Отчет по курсовой работе IX

по курсу: 1 фундаментальная информатика

студента группы М80-101Б-21 Тулина Ивана, № по списку: 22

Контакты www, e-mail, icq, skype: i.tulin0107@gmail.com

Работа выполнена: «23» августа 2022г.

Преподаватель: Титов В. К. каф. 806

Входной контроль знаний с оценкой _____

Отчет сдан « » _____201 __ 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 методом и двоичного поиска по ключу в таблице.

3. **Задание** (*вариант №* 22)**:** *Метод сортировки:* сортировка Шелла. *Структура таблицы:*

№	тип ключа	длина ключа в байтах	хранение данных и ключей	минимальное число элементов таблицы
1	целый	8	вместе	11

4. **Оборудование** (лабораторное):
ЭВМ____-_____, процессор_____-_____, имя узла сети _____-_____ с ОП _____-_____ Мб,
НМД _____ Мб. Терминал _____ адрес _____. Принтер _____
Другие устройства _____
_____

Оборудование ПЭВМ студента, если использовалось:
Процессор Intel Core i5-7300HQ с ОП 7,87 Мб, НМД 15360 Мб. Монитор: встроенный
Другие устройства _____
_____

5. **Программное обеспечение (лабораторное):**
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 ___UNIX___, наименование _____Ubuntu_____ версия ___20.04.3 LTS_____
интерпретатор команд _____bash____ версия _____.
Система программирования _____версия _____
Редактор текстов _____Emacs_____ версия _____3.22.30_____
Утилиты операционной системы _-_
Прикладные системы и программы _____-_____
Местонахождение и имена файлов программ и данных на домашнем компьютере _____-_____

6. Идея, метод, алгоритм решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальные спецификации с пред- и постусловиями)

Идея:
Для описания строки таблицы используется структура, которая состоит из целочисленного ключа строки и символьного массива, к которую будет помещена сама строка.

Работа программы реализована посредством следующих процедур и функций:
- Функция, которая открывает файл с входными данными;
- Функция, которая считает количество строк в файле;
- Функция, формирующая из входных данных таблицу;
- Процедура, печатающая полученную таблицу;
- Процедура, реверсирующая положение строк в таблице;
- Процедура, перемешивающая строки в таблице;
- Процедура, сортирующая строки в таблице;
- Функция двоичного поиска строки в таблице.

Для быстрой проверки отсортированности таблицы выделим отдельную переменную ***isSorted*** в которую будем помещать 1, если таблица отсортирована, и 0, если она не отсортирована. Изначально переменная равна нулю.

Функция, открывающая входные файлы на чтение
Функция открывает файл при помощи ***fopen()***, проверяя успешность её выполнения в условном операторе. В случае неудачного выполнения возвращает нулевой указатель. В случае удачного выполнения выполнения она вернет указатель на файл.

Функция, считающая количество строк в файле
Функция в цикле просматривает файл до конца, увеличивая счетчик каждый раз, когда встречает знак перехода на новую строку. После прочтения файла при помощи процедуры ***rewind()***, возвращает указатель к началу файла. Возвращает полученное значение счетчика.

Функция, формирующая из входных данных таблицу
Функция при помощи ***new*** создает массив строк таблицы (структура из ключа и строки) длины, вычисленной ранее в функции подсчета строк. Затем считывает данные из файла в созданную таблицу. Считывание происходит следующим образом: при помощи ***fscanf()*** считывается ключ строк, затем при помощи ***fgets()*** считывается сама строка. Результатом работы функции является указатель на созданный массив.

Процедура, печатающая полученную таблицу
В цикле по счетчику печатаются строки таблицы: сначала ключ, потом строка.

Процедура, реверсирующая положение строк в таблице
В цикле счетчиком процедура меняет i-ую строку таблицы с (size-i)-ой, где size – количество строк.
В конце выполнения процедура меняет значение ***isSorted*** на ноль.

Процедура, перемешивающая строки в таблице
При помощи ***rand()%size*** генерируются 2 числа – номера строк в таблице, которые процедура меняет местами.
В конце выполнения процедура меняет значение ***isSorted*** на ноль.

