Изменение размеров массива

Размер массива изменить НЕЛЬЗЯ
Но можно создать новый массив, скопировать в него элементы из старого и использовать ту же ссылочную переменную:

```
public class Ara{
public static void main(String[] args) {
        int[] elements = {1, 2, 3, 4, 5}; //массив из 5 элементов
        int[] tmp = new int[10];
        System.arraycopy(elements, 0, tmp, 0, elements.length);
        elements = tmp;
        for(int i:elements) {
                System.out.println(i);
                }
        }
}
```

В аргументах arraycopy() передается исходный массив, начальная позиция копирования в исходном массиве, приемный массив, начальная позиция копирования в приемном массиве и количество копируемых элементов.

Задачи

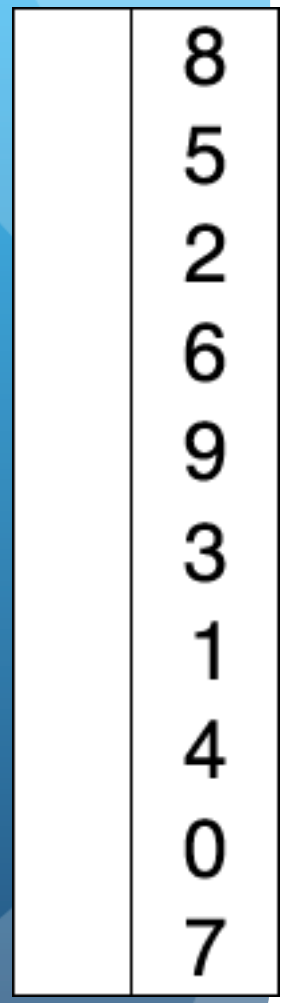
1. Дан массив размера N. Обнулить элементы массива, расположенные между его минимальным и максимальным элементами (не включая минимальный и максимальный элементы).

2. Дан массив размера N и два целых числа K и M ($1 \le K \le N$, $1 \le M \le 10$). После элемента массива с номером K вставить M новых элементов с нулевыми значениями.

Алгоритм сортировки выбором

Шаги алгоритма:

1. находим номер минимального значения в текущем списке
2. производим обмен этого значения со значением первой неотсортированной позиции (обмен не нужен, если минимальный элемент уже находится на данной позиции)
3. теперь сортируем хвост списка, исключив из рассмотрения уже отсортированные элементы



Сортировка выбором

```
   int[] arr = {4, 4, 9, 2, 3};
/*По очереди будем просматривать все подмножества
  элементов массива (0 - последний, 1-последний,
  2-последний,...)*/
for (int i = 0; i < arr.length; i++) {
    /*Предполагаем, что первый элемент (в каждом
      подмножестве элементов) является минимальным */
    int min = arr[i];
    int min_i = i;
    /*В оставшейся части подмножества ищем элемент,
      который меньше предположенного минимума*/
    for (int j = i+1; j < arr.length; j++) {
        //Если находим, запоминаем его индекс
        if (arr[j] < min) {
            min = arr[j];
            min_i = j;
        }
    }
    /*Если нашелся элемент, меньший, чем на текущей позиции,
      меняем их местами*/
    if (i != min_i) {
        int tmp = arr[i];
        arr[i] = arr[min_i];
        arr[min_i] = tmp;
    }
 }
```

Задание

Модифицировать алгоритм таким образом, чтобы одновременно искать максимальное и минимальное значения массива и расставлять их на необходимые позиции.

Алгоритм сортировки «пузырьком»

Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются N-1 раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу массива («всплывает» до нужной позиции как пузырёк в воде, отсюда и название алгоритма).

Сортировка пузырьком

```
for(int i = 0; i < a.length - 1; i++)
    for(int j = 0; j < a.length - i - 1; j++)
        if(a[j] > a[j + 1])
          {
                    Int temp=a[j];
                    a[j] = a[j + 1];
                    a[j+1]=temp;
          }
```

Задания

1. В предидущем алгоритме избавиться от “холостых” проработок алгоритма.
2. Отсортировать массив с помощью алгоритма Шелла (смотреть слайд 10)

Array sort для сортировки по возрастанию

Метод sort() из класса Arrays использует усовершенствованный алгоритм Быстрой сортировки (Quicksort), который эффективен в большинстве случаев. Для того чтобы отсортировать массив, необходимо написать всего одну строку.

Arrays.sort(arr);// где arr это имя массива

Примечание: в начале файла предварительно нужно подключить библиотеку java.util.

import java.util.*;

Алгоритм Шелла

При сортировке Шелла сначала сравниваются и сортируются между собой значения, стоящие один от другого на некотором расстоянии d (о выборе значения d см. ниже). После этого процедура повторяется для некоторых меньших значений d, а завершается сортировка Шелла упорядочиванием элементов при d=1 (то есть обычной сортировкой вставками). Эффективность сортировки Шелла в определённых случаях обеспечивается тем, что элементы «быстрее» встают на свои места (в простых методах сортировки, например, пузырьковой, каждая перестановка двух элементов уменьшает количество инверсий в списке максимум на 1, а при сортировке Шелла это число может быть больше).

Пусть дан список A = (32, 95, 16, 82, 24, 66, 35, 19, 75, 54, 40, 43, 93, 68) и выполняется его сортировка методом Шелла, а в качестве значений d выбраны 5, 3, 1.

Пример:

На первом шаге сортируются подсписки A, составленные из всех элементов A, различающихся на 5 позиций, то есть подсписки A_{5,1} = (32, 66, 40), A_{5, 2} = (95, 35, 43), A_{5, 3} = (16, 19, 93), A_{5, 4} = (82, 75, 68), A_{5, 5} = (24, 54).

В полученном списке на втором шаге вновь сортируются подсписки из отстоящих на 3 позиции элементов.

Процесс завершается обычной сортировкой вставками получившегося списка.

Выбор d – как последовательность чисел Фибоначчи.

Алгоритм сортировки вставками

На каждом шаге алгоритма мы выбираем один из элементов входных данных и вставляем его на нужную позицию в уже отсортированном списке до тех пор, пока набор входных данных не будет исчерпан. Метод выбора очередного элемента из исходного массива произволен; может использоваться практически любой алгоритм выбора. Обычно (и с целью получения устойчивого алгоритма сортировки), элементы вставляются по порядку их появления во входном массиве.