Отчет к лабороторной работе №3

Задача:

Вводится последовательность предложений, оканчивающихся точкой. Предложения могут занимать более одной строки. Выведите все предложения в порядке неубывания длин ровно по одному на строку, заменяя перенос строки в исходном вводе пробельным символом. Код программы должен содержать ввод с консоли флага --tofile и --fromfile. Первый говорит нам, что ввод будет задан в заданный пользователем в командной строке файл (в случае отсутствия файла создать его), а второй говорит о считывании из файла заданного пользователем. Предусмотреть возможность исполнения обоих флагов. Учесть возможность некорректного ввода и обработать исключючения. На дополнительный балл (на 10 баллов) добавьте тесты, по возможности Google-Tests.

Алгоритм:

```
#include "header.hpp"

int main(int argc, char **argv) {
    const char *buf_name = "bufer.txt";
    size_t count = 0;
    size_t sum = 0;
    size_t *len = new size_t[200];
    if (argc == 1) {
        write_file(buf_name, count, len);
        std::ifstream file(buf_name);
        for (int i = 0; i < count; ++i) {
            sum += len[i];
        }
        char *text = new char[sum];
        file.getline(text, sum + 1);
        nine(text, count, len);
        size_t cur = 0;
        size_t start[count];
        size_t end[count];
        start[0] = 0;
        for (size_t i = 1; i < count; ++i) {
            cur += len[i - 1];
            start[i] = cur;
            end[i - 1] = start[i - 1] + len[i - 1];
        }
        end[count - 1] = start[count - 1] + len[count - 1];
        cout << "-----------------------------------------------------"
                "-----------------------------------------------------\n";
        cout << "Answer:\n";
        res_print(text, count, len, start, end);
        remove(buf_name);
        delete[] text;
    } else if (argc == 3) {
        if (strcmp(argv[1], "--fromfile") == 0) {
            const char *from_name = argv[2];
            std::ifstream file(from_name);
            if (!file.is_open()) {
                std::cerr << "ERROR";
                return EXIT_FAILURE;
            }
            size_t cur = 1;
            char f = '\0';
            while (file.get(f)) {//считываем строку
                if (f != '\n' && f != 13) {
                    cur++;
                    if (f == '.') {
```

```
                    len[count] = cur;
                    sum += cur;
                    count++;
                    cur = 0;
                }
            }
        }
        file.close();
        std::ifstream file_t(from_name);
        char *text = new char[sum];
        size_t index = 1;
        while (file_t.get(f)) {//считываем строку
            if (f != '\n' && f != 13) {
                text[index] = f;
                index++;
            }
        }
        nine(text, count, len);
        text[0] = ' ';
        cur = 0;
        size_t start[count];
        size_t end[count];
        start[0] = 0;
        for (size_t i = 1; i < count; ++i) {
            cur += len[i - 1];
            start[i] = cur;
            end[i - 1] = start[i - 1] + len[i - 1];
        }
        end[count - 1] = start[count - 1] + len[count - 1];
        cout << "-----------------------------------------------------"
                "-----------------------------------------------------\n";
        cout << "Answer:\n";
        res_print(text, count, len, start, end);
        delete[] text;
    } else if (strcmp(argv[1], "--tofile") == 0) {
        const char *to_name = argv[2];
        write_file(buf_name, count, len);
        std::ifstream file(buf_name);
        for (int i = 0; i < count; ++i) {
            sum += len[i];
        }
        char *text = new char[sum];
        file.getline(text, sum + 1);
        nine(text, count, len);
        size_t cur = 0;
        size_t start[count];
        size_t end[count];
        start[0] = 0;
        for (size_t i = 1; i < count; ++i) {
            cur += len[i - 1];
            start[i] = cur;
            end[i - 1] = start[i - 1] + len[i - 1];
        }
        end[count - 1] = start[count - 1] + len[count - 1];
        cout << "-----------------------------------------------------"
                "-----------------------------------------------------\n";
        cout << "Answer in " << to_name << std::endl;
        file_res_print(text, count, len, start, end, to_name);
        remove(buf_name);
        delete[] text;
    } else {
        std::cerr << "Enter the correct flags\n";
        return EXIT_FAILURE;
    }
```

```
    } else if (argc == 5) {
        const char *from_name = argv[2];
        char *to_name = argv[4];
        if (strcmp(argv[1], "--fromfile") == 0 && strcmp(argv[3], "--tofile")
== 0) {
            from_name = argv[2];
            to_name = argv[4];
        } else if (strcmp(argv[3], "--fromfile") == 0 && strcmp(argv[1], "--
tofile") == 0) {
            from_name = argv[4];
            to_name = argv[2];
        } else {
            std::cerr << "Enter the correct flags\n";
            return EXIT_FAILURE;
        }
        std::ifstream file(from_name);
        if (!file.is_open()) {
            std::cerr << "ERROR";
            return EXIT_FAILURE;
        }
        size_t cur = 1;
        char f = '\0';
        while (file.get(f)) {//считываем строку
            if (f != '\n' && f != 13) {
                cur++;
                if (f == '.') {
                    len[count] = cur;
                    sum += cur;
                    count++;
                    cur = 0;
                }
            }
        }
        file.close();
        std::ifstream file_t(from_name);
        char *text = new char[sum];
        size_t index = 1;
        while (file_t.get(f)) {//считываем строку
            if (f != '\n' && f != 13) {
                text[index] = f;
                index++;
            }
        }
        text[0] = ' ';
        nine(text, count, len);
        cur = 0;
        size_t start[count];
        size_t end[count];
        start[0] = 0;
        for (size_t i = 1; i < count; ++i) {
            cur += len[i - 1];
            start[i] = cur;
            end[i - 1] = start[i - 1] + len[i - 1];
        }
        end[count - 1] = start[count - 1] + len[count - 1];
        cout << "-------------------------------------------------"
                "-------------------------------------------------\n";
        cout << "Answer:\n";
        file_res_print(text, count, len, start, end, to_name);
        delete[] text;
    }
    delete[] len;
```

```
    return 0;
}
```

