# Московский авиационный институт (национальный исследовательский университет)

## Факультет информационных технологий и прикладной математики

## Кафедра вычислительной математики и программирования


**Лабораторная работа №1 по курсу «Дискретный анализ»**


Студент: М. Ю. Курносов
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б

Дата:
Оценка:
Подпись:

**Москва, 2025**

# Лабораторная работа №1

**Задача:** 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:

**Вариант сортировки:** Поразрядная сортировка.

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

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

# 1 Описание

Требуется написать реализацию алгоритма поразрядной сортировки.
Основная идея поразрядной сортировки[4] заключается в том, чтобы сравнить все разряды элементов входного массива, и отсортировав все разряды элементов сортировкой№2[2] отсортировать сам входной массив.. [1].

# 2 Исходный код

Программа считывает значения из файла, подающегося на поток, сортирует их поразрядной сортировкой и выводит в поток вывода.

На каждой непустой строке входного файла располагается пара «ключ-значение», поэтому создадим новую структуру $TKeyValuePair$, в которой будем хранить ключ и значение. Для хранения пар будем использовать динамический массив $pairs$ ($TKeyValuePair[]$). Для сортировки создадим функцию $RadixSort$ . После выполнения программа выводит отсортированные пары в том же виде что и принимала(на каждой строке ключ-значение).

main3.0.cpp:

```
1  // This is a personal academic project. Dear PVS-Studio, please check it.
2  // PVS-Studio Static Code Analyzer for C, C++ and C#: http://www.viva64.com
3  #include <iostream>
4  #include <cstring>
5  #include <cstdlib>
6
7  const int NUMBER_OF_DIGITS = 6;
8  const int MAX_DIGIT_VALUE = 256;
9  const int MAX_STRING_LENGTH = 2048;
10
11 struct TKeyValuePair {
12     char number[NUMBER_OF_DIGITS];
13     char* value;
14
15     TKeyValuePair() : value(nullptr) {
16         memset(number, ' ', NUMBER_OF_DIGITS);
17     }
18
19     TKeyValuePair(const char* num, const char* val) {
20         int j = 0;
21         for (int i = 0; i < strlen(num) && j < NUMBER_OF_DIGITS; i++) {
22             if (num[i] != ' ') {
23                 number[j++] = num[i];
24             }
25         }
26         while (j < NUMBER_OF_DIGITS) {
27             number[j++] = ' ';
28         }
29
30         value = new char[strlen(val) + 1];
31         strcpy(value, val);
32     }
33
34     ~TKeyValuePair() {
35         delete[] value;
36     }
```

```
37
38      TKeyValuePair(const TKeyValuePair& other) {
39          memcpy(number, other.number, NUMBER_OF_DIGITS);
40          value = new char[strlen(other.value) + 1];
41          strcpy(value, other.value);
42      }
43
44      TKeyValuePair& operator=(const TKeyValuePair& other) {
45          if (this != &other) {
46              delete[] value;
47              memcpy(number, other.number, NUMBER_OF_DIGITS);
48              value = new char[strlen(other.value) + 1];
49              strcpy(value, other.value);
50          }
51          return *this;
52      }
53  };
54
55  void RadixSort(TKeyValuePair* pairs, size_t size) {
56      int countArray[MAX_DIGIT_VALUE];
57      TKeyValuePair* tempArray = new TKeyValuePair[size];
58
59      for (int pos = NUMBER_OF_DIGITS - 1; pos >= 0; pos--) {
60          memset(countArray, 0, sizeof(countArray));
61
62          for (size_t i = 0; i < size; i++) {
63              unsigned char c = pairs[i].number[pos];
64              countArray[c]++;
65          }
66
67          for (int i = 1; i < MAX_DIGIT_VALUE; i++) {
68              countArray[i] += countArray[i-1];
69          }
70
71          for (int i = size - 1; i >= 0; i--) {
72              unsigned char c = pairs[i].number[pos];
73              tempArray[--countArray[c]] = pairs[i];
74          }
75
76          for (size_t i = 0; i < size; i++) {
77              pairs[i] = tempArray[i];
78          }
79      }
80      delete[] tempArray;
81  }
82
83  int main() {
84      std::ios::sync_with_stdio(false);
85      std::cin.tie(nullptr);
```

```
86
87      size_t capacity = 16;
88      size_t size = 0;
89      TKeyValuePair* pairs = new TKeyValuePair[capacity];
90
91      char line[MAX_STRING_LENGTH * 2];
92      while (std::cin.getline(line, sizeof(line))) {
93          if (strlen(line) == 0) continue;
94
95          if (size >= capacity) {
96              capacity *= 2;
97              TKeyValuePair* newPairs = new TKeyValuePair[capacity];
98              for (size_t i = 0; i < size; i++) {
99                  newPairs[i] = pairs[i];
100             }
101             delete[] pairs;
102             pairs = newPairs;
103         }
104
105         char* delimiter = strrchr(line, '\t');
106         if (!delimiter) {
107             char* last_space = nullptr;
108             char* p = line;
109             while (*p) {
110                 if (*p == ' ' && *(p+1) != ' ' && *(p+1) != '\0') {
111                     last_space = p;
112                 }
113                 p++;
114             }
115             delimiter = last_space;
116         }
117
118         if (!delimiter) continue;
119
120         *delimiter = '\0';
121         char* number = line;
122         char* value = delimiter + 1;
123
124         while (*value == ' ') value++;
125
126         pairs[size] = TKeyValuePair(number, value);
127         size++;
128     }
129     RadixSort(pairs, size);
130
131     //
132     for (size_t i = 0; i < size; i++) {
133         std::cout << pairs[i].number[0] << ' '
134                 << pairs[i].number[1] << pairs[i].number[2] << pairs[i].number[3] << '
```

```
                    ’
                << pairs[i].number[4] << pairs[i].number[5] << ’\t’
                << pairs[i].value << ’\n’;
    }

    delete[] pairs;
    return 0;
}
```

