Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовая работа**

**по курсу «Языки и методы программирования»**

**ll Семестр**

**Задание 9**

**Сортировка и поиск**

|  |  |
| --- | --- |
| Студент: | Аксенов А.Е. |
| Группа: | М8О-108Б-18 |
| Преподава-тель: | Поповкин А.В. |
| Оценка: |  |
| Дата: | **2019** |

# Содержание

[Задание 3](#__RefHeading___Toc1230_674285793)

[Общий метод решения 3](#__RefHeading___Toc1232_674285793)

[Общие сведения о программе 4](#__RefHeading___Toc1234_674285793)

[Текст программы 4](#__RefHeading___Toc1236_674285793)

[Распечатка протокола 7](#__RefHeading___Toc1238_674285793)

[Журнал отладки 9](#__RefHeading___Toc1240_674285793)

[Заключение 10](#__RefHeading___Toc1242_674285793)

[Список используемых источников 11](#__RefHeading___Toc1244_674285793)

# Задание

Составить программу на языке Си с использованием процедур и функций для сортировки таблицы заданным методом и двоичного поиска по ключу в таблице. Программа должна вводить значения элементов неупорядоченной таблицы и проверять работу процедуры сортировки в трех случаях:

1. элементы таблицы с самого начала упорядочены;
2. элементы таблицы расставлены в обратном порядке;
3. элементы таблицы не упорядочены.

В последнем случае можно использовать встроенные процедуры генерации псевдослучайных чисел.

Для каждого вызова процедуры сортировки необходимо печатать исходное состояние таблицы и результаты сортировки. После выполнения сортировки программа должна вводить ключи и для каждого из них выполнять поиск в упорядоченной таблице с помощью процедуры двоичного поиска и печатать найденные элементы, если они присутствуют в таблице.

*Вариант N=1.*

Вариант задания определяется двумя числами: (1) - номер метода сортировки = ((N - 1) % 11) + 1, (2) – номер структуры таблицы = ((N + 6) % 9) + 1, где N - номер cтудента по списку в группе.

*Метод сортировки (Простое двупоточное слияние):*

*Структура таблицы:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *№* | *Тип ключа* | *Длина ключа в байтах* | *Хранение данных и ключей* | *Число элементов таблицы* |
| 8 | комбинированный(строка + целое) | 8 | вместе | 8-10 |

# Общий метод решения

Строки заданной таблицы сортируются по ключу методом простого двупоточного слияния.

Слияние означает объединение двух или более упорядоченных массивов в один упорядоченный массив. Например, слияние двух массивов – [5 8 9] и [1 3 4 10] дает массив [1 3 4 5 8 9 10]. Сделать это можно, сравнив два наименьших элемента, вывести наименьший из них, а затем повторить эту процедуру. Необходимо позаботиться о действиях на случай когда один из массивов будет исчерпан.

Поиск строки таблицы по ключу реализуется методом бинарного поиска.

В программе предусмотрено случайное перемешивание строк таблицы (sort -R), а также перестановка строк таблицы в обратном порядке (sort -r).

В программе также предусмотрен запуск исполняемого файла с параметром - именем файла, содержащего исходную таблицу. Если этот параметр не указан, то по умолчанию предполагается файл с именем data.txt.

# **Общие сведения о программе**

**Язык и система программирования:** GNU C

**Местонахождение файлов: ~/course9/**

**Способ компиляции и запуска:**

makefile && ./bin/main data.txt

# Текст программы

***data.h:***

#ifndef DATA\_H

#define DATA\_H

typedef struct {

int key2;

char key1[5];

char \*value;

} data;

void data\_sort(data \*d, size\_t l);

#endif

***data.c:***

#include <stdlib.h>

#include <pthread.h>

#include <string.h>

#include "data.h"

typedef struct {

data \*d;

size\_t len;

} astruct;

int is\_less(data \*s1, data \*s2) {

int res = strcmp(s1->key1, s2->key1);

if (res == 0)

res = s1->key2 < s2->key2;

else

res = res == -1 ? 1 : 0;

return res;

}

data \*merge(astruct \*s1, astruct \*s2){

data \*m = NULL;

if (s1 && s2){

m = (data \*)realloc(m, sizeof(data) \* (s1->len + s2->len));

size\_t l1 = 0, l2 = 0;

while (l1 < s1->len || l2 < s2->len) {

if (s1->len > l1 && (s2->len >= l2 || (l2 < s2->len && is\_less(s1->d + l1, s2->d + l2)))) {

m[l1 + l2] = s1->d[l1];

//printf("%s\n", s1->d[l1].key1);

//printf("m %s\n", m[l1 + l2].key1);

l1++;

//printf("%s\n", s1->d[l1].key1);

}

if (s2->len > l2 && (s1->len >= l1 || (l1 < s1->len && is\_less(s2->d + l2, s1->d + l1)))) {

m[l1 + l2] = s2->d[l2];

//printf("%s\n", s1->d[l2].key1);

//printf("m %s\n", m[l1 + l2].key1);

l2++;

