# Лабораторная работа № 1 по курсу дискретного анализа: сортировка за линейное время

Выполнил студент группы М8О-207Б-20 МАИ *Белоусов Егор Владимирович.*

## Условие

Требуется разработать программу, осуществляющую ввод пар «ключ- значение», их упорядочивание по возрастанию ключа указанным алгоритмом сортировки за линейное время и вывод отсортированной последовательности.

*Вариант 9-2: Карманная сортировка,*
*Ключ: вещественные числа в промежутке [−100, 100],*
*Значение: строки переменной длины (до 2048 символов).*

## Метод решения

Алгоритм карманной сортировки используется для сортировки данных с нецелыми ключами, этот алгоритм является эффективным (работающим за $O(n)$) только при данных, которые расположены *равномерно.*
Сперва алгоритм разбивает промежуток ключей на $n$ равных полуинтервалов, так называемые *«карманы»*, каждый элемент массива отправляется в соответствующий карман для собственного ключа. Затем каждый карман независимо от других сортируется с помощью *сортировки вставкой.* В конечном итоге все карманы объединяются в результирующий массив.

## Описание программы

Структурно в программе есть два файла: **main.cpp** и **BucketSort.hpp**. В **BucketSort.hpp** реализуется класс *BucketSort*, который содержит публичный статический метод сортировки *sort*, приватную функцию *InsertionSort* и компаратор для пар *PairComparator*. В основном файле происходит считывание данных в *std::vector*, сортировка и последующий вывод.

## Дневник отладки

При выполении лабораторной работы возникло несколько проблем. Первой из них было использование стандартного функционального класса *std::less* для пар, из-за этого программа:

1. Нарушала условие стабильности ввиду сравнения вторых значений пар.
2. Сильно замедлялась, так как часто сравнивались строки.

Для решения этой проблемы был реализован собственный функциональный класс *PairComparator*. Второй проблемой был неправильный вывод отсортированных данных: между ключом и значением стоял пробел, а не табуляция.

## Тест производительности

Решение было протестировано с помощью генератора тестов на C++ на различных размерах данных:

| N | Time in ms |
|---|---|
| $10^3$ | 9 ms |
| $10^4$ | 47 ms |
| $10^5$ | 178 ms |
| $10^6$ | 1856 ms |
| $5 \cdot 10^6$ | 11345 ms |

## Выводы

Алгоритмы сортировки используются ежедневно в работе почти любого программиста, а так как в современном мире крайне важна производительность программ, то требуется реализовывать частые участки программы эффективно при удобном случае. Карманная сортировка лучшего всего подходит, когда нужно отсортировать данные, полученные при каком-то опыте, в котором присутствует равномерное распределение результатов. В таких случаях сложность сортировки $O(n)$ будет показывать себя лучше, чем классическая быстрая сортировка с временной сложностью $O(nlog(n))$.