**Министерство образования науки Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский авиационный институт (национальный исследовательский университет)»

«Информационные технологии и прикладная математика»

**Курсовой проект**

**По курсу «Практикум на ЭВМ»**

**2 семестр**

**Задание 8:**

«Сортировка и поиск»

|  |  |
| --- | --- |
| Группа: | М8О-106Б-21 |
| Студент: | Орусский В.Р. |
| Преподаватель: | Дубинин А.В. |
| Оценка: |  |
| Дата: |  |

**Оглавление**

[Введение 3](#_Toc104750260)

[Постановка задачи 4](#_Toc104750261)

[Вариант 5-3-11: 4](#_Toc104750262)

[Теоретическая часть 4](#_Toc104750263)

[Описание алгоритма 4](#_Toc104750264)

[Использованные в программе переменные 5](#_Toc104750265)

[Пользовательские структуры данных в программе: 5](#_Toc104750266)

[Исходники программы 6](#_Toc104750267)

[Протокол исполнения и тесты 8](#_Toc104750268)

[Вывод 11](#_Toc104750269)

# Введение

Для данной работы нам понадобится изучить различные алгоритмы для поиска и сортировки данных в различных структурах. Надо будет оценить скорость работы того или иного алгоритма. В качестве структуры данных будут выступать таблицы (словари).

# Постановка задачи

Составить программу на языке Си с использованием процедур и функций для сортировки таблицы заданным методом и двоичного (бинарного) поиска по ключу в таблице.

## Вариант 3-1:

Необходимо реализовать сортировку пузырьком для таблицы, в которой тип ключа – целый, длина ключа – 8 байт, хранение данных и ключа осуществляется вместе.

# Теоретическая часть

Алгоритмы поиска и сортировки в мире программирования – очень важные вещи. Поскольку, они влияют на кол-во обрабатываемых данных для успешного достижения цели, а, следовательно, и на скорость выполнения этих самых операций. А как известно, время – деньги, поэтому скорость работы тех или иных функций – критически важная составляющая любой программы.

Скажу пару слов о самой простой и самой интуитивно понятной сортировке – сортировке пузырьком (Bubble Sort). Самая известная сортировка данных, сложность которой где n – кол-во входных данных. Суть данной сортировки максимально просто – сравнивать попарно элементы слева направо, оставляя справа по одному отсортированному элементу на каждой итерации.

Бинарный поиск – не менее именитый алгоритм в мире программирования. Из названия понятно, что поиск как-то связан с числом два, а если быть точнее, он связан с поиском необходимых данных путём деления промежутка пополам и дальнейшего сравнения полученного значения с тем, которое необходимо найти. Сложность этого алгоритма где n – кол-во входных данных.

# Описание алгоритма

В начале была реализована таблица путём создания структуры для строки. Таблица в свою очередь представляет из себя массив строк. В строке содержится ключ в виде 8 байтового целого числа и значение, в виде строки (не более 120 символов).

Была реализована функция «GetRow» для обработки строки из файла и добавления её в таблицу.

Далее из файла считывается сначала номер строки, потом пробел в специальную переменную заглушку, а после считывается значение строки (не более 120 символов).

Реализация всех остальных алгоритмов была вкратце приведена выше.

# Использованные в программе переменные

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | За что отвечает |
| N | const integer | Ограничение на кол-во записей в таблице, просто так. |
| i | integer | Переменная для хранения индекса результата бинарного поиска |
| action | integer | Выбор действия пользователя |
| cnt | int | Количество строк в таблице |
| key | long integer | Переменная для хранения ключа |
| ch | char | Заглушка для считывания пробела в строке |
| arr | row[] | Массив строк - таблица |
| file | FILE\* | Файл, в котором лежат строки |

# Пользовательские структуры данных в программе:

|  |  |  |
| --- | --- | --- |
| Название | Поля | Значения |
| row | char[] \_str | Хранение значения в строке |
| long int \_key | Номер ключа (8 байт) |

# Исходники программы

Файлы «table.h» и «func.h» – заголовочные файлы, в котором объявлена структура таблицы и функций, необходимых для работы с ней.

#ifndef **TABLE\_H**#define **TABLE\_H**#include <stdio.h>  
  
*typedef struct* \_row {  
 *long int* \_key;  
 *char* \_str[120];  
} row;  
  
*void* PrintTable(row \*arr, *const int* size); *// Печать таблицы  
void* GetRow(FILE \*stream, *char* \*str, *const int* size); *// Получить одну строку из файла  
void* SwapRows(row \*r1, row \*r2); *// Поменять две строки местами*#endif

#ifndef **FUNC\_H**#define **FUNC\_H**#include "table.h"  
  
*int* BinSearch(*const* row \*arr, *const int* size, *const long int* key); *// Бинарный поиск в таблице по ключу  
void* Sort(row \*arr, *const int* size); *// Сортировка методом пузырька  
void* Scramble(row \*arr, *const int* size); *// Перемешивание данных в таблице  
void* Reverse(row \*arr, *const int* size); *// Реверс данных в таблице  
int* RandomAb(*const int* a, *const int* b); *// Рандомное значение числа  
int* IsSorted(*const* row \*arr, *const int* size); *// Проверка на отсортированность таблицы*#endif

Файлы «table.c» и «func.c» - реализация указанных выше функций.

