**Лабораторная работа №3 - ЦИКЛЫ И МАССИВЫ**

**Часть 2 – Сортировка массивов**

Сведения из теории

В программе может возникнуть необходимость повторять в цикле некоторую группу команд (тело цикла). Однако, количество повторений не является фиксированным и зависит от результатов выполнения тела цикла.

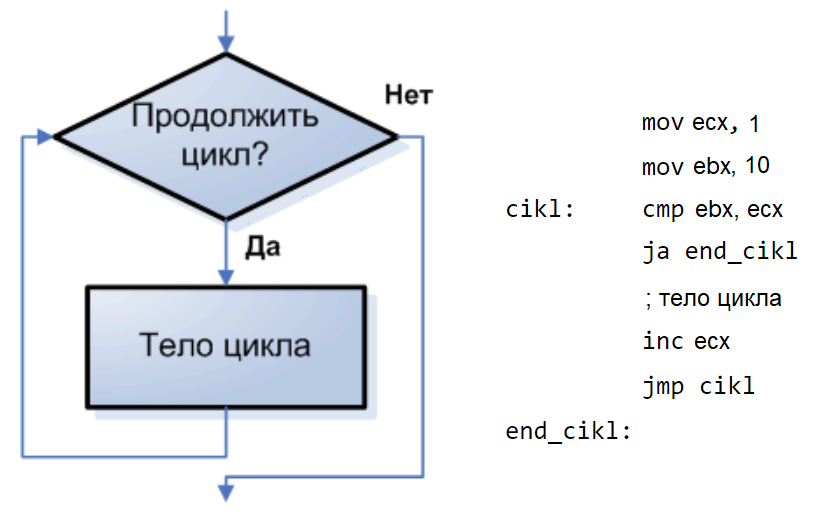


Рисунок 1 – Цикл с предусловием

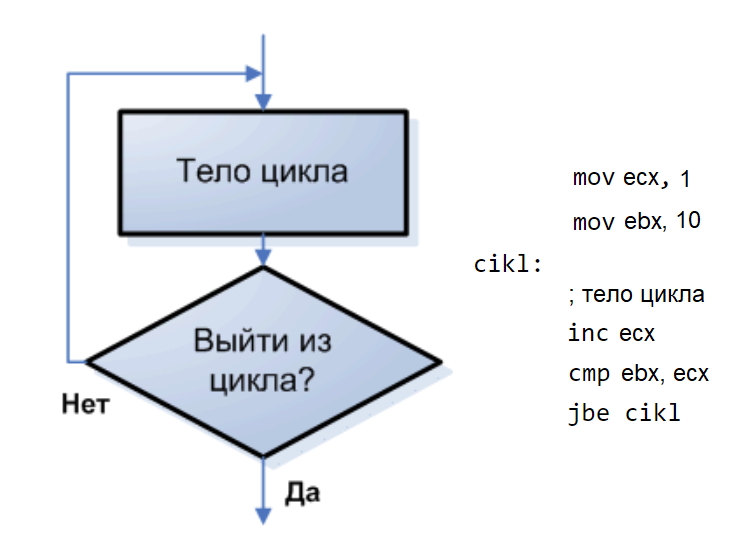


Рисунок 2 – Цикл с постусловием

Для организации таких циклов используются те же команды (сравнения, условных и безусловных переходов), что и при программировании разветвляющихся алгоритмов (см. ЛР №2 часть 1).

В качестве примера на рисунках показана реализация повторения тела цикла, пока значение в регистре ECX не превысит значение в регистре EBX

В данной работе предполагается использование таких циклов в задачах сортировки массивов. Существует несколько методов сортировки, отличающихся скоростью выполнения и сложностью алгоритма.

1. **Сортировка методом «пузырька**» (сортировка простыми обменами).

Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется перестановка 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).

Особенность данного алгоритма заключается в следующем: после первого завершения внутреннего цикла максимальный элемент массива всегда находится на N-ой позиции. При втором проходе, следующий по значению максимальный элемент находится на N-1 месте. И так далее. Таким образом, на каждом следующем проходе число обрабатываемых элементов уменьшается на 1 и нет необходимости «обходить» весь массив от начала до конца каждый раз.

Так как подмассив из одного элемента не нуждается в сортировке, то для сортировки требуется делать не более N-1 итераций внешнего цикла. Поэтому в некоторых реализациях внешний цикл всегда выполняется ровно N-1 и не отслеживается, были или не были обмены на каждой итерации.

