**Ход выполнения работы**

Учебно-методические материалы содержат 20 вариантов заданий на реализацию алгоритмов сортировки массивов, поиска элементов и работы с множествами чисел.

Для выполнения предложенных заданий учебной практики необходимы навыки, полученные в ходе изучения курса «Программирование», а также базовые знания по алгоритмизации. Для реализации программ допустимо использовать любые доступные языки программирования, за исключением случаев использования готовых или стандартных библиотек. Предпочтительным является использование языка Си.

Некоторые варианты заданий требуют дополнительного изучения алгоритмов представления и обработки данных в графах (деревьях) — такие варианты стоит рассматривать как задания повышенной сложности.

В качестве дополнительного фактора сложности может быть предложено выполнение реализации программы с графическим интерфейсом пользователя, позволяющим отображать входные и выходные данные программы.

При проверке качества реализации программы необходимо учитывать корректность реализации основного алгоритма, а также реакцию программы на «граничные» наборы данных, при которых возможно выявление ошибок реализации основного алгоритма или кода ввода и вывода данных.

**Задание на учебную практику**

Необходимо реализовать и продемонстрировать работу алгоритмов сортировки или поиска элементов из списка, представленного ниже. Конечная программа должна реализовывать непосредственно алгоритм, а также выполнять считывание входных данных из файла и запись результатов в файл и отображать все данные в графическом интерфейсе.

Форматы входных и выходных файлов определяются произвольно, для массивов желательно использование формата CSV (значения разделены запятой). Программа должна обрабатывать данные, предоставленные в корректно сформированных входных файлах. Если данные в файле некорректны, то программа должна выводить соответствующее сообщение.

Конечные программы не должны использовать реализации алгоритмов сортировки или поиска элементов, предоставляемые стандартными или сторонними библиотеками, включая системные.

В случае наличия реализации требуемого алгоритма в стандартных библиотеках языка программирования выполнить сравнение производительности реализованного алгоритма со стандартным. Сделать выводы.

**Отчет должен содержать следующие элементы:**

* титульный лист;
* индивидуальный план;
* лист отчета с краткими итогами;
* отзыв руководителя;
* краткое описание алгоритма;
* блок-схема алгоритма;
* блок-схема программы;
* преимущества и недостатки алгоритма;
* типичные сценарии применения;
* непосредственная реализация алгоритма без использования стандартных или сторонних библиотек;
* **анализ работы алгоритма, включая количество базовых операций (для алгоритмов сортировки — число перестановок), производительность для наилучшего и наихудшего сценариев в секундах (для алгоритмов сортировки — предварительно отсортированный, инвертированный и случайный наборы данных, для алгоритмов поиска — поиск элементов, заведомо находящихся в начале или конце набора данных, включая случай, когда набор данных состоит из многократно повторяющихся элементов).**
* анализ и реализация графического интерфейса;
* раздел тестирования готовой программы;
* анализ полученных результатов тестирования.

В качестве результатов выполнения работы предоставляются: **отчет с описанием реализованного алгоритма и сферы его применения, исполняемая программа, включая ее исходный код, наборы тестовых данных (в зависимости от варианта).**

**Анализ работы алгоритма должен выполняться на нескольких различных наборах данных.**

**Варианты заданий**

1. Сортировка пузырьком (bubble sort)
2. Шейкерная сортировка (cocktail shaker sort)
3. Сортировка выбором (selection sort)
4. Сортировка вставками (insertion sort)
5. Сортировка Шелла (Shell sort)
6. Быстрая сортировка (quicksort)
7. Двоичная сортировка (binary tree sort)
8. Сортировка слияниями (merge sort)
9. Поразрядная сортировка (radix sort)
10. Пирамидальная сортировка (heapsort)
11. Поиск в неупорядоченных таблицах
12. Поиск в упорядоченных таблицах
13. Поиск подстроки в строке: алгоритм Бауэра-Мура (Bauer-Moore)
14. Поиск подстроки в строке: алгоритм Кнута-Мориса-Пратта (Knuth-Morris-Pratt)
15. Линейный и бинарный поиск (linear/binary search algorithm)
16. Сформировать все возможные подмножества из k элементов заданного множества
17. Сформировать все возможные пары подмножеств при объединении которых получается исходное множество
18. Сформировать все возможные подмножества из заданного множества, элементы которых располагаются по возрастанию в коде Грея
19. Сформировать все возможные подмножества из заданного множества, элементы которых располагаются по возрастанию в лексикографическом представлении
20. Сформировать случайное подмножество из исходного методом подброса монетки

