Практическое занятие №4
Методы внутренней сортировки. Логические выражения

Задание: обсудить 1) Методы выбора и «пузырька» 2) Правила написания логических выражений; 3) Контрольная работа №1 (1 ак.час)

4.1 Упорядочение одномерного массива методом выбора.

1. Задача *Sort*. ПЗ.

Задание: Упорядочить массив «на месте» (без использования дополнительного массива) в порядке и направлении перемещения, указанными в условии. Для наглядности выводить измененный массив после каждого шага сортировки (прохода по массиву).

Условие: Упорядочить в порядке убывания (максимум в начало) элементы одномерного массива.

2. Уточненная ПЗ.

Задан вещественный одномерный массив ***a***, состоящий из ***n*** элементов.

Упорядочить в порядке убывания (максимум в начало) элементы заданного массива ***a***.

Замечание. Упорядочение делается принципиально в том же массиве (на месте), т.е. входной массив «портится». Чтобы не изменять исходные данные (это важное программистское правило, которое следует по возможности выполнять), можно использовать другой массив, например, ***b***, который и будет выходным. Тогда первым шагом обработки будет перепись ***a*** в ***b*** (при совпадении типа по имени для статических массивов выполняется простым присваиванием), а вторым – упорядочение ***b***.

3. Пример.

Пусть ***n***=6.

a

4	3	5	9	5	2
1	2	3	4	5	6

В результате должен получиться массив

a

9	5	5	4	3	2
1	2	3	4	5	6

4. Таблица данных

Класс	Имя	Описание (смысл), диапазон, точность	Тип	Структура	Формат
Входные данные	***a***	заданный массив, $\|a_i\|<100$, точн. 0.1	вещ	одномерный массив (10)	+XX.X (:5:1)
	n	число элементов массива a, $0 < n \le 10$	цел	простая переменная	XX (:2)
Выходные данные	***a***	упорядоченный массив, $\|a_i\|<100$, точн. 0.1	вещ	*	*
Промежу-точные	***i***	индекс текущего элемента, . . .*	*	*	---
	dat	входной файл *Sort_dat*<№теста>.*txt*	*	*	---
	res	выходной файл *Sort_res*<№теста>.*txt*	*	*	---
	amax	значение максимального элемента, . . .	*	*	---
	k	номер максимального элемента, . . .	*	*	---
	z	шаг упорядочения, $1 \le z \le 9$	*	*	---
			*	*	---

**Диапазоны, типы, точность и структуру для промежуточных параметров заполнить самим.*

5. Форма ввода

Сами, по аналогии с предыдущими задачами, согласуясь с типом элементов заданного массива

6. Форма вывода

```
обр1   Вариант 32. Упорядочение массива в порядке убывания
                        (максимум в начало)
обр2   Число элементов  = <n>
обр3   Значения элементов:
       <a[1]> <a[2]> ..... <a[n]>
обр4   По шагам:
обр5   <a[1]> <a[2]> ..... <a[n]>
обр6   Упорядоченный массив:
       <a[1]> <a[2]> ..... <a[n]>
```

7. Аномалии не рассматриваем

8. Функциональные тесты составить самостоятельно.

Обязательны тесты, где:

1) частично неупорядоченный массив (см Пример);
2) массив, упорядоченный в обратном порядке;
3) массив, уже упорядоченный в требуемом порядке.

И заполните полностью таблицу, указав номера подходящих тестов в пустых ячейках:

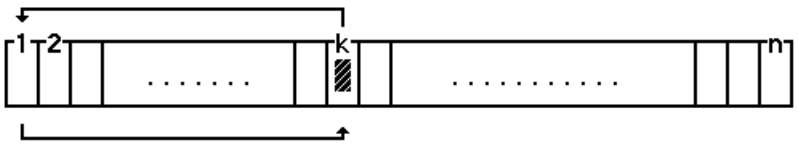
Исходные данные								**Результаты**		Тест№
	аном	*граница*	*сред*		*сред*	*граница*	*аном*	***перестановок***	*макс* = *N*-1	2
N	<1	1	(2	,	9)	10	>10		*мин* = 0	
Тест№	---			←	←		---		*сред* = (0,*N*-1)	1
	аном	*граница*	*сред*	*0*	*сред*	*граница*	*аном*		*не сущ* = при *N*=1	5
A[i]	<-99	-99	(-99,0)	0	(0,99)	99	>99		*0* = см.*мин*	
Тест№	---						---		*Макс.вычисл. нагрузка* = см. *макс*	2

