Отчет по лабораторной работе №IX по курсу

Языки и методы программирования _____

Студент группы М8О-101Б-21 Постнов Александр Вячеславович, № по списку 17

Контакты www, e-mail: 61pav03@mail.ru

Работа выполнена: «» 202 2г.

Преподаватель: ___ каф. 806 _____Титов В.К._____

Входной контроль знаний с оценкой _____

Отчет сдан « » _____ 2022_ г., итоговая оценка _____

Подпись преподавателя _____

1. **Тема:** Сортировка и поиск _____
   _____

2. **Цель работы:** Составить программу на языке Си с использованием процедур и функций для сортировки таблицы заданным методоми двоичного поиска по ключу в таблице.

3. **Задание** (*вариант №17 -> метод сортировки: 2, структура таблицы: 5; изображение ASCII-графики):*
   Реализовать метод сортировки: 2. Линейный выбор с подсчетом(индексная).
   Структура таблицы:
   1) тип ключа - вещественный
   2) длина ключа в байтах - 16
   3) хранение данных и ключей - вместе
   4) минимальное число элементов таблицы - 15
4. **Оборудование(лабораторное)**:
   ЭВМ -, процессор -, имя узла сети - с ОП - ГБ,
   НМД - ГБ, терминал- адрес -, принтер -
   Другие устройства -

   Оборудование ПЭВМ студента, если использовалось:
   Процессор AMD Ryzen 5 4500U, с ОП 8 ГБ
   Другие устройства -

5. **Программное обеспечение:**
   Операционная система семейства -, наименование - версия -
   интерпретатор команд - версия
   Система программирования - версия -
   Редактор текстов - версия -
   Утилиты операционной системы -
   Прикладные системы и программы -
   Местонахождение и имена файлов программ и данных -

   Программное обеспечение ЭВМ студента, если использовалось:
   Операционная система семейства GNU/Linux, наименование Manjaro версия 5-13-12-1
   интерпретатор команд GNOME Terminal версия 3.38.2.
   Система программирования _____ версия _____
   Редактор текстов emacs версия 3.27.20
   Утилиты операционной системы
   Прикладные системы и программы -
   Местонахождение и имена файлов программ и данных -

6. **Идея, метод, алгоритм** решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок,

Индексная сортировка:

Данный метод впервые упоминается в работе Э.Х. Фрэнда, хотя он и не заявил о ней как о своей собственной разработке.

При упорядочении таблицы этим методом необходима память для хранения исходной таблицы, а также дополнительно должна быть выделена память под счетчик для каждого элемента таблицы.

В этом алгоритме элементы не перемещаются. Он подобен сортировке таблицы адресов, поскольку таблица счетчиков определяет конечное положение элементов.

Просмотр таблицы начинается с первой записи. Ее ключ сравнивается с ключами последующих записей. При этом счетчик большего из сравниваемых ключей увеличивается на 1. При втором просмотре таблицы первый ключ уже не рассматривается, второй ключ сравнивается со всеми последующими. Результаты сравнений фиксируются в счетчиках. Для таблицы, содержащей N элементов, этот процесс повторяется N-1 раз.

После выполнения всех просмотров счетчик каждого элемента указывает, какое количество ключей таблицы меньше ключа этого элемента. Эти счетчики используются затем в качестве индексов элементов результирующей таблицы. Поместив записи в результирующую таблицу в соответствии со значениями их счетчиков, получим упорядоченную таблицу. Другими словами, если известно, что некоторый ключ превышает ровно 27 других, то после сортировки соответствующий элемент должен занять 28-е место.

Бинарный поиск – классический алгоритм поиска элемента в отсортированном массиве, использующий дробление массива на половины. Алгоритм:

1. Определение значения элемента в середине структуры данных. Полученное значение сравнивается с ключом.
2. Если ключ меньше значения середины, то поиск осуществляется в первой половине элементов, иначе — во второй.
3. Поиск сводится к тому, что вновь определяется значение серединного элемента в выбранной половине и сравнивается с ключом.
4. Процесс продолжается до тех пор, пока не будет найден элемент со значением ключа или не станет пустым интервал для поиска.

