МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

ТЕМА: ОБРАБОТКА СТРОК НА ЯЗЫКЕ СИ

Студентка гр. 2381 _____ Слабнова Д.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2022

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студентка Слабнова Дарья

Группа 2381

Тема работы: Обработка строк на языке Си

Исходные данные:

Вариант 12

Программе на вход подается текст (текст представляет собой предложения, разделенные точкой. Предложения - набор слов, разделенные пробелом или запятой, слова - набор латинских или кириллических букв, цифр и других символов кроме точки, пробела или запятой) Длина текста и каждого предложения заранее не известна.

Для хранения предложения и для хранения текста требуется реализовать структуры Sentence и Text

Программа должна сохранить (считать) текст в виде динамического массива предложений и оперировать далее только с ним. Функции обработки также должны принимать на вход либо текст (Text), либо предложение (Sentence).

Программа должна найти и удалить все повторно встречающиеся предложения (сравнивать их следует посимвольно, но без учета регистра). Далее, программа должна запрашивать у пользователя одно из следующих доступных действий (программа должна печатать для этого подсказку. Также следует предусмотреть возможность выхода из программы):

1)Сделать транслитерацию всех кириллических символов в тексте. Например, подстрока “Какой nice пень” должна принять вид “Kakoj nice pen’” (использовать ГОСТ 7.79-2000)

2)Для каждого предложения вывести все специальные символы в порядке уменьшения их кода.

3)Заменить все цифры в тексте их двоичным кодом.

4)Удалить все предложения в которых есть нечетные цифры.

Все сортировки и операции со строками должны осуществляться с использованием функций стандартной библиотеки. Использование собственных функций, при наличии аналога среди функций стандартной библиотеки, запрещается.

Каждую подзадачу следует вынести в отдельную функцию, функции сгруппировать в несколько файлов (например, функции обработки текста в один, функции ввода/вывода в другой). Также, должен быть написан Makefile.

Предполагаемый объём пояснительной записки:
Не менее 20 страниц.

Дата выдачи задания: 20.10.2022

Дата сдачи реферата: 19.01.2022

Дата защиты реферата:20.01.2022

Студентка гр. 2381 _____ Слабнова Д.А.

Преподаватель _____ Жангиров Т.Р.

АННОТАЦИЯ

Выполнением курсовой работы заключается в написании программы на языке Си, которая предназначена для обработки текста введённого пользователем.

После ввода текста, программа выводит доступные пользователю команды. Далее согласно введённой пользователем команде, программа обрабатывает текст, выводит результат и завершается.

Программа должна быть написана с использованием стандартных библиотек языка Си.

SUMMARY

The purpose of the course work is to write a program in C, which is designed to process the text entered by the user.

After entering the text, the program displays the commands available to the user. Then, according to the command entered by the user, the program processes the text, outputs the result and ends.

The program must be written using standard C libraries.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 6
1.СТРУКТУРЫ 7
1.1Структура word 7
1.2Структура sentence 7
1.3Структура text 7
2.ВВОД И ВЫВОД ТЕКСТА 8
2.1Функции ввода слова 8
2.2Функции ввода предложения 8
2.3Функция ввода текста 9
2.4Функция вывода предложения 9
2.5Функция вывода текста 10
3.ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ 11
3.1Функция сравнения слов 11
3.2Функция сравнения предложений 11
3.3Функции освобождения памяти 11
4.ОБРАБОТКА ТЕКСТА 12
4.1Команда 1 – транслитерация 12
4.2Команда 2 - вывод специальных символов в порядке уменьшения кода 13
4.3Команда 3 - замена цифр двоичным кодом 13
4.4Команда 4 – удаление предложений с нечётными цифрами 14
5.ОСНОВНАЯ ФУНКЦИЯ, СБОРКА ПРОГРАММЫ 16
5.1Функция main 16
5.2Makefile 16
ЗАКЛЮЧЕНИЕ 17
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 18
ПРИЛОЖЕНИЕ А. 19
ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ 19
ПРИЛОЖЕНИЕ Б. 22
ИСХОДНЫЙ КОД ПРОГРАММЫ 22

ВВЕДЕНИЕ

Цель работы:

Написать программу на языке Си для обработки текста. Программа должна принимать на вход текст и предлагать пользователю список возможных команд. Главные задачи:

1)Реализовать функции считывания и вывода текста.

2)Реализовать функции выполнения каждой из команд.

3)Реализовать функцию, предоставляющую пользователю выбор действия.

4)Написать Makefile

1.СТРУКТУРЫ

1.1Структура word

Структура word имеет три поля: wchar_t* symb – указатель на первый элемент массива широких символов (строку, являющаяся одним словом), int size – длина слова (количество символов), int punct – флаг, указывающий на то, какой знак препинания находится после слова(0 – нет (пробел), 1 – запятая, 2 – точка).

1.2Структура sentence

Структура sentence содержит три поля: struct word* sent – указатель на первый элемент массива слов (структур word), входящих в одно предложение, int size – количество слов в предложении, int paragraph – флаг, показывающий, является ли предложение первым в новом абзаце (начинается ли с красной строки).

1.3Структура text

Структура текст содержит два поля: struct sentence* sents – указатель на первый элемент массива предложений (структур sentence), int size – количество предложений в тексте.

2.ВВОД И ВЫВОД ТЕКСТА

2.1Функции ввода слова

Функция get_word получает на вход в качестве аргумента указатель на структуру word, в которую «записывается» считываемое из стандартного потока слово. Функция с помощью malloc выделяет память в куче для динамического массива широких символов. Далее функция считывает широкие символы по одному с помощью getwchar пробелы до первого символа, не являющегося пробелом. Далее начинается цикл, в котором функция записывает символ в соответствующую ячейку выделенного массива данных и получает новый символ на вход с помощью getwchar до тех пор, пока полученный символ не является ни пробелом, ни точкой, ни запятой. Если длина слова в процессе выполнения цикла превосходит выделенную память, то функция увеличивает место в куче с помощью realloc. Если выполнить realloc не удалось, то цикл прерывается и no_error_flag приравнивается к нулю. Далее, в зависимости от последнего символа, функция приравнивает поле punct структуры word, на которую указывает аргумент, если no_error_flag равен нулю, то punct приравнивается к 2. Длина слова подсчитывается в процессе выполнения цикла. Поле symb структуры word равно полученному слову. Функция возвращает no_error_flag (1, если слово введено успешно, иначе 0).

