Эта программа состоит из нескольких функций, которые работают вместе, чтобы считать слова из файла, найти самое частое длинное слово и самое редкое короткое слово и заменить их в файле. Давайте рассмотрим каждую функцию более подробно.

1. read\_words\_from\_file: Эта функция считывает слова из файла и сохраняет их в массиве структур Word. Каждая структура Word содержит само слово, количество его вхождений в файл и флаг, указывающий на то, было ли оно использовано. Функция открывает файл с помощью fopen и выделяет память для массива words с помощью malloc. Затем функция считывает слова из файла с помощью fscanf и проверяет, есть ли уже это слово в массиве words. Если слово уже есть в массиве, функция увеличивает его счетчик на 1. Если слово еще не встречалось, функция выделяет дополнительную память для массива words с помощью realloc, копирует слово в новую структуру Word с помощью strcpy, устанавливает счетчик равным 1 и флаг was\_used равным 0. После того, как все слова считаны из файла, функция закрывает файл с помощью fclose и возвращает указатель на массив структур Word.
2. find\_and\_save\_words: Эта функция принимает массив структур Word, количество слов в массиве и два указателя на указатели Word. Функция находит самое частое длинное слово и самое редкое короткое слово в массиве, которые еще не были использованы. Для этого функция проходит по всему массиву words и проверяет флаг was\_used каждого элемента. Если флаг равен 0 (то есть слово еще не было использовано), функция проверяет длину слова и его количество. Если это самое длинное и самое частое слово из тех, которые еще не были использованы, функция сохраняет его адрес в переменной most\_frequent\_long\_word. Аналогично, если это самое короткое и самое редкое слово из тех, которые еще не были использованы, функция сохраняет его адрес в переменной least\_frequent\_short\_word. После того, как все слова проверены, функция устанавливает флаги was\_used равными 1 для найденных слов и сохраняет их адреса в местах, на которые указывают указатели most\_frequent\_long\_word\_ptr и least\_frequent\_short\_word\_ptr.
3. read\_line: Эта функция считывает строку из файла и возвращает ее как строку C. Функция выделяет память для буфера с помощью malloc и использует функцию fgets для чтения строки из файла. Если строка не помещается в буфер, функция увеличивает размер буфера с помощью realloc и продолжает чтение строки. Когда строка полностью считана (то есть достигнут символ новой строки или конец файла), функция возвращает строку. Если достигнут конец файла или произошла ошибка чтения, функция освобождает память буфера с помощью free и возвращает NULL.
4. replace\_words\_in\_file: Эта функция принимает имя файла и два указателя на структуры Word. Функция считывает содержимое указанного файла и заменяет все вхождения первого слова на второе слово и все вхождения второго слова на первое слово. Для этого функция открывает файл с помощью fopen и создает временный файл с помощью tmpfile. Затем функция считывает строки из файла с помощью функции read\_line и разбивает их на слова с помощью функции strtok. Для каждого слова функция проверяет, равно ли оно первому или второму слову. Если слово равно первому слову, функция записывает второе слово во временный файл с помощью fprintf. Если слово равно второму слову, функция записывает первое слово во временный файл. В противном случае функция записывает само слово во временный файл. После того, как все строки обработаны, функция закрывает оба файла с помощью fclose, открывает исходный файл для записи с помощью fopen, перематывает временный файл в начало с помощью rewind и копирует содержимое временного файла в исходный файл с помощью fgets и fputs. Наконец, функция закрывает оба файла с помощью fclose.
5. find\_old\_size: Эта функция принимает два указателя на структуры Word и вычисляет старый размер этих двух слов. Старый размер равен произведению длины первого слова на его количество плюс произведение длины второго слова на его количество.
6. find\_new\_size: Эта функция принимает два указателя на структуры Word и вычисляет новый размер этих двух слов. Новый размер равен произведению длины первого слова на количество второго слова плюс произведение длины второго слова на количество первого слова плюс сумма длин обоих слов плюс 2.

