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

Курсовая работа по дисциплине «Программирование» Тема: Обработка строк.

Студентка гр.9382 Тимофеева А.А.

Преподаватель Жангиров Т.Р.

Санкт-Петербург
2019

ЗАДАНИЕ НА
КУРСОВУЮ РАБОТУ

Студентка Тимофеева А.А.

Группа 9382

Тема работы: обработка строк.

Исходные данные: программа должна быть модульной, обработка данных должна производится при возможности с использованием функций стандартных библиотек языка. Сохранение, обработка и вывод данных должны осуществляться с помощью работы со структурами и динамической памятью.

Содержание пояснительной записки:
«Содержание», «Введение», «Задание работы», «Ход выполнения работы» «Заключение», «Список использованных источников».

Предполагаемый объем пояснительной записки:
Не менее 15 страниц.

Дата выдачи задания: 20.12.2019
Дата сдачи реферата:
Дата защиты реферата:

Студентка гр. 9382 Тимофеева А.А.

Преподаватель Жангиров Т.Р.

АННОТАЦИЯ

В курсовой работе была реализована обработка текста произвольной длины, для этого были использованы структуры и динамические массивы. Также в работе активно использовались стандартные библиотеки языка Си, содержащиеся в них функции и типы данных (как, например, в struct tm в заголовочном файле time.h). В работе программы была реализована защита от некорректного ввода данных, неудачного перевыделения памяти в ходе работы. О каждой из ошибок выводится соответствующее сообщение. В программе реализован элементарный интерфейс общения с пользователем и выполнение запрашиваемых им действий.

Пример работы программы приведён в приложении А.

SUMMARY

In the course work was implemented text processing of arbitrary length, for this were used structures and dynamic arrays. In addition, standard C libraries were actively used in the work, the functions and data types contained in them (as, for example, in the struct tm in the header file time.h). The program was implemented protection from incorrect data entry, unsuccessful memory allocation during the work. Each of the errors displays a corresponding message. The program implements an elementary interface for communicating with the user and performing the actions requested by him. An example of the program is given in Appendix A.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы: создать стабильную программу, производящую выбранную пользователем обработку данных. Реализация программы должна содержать работу со структурами (для хранения текста и отдельных предложений, а также дополнительных данных), работу с динамически выделенной памятью и использование стандартных библиотек, в том числе для работы с национальным алфавитом (wchar.h).

В результате была создана программа, считывающая вводимый пользователем с консоли текст, выводящая меню со списком доступных функций и выполняющая выбранные. По окончанию работы программа выводит обработанных массив предложений – текст. Также производятся действия по очистке динамически выделенной памяти и корректному завершению работы.

1. ЗАДАНИЕ

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) Распечатать каждое слово, которое встречается не более одного раза в тексте.
2) Каждую подстроку в тексте имеющую вид “<день> <месяц> <год> г” заменить на подстроку вида “ДД/ММ/ГГГГ”. Например, подстрока “20 апреля 1889 г.” должна быть заменена на “20/04/1889”.
3) Отсортировать предложения по произведению длин слов в предложении. 4) Удалить все предложения, которые содержат символ ‘#’ или ‘№’, но не содержат ни одной 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.

2. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

2.1. Создание структур и общего заголовочный файл.

Для работы с текстом были созданы две структуры Text и Sentense, при объявлении при помощи оператора typedef они были определены как новые типы данных text_s и sentense_s соответственно. Структуры были объявлены в заголовочном файле, и была подключена препроцессорная директива pragma once для избежание повторного объявления. (Далее будем опускать, что каждый заголовочный файл проходил аналогичное подключение).

Содержание структур: в Text содержится переменная – счётчик строк и динамический массив указателей на структуру Sentense. В которой соответственной содержаться: сама считанная строка, её длина, динамический массив слов строки, приведённых к нижнему регистру, количество слов в строке и их произведение (слова и произведение необходимы для работы функций программы).

В заголовочном файле standcl.h содержится подключение всех стандартных библиотек (используемых везде) и заголовочного файла со структурами.

2.2. Чтение текста и заведение его в структуру.

В файле inputcww.c содержаться три функции, созданных для первоначального считывания текста и заведения его в созданные ранее структуры. Все функции имеют возвращаемое значение типа int для сигнализации об ошибках внутри своей работы и их соответствующей обработке вне данного файла.

Функция int ERNUTSYMBSTR(wchar_t **str) производит считывание всего входного теста во временную строку. Для получения данных со стандартного потока ввода используется безопасное посимвольное считывание до символа

переноса строки (‘\n’). Перевыделение памяти происходит с использованием временного указателя и проверки возвращаемых значений функций библиотек malloc.h и stdlib.h

Функция int MAKEARRAYPLEASE(wchar_t *temp, text_s *text), путём использования библиотечной функции wcstok(), делит полученную временную строку по символу-разделителю (точке) на отдельные токены, т.е. предложения, которые потом заносятся в соответствующую структуру. Фрагменты памяти, содержащие необходимую информацию переносятся при помощи функции wmemcpy().