2.2Функции ввода предложения

Функция get_sentence получает на вход указатель на структуру sentence и считывает один символ, и в зависимости от этого поле paragraph становится равным 1 – если предложение с красной строки, или 0 – если нет. Далее функция выделяет память под массив слов с помощью malloc, и затем, аналогично предыдущей функции, с помощью цикла собирает слова (с помощью функции get_word) до первого слова с полем punct равным 2 (т.е. до слова, после которого следует точка). Если в цикле количество слов превышает длину выделенного массива, то, с помощью realloc, функция увеличивает выделенный объём память. Если увеличить объём не удалось,

или не удалось считать слово (функция get_word вернула 0), то цикл завершается и no_error_flag присваивается значение 0. Указатель на полученный массив присваивается полю sent структуре sentence, на которую указывает аргумент. Размер массива (количество слов) присваивается полю size той же структуре sentence. Функция возвращает no_error_flag.

2.3Функция ввода текста

Функция get_text получает на вход в качестве аргумента указатель на структуру text. С помощью malloc функция выделяет память в куче под массив предложений (структур sentence). Далее, в цикле функция считывает предложения с помощью функции get_sentence и записывает в соответствующие ячейки массива. Одновременно с этим, каждое считанное предложение сравнивается со всеми уже введёнными (функции сравнения см. раздел 3) и если такое же предложение без учёта регистра уже есть в «тексте», то память, занимаемая полученным предложением сразу освобождается (см. раздел 3) и цикл продолжается. Если длина выделенного в памяти массива меньше количества предложений, то с помощью функции realloc функция увеличивает выделенный в куче объём памяти. Если при считывании предложений функция get_sentence вернула 0, то цикл прерывается и начинается новый, в котором функция считывает слово и освобождает память, занимаемую им, до терминального слова «ENDENDEND.». Иначе, функция продолжает считывать предложения до терминального предложения ENDENDEND. Полю sents структуры text, на которую указывает аргументит, присваивается указатель на полученный массив. Полю size присваивается число, соответствующие количеству предложений.

2.4Функция вывода предложения

Функция out_sent получает в качестве аргумента указатель на структуру sentence, после чего с помощью цикла for функция берёт структуры word из массива структур word – поля структуры sentence, на которую указывает аргумент. Далее слова выводятся в стандартный поток вывода, печатается

сначала само слово, далее, в зависимости от значения поля punct выводится пробел, запятая – пробел, или точка.

2.5Функция вывода текста

Функция out_text с помощью цикла for выводит в стандартный поток вывода пробел и затем предложения (с помощью функции out_sent). В зависимости от поля paragraph, каждого предложения, выводит до него и пробела перенос на другую строку.

3.ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ

3.1Функция сравнения слов

Функция cmp_wrd получает на вход два указателя на структуры word. Далее функция их поля punct и массивы широких символов без учёта регистра символов (сравнивает широкие символы по одному и тому же индексу в соответствующих строках, без учёта регистра, если таким образом все символы до нулевого символа включительно равны, то строки равны). Если поля punct и все широкие символы без учёта регистра равны, то слова считаются одинаковыми(равными). Функция возвращает 1, если слова (struct word) равны, иначе возвращает 0.

3.2Функция сравнения предложений

Функция cmp_sent получает на вход два указателя на структуры sentence, предложения (структуры sentence) равны, если равны их длины и структуры word с одинаковым индексом из соответствующих массивов слов(struct word) равны по вышеописанному правилу. Функция сначала сравнивает длины предложений(поля size), и если они не равны, возвращает 0, иначе сравнивает соответствующие слова, и если они все соответственно равны, то возвращает 1, иначе 0.

3.3Функции освобождения памяти

free_sent принимает в качестве аргумента указатель на структуру sentence. С помощью цикла for освобождает память занимаемую динамическими массивами символов – полями symb, слов(структур word), входящих в массив структур word на который указывает поле sent структуры sentence, на которую указывает аргумент. Далее функция освобождает память, выделенную под массив структур word.

free_text принимает в качестве аргумента указатель на структуру text и с помощью цикла for освобождает память, занимаемую предложениями (struct sentence и «их» словами) с помощью функции free_sent. Далее функция освобождает память, выделенную под массив структур sentence.

4.ОБРАБОТКА ТЕКСТА

4.1Команда 1 – транслитерация

translit_letter – функция транслитерирует i-ый символ строки. Получает на вход указатель на строку, индекс транслитерируемого символа и размер строки. Функция, если символ заменяется при транслитерации двумя или тремя символами, с помощью функции memmove перемещает на символ или два вправо содержимое строки и заполняет оставшуюся память буквосочетанием, являющимся транслитерацией символов. Возвращает количество символов, на которое был заменен изначальный символ(«щ» при транслитерации переходит в «shh», следовательно, функция возвращает 3).

Функция trans_word транслитерирует слово(получает в качестве аргументов указатель на указатель на первый элемент строки и указатель на целое число – размер строки(длинна слова)). С помощью цикла while (до нулевого символа, т.е. конца строки), функция транслитерирует каждый символ в строке, используя translit_letter , при этом, переменная k – возвращаемое translit_letter число. Размер слова увеличивается на k-1, переменная i (индекс текущего транслитерируемого символа) – увеличивается на k. Если, во время выполнения цикла длина слова увеличивается, то функция создаёт новый динамический массив широких символов нужного размера (на самом деле на 4*sizeof(wchar_t) больше, так как, сначала добавляется новый символ в массив, а затем увеличивается память, т.е. должен быть запас на потенциальные следующие новые символы). Далее в новый массив копируется транслитерированная старая строка, затем память старой строки освобождается и указателю, на который указывает аргумент, присваивается значение равное указателю на новую строку. Если во время выполнения цикла не удалось создать новый массив(не удалось до конца транслитерировать строку), то цикл прерывается и функция возвращает 1, иначе 0.

