

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

КУРСОВАЯ РАБОТА
по дисциплине «Программирование»
Тема: Обработка текстовых данных

Студент гр. 1304

Кардаш Я. Е.

Преподаватель

Чайка К. В.

Санкт-Петербург

2021

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент Кардаш Я. Е.

Группа 1304

Тема работы: Обработка текстовых данных

Исходные данные:

Обработка введенного текста в соответствии с запросами пользователя.

Пользователь сам выходит из программы. Используются структуры и широкосимвольный тип данных.

Содержание пояснительной записки:

Содержание, Введение,

Предполагаемый объем пояснительной записки:

Не менее 12 страниц.

Дата выдачи задания: 15.10.20.21

Дата сдачи реферата: 19.12.2021

Дата защиты реферата: 22.12.2021

Студент

Кардаш Я. Е.

Преподаватель

Чайка К. В.

АННОТАЦИЯ

Разработана программа по работе с текстом. С консоли считывается текст. Текст состоит из предложений, разделенных точкой. Предложения состоят из слов, разделенных пробелом или запятой. Ввод заканчивается двойным нажатием enter. Затем из текста удаляются повторяющиеся предложения. После текст обрабатывается в соответствии с заданием, результаты обработки выводятся в консоль.

СОДЕРЖАНИЕ

	Введение	5
1.	Разработка кода	6
1.1.	Техническое задание	6
1.2.	Разбиение на подзадачи	6
1.3.	Запись текста	7
1.4.	Общие функции обработки и вывода	8
1.5	Функции обработки и вывода, вызываемые пользователем	9
2.	Сборка программы	10
3.	Тестирование	11
4.	Инструкция для пользователя	15
5.	Заключение	15
	Список использованных источников	15
	Приложение А Исходный код	15

ВВЕДЕНИЕ

Цель работы – изучение средств языка Си для работы с текстовыми типами данных и разработка программы с их использованием.

Для выполнения задания необходимо:

Изучить нужные для работы материалы (структуры, работа с памятью, работа с широкоформатными символами данных).

Написать код в соответствии с техническим заданием.

Собрать программу с помощью заголовочных файлов и Makefile

Произвести тестирование программы

1. РАЗРАБОТКА КОДА

1.1. Техническое 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. “Раскрасить” каждое слово в зависимости от остатка от деления его длины на 4. Если остаток равен 0 - красный цвет, 1 - синий, 2 - зеленый, 3 - желтый.
2. Распечатать каждое слово которое начинается и заканчивается на заглавную букву и номера предложений в которых оно встречается.
3. Отсортировать предложения по длине последнего слова в предложении.
4. Удалить все числа из предложений. Число - набор подряд идущих цифр, перед и после которого стоят пробелы.

- 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.

1.2. Разбиение на подзадачи

Для выполнения поставленной задачи необходимо выделить следующие подзадачи:

Записать поступивший текст в структуру.

Сравнить предложения и удалить повторяющиеся (без учета регистра)

Реализовать консольное меню для пользователя с выводом информации о возможных функциях.

Написать функцию поиска слова и его записью в удобный для обращения формат – эта функция понадобится во всех подзадачах.

Написать функцию для раскраски вывода по словам в зависимости от остатка деления длины слова.

Написать функцию, которая ищет слова, начинающиеся и заканчивающиеся с заглавной буквы и выводящая их и предложения, где эти слова были найдены.

Написать функцию-компаратор для передачи в функцию сортировки qsort.

Написать функцию, находящую длину последнего слова для передачи этой информации в функцию-компаратор.

Написать функцию удаления найденного числа.

Написать функцию, проверяющую, является ли числом найденное слово (для передачи в функцию удаления чисел).

Написать функцию вывода текста в консоль, для демонстрации изменений в тексте.

Написать функцию, освобождающую выделенную под текст память.

1.3. Запись текста.

Для записи текста используется динамическая память.

Реализуем структуры Text и Sentence. В структуре Text хранятся struct Sentence, количество предложений и размер выделенной памяти.**

В Sentence хранятся указатель на широкий символ и размер выделенной памяти.

В функции get_Text происходит выделение памяти под struct Sentence и запись полученной из функции get_Sent строки до тех пор, пока пользователь не нажмет enter два раза подряд. Функция возвращает struct Text.**

В функции get_Sent происходит выделение памяти под struct Sentence* и wchar_t* а также посимвольная запись (с помощью getwchar) в широкую строку (wchar_t*). Функция возвращает struct Sentence*

1.4. Общие функции обработки и вывода.

