Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Кафедра компьютерной безопасности

Отчёт

к лабораторной работе №3

по дисциплине

«Языки программирования»

Работу выполнила

студентка группы СКБ221

Е. В. Гриднева

Москва 2022

Постановка задачи:

Вводится последовательность предложений, оканчивающихся точкой. Предложения могут занимать более одной строки. Вывести все предложения в порядке возрастания длин ровно по одному на строку, заменяя перенос строки в исходном вводе пробельным символом. Код программы должен поддерживать ввод с консоли флагов --tofile и --fromfile. Первый говорит нам, что вывод будет задан в заданный пользователем в командной строке файл (в случае отсутствия файла создает его), а второй говорит о считывании текста из файла, заданного пользователем.

Необходимо предусмотреть возможность исполнения обоих флагов. Учесть возможность некорректного ввода и обработать исключения.

Создать Makefile с возможностью сборки, clean и distclean.

Алгоритм решения задачи

Для выполнения поставленной задачи разработан следующий алгоритм:

1) После написания программы, используя компилятор g++ внутри терминала с дополнением WSL, я скомпилировала свой файл командой: g++ lab.cpp -o lab_3, при этом назвав исполняемый файл “lab_3”. В дальнейших пунктах будет описана суть кода.

2) Далее необходимо было запустить исполняемый файл с нужными мне флагами командой “./lab_3 …”, где вместо “…” пользователь мог ввести флаги, названия файлов. В зависимости от флагов или их отсутствия, ввод и вывод могли быть как из файла, так и из терминала.

3) В случае, если пользователь ввел неверное название файла для флага – fromfile, выводится ошибка, которая предупреждает пользователя. Флаг –tofile адаптирован под пользователя в нескольких случаях ввода названия файла сразу. Если название было введено и файл существует, то название файла просто запоминается до функции вывода. Если название файла было введено, но файла не существует, то моя программа создает файл с таким же именем, какое задал пользователь. Если название не задано, то программа создаст файл с дефолтным именем “tofile.txt”.

4) После полного считывания текста в массив я находила координаты каждой точки (каждого конца предложения), чтобы в дальнейшем посчитать размер предложения “от точки до точки” и поставить эта размеры в соответствие с каждой координатой точки. Таким образом, я узнавала сразу и размер предложения (без пробелов и переносов строки), и место нахождения этого предложения в массиве. Вывод в файл и в терминал реализованы похожим образом.

5) Для упрощения работы с терминалом мной был создан Makefile с целями all, clean, tests, lab, distclean. Цель «tests» отвечает за автоматические тесты программы на текстовых файлах, «lab» отвечает за автоматическую компиляцию программы, «clean» - за отчистку временных файлов, «distclean» - за очистку временных файлов, а также объектных файлов.

Выполнение задачи

1) В функции main (int argc, char** argv) я считала флаги, записала имена файлов ввода/вывода и, в случае ошибки ввода, выводила сообщение об этом. Существование файлов проверяла функцией fopen. Функция strcpy помогала сохранять имя файла для дальнейшей работы с файлами. Далее вызывалась функция read_from_terminal (char* text) или read_from_file (FILE* fp, char *filename, char* text).

2) В функции read_from_terminal (char* text) я считывала текст из терминала в массив функцией getline. Конец ввода обозначала знаком “~”. В функции read_from_terminal (FILE* fp, char *filename, char* text) я считывала текст из файла с именем ‘‘filename’’ в массив ‘’text’’ с помощью fgetc.

3) После прочтения всего текста в массив я переходила к функции sort_and_out(const char *text, char *tofile), которая выполняла основную работу по выводу результата. Первым делом функцией get_point_coord(const char *text, int *points) я нашла все координаты точек, при этом возвращала ей количество этих самых точек. Так как я знала количество и координаты точек, то для упрощения координаты сразу были записаны в массив. Далее оставалась циклом while начать подсчет количества символов в предложении перемещением указателя (переменной “j”) в начало предложения именно до первой буквы(!). Уже оттуда я снова возвращалась до конца этого предложения и записывала результат длины в массив длин. При этом элемент массива длин соответствовал элементу массива координат точек.

4) Далее требовалось найти предложение минимальной длины, поэтому внутри функции find_min(int *lenght, int counter) я делали именно это. И так как номера элементов массивов длины и конца предложения идентичны, то оставалось лишь каждый раз выводить предложение (это я делала посимвольно) с минимальной длиной, а предыдущую минимальную длину я делала максимальной, чтобы она больше не помешала. Функции print_to_file и print_to_terminal имеют похожих функционал с функцией sort_and_out в моментах перемещения указателя в начало предложения. Далее, соответственно, посимвольно предложения выводились в терминал (функцией cout), либо в открытый для записи файл (функцией fprintf). Для записи файл открывался функцией fopen(tofile, "w").