Процедура сортировки строк в таблице по ключу
Процедура сортирует строки по ключу методом сортировки Шелла. Сортировка Шелла является улучшением метода вставки. Её скорость зависит от выбора шага сортировки.
Мы изначально будем использовать шаг сортировки равный половине размера массива. Сортируется массив в цикле, который останавливается, когда новый шаг сортировки принимает нулевое значение. В начале каждой итерации цикла будем уменьшать значение шага сортировки в 2 раза. В каждой итерации будет происходить сортировка вставкой для элементов, отстоящих друг от друга на один шаг сортировки. Т. е. если значение i-ой строки меньше, чем значение (s-i)-ой строки в таблице, то процедура будет менять эти строки местами.
Полезная особенность сортировки Шелла состоит в отсутствии необходимости создания дополнительных массивов для выполнения сортировки. Это очень удобно при сортировки таблиц.
В конце выполнения процедура меняет значение ***isSorted*** на единицу.

Функция бинарного поиска строки в таблице.
Изначально функция проверяет отсортирована ли таблица. В случае если она не отсортирована, функция печатает сообщение об ошибке и возвращает -1.
Функция принимает на вход указатель на таблицу, её размер и значение ключа искомой строки.
Для поиска задаются левая и правая граница поиска, изначально равные граница массива. Поиск по ключу происходит в цикле, который останавливается, когда значение левой границы оказывается больше правой. В каждой итерации случайным образом выбирается строка между левой и правой границей. Значение её ключа сравнивается со значением ключа искомой строки. Если значения ключей совпадают, то функция возвращает номер найденной строки (индекс элемента массива). Если значение ключа искомой строки больше чем то, с которым происходит сравнение– левая граница поиска смещается к индексу строки, с которой проводилось сравнение. В противоположном случае – смещается правая граница. Если за все итерации цикла строка не была найдена, функция возвращает -1.

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>
#include<time.h>

const int N=128;
int is_sorted=0;

struct iString
{ long key;
  char *string;
};


void randomize()
{ long a=time(0);
  srand(a);
}


FILE* openFile(int argc, char *argv)
{  FILE *input;
  if(argc==1)
    { if(!(input=fopen("input.txt", "r")))
      { printf("Error: can't open file input.txt\n");
        char n=getchar(); printf("%c\n", n);
        return 0;
      }
    }
  else if(argc==2)
    { if(!(input=fopen(argv, "r")))
      { printf("Error: can't open file %s\n", argv);
        return 0;
      }
    }
  else
    { printf("Error: wrong input\n");
      return 0;
    }
  return input;
}


int iSize(FILE* input)
{ char buffer; int n=0;
  while(!feof(input))
  { buffer=fgetc(input);
    if(buffer=='\n')
        n++;
  }
  rewind(input);
  return n;
}


iString *makeTable(FILE *input, int size)
{ iString *table = new iString[size];
  for(int i=0; i<size; i++)
    table[i].string= new char[N];
  int l=N*sizeof(char);
  for(int i=0; i<size; i++)
  { fscanf(input, "%d", &table[i].key);
    fgets(table[i].string, l, input);
  }
 return table;
}


void printBoard()
{ printf("+---+-----------------------------"
         "---------------------------------\n" );
}
```

```
void printTable(iString *table, int size)
{ printBoard(); printf("|KEY|STRING\n");
  printBoard();
  for(int i=0; i<size; i++)
  { printf("|%3d|", table[i].key);
    printf(" %s", table[i].string);
  }
  printBoard();
  printf("\n\n");
}


void swap(iString *table, int first, int second)
{ iString buffer;
  buffer = table[first];
  table[first] = table[second];
  table[second] = buffer;
}


void reverse(iString *table, int size)
{ for(int i=0; i<size/2; i++)
    swap(table, i, size-i-1);
  is_sorted=0;
}


void mix(iString *table, int size)
{ int first, second;
  for(int i=0; i<size/2; i++)
  { first=rand()%size;
    second=rand()%size;
    swap(table, first, second);
  }
  is_sorted=0;
}


void sortTable(iString *table, int size)
{ int s=1, i, j;
  while(s<size) s*=2;
  while(s>0)
  { s=(s-1)/2;
    for(i=0; i<size-s; i++)
    for(j=i+s; j-s>=0; j-=s)
      if(table[j-s].key>table[j].key)
        swap(table, j-s, j);
      else break;
  }
  is_sorted=1;
}


int binSearch(iString *table, int size, long x)
{ if(!is_sorted)
  { printf("Error: table isn`t sorted\n");
    return -1;
  }
  int l=0, r=size-1, m;
  while(l<=r)
  { m=rand()%(r-l)+l;
    if(table[m].key==x) return m;
    if(table[m].key<x) l=m+1;
    else r=m;
  }
  return -1;
}
```