If_Del: Принимает на вход структуру текста и проверяет есть ли в тексте повторяющиеся предложения с помощью прохода цикла в цикле. Если есть, то удаляет повторно встреченное предложение с помощью функции Del_Repeat.

Del_Repeat: Принимает текст и индекс предложения в нем. Освобождает память под предложение выделенного индекса а затем смещает все предложения и уменьшает значение поля структуры текста, отвечающее за длину текста.

menu: Принимает на вход структуру текста. Выводит сообщение для пользователя с просьбой выбрать одно из перечисленных и описанных действий либо завершить программу.

Print_Text: Принимает структуру текста и выводит находящийся там на данный момент текст на экран консоли. Вызывается после функций меняющих текст (как до вызова меню так и в нем) для того, чтобы пользователь мог увидеть что текст обработан правильно.

Free_Text: Принимает на вход текст и освобождает всю выделенную динамически память.

Find_Word: используется в любой вызываемой пользователем подпрограмме. Принимает на вход строку текста и индекс символа-разделителя перед словом (При первом входе на строку принимает -1). В функции используется структура слова **struct Word**, состоящая из **wchar_t*** (самого слова), и индексов начала и конца слова в строке.

Функция записывает в структуру слово, индекс его первого символа и индекс разделителя после слова. После функция возвращает эту структуру, с которой работают другие функции.

1.5. Функции обработки текста, вызываемые пользователем.

Подзадача 1:

Colour_Words: Принимает на вход структуру текста.

Принимает структуру слова из **Find_Word** а затем проверяет его на остаток деления на 4 и выводит раскрашивая в соответствующий цвет. После выводит символ-разделитель.

Подзадача 2:

Upp_Words: Принимает структуру текста. После берет структуру слова с помощью **Find_Word** и проверяет, являются ли первая и последняя буквы слова заглавными (это происходит с помощью функции **iswupper** стандартной библиотеки Си). Если является выводит на экран слово и предложение, в котором это слово встретилось.

Подзадача 3:

cmp: Компаратор для передачи в **qsort**. Вызывает внутри себя функцию поиска длины последнего слова. После возвращает значение **+1** или **0** если значения длин последних слов переданных **qsort** строк **>**, **<** или равны между собой.

Len_Last_Word: Принимает структуру предложения. С помощью **Find_Word** ищет последнее слово и определяет его длину. Возвращает эту длину.

Подзадача 4:

Del_Digit: Принимает на вход структуру текста. Получает слово с помощью **Find_Word** и с помощью функции проверки проверяет, является ли это слово числом. Если да, то удаляет его с помощью функции стандартной библиотеки **wcscpy**.

If_Number: Принимает на вход структуру слова и проверяет, является ли это слово числом (с помощью функции стандартной библиотеки **iswdigit**). Возвращает 1/0 Если слово соответственно число/не число.

2. СБОРКА ПРОГРАММЫ

Программа разделена на несколько файлов.

Функции **get_Text**, **get_Sent**, **menu**, **Print_Text**, **Free_Text** находятся в файле **inp_out_free.c** Соответственно, это функции, занимающиеся вводом, выводом текста и освобождением памяти.

Все объявления функций из **inp_out_free.c** написаны в **inp_out_free.h**

Также в **inp_out_free.c** импортирован заголовочный файл **processing.h** так как это необходимо для работы некоторых функций из этого файла.

