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

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

Студентка гр. 0382 _____ Чегодаева Е.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студентка Чегодаева Е.А.

Группа 0382.

Тема работы: Обработка строк на языке Си.

Вариант 5.

Программе на вход подается текст (текст представляет собой предложения, разделенные точкой). Предложения - набор слов, разделенные пробелом или запятой, слова - набор латинских букв, цифр и символ ‘/’. Длина текста и каждого предложения заранее не известна.

Программа должна сохранить этот текст в динамический массив строк и оперировать далее только с ним.

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) Вывести все предложения, в которых есть даты в формате “DD/MM/YYYY”, даты которые еще не наступили надо выделить красным цветом.

2) Удалить все предложения, в которых каждое слово содержит нечетное количество букв.

3) Преобразовать предложения так, чтобы перед каждым словом стояло количество букв в нем. Например, для предложения “57 ab:e r4” результатом будет “57 3ab:e 1r4”.

4) Отсортировать предложения по возрастанию длины последнего слова.

Все сортировки должны осуществляться с использованием функции стандартной библиотеки. Использование собственных функций, при наличии аналога среди функций стандартной библиотеки, запрещается.

Все подзадачи, ввод/вывод должны быть реализованы в виде отдельной функции

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

Предполагаемый объем пояснительной записки:
Не менее 15 страниц.

Дата выдачи задания: 02.11.2020
Дата сдачи реферата: 27.12.2020
Дата защиты реферата: 29.12.2020

Студентка гр. 0382 _____ Чегодаева Е.А.

Преподаватель _____ Жангиров Т.Р.

АННОТАЦИЯ

В курсовой работе была реализована программа для обработки заданного текста в зависимости от команд пользователя.

По ходу выполнения задач программа печатает подсказки для пользователя.

Для хранения текста память выделяется динамически. В работе используются стандартные библиотеки языка Си. Программа предусматривает сообщение о некорректном вводе команд. Также реализован вывод текста в текущем состоянии (после выполнения того или иного действия). По окончанию очищается динамически выделенная память. Для завершения работы программы предусмотрена отдельная команда.

Пример работы программы приведён в приложении А.

Код приведён в приложении В.

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ.

Цель работы: создать программу, принимающую текст и производящую с ним действия в соответствии с введённой командой. Реализовать корректный ввод и вывод, наличие подсказок по ходу выполнения, выход из программы. Программа должна включать работу с динамической памятью, стандартными библиотеками языка.

Задачи:

1. Реализовать корректный ввод пользовательского текста с удалением незначащих пробелов в начале предложений и табуляции;
2. Произвести очистку текста от повторяющихся предложений;
3. Реализовать оператор switch{};
4. Произвести операции:

   4.1. Вывести все предложения, в которых есть даты в формате “DD/MM/YYYY”, даты которые еще не наступили надо выделить красным цветом.
   4.2. Удалить все предложения, в которых каждое слово содержит нечетное количество букв.
   4.3. Преобразовать предложения так, чтобы перед каждым словом стояло количество букв в нем. Например, для предложения “57 ab:e r4” результатом будет “57 3ab:e 1r4”.
   4.4. Отсортировать предложения по возрастанию длины последнего слова.

5. Реализовать корректный вывод;
6. Очистить динамически выделенную память;
7. Реализовать интерфейс общения с пользователем.

В результате была создана программа, которая производит обработку текста, введённого пользователем. Для каждого действия печатаются советующие подсказки. Текст представляет собой динамический массив строк. Также производятся необходимые действия по очистке динамически выделенной памяти. В работе использовались стандартные библиотеки языка Си: stdio.h, stdlib.h, string.h, ctype.h, time.h. В ходе работы программы предусмотрен вывод отредактированного текста, выход из работы программы.

В тексте могут встречаться латинские символы, цифры, дроби, знак “/”, табуляция. Слова в тексте разделены пробелом, запятой или “, “. Каждое предложение заканчивается точкой. Ввод текста оканчивается символом переноса строки (“enter”).

Для корректной работы программы предполагается использование реальных дат (примеры несуществующих дат: 22/33/1111, 31/02/1999 и т.д.).

2. ЗАДАНИЕ.

Программе на вход подается текст (текст представляет собой предложения, разделенные точкой). Предложения - набор слов, разделенные пробелом или запятой, слова - набор латинских букв, цифр и символ ‘/’. Длина текста и каждого предложения заранее не известна.

