



МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

НГТУ



НЭТИ

Кафедра теоретической и прикладной информатики

Лабораторная работа № 5
по дисциплине «Структуры данных и алгоритмы»
Упорядочение данных



Группа

ПМИ-03

Бригада

9

МАЛЫГИН СЕРГЕЙ

СИДОРОВ ДАНИИЛ

Преподаватели

ЕЛАНЦЕВА ИРИНА ЛЕОНИДОВНА
ТРАКИМУС ЮРИЙ ВИКТОРОВИЧ

Дата

06.01.2026

Новосибирск

1. Условие задачи

Упорядочить таблицу, построенную в лабораторной работе № 4 по новому ключу методом:

- а) прямого включения;
- б) прямого выбора;
- в) прямого обмена;
- г) методом Шелла.

Используя раздел операторов, дополнить элементы таблицы числом раз использования каждого ключа. Для поиска элементов в таблице использовать:

- а) последовательный поиск;
- б) бинарный поиск.

2. Анализ задачи

Входные данные: Данные из файла “owners”.

Выходные данные: Упорядоченная по новому ключу таблица автовладельцев с числом раз использования каждого ключа и результат поиска.

Решение: Возьмем таблицу из лабораторной работы №4 и упорядочим её по новому ключу – номеру машины. Пользователю дается выбор между 4 разными сортировками: сортировка прямого включения, прямого выбора, прямого обмена и методом Шелла. Во время сортировки дополним таблицу числом раз использования каждого ключа. Выведем упорядоченную и расширенную таблицу на консоль. Далее пользователю предлагается найти машину и её автовладельца по номеру машины. Сделать это можно 2 способами: последовательным поиском и бинарным поиском. Выведем результат поиска на консоль.

3. Структура входных и выходных данных

Внешнее представление входных данных:

Данные из файла owners.txt.

Внутреннее представление входных данных:

Таблица в статической памяти для хранения данных автовладельцев и переменная sought_number для номера искомой машины.

```
struct owners
{
    UINT number;
    char name[30];
    char brand[10];
};
struct ownerstable
{
    struct owners S[N];
    UINT n;
};
```

Внешнее представление выходных данных:

Упорядоченная и расширенная по новому ключу таблица автовладельцев и результат поиска. Всё выведено на консоль.

Внутреннее представление выходных данных:

```
struct owners
{
    UINT number;
    char name[30];
    char brand[10];
    UINT use;
};
struct ownerstable
{
    struct owners S[N];
    UINT n;
}T;
```

4. Текст программы

```
#include <fstream>
#include <iostream>
#include <windows.h>

using namespace std;

const size_t N = 20;
const size_t K = 10;
const size_t L = 30;

struct owners
{
    UINT number;
    char name[L];
    char brand[K];
    UINT use;
};
struct ownerstable
{
    struct owners S[N];
    UINT n;
}T;

void inclusionSort()
{
    for (UINT i = 1; i < T.n; i++)
    {
        owners temp = T.S[i];
        UINT index = i;
        while ((index > 0) && (T.S[index - 1].number > temp.number))
        {
            T.S[index-1].use++;
            temp.use++;
            T.S[index] = T.S[index - 1];
            index--;
        }
        T.S[index] = temp;
    }
}

void selectionSort()
{

```

```

    UINT min;
    owners temp;
    for (UINT i = 0; i < T.n-1; i++)
    {
        min = i;
        for (UINT j = i + 1; j < T.n; j++)
        {
            if (T.S[j].number < T.S[min].number)
            {
                T.S[j].use++;
                T.S[min].use++;
                min = j;
            }
        }
        if (min != i)
        {
            temp = T.S[i];
            T.S[i] = T.S[min];
            T.S[min] = temp;
        }
    }
}

void bubbleSort()
{
    for (UINT i = 0; i < T.n - 1; i++)
    {
        for (UINT j = (T.n - 1); j > i; j--)
        {
            if (T.S[j - 1].number > T.S[j].number)
            {
                T.S[j - 1].use++;
                T.S[j].use++;
                owners temp = T.S[j - 1];
                T.S[j - 1] = T.S[j];
                T.S[j] = temp;
            }
        }
    }
}

void shellSort()
{
    UINT increment = 3;
    while (increment > 0)
    {
        for (UINT i = 0; i < T.n; i++)
        {
            UINT j = i;
            owners temp = T.S[i];
            while ((j >= increment) && (T.S[j - increment].number > temp.number))
            {
                T.S[j - increment].use++;
                temp.use++;
                T.S[j] = T.S[j - increment];
                j = j - increment;
            }
            T.S[j] = temp;
        }
        if (increment > 1)
            increment = increment / 2;
        else if (increment == 1)
            break;
    }
}

int sequentialSearch(UINT number)

