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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка текстовых данных

Студент гр. 1304 _____ Стародубов М.В.

Преподаватель _____ Чайка К.В.

Санкт-Петербург

2021

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Стародубов М.В.

Группа 1304

Тема работы: Обработка текстовых данных

Исходные данные:
Реализовать программу на языке программирования Си с определенным функционалом, выполняющую обработку текстовых данных согласно инструкциям пользователя, пока не будет введена команда завершения работы.
Содержание пояснительной записки:
«Содержание», «Введение», «Ход работы», «Тестирование и примеры работы программы», «Заключение», «Список использованных источников», «Приложение А. Исходный код программы»

Предполагаемый объем пояснительной записки:
Не менее 12 страниц.

Дата выдачи задания: 15.10.2021

Дата сдачи реферата: 18.12.2021

Дата защиты реферата: 21.12.2021

Студент _____ Стародубов М.В.

Преподаватель _____ Чайка К.В.

АННОТАЦИЯ

В ходе выполнения курсовой работы была разработана программа на языке программирования Си, выполняющая некоторые действия над строками. Для хранения идущего на вход программе текста, а также для удобства дальнейшей работы с ним были реализованы структуры, в ходе работы программа использует функции стандартных библиотек языка Си. В ходе разработки программы было решено разделить исходный код на 5 файлов: *main.c*, *get_text.c*, *main_functionality.c*, *time_diffs.c*, *useful.c*. В ходе тестирования ошибок работы программы выявлено не было.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цели.

Целью данной работы является улучшение навыков написания программ на языке программирования Си, и, как результат этого, разработка программы, которая выполняет команды пользователя по обработке идущего на вход программы текста.

Задачи.

Для достижения поставленной цели требуется решить следующие задачи:

– Более глубокое изучение языка программирования Си.

– Выполнение задания курсовой работы по написанию программного продукта.

– Тестирование и отладка разработанной программы.

1. ХОД РАБОТЫ

1.1. Структура файлов исходного кода программы.

Программа разделена на 5 основных файлов с исходным кодом:

main.c – файл содержит в себе определение функции *main*. Внутри функции *main* происходит основное взаимодействие программы и пользователя: функция запрашивает у пользователя ввод текста и команды, в соответствии с которыми вызываются необходимые функции.

get_text.c – файл содержит в себе определения функций, которые служат для получения текста, с которым позже и будет оперировать программа. Объявление функций, содержащихся в данном файле находятся в заголовочном файле *get_text.h*.

main_functionality.c – в файле находятся определения функций по обработке текста. Данные функции являются основным функционалом программы. Объявление функций, содержащихся в данном файле находятся в заголовочном файле *main_functionality.h*.

time_diffs.c – в файле содержатся описания функций для реализации работы первой подпрограммы в задании курсовой работы (Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер предложения в котором она встречается и количество минут до текущего времени (время в которое начала выполняться данная задача)). Данные функции были вынесены в отдельный файл, так как помимо обработки текста они производят вывод сообщений в терминал. Объявление функций, содержащихся в данном файле находятся в заголовочном файле *time_diffs.h*.

useful.c – функции, описанные в этом файле, созданы для удобства, они выполняют действия, которые могут быть многократно совершены в процессе работы программы (к примеру функция вывода текста в консоль). Объявление функций, содержащихся в данном файле находятся в заголовочном файле *useful.h*.

Также в структуру исходного кода программы входят заголовочные файлы *structs.h* и *colors.h*. В файле *structs.h* описаны структуры, которые используются для хранения текста. В файле *colors.h* написаны определения кодов для цветного вывода текста в консоль. Также был написан *Makefile* для быстрой сборки программы.

1.2. Файл *structs.h*. Структуры хранения текста и предложения.

Для хранения идущего на вход программе текста были созданы структуры *Sentence* и *Text*.

Структура *Sentence* предназначена для хранения одного предложения. Основное поле структуры – поле *content*, оно содержит в себе указатель на динамически выделенный массив широких символов, т. е. на символы, из которых и состоит предложение. Поле *number_of_cirillic_characters* хранит в себе количество кириллических символов в пердложении, оно необходимо для выполнения одной из подзадач программы. Поле *are_there_sp_symbols* принимает значение 1 или 0 в зависимости от того, присутствуют ли в предложении специальные символы (Небольшое уточнение: считается, что в предложении отсутствуют специальные символы, если предложение состоит только из букв кириллического и латинского алфавитов или цифр, точка в конце каждого предложения, в данном слуяае, не считается символом предложения). Поле *size* хранит в себе количество символов в пердложении, включая точку. Уточнение: поля *number_of_cirillic_characters* и *are_there_sp_symbols* определяются при получении предложения и не изменяются в процессе дальнейшей работы программы.

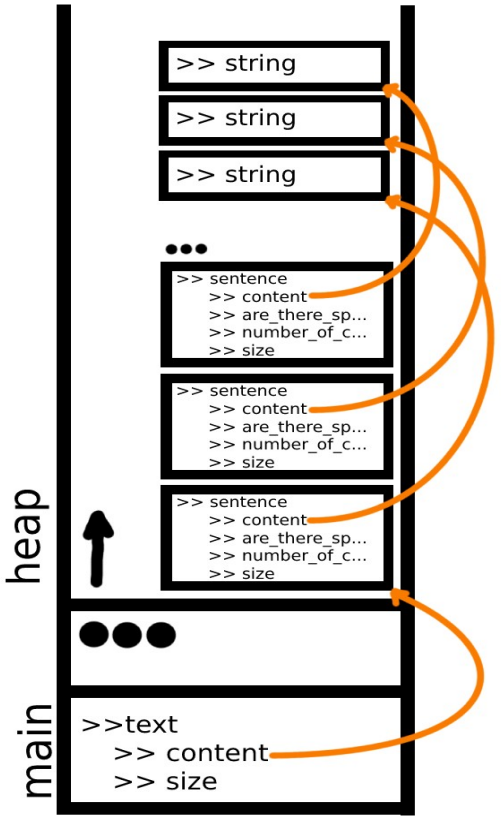
Структура *Text* предназначена для хранения всего текста. Основное поле – *content* – содержит в себе указатель на динамический массив структур *Sentence*. Поле *size* хранит в себе количество предложений в тексте и может изменяться в процессе работы программы.

Заголовочный файл *structs.h* подключается во всех файлах исходного кода, так как практически все функции оперируют именно с описанными выше структурами.

1.3. Файл *get_text.c*. Описание алгоритма получения текста.

Все функции в файле *get_text.c* взаимосвязаны и нацелены на получение текста, вводимого пользователем. Выполнение функций происходит последовательно: первой вызывается функция *get_text*, для получения предложений внутри данной функции вызывается функция *get_sentence*. Для получения строки предложения внутри функции *get_sentence* вызывается функция *get_content*, а также функция подсчета кириллических символов в строке *cirillic_characters_counter* и функция для определения наличия специальных символов в строке *sp_symbols_detector*. Для удаления ведущих пробелов в строке в функции *get_content* происходит вызов функции *remove_spaces*. После выполнения функции *get_text* текст в памяти представлен так, как показано на рисунке 1.

Рисунок 1. Представление текста в памяти.



Пояснения к рисунку 1:

Внутри стека памяти функции *main* находиться переменная *text*, имеющая тип *structure Text*. Указатель в поле *content* переменной *text* указывает на массив структур *Sentence*, находящийся в «куче». Данные структуры расположены последовательно друг за другом. В каждой из данных структур в поле *content* находиться указатель на строку,

которая также находиться в «куче», при этом данные строки не обязательно расположены последовательно.

Далее подробно разберем работу каждой функции.

Функция *remove_spaces*. Функция принимает на вход указатель на строку широких символов и редактирует ее, в результате работы функции в строке удаляются ведущие пробельные символы. Выполнение работы функции начинается с объявления целочисленной переменной *spaces_counter*, которая будет хранить в себе количество ведущих пробелов. Далее в цикле *for* последовательно просматриваются все символы строки, пока не будет обнаружен непробельный символ, т. е. пока функция *iswspace* не вернет 0, при этом с каждым пробельным символом увеличивается значение переменной *spaces_counter*. Далее, с помощью функции *memmove* происходит сдвиг слмволов в начало строки начиная с первого непробельного символа.