Введение индикатора (флажка F) действительно произошедших во внутреннем цикле обменов уменьшает число лишних проходов в случаях с частично отсортированными массивами на входе. Перед каждым проходом по внутреннему циклу флажок сбрасывается в 0, а после действительно произошедшего обмена устанавливается в 1. Если после выхода из внутреннего цикла флажок равен 0, то обменов не было, то есть массив отсортирован и можно досрочно выйти из программы сортировки.

1. **Сортировка перемешиванием.**

Этот алгоритм является разновидностью пузырьковой сортировки. Также этот алгоритм называют «шейкерной» сортировкой или двунаправленной сортировкой. Основное отличие от обычной сортировки пузырьком в том, что массив сначала просматривается слева направо и максимальный элемент перемещается вправо, а после осуществляется проход по массиву справа налево (от последнего отсортированного элемента) и наименьший элемент перемещается влево.

Границы рабочей части массива (то есть части массива, где происходит движение) устанавливаются в месте последнего обмена на каждой итерации. Массив просматривается поочередно справа налево и слева направо.

1. **Сортировка «расческой».**

Сортировка расчёской улучшает сортировку пузырьком.

В сортировке пузырьком, когда сравниваются два элемента, промежуток (расстояние друг от друга) равен 1. Основная идея «расчёски» в том, чтобы первоначально брать достаточно большое расстояние между сравниваемыми элементами и по мере упорядочивания массива сужать это расстояние вплоть до минимального. Таким образом, мы как бы причёсываем массив, постепенно разглаживая на всё более аккуратные пряди. Сначала расстояние между элементами максимально, то есть равно размеру массива минус один. Затем, пройдя массив с этим шагом, необходимо поделить шаг на фактор уменьшения A=1,247 и пройти по списку вновь. Так продолжается до тех пор, пока разность индексов не достигнет единицы. В этом случае сравниваются соседние элементы, как и в сортировке пузырьком, но такая итерация одна.

1. **Сортировка выбором.**

Суть алгоритма заключается в поиске минимального значения (максимального для сортировки по убыванию) и обмене найденного значения с первым неотсортированным значением. Шаги алгоритма:

* находим номер минимального значения в текущем массиве
* производим обмен этого значения со значением первой неотсортированной позиции (обмен не нужен, если минимальный элемент уже находится на данной позиции)
* сортируем оставшуюся часть массива, исключив из рассмотрения уже отсортированные элементы.

1. **Комбинированный метод «пузырёк-выбор».**

В основе алгоритм метода «пузырька», но при каждом проходе также находится минимальный (или максимальный при сортировке по убыванию) элемент и меняется с первым неотсортированным элементом слева.

1. **Сортировка вставками.**

Суть метода заключается в следующем: есть часть массива, которая уже отсортирована, и требуется вставить остальные элементы массива в отсортированную часть, сохранив при этом упорядоченность.

Так как в начальный момент отсортированная часть отсутствует, то в качестве таковой может быть выбран первый элемент массива (т.е. отсортированная часть содержит 1 элемент).

Далее на каждом шаге алгоритма мы выбираем один из элементов входных данных (с целью получения устойчивого алгоритма сортировки элементы вставляются по порядку их появления в неотсортированной части массива) и вставляем его на нужную позицию в уже отсортированной части массива, до тех пор, пока весь набор входных данных не будет отсортирован.

