1. АЛГОРИТМЫ ОБРАБОТКИ МАССИВОВ: ЗАДАНИЯ

1.1. Задания по сортировке прямого выбора SelectSort

1. Для набора из 8 первых последовательных символов ФИО студента выполнить вручную сортировку методом прямого выбора SelectSort. При тестировании программы можно использовать данный пример для проверки правильности реализации алгоритма.

2. Разработать подпрограмму сортировки массива целых чисел по возрастанию методом прямого выборам SelectSort. Правильность сортировки проверить путем подсчета контрольной суммы и числа серий в массиве. Предусмотреть подсчет фактического количества пересылок и сравнений (Мф и Сф) и сравнить их с теоретическими оценками (М и С) для различных типов массивов.

3. Исследовать метод прямого выбора SelectSort на массивах убывающих, возрастающих и случайных чисел и сделать вывод о зависимости (или независимости) метода SelectSort от исходной упорядоченности массива.

4*. Придумать способ устранения фиктивных перестановок и оценить его влияние на фактические значения Мф и Сф для метода прямого выбора. Исследовать улучшенный метод прямого выбора SelectSort на массивах убывающих, возрастающих и случайных чисел, сделать выводы.

5. Составить таблицу:

Трудоемкость метода прямого выбора

N	M+C теоретич.	Исходный $M_{факт}+C_{факт}$			Улучшенный $M_{факт}+C_{факт}$		
		Убыв.	Случ.	Возр.	Убыв.	Случ.	Возр.
10							
100							

5*. Построить на экране в координатной плоскости график зависимости $M_ф+C_ф$ от размера массива n для метода прямого выбора SelectSort (для массива случайных чисел).

1.2. Задания по пузырьковой сортировке BubbleSort

1. Для набора из 8 первых последовательных символов ФИО студента выполнить вручную пузырьковой сортировки BubbleSort. При тестировании программы можно использовать данные примеры для проверки правильности реализации алгоритмов.

2. Разработать подпрограмму пузырьковой сортировки BubbleSort для массива целых чисел по возрастанию. Правильность сортировки проверить путем подсчета контрольной суммы и числа серий в массиве. Предусмотреть подсчет фактического количества пересылок и сравнений (Мф и Сф), сравнить с теоретическими оценками М и С.

3. Сравнить время работы пузырьковой сортировки BubbleSort на массивах убывающих, возрастающих и случайных чисел (по сумме Мф+Сф) и сделать вывод о зависимости (или независимости) метода BubbleSort от исходной упорядоченности массива.

4. Составить таблицу:

Трудоемкость пузырьковой сортировки

N	М+С теоретич.	$М_{факт}+С_{факт}$		
		Убыв.	Случ.	Возр.
100				
200				
300				
400				
500				

5*. Построить на экране в координатной плоскости график зависимости $М_ф+С_ф$ от размера массива n для метода прямого выбора SelectSort и пузырьковой сортировки BubbleSort (для массива случайных чисел).

1.3. Задания по шейкерной сортировке ShakerSort

1. Для набора из 8 первых последовательных символов ФИО студента выполнить вручную сортировку шейкерную сортировку ShakerSort. При тестировании программы можно использовать данные примеры для проверки правильности реализации алгоритмов.

2. Разработать подпрограмму шейкерной сортировки ShakerSort для массива целых чисел по возрастанию. Правильность сортировки проверить путем подсчета контрольной суммы и числа серий в массиве. Предусмотреть подсчет фактического количества пересылок и сравнений (Мф и Сф), сравнить с теоретическими оценками М и С.

3. Сравнить время работы шейкерной сортировки ShakerSort на массивах убывающих, возрастающих и случайных чисел (по сумме Мф+Сф) и сделать вывод о зависимости (или независимости) метода ShakerSort от исходной упорядоченности массива.

4. Составить таблицу:

Трудоемкость пузырьковой и шейкерной сортировок

n	$M_ф+C_ф$ пузырьковой			$M_ф+C_ф$ шейкерной		
	Убыв.	Случ.	Возр.	Убыв.	Случ.	Возр.
100						
200						
300						
400						
500						

5*. Построить на экране в одной координатной плоскости графики зависимости $M_ф+C_ф$ от размера массива n для метода прямого выбора SelectSort и пузырьковой сортировки BubbleSort и шейкерной сортировки ShakerSort (для массива случайных чисел).

1.4. Задания по сортировке прямого включения InsertSort

1. Для набора из 8 первых последовательных символов ФИО студента выполнить вручную сортировку методом прямого включения InsertSort. При тестировании программы можно использовать данные примеры для проверки правильности реализации алгоритмов.

2. Разработать подпрограмму сортировки массива целых чисел по возрастанию методом прямого включения InsertSort. Предусмотреть подсчет фактического количества пересылок и сравнений (Мф и Сф), сравнить с теоретическими оценками М и С.

3. Сравнить время работы метода прямого включения InsertSort на массивах убывающих, возрастающих и случайных чисел (по сумме Мф+Сф) и сделать вывод о зависимости (или независимости) метода InsertSort от исходной упорядоченности массива.

