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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Работа с текстом

Студент гр. 5341 _____ Бейзер Л.В.

Преподаватель _____ Государкин Я.С.

Санкт-Петербург

2025

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент: Бейзер Л.В.

Группа: 5341

Тема работы:

Работа с текстом

Исходные данные:

Текст представляет собой предложения, разделенные точкой.

Предложения – набор слов. Слова состоят из латинских букв, цифр и символа '-'. Остальные символы являются разделителями между словами. Длина текста и каждого предложения заранее не известна. Ввод данных в программе в стандартный поток ввода: stdin.

Содержание пояснительной записки:

«Содержание», «Введение», «Заключение», «Список использованных источников», «Приложение А. Примеры работы программы».

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания: 17.10.2025

Дата сдачи реферата: 26.11.2025

Дата защиты реферата: 28.11.2025

Студент _____ Бейзер Л.В.

Преподаватель _____ Государкин Я.С.

АННОТАЦИЯ

Программа должна найти и удалить все повторно встречающиеся предложения (сравнивать их следует посимвольно, но без учета регистра).
Программа должна выполнить одно из введенных пользователем действий и завершить работу:

1) Вывести все предложения в которых есть дата с текущим годом и месяцем. (дата записывается в формате DD/MM/YYYY)

2) Отсортировать предложения по увеличению минимальной даты в них. Если в предложении нет даты, то следует считать, что "мнимая минимальная дата" == ∞. В случае равенства сохранить порядок вхождения.

3) Удалить все предложения в которых все даты относятся к 19 веку.

4) Для каждого предложения вывести самую раннюю и позднюю дату. Предложения выводит не нужно. Если предложение не содержит дату, то его не нужно выводить. Для каждого предложения последовательно выводятся строки формата "Earliest date: <date>" и "Latest date: <date>". Если в предложении одна дата -- то она одновременно и ранняя и поздняя.

В случае вызова справки (опция 5) текст на вход подаваться не должен, во всех остальных случаях после выбора опции должен быть считан текст.
Каждая подзадача вынесена в отдельную функцию в разных файлах. Для сборки проекта и альтернативных возможностей используется Makefile.
При выводе каждое предложение печатается в отдельной строке (перевод строки).

SUMMARY

The program must find and delete all duplicate sentences
(they should be compared character by character, but without regard to case).
The program must perform one of the actions entered by the user and
then terminate:

1) Display all sentences that contain a date with the current year and month. (the date is written in DD/MM/YYYY format)

2) Sort the sentences by increasing the minimum date in them.

If there is no date in the sentence, then it should be considered that the "imaginary minimum date" == ∞. In case of equality, keep the order of
occurrence.

3) Delete all sentences in which all dates refer to the 19th century.

4) For each sentence, output the earliest and latest dates.

There is no need to output the sentences. If a sentence does not contain a date, then it does not need to be output. For each sentence, the following lines are output in sequence:

"Earliest date: <date>“ and ”Latest date: <date>".

If there is only one date in the sentence, it is both the earliest and
the latest. Example: original sentence "15/08/1722 UOEAkpDq
06/08/1832 oFtEgRt 05/10/1972." Two lines are displayed: "Earliest date:
15/08/1722“ and ”Latest date: 05/10/1972".

When calling up the help (option 5), no text should be entered; in all other cases, the text should be read after selecting the option.

Each subtask is placed in a separate function in different files. Makefile is used to build the project and alternative options.

When outputting, each sentence is printed on a separate line (line break).

СОДЕРЖАНИЕ

	Введение	7
1.	Структуры	8
1.1.	Структура str_inform	8
1.2.	Структура strs_all	8
1.3.	Структура dataStrs	9
1.4.	Структура comms_opts	9
1.5.	Перечисление Boolean	9
1.6.	Перечисление ErrCode	9
1.7.	Перечисление Months	10
2.	Функции/файлы	10
2.1.	Файл main.c	10
2.1.1	Функция main	11
2.1.2	Функция print_sents	11
2.1.3	Функция help_comm	11
2.1.4	Функция free_strs	12
2.2.	Файл readaform.c	12
2.2.1	Функция read_a_formt	12
2.2.2	Функция parse_dates	12
2.2.3	Функция check_dates	12
2.2.4	Функция incr_str_arr	12
2.2.5	Функция cmp_dates	13
2.2.6	Функция malloc_ptr	13
2.2.7	Функция realloc_ptr	13
2.2.8	Функция init_strs_mem	13
2.2.9	Функция expand_str_buffer	13
2.2.10	Функция find_ellipsis	13
2.2.11	Функция check_and_reset_duplicate	13