Функция trans_sent принимает в качестве аргумента указатель на структуру sentence, после чего с помощью цикла for транслитерирует все

слова(структуры word) в массиве структур word, на который указывает поле sent структуры sentence, на которую указывает аргумент, с помощью функции trans_word. Если во время выполнения цикла trans_word возвращает 1, то цикл прерывается (транслитерация останавливается), и функция возвращает 1, иначе 0.

trans_text принимает в качестве аргумента указатель на структуру text, после чего с помощью цикла for транслитерирует все предложения(структуры sentence) в массиве структур sentence, на который указывает поле sents структуры text, на которую указывает аргумент, с помощью функции trans_sent. . Если во время выполнения цикла trans_sent возвращает 1, то цикл прерывается (транслитерация останавливается).

4.2Команда 2 - вывод специальных символов в порядке уменьшения кода

cmp_wc(const void* a, const void* b) – функция компаратор, сравнивает коды символов, на которые указывают аргументы. Возвращает 1, если (*a) < (*b), если равны, то 0, иначе -1.

out_spc_sent получает в качестве аргумента указатель на предложение, создаёт новый динамический массив buff, длина которого равна сумме длин всех слов(полей size структур word) плюс 1, в который записывает все строки и с помощью функции стандартной библиотеки qsort и описанной выше функции – компаратора, сортирует полученный массив – строку, не включая нулевой символ, и выводит полученный результат. Далее динамическая память, выделенная под массив buff, освобождается.

out_spc_text получает в качестве аргумента указатель на текст, и с помощью цикла for и вышеописанной функции выводит специальные символы каждого предложения в порядке убывания кода.

4.3Команда 3 - замена цифр двоичным кодом

bin_dig действует аналогичным translit_letter способом (так же заменяет I-ый символ строки на набор символов), только вместо транслитерации кириллических букв он заменяет цифру на её двоичную запись.

dbin_word действует аналогично trans_word, но на вход получает указатель на структуру word. Возвращает 1, если заменить одну из цифр слова двоичным кодом не удалось, иначе 0.

dbin_sent принимает в качестве аргумента указатель на структуру sentence, после чего с помощью цикла for заменяет цифры на двоичный код во всех словах(структурах word) в массиве структур word, на который указывает поле sent структуры sentence, на которую указывает аргумент, с помощью функции dbin_word . Если во время выполнения цикла dbin_word возвращает 1, то цикл прерывается, и функция возвращает 1, иначе 0.

dbin_text принимает в качестве аргумента указатель на структуру text, после чего с помощью цикла for заменяет цифры на двоичный код во всех словах(структурах sentence) в массиве структур sentence, на который указывает поле sents структуры text, на которую указывает аргумент, с помощью функции dbin_sent. . Если во время выполнения цикла dbin_sent возвращает 1, то цикл прерывается.

4.4Команда 4 – удаление предложений с нечётными цифрами

del_sent_ind – принимает на вход указатель на структуру text и целое число – индекс удаляемого предложения. Функция освобождает с помощью функции free_sent удаляемое предложение и с помощью цикла for «смещает» все предложения с большим индексом, чем заданный, влево (уменьшает каждый индекс на 1). Также уменьшает значения поля size структуры text на 1.

is_odd_word получает на вход указатель на структуру word, проверяет, есть ли в слове нечётная цифра. Функция посимвольно проверяет массив широких символов, является ли символ цифрой, с помощью библиотечной функции iswdigit, и, если символ является цифрой, то преобразует с помощью wcstol в целое число, и проверяет, является ли оно нечётным. Возвращает 1, если есть нечётная цифра, иначе 0.

is_odd_sent получает на вход указатель на структуру sentence, проверяет с помощью вышеописанной функции, есть ли в предложении слова с

нечётными цифрами. Возвращает 1, если есть, иначе 0.

del_odd_sent получает на вход указатель на структуру text. С помощью цикла for проверяет каждое предложение на то, есть ли в нем нечётные цифры (с помощью вышеописанной функции), и если есть, с помощью del_sent_ind удаляет.

5.ОСНОВНАЯ ФУНКЦИЯ, СБОРКА ПРОГРАММЫ

5.1Функция main

В функции main инициализируется переменная struct text tmp, в которой будет храниться обрабатываемый текст. Далее функция выводит текст, предлагающий пользователю ввести текст, после ввода функция выводит подсказки, с помощью которых пользователь может выбрать способ обработки текста и ввести соответствующий символ. После ввода – команды пользователя, в зависимости от введенного символа с помощью оператора switch, текст соответствующим образом обрабатывается, результат выводится на экран с помощью out_text и память освобождается с помощью free_text. Функция завершается.

5.2Makefile

Исходная программа содержит 5 фвайлов

1)input_output_free.c – программа на языке си, содержащая функции ввода, вывода текста и освобождения памяти, и iof.h – соответствующий ей заголовочный файл.

2)func.c – программа на языке си, содержащая функции обработки текста, и func.h – соответствующий ей заголовочный файл.

3)m.c – программа, содержащая функцию main. Включает в себя заголовочные файлы iof.h и func.h

Так как стандартные библиотеки и некоторые объявления функций и структур включены в оба заголовочных файла(и они оба включаются в m.c), в func.h перед включениями, присутствующими в файле iof.h используется директива ifndef.

Также изпользовалось ПО valgrind для проверки на отсутствие утечек памяти.

ЗАКЛЮЧЕНИЕ

Все цели перечисленные в введении были достигнуты: реализованы функции ввода, вывода и освобождения текста, функции обработки текста, написан Makefile. В результате работы были изучены стандартные библиотеки языка Си и получены навыки работы с ними.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Брайан Керниган, Деснис Ритчи «Язык программирования Си»
2. https://cplusplus.com

ПРИЛОЖЕНИЕ А.
ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

```
dasha@dasha:~/hardcore_pron/pr-2022-2381/Slabnova_Darya_cw/src$ make -f Makefile2
gcc -c m.c
gcc m.o input_output_free.o func.o -o CW
dasha@dasha:~/hardcore_pron/pr-2022-2381/Slabnova_Darya_cw/src$ ./CW



Курсовая работа по программированию, 1 семестр.
Выполнена Слабновой Дарьей 2381


Введите текст.
Не забывайте о красной строке(пробеле) перед первым предложением абзаца.
Текст должен заканчиваться терминальным предложением "ENDENDEND."
 Одинаковое Предложение. Транслитераторы транслитерировали транслитируемый текст. ОДИНАкоВОЕ ПРЕДЛОЖЕНИЕ. ENDENDEND.



выберите действие:
1 - транслитерировать текст
2 - вывести специальные символы каждого предложения в порядке уменьшения кода
3 - заменить все цифры в тексте их двоичным кодом
4 - Удалить все предложения в которых есть нечетные цифры
1
 Odinakovoe Predlozhenie. Transliteratory' transliterirovali translitiruemy'j tekst.


Спасибо за внимание!


dasha@dasha:~/hardcore_pron/pr-2022-2381/Slabnova_Darya_cw/src$
```

