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

Выполнил студент группы 08-207 МАИ *Днепров Иван.*

Условие

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

2. Вариант задания 1-2. Тип сортировки: сортировка подсчетом, тип ключа: числа от 0 до 65535.

Метод решения

Порграмма зачитывает пары (ключ, значение) которые хранит в динамическом списке. В процессе считывания определяется наименьший и наибольший элемент. В прецессе сортировки программа создает массив списков, размерность которого равна разности наибольшего и наименьшего ключей увеличенной на единицу, в каждой ячейке которого хранятся все эллементы с соответствующем ключем. Процесс сортировки заключается в том, что мы последовательно зачитываем элементы из списка и добваляем их в конец списка, соответствующего значению ключа. По завершении сортировки программа последовательно выводит элементы списков в массиве, начиная со списка, хранящегося в нулевой ячейке, пропуская ячейки с нулевыми списками.

Описание программы

Весь код хранится в одном файле, в котором определены два типа: элемент списка *KeyPair*, состоящего из ключа типа *unsigned short*, значения в виде массива типа *char* и ссылки на следующий элемент списка и списко *IndexedElement*, состоящий из двух ссылок тина *KeyPair* на первый и последний элемент списка.

Дневник отладки

Я не проверял, введен ли ключ перед значением. Чтобы это исправить я добавил переменную *scannedKey* в которой я храню то, что ввернуля функция *scanf* при считывании ключа.

Ключ я сразу зачитывал в *unsigned short*, то есть я не проверял, входит ли ключ в нужный промежут. Эту проблемму я решил введением дополнительной переменной *currentID* типа *long*, после зачитывания которой, я проверяю, принадлежит ли ключ моему заданному промежутку.

Также я забыл добавить проверку того, является ли список элементов пустым, чтобы это исправить я ввел переменную *isFirst*, если после завершения цикла считывания ее значение осталось равным нулю, я зпавершаю программу.

Строку я сразу зачитывал в массив, но так как размер строки неизвестен заранее, выделение памяти происходило нерационально, и я начал уменьшать размер массива, в котором храниться строка до необходимого размера.

При передаче массива спиков из функции сортировку в функцию вывода я его копировал, что существенно замедляло работу программы и использовало лишнюю память. Для того, чтобы это исправить, я создаю массив а *main* и передаю его по ссылке.

Выводы

Данный алгоритм оптимален для сортировки ключей в небольшом диапазоне, при частом повторении ключей. Скорость работы сортировки повышена за счёт увеличения количества используемой памяти. Поэтому в случаях, где количество используемой памяти критичнее скорости работы, данную сортировку применять не целесообразно. Типовые задачи сортировки: сортировка студентов, по полученной ими оценке, сортировка большего числа людей по их росту с округлением до целых, и любые другие случаи с небольшим диапазоном ключей и высокой кучностью. Также сортировку подсчётом удобно внедрять в сортировку вставкой. Сортировка работает за $O(n + k)$. Где n — количество элементов и k — диапазон их значений. Сортировка подсчётом проста в реализации. И хотя я написал не классическую ее реализацию, сделать это было несложно. Основная сложность лабораторной работы заключается в обработке исключений и оптимизации работы программы.