МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Отчет по выполнению практического задания № 4

Тема:

«Алгоритмы внешних сортировок»

Дисциплина: «Структуры и алгоритмы обработки данных»

Выполнил студент: Боргачев Т.М.

Группа: ИНБО-10-23

Москва – 2024

СОДЕРЖАНИЕ

Цель: освоить приёмы сортировки данных из файлов.

1 ФОРМУЛИРОВКА ЗАДАЧИ

1.1 Задание 1

Разработать программу и применить алгоритм внешней сортировки прямого слияния к сортировке файла данных индивидуального варианта 5 по значению ключевого поля «Издательство».

1) Реализовать функцию сортировки (возможно, с вспомогательными функциями) и основную подпрограмму main.

2) Отладить программу, протестировать на примере.

3) Предварительно подготовить файл данных в соответствии с вариантом (не менее 32 записей).

4) Адаптировать программу для сортировки файла с записями, протестировать на подготовленном ранее файле.

5) Определить практическую сложность алгоритма для файлов с увеличивающимся количеством записей (8, 16, 32). Сформировать таблицу результатов, указав количество записей и время сортировки.

1.3 Задание 2

Разработать программу и применить алгоритм сортировки естественного слияния к сортировке файла с данными варианта (файл уже должен быть подготовлен в задании 1).

1) Реализовать функцию сортировки (возможно, с вспомогательными функциями) и основную подпрограмму main.

2) Отладить программу, протестировать на примере.

3) Адаптировать программу для сортировки файла с записями, протестировать на подготовленном ранее файле.

4) Сформировать таблицу результатов, указав количество записей и время сортировки.

2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ

2.1 Алгоритм сортировки прямого слияния для файлов

Фаза разделения:

1. Открыть файл A как входной.

2. Открыть файлы B и C как выходные (для записи).

3. Считываемые из A записи попеременно записываем в файлы B и C.

4. Закрываем файлы A, B, C.

Фаза слияния:

1. Открыть файл A как выходной (для записи).

2. Открываем файлы B и C как входные (для чтения).

3. Установить размер порции сливаемых данных: 1, 2, 4, 8 и т.д. для этого и следующих этапов.

4. Для каждой порции считываются по одной записи из файлов B и C. 5. Меньшая запись записывается в файл A, и считывается очередная запись из того файла, запись которого была переписана в файл A.

6. Пункты 4 и 5 повторяются до тех пор, пока записи очередной порции одного из файлов не будут исчерпаны.

7. Оставшиеся записи из порции другого файла переписываются в файл A.

8. Пункты с 4 по 7 повторяются до тех пор, пока не будет достигнут конец одного из файлов B и C. Тогда оставшиеся записи из другого файла переписываются в файл A.

9. Закрываются файлы A B C. Сортировка завершается тогда, когда длина порции достигнет n.

2.1.1 Реализация Алгоритма в виде функции

Реализация алгоритма в виде блок-схем представлена на рис. 1, 2 и 3.

