Задача 1

Полностью объяснить задачу из второго задания № 15. 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 сортировки массива простыми вставками (см. ниже). Оценить асимптотическую сложность алгоритма.

Формат сдачи — устная беседа, в ходе беседы можно использовать свои картинки, но нельзя использовать литературу.

Задача 3

Добавить в калькулятор (задача 1.5) функцию возведения в степень (обозначается ^, 5^2 = 25).

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

Сортировка массива простыми вставками

Будем разбирать алгоритм, рассматривая его действия на i-м шаге. Последовательность к этому моменту разделена на две части: готовую a[0]...a[i] и неупорядоченную a[i+1]...a[n].

На следующем, (i+1)-м каждом шаге алгоритма берем a[i+1] и вставляем на нужное место в готовую часть массива.

Поиск подходящего места для очередного элемента входной последовательности осуществляется путем последовательных сравнений с элементом, стоящим перед ним. В зависимости от результата сравнения элемент либо остается на текущем месте(вставка завершена), либо они меняются местами и процесс повторяется.

Таким образом, в процессе вставки мы "просеиваем" элемент x к началу массива, останавливаясь в случае, когда

1. Найден элемент, меньший x или
2. Достигнуто начало последовательности.

Пример реализации:

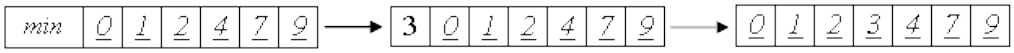
```
template<class T>
void insertSort(T a[], long size) {
  T x;
  long i, j;
  for ( i=0; i < size; i++) {  // цикл проходов, i - номер прохода
    x = a[i];
       // поиск места элемента в готовой последовательности
    for ( j=i-1; j>=0 && a[j] > x; j--)
      a[j+1] = a[j];    // сдвигаем элемент направо, пока не дошли
       // место найдено, вставить элемент
    a[j+1] = x;
  }
}
```

Алгоритм можно слегка улучшить. Заметим, что на каждом шаге внутреннего цикла проверяются 2 условия. Можно объединить из в одно, поставив в начало массива специальный сторожевой элемент. Он должен быть заведомо меньше всех остальных элементов массива.

Тогда при j=0 будет заведомо верно a[0] <= x. Цикл остановится на нулевом элементе, что и

было целью условия j>=0.

Таким образом, сортировка будет происходить правильным образом, а во внутреннем цикле станет на одно сравнение меньше. Однако, отсортированный массив будет не полон, так как из него исчезло первое число. Для окончания сортировки это число следует вернуть назад, а затем вставить в отсортированную последовательность a[1]...a[n].




```
// сортировка вставками со сторожевым элементом
template<class T>
inline void insertSortGuarded(T a[], long size) {
  T x;
  long i, j;
  T backup = a[0];                  // сохранить старый первый элемент
  setMin(a[0]);                     // заменить на минимальный
  // отсортировать массив
  for ( i=1; i < size; i++) {
    x = a[i];
    for ( j=i-1; a[j] > x; j--)
          a[j+1] = a[j];
        a[j+1] = x;
  }
  // вставить backup на правильное место
  for ( j=1; j<size && a[j] < backup; j++)
    a[j-1] = a[j];
  // вставка элемента
  a[j-1] = backup;
}
```

Функция setmin(T& x) задается отдельно. Она заменяет x на элемент, заведомо меньший(меньший или равный, если говорить точнее) всех элементов массива.