Составить таблицу:

Трудоемкость метода прямого включения InsertSort

N	М+С теоретич.	$M_{факт}+C_{факт}$		
		Убыв.	Случ.	Возр.
100				
200				
300				
400				
500				

4. Сравнить время работы методов квадратичной трудоемкости SelectSort, BubbleSort, ShakerSort, InsertSort на массивах случайных чисел (по сумме Мф+Сф), использовать результаты выполнения предыдущих заданий.

Проанализировать полученные результаты и сделать вывод о фактической трудоемкости рассмотренных методов.

Составить таблицу:

Трудоемкость квадратичных методов сортировки

n	$M_ф+C_ф$			
	Select	Bubble	Shaker	Insert
100				
200				
300				
400				
500				

5*. Построить на экране в одной координатной плоскости графики зависимости $M_ф+C_ф$ от размера массива n для четырех методов сортировки квадратичной трудоемкости (для массива случайных чисел).

1.5. Задания по сортировке Шелла ShellSort

1. Для набора из 8 первых последовательных символов ФИО студента выполнить вручную сортировку методом Шелла ShellSort (взять последовательность шагов h=2, h=1). При тестировании программы можно использовать данные примеры для проверки правильности реализации алгоритмов.

2. Разработать подпрограмму сортировки массива целых чисел по возрастанию методом Шелла ShellSort. Правильность сортировки проверить путем подсчета контрольной суммы и числа серий в массиве. Предусмотреть подсчет фактического количества пересылок и сравнений (Мф и Сф), сравнить с теоретическими оценками М и С.

3. Исследовать трудоемкость метода Шелла ShellSort (по сумме Мф+Сф) для n=100, …, 500, n – количество элементов в массиве. Определить последовательность шагов для предварительных сортировок по формуле Д.Кнута. Сравнить с методом прямого включения InsertSort и проанализировать полученные результаты.

Построить таблицу:

Трудоемкость метода Шелла

n	$h_1 \dots h_m$ по формуле Д.Кнута	Insert $M_ф+C_ф$	Shell $M_ф+C_ф$
100			
200			
300			

400			
500			

4*. Исследовать метод Шелла ShellSort на зависимость трудоемкости (Мф+Сф) от выбора последовательности шагов. Подтвердить рекомендацию Д.Кнута или предложить лучший вариант выбора последовательности шагов.

Построить таблицу:

Исследование трудоемкости метода Шелла

n	$h_1 \dots h_m$ по формуле Д.Кнута	Shell $M_ф+C_ф$	$h_1 \dots h_m$ по другой формуле	Shell $M_ф+C_ф$
100				
200				
300				
400				
500				

5*. Построить на экране в одной координатной плоскости графики зависимости $M_ф+C_ф$ от размера массива n для четырех методов сортировки квадратичной трудоемкости и метода Шелла ShellSort (для массива случайных чисел).

1.6. Задания по быстрому двоичному поиску

1. Для набора из 12 первых последовательных символов ФИО студента выполнить вручную быстрый поиск (двумя версиями) той буквы ФИО, которая наиболее часто встречается. При тестировании программы можно использовать данные примеры для проверки правильности реализации алгоритмов.

2. Разработать подпрограммы для двоичного поиска заданного элемента в упорядоченном массиве (две версии алгоритма). Ключ поиска передается в подпрограмму в качестве параметра или вводится с клавиатуры. На экран вывести найденный элемент и номер его позиции в массиве. Правильность работы проверить путем поиска первого элемента массива, последнего элемента массива и элемента, которого нет в массиве. Предусмотреть подсчет фактического количества сравнений ($C_ф$), сравнить с теоретическими оценками С.

3. Сравнить трудоемкости ($C_ф$) двух версий алгоритма двоичного поиска заданного элемента в упорядоченном массиве. Проанализировать полученные результаты.

Построить таблицу:

Трудоемкость двоичного поиска элемента

N	$C_ф$ I версия	$C_ф$ II версия
100 … 500 … 1000		

4*. Разработать подпрограммы поиска в упорядоченном массиве всех элементов с заданным ключом (две версии) BSearchAll1 и BSearchAll2., ключ поиска передается в подпрограмму в качестве параметра или вводится с клавиатуры. Предусмотреть подсчет фактического количества сравнений ($C_ф$).

Построить таблицу:

Трудоемкость двоичного поиска всех элементов

N	$C_ф$ All I версия	$C_ф$ All II версия
100 … 500 … 1000		

5*. Построить на экране в одной координатной плоскости графики зависимости $C_ф$ от n для двух версий двоичного поиска элемента с заданным ключом.

1.7. Задания по сортировке массивов структур

1. Для набора из четырех ФИО (ФИО студента, его однофамильца и двух друзей) выполнить вручную сортировку по сложному ключу, состоящему из фамилии и имени. При тестировании программы можно использовать данные примеры для проверки правильности реализации алгоритмов.

