Практическая работа №1. Алгоритмы сортировки

Цель: Изучить несколько алгоритмов сортировки и сравнить их свойства.

Задача: Написать программу, реализующую два заданных алгоритма сортировки (согласно варианту). Сравнить скорость работы алгоритмов на массивах различной длины.

Общие сведения

Если над некоторым множеством элементов определена операция сравнения, то такое множество может быть упорядочено. Для упорядочивания наборов данных, представленных в виде массивов, существуют различные алгоритмы. Алгоритмы классифицируются по различным характеристикам. В частности, по зависимости количества необходимых операций от размера входных данных, называемой вычислительной эффективностью (или просто быстродействием). В данной лабораторной работе рассматриваются примеры алгоритмов, обладающих квадратичной и субквадратичной вычислительной эффективностью (такие алгоритмы называют быстрыми).

Задание:

1. Написать две функции сортировки массива целых чисел, реализующих заданные алгоритмы сортировки – один из класса квадратичных алгоритмов, другой из класса быстрых алгоритмов.

2. Исследовать вычислительную эффективность этих алгоритмов. Для этого необходимо использовать функции стандартной библиотеки, которые позволяют измерять время работы участка программы. Программа для измерения вычислительной эффективности должна создавать массив для сортировки, заполняя его случайными числами, потом вызывать функцию сортировки, замеряя время её работы. Если время одной сортировки слишком мало, надо проводить генерацию случайного массива и его сортировку в цикле большое число раз и измерять суммарное (либо среднее) время сортировки. Результатом данного пункта должны быть времена выполнения сортировки обоими методами для размеров массивов примерно от 1000 до 100000 (6-7 значений размера).

Варианты задания

1. Шейкер-сортировка и пирамидальная сортировка (heapsort).
2. Шейкер-сортировка и сортировка слиянием.
3. Шейкер-сортировка и быстрая сортировка (quicksort).
4. Пузырьковая сортировка и сортировка слиянием.
5. Пузырьковая сортировка и быстрая сортировка (quicksort).

Примечания:

1. Для измерения времени работы программы можно воспользоваться функцией **clock( )** (заголовочный файл **time.h**), которая при вызове возвращает значение, равное количеству «тиков» системных часов, прошедших от начала работы программы домомента вызова. В **time.h** также определена константа для перевода «тиков» в секунды.
2. В алгоритме быстрой сортировки опорный элемент определять случайным образом.