2.2.12 Функция resize_mem 13
2.2.13 Функция update_dates 13
2.2.14 Функция init_date_storage 13
2.2.15 Функция add_date 13
2.3. Файл control.c 14
2.3.1 Функция inp_oup_ctrl 14
2.4. Файл findcntdel.c 14
2.4.1 Функция find_cnt_del 14
2.5. Файл currdate.c 14
2.5.1 Функция curr_date 14
2.7. Файл maxmindate.c 15
2.7.1 Функция max_min_date 15
2.7.2 Функция cmp_sents 15
3. Сборка проекта (Makefile) 16
4. Заключение 18
5. Список использованных источников 19
6. Приложение А. Инструкция запуска программы. 20
7. Приложение Б. Примеры работы программы 22

ВВЕДЕНИЕ

Курсовая работа представляет собой разработку программы для обработки текста. Проект реализуется на язяыке C, с соблюдением требований к отказоустойчивости. Особенности реализации описаны в общих сведениях и описании работы функций. Примеры использования CLI в Приложении B (см. Приложение B). Инструкция по запуску программы в приложении A (см. Приложение A). Программа корректно обрабатывает ошибки с выводом понятных сообщений пользователю

Цель работы: написать программу на языке C, которая получает аргументы переданные через командную строку в поток stdin, считывает и обрабатывает текст требуемым образом/опцией. Для этого требовалось написать:

- Функцию получения аргументов из командной строки;
- Функции валидации аргументов;
- Функции для хранения текста и его минимальную обработку;
- Функции обработки текста по опциям.

1. СТРУКТУРЫ

Чтобы облегчить написание и чтение кода, было принято решение реализовать и использовать основные структуры: str_inform; strs_all; comms_opts, вспомогательная: dataStrs и перечисление: Boolean.

1.1. Структура str_inform

Структура представляет информацию об тексте и дат в нём.

Она содержит следующие поля:

- char* str: указатель на строку текущего предложения обработки;
- size_t lenstr: длина строки текущего предложения обработки;
- dateStrs* dates_str: Указатель на все вхождения дат в строку;
- dateStrs* minDate: Указатель на минимальную дату в строке;
- dateStrs* minDate: Указатель на максимальную дату в строке;
- size_t inf: флаг наличия бесконечности (по условию ТЗ);
- size_t date_c: Количество вхождений дат в строку;
- size_t index_str: Индекс первоначального вхождения предложения (до форматирования).

1.2. Структура strs_all

Эта структура предназначена для хранения структуры предложений текста и их количество.

Содержит следующие поля:

- str_inform* str_curr: Указатель на структуру строк/предложений;
- size_t total_count: Количество считанных строк;

1.3. Структура dataStrs

Вспомогательная структура, которая предназначена для хранения общей служебной информации о датах в предложении/строке.

Содержит следующие поля:

- int day: День (1 — 31);
- int month: Месяц (1 — 12);
- int year: (1 — 9999);

1.4. Структура comms_opts

Эта структура предназначена для хранения желаемой опции от пользователя.

Содержит следующее поле:

- size_t opts : 3: Номер функции побитового поля (до 7 максимум);

1.5. Перечисление Boolean

Перечисление для хранения булевых знаечний

Содержит следующие поля:

- FALSE: = 0;
- TRUE: = 1;

1.6. Перечисление ErrCode

Перечисление для хранения кодов возвратов.

Но по ТЗ используется только SUCCESS в задаче.

Содержит следующие поля:

- SUCCESS: = 0;
- ERR_MALLOC: = 1;
- ERR_REALLOC: = 2;
- ERR_INVALID_ARG: = 3;

1.7. Перечисление Months

Перечисление для хранения кодов возвратов.

Содержит следующие поля:

- JAN = 1,
- FEB = 2,
- MARCH = 3,
- APR = 4,
- MAY = 5,
- JUNE = 6,
- JUL = 7,
- AUGS = 8,
- SEPT = 9,
- OCTOB = 10,
- NOVEM = 11,
- DECEM = 12,

2. ФУНКЦИИ

2.1 Файл main.c

2.1.1 Функция main

Функция является вступительной функцией, располагается в файле main.c, и начинает запуск других функций. Возвращает на выходе целочисленный код возврата приписанный к enum структуре.

Внутри функции вызываются следующие основные шаги:

1) Сначала выводится информация об авторе исходного кода и номер варианта.

2) Происходит инициализация структуры comms_opts и затем вызов функции inp_oup_ctrl (control.c) для считывания номера опции из stdin потока данных.

