МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Отчет по выполнению практического задания 5

Тема:

Алгоритмы внешних сортировок

Выполнил студент Цемкало А. Р.

Фамилия И. О.

группа ИКБО-10-20

Москва 2021

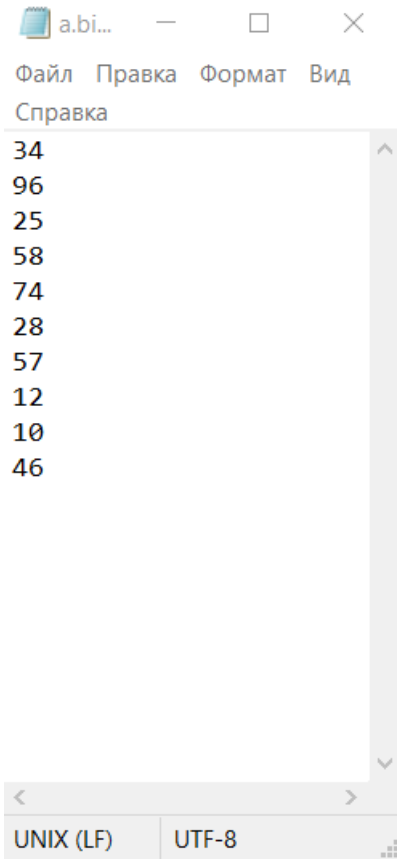
СОДЕРЖАНИЕ

Цель. Получить навыки управления и сортировки файлов.

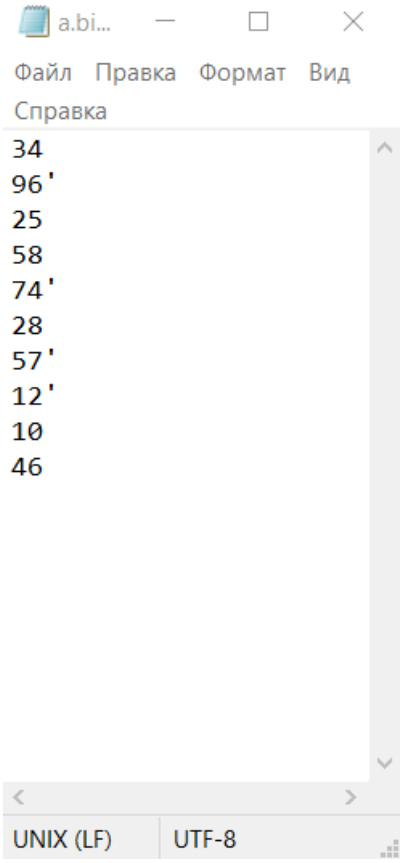
Вариант 10.

1. Демонстрация процесса сортировки методом естественного слияния на массиве, заполненном случайными целыми числами.

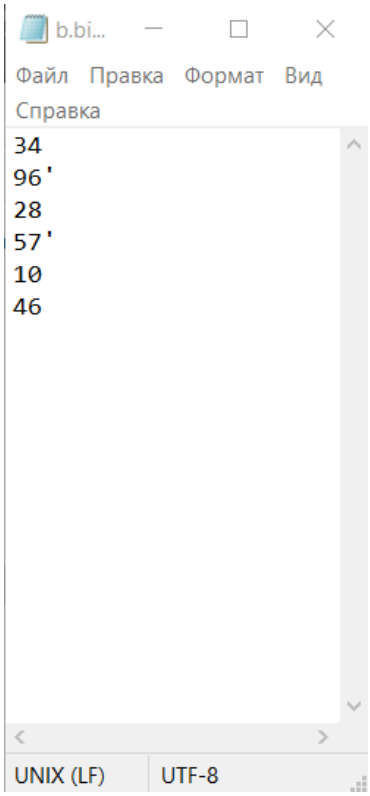
Исходный файл A:



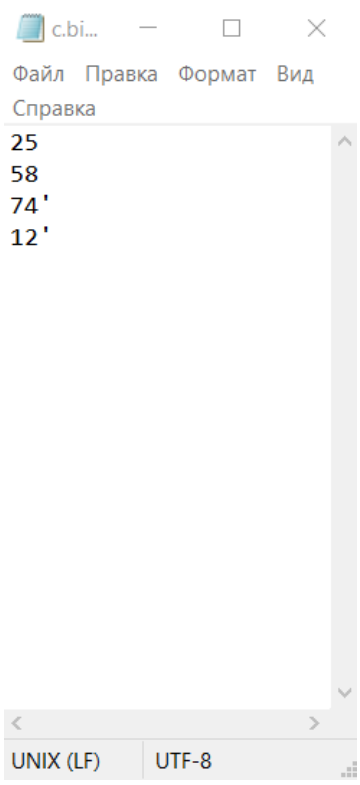
A с выделенными сериями:



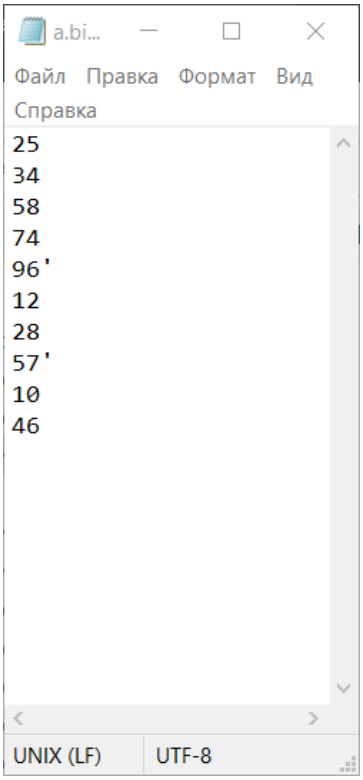
B:


b.bi...
Файл Правка Формат Вид
Справка
34
96'
28
57'
10
46
UNIX (LF)
UTF-8

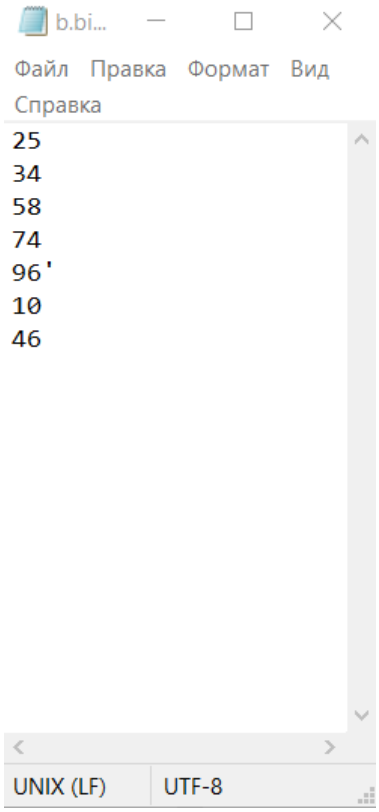
C:


c.bi...
Файл Правка Формат Вид
Справка
25
58
74'
12'
UNIX (LF)
UTF-8

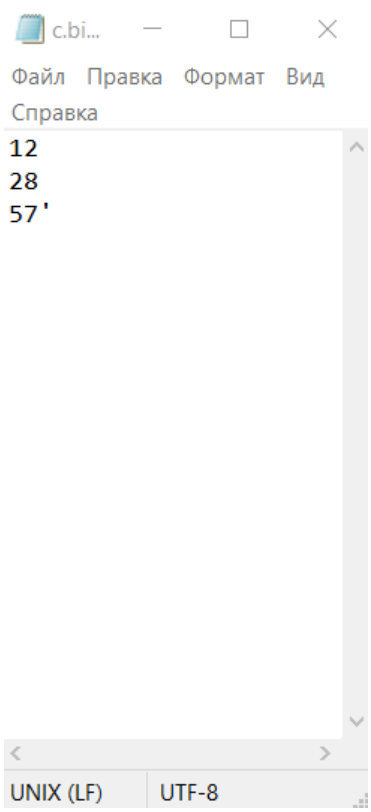
A:


a.bi...
Файл Правка Формат Вид
Справка
25
34
58
74
96'
12
28
57'
10
46
UNIX (LF)
UTF-8

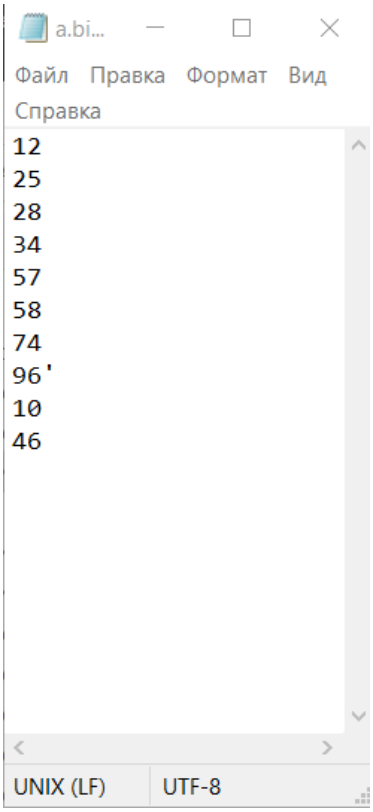
B:


b.bi...
Файл Правка Формат Вид
Справка
25
34
58
74
96'
10
46
UNIX (LF)
UTF-8

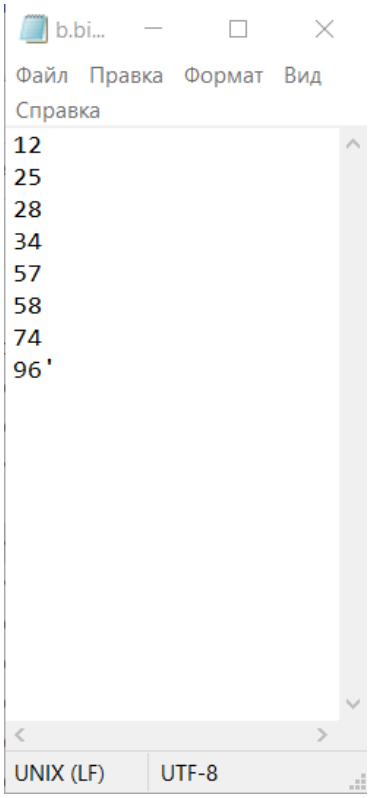
C:


c.bi...
Файл Правка Формат Вид
Справка
12
28
57'
UNIX (LF)
UTF-8

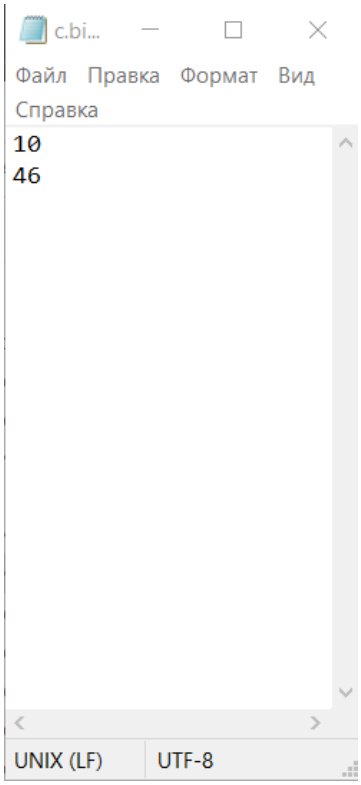
A:


a.bi...
Файл Правка Формат Вид
Справка
12
25
28
34
57
58
74
96'
10
46
UNIX (LF)
UTF-8

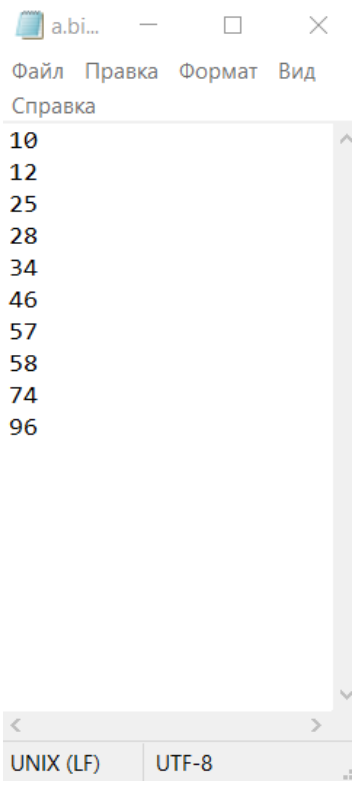
B:



C:

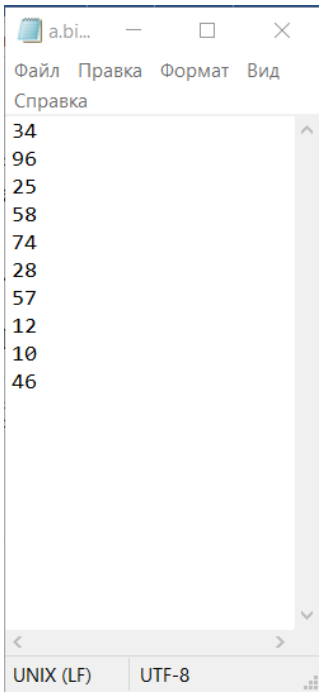


A:

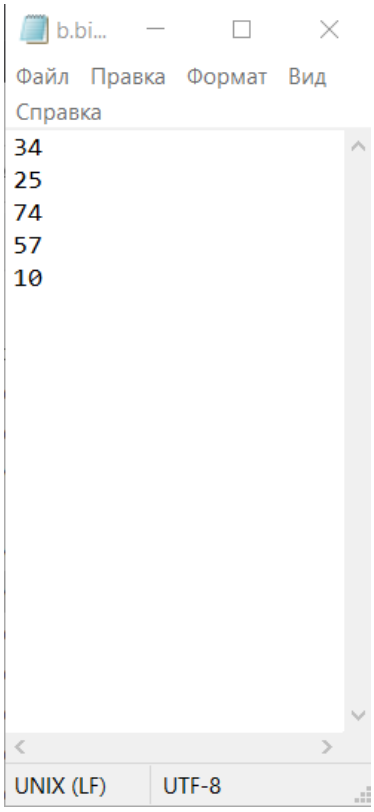


2. Демонстрация процесса сортировки методом прямого слияния на массиве, заполненном случайными целыми числами.