Работа с таблицей будет реализована с помощью меню

На вход будет подаваться файл drawing.txt, в котором находится таблица заданной структуры. Строки образуют *изображение (ASCII-график).*

7. **Сценарий выполнения работы** [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты либо соображения по тестированию].

```
#include <stdio.h>
#include <stdlib.h>

int is_sorted = 0;



typedef long double Key; //вещественный ключ



struct DrawingString { //строка таблицы
 Key key;
 char *string;
};

int get_size(FILE *input) { //количество строк в таблице
 int n = 0;
 char *buffer = new char[128];
 size_t l = sizeof(buffer);
 while (!feof(input)) {
   getline(&buffer, &l, input);
   n++;
 }
 delete[] buffer;
 rewind(input);
 return n;
}

void print_table(DrawingString *table, int size) { //вывод таблицы на экран
 for (int i = 0; i < size; ++i) {
   printf("%3Lf  -----  %s", table[i].key,
          table[i].string);
 }
 printf("\n");
```

```
}

DrawingString *make_table(FILE *input, int size) { //создание таблицы
 DrawingString *table = new DrawingString[size];
 for (int i = 0; i < size; ++i) {
   table[i].string = new char[128];
 }
 for (int i = 0; i < size; ++i) {
   fscanf(input, "%Lf", &table[i].key);
   fgets(table[i].string, 128, input);
 }
 return table;
}

void sort_table(DrawingString *a, int n) { //индексная сортировка таблицы
 int i, j, k;
 is_sorted = 1;
 DrawingString *b = new DrawingString[n];
 for (int i = 0; i < n; ++i) {
   b[i].string = new char[128];
 }
 for (i = 0; i < n; i++) {
   for (j = k = 0; j < n; j++)
     if (a[j].key < a[i].key || a[j].key == a[i].key && i < j)
       k++;
   b[k] = a[i];
 }
 for (i = 0; i < n; i++)
   a[i] = b[i];
}

void swap_strings(DrawingString *table, int a, int b) { //поменять местами строки
 DrawingString temp;
 temp = table[a];
```

```
 table[a] = table[b];
 table[b] = temp;
}

void mix_table(DrawingString *table, int size) { //перемешивание строк
 is_sorted = 0;
 int a, b;
 for (int i = 0; i < size; ++i) {
   a = rand() % size;
   b = rand() % size;
   swap_strings(table, a, b);
 }
}

int bin_search(DrawingString *table, Key key, int size) { //бинарный поиск в
отсортированной таблице
 int l = 0, r = size - 1, mid;
 while (l <= r) {
   mid = (r + l) / 2;
   if (key == table[mid].key)
     return mid;
   if (l == r) { //интервал уменьшен до нуля
     return -1;
   }
   if (key < table[mid].key)
     r = mid;
   if (key > table[mid].key)
     l = mid + 1;
 }
 return -1;
}

void reverse_strings(DrawingString *table, int size) { //реверс таблицы
 is_sorted = 0;
```

```
 for (int i = 0, j = size - 1; i < j; ++i, --j) {
   swap_strings(table, i, j);
 }
}

int main() {
 FILE *input;
 if ((input = fopen("drawing.txt", "r")) == NULL) {
   printf("Error:  can't  input  from  drawing.txt\n");
   return 1;
 }
 int n = get_size(input), action;
 n--;
 DrawingString *table = make_table(input, n);
 print_table(table, n);
 while (1) {
   printf("Menu\n");
   printf("1)  Binary  search\n");
   printf("2)  Sort\n");
   printf("3)  Mix\n");
   printf("4)  Reverse\n");
   printf("5)  Exit\n");
   printf("Choose  an  action\n");
   scanf("%d", &action);
   switch (action) {
   case 1: {
     if (is_sorted) {
       printf("Enter the real key: ");
       Key k;
       scanf("%Lf", &k);
       int search;
       printf("\n");
       search = bin_search(table, k, n);
       if (search == -1) {
```

```
        printf("Element with such key is not found!\n");
        break;
      }
      printf("Found the string:\n %s", table[search].string);
    } else
      printf("Table is not sorted!\n");
    break;
  }
  case 2: {
    sort_table(table, n);
    print_table(table, n);
    break;
  }
  case 3: {
    mix_table(table, n);
    print_table(table, n);
    break;
  }
  case 4: {
    reverse_strings(table, n);
    print_table(table, n);
    break;
  }
  case 5:
    return 0;
  }
 }
}
```

Тестирование:

проверять работу процедуры

сортировки буду в трех случаях: (1) элементы таблицы с самого начала упорядочены; (2) элементы таблицы расставлены в

обратном порядке; (3) элементы таблицы не упорядочены.

проверка бинарного поиска:

1) попробую найти несуществующий элемент(не должен)
2) найду существующий элемент таблицы. (повторю эту процедуру несколько раз)

Пункты 1-7 отчета составляются строго до начала лабораторной работы.