3) После успешного считывания выделяется (если пользователь не выбрал опцию 5) память для первичной структуры strs_all и начинается считывания текста из потока stdin с помощью функции read_a_format (readaform.c).

4) После успешной первичной обработки/считывания текста вызываются опции от 1 до 5

5) Если опции корректно считаны, то под конец программа выволит предложения (по треб опций) с помощью print_sents и затем программа завершается штатно очистив выделенную динамическую память с помощью free_strs.

Возвращает код статуса

2.1.2 Функция print_sents

Данная функция выводит предложения после обработки функциями опций 2, 3.

2.1.3 Функция help_comm

Данная функция выводит справочную информацию при вызове опции 5

2.1.4 Функция free_strs

Данная функция освобождает память выделенную динамически после успешного выполнения/ошибок.

2.2. Файл readaform.c

2.2.1 Функция read_a_format

Данная функция отвечает за первичную обработку предложений (проверка на повторные вхождения), сохранения предложений/строк, поиск дат в предложениях.

Алгоритм работы функции:

1) для начала происходит инициализация структур строк по умолчанию.

2) затем происходит посимвольное чтение из stdin потока до «EOF» или «\n\n» с одновременной проверкой на вхождения повторов с помощью strcasecmp.

3) далее отработка функции parse_dates

Возвращает код статуса.

2.2.2 Функция parse_dates

Эта функция является вспомогательной для read_a_format функции.

Её цель:

После успешного считывания и отсеивания повторов начинается поиск и запись минимальной/максимальной/общих дат/ы используя собственный компаратор cmp_dates и функцию проверки дат check_dates.

Возвращает код статуса.

2.2.3 Функция check_dates

Эта функция отвечает за то, что дата является валидной (включая проверку високосного года). Возвращает код статуса

2.2.4 Функция incr_str_arr

Данная функция отвечает за перераспределение памяти (экономия). Возвращает код статуса

2.2.5 Функция cmp_dates

Эта функция является компаратором для сравнения дат

Возвращает число с ведущим знаком сравнения

2.2.6 Функция malloc_ptr

Данная функция отвечает за выделение памяти под массив и обработкой ошибки выделения в случае провала.

2.2.7 Функция realloc_ptr

Данная функция отвечает за перераспределение памяти под массив и обработкой ошибки перераспределения в случае провала.

2.2.8 Функция init_strs_mem

Инициализация хранилища строк.

2.2.9 Функция expand_str_buffer

Расширение буфера строки.

2.2.10 Функция find_ellipsis

Обработка троеточий в строке.

2.2.11 Функция check_and_reset_duplicate

Обработка дубликатов в тексте

2.2.12 Функция resize_mem

Сужение выделенной, но не использованной памяти.

2.2.13 Функция update_dates

Обновление max/min дат.

2.2.14 Функция init_date_storage

Инициализация хранилища дат.

2.2.15 Функция add_date

Запись даты предложения в его структуру.

2.3. Файл control.c

2.3.1 Функция inp_oup_ctrl

Эта функция отвечает за чтение номера опции из stdin и запись в структуру comms_opts в поле opts.

2.4. Файл findcntdel.c

2.4.1 Функция find_cnt_del

Эта функция отвечает за отсеивание дат, которые входят в промежуток 19 века.

Основана на принципе того, что с помощью структуры strs и указателя в ней связанного с другой структурой на указатель дат вытаскивает даты из предложения в формате dd/mm/yyyy и если оказывается истинной все предложений, то смещает все след указатели массива на прошлые. Под конец мусорные указатели урезаются благодаря realloc.

2.5. Файл currdate.c

2.5.1 Функция curr_date

Функция отвечает за нахождение предложений с вхождением текущей датой запуска

Основана на использование библиотеки time.h для того, чтобы можно было работать с текущей датой:

Получает текущее время и затем закидывает его с структуру библиотеки tm. И затем с помощью sscanf по отдельным аргументам Далее благодаря strftime вытаскивает из структуры нужные поля и с форматированием вставляет в char поле в формате dd/mm/yyyy

Затем начинает сравнение из полей dates_str с текущими аргументами. Если истинно, то выводит предложение

2.7. Файл maxmindate.c

2.7.1 Функция max_min_date

Текущая функция отвечает за нахождение поздней и ранней даты в поданном предложении/строке.

Работает с заполненными полями в указатели внутри strs благодаря функции read_a_form.

Реализована по средствам использования функции qsort из stdlib.h.

