**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ**

(**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**)» (**МАИ)**

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра 304 Вычислительные машины, системы и сети

СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ **№ 3**

**“Алгоритмы сортировки данных”**

Группа: *М30-209Б-19*

Выполнил:

*Кузнецов И.И*

Проверил:

*Ким Роман Валерьевич*

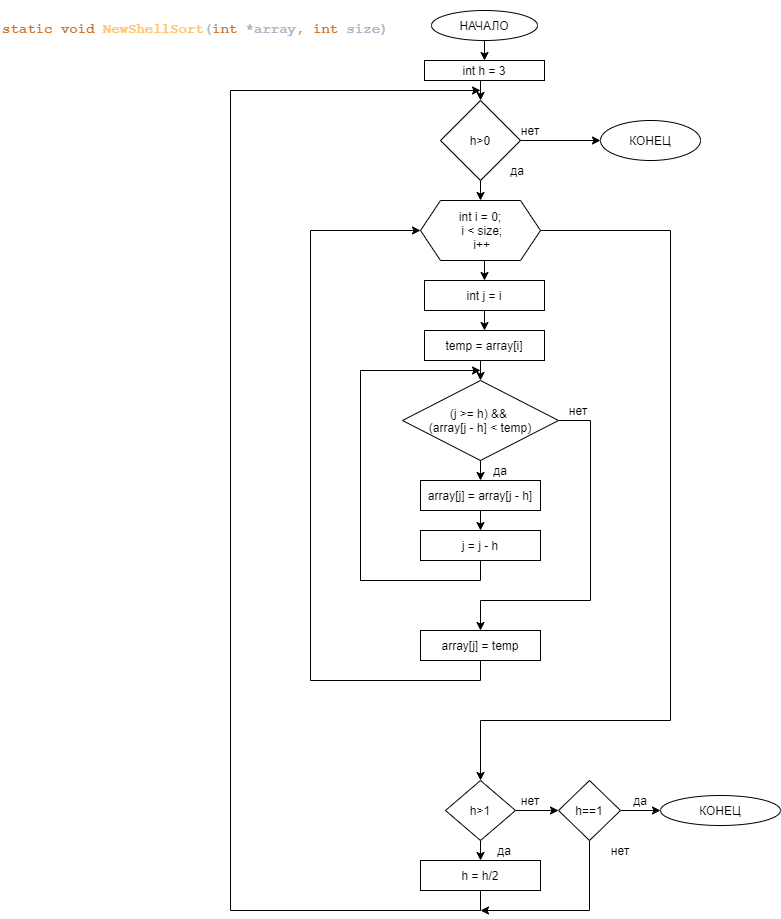
**Москва 2020**

**Описание сортировки Шелла**

Сортировка Шелла является улучшенной версией сортировки вставками:

Вначале сортируются элементы, находящиеся друг от друга на определенном расстоянии, после этого расстояние постепенно уменьшается, и в конце мы получаем массив в котором элементы которые нужно обменять будут стоять достаточно близко друг от друга.

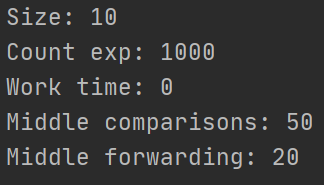
**Структурная схема алгоритма**

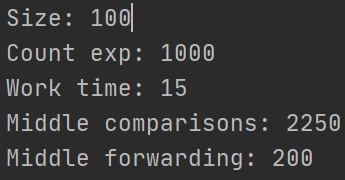


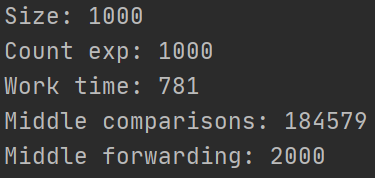
**Код алгоритма**

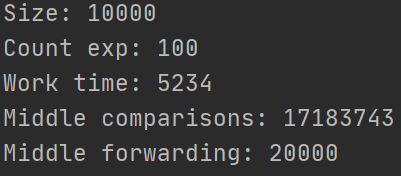
**int h = 3; // начальное приращение сортировки  
while (h > 0) { // пока существует приращение  
 for (int i = 0; i < size; i++) { // для всех элементов массива  
 int j = i; // сохраняем индекс и элемент  
 int temp = array[i];  
 // просматриваем остальные элементы массива, отстоящие от j-ого  
 // на величину приращения  
 while ((j >= h) && (array[j - h] < temp)) { // пока отстоящий элемент меньше текущего  
 array[j] = array[j - h]; // перемещаем его на текущую позицию  
 j = j - h; // переходим к следующему отстоящему элементу  
 }  
 array[j] = temp; // на выявленное место помещаем сохранённый элемент  
 }  
 if (h > 1) {  
 h = h / 2; // делим приращение на 2  
 } else if (h == 1) { // последний проход завершён,  
 break; // выходим из цикла  
 }**