Пример 1. Проверка команды 1. Команда работает корректно, повторное предложение было удалено.

```
dasha@dasha:~/hardcore_pron/pr-2022-2381/Slabnova_Darya_cw/src$ ./CW



Курсовая работа по программированию, 1 семестр.
Выполнена Слабновой Дарьей 2381


Введите текст.
Не забывайте о красной строке(пробеле) перед первым предложением абзаца.
Текст должен заканчиваться терминальным предложением "ENDENDEND."
 Одинаковое Предложение. Транслитераторы транслитерировали транслитируемый текст. ОДИНАкоВОЕ ПРЕДЛОЖЕНИЕ. ENDENDEND.



выберите действие:
1 - транслитерировать текст
2 - вывести специальные символы каждого предложения в порядке уменьшения кода
3 - заменить все цифры в тексте их двоичным кодом
4 - Удалить все предложения в которых есть нечетные цифры
2

предложение №0 - рооонннлкиижеееедквапО
предложение №1 - ыыуттттттттсссспрррррррроонннмллллкйиииииииеееевааааат


Спасибо за внимание!


dasha@dasha:~/hardcore_pron/pr-2022-2381/Slabnova_Darya_cw/src$
```

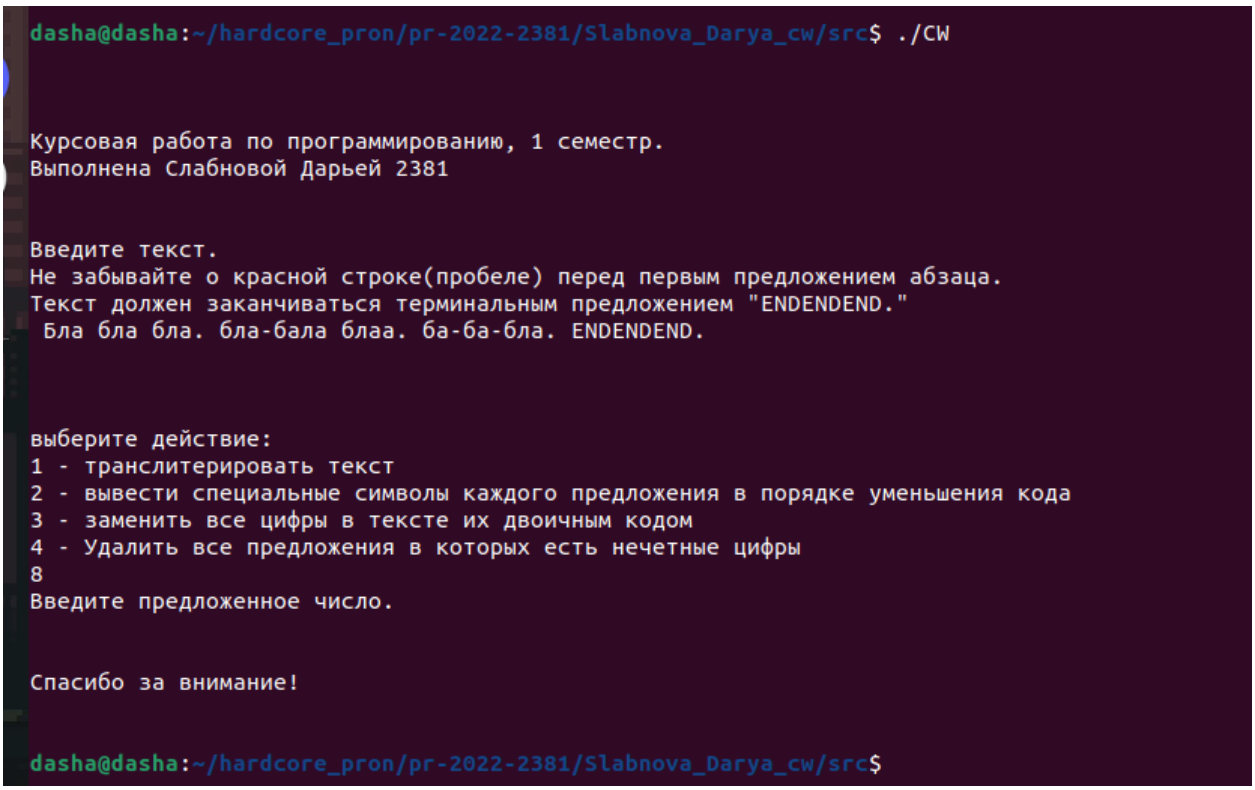
Пример 2. Проверка команды 2. Команда работает корректно, повторное предложение было удалено.

```
dasha@dasha:~/hardcore_pron/pr-2022-2381/Slabnova_Darya_cw/src$ ./CW


Курсовая работа по программированию, 1 семестр.
Выполнена Слабновой Дарьей 2381


Введите текст.
Не забывайте о красной строке(пробеле) перед первым предложением абзаца.
Текст должен заканчиваться терминальным предложением "ENDENDEND."
 7 гномов живут вместе. Плюс 1 Белоснежка. ENDENDEND.



выберите действие:
1 - транслитерировать текст
2 - вывести специальные символы каждого предложения в порядке уменьшения кода
3 - заменить все цифры в тексте их двоичным кодом
4 - Удалить все предложения в которых есть нечетные цифры
3
 111 гномов живут вместе. Плюс 1 Белоснежка.


Спасибо за внимание!


dasha@dasha:~/hardcore_pron/pr-2022-2381/Slabnova_Darya_cw/src$
```

Пример 3. Проверка команды 3. Команда работает корректно.

```
dasha@dasha:~/hardcore_pron/pr-2022-2381/Slabnova_Darya_cw/src$ ./CW


Курсовая работа по программированию, 1 семестр.
Выполнена Слабновой Дарьей 2381


Введите текст.
Не забывайте о красной строке(пробеле) перед первым предложением абзаца.
Текст должен заканчиваться терминальным предложением "ENDENDEND."
 7 гномов живут вместе. Плюс 2 - Белоснежка и принц. ENDENDEND.



выберите действие:
1 - транслитерировать текст
2 - вывести специальные символы каждого предложения в порядке уменьшения кода
3 - заменить все цифры в тексте их двоичным кодом
4 - Удалить все предложения в которых есть нечетные цифры
4
 Плюс 2 - Белоснежка и принц.


Спасибо за внимание!


dasha@dasha:~/hardcore_pron/pr-2022-2381/Slabnova_Darya_cw/src$
```

Пример 4. Проверка команды 4. Команда работает корректно.