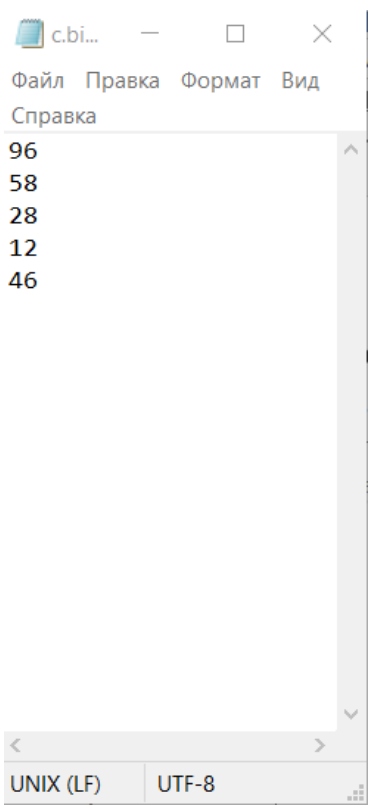
Исходный файл A:



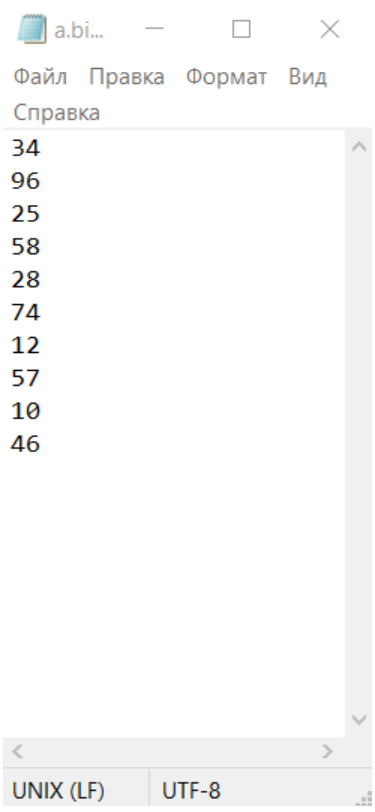
B (порция = 1):



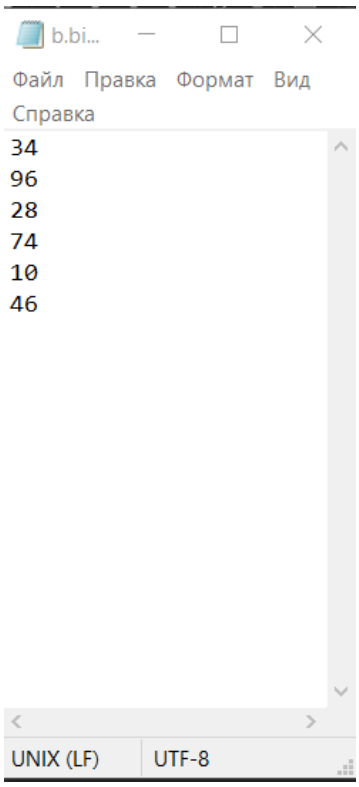
C (порция = 1):



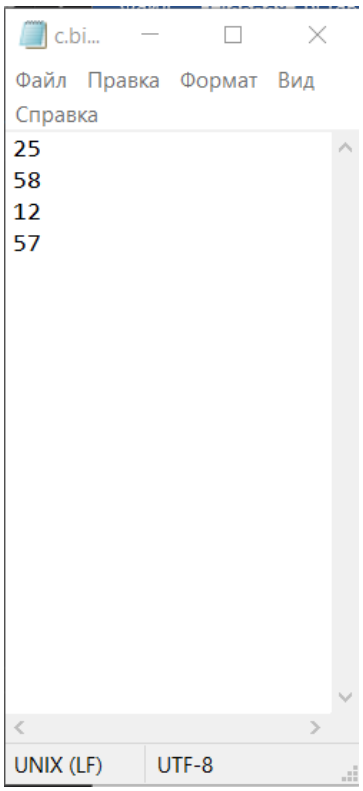
A:



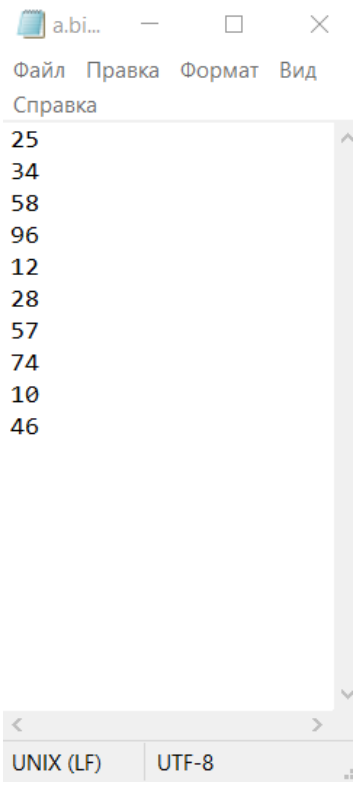
B (порция = 2):



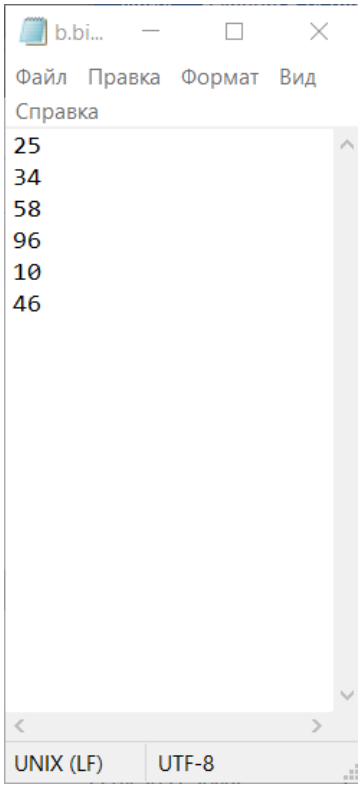
C (порция = 2):



