOUND.

DELETE A — найти и удалить элемент со значением A. Если элемента нет — ничего не делать. Если элементов несколько — удалить одну из копий.

DELETE_ALL A — найти и удалить все вхождения элемента со значением A.

Пример:

Инструкции в файле:

INSERT alice

INSERT alice

INSERT alice

DELETE alice

FIND alice

DELETE_ALL alice

FIND alice

Вывод программы:

FOUND (так как две alice еще остались)

NOT FOUND (так как теперь все alice удалены)

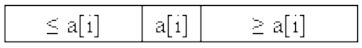
Формат сдачи — предоставленный исходный код плюс устная беседа по нему.

Быстрая сортировка массива

"Быстрая сортировка", хоть и была разработана более 40 лет назад, является наиболее широко применяемым и одним их самых эффективных алгоритмов.

Метод основан на подходе "разделяй-и-властвуй". Общая схема такова:

1. из массива выбирается некоторый опорный элемент a[i],
2. запускается процедура разделения массива, которая перемещает все ключи, меньшие, либо равные a[i], влево от него, а все ключи, большие, либо равные a[i] - вправо,
3. теперь массив состоит из двух подмножеств, причем левое меньше, либо равно правого,



4. для обоих подмассивов: если в подмассиве более двух элементов, рекурсивно запускаем для него ту же процедуру.

В конце получится полностью отсортированная последовательность.

Рассмотрим алгоритм подробнее.

Разделение массива

На входе массив a[0]...a[N] и опорный элемент p, по которому будет производиться разделение.

1. Введем два указателя: i и j. В начале алгоритма они указывают, соответственно, на левый и правый конец последовательности.
2. Будем двигать указатель i с шагом в 1 элемент по направлению к концу массива, пока не будет найден элемент a[i] >= p. Затем аналогичным образом начнем двигать указатель j от конца массива к началу, пока не будет найден a[j] <= p.
3. Далее, если i <= j, меняем a[i] и a[j] местами и продолжаем двигать i,j по тем же правилам...
4. Повторяем шаг 3, пока i <= j.

Рассмотрим работу процедуры для массива a[0]...a[6] и опорного элемента p = a[3].

Теперь массив разделен на две части: все элементы левой меньше либо равны p, все элементы правой - больше, либо равны p. Разделение завершено.

Общий алгоритм

Псевдокод.

```
quickSort ( массив a, верхняя граница N ) {
   Выбрать опорный элемент p - середину массива
   Разделить массив по этому элементу
   Если подмассив слева от p содержит более одного элемента,
      вызвать quickSort для него.
```

Если подмассив справа от p содержит более одного элемента,
вызвать quickSort для него.
}

Реализация на Си:

```
template<class T>
void quickSortR(T* a, long N) {
// На входе - массив a[], a[N] - его последний элемент.
  long i = 0, j = N;           // поставить указатели на исходные места
  T temp, p;
  p = a[ N>>1 ];               // центральный элемент
  // процедура разделения
  do {
    while ( a[i] < p ) i++;
    while ( a[j] > p ) j--;
    if (i <= j) {
      temp = a[i]; a[i] = a[j]; a[j] = temp;
      i++; j--;
    }
  } while ( i<=j );
  // рекурсивные вызовы, если есть, что сортировать
  if ( j > 0 ) quickSortR(a, j);
  if ( N > i ) quickSortR(a+i, N-i);
}
```