Пример 5. Некорректно введённая команда.

ПРИЛОЖЕНИЕ Б.
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: m.c

```
#include "iof.h"
#include "func.h"

int main(){
	setlocale(LC_CTYPE, "");
	struct text tmp;
	printf("\n\n\nКурсовая работа по программированию, 1 семестр.\nВыполнена Слабновой Дарьей 2381\n\n\nВведите текст.\nНе забывайте о красной строке(пробеле) перед первым предложением абзаца.\nТекст должен заканчиваться терминальным предложением \"ENDENDEND.\"\n");
	get_text(&tmp);
	printf("\n\n\nвыберите действие:\n1 - транслитерировать текст\n2 - вывести специальные символы каждого предложения в порядке уменьшения кода\n3 - заменить все цифры в тексте их двоичным кодом\n4 - Удалить все предложения в которых есть нечетные цифры\n");
	wchar_t c;
	wchar_t trash;
	trash = getwchar();
	c = getwchar();
	do{
		trash = getwchar();
	}while(trash != L'\n');

	switch(c){
		case L'1':
			trans_text(&tmp);
			out_text(&tmp);
			break;
		case L'2':
			out_spc_text(&tmp);
			break;
		case L'3':
			dbin_text(&tmp);
			out_text(&tmp);
			break;
		case L'4':
			del_odd_sent(&tmp);
			out_text(&tmp);
			break;
		default:
			printf("Введите предложенное число.\n");
	}
	free_text(&tmp);
	printf("\n\nСпасибо за внимание!\n\n\n");
	return 0;
}
```

Название файла: input_output_free.c

```
#include "iof.h"
```

```
int get_word(struct word* my_word){
        setlocale(LC_CTYPE, "");
        wchar_t* res_symb = malloc(sizeof(wchar_t)*MW);
        int punct = 0;

        wchar_t* trash; //for overload
        int k=0;

        wchar_t c;
        int i = 0;
        int no_error_flag = 1;
     static int err_wrd_num = 0;

        do{                     //to make spaces go away
                c = getwchar();
        }while(c == ' ');


         //собираем слово
         while(c != ' ' && c != ',' && c!= '.'){
                res_symb[i] = c;
                i++;
                c = getwchar();
           if(i == MW + k-4)
                           k+=AW;
                           trash          =          realloc(res_symb,
(MW+k)*sizeof(wchar_t));
                           if(trash == NULL){
                           printf("%sнеудалось  ввести  слово  №%d%s\n",
RED, err_wrd_num ,NONE);
                      i--;
                                no_error_flag = 0;
                                break;
                           }
                      res_symb = trash;
           }
                        //конец случая переполнения


     }

     my_word->size = i;
     res_symb[i] = '\0';
     my_word->symb = res_symb;
          if(no_error_flag){
                if(c == '.'){
                           punct = 2;
                }
                if(c == ','){
                     punct = 1;
                }
                my_word->punct = punct;

          }else{
                my_word->punct = 2;

          }
```

```
      err_wrd_num++;
      return no_error_flag;

}


int cmp_wrd(struct word* w1, struct word* w2){
        int tmp = 1;
        if((*w1).punct != (*w2).punct){
                return 0;
        }
        int i = 0;
        while((*w1).symb[i] != L'\0' && (*w2).symb[i] != L'\0'){
                if(towlower((*w1).symb[i])                                  !=
towlower((*w2).symb[i])){
                        tmp = 0;
                        break;
                }
                i++;
        }
        return tmp;
}



int get_sentence(struct sentence* predlog){
      setlocale(LC_CTYPE, "");
        struct word* res = malloc(sizeof(struct word)*MS);
        int i = 0;

        struct word* trash;
        int k = 0;
        int no_error_flag = 1;
      static int num_err = 0;

      wchar_t c = getwchar();
      if(c == ' '){
           predlog->paragraph = 0;
      }else{
           predlog->paragraph = 1;
      }

        do{
                if(i == MS+k-1){
                        k+=AS;
                        trash   =   realloc(res,   (MS+k)*sizeof(struct
word));
                        if(trash == NULL){
                                fprintf(stderr,          "\n%sпоследнее
введённое слово - %ls, в предложении №%d%s\n", RED, (res[i-1]).symb,
num_err, NONE);
                                no_error_flag = 0;
                                break;
                        }
                        res = trash;

                }
```

```
                no_error_flag = get_word(&res[i]);
            if(!no_error_flag){
                break;
            }
                i++;
        }while((res[i-1]).punct != 2);

    char f;
    if(!no_error_flag){
        do{
            f = getchar();
        }while(c != '.');
    }

        predlog->sent = res;
        predlog->size = i;

    num_err++;
    return no_error_flag;
}

void out_sent(struct sentence* predlog){
    setlocale(LC_CTYPE, "");
        struct word tmp;
        for(int i = 0; i < predlog->size; i++){
                tmp = (predlog->sent)[i];
                if(tmp.punct == 1){
                        printf("%ls, ", tmp.symb);
                }else if(tmp.punct == 2){
                        printf("%ls.", tmp.symb);
                }else{
                        printf("%ls ", tmp.symb);
                }
        }
}

void free_sent(struct sentence* predlog){
        for(int i = 0; i < predlog->size; i++){
                free(((*predlog).sent)[i].symb);
        }
        free(predlog->sent);

}


int cmp_sent(struct sentence* s1, struct sentence* s2){
        if(s1->size != s2->size){
                return 0;
        }else{
                for(int i = 0; i < s1->size; i++){
                        if(!cmp_wrd(s1->sent + i, s2->sent + i)){
                                return 0;
                        }
                }

                return 1;
        }
```

```
}

void get_text(struct text* essay){
        struct sentence* res = malloc(sizeof(struct sentence)*MT);
        int i = 0;
        struct sentence tmp = {NULL, 0, 0};
        int is_same_flag = 0;

      struct word term_w = {L"ENDENDEND", 9, 2};
      struct word t = {NULL, 0, 0};
      struct sentence term_s = {&term_w, 1, 0};

        int k = 0;
        struct sentence* trash;
        int no_error_flag = 1;


        do{
                is_same_flag = 0;

                if(i == MT+k){
                        k+=AT;
                        trash     =     realloc(res,     sizeof(struct
sentence)*(MT+k));
                        if(trash == NULL){
                                fprintf(stderr ,"\nпоследнее введённое
предложение - %d\n", i);
                                no_error_flag = 0;
                                break;
                        }
                        res = trash;
                }

                no_error_flag = get_sentence(&tmp);

           if(!no_error_flag){
                break;
           }

                for(int l = 0; l < i; l++){
                        if(cmp_sent(&tmp, &res[l])){
                                is_same_flag = 1;
                                break;
                        }
                }

                if(is_same_flag){
                        free_sent(&tmp);
                }else{
                        res[i] = tmp;
                        i++;
                }

        }while(!cmp_sent(&term_s, &res[i-1]) && no_error_flag);
```

```
	if(no_error_flag){
		free_sent(&(res[i-1]));
		i--;
	}else{
		do{
			free(t.symb);
			get_word(&t);
		}while(!cmp_wrd(&term_w, &t));
		free(t.symb);
	}

		(*essay).sents = res;
		(*essay).size = i;
}

void out_text(struct text* essay){
	for(int i = 0; i < (*essay).size; i++){
		if((essay->sents)[i].paragraph){
			printf("\n");
		}
		printf(" ");
			out_sent((*essay).sents+i);
	}
	printf("\n");
}

void free_text(struct text* essay){
	for(int i = 0; i < (*essay).size; i++){
			free_sent((*essay).sents+i);
	}
	free((*essay).sents);
}
```