Функция int DELETESPACES(text_s *text) была создана для приведение предложений внутри структур к изначальному виду до обработки: перед каждым предложением удаляются все пробелы, так как они являются лишь разделителями между словами в тексте. Данная операция реализована при помощи подсчёта их количества и последующего сдвига всей строки для удаления. Далее идёт добавление точки в предложения (она исчезла на этапе разделения на токены, так как wcstok() преобразует символ-разделитель в терминальный ноль).

2.3. Начальная обработка текста.

В файле before_procession.c содержаться функции для подготовки заданного текста. Так функция int delete_repeat(text_s *text) создаёт две временные строки: одна – текущая в цикле по всем строкам, другая – последующая, приводит с помощью библиотечной функции towlower() их к нижнему регистру (по условию задания предложения считаются одинаковыми вне зависимости от регистра) и последовательно сравниваются с помощью стандартной функции wcscmp(). Такая операция выполняется для каждой строки с начала текста. В случае нахождения строки 2, равной строке 1, происходит удаление второй, путём сдвига массива строк и очистка последнего элемента. Также по завершении работы происходит пересчёт длин предложений для корректной работы функции сортировки по длине слов в предложении.

Для выполнения функций с произведениями слов в предложении и замены формата даты осуществляется работа функции int split_words (text_s *text), которая разделяет предложения на отдельные слова и заносит их в динамический массив, приводя все к нижнему регистру. Разделение слов осуществляется с использованием уже упомянутой функции wcstok(), разделение же осуществляется по таким знакам: пробел, запятая и точка (для обработки последнего слова).

2.4. Функции для работы с пользователем.

В файле outcwww.c созданы функции для осуществления взаимодействия с пользователем. Перед ними в препроцессорных директивах описаны спецификаторы формата вывода для осуществления цветового выделения выводимой на консоль информации. Функция void print_usr_hello() печатает приветствие пользователя и предупреждение, что одинаковые предложения в ходе работы программы будут удалены. Функции void all_good() и void chachacha() созданы для оповещение пользователя о соответственном успешном или неудачном выполнении функции или операции. Функция void print_menu() осуществляет печать доступных пользователю функций с указаниями на какую кнопку для этого нужно нажать. Функция void print_bye() выполняется после указанием пользователем опции выхода из программы. Она выводит на консоль информацию об получившимся в итоге текста, его самого и прощание с пользователем.

Также в данном файле реализована функция int scan_func(text_s *text) реализующая основное взаимодействие с пользователем в выборе функции. В ней выполнена защита на некорректный ввод и, в случае ввода пользователем цифры, означающей несуществующую функцию, или же иного непредусмотренного текста будут выводится разные сообщения об ошибке, но программа будет продолжать работу. После получения на вход номера требуемой для выполнения операции, функция при помощи оператора switch() осуществляет перенаправление входных данных в одну из функций,

последующий вывод сообщения о завершении работы и переходит на новую итерацию (если не был введён 0 - признак окончания работы программы).

2.5. Функции для выполнения указанных операций.

В данном разделе описываются функции, на которые ссылается scan_func(), каждая и них выполняет определённые операции над текстом через указатель.

2.5.1 Печать уникальных слов.

В файле words_meet_once.c содержаться функции, выполняющие вывод на экран слов, встречающихся тексте не более одного раза. В начала вызывается функция void print_words(text_s *text) осуществляющая последовательный проход по всему тексту и вызов для каждого слова функции int check_word_unic(text_s *text, wchar_t *check, int non_include_str, int non_include_word), которая игнорирую текущее переданное слово ищет во всём тексте совпадения. Если таковых не найдено, и функция вернула соответствующее значение, то оно выводится на консоль. Если данная функция идет после операции замены формата дат, то они не учитываются, так как их значения в виде единицы текста не определено (формат времени).

2.5.2 Замена формата дат

В файле changedate.c содержаться функции, необходимые для замены дат формата “<день> <месяц> <год> г” на “ДД/ММ/ГГГГ”. В начале вызывается функция void find_date(text_s *text), выполняющая последовательный проход по всем строкам и словам, содержащимся в ней и проверку на содержание в них даты строго приведённого формата, также проверяя корректность даты. В случае нахождения данной подстроки, вызывается функция wchar_t *changedate(text_s *text, int numb_of_str, int beg_of_word, int length_month), которой передаются: исходный текст, номер строки, в которой была найдена дата на замену, номер позиции начала даты в строке и длина месяца в найденной дате (для осуществления последующего смещения строки).

В вызванной функции выполняется копирование даты во временную строки, из которой затем при помощи функции swscanf() сканируются значения текущего дня, месяца и года. Преобразования года к корректному значению производится путём вычитания 1900 и преобразование месяца осуществляется с помощью функции wcsmon(), описанной в заголовочном файле exeptional_clean.h, сопоставляющей буквенное значение месяца его цифровому значению. (данная функция реализована из-за отсутствия поддержки в стандартных библиотеках языка Си функций наподобие wcsptime() и т.п. для совместной работы с “широкими символами” и временными структурами). Далее, при помощи стандартной функции wcsftime() (заголовочный файл time.h подключен в данной функции), осуществляется конвертация полученных данных в необходимый формат представления. Также в данной функции осуществляется последовательная замена даты старого формата на новый и соответствующий сдвиг строки для их корректного отображения. После этого происходит сдвиг слов в их динамическом массиве и выход из текущей функции. В данной функции сделаны дополнительные средства проверки сдвигов строки для случаев, например, с годом, состоящим из 3 и менее цифр и т.д.