```
void menu()
{ printf("\n====================\n"
       "_____Menu_____\n"
       "1. Print table\n"
       "2. Binary search\n"
       "3. Sort\n"
       "4. Mix\n"
       "5. Reverse\n"
       "6. Menu\n"
       "0. Exit\n"
       "====================\n");
}

int main(int argc, char *argv[])
{ FILE *input; int size;
  iString *table;
  randomize();
  input = openFile(argc,argv[1]);
  if(!input) return 1;
  size=iSize(input);
  table = makeTable(input, size);
  int action=6;
  while(action)
  { if(action==1)
       printTable(table, size);
    else if(action==2)
    { printf("\nInput key\n");
      long x; int i;
      scanf("%d", &x);
      i=binSearch(table, size, x);
      if(i<0)
        printf("String not found\n");
      else
        printf("Found the string:\n %s",
                    table[i].string);
    }
    else if(action==3)
    { sortTable(table, size);
      printf("\nTable was sorted\n");
    }
    else if(action==4)
    { mix(table, size);
      printf("\nTable was mixed\n");
    }
    else if(action==5)
    { reverse(table, size);
      printf("\nTable was reversed\n");
    }
    else if(action==6)
      menu();
    printf("\nInput number of action\n");
    scanf("%d", &action);
  }

return 0;}
```

Тестовый файл input.txt:

```
241   ___   _____ _________________     ___    ____  ______
333    /   | / ___// _____/  _/  _/     /   |  / __ \/_  __/
357   / /| | \__ \/ /     / / / /      / /| | / /_/ / / /
412  / ___ |___/ / /____/ /_/ /      / ___ |/ _, _/ / /
431 /_/  |_/____/\_____/___/___/     /_/  |_/_/ |_| /_/
479
500   ___   _____ _________________     ___    ____  ______
742    /   | / ___// _____/  _/  _/     /   |  / __ \/_  __/
788   / /| | \__ \/ /     / / / /      / /| | / /_/ / / /
890  / ___ |___/ / /____/ /_/ /      / ___ |/ _, _/ / /
999 /_/  |_/____/\_____/___/___/     /_/  |_/_/ |_| /_/
```

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

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

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

```
yusayu@YS:~/Рабочий стол/cppProjects/kr9$ cat head
*************************************************************
*                       Курсовая работа IX                  *
*                       Сортировка и поиск                  *
*                 Выполнил: Тулин Иван Денисович            *
*                      (номер по списку: 22)                *
*                       Группа: М8О-101Б-21                 *
*************************************************************
yusayu@YS:~/Рабочий стол/cppProjects/kr9$ cat kr9.cpp
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

const int N=128;
int is_sorted=0;

struct iString
{ long key;
  char *string;
};


void randomize()
{ long a=time(0);
  srand(a);
}


FILE* openFile(int argc, char *argv)
{  FILE *input;
  if(argc==1)
    { if(!(input=fopen("input.txt", "r")))
      { printf("Error: can't open file input.txt\n");
        char n=getchar(); printf("%c\n", n);
        return 0;
      }
    }
  else if(argc==2)
    { if(!(input=fopen(argv, "r")))
      { printf("Error: can't open file %s\n", argv);
        return 0;
      }
    }
  else
    { printf("Error: wrong input\n");
      return 0;
    }
  return input;
}


int iSize(FILE* input)
{ char buffer; int n=0;
  while(!feof(input))
  { buffer=fgetc(input);
    if(buffer=='\n')
        n++;
  }
  rewind(input);
  return n;
```

```
}


iString *makeTable(FILE *input, int size)
{ iString *table = new iString[size];
  for(int i=0; i<size; i++)
    table[i].string= new char[N];
  int l=N*sizeof(char);
  for(int i=0; i<size; i++)
  { fscanf(input, "%ld", &table[i].key);
    fgets(table[i].string, l, input);
  }
 return table;
}


void printBoard()
{ printf("+---+------------------------------"
         "--------------------------------\n" );
}

void printTable(iString *table, int size)
{ printBoard(); printf("|KEY|STRING\n");
  printBoard();
  for(int i=0; i<size; i++)
  { printf("|%3ld|", table[i].key);
    printf(" %s", table[i].string);
  }
  printBoard();
  printf("\n\n");
}


void swap(iString *table, int first, int second)
{ iString buffer;
  buffer = table[first];
  table[first] = table[second];
  table[second] = buffer;
}


void reverse(iString *table, int size)
{ for(int i=0; i<size/2; i++)
    swap(table, i, size-i-1);
  is_sorted=0;
}


void mix(iString *table, int size)
{ int first, second;
  for(int i=0; i<size/2; i++)
  { first=rand()%size;
    second=rand()%size;
    swap(table, first, second);
  }
  is_sorted=0;
}


void sortTable(iString *table, int size)
{ int s=1, i, j;
  while(s<size) s*=2;
  while(s>0)
```