**Результаты работы программы**

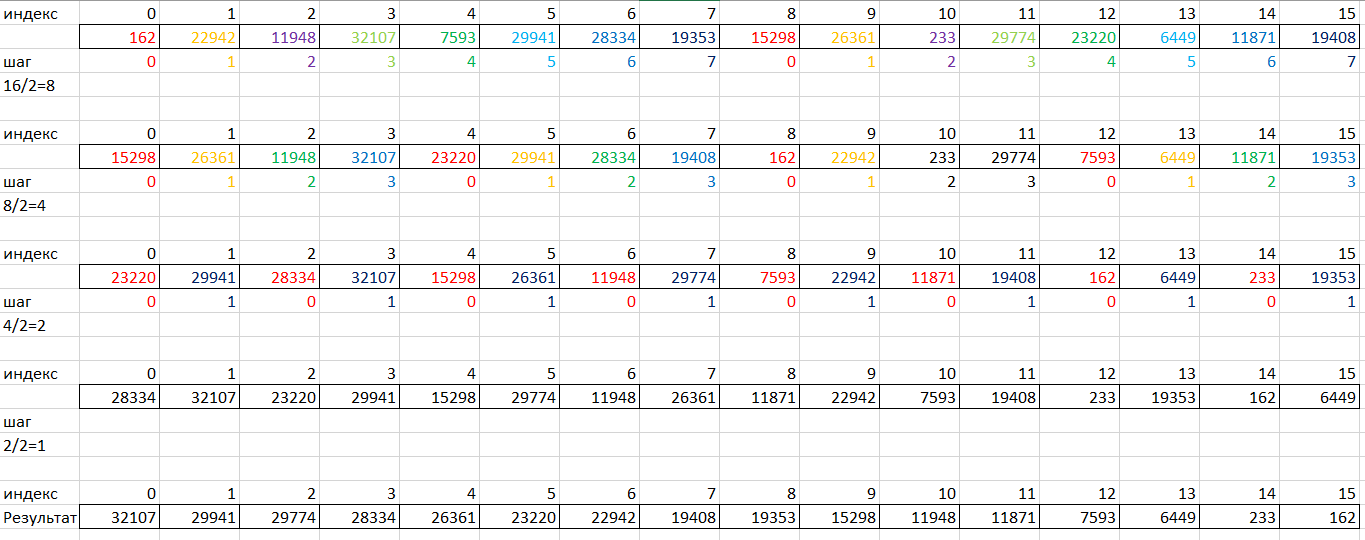








**Пример сортировки с массивом на 16 элементов**

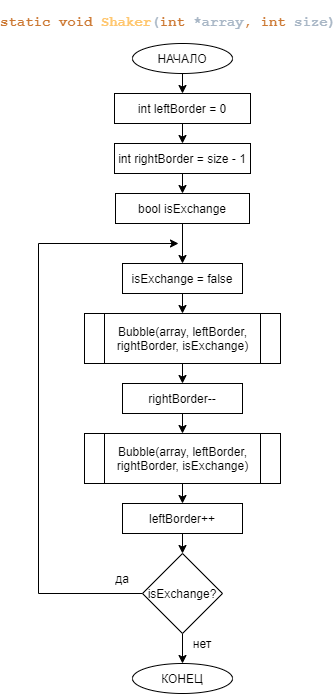


**Описание сортировки шейкера**

Сортировка шейкером, это улучшенная версия сортировки пузырьком:

Движемся от начала массива к концу, в мин(макс) элемент “всплывает” на конец массива, и соответственно наоборот, после мы уменьшаем границы, и сортируем по такому же принципу, таким образом расставляя элементы с краев, мы сортируем центр, и если при очередной проходе, никаких перестановок не осуществлялось => массив отсортирован.

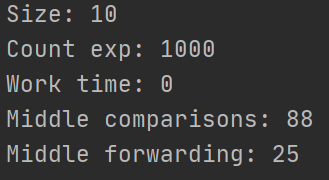
**Структурная схема алгоритма**

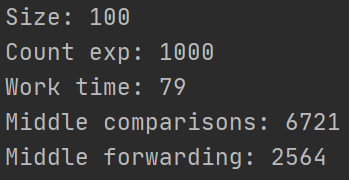
****

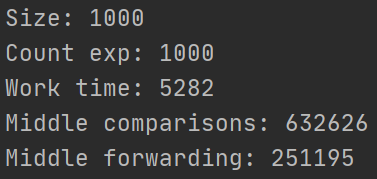
**Код алгоритма**

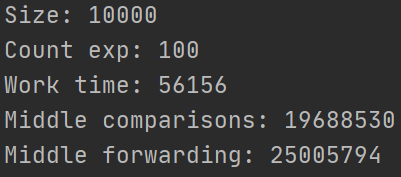
**static void Shaker(int \*array, int size, unsigned int &comparisons, unsigned int &forwarding) {  
 int leftBorder = 0; //левая граница  
 int rightBorder = size - 1; //правая граница  
 bool isExchange; //отсортирован ли массив?  
 do {  
 isExchange = false;  
 Bubble(array, leftBorder, rightBorder, isExchange, comparisons, forwarding);  
 rightBorder--;  
 Bubble(array, rightBorder, leftBorder, isExchange, comparisons, forwarding);  
 leftBorder++;  
 } while (isExchange);  
}**