2.5.3 Функция сортировки текста по произведению длин слов в предложении

В файле sort_sent.c содержаться функции для сортировки текста по произведению длин слов в предложении. Изначальное вызывается функция void sort_sent(text_s *text), вызывающая последовательно функции void count_prod(text_s *text) и qsort(text->sent, text->quant, sizeof(text->sent[0]), compar). Первая выполняет в цикле последовательное вычисление произведения длин слов в тексте. Вторая - стандартная функция библиотеки stdlib.h, выполняющая быструю сортировку массива структур (предложений) по вычисленному полю произведении.

Особенностью данной функции является то, что её вызов после операции замены дат на требуемый формат, она перестаёт учитывать исходные слов

(они удалены смещением из массива слов), но и не учитывает в произведении новую строку (дату), так как её значения в виде единицы текста не определено (формат времени).

2.5.4 Функция удаления некорректных предложений

В файле delete_symbols_not_numbers.c содержаться функции для удаления некорректных (прим. по условию задания некорректные - содержащие символ “#” или “№”, но не содержащие цифры) предложений из текста. Так как вы работе функции iswidigit() совместно с функцией wcscspn() для поиска указанных символов были обнаружены неточности, то поиск выполнения данных условий осуществляется при помощи двух вызовов функции wcscspn(), вторая из которых осуществляет поиск символов - цифр в диапазоне 0-9. Проверка основана на возвращаемом значении данной функции, так как в случае отсутствия заданных элементов, она возвращает длину строки, в которой осуществляется поиск. Соответственной комбинируя данные условия функция определяет необходимые для удаления предложения. Удаление реализовано уже описанным алгоритмом последовательного сдвига текста в строке и очистки последнего. Также по завершении работы происходит пересчёт длин предложений для корректной работы функции сортировки по длине слов в предложении.

2.5.5 Промежуточный вывод текста

Также у пользователя есть возможность просмотреть промежуточный результат работы программы над текстом, для этого по вызову функции 5 происходит вывод обработанного текста.

2.6. Очистка памяти и завершение работы.

После вызова пользователем функции завершения работы программы происходит вызов функции void CLEANINGMEM(text_s *text) из заголовочного файла exeptional_clean.h. Она последовательно очищает память в структурах и загуляет (присваивает NULL) указатели.

2.7. Создание Make файлы.

После создания всех файлов с исходным кодом необходимых функций был создан файл makefile для консольной утилиты make, в которой были установлены корректные зависимости компиляции и цели для каждого бинарного файла отдельных исходников.

ЗАКЛЮЧЕНИЕ

В результате выполнения данной работы были на практике применены приёмы работы со структурами, динамической память, стандартными функциями библиотек языка Си. Была создана стабильная программа с обработкой нестандартных ситуаций (ввода) и выполняющая функции, указанные в задании и выбранные пользователем. Программа осуществляет взаимодействие с пользователем через примитивный консольный интерфейс и корректно обрабатывает различные виды корректных входных данных (в соответствии с условием).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1.Керниган Б. и Ритчи Д. Язык программирования Си. М.: Вильямс, 1978 288 с.

2.Официальный сайт IBM // IBM Knowledge Center. URL: https:// www.ibm.com/support/knowledgecenter/en/ (дата обращения: 02.12.2018).

3. Стив Оолайн C Elements of Style M.: M&T books, 1992 456 с.

ПРИЛОЖЕНИЕ А
ПРИМЕР РАБОТЫ ПРОГРАММЫ

Пример вывода приветствия пользователя и ввод им текста (An example of the output of user greetings and text input):

```
                Добрый день!
                Введите, пожалуйста, текст, требующий обработки:
Замечание: одинаковые предложения (без учёта регистра) будут удалены.
Остальная пехота поспешно проходила по мосту, спираясь воронкой у входа. Наконец повозки все прошли, давка стала меньше, и последний батальон вступил на мост. Одни гусары эскадрона Денисова оставались по ту сторону моста против неприятеля. Неприятель, вдалеке видный с противоположной горы, снизу, от моста, не был еще виден, так как из лощины, по которой текла река, горизонт оканчивался противоположным возвышением не дальше полуверсты. Впереди была пустыня, по которой кое-где шевелились кучки наших разъездных казаков. Вдруг на противоположном возвышении дороги показались войска в синих капотах и артиллерия. Это были французы. Разъезд казаков рысью отошел под гору. Все офицеры и люди эскадрона Денисова, хотя и старались говорить о постороннем и смотреть по сторонам, не переставали думать только о том, что было там, на горе, и беспрестанно всё вглядывались в выходившие на горизонт пятна, которые они признавали за неприятельские войска.          Введите действие, которое вы хотите выполнить: Чтобы распечатать каждое слово, которое встречается в тексте не более одного раза нажмите - 1;
Чтобы заменить все подстроки вида "<день> <месяц> <год> г." на "ДД/ММ//ГГГГ" нажмите - 2;
Чтобы отсортировать предложения по произведению длин слов нажмите - 3;
Чтобы удалить все предложения, содержащие символ # или №, но не содержащих цифры, нажмите - 4; Чтобы выйти из программы нажмите - 0.
```