```

```

{
    for (UINT i = 0; i < N; i++)
    {
        if (T.S[i].number == number)
            return i;
    }
    return -1;
}
int binarySearch(UINT left, UINT right, UINT key)
{
    UINT midd = 0;
    while (1)
    {
        midd = (left + right) / 2;

        if (key < T.S[midd].number)
            if (midd!=0) right = midd - 1; else return -1;
        else if (key > T.S[midd].number)
            left = midd + 1;
        else
            return midd;
        if (left > right) return -1;
    }
}
int main()
{
    setlocale(LC_ALL, "rus");
    ifstream fin;
    UINT number;
    char name[L], brand[K];
    fin.open("owners.txt");
    fin >> T.S[0].number >> T.S[0].brand >> T.S[0].name;
    T.n = 1;
    bool insert = false, limit = true;
    while (!fin.eof() && limit)
    {
        if (T.n < N)
        {
            fin >> number >> brand >> name;
            insert = false;
            for (int i = 0; i < T.n && !insert; i++)
            {
                if (strcmp(T.S[i].name, name) > 0)
                {
                    for (int j = T.n - 1; j >= i; j--)
                    {
                        strcpy_s(T.S[j + 1].name, T.S[j].name);
                        strcpy_s(T.S[j + 1].brand, T.S[j].brand);
                        T.S[j + 1].number = T.S[j].number;
                    }
                    strcpy_s(T.S[i].name, name);
                    strcpy_s(T.S[i].brand, brand);
                    T.S[i].number = number;
                    insert = true;
                    T.n++;
                }
            }
            if (!insert)
            {
                T.S[T.n].number = number;
                strcpy_s(T.S[T.n].name, name);
                strcpy_s(T.S[T.n].brand, brand);
                T.n++;
            }
        }
    }
}

```

```

        else {
            cout << "В таблице больше " << N << " элементов, взяты первые " << N << "
строки!" << endl;
            limit = false;
        }
    }
    fin.close();
    cout << "Построенная таблица в 4 лабе:" << endl;
    for (UINT i = 0; i < T.n; i++)
    {
        cout << T.S[i].number << " " << T.S[i].brand << " " << T.S[i].name << endl;
        T.S[i].use = 0;
    }
    UINT key;
    cout << "Выберите сортировку:" << endl;
    cout << "1 - сортировка прямого включения:" << endl;
    cout << "2 - сортировка прямого выбора:" << endl;
    cout << "3 - сортировка прямого обмена:" << endl;
    cout << "4 - сортировка методом Шелла:" << endl;
    cin >> key;
    switch (key)
    {
        case 1:
            inclusionSort();
            break;
        case 2:
            selectionSort();
            break;
        case 3:
            bubbleSort();
            break;
        case 4:
            shellSort();
            break;
    }
    cout << "Упорядоченная таблица по другому ключу - номеру машины"<<endl;
    for (UINT i = 0; i < T.n; i++)
    {
        cout << T.S[i].number << " " << T.S[i].brand << " " << T.S[i].name << " " <<
T.S[i].use<< endl;
    }
    UINT sought_number;
    cout << "Введите номер машины, которую вы хотите найти:"<<endl;
    cin >> sought_number;
    cout << "Выберите метод поиска:"<<endl;
    cout << "1 - последовательный поиск:" << endl;
    cout << "2 - бинарный поиск:" << endl;
    cin >> key;
    UINT find;
    switch (key)
    {
        case 1:
            find = sequentialSearch(sought_number);
            cout << "Искомый элемент:" << endl;
            if (find != -1) cout << T.S[find].number << " " << T.S[find].brand << " " <<
T.S[find].name << endl; else cout << "Машина с таким номером не найдена!";
            break;
        case 2:
            find = binarySearch(0, N, sought_number);
            cout << "Искомый элемент:" << endl;
            if (find != -1) cout << T.S[find].number << " " << T.S[find].brand << " " <<
T.S[find].name << endl; else cout << "Машина с таким номером не найдена!";
            break;
    }
}

