МИНИСТЕРСТВО НАУКИ ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
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: Подоплелов А.С.

Группа: ИНБО-21-23

Москва - 2023

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

ЦЕЛЬ РАБОТЫ

Освоить приемы сортировки данных из файлов.

1.1 УСЛОВИЕ ЗАДАНИЯ 1

Разработать программу и применить алгоритм внешней сортировки прямого слияния к сортировке файла данных индивидуального варианта (приложение 1) по значению ключевого поля (ключ в структуре записи варианта – подчеркнутое поле). Мой вариант обладает номером 6: «Сведения об успеваемости одного студента по одной дисциплине: Номер зачетной книжки, Шифр группы, Название дисциплины, Дата получения оценки, Оценка, Фамилия преподавателя»

1.2 УСЛОВИЕ ЗАДАНИЯ 2

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

РЕШЕНИЕ ЗАДАНИЯ 1

Алгоритм состоит из двух этапов: разделение и слияние.

Согласно размеру порции из исходного файла выбирается последовательность элементов и записывается в созданный файл. Когда количество записанных в файл элементов становится равным числу порции, запись происходит в другой файл по тому же критерию остановки записи. Эти действия повторяются до момента, когда все элементы исходного файла не будут переписаны в два других.

Слияние двух файлов в один происходит по следующему правилу: считываются по одной порции из каждого файлов и, посредством сравнения элементов порций обоих файлов в файл записывается отсортированная комбинация двух порций. Продолжается до тех пор, пока не будут достигнуты концы каждого из файлов.

Сами процессы разделения и слияния продолжаются до тех пор, пока размер порции не будет равен количеству записей.

Требование для сортировки: количество сортируемых элементов должно быть степенью двойки, а элементы строго соответствовать шаблону, указанному в индивидуальном задании.

Так как сортировка реализуется не на числовом массиве, а на файле с базой данных, то необходимо реализовать функцию, которая будет извлекать из строки базы необходимый ключ сортировки. (См. Рисунок 1).

```
string key(string l) {
    string zbook;
    int s = 0;
    for (int i = 0; i < size(l); i++) {
        if (l[i] == '|') {
            s++;
        }
        while (s != 5) {
            zbook += l[i];
        }
        if (s == 5) {
            return zbook;
        }
    }
}
```

Рисунок 1 - Реализация функции нахождения и выдачи ключевого элемента на C++

Реализация функция разделения и функция слияния на C++ (См. Рисунок 2).

```
void sp(int size) {
    ifstream file("C:/Users/Леша/Desktop/СИАОД/4.1/4.1/file.txt");
    ofstream A("A.txt");
    ofstream B("B.txt");
    int cnt = 0;
    string l;
    bool perem = 1;
    bool endlA = 1;
    bool endlB = 1;
    if (file.is_open() and A.is_open() and B.is_open()) {
        while (getline(file, l)) {
            A << l << endl;
            for (int i = 0; i < size - 1 and getline(file, l); i++) {
                A << l << "\n";
            }
            for (int i = 0; i < size and getline(file, l); i++) {
                A << l << "\n";
            }
        }
        file.close();
        A.close();
        B.close();
    }
}
int sliv(int size) {
    ofstream file("C:/Users/Леша/Desktop/СИАОД/4.1/4.1/file.txt");
    ifstream A("A.txt");
    ifstream B("B.txt");
    bool first = 1;
    int sizeA = 0;
    int sizeB = 0;
    int port = 0;
    string lA;
    string lB;
    string a;
```

```
    string b;
    if (file.is_open() and A.is_open() and B.is_open()) {
        getline(A, lA);
        getline(B, lB);
        a = key(lA);
        b = key(lB);
        while (A and B) {
            if (sizeA < size and sizeB < size) {
                if (a <= b) {
                    file << lA << "\n";
                    sizeA++;
                    port++;
                    getline(A, lA);
                    a = key(lA);
                }
                else {
                    file << lB << "\n";
                    sizeB++;
                    getline(B, lB);
                    b = key(lB);
                }
            }
            else if(sizeB < size) {
                file << lB << "\n";
                sizeB++;
                getline(B, lB);
                b = key(lB);
            }
            else if (sizeA < size) {
                file << lA << "\n";
                sizeA++;
                port++;
                getline(A, lA);
                a = key(lA);
            }
            else {
```