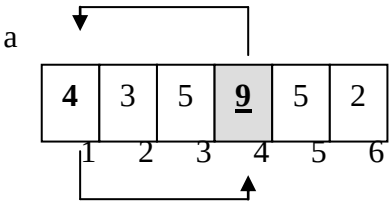
№	Входные данные	Ожидаемый результат	Смысл теста
1	*n*=6 ***A*** 4 3 5 **9** 5 2	Упорядоченный массив: 9 5 5 4 3 2	Частично упорядоченный массив из примера
2	*n*=10 ***A*** -99 **99** -7 50 -9 8 -8 6 -5 1	Пошагово (9 шагов): **99** -99 -7 50 -9 8 -8 6 -5 1 **99 50** -7 -99 -9 8 -8 6 -5 1 **99 50 8** -99 -9 -7 -8 6 -5 1 **99 50 8 6** -9 -7 -8 -99 -5 1 **99 50 8 6 1** -7 -8 -99 -5 -9 **99 50 8 6 1 -5** -8 -99 -7 -9 **99 50 8 6 1 -5 -7** -99 -8 -9 **99 50 8 6 1 -5 -7 -8** -99 -9 **99 50 8 6 1 -5 -7 -8 -9** -99 Упорядоченный массив: 99 50 8 6 1 -5 -7 -8 -9 -99	Частично упорядоченный массив для пошагового просмотра Видно направление сортировки – максимум в начало – за первый же шаг максимум встал в начало, затем второй по величине элемент встал на вторую позицию и т.д. **Максимальное число обменов**
3	*n*=10* ***A***	*	Массив, упорядоченный в обратном порядке
4	*	*	Массив, уже упорядоченный в требуемом порядке
5	*n* =1* ***A***	*	Особый случай

*Продолжить составление тестовых примеров самостоятельно

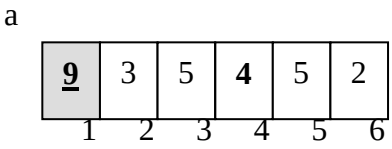
9. Метод сортировки – метод выбора (максимум в начало):

1 шаг (*z*=1). Ищем в массиве, начиная с *первого* элемента, значение максимального элемента ***amax*** и его номер ***k***, затем меняем значениями первый и ***k***-й элементы.

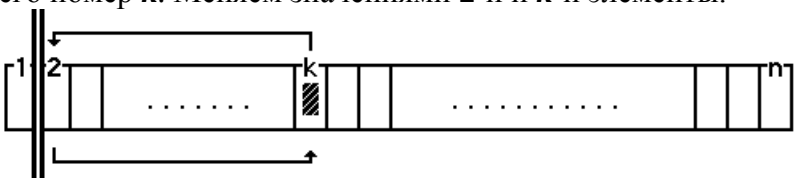


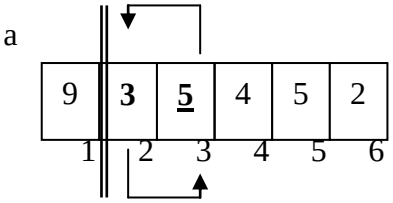


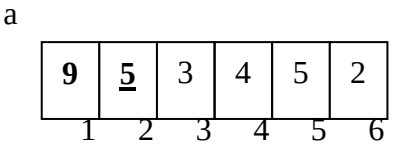
Первый элемент поставлен на место.



2-й шаг (z=2). Ищем в массиве, начиная со *второго* элемента, значение максимального элемент ***amax*** и его номер ***k***. Меняем значениями **2**-й и ***k***-й элементы.

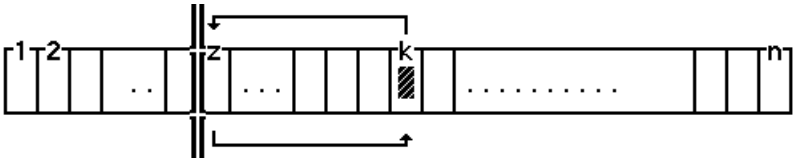






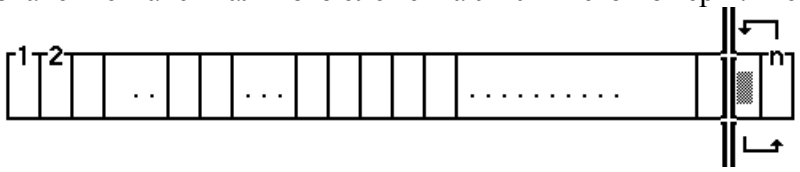
Теперь и *второй* элемент поставлен на место.