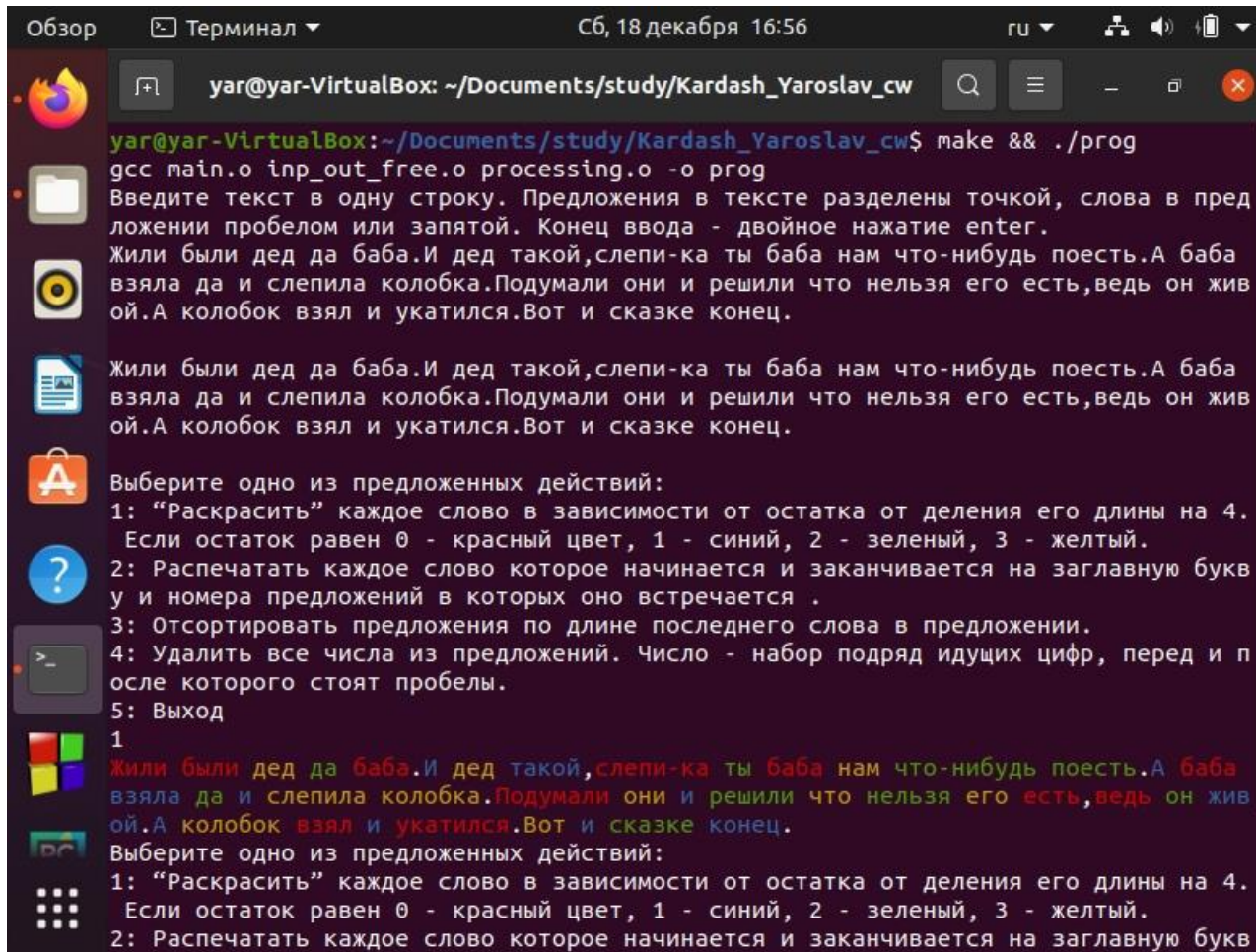
Остальные функции так или иначе обрабатывают текст.

Функции **Del_Repeat**, **If_Del**, **Find_Word**, **Colour_Words**, **If_Number**, **Del_Digit**, **Len_Last_Word**, **cmp**, **Upp_Words** находятся в файле **processing.h**

Все заголовочные файлы стандартной библиотеки, необходимые объявления а также описание структур находятся в файле **define.h** (этот файл подключается ко всем .c файлам).

В функции `main()` импортированы все три заголовочных файла. В ней происходит вызов всех функций из остальных файлов. Все объявления функций из `processing.c` написаны в `processing.h`. Сборка осуществляется с помощью утилиты `make` и `Makefile`, в котором прописаны цели для каждого файла.