Программа должна сохранить этот текст в динамический массив строк и оперировать далее только с ним.

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) Вывести все предложения, в которых есть даты в формате “DD/MM/YYYY”, даты которые еще не наступили надо выделить красным цветом.

2) Удалить все предложения, в которых каждое слово содержит нечетное количество букв.

3) Преобразовать предложения так, чтобы перед каждым словом стояло количество букв в нем. Например, для предложения “57 ab:e r4” результатом будет “57 3ab:e 1r4”.

4) Отсортировать предложения по возрастанию длины последнего слова.

Все сортировки должны осуществляться с использованием функции стандартной библиотеки. Использование собственных функций, при наличии аналога среди функций стандартной библиотеки, запрещается.

Все подзадачи, ввод/вывод должны быть реализованы в виде отдельной функции.

3. ХОД ВЫПОЛНЕНИЯ РАБОТЫ.

3.1. Ввод текста.

Для ввода текста реализован алгоритм, считывающий предложения посимвольно. В функции *main* объявляется переменная *after* (хранит количество предложений в тексте до первичной обработки). Далее динамически объявляются: массив *for_check,* хранящий сами строки - далее, хранящиеся в нём предложения будут направлены на первичную (обязательную) проверку; массив *sentense,* который будет хранить отдельное предложение, полученное из функции *get_text().* Далее посредством цикла *while*(), выходом из которого является встреча символа переноса строки (он же свидетельствует об окончании текста), происходит создание текста, при этом, на каждой итерации, увеличивая значение *after* и выделяется дополнительная память.

Функция *get_text():*

Объявляется переменная *length* для хранения количества символов, поступаемого предложения. Объявляется *sym* - для последующего ввода каждого символа. Объявляется *text* и выделяется память для него посредством *malloc,* хранит всё предложение. Происходит считывание первого символа. Далее посредством оператора *if* идёт проверка символа на пробел и табуляцию, если таковых нет - символ записывается в предложение и *length* увеличивается на 1. Далее реализован цикл *while,* выходом из которого является встреча символа окончания предложения. Внутри цикла поступают символы и, вместе с тем, каждый символ посредством *if* проверяется на табуляцию, если таковой нет - символ записывается в предложение, *length* увеличивается на 1 и выделяется необходимая память при помощи *realloc*. Затем в конце каждого предложения записывается символ конца строки. Функция возвращает предложение исходного текста без (возможно имеющихся) пробела в начале, табуляции и переноса строки.

3.2. Первичная обработка.

Первичная обработка заключается в очистке введённого текста от повторно встречающихся предложений (при посимвольном сравнении без учета регистра.)

При наличии нескольких (>2) одинаковых предложений программа сохраняет только его последнее вхождение.

Обработка реализована посредством вложенного цикла и использования флага. Перед самими циклами объявляется *text* - массив строк, с которым будут проводиться все последующие операции и выделяется необходимая память посредством *malloc* (далее эта память будет увеличиваться при помощи *realloc*). Также объявляется переменная *j*, которая будет хранить количество предложений в тексте после обработки. Задаётся флаг равный 1. Далее с помощью двух циклов *for()* реализуется сопоставление каждого предложения со всеми имеющимися предложениями, стоящими впереди него. Проверка выполнена посредством функции *strncasecmp().* Если встретилось приложение идентичное (без учёта регистра) текущему предложению → флаг становится равным нулю. Внутри внешнего цикла: если флаг остался равен 1, то предложение не имеет последующих повторений и, следовательно, оно записывается в text. На каждой итерации (на каждом новом предложении) значение флага возвращается к первоначальному.

3.3. Выбор команд.

Далее, с помощью оператора switch{}, реализован выбор дальнейшей команды пользователя. Благодаря циклу *while()* - программа продолжает запрашивать команды, пока не будет введён символ, обозначающий окончание работы программы. В работе реализованы 6 команд и сообщение о некорректности ввода, при его наличии. В зависимости от введённой команды программа реализует обработку текста.

При вводе ‘0’ - программа завершает работу и выводит сообщение об этом.

3.4. Операции команды 1.

