**Отчет по лабораторной работе № IX** по курсу Практикум на ЭВМ

Студент группы М8О-110Б-21 Агеева Алиса № по списку 2

Контакты e-mail, telegram, skype

Работа выполнена: « » \_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

Преподаватель: доцент каф. 806 Никулин Сергей Петрович

Отчет сдан « » \_\_\_\_\_\_\_\_\_20 \_\_\_ г., итоговая оценка \_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* 1. **Тема:** Сортировка и поиск.

1. **Цель работы:** Составить программу на языке Си с использованием процедур и функций для сортировки таблицы заданным методом и двоичного поиска по ключу в таблице.
2. **Задание** (*вариант №* **2**) **:** Метод сортировки: Линейный выбор с подсчетом

Структура таблицы:

****

1. **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 i3 с ОП 4 Гб НМД 128 Гб. Монитор 1920x1080~60Hz

Другие устройства \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **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 Windows 10

интерпретатор bash

Система программирования Code::Blocks IDE версия 20.03

Утилиты операционной системы gcc start

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)

Опишем следующие структуры:

struct str\_int{

int value;

char\* string;

};

struct Node

{

struct str\_int key;

char\* data;

struct Node\* next;

};

struct List

{

struct Node\* head;

int size;

struct Node\* end;

};

В которых хранятся ключ и данные и саму структура Листа

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

Алгоритм программы:

Вводятся строки с ключом, сначала строковая часть ключа, потом целочисленная.

Введенная таблица сортируется.

Далее в ней можно произвести бинарный поиск.

Сравнение происходит лексикографически по строковой части ключа, и если совпадает, то сравнивается целая часть.

**8. Распечатка протокола** (подклеить листинг окончательного варианта программы с тестовыми примерами,подписанныйпреподавателем).

