



МИНОБРНАУКИ РОССИИ

*Федеральное государственное бюджетное образовательное учреждение высшего
образования*

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Отчет по выполнению практического задания №5.2

Тема:

**«Алгоритмы поиска в таблице (массиве). Применение алгоритмов поиска
к поиску по ключу записей в файле»**

Дисциплина: «Структуры и алгоритмы обработки данных»

Выполнила студент группы ИКБО-42-23

Туляшева А.Т.

Принял ассистент

Муравьева Е.А.

Москва 2024

Содержание

ЦЕЛЬ РАБОТЫ	3
ЗАДАНИЕ 1.....	3
1.1 Упражнение 1.....	3
1.2 Упражнение 2.....	5
ВЫВОД.....	8

ЦЕЛЬ РАБОТЫ

Получить практический опыт по применению алгоритмов поиска в таблицах данных.

ЗАДАНИЕ 1

1.1 Упражнение 1.

Формулировка упражнения:

Создать двоичный файл из записей (структура записи определена вариантом – смотрите в конце файла). Поле ключа записи в задании варианта подчеркнуто. Заполнить файл данными, используя для поля ключа датчик случайных чисел. Ключи записей в файле уникальны.

Рекомендация: создайте сначала текстовый файл, а затем преобразуйте его в двоичный.

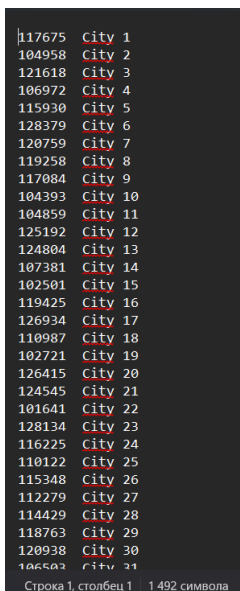
При открытии файла обеспечить контроль существования и открытия файла.

Код программы

Были написаны две функции и структура для данных, которые нужно записать в файл. Заданная вариантом структура:

Справочная межгорода: код города, название города.

Функция `text_file` создает текстовый файл с нужными данными (рис. 1).



```
117675 City 1
104958 City 2
121618 City 3
106972 City 4
115930 City 5
128379 City 6
120759 City 7
119258 City 8
117084 City 9
104393 City 10
104859 City 11
125192 City 12
124804 City 13
107381 City 14
102501 City 15
119425 City 16
126934 City 17
110987 City 18
102721 City 19
126415 City 20
124545 City 21
101641 City 22
128134 City 23
116225 City 24
110122 City 25
115348 City 26
112279 City 27
114429 City 28
118763 City 29
120938 City 30
106503 City 31
```

Строка 1, столбец 1 1 492 символа

Рисунок 1 – Данные в текстовом файле

```

struct record {
    int key;
    char data[100];
};

void text_file(string file_name, int count) { //создание текстового файла
    ofstream file(file_name); //создание файла
    if (!file) {
        cout << "Ошибка при открытии файла\n";
        return;
    }
    srand(static_cast<unsigned>(time(0))); //генератор случайных чисел, зн
    for (int i = 0; i < count; i++) {
        int numb_city = rand() % 900000 + 100000; //код города
        string name = "City " + to_string(i + 1);
        file << numb_city << "\t" << name << endl; //в строку в файл
    }
    file.close();
}

```

Рисунок 2 – Код упражнения 1

Еще была написана функция to_bin, которая преобразовывает текстовый файл в двоичный файл.

```

void to_bin(string text_file, string binary_file) { //преобразование в двоичный файл
    ifstream txt_file(text_file);
    if (!txt_file) {
        cout << "Файл не существует\n";
        return;
    }
    ofstream bin_file(binary_file, ios::binary); //создаем bin_file для записи в двоичный файл
    if (!bin_file) {
        //ios::binary - файл используется в двоичном режиме
        cout << "Ошибка при открытии двоичного файла." << endl;
        return;
    }
    record r;
    while (!txt_file.eof()) {
        txt_file >> r.key;
        txt_file.get();
        txt_file.getline(r.data, 100, '\n');
        //getline(txt_file, r.data);
        bin_file.write((char*)&r, sizeof(record));
    }
    txt_file.close();
    bin_file.close();
}