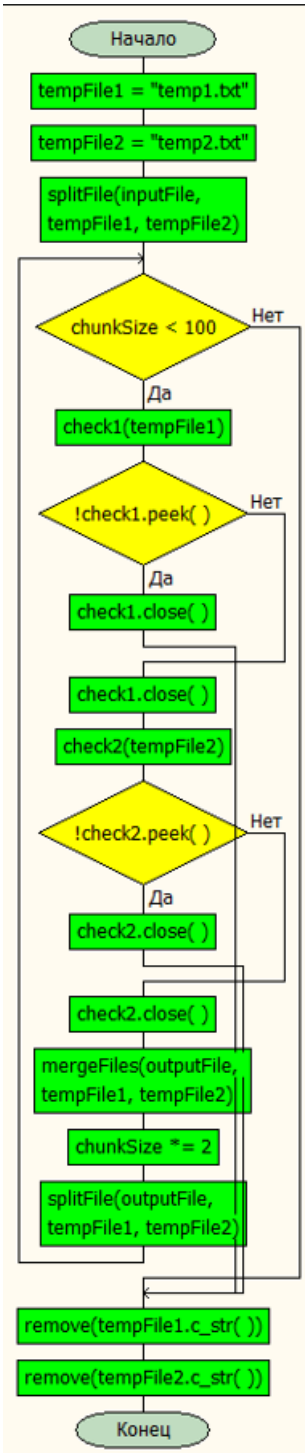

Рисунок 1 – Блок схема функции сортировки вставками для файлов

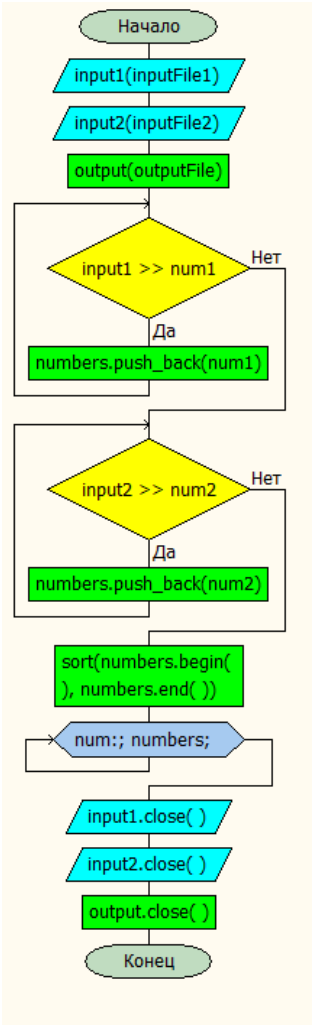

Рисунок 2 – Блок схема функции слияния данных из двух файлов

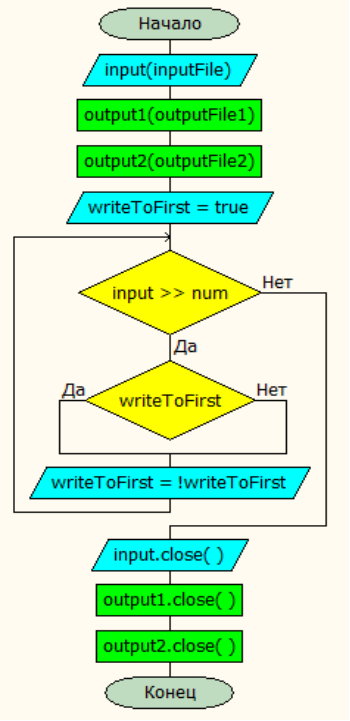

Рисунок 3 – Блок-схема функции разделения данных из файла на два

Код на языке C++, реализующий алгоритм слияния в виде функций представлен на рис. 4, 5 и 6.

```cpp
// Функция для разделения данных из одного файла на два файла
void splitFile(const string& inputFile, const string& outputFile1, const string& outputFile2) {
    ifstream input(inputFile);
    ofstream output1(outputFile1);
    ofstream output2(outputFile2);

    int num;
    bool writeToFirst = true;
    while (input >> num) {
        if (writeToFirst)
            output1 << num << " ";
        else
            output2 << num << " ";

        writeToFirst = !writeToFirst;
    }

    input.close();
    output1.close();
    output2.close();
}
```

Рисунок 4 - Функция для разделения данных из одного файла на два файла

```cpp
// Функция для слияния данных из двух файлов в один файл
void mergeFiles(const string& outputFile, const string& inputFile1, const string& inputFile2) {
    ifstream input1(inputFile1);
    ifstream input2(inputFile2);
    ofstream output(outputFile);

    vector<int> numbers;

    int num1, num2;
    while (input1 >> num1) {
        numbers.push_back(num1);
    }
    while (input2 >> num2) {
        numbers.push_back(num2);
    }

    sort(numbers.begin(), numbers.end());

    for (int num : numbers) {
        output << num << " ";
    }

    input1.close();
    input2.close();
    output.close();
}
```

Рисунок 5 - Функция для слияния данных из двух файлов в один файл

```cpp
// Функция для сортировки файла методом слияния
void mergeSort(const string& inputFile, const string& outputFile, int chunkSize) {
    // Разделение данных из исходного файла на два файла
    string tempFile1 = "temp1.txt";
    string tempFile2 = "temp2.txt";
    splitFile(inputFile, tempFile1, tempFile2);

    // Последовательное слияние данных из двух файлов обратно в исходный
    while (chunkSize < 100) {
        // Проверяем, не пуст ли tempFile1
        ifstream check1(tempFile1);
        if (!check1.peek()) {
            check1.close();
            break; // tempFile1 пуст, выходим из цикла
        }
        check1.close();

        // Проверяем, не пуст ли tempFile2
        ifstream check2(tempFile2);
        if (!check2.peek()) {
            check2.close();
            break; // tempFile2 пуст, выходим из цикла
        }
        check2.close();

        mergeFiles(outputFile, tempFile1, tempFile2);
        chunkSize *= 2;
        splitFile(outputFile, tempFile1, tempFile2);
    }

    // Удаление временных файлов
    remove(tempFile1.c_str());
    remove(tempFile2.c_str());
}
```

Рисунок 6 - Функция для сортировки файла методом слияния

Результаты тестирования алгоритма на примере (8 2 13 4 15 6 9 11 3 7 5 10 1 12 14) представлены на рис. 7.