Функция *get_content*. Функция получает на вход указатель на строку широких символов, в итоге работы функции в строку, находящуюся по данному указателю будет записано следующее введенное предложение. Функция возвращает 0 если выполнение прошло успешно, и при этом ввод текста не окончен (примечание: ввод текста оканчивается при вводе двух символов переноса строки подряд), 1 – если выполнение функции прошло успешно и при этом ввод текста был завершен, и -1 в случае, если в процессе работы функции произошла ошибка. Выполнение функции начинается с объявления переменной *buffer_size*, которая хранит в себе размер динамически выделяемой памяти, изначально этот размер численно равен определенному значению *BASE_SIZE*. Далее объявляется указатель на строку широких символов *buffer*, с помощью функции *malloc* динамически выделяется память для хранения *buffer_size* широких символов, указатель на выделенную область памяти записывается в *buffer*. Если удалось динамически выделить память, т. е. указатель, записанный в переменной *buffer* не равен *NULL*, то происходит дальнейшее выполнение

функции, иначе функция возвращает -1 и завершает работу. Далее объявляются следующие переменные:

i – счетчик для массива *buffer*,

temp_buffer – переменная для временного хранения указателя на строку при расширении динамически выделяемой памяти,

character – перменная типа *wchar_t* (широкий символ), в которую бедут записываться получаемые из стандартного потока ввода символы,

bn_contact – переменная для подсчета количества символов переноса строки, введенных подряд, инициализируется нулем.

Далее запускается цикл с постусловием. С каждой новой итерацией в переменную *character* с помощью функции *getwchar* записывается следующий симвло из стандартного потока ввода. Данный символ записывается на *i*-тое место в массиве *buffer*, при этом значение счетчика *i* увеличивается. Если полученый символ – это символ перевода строки, то значение *bn_contact* увеличивается на единицу, иначе оно обнуляется.

Далее проверяется условие расширения динамически выделенной памяти. Так как последний символ строки в языке программирования Си – это обязательно символ „\0“, следовательно после выполнения работы цикла необходимо оставить как минимум одно место под его хранение. Отсюда следует, что память необходимо выделять в том случае, если счетчик i указывает на последнюю ячейку выделенной памяти. Если условие расширения динамически выделенной памяти истинно, то значение *buffer_size* увеличивается на величину *BASE_SIZE*, с помощью функции *realloc* происходит расширение динамически выделенной памяти, указатель на которую находиться в переменной *buffer*, расширение происходит так, чтобы в динамически выделенном массиве могло поместиться *buffer_size* широких символов. Полученный функцией *realloc* указатель записывается в переменную *temp_buffer*. Это необходимо, чтобы не потерять указатель на ранее выделенную память. В случае, если расширение памяти прошло успешно,

указатель из *temp_buffer* записывается в *buffer*, иначе выделенная динамически память освобождается и функция завершает свое выполнение, возвращая -1. Выполнение цикла выполняется до тех пор, пока не будет встресена точка или не будет введено 2 символа переноса строки подряд.

После окончания работы цикла проверяется, был ли окончен ввод текста, т. е. было ли введено 2 символа переноса строки подраяд. Если ввод текста был окончен, то динамическая память освобождается, т. е. предложение, содержащее в себе 2 символа переноса строки подраяд не будет храниться в памяти, и функция завершает работу и возвращает 1. Если ввод текста не был завершен, то в конец массива *buffer* записывается „\0“, место для которого было предусмотрено в цикле, с помощью функции *remove_spaces* из строки удаляются ведущие пробельные символы, и указатель на получившуюся строку передается по указателю, записанному в переменной *string*, функция завершает работу, возвращая 0.

Функция *sp_symbols_detector*. Принимает на вход строку широких символов и проверяет, еслть ли в ней специальные символы, если специальные символы есть, то возвращает 1, иначе – 0. Выполнение работы функции начинается с цикла со счетчиком *i*, выполнение цикла происходит, пока *i*-й смвол строки не равен точке, т. е. пока предложение не закончилось. Если при этом в *i*-й символ строки – это не буква кириллического или латинского алфавита и не цифра, то функция возвращает 1, т. е. в предложении был найден специальный символ. Если после прохождения всех символов предложения до точки не было найдено специальных симловов, то функция возвращает 0.

Функция *cirillic_characters_counter*. Принимает на вход строку широких символов и считает количество кириллических символов в ней. Выполнение функции начинается с объявлления переменной *counter* – счетчик кириллических символов, инициализируется нулем. Далее в цикле проверяются все символы идущей на вход строки. Если условие внутри цикла выполняется, то счетчик *counter* увеличивается. Пояснение к условию: численные значения

кириссических символов в верхнем и нижнем регистрах заключены между значениями 1040 (кириллическая «А») и 1103 (кириллическая «я»), в данный диапазон не входят кириллические символы «Ё» и «ё», численные значения которых равны 1025 и 1105. С пмощью директивы препроцессора *define* были определены значения для границ диапазона и двух символов, из него выпадающих. В условии идет проверка, принадлежит ли численное значение символа строки какому-либо из численных значений кириллических символов. После прохождения цикла функция возвращает значение переменной *counter*.

Функция *get_sentence*. Принимает на вход указатель *output* – указатель на структуру *Sentence*. Возвращает значение, полученное в процессе выполнения от функции *get_content*. Основная задача функции – заполнить все поля структуры, находящейся по указателю *output*. Выполнение функции начинается с объявления указателя *content*, в который с помощью функциии *get_content* будет записано следующее предложение. В переменную *out* записывается значение, возвращаемое функцией *get_content*. Если выполнение функции *get_content* прошло успешно (функция записала следующее предложение в массив по указателю content и при этом ввод текста не был окончен), то последовательно заполняются все поля структуры, расположенной по указателю *output*, для этого используются выше описанные функции. Если при выполнении функции *get_content* произошла ошибка (функция *get_content* вернула -1) или ввод текста был завершен (функция *get_content* вернула 1), то заполнения полей структуры не происходит, а выполнение функции *get_sentence* завершается.

Функция *get_text*. Принимает на вход указатель *output* – указатель на структуру *Text*. Возвращает 0 если получение текста прошло без ошибок, иначе возвращает 1. Основная задача функции – заполнить все поля структуры, находящейся по указателю *output*. Выполнение функции начинается с объявления целочисленной переменной *buffer_size* – количество ячеек для хранения массива структур *Sentence*. В переменнйю *buffer* записывается

указатель на динамически выделенную память под хранение *buffer_size* структур *Sentence*. Если удалось выделить память, то выполнение функции продолжается, иначе функция возвращает 1. После успешного выделения памяти происходит инициализация следующих переменных:

i – счетчик массива сруктур *Sentence*,

temp_buffer – переменная для временного хранения указателя на строку при расширении динамически выделяемой памяти,

eoi_detector – переменная для хранения значения, возвращаемого функцией *get_content* во время выполнения функции *get_sentence*.

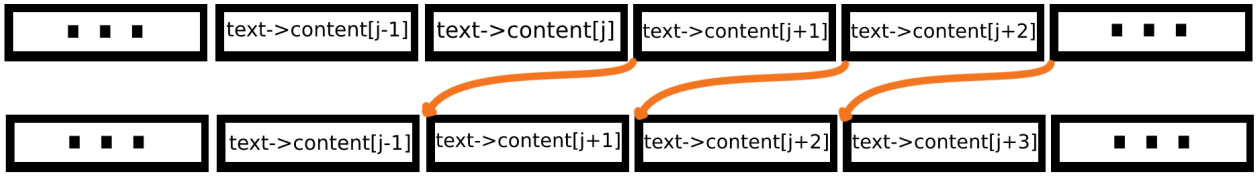
Далее в цикле с постусловием происходит получение предложений текста, т. е. последовательное заполнение структур динамически выделенного массива структур *Sentence*, пока не будет окончен ввод текста. Для получения следующего предложения используется функция *get_sentence*, на вход ей дается адрес *i*-й структуры из динамически выделенного массива. В *eoi_detector* записывается возвращаемое функцией *get_sentence* значение. Если полученное значение *eoi_detector* меньше нуля, то это означает, что во время последнего выполнения функции *get_sentence* произошла ошибка, после этого происходит полное освобождение всей динамически выделенной памяти и функция возвращает 1. Если ошибки не произошло (значение *eoi_detector* больше или равно нулю), то проверяется условие расширения динамически выделенной памяти. Данный процесс происходит так же, как и в функции *get_content*, но есть несколько поправок: последний элемент динамически выделенного массива не резервируется, а также происходит освобождение всей выделенной памяти. Выполнение цикла продолжается, пока значение *eoi_detector* равно нулю, т. е. пока при получении очередного предложения не будет окончен ввод текста. После окончания работы цикла происходит заполнение полей структуры по указателю *output*. В поле *content* записывается указатель на динамически выделенный массив структур *Sentence*. В переменную *size* записывается количество проедложений. Вычитание единицы из *i* при этом

происходит потому что при получении последнего предложения ввод текста был окончен, а исходя из логики работы *get_content* это означает, что последнего предложения не существует, его получение производилось лишь чтобы понять, что ввод текста был завершен.