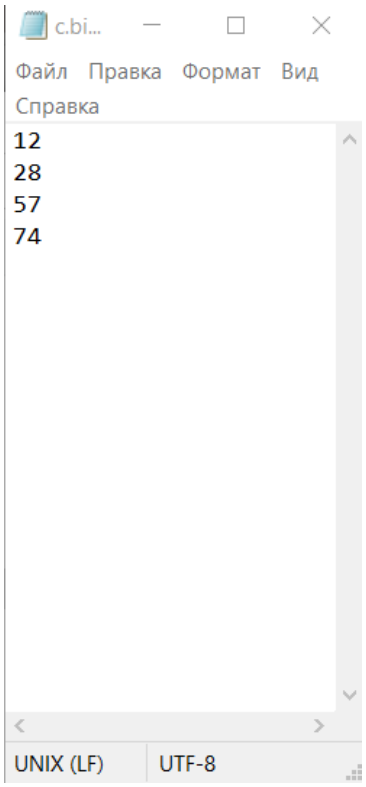
A:



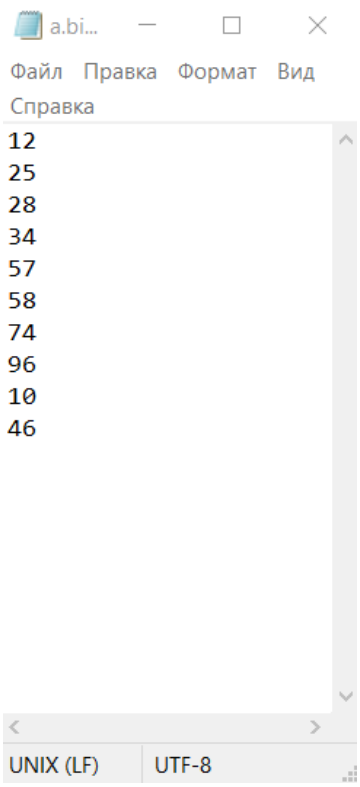
B (порция = 4):



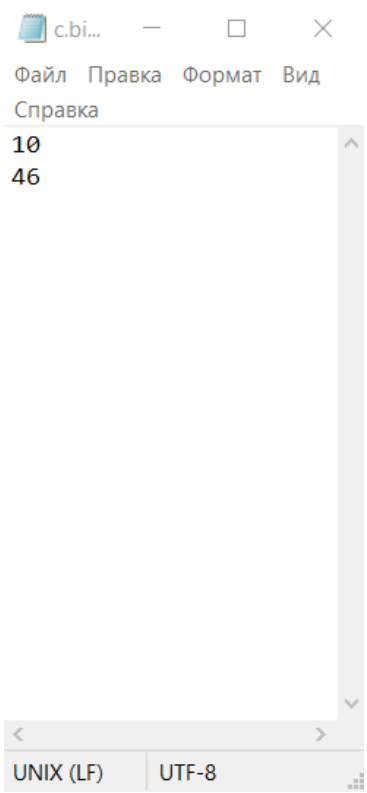
C (порция = 4):



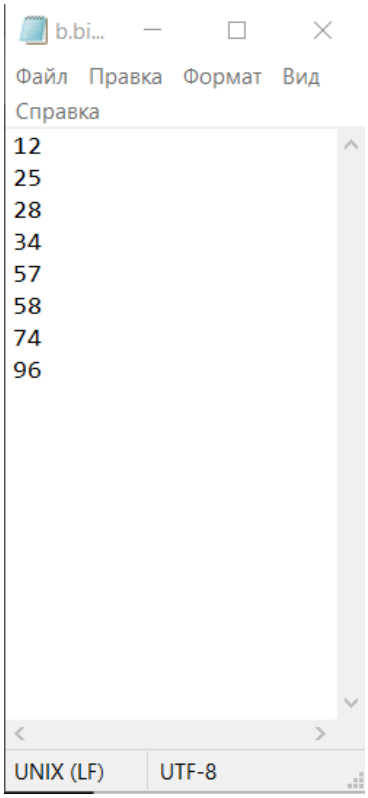
A:



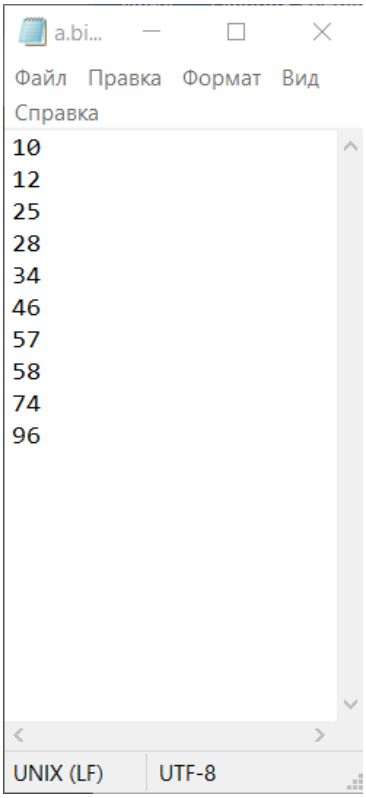
B (порция = 8):



C (порция = 8):



A:



3. Отчёт по разработке программы сортировки прямого слияния

Постановка задачи: разработать программу и применить алгоритм внешней сортировки прямого слияния к сортировке файла данных варианта по значению ключевого поля. Ключ в структуре записи варианта – подчеркнутое поле.

Декомпозиция задачи:

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 A и B.
4. Закрываем файлы A, B, C.

Фаза слияния

1. Открыть файл A как выходкой (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, когда длина порции достигнет n.

Код программы:

```
#include <iostream>
#include <fstream>
#include <chrono>
#include <vector>
using namespace std;

void divide(int portion) {
	ifstream a("a.bin", ios::binary);
	ofstream b;
	ofstream c;
	b.open("b.bin", ios::binary);
	c.open("c.bin", ios::binary);
	string line;
	int n = 0;
	while (getline(a, line, '\n')) {
		if (n < portion) {
			b << line;
			b << '\n';
		}
		else {
			c << line;
			c << '\n';
		}
		n++;
		n %= (portion * 2);
	}
	a.close();
	b.close();
	c.close();
}

void unite(int portion, int n) {
	ofstream a;
	a.open("a.bin", ios::binary);
	ifstream b("b.bin", ios::binary);
	ifstream c("c.bin", ios::binary);
	string line_b, line_c;
	bool eof_b = false;
	bool eof_c = false;
	int enters = 0;
	int portion_b = portion;
	int portion_c = portion;
	while (!eof_b || !eof_c) {
		portion_b = portion - 1;
		portion_c = portion - 1;
		if (!getline(b, line_b, '\n')) {
			eof_b = true;
		}
		if (!getline(c, line_c, '\n')) {
			eof_c = true;
		}
		while (portion_b >= 0 || portion_c >= 0) {
			if (eof_b && eof_c) {
				break;
			}
			if (portion_b < 0) {
				a << line_c;
				a << '\n';
```

```
                        if (portion_c > 0) {
                                if (!getline(c, line_c, '\n')) {
                                        portion_c = 0;
                                        eof_c = true;
                                }
                        }
                        portion_c--;
                }
                else if (portion_c < 0) {
                        a << line_b;
                        a << '\n';
                        if (portion_b > 0) {
                                if (!getline(b, line_b, '\n')) {
                                        portion_b = 0;
                                        eof_b = true;
                                }
                        }
                        portion_b--;
                }
                else {
                        if (line_b < line_c) {
                                a << line_b;
                                a << '\n';
                                if (portion_b > 0) {
                                        if (!getline(b, line_b, '\n')) {
                                                portion_b = 0;
                                                eof_b = true;
                                        }
                                }
                                portion_b--;
                        }
                        else {
                                a << line_c;
                                a << '\n';
                                if (portion_c > 0) {
                                        if (!getline(c, line_c, '\n')) {
                                                portion_c = 0;
                                                eof_c = true;
                                        }
                                }
                                portion_c--;
                        }
                }
        }
    }

    a.close();
    b.close();
    c.close();
}
```

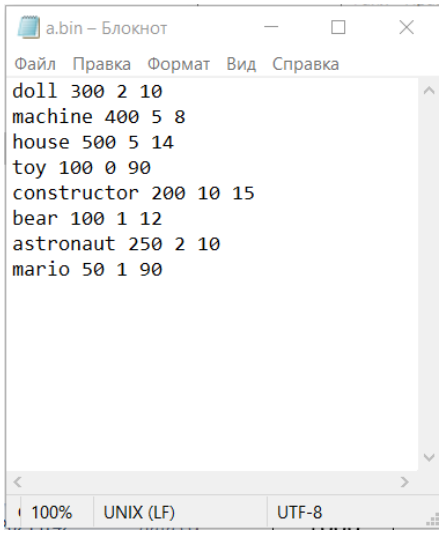
```
void merge_sort() {
	int portion = 1;

	ifstream a("a.bin", ios::binary);
	string line;
	int n = 0;
	while (getline(a, line, '\n')) {
		n++;
	}
	a.close();

	while (portion < n * 2) {
		divide(portion);
		unite(portion, n);
		portion *= 2;
	}
}

int main() {
	using clock_t = chrono::high_resolution_clock;
	using second_t = chrono::duration<double, std::ratio<1> >;
	chrono::time_point<clock_t> start;
	start = clock_t::now();
	merge_sort();
	cout << chrono::duration_cast<second_t>(clock_t::now() - start).count();
	return 0;
}
```