```
  { s=(s-1)/2;
    for(i=0; i<size-s; i++)
    for(j=i+s; j-s>=0; j-=s)
      if(table[j-s].key>table[j].key)
        swap(table, j-s, j);
      else break;
  }
  is_sorted=1;
}


int binSearch(iString *table, int size, long x)
{ if(!is_sorted)
  { printf("Error: table isn`t sorted\n");
    return -1;
  }
  int l=0, r=size-1, m;
  while(l<=r)
  { m=rand()%(r-l)+l;
    if(table[m].key==x) return m;
    if(table[m].key<x) l=m+1;
    else r=m;
  }
  return -1;
}


void menu()
{ printf("\n====================\n"
        "_____Menu_____\n"
        "1. Print table\n"
        "2. Binary search\n"
        "3. Sort\n"
        "4. Mix\n"
        "5. Reverse\n"
        "6. Menu\n"
        "0. Exit\n"
        "====================\n");
}

int main(int argc, char *argv[])
{ FILE *input; int size;
  iString *table;
  randomize();
  input = openFile(argc,argv[1]);
  if(!input)
    return 1;
  size=iSize(input);
  table = makeTable(input, size);
  int action=6;
  while(action)
  { if(action==1)
      printTable(table, size);
    else if(action==2)
    { printf("\nInput key\n");
      long x; int i;
      scanf("%ld", &x);
      i=binSearch(table, size, x);
      if(i<0)
        printf("String not found\n");
      else
        printf("Found the string:\n %s",
                        table[i].string);
```

```
    }
    else if(action==3)
    { sortTable(table, size);
      printf("\nTable was sorted\n");
    }
    else if(action==4)
    { mix(table, size);
      printf("\nTable was mixed\n");
    }
    else if(action==5)
    { reverse(table, size);
      printf("\nTable was reversed\n");
    }
    else if(action==6)
      menu();
    printf("\nInput number of action\n");
    scanf("%d", &action);
  }

return 0;}
yusayu@YS:~/Рабочий стол/cppProjects/kr9$ cat input.txt
241   ___   _____ _____________     ___    ____  ______
333    /   | / ___// ____/  _/  _/    /   |  / __ \/_  __/
357   / /| | \__ \/ /     / / / /     / /| | / /_/ / / /
412  / ___ |___/ / /____/ /_/ /     / ___ |/ _, _/ / /
431 /_/  |_/____/\____/___/___/    /_/  |_/_/ |_| /_/
479
500   ___   _____ _____________     ___    ____  ______
742    /   | / ___// ____/  _/  _/    /   |  / __ \/_  __/
788   / /| | \__ \/ /     / / / /     / /| | / /_/ / / /
890  / ___ |___/ / /____/ /_/ /     / ___ |/ _, _/ / /
999 /_/  |_/____/\____/___/___/    /_/  |_/_/ |_| /_/
yusayu@YS:~/Рабочий стол/cppProjects/kr9$ g++ -o kr9.out kr9.cpp
yusayu@YS:~/Рабочий стол/cppProjects/kr9$ ./kr9.out

====================
_____Menu_____
1. Print table
2. Binary search
3. Sort
4. Mix
5. Reverse
6. Menu
0. Exit
====================

Input number of action
1
+---+------------------------------------------------------------
|KEY|STRING
+---+------------------------------------------------------------
|241|    ___   _____ _____________     ___    ____  ______
|333|     /   | / ___// ____/  _/  _/    /   |  / __ \/_  __/
|357|    / /| | \__ \/ /     / / / /     / /| | / /_/ / / /
|412|   / ___ |___/ / /____/ /_/ /     / ___ |/ _, _/ / /
|431|  /_/  |_/____/\____/___/___/    /_/  |_/_/ |_| /_/
|479|
|500|    ___   _____ _____________     ___    ____  ______
|742|     /   | / ___// ____/  _/  _/    /   |  / __ \/_  __/
|788|    / /| | \__ \/ /     / / / /     / /| | / /_/ / / /
|890|   / ___ |___/ / /____/ /_/ /     / ___ |/ _, _/ / /
|999|  /_/  |_/____/\____/___/___/    /_/  |_/_/ |_| /_/
+---+------------------------------------------------------------
```