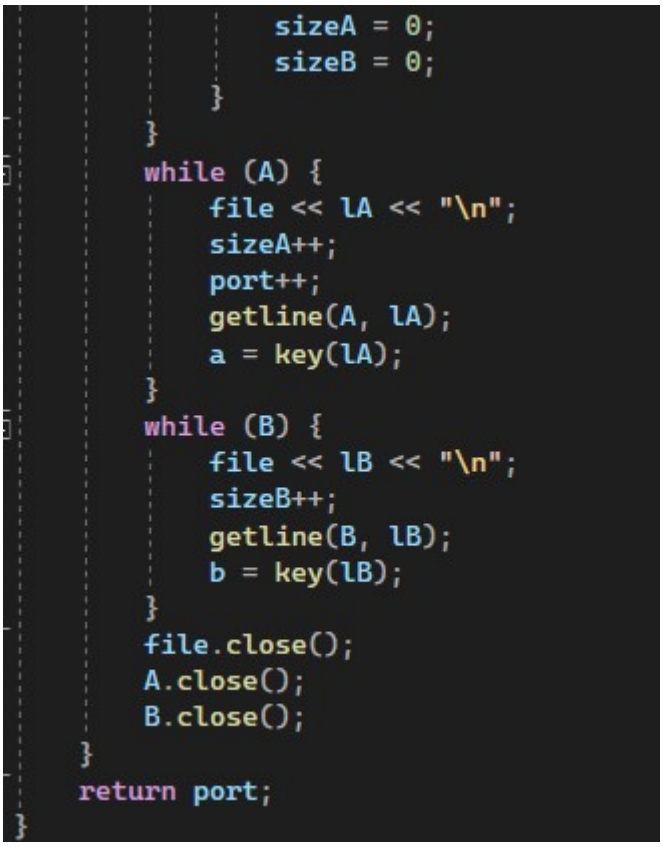


Рисунок 2 - Реализация функция разделения и функция слияния на C++

Функция, объединявшая в себе все написанные раннее функции (См. Рисунок 3). Именно ее время работы измерялось в моей работе далее.

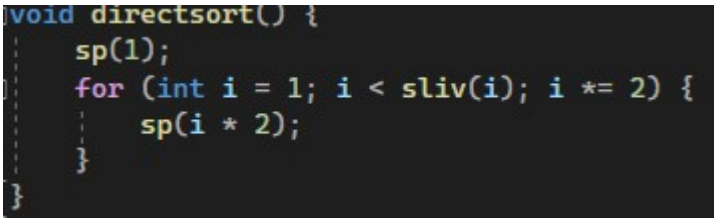


Рисунок 3 — Основная функция, объединяющая все подпрограммы

2.2 ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ

В качестве примера работы программы я возьму базу данных, состоящую из 16 записей. База данных до работы алгоритма (См. Рисунок 4) и версия после выполнения программы (См. Рисунок 5). Так как не ключевые значения не играют роли в работе сортировки, я их оставил одинаковыми для создания акцента на ключевых элементах.