Тестирования:

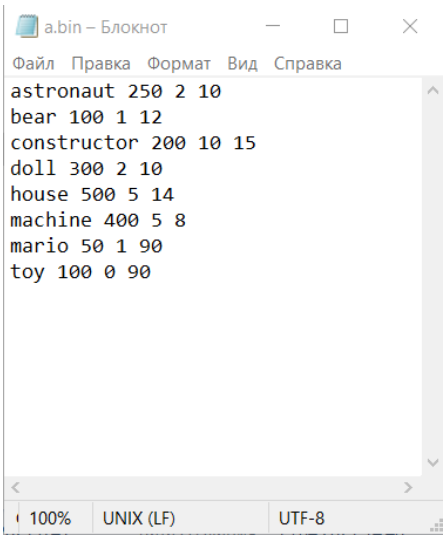
Тестирование сортировки файла данных варианта по значению ключевого поля:

Магазин игрушек. Сведения об игрушке: Название (например: кукла, конструктор и т.д.), стоимость, возрастные границы детей (для кого игрушка предназначена) два поля – начальный возраст и конечный).

Исходный файл A:



Отсортированный файл A:



Тестирование на массивах, заполненных случайными числами:

Таблица 1

n	T(n), сек	Количество записей
100	0.0496986	1614
1000	0.160446	22017
10000	1.00121	300027
100000	8.78746	3600032
1000000	101.553	42000037

4. Отчёт по разработке программы сортировки естественного слияния

Постановка задачи: разработать программу и применить алгоритм сортировки естественного слияния к сортировке файла с данными варианта

Декомпозиция задачи:

Исходный файл разделяется на два файла, каждый из которых содержит по n упорядоченных 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.

Процесс сортировки заканчивается, если при очередном проходе в файл будет перелита только одна серия.

Код программы:

```
#include <iostream>
#include <fstream>
#include <chrono>
#include <vector>
using namespace std;

void divide() {
	ifstream a("a.bin", ios::binary);
	ofstream b("b.bin", ios::binary);
	ofstream c("c.bin", ios::binary);
	string line;
	int k = 0;
	bool new_line_b = false, new_line_c = false;
	while (getline(a, line, '\n')) {
```

```
			if (k % 2 == 0) {
				if (new_line_b) {
					b << '\n';
				}
				new_line_b = true;
				b << line;
			}
			else {
				if (new_line_c) {
					c << '\n';
				}
				new_line_c = true;
				c << line;
			}
			if (line.find("'") != -1) {
				k++;
			}
		}
		a.close();
		b.close();
		c.close();
}


int unite() {
		int s = 0;
		ofstream a;
		a.open("a.bin", ios::binary);
		ifstream b("b.bin", ios::binary);
		ifstream c("c.bin", ios::binary);
		string line_b, line_c;
		bool eof_b = false, eof_c = false;
		int mark_found_b, mark_found_c;
		bool new_line = false;

		while (!eof_b || !eof_c) {
			mark_found_b = false;
			mark_found_c = false;
			if (!getline(b, line_b, '\n')) {
				eof_b = true;
			}
			if (!getline(c, line_c, '\n')) {
				eof_c = true;
			}
			while (!(mark_found_b && mark_found_c)) {
				if (eof_b && eof_c) {
					break;
				}
				if (new_line) {
					a << '\n';
				}
				new_line = true;
				if (mark_found_b || eof_b) {
					if (line_c.find("'") != -1) {
						mark_found_c = true;
						line_c = line_c.substr(0, line_c.find("'"));
					}
					a << line_c;
					if (!mark_found_c) {
						if (!getline(c, line_c, '\n')) {
							eof_c = true;
						}
					}
				}
				else if (mark_found_c || eof_c) {
```

```
                        if (line_b.find("'") != -1) {
                                mark_found_b = true;
                                line_b = line_b.substr(0, line_b.find("'"));
                        }
                        a << line_b;
                        if (!mark_found_b) {
                                if (!getline(b, line_b, '\n')) {
                                        eof_b = true;
                                }
                        }

                }
                else {
                        if (line_b < line_c) {
                                if (line_b.find("'") != -1) {
                                        mark_found_b = true;
                                        line_b = line_b.substr(0, line_b.find("'"));
                                }
                                a << line_b;
                                if (!mark_found_b) {
                                        if (!getline(b, line_b, '\n')) {
                                                eof_b = true;
                                        }
                                }
                        }
                        else {
                                if (line_c.find("'") != -1) {
                                        mark_found_c = true;
                                        line_c = line_c.substr(0, line_c.find("'"));
                                }
                                a << line_c;
                                if (!mark_found_c) {
                                        if (!getline(c, line_c, '\n')) {
                                                eof_c = true;
                                        }
                                }

                        }

                }
                if (mark_found_b) {
                        eof_b = b.eof();
                }
                if (mark_found_c) {
                        eof_c = c.eof();
                }
        }
        if (!(eof_b && eof_c)) {
                a << "'";
                s++;
        }
    }

    a.close();
    b.close();
    c.close();
    return s;
}

int series_divide() {
    int s = 0;
    ifstream a("a.bin", ios::binary);
    ofstream b;
    b.open("b.bin", ios::binary);
    string line_1, line_2;
```