2.7.2 Функция cmp_sents

Данная функция отвечает за сравнение данных (компаратор для дат)

Возвращает целочисленное значение с знаком его вхождения. От первого ко второму.

Замечание: случай, когда даты равны сравнивается по индексам вхождений из-за особенностей qsort.

3. СБОРКА ПРОЕКТА
MAKEFILE

Makefile проекта находится в *[1]/src папке и имеет следующий функционал (исполнение происходит из папки src соответственно):

1) make / make all: данная команда собирает релизную версию программы с соответствующими флагами;

2) make debug: команда собирает отладочную(дебаг) версию;

3) make test: команда собирает дебаг версию и затем проверяют на входных тестах с помощью простенького sh скрипта;

4) make doxy: команда благодаря утилите Doxygen создаёт кросс платформенную документацию с различными параметрами/настройками (формат html внутри */docs/html;

5) make doxy_pdf: команда аналогичная make doxy, но в формате PDF (LaTeX);

6) make clean: команда очищает все сборочные/отладочные файлы;

Флаги:

- релизной версии: -Wall -Wextra -Werror -MMD -MP -Iinclude
- дебаг версии: -fno-common -std=c11 -g -fsanitize=address

[1] * - корень проекта

Примечание: MMD и MP флаг требуется исключительно для корректной работы Makefile (линковка объектных файлов/их проверка изменений/статус заголовков)

Директории:

*/build — файлы сборки исходного кода

*/docs — документация Doxygen/отчёт

*/src — исходные файлы

*/src/include — заголовки

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы были изучены теоретические основы и практические методы обработки строковых данных в языке программирования C. Были рассмотрены алгоритмы парсинга текста, извлечения данных и форматированного вывода.

В результате проделанной работы было спроектировано и реализовано программное приложение для обработки текста. В частности, были выполнены следующие задачи:

1. Реализована работа с динамической памятью: обеспечено корректное выделение и освобождение ресурсов для обработки текстов произвольного размера без утечек памяти.
2. Изучены и применены стандартные библиотеки: использованы возможности библиотек `<string.h>` и `<stdlib.h>` для токенизации строк, их сравнения (в том числе без учета регистра) и конвертации данных.
3. Реализованы сложные алгоритмы обработки: написаны функции для сортировки предложений с использованием стандартной функции `qsort` и пользовательских компараторов, а также функции для специфических преобразований текста ("сшивание" предложений, фильтрация по датам и содержимому).
4. Проведено тестирование: работоспособность программы проверена на различных наборах входных данных, включая пограничные случаи.

Полученные в ходе тестирования результаты полностью соответствуют требованиям технического задания. Цель курсовой работы достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Керниган, Б. У. Язык программирования С / Б. У. Керниган, Д. М. Ритчи ; пер. с англ. — 2-е изд., перераб. и доп. — Москва : Вильямс, 2019. — 304 с.

2. Википедия : свободная энциклопедия [Электронный ресурс]. – https://ru.wikipedia.org

3. Программирование на С. Практические занятия. Первый семестр [Электронный ресурс]. – https://e.moevm.info/course/view.php?id=8

ПРИЛОЖЕНИЕ А
ИНСТРУКЦИЯ ЗАПУСКА ПРОГРАММЫ

Минимальные зависимости:

- GCC

Рекомендуемые зависимости (для Makefile):

- Git
- Make
- clang-format
- Doxygen

Перед запуском программы обязательными являются предустановленные минимальные зависимости. Рекомендуемые по желанию (надобности). Если нет Git, то скачивание проекта вручную и пункты с git пропускаются.

Сборка программы

Make:

Для запуска через Make достаточно перейти в директорию проекта (перед этим его скачать) и затем в дочернею папку src и оттуда вызвать команду make/make all

1. cd ~
2. git clone https://github.com/moevm/pr-2025-5341.git
3. cd pr-2025-5341
4. git switch
5. cd Beizer_Lev_cw/src
6. make
7. ./cw

Manual:

1. cd ~
2. git clone https://github.com/moevm/pr-2025-5341.git
3. cd pr-2025-5341
4. git switch
5. cd Beizer_Lev_cw/src
6. gcc -Wall -Wextra -Werror -Iinclude -O1 *.c -o cw
7. ./cw

ПРИЛОЖЕНИЕ Б

ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

На рисунках А.1–А.7 представлены результаты работы программы.