Пример вывода прощания с пользователем и результата (Example output of the farewell to the user and the result):

```
Полученный после работы программы текст.
Всего предложений: 9
Остальная пехота поспешно проходила по мосту, спираясь воронкой у входа.
Наконец повозки все прошли, давка стала меньше, и последний батальон вступил на мост.
Одни гусары эскадрона Денисова оставались по ту сторону моста против неприятеля.
Неприятель, вдалеке видный с противоположной горы, снизу, от моста, не был еще виден, так как из лощины, по которой текла река, горизонт оканчивался противоположным возвышением не дальше полуверсты. Впереди была пустыня, по которой кое-где шевелились кучки наших разъездных казаков.
Вдруг на противоположном возвышении дороги показались войска в синих капотах и артиллерия.
Это были французы.
Разъезд казаков рысью отошел под гору.
Все офицеры и люди эскадрона Денисова, хотя и старались говорить о постороннем и смотреть по сторонам, не переставали думать только о том, что было там, на горе, и беспрестанно всё вглядывались в выходившие на горизонт пятна, которые они признавали за неприятельские войска.
До свидания!
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ПРОГРАММЫ

название файла: main.c

```
#include "standcl.h"
#include "outcwww.h"
#include "before_procession.h" #include
"inputcww.h"

int main(){
   setlocale(LC_CTYPE, "");
   GREETING(); //выводит приветсвие пользователя
wchar_t *temp;
   if (VERNUTSYMBSTR(&temp)){ //читаем текст во временную строку
chachacha();
      return 0;
   }
   text_s text;
```

```
   if (MAKEARRAYPLEASE(temp, &text)){ //создаём двумерный динамический массив
chachacha();
       return 0;
   }
   free(temp);
   if (DELETESPACES(&text)){ //удаляем лишние пробелы и возвращаем точки
chachacha();
       return 0;
   }
   if (delete_repeat(&text)){ //удаляем повторяющиеся предложения
chachacha();
       return 0;
   }

   if (split_words(&text)){ //разделяем каждую строку на слова
chachacha();
       return 0;
   }

   scan_func(&text); //запрашиваем, считываем и выполняем требуемую функцию
return 0;
}
```

название файла inputcwww.c:

```
#include "standcl.h"

#ifndef MAKE_BIG
#define MAKE_BIG 10 //для увеличение размера и т.п. #endif

//вовзращает кол-во символов в строке int
VERNUTSYMBSTR(wchar_t **str){
   int avl_sy = MAKE_BIG; //доступное кол-во символов
int symb = 0; //текущее кол-во символов
   *str = (wchar_t *)malloc(avl_sy *
sizeof(wchar_t));     if (!str)         return 1;
   wchar_t input = '.'; //переменная для посимвольного ввода
while (input != L'\n'){         wscanf(L"%lc", &input);         if (input
== L'\n')             break;         if (avl_sy <= symb){             avl_sy
+= MAKE_BIG;
           void *tmp = realloc(*str, avl_sy *
sizeof(wchar_t));             if (!tmp)                 return 1;
           *str = (wchar_t *)tmp;
       }
       (*str)[symb] = input;
symb++;
   }
   (*str)[symb] = L'\0';
return 0;
}

int MAKEARRAYPLEASE(wchar_t *temp, text_s *text){     text-
>quant = 0; //текущее кол-во предложений
   long int avl_sent = MAKE_BIG; //доступное кол-во предложений
```

```
   text->sent = (sentense_s *)malloc(avl_sent * sizeof(sentense_s)); //выделение памяти для
предложения
   if (!text->sent)
      return 1;
   wchar_t *token = NULL;    wchar_t
*buffer = NULL;    token =
wcstok(temp, L".", &buffer);    while
(token != NULL){       if (avl_sent <=
text->quant){          avl_sent +=
MAKE_BIG;
         void *tmp = realloc(text->sent, avl_sent * sizeof(sentense_s));
if (!tmp)
            return 1;
         text->sent = (sentense_s *)tmp;
      }
      text->sent[text->quant].str = (wchar_t *)malloc( (wcslen(token) + 2) * sizeof(wchar_t)); //+1
т.к. wclen не учитывает \0       if (!text->sent[text->quant].str)          return 1;
      wmemcpy(text->sent[text->quant].str, token, wcslen(token) + 1); //переносим фрагмент
памяти token в структуру       text->quant++;
      token = wcstok(NULL, L".", &buffer);
   }
   token = NULL;
buffer = NULL;
   return 0; }