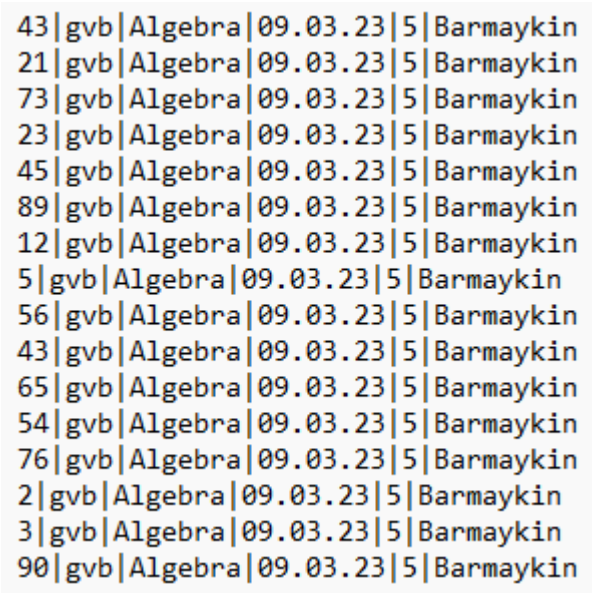


Рисунок 4 — База данных до работы алгоритма

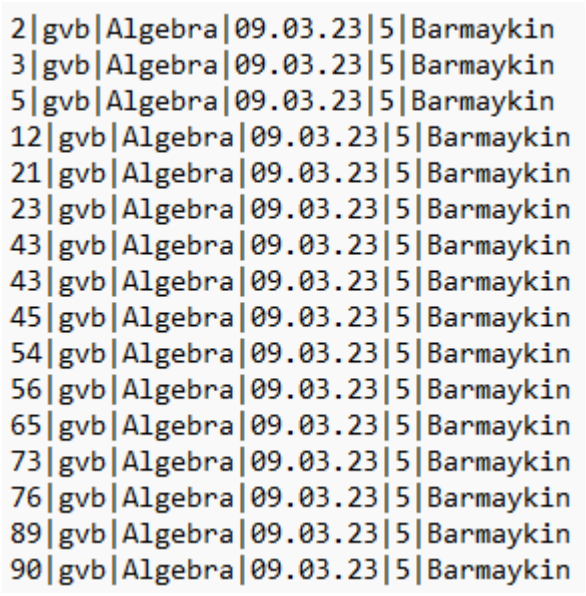


Рисунок 5 — База данных после работы алгоритма

2.3 ТЕСТИРОВАНИЕ РАБОТЫ АЛГОРИТМА

Таблица 1 — Результаты тестирования работы алгоритма сортировки внешним слиянием

Количество строк в исходном файле	**Время работы программы, мс**
8	15
16	37
32	41

64	57
128	74

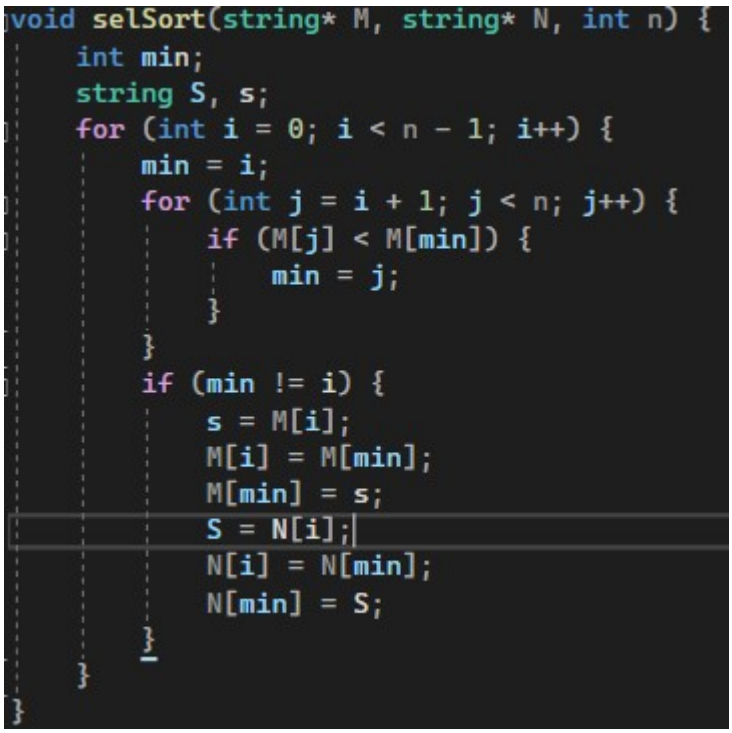
РЕШЕНИЕ ЗАДАНИЯ 2

3.1 ОПИСАНИЕ И РЕАЛИЗАЦИЯ АЛГОРИТМА ВНЕШНЕЙ СОРТИРОВКИ

Первым делом исходный файл разбирается на два фиксированной длины, которые сортируются методом внутренней сортировки (в данном случаем используется сортировка простым выбором). Затем данные копируются обратно в исходный файл.

Алгоритм сортировки естественным слиянием разбивает исходный файл на два по сериям. В каждой серии значения уже отсортированы, при этом размер серии не фиксирован. Конец серии определяется выполнением условия: «i» элемент меньше «i+1». С каждой строки данные считываются в две переменные: в одну будет считываться ключевое поле, а во вторую – вся строка. Процесс разделения и слияния продолжается до тех пор, пока размер серии не будет равен количеству записей.

Алгоритм реализован на языке C++. Функция сортировки выбором продемонстрирована (См. Рисунок 6). Функции разделения фиксированного размера и разделения отсортированного исходного файла (См. Рисунок 7), функция слияния (См. Рисунок 8). Функция, выполняющая внешнюю сортировку прямого слияния (См. Рисунок 9).



```
void selSort(string* M, string* N, int n) {
    int min;
    string S, s;
    for (int i = 0; i < n - 1; i++) {
        min = i;
        for (int j = i + 1; j < n; j++) {
            if (M[j] < M[min]) {
                min = j;
            }
        }
        if (min != i) {
            s = M[i];
            M[i] = M[min];
            M[min] = s;
            S = N[i];
            N[i] = N[min];
            N[min] = S;
        }
    }
}
```