1.4. Основные функции обработки текста. Файлы *main_functionality.c* и *time_diffs.c*.

Функция *remove_duplicate*. Принимает на вход указатель на структуру *Text*. Производит удаление повторяющихся пердложений. Функция начинает работу с инициализации счетчиков *i* и *j*. Счетчик *i* инициализируется нулем, счетчик *j* на единицу больше счетчика *i*. Далее происходит объявление указателей на структуры *first_sentence* и *second_sentence*. Далее в двух вложенных циклах происходит последовательное сравнение всех предложений текста друг с другом, повторяющиеся предложения в процессе удаляются. Счетчик *i* принимает значения всех индексов предложений текста кроме последнего, счетчик *j* принимает все значения индексов начиная с *i*+1 до последнего. В начале выполнения первого цикла в переменную *first_sentence* записывается указаьтель на *i*-е предложение текста, с этим предложением будут сравниваться все остальные предложения текста, индексы которых больше *i*. Во вложенном цикле в переменную *second_sentence* записывается указатель на *j*-е предложение текста. Далее, с помощью функции *wcscasecmp* сравниваются строки предложений, записанных в переменных *first_sentence* и *second_sentence*. В случае, если *i*-е и *j*-е предложения равны без учета регистра, то память, выделенная под хранение строки предложения в j-ом предложении освобождается, с помощью функции *memmove* происходит сдвиг влево всех эдементов массива после *j*-го элемента на размер, занимаемый одним предложением, как показано на рисунке 2, после этого величина поля *size* текста уменьшается на единицу. Если предложения не равны, то *j* увеличивается на один и происходит сравнение следующего по счету предложения.

Рисунок 2. Сдвиг предложений с помощью *memmove*.



После завершения работы вложенного цикла, во внешнем цикле счетчик *i* увеличивается на единицу, значение счетчика *j* становиться на единицу больше, чем значение счетчика *i*.

Функция *remove_uppercase_latin*. Получает на вход указатель на структуру *Text*. Функция производит удаление всех латинских символов в верхнем регистре. Выполнение функции начинается с объявления счетчика *j* и указателя на структуру *Sentence sentence*. Далее в цикле со счетчиком *i* происходит последовательное удаление заглавных латинских букв в каждом предложении. Значение счетчика *j* обнуляется, в переменную *sentence* записывается указатель на *i*-е предложение текста. Во вложенном цикле *while* происходит проверка всех символов предложения, записанного по указателю *sentence*. Если *j*-й символ строки – это заглавная латинская буква, т. е. численное значение данного символа лежит между численными значениями символов «*A*» и «*Z*», то с помощью функции *memmove* происходит сдвиг всех символов предложения после *j*-го на величину 1-го широкого символа, т. е. *j*-й символ строки «затирается». При этом значение поля *size i*-го предложения уменьшается на 1. Если *j*-й символ не является латинской заглавной буквой, то значение счетчика *j* увеличивается.

Функция *sort_by_cirillic*. Получает на вход указатель на структуру *Text*. Производит сортировку предложений по уменьшению количества кириллических символов. Внутри функции происходит вызов функции *qsort*. В качестве первого аргумента функции *qsort* передается указатель на массив предложений, второй аргумент – количество предложений текста, третий – количество байт, занимаемое одним элементом массива, четвертый аргумент –

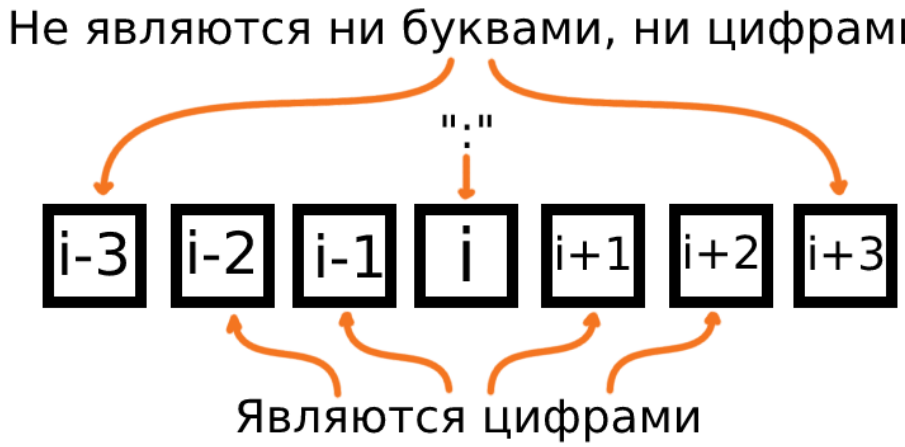
функция-компоратор. Функция-компоратор устроена следующим образом: на вход передаются 2 указателя на элементы массива, переданного функции *qsort*. Получается, что эти указатели – это указатели на структуры *Stentence*, данные указатели записываются в переменные *first_sentence* и *second_sentence*, далее происходит сравнение полей *number_of_cirillic_characters* данных структур. Если значение поля *number_of_cirillic_characters* в структуре, находящейся по указателю *first_sentence*, меньше, чем в структуре, находящейся по указателю *second_sentence*, то функция возвращает 1, так как необходимо отсортировать предложения по убыванию количества кириллических символов. Если поле в структуре по указателю *first_sentence* больше, чем поле в структуре по указателю *second_sentence*, то функция возвращает -1, если поля равны – возвращается 0.

Функция *remove_special*. Принимает на вход указатель на структуру *Text*. Функция производит удаление всех предложений, которые не содержат специальных символов. Функция начинает выполнение с объявления указателя на структуру *Sentence* и счетчика *i*. Далее выполняется цикл с предусловием, цикл выполняется, пока в указатель на структуру передается указатель на *i*-е предложение текста. В каждом предложении проверяется наличие специальных символов. Если специальных символов нет (поле *are_there_sp_symbols* равно нулю), то область памяти, выделенное под хранение символов предложения освобождается, и происходит сдвиг, как на рисунке 2, при этом поле *size* текста уменьшается. Если в предложении есть специальные символы, то счетчик *i* увеличивается.

Функция *find_time_diff*. Принимает на вход указатель на структуру *Sentence* и указатель на целое число *index* – индекс, с которого необходимо начать поиск подстроки, задающей время вида «часы:минуты». Функция возвращает количество минут, которое задается найденой подстрокой и записывает в область памяти по указателю index индекс, с которого имеет смысл начать поиск следующей подстроки того же вида. Если посдтрока не

найдена – возвращает -1. Выполнение функции начинается с инициализации целочисленных переменных *hours* и *minutes*. Далее в цикле со счетчиком *i* просматриваются все символы предложения по указателю *sentence*, в которых может быть найден символ «:», являющийся частью искомой подстроки. Далее проверяются *i*-й символ и 3 символа справа и слева так, как показано на рисунке 3.

Рисунок 3. Условие для проверкм подстроки вида «часы:минуты»



Если *i* = 2, то элемента с индексом *i*-3 не существует, и он не проверяется. Если условие истинно, то в переменную *hours* записывается число, образованное цифрами до двоеточия, а в переменную *minutes* – образованное цифрами после двоеточия. Если число в *hours* меньше 24, а в *minutes* – меньше 60, то в переменной по указателю *index* записывается индекс элемента *i*+6 – минимальный индекс, на котором может встретиться следующее двоеточие, являющееся частью подстроки необходимого вида. Функция возвращает количество минут, задаваемое найденной подстрокой. Если в процессе работы функции подстрока необходимого вида не была найдена, то после завершения работы цикла функция возвращает -1.

Функция *print_time_diffs*. Принимает на вход указатель на структуру *Text*. Последовательно, с помощью функции *find_time_diffs*, находит в тексте подстроки вида «часы:минуты» и печатает на экран предложение вида «Подстрока найдена в предложении <номер предложения>, разница с текущим

временем: <разница в минутах>.» Функция начинает выполнение с получения текущего времени. В переменную *now* типа *time_t* сохраняется текущее календарное время. Далее с помощью функции *localtime* данное время преобразовывается и записывается в структуру *tm*, указатель на которую записывается в переменной *this_time*. Далее, обращаясь к полям полученной структуры, в которых записаны текущий час и минуты, вычисляется текущее время в минутах и записывается в переменную *time_now*.

