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: Т. А. Габдуллин
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б

Дата:
Оценка:
Подпись:

Москва, 2019

Лабораторная работа №1

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

Тип ключа: 7. Автомобильные номера в формате A 999 BC (используются буквы латинского алфавита).

Тип значений: 1. Строки фиксированной длины 64 символа, во входных данных могут встретиться строки меньшей длины, при этом строка дополняется до 64-х нулевыми символами, которые не выводятся на экран.

Тип сортировки: Поразрядная сортировка(Radix sort).

Входные данные: На каждой непустой строке входного файла располагается пара «ключ-значение», в которой ключ указан согласно заданию, затем следует знак табуляции и указано соответствующие значение.

Выходные данные: Выходные данные состоят из тех же строк, что и входные, за исключением пустых и порядка следования.

1 Метод решения

Имеем множество последовательностей одинаковой длины, состоящих из элементов, на которых задано отношение линейного порядка. Требуется отсортировать эти последовательности в лексикографическом порядке.

По аналогии с разрядами чисел будем называть элементы, из которых состоят сортируемые объекты, разрядами. Сам алгоритм состоит в последовательной сортировке объектов какой-либо устойчивой сортировкой по каждому разряду, в порядке от младшего разряда к старшему, после чего последовательности будут расположены в требуемом порядке. **Линейная оценка:** $\emptyset(n)$
Описание взято с сайта *neerc.ifmo.ru*.

2 Листинг

1. main.cpp

```
#include "TStruct.h"
#include "TNumVec.h"
#include "TParser.h"
#include "MyRadixSort.h"
#include <stdio.h>

void PrintKeys(TNumVec * vec);
int main() {
    TNumVec myVec;
    InitTNumVec(&myVec);
    unsigned int number = 0;
    char key[KEY_LEN];
    char string[VALUE_LEN + SPACE_FOR_NULL];
    while (scanf("%8c\t%s", key, string) == 2) {
        number = ReadKey(key);
        int length = MyStrlen(string);
        if (length != VALUE_LEN) {
            for (int i = length; i < VALUE_LEN; i++){
                string[i] = '\0';
            }
        }
        AddItem(&myVec, number, string);
        string[0] = '\0';
        getchar();
    }
    MyRadixSort(&(myVec.items), &myVec.size);
    PrintKeys(&myVec);
    DeleteTNumVec(&myVec);
    return 0;
}

void PrintKeys(TNumVec * vec) {
    for (unsigned int i = 0; i < vec->size; i++) {
        PrintCarPlateNumber(GetItem(vec,i).key);
        PrintValue((GetItem(vec, i).value));
    }
}
```

2. TParser.cpp

```
#include "TNumVec.h"

static void ReallocTNumVec(TNumVec * vec);
static void MyStrcpy(char * output, char * input);

void InitTNumVec(TNumVec * vec) {
```

```
7     unsigned int n = 1;
8     vec->items = (TItem*) malloc(n * sizeof(TItem));
9     vec->capacity = 1;
10    vec->size = 0;
11 }
12 static void ReallocTNumVec(TNumVec * vec) {
13    vec->capacity = ((vec->size + 1) * 2);
14    vec->items = (TItem *) realloc(vec->items, vec->capacity * sizeof(TItem));
15 }
16 static void MyStrcpy(char * output, char * input) {
17    while (*input) {
18        *output++ = *input++;
19    }
20    *output = '\0';
21 }
22 void AddItem(TNumVec * vec, unsigned int number, char * string) {
23    if (vec->size == vec->capacity) {
24        ReallocTNumVec(vec);
25    }
26    vec->items[vec->size].key = number;
27    MyStrcpy(vec->items[vec->size].value, string);
28    vec->size += 1;
29 }
30 TItem GetItem(TNumVec * vec, const int i) {
31    return vec->items[i];
32 }
33 void DeleteTNumVec(TNumVec * vec) {
34    vec->capacity = 0;
35    vec->size = 0;
36    free(vec->items);
37 }
```

3. TNumVec.cpp

```
1  #include "TParser.h"
2  static bool MyIsDigit(const char symbol) {
3     return '0' <= symbol && symbol <= '9';
4  }
5  unsigned int ReadKey(char * key) {
6     unsigned int number = 0;
7     int digit = 10000000;
8     for (int i = 0; i < KEY_LEN; i++) {
9        if (key[i] == ' ') {
10           continue;
11       }
12       if (MyIsDigit(key[i])) {
13           number += (key[i] - '0') * digit;
14       } else {
15           number += (key[i]) * digit;
16       }
```

```
17          digit = digit / 10;
18          if (i == 4 || i == 6) {
19              digit = digit / 10;
20          }
21      }
22      return number;
23  }
24  int MyStrlen(const char *string) {
25      register const char *pStr;
26
27      for (pStr = string; *pStr; ++pStr);
28      return(pStr - string);
29  }
30  void PrintCarPlateNumber(unsigned int number) {
31      char output[KEY_LEN - 2];
32      int digit = 10000000;
33      for (int i = 0; i < KEY_LEN - 2 ; i++) {
34      if ((i >= 1) && (i <= 3 )){
35          output[i] = (char) (number / digit) + '0';
36          } else {
37          output[i] = (char) (number / digit);
38      }
39          number = number % digit;
40          digit = digit / 10;
41          if (i == 3 || i == 4) {
42              digit = digit / 10;
43          }
44      }
45      for (int i = 0; i < KEY_LEN - 2; i++) {
46          if (i == 1 || i == 4) {
47              printf(" ");
48          }
49          printf("%c",output[i]);
50      }
51      printf("\t");
52  }
53  void PrintValue(const char * value) {
54      puts(value);
55  }
```

3 Выводы

Благодаря данной лабораторной работе, я закрепил навыки работы с произвольными структурами и приобрел опыт сортировки таких структур.
Примерами объектов, которые удобно разбивать на разряды и сортировать по ним(Radix sort), являются числа и строки.

- Для чисел уже существует понятие разряда, поэтому будем представлять числа как последовательности разрядов. Конечно, в разных системах счисления разряды одного и того же числа отличаются, поэтому перед сортировкой представим числа в удобной для нас системе счисления.
- Строки представляют из себя последовательности символов, поэтому в качестве разрядов в данном случае выступают отдельные символы, сравнение которых обычно происходит по соответствующим им кодам из таблицы кодировок. Для такого разбиения самый младший разряд — последний символ строки.

Для вышеперечисленных объектов наиболее часто в качестве устойчивой сортировки применяют сортировку подсчетом.
Такой подход к алгоритму называют *LSD сортировкой*

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] *Radix-sort*
URL:`http://neerc.ifmo.ru/wiki/index.php?title=Radix-sort`