#include "table.h"  
*//#include <stdio.h>  
  
// Печатаем таблицу.  
void* PrintTable(row \*arr, *const int* size) {  
 printf("+---------+------------------------------------------------+\n");  
 printf("| Key | Value |\n");  
 printf("+---------+------------------------------------------------+\n");  
 *for* (*int* i = 0; i < size; i++)  
 printf("|%9ld|%48s|\n", arr[i].\_key, arr[i].\_str);  
  
 printf("+---------+------------------------------------------------+\n");  
}  
  
*// Получаем строку из файла  
void* GetRow(FILE \*stream, *char* \*str, *const int* size) {  
 *//счётчик, чтобы строка не была больше указанного кол-ва знаков (один символ уходит на обозначение конца строки).  
 int* cnt = 0;  
 *char* ch;  
 *while* ((ch = getc(stream)) != '\n' && cnt < size - 1)  
 str[cnt++] = ch;  
 str[cnt] = '\0';  
}  
  
*// Смена мест между строками  
void* SwapRows(row \*r1, row \*r2) {  
 row tmp;  
 tmp = \*r1;  
 \*r1 = \*r2;  
 \*r2 = tmp;  
}

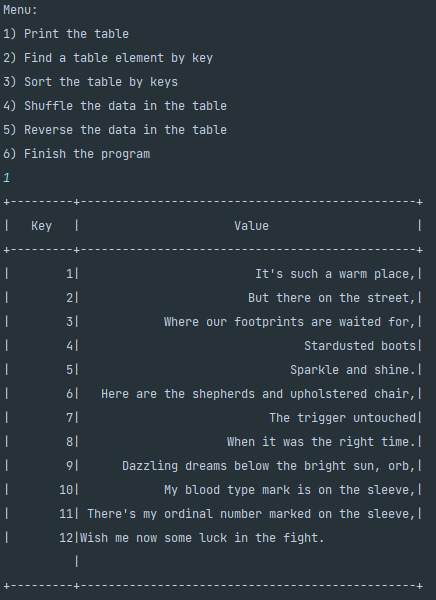
#include <time.h>  
#include <stdlib.h>  
#include "table.h"  
  
*// Бинарный поиск по ключу  
int* BinSearch(*const* row \*arr, *const int* size, *const long int* key) {  
 *int* start = 0, end = size - 1, mid;  
 *if* (size <= 0) *return* -1;  
 *while* (start < end) {  
 mid = (start + end) / 2;  
 *if* (arr[mid].\_key == key)  
 *return* mid;  
 *else if* (arr[mid].\_key < key)  
 start = mid + 1;  
 *else* end = mid;  
 }  
 *if* (arr[end].\_key == key) *return* end;  
 *return* -1;  
}  
  
*// Сортировка пузырьком  
void* Sort(row \*arr, *const int* size) {  
 *for* (*int* j = 1; j < size - 1; j++)  
 *for* (*int* i = 0; i < size - j; i++)  
 *if* (arr[i].\_key > arr[i + 1].\_key) {  
 SwapRows(&arr[i], &arr[i + 1]);  
 }  
}  
  
*// Генерация рандомного числа  
int* RandomAb(*const int* a, *const int* b) {  
 *return* a + rand() % (b - a + 1);  
}  
  
*// Рандомное перемешивание данных в таблице  
void* Scramble(row \*arr, *const int* size) {  
 *int* i, j, k;  
 srand((*unsigned int*)time(0));  
 *for* (k = 0; k < size; k++) {  
 i = RandomAb(0, size - 1);  
 j = RandomAb(0, size - 1);  
 SwapRows(&arr[i], &arr[j]);  
 }  
}  
  
*// Переворачивание таблицы  
void* Reverse(row \*arr, *const int* size) {  
 *int* i, j;  
 *for* (i = 0, j = size - 1; i < j; i++, j--)  
 SwapRows(&arr[i], &arr[j]);  
}  
  
*// Отсортирована ли таблица  
int* IsSorted(*const* row \*arr, *const int* size) {  
 *for* (*int* i = 0; i < size - 1; i++)  
 *if* (arr[i].\_key > arr[i + 1].\_key)  
 *return* 0;  
 *return* 1;  
}

Файл «main.c» - файл для взаимодействия с пользователем и реализация функций, выполняющих основное поставленное задание.