| main3.0.cpp | |
|---|---|
| struct TKeyValuePair | Структура, которую я использую для хранения пар ключ-значение |
| TKeyValuePair() | Конструктор |
| TKeyValuePair(const char* num, const char* val) | Второй конструктор, работающий со входными значениями типов const char* |
| TKeyValuePair(const TKeyValuePair& other) | Конструктор копирования для класса TKeyValuePair |
| TKeyValuePair& operator=(const TKeyValuePair& other) | перегрузка оператора присваивания |
| ∼TKeyValuePair() | Деструктор класса TKeyValuePair |
| void RadixSort(TKeyValuePair* pairs, size$_t size$) | функция-сортировка по разрядам |
| int main() | Основная часть программы. |

# 3 Консоль

```
me@DESKTOP:/mnt/c/Users/Max Kar/Desktop/МАИ/ДА/lab1.25$ g++ -o p main3.0.cpp
me@DESKTOP:/mnt/c/Users/Max Kar/Desktop/МАИ/ДА/lab1.25$ ./p <input.txt
0        TdWYYJgL4MuXq4OxXeq2xxmc7LbvV5Zyg3VCK9VNVPuKUIHzu81q3lSA436Z7dWle1yxATKerEyJV
A 035 JR spexJSsxTEfeoCfvFQD1T4VdRyPKPKcF7GCOg3MAHzm3BPyPq9QHEvudtJyGbYUime7ShRawQMxar
A 045 YF pDcQAkzjFUdNbwRZEjwSzElCcJoNM3P9HOZPiW6x1jIcfi9rR5HOH1auKMHeQeofeL4MsAHrraRUI
A 065 SS coAg1dsYMc8RUDzrXGBx6pznM1Y4LXBvLJZLwRtJ2zkWDkOk0Zh4MgsifOlyvwtEFRXAsOTuOBRbt
A 068 YD bRpWAYihIoWEw9NUFR1fLwbIdTNH872hWpEgLuLcjsqd1B7Ea6uu2VCgyZv4evm9j0n4s9gbzUEyc
A 093 JZ IrbxKRrsOHg6l3mPbhcAWP2G8guDPnFfcqcwGUnejTiVUVs3CSDgrGuxwo9JcOzEDZ9T1u5lLoEpk
A 121 KK KsDmjA32rK57yWdjXodzwc5DGyv0L0ydsBPZLQbAkfHi9sRge3eadhltfgryepbWy03KPfS9IiqRa
A 171 HG irkYnLT3jAEbXrLPXnqveH1CE31zMGy25ibs14vkC8Kjzd7UOxQ3DRFRSEQoSOrX4SN3UHogP8NMj
A 172 IN 5yBknsQV7wL5kJ1nlNwjXP6SN6kOKYcPXl9IeYljU7oCOo09BwqgKx8f1t3LPdjuPqC1Owkr1X3Pk
A 208 LS 2TKmmwr6fVWVUDhjRp4KichWzUmSbaud2ENmAFsnkNHCayv1nzJUZy0YTly4JsfL607FFy2xLHAvC
A 212 IR g4Gvu85OAO8wd52984akaiv1gitXprzTuDMoLSAVoGPRJSYRW695o26Vk02XpzzjDLVWnf0bvP1Cp
A 275 LP KNWY8xRkJN6aoEEUB6YEseNObAAyTAelXBHf6hNP4RxqdBLpGr38WOW5Ye1zofiJqxwueHKgiHXMP
A 286 SR wFWbKD4e5ukOVATYEpqfc9TZw0xgtBQnQxPk8RNCM5apF3LUqB7RIa0EYxvQ7JDXGaGP2dZMg9BwP
A 290 OV yxFKaLMZsoVuzxHbCByAAZhWqv7gp1Snyi5X1R6tDblDW2miBiqLHYr7TylG0D4wt9RsZXlm6UxcV
A 300 PN m8FoWxw61Cx37USChgScvxkB48sbAUpwa4i7zeD1o92udU6I8Zu1UeDYn57xXwr7yZEyBQz0ZzuAT
A 342 YN ybQUzojDjAUZHxeWBLvpT4r31yraYWKW7ky5WhGGrkn6iSctlVgCZYDZU392ZTWeBUji9zw0ij6OS
