***Вариант***

**A-1**(слова из русских букв длины от 1 до 10 (для любой кодировки, т.е. коды букв неизвестны)) **B-1**(по возрастанию)

**C-4** (челночная сортировка) **D-9**(естественное слияние) **E-1** (вход: типизированный файл) **F-2**(выход: текстовый файл)

***Алгоритм работы основной программы***

Программа считывает слова из входного, файла, считает сколько их, создаёт два массива с этими словами (для челночной сортировки и естественного слияния). Затем выполняет челночную сортировку, как описано в книге «Е.А. Бордаченкова, А.А. Панфёров ЗАДАНИЯ ПРАКТИКУМА 1 курс» на странице 29. После челночной сортировки со вторым массивом, копированным с первого в начале выполняется сортировка естественным слиянием. С обоих концов массива находятся упорядоченные части, которые затем сливаются и поочерёдно, то в начало, то в конец, записываются во вспомогательный массив Y, пока он не будет заполнен. После чего эти операции проделываются с массивом Y, пока один из массивов не окажется отсортированным.

***Алгоритм сравнения двух величин***

Сравнение слов выполняется посимвольно, со вспомогательной функцией ord, определяющей порядок буквы в алфавите вида: Аа Бб … . Сравнение прекращается, при обнаружении того, что второе слово «больше» первого.

При этом функция возвращает true, если первое слово идёт в алфавитном порядке раньше (в случае передачи последним параметром false), и false в противном случае. В случае же передачи последним параметром true, функция вернёт true, если первое слово в алфавитном порядке первее, или на том же месте, что и второе.

***Трудности и ошибки***

До прохождения файлов я думала, как будет лучше хранить слова. В каждое дописывать chr(1) до длины 10 или хранить слова в виде записей с двумя полями: количество значащих символов и само слово.

Русские буквы в типе char не идут подряд, поэтому не написать слово из случайных букв используя случайное число из отрезка [ord(‘а’),ord(‘я’)] и применяя к нему chr().

Написать естественное слияние было трудно. Потому что:

1. Не сразу понятно, что значит «*Массив 𝑋 просматривается с начала, определяется наиболее длинный упорядоченныи̮ по неубыванию отрезок; затем массив 𝑋 просматривается с конца и выбирается наиболее длинный упорядоченный (также по неубыванию), отрезок в конце массива*». Можно подумать, что выбираются 2 самых длинных, не пересекающихся, упорядоченных по неубыванию отрезка.
2. Сложно в самой процедуре слияния определить, где в итоге отсортированный массив: в X или Y, так как меняя их местами, следует помнить, что процедура рекурсивна.
3. Необходимо понять, как действовать, когда отрезки на концах совпадут. Это может произойти сразу по получении процедурой массива или в процессе сортировки. Следует понимать, что эти отрезки могут или совпадать или не пересекаться.
4. Довольно сложно организовать отладочный вывод так, чтобы было понятно, что происходит.

Также существуют ограничения на количество вводимых с консоли символов. Это я поняла, когда пыталась создать файл для тестирования, вводя 30 слов из 10 букв.

Ещё из-за того, что файлов в проекте целых 4, приходится при тестировании множество раз компилировать 3 из них и ничего не забывать.

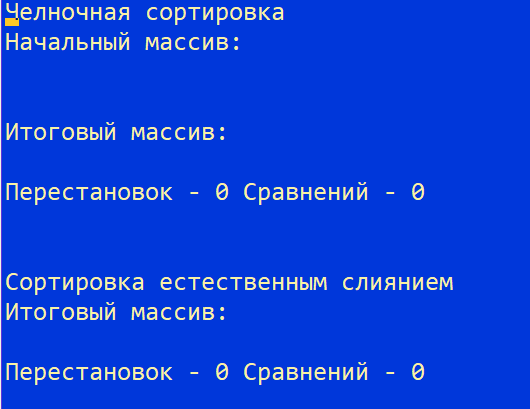
***Оценка работы рассмотренных методов сортировки***

На небольших массивах метод челночной сортировки выигрывает по количеству перестановок и сравнений. Однако, на большом количестве данных (30) челночная сортировка делает значительно больше перестановок, и не на столько меньше сравнений, чем естественное слияние. Также на массивах, близких к обратно упорядоченным, естественное слияние делает меньше перестановок.

***Результаты экспериментов***

*Файлы для тестирования*

1. T1 – пустой файл 2) T2 – файл с одним словом

 Изображение выглядит как текст

Автоматически созданное описание

1. T3 – файл с максимальным количеством элементов (случайный порядок)

Челночная сортировка проигрывает по количеству перестановок (довольно сильно, на 40% больше перестановок), но выигрывает по сравнениям: на 24% сравнений меньше.

1. T4 – файл с произвольным количеством элементов

4 элемента – челночная сортировка значительно лучше по сравнениям и перестановкам

Изображение выглядит как текст

Автоматически созданное описание

1. T5 – файл, с элементами, упорядоченными в обратном порядке

Естественное слияние делает почти в 2 раза меньше перестановок, но на 32% больше сравнений.

1. T6 – файл, с упорядоченными элементами

Сортировки одинаковы по перестановкам и сравнениям.