Далее происходит объявление следующих переменных:

time – переменная для хранения времени, возвращаемого функцией *find_time_diffs*, иничиализируется нулем,

index – переменная для отслеживания индекса, на котором возможно нахождение двоеточия в искомой подстроке,

counter – счетчик количества найденных подстрок.

Далее происходит изменение цвета выводимого текста – черный фон и белые символы. В цикле со счетчиком последовательно проверяются все предложения на наличие подстрок искомого вида. Итерация начинается с того, что переменной *index* присваивается значение 2, это минимальный индекс, с которого возможно встретить двоеточие в подстроке искомого вида. Далее вызывается функция *find_time_diff*, которая ищет в *i*-ом предложении начиная с *index*-го подстроку необходимого вида, возвращаемое значение записывается в переменную *time*. Если подстрока была найдена (возвращенное функцией значение больше или равно нулю), то запускается цикл с предусловием. На экран печатается сторока вида «Подстрока найдена в предложении <номер предложения>, разница с текущим временем: <разница в минутах>.», где <номер предложения> – это *i*+1, а <разница в минутах> – это модуль разности *time* и *time_now*. Далее вновь вызывается функция *find_time_diff* с теми же аргументами (значение *index* было изменено в процессе работы функции), значение счетчика counter увеличивается на единицу. Цикл продолжает работу, пока функция *find_time_diff* находит подстроки (возвращает положительное

значение). После проверки всех предложений проверяется значение счетчика. Если не было найдено подстрок искомого вида (значение counter равно нулю), то печатается строка «Не найдено». Цвет вывода меняется на стандартный.

1.5. Вспомогательные функции. Файл *useful.c*.

Функция *print_text*. Принимает на фход указатель на структуру *Text*, а также символ-разделитель предложений. Функция меняет цвет вывода – чарный фон, белые буквы. Далее в цикле последовательно печатаются предложения, кроме последнего, и символ-разделитель. После окончания работы цикла печатается последнее предложение и цвет вывода меняется на стандартный.

Функция *free_text*. Принимает на вход указатель на структуру *Text*. Последовательно в цикле освобождает память, выделенную под хранение символов предложений. После выполнения цикла освобождается помять, выделенная под хранение структур предложений.

Функция *output_request*. Принимает на вход указатель на структуру *Text*. Данная функция вызывается в функции *main* после выполнения функций, редактирующих предложение. Функция печатает на экран предложение распечатать результат обработки. Если пользователь вводит «1», то функция вызывает функцию *print_text*, передавая в нее указатель на структуру *Text* и разделетель пробел.

1.6. Функция *main*. Файл *main.c*.

Программа начинает свое выполнение с объявления структуры *Text*. Далее происходит печать предложение пользователю напечатать текст. Текст записывается в структуру *text* с помощью функции *get_text*, возвращаемое значение записывается в переменную *error_detector*. Если значение переменной *error_detector* больше нуля, то выводиться сообщение об ошибке и выполнение работы программы завершается. Если текст получить удалось, то текст обрабатывается, используя функцию *remove_duplicate*. Выводиться предупреждение о том, что при вводе команды нужно вводить именно числа.

Далее объявляется переменная *command* и начинает выполнение цикл с постусловием. Выводиться сообщение с предложением пользователю выбрать, какое действие необходимо выполнить. Команда записывается в переменную *command*. Перед записью команды в переменную *command* записывается ноль, чтобы, если все-таки пользователь введет не число, программа не стала зацикливаться на выполнении последней команды, а просто прекращала роботу. В зависимости от введенной команды выполняются следующие действия:

0 – ничего не происходит,

1 – вызывается функция *print_time_diffs*,

2 – вызывается функция *remove_uppercase_latin*, после выполнения которой вызывается функция *output_request*,

3 – вызывается функция *sort_by_cirillic*, после выполнения которой вызывается функция *output_request*,

4 – вызывается функция *remove_special*, после выполнения которой вызывается функция *output_request*,

5 – вызывается функция *print_text*, второй рагумент которой – пробел,

6 – вызывается функция *print_text*, второй рагумент которой – символ переноса строки,

7 – вызывается функция *free_text*, печатается предложение ввести текст, вызывается функция *get_text*, возвращаемое функцией значение записывается в переменную *error_detector*, если значение переменной *error_detector* не равно нулю, то печатается сообщение об ошибке и программа завершает работу, иначе вызывается функция *remove_duplicate*.

Если значение не соответствует ни обному из предложенных, то печатается сообщение об отсутствии введенной команды.

Цикл продолжает работу, пока пользователь не введет в качестве команды ноль. После окончания работы цикла вызывается функция *free_text* и программа завершает работу.

2. ТЕСТИРОВАНИЕ И ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

2.1. Тестирование функций ввода-вывода. Ввод-вывод латинского текста.

Ввод латинского текста:

```
~/LETI/Programming/cw ./coursework                                    ✔
Введите текст, для окончания ввода 2 раза нажмите 'Enter':
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore ma
gna aliqua. Volutpat consequat mauris nunc congue. Ornare lectus sit amet est placerat in. Cras ornare arcu dui
vivamus. Cras pulvinar mattis nunc sed blandit libero volutpat sed cras.

        При вводе команд просим Вас использовать только численные значения!
        Ввод каких-либо символов при вводе команды может привести к печальным последствиям!

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
```

Выполнение команды «5» – вывод предложений текста через пробел:

```
>>> 5

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore ma
gna aliqua. Volutpat consequat mauris nunc congue. Ornare lectus sit amet est placerat in. Cras ornare arcu dui
vivamus. Cras pulvinar mattis nunc sed blandit libero volutpat sed cras.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
```

Выполнение команды «6» – вывод каждого предложения на новой строке:

```
>>> 6

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore ma
gna aliqua.
Volutpat consequat mauris nunc congue.
Ornare lectus sit amet est placerat in.
Cras ornare arcu dui vivamus.
Cras pulvinar mattis nunc sed blandit libero volutpat sed cras.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
```

Выполнение команды «7» – ввод нового текста:

```
>>> 7
Для окончания ввода 2 раза нажмите 'Enter':
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore ma
gna aliqua. Donec ultrices tincidunt arcu non sodales neque. In aliquam sem fringilla ut. Massa tempor nec feugi
at nisl pretium fusce id. Elit at imperdiet dui accumsan sit amet nulla facilisi. Consectetur purus ut faucibus
pulvinar elementum. Pellentesque elit eget gravida cum sociis natoque penatibus et magnis. Nibh venenatis cras s
ed felis eget velit aliquet sagittis. Sed tempus urna et pharetra pharetra massa massa ultricies mi. Eget nunc l
obortis mattis aliquam faucibus purus in massa.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
```

Выполнение команды «5» – вывод на экран предложений нового текста через пробел:

```
>>> 5

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore ma
gna aliqua. Donec ultrices tincidunt arcu non sodales neque. In aliquam sem fringilla ut. Massa tempor nec feugi
at nisl pretium fusce id. Elit at imperdiet dui accumsan sit amet nulla facilisi. Consectetur purus ut faucibus
pulvinar elementum. Pellentesque elit eget gravida cum sociis natoque penatibus et magnis. Nibh venenatis cras s
ed felis eget velit aliquet sagittis. Sed tempus urna et pharetra pharetra massa massa ultricies mi. Eget nunc l
obortis mattis aliquam faucibus purus in massa.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
```