A 353 WF KuT7Gaw9Ca1bkgtlPHct2UwMm24EeRHwMk2aIyjSYk3HRw2oCfhE7dYudc6F2NCM6EwOAgrgOuxnp
A 409 MF vzSGwqXOUd5PgdK80wanveXB8r1I06ev369zxgLPHRpv27224cozEJ8MA7c9BE4FJECGsYf8zS3za
A 426 PA iF5dTQOSH8wOOROQHX7FjkuNc7l8nBwTQ16tQ4KhCEhag6zxb6ALq2gT7RZuaVNOURtuVBaiQFG6J
A 445 PO yVpfbENLvyIa28WY9htJpztdLbj1HcfG7TvghG2bEIBERgkYNdrCbjqwKXvZAbpF4iwjywIAEUOdA
A 452 MA GXpZyVnksGtDUZCwvsxlfIGDsgbAhp3xLrVHKI0AWrNzQZvLPs439KG11qBgfDby26EMMEWr5rqUC
A 625 AY okomhtAxCVbWghaSz5tp2JBdXwtLsy1ggpRNgbIt5rNjXvBU13K3MTgrQYDGWCxDzOYgxpX0gsjDn
A 690 WM OYkw7WEXHGx1rFCO5D7uqKsWmdVTzxSNTCHYhT3yj00ZDCxHO2BCK2i7fBYc60zZAE5rh6oP6oyIy
A 706 OL lCUoVhhAXvMQKOOwXcEVHWHEAMr6XLHIXj72PmAwfWMxXiu4L6ZccrqkBfqh05xVn4VCpg8UCSShA
A 751 LT r1w1n3wdus9fsNrPb3bX6e0Xu4VCb5wS6sTrtNTnEcS4xITYJ45Qh4vb8PlhUf9YVaNOwqBAQcENu
A 807 SY fsMXk50Noe1ulAKbRKMt0paOtc0B1rQeklCUqApemoWXyo7P8TH6HrU8RSISIi72RHUFPHrB6Ng4P
A 846 AR Zg9vwiBFSNiha3M3BpltkHkq8PjECf8lJHeFxpUNBA3lEPmNDXExmzouMV6Z8FKRUzhSm9nvKqeWP
A 857 AN woLoqNWjLc9pzbGDuiyBy43zXvVlhGxd5JRvgvezWmoVN3gFlcOjgShBLAw2RudWB3Ppy1oSnby9e
A 864 CL eVjnOCJZX651SazHThFUhTLOXxI152ZjYIUuUoT1sW0J50aYfn2NHLlmI1lL3K3ZbVT3Hu38R3RW3
A 879 KE gGmB59dhDxKckVu2BMoLrYWQ6n3slu9PAvaF3DvG8Dssimsr6gCxCiOIWP9FJIfSEDfHRYXXlNQTS
A 902 PQ cG9gtQhVC8YXrIhEO0mDPSRrF4reH3irKpXDFDjPJFxAVeNtf762ZVtmaiQpm6g4vCH8NyYgBVpgT
U 749 UM uJWvNwJVlZD2ksU64wOH3E1yVpWg1FnwWHptB8OufZwPSPVWJulN6kJbZqFb32VaIKTTRqM6NIUpi
U 805 OQ DXJOCNCuYTQQ82k6M1w4CgYJ9B2P0IhDn0ZxLlrrFFGNIyScOOhA2DUAMWZMmGYaE7VXtNO6aeTqk
U 805 OQ YkLGqGVWpyO5SWxGA7vVHvWihZOrlcmHM5VCL0jBy7GPbDfmJbFYUjG9Ie01GlIdqnnAmWLicZ7Bl
U 805 OQ nQUMgYUFsYQ399ThqMOZs42W0MAlBqsyEKKssn7iJWlSfC7UYV1QX4xYO5JXt9U6SmyIZ41samIDz
U 884 XR I2tGxM1eMS4KaAErASmcvYJD6kzPb4Ds458zR7elZg58qJzzjlZcHspNbpkArw2t18tQFVBpBEv0W
U 888 WC oNvW4cdM8QqYqbeLcbN30cmXIdnIHfj40ca5EDRMbHsPqWlT78W5iIb0tMJ922D2fl7rxWCWl2wcZ
```

```
U 891 BA HD7P0yAxUFIWHcxTPOyC7dLOd6ws2R3HeBfc7pZa5p6MS3ppPl2VNLt0RpqREuhq3MS8B0iEnpZDc
U 939 LW ABqsMr87dh3DATMlqsRq94ZzrvE0ivUq5KjR9rYmWZxg0JSqCreJuEGl7SlnMDeRXNsfCORivPNvg
U 950 ZQ x5kmr0JkiuB2TtS9TbZWmspfzGH92ibxnJjcH0K0uW2MNSVo32Jnv8SsOk1OQdKDu3nC47CwbEGyg
V 002 EP K6UtsWAWxNFmoOHJj1vh17hVla8xHapbeHTWld1gyGQkci4Ljx2i2hCmHKjZtW8Xnb2XC3D8HettP
```