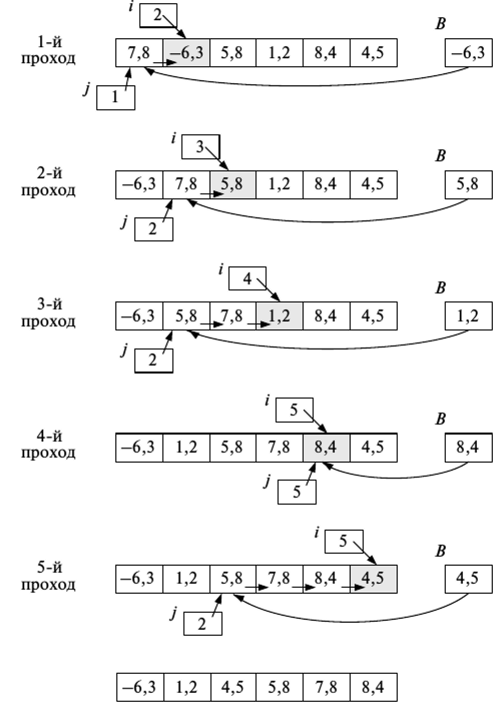


Рисунок 3 – Сортировка вставками

Для нахождения нужного места для вставки при сортировке по возрастанию выбранный «неотсортированный» элемент сравнивается с крайним правым отсортированным элементом. Если выбранный элемент оказывается больше этого элемента, то он сохраняется на своей позиции. Если он оказывается меньше, то сравнивается с предпоследним и так далее (т.е. проверяется в направлении справа-налево). Вставка происходит правее первого же «отсортированного» элемента, который окажется меньше выбранного «неотсортированного» элемента. Если таковых в отсортированной части не найдется вообще, то «неотсортированный» элемент вставляется на первую позицию. В любом случае, после вставки этот элемент становиться «отсортированным», т.е. элементом отсортированной части массива.

1. **Сортировка Шелла.**

Данный метод является усовершенствованным методом вставок. Идея ускорения вставок схожа с идеей метода «расчески». При сортировке Шелла сначала сортируются «подсписки», состоящие из элементов, стоящих один от другого на некотором расстоянии d. После этого процедура повторяется для некоторых меньших значений d, а завершается сортировка Шелла упорядочиванием элементов при d=1 (то есть обычной сортировкой вставками всего полученного на предыдущих шагах массива).

Данный процесс иллюстрируется примером:



Среднее время работы алгоритма зависит от длин промежутков d, на которых будут находиться сортируемые элементы исходного массива ёмкостью N на каждом шаге алгоритма. Существует несколько подходов к выбору этих значений. В оригинальном методе Шеллом использовался подход уменьшения длины промежутка вдвое:



1. **Гномья сортировка.**

Алгоритм сортировки похож на сортировку вставками, но в отличие от последней перед вставкой на нужное место происходит серия обменов, как в сортировке пузырьком. Название происходит от предполагаемого поведения садовых гномов при сортировке линии садовых горшков.

*«Гномья сортировка основана на технике, используемой обычным голландским садовым гномом (нидерл. tuinkabouter). Это метод, которым садовый гном сортирует линию цветочных горшков. По существу он смотрит на текущий и предыдущий садовые горшки: если они в правильном порядке, он шагает на один горшок вперёд, иначе он меняет их местами и шагает на один горшок назад. Граничные условия: если нет предыдущего горшка, он шагает вперёд; если нет следующего горшка, он закончил.»*

*Дик Грун*

Алгоритм концептуально простой, не требует вложенных циклов. Алгоритм находит первое место, где два соседних элемента стоят в неправильном порядке и меняет их местами. Он пользуется тем фактом, что обмен может породить новую пару, стоящую в неправильном порядке, только до или после переставленных элементов. Он не допускает, что элементы после текущей позиции отсортированы, таким образом, нужно только проверить позицию до переставленных элементов.

**9. Пирамидальная сортировка.**

Пирамидальная сортировка или как еще ее называют "Сортировка кучей" использует в своем алгоритме двоичное дерево – это такое дерево, для которого выполнены следующие условия:

* значение в любой вершине не меньше, чем значения её потомков;
* длина веток дерева не отличается друг от друга более чем на 1 слой;
* последний слой заполняется слева направо без «дырок».

Поскольку двоичная куча — это законченное двоичное дерево, ее можно легко представить в виде массива, а представление на основе массива является эффективным с точки зрения расхода памяти. Если родительский узел (вершина кучи или корень дерева) хранится в массиве под индексом К, то индекс левого дочернего элемента может быть вычислен как 2 K, а индекс правого дочернего элемента — как 2 K + 1 (при условии, что индексирование начинается с 0).

Таким образом, если предположить, что регистр ESI содержит индекс узла, то индекс его родительского узла может быть вычислен сдвигом вправо (т.е. делением на 2):

**Mov EAX, ESI ; ESI содержит индекс узда**

**Sar EAX, 1 ; в EAX - индекс родительского узла**

Индекс левого дочернего узла будет получен сдвигом влево:

**Mov EAX, ESI ; ESI содержит индекс узда**

**Sal EAX, 1 ; в EAX - индекс левого дочернего узла**

Индекс правого дочернего узла равен индексу левого +1:

**Mov EAX, ESI ; ESI содержит индекс узда**

**Sal EAX, 1**

**Inc EAX ; в EAX - индекс правого дочернего узла**

У узла с индексом 0 родительского узла не существует. Если при вычислении индекса дочернего узла получили число, большее, чем длина дерева (общего количества узлов, из которых состоит дерево), значит такой дочерний узел в дереве отсутствует.