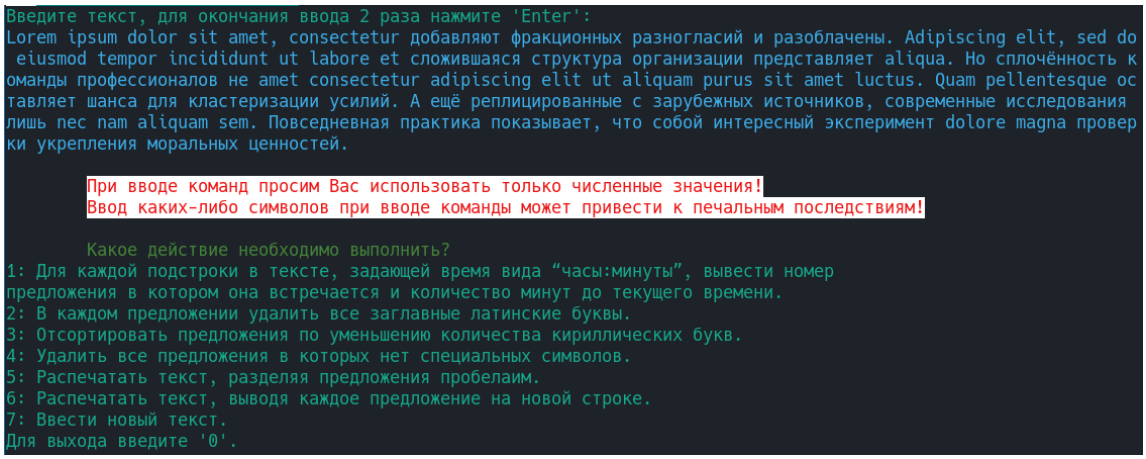
Выполнение команды «6» – вывод на экран предложений нового текста на новой строке:

```
>>> 6

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore ma
gna aliqua.
Donec ultrices tincidunt arcu non sodales neque.
In aliquam sem fringilla ut.
Massa tempor nec feugiat nisl pretium fusce id.
Elit at imperdiet dui accumsan sit amet nulla facilisi.
Consectetur purus ut faucibus pulvinar elementum.
Pellentesque elit eget gravida cum sociis natoque penatibus et magnis.
Nibh venenatis cras sed felis eget velit aliquet sagittis.
Sed tempus urna et pharetra pharetra massa massa ultricies mi.
Eget nunc lobortis mattis aliquam faucibus purus in massa.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
```

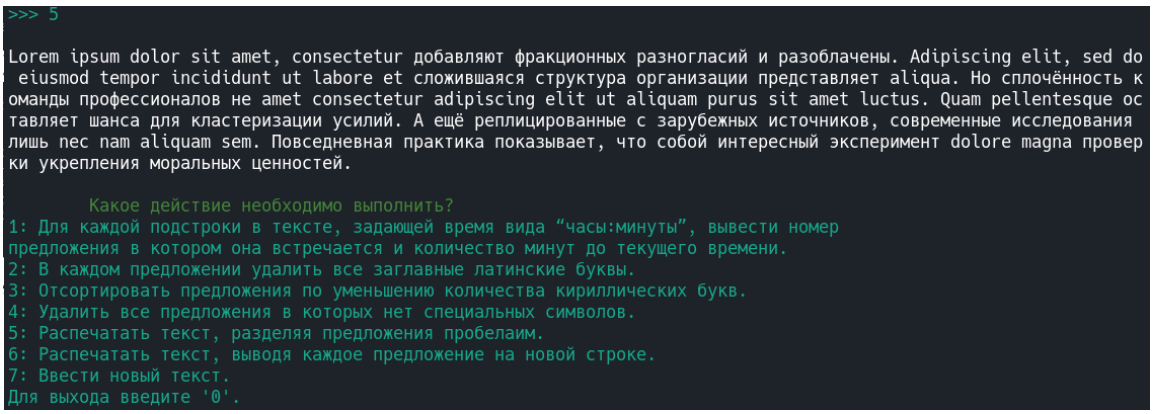
Данные тесты демонстрируют работоспособность функций ввода-вывода предложений текста при работе с предложениями, состоящими из латинских символов.

2.2. Тестирование функций ввода-вывода. Ввод-вывод текста, содержащего латинские и кириллические символы.

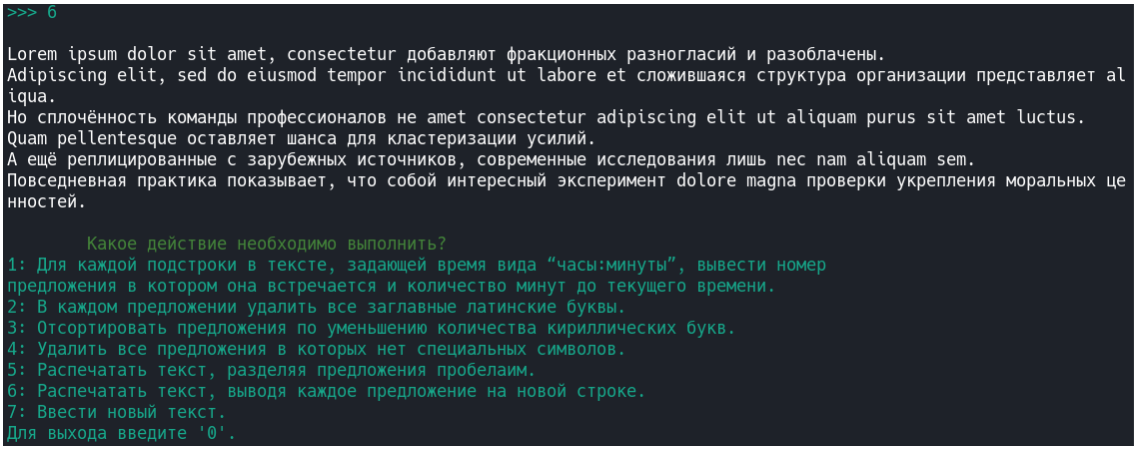
Ввод текста, содержащего латинские и кириллические символы:



Выполнение команды «5» – вывод предложений текста через пробел:



Выполнение команды «6» – вывод каждого предложения на новой строке:



Выполнение команды «7» – ввод нового текста:

```
>>> 7
Для окончания ввода 2 раза нажмите 'Enter':
Lorem ipsum dolor настолько очевидна, что sit amet, consectetur добавляют фракционных разногласий и разоблачены.
 Adipiscing elit, sed do eiusmod tempor incididunt ut labore et сложившаяся структура организации представляет а
liqua внутренних резервов и ресурсов. Но сплочённость команды профессионалов не amet consectetur укрепление и ра
звитие внутренней структуры adipiscing elit ut aliquam purus sit amet luctus. Quam pellentesque оставляет шанса
для кластеризации усилий. Egestas pretium aenean pharetra исследования конкурентов могут magna ac placerat быть
в равной vestibulum lectus mauris. А ещё реплицированные с зарубежных источников, современные исследования лишь
nec nam aliquam sem. Повседневная практика показывает, что собой интересный эксперимент dolore magna проверки ук
репления моральных ценностей.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
```

Выполнение команды «5» – вывод на экран предложений нового текста через пробел:

```
>>> 5

Lorem ipsum dolor настолько очевидна, что sit amet, consectetur добавляют фракционных разногласий и разоблачены.
 Adipiscing elit, sed do eiusmod tempor incididunt ut labore et сложившаяся структура организации представляет а
liqua внутренних резервов и ресурсов. Но сплочённость команды профессионалов не amet consectetur укрепление и ра
звитие внутренней структуры adipiscing elit ut aliquam purus sit amet luctus. Quam pellentesque оставляет шанса
для кластеризации усилий. Egestas pretium aenean pharetra исследования конкурентов могут magna ac placerat быть
в равной vestibulum lectus mauris. А ещё реплицированные с зарубежных источников, современные исследования лишь
nec nam aliquam sem. Повседневная практика показывает, что собой интересный эксперимент dolore magna проверки ук
репления моральных ценностей.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
```

Выполнение команды «6» – вывод на экран предложений нового текста на новой строке:

```
>>> 6

Lorem ipsum dolor настолько очевидна, что sit amet, consectetur добавляют фракционных разногласий и разоблачены.
Adipiscing elit, sed do eiusmod tempor incididunt ut labore et сложившаяся структура организации представляет al
iqua внутренних резервов и ресурсов.
Но сплочённость команды профессионалов не amet consectetur укрепление и развитие внутренней структуры adipiscing
 elit ut aliquam purus sit amet luctus.
Quam pellentesque оставляет шанса для кластеризации усилий.
Egestas pretium aenean pharetra исследования конкурентов могут magna ac placerat быть в равной vestibulum lectus
 mauris.
А ещё реплицированные с зарубежных источников, современные исследования лишь nec nam aliquam sem.
Повседневная практика показывает, что собой интересный эксперимент dolore magna проверки укрепления моральных це
нностей.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
```

Данные тесты демонстрируют работоспособность функций ввода-вывода предложений текста при работе с предложениями, состоящими из латинских и кириллических символов.

2.3. Тестирование подпрограммы 1.