Визуализация работы некоторых из перечисленных алгоритмов может быть найдена на <https://sorting.at/>

**Краткие сведения об алгоритмах**

**Сортировка пузырьком (Bubble sort)**

Сортировка пузырьком, или сортировка простыми обменами — простейший алгоритм сортировки, выполняющий перестановку элементов в зависимости от результата их сравнения: выполняется N–1 проходов по массиву от его начала к концу, в каждом проходе выполняется попарное сравнение соседних элементов массива: при сортировке по возрастанию перестановка элементов выполняется, если a[i] > a[i+1], при сортировке по убыванию: a[i] < a[i+1].

Сортировка является устойчивой (перестановка элементов с одинаковыми значениями не выполняется).

Подробное описание алгоритма:

https://ru.wikipedia.org/wiki/Сортировка\_пузырьком

**Шейкерная сортировка (Coctail shaker sort)**

Шейкерная сортировка, или сортировка перемешиванием — разновидность сортировки пузырьком, при которой выполняется чередование направления прохода по элементам массива. После восходящего прохода правило сравнения изменяется на противоположное и выполняется нисходящий проход. На каждой итерации запоминается позиция крайней левой и крайней правой перестановки: эти позиции используются для определения диапазона очередного прохода по массиву. Алгоритм завершает работу по достижении единичного или нулевого диапазона (в зависимости от входного набора данных).

Подробное описание алгоритма:

https://ru.wikipedia.org/wiki/Сортировка\_перемешиванием

**Сортировка выбором (Selection sort)**

Данный алгоритм работает следующим образом: в неотсортированном подмножестве находится элемент, имеющий минимальное значение, и записывается в конец отсортированного подмножества.

В зависимости от реализации алгоритм может быть устойчивым или неустойчивым.

Подробное описание алгоритма:

https://ru.wikipedia.org/wiki/Сортировка\_выбором

**Сортировка вставками (Insertion sort)**

Имеется набор входных данных и пустая отсортированная последовательность. На каждом шаге алгоритма выбирается один из элементов входных данных и помещается на нужную позицию, увеличивая отсортированную последовательность до тех пор, пока набор входных данных не будет исчерпан. В отсортированной последовательности элементы упорядочены в любой момент времени.

Подробное описание алгоритма:

https://ru.wikipedia.org/wiki/Сортировка\_вставками

**Сортировка Шелла (Shell sort)**

Сортировка Шелла — алгоритм сортировки, являющийся усовершенствованным вариантом сортировки вставками. Вместо одного прохода мы делаем несколько, причем на i-ом проходе мы сортируем подмассивы из элементов, стоящих друг от друга на расстоянии di. Последний шаг должен являться обычной сортировкой вставками так как это гарантирует корректность алгоритма. Сортируем вставкой подгруппы элементов, но только в подгруппе они идут не в ряд, а равномерно выбираются с некоторой дельтой по индексу. Методично уменьшая расстояние между элементами этих несвязных подмножеств, сортировка проходит быстрее. Однако в данном методе нет строгой формулы, по которой строится оптимальный числовой ряд изменяющихся расстояний между элементами в подгруппах, он строится эмпирически.

Подробное описание алгоритма:

https://ru.wikipedia.org/wiki/Сортировка\_Шелла

**Быстрая сортировка (Quicksort)**

Является наиболее эффективным алгоритмом сортировки с помощью прямого обмена.

Выбирается опорный элемент массива. Элементы меньше опорного располагаются перед ним, больше опорного или равные — после. Данные операции рекурсивно применяются к подмассивам, но только если подмассив не единичный или не пустой.

Подробное описание алгоритма:

https://ru.wikipedia.org/wiki/Быстрая\_сортировка

**Двоичная сортировка (Binary tree sort)**

При выполнении алгоритма создается двоичное дерево ключей, для чего каждый ключ последовательно сравнивается с нижестоящими узлами. Если ключ меньше корня (родителя), то он располагается в левом поддереве; если равен или больше — в правом. Созданное дерево обходится по определенному правилу: сначала обходится левое поддерево и находится самый младший ключ, затем выводится корень, затем обходится правое поддерево и выводится младший ключ. Полученные ключи представляют упорядоченную последовательность.