При вводе ‘1’ - программа выполняет первую подзадачу при помощи цикла *for(),* посредством которого каждое предложение текста перенаправляется в функцию *data().*

Функция *data()*:

Для решения задачи создана система флагов и использована стандартная библиотека языка Си - *time.h.* С помощью функции time в переменную *rawtime* помещается текущее время, далее с помощью функции *localtime*, *rawtime* преобразует данное время в структуру, поделённую на год, месяц, день, час и т.д. Затем объявляются переменные: $i = 0$ (для движения по предложению), *flag* (флаг для определения: равен 2 - если перед нами не дата, равен 1 - если перед нами дата, которая уже была в прошлом, и равен 0 - если дата еще не наступила) изначально равен 2, переменная *consent* (флаг для определения того, есть ли дата в предложении) изначально равен 0, переменная *chek* (флаг для определения того, есть ли символ “/” в предложении - необходим для исключения вероятности зацикливания дальнейших циклов) изначально равен 0, переменная *begin* (индекс первого символа встречающегося слова), переменная *base* (чтобы избежать повторов при чтении символов), переменные *d, m, y* (далее будут хранить день, месяц и год соответственно в целочисленном типе), динамически объявляются *dd, mm, yy* (для хранения дня, месяца и года в типе *char*). Затем посредством цикла *while()*, выходом из которого является встреча символа окончания предложения, реализована проверка всех символов. Внутренним циклом ведётся поиск символа “/” - если такого вовсе не встретилось - переменная *chek* не меняет своё значение и, следовательно, внешний цикл принудительно завершается. Если в предложении встретился символ “/” - то следом идёт проверка (является ли

этот символ частью записи даты), для этого проверяется: есть ли, до символа 2 знака и 7 после, являются ли 2 знака до данного символа цифрами (посредством функции *isdigit()*), 2 после и еще 4 после предыдущих 2 (пропуская 1, так как, подразумевается, что на его месте стоит еще один необходимый символ “/”) и далее, если все условия соблюдены переменная *consent* получает значение 1 (что свидетельствует о том, что предложение содержит дату и при любом дальнейшем раскладе должно быть выведено). *Flag'y* так же присваивается значение 1. Далее реализована проверка самой даты - для определения необходимо ли окрашивать её цветом. День, месяц и год посимвольно добавляются в соответствующие массивы и затем (посредством функции *atoi()*) преобразуются в целочисленное значение и присваиваются соответствующим переменным. Затем, после данных преобразований, происходит поэтапное сравнение полученных данных с полями структуры *timeinfo* (с помощью команды “->”): первым сравнивается год - если значение, хранящееся в переменной *y* больше нынешнего года (значение получено посредством *tm_year*), то дату необходимо выделить и, следовательно, flag получает значение равное 0. Вместе с этим, необходимо учесть то, что при работе со структурой возвращается значение, при начале счёта с 1900-го года. Если даты равны - необходимо сравнить месяцы: значение полученной переменной m сравнивается с нынешним месяцем (значение получено посредством *tm _mon*), если месяц превышает текущий, то *flag'*у присваивается значение 0. Если месяцы тоже равны - необходимо сравнить дни: переменная *d* сравнивается с сегодняшним днём (значение получено посредством *tm_mday),* и если *d* превышает нынешнюю дату *flag'*у присваивается значение 0. При ином исходе значение *flag* остаётся неизменным. После этого (если *consent* =1) происходит сам вывод: если *flag* равен 1 - печатаются все символы начиная с *begin* и до конца даты, далее значение *i* возрастает на 8 (для того, что бы дальнейшая обработка продолжилась после данной даты), а так же осуществляется проверка на возможность окончания предложения в данном месте - если в строке сразу

после даты встречается символ окончания предложения, то к выводу дописывается точка (её не засчитает предыдущий цикл т.к. рассчитан только на запись даты); если *flag* равен 2 - реализован подобный предыдущему алгоритм, но с выделением самой даты красным цветом; если *flag* равен 0 - происходит посимвольный вывод строки, в данном случае *i* увеличивается на 1 на каждом шаге, здесь так же реализована проверка на окончание строки (но в данном случае без лишней записи точки, так как её запись войдёт в цикл). После любого из данных случаев *flag* получает значение 2 - для корректной дальнейшей обработки. Если *consent* остался равен 0 - то происходит дальнейшая посимвольная проверка предложения, так же значение *base* становится равным 0 (это необходимо для того, что бы, если в предложении всё же встретится дата, не потерять индекс начала предложения), реализована проверка на окончание строки (в таком случае - принудительный выход из внешнего цикла). Если в предложении встретилась хоть одна дата - печатается символ переноса строки (для того, что бы все предложения текста, в которых есть дата - печатались с новой стоки), иначе - функция возвращает значение 1. Окраска необходимых строк реализована посимвольно с использованием кодов, соответствующих красному цвету и возвращению к первоначальному цвету.

