Московский Авиационный

Институт

(Национальный Исследовательский Университет)

Институт №8

«Фундаментальная информатика и информационные технологии»

КУРСОВОЙ ПРОЕКТ

по курсу «Практикум на ЭВМ»

II семестр

«Сортировка и поиск»

Выполнил: студент

группы M8O-113Б-21

Хасанов Даниил Рафаилович

Преподаватель:

Довженко Анастасия Александровна

Работа сдана

«\_\_»\_\_\_\_\_\_\_\_\_\_\_2022

Оценка\_\_\_\_\_\_

**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 значение элементов неупорядоченной таблицы и проверять работу процедуры сортировки в трёх 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 12.2**

Метод сортировки: быстрая сортировка Хоара (не рекурсивный вариант).

Тип ключа: целый.

Длина ключа в байтах: 4.

Хранение данных и ключей: отдельно.

Число элементов таблицы: 10-14.

**Алгоритм решения**

Проект состоит из восьми частей:

1. main.c – основной файл программы;
2. stack.h, stack.c – реализация стека для сортировки;
3. table.h, table.c – реализация таблицы и работа с ней;
4. keys.txt, stih.txt – ключи и значения соответственно;
5. makefile – сборка программы.

После сборки и запуска программы в терминал выводиться Menu:

Меню:

1) Вывести таблицу

2) Найти элемент в таблице по ключу

3) Отсортировать таблицу

4) Перемешать данные в таблице

5) Перевернуть данные в таблице

6) Меню

7) Выход

Рассмотрим функции реализованных в программе:

Файл stack.c

stackCreate() – создание стека;

freeStack() – удаление стека;

push() – положить элемент в стек;

pop() – достать элемент из стека.

Файл table.c

printTable() – печать таблицы в терминал;

getRow() – заполнение таблицы ключами и значениями;

swapRow() – меняет ключи со значениями местами, нужна для сортировки;

binSearch() – функция бинарного поиска;

quick\_sort() – функция быстрой сортировки Хоара без рекурсии;

shaffle() – перемешивает данные в таблице;

random\_() – возвращает случайное число для перемешивания таблицы;

reverse() – переворачивает таблицу;

isSorted() – функция, проверяющая отсортирована ли таблица.

**Листинг программы**

main.c

#include <stdio.h>

#include <stdlib.h>

#include "table.h"

void menu()

{

printf("Меню:\n");

printf("1) Вывести таблицу\n");

printf("2) Найти элемент в таблице по ключу\n");

printf("3) Отсортировать таблицу\n");

printf("4) Перемешать данные в таблице\n");

printf("5) Перевернуть данные в таблице\n");

printf("6) Меню\n");

printf("7) Выход\n");

}

int main(void)

{

const int N = 50;

int i, size, rule;

int key\_index;

char ch;

row arr[N];

FILE \*keys = fopen("keys.txt", "r");

FILE \*text = fopen("stih.txt", "r");

if (keys == NULL || text == NULL)

{

printf("Файл не существует\n");

return 0;

}

i = 0;

while (i < N && fscanf(keys, "%d", &arr[i].key) == 1)

{

getRow(text, arr[i].stroke, sizeof(arr[i].stroke));

i++;

}

fclose(keys);

fclose(text);

size = i;

menu();

while (1)

{

scanf("%d", &rule);

switch (rule)

{

case 1:

printTable(arr, size);

break;

case 2:

if (!isSorted(arr, size))

{

printf("Таблица неотсортирована\n");

}

else

{

printf("Введите ключ\n");

scanf("%d", &key\_index);

i = binSearch(arr, size, key\_index);

if (i > -1)

{

printf("По ключу %d найден элемент \"%s\"\n", key\_index, arr[i].stroke);

}

else

printf("Элемент с данным ключом не найден\n");

}

break;

case 3:

qiuck\_sort(arr, size);

printf("Таблица отсортирована\n");

break;

case 4:

printf("Таблица перемешана\n");

shuffle(arr, size);

break;

case 5:

printf("Таблица перевёрнута\n");

reverse(arr, size);

break;

case 6:

menu();

break;

case 7:

return 0;

break;

default:

printf("Выберите корректную команду из меню\n");

break;

}

}

return 0;

}

Stack.h

#ifndef STACK\_H

#define STACK\_H

#define STACK\_INIT\_SIZE 100

typedef struct Stack

{

size\_t size;

size\_t limit;

int \*data;

} Stack;

Stack \*createStack();

void freeStack(Stack \*\*s);

void push(Stack \*s, int item);

int pop(Stack \*s);

#endif

Stack.c

#include <stdio.h>

#include <stdlib.h>

#include "stack.h"

