

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" $\equiv \infty$. 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 выводятся 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" $\equiv \infty$. 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 UOEApDq 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	6
1. Структуры	7
1.1. Структура str_inform	7
1.2. Структура str_all	7
1.3. Структура dataStrs	8
1.4. Структура comms_opts	8
1.5. Перечисление Boolean	8
1.6. Перечисление ErrCode	8
2. Функции/файлы	9
2.1. Файл main.c	9
2.1.1 Функция main	9
2.1.2 Функция print_sents	9
2.1.3 Функция help_comm	9
2.1.4 Функция free_strs	9
2.2. Файл readaform.c	10
2.2.1 Функция read_a_fmt	10
2.2.2 Функция parse_dates	10
2.2.3 Функция check_dates	10
2.2.4 Функция incr_str_arr	10
2.2.5 Функция cmp_dates	10
2.2.6 Функция malloc_ptr	11
2.2.7 Функция realloc_ptr	11
2.3. Файл control.c	11
2.3.1 Функция inp_oup_ctrl	11
2.4. Файл findcntdel.c	11
2.4.1 Функция find_cnt_del	11
2.5. Файл currdate.c	12

2.5.1	Функция curr_date	12
2.6.	Файл maxmindate.c	12
2.6.1	Функция max_min_date	12
2.7.	Файл maxmindate.c	12
2.7.1	Функция max_min_date	12
2.7.2	Функция cmp_sents	12
3.	Сборка проекта (Makefile)	13
4.	Заключение	14
5.	Список использованных источников	15
6.	Приложение А. Инструкция запуска программы.	16
7.	Приложение Б. Примеры работы программы	17

ВВЕДЕНИЕ

Курсовая работа представляет собой разработку программы для обработки текста. Проект реализуется на языке C, с соблюдением требований к отказоустойчивости. Особенности реализации описаны в общих сведениях и описании работы функций. Примеры использования CLI в Приложении А. 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* maxDate`: 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 значений

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;

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 выводит 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 strcasestr.
- 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.

2.3. Файл `control.c`

2.3.1 Функция `inp_our_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/уууу` 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

2.6. Файл maxmindate.c

2.6.1 Функция max_min_date

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

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

2.7. Файл maxmindate.c

2.7.1 Функция max_min_date

Эта функция отвечает за сортировку от ранней к самой поздней дате. 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

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

¹ * - корень проекта

ЗАКЛЮЧЕНИЕ

В результате написания данной работы я изучил то, какими методами можно работать со строками, форматировать их, парсить, извлекать данные. Также узнал о сторонних функциях для работы с сравнением строк, qsort, и многое другое. Итоговые результаты программы полностью соответствуют заданным ожиданиям.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Курс «Программирование на С. Практические занятия. Первый семестр» : <https://e.moevm.info/course/view.php?id=8>
2. Сайт Wikipedia: <https://ru.wikipedia.org>
3. «Язык программирования Си» Брайан Керниган и Деннис Ритчи.

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 дочернюю 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

```
~/EVM_PR/repos/pr-2025-5341/Beizer_Lev_cw/src > on Beizer_lev_cw 12 77 ./cw
Course work for option 4.8, created by Lev Beizer.
3

Error: An empty string is entered
```

Иллюстрация 1: Подана пустая строка

```
~/EVM_PR/repos/pr-2025-5341/Beizer_Lev_cw/src > on Beizer_lev_cw 11 76 ./cw
Course work for option 4.8, created by Lev Beizer.
6

Error: incorrect input 6
```