Подробное описание алгоритма:

http://algolist.manual.ru/sort/faq/q7.php

**Сортировка слияниями (Merge sort)**

Входная последовательность делится на подмассивы (например, единичной длины. Затем сравнивается первые элементы двух подмассивов. Наименьший элемент из двух записывается в будущий массив, состоящий из двух подмассивов, а на место записанного наименьшего назначается следующий элемент и снова сравнивается с элементом из второго подмассива.

Подробное описание алгоритма:

https://ru.wikipedia.org/wiki/Сортировка\_с\_помощью\_двоичного\_дерева

**Поразрядная сортировка (Radix sort)**

Возможна сортировка по младшим (неустойчивая) или старшим (устойчивая) разрядам.

Элементы перебираются по порядку и группируются по самому младшему разряду (сначала все, заканчивающиеся на 0, затем заканчивающиеся на 1, ..., заканчивающиеся на 9). Возникает новая последовательность. Затем группируются по следующему разряду с конца, затем по следующему и т.д. пока не будут перебраны все разряды, от младших к старшим.

Подробное описание алгоритма:

http://algolab.valemak.com/radix

**Пирамидальная сортировка (Heapsort)**

Создается двоичное сортирующее дерево, в котором значение родительских элементов равно или больше значения потомков.

Для создания сортирующего дерева выбирается потомок и сравнивается с родителем. Если значение потомка больше родительского, то они меняются местами. После некоторого числа итераций в корне дерева будет ключ с максимальным значением.

Далее максимум обменивается с последним ключом неотсортированного подмассива. Структура перестанет быть сортирующим деревом, однако его неотсортированная часть станет меньше на один узел. Неотсортированная часть снова преобразуется её в сортирующее дерево с последующей перестановкой найденного максимума в конец. И так до тех пор, пока неотсортированная часть не уменьшится до единственного элемента.

Подробное описание алгоритма:

https://habr.com/ru/post/221095/

**Поиск в неупорядоченных таблицах**

В неупорядоченных таблицах для поиска элемента с заданным ключом используется алгоритм линейного поиска. Линейный поиск выполняет поиск элемента или значения из некоторого множества до тех пор, пока нужный элемент или значение не будет найден, порядок поиска последовательный.

Сравнивается искомый элемент со всеми элементами на некотором множестве, если искомый элемент будет найден, то функция возвращает индекс найденного элемента, в противном случае функция возвращает -1.

Подробное описание алгоритма:

https://nauchforum.ru/studconf/tech/xlviii/25030

**Поиск в упорядоченных таблицах**

В упорядоченных таблицах для поиска элемента с заданным ключом используется алгоритм бинарного поиска. Бинарный поиск делит диапазон значений на половину и продолжает сужать поле поиска до тех пор, пока искомый элемент не будет найден. Это классический пример алгоритма «разделяй и властвуй».

Подробное описание алгоритма:

https://nauchforum.ru/studconf/tech/xlviii/25030

**Поиск подстроки в строке: алгоритм Бауэра-Мура (Bauer-Moore)**

Алгоритма Бойера-Мура состоит из двух этапов. На первом этапе строится таблица смещений для искомой строки. Далее совмещается начало строки и образца и начинается проверка с последнего символа образца. Если последний символ образца и соответствующий ему символ строки не совпадают, то образец сдвигается относительно строки вправо на величину, полученную из таблицы смещений, и снова проводится сравнение, начиная с последнего символа образца. В противном случае, при совпадении символов, проводится сравнение предпоследнего символа образца и т.д. Если все символы образца совпали с соответствующими символами строки, то нужная подстрока найдена. Если же какой-то (не последний) символ образца не совпадает с соответствующим символом строки, то образец сдвигается на один символ вправо и снова начинается проверка с последнего символа.

Подробное описание алгоритма:

http://cybern.ru/algoritm-bojera-mura.html

**Поиск подстроки в строке: алгоритм Кнута-Мориса-Пратта (Knuth-Morris-Pratt)**

Алгоритм работает следующим образом: в строке последовательно сравниваются символы с образцом. Если подстрока не найдена, начало сравнения перемещается на один шаг и снова сравнивается до тех пор, пока не будет найден образец или не будет достигнут конец строки.

Подробное описание алгоритма:

<https://habr.com/ru/post/307220/>

**Линейный и бинарный поиск (Linear/binary search algorithm)**

Алгоритмы линейного и бинарного поиска используются для поиска элемента с указанным значением в массиве данных. Линейный алгоритм использует примитивный подход, при котором выполняется последовательный проход по всем элементам массив, при котором выполняется сравнение текущего элемента с искомым. Исходный массив данных может быть неотсортированным.