2. Разработать программу сортировки телефонного справочника по сложному (составному) ключу с использованием любого метода сортировки, кроме пузырькового. Структура записи справочника должна состоять минимум из четырех полей. Исходный массив записей задать в программе (4-5 записей).

3. Реализовать функцию сравнения абонентов справочника по сложному ключу, состоящему минимум из двух полей записи абонента.

4. Реализовать вывод на экран исходного справочника и справочника, отсортированного по сложному ключу.

5. Предусмотреть возможность изменения ключа сортировки и направления сортировки телефонного справочника путем изменения функции сравнения абонентов.

6*. Реализовать быстрый двоичный поиск в отсортированном справочнике по старшей части сложного ключа сортировки.

1.8. Задания по индексации массивов

1. Для набора из 12 первых последовательных символов ФИО студента, пронумерованных от 1 до 12, выполнить вручную построение четырех индексных массивов: 1) для сортировки исходного массива по возрастанию, 2) для сортировки исходного массива по убыванию, 3) для фильтрации только согласных букв по возрастанию, 4) для фильтрации только гласных букв по убыванию. При тестировании программы можно использовать данные примеры для проверки правильности реализации алгоритмов.

2. Разработать программу сортировки телефонного справочника с использованием индексации. Структура записи справочника должна состоять из четырех полей. Исходный массив записей (структур) задать в программе (4-5 записей).

3. Предусмотреть создание в памяти компьютера двух индексных массивов (для упорядочивания справочника по двум разным полям) с использованием любого метода сортировки, кроме пузырькового.

4. Предусмотреть вывод на экран справочника, отсортированного по двум разным полям с использованием построенных индексных массивов.

5*. Реализовать быстрый двоичный поиск в справочнике (по двум разным ключам) с использованием построенных индексных массивов.

1.9. Задания по пирамидальной сортировке HeapSort

1. Для набора из 12 первых последовательных символов ФИО студента выполнить вручную сортировку методом пирамидальной сортировки HeapSort. При тестировании программы можно использовать данный пример для проверки правильности реализации алгоритма.

2. Разработать подпрограмму построения пирамиды из массива целых чисел. Предусмотреть подсчет фактического количества пересылок и сравнений (Мф и Сф), сравнить с теоретическими оценками М и С.

Составить таблицу:

Трудоемкость построения пирамиды

N	M+C теоретич.	$M_{факт}+C_{факт}$		
		Убыв.	Случ.	Возр.
100				
200				
300				
400				
500				

3. Разработать подпрограмму пирамидальной сортировки HeapSort для массива целых чисел по возрастанию. Правильность сортировки проверить путем подсчета контрольной суммы и числа серий в массиве. Предусмотреть подсчет фактического количества пересылок и сравнений (Мф и Сф), сравнить с теоретическими оценками М и С.

4. Исследовать трудоемкость метода пирамидальной сортировки HeapSort на массивах убывающих, возрастающих и случайных чисел (по сумме Мф+Сф) и сделать вывод о зависимости или независимости метода от исходной упорядоченности массива.

Построить таблицу:

Трудоемкость пирамидальной сортировки

N	HeapSort ($M_ф+C_ф$)		
	Убыв.	Возр.	Случ.
100			
200			
300			
400			
500			

5*. Построить на экране в одной координатной плоскости графики зависимости $M_ф+C_ф$ от размера массива n для пирамидальной сортировки HeapSort и метода Шелла ShellSort (для массива случайных чисел).

1.10. Задания по быстрой сортировке QuickSort

1. Для набора из 12 первых последовательных символов ФИО студента выполнить вручную сортировку методом Хоара QuickSort. При тестировании программы можно использовать данный пример для проверки правильности реализации алгоритма.

2. Разработать подпрограмму сортировки массива целых чисел методом Хоара QuickSort (первая версия). Правильность сортировки проверить путем подсчета контрольной суммы и числа серий в массиве. Предусмотреть

подсчет фактического количества пересылок и сравнений (Мф и Сф), сравнить с теоретическими оценками М и С.

3. Исследовать трудоемкость сортировки методом Хоара QuickSort на массивах убывающих, возрастающих и случайных чисел (по сумме Мф+Сф) и сделать вывод о зависимости или независимости метода от исходной упорядоченности массива.

Построить таблицу:

Трудоемкость метода Хоара

N	QuickSort ($M_ф+C_ф$)		
	Убыв.	Возр.	Случ.
100			
200			
300			
400			
500			

4*. Сравнить две версии сортировки методом Хоара QuickSort по глубине рекурсии (при вычислении глубины рекурсии не путать ее с количеством рекурсивных вызовов). Проанализировать полученные результаты.

Составить таблицу:

Глубина рекурсии сортировки методом Хоара

n	QuickSort1			QuickSort2		
	Убыв.	Случ.	Возр.	Убыв.	Случ.	Возр.
100						
200						
300						
400						
500						

5*. Построить на экране в одной координатной плоскости графики зависимости $M_ф+C_ф$ от размера массива n для пирамидальной сортировки HeapSort, метода Шелла ShellSort и метода Хоара QuickSort (для массива случайных чисел).