**list.h**

**#ifndef KP9\_LIST\_H**

**#define KP9\_LIST\_H**

**#include "stdio.h"**

**#include <stdlib.h>**

**#include "string.h"**

**#include <stdbool.h>**

**struct str\_int{**

**int value;**

**char\* string;**

**};**

**struct Node**

**{**

**struct str\_int key;**

**char\* data;**

**struct Node\* next;**

**};**

**struct List**

**{**

**struct Node\* head;**

**int size;**

**struct Node\* end;**

**};**

**struct List createList(){**

**struct List list;**

**list.head = NULL;**

**list.size = 0;**

**return list;**

**}**

**typedef struct Node\* ListIterator;**

**void set\_element(struct Node\* \_node, struct str\_int key, char\* data){**

**if (\_node != NULL){**

**\_node->key = key;**

**\_node->data = data;**

**}**

**}**

**void set\_next\_element(struct Node\* \_node, struct Node\* \_next){**

**if (\_node != NULL){**

**\_node->next = \_next;**

**}**

**}**

**bool is\_list\_empty(struct List\* \_list){**

**return \_list->size == 0;**

**}**

**ListIterator end(struct List\* \_list){**

**return \_list->end;**

**}**

**ListIterator begin(struct List\* \_list){**

**if (\_list != NULL && \_list->head != NULL){**

**return \_list->head;**

**}**

**else**

**return end(\_list);**

**}**

**ListIterator next(ListIterator It){**

**return It->next;**

**}**

**ListIterator plus(struct List\* \_list, ListIterator It, int \_n){**

**if (is\_list\_empty(\_list)){**

**return begin(\_list);**

**}**

**ListIterator ansIterator = It;**

**int i = 0;**

**while (i < \_n && next(ansIterator) != end(\_list)){**

**ansIterator = next(ansIterator);**

**++i;**

**}**

**return ansIterator;**

**}**

**int distance(ListIterator \_begin, ListIterator \_end){**

**int ans = 0;**

**ListIterator curIt = \_begin;**

**while (curIt != \_end){**

**++ans;**

**curIt = next(curIt);**

**}**

**return ans;**

**}**

**void increase(struct List\* \_list, ListIterator \_lIt, int \_n){**

**int i = 0;**

**while (i < \_n && \_lIt != end(\_list)){**

**\_lIt = next(\_lIt);**

**}**

**}**

**int getListLength(struct List\* \_list){**

**return distance(begin(\_list), end(\_list));**

**}**

**void pushBack(struct List\* \_list, struct str\_int key, char\* data){**

**struct Node\* newNode = (struct Node\*) malloc(sizeof(struct Node));**

**set\_element(newNode, key, data);**

**if (is\_list\_empty(\_list)){**

**set\_next\_element(newNode, end(\_list));**

**\_list->head = newNode;**

**++\_list->size;**

**return;**

**}**

**struct Node\* lastNode = begin(\_list);**

**while (next(lastNode) != end(\_list)){**

**lastNode = next(lastNode);**

**}**

**set\_next\_element(lastNode, newNode);**

**set\_next\_element(newNode, end(\_list));**

**++\_list->size;**

**}**

**void popBack(struct List\* \_list){**

**ListIterator prevNode = plus(\_list, begin(\_list), \_list->size - 2);**

**if (getListLength(\_list) > 1){**

**ListIterator deletedNode = next(prevNode);**

**free(deletedNode);**

**set\_next\_element(prevNode, end(\_list));**

**--\_list->size;**

**}**

**else if (getListLength(\_list) == 1){**

**free(\_list->head);**

**\_list->head = end(\_list);**

**--\_list->size;**

**}**

**}**

**void printList(struct List\* \_list){**

**printf("list: \n");**

**struct Node\* curNode = begin(\_list);**

**while (curNode != end(\_list)){**

**printf("%s%s\t%d\t", "key: ", curNode->key.string, curNode->key.value);**

**printf("%s%s\n", "data: ", curNode->data);**

**curNode = next(curNode);**

**}**

**printf("\n");**

**}**

**void destroyList(struct List\* \_list){**

**struct Node\* deletedNode = begin(\_list);**

**if (deletedNode == end(\_list)){**

**return;**

**}**

**struct Node\* nextNode = next(deletedNode);**

**while (nextNode != end(\_list)){**

**printf("%s %s\n", deletedNode->data, nextNode->data);**

**free(deletedNode->key.string);**

**free(deletedNode->data);**

**free(deletedNode);**

**deletedNode = nextNode;**

**nextNode = next(nextNode);**

**}**

**free(deletedNode->key.string);**

**free(deletedNode->data);**

**free(deletedNode);**

**}**

**bool tableSorted = false;**

**void linear\_sort(struct List\* \_list){**

**struct Node\* curNode = begin(\_list);**

**while(next(curNode) != end(\_list)){**

**if(strcmp(curNode->key.string, next(curNode)->key.string) > 0){**

**struct Node tmp = { .key = curNode->key, .data = curNode->data };**

**curNode->key = curNode->next->key;**

**curNode->data = curNode->next->data;**

**next(curNode)->key = tmp.key;**

**next(curNode)->data = tmp.data;**

**curNode = begin(\_list);**

**}**

**else if (strcmp(curNode->key.string, next(curNode)->key.string) == 0){**

**if(curNode->key.value > next(curNode)->key.value){**

**struct Node tmp = { .key = curNode->key, .data = curNode->data };**

**curNode->key = next(curNode)->key;**

**curNode->data = next(curNode)->data;**

**next(curNode)->key = tmp.key;**

**next(curNode)->data = tmp.data;**

**curNode = begin(\_list);**

**}**

**else curNode = next(curNode);**

**}**

**else curNode = next(curNode);**

**}**

**tableSorted = true;**

**}**

**struct Node\* get\_Item\_by\_index(ListIterator begin, int index){**

**ListIterator curIterator = begin;**

**for (int i = 0; i < index; i++){**

**curIterator = next(curIterator);**

**}**

**return curIterator;**

**}**

**void ValuebyKey(struct List\* \_list, struct str\_int \_key){**

**int count = 0;**

**struct Node\* curNode = begin(\_list);**

**while (curNode != end(\_list)){**

**if ((strcmp(curNode->key.string, \_key.string) == 0) && (curNode->key.value == \_key.value)){**

**printf("%s%s", "Data by key: ", curNode->data);**

**count += 1;**

**}**

**curNode = next(curNode);**

**}**

**if (count == 0){**

**printf("No data by that key has founded");**

**}**

**}**

**void binarysearch(struct List\* \_list, ListIterator begin, ListIterator end, struct str\_int key){**

**bool element\_found = false;**

**int left = 0;**

**int right = distance(begin, end) - 1;**

**int mid = (distance(begin, end) + 0) / 2;**

**while(left <= right){**

**mid = (left + right) / 2;**

**if (strcmp(key.string, get\_Item\_by\_index(begin, mid)->key.string) == 0 && key.value == get\_Item\_by\_index(begin, mid)->key.value){**

**element\_found = true;**

**printf("%s%d%s", "element found. it's position is ", mid);**

**ValuebyKey(\_list, key);**

**break;**

**}**

**if (strcmp(key.string, get\_Item\_by\_index(begin, mid)->key.string) < 0 ||**

**(strcmp(key.string, get\_Item\_by\_index(begin, mid)->key.string) == 0 && key.value < get\_Item\_by\_index(begin, mid)->key.value)){**

