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

Выполнил студент группы 08-208 МАИ *Коростелев Дмитрий Васильевич.*

Условие

1. Общая постановка задачи.

   Требуется разработать программу, осуществляющую ввод пар «ключ-значение», их упорядочивание по возрастанию ключа указанным алгоритмом сортировки за линейное время и вывод отсортированной последовательности. Вариант задания определяется типом ключа (и соответствующим ему методом сортировки) и типом значения:

2. Вариант задания.

   (a) Номер варианта: 7-2

   (b) Сортировка: поразрядная сортировка.

   (c) Тип ключа: автомобильные номера в формате A 999 BC (используются буквы латинского алфавита).

   (d) Тип значения: строки переменной длины (до 2048 символов).

Метод решения

В данном варианте лабораторной работы требуется отсортировать значения по соответствующим им ключам, где значения – это строки переменной длины (до 2048 символов), а ключи – автомобильные номера типа: A 123 BC. Метод сортировки – поразрядная.

Требуется правильно прочитать входной файл и сохранить все ключи и соответствующие им значения. Считывание данных производится со стандартного потока ввода и выводы - std::cin и std::cout, но, так как требуется написать эффективную по памяти и времени программу, нужно ограничить до минимума вызовы данных операторов. Так же для этого нужно еще игнорировать пустые строки, для этого будем использовать следующий алгоритм: пока есть возможность прочитать первый символ ключа, чиатем его, затем получаем остальной кусок ключа и саму строку.

Далее нужно отсортировать вектор используя поразрядную сортировку. Для этого используется структура, в которой будет записан ключ и индекс. Сортировка проходит по каждому элементу ключа, начиная с конца и игнорируя пробелы. Стоит отметить, что в целях экономии времени меняются только ключи и значения, указывающие на строки в другом массиве.

Описание программы

Для решения данной задачи требовалось реализовать шаблонный класс вектора, функцию сортировки вектора, а также простой класс строки и структуру, которая хранит ключ с индексом.

Вектор – массив, который может расширятся посредством добавления в него новых элементов. Основные методы:

- void PushBack(T value) – добавляет новый элемент в вектор. Если в векторе есть место, значение просто добавляется, если места нет, происходит увеличение вектора в два раза, затем новый элемент добавляется в вектор.
- size_t Size() const – возвращает кол-во элементов в векторе
- T At(int index) const и T operator[](int index) const – возвращает элемент из хранилища по соответствующему индексу.

Простой класс строки представляет из себя массив для хранеия 2048 символов, конструктор по умолчанию, конструктор через ввод, и перегруженный оператор ввывода в поток std::cout

Структура для хранения ключа и индекса состоит из char массива на 9 символов, а также переменной типа int.

Поразрядная сортировка void RadixSort(TVector<Element>& arr) начинает проверку с предпоследнего элемента строки (так как последний - символ окончания строки), игнорирует пробелы (так как строгая типизация ключа) и далее вниз, сравнивает значения от '0' до 'Z' так как остальные значения в автомоильных знаках не исопльзуются.

Дневник отладки

№	Вердикт	Проблема и решение
1-3	CE	Пропущенные символы, неправильное использование scanf
4	WA	Неправильный формат вывода данных
5-21	WA, CE	Некоректная обработка пустых строк, переход на std::cin и std::cout
22-45	TL	Оптимизация программа, уменьщение кол-ва системных вызовов
46	WO	Оптимизация шаблонного класса вектора
46-51	WO	Корректировка кода в соответствии с код-стайлом

Тест производительности

Проверять линейности данной сортировки будем на тестах разного объема, по комплектам из 10, 50, 100, 200, 500 и 1000 строк, результаты заносить в таблицу, в которой

первая колонка - номер теста, вторая - объем строк, третья - время работы сортировки, данные представлены в милисекундах.

№	Объем теста	Время
1	10	0.0205
2	50	0.0787
3	100	0.1788
4	200	0.2804
5	500	0.7316
6	1000	1.3272
7	5000	6.5057
8	10000	24.4603
9	100000	72.6016
10	200000	161.473

Недочёты

Данная программа во время сортировки использует char массив на 256 символов, по факту, используется только некоторая малая часть этого, с одной стороны это плохо, потому что бесцельно тратиться и не используется память, с другой стороны данную программу можно расширить и "научить"соритровать не только автомобильные номера, состоящие из латинского алфавита и цифр, а из любого набора чисел.

Выводы

Данный алгоритм будет крайне полезен для сортировки различных баз данных, так как при линейной сложности сортировка выполняется быстро по сравнению с сортирвками сравнениями. В то же время стоит отметить, что сам алгоритм сортировки программируется довольно легко, а итоговый код легок в прочтении, но алгоритм поразрядной сортировки требует правильного и бережного отношения к памяти, так как при неверном написании можно получить программу, которая будет требовать неоправдано много памяти.