3.5. Операции команды 2.

При вводе ‘2’ - программа выполняет вторую подзадачу при помощи цикла *for(),* посредством которого каждое предложение текста перенаправляется в функцию *deletion().* Если флаг, который возвращает эта функция равен 0, то предложение встаёт на *l*-е место в *text'*е. Переменная *l,* объявленная ранее, изначально принимает значение 0 и увеличивается на 1 при встрече предложения, которое вернёт 0 в ранее названной функции. После окончания работы цикла переменная *j* принимает значение *l.*

Функция *deletion():*

Принимает одно предложение текста. Объявляются переменные *i* (для движения по предложению) и *count* (хранящую количество букв в каждом слове) - изначально равны ноль. Переменной *flag* присваивается значение 1. Далее посредством цикла *while()*, выходом из которого является встреча ".", реализована проверка всех символов. Затем, при помощи вложенного цикла *while()*, выходом из которого является встреча символа разделения слов или точки, происходит подсчёт букв (посредством *isalpha*), в каждом слове - и это значение проверяется на чётность - если в предложении встретилось хоть одно слово с чётным количеством букв *flag*'у присваивается значение 0 - что является, в дальнейшем, показателем того, что предложение необходимо оставить в тексте. При выполнении функции предусмотрено то, что помимо пробела и запятой слова могут разделяться ", ", вместе с этим исключено зацикливание. В результате строка возвращает значение переменной *flag*.

3.6. Операции команды 3.

При вводе '3' - программа выполняет третью подзадачу при помощи цикла *for()*, посредством которого каждое предложение текста меняется на предложение, возвращённое функцией *word_length()*, в которую поступает данное предложение.

Функция *word_length()*:

Принимает одно предложение текста. Объявляются переменные *i* (для движения по предложению *text*) и *m* (для движения по предложению *res*) - равные 0. Также объявляется *res* (который и будет хранить преобразованное предложение) и выделяется память, которая в дальнейшем будет увеличиваться. Далее посредством цикла *while()*, выходом из которого является встреча ".", реализована проверка всех символов. В начале каждой итерации объявляются переменные *begin = i* (который хранит индекс начала каждого слова) и *count = 0* (хранит количество букв каждого слова). Затем при

помощи вложенного цикла *while(),* выходом из которого является встреча символа разделения слов или точки, происходит подсчёт букв (посредством *isalpha*) в каждом слове. Если в “слове” нет букв (если к примеру это число или дата) в *res* записывается это “слово” без изменений и с соответствующими символами разделения слов в предложении. Если *count* не равен нулю, то его значение преобразуется в строку при помощи значений таблицы ASCII и добавляется в *res* (предусмотрено то, что количество букв в слове может быть двузначным), сразу за данным значением в *res* записывается всё слово с сохранением следующих символов разделения слов в предложении. После выхода из внешнего цикла в res добавляется символ переноса строки. При выполнении функции предусмотрено то, что помимо пробела и запятой слова могут разделяться “, “, вместе с этим исключено зацикливание. В результате функция возвращает преобразованную строку.

3.7. Операции команды 4.

При вводе ‘4’ - программа выполняет четвёртую подзадачу при помощи функции *qsort(),* в которую поступает *text* - массив указателей (*char***), то есть элемент массива будет указаетель на символ (*char**), в функцию *-comp*() - компаратор будут поступать два указателя на начало предложения (*char*),* при этом *qsort* оборачивает элементы, поступающие в компаратор ещё в один указатель и поэтому в функцию компаратор поступает *a=char**c.*

Функция *comp():*

Принимает 2 предложения текста. Переменные *len1* и *len2* для подсчёта длины соответствующих предложений и, следовательно, определения индекса последнего символа последнего слова. Объявляются переменные *i* и *j* для движения по первому и второму предложениям. Далее посредством *while(),* выходом из которого является встреча символа окончания предложения, в *len1* и *len2* записываются необходимые значения. Затем, для подсчёта букв в последнем слове, объявляются переменные *count1* и *count2*. Счёт реализован