Допущен к выполнению работы. **Подпись преподавателя _____**

8. Распечатка протокола (подклеить листинг окончательного варианта программы с тестовыми примерами, подписанный преподавателем).

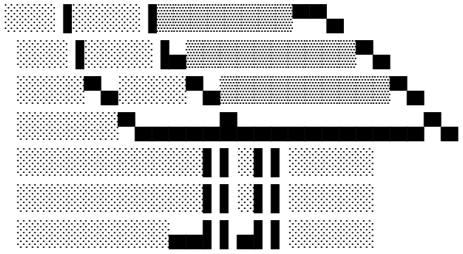
```
  ~/P/mai_labs/2/9(не делал)   main !2 ?9  cat head.txt    ✔
------------------------------------------------------------
|          Лабораторная работа №IX                          |
|          Сортировка и поиск                                                  |
|          Выполнил:  студент группы М8О-101Б-21        |
|          Постнов Александр Вячеславович                  |
------------------------------------------------------------
  ~/P/mai_labs/2/9(не делал)   main !2 ?9  cat drawing.txt  ✔
1.1110
2.1111          ЗАПУСКАЕМ
3.1112    ГУСЯ          РАБОТЯГИ
4.1113
5.1114
6.1115
7.1116
8.1117
9.1118
10.1119
11.1120
12.1121
13.1122
14.1123
15.1124
  ~/P/mai_labs/2/9(не делал)   main !2 ?9  cat main.cpp    ✔
#include <stdio.h>
#include <stdlib.h>

int is_sorted = 0;


typedef long double Key; //вещественный ключ


struct DrawingString { //строка таблицы
  Key key;
```

```
 char *string;
};

int get_size(FILE *input) { //количество строк в таблице
 int n = 0;
 char *buffer = new char[128];
 size_t l = sizeof(buffer);
 while (!feof(input)) {
  getline(&buffer, &l, input);
  n++;
 }
 delete[] buffer;
 rewind(input);
 return n;
}

void print_table(DrawingString *table, int size) { //вывод таблицы на экран
 for (int i = 0; i < size; ++i) {
  printf("%3Lf  -----  %s", table[i].key,
       table[i].string);
 }
 printf("\n");
}

DrawingString *make_table(FILE *input, int size) { //создание таблицы
 DrawingString *table = new DrawingString[size];
 for (int i = 0; i < size; ++i) {
  table[i].string = new char[128];
 }
 for (int i = 0; i < size; ++i) {
  fscanf(input, "%Lf", &table[i].key);
  fgets(table[i].string, 128, input);
 }
 return table;
}

void sort_table(DrawingString *a, int n) { //индексная сортировка таблицы
 int i, j, k;
 is_sorted = 1;
 DrawingString *b = new DrawingString[n];
 for (int i = 0; i < n; ++i) {
  b[i].string = new char[128];
 }
 for (i = 0; i < n; i++) {
  for (j = k = 0; j < n; j++)
   if (a[j].key < a[i].key || a[j].key == a[i].key && i < j)
    k++;
  b[k] = a[i];
 }
 for (i = 0; i < n; i++)
  a[i] = b[i];
}

void swap_strings(DrawingString *table, int a, int b) { //поменять местами строки
```

```
  DrawingString temp;
  temp = table[a];
  table[a] = table[b];
  table[b] = temp;
}

void mix_table(DrawingString *table, int size) { //перемешивание строк
  is_sorted = 0;
  int a, b;
  for (int i = 0; i < size; ++i) {
    a = rand() % size;
    b = rand() % size;
    swap_strings(table, a, b);
  }
}

int bin_search(DrawingString *table, Key key, int size) { //бинарный поиск в отсортированной таблице
  int l = 0, r = size - 1, mid;

  while (l <= r) {
    mid = (r + l) / 2;
    if (key == table[mid].key)
      return mid;
    if (l == r) { //интервал уменьшен до нуля
      return -1;
    }
    if (key < table[mid].key)
      r = mid;
    if (key > table[mid].key)
      l = mid + 1;
  }
  return -1;
}

void reverse_strings(DrawingString *table, int size) { //реверс таблицы
  is_sorted = 0;
  for (int i = 0, j = size - 1; i < j; ++i, --j) {
    swap_strings(table, i, j);
  }
}

int main() {
  FILE *input;
  if ((input = fopen("drawing.txt", "r")) == NULL) {
    printf("Error: can't input from drawing.txt\n");
    return 1;
  }
  int n = get_size(input), action;
  n--;
  DrawingString *table = make_table(input, n);
  print_table(table, n);
  while (1) {
    printf("Menu\n");
    printf("1) Binary search\n");
```

```
   printf("2)  Sort\n");
   printf("3)  Mix\n");
   printf("4)  Reverse\n");
   printf("5)  Exit\n");
   printf("Choose  an  action\n");
   scanf("%d", &action);
   switch (action) {
   case 1: {
    if (is_sorted) {
      printf("Enter the real key: ");
      Key k;
      scanf("%Lf", &k);
      int search;
      printf("\n");
      search = bin_search(table, k, n);
      if (search == -1) {
        printf("Element  with  such  key  is  not  found!\n");
        break;
      }
      printf("Found  the  string:\n  %s", table[search].string);
    } else
      printf("Table  is  not  sorted!\n");
    break;
   }
   case 2: {
    sort_table(table, n);
    print_table(table, n);
    break;
   }
   case 3: {
    mix_table(table, n);
    print_table(table, n);
    break;
   }
   case 4: {
    reverse_strings(table, n);
    print_table(table, n);
    break;
   }
   case 5:
    return 0;
   }
  }
}
   ~/P/mai_labs/2/9(не делал)   main !2 ?9  g++ main.cpp    ✔
   ~/P/mai_labs/2/9(не делал)   main !2 ?9  ./a.out        ✔
1.111000 -----
2.111100 -----   ЗАПУСКАЕМ
3.111200 -----  ГУСЯ  РАБОТЯГИ
4.111300 -----
5.111400 -----
6.111500 -----
7.111600 -----
8.111700 -----
```

