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: | Жерлыгин М.А |
| Группа: | М8О-108Б-18 |
| Преподаватель: | Поповкин А.В. |
|  |  |
| Оценка: |  |
| Дата: |  |

# Содержание

[Содержание 2](#_Toc513831098)

[Задание 3](#_Toc513831099)

[Общий метод решения 3](#_Toc513831100)

[Общие сведения о программе 4](#_Toc513831101)

[Функциональное назначение 4](#_Toc513831102)

[Текст программы 4](#_Toc513831103)

[Описание переменных и функций 7](#_Toc513831104)

[Распечатка протокола 8](#_Toc513831105)

[Журнал отладки 12](#_Toc513831106)

[Заключение 12](#_Toc513831107)

[Список литературы 13](#_Toc513831108)

# 1. 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=5.*

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 (в терминах Н.Вирта ):*

2. Линейный выбор с подсчётом

*Структура таблицы:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***№*** | ***Тип ключа*** | ***Длина ключа в байтах*** | ***Хранение данных и ключей*** | ***Число элементов таблицы*** |
| 2 | целый | 4 | отдельно | 10-14 |

# Общий метод решения

Строки заданной таблицы сортируются по ключу методом линейного выбора с подсчётом количества перестановок. Линейный выбор является усовершенствованным методом пузырьковой сортировки.

Анализируя метод пузырьковой сортировки, можно отметить два обстоятельства:

* если при движении по части массива перестановки не происходят, то эта часть массива уже отсортирована и, следовательно, ее можно исключить из рассмотрения.
* при движении от конца массива к началу минимальный элемент "всплывает" на первую позицию, а максимальный элемент сдвигается только на одну позицию вправо.

Эти две идеи приводят к модификациям в методе пузырьковой сортировки:

* От последней перестановки до конца (начала) массива находятся отсортированные элементы. Учитывая данный факт, просмотр осуществляется не до конца (начала) массива, а до конкретной позиции. Границы сортируемой части массива сдвигаются на 1 позицию на каждой итерации.
* Просмотр массива осуществляется до тех пор, пока все элементы не встанут в порядке возрастания (убывания).
* Поиск строки таблицы по ключу реализуется методом бинарного поиска.

В программе предусмотрено случайное перемешивание строк таблицы при помощи генератора случайных чисел, а также перестановка строк таблицы в обратном порядке.

Для поиска строки программа требует ввести значение ключа.

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 in1.txt .

# **2.** **Общие сведения о программе**

**Язык и система программирования:** GNU C

**Местонахождение файлов: ~/**2sem/kursach8

# 3. Функциональное назначение

Программа предназначена для демонстрации использования метода сортировки линейным выбором при сортировке строк любой заданной таблицы по заданному ключу, а также для демонстрации использования поиска по ключу методом бинарного поиска в уже отсортированной по ключу таблице.

Показано, что эта сортировка работает при любом случайном перемешивании строк исходной таблицы, даже в худшем случае, когда таблица по ключу упорядочена в обратном порядке.

# 4. Текст программы

***main.c:***

#include <stdio.h>  
#include <string.h>  
#include <stdlib.h>  
#include <time.h>  
   
int n = -1;  
typedef struct  
{  
 int key;  
 char string[100];  
} line;  
   
line L[100];  
   
void prnt()  
{  
 puts("\nKey: \t\tString:");  
 for (int i = 0; i<n; i++)  
 printf("%6d│ %s", L[i].key, L[i].string);  
}  
   
void randomize()  
{  
 long a = time(0);  
 srand(a);  
}  
   
void randLines()  
{  
 randomize();  
 line temp;  
 for (int i = 0; i<n; i++)  
 {  
 int j = rand() % n;  
 temp = L[i];  
 L[i] = L[j];  
 L[j] = temp;  
 }  
}  
   
void revers()  
{  
 line temp;  
 for (int i = 0; i<n / 2; i++)  
 {  
 temp = L[i];  
 L[i] = L[n - i - 1];  
 L[n - i - 1] = temp;  
 }  
}  
   
void bsrch(int k)  
{  
 int m, left = 0, right = n - 1;  
 while (left < right)  
 {  
 m = (left + right) / 2;  
 if (k > L[m].key)  
 left = m + 1;  
 else right = m;  
 }  
 if (k == L[left].key)  
 printf("Key is found:\n%d %s", k, L[left].string);  
 else puts("Key not found");  
}  
   
void title()  
{  
 puts("Commands:\n\  
 1:\t Print file\n\  
 2:\t Sort file\n\  
 3:\t Binary search\n\  
 4:\t Randomize lines\n\  
 5:\t Reverse lines\n\  
 0:\t Exit");  
 return;  
}  
   