#define f\_name "C:/Users/misha/Desktop/OAiP/Lab\_ty\_ty\_ty/Tol.txt" // Определение имени файла

#define \_CRT\_SECURE\_NO\_WARNINGS // Отключение предупреждений безопасности

#define MAX\_WORD\_LENGTH 100 // Определение максимальной длины слова

#include "func.h" // Подключение заголовочного файла с объявлениями функций

int main() {

int num\_words = 0; // Объявление и инициализация переменной для хранения количества слов

Word\* words = read\_words\_from\_file(f\_name, &num\_words); // Считывание слов из файла и сохранение их в массиве структур Word

Word\* most\_frequent\_long\_word; // Объявление указателя на самое частое длинное слово

Word\* least\_frequent\_short\_word; // Объявление указателя на самое редкое короткое слово

int max\_pairs = num\_words / 2; // Вычисление максимального количества пар слов

char\*\* pairs = malloc(max\_pairs \* sizeof(char\*)); // Выделение памяти для массива пар слов

for (int i = 0; i < max\_pairs; i++) {

pairs[i] = malloc(MAX\_WORD\_LENGTH \* 2 + 1); // Выделение памяти для каждой пары слов

pairs[i][0] = '\0'; // Инициализация пары слов пустой строкой

}

int pair\_count = 0; // Объявление и инициализация счетчика пар слов

while (1) {

find\_and\_save\_words(words, num\_words, &most\_frequent\_long\_word, &least\_frequent\_short\_word); // Поиск самого частого длинного слова и самого редкого короткого слова

int old\_size = find\_old\_size(most\_frequent\_long\_word, least\_frequent\_short\_word); // Вычисление старого размера этих двух слов

//

int new\_size = find\_new\_size(most\_frequent\_long\_word, least\_frequent\_short\_word); // Вычисление нового размера этих двух слов

//

if (new\_size >= old\_size) { // Если новый размер больше или равен старому размеру,

break; // выходим из цикла

}

replace\_words\_in\_file(f\_name, most\_frequent\_long\_word, least\_frequent\_short\_word); // Замена слов в файле

sprintf(pairs[pair\_count], "%s %s", most\_frequent\_long\_word->word, least\_frequent\_short\_word->word); // Сохранение пары слов в массиве pairs

pair\_count++; // Увеличение счетчика пар слов на 1

}

FILE\* file = fopen(f\_name, "a"); // Открытие файла для добавления данных в конец файла

fprintf(file, " --- "); // Запись разделителя в файл

for (int i = 0; i < pair\_count; i++) {

fprintf(file, "%s\n", pairs[i]); // Запись всех пар слов в файл

}

fclose(file); // Закрытие файла

for (int i = 0; i < max\_pairs; i++) {

free(pairs[i]); // Освобождение памяти для каждой пары слов

}

free(pairs); // Освобождение памяти для массива пар слов

free(words); // Освобождение памяти для массива структур Word

return 0; // Возврат значения 0 из функции main

}

#define f\_name "C:/Users/misha/Desktop/OAiP/Lab\_ty\_ty\_ty/Tol.txt" // Определение имени файла

#define \_CRT\_SECURE\_NO\_WARNINGS // Отключение предупреждений безопасности

#define MAX\_WORD\_LENGTH 100 // Определение максимальной длины слова

#include "func.h" // Подключение заголовочного файла с объявлениями функций

int main() {

int num\_words = 0; // Объявление и инициализация переменной для хранения количества слов

Word\* words = read\_words\_from\_file(f\_name, &num\_words); // Считывание слов из файла и сохранение их в массиве структур Word

Word\* most\_frequent\_long\_word; // Объявление указателя на самое частое длинное слово