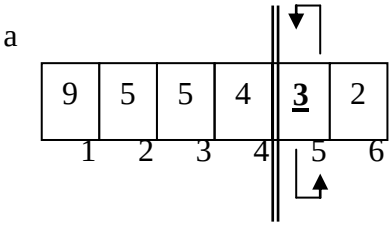
z*-ый шаг**. Часть массива с первого по (z***-1)-й элемент уже упорядочена.
Ищем в массиве, начиная с ***z***-го элемента, значение максимального элемента ***amax*** и его номер ***k***. Меняем значениями ***z***-й и ***k***-й элементы.



z-й элемент тоже поставлен на своё место.

Последний (***z*** = ***n***-1) шаг. Часть массива с первого по (***n***-2)-й элемент уже упорядочена.
Ищем в массиве, начиная с **(n-1)**-го элемента (их осталось всего два: последний и предпоследний), значение максимального элемента ***amax*** и его номер ***k***. Меняем значениями (***n*-1)**-й и ***k***-й элементы.



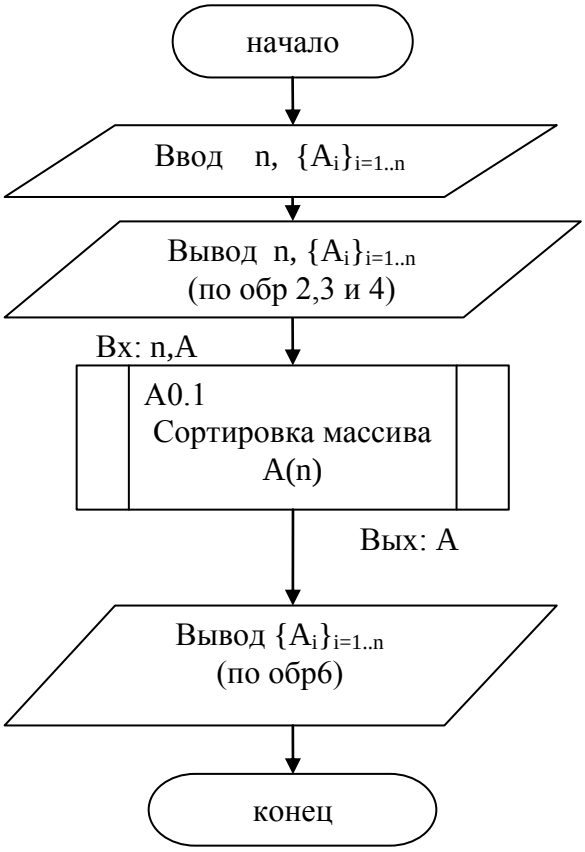


Теперь и последние два элемента тоже стоят в правильном порядке.
Массив упорядочен.

Замечание 1. Обратите внимание, что в данном массиве элементы фактически оставались на своих местах уже с третьего шага, и далее все элементы менялись местами сами с собой. Внесем небольшую модификацию в алгоритм: будем проверять индексы ***z*** и ***k*** меняющихся значениями элементов.
Замечание 2. Для направления перемещения «максимум/минимум *в конец*» удобнее шаги отсчитывать в обратном порядке с (*n*-1) до 2, и, соответственно, использовать цикл *for z*:=*n*-1 ***downto*** 2 *do* вместо *for z*:=1 ***to*** n-1 *do*. Сам максимум/минимум тоже логичнее начинать искать *с конца* неупорядоченной части массива, либо искать *последний* из элементов с максимальным/минимальным значением.

10. Алгоритм.

A0 Основной алгоритм
(отделяем ввод-вывод от обработки)



Раскрываем абстракцию A0.1
(упорядочение методом выбора)