посредством двух циклов *for(),* которые идут справа налево: от индекса конца последнего слова, до встречи символа разделения слов в предложении (то есть начала последнего слова). Внутри соответствующего цикла в переменные *count1* и *count2* записываются количество букв в слове (посредством *isalpha*). После выхода из обоих циклов реализуется сравнения: если *count1>=count2* то возвращается “1”, иначе “-1”. За счёт возвращаемого значения и реализуется сортировка предложения по заданному критерию.

3.8. Вывод текста.

При вводе ‘5’ - программа выполняет вывод текста в текущем состоянии при помощи цикла *for()* и функции *puts().* Данный пункт необходим для отслеживания состояния текста после совершения операция.

3.9. Очистка памяти.

Реализована очистка динамически выделенной памяти посредством функции *free().*

4. ТЕСТИРОВАНИЕ.

- Ввод:

  11/11/2222 hh 22/01/3333. 5/4 09/07/1090. hello. 33/2222/ jj. 22/01/2020 o/l 22/01/2021.

  - Команда 1

  Вывод:

  11/11/2222 hh 22/01/3333.

  5/4 09/07/1090.

  22/01/2020 o/l 22/01/2021.

Программа работает верно.

- Ввод:

  Koshka, kot. Dog in home. Kot Kot Kot. plot in reka.

  - Команда 2

  Вывод:

  Koshka, kot.

  Dog in home.

  plot in reka.

Программа работает верно.

- Ввод:

  aaa, aaaa. Bb3bb, ccc. 12/12/2012 hh. l-l-l. ddd, dhd.

  - Команда 3

  Вывод:

  3aaa, 4aaaa.

  4Bb3bb, 3ccc.

  12/12/2012 2hh.

  3l-l-l.

  3ddd, 3dhd.

Программа работает верно.

- Ввод:

  FFF 2 llllll. FFF 4 k. GGG 5/4 HG. DDDDD yy5ttt. 5544. 876/65 ss.

  - Команда 4

  Вывод:

  5544.

  FFF 4 k.

  876/65 ss.

  GGG 5/4 HG.

  DDDDD yy5ttt.

  FFF 2 llllll.

Программа работает верно.

- Ввод:

  Hello. I am Santa. HO-HO-HO-HO. We wish you a Merry Christmas. We wish you a Merry Christmas and a Happy New Year. 31/12/2020 and 01/12/2021.

  - Команда 3
  - Команда 2
  - Команда 4

  Вывод:

  31/12/2020 3and 01/12/2021.

  2We 4wish 3you 1a 5Merry 9Christmas 3and 1a 5Happy 3New 4Year.

  1I 2am 5Santa.

  8HO-HO-HO-HO.

  2We 4wish 3you 1a 5Merry 9Christmas.

Программа работает верно.

5. ЗАКЛЮЧЕНИЕ.

В процессе курсовой работы была создана корректно работающая программа по обработке строк на языке Си. Программа предоставляет пользователю возможность ввода текста и его дальнейшую обработку в зависимости от вводимых команд. В программе предусмотрена печать сообщений для пользователя, в зависимости от ситуации. Исключено возникновение ошибок при некорректном вводе. Программа будет запрашивать команды, и выполнять соответствующие действия до тех пор, пока не будет введён символ окончания работы программы. Для реализации всех задач были применены динамические массивы и стандартные библиотеки языка Си. Память, выделенная динамически, в процессе работы программы - очищается.

6. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.

1. Cplusplus URL: https://www.cplusplus.com.

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

ПРИЛОЖЕНИЕ А
ПРИМЕР РАБОТЫ ПРОГРАММЫ

```
Пожалуйста, введите текст, для дальнейшей обработки.
(Для завершения ввода необходимо нажать 'enter')
```

Ввод:

Hello. My na-me is Tom. Every day I drink 2/3 of a cube of coffee. I 5was in London on 05/12/2005. I am plan/ning to g1o to Paris on 01/22/2021. hello. This event will happen on 12/09/2022 and 12/01/3022. My house code is 22/1111. I have a lot of 2friends.Every day5 at 6 in the morning I run. This is a very interesting and informative story.

```
Для совершения необходимого действия выберете соответсвующую команду:
'1' - Вывести все предложения в которых есть даты в формате
“DD/MM/YYYY”, даты которые еще не наступили будут выделены красным
цветом;
'2' - Удалить все предложения в которых каждое слово содержит
нечетное количество букв;
'3' - Преобразовать предложения так, чтобы перед каждым словом стояло
количество букв в нем;
'4' - Отсортировать предложения по возрастанию длины последнего слова;
'5' - Вывод текста в текущем состоянии;
'0' - Выход из программы.
Введите команду: 7
Данные некорректны
Введите команду: 2
Введите команду: 5
My na-me is Tom.
Every day I drink 2/3 of a cube of coffee.
I 5was in London on 05/12/2005.
I am plan/ning to g1o to Paris on 01/22/2021.
This event will happen on 12/09/2022 and 12/01/3022.
My house code is 22/1111.
I have a lot of 2friends.
Every day5 at 6 in the morning I run.
This is a very interesting and informative story.
Введите команду: 3
Введите команду: 5
2My 4na-me 2is 3Tom.
5Every 3day 1I 5drink 2/3 2of 1a 4cube 2of 6coffee.
1I 35was 2in 6London 2on 05/12/2005.
1I 2am 8plan/ning 2to 2g1o 2to 5Paris 2on 01/22/2021.
4This 5event 4will 6happen 2on 12/09/2022 3and 12/01/3022.
2My 5house 4code 2is 22/1111.
1I 4have 1a 3lot 2of 72friends.
5Every 3day5 2at 6 2in 3the 7morning 1I 3run.
4This 2is 1a 4very 11interesting 3and 11informative 5story.
Введите команду: 4
Введите команду: 5
```

```
1I 2am 8plan/ning 2to 2g1o 2to 5Paris 2on 01/22/2021.
1I 35was 2in 6London 2on 05/12/2005.
2My 5house 4code 2is 22/1111.
4This 5event 4will 6happen 2on 12/09/2022 3and 12/01/3022.
2My 4na-me 2is 3Tom.
5Every 3day5 2at 6 2in 3the 7morning 1I 3run.
4This 2is 1a 4very 11interesting 3and 11informative 5story.
5Every 3day 1I 5drink 2/3 2of 1a 4cube 2of 6coffee.
1I 4have 1a 3lot 2of 72friends.
Введите команду: 3
Введите команду: 5
11I 22am 88plan/ning 22to 22g1o 22to 55Paris 22on 01/22/2021.
11I 335was 22in 66London 22on 05/12/2005.
22My 55house 44code 22is 22/1111.
44This 55event 44will 66happen 22on 12/09/2022 33and 12/01/3022.
22My 44na-me 22is 33Tom.
55Every 33day5 22at 6 22in 33the 77morning 11I 33run.
44This 22is 11a 44very 1111interesting 33and 1111informative 55story.
55Every 33day 11I 55drink 2/3 22of 11a 44cube 22of 66coffee.
11I 44have 11a 33lot 22of 772friends.
Введите команду: 8
Данные некорректны
Введите команду: 0
Работа программы завершена.
```

ПРИЛОЖЕНИЕ Б
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: cw.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <time.h>
#define RED  "\033[0;31m"
#define NONE "\033[0m"

char* get_text(){
    int length = 0;
    char sym = 0;
    char *text = malloc(1*sizeof(char));
    sym = getchar();
    if(sym!=' ' && sym!='\t'){
        text[length] = sym;
        length++;
    }
    while (sym != '.' && sym != '\n'){
        sym = getchar();
        if (sym!='\t' && sym!='\n'){
            text[length] = sym;
            length++;
            text= realloc(text,(length+1)*sizeof(char*));
        }
    }
    text[length] = '\0';
    return text;
}
char* word_length(char* text){
    int i=0;
```

```
    int m=0;
    char *res= malloc(1*sizeof(char));
    while (text[i]!='.') {
        int begin = i;
        int count = 0;
        while (1)
        {
            i++;
            if (text[i] == ' ' || text[i] == ','|| text[i] ==
'.') {
                break;
            }
        }
        if (text[i+1] == ' '){
            i++;
        }
        for(int l=begin;l<i; l++)
            if (isalpha(text[l])) {
                count++;
            }
        if (count==0){
            for(int k = begin; k<=i; k++ ){
                res = realloc(res,(m*2)*sizeof(char));
                res[m]=text[k];
                m++;
            }
        }
        else{
            char *base=malloc(10*sizeof(char));
            int u=0;
            while (count>0)
            {
                base[u]=48+(count%10);
                count=count/10;
                u++;
```