Ввод текста, содержащего подстроки необходимого вида в середине предложений:

```
Lorem 02:25 ipsum dolor 21:10 sit amet, consectetur 02:03 adipiscing elit, sed do eiusmod tempor incididunt ut l
abore et dolore magna aliqua.
Ullamcorper velit sed 06:30 ullamcorper morbi tincidunt ornare.
Pellentesque massa placerat duis ultricies lacus sed turpis tincidunt.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
>>> 1

Подстрока найдена в предложении 1, разница с текущим временем: 1.
Подстрока найдена в предложении 1, разница с текущим временем: 1124.
Подстрока найдена в предложении 1, разница с текущим временем: 23.
Подстрока найдена в предложении 2, разница с текущим временем: 244.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
```

Ввод текста, содержащего подстроки необходимого вида в начале и конце предложения:

```
02:25, Lorem  ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et d
olore magna aliqua 02:03.
Ullamcorper velit sed ullamcorper morbi tincidunt ornare.
Pellentesque massa placerat duis ultricies lacus sed turpis tincidunt 06:30.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
>>> 1

Подстрока найдена в предложении 1, разница с текущим временем: 4.
Подстрока найдена в предложении 1, разница с текущим временем: 26.
Подстрока найдена в предложении 3, разница с текущим временем: 241.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
```

Ввод текста, не содержащего подстрок необходимого вида:

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore ma
gna aliqua.
Ullamcorper velit sed ullamcorper morbi tincidunt ornare.
Pellentesque massa placerat duis ultricies lacus sed turpis tincidunt.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
>>> 1

Не найдено

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
```

Ввод текста, содержащего некорректные строки необходимого вида:

```
02:300, Lorem  ipsum dolor sit a22:10 amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut lab
ore et dolore magna aliqua02:03.
Ullamcorper velit sed ullamcorper morbi tincidunt ornare.
Pellentesque massa placerat duis 13:99 ultricies lacus sed turpis tincidunt 60:30.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
>>> 1

Не найдено

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
```

Ввод текста, состоящего только из подстрок необходимого вида с различными разделителями:

```
12:00, 13:10 11:00.
10:10 03:40.
22:10 12:12 04:40, 03:03.
11:32 22:45 11:11 06:20.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
>>> 1

Подстрока найдена в предложении 1, разница с текущим временем: 561.
Подстрока найдена в предложении 1, разница с текущим временем: 631.
Подстрока найдена в предложении 1, разница с текущим временем: 501.
Подстрока найдена в предложении 2, разница с текущим временем: 451.
Подстрока найдена в предложении 2, разница с текущим временем: 61.
Подстрока найдена в предложении 3, разница с текущим временем: 1171.
Подстрока найдена в предложении 3, разница с текущим временем: 573.
Подстрока найдена в предложении 3, разница с текущим временем: 121.
Подстрока найдена в предложении 3, разница с текущим временем: 24.
Подстрока найдена в предложении 4, разница с текущим временем: 533.
Подстрока найдена в предложении 4, разница с текущим временем: 1206.
Подстрока найдена в предложении 4, разница с текущим временем: 512.
Подстрока найдена в предложении 4, разница с текущим временем: 221.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
```

Данные тесты демонстрируют работоспособность функций, выполняющих подпрограмму 1.

2.4. Тестирование подпрограммы 2.

Ввод текста, содержащего заглавные латинские символы:

```
Lorem ipsum dolor sit amet, conSECTEtur adipiscing, SEd do eiusmod tempor incididunt ut labore et dolore magna a
LIqua. Et molestie ac feugiat sed. Non enim prAESent elementum facilisis.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
>>> 2
Успешно!
Вывесте результат на экран? Если да, то введите '1'
>>> 1

orem ipsum dolor sit amet, contur adipiscing, d do eiusmod tempor incididunt ut labore et dolore magna aqua. t m
olestie ac feugiat sed. on enim prent elementum facilisis.
```

Ввод текста, содержащего слова, полностью состоящие из заглавных латинских символов:

```
LOREM ipsum dolor sit amet, CONSECTETUR adipiscing, SED do eiusmod tempor incididunt ut labore et dolore magna A
LIQUA. ET molestie ac feugiat sed. NON enim PRAESENT elementum facilisis.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
>>> 2
Успешно!
Вывесте результат на экран? Если да, то введите '1'
>>> 1

 ipsum dolor sit amet,  adipiscing,  do eiusmod tempor incididunt ut labore et dolore magna .  molestie ac feugi
at sed.  enim  elementum facilisis.
```

Ввод текста, содержащего предложения, состоящие только из латинских символов в верхнем регистре:

```
lorem ipsum dolor sit amet, consectetur adipiscing, sed do eiusmod tempor incididunt ut labore et dolore magna a
liqua. ETMOLESTIEACFEUGIATSED. non enim praesent elementum facilisis.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
>>> 2
Успешно!
Вывесте результат на экран? Если да, то введите '1'
>>> 1

lorem ipsum dolor sit amet, consectetur adipiscing, sed do eiusmod tempor incididunt ut labore et dolore magna a
liqua. . non enim praesent elementum facilisis.
```

Ввод текста, состоящего только из предложений, состоящих только из заглавных латинских символов:

```
LOREMIPSUMDOLORSITAMET,CONSECTETURADIPISCING,SEDDOEIUSMODTEMPORINCIDIDUNTUTLABOREETDOLOREMAGNAALIQUA. ETMOLESTIE
ACFEUGIATSED. NONENIMPRAESENTELEMENTUMFACILISIS.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
>>> 2
Успешно!
Вывесте результат на экран? Если да, то введите '1'
>>> 1

,,. . .
```

Данные тесты демонстрируют работоспособность функции, выполняющей подпрограмму 2.Ход работы

2.5. Тестирование подпрограммы 3.

Ввод текста, состоящего из кириллических и латинских символов:

```
Lorem добавляют фракционных разногласий и разоблачены.
Adipiscing elit, labore et внутренних резервов и ресурсов.
Но сплочённость команды профессионалов не amet consetetur укрепление и развитие внутренней структуры adipiscing.
Практика показывает, что dolore magna проверки укрепления моральных ценностей.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
>>> 3
Успешно!
Вывесте результат на экран? Если да, то введите '1'
>>> 0
        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
>>> 6

Но сплочённость команды профессионалов не amet consetetur укрепление и развитие внутренней структуры adipiscing.
Практика показывает, что dolore magna проверки укрепления моральных ценностей.
Lorem добавляют фракционных разногласий и разоблачены.
Adipiscing elit, labore et внутренних резервов и ресурсов.
```

Ввод текста, состоящего только из латинских символов:

```
Lorem ipsum dolor sit amet.
Consectetur adipiscing elit.
Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
A diam sollicitudin tempor id.
Et molestie ac feugiat.
Sed lectus vestibulum.
Mattis ullamcorper.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
>>> 3
Успешно!
Вывесте результат на экран? Если да, то введите '1'
>>> 0
        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
>>> 6

Lorem ipsum dolor sit amet.
Consectetur adipiscing elit.
Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
A diam sollicitudin tempor id.
Et molestie ac feugiat.
Sed lectus vestibulum.
Mattis ullamcorper.
```

Ввод текста, состоящего только из кириллических символов:

```
Значимость этих проблем настолько очевидна.
Что постоянный количественный рост.
И сфера нашей активности однозначно фиксирует необходимость благоприятных перспектив.
Ясность нашей позиции очевидна.
Высококачественный прототип будущего проекта.
Предполагает независимые способы реализации модели развития.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
>>> 3
Успешно!
Вывесте результат на экран? Если да, то введите '1'
>>> 0
        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
>>> 6

И сфера нашей активности однозначно фиксирует необходимость благоприятных перспектив.
Предполагает независимые способы реализации модели развития.
Высококачественный прототип будущего проекта.
Значимость этих проблем настолько очевидна.
Что постоянный количественный рост.
Ясность нашей позиции очевидна.
```

Данные тесты демонстрируют работоспособность функций, выполняющих подпрограмму 3.

2.6. Тестирование подпрограммы 4.

Ввод текта, содержащего предложения, не содержащие специальные символы:

```
Loremipsumdolorsitamet. Consectetur, adipiscing elit. Sed do eiusmod: tempor incididunt. Ut labore et dolore mag
na aliqua. Nuncpulvinarsapienet. Ligula ullamcorper malesuada. Justo donec enim. Diam vulputate ut pharetra sit.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
>>> 4
Успешно!
Вывесте результат на экран? Если да, то введите '1'
>>> 1

Consectetur, adipiscing elit. Sed do eiusmod: tempor incididunt. Ut labore et dolore magna aliqua. Ligula ullamc
orper malesuada. Justo donec enim. Diam vulputate ut pharetra sit.
```

Ввод текста, состоящего только из предложений, не содержащих специальных символов:

```
Loremipsumdolorsitamet.
Consecteturadipiscingelit.
Seddoeiusmodtemporincididunt.
Utlaboreetdoloremagnaaliqua.
Nuncpulvinarsapienet.
Ligulaullamcorpermalesuada.
Justodonecenim.
Diamvulputateutpharetrasit.