Word\* least\_frequent\_short\_word; // Объявление указателя на самое редкое короткое слово

int max\_pairs = num\_words / 2; // Вычисление максимального количества пар слов

char\*\* pairs = malloc(max\_pairs \* sizeof(char\*)); // Выделение памяти для массива пар слов

for (int i = 0; i < max\_pairs; i++) {

pairs[i] = malloc(MAX\_WORD\_LENGTH \* 2 + 1); // Выделение памяти для каждой пары слов

pairs[i][0] = '\0'; // Инициализация пары слов пустой строкой

}

int pair\_count = 0; // Объявление и инициализация счетчика пар слов

while (1) {

find\_and\_save\_words(words, num\_words, &most\_frequent\_long\_word, &least\_frequent\_short\_word); // Поиск самого частого длинного слова и самого редкого короткого слова

int old\_size = find\_old\_size(most\_frequent\_long\_word, least\_frequent\_short\_word); // Вычисление старого размера этих двух слов

//

int new\_size = find\_new\_size(most\_frequent\_long\_word, least\_frequent\_short\_word); // Вычисление нового размера этих двух слов

//

if (new\_size >= old\_size) { // Если новый размер больше или равен старому размеру,

break; // выходим из цикла

}

replace\_words\_in\_file(f\_name, most\_frequent\_long\_word, least\_frequent\_short\_word); // Замена слов в файле

sprintf(pairs[pair\_count], "%s %s", most\_frequent\_long\_word->word, least\_frequent\_short\_word->word); // Сохранение пары слов в массиве pairs

pair\_count++; // Увеличение счетчика пар слов на 1

}

FILE\* file = fopen(f\_name, "a"); // Открытие файла для добавления данных в конец файла

fprintf(file, " --- "); // Запись разделителя в файл

for (int i = 0; i < pair\_count; i++) {

fprintf(file, "%s\n", pairs[i]); // Запись всех пар слов в файл

}

fclose(file); // Закрытие файла

for (int i = 0; i < max\_pairs; i++) {

free(pairs[i]); // Освобождение памяти для каждой пары слов

}

free(pairs); // Освобождение памяти для массива пар слов

free(words); // Освобождение памяти для массива структур Word

return 0; // Возврат значения 0 из функции main

}

Word\* read\_words\_from\_file(char\* filename, int\* num\_words) {

FILE\* file = fopen(filename, "r"); // Открытие файла для чтения

if (file == NULL) { // Если файл не удалось открыть,

printf("Error\n"); // выводим сообщение об ошибке

exit(1); // и завершаем программу с кодом ошибки 1

}

Word\* words = malloc(sizeof(Word)); // Выделение памяти для массива структур Word

char buffer[MAX\_WORD\_LENGTH]; // Объявление буфера для хранения слова

while (fscanf(file, "%s", buffer) != EOF) { // Пока не достигнут конец файла,

int found = 0; // Объявление и инициализация флага найденного слова

for (int i = 0; i < \*num\_words; i++) { // Для каждого слова в массиве words

if (strcmp(buffer, words[i].word) == 0) { // Если слово в буфере равно текущему слову в массиве,

words[i].count++; // увеличиваем счетчик этого слова на 1

found = 1; // устанавливаем флаг найденного слова равным 1

break; // и выходим из цикла

}

}

if (!found) { // Если слово не найдено в массиве,

words = realloc(words, (\*num\_words + 1) \* sizeof(Word)); // выделяем дополнительную память для массива words

strcpy(words[\*num\_words].word, buffer); // копируем слово из буфера в новую структуру Word

words[\*num\_words].count = 1; // устанавливаем счетчик этого слова равным 1

words[\*num\_words].was\_used = 0; // устанавливаем флаг was\_used этого слова равным 0

(\*num\_words)++; // увеличиваем количество слов на 1

}

}

fclose(file); // Закрытие файла

return words; // Возврат указателя на массив структур Word

}

void find\_and\_save\_words(Word words[], int num\_words, Word\*\* most\_frequent\_long\_word\_ptr, Word\*\* least\_frequent\_short\_word\_ptr) {

Word\* most\_frequent\_long\_word = NULL; // Объявление и инициализация указателя на самое частое длинное слово

