КИРИЛЛ МОИСЕЕНКО
28.11.12

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

В моей лабораторной работе представлены 2 файла: файл ***main.cpp*** и ***Makefile***.

Файл ***main.cpp*** – главный исполняемый файл, в котором использованы библиотеки <iostream>, <fstream>, <cstring>. <iostream> используется для ввода\вывода элементов, <fstream> нужен для работы с файлами, а <cstring> для сравнения флагов при вводе с консоли.

Внутри своей работы самым первом циклом я проверяю наличие флагов –tofile и –fromfile. Если есть флаг –fromfile, то после него следующим токеном обязательно идёт название этого файла (иначе выводится Input Error). Если такого флага нет, то данные записываются из консоли. Наличие флага –tofile означает, что данные мы будем записывать в файл. Если после флага –tofile идёт его название, то мы просто открываем этот файл и записываем туда ответ. Иначе мы создаём свой собственный файл (“solution.txt”), куда мы будем производить ввод. Если же флага –tofile нет, то вывод будет в консоли.

После расставления флагов идёт проверка их наличия. В зависимости от наличия флага –fromfile мы считываем либо из существующего файла, либо же считываем из консоли. В обоих случаях мы считываем все наши символы в строках в массив char-ов symb[], с которым мы и будем работать. Далее также создаются массивы points_check[] и senten_len[], которые хранят индексы точек (концы наших преложений) и длины предложений соответственно. points_check[] создаётся обычным перебором массива symb[], а senten_len[] считает длины соответствующих предложений как количество символом от точки до точки (не считая первого пробела после точки и символа новой строки). Массив positions[] указывает нам, в каком порядке следует считывать предложения. Для этого следует просто перебрать массив senten_len[] (сортировать по длинам) и возвращать в очередном порядке номер предложения. После этого мы выводим сами предложения в зависимости от наличия флага –tofile. Если он есть, то мы создаём его (если его имя не указано, создаётся файл “solution.txt”) и записываем в этот файл наши предложения. Если же флаг –tofile не поднят, то выводим с помощью std::cin нужные нам предложения. В обоих случаях вывод предложений основывается на том, что поочередно берётся номер предложения из массива positions[] и с помощью цикла от точки до точки выводятся все наши символы из массива symb[] в заданном промежутке.

В дополнительном задании для того, чтобы вернуть последнюю строку перевернутой, мы создаём файл NINE.txt, в который мы будем записывать нужные нам слова в нужном нам порядке. Вне зависимости от флагов, мы в последнем переборе нашего цикла, записывающего уже готовые предложения, находим последний шаг цикла (в моём коде это момент равенства j=l). Когда этот шаг начинает работу, мы создаём отдельный массив arr9[], в который мы также дублируем последнее предложение (что на самом деле необязательно). После этого мы создаем массив space_check[], в котором будут хранится индексы наших пробелов этого предложения (по сути мы просто разделяем наше предложение по пробелам, как мы разделяли весь наш ввод по точкам). После этого мы создаём цикл, в котором проходимся от space_check[n-1] до space_check[n] индексов пробелов и также посимвольно вводим в файл NINE.txt . В итоге получаем то, что требовалось.

Файл ***Makefile*** содержит в себе файл ***main.cpp*** и непосредственно его сборку. Также там указаны функции clean и distclean, каждая из которых удаляет после сборки объектные файлы и сам проект, но distclean также удаляет и все файлы, созданные пользователем.

*Листинг 1: файл **main.cpp***

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

int main(int argc, char* argv[]) {
    bool mtofile = 0;
    bool mfromfile = 0;
    std::string tofile_name;
    std::string fromfile_name;
    std::ofstream nine("NINE.txt");
    char arr9[1000] = {'0'};
    int count9 = 0;

    if (argc != 1) {
        if (std::strcmp(argv[1], "--tofile") == 0) {
            mtofile = 1;
            if (argc != 2) {
                tofile_name = (argv[2]);
                if (argc != 3) {
                    if (std::strcmp(argv[3], "--fromfile") == 0) {
                        mfromfile = 1;
                        if (argc != 4) {
                            fromfile_name = (argv[4]);
                        } else {
                            std::cout << "Input Error" << std::endl;
                            return -1;
                        }
                    }
                }
            } else {
                tofile_name = "solution.txt";
            }
        } else if (std::strcmp(argv[1], "--fromfile") == 0) {
            mfromfile = 1;
            if (argc != 2) {
                fromfile_name = (argv[2]);
                if (argc != 3) {
                    if (std::strcmp(argv[3], "--tofile") == 0) {
                        mtofile = 1;
                        if (argc != 4) {
                            tofile_name = (argv[4]);
                        } else {
                            tofile_name = "solution.txt";
                        }
                    }
                }
            } else {
                std::cout << "Input Error" << std::endl;
                return -1;
            }
        }
    }


    if (mfromfile) {
        std::ifstream fromfile(fromfile_name);
        if (!fromfile) {
            std::cout << "Input Error" << std::endl;
            return -1;
        }
```