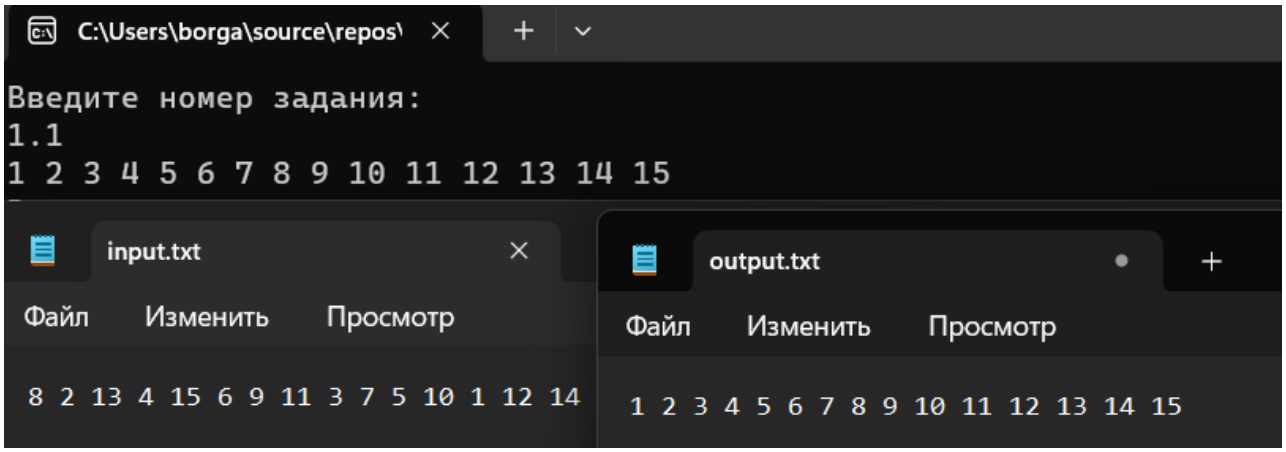

Рисунок 7 – Тестирования алгоритма на примере

Теперь необходимо модернизировать программу для работы с индивидуальным вариантом. Файл с данными представлен на рис. 8.