//printf("%s\n", s1->d[l1].key1);

}

}

}

return m;

}

void \*data\_msort(void \*vs){

astruct \*s = (astruct \*)vs;

if (s && s->len > 1) {

pthread\_t pid1, pid2;

astruct st1 = {s->d, s->len / 2};

astruct st2 = {s->d + s->len / 2, s->len - (s->len / 2)};

pthread\_create(&pid1, NULL, data\_msort, &st1);

pthread\_create(&pid2, NULL, data\_msort, &st2);

pthread\_join(pid1, NULL);

pthread\_join(pid2, NULL);

data \*n = merge(&st1, &st2);

memcpy(s->d, n, s->len);

}

return NULL;

}

void data\_sort(data \*d, size\_t l) {

astruct st = {d, l};

data\_msort(&st);

}

***main.c:***

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include "data.h"  
  
#define MAX\_VLEN 1000  
  
int data\_read(char \*fname, data \*\*d, size\_t \*len) {  
 FILE \*fin = fname ? fopen(fname, "r") : stdin;  
 int err = 0;  
 if (d && len) {  
 \*d = NULL;  
 \*len = 0;  
 }  
 if (fin) {  
 if (d) {  
 int key2;  
 char key1[5];  
 char value[MAX\_VLEN];  
 while (fscanf(fin, "%s%d", key1, &key2) == 2) {  
 fgets(value, MAX\_VLEN, fin);  
 \*d = (data \*)realloc(\*d, sizeof(data) \* ++\*len);  
 (\*d)[\*len - 1].value = strdup(value);  
 (\*d)[\*len - 1].key2 = key2;  
 //memcpy((\*d)[\*len - 1].key1, key1, 4);  
 strcpy((\*d)[\*len - 1].key1, key1);  
 }  
 }  
 if (fin != stdin)  
 fclose(fin);  
 }  
 else  
 err = 1;  
 return err;  
}  
  
void data\_print(data \*d, size\_t len){  
 if (d){  
 size\_t i;  
 printf ("Key |Value\n");  
 printf ("------|----------------------\n");  
 for (i = 0; i < len; i++) {  
 printf("%4s %d|%s", d[i].key1, d[i].key2, d[i].value);  
 }  
 }  
}  
  
void program(char \*name) {  
 data \*d;  
 size\_t len;  
 printf("LAUNCHING PROGRAM...\n");  
 printf("READING DATA FROM FILE...\n");  
 data\_read(name, &d, &len);  
 printf("PRINTING DATA...\n");  
 data\_print(d, len);  
 printf("SORTING...");  
 data\_sort(d, len);  
 printf("DONE\n");  
 printf("PRINTING SORTED DATA...\n");  
 data\_print(d, len);  
 printf("PROGRAM COMPLETE\n");  
}  
  
int main(int argc, char \*\*argv) {  
 if (argc > 1)  
 program(argv[1]);  
 return 0;  
}

***data.txt:***

abcd 1 \_.--""--.\_

aceg 2 / \_ \_ \

adma 3 \_ ( (\_\ /\_) ) \_

ansi 9 { \.\_\ /\ /\_./ }

hjkl 1 /\_"=-.}\_\_\_\_\_\_{.-="\_\

hjkl 8 \_ \_.=("""")=.\_ \_

vt10 0 (\_'"\_.-"`~~`"-.\_"'\_)

wyse 5 {\_" "\_}

***data\_rev.txt:***

wyse 5 {\_" "\_}

vt10 0 (\_'"\_.-"`~~`"-.\_"'\_)

hjkl 8 \_ \_.=("""")=.\_ \_

hjkl 1 /\_"=-.}\_\_\_\_\_\_{.-="\_\

ansi 9 { \.\_\ /\ /\_./ }

adma 3 \_ ( (\_\ /\_) ) \_

aceg 2 / \_ \_ \

abcd 1 \_.--""--.\_

***data\_rand.txt:***

abcd 1 \_.--""--.\_

wyse 5 {\_" "\_}

adma 3 \_ ( (\_\ /\_) ) \_

hjkl 1 /\_"=-.}\_\_\_\_\_\_{.-="\_\

ansi 9 { \.\_\ /\ /\_./ }

hjkl 8 \_ \_.=("""")=.\_ \_

aceg 2 / \_ \_ \

vt10 0 (\_'"\_.-"`~~`"-.\_"'\_)

# Распечатка протокола

user@PSB133S01ZFH:/mnt/c/Users/User/Desktop/course9/sort$ make && ./bin/main data.txt

mkdir -p ./build

mkdir -p ./bin

gcc -Wall -c src/main.c -o build/main.o

gcc -Wall -c src/data.c -o build/data.o

gcc build/main.o build/data.o -lpthread -o bin/main

LAUNCHING PROGRAM...

READING DATA FROM FILE...

PRINTING DATA...