# 4 Тест производительности

*Сравнение Моей реализации с уже существующей.*

Тест производительности представляет из себя следующее: сортировка двух динамических массивов с помощью функции, созданной в файле main3.0.cpp, и стандартной функции сортировки.

```
me@DESKTOP:/mnt/c/Users/Max Kar/Desktop/МАИ/ДА/lab1.25$ ./t 1000 >input.txt
me@DESKTOP:/mnt/c/Users/Max Kar/Desktop/МАИ/ДА/lab1.25$ ./p <input.txt
Radix sort time: 4401us
STL stable sort time: 2164us
Input array size: 1000
me@DESKTOP:/mnt/c/Users/Max Kar/Desktop/МАИ/ДА/lab1.25$ ./t 10000 >input.txt
me@DESKTOP:/mnt/c/Users/Max Kar/Desktop/МАИ/ДА/lab1.25$ ./p <input.txt
Radix sort time: 81708us
STL stable sort time: 41788us
Input array size: 10000
me@DESKTOP:/mnt/c/Users/Max Kar/Desktop/МАИ/ДА/lab1.25$ ./t 1000000 >input.txt
^[[Ame@DESKTOP:/mnt/c/Users/Max Kar/Desktop/МАИ/ДА/lab1.25$ ./p <input.txt
Radix sort time: 11226569us
STL stable sort time: 7176365us
Input array size: 1000000
```

Как видно, сортировка из стандартной библиотеки выигрывает у поразрядной сортировки, так как она оптимальнее реализована. Моя реализация медленнее из-за: избыточных операций копирования, работы с неоптимальными типами данных, нагрузки на систему управления памятью Для строк фиксированной длины Radix Sort действительно должен быть быстрее, но только при правильной оптимизированной реализации и на достаточно больших массивах данных. Как видно из теста производительности при увеличении массива данных разрыв в скорости работы сортировок сокращается.

# 5 Выводы

В результате выполнения лабораторной работы по курсу «Дискретный анализ», были получены навыки использования поразрядной сортировки, работы с командами new и delete, работы со строками при считывании из файла, и закреплены навыки использования классов.

# Список литературы


[1] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. *Алгоритмы: построение и анализ, 2-е издание.* — Издательский дом «Вильямс», 2007. Перевод с английского: И. В. Красиков, Н. А. Орехова, В. Н. Романов. — 1296 с. (ISBN 5-8459-0857-4 (рус.))

[2] *Сортировка подсчётом — Википедия.*
URL: `http://ru.wikipedia.org/wiki/Сортировка_подсчётом` (дата обращения: 14.10.2020).

[3] *ГОСТ Р 7.05-2008*
URL: `https://internet-law.ru/gosts/gost/44298`(дата обращения: 14.10.2020).

[4] *Поразрядная сортировка — Википедия.*
URL: `https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D0%B0%D0%B7%D1%80%D1%8F%D0%B4%D0%BD%D0%B0%D1%8F_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0`(дата обращения: 14.10.2020).