9.111800 -----
10.111900 -----
11.112000 -----
12.112100 -----
13.112200 -----
14.112300 -----
15.112400 -----



Menu

1) Binary search
2) Sort
3) Mix
4) Reverse
5) Exit

Choose an action

2

1.111000 -----
2.111100 -----
3.111200 -----
4.111300 -----
5.111400 -----
6.111500 -----
7.111600 -----
8.111700 -----
9.111800 -----
10.111900 -----
11.112000 -----
12.112100 -----
13.112200 -----
14.112300 -----
15.112400 -----

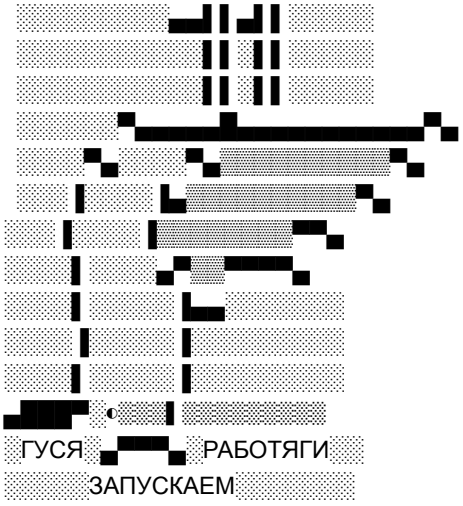
Menu

1) Binary search
2) Sort
3) Mix
4) Reverse
5) Exit

Choose an action

4

15.112400 -----
14.112300 -----
13.112200 -----
12.112100 -----
11.112000 -----
10.111900 -----
9.111800 -----
8.111700 -----
7.111600 -----
6.111500 -----
5.111400 -----
4.111300 -----
3.111200 -----
2.111100 -----



1.111000 -----

Menu
1) Binary search
2) Sort
3) Mix
4) Reverse
5) Exit
Choose an action
2
1.111000 -----
2.111100 ----- ЗАПУСКАЕМ
3.111200 ----- ГУСЯ РАБОТЯГИ
4.111300 -----
5.111400 -----
6.111500 -----
7.111600 -----
8.111700 -----
9.111800 -----
10.111900 -----
11.112000 -----
12.112100 -----
13.112200 -----
14.112300 -----
15.112400 -----

Menu
1) Binary search
2) Sort
3) Mix
4) Reverse
5) Exit
Choose an action
3
7.111600 -----
3.111200 ----- ГУСЯ РАБОТЯГИ
2.111100 ----- ЗАПУСКАЕМ
4.111300 -----
6.111500 -----
8.111700 -----
1.111000 -----
9.111800 -----
12.112100 -----
14.112300 -----
13.112200 -----
10.111900 -----
11.112000 -----
5.111400 -----
15.112400 -----