Иллюстрация 2: Подан некорректный аргумент

```
~/EVM_PR/repos/pr-2025-5341/Beizer_Lev_cw/src > on Beizer_lev_cw 11 76 ./cw
Course work for option 4.8, created by Lev Beizer.
1
Hello world, this is a simple sentence. This sentence (01/01/2000) has a date in brackets. This is a duplicate sentence. And here is a valid leap year date: 29/02/2024. But this is NOT a valid leap year: 29/02/2025. This is a duplicate sentence. This sentence...10/10/2010...has junk...and...01/01/2001. This sentence has only ONE 19th century date: 04/04/1850. ALL 19th CENTURY: 01/01/1801, 02/02/1850, 31/12/1900. THIS IS A CASE-INSENSITIVE DUPLICATE. Mixed dates: 01/01/1899 (19th C) and 01/01/2001 (not 19th C). Invalid formats: 30-01-2000 and 1/300/2000 and 05/May/2005. Min/Max test (Min=2000, Max=4000): 03/03/3000, 02/02/2000, 01/01/4000. A sentence with today's date for task 1: 15/11/2025. all 19th century: 01/01/1801, 02/02/1850, 31/12/1900. This one should also be deleted: 05/05/1805. Dates stuck together: 01/01/2010and02/02/2011. Dates with lots of spaces: 01/01/1999 12/12/1998. Invalid values: 32/01/2000, 01/13/2000, 50/50/50, 00/00/0000. Just symbols: 5%*(.). An empty sentence.. The boundary of the 19th century: 31/12/1800 (NO) and 01/01/1901 (NO).

A sentence with today's date for task 1: 15/11/2025.
```

Иллюстрация 5: Пример работы опции 1

```
~/EVM_PR/repos/pr-2025-5341/Beizer_Lev_cw/src > on Beizer_lev_cw 11 76 ./cw
Course work for option 4.8, created by Lev Beizer.
3
Hello world, this is a simple sentence. This sentence (01/01/2000) has a date in brackets. This is a duplicate sentence. And here is a valid leap year date: 29/02/2024. But this is NOT a valid leap year: 29/02/2025. This is a duplicate sentence. This sentence...10/10/2010...has junk...and...01/01/2001. This sentence has only ONE 19th century date: 04/04/1850. ALL 19th CENTURY: 01/01/1801, 02/02/1850, 31/12/1900. THIS IS A CASE-INSENSITIVE DUPLICATE. Mixed dates: 01/01/1899 (19th C) and 01/01/2001 (not 19th C). Invalid formats: 30-01-2000 and 1/300/2000 and 05/May/2005. Min/Max test (Min=2000, Max=4000): 03/03/3000, 02/02/2000, 01/01/4000. A sentence with today's date for task 1: 15/11/2025. all 19th century: 01/01/1801, 02/02/1850, 31/12/1900. This one should also be deleted: 05/05/1805. Dates stuck together: 01/01/2010and02/02/2011. Dates with lots of spaces: 01/01/1999 12/12/1998. Invalid values: 32/01/2000, 01/13/2000, 50/50/50, 00/00/0000. Just symbols: 5%*(.). An empty sentence.. The boundary of the 19th century: 31/12/1800 (NO) and 01/01/1901 (NO).

Hello world, this is a simple sentence.
This sentence (01/01/2000) has a date in brackets.
This is a duplicate sentence.
And here is a valid leap year date: 29/02/2024.
But this is NOT a valid leap year: 29/02/2025.
This sentence...10/10/2010...has junk...and...01/01/2001.
THIS IS A CASE-INSENSITIVE DUPLICATE.
Mixed dates: 01/01/1899 (19th C) and 01/01/2001 (not 19th C).
Invalid formats: 30-01-2000 and 1/300/2000 and 05/May/2005.
Min/Max test (Min=2000, Max=4000): 03/03/3000, 02/02/2000, 01/01/4000.
A sentence with today's date for task 1: 15/11/2025.
Dates stuck together: 01/01/2010and02/02/2011.
Dates with lots of spaces: 01/01/1999 12/12/1998.
Invalid values: 32/01/2000, 01/13/2000, 50/50/50, 00/00/0000.
Just symbols: 5%*(.).
An empty sentence.. The boundary of the 19th century: 31/12/1800 (NO) and 01/01/1901 (NO).
```