Word\* least\_frequent\_short\_word = NULL; // Объявление и инициализация указателя на самое редкое короткое слово

for (int i = 0; i < num\_words; i++) { // Для каждого слова в массиве words

if (!words[i].was\_used) { // Если флаг was\_used этого слова равен 0 (то есть слово еще не было использовано),

if (most\_frequent\_long\_word == NULL || (strlen(words[i].word) > strlen(most\_frequent\_long\_word->word) && words[i].count > most\_frequent\_long\_word->count)) {

most\_frequent\_long\_word = &words[i];

}

if (least\_frequent\_short\_word == NULL || (strlen(words[i].word) < strlen(least\_frequent\_short\_word->word) && words[i].count < least\_frequent\_short\_word->count)) {

least\_frequent\_short\_word = &words[i];

}

}

}

if (!most\_frequent\_long\_word->was\_used && !least\_frequent\_short\_word->was\_used) {

most\_frequent\_long\_word->was\_used = 1;

least\_frequent\_short\_word->was\_used = 1;

\*most\_frequent\_long\_word\_ptr = most\_frequent\_long\_word;

\*least\_frequent\_short\_word\_ptr = least\_frequent\_short\_word;

}

}

char\* read\_line(FILE\* file) {

size\_t buffer\_size = 1024; // Объявление и инициализация размера буфера

char\* buffer = malloc(buffer\_size); // Выделение памяти для буфера

size\_t len = 0; // Объявление и инициализация длины строки

while (fgets(buffer + len, buffer\_size - len, file) != NULL) { // Пока не достигнут конец файла или не произошла ошибка чтения,

len += strlen(buffer + len); // увеличиваем длину строки на длину считанной части строки

if (buffer[len - 1] == '\n') { // Если последний символ в буфере равен символу новой строки,

break; // выходим из цикла

}

buffer\_size \*= 2; // Увеличиваем размер буфера в 2 раза

buffer = realloc(buffer, buffer\_size); // Выделяем дополнительную память для буфера

}

if (len == 0) { // Если длина строки равна 0 (то есть достигнут конец файла или произошла ошибка чтения),

free(buffer); // освобождаем память буфера

return NULL; // и возвращаем NULL

}

return buffer; // Возвращаем указатель на строку

}

void replace\_words\_in\_file(char\* filename, Word\* word\_a, Word\* word\_b) {

FILE\* file = fopen(filename, "r"); // Открытие файла для чтения

FILE\* new\_file = tmpfile(); // Создание временного файла

char buffer[MAX\_WORD\_LENGTH]; // Объявление буфера для хранения слова

char\* line = NULL; // Объявление и инициализация указателя на строку

char\* next\_line = read\_line(file); // Считывание первой строки из файла

while (next\_line != NULL) { // Пока не достигнут конец файла,

line = next\_line; // сохраняем указатель на текущую строку

char\* word = strtok(line, " \n"); // Разбиваем строку на слова с помощью функции strtok

while (word != NULL) { // Для каждого слова в строке

if (strcmp(word, word\_a->word) == 0) { // Если слово равно первому слову,

fprintf(new\_file, "%s", word\_b->word); // записываем второе слово во временный файл

}

else if (strcmp(word, word\_b->word) == 0) { // Если слово равно второму слову,

fprintf(new\_file, "%s", word\_a->word);

}

else {

fprintf(new\_file, "%s", word);

}

word = strtok(NULL, " \n");

if (word != NULL) {

fprintf(new\_file, " ");

}

}

next\_line = read\_line(file);

if (next\_line != NULL) {

fprintf(new\_file, "\n");

}

free(line);

}

fclose(file);

file = fopen(filename, "w");

rewind(new\_file);

while (fgets(buffer, MAX\_WORD\_LENGTH, new\_file) != NULL) {

fputs(buffer, file);

}

fclose(file);

fclose(new\_file);

}

int find\_old\_size(Word\* word1, Word\* word2) {

return strlen(word1->word) \* word1->count + strlen(word2->word) \* word2->count;

}

int find\_new\_size(Word\* word1, Word\* word2) {

return strlen(word1->word) \* word2->count + strlen(word2->word) \* word1->count + strlen(word1->word) + strlen(word2->word)+2;

}