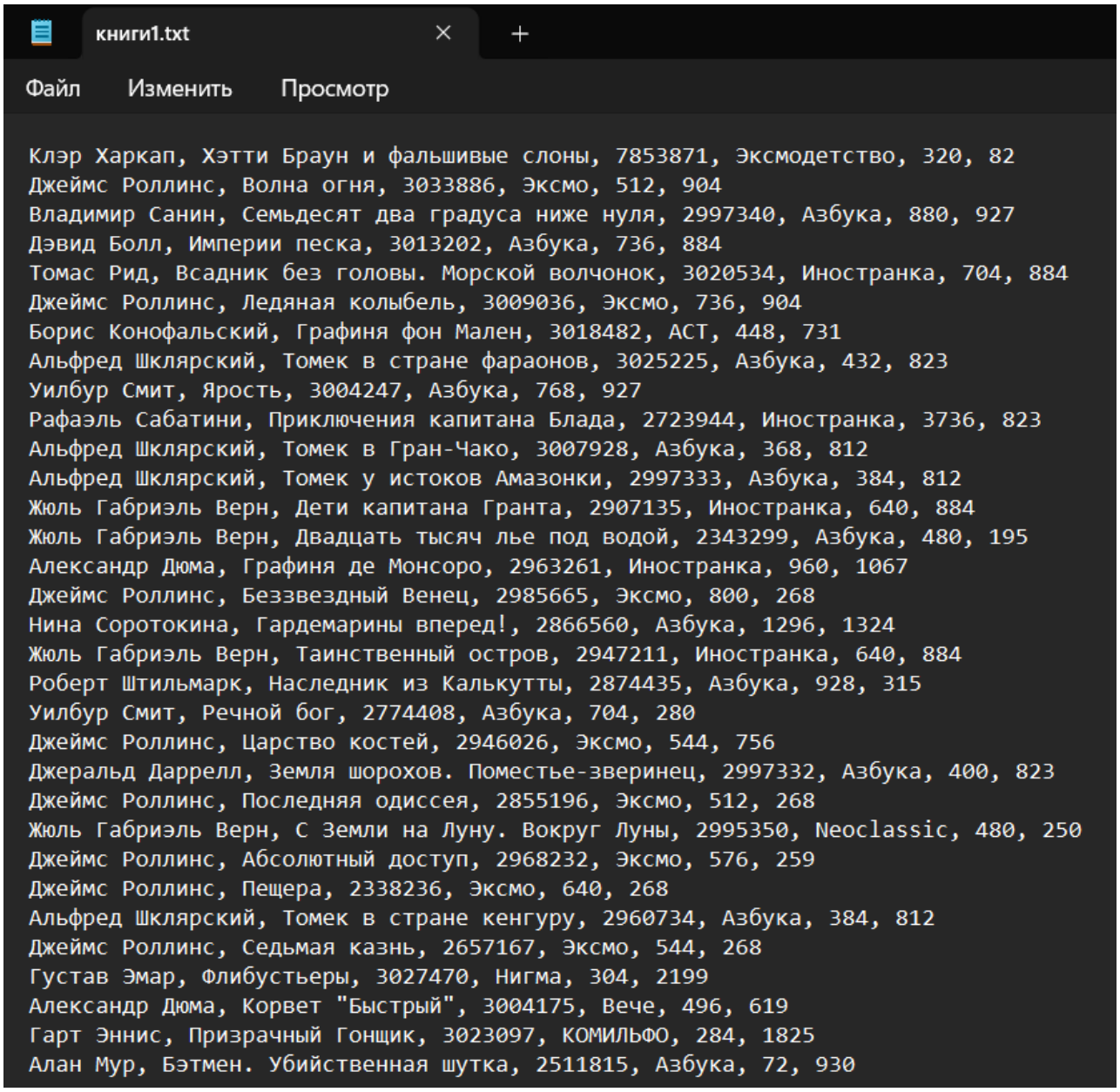

Рисунок 8 – Тестовые данные

Измененные и добавленные функции представлены на рис. 9.

```cpp
struct Book {
    string author;
    string title;
    int ISBN;
    string publisher;
    int pages;
    int price;
};

// Функция для разделения строки на элементы по разделителю ','
vector<string> split(const string& s, char delimiter) {
    vector<string> tokens;
    string token;
    istringstream tokenStream(s);
    while (getline(tokenStream, token, delimiter)) {
        tokens.push_back(token);
    }
    return tokens;
}

// Функция для сравнения двух книг по издателю
bool compareBooks(const Book& a, const Book& b) {
    return a.publisher < b.publisher;
}

// Сортировка слиянием для вектора книг
void mergeSort1(vector<Book>& books) {
    if (books.size() <= 1) {
        return;
    }

    auto middle = books.begin() + books.size() / 2;
    vector<Book> left(books.begin(), middle);
    vector<Book> right(middle, books.end());

    mergeSort1(left);
    mergeSort1(right);

    merge(left.begin(), left.end(), right.begin(), right.end(), books.begin(), compareBooks);
}
```

Рисунок 9 – Сортировка слиянием книг

Результаты тестирования алгоритма представлены на рис. 10.

```
Жюль Габриэль Верн,  С Земли на Луну. Вокруг Луны, 2995350,  Neoclassic, 480, 250
Борис Конофальский,  Графиня фон Мален, 3018482,  АСТ, 448, 731
Владимир Санин,  Семьдесят два градуса ниже нуля, 2997340,  Азбука, 880, 927
Дэвид Болл,  Империи песка, 3013202,  Азбука, 736, 884
Альфред Шклярский,  Томек в стране фараонов, 3025225,  Азбука, 432, 823
Уилбур Смит,  Ярость, 3004247,  Азбука, 768, 927
Альфред Шклярский,  Томек в Гран-Чако, 3007928,  Азбука, 368, 812
Альфред Шклярский,  Томек у истоков Амазонки, 2997333,  Азбука, 384, 812
Жюль Габриэль Верн,  Двадцать тысяч лье под водой, 2343299,  Азбука, 480, 195
Нина Соротокина,  Гардемарины вперед!, 2866560,  Азбука, 1296, 1324
Роберт Штильмарк,  Наследник из Калькутты, 2874435,  Азбука, 928, 315
Уилбур Смит,  Речной бог, 2774408,  Азбука, 704, 280
Джеральд Даррелл,  Земля шорохов. Поместье-зверинец, 2997332,  Азбука, 400, 823
Альфред Шклярский,  Томек в стране кенгуру, 2960734,  Азбука, 384, 812
Алан Мур,  Бэтмен. Убийственная шутка, 2511815,  Азбука, 72, 930
Александр Дюма,  Корвет "Быстрый", 3004175,  Вече, 496, 619
Томас Рид,  Всадник без головы. Морской волчонок, 3020534,  Иностранка, 704, 884
Рафаэль Сабатини,  Приключения капитана Блада, 2723944,  Иностранка, 3736, 823
Жюль Габриэль Верн,  Дети капитана Гранта, 2907135,  Иностранка, 640, 884
Александр Дюма,  Графиня де Монсоро, 2963261,  Иностранка, 960, 1067
Жюль Габриэль Верн,  Таинственный остров, 2947211,  Иностранка, 640, 884
Гарт Эннис,  Призрачный Гонщик, 3023097,  КОМИЛЬФО, 284, 1825
Густав Эмар,  Флибустьеры, 3027470,  Нигма, 304, 2199
Джеймс Роллинс,  Волна огня, 3033886,  Эксмо, 512, 904
Джеймс Роллинс,  Ледяная колыбель, 3009036,  Эксмо, 736, 904
Джеймс Роллинс,  Беззвездный Венец, 2985665,  Эксмо, 800, 268
Джеймс Роллинс,  Царство костей, 2946026,  Эксмо, 544, 756
Джеймс Роллинс,  Последняя одиссея, 2855196,  Эксмо, 512, 268
Джеймс Роллинс,  Абсолютный доступ, 2968232,  Эксмо, 576, 259
Джеймс Роллинс,  Пещера, 2338236,  Эксмо, 640, 268
Джеймс Роллинс,  Седьмая казнь, 2657167,  Эксмо, 544, 268
Клэр Харкап,  Хэтти Браун и фальшивые слоны, 7853871,  Эксмодетство, 320, 82
```

Рисунок 10 – Тестирование алгоритма

Практическая сложность алгоритма для различных количеств записей представлена в табл. 1.

Таблица 1 – Практическая сложность алгоритма прямого слияния

Количество записей	Время работы алгоритма, мс
8	11
16	16
32	24

2.2 Алгоритм естественного слияния

Сортировка естественного слияния, рассматривает две сливаемые подпоследовательности, как упорядоченные. Упорядоченные

подпоследовательности принято называть сериями. Пусть исходный файл разделен на два файла, каждый из которых содержит по n – серий (один может содержать n-1 серию). Тогда при слиянии этих файлов будет получен файл из n серий. При каждом проходе число серий уменьшается вдвое, и общее число пересылок в худшем случае равно n log2n, а в среднем меньше. Процесс сортировки заканчивается, если при очередном проходе в файл будет перелита только одна серия.