Иллюстрация 4: Пример работы опции 3

```
~/EVM_PR/repos/pr-2025-5341/Beizer_Lev_cw/src > on Beizer_lev_cw 110 78 ./cw
Course work for option 4.8, created by Lev Beizer.
4
Hello world, this is a simple sentence. This sentence (01/01/2000) has a date in brackets. This is a duplicate sentence. And here is a valid leap year date: 29/02/2024. But this is NOT a valid leap year: 29/02/2025. This is a duplicate sentence. This sentence...10/10/2010...has junk...and...01/01/2001. This sentence has only ONE 19th century date: 04/04/1850. ALL 19th CENTURY: 01/01/1801, 02/02/1850, 31/12/1900. THIS IS A CASE-INSENSITIVE DUPLICATE. Mixed dates: 01/01/1899 (19th C) and 01/01/2001 (not 19th C). Invalid formats: 30-01-2000 and 1/300/2000 and 05/May/2005. Min/Max test (Min=2000, Max=4000): 03/03/3000, 02/02/2000, 01/01/4000. A sentence with today's date for task 1: 15/11/2025. all 19th century: 01/01/1801, 02/02/1850, 31/12/1900. This one should also be deleted: 05/05/1805. Dates stuck together: 01/01/2010and02/02/2011. Dates with lots of spaces: 01/01/1999 12/12/1998. Invalid values: 32/01/2000, 01/13/2000, 50/50/50, 00/00/0000. Just symbols: 5%*(.). An empty sentence.. The boundary of the 19th century: 31/12/1800 (NO) and 01/01/1901 (NO).

Earliest date: 01/01/2000
Latest date: 01/01/2000
Earliest date: 29/02/2024
Latest date: 29/02/2024
Earliest date: 01/01/2001
Latest date: 10/10/2010
Earliest date: 04/04/1850
Latest date: 04/04/1850
Earliest date: 01/01/1801
Latest date: 31/12/1900
Earliest date: 01/01/1899
Latest date: 01/01/2001
Earliest date: 02/02/2000
Latest date: 01/01/4000
Earliest date: 15/11/2025
Latest date: 15/11/2025
Earliest date: 05/05/1805
Latest date: 05/05/1805
Earliest date: 01/01/2010
Latest date: 02/02/2011
Earliest date: 12/12/1998
Latest date: 01/01/1999
Earliest date: 31/12/1800
Latest date: 01/01/1901
```

Иллюстрация 7: Пример работы опции 4

```
~/EVM_PR/repos/pr-2025-5341/Beizer_Lev_cw/src > on Beizer_lev_cw 11 76 ./cw
Course work for option 4.8, created by Lev Beizer.
2
test 03/02/2026. jsjsjjs. test/test/. \n. jddjdd \n. check suppiort. 02/03/1036 dfdkkdf 09/02/1036.

02/03/1036 dfdkkdf 09/02/1036.
test 03/02/2026.
jsjsjjs.
test/test/.
n.
jddjdd \n.
check suppiort.
```

Иллюстрация 3: Пример работы опции 2

```
~/EVM_PR/repos/pr-2025-5341/Beizer_Lev_cw/src > on Beizer_lev_cw 110 78 ./cw
Course work for option 4.8, created by Lev Beizer.
5
Команды от 1 до 5:

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

2
Отсортированные предложения по увеличению минимальной даты в них.
Если в предложении нет даты, то следует считать, что 'мнимая
минимальная дата' == inf. В случае равенства сохранить порядок
вхождения.

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

4
Для каждого предложения выводит самую раннюю и позднюю дату.
Предложения не выводятся, только строки в формате: 'Earliest date: <date>' и 'Latest date: <date>'.

5
Справка об функциях исполняемой программы.
```

Иллюстрация 6: Пример работы опции 5