Рисунок 6 — Реализация функции разделения и слияния внешней сортировки прямого слияния

```
void seriesFixedDivideFiles(int size)
{
    // открытие файлов
    ifstream base("C:/Users/User/CLionProjects/prac_4/cmake-build-
debug/Base.txt");
    ofstream fA("A.txt");
    ofstream fB("B.txt");


    string *massId = new string[size];
    string *massStr = new string[size];
    int iMass = 0;


    int j;
    string l, obj;
    bool nextB = true;


    if (base.is_open() && fA.is_open() && fB.is_open())
    {

        getline(base, l);
        obj = Translate(l);

        while (base)
        {

            massId[iMass] = obj;
            massStr[iMass] = l;
            iMass++;


            if (iMass == size)
            {

                SelectionSort(massId, massStr, iMass);


                if (nextB)
                {
                    for (j = 0; j < iMass; j++)
                    {
```

Рисунок 7 — Реализация функции разделения и слияния внешней сортировки прямого слияния

```
                        fA << massStr[i] << endl;
                    }
                    nextB = false;
                } else
                {
                    for (i = 0; i < iMass; i++)
                    {
                        fB << massStr[i] << endl;
                    }
                    nextB = true;
                }
                iMass = 0;

            }


            getline(base, l);
            obj = Translate(l);
        }


        if (iMass > 0)
        {
            SelectionSort(massId, massStr, iMass);
            if (nextB)
            {
                for (i = 0; i < iMass; i++)
                {
                    fA << massStr[i] << endl;
                }
                nextB = false;
            } else
            {
                for (i = 0; i < iMass; i++) {
                    fB << massStr[i] << endl;
                }
                nextB = true;
            }


        }

        base.close();
        fA.close();
        fB.close();
    }

    delete[] massId;
    delete[] massStr;
}
void seriesDivideFiles() {
    ifstream base("C:/Users/User/CLionProjects/prac_4/cmake-build-
debug/Base.txt");
    ofstream fA("A.txt");
    ofstream fB("B.txt");


    string l, lnext;
    string id, idNext;
    bool fileB = true;
```

Рисунок 8 — Реализация функции разделения и слияния внешней сортировки прямого слияния

```
                }
                base << lA << endl;

                idA = idAN;
                lA = lANext;
                getline(fA, lANext);
                idAN = Translate(lANext);

                if (lastInFileA)
                {
                    base << lB << endl;
                    if (!lastInFileB)
                        base << lBNext << endl;
                    break;
                }

            } else
            {
                if (idB > idBN)
                {
                    lastB = true;
                }
                base << lB << endl;

                idB = idBN;
                lB = lBNext;
                getline(fB, lBNext);
                idBN = Translate(lBNext);

                if (lastInFileB)
                {
                    base << lA << endl;
                    if (!lastInFileA)
                        base << lANext << endl;
                    break;
                }
            }
        } else if (!lastB)
        {
            if (idB > idBN)
            {
                lastB = true;
            }
            base << idB << lB << endl;
            idB = idBN;
            lB = lBNext;
            fB >> idBN;
            getline(fB, lBNext);
            if (lastInFileB)
            {
                base << idA << lA << endl;
                if (!lastInFileA)
                    base << idAN << lANext << endl;
                break;
            }
        } else if (!lastA)
        {
            if (idA > idAN)
            {
                lastA = true;
            }
            base << idA << lA << endl;
            idA = idAN;
```

Рисунок 9 — Реализация функции разделения и слияния внешней сортировки прямого слияния

3.2 ТЕСТИРОВАНИЕ АЛГОРИТМА ВНЕШНЕЙ СОРТИРОВКИ ЕСТЕСТВЕННОГО СЛИЯНИЯ

Результаты тестирования программы при различных размерах входных данных представлены в таблице 2.

Таблица 2 — Результаты тестирования алгоритма внешней сортировки естественным слиянием

Количество строк в исходном файле	**Время работы программы, мс**
8	8
16	14
32	23
64	29
128	45

Согласно результатам, вычислительная сложность алгоритма $O(n * \log_2 n)$.

ВЫВОД

Освоены алгоритмы внешних. Реализованы сортировки прямым и естественным слиянием для файла, структура которого выстроена согласно индивидуальному варианту №6. Согласно тестированию, алгоритмы сортировок работают верно. Проанализированы времена работы сортировок при разном количестве строк в исходном файле. Согласно результатам, сортировка естественным слиянием в среднем случае работает немного быстрее сортировки прямым слиянием. Функции роста обоих алгоритмов равны.