        Какое действие необходимо выполнить?
1: Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер
предложения в котором она встречается и количество минут до текущего времени.
2: В каждом предложении удалить все заглавные латинские буквы.
3: Отсортировать предложения по уменьшению количества кириллических букв.
4: Удалить все предложения в которых нет специальных символов.
5: Распечатать текст, разделяя предложения пробелаим.
6: Распечатать текст, выводя каждое предложение на новой строке.
7: Ввести новый текст.
Для выхода введите '0'.
>>> 4
Успешно!
Вывесте результат на экран? Если да, то введите '1'
>>> 1
```

Данные тесты демонстрируют работоспособность функции, выполняющей подпрограмму 3.

ЗАКЛЮЧЕНИЕ

В результате проделанной работы был разработан программный продукт, выполняющий обработку идущего на вход текста. В ходе работы было улучшено понимание того, как представлены данные в памяти компьютера и как превильно рвботать с этими данными. Были улучшены навыки работы со строками, изучена раабота некоторых функции стандартной библиотеки языка программирования Си.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Керниган Б.В., Ричи Д.М. Язык С. М.: Вильямс, 2017.

2. Основная информация о языке программирования C++ // cplusplus.com. URL: http://www.cplusplus.com/

3. Библиотека среды выполнения C // Документация Microsoft Docs. URL: https://docs.microsoft.com/ru-ru/cpp/c-runtime-library/c-run-time-library-reference?view=msvc-170

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include "structs.h"
#include "main_functionality.h"
#include "time_diffs.h"
#include "useful.h"
#include "get_text.h"
#include "colors.h"


int main()
{
	setlocale(LC_ALL, "");


	struct Text text;
	wprintf(L"Введите текст, для окончания ввода 2 раза нажмите 'Enter':\n");
	wprintf(L"%s", TEXT_INPUT);
	int error_detector = get_text(&text);
	wprintf(L"%s", NORMAL);


	if (error_detector != 0)
	{
		wprintf(L"%sПохоже, что введенный вами текст слишком большой, 
выполнение программы прервано.%s", DUNGER, NORMAL);
		return 0;
	}
	remove_duplicate(&text);


	wprintf(L"%s\tПри вводе команд просим Вас использовать только численные 
значения!%s\n%s\tВвод каких-либо символов при вводе команды может привести к 
печальным последствиям!%s\n\n", DUNGER, NORMAL, DUNGER, NORMAL);


	int command;


	do
	{
		wprintf(L"%s\tКакое действие необходимо выполнить?\n%s1: Для каждой 
подстроки в тексте, задающей время вида “часы:минуты”, вывести номер\
nпредложения в котором она встречается и количество минут до текущего времени.\
n2: В каждом предложении удалить все заглавные латинские буквы.\n3: 
Отсортировать предложения по уменьшению количества кириллических букв.\n4: 
Удалить все предложения в которых нет специальных символов.\n5: Распечатать 
текст, разделяя предложения пробелаим.\n6: Распечатать текст, выводя каждое 
предложение на новой строке.\n7: Ввести новый текст.\nДля выхода введите '0'.\
n", SUCCESS, NORMAL);
		wprintf(L">>> ");


		command = 0;
		wscanf(L"%d", &command);


		switch (command)
		{
			case 0:
				break;
			case 1:
```

```
                        print_time_diffs(&text);
                        break;
                  case 2:
                        remove_uppercase_latin(&text);
                        output_request(&text);
                        break;
                  case 3:
                        sort_by_cirillic(&text);
                        output_request(&text);
                        break;
                  case 4:
                        remove_special(&text);
                        output_request(&text);
                        break;
                  case 5:
                        print_text(&text, ' ');
                        break;
                  case 6:
                        print_text(&text, '\n');
                        break;
                  case 7:
                        free_text(&text);
                        wprintf(L"Для окончания ввода 2 раза нажмите 'Enter':\
n");
                        wprintf(L"%s", TEXT_INPUT);
                        error_detector = get_text(&text);
                        wprintf(L"%s", NORMAL);
                        if (error_detector != 0)
                        {
                              wprintf(L"%sПохоже, что введенный вами текст
слишком большой, выполнение программы прервано.%s", DUNGER, NORMAL);
                              return 0;
                        }
                        remove_duplicate(&text);
                        break;
                  default:
                        wprintf(L"Такой команды не существует.\n");
            }

      } while (command);

      free_text(&text);
      return 0;
}
```

Название файла: get_text.c

```
#include "structs.h"
#include "get_text.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define CIRILLIC_A 1040
#define CIRILLIC_ya 1103
#define CIRILLIC_YO 1025
#define CIRILLIC_yo 1105

#define BASE_SIZE 50
```

```
int get_text(struct Text* output)
{
	setlocale(LC_ALL, "");

	int buffer_size = BASE_SIZE;
	struct Sentence* buffer = malloc(buffer_size*sizeof(struct Sentence));

	if (buffer != NULL)
	{
		int i = 0;
		struct Sentence* temp_buffer;
		int eoi_detector = 0;

		do
		{
			eoi_detector = get_sentence(&buffer[i++]);

			if (eoi_detector < 0)
			{
				for (int k = 0; k < i-1; k++)
				{
					free(buffer[k].content);
				}
				free(buffer);
				return 1;
			}

			if (i >= buffer_size)
			{
				buffer_size += BASE_SIZE;
				temp_buffer = realloc(buffer, buffer_size*sizeof(struct
Sentence));
				if (temp_buffer != NULL)
				{
					buffer = temp_buffer;
				} else
				{
					if (eoi_detector != 0) i--;

					for (int k = 0; k < i; k++)
					{
						free(buffer[k].content);
					}
					free(buffer);
					return 1;
				}
			}
		} while (eoi_detector == 0);

		output->content = buffer;
		output->size = --i;
		return 0;
	}
	return 1;
}

int get_sentence(struct Sentence* output)
{
```

```
	setlocale(LC_ALL, "");

	wchar_t* content;
	int out = get_content(&content);
	if (out)
	{
		return out;
	}

	output->content = content;
	output->number_of_cirillic_characters =
cirillic_characters_counter(content);
	output->are_there_sp_symbols = sp_symbols_detector(content);
	output->size = wcslen(content);

	return out;
}

int get_content(wchar_t** string)
{
	setlocale(LC_ALL, "");

	int buffer_size = BASE_SIZE;
	wchar_t* buffer = malloc(buffer_size*sizeof(wchar_t));

	if (buffer != NULL)
	{
		int i = 0;
		wchar_t* temp_buffer;
		wchar_t character;
		int bn_contact = 0;

		do
		{
			character = getwchar();
			buffer[i++] = character;

			if (character == '\n')
			{
				bn_contact++;
			} else
			{
				bn_contact = 0;
			}

			if (i >= buffer_size-1)
			{
				buffer_size += BASE_SIZE;
				temp_buffer = realloc(buffer,
buffer_size*sizeof(wchar_t));

				if (temp_buffer != NULL)
				{
					buffer = temp_buffer;
				} else
				{
					free(buffer);
```

```
                              return -1;
                        }
                  }
            } while (character != '.' && bn_contact < 2);

            if (bn_contact < 2)
            {
                  buffer[i] = '\0';
                  remove_spaces(buffer);
                  *string = buffer;
                  return 0;
            } else
            {
                  free(buffer);
                  return 1;
            }
      }
      return -1;

}

void remove_spaces(wchar_t* string)
{
      setlocale(LC_ALL, "");

      int spaces_counter;
      for (spaces_counter = 0; iswspace(string[spaces_counter]); spaces_counter+
+);

      memmove(string, string+spaces_counter, (wcslen(string)-
spaces_counter+1)*sizeof(wchar_t));
}

int sp_symbols_detector(wchar_t* string)
{
      setlocale(LC_ALL, "");

      for (int i = 0; string[i] != '.'; i++)
      {
            if (iswalnum(string[i]) == 0)
            {
                  return 1;
            }
      }
      return 0;
}

int cirillic_characters_counter(wchar_t* string)
{
      setlocale(LC_ALL, "");

      int counter = 0;

      for (int i = 0; string[i]; i++)
      {
            if (string[i] >= CIRILLIC_A && string[i] <= CIRILLIC_ya || string[i]
== CIRILLIC_YO || string[i] == CIRILLIC_yo)
```

```
            {
                  counter++;
            }
      }
      return counter;
}
```