Бинарный поиск выполняет поиск элемента с указанным значением в *отсортированном* массиве. В ходе выполнения поиска диапазон поиска делится пополам и, в зависимости от значения элемента, находящегося в середине очередного диапазона, выбирается следующий поддиапазон.

**Сформировать все возможные подмножества из k элементов заданного множества**

Для того, чтобы сформировать все возможные подмножества необходим перебор всех подмножеств. Однако, для начала необходимо найти минимальное подмножество, то есть состоящее как можно из меньшего числа элементов (либо максимальное подмножество). Тогда эффективнее всего организовать перебор так, чтобы сначала проверялись все подмножества, состоящие из одного элемента, затем из двух, трех и т.д. элементов (для максимального подмножества — в обратном порядке). Таким образом, все подмножества, удовлетворяющие условию задачи, будут являться подмножествами заданного множества.

**Сформировать все возможные пары подмножеств при объединении которых получается исходное множество**

Для того, чтобы сформировать все возможные подмножества необходим перебор всех подмножеств. Тогда для начала необходимо найти первый элемент пары, который будет являться минимальным подмножеством, чаще всего состоящее из одного элемента. Тогда второй элемент пары будет являться всей остальной цепочной элементов. Далее необходимо сформировать следующую пару, состоящую из двух элементов и всей остальной цепочки и так далее. Таким образом, все пары подмножеств, удовлетворяющие условию задачи, будут являться парами подмножеств заданного множества.

**Сформировать все возможные подмножества из заданного множества, элементы которых располагаются по возрастанию в коде Грея**

Двоичным кодом Грея порядка n называется последовательность всех 2^n n-битных кодов, в которой любые два соседних кода различаются ровно в одном разряде. Для начала кодируем первые 2 числа "0" и "1". Тогда для следующих 2-х чисел нам необходимо отразить коды и дописать их спереди. Такую операцию проделываем для каждого элемента последовательности. Для возрастающей последовательности n-битных кодов следует действовать следующим образом:

- дописать спереди 0 к возрастающей (n-1)-битной последовательности

- дописать спереди 1 к убывающей (n-1)-битной последовательности

Подробное описание алгоритма:

https://habr.com/ru/post/200806/

**Сформировать все возможные подмножества из заданного множества, элементы которых располагаются по возрастанию в лексикографическом представлении**

Перестановки сравниваются слева направо поэлементно. На некотором шаге работы алгоритма получена перестановка. Для того чтобы определить непосредственно следующую за ней перестановку, необходимо, пересматривая данную перестановку справа налево, следить за тем, чтобы каждое следующее число было больше предыдущего, и остановиться сразу же, как только это правило нарушится.

Подробное описание алгоритма:

http://algolist.manual.ru/olimp/per\_sol.php

**Сформировать случайное подмножество из исходного методом подброса монетки**

Метод подброса монетки состоит в том, что из исходной последовательности выбираются только те элементы, для которых подброс монетки выдал ‘1’, а все элементы с ‘0’ не учитываются.

**Требования к оформлению отчета**

Отчеты представляются в виде одного файла в формате Microsoft Word.

**Требования к основному тексту.**

При оформлении основного текста работы следует соблюдать следующие параметры:

* шрифт «TimesNewRoman»;
* размер шрифта – 14;
* межстрочный интервал 1,5;
* в параметрах абзаца:

1. отступ слева – 0 см;
2. отступ справа – 0 см;
3. интервал перед – 0 пт;
4. интервал после 0 пт.

При использовании для оформления пояснительной записки текстового редактора MicrosoftOfficeWord в настройках параметров страницы рекомендуется выставить следующие значения полей: левое – 3 см; верхнее, нижнее и правое – 1,5 см. Отступ красной строки (отступ первой строки), должен составлять 1,25 см.

Все заголовки не имеют абзацного отступа (красной строки), выравниваются по центру и оформляются полужирным шрифтом «TimesNewRoman» размер шрифта – 14, межстрочный интервал 1,5, в параметрах абзаца отступ слева – 0 см, отступ справа – 0 см, интервал перед – 10 пт, интервал после 10 пт.

Все страницы работы должны быть пронумерованы последовательно арабскими цифрами. Нумерация страниц должна быть сквозной от титульного листа (на титульном листе, индивидуальном плане, листе отчета с краткими итогами и отзыве на работу нумерация не ставится) до последнего листа текста, включая иллюстративный материал (таблицы, графики, диаграммы и т. п.), расположенный внутри текста, а также приложения.