Для усовершенствования этой сортировки был предложен вариант предварительного разделения данных в файле на серии одной длины, загрузки каждой серии в оперативную память, сортировки этой серии, например, алгоритмом быстрой сортировки, и запись этих серий в исходный файл. Чем длиннее серию возможно выгрузить в память, отсортировать и вернуть в файл, тем эффективнее будет алгоритм самой сортировки. Такое решение предлагается вам исследовать и разобраться в реализации. Рассмотрим алгоритм и его фазы. Он так же является двухфазным.

1. Определить размер свободной оперативной памяти для выгрузки в нее серии из файла. В программе создаем массив для хранения серии buf.

2. Открыть исходный файл A, подлежащий сортировке.

3. Открыть два файла для записи B и C.

4. Считать последовательность данных в количестве достаточном для размещения в массиве buf. Отсортировать в массиве методом внутренней сортировки и записать в файл B.

5. Считать следующую последовательность данных в количестве достаточном для размещения в массиве buf. Отсортировать в массиве методом внутренней сортировки и записать в файл C.

6. Пункты 4 и 5 выполнять, пока все данные из файла A не будут переписаны отсортированными во вспомогательные файлы B и C.

7. Слить данные в файл A сначала из файла B, затем из файла C. Теперь файл A содержит длинные упорядоченные серии, считаем, что данные в сериях упорядочены по возрастанию.

8. Фаза разделения включает поочередную запись серий из A в файлы B и C.

9. Фаза слияния имеет теперь следующий алгоритм:

• Считываем данные из одного и другого файлов, пока $a_i < a_{i+1}$

• После этого считываем следующую серию и так пока один из файлов не станет пустым, тогда серии другого переписываются в файл A

10. Пункты 8 и 9 повторяются пока в файл A, в результате слияния не будет переписана только одна серия.

2.2.1 Задание 2

Реализация алгоритма в виде блок-схем представлена на рис. 11, 12 и 13.