Название файла: func.c

```
#include "func.h"

int	translit_letter(wchar_t*	str,int	i,	int	size){
	setlocale(LC_CTYPE, "");
	wchar_t tmp = str[i];
	switch(towlower(tmp)){
		//буквы от A до E
		case L'a':
			if(iswupper(tmp)){
					str[i] = L'A';
				}else{
					str[i] = L'a';
				}
				return 1;
		case L'б':
				if(iswupper(tmp)){
					str[i] = L'B';
				}else{
					str[i] = L'b';
				}
				return 1;
		case L'в':
				if(iswupper(tmp)){
					str[i] = L'V';
```

```
                        }else{
                                str[i] = L'v';
                        }
                        return 1;
                case L'г':
                     if(iswupper(tmp)){
                          str[i] = L'G';
                     }else{
                          str[i] = L'g';
                     }
                     return 1;
                case L'д':
                        if(iswupper(tmp)){
                                str[i] = L'D';
                        }else{
                                str[i] = L'd';
                        }
                        return 1;
                case L'е':
                        if(iswupper(tmp)){
                                str[i] = L'E';
                        }else{
                                str[i] = L'e';
                        }
                        return 1;
                //буквы от Ё до К
                case L'ё':
                        if(iswupper(tmp)){
                                wmemmove(str+i+1,   str+i,   size-i+1);
str[i] = L'Y';
                                str[i+1] = L'o';
                        }else{
                                wmemmove(str+i+1, str+i, size-i+1);
                                str[i] = L'y';
                                str[i+1] = L'o';

                        }
                        return 2;
                 case L'ж':
                        if(iswupper(tmp)){
                                wmemmove(str+i+1, str+i, size-i+1);
                                str[i] = L'Z';
                                str[i+1] = L'h';
                        }else{
                                wmemmove(str+i+1,   str+i,   size-i+1);
str[i] = L'z';
                                str[i+1] = L'h';

                        }
                        return 2;
                case L'з':
                        if(iswupper(tmp)){
                                str[i] = L'Z';
                        }else{
                                str[i] = L'z';
                        }
                        return 1;
```

```
case L'и':
		if(iswupper(tmp)){
			str[i] = L'I';
		}else{
			str[i] = L'i';
		}
		return 1;
case L'й':
		if(iswupper(tmp)){
			str[i] = L'J';
		}else{
			str[i] = L'j';
		}
		return 1;
case L'к':
		if(iswupper(tmp)){
			str[i] = L'K';
		}else{
			str[i] = L'k';
		}
		return 1;
//буквы от Л до Р
case L'л':
		if(iswupper(tmp)){
			str[i] = L'L';
		}else{
			str[i] = L'l';
		}
		return 1;
	case L'м':
		if(iswupper(tmp)){
			str[i] = L'M';
		}else{
			str[i] = L'm';
		}
		return 1;
	case L'н':
		if(iswupper(tmp)){
			str[i] = L'N';
		}else{
			str[i] = L'n';
		}
		return 1;
	case L'о':
		if(iswupper(tmp)){
			str[i] = L'O';
		}else{
			str[i] = L'o';
		}
		return 1;
case L'п':
		if(iswupper(tmp)){
			str[i] = L'P';
		}else{
			str[i] = L'p';
		}
		return 1;
```

```
                case L'р':
                        if(iswupper(tmp)){
                                str[i] = L'R';
                        }else{
                                str[i] = L'r';
                        }
                        return 1;
                     case L'с':
                        if(iswupper(tmp)){
                                str[i] = L'S';
                        }else{
                                str[i] = L's';
                        }
                        return 1;
                case L'т':
                        if(iswupper(tmp)){
                                str[i] = L'T';
                        }else{
                                str[i] = L't';
                        }
                        return 1;
                case L'у':
                        if(iswupper(tmp)){
                                str[i] = L'U';
                        }else{
                                str[i] = L'u';
                        }
                        return 1;
                case L'Ф':
                        if(iswupper(tmp)){
                                str[i] = L'F';
                        }else{
                                str[i] = L'f';
                        }
                        return 1;
                case L'х':
                        if(iswupper(tmp)){
                                str[i] = L'X';
                        }else{
                                str[i] = L'x';
                        }
                        return 1;
            case L'ц':
                        if(iswupper(tmp)){
                                wmemmove(str+i+1, str+i, size-i+1);
                                str[i] = L'C';
                                str[i+1] = L'z';
                        }else{
                                wmemmove(str+i+1,   str+i,   size-i+1);
str[i] = L'c';
                                str[i+1] = L'z';

                        }
                        return 2;
            case L'ч':
                        if(iswupper(tmp)){
                                wmemmove(str+i+1, str+i, size-i+1);
```

```
                                str[i] = L'C';
                                str[i+1] = L'h';
                        }else{
                                wmemmove(str+i+1,   str+i,   size-i+1);
str[i] = L'c';
                                str[i+1] = L'h';

                        }
                        return 2;
        case L'ш':
                        if(iswupper(tmp)){
                                wmemmove(str+i+1, str+i, size-i+1);
                                str[i] = L'S';
                                str[i+1] = L'h';
                        }else{
                                wmemmove(str+i+1, str+i, size-i+1);
                                str[i] = L's';
                                str[i+1] = L'h';

                        }
                        return 2;
        case L'щ':
                        if(iswupper(tmp)){
                                wmemmove(str+i+2, str+i, size-i+1);
                                str[i] = L'S';
                                str[i+1] = L'h';
                    str[i+2] = L'h';
                        }else{
                                wmemmove(str+i+2, str+i, size-i+1);
                                str[i] = L's';
                                str[i+1] = L'h';
                    str[i+2] = L'h';

                        }
                        return 3;
             case L'ъ':

                           wmemmove(str+i+1,     str+i,     size-i+1);
    str[i] = 39;
                           str[i+1] = 39;

                        return 2;
        case L'ы':
                        if(iswupper(tmp)){
                                wmemmove(str+i+1, str+i, size-i+1);
                                str[i] = L'Y';
                                str[i+1] = 39;
                        }else{
                                wmemmove(str+i+1, str+i, size-i+1);
                                str[i] = L'y';
                                str[i+1] = 39;

                        }
                        return 2;
        case L'ь':
                                str[i] = 39;
                        return 1;