```
            }
            base[u]='\0';
            for (u=0;u<strlen(base)/2;u++)
            {
                char temp;
                temp=base[u];
                base[u]=base[strlen(base)-1-u];
                base[strlen(base)-1-u]=temp;
            }
            res[m]=base[0];
            m++;
            if (strlen(base)>1){
                res[m]=base[1];
                m++;
            }
            free(base);
            res = realloc(res,(m*2)*sizeof(char));
            for(int k = begin; k<=i; k++ ){
                res = realloc(res,(m*2)*sizeof(char));
                res[m]=text[k];
                m++;
            }

        }
        i++;
        if (text[i-1] == '.') {
            break;
        }
    }
    res = realloc(res,(m*2)*sizeof(char));
    res[m]="\0";
    return res;
}
int data(char *text){
    time_t rawtime;
```

```
    time(&rawtime);
    struct tm * timeinfo;
    timeinfo = localtime(&rawtime);
    int i = 0;
    int flag = 2;
    int base=0;
    int consent = 0;
    int chek=0;
    int d = 0;
    int m = 0;
    int y = 0;
    int begin= 0;
    char* dd = malloc(3*(sizeof(char)));
    char* mm = malloc(3*(sizeof(char)));
    char* yy = malloc(5*(sizeof(char)));
    while (text[i]!='.') {
        begin=i-base;
        while (text[i] != '.') {
            if (text[i] == '/') {
                chek=1;
                break;
            }
            if (text[i] == '.') {
                break;
            }
            i++;
        }
        if(chek==0){
            break;
        }
        if ((strlen(text) >= i + 7) && (strlen(text) - 2 - i >=
0)) {
            if (isdigit(text[i - 2]) && isdigit(text[i - 1])) {
                if (isdigit(text[i + 1]) && isdigit(text[i + 2]))
{
```

```
                    if (isdigit(text[i + 4]) && isdigit(text[i +
5]) && isdigit(text[i + 6]) && isdigit(text[i + 7])) {
                        consent=1;
                        flag=1;
                        dd[0] = text[i - 2];
                        dd[1] = text[i - 1];
                        dd[2] = "\0";
                        d = atoi(dd);
                        mm[0] = text[i + 1];
                        mm[1] = text[i + 2];
                        mm[2] = "\0";
                        m = atoi(mm);
                        yy[0] = text[i + 4];
                        yy[1] = text[i + 5];
                        yy[2] = text[i + 6];
                        yy[3] = text[i + 7];
                        yy[4] = '\0';
                        y = atoi(yy);
                        if ((y - 1900) > (timeinfo->tm_year)) {
                            flag = 0;
                        }
                        if ((y - 1900) == (timeinfo->tm_year)) {
                            if ((m - 1) > (timeinfo->tm_mon)) {
                                flag = 0;
                            }
                            if ((m - 1) == (timeinfo->tm_mon)) {
                                if ((d) > (timeinfo->tm_mday)) {
                                    flag = 0;
                                }
                            }
                        }
                    }
                }
            }
            if (flag==2){
```

```
            i=i+1;
        }
    }

    if (consent == 1) {
        if (flag==2){
            for (int n = begin; n <= i; n++) {
                printf("%c", text[n]);
            }
            i=i+1;
            if (text[i-1] == '.') {
                break;
            }
        }

        if (flag == 1) {
            for (int g = begin; g <= i + 7; g++) {
                printf("%c", text[g]);
            }
            i = i + 8;
            base=0;
            if (text[i] == '.') {
                printf(".");
                break;
            }
        }
        if (flag == 0) {
            for (int g = begin; g < i - 2; g++) {
                printf("%c", text[g]);
            }
            for (int l = i - 2; l <= i + 7; l++) {
                printf("%s%c%s", RED, text[l], NONE);
            }
            i = i + 8;
            base=0;
```

