Задание №2. Анализ кода

Функция Func1 представляет собой бинарный поиск - искомый ключ key сравнивается с ключом элемента находящегося посередине отрезка [low; high] (значение middle). В случае если key больше значения середины, то поиск продолжается в отрезке [middle+1; high], иначе в отрезке [low, middle].

Назначение функции Func1 найти позицию, в которую будет добавляться новый элемент, так чтобы сохранить сортировку массива по ключу key из KeyValuePair.

Функция Func2 добавляет в отсортированный массив элемент со структурой “ключ-значение”. На вход передается массив a (по ссылке), а также ключ key и значение value.

Оптимизация (чтобы не было повторяющихся строк):

```
static void Func2(ref KeyValuePair<int, string>[] a, int key, string value)
{
  int pos;
  KeyValuePair<int, string> keyValuePair;

  Array.Resize(ref a, a.Length + 1);

  if (a.Length != 0)
  {
     if (key < a[0].Key)
        pos = 0;
     else if (key > a[a.Length - 1].Key)
        pos = a.Length;
     else
        pos = Func1(a, 0, a.Length - 1, key);

     for (int i = a.Length - 1; i > pos; i--)
         a[i] = a[i - 1];
  } else {
      pos = 0;
  }

  keyValuePair = new KeyValuePair<int, string>(key, value);
  a[pos] = keyValuePair;
}
```

Чтобы оптимизировать код нужно рассмотреть следующие строки:

Func1

```
 if (key > a[middle].Key)
   return Func1(a, middle + 1, high, key);

return Func1(a, low, middle, key);
```

Func2

```
for (int i = a.Length - 1; i > pos; i--)
   a[i] = a[i - 1];
```

Если в массиве *a* существует элемент с ключом равным ключу добавляемого элемента, то новый элемент следует расположить так, чтобы сдвигать наименьшее количество элементов. Значит нужно расположить новый элемент после существующего/их.

Допустим:
a = {<1, string>, <2, string>, <3, string>, <3, string>, <3, string>, <4, string>, <5, string>}
key = 3;

Пройдемся по Func2, вызывается Func1 со следующими аргументами:

```
pos = Func1(a, 0, a.Length - 1, key);
pos = Func1(a, 0, 6, 3);
```

1 итерация:
low = 0;
high = 6;
middle = 3;
Сравнивается *key = 3* и *a[3].Key = 3.* Так как условие *if (key > a[middle].Key)* не срабатывает, выполняется следующий код *return Func1(a, low, middle, key);* т.е. поиск продолжается в первой половине массива. Но нам необходимо продолжать поиск во второй половине массива. Этого можно достичь если изменить условие на *if (key >= a[middle].Key)*.
В таком случае будет меньше затрат по времени для смещения элементов в Func2.