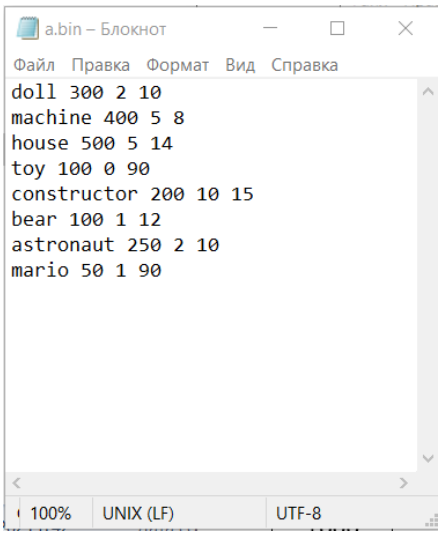
```
	if (getline(a, line_1, '\n')) {
		b << line_1;
		while (getline(a, line_2, '\n')) {
			if (line_2 <= line_1) {
				b << "'";
				s++;
			}
			b << '\n';
			b << line_2;
			line_1 = line_2;
		}
	}

	a.close();
	b.close();

	ifstream bb("b.bin", ios::binary);
	ofstream aa("a.bin", ios::binary);

	while (getline(bb, line_1, '\n')) {
		aa << line_1;
		aa << '\n';
	}

	aa.close();
	bb.close();
	return s;
}

void natural_merge_sort() {
	int s;
	s = series_divide();
	while (s != 0) {
		divide();
		s = unite();
	}
}

int main() {
	using clock_t = chrono::high_resolution_clock;
	using second_t = chrono::duration<double, std::ratio<1> >;
	chrono::time_point<clock_t> start;
	start = clock_t::now();
	natural_merge_sort();
	cout << chrono::duration_cast<second_t>(clock_t::now() - start).count();
	return 0;
}
```

Тестирования:

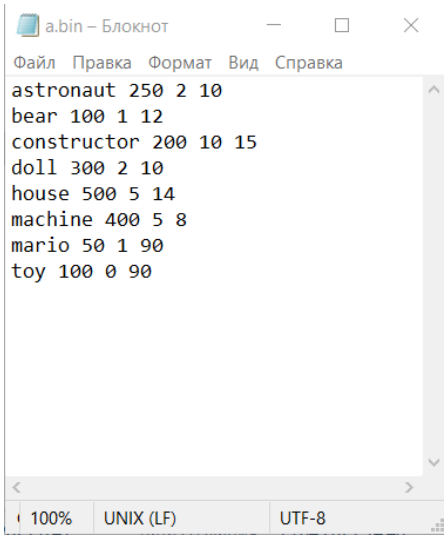
Тестирование сортировки файла данных варианта по значению ключевого поля:

Магазин игрушек. Сведения об игрушке: Название (например: кукла, конструктор и т.д.), стоимость, возрастные границы детей (для кого игрушка предназначена) два поля – начальный возраст и конечный).

Исходный файл A:



Отсортированный файл A:



Тестирование на массивах, заполненных случайными числами:

Таблица 2

n	T(n), сек	Количество записей
100	0.0485471	1200
1000	0.160903	18000
10000	1.09829	260000
100000	8.89786	3200000
1000000	98.023	38000000

5. Выводы об эффективности алгоритмов на основе полученных практических замеров времени выполнения.

Вычислительная сложность обоих алгоритмов равна $O(n*\log_2 n)$, так как сортировки производятся с помощью деления массива (файла) из n элементов на 2 массива (файла). На основе полученных практических замеров времени выполнения можно сделать вывод, что в среднем случае алгоритмы одинаковы по эффективности. Однако в лучшем случае алгоритм естественной сортировки будет эффективнее.

ВЫВОДЫ

В ходе выполнения задания получены практические навыки в:

1. Разработке алгоритмов внешней сортировки прямого и естественного слияний.
2. Управления и сортировки файлов;
3. Оценке эффективности алгоритмов внешних сортировок;
4. В ходе тестирований определено, что алгоритм естественного слияния сортировки эффективнее в лучшем случае, в худшем случае быстрее работает алгоритм сортировки прямым слиянием. В среднем случае (при случайном заполнении файла) алгоритмы работают примерно за одинаковое время.

Тестирования всех операций пройдены успешно.

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

5. Процедурное программирование Языки программирования – Сайт lizochekk! [Электронный ресурс]: URL: https://lizochekk.jimdofree.com/программирование/
6. Документация по Microsoft C/C++ | Microsoft Docs – [Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/cpp/?view=msvc-160
7. C++ – Типизированный язык программирования / Хабр – [Электронный ресурс] URL: https://habr.com/ru/hub/cpp/