3. ТЕСТИРОВАНИЕ

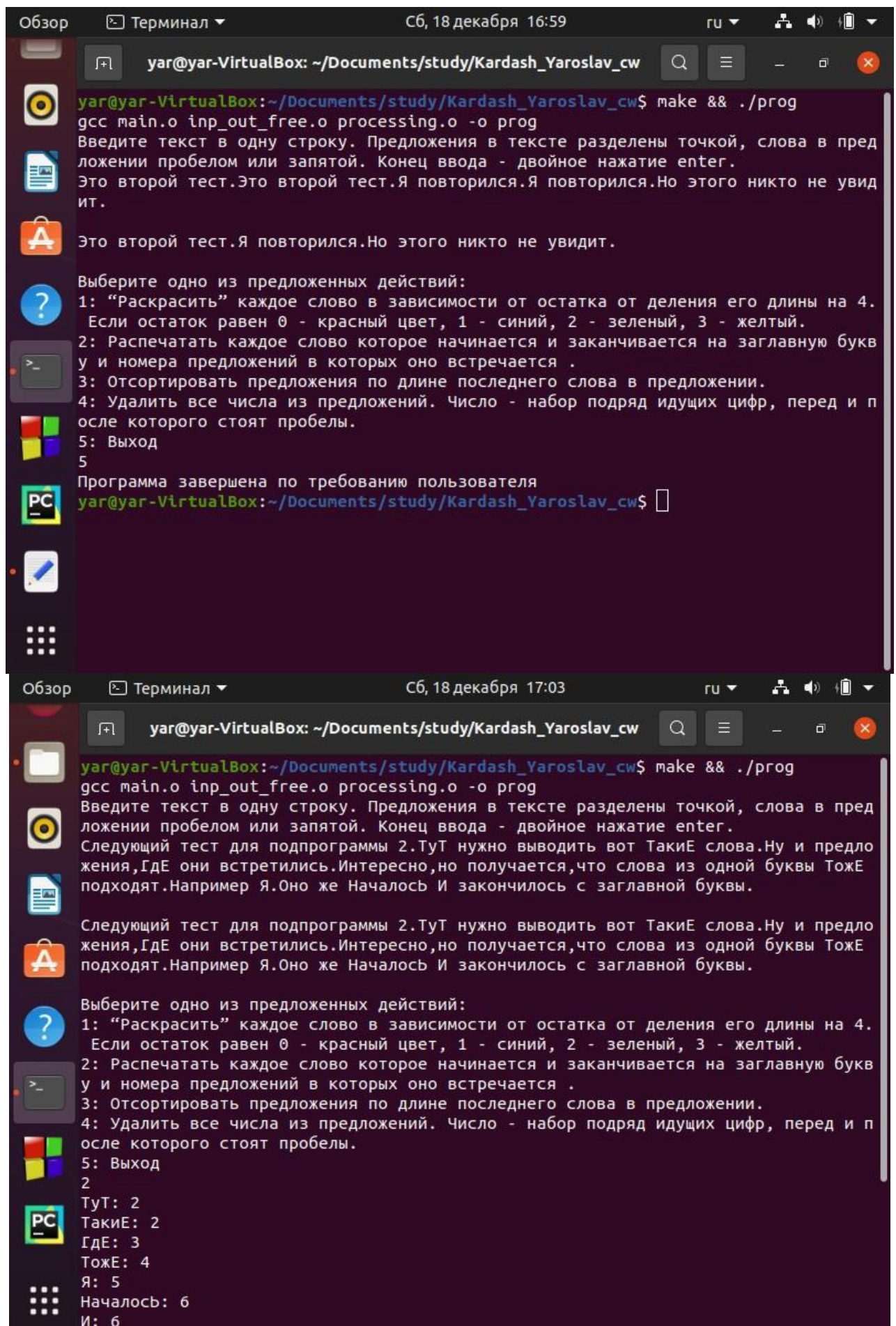


```
Обзор Терминал Сб, 18 декабря 16:56 ru
yar@yar-VirtualBox: ~/Documents/study/Kardash_Yaroslav_cw
yar@yar-VirtualBox:~/Documents/study/Kardash_Yaroslav_cw$ make && ./prog
gcc main.o inp_out_free.o processing.o -o prog
Введите текст в одну строку. Предложения в тексте разделены точкой, слова в пред-
ложении пробелом или запятой. Конец ввода - двойное нажатие enter.
Жили были дед да баба.И дед такой,слепи-ка ты баба нам что-нибудь поесть.А баба
взяла да и слепила колобка.Подумали они и решили что нельзя его есть,ведь он жив
ой.А колобок взял и укатился.Вот и сказке конец.

Жили были дед да баба.И дед такой,слепи-ка ты баба нам что-нибудь поесть.А баба
взяла да и слепила колобка.Подумали они и решили что нельзя его есть,ведь он жив
ой.А колобок взял и укатился.Вот и сказке конец.

Выберите одно из предложенных действий:
1: "Раскрасить" каждое слово в зависимости от остатка от деления его длины на 4.
   Если остаток равен 0 - красный цвет, 1 - синий, 2 - зеленый, 3 - желтый.
2: Распечатать каждое слово которое начинается и заканчивается на заглавную букв
у и номера предложений в которых оно встречается .
3: Отсортировать предложения по длине последнего слова в предложении.
4: Удалить все числа из предложений. Число - набор подряд идущих цифр, перед и п
осле которого стоят пробелы.
5: Выход
1
Жили были дед да баба.И дед такой,слепи-ка ты баба нам что-нибудь поесть.А баба
взяла да и слепила колобка.Подумали они и решили что нельзя его есть,ведь он жив
ой.А колобок взял и укатился.Вот и сказке конец.

Выберите одно из предложенных действий:
1: "Раскрасить" каждое слово в зависимости от остатка от деления его длины на 4.
   Если остаток равен 0 - красный цвет, 1 - синий, 2 - зеленый, 3 - желтый.
2: Распечатать каждое слово которое начинается и заканчивается на заглавную букв
```