**Результаты работы программы**

****

****

****

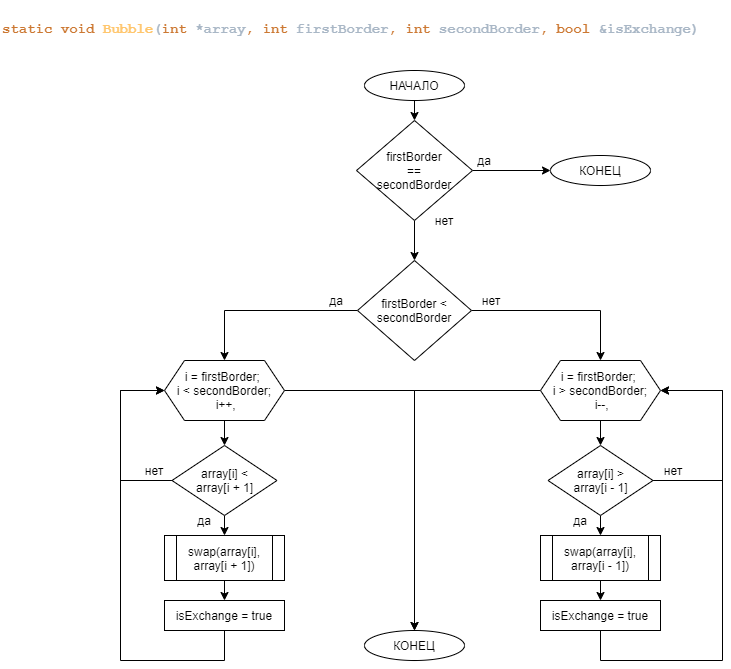
****

**Пример сортировки с массивом на 16 элементов**

****

**Описание вспомогательной функции для сортировки Шейкером (пузырьковая)**

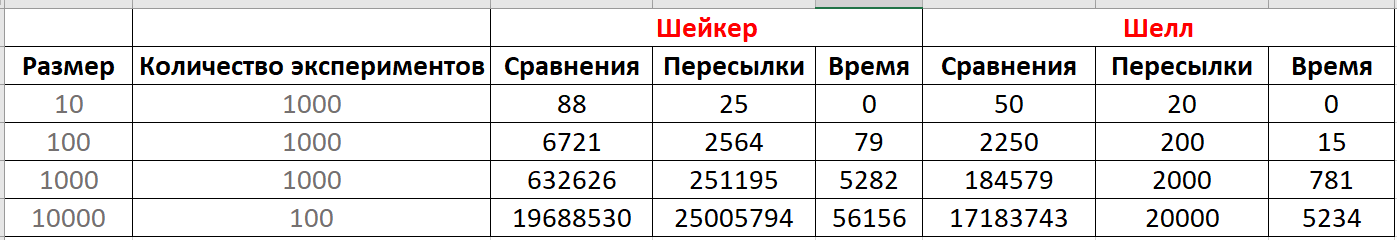
**Структурная схема алгоритма**

****

**Код алгоритма**

**static void Bubble(int \*array, int firstBorder, int secondBorder, bool &isExchanged) {  
  
 if (firstBorder == secondBorder) return; //если сомкнулись границы  
  
 if (firstBorder < secondBorder) { //сортировка слева направо  
 for (int i = firstBorder; i < secondBorder; i++, comparisons++) {  
 if (array[i] < array[i + 1]) {  
 swap(array[i], array[i + 1]);  
 isExchange = true;  
 }  
 }  
 } else {  
 for (int i = firstBorder; i > secondBorder; i--, comparisons++) { //сортировка справа налево  
 if (array[i] > array[i - 1]) {  
 swap(array[i], array[i - 1]);  
 isExchange = true;  
 }  
 }  
 }  
}**

**Сопоставительный анализ**



Таким образом, мы изучили сортировки Шейкера и Шелла, и проведя эксперименты выяснили, что сортировка Шелла работает быстрее, за счет меньшего количество обмена и сравнений и за счет самой идеи алгоритма.

**Вывод:**

Мы ознакомились с сортировкой Шелла и сортировкой Шейкером, рассмотрели на маленьком массиве каким образом работают данные сортировки, посмотрели на скорость работы, количество пересылок и сравнений и провели сопоставительный анализ.