## Выполнил Семёнов М. СКБ223

**1) Использование программы**

Флаг --fromfile считывает исходный текст из заданного файла и выводит результат в консоль.
Например:
./main –fromfile “test.txt”
Флаг --tofile считывает исходный текст с консоли и записывает его в файл “result.txt”
После ввода текста нажмите ESC + enter
При сочетании флагов программа считывает данные из заданного пользователем файла и выводит результат в файл “result.txt”

При использовании любых флагов создается файл “NINE.txt” куда программа выводит перевернутое самое длинное предложение

**2) Структура программы**

В функции main производится проверка и сопоставление флагов.
При наличии одного или обоих флагов будет вызвана соответствующая функция.
В программе написаны 2 функции для разного типа ввода:
read и read_file. Первая считывает ввод с консоли вторая - с файла заданного пользователем
Далее были написаны 2 функции для разного типа вывода:
to_file - выводит в файл и to_console - выводит в консоль.
Так же есть функция from_f_to_f которая и считывает данные с файла и выводит результат в другой файл
Вспомогательные функции:
sort - сортирует массивы
reverse - переворачивает самое длинное предложение.

```
int read(char data[100][1024], int* len, int* nomer, int
temp2){
    char Ch;
    int temp3 = 0;
    temp2 = 0;
    int temp1 = 1;
    while(Ch != 27){
        Ch = cin.peek();
        if(Ch == '\n'){
            Ch = ' ';
        }
        else if(Ch == 27){
            break;
        }
        if(Ch == '.'){
            cin.ignore();
            char Ch2 = cin.peek();
            Ch = Ch2;
            len[temp2] = temp1;
```

```
            nomer[temp2] = temp2;
            temp2++;
            temp3 = temp1;
            temp1 = 0;
        }
        data[temp2][temp1-1] = Ch;
        temp1++;
        cin.ignore();
    }
    reverse(temp3, data[temp2-1]);
    return temp2;
}

void tofile(){
    char data[100][1024];
    int len[1024];
    int nomer[1024];
    int temp2 = read(data, len, nomer, temp2);
    sort(len, nomer, temp2);
    char temp3[100][1024];
    ofstream fout("result.txt");
    for (int i = 0; i < temp2; i++) {
        fout << data[nomer[i]] << '.' << endl;
    }
    fout.close();
    reverse(strlen(data[nomer[temp2-1]]),
data[nomer[temp2-1]]);
}


void sort(int len[], int nomer[], int temp2) {
    int var;
    int var1;
    for (int i = 0; i < temp2-1; i++) {
        for (int j = 0; j < temp2 - i-1; j++) {
            if (len[j] > len[j + 1]) {
                var = nomer[j + 1];
                nomer[j + 1] = nomer[j];
                nomer[j] = var;
                var1 = len[j + 1];
                len[j + 1] = len[j];
                len[j] = var1;
            }
        }
    }
}

void to_console(char* file_name){
```

```
    char data[100][1024];
    int len[1024];
    int nomer[1024];
    int temp2 = read_file(file_name, data, len, nomer, temp2);
    sort(len, nomer, temp2);
    for (int i = 0; i < temp2; i++) {
        cout << data[nomer[i]] << '.' << endl;
    }
    reverse(strlen(data[nomer[temp2-1]]),
data[nomer[temp2-1]]);
}

int read_file(char* filename, char data0[100][1024], int*
len, int* nomer, int temp2){
    ifstream in(filename,ios::binary);
    int size_of_file = in.seekg(0,ios::end).tellg();
    in.seekg(0);
    char * data = new char[size_of_file+1];
    in.read(data,size_of_file);
    data[size_of_file] = 0;
    temp2 = 0;
    int temp1 = 1;
    for (int i = 0; i < size_of_file; i++){
        char Ch = data[i];
        if(Ch == '\n'){
            Ch = ' ';
        }
        if(Ch == '.'){
            len[temp2] = temp1;
            nomer[temp2] = temp2;
            temp2++;
            temp1 = 0;
            continue;
        }
        data0[temp2][temp1-1] = Ch;
        temp1++;
    }
    return temp2;
}

void from_f_to_f(char* filename){
    char data[100][1024];
    int len[1024];
    int nomer[1024];
    int temp2 = read_file(filename, data, len, nomer, temp2);
    sort(len, nomer, temp2);
    ofstream fout("result.txt");
    for (int i = 0; i < temp2; i++) {
```

```
        fout << data[nomer[i]] << '.' << endl;
    }
    fout.close();
    reverse(strlen(data[nomer[temp2-1]]),
data[nomer[temp2-1]]);
}

void reverse(int len, char* str){
    int temp = len;
    ofstream nine("NINE.txt");
    nine << ">>> Original sentence: " << str << endl;
    nine << ">>> Reversed sentence:";
    for (int i = len-1; i > -1; i--){
        if(str[i] == ' ' || i == 0){
            int j = i;
            while(j <= temp){
                nine << str[j];
                j++;
            }
            nine << ' ';
            temp = i;
        }
    }
    nine << endl;
    nine.close();
}
```

**3) Логика программы**

В случае ввода с консоли данные считываются посимвольно, параллельно заполняются 2 массива len и nomer, в первом длина предложения во втором его номер. Также данные делятся на предложения и заполняют массив строк data, где каждое предложение - новая строка

В случае ввода с файла он открывается как бинарник и все его содержимое копируется сразу в массив char*, потом точно так же как и при вводе с консоли, заполняется массив предложений, и[ длин и номеров.

Далее происходит одновременная сортировка пузырьком массивов длин и номеров. Далее имеется самое главное - массив номеров позволяющий просто по порядку вывести предложения в порядке возрастания. После сортировки вызывается функция reverse.

**4) Файлы программы**

main.cpp

makefile