Menu
1) Binary search
2) Sort
3) Mix

4) Reverse
5) Exit
Choose an action
2
1.111000 -----
2.111100 ----- ЗАПУСКАЕМ
3.111200 ----- ГУСЯ РАБОТЯГИ
4.111300 -----
5.111400 -----
6.111500 -----
7.111600 -----
8.111700 -----
9.111800 -----
10.111900 -----
11.112000 -----
12.112100 -----
13.112200 -----
14.112300 -----
15.112400 -----

Menu
1) Binary search
2) Sort
3) Mix
4) Reverse
5) Exit
Choose an action
3
6.111500 -----
1.111000 -----
14.112300 -----
12.112100 -----
5.111400 -----
3.111200 ----- ГУСЯ РАБОТЯГИ
4.111300 -----
7.111600 -----
8.111700 -----
15.112400 -----
11.112000 -----
10.111900 -----
13.112200 -----
2.111100 ----- ЗАПУСКАЕМ
9.111800 -----

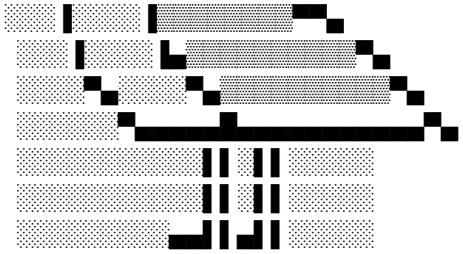
Menu
1) Binary search
2) Sort
3) Mix
4) Reverse
5) Exit
Choose an action
2
1.111000 -----
2.111100 ----- ЗАПУСКАЕМ

3.111200 ----- ГУСЯ РАБОТЯГИ
4.111300 -----
5.111400 -----
6.111500 -----
7.111600 -----
8.111700 -----
9.111800 -----
10.111900 -----
11.112000 -----
12.112100 -----
13.112200 -----
14.112300 -----
15.112400 -----

Menu
1) Binary search
2) Sort
3) Mix
4) Reverse
5) Exit
Choose an action
3
15.112400 -----
10.111900 -----
14.112300 -----
6.111500 -----
2.111100 ----- ЗАПУСКАЕМ
4.111300 -----
9.111800 -----
12.112100 -----
1.111000 -----
7.111600 -----
11.112000 -----
8.111700 -----
13.112200 -----
3.111200 ----- ГУСЯ РАБОТЯГИ
5.111400 -----

Menu
1) Binary search
2) Sort
3) Mix
4) Reverse
5) Exit
Choose an action
2
1.111000 -----
2.111100 ----- ЗАПУСКАЕМ
3.111200 ----- ГУСЯ РАБОТЯГИ
4.111300 -----
5.111400 -----
6.111500 -----
7.111600 -----
8.111700 -----

9.111800 -----
10.111900 -----
11.112000 -----
12.112100 -----
13.112200 -----
14.112300 -----
15.112400 -----



Menu
1) Binary search
2) Sort
3) Mix
4) Reverse
5) Exit
Choose an action
1
Enter the real key: 11.56

Element with such key is not found!
Menu
1) Binary search
2) Sort
3) Mix
4) Reverse
5) Exit
Choose an action
1
Enter the real key: 15

Element with such key is not found!
Menu
1) Binary search
2) Sort
3) Mix
4) Reverse
5) Exit
Choose an action
1
Enter the real key: 1.111

Found the string:

Menu
1) Binary search
2) Sort
3) Mix
4) Reverse
5) Exit
Choose an action
1
Enter the real key: 4.1113

Found the string:

Menu
1) Binary search
2) Sort
3) Mix
4) Reverse
5) Exit
Choose an action
1
Enter the real key: 15.1124

Found the string:

Menu
1) Binary search
2) Sort
3) Mix
4) Reverse
5) Exit
Choose an action
5

9. Дневник отладки должен содержать дату и время сеансов отладки и основные события (ошибки в сценарии и программе, нестандартные ситуации) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

№	Лаб. или дом.	Дата	Время	Событие	Действие по исправлению	Примечание
1	дом	11.05	16:00	Бинарный поиск бесконечно работал	Проверял размер интервала(если он равен нулю, то выходил)	

10. Замечания автора

11.**Выводы**

В ходе лабораторной работы и в ходе подготовке к ней изучил классические алгоритмы сортировки, разобрался в них

_____
_____
_____
_____
_____
_____

Недочёты при выполнении задания могут быть устранены следующим образом:

_____
_____
_____
_____
_____

Подпись студента ____Постнов_____