Обзор Терминал Сб, 18 декабря 17:09 ru

yar@yar-VirtualBox: ~/Documents/study/Kardash_Yaroslav_cw

```
yar@yar-VirtualBox:~/Documents/study/Kardash_Yaroslav_cw$ make && ./prog
gcc main.o inp_out_free.o processing.o -o prog
Введите текст в одну строку. Предложения в тексте разделены точкой, слова в предложении пробелом или запятой. Конец ввода - двойное нажатие enter.
Ну и напоследок тест 4.Он удаляет все числа из предложений. Например 123 удалится.И 124657843 тоже.А вот такое слово нет.Оно ведь не число.
```

Выберите одно из предложенных действий:

- 1: "Раскрасить" каждое слово в зависимости от остатка от деления его длины на 4. Если остаток равен 0 - красный цвет, 1 - синий, 2 - зеленый, 3 - желтый.
- 2: Распечатать каждое слово которое начинается и заканчивается на заглавную букву и номера предложений в которых оно встречается .
- 3: Отсортировать предложения по длине последнего слова в предложении.
- 4: Удалить все числа из предложений. Число - набор подряд идущих цифр, перед и после которого стоят пробелы.
- 5: Выход

4

Ну и напоследок тест.Он удаляет все числа из предложений.Например удалится.И тоже.А вот такое слово нет.Оно ведь не число.

Выберите одно из предложенных действий:

- 1: "Раскрасить" каждое слово в зависимости от остатка от деления его длины на 4. Если остаток равен 0 - красный цвет, 1 - синий, 2 - зеленый, 3 - желтый.
- 2: Распечатать каждое слово которое начинается и заканчивается на заглавную букву и номера предложений в которых оно встречается .
- 3: Отсортировать предложения по длине последнего слова в предложении.
- 4: Удалить все числа из предложений. Число - набор подряд идущих цифр, перед и после которого стоят пробелы.

Обзор Терминал Сб, 18 декабря 17:07 ru

yar@yar-VirtualBox: ~/Documents/study/Kardash_Yaroslav_cw

```
yar@yar-VirtualBox:~/Documents/study/Kardash_Yaroslav_cw$ make && ./prog
gcc main.o inp_out_free.o processing.o -o prog
Введите текст в одну строку. Предложения в тексте разделены точкой, слова в предложении пробелом или запятой. Конец ввода - двойное нажатие enter.
Хорошо,дальше.А дальше надо отсортировать текст по длине последнего слова.Вот это например будет первым, ведь оно закончится на Ъ.Ну а вот последним,ведь последнее слово будет очень длиинииииииииинным.Как-то так.
```

Хорошо,дальше.А дальше надо отсортировать текст по длине последнего слова.Вот это например будет первым, ведь оно закончится на Ъ.Ну а вот последним,ведь последнее слово будет очень длиинииииииииинным.Как-то так.

Выберите одно из предложенных действий:

- 1: "Раскрасить" каждое слово в зависимости от остатка от деления его длины на 4. Если остаток равен 0 - красный цвет, 1 - синий, 2 - зеленый, 3 - желтый.
- 2: Распечатать каждое слово которое начинается и заканчивается на заглавную букву и номера предложений в которых оно встречается .
- 3: Отсортировать предложения по длине последнего слова в предложении.
- 4: Удалить все числа из предложений. Число - набор подряд идущих цифр, перед и после которого стоят пробелы.
- 5: Выход

3

Вот это например будет первым, ведь оно закончится на Ъ.Как-то так.А дальше надо отсортировать текст по длине последнего слова.Хорошо,дальше.Ну а вот последним, ведь последнее слово будет очень длиинииииииииинным.

Выберите одно из предложенных действий:

- 1: "Раскрасить" каждое слово в зависимости от остатка от деления его длины на 4. Если остаток равен 0 - красный цвет, 1 - синий, 2 - зеленый, 3 - желтый.
- 2: Распечатать каждое слово которое начинается и заканчивается на заглавную букву