Тестирование

1) Ввод из файла. Вывод в терминал (Тесты 3, 4) и вывод в файл(Тесты 1, 5).

```
-------Test #1: File output-------
./lab_3 --fromfile read1.txt --tofile output.txt

cat output.txt
Test.
It was released on 26 November 1865.
It has been adapted to numerous movies.
Charles Lutwidge Dodgson wrote the book.
He wrote it using the pen name Lewis Carroll.
John Tenniel drew the 42 pictures in the book.
In 2010, it has been adapted to Hollywood movie.
The book was published by Macmillan and Co in London.
The title is usually shortened to Alice in Wonderland.
Alice's Adventures in Wonderland is a children's novel.
```

```
-------Test #3: Output to Terminal-------
./lab_3 --fromfile read1.txt
Test.
It was released on 26 November 1865.
It has been adapted to numerous movies.
Charles Lutwidge Dodgson wrote the book.
He wrote it using the pen name Lewis Carroll.
John Tenniel drew the 42 pictures in the book.
In 2010, it has been adapted to Hollywood movie.
The book was published by Macmillan and Co in London.
The title is usually shortened to Alice in Wonderland.
Alice's Adventures in Wonderland is a children's novel.
```

```
-------Test #4: Output to Terminal (Sentences with the same length)-------
./lab_3 --fromfile read2.txt
Yes.
Hey.
TEst.
test.
Test.
Test.
Tester.
Toster.
Home alone.


-------Test #5: Unnamed File output-------
./lab_3 --fromfile read1.txt --tofile

cat tofile.txt
Test.
It was released on 26 November 1865.
It has been adapted to numerous movies.
Charles Lutwidge Dodgson wrote the book.
He wrote it using the pen name Lewis Carroll.
John Tenniel drew the 42 pictures in the book.
In 2010, it has been adapted to Hollywood movie.
The book was published by Macmillan and Co in London.
The title is usually shortened to Alice in Wonderland.
Alice's Adventures in Wonderland is a children's novel.
```

2) Ввод с терминала и вывод в терминал (без флагов, соответственно)

```
Enter text.
To mark the end of input, type '~' and press Enter.

Test. Hey.
My name
is Katya. Let us be
friends.

~
Hey.
Test.
My name is Katya.
Let us be friends.
```

3) Ввод команды с ошибкой

```
-------Test #2: Output file doesn't exist-------
./lab_3 --fromfile --tofile
Incorrect file for reading OR flag
```