```

Рисунок 3 – Код упражнения 1

Результаты тестирования

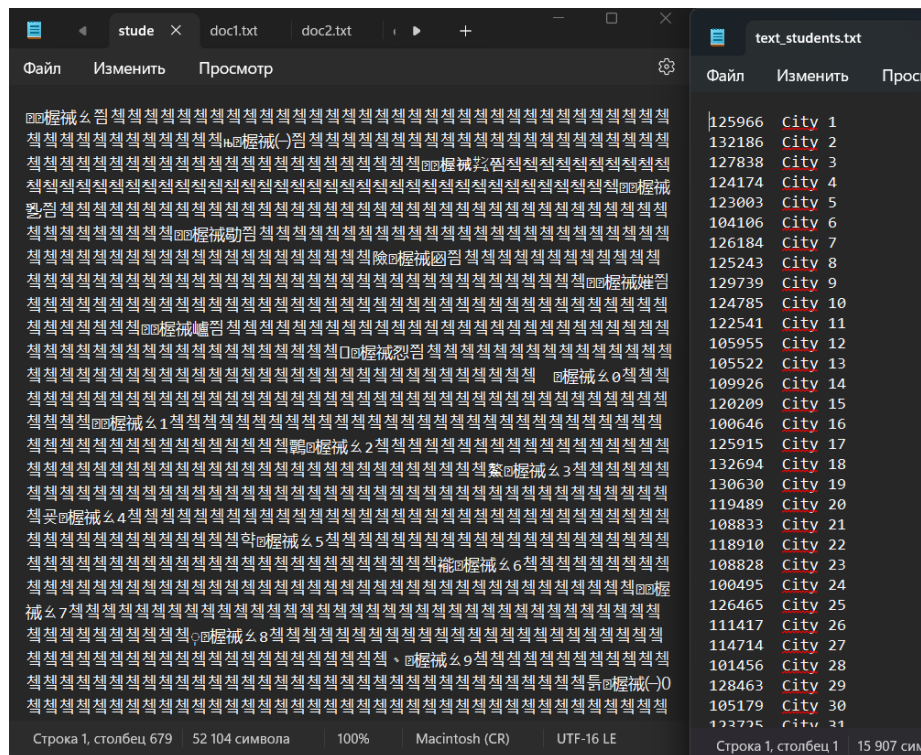


Рисунок 4 – Тестирование упражнения 1

Была написана функция для чтения бинарного файла, которая показала, что программа перевода работает корректно (рис. 5-6).

```
127671 City 1
106758 City 2
114055 City 3
102311 City 4
100491 City 5
122243 City 6
112484 City 7
104533 City 8
130805 City 9
126757 City 10
110011 City 0
```

Рисунок 5 – Тестирование функции перевод из бинарного в текстовый

```
void from_bin(string file_name) { //чтение бинарного файла
    ifstream bin_file(file_name, ios::in | ios::binary);
    record r;
    while (bin_file.read((char*)&r, sizeof(record))) {
        cout << r.key << " " << r.data << "\n";
    }
    bin_file.close();
}
```

Рисунок 6 – Код для чтения бинарного файла

1.2 Упражнение 2.

Формулировка упражнения:

Поиск в файле с применением линейного поиска:

1. Разработать программу поиска записи по ключу в бинарном файле, созданном в первом задании, с применением алгоритма линейного поиска.
2. Провести практическую оценку времени выполнения поиска на файле объемом 100, 1000, 10 000 записей.
3. Составить таблицу с указанием результатов замера времени.

Описание алгоритма:

Начиная с первого, все элементы массива последовательно просматриваются и сравниваются с искомым. Если на каком-то шаге текущий элемент окажется равным искомому, тогда элемент считается найденным, и в качестве результата возвращается информация о нем.

Код программы

Была написана функция для линейного поиска `search_bin`.

```
int search_bin(string file_name, int key) {  
    ifstream bin_file(file_name, ios::in | ios::binary);  
    if (bin_file) {  
        record r;  
        int start = clock();  
        while (bin_file.read((char*)&r, sizeof(record))) {  
            if (r.key == key) {  
                cout << "Искомое значение: " << r.key << " " << r.data << "\n";  
                bin_file.close();  
                cout << "Time: " << clock() - start << " ms\n";  
                return 0;  
            }  
        }  
    }  
    else {  
        cout << "Ошибка при открытии файла\n";  
    }  
}
```

Рисунок 7 – Код упражнения 2

Результаты тестирования

Проведем тестирование на разном количестве записей (табл. 1).

Таблица 1 – Результаты замера времени

Количество записей	Время выполнения (в мс)
100	1
1000	1
10000	3

```
Искомое значение: 110011 City 0  
Time: 1 мс
```

Рисунок 8 – Тестирование упражнения (100 записей)

```
Искомое значение: 110011 City 0  
Time: 1 мс
```

Рисунок 9 – Тестирование упражнения (1000 записей)

```
Искомое значение: 110011 City 9476  
Time: 3 мс
```

Рисунок 10 – Тестирование упражнения (1000 записей)

ВЫВОД

Цель работы была достигнута: получить практический опыт по применению алгоритмов поиска в таблицах данных.