Так как в процессе сортировки размер дерева на каждой итерации уменьшается, то целесообразно иметь две независимые переменные – MassSize и TreeSize для указания размера исходного массива и текущего размера дерева (на первом шаге TreeSize = MassSize).

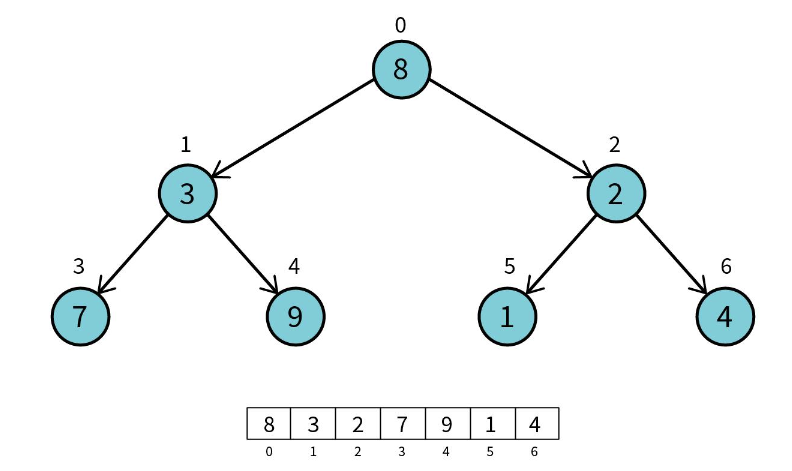


Рисунок 4 – Неотсортированный массив в виде дерева

Алгоритм пирамидальной сортировки состоит из следующих шагов:

1) Построение исходного массива в виде дерева (рис. 4).

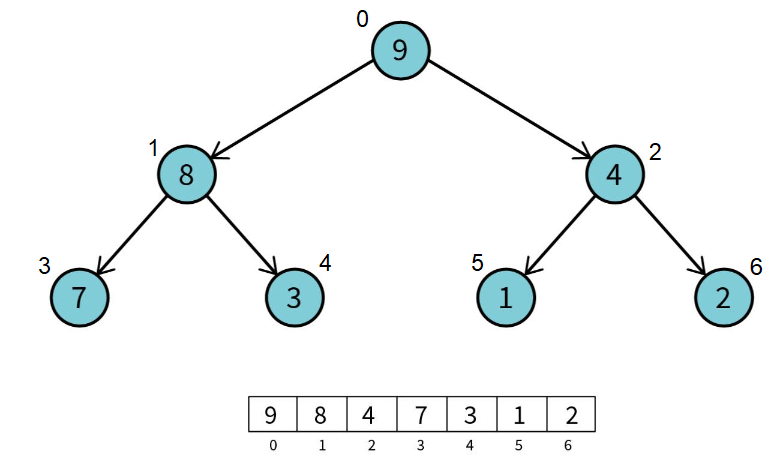
2) Проверка всех узлов, начиная с предпоследнего ряда, на выполнение правила двоичного дерева. При нарушении правила наибольший из потомков меняется местами с родительским узлом. При наличии у вышеупомянутого потомка своего дочернего поддерева, оно повторно перепроверяется. В результате получится исправленное дерево, в корне которого – элемент с максимальным значением (рис. 5).

Рисунок 5 – Исправленное двоичное дерево

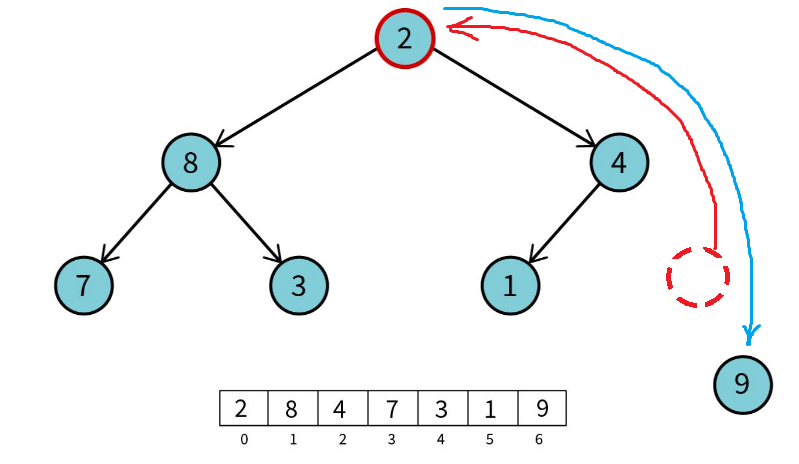
3) Исключение корня дерева (максимального значения массива) из массива и перенос его в конец последовательности.

