Таблица 14. Варианты индивидуальных заданий.

№	Алгоритм задания 1	Алгоритм задания 3
1	Простой вставки (***Insertion sort***)	Простого выбора ***(Selection sort)***
2	Простого обмена (***Exchange sort***)	Простой вставки (***Insertion sort***)
3	Простого выбора ***(Selection sort)***	Простого обмена (***Exchange sort***)

Отчёт:

В отчёте по каждой сортировке необходимо привести словесное описание алгоритма и его блок-схему, а также программный код (с комментариями), результаты тестирования на массиве n=10 и контрольных прогонов на массивах длиной 100, 1000, 10000, 100000 и 1000000 элементов.

По каждому пункту в заданиях 1-3 приведите полученные результаты в виде таблиц A, B и C и графиков.

По итогам каждого задания сформулируйте соответствующие выводы.

В выводах по всей работе опишите, какие знания, умения и практические навыки получены в ходе выполнения практической работы.

11.3. Практическая работа 3

Тема: «Асимптотический анализ эффективности на примерах алгоритмов сортировки».

Задание 1. Эмпирическая оценка эффективности алгоритмов в среднем случае.

Требования по выполнению задания

1. Разработать алгоритм ускоренной сортировки, определенной в варианте (табл. 17), реализовать код на языке C++. Сформировать таблицу 1.1 результатов эмпирической оценки сложности сортировки по формату табл. 15 для массива, заполненного случайными числами. Определить ёмкостную сложность алгоритма.

Таблица 15. Сводная таблица результатов

n	T(n), мс	$T_п(n)=C_ф+M_ф$
100		
1000		
10000		
100000		
1000000		

1. Разработать алгоритм быстрой сортировки, определенной в варианте (приложение 1), реализовать код на языке C++. Сформировать таблицу 2.1 результатов эмпирической оценки сортировки по формату табл. 15 для массива, заполненного случайными числами. Определить ёмкостную сложность алгоритма.

2. Представить на графике зависимости $\mathbf{T_п(n)=C_ф+M_ф}$ для двух анализируемых алгоритмов.
3. На основе анализа полученных данных определите наиболее эффективный из алгоритмов в среднем случае.

Задание 2. Асимптотический анализ сложности алгоритмов.

Требования по выполнению задания:

1. Провести дополнительные прогоны программы на рабочих массивах, отсортированных а) строго в убывающем и б) строго возрастающем порядке значений элементов. Заполнить по соответствующие таблицы 1.2, 1.3, 2.2, 2.3 для каждого алгоритма по формату табл. 15.

   Сделайте вывод о зависимости (или независимости) алгоритмов сортировок от исходной упорядоченности массива на основе результатов, представленных в таблицах**.**
2. На основе анализа кода этих алгоритмов определить, если это возможно, формулы $T_т(n)$ функций роста в лучшем и худшем случае.

   Добавьте в таблицы 1.2, 1.3, 2.2, 2.3 столбец $T_т(n)=C+M$ и заполните соответствующими расчётными значениями.
3. Получить асимптотическую оценку обоих алгоритмов в худшем случае (сверху) и в лучшем случае (снизу).

   Получите (если это возможно) асимптотически точную оценку вычислительной сложности алгоритма.

   При невозможности получения асимптотической оценки привести справочную информацию.

   Для наиболее эффективного алгоритма простой сортировки по вашему варианту предыдущей практической работы получить асимптотическую оценку сложности.

   Общие результаты свести в табл. 16.

Таблица 16. Сводная таблица результатов

Алгоритм	Асимптотическая сложность алгоритма			
	Наихудший случай (сверху)	Наилучший случай (снизу)	Средний случай (точная оценка)	Ёмкостная сложность
Простой				
Усовершенствованный				
Быстрый				

Сделать вывод о наиболее эффективном алгоритме из трёх.

Отчёт:

В отчёте по каждой сортировке необходимо привести словесное описание алгоритма и его блок-схему, а также программный код (с комментариями), результаты тестирования на массиве n=10 и контрольных прогонов на массивах длиной 100, 1000, 10000, 100000 и 1000000 элементов.

По итогам выполнения каждого задания сформулируйте соответствующие выводы.

В выводах по всей работе опишите, какие знания, умения и практические навыки получены в ходе выполнения практической работы.

Таблица 17. Варианты индивидуальных заданий

Вариант	**Усовершенствованный алгоритм**	**Быстрый алгоритм**
1	Сортировка обменами с условием Айверсона	Простое слияние
2	Шейкерная сортировка	Простое слияние
3	Шейкерная с условием Айверсона	Простое слияние
4	Сортировка Шелла со смещениями Д. Кнута. Способ 1	Простое слияние
5	Шелла со смещениями Д. Кнута. Способ 2	Простое слияние
6	Шелла со смещениями Р. Седжвика.	Простое слияние
7	Пирамидальная сортировка	Простое слияние
8	Турнирная сортировка	Простое слияние
9	Сортировка обменами с условием Айверсона	Быстрая сортировка (Хоара)

Продолжение табл. 17.

10	Шейкерная сортировка	Быстрая сортировка (Хоара)
11	Шейкерная с условием Айверсона	Быстрая сортировка (Хоара)
12	Сортировка Шелла со смещениями Д. Кнута. Способ 1	Быстрая сортировка (Хоара)
13	Шелла со смещениями Д. Кнута. Способ 2	Быстрая сортировка (Хоара)
14	Шелла со смещениями Р. Седжвика.	Быстрая сортировка (Хоара)
15	Пирамидальная сортировка	Быстрая сортировка (Хоара)
16	Турнирная сортировка	Быстрая сортировка (Хоара)

11.4. Практическая работа 4

Тема: «Алгоритмы внешних сортировок».

Задание 1. Разработать программу и применить алгоритм внешней сортировки *прямого слияния* к сортировке файла данных варианта по значению ключевого поля (табл. 8). Ключ в структуре записи варианта – подчеркнутое поле.

1) Файл данных варианта предварительно подготовить в тестовом файле с помощью любого тестового редактора.
2) Так как записи должны перемещаться, то удобнее хранить данные в двоичном файле, записи которого имеют фиксированную длину. Создать двоичный файл из записей, представленных в текстовом файле.
3) Разработать программу (функцию или несколько) сортировки.
4) Отладить программу, определить практическую сложность алгоритма для файлов с увеличивающимся количеством записей.
5) Сформировать таблицу результатов, указав количество записей и время сортировки.

Алгоритм сортировки прямого слияния для файлов

Фаза разделения:

1. Открыть файл A как входной.
2. Открыть файлы B и C как выходные(для записи).
3. Считываемые из A записи попеременно записываем в файлы B и C.
4. Закрываем файлы A, B, C.

Фаза слияния:

1. Открыть файл A как выходкой (для записи).
2. Открываем файлы B и C как входные (для чтения).