```
        char symb[10000] = {'0'};
        symb[0] = fromfile.get();
        symb[1] = fromfile.get();
        int i = 2;
        while (1) {
            if (fromfile.eof()) {
                break;
            }
            symb[i] = fromfile.get();
            i++;
        }
        int points_check[100] = {0};
        int counter_points = 1;
        points_check[0] = -1;
        for (int j = 0; j < i-1; j++) {if (symb[j] == '.')
{points_check[counter_points] = j; counter_points++;}}
        int counter_len = 0;
        int senten_len[100] = {0};
        int l = 0;
        for (int j = 0; j < i-1; j++) {
            if ((symb[j] != '.') && (symb[j] != '\n') && (symb[j] != ' ')) {
                counter_len++;
            } else if (symb[j] == ' ') {
                if (symb[j-1] != '.') {
                    counter_len++;
                }
            } else if (symb[j] == '.') {
                senten_len[l] = counter_len;
                l++;
                counter_len = 0;
            }
        }
        int positions[100] = {0};
        for (int j = 0; j < l; j++) {
            int index_min = 0;
            for (int k = 0; k < l; k++) {
                if (senten_len[k] < senten_len[index_min]) {index_min = k;}
            }
            positions[j] = index_min+1;
            senten_len[index_min] = 10000;
        }
        if (mtofile) {
            std::ofstream tofile(tofile_name);
            if (!tofile) {
                std::cout << "Input Error" << std::endl;
                return -1;
            }
            for(int j = 1; j < l+1; j++) {
                int k = positions[j-1];
                for (int p = points_check[k-1]+1; p < points_check[k]; p++) {
                    if ((symb[p-1] == '.') && (symb[p] == ' ')) {
                        tofile << "";
                    } else if ((symb[p-1] == '.') && (symb[p] == '\n')) {
                        tofile << "";
                    } else {tofile << symb[p];}
                }
                tofile << '.' << std::endl;
                if (j == l) {
                    for (int p = points_check[k-1]+1; p < points_check[k]; p++) {
                    if ((symb[p-1] == '.') && (symb[p] == ' ')) {
                        continue;
                    } else if ((symb[p-1] == '.') && (symb[p] == '\n')) {
```

```
                        continue;
                    } else {arr9[count9] = symb[p]; count9++;}}
                    arr9[count9] = ' ';
                    int space_check[100] = {0};
                    int counter_space = 1;
                    space_check[0] = -1;
                    for (int n = 0; n < (points_check[k] - (points_check[k-1]+1));
n++) {
                        if (arr9[n] == ' ') {space_check[counter_space] = n;
counter_space++;}}

                    space_check[counter_space] = (points_check[k] - points_check[k-
1] - 1);
                    for (int d = 0; d < counter_space + 1; d++) {
                        for (int q = space_check[counter_space - d - 1]+1; q <
space_check[counter_space - d]; q++) {
                            nine << arr9[q];
                        }
                        nine << " ";
                    }
                }
            }
        } else {
            for(int j = 1; j < l+1; j++) {
                int k = positions[j-1];
                for (int p = points_check[k-1]+1; p < points_check[k]; p++) {
                    if ((symb[p-1] == '.') && (symb[p] == ' ')) {
                        std::cout << "";
                    } else if ((symb[p-1] == '.') && (symb[p] == '\n')) {
                        std::cout << "";
                    } else {std::cout << symb[p];}
                }
                std::cout << '.' << std::endl;
                if (j == l) {
                    for (int p = points_check[k-1]+1; p < points_check[k]; p++) {
                    if ((symb[p-1] == '.') && (symb[p] == ' ')) {
                        continue;
                    } else if ((symb[p-1] == '.') && (symb[p] == '\n')) {
                        continue;
                    } else {arr9[count9] = symb[p]; count9++;}}
                    arr9[count9] = ' ';
                    int space_check[100] = {0};
                    int counter_space = 1;
                    space_check[0] = -1;
                    for (int n = 0; n < (points_check[k] - (points_check[k-1]+1));
n++) {
                        if (arr9[n] == ' ') {space_check[counter_space] = n;
counter_space++;}}

                    space_check[counter_space] = (points_check[k] - points_check[k-
1] - 1);
                    for (int d = 0; d < counter_space + 1; d++) {
                        for (int q = space_check[counter_space - d - 1]+1; q <
space_check[counter_space - d]; q++) {
                            nine << arr9[q];
                        }
                        nine << " ";
                    }
                }
            }
        }
    }
```