//удаление пробелов в начале предложений, появившихся из-за wcstok. Также
добавление в них точек. int DELETESPACES(text_s *text){
   int c_tabs = 0; //кол-во пробелов в начале для удаления
int i, j;
   int now_len = 0;
   for (i = 0; i < text->quant;
i++){       now_len = wcslen(text-
>sent[i].str);       while (text-
>sent[i].str[c_tabs] == L' ')
c_tabs++;

      for (j = 0; j < wcslen(text->sent[i].str); j++)          text-
>sent[i].str[j] = text->sent[i].str[j + c_tabs];

      if (c_tabs == 0){
         void *tmp = realloc(text->sent[i].str, (now_len + 2) * sizeof(wchar_t)); //+1 для точки
if (!tmp)
            return 1;
         text->sent[i].str = (wchar_t *)tmp;
      }
      text->sent[i].str[now_len - c_tabs] = L'.'; //на место терминального 0 пишем
точку       text->sent[i].str[now_len - c_tabs + 1] = L'\0'; //после пишем терминальный
0       text->sent[i].length = wcslen(text->sent[i].str);       c_tabs = 0;
   }
   return 0;
}
```

название файла outcwww.c:

```
#include "standcl.h"

#include "words_meet_once.h"
#include "changedate.h"
#include "sort_sent.h"
#include "delete_symbols_no_nubers.h"
#include "exceptional_clean.h"

#define RED_ERR "\033[1;31m" //красный полужирный
#define GREEN "\033[5;32m" //зелёный мигающий
#define CYAN  "\033[1;36m" //сине-зелёный полужирный
#define NONE "\033[0m" //возвращение к дефолту
#define FUNC "\033[3;97m" //белый италик
#define START "\033[1;97m" //полужирный белый

void GREETING(){
   wprintf(L"\n\t\t%sДобрый день!\n\t\t%sВведите, пожалуйста, текст, требующий
обработки:%s\n", START, START, NONE);     wprintf(L"%sЗамечание: одинаковые
предложения (без учёта регистра) будут удалены. %s\n", GREEN, NONE);
}

void all_good(){
   wprintf(L"%sФункция успешно выполнена!%s\n", CYAN, NONE);
}

void chachacha(){
   wprintf(L"%sSomething went wrong with your memmory!%s\n", RED_ERR, NONE);
}

void print_menu(){
   wprintf(L"\t%sВведите действие, которое вы хотите выполнить:%s\n", START, NONE);
wprintf(L"%sЧтобы распечатать каждое слово, которое встречается в тексте не более
одного раза нажмите - 1;%s\n", FUNC, NONE);
   wprintf(L"%sЧтобы заменить все подстроки вида \"<день> <месяц> <год> г.\" на \"ДД/
ММ//ГГГГ\" нажмите - 2;%s\n", FUNC, NONE);
   wprintf(L"%sЧтобы отсортировать предложения по произведению длин слов нажмите -
3;%s\n", FUNC, NONE);
   wprintf(L"%sЧтобы удалить все предложения, содержащие символ # или №, но не
содержащих цифры, нажмите - 4;%s\n", FUNC, NONE);
   wprintf(L"%sЧтобы вывести состояние текста на данный момент, нажмите - 5;%s\n",
FUNC, NONE);
   wprintf(L"%sЧтобы выйти из программы нажмите - 0.%s\n", FUNC, NONE);
}

void print_text(text_s *text){
   int i;
   for (i = 0; i < text->quant; i++)
wprintf(L"%ls\n", text->sent[i].str);
}

int scan_func(text_s
*text){     int need_func =
0;     int check_cont = 1;
wchar_t bufStub = 0;
while (check_cont)
```

```
{       print_menu();
bufStub = 0;
      if((wscanf(L"%i%lc",&need_func, &bufStub) != 2) || bufStub !=
L'\n'){          wprintf(L"%sWrong input!%s\n", RED_ERR, NONE);
while(bufStub != '\n')              wscanf(L"%lc",&bufStub);
      } else {
         switch
(need_func){             case 1:
print_words(text);
check_cont = 1;
all_good();
             break;
          case 2:
FINDINGDATE(text);
check_cont = 2;
all_good();                 break;
          case 3: sort_sent(text);
check_cont = 3;                 all_good();
             break;
          case 4: DELETEWITHSYMBNONUMB(text);
             check_cont = 4;
all_good();
break;
          case 5: print_text(text);
check_cont = 5;
break;              case 0:
             wprintf(L"\n%sПолученный после работы программы текст.%s\n", CYAN, NONE);
             wprintf(L"Всего предложений: %li\n", text->quant);
print_text(text);
             wprintf(L"\nДо свидания!\n");
check_cont = 0;
//CLEANINGMEM(text);
break;
          default: wprintf(L"%sNo option selected!%s\n", RED_ERR, NONE);
check_cont = 5;                 break;
        }
      }
   }
   return 0;
}
```

название файлы changedate.c:

```
#include "standcl.h"
#include <time.h>
#include "changedate.h" #include
"exceptional_clean.h"

int changedate(text_s *text, int numb_of_str, int beg_of_word, int length_month, int
numb_offset, int numb_of_word, int ln, int check){
   const int length_start = numb_offset + length_month + 2; //длина изначальной
подстрокидаты; +2 - это пробел и г.
   wchar_t *temp = malloc(50 * sizeof(wchar_t));
if (ln == 1)
```

```
        wmemcpy(temp, &text->sent[numb_of_str].str[beg_of_word], length_start);
else
        wmemcpy(temp, &text->sent[numb_of_str].str[beg_of_word - 1], length_start);
temp[length_start] = L'\0';     struct tm new_time;     new_time.tm_sec = 0;
new_time.tm_min = 0;     new_time.tm_hour = 0;     new_time.tm_isdst = 0;
    wchar_t *month = malloc((length_month + 1) * sizeof(wchar_t));
    swscanf(temp, L"%i %ls %i", &new_time.tm_mday, month, &new_time.tm_year);
new_time.tm_year -= 1900;     new_time.tm_mon = MONTH(month);
    wchar_t *date_out = (wchar_t *)malloc(12 * sizeof(wchar_t));
wcsftime(date_out, 12, L"%e/%m/%Y", &new_time);

    if(check) //если в предыдущей дате не однозначное число и не буквы обр.
beg_of_word--;

    int i;
    //запись даты нового формата в строку
for (i = 0; i < wcslen(date_out); i++){
        text->sent[numb_of_str].str[beg_of_word++] = date_out[i];
    }

    //смещение строки для новой даты
    int diff = length_month;
    for (beg_of_word; beg_of_word < text->sent[numb_of_str].length; beg_of_word++){
        text->sent[numb_of_str].str[beg_of_word] = text->sent[numb_of_str].str[beg_of_word + diff];
    }

    text->sent[numb_of_str].length = wcslen(text->sent[numb_of_str].str);

    //изменение массива слов к виду с новой датой     text-
>sent[numb_of_str].splword[numb_of_word] = date_out;
    for (i = numb_of_word + 1; i <= text->sent[numb_of_str].count_words - 3; i++){        text-
>sent[numb_of_str].splword[i] = text->sent[numb_of_str].splword[i + 3];
    }
    //очистка
    for (i = text->sent[numb_of_str].count_words; i > text->sent[numb_of_str].count_words - 3; i--)
{
        //free делать нельзя, т.к. просто перевешиваем указатели;        text-
>sent[numb_of_str].splword[i] = NULL;
    }
    text->sent[numb_of_str].count_words -= 3;
free(date_out);     date_out = NULL;
free(temp);     temp = NULL;     return 0;
}

void FINDINGDATE(text_s
*text){     int i, j;
    int length_month = 0; //длина месяца в найденной строке даты
long int pos = 0; //позиция начала подстроки-даты в строке     int
now_words = 0; //текущее число пройденных слов     int
numb_offset = 0;
    int old_ln = 0, temp_ln, old_y = 4, temp_y = 4;     int
func_call = 1, check = 1, check_func_alr = 0;     wchar_t
*months = (wchar_t *)malloc(200 * sizeof(wchar_t));
```

```
   months = L"января февраля марта апреля мая июня июля августа сентября октября
ноября декабря january february march april may june july august september october november
december\0";
   for (i = 0; i < text->quant; i++){ //идём по строкам
      while(now_words < text->sent[i].count_words - 3){ //идём по словам
pos += wcslen(text->sent[i].splword[now_words]);            //проверяем
корректность числа и названия месяца
         //wcstol - converts the initial portion of the wide-character string pointed to by nptr to a
long integer value
         if ( (wcstol(text->sent[i].splword[now_words], NULL, 10) <= 31) &&
(wcstol(text>sent[i].splword[now_words], NULL, 10) > 0) && (wcsstr(months, text-
>sent[i].splword[now_words + 1])) ){
               length_month = wcslen(text->sent[i].splword[now_words + 1]);
               if ( ( wcstol(text->sent[i].splword[now_words + 2], NULL, 10) > 0) &&
( wcslen(text>sent[i].splword[now_words + 3]) == 1 ) && (( text->sent[i].splword[now_words +
3][0] == L'г' ) ||
(text->sent[i].splword[now_words + 3][0] == L'y')) ){
                  numb_offset += wcslen(text->sent[i].splword[now_words]); //добавили длину
числа месяца
                  numb_offset += (wcslen(text->sent[i].splword[now_words + 2]) + 2); //пробелы
справа/слева от месяца
                  temp_ln = wcslen(text->sent[i].splword[now_words]);
temp_y = wcslen(text->sent[i].splword[now_words + 2]);
                  if (((func_call == 0) && (check_func_alr == 1)) ||(old_ln == 1) || (old_y < 3) ||
(temp_y
< 3)){
                     check = 0;
                  } else
check = 1;
                  changedate(text, i, pos - 1, length_month, numb_offset, now_words, wcslen(text-
>sent[i].splword[now_words]), check);
check_func_alr = 1;
                  func_call++;
                  pos += numb_offset; //число+месяц+год в числовом виде
old_ln = temp_ln;                  old_y = temp_y;
                  numb_offset = 0;
               }
  } else
            func_call = 0;
         pos++; //пробел между словами
         now_words++;
      }
      check_func_alr = 0;
func_call = 1;
check = 1;       old_ln =
0;       old_y = 1000;
pos = 0;
      now_words = 0;
   }
   months = NULL;
}
```

название файла exeptional_clean.c:

```
#include "standcl.h"

int MONTH(wchar_t *month){
   if (!(wcscmp(month, L"january")) || !(wcscmp(month, L"января")))
return 0;
   if (!(wcscmp(month, L"february")) || !(wcscmp(month, L"февраля")))
return 1;
   if (!(wcscmp(month, L"march")) || !(wcscmp(month, L"марта")))
return 2;
   if (!(wcscmp(month, L"april")) || !(wcscmp(month, L"апреля")))
return 3;
   if (!(wcscmp(month, L"may")) || !(wcscmp(month, L"мая")))
return 4;
   if (!(wcscmp(month, L"june")) || !(wcscmp(month, L"июня")))
return 5;
   if (!(wcscmp(month, L"july")) || !(wcscmp(month, L"июля")))
return 6;
   if (!(wcscmp(month, L"august")) || !(wcscmp(month, L"августа")))
return 7;
   if (!(wcscmp(month, L"september")) || !(wcscmp(month, L"сентября")))
return 8;
   if (!(wcscmp(month, L"october")) || !(wcscmp(month, L"октября")))
return 9;
   if (!(wcscmp(month, L"november")) || !(wcscmp(month, L"ноября")))
return 10;
   if (!(wcscmp(month, L"december")) || !(wcscmp(month, L"декабря")))
return 11;
}

void CLEANINGMEM(text_s *text){
   int i, j;
   for (i = 0; i < text->quant;
i++){        free(text->sent[i].str);        text-
>sent[i].str = NULL;
       for (j = 0; j < text->sent[i].count_words; j++){
          free(text->sent[i].splword[j]);
          text->sent[i].splword[j] = NULL;
       }
       free(text->sent[i].splword);
       text->sent[i].splword = NULL;
   }
   free(text->sent);     text-
>sent = NULL;
   text = NULL;
}
```

название файла: delete_symbols_no_numbers.c:

```
#include "standcl.h"

#include "before_procession.h"

//удаление предложений, содержащий # or №, но не содержащих цифры
void DELETEWITHSYMBNONUMB(text_s *text){     int k;
   int check_dig = 0; //проверка на наличие цифр
```

```
   int check_symb = 0; //проверка на наличие указанных символов
long int str_now = 0; //номер строки сейчас     while(str_now != text-
>quant){
       check_symb = wcscspn(text->sent[str_now].str, L"#№");
check_dig = wcscspn(text->sent[str_now].str, L"1234567890");
       /*iswidigit иногда отрабатывает некорректно. Случай с предложением - симвлолом/
цифрой
       невозможен из-за точки*/

       if ((check_symb != wcslen(text->sent[str_now].str)) && (check_dig == wcslen(text-
>sent[str_now].str))){
          for (k = str_now; k < text->quant - 1; k++){
             text->sent[k].str = (wchar_t *)realloc(text->sent[k].str, (wcslen(text->sent[k + 1].str) + 1)
* sizeof(wchar_t));
             wmemcpy(text->sent[k].str, text->sent[k + 1].str, wcslen(text->sent[k + 1].str) + 1);
          }
          free(text->sent[text->quant - 1].str);
text->sent[text->quant - 1].str = NULL;
text->quant--;            recount_length(text);
       } else
          str_now++; //чтобы проверять предложения подряд в случае удаления
check_dig = 0;
       check_symb = 0;
   }
}
```

название файла: cw_struct.h:

```
#pragma once
#include <wchar.h>

//объявляем структуру предложений и новый тип, чтобы обращаться к сткруктуре по
всевдониму typedef struct Sentense{     long int length; //длина строки     wchar_t *str;
//непосредственно строка
   wchar_t **splword; //массив, содержащихся в строке, слов
long int count_words; //кол-во слов в строке
   long long int prod; //произведение длин слов в предложении }
sentense_s;

//объявляем структуру текста
typedef struct Text{
   long int quant; //кол-во предложений
sentense_s *sent;
} text_s;
```

название файла pre_process_treat.c:

```
#include "standcl.h"

#ifndef MAKE_BIG
#define MAKE_BIG 10 //для увеличение размера и т.п.
#endif

void recount_length(text_s
*text){     int i;
   for (i = 0; i < text->quant; i++){
```