Рисунок 6 – Исключение корня

4) Возврат на шаг 2 с учетом, что размер подлежащего проверке дерева (длина неотсортированной части массива) уменьшилась на единицу.

Таким образом, в конце массива складываются уже отсортированные элементы.

Итерации выполняются до тех пор, пока массив не будет полностью отсортирован, т.е. после очередного исключения корня дерева не останется только один элемент.

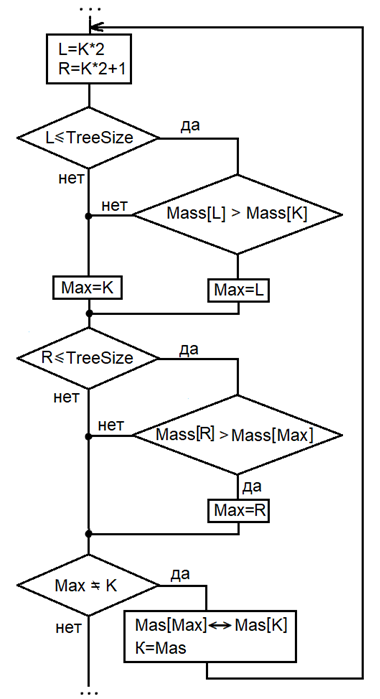


Рисунок 6 – Упорядочивание дерева от узла К

Как следует из описания метода, ключевым элементом алгоритма сортировки является выстраивание упорядоченного дерева, начиная с узла с определенным номером.

Схема соответствующего фрагмента алгоритма (для дерева размером TreeSize и текущего узла К) будет иметь вид, представленный на рис. 6.

Данный мини-алгоритм (рис. 6) будет использоваться:

- во-первых, при первоначальном упорядочивании дерева (для этого надо последовательно пройти по всем узлам, имеющим дочерние узлы, начиная с предпоследнего ряда древовидной структуры;

- во-вторых, для восстановления упорядоченности дерева при исключении корневого узла.

Таким образом, общий алгоритм будет выглядеть так, как показано на рис. 6.

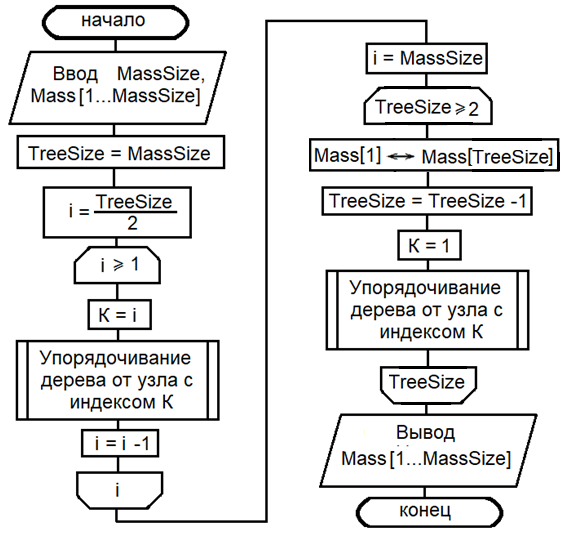


Рисунок 7 – Алгоритм пирамидальной сортировки

**10. Быстрая сортировка (сортировка Хоара).**

Алгоритм быстрой сортировки - один из самых быстрых и эффективных и часто используется в практике. При этом он достаточно простой.

Суть алгоритма в следующем:

1. Выбрать из массива опорный элемент. Например, взять элемент в середине массива (в целом это может быть любой из элементов).
2. Сравнить остальные элементы массива с выбранным опорным элементов и разбить массив на 2 части:

* элементы, которые меньше или равны опорному элементу;
* элементы, которые больше опорного.

1. Далее пункты 1 и 2 повторяются рекурсивно для каждой части массива, до тех пор, пока размер части состоит из более чем 1 элемента.