Key |Value

------|----------------------

abcd 1| \_.--""--.\_

aceg 2| / \_ \_ \

adma 3| \_ ( (\_\ /\_) ) \_

ansi 9| { \.\_\ /\ /\_./ }

hjkl 1| /\_"=-.}\_\_\_\_\_\_{.-="\_\

hjkl 8| \_ \_.=("""")=.\_ \_

vt10 0| (\_'"\_.-"`~~`"-.\_"'\_)

wyse 5| {\_" "\_}

SORTING...DONE

PRINTING SORTED DATA...

Key |Value

------|----------------------

abcd 1| \_.--""--.\_

aceg 2| / \_ \_ \

adma 3| \_ ( (\_\ /\_) ) \_

ansi 9| { \.\_\ /\ /\_./ }

hjkl 1| /\_"=-.}\_\_\_\_\_\_{.-="\_\

hjkl 8| \_ \_.=("""")=.\_ \_

vt10 0| (\_'"\_.-"`~~`"-.\_"'\_)

wyse 5| {\_" "\_}

PROGRAM COMPLETE

user@PSB133S01ZFH:/mnt/c/Users/User/Desktop/course9/sort$ sort -r < data.txt > data\_rev.txt

user@PSB133S01ZFH:/mnt/c/Users/User/Desktop/course9/sort$ make && ./bin/main data\_rev.txt

mkdir -p ./build

mkdir -p ./bin

LAUNCHING PROGRAM...

READING DATA FROM FILE...

PRINTING DATA...

Key |Value

------|----------------------

wyse 5| {\_" "\_}

vt10 0| (\_'"\_.-"`~~`"-.\_"'\_)

hjkl 8| \_ \_.=("""")=.\_ \_

hjkl 1| /\_"=-.}\_\_\_\_\_\_{.-="\_\

ansi 9| { \.\_\ /\ /\_./ }

adma 3| \_ ( (\_\ /\_) ) \_

aceg 2| / \_ \_ \

abcd 1| \_.--""--.\_

SORTING...DONE

PRINTING SORTED DATA...

Key |Value

------|----------------------

abcd 1| \_.--""--.\_

aceg 2| / \_ \_ \

adma 3| \_ ( (\_\ /\_) ) \_

ansi 9| { \.\_\ /\ /\_./ }

hjkl 1| /\_"=-.}\_\_\_\_\_\_{.-="\_\

hjkl 8| \_ \_.=("""")=.\_ \_

vt10 0| (\_'"\_.-"`~~`"-.\_"'\_)

wyse 5| {\_" "\_}

PROGRAM COMPLETE

user@PSB133S01ZFH:/mnt/c/Users/User/Desktop/course9/sort$ sort -R < data.txt > data\_rand.txt

user@PSB133S01ZFH:/mnt/c/Users/User/Desktop/course9/sort$ make && ./bin/main data\_rand.txt

mkdir -p ./build

mkdir -p ./bin

LAUNCHING PROGRAM...

READING DATA FROM FILE...

PRINTING DATA...

Key |Value

------|----------------------

abcd 1| \_.--""--.\_

wyse 5| {\_" "\_}

adma 3| \_ ( (\_\ /\_) ) \_

hjkl 1| /\_"=-.}\_\_\_\_\_\_{.-="\_\

ansi 9| { \.\_\ /\ /\_./ }

hjkl 8| \_ \_.=("""")=.\_ \_

aceg 2| / \_ \_ \

vt10 0| (\_'"\_.-"`~~`"-.\_"'\_)

SORTING...DONE

PRINTING SORTED DATA...

Key |Value

------|----------------------

abcd 1| \_.--""--.\_

aceg 2| / \_ \_ \

adma 3| \_ ( (\_\ /\_) ) \_

ansi 9| { \.\_\ /\ /\_./ }

hjkl 1| /\_"=-.}\_\_\_\_\_\_{.-="\_\

hjkl 8| \_ \_.=("""")=.\_ \_

vt10 0| (\_'"\_.-"`~~`"-.\_"'\_)

wyse 5| {\_" "\_}

PROGRAM COMPLETE

# Журнал отладки

|  |  |  |  |
| --- | --- | --- | --- |
| Дата | Время | Описание ошибки | Решение |
| 30.05.2019 | 16:30 | Не работал алгоритм сортировки | Изменены процедуры слияния |

# Заключение

Существует два основных класса сортировок — простые и эффективные. Первые работают за O(N2), вторые за O(N\*log(N)). Однако пренебрегать одним из этих классов не стоит, т. к. разные классы ведут себя лучше на разных объемах данных. При небольшом количестве выигрывают простые, при большом — эффективные.

# Список используемых источников

1. Виртуальная энциклопедия Linux по-русски [Электронный ресурс]. –

<http://rus-linux.net/>

1. Агрегатор гиковских статей [Электронный ресурс]. –

<https://habr.com/post/117050/>

1. Электронная энциклопедия [Электронный ресурс]. –

<https://ru.wikipedia.org/wiki/Dd>

1. Linux Open Source Software Technologies [Электронный ресурс]. –

<https://losst.ru/komanda-dd-linux>