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

Выполнил студент группы М80-208Б-18 МАИ *Синявский Андрей.*

Условие

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

2. Вариант 5-3.

   Поразрядная сортировка.

   Тип ключа: MD5-суммы (32-разрядные шестнадцатиричные числа).

   Тип значения: числа от 0 до 264 - 1.

Метод решения

1. Данные на вход программе подаются через стандартный поток ввода, так что наиболее лаконичным решением на мой взгляд будет считывание циклом **while** (пока значение может быть прочитано, продолжать цикл).

   ```
   while(std::cin >> tmp.key && std::cin >> tmp.val)
   ```

2. Так как количесво данных заранее не известно, необходимо будет реализовать вектор (по условию задания стандартный вектор из библиотеки STL использовать запрещается).

3. Ключи будут представляться в виде символьных строк, так что для корректной работы алгоритма сортировки необходимо реализовать функцию, переводящую символ в число (0 в 0, 5 в 5, b в 11 и т. д.).

4. Алгоритм поразрядной сортировки принимает на вход вектор со входными значениями, и вызывает сортировку подсчётом от последнего до первого элемента ключа.

5. Сортировка подсчётом принимает тот же вектор входных данных, а также создаёт 2 новых массива: **res**, в который будет записана отсортированная последовательность, и **count**, в котором подсчитывается число повторений каждого возможного символа (всего 16, 10 цифр и 6 букв) в сортируемом разряде ключа.

Описание программы

На каждой непустой строке входного файла располагается пара «ключ-значение», поэтому создадим новую структуру **TData**, в которой будем хранить ключ и значение.

А также мы не знаем количество входных данных, поэтому мы напишем динамический массив - вектор, в который будут помещаться структуры **TData**.

main.cpp	
Тип данных	Значение
struct TData	Структура для хранения пары "ключ-значение"
class TVector	Вектор для хранения структур **TData**
Функция	Значение
TVector()	Конструктор класса **TVector**
TVector($size_t$)	Конструктор класса **TVector**
$\sim$ TVector()	Деструктор класса **TVector**
void Add(TData)	Добавить элемент в конец вектора
void CountingSort(size_t)	Функция сортировки подсчётом
void RadixSort()	Функция порарядной сортировки
int CharToNum(size_t i, int cInd)	Функция, преобразующая символ в число
int main()	Главная функция, в которой происходит чтение данных, вызов функции сортировки и вывод.

Дневник отладки

При создании этой таблицы была использована история посылок.

Время	Проблема	Описание
2019/10/09 21:47:00	Ошибка выполнения	Оказалось, что я неверно истолковал задание и сделал ввод данных через входной файл, указываемый в аргументах программы.
2019/10/10 20:59:26	Превышено реальное время работы	Выделение памяти в векторе происходило при каждом добавлении нового элемента. Решено введением вместимости вектора, которая удваивается при достижении размера вектора предела вместимости.
2019/10/10 21:08:10	Превышено реальное время работы	Проблема в неоптимизированном выводе. Использованные методы оптимизации: 1) Опция std::ios_base::sync_with_stdio(false), которая отключает синхронизацию потоков Си и C++, т. е. данные перестат копироваться из буфера в буфер. 2) Опция std::cin.tie(nullptr). Она отделяет изначально связанные потоки cin и cout. При их связи один из потоков сбрасывается перед каждой операцией в другом потоке.
2019/10/11 08:34:07	Работает	Далее убирались комментарии и лишние функции, используемые только для отладки программы, вводились незначительные косметические изменения.

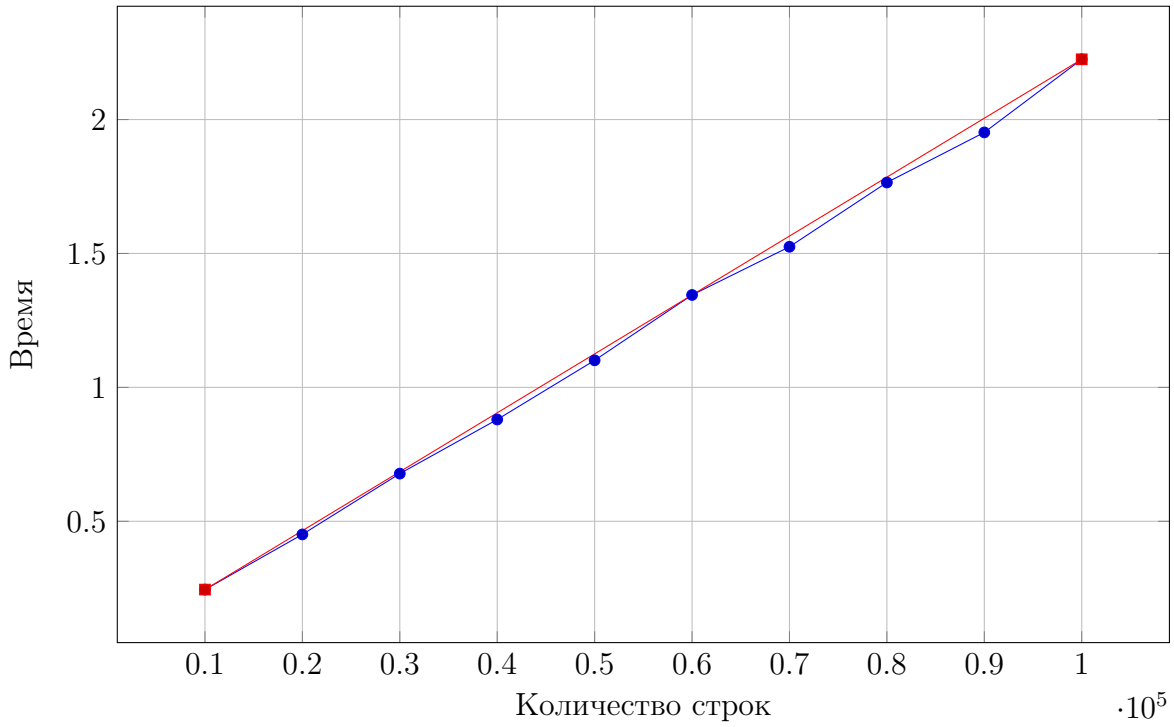
Тест производительности

Для генерации тестов использовалась следующая программа:

```
#include <iostream>
#include <fstream>
#include <ctime>
#include <cmath>
#include <cstdlib>

int main(int argc, char *argv[]) {
    std::ofstream file(argv[1]);
    srand(time(0));
    size_t size = atoi(argv[2]);
    char key[33];
    int flag;
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < 32; j++) {
            flag = rand() % 2;
```

```
            if (flag) {
                key[j] = (char) (rand() % 10 + 48);
            } else {
                key[j] = (char) (rand() % 6 + 97);
            }
        }
        key[32] = '\0';
        file << key << "\t" << abs(rand()) << "\n";
    }
    return 0;
```



Недочёты

Параметр вместимости у вектора, показавшийся мне лишним, как оказаось, в разы ускоряет работу программы.

Выводы

Данную программу можно использовать для сортировки файлов по их MD5-суммам, где сумма является ключом, а имя файла - значением.

В процессе работы я получил ценный опыт отладки программы и её оптимизации при помощи утилит valgrind и time, углубил и расширил своё понимание принципов языка C++.