Описание алгоритма:

На вход подается несколько флагов и от этого зависит дальнейшие действия. Если без флагов, то все что ввел пользователь записывается во временный файл в одну строку. Потом все считывается в массив char *text , который передается в функцию res_print, где происходит сортировка предложений и вывод ответа. Если флаг –tofile, то происходит тоже самое, что и без флагов, только результат не выводиться в консоли, а записывается в файл. Если флаг –fromfile, то происходит считывание с файла в char *text и также с помощью функции res_print происходит вывод в консоли. Когда присутствуют оба флага, то считывание происходит как при флаге –fromfile, а вывод как при флаге –tofile.

Функции:

1) Функция void write_file()

Содержание:

```
void write_file(const char *name, size_t &count, size_t *&len) {
    std::ofstream file(name, std::ios_base::out | std::ios_base::trunc);
    file.clear();
    std::noskipws(cin);
    size_t cur_len = 0;
    char symb = '\0';
    file << " ";
    while (cin.peek() != EOF) {
        for (;;) {
            if (cin.peek() == '\n') {
                break;
            }
            cin >> symb;
            ++cur_len;
            file << symb;
            if (symb == '.') {
                len[count] = cur_len;
                ++count;
                cur_len = 0;
            }
        }
        cin >> symb;
    }
    len[0]++;
    file.close();
}
```

Описание:

Функция принимает аргументы названия временного файла name, кол-во предложений count, массив длин предложений len.

Перед началом работы, она очищает временный файл и сообщает програме, чтобы оператор cin не пропускал пробелы. Далее функция проверяет есть ли дальше элементы, если да, то запускается бесконечный цикл пока не встретиться символ перехода на другую строку, записывая их во временный файл, и по пути программа считает кол-во предложений и их длину.