Stack \*createStack()

{

Stack \*tmp = (Stack \*)malloc(sizeof(Stack));

tmp->limit = STACK\_INIT\_SIZE;

tmp->size = 0;

tmp->data = (int \*)malloc(tmp->limit \* sizeof(int));

return tmp;

}

void freeStack(Stack \*\*s)

{

free((\*s)->data);

free(\*s);

\*s = NULL;

}

void push(Stack \*s, int item)

{

if (s->size >= s->limit)

{

s->limit \*= 2;

s->data = (int \*)realloc(s->data, s->limit \* sizeof(int));

}

s->data[s->size++] = item;

}

int pop(Stack \*s)

{

if (s->size == 0)

{

exit(7);

}

s->size--;

return s->data[s->size];

}

Table.h

#ifndef TABLE\_H

#define TABLE\_H

#include <stdio.h>

#include "table.h"

typedef struct row

{

int key;

char stroke[150];

} row;

void printTable(row \*arr, const int size);

void getRow(FILE \*stream, char \*str, const int size);

void swapRows(row \*r1, row \*r2);

int binSearch(const row \*arr, const int size, const long int key);

void quick\_sort(row \*a, size\_t size);

void shuffle(row \*arr, const int size);

void reverse(row \*arr, const int size);

int random\_(const int a, const int b);

int isSorted(const row \*arr, const int size);

#endif

Table.c

#include <time.h>

#include <stdlib.h>

#include "table.h"

#include "stack.h"

void printTable(row \*arr, const int size)

{

printf("╔═════════╦══════════════════════════════════════════════════╗\n");

printf("║ Ключ ║ Значение ║\n");

printf("╠═════════╬══════════════════════════════════════════════════╣\n");

for (int i = 0; i < size; i++)

printf("║%9d║%50s║\n", arr[i].key, arr[i].stroke);

printf("╚═════════╩══════════════════════════════════════════════════╝\n");

}

void getRow(FILE \*stream, char \*str, const int size)

{

int cnt = 0;

char ch;

while ((ch = getc(stream)) != '\n' && cnt < size - 1)

str[cnt++] = ch;

str[cnt] = '\0';

}

void swapRows(row \*r1, row \*r2)

{

row tmp;

tmp = \*r1;

\*r1 = \*r2;

\*r2 = tmp;

}

int binSearch(const row \*arr, const int size, const long int key\_index)

{

int start = 0, end = size - 1, mid;

if (size <= 0)

return -1;

while (start < end)

{

mid = start + (end - start) / 2;

if (arr[mid].key == key\_index)

return mid;

else if (arr[mid].key < key\_index)

start = mid + 1;

else

end = mid;

}

if (arr[end].key == key\_index)

return end;

return -1;

}

void quick\_sort(row \*a, size\_t size)

{

size\_t i, j;

int tmp, pivot;

Stack \*lows = createStack();

Stack \*highs = createStack();

size\_t low, high;

push(lows, 0);

push(highs, size - 1);

while (lows->size > 0)

{

low = pop(lows);

high = pop(highs);

i = low;

j = high;

pivot = a[(low + (high - low) / 2)].key;

do

{

while (a[i].key < pivot)

i++;

while (a[j].key > pivot)

j--;

if (i <= j)

{

if (a[i].key > a[j].key)

{

swapRows(&a[i], &a[j]);

}

i++;

if (j > 0)

{

j--;

}

}

} while (i <= j);

if (i < high)

{

push(lows, i);

push(highs, high);

}

if (j > low)

{

push(lows, low);

push(highs, j);

}

}

freeStack(&lows);

freeStack(&highs);

}

int random\_(const int a, const int b)

{

return a + rand() % (b - a + 1);

}

void shuffle(row \*arr, const int size)

{

int i, j, k;

srand((unsigned int)time(0));

for (k = 0; k < size; k++)

{

i = randomtwo(0, size - 1);

j = randomtwo(0, size - 1);

swapRows(&arr[i], &arr[j]);

}

}

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]);

}

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;

}

Работа программы:

reversetm@ReverseTM:~/CP9$ make

gcc -c stack.c -o stack.o

gcc -c table.c -o table.o

gcc -c main.c -o main.o

gcc stack.o table.o main.o -o cp9

reversetm@ReverseTM:~/CP9$ ./cp9

Меню:

1) Вывести таблицу

2) Найти элемент в таблице по ключу

3) Отсортировать таблицу

4) Перемешать данные в таблице

5) Перевернуть данные в таблице

6) Меню

7) Выход

1

╔═════════╦══════════════════════════════════════════════════╗

║ Ключ ║ Значение ║

╠═════════╬══════════════════════════════════════════════════╣

║ 4║Цветешь для радостей, для света, для свободы. ║

║ 7║И сладостных забот, и милых заблуждений! ║

║ 3║Теперь ты юноша — и полною душой ║

║ 1║Брат милый, отроком расстался ты со мной — ║

║ 2║В разлуке протекли медлительные годы; ║

║ 10║Зовешь, вверяясь им, и дружбу и любовь. ║

║ 5║Какое поприще открыто пред тобой, ║

║ 9║Ты сердце пробуешь, в надежде торопливой, ║

║ 6║Как много для тебя восторгов, наслаждений ║

║ 8║Как часто новый жар твою волнует кровь! ║

╚═════════╩══════════════════════════════════════════════════╝

2

Таблица неотсортирована

3

Таблица отсортирована

1

╔═════════╦══════════════════════════════════════════════════╗

║ Ключ ║ Значение ║

╠═════════╬══════════════════════════════════════════════════╣

║ 1║Брат милый, отроком расстался ты со мной — ║

║ 2║В разлуке протекли медлительные годы; ║

║ 3║Теперь ты юноша — и полною душой ║

║ 4║Цветешь для радостей, для света, для свободы. ║

║ 5║Какое поприще открыто пред тобой, ║

║ 6║Как много для тебя восторгов, наслаждений ║

║ 7║И сладостных забот, и милых заблуждений! ║

║ 8║Как часто новый жар твою волнует кровь! ║

║ 9║Ты сердце пробуешь, в надежде торопливой, ║

║ 10║Зовешь, вверяясь им, и дружбу и любовь. ║

╚═════════╩══════════════════════════════════════════════════╝

5

Таблица перевёрнута

1

╔═════════╦══════════════════════════════════════════════════╗

║ Ключ ║ Значение ║

╠═════════╬══════════════════════════════════════════════════╣

║ 10║Зовешь, вверяясь им, и дружбу и любовь. ║

║ 9║Ты сердце пробуешь, в надежде торопливой, ║

║ 8║Как часто новый жар твою волнует кровь! ║

║ 7║И сладостных забот, и милых заблуждений! ║

║ 6║Как много для тебя восторгов, наслаждений ║

║ 5║Какое поприще открыто пред тобой, ║

║ 4║Цветешь для радостей, для света, для свободы. ║

║ 3║Теперь ты юноша — и полною душой ║

║ 2║В разлуке протекли медлительные годы; ║

║ 1║Брат милый, отроком расстался ты со мной — ║

╚═════════╩══════════════════════════════════════════════════╝

4

Таблица перемешана

1

╔═════════╦══════════════════════════════════════════════════╗

║ Ключ ║ Значение ║

╠═════════╬══════════════════════════════════════════════════╣

║ 9║Ты сердце пробуешь, в надежде торопливой, ║

║ 2║В разлуке протекли медлительные годы; ║

║ 3║Теперь ты юноша — и полною душой ║

║ 10║Зовешь, вверяясь им, и дружбу и любовь. ║

║ 7║И сладостных забот, и милых заблуждений! ║

║ 1║Брат милый, отроком расстался ты со мной — ║

║ 6║Как много для тебя восторгов, наслаждений ║

║ 4║Цветешь для радостей, для света, для свободы. ║

║ 8║Как часто новый жар твою волнует кровь! ║

║ 5║Какое поприще открыто пред тобой, ║

╚═════════╩══════════════════════════════════════════════════╝

3

Таблица отсортирована

1

╔═════════╦══════════════════════════════════════════════════╗

║ Ключ ║ Значение ║

╠═════════╬══════════════════════════════════════════════════╣

║ 1║Брат милый, отроком расстался ты со мной — ║

║ 2║В разлуке протекли медлительные годы; ║

║ 3║Теперь ты юноша — и полною душой ║

║ 4║Цветешь для радостей, для света, для свободы. ║

║ 5║Какое поприще открыто пред тобой, ║

║ 6║Как много для тебя восторгов, наслаждений ║

║ 7║И сладостных забот, и милых заблуждений! ║

║ 8║Как часто новый жар твою волнует кровь! ║

║ 9║Ты сердце пробуешь, в надежде торопливой, ║

║ 10║Зовешь, вверяясь им, и дружбу и любовь. ║

╚═════════╩══════════════════════════════════════════════════╝

2

Введите ключ

5

По ключу 5 найден элемент "Какое поприще открыто пред тобой, "

2

Введите ключ

2

По ключу 2 найден элемент "В разлуке протекли медлительные годы; "

7

**Вывод:**