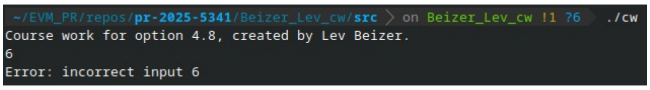

Иллюстрация 1: Подан некорректный аргумент

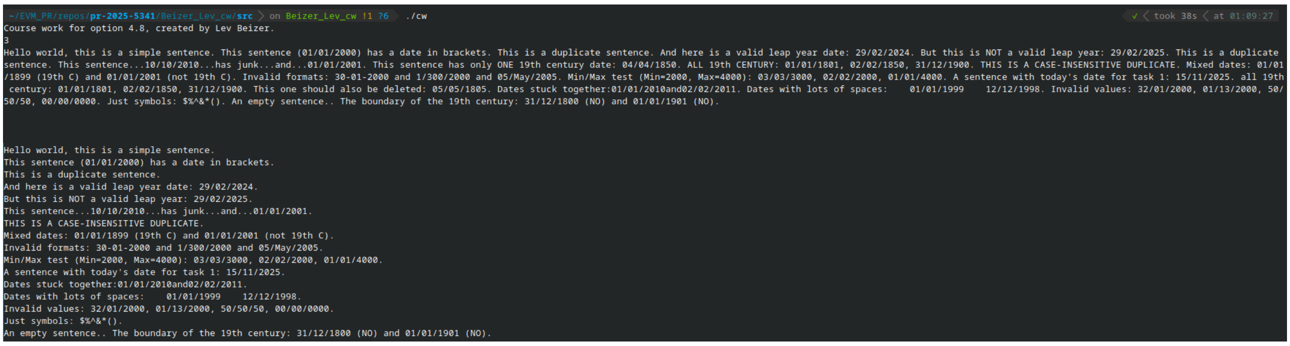

Иллюстрация 2: Пример работы опции 3

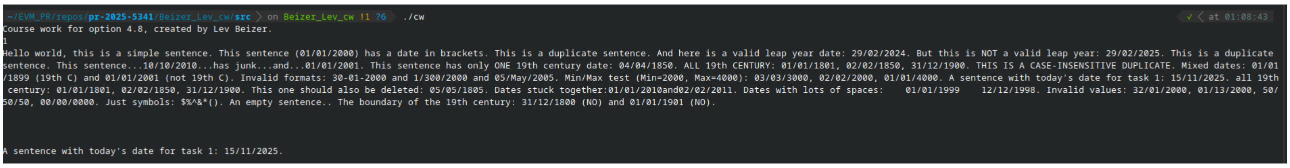

Иллюстрация 3: Пример работы опции 1

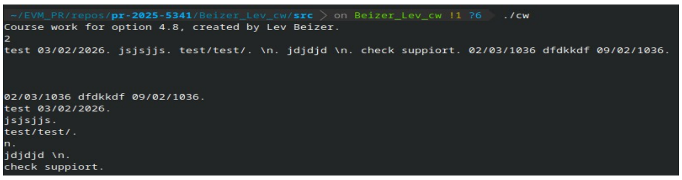

Иллюстрация 4: Пример работы опции 2

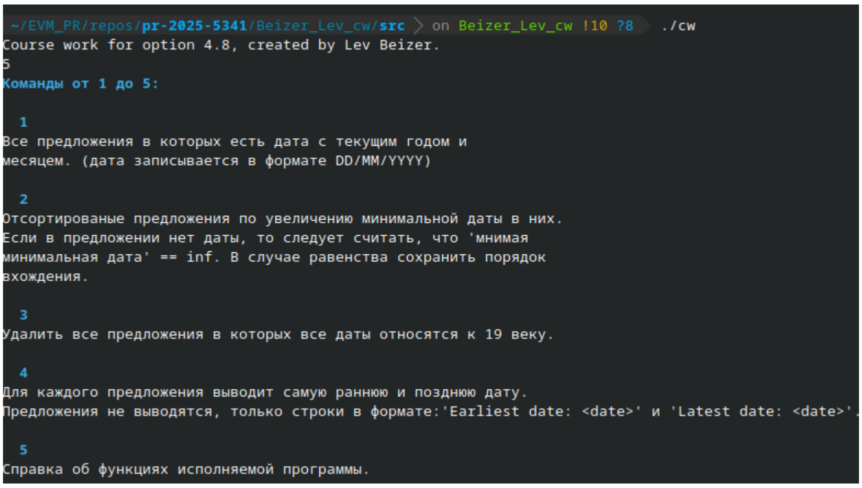

Иллюстрация 5: Пример работы опции 5

Иллюстрация 6: Пример работы опции 4

Иллюстрация 7: Подана пустая строка