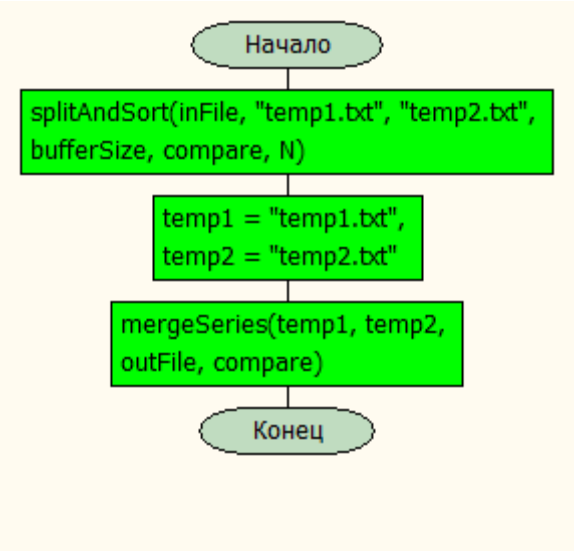

Рисунок 11 – Основная функция алгоритма

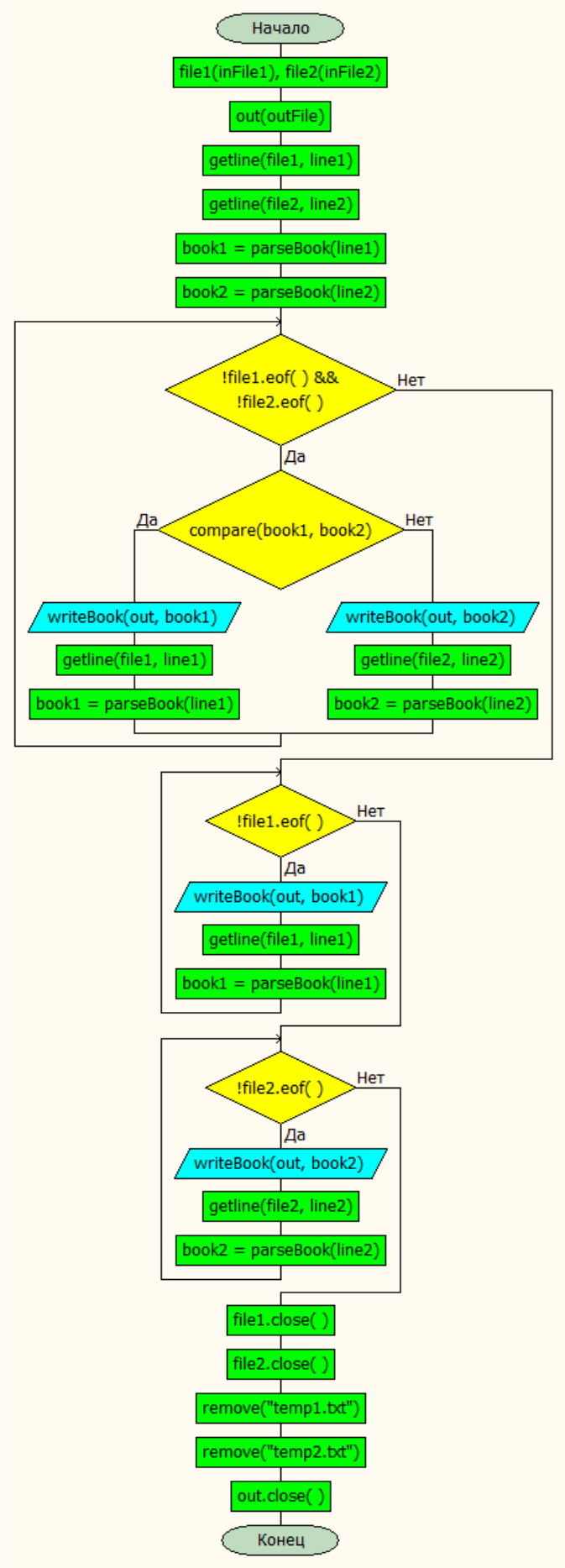

Рисунок 12 – Функция сортировки серий и записи в файл

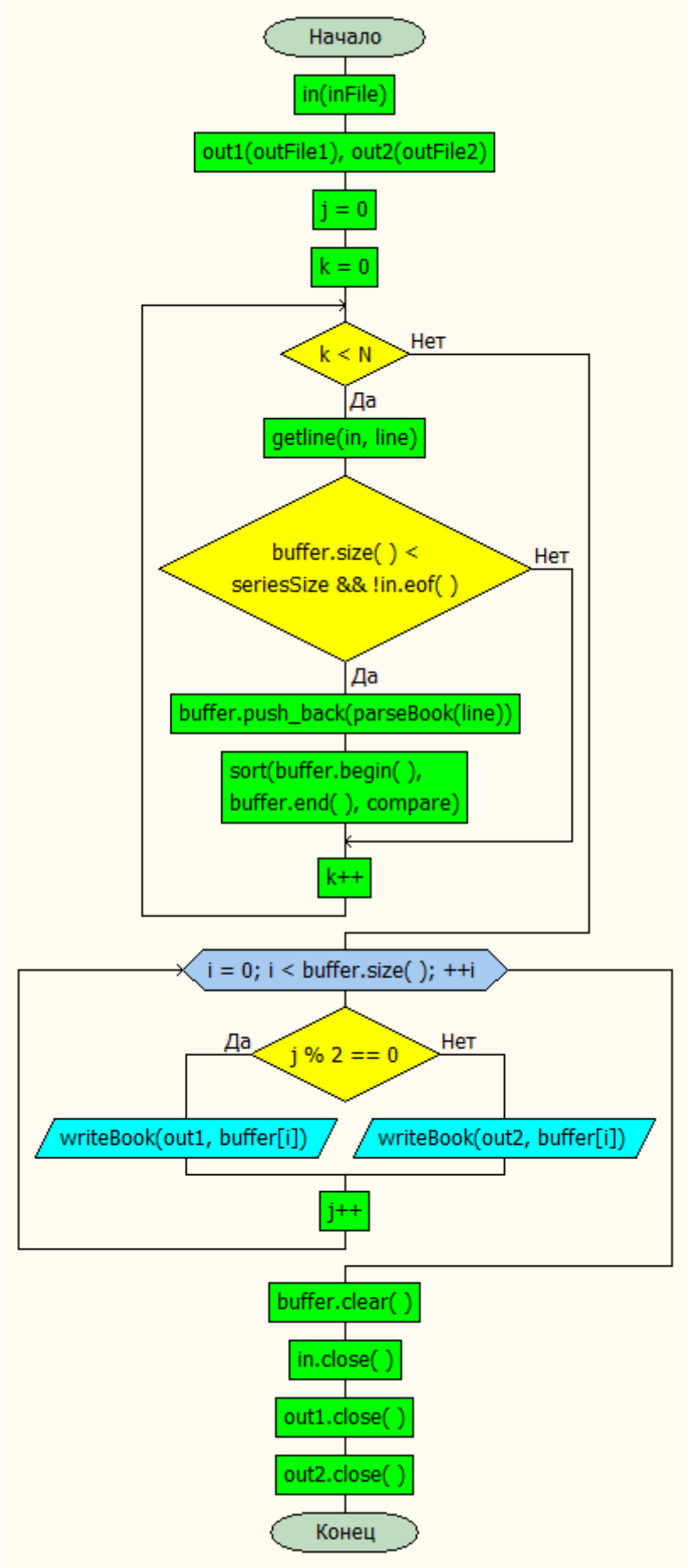

Рисунок 13 – Функция разделения данных на 2 файла

Разработаем алгоритм естественного слияния в виде функций на языке программирования C++. Код реализации представлен на рис. 14, 15 и 16.

```
void naturalMergeSort(const string& inFile, const string& outFile, int bufferSize) {
    splitAndSort(inFile, "temp1.txt", "temp2.txt", bufferSize);

    bool flag = true;
    while (flag) {
        mergeSeries("temp1.txt", "temp2.txt", outFile);
        mergeSeries("temp2.txt", "temp1.txt", outFile);
        flag = false;
        ifstream in(outFile);
        int num, prev;
        in >> prev;
        while (in >> num) {
            if (num < prev) {
                flag = true;
                break;
            }
            prev = num;
        }
        in.close();
    }
}
```