№ теста	Входные данные	Данные на выходе	Комментарий
1	Жили были дед да баба.И дед такой,слепи-ка ты баба нам что-нибудь поесть.А баба взяла да и слепила колобка.Подумали они и решили что нельзя его есть,ведь он живой.А колобок взял и укатился.Вот и сказке конец.	Жили были дед да баба.И дед такой,слепи-ка ты баба нам что-нибудь поесть.А баба взяла да и слепила колобка.Подумали они и решили что нельзя его есть,ведь он живой.А колобок взял и укатился.Вот и сказке конец.	Программа успешно раскрасила слова в зависимости от остатка деления на 4 их длины
2	Это второй тест.Это второй тест.Я повторился.Я повторился.Но этого никто не увидит.	Это второй тест.Я повторился.Но этого никто не увидит. Программа завершена по требованию пользователя	Программа успешно удалила повторяющиеся предложения и завершилась по требованию пользователя
3	Следующий тест для подпрограммы 2.ТуТ нужно выводить вот ТакиЕ слова.Ну и предложения,Где они встретились.Интересно,но получается,что слова из одной буквы Тоже подходят.Например Я.Оно же Началось И закончилось с заглавной буквы.	ТуТ: 2 ТакиЕ: 2 Где: 3 Тоже: 4 Я: 5 Началось: 6 И: 6	Программа успешно вывела все слова начинающиеся и заканчивающиеся с заглавной буквы а также номера предложений в которых они встретились
4	Хорошо,дальше.А дальше надо отсортировать текст по длине последнего слова.Вот это например будет первым, ведь оно закончится на ь.Ну а вот последним,ведь последнее слово будет очень длиiiiiiiiiiiiiiiiiинным.Как-то так.	Вот это например будет первым, ведь оно закончится на ь.Как-то так.А дальше надо отсортировать текст по длине последнего слова.Хорошо,дальше.Ну а вот последним,ведь последнее слово будет очень длиiiiiiiiiiiiiiiiiинным.	Программа успешно отсортировала текст по длине последнего слова.

5	Ну и напоследок тест 4.Он удаляет все числа из предложений. Например 123 удалится.И 124657843 тоже.А вот такое слово нет.Оно ведь не число.	Ну и напоследок тест.Он удаляет все числа из предложений.Например удалится.И тоже.А вот такое слово нет.Оно ведь не число.	Программа успешно удалила из текста все числа
---	---	--	---

4. ИНСТРУКЦИЯ ДЛЯ ПОЛЬЗОВАТЕЛЯ

1. Запустите терминал в папке с курсовой работой.
2. Введите `make && ./prog`
3. Введите в терминал текст, разделяя предложения точкой, а слова пробелом или запятой. Чтобы закончить ввод нажмите enter два раза подряд.
4. На экране появится информация о действиях, которые можно выбрать. Выберите одно из них (у каждого свое число для выбора) и нажмите enter.
5. При желании вы можете выбирать любое из действий неограниченное число раз либо завершить программу, нажав кнопку 5.

5. ЗАКЛЮЧЕНИЕ

Изучена работа с текстовыми типами данных на языке Си. Создано консольное приложение для отработки знаний на практике. Программа собрана, протестирована, отлажена. Создана инструкция для пользователя.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

<http://cppstudio.com> – функции стандартной библиотеки Си и описание их применения.

<http://se.moevm.info/doku.php/courses:programming:report> – оформление пояснительной записки к курсовой работе и ее шаблон.

ПРИЛОЖЕНИЕ А

НАЗВАНИЕ ПРИЛОЖЕНИЯ

Main.c

```
#include "define.h"
#include "inp_out_free.h"
#include "processing.h"
int main(){
    setlocale(LC_ALL, "");
    wprintf(L"Введите текст в одну строку. Предложения в тексте разделены точкой, слова в предложении пробелом или запятой. Конец ввода - двойное нажатие enter.\n");
    struct Text text = get_Text();
    text=If_Del(text);
    Print_Text(text);
    wprintf(L"\n");
    menu(text);
    Free_Text(text);
    return 0;
}
```

Define.h

```
#include <stdio.h>
#include <wchar.h>
#include <wctype.h>
#include <stdlib.h>
#include <locale.h>
#include <math.h>
#define STAND_TXT 30
#define STAND_SENT 100
#define STAND_WORD 50
struct Sentence{
    wchar_t *s;
    int size;
};

struct Text{
    struct Sentence **t;
    int size;
    int len;
};

struct Word{
    wchar_t* w;
    int begin;
    int end;
};
```