```
                if (text[i] == '.') {
                    printf(".");
                    break;
                }
            }
            flag = 2;
        }
        else{
            i++;
            base=i;
            if (i > strlen(text)) {
                break;
            }
        }

    }
    free(dd);
    free(mm);
    free(yy);
    if (consent==1){
        printf("\n");
    }
    else{
        return 1;
    }
}

int deletion(char *text){
    int i=0;
    int count = 0;
    int flag = 1;
    while (text[i]!='.'){
        count=0;
        while(1){
```

```
            if (text[i]==' ' | text[i]==',' | text[i]=='.'){
                break;
            }
            if(isalpha(text[i])) {
                count++;
            }
            i++;
        }
        if(text[i+1]==' '){
            i++;
        }
        if ((count)%2==0){
            flag = 0;
        }
        i++;
        if (text[i-1] == '.') {
            break;
        }
    }
    return flag;

}
int comp(const void* it1, const void* it2){
    int len1 = 0;
    int len2 = 0;
    int i = 0;
    int j = 0;
    char** item1=(char **)it1;
    char** item2=(char **)it2;
    char* text1=*item1;
    char* text2=*item2;
    while (*(text1+len1)!='.'){
        len1++;
    }
    while (*(text2+len2)!='.'){
```

```
        len2++;
    }
    len1=len1-1;
    len2=len2-1;
    int count1=0;
    int count2=0;
    for(i = len1; i>=0; i--){
        if(isalpha(text1[i])){
            count1++;
        }
        if ((text1[i])==' ' | (text1[i])==','){
            break;
        }
    }
    for(j = len2; j>=0; j--){
        if(isalpha(text2[j])){
            count2++;
        }
        if ((text2[j])==' ' | (text2[j])==','){
            break;
        }
    }
    if (count1>=count2){
        return 1;
    }
    else {
        return -1;
    }
}

int main(){
    printf("Пожалуйста, введите текст, для дальнейшей
обработки.\n"
           "(Для завершения ввода необходимо нажать 'enter')\n
");
```

```
    int after= 0;
    char* sentense;
    char** for_check = malloc(1*sizeof(char*));
    while (strcmp(sentense,"\n")){
        sentense=get_text();
        for_check[after] = sentense;
        after++;
        for_check = realloc(for_check, (after+1)*sizeof(char*));
    }
    int j = 0;
    char** text = malloc((after)*sizeof(char*));
    for (int i = 0; i<=after-2; i++){
        int k=1;
        for (int h = i+1; h<=after-1; h++ ){
            if
(!strncasecmp(for_check[i],for_check[h],strlen(for_check[i]))){
                k=0;
                break;
            }
        }
        if (k==1){
            text[j]=for_check[i];
            j++;
        }
    }
    int n;
    int l=0;
    printf("Для совершения необходимого действия выберете
соответсвующую команду:\n"
           "'1' - Вывести все предложения в которых есть даты в
формате\n"
           "“DD/MM/YYYY”, даты которые еще не наступили будут
выделены красным\n"
           "цветом;\n"
           "'2' - Удалить все предложения в которых каждое слово
```

```
содержит \n"
           "нечетное количество букв;\n"
           "'3' - Преобразовать предложения так, чтобы перед
каждым словом стояло\n"
           "количество букв в нем;\n"
           "'4' - Отсортировать предложения по возрастанию длины
последнего слова;\n"
           "'5' - Вывод текста в текущем состоянии;\n"
           "'0' - Выход из программы.\n");
    while(1) {
        printf("Введите команду: ");
        scanf("%d", &n);
        switch (n) {
            case 0:
                free(for_check);
                free(text);
                printf("Работа программы завершена.\n");
                return 0;
            case 1:
                for (int i = 0; i < j; i++) {
                    data(text[i]);
                }
                break;
            case 2:
                l = 0;
                for (int i = 0; i < j; i++) {
                    if (deletion(text[i]) == 0) {
                        text[l] = text[i];
                        l++;
                    }
                }
                j=l;
                break;
            case 3:
                for (int i = 0; i < j; i++) {
```

```
                    text[i] = word_length(text[i]);
                }
                break;
            case 4:
                qsort(text, j, sizeof(char *), comp);
                break;
            case 5:
                for (int i = 0; i < j; i++) {
                    puts(text[i]);
                }
                break;
            default:
                printf("Данные некорректны\n");
                break;
        }
    }
}
```