Лабораторная работа №4

«Продвинутые алгоритмы сортировки и алгоритмы поиска»

Цель работы: изучить продвинутые алгоритмы сортировки и поиска. Дополнительно реализовать шаблонный класс – аналог std::vector из C++ 23 и сделать их стандартными в этом контейнере.

Краткие теоретические сведения:

Быстрая сортировка — эффективный алгоритм сортировки на месте, который обычно работает примерно в два-три раза быстрее, чем Сортировка слиянием а также *сортировка кучей* при хорошей реализации. Быстрая сортировка — это сортировка сравнением, то есть она может сортировать элементы любого типа, для которых *меньше, чем* отношение определено. В эффективных реализациях это обычно нестабильная сортировка.

Быстрая сортировка в среднем дает O (n * log(n)) сравнения для сортировки n Предметы. В худшем случае получается $O(n^2)$ сравнения, хотя такое поведение встречается очень редко.

Задание 1.

Разработать программу, которая принимает n размерный массив, заполненный целыми числами. Отсортировать его по возрастанию используя перечисленные сортировки и вывести скорость работы каждой сортировки. Затем пользователь вводит число. Реализовать процедуру binsearch (int *arr, int digit), которая находит число бинарным поиском и выводит индекс, по которому оно расположено в массиве. Если число не найдено – выведите -1.

1) Heap sort (Сортировка кучей)
2) Quick sort (Быстрая сортировка)
3) Merge sort (Сортировка слиянием)

Задание 2.

Реализовать функционал задания 2 с помощью интерполяционной сортировки. После каждого шага сортировки необходимо выводить массив на экран. Определить функцию бинарного возведения в степень binpow (int digit, int powder, int mod). С ее помощью в ответ вывести индекс найденного элемента в степени длинны массива по модулю числа.

Задание 3.

Медианой последовательности с нечётным числом членов будем называть значение, которое встаёт в середину, если последовательность отсортировать. Т. е. половина значений последовательности не меньше медианного элемента и половина значений не больше медианы. Для заданного вектора a построить вектор b из медиан подряд идущих троек элементов. Для неполных троек брать арифметическое среднее.

Пример:

Дано a = { 1, 5, 1, 4, 5, 6, 2, 1, 3, 4, 4, 4, 5, 7 }. Разбиваем на тройки: { 1, 5, 1 }, { 4, 5, 6 }, { 2, 1, 3 }, { 4, 4, 4 }, { 5, 7 }, последняя “тройка” неполная — два элемента. Получаем

набор медиан (последнее значение — арифметическое среднее последних двух элементов): b = {1, 5, 2, 4, 6}.

Данных задач достаточно, чтобы защитить лабораторную на минимальную оценку.

Задание 4.

Реализовать статическую библиотеку Vector (аналог std::vector) не используя стандартные библиотеки C++. Использовать шаблоны. Реализовать простейший итератор и наследоваться от него в классе vector. Для работы класса итератора перегрузите операторы по аналогии с std::vector. В библиотеке vector необходимо реализовать следующие методы:

- assign; Удаляет вектор и копирует указанные элементы в пустой вектор.
- at; Возвращает ссылку на элемент в заданном положении в векторе.
- back; Возвращает ссылку на последний элемент вектора.
- begin; Возвращает итератор произвольного доступа, указывающий на первый элемент в векторе.
- capacity; Возвращает число элементов, которое вектор может содержать без выделения дополнительного пространства.
- cbegin; Возвращает *константный* итератор произвольного доступа, указывающий на первый элемент в векторе.
- clear; Очищает элементы вектора.
- data; Возвращает указатель на первый элемент в векторе.
- emplace; Вставляет элемент, созданный на месте, в указанное положение в векторе.
- emplace_back; Добавляет элемент, созданный на месте, в конец вектора.
- empty; Проверяет, пуст ли контейнер вектора.
- end; Возвращает итератор произвольного доступа, который указывает на конец вектора.
- erase; Удаляет элемент или диапазон элементов в векторе из заданных позиций.
- front; Возвращает ссылку на первый элемент в векторе.
- insert; Вставляет элемент или множество элементов в заданную позиции в вектор.
- max_size; Возвращает максимальную длину вектора.
- pop_back; Удаляет элемент в конце вектора.
- push_back; Добавляет элемент в конец вектора.
- rbegin; Возвращает итератор, указывающий на первый элемент в обратном векторе.
- rend; Возвращает итератор, который указывает на последний элемент в обратном векторе.
- reserve; Резервирует минимальную длину хранилища для объекта вектора.
- resize; Определяет новый размер вектора.
- size; Возвращает количество элементов в векторе.
- swap; Меняет местами элементы двух векторов.

Задание 5.

Реализовать контейнер pair. Предусмотреть вариант pair<pair<T, T>, pair<T, T>> a; Разработайте оконное приложение используя собственную библиотеку vector. Необходимо создать объект класс vector, где каждый объект это pair< vector<int>, vector<pair<int,double>> >. Вывести две матрицы на экран, где первая матрица это первый аргумент pair (vector<int>), вторая – второй аргумент pair (vector <pair<int, double>>). Продемонстрировать работу оставшихся методов вашего класса vector из задания 4 используя оконное приложение.