Название файла: main_functionality.c

```
#include "structs.h"
#include "main_functionality.h"
#include <stdlib.h>
#include <string.h>

void remove_duplicate(struct Text* text)
{
	setlocale(LC_ALL, "");

	int i = 0;
	int j = i + 1;
	struct Sentence* first_sentence;
	struct Sentence* second_sentence;

	while (i < text->size-1)
	{
		first_sentence = &text->content[i];
		while (j < text->size)
		{
			second_sentence = &text->content[j];
			if (wcscasecmp(first_sentence->content, second_sentence-
>content) == 0)
			{
				free(second_sentence->content);
				memmove(second_sentence, (second_sentence+1), (text-
>size-j+1)*sizeof(struct Sentence));
				text->size -= 1;
			} else
			{
				j++;
			}
		}
		i++;
		j = i + 1;
	}
}

void remove_uppercase_latin(struct Text* text)
{
	setlocale(LC_ALL, "");

	int j;
	struct Sentence* sentence;

	for (int i = 0; i < text->size; i++)
	{
		j = 0;
		sentence = &text->content[i];
		while (j < sentence->size)
		{
			if (sentence->content[j] >= 'A' && sentence->content[j] <=
'Z')
			{
				memmove(&sentence->content[j], &sentence->content[j+1],
(sentence->size-j+1)*sizeof(wchar_t));
				sentence->size -= 1;
			} else
```

```
                {
                        j++;
                }
        }
    }
}

void sort_by_cirillic(struct Text* text)
{
    setlocale(LC_ALL, "");

    qsort(text->content, text->size, sizeof(struct Sentence), compare);
}

int compare(const void* a, const void* b)
{
    setlocale(LC_ALL, "");

    struct Sentence* first_sentence = (struct Sentence*) a;
    struct Sentence* second_sentence = (struct Sentence*) b;

    if (first_sentence->number_of_cirillic_characters < second_sentence-
>number_of_cirillic_characters)
    {
        return 1;
    } else if (first_sentence->number_of_cirillic_characters >
second_sentence->number_of_cirillic_characters)
    {
        return -1;
    }
    return 0;
}

void remove_special(struct Text* text)
{
    setlocale(LC_ALL, "");

    struct Sentence* sentence;

    int i = 0;
    while (i < text->size)
    {
        sentence = &text->content[i];
        if (sentence->are_there_sp_symbols == 0)
        {
            free(sentence->content);
            memmove(sentence, (sentence+1), (text->size-i+1)*sizeof(struct
Sentence));
            text->size -= 1;
        } else
        {
            i++;
        }
    }
}
```

Название файла: time_diffs.c

```
#include "structs.h"
#include "time_diffs.h"
#include "colors.h"
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

void print_time_diffs(struct Text* text)
{
	setlocale(LC_ALL, "");

	time_t now = time(NULL);
	struct tm* this_time = localtime(&now);
	int time_now = this_time->tm_hour*60 + this_time->tm_min;

	int time = 0;
	int index;
	int counter = 0;

	wprintf(L"%s\n", TEXT_OTPUT);
	for (int i = 0; i < text->size; i++)
	{
		index = 2;
		time = find_time_diff(&text->content[i], &index);
		while (time >= 0)
		{
			wprintf(L"Подстрока найдена в предложении %d, разница с
текущим временем: %d.\n", i+1, abs(time-time_now));
			time = find_time_diff(&text->content[i], &index);
			counter++;
		}
	}
	if (counter == 0)
	{
		wprintf(L"Не найдено\n");
	}
	wprintf(L"%s\n", NORMAL);
}

int find_time_diff(struct Sentence* sentence, int* index)
{
	setlocale(LC_ALL, "");

	int hours, minutes;
	for (int i = *index; i < sentence->size-3; i++)
	{
		if (sentence->content[i] == ':' && (iswdigit(sentence->content[i-2])
&& iswdigit(sentence->content[i-1]) && iswdigit(sentence->content[i+1]) &&
iswdigit(sentence->content[i+2])) && ((i-2 == 0 || iswalnum(sentence->content[i-
3]) == 0) && (iswalnum(sentence->content[i+3]) == 0)))
		{
			hours = (sentence->content[i-2]-'0')*10 + (sentence-
>content[i-1]-'0');
			minutes = (sentence->content[i+1]-'0')*10 + (sentence-
>content[i+2]-'0');
			if (hours < 24 && minutes < 60)
			{
```

```
                        *index = i + 6;
                        return hours*60 + minutes;
                  }
            }
      }
      return -1;
}
```

Название файла: useful.c

```
#include "structs.h"
#include "useful.h"
#include "colors.h"
#include <stdio.h>
#include <stdlib.h>

void print_text(struct Text* text, wchar_t sep)
{
      setlocale(LC_ALL, "");

      wprintf(L"%s\n", TEXT_OTPUT);
      for (int i = 0; i < text->size-1; i++)
      {
            wprintf(text->content[i].content);
            wprintf(L"%c", sep);
      }
      wprintf(text->content[text->size-1].content);
      wprintf(L"%s\n\n", NORMAL);
}

void free_text(struct Text* text)
{
      setlocale(LC_ALL, "");

      for (int i = 0; i < text->size; i++)
      {
            free(text->content[i].content);
      }
      free(text->content);
}

void output_request(struct Text* text)
{
      setlocale(LC_ALL, "");

      int command = 0;
      wprintf(L"%sУспешно!%s\nВывесте результат на экран? Если да, то введите
'1'\n>>> ", SUCCESS, NORMAL);
      wscanf(L"%d", &command);
      if (command == 1)
      {
            print_text(text, ' ');
      }
}
```

Название файла: structs.h

```
include <wchar.h>
#include <wctype.h>
#include <locale.h>


struct Sentence
{
	wchar_t* content;
	int number_of_cirillic_characters;
	int are_there_sp_symbols;
	int size;


};


struct Text
{
	struct Sentence* content;
	int size;


};
```

Название файла: colors.h

```
#define TEXT_OTPUT "\033[40;97m"
#define NORMAL "\033[0m"
#define SUCCESS "\033[40;32m"
#define DUNGER "\033[107;31m"
#define TEXT_INPUT "\033[40;94m"
```

Название файла: get_text.h

```
int get_text(struct Text* output);
int get_sentence(struct Sentence* output);
int get_content(wchar_t** string);
void remove_spaces(wchar_t* string);
int sp_symbols_detector(wchar_t* string);
int cirillic_characters_counter(wchar_t* string);
```

Название файла: main_functionality.h

```
void remove_duplicate(struct Text* text);
void remove_uppercase_latin(struct Text* text);
void sort_by_cirillic(struct Text* text);
void remove_special(struct Text* text);
int compare(const void* a, const void* b);
```

Название файла: time_diffs.h

```
void print_time_diffs(struct Text* text);
int find_time_diff(struct Sentence* sentence, int* index);
```

Название файла: useful.h

```
void print_text(struct Text* text, wchar_t sep);
void free_text(struct Text* text);
void output_request(struct Text* text);
```

Название файла: Makefile

```
all: main.o main_functionality.o time_diffs.o useful.o get_text.o structs.h main_functionality.h time_diffs.h useful.h get_text.h colors.h
	gcc main.o main_functionality.o time_diffs.o useful.o get_text.o -o coursework

main.o: main.c structs.h main_functionality.h time_diffs.h useful.h get_text.h colors.h
	gcc -c main.c

main_functionality.o: main_functionality.c structs.h main_functionality.h
	gcc -c main_functionality.c

time_diffs.o: time_diffs.c structs.h time_diffs.h colors.h
	gcc -c time_diffs.c

useful.o: useful.c structs.h useful.h colors.h
	gcc -c useful.c

get_text.o: get_text.c structs.h get_text.h
	gcc -c get_text.c

clean:
	rm *.o coursework
```