```
    if (!mfromfile) {
        char symb[10000] = {'0'};
        symb[0] = std::cin.get();
        symb[1] = std::cin.get();
        int i = 2;
        while ((symb[i-2] == '\n' && symb[i-1] == '\n') != 1) {
            symb[i] = std::cin.get();
            i++;
        } //for (int j = 0; j < i-1; j++) {std::cout << symb[j];}
        int points_check[100] = {0};
        int counter_points = 1;
        points_check[0] = -1;
        for (int j = 0; j < i-1; j++) {if (symb[j] == '.')
{points_check[counter_points] = j; counter_points++;}}
        int counter_len = 0;
        int senten_len[100] = {0};
        int l = 0;
        for (int j = 0; j < i-1; j++) {
            if ((symb[j] != '.') && (symb[j] != '\n') && (symb[j] != ' ')) {
                counter_len++;
            } else if (symb[j] == ' ') {
                if (symb[j-1] != '.') {
                    counter_len++;
                }
            } else if (symb[j] == '.') {
                senten_len[l] = counter_len;
                l++;
                counter_len = 0;
            }
        }
        int positions[100] = {0};
        for (int j = 0; j < l; j++) {
            int index_min = 0;
            for (int k = 0; k < l; k++) {
                if (senten_len[k] < senten_len[index_min]) {index_min = k;}
            }
            positions[j] = index_min+1;
            senten_len[index_min] = 10000;
        }
        if (mtofile) {
            std::ofstream tofile(tofile_name);
            if (!tofile) {
                std::cout << "Input Error" << std::endl;
                return -1;
            }
            for(int j = 1; j < l+1; j++) {
                int k = positions[j-1];
                for (int p = points_check[k-1]+1; p < points_check[k]; p++) {
                    if ((symb[p-1] == '.') && (symb[p] == ' ')) {
                        tofile << "";
                    } else if ((symb[p-1] == '.') && (symb[p] == '\n')) {
                        tofile << "";
                    } else {tofile << symb[p];}
                }
                tofile << '.' << std::endl;
                if (j == l) {
                    for (int p = points_check[k-1]+1; p < points_check[k]; p++) {
                    if ((symb[p-1] == '.') && (symb[p] == ' ')) {
                        continue;
                    } else if ((symb[p-1] == '.') && (symb[p] == '\n')) {
                        continue;
                    } else {arr9[count9] = symb[p]; count9++;}}
```

```
                    arr9[count9] = ' ';
                    int space_check[100] = {0};
                    int counter_space = 1;
                    space_check[0] = -1;
                    for (int n = 0; n < (points_check[k] - (points_check[k-1]+1));
n++) {
                        if (arr9[n] == ' ') {space_check[counter_space] = n;
counter_space++;}}

                    space_check[counter_space] = (points_check[k] - points_check[k-
1] - 1);
                    for (int d = 0; d < counter_space + 1; d++) {
                        for (int q = space_check[counter_space - d - 1]+1; q <
space_check[counter_space - d]; q++) {
                            nine << arr9[q];
                        }
                        nine << " ";
                    }
                }
            }
        } else {
            for(int j = 1; j < l+1; j++) {
                int k = positions[j-1];
                for (int p = points_check[k-1]+1; p < points_check[k]; p++) {
                    if ((symb[p-1] == '.') && (symb[p] == ' ')) {
                        std::cout << "";
                    } else if ((symb[p-1] == '.') && (symb[p] == '\n')) {
                        std::cout << "";
                    } else {std::cout << symb[p];}
                }
                std::cout << '.' << std::endl;
                if (j == l) {
                    for (int p = points_check[k-1]+1; p < points_check[k]; p++) {
                    if ((symb[p-1] == '.') && (symb[p] == ' ')) {
                        continue;
                    } else if ((symb[p-1] == '.') && (symb[p] == '\n')) {
                        continue;
                    } else {arr9[count9] = symb[p]; count9++;}}
                    arr9[count9] = ' ';
                    int space_check[100] = {0};
                    int counter_space = 1;
                    space_check[0] = -1;
                    for (int n = 0; n < (points_check[k] - (points_check[k-1]+1));
n++) {
                        if (arr9[n] == ' ') {space_check[counter_space] = n;
counter_space++;}}

                    space_check[counter_space] = (points_check[k] - points_check[k-
1] - 1);
                    for (int d = 0; d < counter_space + 1; d++) {
                        for (int q = space_check[counter_space - d - 1]+1; q <
space_check[counter_space - d]; q++) {
                            nine << arr9[q];
                        }
                        nine << " ";
                    }
                }
            }
        }
    }
}
```

*Листинг 2: файл **Makefile***

```
MyProject : main.o
	g++ main.o -o MyProject

main.o : main.cpp
	g++ -c main.cpp -o main.o

clean :
	rm MyProject *.o

distclean :
	rm MyProject *.o NINE.txt solution.txt answer.txt
```