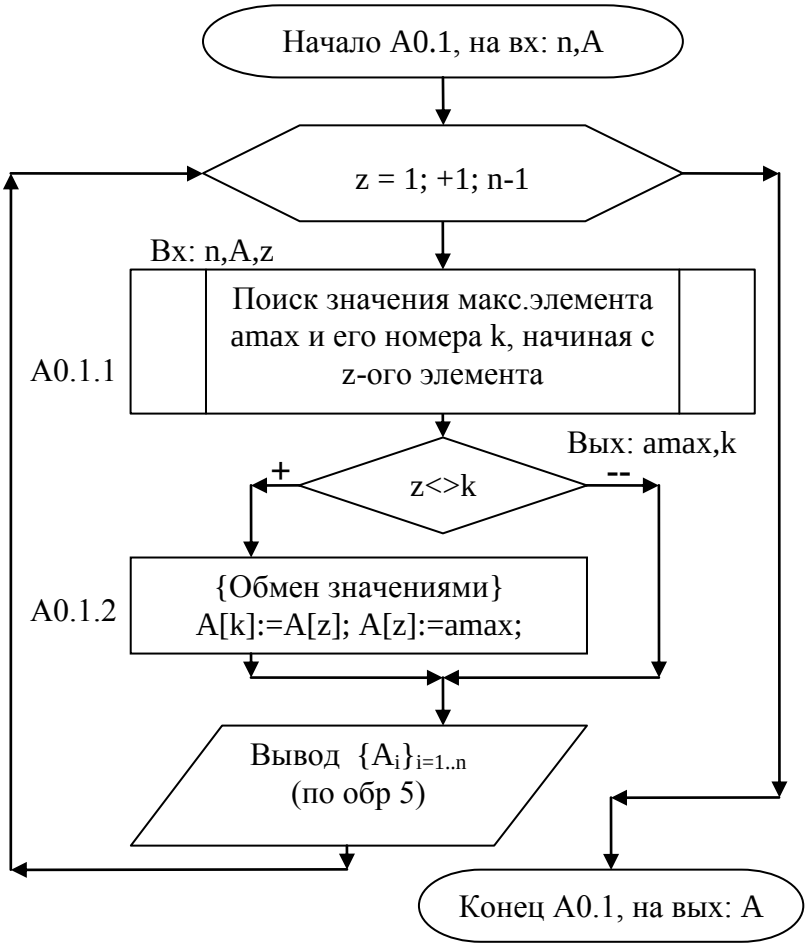
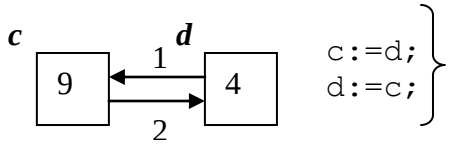


Рис. Блок-схемы алгоритма для задачи *Extremum*

Подзадача A0.1.1 – модификация задачи ***Extremum***: начальное значение индекса элемента, с которого начинается поиск, заменяется с 1 на ***z***.

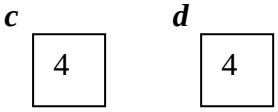
Подзадача A0.1.2 – обмен значениями ***z***-го и ***k***-го элементов (в ***amax*** лежит ***A***[***k***]).

Пусть в общем случае необходимо обменять значениями переменные ***c*** и ***d***.



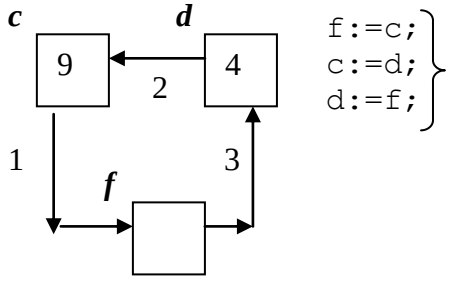
Способ 1).

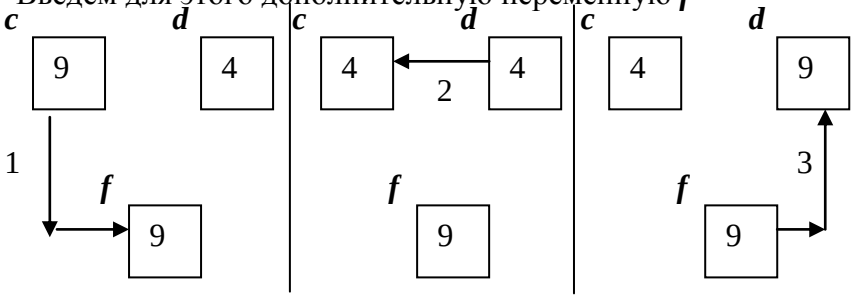
Неверно, т.к. после первого же присваивания теряется *начальное значение переменной c*:



Надо его сохранить…

Введем для этого дополнительную переменную ***f***





В рассмотренном алгоритме сортировки роль переменной ***c*** играет ***A***[***z***], роль переменной ***d*** играет ***A***[***k***].
Другие способы обмена: 2) ***c***:= ***c***+***d***; ***d***:= ***c***-***d***; ***c***:= ***c***-***d***; 3) для целых чисел ***c***:=***c*** *xor* ***d***; ***d***:= ***c*** *xor* ***d***; ***c***:=***c*** *xor* ***d***;

11. Программный код.