```

```
            case L'э':
                        if(iswupper(tmp)){
                                wmemmove(str+i+1, str+i, size-i+1);
                                str[i] = L'E';
                                str[i+1] = 39;
                        }else{
                                wmemmove(str+i+1, str+i, size-i+1);
                                str[i] = L'e';
                                str[i+1] = 39;

                        }
                        return 2;
            case L'ю':
                        if(iswupper(tmp)){
                                wmemmove(str+i+1,   str+i,   size-i+1);
str[i] = L'Y';
                                str[i+1] = L'u';
                        }else{
                                wmemmove(str+i+1, str+i, size-i+1);
                                str[i] = L'y';
                                str[i+1] = L'u';

                        }
                        return 2;
            case L'я':
                  if(iswupper(tmp)){
                        wmemmove(str+i+1, str+i, size-i+1);
      str[i] = L'Y';
                        str[i+1] = L'a';
                  }else{
                        wmemmove(str+i+1,        str+i,        size-i+1);
str[i] = L'y';
                                str[i+1] = L'a';

                  }
                  return 2;
            default:
                  return 1;
      }

}

int trans_word(wchar_t** a, int* size){
      int i = 0;
      int k;

      static int num_err = 1;
      int err = 0;

      wchar_t* trash;
      do{
            k = translit_letter(*a, i, *size);
            i+=k;
            *size += (k - 1);
            if(k > 1){
                  trash = malloc((*size + k + 2)*sizeof(wchar_t)); //+
```

```
(k-1) + 3
                if(trash == NULL){
                     fprintf(stderr,  "\n%sНеудалось  транслитерировать
слово - %ls%s\n", RED, *a, NONE);
                     err = 1;
                     break;
                }
                wcscpy(trash, *a);
                free(*a);
                *a = trash;

           }

      }while((*a)[i] != L'\0');
      num_err++;
      return err;
}

int trans_sent(struct sentence* a){
      int k = 0;
      for(int i = 0; i < (*a).size; i++){
           k           =           trans_word(&((((*a).sent)[i]).symb),
&((((*a).sent)[i]).size));
           if(k) break;
      }
      return k;
}
void trans_text(struct text* a){
      int k = 0;
      for(int i = 0; i < (*a).size; i++){
           k = trans_sent((*a).sents+i);
           if(k) break;
      }
}



//2nd task
//
int cmp_wc(const void* a, const void* b){
      wchar_t* chr1 = (wchar_t*)a;
      wchar_t* chr2 = (wchar_t*)b;
      if((*chr1) > (*chr2)){
           return -1;
      }
      if((*chr1) < (*chr2)){
           return 1;
      }
      return 0;
}

void out_spc_sent(struct sentence* a){

      int size = 0;
      for(int i = 0; i < (*a).size; i++){
           size+=(a->sent)[i].size;
      }
```

```
	wchar_t* buff = malloc(sizeof(wchar_t)*(size+1));
	int count = 0;

	struct word tmp;
	for(int i = 0; i < (*a).size; i++){
		tmp = (*a).sent[i];
		for(int j = 0; j < tmp.size; j++){
		buff[count] = tmp.symb[j];
		count++;

		}
	}

	buff[count] = L'\0';
	qsort(buff, count, sizeof(wchar_t), cmp_wc);
	printf("%ls\n", buff);
	free(buff);

}

void out_spc_text(struct text* a){
	printf("\n");
	for(int i = 0; i < (*a).size; i++){
			printf("предложение №%d - ", i);
			out_spc_sent((*a).sents+i);
	}
}

//3rd task
//
int bin_dig(wchar_t* str,int i, int size){
	wchar_t buff[2];
	buff[0] = str[i];
	buff[1] = L'\0';
	   long int tmp = wcstol(buff, NULL, 10);
	   switch(tmp){
	      case 0:
			str[i] = L'0';
			return 1;
	      case 1:
			str[i] = L'1';
			return 1;
	      case 2:
			wmemmove(str+i+1, str+i, size-i+1);
			str[i] = L'1';
			str[i+1] = L'0';
			return 2;
	      case 3:
			wmemmove(str+i+1, str+i, size-i+1);
				str[i] = L'1';
				str[i+1] = L'1';
			return 2;
	      case 4:
				wmemmove(str+i+2, str+i, size-i+1);
				str[i] = L'1';
				str[i+1] = L'0';
```

```
                str[i+2] = L'0';
                return 3;
           case 5:
                wmemmove(str+i+2, str+i, size-i+1);
                        str[i] = L'1';
                        str[i+1] = L'0';
                        str[i+2] = L'1';
                return 3;
           case 6:
                wmemmove(str+i+2, str+i, size-i+1);
                        str[i] = L'1';
                        str[i+1] = L'1';
                        str[i+2] = L'0';
                return 3;
           case 7:
                        wmemmove(str+i+2, str+i, size-i+1);
                        str[i] = L'1';
                        str[i+1] = L'1';
                        str[i+2] = L'1';
                return 3;
           case 8:
                wmemmove(str+i+3, str+i, size-i+1);
                        str[i] = L'1';
                        str[i+1] = L'0';
                        str[i+2] = L'0';
                str[i+3] = L'0';
                return 4;
           case 9:
                wmemmove(str+i+3, str+i, size-i+1);
                        str[i] = L'1';
                        str[i+1] = L'0';
                        str[i+2] = L'0';
                        str[i+3] = L'1';
                return 4;
      }
}


int dbin_word(struct word* a){
     int i = 0;
        int k;

        static int num_err = 1;
        int err = 0;

        wchar_t* trash;
     do{
           if(iswdigit((*a).symb[i])){
                k = bin_dig((*a).symb, i, (*a).size);
                i+=k;
                (*a).size += (k - 1);
                if(k > 1){
                     trash = malloc((a->size + 3)*(sizeof(wchar_t)));
                     if(trash == NULL){
                          fprintf(stderr,    "\n%sНеудалось    изменить
цифру в слове - %ls%s\n", RED, (*a).symb, NONE);
                          err = 1;
```

```
                              break;
                        }
                        wcscpy(trash, a->symb);
                              free(a->symb);
                              a->symb = trash;
                  }
            }else{
                  i++;
            }
      }while((*a).symb[i] != L'\0');
      num_err++;
      return err;

}

int dbin_sent(struct sentence* a){
      int k = 0;
         for(int i = 0; i < (*a).size; i++){
                  k = dbin_word((*a).sent+i);
            if(k) break;
         }
      return k;
}
void dbin_text(struct text* a){
      int k = 0;
         for(int i = 0; i < (*a).size; i++){
                  k = dbin_sent((*a).sents+i);
            if(k) break;
         }
}

//4th
//
int is_odd_word(struct word* a){
      long int tmp;
      wchar_t buff[2];
      buff[1] = L'\0';
      for(int i = 0; i < (*a).size; i++){
            if(iswdigit((*a).symb[i])){
                  buff[0] = (*a).symb[i];
                  tmp = wcstol(buff, NULL, 10);
                  if(tmp % 2 == 1){
                        return 1;
                  }
            }
      }
      return 0;
}



int is_odd_sent(struct sentence* a){
      for(int i = 0; i < (*a).size; i++){
            if(is_odd_word((*a).sent+i)){
                  return 1;
            }
      }
```

```
	return 0;
}

void del_sent_ind(struct text* a, int ind){
	free_sent((a->sents)+ind);
	for(int i = ind+1; i < a->size; i++){
		((*a).sents)[i-1] = ((*a).sents)[i];
	}
	(*a).size--;
}

void del_odd_sent(struct text* a){
	int i = 0;
	for(int i = 0; i < (*a).size; i++){
		if(is_odd_sent((*a).sents+i)){
			del_sent_ind(a, i);
		}
	}
	}
```