inp_out_free.c

```
#include "define.h"
#include "inp_out_free.h"
#include "processing.h"
void menu(struct Text text) {

    int menu_input = 0;
    do{
        wprintf(L"Выберите одно из предложенных действий:\n");
        wprintf(L"1: "Раскрасить" каждое слово в зависимости от остатка от деления его длины на 4. Если остаток равен 0 - красный цвет, 1 - синий, 2 - зеленый, 3 - желтый.\n");
    } while (menu_input != 1);
}
```



```

        wprintf(L"2: Распечатать каждое слово которое начинается и заканчивается на
заглавную букву и номера предложений в которых оно встречается .\n");
        wprintf(L"3: Отсортировать предложения по длине последнего слова в
предложении.\n");
        wprintf(L"4: Удалить все числа из предложений. Число - набор подряд идущих
цифр, перед и после которого стоят пробелы.\n");
        wprintf(L"5: Выход\n");
        wscanf(L"%d", &menu_input);
        switch (menu_input) {
            case 1: Colour_Words(text);
                    break;

            case 2: Upp_Words(text);
                    break;
            case 3: qsort(text.t, text.len, sizeof(struct Sentence*), cmp);
                    Print_Text(text);
                    break;
            case 4: text= Del_Digit(text);
                    Print_Text(text);
                    break;
            case 5: wprintf(L"Программа завершена по требованию пользователя\n");
                    break;
        }
    }while(menu_input!=5);
}

```

```

struct Sentence* get_Sent(){
    int size = STAND_SENT;
    wchar_t *buf = malloc(size*sizeof(wchar_t));
    if (buf){
        wchar_t temp;
        int i=0;
        do{
            if(i>=size-2){// выделяем доп память
                wchar_t *t = realloc(buf, size+STAND_SENT);
                if (t){
                    size+=STAND_SENT;
                    buf = t;}
            }else{
                wprintf(L"Ошибка при довыделении памяти, пожалуйста, завершите ввод а
затем завершите программу");
                break;
            }
        }

        temp=getwchar();
        buf[i]=temp;
        i++;
    }while(temp!='\n' && temp!='. ');
    buf[i]='\0';
    struct Sentence *b = malloc(sizeof(struct Sentence));
    if(b){
        struct Sentence * sent=b;
        sent->s = buf;
        sent->size = size;
        return sent;
    }
    else{
        struct Sentence* er_sent;
        wprintf(L"Ошибка при выделении памяти, пожалуйста, закончите ввод и завершите
программу");
    }
}

```

```

        return er_sent;
    }
}
else{
    struct Sentence* er_sent;
    wprintf(L"Ошибка при выделении памяти, пожалуйста, закончите ввод и завершите
программу");
    return er_sent;
}
}

struct Text get_Text(){
    int size=STAND_TXT;
    struct Sentence** text = malloc(size*sizeof(struct Sentence*));
    if (text){
        int len=0;
        struct Sentence* temp;
        int n_leack=0;
        do{
            temp = get_Sent();
            if(temp->s[0]=='\n'){
                n_leack++;
                free(temp->s);
                free(temp);
            }else{
                n_leack=0;
                text[len]=temp;
                len++;
            }
        }while(n_leack<2);

        struct Text all_text;
        all_text.size = size;
        all_text.t = text;
        all_text.len =len;

        return all_text;
    }
    else{
        struct Text er_text;
        er_text.len=0;
        wprintf(L"Ошибка при выделении памяти, пожалуйста, завершите программу.");
        return er_text;
    }
}

void Print_Text(struct Text text){
    for(int i=0;i<text.len;i++){
        wprintf(L"%ls",text.t[i]->s);
    }
    wprintf(L"\n");
}

}

void Free_Text(struct Text text){
    for(int i=0;i<text.len;i++){
        free(text.t[i]->s);
    }
    free(*text.t);
    free(text.t);
}
}