Рисунок 14 – Основная функция алгоритма естественного слияния

```
void splitAndSort(const string& inFile, const string& outFile1, const string& outFile2, int bufferSize) {
    ifstream in(inFile);
    ofstream out1(outFile1), out2(outFile2);

    vector<int> buffer(bufferSize);
    int num;

    while (!in.eof()) {
        int i = 0;
        while (i < bufferSize && !in.eof()) {
            in >> num;
            buffer[i++] = num;
        }
        sort(buffer.begin(), buffer.begin() + i);

        for (int j = 0; j < i; ++j) {
            if (j % 2 == 0) out1 << buffer[j] << " ";
            else out2 << buffer[j] << " ";
        }
    }

    in.close();
    out1.close();
    out2.close();
}
```

Рисунок 15 - Функция для разделения файла на серии

```cpp
void mergeSeries(const string& inFile1, const string& inFile2, const string& outFile) {
    ifstream file1(inFile1), file2(inFile2);
    ofstream out(outFile);

    int num1, num2;
    file1 >> num1;
    file2 >> num2;

    while (!file1.eof() && !file2.eof()) {
        if (num1 < num2) {
            out << num1 << " ";
            file1 >> num1;
        }
        else {
            out << num2 << " ";
            file2 >> num2;
        }
    }

    while (!file1.eof()) {
        out << num1 << " ";
        file1 >> num1;
    }

    while (!file2.eof()) {
        out << num2 << " ";
        file2 >> num2;
    }

    file1.close();
    file2.close();
    out.close();
}
```

Рисунок 16 – Функция для слияния двух серий

Результат работы алгоритма на тестовых данных в виде файла A с записями 17 31 5 59 13 41 43 67 11 23 29 47 3 7 71 2 19 57 37 61 представлен на рис. 17.

Рисунок 17 – Результат работы алгоритма на тестовых данных

Можно сделать вывод о том, что алгоритм работает корректно.

Модернизируем алгоритм так, чтобы он работал со строками из файлов и сортировал их по ключевому элементу.

Код, реализующий алгоритм естественного слияния для файлов по ключевому элементов в виде функций представлен на рис. 18, 19.

```cpp
void splitAndSort(const string& inFile, const string& outFile1, const string& outFile2, int seriesSize, bool (*compare)(const Book&, const Book&), int N) {
    ifstream in(inFile);
    ofstream out1(outFile1), out2(outFile2);

    vector<Book> buffer;
    string line;
    int j = 0;
    int k = 0;
    while (k<N) {
        getline(in, line);
        if (buffer.size() < seriesSize && !in.eof()) {
            buffer.push_back(parseBook(line));
            sort(buffer.begin(), buffer.end(), compare);
        }
        k++;
    }
    for (size_t i = 0; i < buffer.size(); ++i) {
        if (j % 2 == 0) {
            writeBook(out1, buffer[i]);
        }
        else {
            writeBook(out2, buffer[i]);
        }
        j++;
    }

    buffer.clear();
    in.close();
    out1.close();
    out2.close();
}


void naturalMergeSort(const string& inFile, const string& outFile, int bufferSize, bool (*compare)(const Book&, const Book&),int N) {
    splitAndSort(inFile, "temp1.txt", "temp2.txt", bufferSize, compare, N);
    string temp1 = "temp1.txt", temp2 = "temp2.txt";
    mergeSeries(temp1, temp2, outFile, compare);
}
```

Рисунок 18 – Основная функция сортировки и разбиение файлов на временные

```cpp
void mergeSeries(const string& inFile1, const string& inFile2, const string& outFile, bool (*compare)(const Book&, const Book&)) {
    ifstream file1(inFile1), file2(inFile2);
    ofstream out(outFile);

    Book book1, book2;
    string line1, line2;

    // Считываем первую запись из каждого файла
    getline(file1, line1);
    getline(file2, line2);
    book1 = parseBook(line1);
    book2 = parseBook(line2);

    // Пока не достигнем конца обоих файлов
    while (!file1.eof() && !file2.eof()) {
        if (compare(book1, book2)) {
            writeBook(out, book1);
            getline(file1, line1);
            book1 = parseBook(line1);
        }
        else {
            writeBook(out, book2);
            getline(file2, line2);
            book2 = parseBook(line2);
        }
    }

    // Дописываем оставшиеся записи из первого файла
    while (!file1.eof()) {
        writeBook(out, book1);
        getline(file1, line1);
        book1 = parseBook(line1);
    }

    // Дописываем оставшиеся записи из второго файла
    while (!file2.eof()) {
        writeBook(out, book2);
        getline(file2, line2);
        book2 = parseBook(line2);
    }
    file1.close();
    file2.close();
    remove("temp1.txt");
    remove("temp2.txt");
    out.close();

}
```