Написать самостоятельно, используя циклы for и ветвление if.

Подсказки в файлах Кодирование-алгоритмов.pdf и Базовые-алгоритмы.pdf, а также Пример-отчета-для-лабораторной-работы-2.doc

Далее будет рассмотрен еще один метод сортировки (пузырьком), и в итоге **документация по задачам сортировки** должна иметь вид:
– Пункты 1-8 общие, добавится одна логическая переменная;
– Основной алгоритм и программа с пустой заглушкой (пункты 10-11), куда можно будет вставить алгоритм упорядочения массива;
– Далее – решение методом выбора (с новой страницы пункты 9-11: метод, алгоритм, фрагмент кода, вставляемый вместо заглушки);
– Далее – решение методом пузырька (с новой страницы пункты 9-11: метод, алгоритм, фрагмент кода, вставляемый вместо заглушки).
Оба метода сортировки можно совместить в одной программе. При этом не забудьте заранее сделать копию исходного массива, чтобы не пытаться упорядочивать вторым методом уже упорядоченный первым методом массив.

4.2 Упорядочение одномерного массива методом "пузырька" (простым обменом)

1. Задача Bubble. Упорядочить элементы одномерного массива $A(n)$ в заданном порядке и направлении.

32 вариант: массив из вещественных чисел, по убыванию, максимум «всплывает пузырьком» *в начало* массива (значит, в этом варианте направление просмотра элементов *с конца в начало* массива).

6. Метод (сортировка «пузырьком»)

Сортировка «пузырьком» (***обменом***) (максимум в начало) – метод, при котором все *соседние* элементы массива попарно сравниваются друг с другом, начиная с конца, и меняются местами в том случае, если предшествующий элемент меньше последующего. В результате этого максимальный элемент постепенно смещается влево и за первый же проход по массиву занимает свое крайнее левое место в массиве, после чего он исключается из дальнейшей обработки. Затем процесс повторяется, и свое место занимает второй по величине элемент, который также исключается из дальнейшего рассмотрения. Для полной сортировки надо выполнить (n-1) проходов по массиву.

Сортировка пузырьком – один из самых простых, но медленных алгоритмов, имеющий много модификаций. Например, можно изменить его, добавив флажок, показывающий, были ли на данном проходе неупорядоченные *пары* элементов. Если таких пар не нашлось, закончить сортировку досрочно, а не за (n–1) шаг (проход).

Если последовательность сортируемых чисел расположить вертикально (первый элемент – вверху) и проследить за перемещениями элементов, то можно увидеть, что большие элементы, подобно пузырькам воздуха в воде, «всплывают» на соответствующую позицию. Поэтому упорядочение таким образом и называют еще сортировкой методом «пузырька», или пузырьковой сортировкой.

Добавим в таблицу данных две переменные:

Цел z – номер прохода. За один проход сравниваем пары *соседних* элементов $(A_i, A_{i+1})_{i=(n-1)..z}$, т.е. первая рассматриваемая пара (A_{n-1}, A_n), последняя – (A_z, A_{z+1}).

лог y = { истина, если массив упорядочен (неупорядоченных пар нет),
ложь, в противном случае (есть хотя бы одна пара неупорядоченных соседних элементов).

Вначале каждого прохода следует положить y=истина (массив упорядочен), но если встретим хотя бы одну неупорядоченную пару элементов, поменяем значение переменной y на ложь.

Если в конце прохода значение переменной y осталось истинным, значит, массив упорядочен.

Получаем:

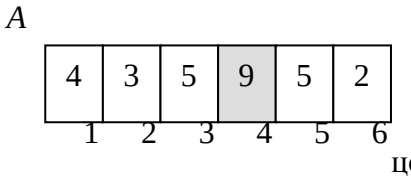
Начинам с первого прохода (z:=1)
Повторяем в цикле ДО следующие действия:
Положим, что массив упорядочен (y:=**истина**)
В цикле с *убывающим* параметром i= (n–1); -1; z
Сравниваем пары соседних элементов:
Если пара не упорядочена (A_i<A_{i+1}), то
Меняем значение y на **ложь**, и
выполняем обмен значениями A_i и A_{i+1}.
Увеличиваем номер прохода (z:=z+1), готовясь к следующему заходу
ДО тех пор, пока после очередного прохода y не останется истиной, либо не будет сделан (n–1) проход.