2) Функция void res_print()

Содержание:

```
void res_print(char *text, size_t count, size_t len[], size_t start[], size_t
end[]) {
    for (size_t i = 0; i < count - 1; ++i) {
        for (size_t j = count - 1; j > i; --j) {
            if (len[j - 1] > len[j]) {
                std::swap(len[j - 1], len[j]);
```

```
                std::swap(start[j - 1], start[j]);
                std::swap(end[j - 1], end[j]);
            }
        }
    }

    for (size_t i = 0; i < count; ++i) {
        for (size_t j = start[i]; j < end[i]; ++j) {
            cout << text[j];
        }
        cout << "\n";
    }
}
```

Описание:

На вход подаётся 5 аргументов: массив с предложениями text, кол-во предложений count, их длина len, массив с индексами старта start и конца end предложений.

Функция, с помощью пузырьковой сортировки, сортирует длины по не убыванию, сортируя по длине мы также меняем индексы в start и end. Затем запускается цикл for, который повторяется count-1 раз, в нем выводится элементы массива text с start[j] до end[j].

3)Функция void file_res_print()

Содержание:

```
void file_res_print(char *text, size_t count, size_t len[], size_t start[],
size_t end[], const char *file_name) {
    for (size_t i = 0; i < count - 1; ++i) {
        for (size_t j = count - 1; j > i; --j) {
            if (len[j - 1] > len[j]) {
                std::swap(len[j - 1], len[j]);
                std::swap(start[j - 1], start[j]);
                std::swap(end[j - 1], end[j]);
            }
        }
    }
    std::ofstream file(file_name);
    for (size_t i = 0; i < count; ++i) {
        for (size_t j = start[i]; j < end[i]; ++j) {
            file << text[j];
        }
        file << "\n";
    }
}
```

Описание:

Аналогичная с функцией void res_print(), только выводит все не в консоль, а записывает в файл, название которого передано в аргументах.

4)Функция void nine()

Содержание:

```
void nine(const char *text, const size_t count, const size_t len[]) {
    std::ofstream file_b("NINE.txt", std::ios_base::out |
std::ios_base::trunc);
    size_t number_of_ms = 0, dought_index1 = 1, len_max = 0;
    for (size_t i = 0; i < count; ++i) {
        if (len[i] > len_max) {
            number_of_ms = i;
```

```
            len_max = len[i];
        }
    }
    len_max--;
    for (size_t i = 0; i < number_of_ms; ++i) {
        dought_index1 += len[i];
    }
    char NINE[len_max];
    size_t count_word = 0, cur_word = 0;
    int j = 0;
    for (int i = dought_index1; i < dought_index1 + len_max; ++i) {
        if (text[i] == ' ' || text[i] == '.') {
            count_word++;
        }
        NINE[j] = text[i];
        j++;
    }
    NINE[len_max - 1] = ' ';
    if (count_word > 1) {
        size_t len_word[count_word];
        size_t start[count_word];
        start[0] = 0;
        j = 0;
        for (int i = 0; i < len_max; i++) {
            cur_word++;
            if (NINE[i] == ' ' || NINE[i] == '.') {
                len_word[j] = cur_word;
                j++;
                if (j < count_word) {
                    start[j] = start[j - 1] + cur_word;
                }
                cur_word = 0;
            }
        }
        for (int i = count_word - 1; i >= 0; i--) {
            for (j = start[i]; j < start[i] + len_word[i]; j++) {
                file_b << NINE[j];
            }
        }
    } else {
        file_b << NINE;
    }
    file_b.close();
}
```

Описание:

В аргументах передается массив предложений text, кол-во предложений count, длины предложений len.

В первом for находится номер самого длинного предложения и его длина. В следующем for находится индекс первой буквы самого длинного предложения. Далее

```
for (int i = dought_index1; i < dought_index1 + len_max; ++i) {
      if (text[i] == ' ' || text[i] == '.') {
         count_word++;
      }
      NINE[j] = text[i];
      j++;
   }
   NINE[len_max - 1] = ' ';
```

записывает самое длинное предложение в массив NINE. Следующим шагом будет проверка из сколько слов состоит предложение, если из одного, то программа запишет его в файл NINE.txt, если несколько, то программа запишет индекс начала каждого слова и будет записывать слов в файл в обратном порядке.