Рисунок 19 – Функция сортировки файла по сериям и записи в файл

Результаты тестирования алгоритма представлены на рис. 20.

```
Жюль Габриэль Верн, С Земли на Луну. Вокруг Луны, 2995350,   Neoclassic, 480, 250
Борис Конофальский, Графиня фон Мален, 3018482,   АСТ, 448, 731
Дэвид Болл, Империи песка, 3013202,   Азбука, 736, 884
Уилбур Смит, Ярость, 3004247,   Азбука, 768, 927
Альфред Шклярский, Томек у истоков Амазонки, 2997333,   Азбука, 384, 812
Нина Соротокина, Гардемарины вперед!, 2866560,   Азбука, 1296, 1324
Уилбур Смит, Речной бог, 2774408,   Азбука, 704, 280
Альфред Шклярский, Томек в стране кенгуру, 2960734,   Азбука, 384, 812
Владимир Санин, Семьдесят два градуса ниже нуля, 2997340,   Азбука, 880, 927
Альфред Шклярский, Томек в стране фараонов, 3025225,   Азбука, 432, 823
Альфред Шклярский, Томек в Гран-Чако, 3007928,   Азбука, 368, 812
Жюль Габриэль Верн, Двадцать тысяч лье под водой, 2343299,   Азбука, 480, 195
Роберт Штильмарк, Наследник из Калькутты, 2874435,   Азбука, 928, 315
Джеральд Даррелл, Земля шорохов. Поместье-зверинец, 2997332,   Азбука, 400, 823
Александр Дюма, Корвет "Быстрый", 3004175,   Вече, 496, 619
Томас Рид, Всадник без головы. Морской волчонок, 3020534,   Иностранка, 704, 884
Жюль Габриэль Верн, Дети капитана Гранта, 2907135,   Иностранка, 640, 884
Жюль Габриэль Верн, Таинственный остров, 2947211,   Иностранка, 640, 884
Рафаэль Сабатини, Приключения капитана Блада, 2723944,   Иностранка, 3736, 823
Александр Дюма, Графиня де Монсоро, 2963261,   Иностранка, 960, 1067
Гарт Эннис, Призрачный Гонщик, 3023097,   КОМИЛЬФО, 284, 1825
Густав Эмар, Флибустьеры, 3027470,   Нигма, 304, 2199
Джеймс Роллинс, Ледяная колыбель, 3009036,   Эксмо, 736, 904
Джеймс Роллинс, Царство костей, 2946026,   Эксмо, 544, 756
Джеймс Роллинс, Абсолютный доступ, 2968232,   Эксмо, 576, 259
Джеймс Роллинс, Седьмая казнь, 2657167,   Эксмо, 544, 268
Джеймс Роллинс, Волна огня, 3033886,   Эксмо, 512, 904
Джеймс Роллинс, Беззвездный Венец, 2985665,   Эксмо, 800, 268
Джеймс Роллинс, Последняя одиссея, 2855196,   Эксмо, 512, 268
Джеймс Роллинс, Пещера, 2338236,   Эксмо, 640, 268
Клэр Харкап, Хэтти Браун и фальшивые слоны, 7853871,   Эксмодетство, 320, 82
```

Рисунок 20 – Результаты тестирования алгоритма

Данные, дающие оценку практической сложности алгоритма, представлены в табл. 2.

Таблица 2 – Практическая сложность алгоритма естественного слияния

Количество записей	Время выполнения, мс
8	3
16	14
32	12

3 ВЫВОДЫ

В ходе выполнения работы были получены знания и умения работать с сортировками файлов, освоены приёмы сортировки данных из файлов.

По результатам тестирования алгоритмов можно убедиться в корректности их выполнения, как в случае с числовыми данными в файлах, так и со строковыми в случае наличия ключевого элемента.

По данным о практической сложности алгоритма, можно сделать два вывода:

1. Существует зависимость между временем работы алгоритмов от количества записей, с которыми они работают;
2. Алгоритм естественного слияния более эффективен, нежели алгоритм прямого слияния.

4 ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Сартаков М.В., ПР-1.1 (Теоретическая сложность алгоритма) М., МИРЭА — Российский технологический университет – 12 с. - URL: https://online-edu.mirea.ru/pluginfile.php?file=%2F1042738%2Fmod_assign%2Fintroattachment%2F0%2FПР1.1%20%28Теоретическая%20сложность%20алгоритма%29.pdf&forcedownload=1 (дата обращения: 15.02.2024). - Режим доступа: Электронно-облачная система – Cloud MIREA РТУ МИРЭА. - Текст: электронный.

2. Рысин М.Л., Сартаков М.В., Туманова М.Б., Введение в структуры и алгоритмы обработки данных. Ч. 1 - учебное пособие, 2022, МИРЭА – Российский технологический университет. – 2022, 109с. – URL: file:///C:/Users/borga/Downloads/Рысин%20М.Л.%20и%20др.%20Введение%20в%20структуры%20и%20алгоритмы%20обработки%20данных.%20Ч.%201%20-%20учебное%20пособие,%202022.pdf (дата обращения: 15.02.2024). – Режим доступа: Электронно-облачная система – Cloud MIREA РТУ МИРЭА. - Текст: электронный.