**right = mid - 1;**

**}**

**else left = mid + 1;**

**}**

**if (element\_found == false) printf("%s", "element not found");**

**}**

**#endif //KP9\_LIST\_H**

**main.c**

**#include "list.h"**

**int main(){**

**printf("+ - add el into end of table, s - sort table, ? - search by key\n");**

**struct List list = createList();**

**int symb;**

**while ((symb = getchar()) != EOF) {**

**switch (symb) {**

**case '+': {**

**struct str\_int key;**

**char text[100];**

**char\* data = (char\*) malloc(sizeof(data));**

**char\* string = (char\*) malloc(sizeof(string));**

**int value;**

**fgets(text, 100, stdin);**

**sscanf(text," %s%d%s", string, &value, data);**

**key.string = string;**

**key.value = value;**

**pushBack(&list, key, data);**

**break;**

**}**

**case 'p': {**

**printList(&list);**

**break;**

**}**

**case '-': {**

**popBack(&list);**

**break;**

**}**

**case 's':{**

**linear\_sort(&list);**

**break;**

**}**

**case '?':{**

**struct str\_int key;**

**char text[100];**

**char\* string = (char\*) malloc(sizeof(string));**

**int value;**

**struct Node\* curNode = begin(&list);**

**fgets(text, 100, stdin);**

**sscanf(text," %s%d", string, &value); //ввожу ключ (строка + целое)**

**key.string = string;**

**key.value = value;**

**struct str\_int memory;**

**memory.string = key.string;**

**memory.value = key.value;**

**if(tableSorted) binarysearch((&list), begin(&list), end(&list), memory);**

**free(string);**

**break;**

**}**

**}**

**}**

**destroyList(&list);**

**return 0;**

**}**

**Tests:**

+ - add el into end of table, s - sort table, ? - search by key

+ aaa 23 gogo

+ amm 45 doping

+ amm 13 fugire

+ ffkl 12 automat

+ ffg 34 gorod

+ zzz 5666 podoshvoi

+ zzc 45 v

+ zzc 44 met menia

+ fgh 56 kokoro

p

list:

key: aaa 23 data: gogo

key: amm 45 data: doping

key: amm 13 data: fugire

key: ffkl 12 data: automat

key: ffg 34 data: gorod

key: zzz 5666 data: podoshvoi

key: zzc 45 data: v

key: zzc 44 data: met

key: fgh 56 data: kokoro

s

p

list:

key: aaa 23 data: gogo

key: amm 13 data: fugire

key: amm 45 data: doping

key: ffg 34 data: gorod

key: ffkl 12 data: automat

key: fgh 56 data: kokoro

key: zzc 44 data: met

key: zzc 45 data: v

key: zzz 5666 data: podoshvoi

? amm 13

element found. it's position is 1 Data by key: fugire

? zzz 5666

element found. it's position is 8 Data by key: podoshvoi

? ffg 35

element not found

-

p

list:

key: aaa 23 data: gogo

key: amm 13 data: fugire

key: amm 45 data: doping

key: ffg 34 data: gorod

key: ffkl 12 data: automat

key: fgh 56 data: kokoro

key: zzc 44 data: met

key: zzc 45 data: v

+ - add el into end of table, s - sort table, ? - search by key

+ ahg 23 uu

+ agh 22 uu

+ agh 21 o

+ rrrrt 67 ratata

+ rt 34 rat

+ uoi 34 korotaska

+ jop 67 ioi

+ opo 1 priv

+ opq 1 priv

+ jip 12 povin

p

list:

key: ahg 23 data: uu

key: agh 22 data: uu

key: agh 21 data: o

key: rrrrt 67 data: ratata

key: rt 34 data: rat

key: uoi 34 data: korotaska

key: jop 67 data: ioi

key: opo 1 data: priv

key: opq 1 data: priv

key: jip 12 data: povin

s

p

list:

key: agh 21 data: o

key: agh 22 data: uu

key: ahg 23 data: uu

key: jip 12 data: povin

key: jop 67 data: ioi

key: opo 1 data: priv

key: opq 1 data: priv

key: rrrrt 67 data: ratata

key: rt 34 data: rat

key: uoi 34 data: korotaska

? rt 34

element found. it's position is 8 Data by key: rat

? opo 1

element found. it's position is 5 Data by key: priv

? agh 22

element found. it's position is 1 Data by key: uu

-

p

list:

key: agh 21 data: Ё§╞

key: agh 22 data: uu

key: ahg 23 data: uu

key: jip 12 data: povin

key: jop 67 data: ioi

key: opo 1 data: priv

key: opq 1 data: priv

key: rrrrt 67 data: ratata

key: rt 34 data: rat

? opo 3344

element not found

**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. | Дата | Время | Событие | Действие по исправлению | Примечание |
|  | или |  |  |  |  |  |
|  | дом. |  |  |  |  |  |
|  |  |  |  |  |  |  |

1. **Замечания автора** по существу работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Выводы**

Я изучила метод линейной сортировки с подсчетом.

Недочёты при выполнении задания могут быть устранены следующим образом: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_