**Требования к оформлению таблиц.**

Необходимым условием оформления работы является наличие иллюстративного материала, который может быть представлен в виде рисунков, схем, таблиц, графиков и диаграмм. Иллюстрации должны наглядно дополнять и подтверждать содержание текстового материала и отражать тему работы. На каждую единицу иллюстративного материала должна быть хотя бы одна ссылка в тексте пояснительной записки.

В том случае, когда текст иллюстрируется таблицами, они оформляются следующим образом. Таблицы следует размещать сразу после ссылки на них в тексте. Таблицы последовательно нумеруются арабскими цифрами в пределах всей работы.

Название таблиц размещается над левым верхним углом соответствующей таблицы, выравниваются по левому краю и оформляются шрифтом «TimesNewRoman» размер шрифта – 14, межстрочный интервал 1, в параметрах абзаца интервал перед – 15 пт, интервал после 10 пт.

Обратите внимание на то, что если требуется перенести таблицу на другую страницу, то ее «шапку» необходимо будет продублировать на каждой новой странице пояснительной записки, а прерывание таблицы внизу сопровождается отсутствием нижней границы у всей строки.

Если таблица размещается на двух страницах пояснительной записки, то ее заголовок на второй странице меняется на – «Продолжение таблицы N», где N номер соответствующей таблицы. В случае если таблица занимает более чем 2 страницы, то на последней странице с таблицей в заголовке указывается «Окончание таблицы N».

В случае если в таблице располагается большой объем текстового материала, то для корректного его отображения допускается изменить шрифт внутри таблицы на «TimesNewRoman» размер шрифта 10 или 12, а межстрочный интервал установить равным 1.

**Требования к оформлению рисунков, схем и тд.**

Все иллюстрации, не относящиеся к таблицам (схемы, графики, диаграммы и т.д.), именуются рисунками. Им присваивается сквозная нумерация в пределах всей работы. Все рисунки должны иметь полные наименования. Номер и наименование рисунка записываются через тире в строчку под его изображением без закрывающей точки.

Подрисуночная надпись начинается без абзацного отступа (красной строки), выравнивается по центру и оформляется обычным шрифтом «TimesNewRoman» размер шрифта – 14, межстрочный интервал 1, в параметрах абзаца отступ слева – 0 см, отступ справа – 0 см, интервал перед – 10 пт, интервал после 15 пт.

Если в тексте документа имеется иллюстрация, на которой изображены составные части изделия, то на этой иллюстрации должны быть указаны номера позиций этих составных частей в пределах данной иллюстрации, которые располагают в возрастающем порядке, за исключением повторяющихся позиций и описание составных частей вынесено в основной текст пояснительной записки.

На все рисунки и графики в пояснительной записке обязательно должны быть ссылки в тексте. Обратите внимание на что, что ссылка на рисунок должна стоять перед самим рисунком, на который она ссылается.

**Требования к оформлению листинга.**

Листингу присваивается сквозная нумерация в пределах всей работы. Листинги должны иметь полные наименования. Номер и наименование листинга записываются через тире в строчку непосредственно перед программным кодом без отступа красной строки и без закрывающей точки.

К оформлению программного кода в тексте пояснительной записке и приложениях предъявляются следующие основные требования:

- шрифт Courier или CourierNew, размер шрифта 10-12 pt, межстрочный интервал одинарный, выравнивание по левому краю, отступ красной строки 0 см;

- используйте табуляцию, а не пробелы, для отступов и оформления блоков. Для сдвига блока используйте один символ табуляции;

При расстановке в программном коде пустых строк и пробелов следует руководствоваться следующими правилами:

- запрещено использование двух пустых строк подряд;

- используйте одну пустую строку между процедурами и функциями;

- используйте одну пустую строку между логическими частями в процедуре;

Комментарии к программному коду пишутся на языке конфигурации (русский). Не рекомендуется использование комментария в одной строке с инструкциями. Такие комментарии рекомендуется располагать непосредственно над инструкцией. Комментируя код, старайтесь объяснять, что он делает, а не какаяоперация производится. Описывая действия, старайтесь описывать суть происходящего, а не то, что делают те или иные операторы. Так, совершенно бессмысленны комментарии вроде: «Присваиваем переменной a значение b» или «вызываем метод f». Отделяйте текст комментария от символов комментария одним пробелом: «// Текст комментария». Пишите каждое предложение комментария, начиная с большой буквы. Не забывайте, что точка в конце комментария не ставится.