Замечание. Если у вас в задании *максимум* надо заставить всплывать не в начало, а *в конец* массива, то просмотр элементов надо начинать с начала массива, а скапливаться уже упорядоченные элементы будут в конце. При поиске *минимума* знак сравнения «<» смениться на «>».

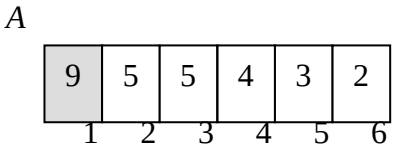
Получаем:
Начинам с первого прохода (z:=1)
Повторяем в цикле ДО следующие действия:
Положим, что массив упорядочен (y:=**истина**)
В цикле с *возрастающим* параметром i=1; +1; (n-z)
Сравниваем пары соседних элементов:
Если пара не упорядочена (A_i>A_{i+1}), то
Меняем значение y на **ложь**, и
выполняем обмен значениями A_i и A_{i+1}.
Увеличиваем номер прохода (z:=z+1), готовясь к следующему проходу
До тех пор, пока после очередного прохода y не останется истиной, либо не будет сделан (n–1) проход.

Пример

Пусть n=6,

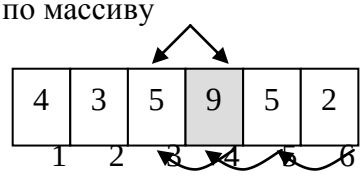


Упорядоченный массив:

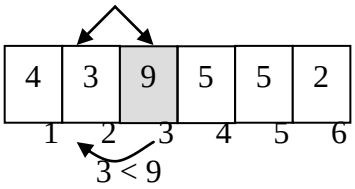


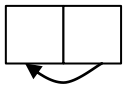
Просматриваем массив *с конца до начала*, меняя значениями элементы неупорядоченных пар ***соседних*** элементов. ***За первый проход максимальный элемент всплывет в самое начало***. Но не только он: если бы последние два элемента были не упорядочены, то 5 бы тоже продвинулась на шаг:

1-й проход по массиву
z=1

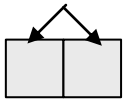


5 < 9, меняем местами и продолжаем обход

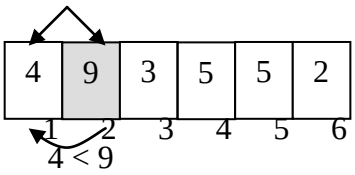


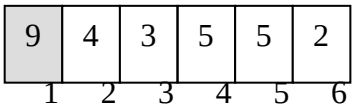


– сравнение пары соседних элементов



– неупорядоченная пара элементов; меняем их значениями.

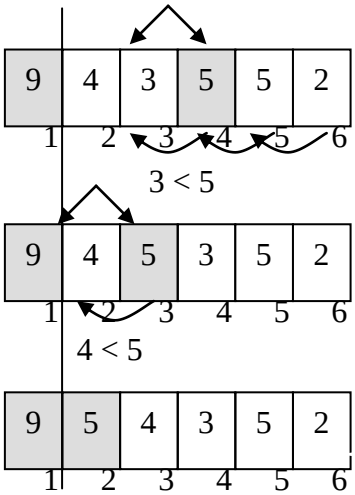




Максиму встал на своё место – «***максимум в начало***».
При первом обходе ***были*** неупорядоченные пары.

2-й проход (первый элемент встал на место, и его больше не сравниваем)

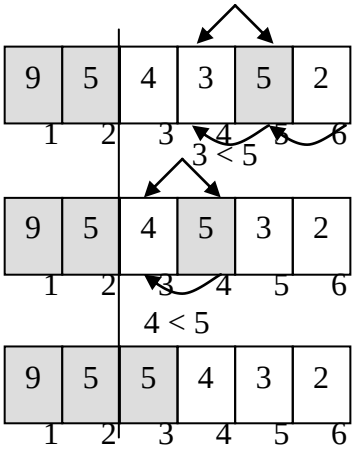
z=2



На данном шаге **были** обмены (неупорядоченные пары).

3-й проход (первые два элемента встали на место, и их больше не сравниваем)

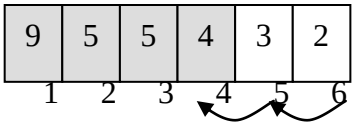
z=3



На данном шаге **были** обмены (неупорядоченные пары).

4-й проход (первые три элемента встали на место, и их больше не сравниваем)

z=4



На данном шаге обменов (неупорядоченных пар) **не было**.
Массив упорядочен.

Выполнение Контрольной работы №1 (1 ак.час)