Код программы

Приложение А

```
#include <iostream>
#include <cstring>
#include <fstream>

void read_from_terminal(char* text);
void read_from_file(FILE* fp, char *filename, char* text);
void sort_and_out(const char *text, char *tofile);
int get_point_coord(const char *text, int *points);
int find_min(int *lenght, int counter);
void print_to_terminal(int points, const char *text);
void print_to_file(int points, const char *text, FILE* fp);

int main(int argc, char** argv) {
   FILE* fp;
   char tofile_name[32] = {0};
   int fromfile = 0;
   char text[1024];
   for (int i = 1; i < argc; i++) {
      if (argv[i][0] == '-') {
                        if ((std::strcmp(argv[i], "--tofile")) == 0) {
            if (i + 1 >= argc || (fp = fopen(argv[i+1], "r")) == NULL) {
               if (i + 1 >= argc || argv[i+1][0] == '-') {
                  std::ofstream oFile ( "tofile.txt" );
                  std::strcpy(tofile_name, "tofile.txt");
               }
               else {
                  std::ofstream oFile (argv[i+1]);
                  std::strcpy(tofile_name, argv[i+1]);
               }
            } else {
               std::strcpy(tofile_name, argv[i+1]);
               fclose(fp);
            }
                     }
                     else if ((std::strcmp(argv[i], "--fromfile")) == 0) {
                              if (i + 1 >= argc || (fp = fopen(argv[i+1], "r")) == NULL) {
               fromfile = -1;
            } else {
               fromfile = i + 1;
               fclose(fp);
            }
                     }
        }
```

```
    }
    if (fromfile >= 0) {
        if (fromfile == 0) {
            read_from_terminal(text);
        } else {
            read_from_file(fp, argv[fromfile], text);
        }
        sort_and_out(text, tofile_name);
    } else {
        std::cout << "Incorrect file for reading OR flag\n";
    }
    return 0;
}

void read_from_terminal(char* text) {
    std::cout << "Enter text. \nTo mark the end of input, type '~' and press Enter.\n" << std::endl;
    std::cin.getline(text, 1024, '~');
}

void read_from_file(FILE* fp, char *filename, char* text) {
    int j = 0;
    fp = fopen(filename, "r");
    while ((text[j] = fgetc(fp)) != EOF) {
                j++;
        }
        text[j] = '\0';
    fclose(fp);
}

void sort_and_out(const char *text, char *tofile) {
    int counter;
    int points[128];
    int j;
    counter = get_point_coord(text, points);
    int flag;
    int lenght[128] = {0};
    FILE* fp;
    for (int k = 0; k < counter; k++) {
        flag = 0;
        j = points[k] - 1;
        while (text[j - 1] != '.' && j != -1 && flag == 0) {
            j--;
            if (text[j - 1] == '.' && text[j] == ' ') {
                flag = 1;
                j++;
            }
        }
```

```
        for (int h = j; h < points[k] + 1; h++) {
            if (text[h] != '\n' && text[h] != ' ') {
                lenght[k]++;
            }
        }
    }
    flag = 0;
    flag = std::strlen(tofile);
    if (flag) {
        fp = fopen(tofile, "w");
    }
    for (int l = 0; l < counter; l++) {
        j = find_min(lenght, counter);
        if (!flag) {
            print_to_terminal(points[j], text);
            std::cout << '\n';
        } else {
            print_to_file(points[j], text, fp);
            if (l < counter - 1) {
                fprintf(fp, "%c", '\n');
            }
        }
        lenght[j] = 2048;
    }
    if (flag) {
        fclose(fp);
    }
}

int get_point_coord(const char *text, int *points) {
    int j = 0;
    int k = 0;
    int len = std::strlen(text);
    for (j = 0; j < len; j++) {
        if (text[j] == '.') {
            points[k] = j;
            k++;
        }
    }
    return k;
}

int find_min(int *lenght, int counter) {
    int min = 0;
    for (int i = 1; i < counter; i++) {
        if (lenght[i] < lenght[min]) {
            min = i;
```

```
        }
    }
    return min;
}

void print_to_terminal(int points, const char *text) {
    int flag = 0;
    int j = points - 1;
    while (text[j - 1] != '.' && j != -1 && flag == 0) {
        j--;
        if (text[j - 1] == '.' && text[j] == ' ') {
            flag = 1;
            j++;
        }
    }
    for (int h = j; h < points + 1; h++) {
        if (text[h] == '\n' && text[h + 1] != ' ' && text[h - 1] != ' ' && text[h-1] != '.') {
            std::cout << ' ';
        }
        else if (text[h] != '\n'){
            std::cout << text[h];
        }
    }
}

void print_to_file(int points, const char *text, FILE* fp) {
    int flag = 0;
    int j = points - 1;
    while (text[j - 1] != '.' && j != -1 && flag == 0) {
        j--;
        if (text[j - 1] == '.' && text[j] == ' ') {
            flag = 1;
            j++;
        }
    }
    for (int h = j; h < points + 1; h++) {
        if (text[h] == '\n' && text[h + 1] != ' ' && text[h - 1] != ' ' && text[h-1] != '.') {
            fprintf(fp, "%c", ' ');
        }
        else if (text[h] != '\n' && text[h] != '\0'){
            fprintf(fp, "%c", text[h]);
        }
    }
}
```

Приложение Б (Makefile)

```
G++ = g++ -Wall -Werror -Wextra

all: distclean lab_3

lab_3:
	$(G++) lab.cpp -o lab_3

tests:
	@echo "-------Test #1: File output-------"
	./lab_3 --fromfile read1.txt --tofile output.txt
	@echo " "
	cat output.txt
	@echo "\n"

	@echo "-------Test #2: Output file doesn't exist-------"
	./lab_3 --fromfile --tofile
	@echo "\n"

	@echo "-------Test #3: Output to Terminal-------"
	./lab_3 --fromfile read1.txt
	@echo "\n"

	@echo "-------Test #4: Output to Terminal (Sentences with the same length)-------"
	./lab_3 --fromfile read2.txt
	@echo "\n"

	@echo "-------Test #5: Unnamed File output-------"
	./lab_3 --fromfile read1.txt --tofile
	@echo " "
	cat tofile.txt
	@echo "\n"
```

```
clean:
	rm -rf tofile.txt

distclean: clean
	rm -rf lab_3

rebuild: distclean all
```

Вывод

В ходе данной лабораторной работы я научилась больше работать с вводом/выводом из файла, освоила больше функций из библиотеки “cstring”, а также новой целью для Makefile “distclean”.