На все листинги в тексте пояснительной записки обязательно должны быть ссылки в тексте. Ссылка следует ставить всегда выше по тексту самого листинга.

**Требования к оформлению списка литературы.**

Использованные в процессе работы специальные литературные источники указываются в конце пояснительной записки перед приложениями. Список использованных источников входит в основной объем работы. На каждый литературный источник в тексте работы обязательно должна быть хотя бы одна ссылка.

Список литературы составляется в порядке упоминания литературных источников в работе.

Для многотиражной литературы при составлении списка указываются: полное название источника, фамилия и инициалы автора, издательство и год выпуска (для статьи – название издания и его номер). Полное название литературного источника приводится в начале книги на 2-3 странице. Для законодательных актов необходимо указывать их полное название, принявший орган и дату принятия. При указании адресов серверов Internet сначала указывается название организации, которой принадлежит сервер, а затем его полный адрес.

*Пример библиографического описания книги:*

1. Курейчик, В. М. Ориентированные графы и конечные автоматы / В. М. Курейчик, В. П. Карелин, А. Н. Мелихов. – М. :Наука, 1971. – 416 с.

2. Абрамович, Г. Н. Турбулентные течения при воздействии объемных сил / Г. Н. Абрамович. – М. : Машиностроение, 1975. – 94 с.

*Пример библиографического описания статьи из журнала:*

3. Азаров, А. И. Промышленное применение многоцелевых воздухоохладителей / А. И. Азаров // Химическое и нефтегазовое машиностроение. – 1999. – №7. – С. 29 – 31.

*Пример библиографического описания статьи из сборника:*

4. Азаров, А. И. Расчет предельных температурно-энергетических характеристик противоточной вихревой трубы / А. И. Азаров, А. А. Кузьмин, С. О. Муратов // Вихревой эффект и его применение в технике : материалы V Всесоюзн. науч.-техн. конф. – Куйбышев : Изд-во Куйбыш. авиац. ин-та, 1988. – С. 23 – 27.

*Пример библиографического описания законодательных материалов:*

5. Российская Федерация. Конституция (1993). Конституция Российской Федерации [Текст] : офиц. текст. – М. : Маркетинг, 2001. – 39, [1] с. ; 20 см. – 10000 экз. – ISBN 5-94462-025-0.

*Пример библиографического описания патента:*

6. Пат. 2177591, Российская Федерация, МПК6 F 25 B 29/00. Термогенератор / Курносов Н. Е. ; заявитель и патентообладатель ООО «Термовихрь». – № 2000130684/06 ; заявл. 08.12.2000 ; опубл. 27.12.2001, Бюл. № 36, 2001. – 9 с.

*Пример библиографического описания диссертации:*

7. Виноградов, Ю. В. Моделирование процесса резания металла методом конечных элементов :дис. ... канд. техн. наук : 05.13.18 – Тула, 2004. – 119 с.

*Пример библиографического описания стандартов:*

8. ГОСТ Р 517721–2001. Аппаратура радиоэлектронная бытовая. Входные и выходные параметры и типы соединений. Технические требования. – Введ. 2002–01–01. – М. : Изд-во стандартов, 2001. – IV, 27 с.

*Пример библиографического описания интернет ресурсов:*

9. Федорова К.П. Объектный подход в проектировании баз данных // Портал научно-практических публикаций [Электронный ресурс]. URL: http://portalnp.ru/2014/06/1987 (дата обращения: 07.05.2017)

10. Тучкова А.С., Шерстянкина А.А., Зыкова Е.Н. Облачные технологии как способ решения бизнес-задач // Современные научные исследования и инновации. 2017. № 4 [Электронный ресурс]. URL: http://web.snauka.ru/issues/2017/04/80536 (дата обращения: 29.04.2017).

На все литературные источники в отчете обязательно должны быть ссылки в тексте. Ссылки на литературу оформляются как порядковый номер источника в соответствии со списком использованных источников, заключенный в квадратные скобки.

**ГОСТы для оформления отчета**

ГОСТ 7.32–2001 «Отчет о научно-исследовательской работе. Структура и правила оформления»;

ГОСТ 2.105–95 «Общие требования к текстовым документам»;

ГОСТ Р 7.0.5–2008 «Библиографическая ссылка. Общие требования и правила составления»;

ГОСТ 7.1–2003 «Библиографическая запись. Библиографическое описание. Общие требования и правила составления»;

ГОСТ 7.80–2000. «Библиографическая запись. Заголовок. Общие требования и правила составления».