```

}

processing.c

```
#include "define.h"
#include "inp_out_free.h"
#include "processing.h"

struct Text Del_Repeat(struct Text text, int j){
    free(text.t[j]);
    for(int i=j; i<text.len-1;i++){
        text.t[i]=text.t[i+1];
    }
    text.len--;
    return text;
}

struct Text If_Del(struct Text text){
    int i =0;
    int j=0;
    int t_len=text.len;
    while(i< (text.len)-1){
        j =i+1;

        while (j<text.len){
            if (wcscasecmp(text.t[i]->s, text.t[j]->s)==0){
                text=Del_Repeat(text, j);
            }
            else{
                j++;
            }
        }
        i++;
    }

    return text;
}

struct Word Find_Word(wchar_t* sent, int begin)
{
    int size = STAND_WORD;
    wchar_t* buf;
    buf = malloc(size*sizeof(wchar_t*));
    if(buf){
        struct Word word;
        //word.w = malloc(size*sizeof(wchar_t*));
        word.w=buf;
        int i = begin+1;
        word.begin = begin+1;
        while(sent[i]!=' ' && sent[i]!=',' && sent[i]!='.')
        {
            i++;
        }

        word.end = i;
        wchar_t t;
        t = sent[i];
        sent[i] = '\0';
        wcsncpy(word.w, &sent[word.begin]);
        sent[i] = t;
        return word;
    }
    else{
        struct Word er_word;
```

```

        wprintf(L"Ошибка при работе функции. Память выделена некорректно. Пожалуйста,
завершите программу");
        return er_word;
    }
}

void Colour_Words(struct Text text){
    for(int i=0;i<text.len;i++){
        int begin=-1;
        int j=0;
        while(j<wcslen(text.t[i]->s)-1){
            //int size = STAND_WORD;
            struct Word word;
            word = Find_Word(text.t[i]->s,begin);
            begin=word.end;
            j=begin;
            if((word.end-word.begin)%4==0){
                wprintf(L"\033[31m%s",word.w);
            }
            if((word.end-word.begin)%4==1){
                wprintf(L"\033[34m%s",word.w);
            }
            if((word.end-word.begin)%4==2){
                wprintf(L"\033[32m%s",word.w);
            }
            if((word.end-word.begin)%4==3){
                wprintf(L"\033[33m%s",word.w);
            }
            wprintf(L"\033[0m%c",text.t[i]->s[j]);
            free(word.w);
        }

        wprintf(L"\n");
    }
}

int If_Number(struct Word word){
    int f=1;
    for(int i=0;i<word.end-word.begin;i++){
        if (iswdigit(word.w[i])){
            f=1;
        }
        else{
            f=0;
            break;
        }
    }
    return f;
}

struct Text Del_Digit(struct Text text){
    for(int i=0;i<text.len;i++){
        int j=-1;
        int len = wcslen(text.t[i]->s);
        while(j<len-1){
            struct Word word;
            word = Find_Word(text.t[i]->s,j);
            if(If_Number(word)){
                wcscpy(&text.t[i]->s[word.begin],&text.t[i]->s[word.end+1]);
                len =wcslen(text.t[i]->s);
                if (j==len-1){
                    text.t[i]->s[j+1]='\0';
                    text.t[i]->s[j]='.';
                    len =wcslen(text.t[i]->s);
                }
                continue;
            }
        }
    }
}

```

```

        j=word.end;
        free(word.w);
    }

}

return text;
}

int Len_Last_Word (struct Sentence sent){
    int size = STAND_WORD;
    int len_last_word = 0;
    int j=-1;
    int len = wcslen(sent.s);
    struct Word word;
    while(j<len-1){
        word.w = malloc(size*sizeof(wchar_t*));
        word = Find_Word(sent.s,j);
        j=word.end;
        if(j<len-1){
            free(word.w);
        }
    }
    len_last_word= word.end - word.begin;
    free(word.w);
    return len_last_word;
}

int cmp(const void* a,const void* b){
    struct Sentence* sent_1= *((struct Sentence**)a);
    struct Sentence* sent_2= *((struct Sentence**)b);
    int len_1=Len_Last_Word(*sent_1);
    int len_2=Len_Last_Word(*sent_2);
    if (len_1<len_2)
        return -1;
    if (len_1>len_2)
        return 1;
    return 0;
}

void Upp_Words(struct Text text){
    for(int i=0;i<text.len;i++){
        int j=-1;
        int len = wcslen(text.t[i]->s);
        struct Word word;
        while(j<len-1){
            word = Find_Word(text.t[i]->s,j);
            if (iswupper(word.w[0]) && iswupper(word.w[wcslen(word.w)-1])){
                wprintf(L"%ls: %d\n",word.w,i+1);
            }
            j=word.end;
            free(word.w);
        }
    }
}

```

inp_out_free.h

```

void menu(struct Text text);
struct Sentence* get_Sent();

```

```
struct Text get_Text();  
void Print_Text(struct Text text);  
void Free_Text(struct Text text);
```

processing.h

```
struct Word Find_Word(wchar_t* sent,int begin);  
void Colour_Words(struct Text text);  
struct Text Del_Digit(struct Text text);  
int If_Number(struct Word word);  
void Print_Text(struct Text text);  
int Len_Last_Word (struct Sentence sent);  
struct Text Sort_Text (struct Text text);  
int cmp(const void* a,const void* b);  
struct Text Del_Repeat(struct Text text,int j);  
struct Text If_Del(struct Text text);  
void Upp_Words(struct Text text);
```

Makefile

```
all: main.o inp_out_free.o processing.o  
    gcc main.o inp_out_free.o processing.o -o prog  
main.o: main.c  
    gcc -c main.c  
inp_out_free.o: inp_out_free.c  
    gcc -c inp_out_free.c  
processing.o: processing.c  
    gcc -c processing.c
```