```

}

5. Набор тестов

№	Входные данные				Выходные данные	Примечание
	Файл owners.txt	Номер сортировки	Номер авто	Номер поиска		
1	123 KIA Sidorov 345 BMW Ivanov 404 FORD Petrov 256 GAZ Acinov 300 Mercedes Boosin 204 Toyota Corov 250 Honda Davidov 799 Nissan Jokov 686 Hondai Worotov 999 Ferrari Yaniv 101 Ferrari Kokarev 121 Ferrari Maneskin 412 Toyota Washington 888 GAZ Lobanov 777 BMW Ugodov 666 BMW Akarapev 436 Ferrari Konovalov 467 KIA Horonov 869 Toyota Jovanov 258 FORD Imanbek	1	101	1	Построенная таблица в 4 ла- бе: 256 GAZ Acinov 666 BMW Akarapev 300 Mercedes Boosin 204 Toyota Corov 250 Honda Davidov 467 KIA Horonov 258 FORD Imanbek 345 BMW Ivanov 799 Nissan Jokov 869 Toyota Jovanov 101 Ferrari Kokarev 436 Ferrari Konovalov 888 GAZ Lobanov 121 Ferrari Maneskin 404 FORD Petrov 123 KIA Sidorov 777 BMW Ugodov 412 Toyota Washington 686 Hondai Worotov 999 Ferrari Yaniv Выберите сортировку: 1 - сортировка прямого включения: 2 - сортировка прямого вы- бора: 3 - сортировка прямого об- мена: 4 - сортировка методом Шел- ла: 1 Упорядоченная таблица по другому ключу - номеру ма- шины 101 Ferrari Kokarev 10 121 Ferrari Maneskin 12 123 KIA Sidorov 13 204 Toyota Corov 6 250 Honda Davidov 6 256 GAZ Acinov 5 258 FORD Imanbek 6 300 Mercedes Boosin 7 345 BMW Ivanov 5 404 FORD Petrov 7 412 Toyota Washington 7 436 Ferrari Konovalov 8 467 KIA Horonov 9	Программа работает верно

					666 BMW Akarapev 12 686 Hondai Worotov 4 777 BMW Ugodov 5 799 Nissan Jokov 8 869 Toyota Jovanov 8 888 GAZ Lobanov 6 999 Ferrari Yaniv 0 Введите номер машины, которую вы хотите найти: 101 Выберите метод поиска: 1 - последовательный поиск: 2 - бинарный поиск: 1 Искомый элемент: 101 Ferrari Kokarev	
2	123 KIA Sidorov 345 BMW Ivanov 404 FORD Petrov 256 GAZ Acinov 300 Mercedes Boosin 204 Toyota Corov 250 Honda Davidov 799 Nissan Jokov 686 Hondai Worotov 999 Ferrari Yaniv 101 Ferrari Kokarev 121 Ferrari Maneskin 412 Toyota Washington 888 GAZ Lobanov 777 BMW Ugodov 666 BMW Akarapev 436 Ferrari Konovalov 467 KIA Horonov 869 Toyota Jovanov 258 FORD Imanbek	2	102	2	Построенная таблица в 4 ла- бе: 256 GAZ Acinov 666 BMW Akarapev 300 Mercedes Boosin 204 Toyota Corov 250 Honda Davidov 467 KIA Horonov 258 FORD Imanbek 345 BMW Ivanov 799 Nissan Jokov 869 Toyota Jovanov 101 Ferrari Kokarev 436 Ferrari Konovalov 888 GAZ Lobanov 121 Ferrari Maneskin 404 FORD Petrov 123 KIA Sidorov 777 BMW Ugodov 412 Toyota Washington 686 Hondai Worotov 999 Ferrari Yaniv Выберите сортировку: 1 - сортировка прямого включения: 2 - сортировка прямого вы- бора: 3 - сортировка прямого об- мена: 4 - сортировка методом Шел- ла: 2 Упорядоченная таблица по другому ключу - номеру ма- шины 101 Ferrari Kokarev 1 121 Ferrari Maneskin 1 123 KIA Sidorov 1 204 Toyota Corov 6 250 Honda Davidov 0 256 GAZ Acinov 2 258 FORD Imanbek 2 300 Mercedes Boosin 4 345 BMW Ivanov 2 404 FORD Petrov 3 412 Toyota Washington 1 436 Ferrari Konovalov 6 467 KIA Horonov 7 666 BMW Akarapev 3	Сортировка была проведена, но искомая машина не найдена

					686 Hondai Worotov 1 777 BMW Ugodov 3 799 Nissan Jokov 4 869 Toyota Jovanov 3 888 GAZ Lobanov 2 999 Ferrari Yaniv 0 Введите номер машины, которую вы хотите найти: 102 Выберите метод поиска: 1 - последовательный поиск: 2 - бинарный поиск: 2 Искомый элемент: Машина с таким номером не найдена!	
3		1	230	2	Построенная таблица в 4 ла- бе: 0 Выберите сортировку: 1 - сортировка прямого включения: 2 - сортировка прямого вы- бора: 3 - сортировка прямого об- мена: 4 - сортировка методом Шел- ла: 1 Упорядоченная таблица по другому ключу - номеру ма- шины 0 0 Введите номер машины, кото- рую вы хотите найти: 230 Выберите метод поиска: 1 - последовательный поиск: 2 - бинарный поиск: 2 Искомый элемент: Машина с таким номером не найдена!	Дан пустой файл

6. Результаты работы программы

Программа работает правильно, что подтверждают тесты.