void sort()  
{  
 int i, j, min;  
 int count = 0;  
 line temp;  
 for (i = 0; i < n - 1; i++)  
 {  
 min = i;  
 for (j = i + 1; j < n; j++)  
 {  
 if (L[j].key < L[min].key)  
 min = j;  
 }  
 if (i != min)  
 {  
 temp = L[i];  
 L[i] = L[min];  
 L[min] = temp;  
 count++;  
 }  
 }

pritnf("%d", count);  
 return;  
}  
   
int main(int argc, char \*\*argv)  
{  
 FILE \*in;  
 int input, k = 0, ind = 0;  
   
 if (argc > 1)  
 {  
 if(!(in = fopen(argv[1],"r")))  
 {  
 puts("Can\'t open");  
 return 1;  
 }  
 }  
 else if(!(in = fopen("in1.txt","r")))  
 {  
 puts("Can\'t open");  
 return 1;  
 }  
 while (!feof(in))  
 {  
 fscanf(in, "%d", &L[++n].key);  
 fgets(L[n].string, 100, in);  
 }  
 title();  
 printf(">> ");  
   
 while (scanf("%d", &input) == 1)  
 {  
 switch(input)  
 {  
 case 1:  
 prnt();  
 break;  
 case 2:  
 sort();  
 prnt();  
 ind = 1;  
 break;  
 case 3:  
 if (ind)  
 {  
 printf("Enter the key:");  
 scanf("%d", &k);  
 bsrch(k);  
 }  
 else puts("Sort file first");  
 break;  
 case 4:  
 randLines();  
 prnt();  
 break;  
 case 5:  
 revers();  
 prnt();  
 break;  
 case 0:  
 fclose(in);  
 return 0;  
 default:  
 puts("Wrong command");  
 break;  
 }  
 title();  
 printf(">> ");  
 }  
 return 0;  
}

# 5. Описание переменных и функций

***main.c:***

|  |  |  |
| --- | --- | --- |
| **Название и тип**  **функции** | **Аргументы и их тип** | **Описание** |
| ***глобальные переменные*** | **int n** | **Количество строк таблицы** |
|  | **line L[100]** | Структура, содержащая ключи и соответствующие им строки |
| ***void prnt*** |  | Выводит все ключи и соответствующие им строки |
| ***void randomize*** |  | Вспомогательная функция для генерации рандомных значений |
| ***void randLines*** | **line temp** | Перемешивает все строки таблицы |
| ***void reverse*** | **line temp** | Ставит все строки таблицы в обратном порядке |
| ***void bsrch*** |  | Указатель на список |
|  | **int k** | Ключ для поиска |
|  | **int m** | Вспомогательная переменная для сужения области поиска |
|  | **int left** | Левая граница поиска |
|  | **int right** | Правая граница поиска |
| ***void title*** |  | Выводит спецификацию программы |
| ***void sort*** |  | Сортирует строки таблицы по ключу с помощью метода линейного выбора |
|  | **int count** | Количество сделанных перестановок |
|  | **int i** | Итерационная переменная |
|  | **int j** | Итерационная переменная |
|  | **int min** | Вспомогательная переменная для поиска минимального элемента |
| ***int main*** | **int argc**  **char \*\*argv** | Основная функция |
|  | **FILE \*in** | Файл с таблицей |
|  | **int input** | Вспомогательная переменная для ввода |
|  | **int k** | Вспомогательная переменная для поиска по ключу |
|  | **int ind** | Индикатор (отсортирована таблица или нет) |

# Таблица 1. Описание переменных и функций.

# 6. Распечатка протокола

mmaxim2710@DESKTOP-RDPBU3D:~$ ./a.out in3.txt

Commands:

1: Print file

2: Sort file

3: Binary search

4: Randomize lines

5: Reverse lines

0: Exit

>> 1

Key: String:

1│ \_\_\_ \_\_\_ \_\_\_ \_\_\_

2│ /\ \ /\ \ /\ \ /\ \

3│ \:\ \ /::\ \ /::\ \ \:\ \

4│ \:\ \ /:/\:\ \ /:/\ \ \ \:\ \

5│ /::\ \ /::\~\:\ \ \_\:\~\ \ \ /::\ \

6│ /:/\:\\_\_\ /:/\:\ \:\\_\_\ /\ \:\ \ \\_\_\ /:/\:\\_\_\

7│ /:/ \/\_\_/ \:\~\:\ \/\_\_/ \:\ \:\ \/\_\_/ /:/ \/\_\_/

8│ /:/ / \:\ \:\\_\_\ \:\ \:\\_\_\ /:/ /

9│ \/\_\_/ \:\ \/\_\_/ \:\/:/ / \/\_\_/

10│ \:\\_\_\ \::/ /

11│ \/\_\_/ \/\_\_/

>> 4

Key: String:

11│ \/\_\_/ \/\_\_/

8│ /:/ / \:\ \:\\_\_\ \:\ \:\\_\_\ /:/ /

6│ /:/\:\\_\_\ /:/\:\ \:\\_\_\ /\ \:\ \ \\_\_\ /:/\:\\_\_\

4│ \:\ \ /:/\:\ \ /:/\ \ \ \:\ \

3│ \:\ \ /::\ \ /::\ \ \:\ \

5│ /::\ \ /::\~\:\ \ \_\:\~\ \ \ /::\ \

10│ \:\\_\_\ \::/ /

9│ \/\_\_/ \:\ \/\_\_/ \:\/:/ / \/\_\_/

2│ /\ \ /\ \ /\ \ /\ \

7│ /:/ \/\_\_/ \:\~\:\ \/\_\_/ \:\ \:\ \/\_\_/ /:/ \/\_\_/

1│ \_\_\_ \_\_\_ \_\_\_ \_\_\_

>> 2

10

Key: String:

1│ \_\_\_ \_\_\_ \_\_\_ \_\_\_

2│ /\ \ /\ \ /\ \ /\ \

3│ \:\ \ /::\ \ /::\ \ \:\ \

4│ \:\ \ /:/\:\ \ /:/\ \ \ \:\ \

5│ /::\ \ /::\~\:\ \ \_\:\~\ \ \ /::\ \

6│ /:/\:\\_\_\ /:/\:\ \:\\_\_\ /\ \:\ \ \\_\_\ /:/\:\\_\_\

7│ /:/ \/\_\_/ \:\~\:\ \/\_\_/ \:\ \:\ \/\_\_/ /:/ \/\_\_/

8│ /:/ / \:\ \:\\_\_\ \:\ \:\\_\_\ /:/ /

9│ \/\_\_/ \:\ \/\_\_/ \:\/:/ / \/\_\_/

10│ \:\\_\_\ \::/ /

11│ \/\_\_/ \/\_\_/

>> 5

Key: String:

11│ \/\_\_/ \/\_\_/

10│ \:\\_\_\ \::/ /

9│ \/\_\_/ \:\ \/\_\_/ \:\/:/ / \/\_\_/

8│ /:/ / \:\ \:\\_\_\ \:\ \:\\_\_\ /:/ /

7│ /:/ \/\_\_/ \:\~\:\ \/\_\_/ \:\ \:\ \/\_\_/ /:/ \/\_\_/

6│ /:/\:\\_\_\ /:/\:\ \:\\_\_\ /\ \:\ \ \\_\_\ /:/\:\\_\_\

5│ /::\ \ /::\~\:\ \ \_\:\~\ \ \ /::\ \

4│ \:\ \ /:/\:\ \ /:/\ \ \ \:\ \

3│ \:\ \ /::\ \ /::\ \ \:\ \

2│ /\ \ /\ \ /\ \ /\ \

1│ \_\_\_ \_\_\_ \_\_\_ \_\_\_

>> 2

10

Key: String:

1│ \_\_\_ \_\_\_ \_\_\_ \_\_\_

2│ /\ \ /\ \ /\ \ /\ \

3│ \:\ \ /::\ \ /::\ \ \:\ \

4│ \:\ \ /:/\:\ \ /:/\ \ \ \:\ \

5│ /::\ \ /::\~\:\ \ \_\:\~\ \ \ /::\ \

6│ /:/\:\\_\_\ /:/\:\ \:\\_\_\ /\ \:\ \ \\_\_\ /:/\:\\_\_\

7│ /:/ \/\_\_/ \:\~\:\ \/\_\_/ \:\ \:\ \/\_\_/ /:/ \/\_\_/

8│ /:/ / \:\ \:\\_\_\ \:\ \:\\_\_\ /:/ /

9│ \/\_\_/ \:\ \/\_\_/ \:\/:/ / \/\_\_/

10│ \:\\_\_\ \::/ /

11│ \/\_\_/ \/\_\_/

>> 3

Enter the key:6

Key is found:

6 /:/\:\\_\_\ /:/\:\ \:\\_\_\ /\ \:\ \ \\_\_\ /:/\:\\_\_\

>>

# 7. 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>