```
      text->sent[i].length = wcslen(text->sent[i].str);
   } }

//удаление повторяющихся предложений
int delete_repeat(text_s *text){
   int numb_string = 0; //номер текущей обрабатываемой строки
int i, j;
   wchar_t *str1 = NULL;
   wchar_t *str2 = NULL; //временные строки для нижнего индекса и сканирования
while(numb_string != text->quant){        //работа с 1-ой сравниваемой строкой
      str1 = (wchar_t *)malloc((wcslen(text->sent[numb_string].str) + 1) *
sizeof(wchar_t));        if (!str1)            return 1;
      wmemcpy(str1, text->sent[numb_string].str, wcslen(text->sent[numb_string].str) + 1);
for (i = 0; i < wcslen(str1); i++)
         str1[i] = towlower(str1[i]);
      //проверяем последующие строки на равенство
for (i = numb_string + 1; i < text->quant;
i++){           //работа с 2-ой сравниваемой строкой
         str2 = (wchar_t *)malloc((wcslen(text->sent[i].str) + 1) *
sizeof(wchar_t));           if (!str2)               return 1;
         wmemcpy(str2, text->sent[i].str, wcslen(text->sent[i].str) + 1);
for (j = 0; j < wcslen(str2); j++)               str2[j] = towlower(str2[j]);
//проверка на равенство           if ( !( wcscmp(str1, str2)) ){
            for (j = i; j < text->quant - 1; j++){ //на эл. меньше, т.к.последний уже не
рассматриваем
               text->sent[j].str = (wchar_t *)realloc(text->sent[j].str, (wcslen(text->sent[j + 1].str) + 1)
* sizeof(wchar_t)); //на 1 больше, т.к. фу-ия не учитывает
терм. 0                  if (!text->sent[j].str)                     return 1;
               wmemcpy(text->sent[j].str, text->sent[j + 1].str, wcslen(text->sent[j + 1].str) + 1);
            }
            free(text->sent[text->quant - 1].str); //т.к. индекс на 1 меньше, чем кол-во
text->sent[text->quant - 1].str = NULL;               text->quant--;
         }
         free(str2);
      }        free(str1);
recount_length(text);
      numb_string++;
   }
   return 0; }

//разделяем строку на слова int
split_words(text_s *text){
   int i, j;
   int avl_words;
wchar_t *temp;
wchar_t *token;
wchar_t *buffer;
   for(i = 0; i < text->quant; i++){
      avl_words = MAKE_BIG; //доступное кол-во слов
      text->sent[i].splword = (wchar_t **)malloc(avl_words * sizeof(wchar_t *));
//выделение памяти на массив слов        text->sent[i].count_words = 0; //текущее кол-
во слов
```

```
	//перекопировали во временную строку, т.к. wcstok меняет исходную
temp = (wchar_t *)malloc((text->sent[i].length + 1) * sizeof(wchar_t));	if
(!temp)		return 1;
	wmemmove(temp, text->sent[i].str, text->sent[i].length + 1);

	token = wcstok(temp, L" ,.", &buffer); //точка, чтобы отдельно не работать с последним
словом
	while (token != NULL){
	  if (avl_words <= text-
>sent[i].count_words){		avl_words += MAKE_BIG;
	    void *tmp = realloc(text->sent[i].splword, avl_words * sizeof(wchar_t *));
if (!tmp)
		return 1;
	    text->sent[i].splword = (wchar_t **)tmp;
	  }
	  text->sent[i].splword[text->sent[i].count_words] = (wchar_t *)malloc((wcslen(token) + 1) *
sizeof(wchar_t));
	  if (!text->sent[i].splword[text->sent[i].count_words])
return 1;
	  wmemmove(text->sent[i].splword[text->sent[i].count_words], token, wcslen(token) + 1);
	  //приведение слов к нижнему регистру
	  for (j = 0; j < wcslen(text->sent[i].splword[text->sent[i].count_words]); j++)		text-
>sent[i].splword[text->sent[i].count_words][j] = towlower(text-
>sent[i].splword[text->sent[i].count_words][j]);		text-
>sent[i].count_words++;
	  token = wcstok(NULL, L" ,.", &buffer);
	}
	free(temp);
temp = NULL;
token = NULL;
	buffer = NULL;
   }
   return 0;
}
```

название файла sort_sent.c:

```
#include "standcl.h"

//компаратор произведения длин слов в тексте
int compar(const void *ptr1, const void *ptr2)
{	long int p1 = ((sentense_s *)ptr1)->prod;
long int p2 = ((sentense_s *)ptr2)->prod;
    if (p1 == p2)
return 0;	else
if (p1 > p2)
return 1;
else
return -1; }

//считаем произведение каждого предложения void
count_prod(text_s *text){
   int i, j;
```

```
   for (i = 0; i < text->quant; i++){        text-
>sent[i].prod = 1;
       for (j = 0; j < text->sent[i].count_words;
j++){            if (wcslen(text->sent[i].splword[j]) != 0)
               text->sent[i].prod *= wcslen(text->sent[i].splword[j]);
       }
   } }

//сортируем предложения
void sort_sent(text_s
*text){     count_prod(text);
   qsort(text->sent, text->quant, sizeof(text->sent[0]), compar);
}
```

название файла standcl.h:

```
#pragma once

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>
#include <locale.h>
#include <wctype.h>
#include "cw_struct.h"
```