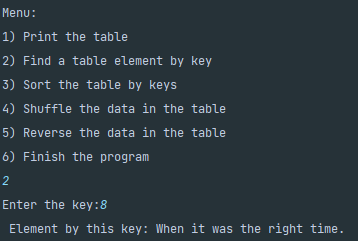
#include <stdio.h>  
#include "func.h"  
  
*void* printMenu() {  
 printf("Menu:\n");  
 printf("1) Print the table\n");  
 printf("2) Find a table element by key\n");  
 printf("3) Sort the table by keys\n");  
 printf("4) Shuffle the data in the table\n");  
 printf("5) Reverse the data in the table\n");  
 printf("6) Finish the program\n");  
}  
  
*int* main() {  
 *const int* N = 50;  
 *int* i, cnt, action;  
 *long* key;  
 *char* ch;  
 row arr[N];  
 FILE \*file = fopen("input.txt", "r");  
 *if* (file == **NULL**) {  
 printf("Error opening file\n");  
 *return* 0;  
 }  
 i = 0;  
 *while* (i < N && fscanf(file, "%ld", &arr[i].\_key) == 1) {  
 fscanf(file, "%c", &ch);  
 GetRow(file, arr[i].\_str, *sizeof*(arr[i].\_str));  
 i++;  
 }  
 fclose(file);  
 cnt = i;  
 *do* {  
 printMenu();  
 scanf("%d", &action);  
 *switch* (action) {  
 *case* 1: PrintTable(arr, cnt);  
 *break*;  
 *case* 2:  
 *if* (!IsSorted(arr, cnt)) {  
 printf("The table is not sorted\n");  
 } *else* {  
 printf("Enter the key: ");  
 scanf("%ld", &key);  
 i = BinSearch(arr, cnt, key);  
 *if* (i > -1) {  
 printf("Element by this key: %s\n", arr[i].\_str);  
 } *else* printf("There is no element with this key in the table!\n");  
 }  
 *break*;  
 *case* 3: Sort(arr, cnt);  
 *break*;  
 *case* 4: Scramble(arr, cnt);  
 *break*;  
 *case* 5: Reverse(arr, cnt);  
 *break*;  
 *case* 6: *break*;  
 *default*: printf("There is no such item in the menu! Try a different value\n");  
 *break*;  
 }  
 } *while* (action != 6);  
 *return* 0;  
}

# Протокол исполнения и тесты

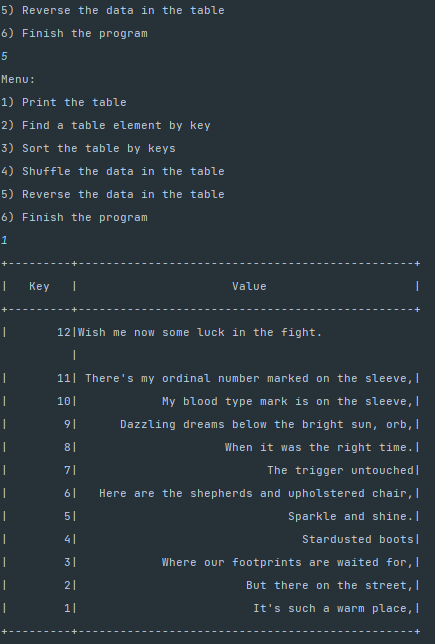
Тест №1



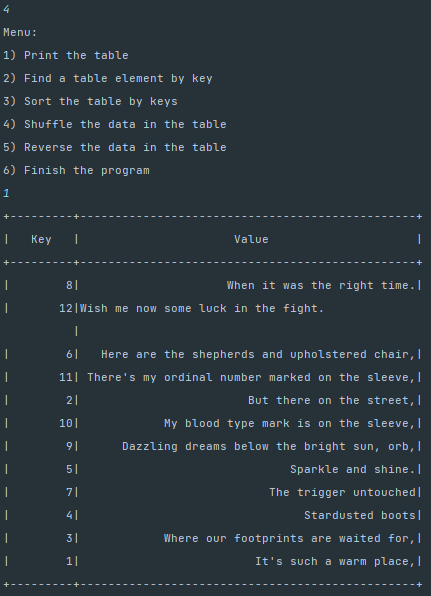
Тест №2



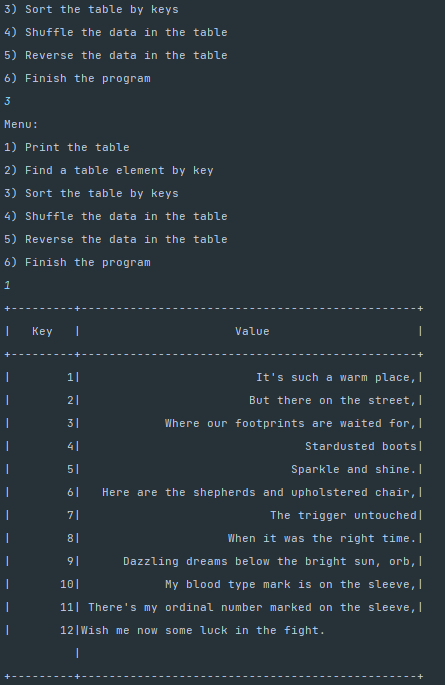
Тест №3



Тест №4



Тест №5



# Вывод

Как уже говорилось в теоретической части, сложность алгоритмов — это очень важная характеристика для успешной реализации программ. Следовательно, умение писать алгоритм с низкой сложность – хорошо и полезно для программиста и его карьеры. А значит, знания из данной работы определённо отложатся и помогут в дальнейшем. Я бы даже прорешал все варианты из задания, чтобы узнать и изучить все представленные виды сортировок, ведь каждый из них по-своему удобен и полезен.