Название файла: iof.h

```
#define JUST_HERE

#include <stdio.h>
#include <stdlib.h>
#include <wchar.h>
#include <wctype.h>
#include <locale.h>

#define MW 6
#define AW 2
#define MS 15
#define AS 3
#define MT 10
#define AT 3

#define RED "\033[0;31m"
#define NONE   "\033[0m"

struct word{
	wchar_t* symb;
	int size;
	int punct;
};
struct sentence{
	struct word* sent;
	int size;
	int paragraph;
};
struct text{
	struct sentence* sents; //pointer to the
	int size;
};
int get_word(struct word* my_word);
int cmp_wrd(struct word* w1, struct word* w2);
void out_sent(struct sentence* predlog);
int get_sentence(struct sentence* predlog);
```

```
void free_sent(struct sentence* predlog);
int cmp_sent(struct sentence* s1, struct sentence* s2);
void get_text(struct text* essay);
void out_text(struct text* essay);
void free_text(struct text* essay);
```

Название файла: func.h

```
#ifndef JUST_HERE

#include <stdio.h>
#include <stdlib.h>
#include <wchar.h>
#include <wctype.h>
#include <locale.h>

#define RED "\033[0;31m"
#define NONE   "\033[0m"

struct word{
        wchar_t* symb;
        int size;
        int punct;
};

struct sentence{
        struct word* sent;
        int size;
        int paragraph;
};

struct text{
        struct sentence* sents;
        int size;
};

void free_sent(struct sentence* predlog);
#endif

int translit_letter(wchar_t* str,int i, int size);
int trans_word(wchar_t** a, int* size);
int trans_sent(struct sentence* a);
void trans_text(struct text* a);

int cmp_wc(const void* a, const void* b);
void out_spc_sent(struct sentence* a);
void out_spc_text(struct text* a);

int bin_dig(wchar_t* str,int i, int size);
int dbin_word(struct word* a);
int dbin_sent(struct sentence* a);
void dbin_text(struct text* a);

int is_odd_word(struct word* a);
int is_odd_sent(struct sentence* a);
void del_sent_ind(struct text*a, int ind);
void del_odd_sent(struct text* a);
```

Название файла: Makefile2

```
all: CW

CW: m.o input_output_free.o func.o
	gcc m.o input_output_free.o func.o -o CW

m.o: m.c iof.h func.h
	gcc -c m.c

input_output_free.o: input_output_free.c iof.h
	gcc -c input_output_free.c

func.o: func.c func.h
	gcc -c func.c
clean:
	rm -rf *.o
memcheck:
	valgrind --leak-check=full ./CW
```