**11. Сортировка слиянием.**

Суть алгоритма сортировки слиянием состоит в том, чтобы разбить исходный массив на более мелкие массивы, отсортировать каждый по отдельности, а после объединить результаты.

*Алгоритм*

1. *Сортируемый массив разбивается на две части примерно одинакового размера;*
2. *Каждая из получившихся частей сортируется отдельно, например — тем же самым алгоритмом;*

*Шаги 1 и 2 представляют собой рекурсивное разбиение массива на меньшие, которое происходит до тех пор, пока размер массива не достигнет единицы (любой массив длины 1 можно считать упорядоченным).*

*3) Два упорядоченных массива половинного размера соединяются в один.*

Основную идею слияния двух отсортированных массивов можно объяснить на следующем примере. Пусть имеются два уже отсортированных по возрастанию подмассива. Тогда на каждом шаге берется меньший из двух первых элементов подмассивов и записывается в результирующий массив. Счётчики номеров элементов результирующего массива и подмассива, из которого был взят элемент, увеличиваем на 1.

Когда один из подмассивов закончился, добавляются все оставшиеся элементы второго подмассива в результирующий массив.

Например, массив содержит числа 5 2 1 3 4. Разбиваем его на две части: 5,2,1 и 3,4. Первую часть 5,2,1 разбиваем еще на две части 5,2 и 1. Далее 5,2 еще на две части 5 и 2. А теперь идем обратно, сортируем и сливаем массивы. Получается 2,5 и 1, объединим дальше - 1,2,5, последняя итерация отсортирует исходный массив 1 2 3 4 5. При слиянии учитывается тот факт, что массивы уже отсортированы по отдельности, поэтому объединение проходит быстрее.

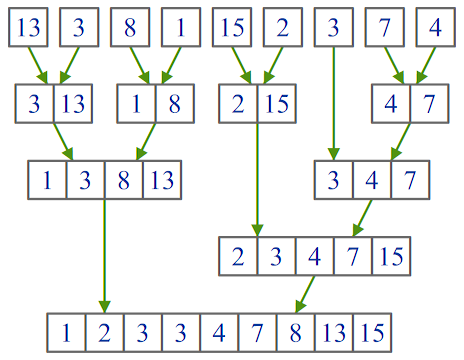
Ещё один пример представлен на рисунке.

Рисунок 8 – Сортировка слиянием

\*\*\*

Сортировка методом слияния и быстрая сортировка предполагают использования в программе рекурсивных процедур (процедур, код которых содержит вызов самой себя). В рамках данной работы эти методы не используются.

Более простые методы сортировки также предполагают использование повторяющихся действий, но без рекурсии, с заранее неизвестным числом повторений. Они могут быть реализованы с помощью циклов с постусловием или предусловием.

*Задание*

Составить схему алгоритма фрагмента программы, отвечающего за сортировку методом, который соответствует варианту из табл. 1.

Написать программу, в которой:

- осуществляется поэлементный ввод неупорядоченного массива из 20 элементов;

- вывод данного массива в одну строку (элементы должны быть разделены минимум одним пробелом);

- сортировка массива в соответствии со схемой алгоритма;

- вывод отсортированного массива в одну строку (элементы должны быть разделены минимум одним пробелом).

Таблица 1 – Варианты заданий

|  |  |  |  |
| --- | --- | --- | --- |
| № по списку | Задание | № по списку | Задание |
| 1 | Сортировка пузырьком по возрастанию | 9 | Пирамидальная сортировка по возрастанию |
| 2 | Сортировка перемешиванием по возрастанию | 10 | Сортировка пузырьком по убыванию |
| 3 | Сортировка расческой по возрастанию | 11 | Сортировка перемешиванием по убыванию |
| 4 | Сортировка выбором по возрастанию | 12 | Сортировка выбором по убыванию |
| 5 | Сортировка «пузырек-выбор» по возрастанию | 13 | Сортировка «пузырек-выбор» по убыванию |
| 6 | Сортировка вставками по возрастанию | 14 | Сортировка вставками по убыванию |
| 7 | Сортировка Шелла по возрастанию | 15 | Сортировка Шелла по убыванию |
| 8 | Гномья сортировка по возрастанию | 16 | Сортировка расческой по убыванию |

Если номер по списку группы N>16, то номер варианта равен N-15.