```
Input number of action
5

Table was reversed

Input number of action
1
+---+------------------------------------------------------------
|KEY|STRING
+---+------------------------------------------------------------
|999|  /_/  |_/____/\____/___/___/    /_/  |_/_/ |_| /_/
|890|   / ___ |___/ / /____/ /_/ /     / ___ |/ _, _/ / /
|788|    / /| | \__ \/ /    / / / /     / /| | / /_/ / / /
|742|     /   | / ___// ____/  _/  _/    /   |  / __ \/_  __/
|500|    ___   _____ _______________     ___    ____  ______
|479|
|431|  /_/  |_/____/\____/___/___/    /_/  |_/_/ |_| /_/
|412|   / ___ |___/ / /____/ /_/ /     / ___ |/ _, _/ / /
|357|    / /| | \__ \/ /    / / / /     / /| | / /_/ / / /
|333|     /   | / ___// ____/  _/  _/    /   |  / __ \/_  __/
|241|    ___   _____ _______________     ___    ____  ______
+---+------------------------------------------------------------



Input number of action
6

====================
_____Menu_____
1. Print table
2. Binary search
3. Sort
4. Mix
5. Reverse
6. Menu
0. Exit
====================

Input number of action
3

Table was sorted

Input number of action
1
+---+------------------------------------------------------------
|KEY|STRING
+---+------------------------------------------------------------
|241|    ___   _____ _______________     ___    ____  ______
|333|     /   | / ___// ____/  _/  _/    /   |  / __ \/_  __/
|357|    / /| | \__ \/ /    / / / /     / /| | / /_/ / / /
|412|   / ___ |___/ / /____/ /_/ /     / ___ |/ _, _/ / /
|431|  /_/  |_/____/\____/___/___/    /_/  |_/_/ |_| /_/
|479|
|500|    ___   _____ _______________     ___    ____  ______
|742|     /   | / ___// ____/  _/  _/    /   |  / __ \/_  __/
|788|    / /| | \__ \/ /    / / / /     / /| | / /_/ / / /
|890|   / ___ |___/ / /____/ /_/ /     / ___ |/ _, _/ / /
|999|  /_/  |_/____/\____/___/___/    /_/  |_/_/ |_| /_/
```

```
+---+------------------------------------------------------------


Input number of action
4

Table was mixed

Input number of action
1
+---+------------------------------------------------------------
|KEY|STRING
+---+------------------------------------------------------------
|241|    ___   _____ _______________     ___    ____  ______
|742|     /   | / ___// ____/  _/  _/    /   |  / __ \/_  __/
|890|   / ___ |___/ / /____/ /_/ /     / ___ |/ _, _/ / /
|412|   / ___ |___/ / /____/ /_/ /     / ___ |/ _, _/ / /
|357|    / /| | \__ \/ /    / / / /     / /| | / /_/ / / /
|479|
|500|    ___   _____ _______________     ___    ____  ______
|999|  /_/  |_/____/\____/___/___/    /_/  |_/_/ |_| /_/
|788|    / /| | \__ \/ /    / / / /     / /| | / /_/ / / /
|333|     /   | / ___// ____/  _/  _/    /   |  / __ \/_  __/
|431|  /_/  |_/____/\____/___/___/    /_/  |_/_/ |_| /_/
+---+------------------------------------------------------------


Input number of action
3

Table was sorted

Input number of action
1
+---+------------------------------------------------------------
|KEY|STRING
+---+------------------------------------------------------------
|241|    ___   _____ _______________     ___    ____  ______
|333|     /   | / ___// ____/  _/  _/    /   |  / __ \/_  __/
|357|    / /| | \__ \/ /    / / / /     / /| | / /_/ / / /
|412|   / ___ |___/ / /____/ /_/ /     / ___ |/ _, _/ / /
|431|  /_/  |_/____/\____/___/___/    /_/  |_/_/ |_| /_/
|479|
|500|    ___   _____ _______________     ___    ____  ______
|742|     /   | / ___// ____/  _/  _/    /   |  / __ \/_  __/
|788|    / /| | \__ \/ /    / / / /     / /| | / /_/ / / /
|890|   / ___ |___/ / /____/ /_/ /     / ___ |/ _, _/ / /
|999|  /_/  |_/____/\____/___/___/    /_/  |_/_/ |_| /_/
+---+------------------------------------------------------------


Input number of action
2

Input key

500
Found the string:
    ___   _____ _______________     ___    ____  ______
```

```
Input number of action
5

Table was reversed

Input number of action
2

Input key
999
Error: table isn`t sorted
String not found

Input number of action
0
```

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

10. Замечания автора по существу работы _____

11. **Выводы**

В ходе работы я научился создавать и отлаживать программы сортировки и поиска в массиве на Си

Недочёты при выполнении задания могут быть устранены следующим образом: _____

Подпись студента _____